

https://dspace.org/

Providing value-added services to cellphone contract clients - a hybrid recommendation approach

Ndlovu, Mpumelelo

2016

http://hdl.handle.net/10539/21650

Ndlovu, Mpumelelo (2016) Providing value-added services to cellphone contract clients - a hybrid recommendation approach, University of Witwatersrand, Johannesburg, <http://wiredspace.wits.ac.za/handle/10539/21650>

Downloaded from a DSpace repository. https://dspace.org/



University of the Witwatersrand

MSc Computer Science Thesis

Providing Value-Added Services to Cellphone Contract Clients – A Hybrid Recommendation Approach

Author:
Mpumelelo Ndlovu
833222@students.wits.ac.za

Supervisor:
Mr. Michael Mchunu

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science Computer Science

in the

School of Computer Science
Faculty of Science

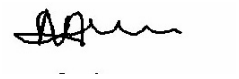
November 21, 2016

Declaration of Authorship

I, Mpumelelo NDLOVU, declare that this Thesis titled, "Providing Value-Added Services to Cellphone Contract Clients – A Hybrid Recommendation Approach" and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Masters degree at this University.
- Where any part of this Research Proposal has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:



Date: ***November 21, 2016***

"Find something that you're really interested in doing in your life. Pursue it, set goals, and commit yourself to excellence. Do the best you can."

Chris Evert

Abstract

There is stiff competition for customers and market share in the South African telecommunications industry amongst the four predominant mobile service providers, namely Vodacom, MTN, Cell C and Telkom Mobile. The First National Bank (FNB) through one of its entities, FNB Connect, has also joined this intensely competitive environment. These companies face a constant challenge of having to come up with new and innovative ways of attracting new customers and retaining their current ones. Cell C has embarked on a good strategy of claiming solid market share and growing itself against the competition by using the Private Label Promotions (PLP) group, a leading BEE Level 3 company that provides a variety of business solutions, to market `GetMore`, its value-added service package. A recommender system could be used to suggest and promote the items available in this package to existing and potential clients (users). There are different approaches to recommendation, the most widely used ones being the *collaborative* and *content-based* recommendation. The collaborative filtering approach uses the ratings of other users to recommend the items the current (active) user might like. In the content-based approach, items are recommended in terms of their content similarity to items a user has previously liked, or elements that have matched a user's attributes (features). Hybrid recommendation approaches are used To eliminate the drawbacks individually associated with the CF and CBF approaches and to leverage their advantages. One of the aims of this research was to design and implement a prototype hybrid recommender system that would be used to recommend Cell C's `GetMore` package to current and potential subscribers. The system was to implement *matrix factorisation* (collaborative) and *cosine similarity* (content-based) techniques. Several experiments were conducted to evaluate its performance and quality. The metrics used included Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Area Under the ROC Curve (AUC). We expected the proposed hybrid recommender system would leverage the advantages provided by its different components and demonstrate its effectiveness in providing Cell C's customers with accurate and meaningful recommendations of its `GetMore` package services.

Keywords:

Content-based Recommendation, Collaborative Recommendation, Hybrid Recommendation, Cosine Similarity, Matrix Factorisation, Association Rule Mining, J48 Classifier, Decision Table, Naïve Bayes, Simple K-means, Expectation Maximization, Farthest First, Predictive Apriori

Acknowledgements

For his guidance, advice, and insight a special acknowledgement goes to Mike Mchunu, my supervisor. I would also like to thank my family for their support, encouragement and prayers. A special mention is needed of my colleagues at the FNB Contact Centre IT who helped me with the architecture of the recommender system. And, above all, the CIO and my former boss at the PLP Group, Mr. Cobus Pretorius. Thank you, sir.

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

Contents v

List of Figures ix

List of Tables xi

Abbreviations xiii

Symbols xiv

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Problem Definition . . . 4
  - 1.2.1 Research Problem . . . 5
- 1.3 Importance of the Problem . . . 7
- 1.4 Research Questions and Objectives . . . 8
- 1.5 Overview of the Approach . . . 10
- 1.6 Contributions of this Study . . . 14
- 1.7 Structure of the Document . . . 14

2 Background and Related Work 16
- 2.1 Introduction . . . 16
- 2.2 Data Mining, Text Mining and Recommender Systems . . . 17
  - 2.2.1 Classification, Estimation and Prediction . . . 18
  - 2.2.2 Association Rule Mining . . . 19
  - 2.2.3 Clustering . . . 20
  - 2.2.4 Concept Description and Profiling . . . 21
- 2.3 Data Mining and Recommender System Applications . . . 22
  - 2.3.1 Applications of Data Mining . . . 22

2.3.2 Recommender System Applications . . . 25
2.4 Recommendation and Recommender Approaches . . . 29
2.4.1 Content-Based Filtering (CBF) Recommender Systems . . . 31
2.4.2 Collaborative Filtering (CF) Recommender Systems . . . 32
2.4.3 Demographic Recommender Systems . . . 35
2.4.4 Utility-based Recommender Systems . . . 35
2.4.5 Knowledge-based Recommender Systems . . . 36
2.4.6 Hybrid Recommender Systems . . . 36
2.5 Recommender Systems Challenges . . . 40
2.5.1 Limitations of Content-based Systems . . . 41
2.5.2 Limitations of Collaborative Systems . . . 42
2.6 Related Work . . . 42
2.6.1 Related Work in Value-Added Service (VAS) Provision . . . 43
2.6.2 Related Work in Hybrid Recommender Systems . . . 45
2.7 Selected Algorithms and Metrics for the Research Problem . . . 46
2.7.1 Collaborative Filtering Algorithms . . . 47
2.7.1.1 Matrix Factorisation and Singular Value Decomposition (SVD) . . . 48
2.7.1.2 Nearest Neighbourhood Method (NNH) . . . 52
2.7.2 Content-Based Filtering Algorithm . . . 54
2.7.2.1 Item-Based CBF . . . 55
2.7.2.2 User-Based CBF . . . 56
2.7.2.3 Combining CBF: The Rating Matrix . . . 57
2.7.3 Selected Data Mining Techniques . . . 58
2.7.3.1 Classification . . . 58
2.7.3.2 Prediction . . . 60
2.7.3.3 Clustering . . . 60
2.7.3.4 Association Rule Mining . . . 61
2.7.3.5 Linear Regression Methods . . . 62
2.7.4 Recommender System Item Ratings . . . 62
2.7.5 Similarity Metrics . . . 64
2.8 Chapter Summary . . . 67

3 Research Method 68
3.1 Introduction . . . 68
3.2 Motivation for the Research Problem . . . 68
3.3 The CRISP-DM Process Model . . . 69
3.3.1 Business Understanding . . . 72
3.3.2 Data Understanding . . . 72
3.3.3 Data Preparation . . . 72
3.3.4 Modelling . . . 72
3.3.5 Evaluation . . . 73
3.3.6 Deployment . . . 73
3.4 Research Setting and Research Subjects . . . 73
3.4.1 Research Setting . . . 73
3.4.2 Research Participants . . . 74
3.5 Prototype Hybrid Recommender System . . . 75
3.5.1 System Architecture . . . 76

3.5.2 Hybrid Recommendation Algorithm . . . 77
3.5.3 System Flow . . . 79
3.5.4 User-Recommender Interaction . . . 80
3.6 Applying the CRISP-DM Process Model to the Research Problem . . . 81
3.6.1 Business Understanding . . . 81
3.6.2 Data Understanding . . . 86
3.6.3 Data Preparation Phase . . . 94
3.6.4 Modeling Phase . . . 104
3.7 Adaptability of the Prototype Hybrid Recommender System . . . 123
3.7.1 Examples of Value-Added Services in other Industries . . . 123
3.7.2 Adapting the Prototype Hybrid Recommender System . . . 124
3.8 Chapter Summary . . . 124

4 Results 126
4.1 Introduction . . . 126
4.2 Extracting Latent Traits . . . 127
4.2.1 Calculating Similarity using Latent Factors . . . 136
4.3 Collaborative Filtering User-User Similarity . . . 137
4.3.1 Using Association Rules (user-user Similarity) . . . 138
4.3.2 Memory-Based Collaborative Filtering . . . 139
4.3.2.1 Distance-based User Similarity . . . 144
4.3.3 Cluster-based User Similarity . . . 145
4.3.4 User Similarity with Matrix Factorization . . . 146
4.4 Content-based Filtering User-user Recommendation . . . 148
4.4.1 Classification of Users by Usage . . . 148
4.4.2 Predicting User Usage . . . 151
4.4.2.1 K-NN with SVD on Content Features . . . 159
4.4.3 CBF Recommendation . . . 162
4.5 Predicting Membership Churn . . . 162
4.5.1 Content Features with SVD and Linear Regression . . . 164
4.5.2 Churn Prediction using Rating Data . . . 166
4.6 Evaluating Prediction Algorithms . . . 168
4.6.1 Evaluation Metrics . . . 169
4.6.2 Evaluation Results . . . 171
4.7 Association Rule Mining (ARM) . . . 171
4.7.1 Mining Frequent Itemsets . . . 172
4.7.2 ARM using the Apriori Algorithm . . . 173
4.7.3 ARM using the Predictive Apriori Algorithm . . . 174
4.8 Prototype Hybrid Recommender System Applications . . . 175
4.8.1 Cold Start Problem . . . 176
4.8.2 System Performance . . . 177
4.9 Evaluating the Prototype Recommender System . . . 178
4.9.1 System Architecture . . . 178
4.9.2 Functionality and Accuracy . . . 179
4.9.3 Usability . . . 179
4.10 Limitations . . . 179
4.10.1 Data Sparsity . . . 180

4.10.2 Scalability . . . 180
4.10.3 Collinearity . . . 180
4.10.4 Attribute Extraction . . . 181
4.10.5 General Application . . . 181
4.10.6 Human Computer Interaction . . . 181
4.11 Chapter Summary . . . 181

5 Discussion **183**
5.1 Introduction . . . 183
5.2 Extracting Latent Features . . . 184
5.3 Rating Prediction . . . 184
5.4 Computing Similarity . . . 185
5.5 Business Problem . . . 185
5.6 Chapter Summary . . . 186

6 Conclusion and Future Work **187**
6.1 Conclusion . . . 187
6.2 Future Work . . . 188

A Examples of Recommender Systems **189**
A.1 Introduction . . . 189
A.2 Amazon . . . 189
A.3 IMDb . . . 190
A.4 Gumtree . . . 190

B Java Code **192**
B.1 Latent Features . . . 192
B.2 Attribute Selection . . . 196
B.3 Calculating Similarity . . . 198

References **204**

List of Figures

1.1 Proposed Hybrid Recommender System . . . 11

2.1 Data Mining Techniques . . . 18
2.2 The Recommendation Process . . . 30
2.3 Items in the database . . . 56
2.4 Example Utility Matrix . . . 64
2.5 Jaccard Similarity . . . 65
2.6 The Pearson Correlation Coefficient . . . 66

3.1 Generic KDD Process . . . 70
3.2 The Cross Industry Standard Process for Data Mining (CRISP-DM)[Chapman *et al.*, 2000] . . . 71
3.3 Application Interface . . . 76
3.4 The Recommendations . . . 76
3.5 System Flow . . . 80
3.6 Usage vs Status . . . 86
3.7 Business Processes . . . 87
3.8 Rating Data Sparsity Distribution . . . 92
3.9 Training Data Distribution of Ratings . . . 93
3.10 Sample Item-User Demographics Correlation . . . 96
3.11 User-Item Combined Matrix . . . 100
3.12 Simplified User-Feature Matrix . . . 102
3.13 Multiplicative Normalisation Formula . . . 102
3.14 Multiplicative Normalisation of User-Feature Matrix . . . 103
3.15 Data used by the recommendation application . . . 105
3.16 The SimpleKMeans in WEKA . . . 113
3.17 The K-Means Algorithm . . . 114
3.18 Users who rated the same item . . . 122

4.1 Latent Feature Model Performance . . . 133
4.2 Correlation Matrix . . . 134
4.3 Sample PCA Variables . . . 135
4.4 Eigenvalues and the Proportion of Variation . . . 136
4.5 The K-Means Clustering Technique . . . 145
4.6 Summary of the raw variables . . . 152
4.7 Area under the ROC Curve . . . 156
4.8 Age vs Number of Requests . . . 157
4.9 Age vs Number of Requests . . . 157

4.10 User Churn Against User Age 163
4.11 Bayes Network Classifier 164
4.12 Binary Rating Matrix 173
4.13 Part Results from Predictive Apriori 175
4.14 New User Screen 176
4.15 Recommending Items Unrelated to Rated Items 177

A.1 The Amazon Recommendation System 189
A.2 The IMDb Movie Recommendation System 190
A.3 The Gumtree Recommendation System 191

List of Tables

2.1 Different Hybridization Techniques . . . 40
2.2 Item-Feature Matrix (I) . . . 56
2.3 User Profile Matrix . . . 57
2.4 Combined Feature Matrix . . . 58
2.5 Collaborative Filtering Usage Accuracy Matrix . . . 59

3.1 Users by Age and Gender . . . 74
3.2 Users by Province and Status . . . 75
3.3 User-Item Binary Relationship . . . 77
3.4 Benefits offered by the GETMORE Service . . . 82
3.5 Customer Profile . . . 89
3.6 Item Attributes . . . 90
3.7 Rating Data . . . 91
3.8 Rating Matrix . . . 91
3.9 User-Feature Item Matrix . . . 96
3.10 Item-Feature Matrix . . . 97
3.11 User-Item Rating Matrix . . . 97
3.12 Gender Sub Matrix . . . 98
3.13 User-Age Matrix . . . 98
3.14 Item-Item_type Matrix . . . 99
3.15 Variables used in Clustering . . . 115
3.16 Numerically Coding User Birth Months . . . 116
3.17 Ranking Months Using Equation 3.11 . . . 116
3.18 Sample Getmore Data For Clustering . . . 117
3.19 Coded Getmore Data For Clustering . . . 117
3.20 Customer Profile Rules . . . 119
3.21 Item Content Matrix . . . 121
3.22 Item Content Matrix with Zeros Removed . . . 121

4.1 Correlation Between User and Item Features . . . 128
4.2 User, Item and Rating Matrices . . . 129
4.3 $X = U_2.S_2.V_2'$ . . . 132
4.4 Explaining Principal Components . . . 135
4.5 Ratings given to six items by selected users . . . 142
4.6 Pearson Correlation Table of Values . . . 142
4.7 Pearson's correlation Coefficient Similarity for Selected Users . . . 143
4.8 K-Means Clustering Results . . . 146
4.9 Collaborative Filtering Usage Accuracy Matrix . . . 148

4.10 Attributes used in this Classification Exercise 149
4.11 Preliminary User Classifications 149
4.12 Classification Algorithms Results 150
4.13 Confusion Matrix of the AprioriC 151
4.14 Area Under ROC for the User Classes 159
4.15 Items Table Extract 161
4.16 Item-item Similarity in the Database 162
4.17 Usage Churn Confusion Matrix 167
4.18 Summary of User-Churn Results 168
4.19 Classification Predictions 169
4.20 Collaborative Filtering Usage Accuracy Matrix 171
4.21 Apriori Dataset 173
4.22 Example Rules from the Apriori Algorithm 174

Abbreviations

BI	Business Intelligence
CRISP-DM	Cross-Industry Standard Process for Data Mining
RS	Recommender/Recommendation Systems
KDD	Knowledge Discovery in Databases
CBF	Content-Based Filtering
CF	Collaborative Filtering
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
PLP	Private Label Promotion
SVD	Singular Value Decomposition
NNH	Nearest Neighbourhood
QA	Quality Assurance
LR	Linear Regression

Symbols

i	Item
a	Active User
u	Collaborating users (Recommender)
$\bar{r_a}$	Average rating by the active user
$\bar{r_u}$	Average rating by the recommender
$r_{u,i}$	Recommender's rating on the item
$w_{a,u}$	Active user's weight to the recommender
$P_{a,i}$	Predicted rating for the active user on the item

Dedicated to my mother who, in her wisdom, set me on this path before she departed.

Chapter 1

Introduction

1.1 Background

The term "Business Intelligence" (BI) refers to

> "*the process of collecting business data and turning it into information that is meaningful and actionable towards a strategic goal.*" [Business Intelligence, 2014].

According to Business Intelligence [2014], BI activities include reporting, analysis, data mining [Dumais, 1992; Jackson, 2002], data quality and interpretation, and predictive analysis. Many South African organizations and companies have not yet begun to use the BI-based intelligent decision-support systems (DSS) [Jantan *et al.*, 2012; Shaw *et al.*, 2001; Sophatsathit, 2013] at their disposal to their fullest extent. In most cases the scope of BI applications is limited to producing pre-written reports, most of which are mere replacements of legacy reporting systems, the difference being the use of colourful formatting in reproduced ones. The real payback for BI is obtained by extracting hidden information from an organization's data. Such information can be discovered by using data mining algorithms and tools [Moss and Atre, 2003]. The success of any organization depends largely on the extent to which it utilizes the data acquired from its business operations. Data mining techniques such as classification, prediction and clustering play a major role in the discovery of information and the generation of knowledge from data [Baitharu and Pani, 2013; Tan *et al.*, 2006].

This research highlights the importance and relevance of data mining to telecommunications companies in developing countries such as South Africa, which have to contend with the challenge posed by globalisation and the restrictive legislative framework that governs accessing,

processing and analysing private user data. The research also looks at how these companies can use data mining methods to get client or customer information from their data, information that could be used to build meaningful, mutually beneficial relationships with their customers. Data mining is the computer-assisted process of exploring and analysing large datasets, from which meaningful information is extracted [Agrawal, 2013; Cisty and Bezak, 2013; Kruse *et al.*, 1999; McGregor *et al.*, 2012]. It attempts to alleviate the challenges of processing and analysing these datasets to identify valid, novel, understandable and potentially useful data patterns [McGregor *et al.*, 2012; Merceron and Yacef, 2008; Tan *et al.*, 2006]. Different data mining algorithms are applied to these datasets to produce models based on various tasks such as classification, association rule mining, prediction, clustering and sequential pattern discovery [Adomavicius and Tuzhilin, 2001; Agrawal and Srikant, 1994; Agrawal, 2013]. Through the succinct and commodious intelligence they can provide, these models offer a lot of hope to organizations facing global competition in inelastic markets such as the telecommunications sector in South Africa.

After operating as a monopoly for many years, the telecommunications industry is now one of the most contested environments Camilovic [2008]. Following the implosion of mobile network companies, an increasing number of service providers have entered the telecommunications market, resulting in stiff competition for customers among different companies. A monopoly is usually static; it does not change much. On the other hand, competitive markets constantly change, with their clients being able to switch easily between different service providers. "For this reason, telecommunications companies explore data mining solutions to achieve competitive advantage. By understanding the demographic characteristics and customers' behaviour, telecommunications companies can successfully tailor their marketing strategies to reach those most likely to use their services, to increase customer loyalty and improve customer profitability", [Camilovic, 2008, p.63]. The South African telecommunications market is dominated by four mobile communication companies, namely Vodacom, MTN, Cell C and Telkom Mobile (also known as `8.ta`. Recently, the First National Bank (FNB) has declared itself a full telecommunications company with its handle, FNB Connect, currently available only to FNB customers. This market is already saturated. It is small and inelastic and is characterized by high call rates and inadequate infrastructure. Furthermore, the majority of the South African population live below the poverty datum line and are not able to afford the high call rates. The telecommunications space is a highly regulated cut-throat industry where the struggle to penetrate the market is only matched, if not overshadowed, by the struggle to maintain one's market share and to retain subscribers [Makwana *et al.*, 2014].

In the face of ever-increasing competition from voice and data service providers the traditional method used by mobile communication providers to retain customers, that of having them sign up to 24-month contracts, with a fancy smartphone thrown in to sweeten the deal, is now neither adequate nor sustainable. Mobile service providers have to find alternative novel ways of attracting and retaining customers [Kamath *et al.*, 2008]. The provision of uniform products,

including the focus on individual client needs, has brought pressure for change to be made in traditional marketing practices and necessitated the introduction and use of bundled cellphone packages. In this context, additional product benefits are generated by communicating with clients through the calculated use of value-added services designed and delivered to match individual customer expectations and needs [Bhavnani *et al.*, 2008; Carlsson and Walden, 2002; Eichelmann *et al.*, 2011; Pattamavorakun and Pattamavorakun, 2010; Zigkolis *et al.*, 2009]. The term "mobile-commerce" (m-commerce) refers to

> "*the buying and selling of goods and services through hand-held wireless devices such as cellular telephone and personal digital assistants (PDAs)*" [Rouse,M, 2015].

This study focusses on the provision of value-added m-commerce services to cell phone contract customers. The services can be provided through direct marketing [Cheung *et al.*, 2003; Dolnicar and Jordaan, 2007; Ling and Li, 1998; Shen and Chuang, 2009], and can be enhanced by using data mining techniques and recommendation systems[Arndt and Gersten, 2001]. Guiding users in selecting relevant items and enticing them to sign up for more products than they otherwise would are important means of attracting new customers and maintaining the loyalty of current clients.

Telecommunications companies possess a lot of information about their clients. They know who their customers are and they can easily keep track of their activities [Milne and Rohm, 2000]. They have accumulated vast amounts of data about their customers, data that cannot be analysed in a traditional manner, using manual data analysis techniques [Camilovic, 2008]. Although almost all mobile providers offer value-added digital services, many of these services are digital and technology-specific. Only smartphone users can enjoy the majority of these services. Currently, MTN provides a total of 32 value-added services, which are either technology-dependent or require a user to follow a series of tedious steps to use them. Similarly, Vodacom offers value-added digital services which are occasionally complimented by voice and data promotions. The most comprehensive service provided by Vodacom is the VoucherCloud [Vodacom VoucherCloud, 2014], which offers discounts to Vodacom cellphone users. The service is open to almost everyone. The package offers discount deals on a variety of items such as food and drink, leisure and entertainment, travel and accommodation, fashion, automotive and electronic goods.

by Cell C offers the most comprehensive value-added service package called $GetMore^{©}$ scheme [CellC GetMore, 2014]. To improve its market share Cell C, one of the dominant mobile phone providers in South Africa, has come up with the **GetMore 24-7** Service, previously known as the *Get-it* Service, a very revolutionary and innovative service for contract customers. This is a value-added customer-loyalty service which Cell C customers can subscribe to for a token

fee. It enables subscribers to focus on important tasks while it handles mundane tasks such as assisting kids with homework and helping to locate cheaper and reliable movers for customers intending to move homes. The service is offered in partnership with over 150 000 suppliers to provide customers with incredible deals, expert advice and services, bookings and outdoor travel arrangements. This study examined customer ratings of the different value-added services provided by Cell C's GetMore scheme.

1.2 Problem Definition

Depending on what individuals need, they have to process the massive information available in order to select and obtain the items they need. People who need to purchase different types of products and services are usually faced with a number of choices. For example, they need to choose the type of furniture they would like to purchase, the kind of holiday package deal they would like to pay for, which restaurant they would like to go, which movie they would like to go and watch, which taxi service operator they would like to fetch them from the airport, and so on. The vast amount of information available on different products and services makes it difficult for customers to select the products or services they need. Recommender systems have been introduced to overcome this "information overload" problem and to enable users (customers) to make the appropriate selection of the products or services they need. Recommender systems are useful for recommending different items such as movies, products and news articles. There are *content-based* and *collaborative filtering* recommendation systems. Content-based systems recommend items based on the features of items similar to them. Collaborative systems make recommendations using similarities between users and/or items. Collaborative recommender systems are used to either predict how much of an item a given user will like or to recommend selected items to the user [Agrawal *et al.*, 2009].

Recommender systems are not without problems or limitations. Regarding content-based recommenders, some of the challenges include difficulty in extracting features; not being able to obtain enough ratings in order to classify items; difficulty in obtaining features that match user preferences; features not being useful enough to determine the similarity score between items; as well as the inability to recommend items for new users. Likewise, collaborative filtering systems have problems of their own. There may not be enough users; not enough ratings may be available for new items (new item problem); there may not be many users who have rated the same items (sparsity); new users must rate enough items for their preferences to be known (cold start problem); and the datasets can be enormous. Hybrid recommender systems, which combine content-based and collaborative systems, are often used to address some of the challenges related to these systems.

Cell C's GetMore program faces some challenges. These include

1. **Getting more cellphone contract users to sign up for the GetMore service. Attempt to recruit more users has proved not to be an easy task**
   - It was hoped that increasing the number of subscribers to this service would lead to its extensive use, which could be interpreted as customer acceptance of the mobile phone package, thereby indicating improved market penetration by the service provider.

2. **Maximizing service usage by persuading existing subscribers to utilize the benefits on offer**
   - Such usage could be viewed as an indication of customer assimilation of the service and a sign of its utility. It was hoped that this reliance on the service would translate into the adoption of the whole package offered by the mobile service provider.

3. **Retaining current users**
   - If the users appreciate the service enough to keep it for long, this could possibly translate to an appreciation of the Cell C package as a whole.

In order to address these problems, Cell C has embarked on a good strategy of claiming solid market share and growing itself against the competition by using the PLP Group as a proxy to market its GetMore service and, by extension, to obtain more customers in its fight against a dominant opposition. The PLP Group was founded in 1992 and is a leading *Level 3 BEE Company* that provides a variety of business solutions.

1.2.1 Research Problem

One of the objectives of this study was to design, develop and deploy a prototype hybrid recommender system to assist the PLP Group in meeting its business objectives, thereby enabling it to solve the business problem related to Cell C's GetMore package. Within the context of this research the following challenges had to be addressed:

1 **New-User Problem**
   - Recommend existing services to new users whose preferences are not known;

2 **Existing-User Problem**

- Based on service usage history, recommend existing services to existing users who have not subscribed to these services;

3 **New-Item Problem**

- Newly introduced services cannot be recommended to both existing and new users unless they (services) have been rated by some user(s) before, in which case they may be recommended through occasional marketing alerts.

Recommending Existing Services to New Users

Collaborative recommender systems are characterised by the *cold-start* problem [Asabere, 2012; Schein *et al.*, 2002], which relates to profiling or handling new users. In this situation, where a new user's previous purchasing profile is unavailable, content-based filtering recommendation approaches often return better results. The recommendation algorithm, in this situation, involved comparing users based on their item interests and recommending new items in which similar users showed interest. In the hybrid recommender system developed as part of this research, a user was recommended items which were selected from the items that a group of users with similar tastes and preferences had previously rated highly. Content-based recommendation systems could provide recommendations for "cold-start" users, where little or no training data was available. However, they typically have lower accuracy compared to collaborative filtering systems [Sarwar *et al.*, 2002; Vozalis and Margaritis, 2006].

The problem with cold start cases was the absence of previous purchasing or rating data that could be used in creating a customer profile or model. In the developed hybrid recommender system the only input data into the recommendation process would be the customer's description of the desired item and the customer's profile, created when the customer applied for a Cell C mobile phone contract. Such a profile consisted mainly of demographic attributes. In this case, the recommendation for the customer would be based on the requested item and on what other users matching the customer's preferences had rated highly.

Recommending Existing Services to Existing Users

This is the common hybrid filtering scenario where previous customer tastes are used to identify items which may be of interest to an existing customer. Item and user similarity are derived using content-based filtering. In the system developed as part of this study a user was recommended items which resembled the ones he or she had shown preference for in the past. Collaborative filtering was used to recommend to the user what similar users had liked in the past. Hybrid

schemes attempt to combine ratings and profiles to yield better recommendations [Gunawardana and Meek, 2009].

Recommending New Services to New and Existing Users

The *new-item* problem was solved in a manner similar to the *new-user* problem discussed in Section 1.2.1, by using item-based collaborative filtering and content-based filtering using item attributes [Ghazanfar and Prugel-Bennett, 2010]. To work around the *new-item, new-user* issue, this research used a combination of CBF and CF techniques. Therefore, a hybrid recommender system [Adomavicius and Tuzhilin, 2005; Rombouts and Verhoef, 2004] provided an ideal solution to the problem facing the PLP Group.

Clustering, classification and association data mining techniques were used in modelling this hybrid recommender system to establish customer characteristics and to determine item similarities using latent features and rating patterns [Dumais, 1992; Salehi, 2014]. These patterns could be used to identify relationships among various data instances, and they were utilised in this research to deduce rules and models [Olson and Delen, 2008]. The scope of this research was limited to identifying and analysing customer and item clusters in the logged requests for the purpose of recommending other items that users could be interested in.

1.3 Importance of the Problem

Since recommender systems are typically designed and implemented to solve real-world problems, the study of recommendation systems is an exciting and rewarding research field, which offers benefits to industry and academic practitioners alike [Aksel and Birtürk, 2010]. The problem facing mobile phone providers like Cell C, through its subcontracted company the PLP Group, is a real-world business problem that confronts most organizations that operate in the current competitive global economy. Information technology has evolved to enable the collection and storage of huge amounts of data [Camilovic, 2008]. Telecommunications organizations collect and store huge quantities of data about their customers, data that could be used to generate meaningful and useful information that could benefit these companies. Data mining techniques have gained acceptance as a viable means of obtaining such information [Antunes and Oliveira, 2004].

One of the problems this research sought to address was the need by the PLP Group to attract new users, retain existing ones and provide utility to contracted users. Shrinking markets, coupled with stiff and increasing competition amongst the major role players in the local telecommunications industry, make it imperative for South African mobile phone providers to extract more value from their existing clientèle, while striving to gain more market penetration.Companies that succeed in effecting these strategies stand a better chance of ensuring shareholder and customer satisfaction and loyalty. Furthermore, their continued success and operation provide a long-term guarantee of service and prosperity for their employees. For Cell C, the successful execution of the PLP Group of its mandate means growth, profitability, employee and shareholder satisfaction.

For their part, users find recommender systems very useful. By using these systems, customers are able to avoid the problem of having to process vast amounts of information, looking for their preferred products or services [Dolnicar and Jordaan, 2007]. This is the information overload problem. In an attempt to overcome the problem of information overload, a recommender system can be used to recommend available goods and services. It can also be used to recommend relevant products or services that are trending and popular, products which are similar to those purchased or selected by customers in the past. Most people hate the inconvenience of having to search for information, whilst others do not really know the exact information that they are looking for. In such circumstances, a recommender system could come in handy as an advertising and filtering instrument.

1.4 Research Questions and Objectives

To improve its market share, Cell C has introduced the GetMore Service, a comprehensive value-added service package offered to its contract customers for a token subscription fee. The package provides a variety of services to subscribers, such as assisting with school homework and helping to locate cheaper and reliable movers for customers intending to relocate. The GetMore Service is offered in partnership with over 150 000 suppliers in order to provide customers with incredible deals, expert advice and services, bookings and outdoor travel arrangements. With a view to claiming a larger share of the market and grow itself against the competition, Cell C tasked the PLP Group to market its GetMore Service. Among other things, this study examined customer ratings of the services provided by the GetMore package. User-based and content-based features were used, including the item and customer data provided by mobile phone service providers. Given the challenging business environment in which Cell C and

the PLP Group operate, the primary question this research sought to answer is formulated as follows:

> *Can an effective prototype hybrid recommender system be designed, developed and implemented to accurately predict item ratings for cellphone contract customers, determine their profiles and characteristics, predict their behaviour, and recommend the PLP Group's services to individual clients based on the preferences of clients with similar interests?*

The following sub-questions were formulated to answer the main research question:

1. *What are the latent traits of the PLP Group's subscribers? How can these traits be used to calculate user similarity?*

2. *Which statistical or data mining technique is most suitable for discovering PLP Group subscribers whose preferences are similar to those of a client for whom a service is being recommended?*

3. *What are the common traits and attributes of frequent users of the PLP Group's services? How can these characteristics be used in recommending these services to similar clients?*

4. *What is known about PLP clients who cancel their subscription to a particular service? Can such information be used to predict service cancellation, and lead to preemptive measures to reduce the number or likelihood of subscriber cancellations?*

5. *Which data mining techniques are best able to predict or determine whether subscribers will like or dislike a particular item or service offered by the PLP Group?*

6. *Which of the products or services offered by the PLP Group should be recommended together or in combinations?*

To answer these questions and to obtain a better understanding of the work that needed to be done in this research the following objectives were formulated:

- To conduct a detailed review of the literature to obtain knowledge on the current trends in recommendation and recommender system techniques

- To study, understand and apply the CRISP-DM, [Cisty and Bezak, 2013; Sharma and Osei-Bryson, 2009; Shearer, 2000], process model as the approach to be followed in conducting this research
- To design, develop and implement a prototype hybrid recommender system
- To analyse the performance and effectiveness of the prototype hybrid recommender system using the PLP Group's customer cellphone contract dataset
- To evaluate the performance and effectiveness of the prototype hybrid recommender system using suitable performance metrics
- To demonstrate the ability of the hybrid recommendation approach to providing better recommendations compared to the individual, content-based and collaborative, recommendation approaches
- To present, analyse and discuss the results obtained from the different experiments conducted in this research

1.5 Overview of the Approach

Due to the sparsity of both the content space and the rating matrix, it was not feasible to obtain item scores using either the content-based or collaborative filtering approaches. The *cold-start* and *first-rater* problems were caused by the sparsity of the data. These problems occur when a first-time user and a newly added item, respectively, have just started using the system and it is hard to find users and items with a similar profile because there is inadequate information about them in the system [Adomavicius and Tuzhilin, 2005; Aksel and Birtürk, 2010; Asabere, 2012; Grivolla *et al.*, 2010; Li and Kim, 2003]. To avoid these and similar problems this study adopted a hybrid approach to make an autonomous set of recommendations from each component. These recommendations were combined, ranked and presented to users by the hybrid recommender system that was implemented as part of this research. The user and item models in our system consisted of a combination of user or item stereotypes, created automatically through a latent-feature extraction process. Each stereotype is defined by a list of item attributes represented by a row vector. This approach provides good recommendations for previously rated items and is also suitable for handling new items [Shani *et al.*, 2007]. *Figure* 1.1 illustrates the conceptual

view of the recommendation approach that was followed in designing and implementing the system.

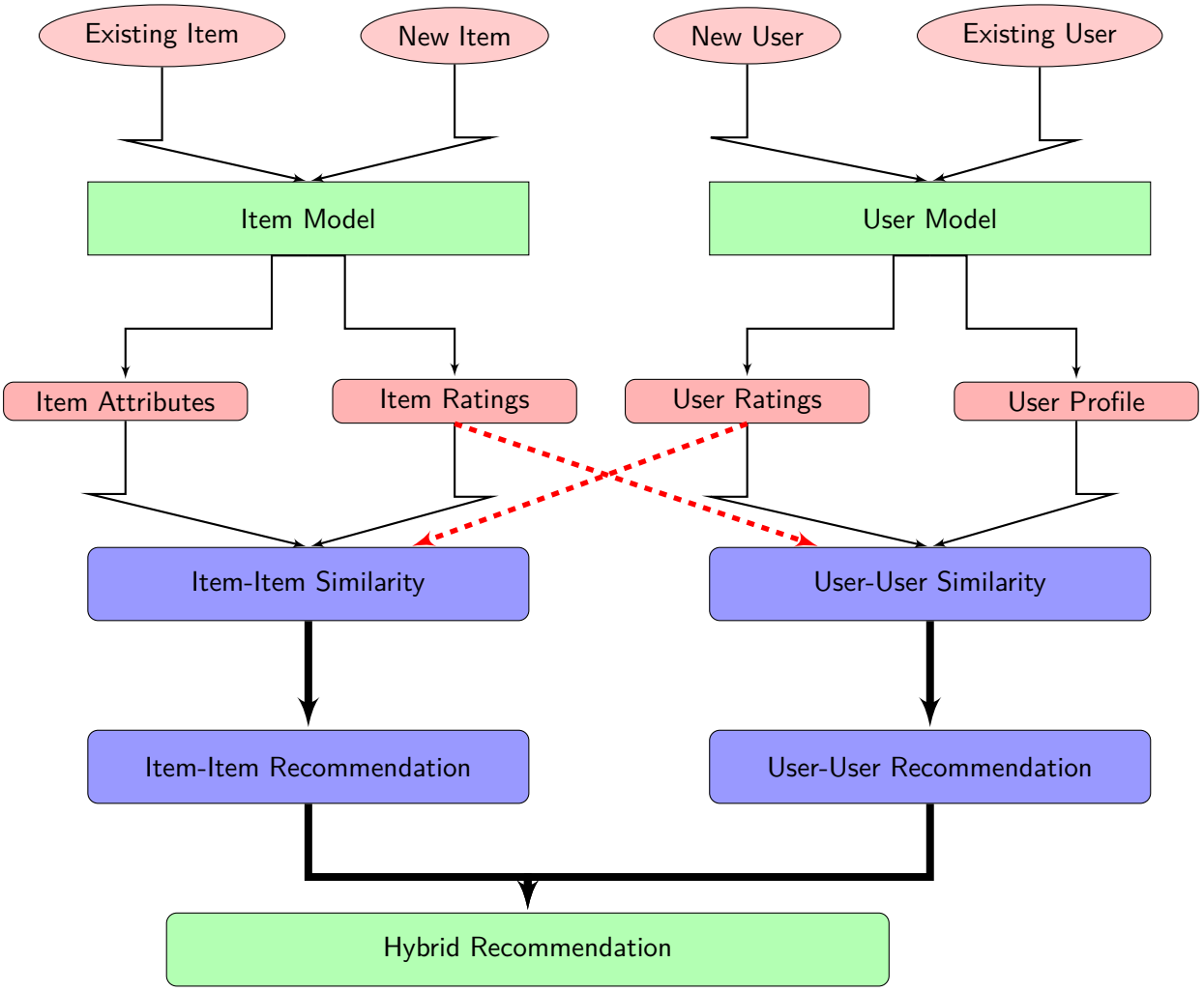


FIGURE 1.1: Proposed Hybrid Recommender System

Figure 1.1 shows that the solution to PLP's problems lay in understanding the user, the item and the rating system. These three entities hold the key to solving the business and research problems discussed earlier, in Section 1.2. Two multi-dimensional matrices were used. The first was used to store item features and the second was used to store user profiles, the link between them being a common, ratings attribute. In this research a dataset consisting of n PLP customer records was used. To answer the main research question the task was broken into several activities whose main focus was to answer the different sub-questions and by extension, provide the answer to the main question. The following activities were carried out:

- To provide effective and accurate recommendations it is important to have some knowledge of the customers who use or intend to use the products and/or services offered by different providers. One of the aims of the PLP Group is to "entice" Cell C contract subscribers to sign up for a wider variety of products and/or services. Knowing or being aware of these subscribers' traits increases the chances of targeting them with accurate recommendations, which may lead to them signing up for the recommended products and/or services on offer. A collaborative filtering technique, enhanced by the matrix factorization technique, was applied to a PLP dataset consisting of 24583 PLP subscriber records to predict PLP item ratings for individual users. Recommendations were generated based on the similarity between the latent factors of subscribers and items. Using the results obtained from this activity it was possible to answer *Sub-question 1.*

- In some instances, a group of subscribers may prefer similar items. An active PLP Group subscriber (one for whom a product and/or service is being recommended) may prefer the same items as other members of the group. Having identified a group of subscribers who prefer the same items as the active user, this information can be used to recommend to the active user that list of items which he or she has not yet subscribed for, but which similar subscribers have signed up for. In this research, the Adjusted Cosine Similarity and Vector Similarity techniques were used in conjunction with the Matrix Factorization technique to identify subscribers with similar preferences as the active user, based on rating and demographic information.

- Some of the PLP Group's subscribers may share similar attributes or characteristics. An interesting aspect to look at is to see whether subscribers with similar characteristics also prefer the same items, in which case this information can be used to recommend the same items to an active user with similar characteristics, who may not have subscribed for these items. The *SimpleKMeans* and *FilteredCluster* clustering techniques were used to identify subscribers with similar traits, to determine whether they preferred the same items. *Sub-question 3* was answered by using the results obtained in this activity.

- For a number reasons some users will cancel their subscription to particular items. For the PLP Group, it makes good business sense for subscribers to sustain their use of the items on offer. For those subscribers who cancel their subscriptions, it is very important for the PLP Group to identify the main factors responsible for such cancellations. Armed with this knowledge, it is then possible to pro-actively implement marketing and business strategies whose purpose will be to halt or reduce the number of subscriber cancellations. The PLP Group's dataset was used to obtain information on service cancellations. Classification techniques, namely the *J48* and *Naïve Bayes* classifiers, were used to classify PLP Group subscribers and, based on their usage of the different services, to determine those likely

to cancel their subscription to some of these services. The results from this activity were analyzed and used to answer *Sub-question 4*.

- Valuable information can be obtained about customer traits by analyzing the PLP Group's dataset. We focus on latent traits, traits that cannot be measured directly, and are for this reason considered "unobservable", hidden. This research examined the relationship between the rating of a PLP Group service and a subscriber's latent traits. Latent Feature Indexing, a Matrix Factorisation method based on Singular Value Decomposition (SVD), was used to compute a low-rank matrix, which was used to identify latent features and to efficiently calculate item and user similarities, resulting in a faster determination of the closest item peers.

- In user-based collaborative filtering systems, an item is recommended to a user based on the ratings of similar users for that item. On the other hand, in item-based collaborative filtering systems, an item is recommended to a user based on the ratings she/he has assigned to similar items in the past. In both types of collaborative filtering systems, similarity metrics are used to compute similarities between users (user-user similarity) and items (item-item similarity). This information is then used to predict the rating an active user would assign to a target item. In this research, item-based similarity techniques were used. Alternative approaches were also studied, to determine their suitability and performance in predicting whether a service offered by the PLP Group will be preferred by a particular subscriber. The focus in this part of the research was to address *Sub-question 6*. Association rule mining and predictive data mining techniques were used. To determine association rules the *PredictiveApriori* and *Tertius* algorithms were used. For prediction linear regression and decision tree methods were used, in the form of *LinearRegression* and *J48* algorithms, respectively.

- Given the diversity of the items or products offered by the PLP Group, individual subscribers will select the item (product or service) or items they prefer the most. The company could make more profit if it had the means of identifying items and then recommending the item or items similar to the one(s) already selected by a subscriber. This recoomendation technique is the idea behind *Sub-question 7*, which focusses or is based on the item recommendation problem. Association rule mining and classification techniques were used to address this question. The decision tree algorithm, *J48*, was used for classification purposes. The *PredictiveApriori* and *Tertius* algorithms were used to generate different association rules.

1.6 Contributions of this Study

To ameliorate the individual deficiencies of content-based and collaborative recommender systems, in this study we decided to adopt a hybrid system that combined both content features and rating data to form an item data model and a user data model. While the majority of hybrid recommender systems in literature focus mainly on item-item similarity and user-user similarity based on the similarity of item preferences, in this study, items are similar, if:

1. they have a satisfactory content features similarity score; or
2. they have been liked by a group of users who rank highly in the content features similarity score.

User content features and item content features consist of latent features extracted through matrix factorisation.

In addition to the above, this study also designed, developed and implemented a web-based prototype hybrid recommender system which can be deployed in an enterprise application container for multiple users.

1.7 Structure of the Document

Chapter 2 provides the background and literature review on the key topics of data mining and recommender systems. The chapter introduces the main paradigms, techniques and state of the art in recommender systems, and provides a detailed discussion of the different approaches and common applications of these systems. The relevant literature on recommender systems and machine learning is also discussed, including a description of the different types of recommendation systems, their approaches and structures. Chapter 3 is the Research Method chapter, which provides a detailed discussion of the approach that was followed in conducting this research. The chapter also provides a detailed explanation of the main ideas behind the design, development and implementation of the prototype hybrid recommender system that was used in this research. Given the strong focus on data mining, it was decided to conduct this research based on the life cycle of the CRISP-DM methodology, [Jackson, 2002; Shearer, 2000], and process model. The chapter discusses the CRISP-DM process model in detail, including its use in guiding this research. Each of the six phases that constitute the model is discussed and implemented (Section 3.3). The architecture of the prototype hybrid recommender system, including its components, are also presented and described in detail in this chapter. The results

obtained in this research are presented and discussed in Chapter 4, which also discusses the experimental setup, and describes the datasets and evaluation metrics that were used. The chapter also provides a summary of the methods used to test the performance of the implemented prototype hybrid recommender system. The results obtained are also presented, mostly in a graphical and easy-to-understand format. Chapter 5 analyses and discusses the findings of the research, focussing mainly on the results presented in Chapter 4. Lastly, Chapter 6 concludes the document and includes some ideas for future work.

Chapter 2

Background and Related Work

2.1 Introduction

Chapter 1 introduced the problem area and provided the context for this research. This chapter presents a comprehensive background discussion of the topics that will be covered. It also includes a detailed review of the literature related to this work. Different data mining techniques were used in this research to extract information and knowledge from the customer-related data that was obtained from the PLP Group, to be used for data mining and recommendation purposes. Section 2.2 introduces the concept of data mining. This is followed by a discussion of the different data mining techniques that were used as part of this research. These methods include classification, prediction, association rule mining, clustering, concept description and profiling. Data mining and recommendation techniques are used in a wide range of disciplines. In Section 2.3, examples are provided for the different areas in which data mining and recommendation techniques are used. The concept of recommendation and the different recommendation approaches are discussed in Section 2.4. There are content-based and collaborative filtering approaches, amongst others. Recommender systems face numerous challenges. A discussion of some of these challenges is provided in Section 2.5. In the telecommunications industry, the provision of value-added services and their recommendation to prospective and existing clients are essential to the survival and growth of a business in a tough and highly competitive environment. A detailed discussion of the literature related to this work is provided in Section 2.6, and it focuses on the work that has been done by others in the area of value-added service provision and hybrid recommendation. Section 2.7 discusses the measures and techniques that were implemented in the recommendation algorithms that were used in our prototype hybrid recommender system. These include matrix factorisation and singular value decomposition, as well as different similarity metrics such as Pearson's correlation coefficient. A summary is provided in Section 2.8 which concludes this chapter.

2.2 Data Mining, Text Mining and Recommender Systems

Data Mining, [Gera *et al.*, 2014; Hüllermeier, 2005; Jackson, 2002; Kruse *et al.*, 1999; Rud, 2001], involves the exploration and analysis of data, from which interesting and useful information can be obtained [Arndt and Gersten, 2001; Cisty and Bezak, 2013]. Finding useful patterns in data, or knowledge discovery in databases, is known by different names which include data mining, knowledge extraction, information discovery, information harvesting, data archaeology, and data processing [Fayyad and Smyth, 1996; Jackson, 2002]. A significant amount of data is collected from different domains, and the main aim of data mining is to extract meaning from it [Cios *et al.*, 2007]. As service provider companies expand their operations and customer base, they lose personal contact with their customers, and must find other ways of forming long-term relationships with these customers. One of the ways in which they can do this is by taking full advantage of the data produced through interaction with their customers [Berry and Linoff, 2004]. According to Gera and others in Gera *et al.* [2014], data mining is mainly focused on "building models" to interpret, predict and forecast user behaviour. They proceed to describe a model as an algorithm or composed of a set of rules that link a collection of inputs like ratings and user features to a particular target or outcome [Gera *et al.*, 2014]. The inputs can range from item ratings to content features while the outputs can be similarity and prediction metrics. In their text, data mining techniques are simply data modeling techniques. Commonly used data mining techniques for creating models include regression, neural networks, decision trees and association rule mining. One way in which models can be used is to predict outcomes,is using a process also known as "scoring" [Hsieh, 2004]. Scoring can be used to refer to metrics for calculating similarity either regarding content features or user ratings. Scores can be numbers, like rating predictions, strings or whole data structures as in content-feature prediction. In business environments such as those in which the PLP Group operates, value scores can be used to classify a list of customers into most loyal and least loyal, or into most likely or least likely to respond to a recommendation, or into most likely or least likely to default on a credit repayment [Berry and Linoff, 2004; Cios *et al.*, 2007; Hsieh, 2004]. Data mining tasks include classification, estimation, prediction, association rule mining, clustering, description and profiling [Berry and Linoff, 2004; Jackson, 2002; Rud, 2001].Matatov *et al.* [2010] classify *Prediction* into *Classification* and *Regression* as illustrated in a topology in Figure 2.1:

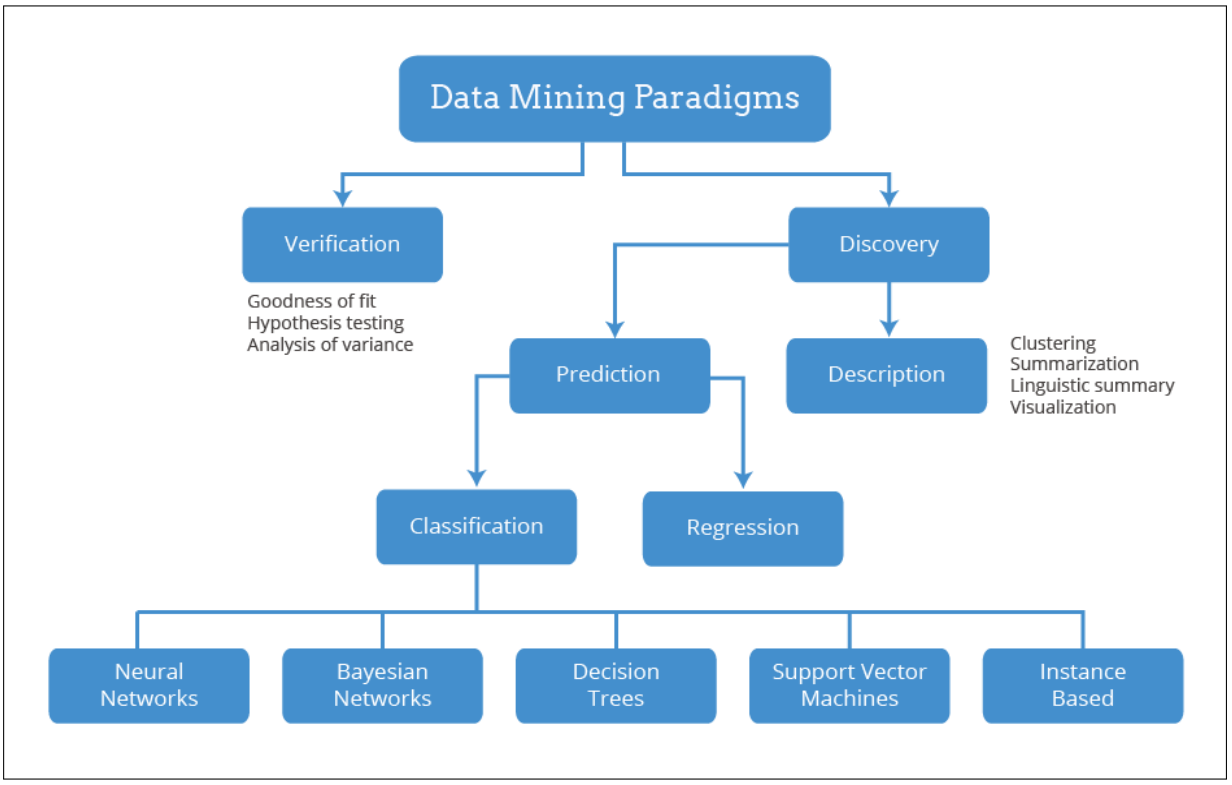


FIGURE 2.1: Data Mining Techniques

[Maimon and Rokach, 2009]

2.2.1 Classification, Estimation and Prediction

Human beings tend to use *classification* as a tool for understanding different phenomena. For example, nature is classified into living and non-living things; living things are further classified into plants and animals, people are classified into male and female, customers are either creditworthy or not creditworthy, there are mobile and landline phone numbers, there are postal and physical addresses, and so on. Within the context of data mining, the classification task involves the assignment of data items into different classes. The purpose of classification,[Blockeel *et al.*, 2002; Pechenizkiy *et al.*, 2008; Phyu, 2009; Turney, 1995], is to predict, as accurately as possible, the class to which a data item belongs. For example, the data items that are collected and stored in different repositories can each be assigned to a different class, as long as discrete-valued data items are used [deVille, 2006; Han, 2011]. Classification techniques include neural networks [Curram and Mingers, 1994], decision trees [Blockeel *et al.*, 2002; deVille, 2006; Turney, 1995], nearest neighbour [Berry and Linoff, 2004; Phyu, 2009; Spiegel *et al.*, 2009] and link analysis. A classification technique is applied to a group of data records (data items) to produce a model. The records constitute the *training set* [Phyu, 2009]. Each record consists of attributes, one of them being the *class* attribute, which indicates the class to which a particular record has

been assigned. The model produced during the training phase is applied to a *test set*, which consists of records whose individual classes are unknown, and the task is to assign each record, as accurately as possible, to a particular class. The different data mining techniques leverage on the availability of data, powerful computing facilities and predictive analytics software.

With classification, each record is associated with a nominal or discrete-valued outcome. *Estimation*, on the other hand, associates each record with a continuous-valued outcome. Given some detail about a user or similar users, an estimation model produces a value for the unknown feature such as gender, income, height, or age. Examples of estimation tasks include a motor vehicle insurance company estimating the number of claims an insured car owner will make in a year, a health insurance provider estimating how much an insured individual will spend in a year on their medical aid scheme, a revenue collection agency estimating how much money will be generated during the tax season, and so on. Data mining techniques such as regression analysis and neural networks are more suited to the task of estimation [Curram and Mingers, 1994; Han, 2011; Ricci *et al.*, 2011].

Prediction is similar to classification and estimation [Sarwar *et al.*, 2002; Vozalis and Margaritis, 2006]. The task involves predicting some future behaviour or estimating future value [Han, 1996; Hand *et al.*, 2001]. Prediction can be regarded as an extrapolation of current trends. Any one of the algorithms used for the estimation and classification tasks can also be used for prediction. This can be done by splitting the data into *training* data which is made up of known values and *testing* data made up of unknown values. [Berry and Linoff, 2004; Cios *et al.*, 2007; Curram and Mingers, 1994]. Examples of prediction tasks include predicting whether a particular PLP Group subscriber will continue or cancel their subscription to the service, a university department dealing with alumni affairs predicting whether certain alumni will donate to the university's Alumni Fund, predicting which of the PLP Group's subscribers will sign up for one of its value-added services, as offered through its GetMore scheme, and so on.

2.2.2 Association Rule Mining

Business enterprises such as the PLP Group collect and store vast amounts of data about their customers. Such data can be processed and analysed to learn more about customers and their behaviour as product and/or service subscribers. The obtained information can be used, in the case of the PLP Group, for example, to promote those value-added services for which subscriptions are too low, or to contact customers who are likely to cancel their subscriptions, "entice" them to maintain their subscriptions (customer relationship management), and so on. The association rule mining technique is often used to extract interesting relationships among

dataset items (that is, records). In other words, the task of association rule mining is to determine which items belong together [Cios *et al.*, 2007; Larose, 2006; Ma *et al.*, 1998; Mashat *et al.*, 2013; Merceron and Yacef, 2008; Vaidya and Clifton, 2002]. *Association rules* are used to specify the relationships discovered among the items in a dataset. An association rule is expressed using the form $X \rightarrow Y$, where X and Y represent sets of items that occur frequently in a dataset, also known as *frequent itemsets*. The set X is the *antecedent* of the rule and Y is the *consequent*. A *transaction* is a subset of items that belong to a dataset. For example, in a retail market environment a transaction could be the set of items purchased by a customer.

Support and *confidence* are the two measures used to determine the strength of an association rule [Gera *et al.*, 2014; Ma *et al.*, 1998; Sharma *et al.*, 2012]. The support s is the proportion of transactions that contain both X and Y , and the confidence c is the percentage of transactions containing X, that also contain Y [Larose, 2006; Tobias, 2001]. A rule has *minimum support* if its support is greater than a specified threshold value. It has *minimum confidence* if its confidence is greater than a specified threshold value. In the retail and similar environments, the association rule mining technique is used to perform market basket analysis [Agrawal and Srikant, 1994; Ma *et al.*, 1998], which seeks to enable a retailer to understand a customer's buying pattern or behaviour, and thereby to maximise profit by promoting items that the clients often purchase together.

2.2.3 Clustering

Clustering is a data mining task which, according to [Berry and Linoff, 2004, p.11],

> "*Clustering is the task of segmenting a heterogeneous population into a number of more homogeneous subgroups or clusters. What distinguishes clustering from classification is that clustering does not rely on predefined classes. In classification, each record is assigned a predefined class on the basis of a model developed through training on pre-classified examples.*"

Clustering maximises the similarity between records that belong to the same cluster [Shepitsen *et al.*, 2008; Shinde and Kulkarni, 2011]. Records that belong in different clusters are less similar to one another. Various measures such as the Euclidean distance [Berkhin, 2006], and Pearson's Correlation are used to place the records in a dataset into different clusters. Clustering algorithms include K-means [Berkhin, 2006], Self-Organising Maps (SOM), DBSCAN and others [Farajian and Mohammadi, 2010; Lai and Liaw, 2008; Ungar and Foster, 1998]. Clustering

techniques are applied in different areas such as medicine, commerce, education, geology, and so on [Li and Kim, 2003]. For example, clusters of medical symptoms might indicate different diseases. In a business environment, customer attributes might be used cluster customers into different market segments. Business organisations may divide their customer base into different clusters, with each cluster consisting of customers whose buying habits are similar, and this information can be used to promote products or services that best suit the profile of the members in the different clusters [Cios *et al.*, 2007; deVille, 2006; Han, 2011; Hsieh, 2004].

2.2.4 Concept Description and Profiling

There are *predictive* and *descriptive* data mining techniques [Fayyad *et al.*, 1996]. Predictive techniques use dataset records to create models for predicting the trends, properties and behaviour of new datasets. Descriptive techniques, on the other hand, provide concise and summarised descriptions of data, focusing on interesting and relevant properties of the data [Shaw *et al.*, 2001]. Descriptive data mining is best understood in terms of *concept description*, a method that that uses domain knowledge to group dataset records. The term *concept* is used to refer to the grouping of data into different "categories" such as *frequent-fliers*, *stokvel-members* and *NSFAS-students* [Shaw *et al.*, 2001]. As a data mining task concept-description generates descriptions which characterise and compare data. Concept-description focuses on the terse and compendious depiction of data. Comparison provides descriptions which compare multiple data collections. Such forms as graphs, charts and logical rules can be used to present concept description.

In business environments, customer *profiling*, [Adomavicius and Tuzhilin, 2001; Berkhin, 2006; Berry and Linoff, 2004; Shaw *et al.*, 2001], is a widely used method of applying the knowledge obtained about individual customers, from which a business can make important marketing decisions. According to Shaw *et al.* [2001], a customer profile is defined as "*a model of the customer, based on which the marketer decides on the right strategies and tactics to meet the needs of that customer.*" A customer profile has *factual* as well as *behavioural* components [Adomavicius and Tuzhilin, 2001; Farajian and Mohammadi, 2010; Hsieh, 2004]). The factual profile contains details or information about a customer such as their gender, professional status, ethnicity, educational qualifications, age, and so on. The behavioural profile models customer behaviour and is mainly based on transactional data [Adomavicius and Tuzhilin, 2001]. There are different applications of customer profiling. For example, customer profiling techniques can be used to identify customers most likely to cancel their subscription to a service, determine how much a customer spends on the products or services on offer, determine the creditworthiness

of a potential customer, and so on. Profiling is performed using demographic, geographic, psychographic and behavioural information. Some of the powerful techniques that are used in customer profiling include clustering, association rules, decision tables and decision trees [Cios *et al.*, 2007; Farajian and Mohammadi, 2010; Han, 2011; Shaw *et al.*, 2001].

2.3 Data Mining and Recommender System Applications

2.3.1 Applications of Data Mining

Data mining techniques play a critical role in numerous applications [Agrawal, 2013; Han, 2011; Hand *et al.*, 2001; Pal, 2011; Shaw *et al.*, 2001]. Search engines and business intelligence systems are two of the most popular areas where data mining applications are implemented. A few examples of the applications of data mining are discussed in the following subsections.

Intrusion Detection and Prevention Systems

Intrusion detection and intrusion prevention systems, [Lee *et al.*, 1999], are used to monitor network traffic or system executions to detect malicious activities [Han, 2011; Schultz *et al.*, 2001; Shah *et al.*, 2003]. The following are some of the ways in which data mining methods are used to enhance performance in these systems:

- Incorporating new data mining algorithms for intrusion detection,
- Using association, correlation, and discrimination pattern analyses techniques to select and build discrimination classifiers
- Analysis of stream data

Wenke Lee and others carried out several studies on the application of data mining in intrusion detection and prevention. In one of their papers, [Lee *et al.*, 1999], they sought to develop a dynamic and adaptive intrusion detection system by employing data mining techniques. Such a system would adjust itself in response to new threats with minimal human involvement. Their system was evaluated by MIT Lincoln Labs and their model performed poorly with the number of false alarms far much higher than 0.5 on the ROC Curve.

In a similar line of study, Matthew G. Schultz in Schultz *et al.* [2001] and two other researchers focused their line of work on using data mining frameworks to identify new malicious executable

software. Their argument was that most of the attacks on Windows systems were a result of malicious programs. They used various algorithms and evaluated them using percentage detection. The Naive Bayes algorithm performed better than other algorithms with a detection rate of over 97%.

Unlike in the studies highlighted above, Hiren Shah and his team sought to use a single data mining technique on behalf of the government in response to the September 11 attacks in the United States [Shah *et al.*, 2003]. They used the clustering technique complemented by matrix factorisation. The results they obtained from evaluating their system were highly inconclusive.

Financial Data Analysis

The data collected in the banking and finance industries is often complete, reliable, and of high quality, which makes it convenient to analyse and mine it systematically. Loan payment prediction and customer credit-worthiness analysis are critical to the business of a bank,Han [2006], as is the clustering of customers for market segmentation and targeted marketing purposes. Data mining techniques are also used to detect money laundering and other fraudulent crimes [Farajian and Mohammadi, 2010; Han, 2011; Zhang and Zhou, 2004].

Mohammad Ali Farajian and Shahriar Mohammadi in Farajian and Mohammadi [2010], noticed that banks were now faced with unprecedented competition which raised the need to look for effective ways to retain customers and acquire more. They sought to help banks solve their problem by using *K-Means* and the *Apriori* algorithms to analyse the behaviour of customers. They hoped that the results of their study would help inform the development of a more effective marketing strategy. Their two-step model first developed customer clusters using the K-means algorithm before analysing the behaviour of the customers in each cluster by using the Apriori algorithm. Dongsong Zhang and Lina Zhou attributed this unprecedented increase in competition in the banking sector to globalisation [Zhang and Zhou, 2004]. However, they too, like Mohammad Ali Farajian and Shahriar Mohammadi, turned to data mining to solve the marketing problems facing the banking sector. In their approach, they sought to investigate various data mining techniques and their applicability in the activities of the banking sector. They faced the problem of scalability, performance and diverse data sources in the banking sector.

In a slightly different angle, Efstathios Kirkos and his research team in Kirkos *et al.* [2007] used data mining to expose fraudulently financial statements released by firms. They limited their study to investigating the relevancy of Neural Networks, Decision Trees and Bayesian Networks

in the identification of fraudulent financial statements published by companies. Their study had too many variables and lacked adequate fraudulent data to feed to their system.

Applications of Data Mining in the Retail and Telecommunication Industries

The retail, health and telecommunications, [Weiss, 2005], industries are classic application areas for data mining, in which vast amounts of data are collected on sales, customer shopping history [Han, 2011; Moro *et al.*, 2010; Nabavi and Jafari, 2013; Rud, 2001], goods transportation, consumption, service, and so on. Retail data mining can help in a variety of ways, such as:

- Identifying customer buying behaviours, discovering customer shopping patterns and trends [Berry and Linoff, 2004; Cios *et al.*, 2007],
- Improving the quality of customer service,
- Achieving better customer retention and satisfaction,
- Enhancing goods consumption ratios,
- Designing more effective goods transportation and distribution policies, and
- Reducing the cost of doing business [Han, 2011; Shen and Chuang, 2009]

There has been extensive study in the use of data mining techniques in the management customer relationships and in direct marketing [Adomavicius and Tuzhilin, 2001; Cheung *et al.*, 2003; Nabavi and Jafari, 2013; Rud, 2001; Shen and Chuang, 2009]. Of these, Ćamilović in Camilovic [2008], specifically studied the application of data mining in customer relationship management in the telecommunications industry. The reasons for focusing on the telecommunications industry have to do with the availability of large quantities of data in the sector, and increased competition which is matched by high churn rates. In India Khushboo Makwana and others carried out a research to establish the factors that caused consumers to switch mobile phone providers. They discovered that value-added services bundled with each cellphone package were the single most significant variable [Makwana *et al.*, 2014]. However, Singh and others had already done a study to investigate the role of value-added services in shaping the Indian telecommunications industry [Singh *et al.*, 2011].

Application of Data Mining in Recommender Systems

Recommender Systems (RS) are designed to help individuals and organisations manage the information overload problem by making it easier for them to make evaluative decisions. Recommender systems help consumers by automatically recommending items that are most likely to be of interest to them. These products include books, compact discs (CDs), movies, restaurants, online news articles, and so on. Recommender systems may use content-based, collaborative or hybrid approaches. The hybrid approach combines content-based and collaborative methods [Baitharu and Pani, 2013; Han, 2011]. Collaborative recommender systems focus solely on analysing historical interactions, while content-based recommendation techniques rely on profile attributes Melville and Sindhwani [2010]; and hybrid techniques combine both demographic data and user usage or ratings.

The degree of personalisation in recommendation systems can be different for each context and domain. Galland Galland [2012] has classified the recommendations of various systems into four different types:

1. **Generic**: everyone receives same recommendations.
2. **Demographic**: everyone in the same category receives same recommendations.
3. **Contextual**: recommendation depends only on current activity.
4. **Persistent**: recommendation depends on long-term interests.

2.3.2 Recommender System Applications

Recommendation systems (see examples in Appendix A) are used in a variety of contexts such as online stores and communities, music players, media streaming sites, social webs and so on [Li *et al.*, 2012]. Social network sites like Facebook, Twitter and LinkedIn use various recommendation algorithms to connect users with one another. Google also uses recommendation techniques to filter spam messages from Gmail, its email facility. [Asabere, 2012] says that "*[Recommender Systems] apply machine learning and data mining techniques to filter undetected information and can predict whether a user of a system would like a given resource based on his/her interests and preferences*". The important points to note in Asabere's description are *undetected, predict and filter*. By using data mining techniques, recommender systems can *detect* hidden patterns in both content features and rating information of items and users. The *filtering* function of recommender systems enables them to ameliorate the issue of information overload while that is only possible if recommender systems can *predict* what is relevant to the active user. The most concise definition of a recommender system is provided by Robin Burke in [Burke, 2007]

who defines recommender systems as information agents that provide personalised suggestions for items that are most likely to have utility for a user [Galland, 2012; Good *et al.*, 1999]. A few of the main application domains of recommender systems are discussed in the following subsections (2.3.2 - 2.3.2):

Recommender Systems in Education

Recommender systems are now being used in educational environments to personalise teaching and learning approaches [Gera *et al.*, 2014; Li *et al.*, 2012; Pechenizkiy *et al.*, 2008]. Data mining techniques can discover useful information that can be used as part of a formative evaluation exercise, to assist educators in establishing a pedagogical basis for decision-making during the design or modification of the teaching environment or approach [Mashat *et al.*, 2013].

Luiz Fernandez-Luque and others in Fernandez-Luque *et al.* [2009] studied the possibility of using recommender systems to reduce the burden of information overload in providing health education over the internet. The intended to investigate the potential of adaptation of health education to one specific person through a computerised process. While Luiz and his team limited their study to the health education sector, Hendrik Drachsler and others, [Drachsler *et al.*, 2008], studied the application of recommender systems in the field continuing and adult education. Life-long learners are distance learning programme participants who get their material from different institutions. Such learners would benefit from a recommender system that introduced them to new learning material that is relevant to their needs and preferences.

In another study, Hendrik Drachsler collaborated with others in studying the application of recommendation systems in technology-enhanced learning [Manouselis *et al.*, 2011]. They designed a system that would recommend relevant materials to students. They broadened the scope of their study by comparing the various recommender systems that purport to support technology-enhanced learning.

Geyer-Schulz and his colleagues investigated the role played by recommender systems and their potential application in the academic and scientific space of a Virtual University intending to exploiting their capabilities to enhance the provision of tutoring and consulting services of a Virtual University automatically [Geyer-Schulz *et al.*, 2001]. In their study, they discovered that recommender systems could be used to ameliorate some of the challenges facing Virtual Universities like:

1. teaching growing numbers of students without an unsustainable increase in staff size;

2. supporting continuing and distance learning for more and more citizens at a socially acceptable cost;

3. and, as a corollary, allow universities to accommodate an increasingly heterogeneous student set.

4. Dealing with information overload caused by the exponential Internet growth the ever increasing number of researchers worldwide.

[Geyer-Schulz *et al.*, 2001]

Recommender Systems in Commerce

Recommender systems are described as applications that are used by e-commerce sites to provide users with personalised information to aid them to make well-informed decisions [Carlsson and Walden, 2002; Pattamavorakun and Pattamavorakun, 2010; Ricci, 2010; Sarwar *et al.*, 2000; Tran and Cohen, 2000].

In e-commerce, according to [Tran and Cohen, 2000], potential customers may wish to receive product recommendations by email, SMS, MMS or other instant messaging applications. In their study, Tran and Cohen used a collaborative filtering system with the argument that clients tend to show interest in commodities which similar users have preferred in the past. They developed a system that combined user-features with rating information. Sarwar and other researchers in [Sarwar *et al.*, 2000] realised that consumers of e-Commerce sites needed an accurate recommender system on which they can rely. However, accuracy usually came at the expense of speed and computer performance. Therefore, they sought to optimise existing recommendation algorithms and also tried to develop new ones. They introduced Singular Value Decomposition to reduce dimensions and improve algorithm performance.

Using the principles developed in Sarwar *et al.* [2000], Suwat and Suwarin Pattamavorakun, [Pattamavorakun and Pattamavorakun, 2010], used the similarity algorithms to improve personalised value-added services recommendations to clients in the Thai mobile commercial space. In this study, value-added services were limited to any service other than SMS and MMS provided by mobile phone companies.

Web Service Recommender Systems

Recommender systems are now more popular on the Web [Burke, 2007; Herlocker *et al.*, 2004], and are commonly used for research (e.g. GroupLens and MovieLens data) and as part of online business-to-consumer and consumer-to-consumer commercial websites (e.g., Amazon.com and CDNow.com, Gumtree) that recommend several ways in which consumers can find products they might like to purchase [Medhi and Dakua, 2005]. The popular website, Amazon.com, uses a recommender system to provide personalised recommendations to individual customers interested in the items offered by its online store [Good *et al.*, 1999].

Adaptive websites sites like Amazon, Gumtree and IMDb use automated recommender systems. Robin Burke in Burke [2007] surveyed the subject of two-step hybrid recommender systems. They compared four different recommendation techniques provided in literature and seven different hybridisation strategies. Under recommendation techniques, he identified collaborative, content-based, demographic and knowledge-based systems [Burke, 2002]. In his study, he concluded that feature augmented hybrid systems showed the best performance [Burke, 2007]. Contributing to the subject of web-based systems, Ziegler [2005], felt there was little research done on distributed recommender systems. He argued that collaborative filtering performed poorly in distributed systems due to the diversity and sparsity of rating information in systems that geographically partition their data. To solve that problem, he advocated the use of a hybridisation technique that catered for incomplete user rating and trust data.

In another dimension, Satoshi Niwa and two others in [Niwa *et al.*, 2006] developed a novel web page recommender system which covered the whole internet, by using Folksonomy and Social Bookmark [Niwa *et al.*, 2006]. They admitted that previous attempt to produce such a system had failed due to a large amount of web pages and unavailability of user-preference data. They tried to solve the problem of lack of user-preference data for web pages by using Social Bookmarks. Their results were profoundly affected by the subjectivity of user responses.

Recommender Systems in Industry

Given the overwhelming information explosion, a lot of time is consumed in performing operations such as searching, extracting , analysing and processing the vast amounts of collected data. Worst still, it is known that activities that involve human beings are inevitably subject to human errors, which can lead to poor or wrong decisions being taken [Sophatsathit, 2013]. Using recommender systems can help to minimise the time taken to perform such demanding tasks.

Nalinee Sophatsathit, Sophatsathit [2013] investigated the use of recommender systems in car dealerships. Nalinee tried to develop a system that matched the needs of the buyers with those of the used car dealers. The study hoped the developed system would save buyers time and improve sales for the used car dealers. However, the research did not use data mining techniques. It used statistical methods. Francesco Ricci studied the application of mobile recommender systems in the tourism sector [Ricci, 2010]. He observed that tourism was a good area for the use of mobile applications and a good number of mobile services were already being provided to support tourists before, during and after their journey. Mobile recommender systems in the tourism sector help recommend places of interests and restaurants to tourists. They also help people explore the city and pick movies on the move.

2.4 Recommendation and Recommender Approaches

It was mentioned in Chapter 1 that recommender systems help consumers by recommending products or services that may be of interest to them. These may include books, compact discs, movies, restaurants, online news articles, and so on [Ricci *et al.*, 2011]. There is a widespread problem of having "too much to choose from", also known as the information overload problem. It not only occurs when one is searching for information, but it also occurs in other cases, such as when one has to manage constant, incoming streams of information such as e-mails, as well as radio and television news stories. People find it difficult to process such information, thus making it difficult for them to use it to make good quality and well-informed decisions. For example, due to information overload, it may not be easy to distinguish between interesting and uninteresting news stories or to differentiate between important and junk e-mail messages [Setten, 2005]. To address the information overload problem and to alleviate its negative effects, recommender systems have been developed to enable users to make well-informed choices with confidence, given the vast amount of information at their disposal.

Figure 2.2 illustrates the general idea behind a recommender system:

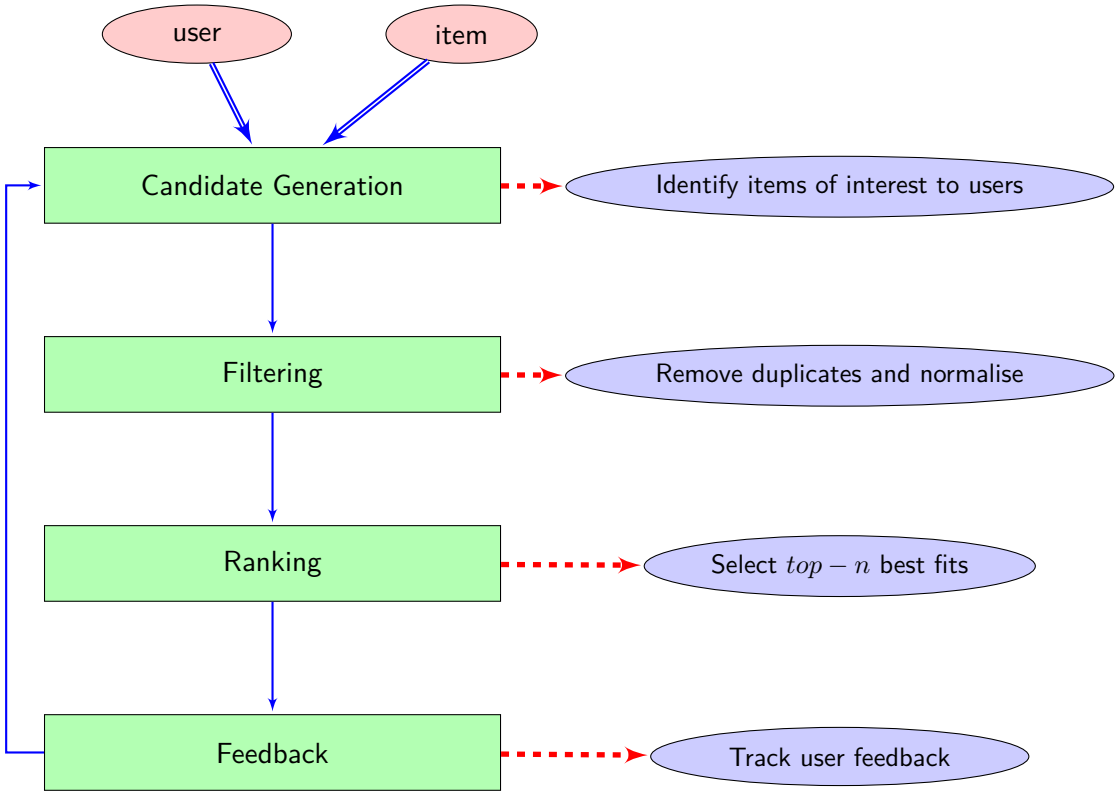


FIGURE 2.2: The Recommendation Process

Figure 2.2 above illustrates a generic recommendation process. For the recommendation process to be effective, some *user* and *item* information must be available and fed into the system. The system needs to know if the active user has previously indicated a preference for certain items. If he has, the system would then seek to establish if those items have similar features to other items in the database that it can identify as *candidate items* for recommendation. If the user has not previously used the system, then the system needs to know the user's attributes so it can compare the user to other users who have used the system before. Filtering and ranking involve the task of presenting a manageable quantity of information to the user. Only the *top* n most relevant items are presented to the active user as recommendations. In order for the system to improve its effectiveness, it requires the user to give feedback on its performance. That feedback will be used to inform future recommendations for the current user. Feedback can be both explicit and implicit. Explicit feedback includes user ratings and comments, while explicit feedback can be gathered by the number of clicks made by the user or the length of time the user spent viewing an item.

There are diverse variations of recommender systems identified in the literature. There are collaborative, content-based, utility-based, demographic and knowledge-based systems [Burke, 2002; Lops *et al.*, 2011]. Each one of them has its own advantages and limitations [Aksel and

Birtürk, 2010; Asabere, 2012; Pazzani, 1999]. Hybrid recommender systems are obtained by combining multiple recommendation systems, such as when content-based and collaborative systems are combined to form a hybrid system. The goal behind recommender systems research is to boost the relevancy of the recommendations produced. Hybrid recommenders play an important role in achieving this goal [Jannach *et al.*, 2011]. The various types of recommender systems are discussed in the following subsections (2.4.1 - 2.4.6).

2.4.1 Content-Based Filtering (CBF) Recommender Systems

Dumais [1992] suggests that the earliest information filtering systems were based on content. CBF systems recommend items based on their content description. To meet users' needs, personalised information systems adapt themselves, including the item-related information they retrieve. Such systems require some information about user characteristics, interests, and preferences. User-related knowledge is usually stored in the form of user profiles that represent users within the system [Setten, 2005]. CBF recommender systems give recommendations that are similar to the items a user has preferred in the past. This method focuses mainly on the association between user profile attributes and item attributes. Some of the key problems associated with content-based recommender systems include limited content analysis [Spiegel *et al.*, 2009], and over-specialisation [Asabere, 2012; Ricci *et al.*, 2011].

Pure CBF systems are very few in literature. Pazzani in [Pazzani, 1999] set out to develop a system that used content features of items and demographic attributes of users. The system he built was for recommending sources of information like Web pages and news articles. The system was meant to determine which web pages could be recommended to given users. The data at his disposal included the content features of the web page, the ratings provided by the user on the previous pages he/she had visited and the content features of these web pages, the ratings of the other similar users to that page and the ratings of these similar users on other pages pages visited by the active user and the demographic features of all the users. As can be seen from the available data, the system inclined itself towards CF filtering due to its reliance on other user's ratings and the ratings of the active user on other pages. His results ranged from 57% to 71% with the best results obtained from collaborative systems built through content features. This was a case for a hybrid system. He attributed the poor results of the CBF system to lack of adequate data and to its inability to take into cognizance user preferences.

Pasquale Lops and others describe CBF systems as recommendation systems that attempt to recommend items based on item-item similarity [Lops *et al.*, 2011]. This definition is classical illustration of the deficiency of CBF systems: inability to accommodate new users. In their

study, they looked at the state-of-the-art and trends in CBF systems. They provided an architecture of CBF systems which relied heavily on creating a user profile first before creating recommendations for him. However, despite the blatant drawbacks of CBF Systems, Pasquale and his colleagues managed to identify three advantages: user independence, transparency and new item. The system did not rely on other users, but the active user to create his own profile. Since the system used only item content features, it was clear and easy to explain how it operated. Any new item could easily be integrated into the system via its attributes.

Despite the above positives, their system had a lot of drawbacks. The main one being the inability to provide recommendations for new users, which together with over-specialisation have made CBF systems unpopular [Burke, 2007; Symeonidis *et al.*, 2007].

John Bruntse Larsen in his MSc Informatics Thesis, [Larsen, 2013], also lists some shortcomings of CBF systems similar to the ones compiled by Pasquale Lops and others above. His main worry was that since CBF systems relied heavily on creating profiles, it was not easy to automatically create profiles which meant that the task of getting more information about a user will remain a manual and a slow one. However, manual profiling is usually impractical as users may wish to remain anonymous.

From the discussion above, we can safely conclude that CBF systems have three main disadvantages [Larsen, 2013; Lops *et al.*, 2011; Pazzani, 1999]:

1. Deficient Content Analysis: Content-based filtering algorithms are limited by the natural limit in the number and type of attributes that can be associated with the objects they recommend [Larsen, 2013; Pazzani, 1999]
2. Overspecialisation: Novelty and serendipity suffer in the face of CBF systems [Lops *et al.*, 2011]
3. New user: New users without a profile do not get accurate recommendations [Larsen, 2013; Lops *et al.*, 2011; Pazzani, 1999].

It is these major shortcomings of CBF systems which forced researchers and developers to look for answers from CF systems covered in Section 2.4.2.

2.4.2 Collaborative Filtering (CF) Recommender Systems

The principle behind collaborative filtering recommender systems is that similar users tend to prefer similar items [Agrawal *et al.*, 2009; Amatriain, 2014; Bell and Koren, 2007; Good *et al.*,

1999; Herlocker *et al.*, 2004; Resnick *et al.*, 1994]. For recommendation purposes, the collaborative filtering approach relies on a database containing user preferences for certain items [Sarwar *et al.*, 2001]. Items are recommended based on the similarities of users who rated the items. To predict the items the active user will most probably like or show interest in, the collaborative filtering approach exploits information about past behaviour or the opinions of an existing user community. Neighbourhood methods and latent factor models play an important role in collaborative filtering. Neighbourhood methods focus primarily on determining the latent relationships between items and other items or between users. Actually, according to Bell and Koren in [Bell and Koren, 2007], the major approach to collaborative filtering is neighbourhood based ("k-nearest neighbors"), where an active user's rating for an item is interpolated from the ratings of similar items or users. Latent factor models use a range of factors inferred from the rating patterns to characterise items and users [Amatriain, 2014; Bell and Koren, 2007; Linden *et al.*, 2003].

A collaborative filtering approach involves the following steps

1. Identify the set of ratings made by the target/active user.
2. Using a similarity function, identify a neighbourhood of users who are the most similar to the target user.
3. Identify the items which these similar users have preferred.
4. For each one of these items, predict a rating the target user would assign to it.
5. Using the predicted rating, recommend the *top N* items.

CF systems are further classified into *model-based* and *memory-based* collaborative filtering approaches [Breese *et al.*, 1998; Sarwar *et al.*, 2001; Yu *et al.*, 2004].

Memory-Based Collaborative Filtering

Memory-based CF algorithms use the entire *user-item* database to perform collaborative filtering (that is, to generate a prediction) [Breese *et al.*, 1998; Ma *et al.*, 2007; Pennock *et al.*, 2000; Sarwar *et al.*, 2001]. These algorithms are referred to as memory-based since the entire rating database they manipulate is stored in memory and used directly for recommendation purposes. To make recommendations the rating data is used to determine the similarity between users or items. Memory-based algorithms generate a prediction by using the ratings of relevant users

or items. There are two types of memory-based CF algorithms. There are *user-based* CF algorithms and *item-based* CF algorithms. A user-based CF algorithm uses a similarity measure to identify a group of users who are similar to the active user (a set of k nearest neighbours). Once this group has been identified the algorithm uses their item ratings to estimate a rating value for the items the active user has not yet rated. Using the predicted ratings, the top N items are recommended to the active user. To date, user-based collaborative filtering algorithms have proven to be the most accurate and efficient approach for developing recommender systems and is used extensively in commerce [Deshpande and Karypis, 2004]. The item-based CF algorithm starts by identifying the group of users who have rated the target item and then looks for other items that have been rated by this group of users. The algorithm uses a similarity function to compute the similarity score between these items and the candidate item. This enables ithe algorithm to select the top k most similar items (neighbours). The item-based CF algorithm then uses the ratings of these k items to predict the ratings for the target item. In this research both the user-based/item-based approaches were adopted as one of the components in the prototype hybrid recommender system that was developed as part of this research.

The problems that bedevil memory-based algorithms can be approached from two fronts [Ma *et al.*, 2007]

- calculating similarity scores between users or items; and
- identifying algorithms to predict missing data.

There are too many algorithms put forward in literature for calculating similarity. Some of these algorithms are highlighted in Section 2.7.5 below. In this study, we use both the Adjusted Cosine Similarity and the Pearson Correlation Coefficient as measures for similarity. However, although memory-based approaches have been widely used in recommendation systems [Resnick *et al.*, 1994], the problem of inaccurate recommendation results still exists in both user-based and item-based approaches [Linden *et al.*, 2003].

The main problem of memory-based approaches is the dearth of rating data which results in a very sparse user-item matrix. Many recent algorithms have been proposed to alleviate the data sparsity problem. These include multiplicative normalisation and Matrix Factorisation. Both these are covered and applied later in this text.

Model-Based Collaborative Filtering

In data mining algorithms are applied to datasets to create models that perform different tasks such as classification and prediction. There are many model-based collaborative filtering algorithms such as matrix factorisation or latent factor models, clustering models, association rule mining models and Bayesian network models. A model-based recommendation system uses a dataset of ratings to create a model. The generated model is then used to make predictions using test data, without having to use the full dataset each time. Model-based algorithms have been developed to deal with some of the problems that affect memory-based algorithms which, for example, perform poorly against huge datasets. Model-based algorithms are fast and also capable of scaling up when the data volume increases [Sarwar *et al.*, 2001]. Some of the advantages of model-based systems include prediction speed and the avoidance of over-fitting. However, these systems tend to be rigid in the sense that they rely on pre-determined models leaving little room for newly-generated relationships. Furthermore, model-based systems use test data which may not be as accurate as the whole data population [Breese *et al.*, 1998; Pennock *et al.*, 2000].

Besides the *first-time-user* or *cold-start* problem, collaborative approaches also exhibit the *newly-added item* problem, according to which a new item must have a considerable number of ratings from other users before the algorithm can accurately recommend it.

2.4.3 Demographic Recommender Systems

Demographic recommender systems, [Pazzani, 1999; Vozalis and Margaritis, 2007], use demographic information to identify the types of users that prefer a given item. Demographic data can include age, gender, education, location, marital status, religion and so on of the people who showed interest in an item. However, obtaining demographic information can be difficult. People usually detest revealing more about themselves. Imagine asking people to provide their age, occupation, marital status, and so on, before purchasing bread in a supermarket. These systems identify users whose demographics resemble those of the active user u, and extrapolate from their ratings of the given item i how the active user would rate it [Burke, 2002].

2.4.4 Utility-based Recommender Systems

This category of recommender systems makes an appearance in Robin Burke's study [Burke, 2002] and further explored in [Martinez *et al.*, 2008; Prangl *et al.*, 2007]. These systems use a function to calculate utility (an Economics term referring to a commodity's ability to satisfy a

want). Items satisfying a similar utility like bread and buns should rank the same. Utility-based recommenders compute recommendations based on an abstract calculation of the utility of each item to the active user.

The major drawback of this system is that utility is a very subjective concept which is difficult to measure. Measuring utility is largely based on the assumption that humans are rational beings who are expected to use their scarce resources to buy food before the purchase of a luxury car.

2.4.5 Knowledge-based Recommender Systems

This category of systems also makes an appearance in Burke's [Burke, 2002]. The system uses inference to determine a user's aims and goals and then uses that information to match the user with items relevant to his goals [Prangl *et al.*, 2007]. For example, if a user wants to have something for breakfast there is an array of light meals to satisfy that need. A recommender system recommending breakfast items to a client would use that fore-knowledge to recommend relevant items. They compute their recommendations by using case-based reasoning processes where the users provide an example similar to his/her aims and the system infers a profile to find the better match product in the database [Burke, 2002; Martinez *et al.*, 2008].

2.4.6 Hybrid Recommender Systems

Multiple recommenders systems can be joined to form a hybrid recommender system, to alleviate the limitations of individual recommenders and to leverage their strengths to improve prediction performance. Hybrid recommender systems are created by bringing together collaborative filtering and content-based filtering systems [Grivolla *et al.*, 2010; Jannach *et al.*, 2011; Setten, 2005; Shani *et al.*, 2007; Shinde and Kulkarni, 2011]. This is the approach that was followed in implementing the prototype hybrid recommender system used in this research. Different hybrid system configurations are produced by combining mainly the collaborative filtering approach with other approaches.

Robin Burke in [Burke, 2002] identifies seven different hybridisation techniques. The following classes of hybrid recommender systems are produced by combining different recommender system approaches [Asabere, 2012; Burke, 2002].

Weighted Hybrid Recommender Systems

The similarity values or ratings of one or two recommendation algorithms are collated to create one recommendation. The principal value of a weighted hybrid recommender is that all of the system's recommendation techniques are used, compared and the best results forwarded to the user. The various algorithms used in the various techniques cancel out each others' shortfalls. However, use of this technique weighs heavily on the system's computing resources. Claypool and others in Claypool *et al.* [1999] developed a "Weighted Hybrid Recommender System" called *P-Tango*. They tested the results over a 3-week period and noticed that the combined recommendation approach was constantly more accurate than the other approaches throughout the period. However, Clayton and his colleagues compared the weighted approach against the CF and CBF techniques, not against the other hybridisation techniques.

Switching Hybrid Recommender Systems

The system alternates between different recommendation algorithms based on the context. In a new-user situation where no previous rating information is available, the CF technique would be of better use. The recommendation system uses an algorithm to switch between recommendation algorithms that constitute the hybrid recommender. Mustansar Ali Ghazanfar and Adam Prugel-Bennett in [Ghazanfar and Prugel-Bennett, 2010] developed a one-of-a-kind hybrid recommendation approach that used a Naive Bayes classification approach with the collaborative filtering technique. They were satisfied with the results they got concerning accuracy and scalability of the system. They compared their system to various CBF and CF algorithms.

Mixed Hybrid Recommender Systems

In this approach, two data models are built, a user data model and an item data model. Two recommendation are made using each model, but the results are combined before their presented to the user. Recommendations from several different user and item recommenders are presented at the same time. It is used in cases where it is where making many predictions at simultaneously is possible. Martinez and others in [Martínez *et al.*, 2009] developed a mixed TV hybridised recommender system. The recommendation engine used both CBF and CF techniques as a mixed approach to generate appropriate recommendations for television viewers. The mixed hybrid system avoids the *cold-start*, *new-user, new-item* problems which bedevil individual CBF and CF systems and can be trusted to recommend new items based on their profile descriptions even if they have not been previously rated by any of the other users [Burke, 2002; Glauber *et al.*, 2013; Martínez *et al.*, 2009].

Feature Combination Hybrid Recommender Systems

Content Features from both item and user recommendation data sources are combined into a single recommendation algorithm [Asabere, 2012; Burke, 2002]. Content features combination presents another efficient and effective way to merge CF and CBF. This technique treats collaborative information as simply additional content attribute data associated with each user or item and use content-based techniques over this combined data set.

In 1998, Basu and two other researchers, [Basu *et al.*, 1998], carried out a study on an inductive learning approach to the recommendation space that used both ratings information and content features of each item in predicting user preferences. The system they developed, *Ripper*, recommended items based on both user ratings and content features, and showed a good improvement in precision when compared to the pure CF approach. However, good results were obtained when they manually selected features to include in the algorithm. This system eliminated total reliance on the rating data which is usually sparse and forms the backbone of CF systems.

Cascade Hybrid Recommender Systems

In this approach, one recommender moderates the recommendations produced by another and then presents a single recommendation to the user [Asabere, 2012; Burke, 2002]. The concept of a cascade hybrid recommender system resembles that of a feature augmentation technique. However, cascade models make recommendations solely with the primary recommender, and just use the secondary recommender simply to moderate item ranking scores. For example, items that were scored and recommended by the main recommender may be re-ranked by the secondary recommender [Spiegel *et al.*, 2009]. It is a staged two-step process. [Burke, 2002] describes a knowledge-based and cascading collaborative recommender system called *EntreeC*. Like Entree, it used its knowledge of restaurants and a user's interests to produce recommendations. The recommendations are pooled in buffers of equal preference, and the collaborative algorithm is used to split tied scores by further sorting the suggestions in each buffer in descending order. Cascading enables the system to avoid using the secondary recommendation techniques on items that are already well-ranked by the first recommender [Asabere, 2012].

Feature Augmentation Hybrid Recommender Systems

Melville, Mooney and Nagarajan, [Melville *et al.*, 2002], developed and designed a feature argumentation system that incorporated CBF and CF techniques to predict items that may be of interest to a user. Their *Content-Boosted Collaborative Filtering (CBCF)* algorithm trains a content-based model over a training dataset to extrapolate ratings for unrated items on a sparse

matrix space. The dense rating matrix produced by this rating extrapolation technique is then used for collaborative recommendation by the recommender. CBCF ameliorates disadvantages of both CF and CBF techniques, and largely improved the estimations of the recommendation system they developed. The output from one recommendation technique is used as a feeder feature in the other system [Asabere, 2012; Burke, 2002; Spiegel *et al.*, 2009].

Libra, a hybrid recommender system developed by Mooney and Roy in 1999, [Mooney and Roy, 1999], was a content-based system for recommending books that utilised machine-learning algorithms to categorise words. The text data used by the system in [Mooney and Roy, 1999] included information about related authors and titles, similar to tagging data together with information generated by Amazon. These *collaborative* content features improved the quality of recommendations made by *Libra* [Asabere, 2012; Burke, 2002]. In a similar study, [Sarwar *et al.*, 1998], used results of results of the *GroupLens* filterbot model to help collaborative filtering systems to solve the data sparsity problem by tapping into the strength of content filtering techniques. A Feature Augmentation Hybrid Recommender System takes the results of CF as its input.

Meta-Level Hybrid Recommender Systems

Meta-Level Hybrid Recommender Systems too are two-step systems. The first system is trained on a training dataset, and the model it learns is used as input to another [Burke, 2002]. This actual recommender simply implements the algorithms learned and recommended by the first one. Although the general approach of meta-level hybrid systems resembles that of feature augmentation techniques, there exist some significant differences between both approaches. Instead of supplying the actual recommender with additional features, a meta-level contributing recommender provides a completely new recommendation space. In a feature augmentation system, a learned model produces features used to feed a secondary algorithm whereas the meta-level hybrid system uses the entire initial model, not individual algorithms. However, it is not always possible to produce a model that suits the recommendation technique of the primary recommender [Asabere, 2012; Spiegel *et al.*, 2009].

Meta-level hybrid recommendation systems first make an appearance in Balabanović [1997], who developed a web-based system called *Fab*. He further develops his system in Balabanović [1998]. In Fab, user-specific attribute selection algorithms perform content-based filtering using a method discussed in [Pazzani and Billsus, 2007] called *Rocchio* to maintain a vector of terms that profile the user's interest space.

The benefit of the meta-level method is that the learned model is a miniature representation of a user's preferences and the subsequent collaborative algorithm can be applied to this detailed representation more efficiently and effectively than on the raw rating data [Burke, 2002].

Table 2.1 below gives a synoptic view of all the hybridisation techniques as described in Burke [2002] and in Asabere [2012].

Hybridization Technique	**Description**
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time
Feature Combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature Augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

[Asabere, 2012; Burke, 2002]

TABLE 2.1: Different Hybridization Techniques

2.5 Recommender Systems Challenges

The exponential growth and increasing diversity of customers (users) and products (items) in recent years raises some key challenges for recommender systems. These challenges relate to producing high quality, accurate recommendations and performing many recommendations per second for millions of customers and products [Sarwar *et al.*, 2002]. There are many issues facing recommender systems. These include privacy, scalability [Sarwar *et al.*, 2002], sparsity, the first-rater or cold-start problem and context-awareness. Each type of recommender system has its advantages and disadvantages. Combining individual systems to form a hybrid system is a way of capitalizing on the strength of one form while neutralising the disadvantages of the other. Collaborative filtering systems are good in providing accurate recommendations, but they suffer from the first-rater or cold-start problem, which content-based systems address pretty well. Issues of trust, transparency and human-computer interactions, are also coming up in literature related to recommender systems [Ricci *et al.*, 2011]. To be effective in their

personalisation task, recommender systems must obtain information about the user, which may infringe on user privacy. On the other hand, without enough information about the user, the system may return some wild recommendations and result in a lack of trust in the system.

The challenge in collaborative recommender systems is the lack of knowledge about items that have few ratings [Fernandez-Luque *et al.*, 2009]. One other problem facing recommender systems is their reliance on technology. This reliance begets a derived type of challenge which is the limitations of technology devices, the short-comings of the internet, the impacts from the external factors, and the behavioural attributes of users, especially in rural Africa [Ricci, 2010].

Another challenge facing recommender systems is raised by [Ma *et al.*, 2011]. They argue that internet sites and applications pose new challenges for traditional recommender systems because most of them do not take into account the influence of relatives when making recommendations. Traditional collaborative recommender systems always ignore social relationships among users and focus only on the similarity of content features. Humans are social beings.

2.5.1 Limitations of Content-based Systems

As already explained in Section 2.4.1, Content-based Systems systems select items to recommend based on their content features. This poses a huge challenge because the system has to extract items, analyse them and extract content attributes. Extracting item features can be difficult for a machine to do especially for items like videos. Therefore, the human element is needed which makes them inappropriate for large recommendations [Cacheda *et al.*, 2011; Sarwar *et al.*, 2002].

Another major problem with content-based filtering is the inability of CBF algorithms to evaluate the quality of an item. Movies can be classified as either *Action* or *Comedy*, but there are many movies which combine varying degrees of both. When extracting features and recommending these items CBF items do not split hairs. The quality of an item is a very subjective concept which depends on culture, religion, taste, season or even the mood of the user.

Another most talked about short-coming of content-based filtering is its inability to find serendipitous items that are interesting for the user, that is, really good items that are not apparently related to the user profile. Something the user did not know he liked already. Novel or new items suffer.

2.5.2 Limitations of Collaborative Systems

Collaborative filtering systems are cushioned from the above problems by the fact that they are not based on the content of items but rather on the opinions of other users [Burke, 2002; Glauber *et al.*, 2013]. However, they have their own problems:

- Sparsity of the rating matrix. For most recommender systems, users rarely stay long enough to rate the system, and most of the cells in the rating matrix are empty. In such cases, finding similarities among different users or items is challenging [Sarwar *et al.*, 2001; Zhang and Huang, 2008].
- The infamous *cold-start* is also related to the previous problem, this one deals with the difficulty in making recommendations for new users to the system. In such cases, the users have not rated enough items yet, so the recommender system does not know them well enough to guess their interests [Schein *et al.*, 2002]. This problem also affects new items which have not yet been rated.
- Spam attacks [Cacheda *et al.*, 2011; Wang *et al.*, 2015]. Recommender systems could suffer spam attacks, mainly from unscrupulous users interested in misleading the system to recommend a certain product [Chirita *et al.*, 2005]

The list above is not in any way exhaustive and is not meant to be.

2.6 Related Work

Many companies operate in the service industry. These include banks, hotels, transport, logistics, insurance, agricultural and telecommunications companies. All of them provide core (primary) services to their clients, services that are essential and which every company operating in a particular service industry must provide. For example, the telecommunications industry provides core services such as voice telephone, telex, telegraph and facsimile. Each service industry is characterised by intense competition for customers among its different entities. In the case of telecommunications companies, there are other existential threats, apart from having to compete against other companies. The market has reached maturity, and this has led to a flattening trend in its growth. There is also the phenomenon of market saturation. Given these and other challenges, for a company that seeks to maintain its existing customer base and to attract new customers, it is not enough to merely continue offering primary services.

Service industry companies need to be innovative to survive in this challenging economic environment. One way of doing this is for companies to provide value-added services (VAS) "*to differentiate themselves, attract new customers and boost margins and ARPU (average revenue per user)*" [Bustos, 2012]. Value-added services are non-core services that are provided at low cost or freely to customers, the sole purpose being to promote and add value to the core (standard) services on offer. Some of the value-added services available in the telecommunications industry include mobile television, online gaming, missed call alerts, SMS chatting and so on. In this study, we focus on the promotion and provision of the different value-added services available in Cell C's GetMore package. The Private Label Promotions (PLP) Group has been engaged by Cell C to promote these services.

A company, having introduced value-added services, will be able to generate profits if these services are publicized and marketed to existing and potential customers. In this regard, recommender systems can play a very important role, by using them to recommend services that may be of particular interest to specific customers. Recently, recommender systems have been deployed in various e-Commerce environments, with the domain of recommender system applications being very extensive. It covers a wide spectrum of application areas, from multimedia rating systems to book recommendations on e-commerce sites such as Amazon, right up to social network sites such as Twitter, Facebook, and Linkedin, which can be used to recommend new contacts for active members. In this section, the work related to this research is discussed. In Subsection 2.6.1 some of the work focusing on value-added service provision is discussed. In Subsection 2.6.2 a discussion on recommendation and recommender system, which focusses specifically on hybrid recommender systems.

2.6.1 Related Work in Value-Added Service (VAS) Provision

According to Singh *et al.* [2011], "*A value-added service (VAS) is popular as a telecommunications industry term for non-core services, or in short, all services beyond standard voice calls and fax transmissions.*" Many companies are increasingly using technologies such as direct calling, email, short message service (SMS) and multimedia messaging service (MMS) as marketing tools. This development has led to an interesting challenge regarding the application of recommendation methods using these technologies [Pattamavorakun and Pattamavorakun, 2010]. Despite all the advances that have been made in the subject of recommender systems, there is still a need for further improvements to be made for recommendations to be more effective and applicable to an even wider spectrum of real-life applications, such as the domain in this study. In their work, Pattamavorakun and Pattamavorakun [2010] investigated a mobile application

whose intelligent personalisation algorithm implemented item-based collaborative filtering to help customers find the items (Thai cakes in this case) they wanted to buy.

M-commerce includes a wide range of value-added services. Carlsson and Walden [2002] investigated m-commerce products and services by focussing on three factors, namely users, producers and management. A survey was conducted whose results were used to support the authors' stance that m-commerce products and services should be understood based on these factors. The study was undertaken in three countries (Finland, Hong Kong and Singapore), and involved 50 companies whose businesses were driven by m-commerce. All the companies performed similar m-commerce activities. Concerning different issues on m-commerce, there were similarities and differences of opinion among experts in these countries. For example, the experts agreed that m-insurance services would not yield any profits "in the very near future" Carlsson and Walden [2002], and that mobile health care was not regarded as an important proposition for m-commerce. However, there were disagreements too. For example, there were different views regarding the prospects of m-education and m-learning. Some felt that m-education and m-learning had a great chance to succeed, whilst others held a different view. Also, there was no unanimity regarding the use of SMS, with views being divided regarding its potential for success. Worth noting is the fact that the research was conducted at a time when m-commerce was still in its infancy, and mobile telephones were not yet as powerful and widespread as they are today.

Francke and Weideman [2008] conducted their research on Instant Messaging (IM), focussing specifically on MXit (pronounced "mix it"), a cross-platform mobile social network developed in South Africa. MXit was one of the platforms used by mobile services providers to provide value-added services (VAS). The research looked at the implications of the application on the lives of South Africans and its security concerns. They concluded that, other than reported violent crimes which emanated from MXit, users had evidently become addicted to it, a sign that users want more out of their mobile telephones. Besides the fact that the introduction of IM was one of the first value-added services, the study introduced another important challenge of VAS: security.

The mushrooming of value-added services in the telecommunications industry is forcing service providers to think differently about their businesses and to find ways of differentiating themselves from and staying ahead of the competition [Lehmann *et al.*, 2008]. In their research, [Lehmann *et al.*, 2008] argue that value-added services will play an important part in computer networks of the future. For this to happen, these networks will have to include such components as application servers (AS) and media servers (MS), with application servers providing the required value-added services and the media servers being used for activities such as video conferencing. These new generation networks, the authors believe, will be capable of providing personalised customer services, and be able to support a wide range of new services.

Singh *et al.* [2011] carried out a study in the Indian telecommunications market to find out if Value-Added services (VAS) had any impact on the service provider consumers chose. Users were asked to rate their service providers of choice. They concluded that customers strongly consider VAS when choosing a particular cellular service over another. The regression analysis results clearly showed that the availability of services like internet banking, internet services had a positive effect on the consumer. These days most people travel a lot and they want to access all services while on the move like pay their bills, access the news, watch sports, and so on.

However, to receive improved benefits from the mobile VAS, users have to be willing to give more personal data in return. For personalised, targeted services more personal data handling is necessary prompting a reluctance in the user community to embrace the use of these services. This fear is dissipating as consumers become smarter and more informed about the way in which their data is used for marketing purposes and appreciate the value exchange in sharing their details; and the transparency of the opt-out options provided. Dolnicar and Jordaan in Dolnicar and Jordaan [2007] looked at responsible information management for aggressive direct marketing.In their study, only about 3% South Africans did not mind receiving calls marketing products and a similar figure thought their information was safe with the companies doing direct marketing.

[Wang *et al.*, 2015] proposed a new web-based recommender system-reputation measurement approach. They detected unscrupulous malicious feedback ratings by using the cumulative sum control chart, tried to neutralise the effects of subjective user feedback preferences by using the Pearson Correlation Coefficient. They evaluated the efficiency and effectiveness of their proposed approach by theoretical analysis and extensive experiments.

2.6.2 Related Work in Hybrid Recommender Systems

Spiegel and others in Spiegel *et al.* [2009], introduced Hydra, a hybrid recommender system combining collaborative algorithms and content-based filtering techniques in the domain of web-based recommendation systems. The data normalization and matrix factorisation models discussed in their article will be adopted in this text. Nabavi and Jafari in Nabavi and Jafari [2013] conducted a similar study to create a customer-retention system based on user preferences and customer data. The main objective of their study was perfecting the art of marketing and improve customer relationship management. They developed a model for predicting customer loyalty which partially satisfies the objective of this text. Moro and others also described a data mining approach to extract valuable knowledge from recent Portuguese bank telemarketing campaign data [Moro *et al.*, 2010], which is similar to what PLP's call centre does.

Symeonidis and others in Symeonidis *et al.* [2007] tried to bring a different dimension to the research in Recommender Systems when they used matrix factorisation and ranking algorithms to eliminate the trade-off between efficiency and effectiveness. As a point of departure they argued that most approaches partially analyse user or profile items when making recommendations. They also describe the various classifications of recommender systems and algorithms for computing similarity.

Breese and others in [Breese *et al.*, 1998] and Sarwar in [Good *et al.*, 1999; Sarwar *et al.*, 2001] discussed both user-based and item-based CF using the nearest neighbour algorithms. Deshpande in [Deshpande and Karypis, 2004] apply item-based CF algorithm combined with conditional-based probability similarity and cosine-similarity in developing their recommendation system.

Finally, the algorithms used to build this system borrow most of their architecture from the work of Vozalis G. and Margaritis in [Vozalis and Margaritis, 2007], [Vozalis and Margaritis, 2006],[Vozalis and Margaritis, 2004] and from Sarwar in [Sarwar *et al.*, 2002], [Sarwar *et al.*, 2001], [Sarwar *et al.*, 2000]. These authors highlighted the most important aspects of this research: understanding the user and the item by emphasising the construction of a demographic model at the onset of the modelling phase.

Research on recommendation systems and data mining is very extensive. Different models, their strength, and their shortcomings are discussed and illustrated. Research efforts in the field of Recommender Systems have been directed at improving accuracy in predicting user preferences. There is still need to focus more research on areas of diversity, serendipity and novel universally applicable algorithms. Matrix Factorisation is used mainly in the field of linear algebra and statistical matrix analysis has emerged as the best innovation yet in the study of recommender systems.

2.7 Selected Algorithms and Metrics for the Research Problem

This research tries to apply data mining to a new problem space where the content of the itemset is very heterogeneous. There is absolutely no prediction as to what the next customer will request next. The requests logged by customers range from classical service requests like booking a doctor to purchasing a car. Ratings used in the system are three-fold: service ratings, supplier ratings and price ratings.

The system built in this text calculates item and user similarity by measuring the similarity between items and between users. Similarity, in this paper, is defined as:

1. Content-based similarity

- Item content (attributes);
- User content (profile)

2. Collaborative-based similarity

- User-User (social/community) Collaboration
- Item-Item collaboration

A recommender system usually focuses on specific, homogeneous types of items like movies, news, books, web pages and accordingly with its architecture, its colourful user interface, and the main recommendation algorithm used to produce the recommendations are all customised to provide useful and relevant suggestions for that specific type of item. In this text, however, the items available for recommendation are differentiated and heterogeneous. They cover almost anything that people need in their daily lives.

Moreover, as much as the speed of recommendation generation is appreciable, the survival of the organisation depends on this system making accurate predictions. The primary objective was accuracy, with speed secondary. A hybrid recommender system was implemented using collaborative and content-based filtering algorithms. In this section, we describe the different algorithms that were selected and implemented as part of this research. Collaborative filtering algorithms are discussed in Subsection 2.7.1, followed by a discussion of content-based filtering algorithms in Subsection 2.7.2. In the context of the recommendation process different data mining techniques were used in conjunction with recommendation algorithms. Linear regression, classification, clustering and association rule mining techniques were used. In Subsection 2.7.3.2 a description is provided of these data mining techniques. Similarity measures are discussed in Subsection 2.7.5.

2.7.1 Collaborative Filtering Algorithms

Different collaborative filtering techniques were used in this research. Specifically, algorithms applicable to latent-factor models, as well as neighbourhood-based algorithms, were used. The Hybrid Recommender Algorithm followed in this study is shown in Figure 1 below:

For each user U, do the following:

Procedure 1 User CF Algorithm

Step 1: Build a neighborhood of size *N* consisting of the most similar users to the current user, where similarity is defined by the Adjusted Cosine Similarity.
Step 2: Generate a candidate list *L* of all distinct items in the system that at least one of the user in the current user's neighbourhood has expressed an interest.
For each item *i* in *L*
1. Generate the total_weight for that item as total_weight = (total_number_of_people_who_liked_it_in_neighborhood /neighborhood_size)
2. Sort *L* in descending order by total_weight.
3. Filter out all the items in *L* which the current user has already expressed an interest.
4. Pick the top *k* items and return them as recommendations

2.7.1.1 Matrix Factorisation and Singular Value Decomposition (SVD)

Recommender systems handle large volumes of data about users and items, data that is used to produce high-quality recommendations. Collaborative filtering is the most successful recommendation technique to date. Most collaborative filtering approaches use Nearest Neighbourhood Heuristic (NNH) algorithms in conjunction with the *Pearson Correlation Coefficient* metric. However, these algorithms fall short regarding accuracy. To enhance their performance, they must be optimised. Nearest neighbourhood approaches usually experience difficulties in obtaining exact matches. Their accuracy in producing good recommendations is poor [Sarwar *et al.*, 2000]. An alternative approach is to use the more superior matrix factorisation methods, which can discover latent features that underlie the interactions between two different kinds of entities, users U and items I [Koren *et al.*, 2009; Takacs *et al.*, 2008b]. Some of the most successful applications of latent factor models are based on matrix factorisation. By using Latent Factor model, hidden features about the relationship between users and items are identified. Singular Value Decomposition (SVD) transforms both items and users into the same latent feature space making it possible to compare them.

In the literature, matrix factorisation techniques are classified into two groups, based on the factorisation algorithms used [Gower, 2014; Takács *et al.*, 2008a; Takacs *et al.*, 2008b].

1. The first group of algorithms includes *QR factorisation* and *LU factorisation* methods, which decompose a matrix A into a product [Lee and Seung, 2000].

$$A = BC,$$

where B and C are matrices of a simpler form compared to matrix A.

2. The second group of algorithms factorises a Matrix $A \in \mathbb{C}^{n \times n}$ as follows:

$$A = S\Lambda S^{-1},$$

where Λ is typically a less complex matrix than the original matrix A.
If S is a complex orthogonal matrix then $S^{-1} = S^T$.

Examples of this factorisation technique include the Jordan Canonical Form and Matrix Diagonalisation. However, these factorisations are mostly used in logic proofs. This type of factorisation is exemplified by the Singular Value Decomposition (SVD) algorithm, which was implemented in this research to reduce the dimension of the user-item rating matrix that was used.

The Singular Value Decomposition (SVD), [Brand, 2003; Sarwar *et al.*, 2000; Spiegel *et al.*, 2009], is a well-known matrix factorisation technique, used for dimensionality reduction. It factorises an u - by - i matrix X into three matrices [Sarwar *et al.*, 2000]. These are an orthogonal matrix, a diagonal matrix and the transpose of an orthogonal matrix. The SVD is based on the fact that, given any matrix X, it is possible to find a set of real positive values σ_i and vectors u_i, v_i, such that, $Av_i = \sigma_i u_i$. The σ_i are known as *singular values* and every matrix has a SVD and the singular values are uniquely identified [Brand, 2003; Gower, 2014]. For example, given the matrix X, its decomposition is expressed as follows:

$$X = USV^T$$

where

- X is a $m \times n$ matrix of rank n, whose elements are either real numbers or complex numbers,
- U and V are orthogonal m x m and n x n matrices, and
- S is a diagonal matrix of size m x n with all singular values of X as its diagonal elements.

This low-dimensional representation of the user-item space reduces the computing complexity of neighbourhood algorithms [Sarwar *et al.*, 2001; Spiegel *et al.*, 2009]. This can be represented

as:

$$
\begin{array}{cccc}
X & & U & S & V^T \\
\begin{array}{c} \\ \uparrow \\ \text{users} \\ \downarrow \end{array}
\begin{pmatrix}
\leftarrow & items & \rightarrow & \\
x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\
x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
x_{m,1} & x_{m,2} & \cdots & x_{m,n}
\end{pmatrix}
&=
\begin{pmatrix}
u_{1,1} & \cdots & u_{1,n} \\
\vdots & \ddots & \vdots \\
u_{m,1} & \cdots & u_{m,n}
\end{pmatrix}
\times
\begin{pmatrix}
s_{1,1} & 0 & \cdots \\
0 & s_{2,2} & \\
\vdots & & s_{r,r}
\end{pmatrix}
\times
\begin{pmatrix}
v_{1,1} & \cdots & v_{1,n} \\
\vdots & \ddots & \\
v_{r,1} & & v_{r,n}
\end{pmatrix} \\
m \times n & & m \times m & m \times n & n \times n
\end{array}
$$

SVD is built on a theorem which originated from the linear programming field which contends that a rectangular matrix X can be expressed as a product of three matrices: an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V [Baker, 2005].

The rectangular matrix X above represents the combined *user-item* rating matrix. In this study, each user u will be composed of a vector of user attributes as will be shown in later sections. The orthogonal matrix U and its transpose V represent the user and item matrices respectively. The 'decomposition' part of SVD splits user and item matrices from the huge original matrix. The middle matrix, S,is a diagonal matrix whose entries $s_{i,j}$ are 0 when $i \neq j$. S represents the Singular Values which explains why the technique is referred to as Singular Value Decomposition. The only non-zero values run diagonally from the upper left corner to the bottom right corner [Baker, 2005; Gower, 2014; Sarwar *et al.*, 2000].

The diagonal matrix S contains r singular values of matrix X, where r is the rank of matrix X [Deerwester *et al.*, 1990]. Two vectors are linearly independent if they cannot be written as the sum or scalar multiple of any other vectors in the space [Brand, 2003]. Linear independence captures the notion of a feature or agglomerative item or user that we are trying to obtain information about. For example, if every user who liked an item (i_1) also liked another item (i_2), then the two items' vectors are linearly dependent and only contribute one to the rank. This means the two linearly dependent vectors are treated as one.

If the singular values in S are ordered by size in descending order, the first k largest values may be kept and the remaining, smaller values, set to zero [Deerwester *et al.*, 1990]. Normally,

k , that is, singular values included, should be large enough to capture all the relevant latent feature relationships and small enough to avoid over-fitting [Spiegel *et al.*, 2009]. The task is to compare items to see if most users who like the first item also like the second one. This is accomplished by keeping the first k singular values in S, where $k < r$. This produces the best *rank-k* approximation to X, thereby reducing the dimensionality of the original space. The product of the resulting matrices is a matrix $\hat{X}$, which is approximately equal to X, and has rank k:

$$\underset{m \times n}{\hat{X}} \approx \underset{m \times r}{U} \times \underset{r \times r}{S} \times \underset{r \times n}{V^T}$$

$$\begin{array}{c} \\ \uparrow \\ \text{items} \\ \downarrow \end{array} \begin{array}{c} \leftarrow \quad \textit{Attributes} \quad \rightarrow \\ \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix} \end{array} \approx \begin{pmatrix} u_{1,1} & \cdots & u_{1,n} \\ \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,n} \end{pmatrix} \times \begin{pmatrix} s_{1,1} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{r,r} \end{pmatrix} \times \begin{pmatrix} v_{1,1} & \cdots & v_{1,n} \\ \vdots & \ddots & \\ v_{r,1} & & v_{r,n} \end{pmatrix}$$

This resultant estimation $\hat{X}$ is able to capture all the relevant latent relationships between users and items, which allows a collaborative recommendation system to predict user-item ratings with ease. Moreover, this low-dimensional representation of the user-item space ($\hat{X}$) reduces the computational overload of the similarity algorithms. It is quite possible to reduce the number of dimensions in a user-item space to a designated number k, in which matrix S only consists the top k largest singular values.

The main problem with using SVD is its computation cost [Brand, 2003]. Even though computation can be done offline Spiegel *et al.* [2009], the process can still be computationally intractable for colossal databases. To address this problem, many researchers have examined incremental techniques to update an existing SVD without recomputing it [Sarwar *et al.*, 2002]. Using SVD also makes sense if there is a relatively small set of item features and users that are a true representation of the reaction of the majority of users to most of the items. Given the diverse nature of the items in our dataset, the SVD matrix factorisation techniques were used in this research for dimensionality reduction [Vozalis and Margaritis, 2006].

Lee and Chang [2013] carried out a study on enhancing the performance of CF algorithms by using SVD. In their paper, they gave a concise algorithm for SVD:

Procedure 2 Singular Value Decomposition

Input: An $m \times n$ matrix A, and a number k
Output: Approximate U_k, V_k and S_k
Algorithm:
1. Generate an $n \times k$ Guassian matrix G
2. Compute $Y = AG$
3. Compute an orthogonal column basis Q of Y
4. Form $B = Q^T A$
5. Compute eigen-decomposition of $BB^T = XS^2X^T$

$U_k = QX, V_k = B^T XS^{-1} and S_k = S$

[Lee and Chang, 2013]

However, in this study, we are not interested much in the backend workings of SVD, but in its proven properties as already given above.

2.7.1.2 Nearest Neighbourhood Method (NNH)

The most used approach in CF is based on neighbourhood models [Sarwar *et al.*, 2001; Schafer *et al.*, 2007]. Originally, CF was mainly user-oriented [Koren, 2008]. User-oriented methods estimate an active user's unknown ratings based on the ratings of like-minded users.The prominent nearest neighbourhood algorithm used in literature is the *k-NN*. Sarwar *et al.* [2001] investigated the various techniques for computing item-item similarities which included item-item correlation and cosine similarities between item feature vectors and different algorithms for obtaining recommendations from them. These algorithms included the linear regression model. In the end, we evaluated their results by comparing them to the basic k-nearest neighbour approach. In their results item-based CF algorithms performed far much better than their user-based counterparts.

An excellent analysis of NNH is provided by [Herlocker *et al.*, 1999], in which NNH methods are divided into three broad steps:

- Weight all users with respect to their similarity to the active user.
- Select a pool of users to use as a neighbourhood set of predictors
- Normalise the ratings and calculate a prediction from a combination of selected neighbour's ratings. [Bell and Koren, 2007; Herlocker *et al.*, 1999; Koren *et al.*, 2009]

The item-based CF approach developed later based on the user-oriented one. Better scalability and improved accuracy make the item-oriented approach more favourable in many cases [Koren, 2008]. We compare and contrast the item-based and the user-based models below.

- **Item-Based Nearest Neighbour Algorithms**. Our recommender system is built around the item model based on the work of Sarwar *et al.* [2001] which proved it to be the most accurate and efficient. Item-based algorithms generate predictions based on similarities between items [Sarwar *et al.*, 2001; Schafer *et al.*, 2007]. The prediction for an item is based on the active user's ratings for similar items. The algorithm for the item-based model can be formalised in Equation (2.1) below:

$$pred(u,i) = \frac{\Sigma_{j \in ratedItems(u)} itemSim(i,j).r_{u,i}}{\Sigma_{j \in ratedItems(u)} itemSim(i,j)} \tag{2.1}$$

[Schafer *et al.*, 2007]

A predicted score or rating for active user u on item i is composed of a weighted sum of the active user's u ratings for items most similar to i. Item similarity *itemSim()*, will be calculated using the adjusted Cosine Similarity metric which is explained in Section 2.7.5.

While item-based models can be huge in size, Sarwar *et al.* [2001] manage the model by retaining only a manageable top n correlations for each item. Such modifications to the model yield optimised algorithms that are relatively efficient in both memory usage and CPU performance.

- **User-Based Nearest Neighbour Algorithms**.This group of algorithms generates predictions for the active user based on ratings of similar users (neighbours).If the active user a is most similar to another user u, consider that u to be the nearest neighbour of a. User-based algorithms generate a prediction for an item i by analysing ratings for item i from users in the active user's a neighbourhood. Consequently, we could easily average all neighbours' ratings for item i, but that would be inadequate because we need to rank neighbours according to their similarity score to identify the nearest neighbours. [Sarwar *et al.*, 2001; Schafer *et al.*, 2007]. Furthermore, calculating user-similarity operates well in an environment where the rating scale is the same, and the rating matrix is perfect, but that is rarely the case. Therefore, the algorithm needs to normalise the rating data that is fed into it to work. We normalise the prediction by dividing it by the sum of the neighbours' similarities and by subtracting the average rating from each rating [Schafer *et al.*, 2007; Spiegel *et al.*, 2009].

  The above algorithm can be formalised in Equation (2.2) below:

$$pred(u,i) = \bar{r}_u + \frac{\Sigma_{n \subset neighbours(u)} userSim(u,n).(r_{ni} - \bar{r}_n)}{\Sigma_{n \subset neighbours(u)} userSim(u,n)} \tag{2.2}$$

userSim() in equation 2.2 will be calculated using the Pearson Correlation Coefficient which will be formally stated in Section 2.7.5. User u is the active user while n represents the neighbour in the algorithm (2.2).

Although in theory, user-based CF systems seem like exactly how recommendation should be made, they have serious shortcomings; most of which relate to the issue of rating data sparsity. Lack of weighting in the Pearson Correlation Coefficient fails to accommodate inconsistencies posed by item popularity. Most popular items tend to be rated by many people while less popular ones are lacking in terms of quantity of ratings given by the previous users [Breese *et al.*, 1998]. Most importantly, computing a user's perfect neighborhood is expensive; especially since they require comparison against all other users. Some neighbourhood algorithms try to eliminate this burden on computing resources by limiting the number of neighbours that can be retained and by employing clustering.

2.7.2 Content-Based Filtering Algorithm

As already indicated in previous sections, CBF systems analyse item descriptions and other content attributes to identify items that may interest the active user. In literature, CBF algorithms have been used extensively, but mainly in conjunction with CF ones [Balabanović, 1997; Melville *et al.*, 2002; Sarwar *et al.*, 2000]. In this study, CBF algorithms will be combined with CF algorithms and SVD for dimensional reduction. The CBF component of our approach constructs a feature profile of a user and that of an item, based on both collaborative and content features.

The CBF recommendation methods used in this study is the *Utility* approach described in Adomavicius and Tuzhilin [2005]; Symeonidis [2008]. The recommendation problem can be formulated as follows. Let C be the matrix of all users while S represents all the over a thousand items and u is a utility function that measures the ability of an item s to satisfy the need of a user c. In a simplified form, $u : CxS \longrightarrow X$, where X is an ordered set of non-negative numbers within a finite range. Then for each user $c \in C$, we need to choose an item $s' \in S$ that maximises the user's utility. This algorithm can be formally embodied in the following equation 2.3:

$$\forall c \in C, s'_c = arg_{s \in S} \quad max \quad u(c,s) \tag{2.3}$$

In this study, a user indicated the item's utility to him by way of ratings and number of requests for the item he has made over a given period. And every user, $c \in C$ is a product of his feature attributes as will be explained in Section 3.6.2 with the *userID* as the unique user identifier. The same applies to each item, $s \in S$, where the unique identifier is the *itemID*. Item data is discussed in table 3.6.

The problem with this approach is that it relies on ratings to calculate utility. Ratings are sparse and are based on historical data which means for the most part utility has to be extrapolated and guessed. In this section, the utility *u(c, s)* of item s for user c was extrapolated based on the rating (u, c_i) assigned by user c to items $s_i \in S$ that are *similar* to item s. *Similarity* in this case was calculated using the Adjusted Cosine Similarity (Section 2.7.5). Several potential items are compared with items previously rated by the active user, and the *top-n* most matching item are recommended. An item content profile CP of user c is a vector of items that the user has previously rated. Given that, the utility function u can be stated as:

$u(c, s) = score(CP(c), Content(s))$, where *CP(c)* is the demographic profile of user c, and *Content(s)* are the content attributes of the active item. The *score* is the Adjusted Cosine Similarity algorithm [Basu *et al.*, 1998; Pennock *et al.*, 2000; Sarwar *et al.*, 2001]. Formally the *utility* algorithm leads us to:

$$u(c, s) = cos(\overrightarrow{w}_c, \overrightarrow{w}_s) = \frac{\overrightarrow{w}_c . \overrightarrow{w}_s}{|\overrightarrow{w}_c|_2 \times |\overrightarrow{w}_s|_2}$$

Where w_c and w_s are the item and user content-feature vectors respectively.

In this recommendation procedure, in order to recommend items to user c, the recommender system uses profile features to try and understand the similarities among the items user c has previously rated. Then, only the most similar items to the user's choice would be recommended.

2.7.2.1 Item-Based CBF

In this text, we had only four item attributes: the *item_id, item_name, item_category, and the item_type*. Therefore, item k (I_k) can be expressed as:

$F(I_k) = \{a_1, a_2, a_3\}$ Where, $F(I_k)$ is the k^th item of the feature matrix I. The item_name and item_id fields are identifiers so only one of them was used in the algorithm.The metadata for the item attributes is given in Section 3.6.2.

The item matrix is pictured in Table 2.2 below:

	item_id	item_category	item_type
I_1	1	20	1,2,4,5,...
I_2	2	21	1,2,4,10,...
I_3	3	20	1,2,4,5,...
I_4	4	20	1,2,4,5,...
I_5	5	23	8,9,2,11,...
...	...	...	...

TABLE 2.2: Item-Feature Matrix (I)

The values assigned to each attribute did not belong to the same scale; neither were they just Boolean. The *item_type* attribute was an array. Some of the items qualified in various sub-categories. For example, a bicycle is both a means of transport and sports item. Subtractive Normalisation, using The Gaussian Method discussed in 3.6.3 was applied to this matrix to convert it to values of 0 and 1 before any modelling could be done.

Table 2.2 above displays the item features transformed to binary forms for use in SVD. Figure 2.3 graphically illustrates the row form of these attributes:

```
mysql> select * from items limit 10;
+----+--------+------------------------------+-----------------------------------------------------------------------------------------+
| id | itemID | name                         | type                                                                                    |
+----+--------+------------------------------+-----------------------------------------------------------------------------------------+
|  1 |   3253 | Call dropped                 | Non-service, missed call                                                                |
|  2 |   3304 | Member called in for info    | Information, enquiry                                                                    |
|  3 |   3805 | Telewheels - Cell C          | Telewheels, car hire, taxi, bus service                                                 |
|  4 |   4823 | On boarding TeleShop         | Onboarding, online shopping, suppliers                                                  |
|  5 |   4827 | On boarding Finance Assist   | Onboarding, financial assistance, business assistance, banking                          |
|  6 |   4829 | On boarding TeleWheels       | Onboarding, information, taxi, car hire                                                 |
|  7 |   4225 | Cancellation                 | Cancellation, membership cancellation                                                   |
|  8 |   3796 | Tax Expert                   | Financial Services, financial assistance, business assistance, tax, accounting, auditing |
|  9 |   3303 | Internal Transfer            | Marketing, Business Services                                                            |
| 10 |   3809 | Get It Membership Cancellation | Cancellation, membership cancellation                                                 |
+----+--------+------------------------------+-----------------------------------------------------------------------------------------+
10 rows in set (0,01 sec)
```

FIGURE 2.3: Items in the database

The figure above shows that every item belongs to an array of an item_types. These item types were instrumental in determining item-item similarity weights.

Due to the scarcity of meaningful attributes in the *item* data space, the *item_type* attribute showed dominance.

2.7.2.2 User-Based CBF

The user profile matrix U was created in the form described in Section 3.6.2. The matrix is constructed as table 2.3.

	f_1	f_2	f_3	...
U_1	1	0	0	...
U_2	0	1	0	...
U_3	0	0	0	...
U_4	1	1	1	...
U_5	1	0	1	...
...	...	...	...	...

TABLE 2.3: User Profile Matrix

Similarly, the profile attributes above were in no way, boolean. They had to be normalised before they could be used. Features were entered at their face value, for example, age could be anything between 1 and 100 while service_join_month ranges between 1 and 12. It was these variations in the rating value scales which necessitated normalisation in the data preparation stage.

2.7.2.3 Combining CBF: The Rating Matrix

The rating matrix R brings together the user matrix U and the item matrix I. We used this matrix to capture the interaction between the user and the item. It included all items that had been used for at least 10 times. Therefore, the combined feature profile of user U_k, is just one row in the rating feature matrix R, whose elements, *P(u,f)*, are formally expressed by the equation below:

$$P(u, f) = \sum_{\forall R(u,i)} F(i, f)$$

The where the profile feature of user u at point *(u,f)* in the matrix, is a summation of all the implicit and explicit ratings of the user given in the feature matrix *F(i,f)*

This combined matrix, which is detailed in Section 3.8 and in Figure 3.11, is summarised below in table 2.4:

	I_1	I_2	I_3	...
U_1	0	0	1	...
U_2	1	1	0	...
U_3	1	1	0	...
U_4	1	0	1	...
...	...	...	...	...

TABLE 2.4: Combined Feature Matrix

SVD was be applied to this combined matrix to deduce latent features and sub-matrices as specified in Section 3.6.2 in Chapter 3. The similarity scores between user-vectors were calculated using the Adjusted Cosine Similarity metric. Finally, we selected the *Top-N* recommendations to present to the client [Symeonidis, 2008; Vozalis and Margaritis, 2004].

2.7.3 Selected Data Mining Techniques

There are a number of studies where data mining techniques have been applied to recommender systems. The study by Haruechaiyasak *et al.* [2004] used association rule mining to produce a content-based filter that was incorporated into a system that was used to provide personalised recommendations of Web pages at an American university. In another study, Adeniyi *et al.* [2016], the K-Nearest Neighbour (KNN) classifier was used online in a system whose purpose was to identify a Web user's behaviour in real-time, based on click stream data, and to match this information to a group of other users (nearest neighbours), whose browsing preferences could then be recommended as links the user could follow. The association rule mining technique was used as part of a collaborative recommender system [Lin *et al.*, 2002]. The authors developed a new association rule algorithm which was used offline to generate rules that were used for collaborative filtering Duan *et al.* [2011]; Gupta and Garg [2014]; Saraee *et al.* [2005]. In this research different data mining techniques were used in the prototype hybrid recommender system that was implemented as part of this research. These included classification, prediction, clustering and association rule mining techniques.

2.7.3.1 Classification

Classification as a data mining technique has already been discussed in Section 2.2.1 above. According to Romero *et al.* [2008], classification, which involves predicting the missing attribute based on the values of other attributes can be classified into five different classification approaches:

- *Statistical Classification*: This is described by Otero and Sánchez [2006] as a procedure in which items are clustered into groups based on the quantitative information of attributes composing the items. Examples of this set of algorithms include k nearest neighbours and linear discriminant analysis [Curram and Mingers, 1994; Minaei-Bidgoli and Punch, 2003].

- *decision tree classification*: According to Quinlan [2014] a decision tree consists of a set of filtering conditions organised hierarchically. The most common decision tree algorithm is C4.5 (J4.8) Quinlan [2014].

- *Rule Induction Classification*: These are IF-THEN production rules that are derived from observations like a sequence of item requests or purchases [Calvo-Flores *et al.*, 2006]. The set of algorithms used by rule induction follow a heuristic search [Romero *et al.*, 2008]. An example of an algorithm in this category is the AprioriC described inJovanoski and Lavrač [2001].

- *Fuzzy Rule Induction Classification*: This category of algorithms apply fuzzy logic in order to interpret the underlying data linguistically [Romero *et al.*, 2008; Woolf and Wang, 2000].An example of a fuzzy rule learning method is LogitBoost which is described in Otero and Sánchez [2006].

- *Neural Networks Classification*: Neural Networks use the same principle as is used in rule induction. They are modelled to mimic the cortical structures in the brain [Curram and Mingers, 1994; Otero and Sánchez, 2006].

A statistical classification algorithm was chosen for this study. The results obtained from testing the combined dataset produced the results displayed on Table 2.5 below:

Algorithm	RMSE	MAE	TP Rate	FP Rate	F-Measure	ROC	Accuracy
C4.5 (J48)	0.45	0.23	0.67	0.25	0.67	0.71	67.13
Filtered Classifier	0.40	0.26	0.69	0.26	0.67	0.75	68.67
Naive Bayes Classifier	0.59	0.35	0.47	0.37	0.38	0.62	47.02
Regression Classifier	0.35	0.24	0.74	0.22	0.72	0.85	74.20
Clustering Classifier	0.59	0.35	0.48	0.46	0.47	0.51	48.06

TABLE 2.5: Collaborative Filtering Usage Accuracy Matrix

Clearly, the Regression classifier had the lowest RMSE and was the best performer in all criteria. Linear Regression (LR) was used to predict a class attribute which could be user status, user-item rating or user service usage. Formally stated, the LR algorithm is stated below:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \varepsilon_i (i = 1, 2, ..., n) \tag{2.4}$$

Regression trees proved very accurate when used with continuous data like ratings and usage.

2.7.3.2 Prediction

To obtain missing rating predictions and make recommendations is the major step in a collaborative filtering system. In the neighbourhood-based CF algorithm used in this system, a pool of nearest neighbours of the current user are selected based on their Pearson Correlation Coefficient similarity scores, and their ratings are used to make predictions for the current user [Koren *et al.*, 2009; Su and Khoshgoftaar, 2009; Zhang and Huang, 2008]. Therefore, to make a prediction for the active user, a, on a certain item, i, we took a weighted average of all the ratings on that item using the formula shown in equation 2.5 below.

$$pred(a,i) = \bar{r}_a + \frac{\sum u \in U(r_{u,i} - \bar{r}_u).w_{a,u}}{\sum u \in U\,|w_{a,u}|} \tag{2.5}$$

All users who are similar to the active user are placed in set U, where $U = \{u_1, u_2, ..., u_k\}$. $\bar{r}_a$ and $\bar{r}_u$ are the average ratings of the active user a and the recommender user u on all other items. $w_{a,u}$ is the weight (similarity) between the users u and a. The ratings of all the users $u \in U$ who have rated the item i are then summed up [Bell and Koren, 2007; Su and Khoshgoftaar, 2009]. The similarity, $w_{u,a}$ between user a and u is calculated using the Pearson Correlation Coefficient discussed in section 2.7.5.

2.7.3.3 Clustering

A cluster is a set of items or users that are similar to each based on some attributes which make them dissimilar to the items or users in other clusters [Han, 2011; Su and Khoshgoftaar, 2009]. According to Han [2011, 361]"Clustering is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters. Dissimilarities and similarities are assessed based on the attribute values". Item- or user-similarity is commonly defined in terms of the closeness of the objects in space, based on a distance function. The most common distance function is the Minkowski Distance [Su and Khoshgoftaar, 2009]. According to Han [2011], the "quality" of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster.

The Minkowski Distance function can be formally stated as:

$$d(X,Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q}$$

where, where n is the dimension number of the object and x_i, y_i are the values of the i^{th} dimension of objects X and Y respectively [Han, 2011; Su and Khoshgoftaar, 2009].

There are different approaches to clustering [Han, 2011].

- Hierarchical methods: Clustering is a hierarchical decomposition technique with multiple levels. However, they are rigid. They cannot correct erroneous splits and merges. They can incorporate other clustering types.
- Density-based methods: This technique can be used to filter outliers.
- Grid-based methods: They use a multi-resolution grid data structure and are quite fast.
- Partitioning methods: These are distance-based algorithms which attempt to find mutually exclusive clusters of spherical shapes. These are effective on small to medium datasets. The *k means* is an example of a partitioning method. The *Simple kMeans* algorithm explained in figure 3.16 is a lighter implementation of the *k Means* clustering algorithm.

In this study, k-means clustering is used as an intermediate step in attribute selection and computing user similarity and the resultant clusters are used again in classification or Linear Regression [Su and Khoshgoftaar, 2009]. The k-means works well for regular clusters and is relatively efficient and scalable [Han, 2011; Su and Khoshgoftaar, 2009]

For a dataset D with n records (objects) in the Euclidean space, the K-means algorithm distributes the records in D into k clusters. *C1,..., Ck*, that is, $C_i \subset D$ and $C_i \cap C_j = \emptyset$ for $(1 \leq i, j \leq k)$.

2.7.3.4 Association Rule Mining

Association rule mining 2.2.2 was done in this study as a two-step process in computing user-user and item-item similarity. In the first step, we find all frequent itemsets. A frequent itemset occurs at least as frequently as a predetermined minimum support count, *min_sup*. Then we generate strong association rules from the frequent itemsets. A strong rule by definition satisfies the minimum support and minimum confidence. *Support* and *Confidence* can be easily illustrated by equation 2.6 below:

$$\begin{aligned} support(A \Longrightarrow B) &= P(A \cup B) \\ confidence(A \Longrightarrow B) &= P(B \mid A) \end{aligned} \tag{2.6}$$

Equation 2.6 leads us to equation 2.7 below.

$$confidence(A \Longrightarrow B) = P(B \mid A) = \frac{support(A \cup B)}{support(A)} = \frac{support_count(A \cup B)}{support_count(A)} \tag{2.7}$$

According to equation 2.7 the confidence of rule $A \Longrightarrow B$ can be easily derived from the support counts of A and $A \cup B$. That is, once the support counts of A, B, and $A \cup B$ are found, it is straightforward to derive the corresponding association rules $A \Longrightarrow B$ and $B \Longrightarrow A$ and check whether they are strong. Thus, the problem of mining association rules can be reduced to that of mining frequent itemsets [Han, 2011].

2.7.3.5 Linear Regression Methods

Linear regression (LR), [Koren *et al.*, 2009; Zhang and Huang, 2008], is the most characteristic of the regression models and is usually applied in various areas. LR is usually used to predict the value a given variable when other related variables change in a certain way. Regression analysis models the relationship between a set of independent variables and a dependent variable. The following algorithm, 2.8, usually calculates this correlation:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \varepsilon, (i = 1, ..., n) \tag{2.8}$$

This method is commonly known as *least square estimation or least square problem* [Koren *et al.*, 2009; Zhang and Huang, 2008]. Regression models can be used to either learn normalized ratings or estimate missing ratings[Bell and Koren, 2007]

2.7.4 Recommender System Item Ratings

To recommend certain items recommender systems rely on the ratings assigned by users to these items. The ratings are assigned *explicitly* or *implicitly*. Explicit ratings represent the opinions users have about items and are assigned by users to these items. A recommender system derives implicit ratings, based on its monitoring of the interaction between users and items. Examples of explicit ratings include a user assigning a numerical rating to an item Melville and

Sindhwani [2010] by, for example, clicking on *like/dislike* buttons, after which the response can be converted to a binary "relevant/not relevant" response Ricci *et al.* [2011]. Other types of explicit rating methods include making comments and asking a user to provide demographic information. Implicit rating occurs, for example, when a system monitors the number of visits to a web page. In short, user feedback information can be obtained explicitly by asking the user or implicitly by monitoring user activities. In-depth research has been conducted in which explicit and implicit rating techniques are discussed within the context of recommender systems [Dooms *et al.*, 2011; Jawaheer *et al.*, 2014; Parra *et al.*, 2011].

Explicit Ratings

In explicit rating, a numeric scale is used, whose values may range between 1 and 15. A widening of the rating scale and higher variance in collected ratings should be normalised before the data can be used [Sarwar *et al.*, 2002; Setten, 2005]. Social network sites like Facebook and video-sharing websites such as Youtube use *'like/dislike'* buttons that can be used to gather user opinion. These websites also use demographic data to suggest content which may be relevant to a particular user. Search tags and labels are also included in this category of explicit ratings. One of the advantages of explicit feedback is its simplicity. However, the use of numeric increases the mental load on the user, and this may not be limited regarding capturing the user's real feelings about items [Dooms *et al.*, 2011; Melville and Sindhwani, 2010].

Implicit Ratings

Implicit ratings are collected passively by studying users' behaviour as they click on links, spend time on a web page, their interactions and login location [Fayyad and Smyth, 1996; Ricci *et al.*, 2011]. On a user's computer, a small text file (*cookie*) may be created by a website to collect information about the user's behaviour, activities and preferences during a session. This information is converted to numeric data for analysis and normalisation. Implicit feedback methods assign a relevance score to specific user actions in relation to an item. Such activities may include saving, discarding, printing, bookmarking, quantity purchased, the frequency of usage, and so on [Melville and Sindhwani, 2010]. The main advantage in adopting the implicit rating methods is the fact that they do not require direct user involvement, even though cases of bias are likely to occur, such as being interrupted by phone calls while reading, resulting in more time being spent on a page. Such user interruptions may be misinterpreted to mean that the user spent more time on the page.

Hybrid Ratings

To capture the accurate feelings of a user, a hybrid rating method combines explicit and default ratings. A user may be asked a simple "yes/no" survey question, accompanied by a text box to provide the reason for responding with a "no." This approach also enables users to provide gray-area ratings such as "Yes, but ...".

2.7.5 Similarity Metrics

In this research collaborative and content-based filtering techniques were combined to create the prototype hybrid recommender system that was implemented. Memory-based collaborative filtering algorithms are divided into user-based and item-based algorithms. Similarity techniques are used in both types of algorithms to determine the similarity between users (user-based CF) and between items (item-based CF), respectively. Figure 2.4 represents a *user-item* utility matrix, of the type that is often used to depict the relationship between users and items in user-based collaborative filtering environments.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	6	4	1	3		5	9
u_2			8	5	8	9	6
u_3	5			4	7	2	7
u_4	1	2		6	5	8	4

FIGURE 2.4: Example Utility Matrix

In the matrix, four users have rated seven items. Worth noting is the difference between users u_1 and u_2 in their rating of the item i_3. In other words, we can say that there is a large *distance* between u_1 and u_2 in terms of how similar they are in rating item i_3. This notion of distance is applied prominently in nearest neighbour based algorithms to determine the similarity between users or between items. The blank spaces in Figure 2.4 indicate that users have not yet rated particular items. These are the items which the recommender system should determine ratings for. The major goal of a recommender system is the generation of *"meaningful recommendations to a collection of users for items or products that might interest them [Melville and Sindhwani, 2010]"*. Filling in the missing ratings in a matrix such as the one shown in Figure 2.4, makes it more convenient and easier for a recommender to help users by recommending items that may be of interest to them. Most of the matrices that are used to store real-life data contain a large

number of blank spaces, much more than in Figure 2.4. These matrices are referred to as *sparse matrices* [Rajaraman and Ullman, 2012].

There are a number techniques that are used to determine the similarity between items or between users. Some of the most common techniques the Jaccard Coefficient [Rajaraman and Ullman, 2012], Cosine similarity [Ricci *et al.*, 2011], Pearson Correlation Coefficient [Antunes, 2008], and hashing algorithms [Rajaraman and Ullman, 2012]. In the following subsections some of the similarity techniques that were implemented in this research as part of the collaborative filtering task are discussed.

The Jaccard Similarity

The Jaccard Index is also known as the *Jaccard similarity coefficient* or the *Jaccard Distance* [Rajaraman and Ullman, 2012]. It uses set notation to compute recommendations. A Venn Diagram intersection is used as a measure of similarity. The following algorithm is used to measure the Jaccard Similarity:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.9}$$

FIGURE 2.5: Jaccard Similarity

where $J(A, B)$ represents the similarity between user A and user B and is calculated by dividing the items they have in common by the total number of items they have rated. However, this algorithm ignores the ratings themselves.

Cosine Distance Similarity

The Cosine Distance Similarity measure is one of the most common nearest neighbour algorithms. Items are represented as vectors in the user space. A similarity score is computed by calculating the cosine of the angle between the two vectors. For this reason, Pennock and Horvitz in Pennock *et al.* [2000] refer to it as the *Vector Similarity*. The values range between 1 (perfect) and -1 (opposite) [Antunes, 2008; Rajaraman and Ullman, 2012]. Using the adjusted Cosine Similarity Measure, which takes into account differences in the rating scales of different

users, the similarity between two items i and j, is defined as:

$$sim(\vec{i}, \vec{j}) = \frac{\vec{i} * \vec{j}}{\left|\vec{i}\right| * \left|\vec{j}\right|}$$

It is represented by Equation 2.10 [Jannach *et al.*, 2011; Li and Kim, 2003; Sarwar *et al.*, 2000]:

$$sim(i, j) = \frac{\Sigma_{u \in U}(R_{u,i} - R_u) * (R_{u,j} - R_u)}{\sqrt{\Sigma_{u \in U}(R_{u,i} - R_u)^2}\sqrt{\Sigma_{u \in U}(R_{u,j} - R_u)^2}} \tag{2.10}$$

where

- U is the set of items rated by both users;
- $R_{u,i}$ is the rating assigned to item i by user u;

$(R_{u,i} - R_u)$ and $(R_{u,j} - R_u)$ normalise a rating by subtracting the respective user averages from the rating, which contributes towards overcoming discrepancies in a user's rating scale.

Cosine Adjusted similarity measure will be used in this research to rank items.

Pearson Correlation Coefficient

One of the basic most common similarity measures uses the Pearson correlation Coefficient [Cios *et al.*, 2007]. The equation for this algorithm is as follows [Adomavicius and Tuzhilin, 2005]:

$$sim(x, y) = \frac{\Sigma_{s \in S_{x,y}}(r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\Sigma_{s \in S_{x,y}}(r_{x,s} - \bar{r}_x)^2(r_{y,s} - \bar{r}_y)^2} \tag{2.11}$$

FIGURE 2.6: The Pearson Correlation Coefficient

where xy represents two users, x and y, who have co-rated the items. This approach is best suited where only users and ratings need to be correlated without first to deduce similarity between users themselves using demographic data.

2.8 Chapter Summary

To conclude, this chapter has briefly presented a large amount of literature coverage for the topics and concepts related to this study. Most of the literature available relates to data mining and its application; recommendation systems; the challenges facing them and the different recommendation techniques. An introduction to the various algorithms relevant to this study was also made

Nowadays, organisations cope with much more challenges than ever before. Challenges of fast changes in a global, complex and competitive business environment, the change and diversification of users' requirements, the growing complexity of systems and technologies, extended regulation and compliance requirements. Organisations generate, use, store and share information with their customers, business partners and providers, At the same time, they have to report periodically certain information to their shareholders and various regulation and control institutes.

However, there are still many challenges with recommender systems just as there different types of recommender systems and recommendation algorithms not applicable in all domains.

Chapter 3

Research Method

3.1 Introduction

In Chapter 2 a detailed review of the literature related to this research was conducted. The chapter also included a discussion of different topics such as data mining and data mining applications, the concept of recommendation and recommendation techniques, and their application in recommender systems. In this chapter, we discuss in detail the methodology that was used to guide this research to completion. More specifically, the discussion is focussed on the adoption and application of the CRISP-DM methodology to our project, in which one of the major aspects was the PLP Group's customer data that had to be processed and analysed. CRISP-DM, given its predominant acceptance in the data mining and data science fraternity as the methodology of choice for data mining projects Piatetsky, G [2014], was adopted for this study. A discussion on the motivation for conducting this research is provided in Section 3.2. The CRISP-DM Process Model and the different phases that constitute it are discussed in Section 3.3. A discussion is provided in Section 3.4, which focusses on the research setting and the research participants. Section 3.5 provides a detailed analysis and discussion of different aspects related to the prototype hybrid recommender system that was implemented as part of this research. Following the introduction and detailed description of the CRISP-DM methodology in Section 3.3, its application to our research context is discussed in Section 3.6. The chapter is summarized and concluded in Section 3.8.

3.2 Motivation for the Research Problem

Individuals and organisations have access to significant amounts of data, which is obtained from different sources. In some cases, the data is highly accessible and available for sharing by

interested parties. The primary challenge facing individuals, organisations and institutions, is how to exploit to its full potential the massive amounts of data at their disposal, by processing it to discern meaningful information patterns, predict trends, and to obtain knowledge about variables of interest [Giudici, 2003]. In many cases, data is not efficiently stored or utilised. It is not effectively processed to provide meaningful and useful information to the people and organisations that need it in the right format and at the most appropriate time for them to use it to make well-informed decisions.

Data mining involves the analysis of large volumes of datasets, to discover hidden relationships between dataset instances and to summarise and present the data in a novel, understandable and useful fashion [Larose, 2006]. To extract meaningful information from data, different types of algorithms have to be used. Information obtained through data mining can be used, for example, by businesses that would like to know more about their suppliers, customers, employees and products. Such information can also enable businesses to supply customers with the products they need and to recommend products they are most likely to need. Recommender Systems (RSs) use data mining techniques to suggest or recommend items of possible interest to specific users. The recommendation problem has to do with estimating the rating for items not yet rated by a user. The user is recommended items that are predicted with higher ratings. [Ghazanfar and Prugel-Bennett, 2010; Glauber *et al.*, 2013]. The hybrid recommender system developed as part of this research study presents itself as an ideal solution to some of the problems the PLP Group would like to solve. The development of such a system was necessary for that it would enable the PLP Group to, among other things:

- Overcome the information overload problem by recommending the most relevant items to clients thus positioning the service the organisation offers as a convenience it really is.
- Enable the PLP Group to target existing and potential clients with item offers that were well suited to their profile. For example, it would be very annoying to call and offer a single young professional Dial-a-Teacher services as he does not have kids.
- Enable the PLP Group to apply pre-emptive retention measures to clients who exhibit tendencies exhibited by users who have cancelled the service in the past.

3.3 The CRISP-DM Process Model

Data mining, [Cios *et al.*, 2007], is the process of identifying, new, useful and easy-to-understand relationships in data [Fayyad and Smyth, 1996]. Other terms that are used to describe data

mining include information extraction, discovery, data harvesting, archaeology, and pattern processing [Cios *et al.*, 2007]. Figure 3.1 illustrates the stages involved in the KDD process.

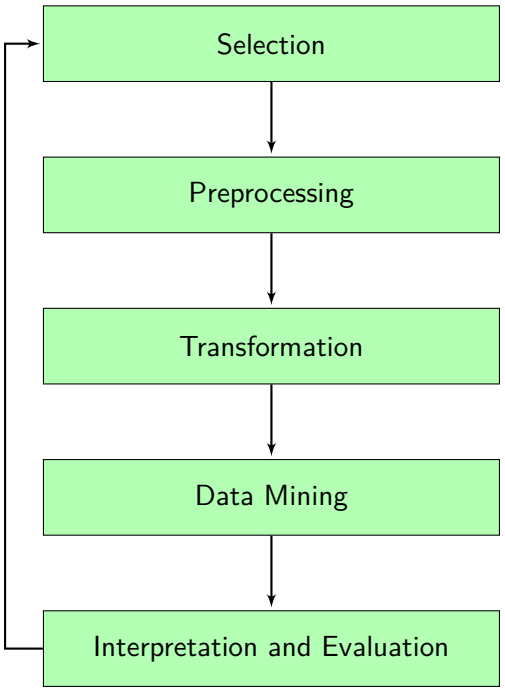


FIGURE 3.1: Generic KDD Process

In the literature, several knowledge discovery process models are discussed [Berry and Linoff, 2004; Cios *et al.*, 2007; deVille, 2006; Fayyad and Smyth, 1996; Larose, 2005]. Apart from minor variations in the number of steps involved, most of these models are similar to the KDD process model illustrated in **Figure** 3.1. Some of these models include:

- Knowledge Discovery in Databases (KDD) Process by [Fayyad and Smyth, 1996] *et al.*
- Information Flow in a Data Mining Life Cycle
- The Sample, Explore, Modify, Model and Assess (SEMMA) process, developed by the Statistical Analysis System (SAS) Institute Inc.[Azevedo, 2008]
- Refined KDD paradigm
- Knowledge Discovery Life Cycle (KDLC) Model [Lee and Kerschberg, 1998]
- Ontology-Driven Knowledge Discovery **(ODKD)** Process [Gottgtroy *et al.*, 2004]
- CRoss-Industry Standard Process for Data Mining (CRISP-DM) [Chapman *et al.*, 2000; Shearer, 2000]

Given the nature of the work that needed to be done in this research the CRISP-DM [Han, 2011; Wirth and Hipp, 2000] process model was selected to guide the activities through different phases. CRISP-DM is an application-neutral standard which defines and validates the data mining process [Han, 2011; Wirth and Hipp, 2000]. CRISP-DM is increasingly being adopted and used as a standard process model in data mining projects. It describes a process model that provides a framework for developing data mining projects and is independent of the industry sector and the technology used. The purpose of using the CRISP-DM process model in projects is, among other things, to make large data mining projects less costly, more reliable, more repeatable, more manageable and faster [Kadav *et al.*, 2003; Wirth and Hipp, 2000]. CRISP-DM consists of six phases, namely business understanding, data understanding, data preparation, modelling, evaluation and deployment [Kadav *et al.*, 2003]. A project based on CRISP-DM is defined as a cyclic process, in which several iterations may be performed to tune data mining towards business goals [Llanes and Puldon, 2008; Moro *et al.*, 2010]. Figure 3.2 shows the different phases of the CRISP-DM process model and the relationships between them.

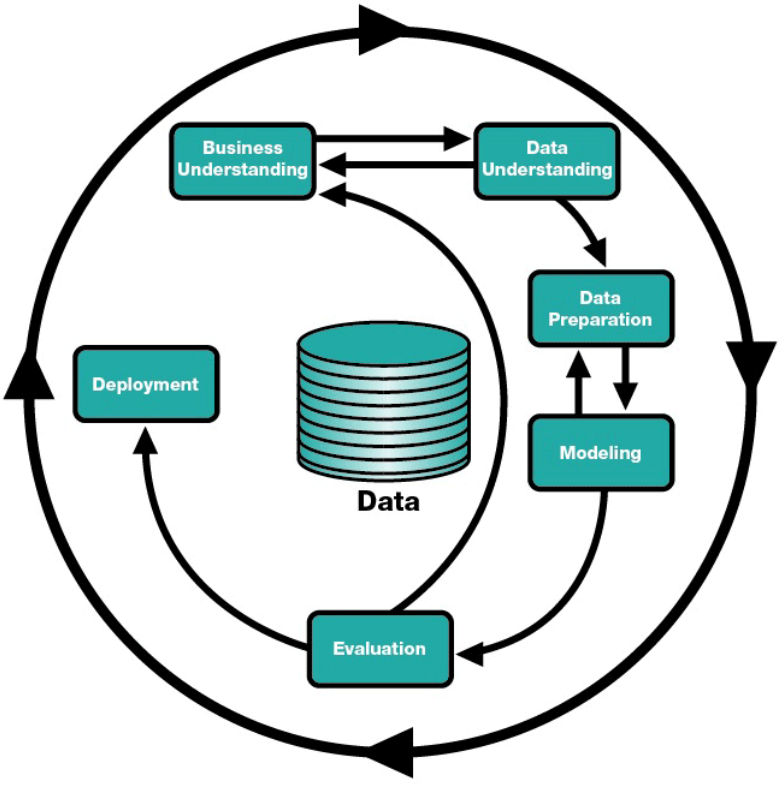


FIGURE 3.2: The Cross Industry Standard Process for Data Mining (CRISP-DM)[Chapman *et al.*, 2000]

As mentioned earlier, the CRISP-DM framework is iterative, with some stages also iterating amongst themselves, as indicated by the two-way arrows.

3.3.1 Business Understanding

This stage is also known as the research understanding phase [Chapman *et al.*, 2000; Larose, 2005; Llanes and Puldon, 2008; Shearer, 2000]. It focuses on assessing and understanding, from a business perspective, the project objectives and requirements, and converts this information into a data mining problem definition. A preliminary plan is then designed to achieve the stated objectives [Kadav *et al.*, 2003; Larose, 2005]. The goals are translated into the formulation of a data mining problem definition.

3.3.2 Data Understanding

This stage begins with data collection, followed by a preliminary analysis, in which the objective is to derive a better understanding of the data and to obtain some insights from it. During this stage the, quality of the data is also evaluated and interesting underlying trends and patterns present within the data identified [Larose, 2005; Llanes and Puldon, 2008].

3.3.3 Data Preparation

Data Preparation is a labour intensive phase in which the data to be used during the project is meticulously extracted and prepared. The cases and the variables to be analysed are selected during this phase, which involves cleaning the data, pre-processing it and preparing it for the modelling phase [Chapman *et al.*, 2000; Kadav *et al.*, 2003; Larose, 2005; Llanes and Puldon, 2008]. Data preparation tasks are usually performed iteratively and in no particular order.

3.3.4 Modelling

Different data mining techniques may be applied to the same problem. For this reason, it is necessary to select and use an appropriate modelling technique. During this phase, it may be necessary to loop back to the data preparation phase to re-optimise and clean the data and to

align it with the specific requirements of a particular data mining technique [Chapman *et al.*, 2000; Larose, 2005; Llanes and Puldon, 2008].

3.3.5 Evaluation

In this phase, the models developed in the modelling stage are evaluated for quality, relevance and effectiveness in solving the specified data mining problem. The models should be evaluated in relation to the extent to which they address the business problem or the extent to which they omit certain aspects of the problem [Llanes and Puldon, 2008; Shearer, 2000; Wirth and Hipp, 2000; Witten *et al.*, 2011]. A decision is made as to whether or not the solution was appropriate for the task at hand.

3.3.6 Deployment

Deployment involves getting the model to production on real business data and producing reports. A deployment plan is outlined before actually carrying out the deployment [Chapman *et al.*, 2000; Kadav *et al.*, 2003]. The models developed have to be put into use in the business.

3.4 Research Setting and Research Subjects

3.4.1 Research Setting

The business motivation for this project has already been outlined in Sections 1.1. The business case for the project at Private Label Promotion (Pty) Ltd (PLP) is a derived case from the contractor, mobile provider Cell C. The organisation handles a service only offered to Cell C contract customers and as such PLP has the onus to improve consumption of the service and thus help Cell C retain its customers. To get more usage from current users, the organisation needs to recommend services which may be of interest to the users judging by the purchasing behaviour of other users with a similar taste and the ratings they give for those items. PLP needs to use data mining techniques to identify common traits and attributes of some of its heaviest users and try to sell the service to users with similar attributes. Similarly, the organisation needs to identify patterns of users who opt out of the service and use this information to initiate pre-emptive membership rescue attempts to reduce member cancellations. There is also a need

to identify the general profile of users who join the service so as to do target marketing to new contract cellphone subscribers to sell the GetMore Service to. Moreover, the service needs to be aligned to the needs of the majority of the subscribers. Requests for services are logged one after the other forming a symbolic sequence. A symbolic sequence consists of an ordered set of elements or events, recorded with or without a concrete notion of time [Han, 2011].

3.4.2 Research Participants

As already explained in Section 3.3.1 the dataset was made up solely of Cell C mobile phone contract customers. Table 3.1 and 3.2 give an overview of the composition of the data used in this study.

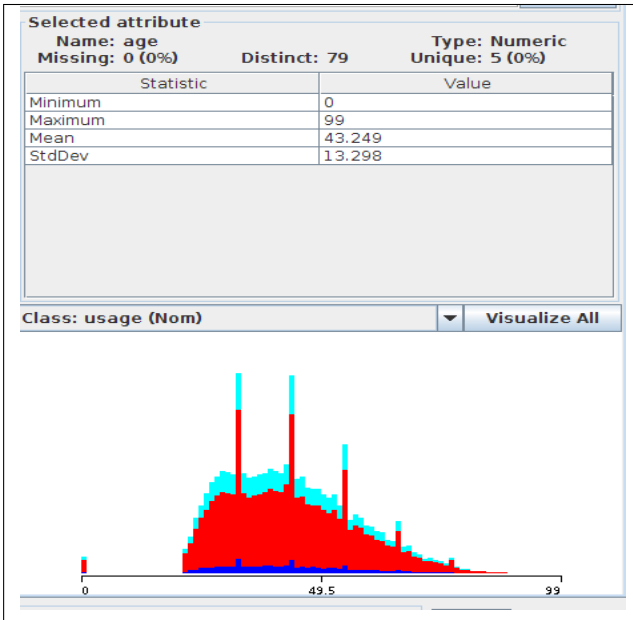
Users by Age	Users by Gender
	

TABLE 3.1: Users by Age and Gender

Users by Province	Users by Status

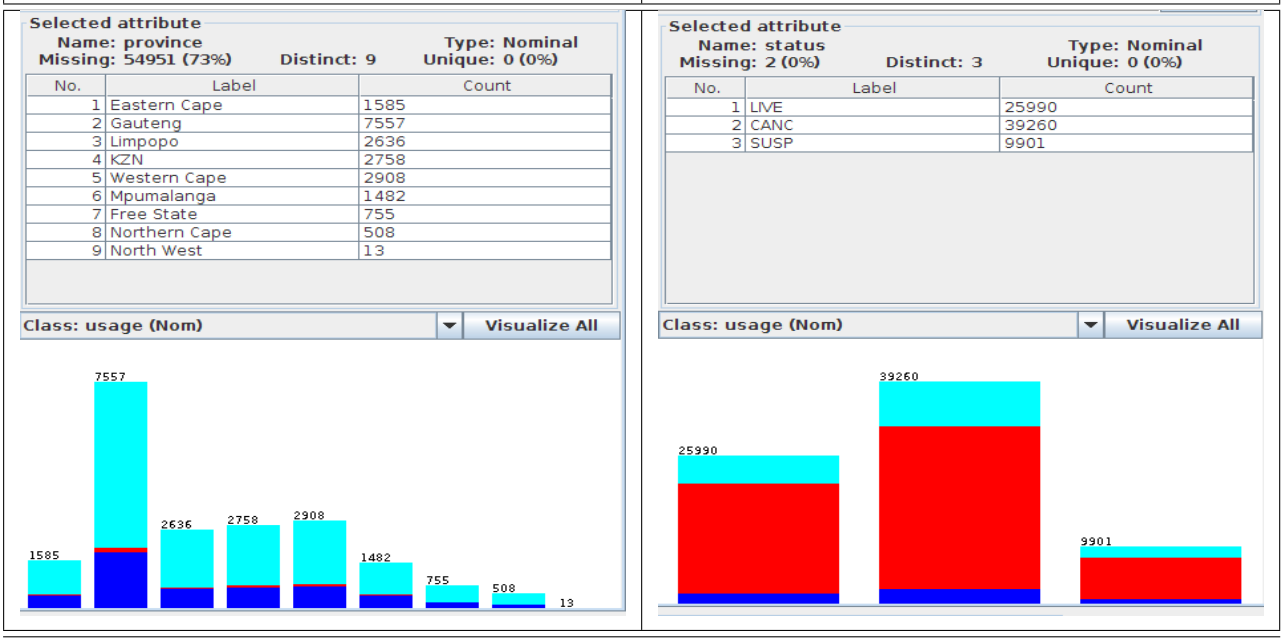


TABLE 3.2: Users by Province and Status

The four attributes chosen for data analysis above, age, gender, province and status, are analysed against their usage statics. Status was used to measure user churn. Suspended members are those who have missed out on their contract payments, while members can choose to cancel the GetMore service and still retain their Cell C contract. The usage statics were arbitrarily grouped into high, medium and low users based on the average requests logged by a client. The 0 in the age analysis was just a default value for those users whose age was not provided.

Features like race, income and education level which could have been very useful for this study were not provided. The dominance of males is very striking. Which, as a matter of opinion may reflect the dominance of men in formal occupations which are preferred by mobile phone contract provides.

3.5 Prototype Hybrid Recommender System

One of the major tasks of this research was to design, develop and implement a prototype hybrid recommender system which would be used to recommend the PLP Group's products or services to existing or potential customers. Apart from being used to collect and process customer-related data, the system also incorporated data mining and hybrid recommendation algorithms. The system's architecture, including the different aspects of its operation, are discussed in Subsections 3.5.1 to 3.5.4.

3.5.1 System Architecture

The system developed in this study uses a Client-Server architecture. The server end consists of an application server and a database server. Wildfly, [RedHat], and MySQL, [Inc]. The application it self consists of a web interface, an item- and a user data model. When the application is launched, the user is presented with the screen shown in Figure 3.3 below:



FIGURE 3.3: Application Interface

After logging in, the application triggers three servlets: the user servlet, the item servlet, and the recommendation servlet. The user servlet prints out the user surname, initial, email address and account number at the top-left corner of the web page. Below that there are two tables, one on the left showing the recommended items and another one on the right showing all the items the user has used in the past. The individual servlets for the user with membership ID 14837778 was something like this:

```
{'membersId':'14837778', 'sname':'Greyling', 'email':'' ,'status':'LIVE' ,'usage':'low' ,'pr
```

These are the user attributes described in table 3.5.

The outputs of these servlets is shown in Figure 3.4 below:

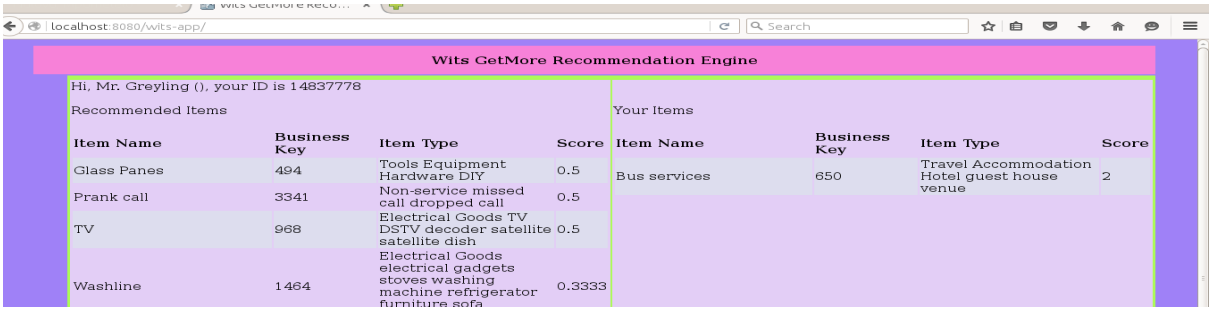


FIGURE 3.4: The Recommendations

The three servlets are triggered by a Javascript engine embedded in the HTML front end. The database structure and data are included in the web archive (war) file. Their similarity score

orders recommended items in a descending order. Similarity scores are given as a fraction of 1. If the user has not rated any items in the past, the user servlet will return an empty list and provide recommendations based on user-user similarity

3.5.2 Hybrid Recommendation Algorithm

In this text, we made various assumptions based on the conclusions drawn by previous studies discussed in Chapter 1. From the previous chapter we realised that we can define a binary relationship between users and items. Where a user, $U_k = \{u_1, u_2, \ldots, u_n\}$ and item, $I_j = \{i_1, i_2, \ldots, i_k\}$ are two sets of objects where the binary relationship, R, between U and I can be expressed as $R \subset UxI$. This binary relationship between the users and items can be roughly illustrated by a two-dimensional table like table 3.3 below:

	i_1	i_2	...	i_k
u_1	1	1	...	0
u_2	0	1	...	1
...	...	...	...	...
u_n	1	1	...	0

TABLE 3.3: User-Item Binary Relationship

Although table 3.3 above is easy to understand, it misses the important point that each of the users and items is an embodiment of their attributes which gives them a third dimension. This is best captured by a database foreign key relationship.

An incremental hybrid algorithm is given by Algorithm 3 below. The inputs to the algorithm include the user id (u_{id}) and two or three other user-specific parameters. These are the total number of items to be recommended (N), and the number of neighbours kN, items rated by user NU . It essentially has four steps.

Procedure 3 The Hybrid Algorithm

Input:u_{id}, N, kN, NU

Output: Items(I_r)

procedure HYBRID ALGORITHM

I_j = list of items used by user u_{id}

X = the user-item matrix for all users.

If $X = \emptyset$;

Or $I_j = \emptyset$; **then**

I_r = recommended items by user data model;

Return I_r

EndIf

$U = \{u_1, u_2, \ldots, u_n\}$

count = 0;

For $u_i inU$ **do**

N_r = Number of items rated by user u_i.

If$N_r \geq NU$ **then**

Count = Count + 1;

EndIf

EndFor

If Count ≤ 0 **then**

I_r = Items rated by top-n most similar users

returnI_r

EndIf

N_k = N * (1 - kN)

I_k = recommended N_k items by the item-based recommender which uses the kNN algorithm;

$N_{rd} = N - |I_k|$

I_{rd} = recommended N_{rd} by the user-based recommender

$I_r = I_k \cup I_{rd}$;

return I_r

The algorithm above goes through the following broad steps:

Step 1 . Initially, there will be usage data in the user-item matrix.In this case, we use user attributes to deal with the cold-start problem. The top-n most similar users are used generate recommendations for the active user. A data model uses user attributes to establish user similarity and to generate recommendations.

Step 2 . NU, the number of items rated by the user, is used to determine whether or not to use the item-based k-NN algorithm or not. We first count the number, N_k,items rated by the

user u_i. If $N_k > 0$,then the item-based kNN algorithm is otherwise the user data model will be used.

Step 3 . When the recommender is not misbehaving, the user model is used to recommend some new items for exploring new interests.

Step 4 . The kNN algorithm is used to recommend items based on item-usage similarity. The other items are recommended based on user attribute similarity.

3.5.3 System Flow

The user logs into the system using his userID and a password given during registration. Once he has logged in, the system reads the user's profile data which includes his personal details and usage history, and prints them on the screen. It lists his latest ten items on the screen together with user's name and email address and proceeds to compute recommendations. However, if the user has never used the system before, the recommender will identify top 10 most similar users using user attributes and compute and proceed to recommend items for the user using the neighbourhood users. Figure 3.5 below gives an outline of the recommender system flow.

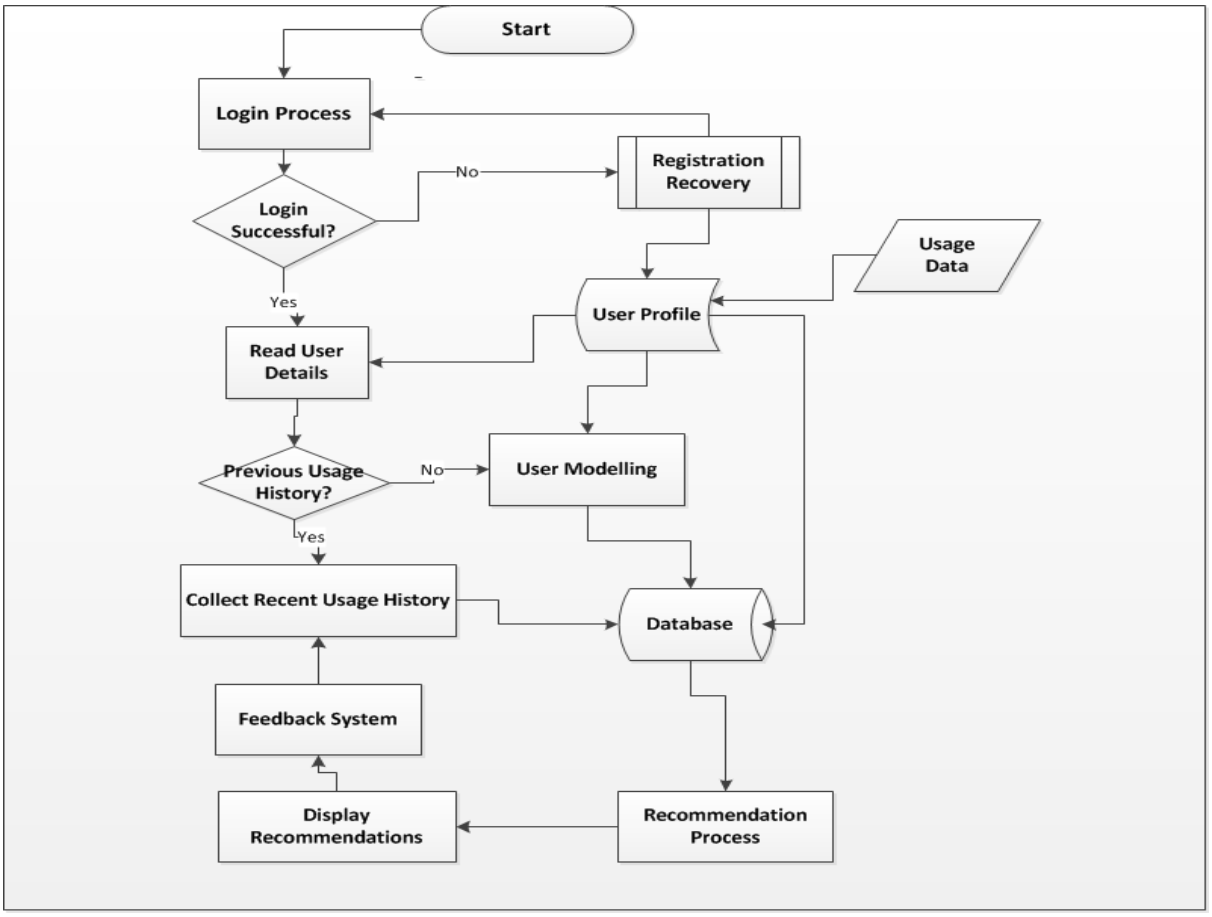


FIGURE 3.5: System Flow

3.5.4 User-Recommender Interaction

The recommender system is divided into the user and system parts. The user logs in and views and logs requests for items. The recommendation part should show the user more items that are similar to the ones he has requested and items that other users similar to him have requested in the past. For each user u, the user-system interaction process is as follows: (1) the user logs into the system and the system quickly retrieves his/her profile; (2) the user gets ten recommendations from the RS based on his profile or previous interactions with the RS; (3) whether or not the user requests an item is based on his/her own interests. If the user requests an item, the process continues. Otherwise, she is not interested in any of the items recommended by RS in which case she is free to choose her item. The recommender will then provide new recommendations based on the user's

3.6 Applying the CRISP-DM Process Model to the Research Problem

In many projects that involve data mining, the CRISP-DM Process Model is increasingly being adopted as the methodology of choice for implementing projects. A detailed introduction and discussion of the model were provided in Section 3.3. In this study the CRISP-DM methodology was used to guide the direction of this project and to help provide the required structure for it. In this section we discuss how the different phases of the methodology were implemented, based on the operations and activities of the PLP Group.

3.6.1 Business Understanding

The GetMore service offers the following main categories of items to Cell C contract customers:

- **Discount Services**
  - Habari discounted services where Cell C customers get fantastic deals when booking or buying these services.
- **Expert Services**
  - This is a *"Rent-an-Expert"* group of services, which links customers with experts registered with Cell C to offer their services at a discount.
- **Assistant Services**
  - The Personal Assistant group of services performs mundane tasks for the customers.
- **SOS Services**
  - Emergency services offered to customers in distress, both at home and in transit.

Table 3.4 lists some of the benefits currently available in each of the above-named categories of the Get-it Service.

Discount Services	**Expert Service**
2-4-1 Dining: over 500 restaurants to choose from.	Teacher On Call,
2-4-1 Movies: view for two, pay for one.	Lawyer On Call,
2-4-1 Well-being: pamper for two, pay for one.	Tax Expert On Call,
2-4-1 Experiences: two activities for the price of one.	Finance Expert On Call,
Travel:flights, accommodation, car hire.	IT Expert On Call,
Wheels: new or used vehicles, tracking, smash & grab, car kits.	Home Expert On Call,
Shop: furniture, toys, decorators, caterers, fashion, jewellery.	Insurance Expert On Call,
Classifieds: sell, swop or buy anything online	Recruiter on Call.
SOS Services	**Assistant Services**
Home SOS: Emergency services at home like plumbing, electricians, swimming pool and the like.	make bookings
Medical SOS: Medical emergency services	ordering, sourcing, selling old antiques,concierge services, etc
Roadside SOS: Roadside emergency assistance.	paying for service tickets like traffic tickets and E-Tutor Services

[CellC GetMore, 2014]

TABLE 3.4: Benefits offered by the GETMORE Service

Briefly, to use the service, customers log requests in sequence, [Pei and Han, 2004; Wang and Han, 2007], via SMS, phone, web portal or email. These requests are then processed, sourced or forwarded to the relevant service providers and feedback given to the customer who logged the request. Follow-up activities may include requesting the customer to rate both the service he or she received and the suppliers of the service recommended to him or her. More details on the logging, sourcing, onboarding, inbound and outbound processes involved in servicing customers will be discussed in Section 3.6.2, the data collection section.

The main business questions addressed in this study are described in Chapter 1. The organisation needed to find customers whose spending patterns over time were similar to a given spending profile. The business requirement needed the study to use both collaborative filtering and content-based algorithms. Content-based Filtering manifested itself when we used item attributes and user profiles to determine similarity and correlation while Collaborative filtering relied on previous ratings of items by other users.

The service items were divided into four major categories listed in *Table* 3.4. To attract more customers, the items on offer had to be appealing and relevant to the needs of the customer. They had to provide sufficient utility to the customer to warrant his attention. While getting more users to join could be easier since many joined out of mere curiosity, getting them to use the service more frequently was a bit of a challenge. This involved creating reliance on the service by users by keeping a step ahead of their needs all the time. That was what the GetMore Service was all about - being everything to everyone. If getting more usage out of the current subscribers was difficult, then maintaining loyalty was almost near impossible. Loyalty required the service to constantly adapt, evolve and introduce novel categories and items to satisfy a diverse clientèle in a fast moving consumerist society. Data mining techniques were thus needed to recommend items to the over 300 000 plus subscribers while also giving an insight to the organisation as to which items are suitable for which category of customers.

Business Problem

The more Cell C contract customers registered for the GetMore service, the higher the likelihood that they would stay with Cell C, thereby improving the company's chances of retaining its customers, which is one of the major objectives of mobile service provider companies in South Africa and elsewhere in the world. By subcontracting the GetMore Service to PLP, Cell C has tasked PLP to help it surmount its challenges and the problems listed above. The motivation for PLP to attain these objectives is simple: Cell C pays this company, depending on the number of old and new contract customers who use the service, as well as on the number of new customers who register for it. PLP needed to use the data at its disposal to identify usage trends and patterns. This information would then be used to increase customer usage of the service, retain users and also attract new users. The challenge that is facing PLP is that of increasing usage and acceptance of the GetMore service, could be met by enrolling more members into this service, prompting existing members to use it more often, keeping existing members longer and hoping that their loyalty would lead to continued usage of the service.

To broaden the scope of usage among current users, PLP needed to recommend other items that could be of interest to these users. By observing the purchasing behaviour of users who shared a similar taste and how well they rated the service(s) they got, PLP could then recommend a

list of similar items to the users. Given the vast amount of customer data at its disposal, the organization could use data mining techniques to obtain meaningful information about some of its customers who used the service consistently. The company could then use this information to try and market the service to other users who shared similar tastes. It was also important for PLP to identify the characteristics of users who opted out of the GetMore service and to use this information to initiate pre-emptive strategies aimed at reducing or stemming the tide of member cancellations. There was also need to identify the general profile of users who joined the GetMore service and to use this information to target the market in order to entice new contract cellphone users to subscribe to the service. Furthermore, the service had to be aligned to the needs of the majority of subscribers.

Worth noting is the fact that direct marketing, or target marketing [Shen and Chuang, 2009], has its challenges. There are legal and privacy issues to deal with [Milne and Rohm, 2000; Weiss, 2005]. Consumers own any information about themselves and access to it should be restricted [Milne and Rohm, 2000]. The buying and selling of such information is illegal. Despite the legal restrictions, according to a study by Dolnicar and Jordaan in [Dolnicar and Jordaan, 2007], only 16% of South African respondents indicated that they did not mind receiving targeted marketing calls. When wrongly applied, targeted marketing can scare away about 84% of potential customers. It was important, therefore, for a recommender system to be used to target customers by recommending products most relevant to them or else risk losing them [Cheung *et al.*, 2003]. This proposition was made with the assumption that the organization feels it in its advantage to maintain a symbiotic relationship with its customers [Dolnicar and Jordaan, 2007] and that the organisation was committed to reducing negative publicity and complaints emanating from violated consumer privacy.

A powerful recommender system could personalize products to customer needs, but to do that it needed to collect customer data, create a customer profile, match it to available goods, make a recommendation and measure the response of the client [Adomavicius and Tuzhilin, 2001]. It used data mining algorithms to suggest items that may be of interest to a given user. The problem of recommendation could be seen as the problem of estimating the user rating for items not yet rated by the user. The top-n items predicted with high rating scores for the active user can be offered for recommendation by the recommender [Glauber *et al.*, 2013]. Recommender systems (RS) applied machine learning algorithms and data mining techniques to filter large amounts of information and predict if a user would like a given item or not [Ghazanfar and Prugel-Bennett, 2010].

Given the context that has been provided above, the following are some of the interesting questions this research sought to address:

1. Given the amount of data at PLP's disposal, how could the organization leverage this data to its advantage and use the information generated to advise new customers on selecting or purchasing items most preferred by their peers? How are the peers identified, and how does the organization determine what their most preferred items are? The organisation needed to find customers whose spending patterns over time were similar to a given spending profile [Gorunescu, 2011].

2. Given the amount of data the organisation had on a customer, could it show him/her more of the same of what he/she has already liked? If this was possible, how could the organisation identify similar items? How could it recommend them to the users?

To answer these and related questions, data mining techniques were applied to a customer and item dataset, and the information obtained was used to provide personalized recommendations to potential customers. In this study collaborative-filtering and content-based filtering techniques were used in the hybrid recommender system that was implemented as part of this research study. Content-based filtering involved using item attributes and user profiles to determine the similarity and correlation between items. Collaborative filtering relied on previous ratings of items by users to recommend items and used data mining algorithms to analyse past trends, item attributes and user knowledge to predict customer behaviour and needs. These techniques reduced or minimized the information overload problem by recommending to users the most relevant information.

Business Success Criteria

In short, this study was commissioned with the objective to improve cross-sales by making better recommendations and to increase customer loyalty with a more personalized service. Tentatively, although this is covered in a later section, the study could be judged a success by business if cross-selling increased by 10% or higher and customers spent more time and requested

more items per request. Figure 3.6 below does not paint a good picture:

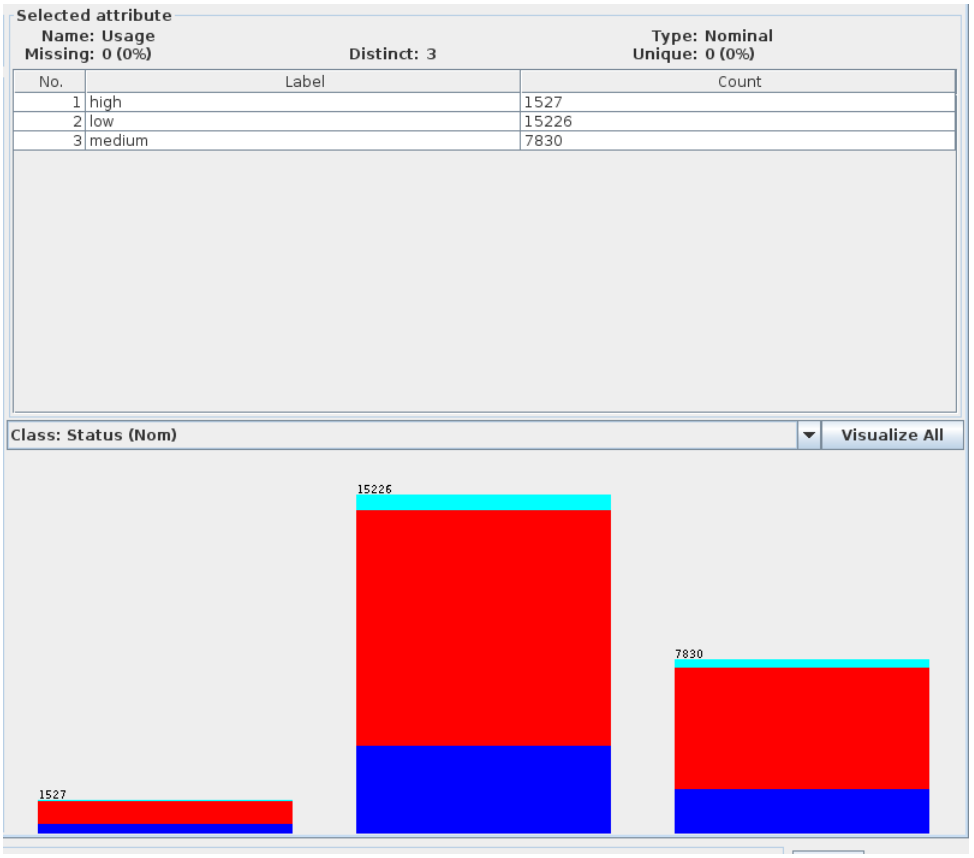


FIGURE 3.6: Usage vs Status

One thing that comes out clearly is that even for high users a larger proportion cancels the program. This trend may be a sign that users do not opt to remain in the programme after their contracts with Cell C expire or simply that the program does not adapt enough to changing user needs.

3.6.2 Data Understanding

The previous section described the major business setting for the processes which generated the data which was used in this study, the business objectives and success criteria. This section describes the actual data generation process,and how it was initially analysed and cleaned. Three intersecting, non-discrete categories of data were used in the study. This consisted of customer data; item attributes data and item requests, which introduced the association between users and items through rating and simple usage information. The data was originally stored in a MySQL Database before it was extracted into text files.

Four business processes generated the data used in this study and were involved in helping the organisation attain its objectives. These were Outbound, Inbound, Onboarding and Quality Assurance (QA) illustrated in Figure 3.7.

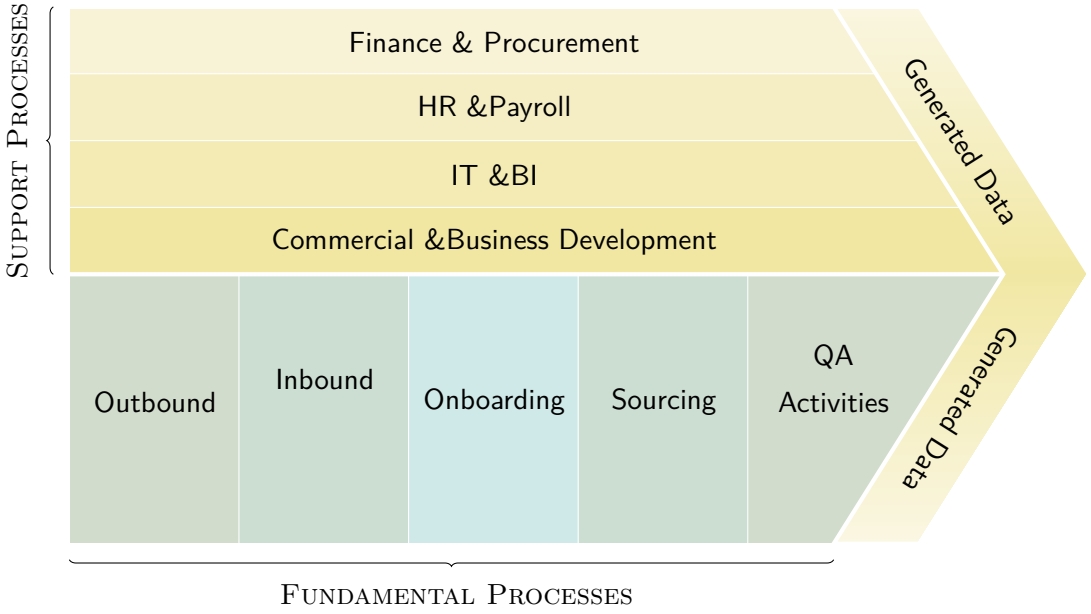


Figure 3.7: Business Processes

Outbound

This is the process which sells the service to new Contract Customers. Once the client has signed a Cell phone contract, his/her details are forwarded to the outbound team through a sophisticated lead management system who do their best to introduce the service and its benefits. This represented the 'New user - existing item' scenario introduced in **Section** 1.2 which is equivalent to the first-rater problem in recommender systems. One of the main objectives of the organisation was to increase the number of new users who joined the service.

Inbound

To use the service, once registered, the customer logged a request with the Call Centre via phone call, SMS, email, web service, mobile application or social networks. When logging a request, the client selected any item from the over 1150 items on offer. It was at this point where the logging agent got to ask the customer a few questions about himself to create a tentative profile of the customer which provided this study with the user profile attributes.

One of the main objectives of the organisation was to increase the number of logged requests.

Onboarding

The Onboarding Process had the unenviable task of retaining users and increasing usage. This department continuously marketed new items to clients encouraging them to use some of the items they had not tried before or which they had used before. They used all marketing skills and principles to increase usage and retain membership. Membership retention was another of the major objectives of this study.The onboarding process would benefit immensely from the recommendation system developed by this study.

Quality Assurance (QA)

Quality Assurance engaged in various activities ranging from Service Level Agreements (SLAs)satisfaction to customer feedback (Follow-up Activity). For this text, we only considered their follow-up activity as it generated the most relevant data for item recommendation. Among other things, the follow-up activity asked the customer to rate the service, the supplier, and the price within 14 days after the item requests. It was a customer satisfaction index survey which generated an 'our-service-to-you' report.

- **Service rating**

  Here the client rates the item he requested and the whole service he got from PLP.

- **Supplier Rating** After logging a request, a customer received a recommendation of up to five suppliers of the item he requested. These were selected based on price, previous ratings and the preferred shopping area of the client. If the customer did not use any of the suggested suppliers, he also gave a reason for his choice, why he opted not to take the service's advice.

- **Price Rating**

  At this point, the customer gave feedback on whether he thought he had saved by using the suggested supplier. The system computed savings by calculating the difference between the highest supplier price and the lowest price. Customer ratings were filtered in to validate the accuracy of the ratings.

Obtaining Data for the Research

Three main datasets were produced by the above data, namely, customer profile data, item usage data and item rating data.

- **Customer Profile Data**

  This dataset contained customer profile attributes identified by an auto incremental member number. The major attributes included the date of birth, residential province, gender, date joined the service. The data structure of the Customer Profile file is illustrated in *Table* 3.5. It shows the major attributes used in user-similarity modelling.

Attribute	Data Type	Description
userID	integer	Auto increment customer number used to track customer activity
user_birth_month	varchar	Customer's date month to be used in determining the relationship between request and birthday month
gender	varchar	For gender breakdown
age_range	varchar	For age analysis
province	varchar	For geographical analysis
user_birth_year	varchar	the user year of birth expected to correlate strongly with age
getmore_join_year	varchar	year user joined the service
getmore_join_month	varchar	month the user joined the service
residential_province	varchar	For geographical breakdown
time_left	interger	Time left before the Cell C contract is paid off
tariff_plan_desc	varchar	There are various cellphone contract plans on offer
contract_duration	integer	users cellphone contract length in months
status	varchar	User membership status for churn

TABLE 3.5: Customer Profile

Market segmentation information was collected during the Onboarding call, with the exception of the personal interests which was collected on a separate call. The amount of

information collected was subject to the clients' availability and willingness to participate. Data in this regard was very sparse.

- **Item Attributes**

The items in this text refer to the possible requests which could be logged by a given customer. The logging agent, who received a call from the customer, selected an option under which the request fell. There were over 1150 options which included periodic competitions and promotions which were be treated differently for this research. Each item is identified by a unique incremental *item_id*. The item attributes file structure is illustrated in **Table** 3.6. It contains some rating information summed up. The rating dataset in subsection 3.6.2 gives a clear connection between users and items.

Attribute	Data Type	Description
Item_id	integer	auto increment number
item_category_id	integer	item major category classification
item_type	varchar	Rough item grouping meant to expand categories and prune items
request_source	integer	An indication whether the customer logged the request by email, phone,web,etc.
Item_Desc	varchar	Text description field
NoOfRequests	integer	Sum of requests logged to date for the particular item.
Avg_item_rating_rats	integer	Average item ratings rounded.

TABLE 3.6: Item Attributes

The rating attributes contained a null rating by default for all newly added or existing items which had not yet been rated. The total number of requests for an item were also used to compute implicit ratings to counter the limitation of data sparsity.

- **Rating Data**

The rating data forms a huge component of the collaborative filtering aspect of this

study. Before processing, the rating dataset is provided as a comma-delimited file in the following format:

userId	itemId	itemCategoryId	dateLogged	requestSourceId	averageRating	...
17509134	3353	80	2014-04-07	1	3	...
17509134	3203	80	2014-04-07	1	2	...
16526153	3261	61	2014-04-08	0	2	...
16526075	3261	61	2014-04-08	1	3	...
16940930	2345	80	2014-04-09	2	null	...

TABLE 3.7: Rating Data

The date field in the rating data above was no used anywhere in this study. Such information is valuable in algorithms that use sequential data modelling. The ratings are given as an average of item rating, supplier rating and price rating. The rating data in Table 3.7 can easily be represented as an extremely sparse two-dimensional matrix with the users in one dimension and the second dimension spanning the items.

The rating matrix, illustrated in table 3.8 below, contained the sum and average of all ratings by the user on an item and the status of the user.

user_id	**status**	**2 for 1 Deals**	**Accessories**	**Accommodation**	**....**	**....**
u1	LIVE	12			3	
u2	CANC			15		
u3	LIVE		1			3
....	SUSP			9		
....	CANC	1			12	

TABLE 3.8: Rating Matrix

This file was used to compute both implicit ratings and explicit ratings.In this case, implicit ratings were derived from the number of requests logged for the item while the ratings determined the explicit ratings. The variances in the rating scales account for some ridiculously large values in the above matrix. The data is to be used as a complete set as it is not split into training and testing data. Fill Rate is given by Equation 3.1.

$$FillRate\% = \frac{No.\ Of\ Ratings\ *\ 100\%}{No.\ Of\ Users\ *\ No.\ Of\ Items} = \frac{16479 * 100\%}{75153 * 1109} \approx 0.02\%. \tag{3.1}$$

The result in equation 3.1 means that less than 1% of the items are rated. The scatter graph in figure 3.8 below gives a visual representation of how scanty the rating the rating information at our disposal was.

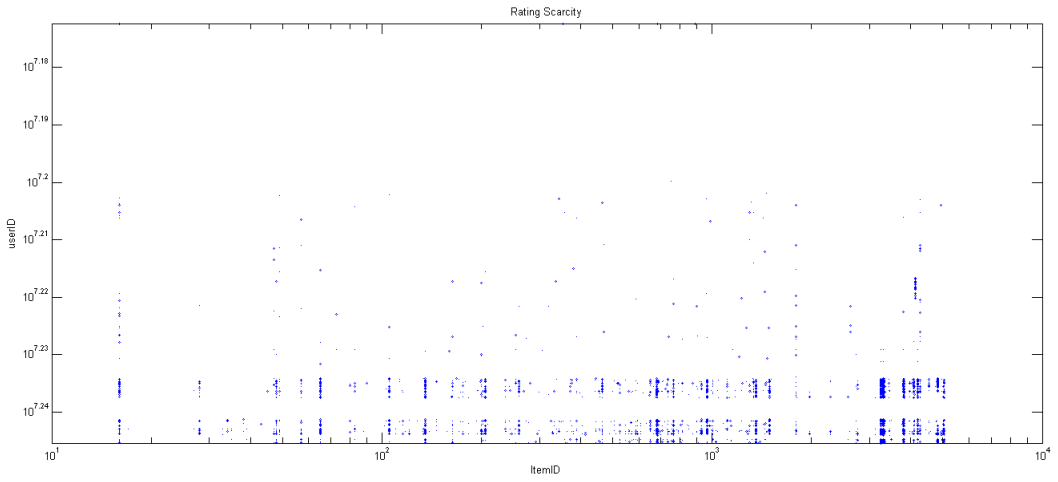
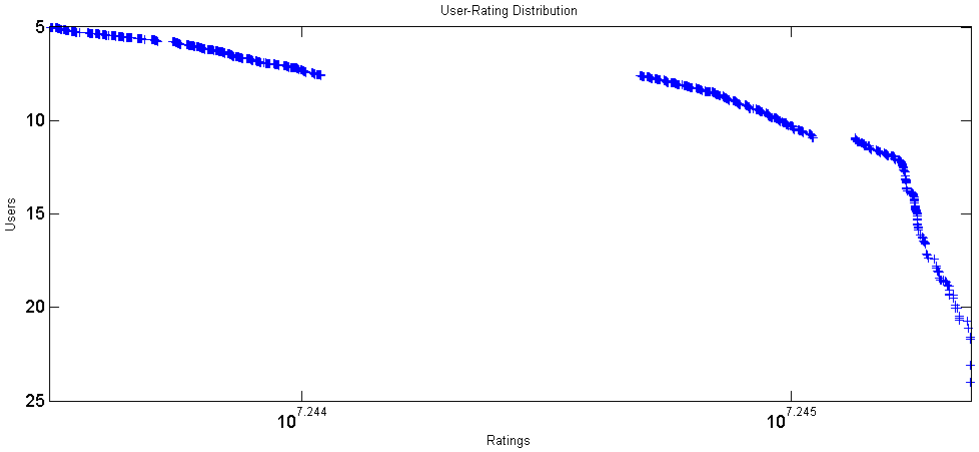
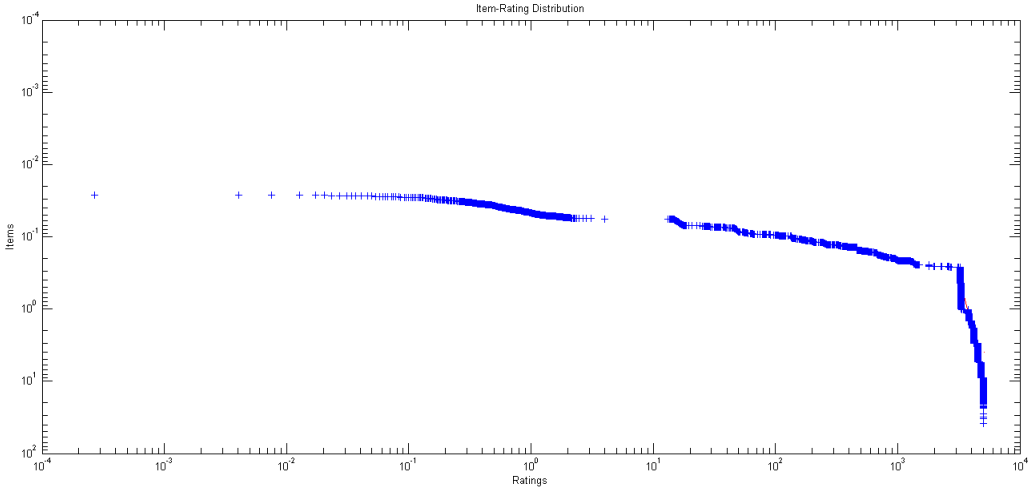


FIGURE 3.8: Rating Data Sparsity Distribution

Figure 3.9 shows the distribution of ratings for both users and items. In tandem with the *power law*, the rating density increases slowly as the number of users or items decreases.



(A) User-Rating Distribution



(B) Item-Rating Distribution

FIGURE 3.9: Training Data Distribution of Ratings

Since one of the objectives of this study is to discover nearest neighbours based on item similarity, the dearth of rating information in item space left us with a predicament.Therefore, this justifies why it is prudent to use both user-based and item-based nearest neighbourhood approaches complemented by user and item attributes to improve prediction accuracy.

3.6.3 Data Preparation Phase

During the preparation of the data, necessary activities were performed to determine the data format that is accepted by the recommendation tool. This phase of data preparation was considered preliminary because in the modelling phase each algorithm required certain data preparation steps to be followed. The phase is characterised by five main tasks, namely data selection, data cleaning, data construction, data integration and data transformation.

Data Selection

This stage involved identification of relevant fields and a justification for their inclusion/exclusion. The feature selection tool of Weka was used for attribute selection. Pearson's correlation coefficient cut-off for this project was set at $r >= 0.95$. In any given pair of variables which have an *r-value* higher than this one of them was removed from the dataset since it indicated redundancy. This was done for both the Customer Profile and Item Attributes data sets. The initial results were also analysed for the possible effects of outliers.

Clustering was the first step in identifying general groupings of users and items and trying to draw user feature-item feature correlation. When clustered, the system should be able to show the user what other users have liked and items similar to the ones he liked. The Java code used in attribute selection is given in Appendix B.2

Data Cleaning

The major tasks in this stage involved the removal of invalid requests where no item was selected and the merging of identical items where the difference was only in the spelling or preference. In some cases, there were items given as 'watch Bafana Bafana' while elsewhere it was given as 'watch the South Africa Soccer team'. There were also items marked as 'inactive' which were not used anywhere else. These had to be removed. Some attributes also needed to be renamed to reflect study-related vocabulary.

Data Construction

Critical attributes like gender, user-birth-month and age had to be derived from ID numbers. The average time between requests and the total number of requests per item per user were also derived from the dataset.

Data Integration

Some of the data had to be normalised while some of it had to be converted from numeric to nominal to make it discrete and truly reflect the data item. The total number of requests (usage) was also included as a user feature to give an indication of how active a user is. The month the request was also added to enable identification of a possible correlation between requests and birthdays.

Data Transformation

Transforming the was a minor task of reformatting dates and converting text zip codes to provinces for consistency.

Using the RapidMiner, selection of features using a correlation matrix proceeded through the following four steps,

(a) Load the data set into RapidMiner without specifying the label or target variable

(b) Connect the data to "Correlation Matrix" operator and run the analysis.

(c) Identify variables which have a correlation coefficient of

$$r > 0.95$$

(d) Among the selected variables, check which ones are correlated to one another and if they are, pick the one with a stronger correlation to the target

Clustering was the first step in identifying general groupings of users and items and try to draw user feature-item feature correlation. When clustered, the system can show the user what other users have liked and items similar to the ones he liked.

- **Feature Correlation**

  A user-item matrix combined attributes of both users and items. Combining these features fitted well with the purpose of this text which intended to create a hybrid recommender system which used both user and item features for the CBF part. The example in Figure 3.10 shows how the strength of correlation between user demographics and item features was exploited.

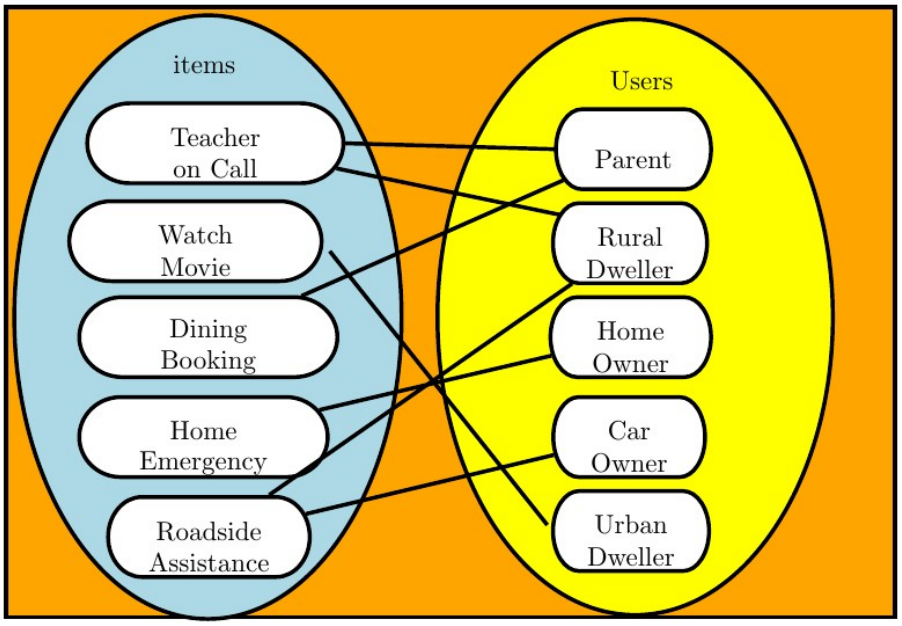


FIGURE 3.10: Sample Item-User Demographics Correlation

The knowledge obtained from the attribute correlation figure in 3.10 can be made useful for our model by representing it as a matrix similar to Table 3.9:

		User Features									
		Age-Range					Area				
		> 25	25-34	35-44	45-54	...	Town	Rural	Farm	Metro	...
Items	Teacher on Call	0	1	0	0	...	0	0	0	1	...
	Home Emmergency	1	0	1	1	...	1	0	0	0	...
	Roadside Assistance	1	0	0	1	...	0	0	0	1	...
	Dining Booking	0	1	0	1	...	1	0	1	1	...
	Business Travel	0	0	1	0	...	0	1	0	1	...
	Watch Movie	0	1	1	1	...	0	1	0	0	...

TABLE 3.9: User-Feature Item Matrix

In the above matrix, feature extraction pointed out some strong relationships between certain user attributes and certain items. A similar matrix for user-item features was created but was less revealing due to few item attributes. These two matrices created a relationship between item features and user attributes. A third matrix, which tried to illustrate the user-item ratings was very easy to create, but needs to be normalised to

reduce collinearity and to standardise rating scales as already indicated in Table 3.8. Item features included category, type, description and average ratings. The item attributes were a bit more complicated to extract. This was because the service provided a very wide range of items which could not be easily classified

The Item Matrix is illustrated in Table 3.10.

		Item Attributes			
		Att_1	Att_2	Att_3	...
Items	Teacher on Call	0	1	0	...
	Home Emergency	1	0	1	...
	Roadside Assistance	1	0	0	...
	Dining Booking	0	1	0	...
	Business Travel	0	0	1	...
	Watch Movie	0	1	1	...

TABLE 3.10: Item-Feature Matrix

Finally, the User-Item matrix looked like Table 3.11. Which is an exact replica of Table 3.8.

		Items			
		$Item_1$	$Item_2$	$Item_3$	...
Users	$User_1$	0	1	0	...
	$User_2$	1	0	1	...
	$User_3$	1	0	0	...
	$User_4$	0	1	0	...
	$User_5$	0	0	1	...
	$User_6$	0	1	1	...

TABLE 3.11: User-Item Rating Matrix

In preparation for the modelling phase, the three matrices illustrated by **Table** 3.9, **Table** 3.10 and **Table** 3.11 needed to be combined in one 3-dimensional matrix. The matrix combined the user features, item attributes and item ratings by individual users.

- **Submatrices**

Both the User and the Item matrices were made up of various submatrices.

– User-Profile Sub Matrices

As already explained in the above subsections, the User matrix, U, is made of various submatrices. These submatrices can be presented in the following format:

$$\begin{pmatrix} U_{1,1} & U_{1,2} \\ U_{2,1} & U_{2,2} \end{pmatrix}$$

The first submatrix, the smallest, represents user gender:

		Gender	
		$Male$	$Female$
Users	$User_1$	0	1
	$User_2$	1	0
	$User_3$	1	0
	$User_4$	0	1
	$User_5$	0	0
	$User_6$	0	1

TABLE 3.12: Gender Sub Matrix

Another submatrix, is the User-Age matrix:

		Age Groups			
		$0-25$	$26-35$	$36-55$	55^+
Users	$User_1$	0	1	0	...
	$User_2$	0	0	1	...
	$User_3$	1	0	0	...
	$User_4$	0	1	0	...
	$User_5$	0	0	1	...
	$User_6$	0	1	0	...

TABLE 3.13: User-Age Matrix

The $User-Age$ submatrix, like user-gender, was characterised by high data sparsity since a user can only belong to only one age group. Similar submatrices were self-evident for location, tariff plan, and status. The biggest of these was the $User-ItemRatings$ Matrix indicated in Figure 3.11. There were over 25000 users and 1150 items under study.

– Item-Attribute Submatrices

Top on the Item-Attribute group of submatrices is the item-item_type matrix:

		Items Groups			
		Discount	*Expert*	*SOS*	*MyPA*
Items	$Item_1$	0	1	0	...
	$Item_2$	1	0	0	...
	$Item_3$	1	0	0	...
	$Item_4$	0	1	0	...
	$Item_5$	0	0	1	...
	$Item_6$	0	0	1	...

TABLE 3.14: Item-Item_type Matrix

The item-supplier submatrix was initially envisaged to be the biggest of them all, could not be constructed due to the absence of supplier data in the study dataset. Similar submatrices were created for item ratings and categories.

- **Combining the Matrices**

Figure 3.11 shows a synoptic view of the item and user matrices combined for analysis. The user matrix attributes consisted of:

i. Demographic data like age, gender, geographical location and user birth month where applicable.

ii. Cellphone contract-related data like tariff plan, contract duration, time left on contract and other minor details.

iii. Item rating data consisted of the average rating given by the user for any of the items the user has used or rated. Average ratings included average service (item) rating, overall average rating and the average price rating. The total number of requests per user per item also formed part of this matrix as well as the average interval in days between item requests The average rating was used just in case the user gave different ratings for an item he used more than once and to keep the rating scale normal.

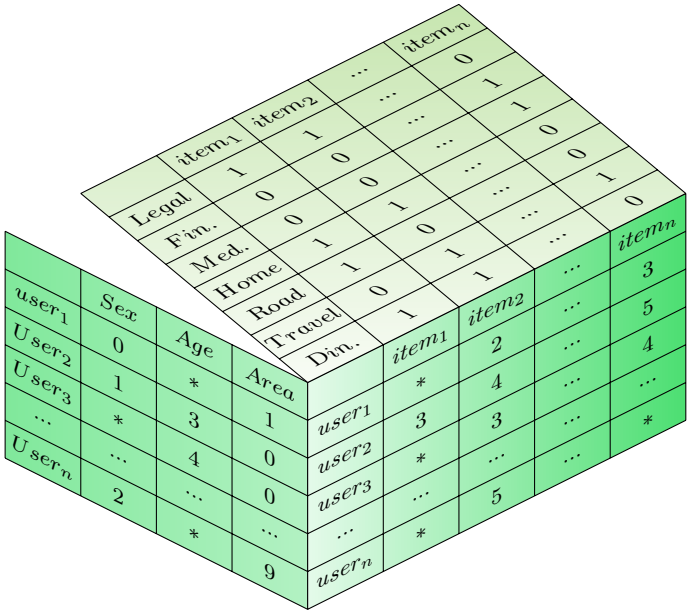


FIGURE 3.11: User-Item Combined Matrix

Item attributes in the item matrix of Figure 3.11 included the item category, item sub-categories, average ratings, item description and type. The user's preferred shopping area and the source of the request were also added to the item attributes.

Data Pre-processing

An overview discussion is provided in this section of the data pre-processing activities that were performed before the data could be used during the modeling phase. The combined feature matrix shown in Figure 3.11 revealed that while common feature attributes were given as either 1, if present, or 0, if not, service and price ratings ranged from $1 - 5$. The two ratings are on a different scale. Secondly, user ratings usually contained a lot of user bias which needed to be accommodated when making recommendations. Some users rated all items higher than other users simply because they had a good first-user experience which left a lasting impression while others who had a bad first-user experience needed more convincing to give all other items they had used a fair rating.The two rating problems we faced in this text as already explained were:

- *Shift of average ratings* - resulting from user bias like rating an item highly because you liked what you got previously or you like the supplier.

- *Different rating scales* - for attributes (zeros and ones) and for explicit ratings where we use a scale of one to five.

Moreover, data mining tools available can now handle complex text data that converting everything to numeric type was not necessary.

Subtractive Normalisation - The Gaussian Method

To initially solve the first problem; ratings of each user r_u were subtracted from the average rating of the user $\bar{r}_u$. To neutralise the second problem,the ratings of each user were divided by the variance in his ratings. The application of this solution provided us with the normalised rating $\hat{R}$ for item i by user u, $\hat{R}_u(i)$ is computed as:

$$\hat{R}_u(i) = \frac{R_u(i) - \bar{R}_u}{\Sigma_i (R_u(i) - \bar{R}_u)^2} \tag{3.2}$$

where $R_u(i)$ stands for the rating of item i by user u and $\bar{R}_u$ is the average rating for user u.

Although The Gaussian Normalisation Method is easier to understand and implement, it proved highly ineffective for our study due to the fact that most of our attributes used either one or zero. The mean was always most likely a one and subtracting one from one eliminated all the attributes. The row and/or column mean average values ($\bar{R}_u$ & $\bar{R}_i$) always produced a one or a near one figure.

Multiplicative Normalisation

To circumvent the problem highlighted in the above paragraph, this text resorted to the use of multiplicative normalisation technique to prepare the user and item feature matrices.The point of departure of the multiplicative technique was, given a feature matrix ,$F_{m\times n}$, normalise all matrix elements in such a way that the sum of each row or column adds up to 1 in a normalised feature matrix, $F_{normalised}$ [Spiegel *et al.*, 2009]. It relied on the lengths of the row, m and column, n of the F matrix. Figure 3.12 shows vectors $\vec{User}_3 = \{1, 1, 1\}$ and $\vec{Att}_3 = \{1, 1\}$. The

zeros, which represent inapplicable attributes, have been replaced by the $\{*\}$ in Figure 3.12 since we opted not to consider non-existent features in our computation.

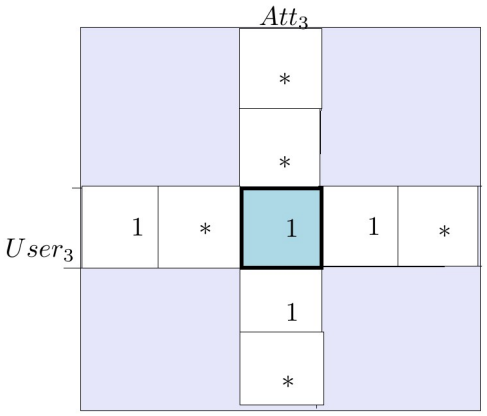


FIGURE 3.12: Simplified User-Feature Matrix

Normalising the User-Feature Matrix in Figure 3.12 was tantamount to normalising each of the row and column vectors. A *normal vector* (or unit vector) is a vector whose length is 1. Any vector whose length > 0 can be normalised by dividing each element in it by the length of the vector. The row vector ($User_3$) can be normalised by the formula $\vec{u} = \frac{\vec{User_3}}{|\vec{User_3}|}$ where $\vec{u}$ is the unit vector. The same applies to the column vector Att_3.

The user vector normalisation constant in Figure 3.12 is : $\vec{u} = \sqrt{1^2 + 1^2 + 1^2} = \sqrt{3}$ The row vector,$User_3$, then has to be normalised by the constant to produce: $\{(\frac{1}{\sqrt{3}}), (\frac{1}{\sqrt{3}}), (\frac{1}{\sqrt{3}})\}$. The column vector, Att_3, has two filled spaces so its normalisation constant is $\sqrt{2}$

This left us with a situation where row vectors were longer than column vectors. Which is when the study decided to borrow from the study by Spiegel and others in [Spiegel *et al.*, 2009] who suggested a multiplicative normalisation formula which accommodated the magnitudes of both the row and column vectors ($|\vec{User_3}|$ and $|\vec{Att_3}|$ and the position value. Their formula, for a Feature Matrix ,$F_m \times n$, to get a normalised feature matrix, $FN_m \times n$:

$$FN_{(m \times n)} = ROW_{m \times m}^{-\frac{1}{2}} \dot{F}_{m \times n} \dot{COL}_{n \times n}^{-\frac{1}{2}} \tag{3.3}$$

FIGURE 3.13: Multiplicative Normalisation Formula

where $ROW_{x \times x}^{-\frac{1}{2}} = \frac{1}{\sqrt{\Sigma_n F_{x,n}}}$ and $COL_{y \times y}^{-\frac{1}{2}} = \frac{1}{\sqrt{\Sigma_m, F_{m,y}}}$

Given equation 3.3, matrix position $Col3, Row3$ of Figure 3.12, the shaded matrix position, is calculated as $\frac{1}{\sqrt{3}} \times \frac{1}{1} \times \frac{1}{\sqrt{2}} = \frac{\sqrt{6}}{6}$

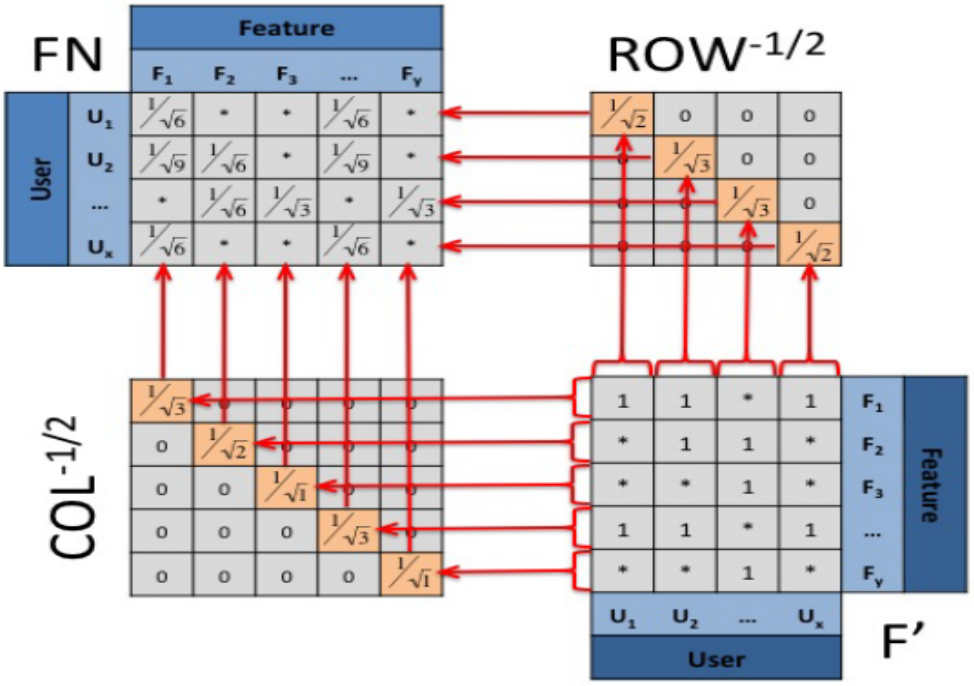


FIGURE 3.14: Multiplicative Normalisation of User-Feature Matrix
[Spiegel *et al.*, 2009]

The point of departure of the multiplicative normalisation technique is to normalise all matrix elements such that the sum of the elements in each row or column adds up to 1.

To best illustrate what has been explained in the above paragraphs, the original feature matrix in Figure 3.14 is transposed (F') and placed in the lower right corner. Non-existent features are already represented by (*) to show that they were not taken into consideration. The multipliers for the single feature matrices ROW and COL matrices are placed in the diagonals ($COL^{-\frac{1}{2}}$ and $ROW^{-\frac{1}{2}}$).

Matrix Factorisation

The literature review section in subsection 2.7.1.1 has described the role of Matrix Factorisation in literature with clear illustrations. This subsection explains the role Matrix Factorisation played in optimizing the results of this study, especially SVD. As already explained, matrix factorisation strategies are used to reduce the dimensionality of the item and user feature space and to extract latent feature relations between items and users of the observed dataset [Deerwester *et al.*, 1990; Spiegel *et al.*, 2009]

As a follow up to the example in Figure 3.12, SVD was applied on the normalised user-item rating matrix $Fnorm_{m\times n}$, which was about 90000 users by 1150 items. The equation pictures this process: $Fnorm_{m\times n} = U \bullet S \dot{\bullet} V^T$ where U, S, V^T were matrices of size $m \times m, m \times n, n \times n$, respectively. And, where U and V were orthonormal while S was diagonal. The column vectors of U are taken from the orthonormal eigenvectors of $F_{norm}F_{norm}^T$, and ordered right to left from largest corresponding eigenvalue to the least. The dimension reduction step of SVD kept only k diagonal entries from the matrix S to obtain a $k \times k$ matrix, S_k. Similarly, matrices U_k and V_k of size $m \times k$ and $k \times n$ were generated. The user-item rating matrix with reduced dimensions, F_{red}, is expressed by $F_{red} = Uk \bullet S_k \bullet V_k^T$. Position $ff_{i,j}$, in the reduced matrix F_{red}, denotes the rating by user u_i on item i_j. Several experiments were carried out to determine the optimal value of k during Modelling

3.6.4 Modeling Phase

Three different data mining techniques and six different algorithms were used to answer the research question as stated in Section 1.4. Most of the algorithms use either the SimpleKMeans algorithm explicitly or any similar implementation of the Euclidean Distance.

Modelling is the essence of data mining. In this phase the selected techniques and modelling algorithms were applied to the data and its parameters were calibrated [Llanes and Puldon, 2008]. In this section we identified algorithms which extracted user similarities based on user demographics, implicit and explicit ratings; item similarity based on both user ratings and item attributes.

The method used here mainly calculated the distance d between two items by measuring the relevance of these items. In the experiments in this text, we used both collaborative-based and content-based similarity for the item distance and for the user distance. The Item Model was made up of a collaboration of item ratings and profile attributes. The algorithm needed to create the item model computed item similarity using mostly attributes and user behaviour. As already noted, users rated items, suppliers and the price on a scale of 1 to 5.

Each item, i, on top of the ratings it received, it was also composed of n attributes a_n

$$i_k = \{a_1, a_2, ..., a_n\}$$

and the user, u_m is also a product of n profile attributes p_n:

$$u_m = \{p_1, p_2, ..., p_n\}$$

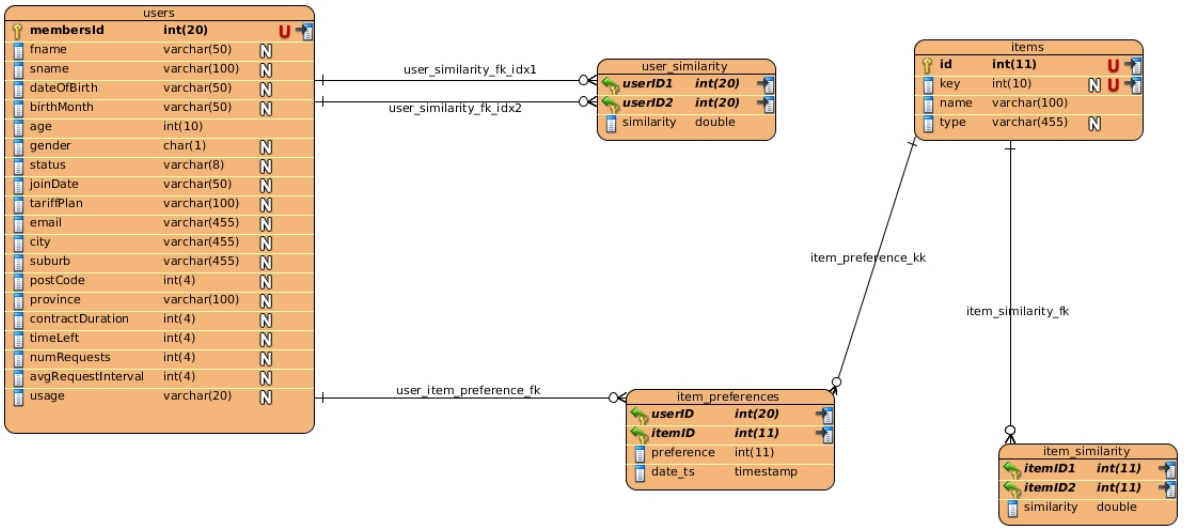


FIGURE 3.15: Data used by the recommendation application

.

Research Datasets

The data used by the recommender system is illustrated in Figure 3.15. The three main datasets are the users, items and rating data which is indicated as item_preferences in the diagram.

When similarity is calculated, similar users are persisted in table user_similarities and similar items in table item_similarities. These two tables act as a cache which alleviates the strain placed on the computing resources by the application running the various algorithms.

Attribute Selection

Attribute selection was performed to improve accuracy, reduce model training time and to control sprawling decision trees. Not all attributes are equal, yet some attributes refer to the same thing. In the customer profile dataset used in the study, age-range and age referred to the same attribute, while the 'contract_duration' attribute was 24 for every user thus rendering it redundant.

Similarly, the item data included a 'guest' account used to record transactions involving non-members who wanted to try out the service before deciding to register for it. Usage and rating values on this account were ridiculously high and quite misleading. Overfitting attributes are split by using decision tree algorithms during attribute selection. The task of identifying and removing redundant overfitting attributes was performed in the data preparation phase 3.6.3.

The AttributeSelectionClass.java, B.2, used to select attributes in this project, uses common WEKA attribute selection algorithms. These algorithms used the evaluation algorithms: *CfsSubsetEval, ClassifierSubsetEval* and *WrapperSubsetEval*. The following methods, available in the WEKA toolkit, were used to search the attribute space: *Exhaustive, BestFirst* and *GreedyStepWise*.

Recommendation and Data Mining Algorithms

Different recommendation and data mining algorithms were incorporated into the prototype hybrid recommender system that was designed and implemented as part of this research. Collaborative filtering and content-based recommendation algorithms were used. The different data mining techniques that were used included classification, clustering and association rule mining algorithms. The complete Java code for some of these of these algorithms can be found in the appendix section of this document. The algorithms selected for this study were built on item-item similarity which used both ratings and item attributes, and user-user similarity which is built on the items users have rated and on user profiles. The following is an example of the Cosine Similarity implementation in Java:

```
1  public class AdjustedCosineSimilarity {
2                  public static final int COSINE = 0;
3                  public static final int L2 = 1;
4                  public static final int L1 = 2;
5                  public static double calculateSimilarity(double[] vec1, double[]
       vec2, int type) {
6                          double similarity = 0;
7                          assert(vec1.length == vec2.length);
8                          for (int i = 0; i < vec1.length; i++) {
9                                  switch (type) {
10                                 case(COSINE):
11                                 ...
12                                 case(L2):
13                                 ...
14                                 case(L1):
15                                         similarity += Math.abs(vec1[i] - vec2[i]){
16                                 ...
17                 public static double calculateSimilarity(double[] vec1, double[]
       vec2) {
18                         ...
19                 }
```

```
20              public static int[] similarVectors(double[][] vectors, int index, int
    type) {
21                      ...
22              }
```

The complete code for this class can be found in Appendix B.

Memory-based Collaborative Filtering Algorithms

Two types of memory-based collaborative filtering algorithms were implemented, namely the user-based and item-based recommendation algorithms. In both the concept of a *neighbourhood* is used, in which the prediction of unknown ratings is based either on similar users or items [Bell and Koren, 2007]. In this research the Euclidean Distance measure was used to determine the similarity between users. This approach is also referred to as "k Nearest Neighbours" (kNN). It identifies pairs of items that are similarly rated or like-minded users with a similar rating or purchasing history, and it uses this information to predict ratings for unobserved user-item pairs [Bell and Koren, 2007; Spiegel *et al.*, 2009]. The following steps summarize the neighbourhood-based algorithms that were implemented as part of the hybrid recommender system developed in this research:

1. Identify the class attribute.
2. Normalise the data.
3. Address the problem of missing data.
4. Select k users who are similar to the active users (neighbourhood selection).
5. Determine the weighted sum of the selected users' ratings.
6. validation and prediction

The item-item and user-user components were combined and incorporated into the algorithm to enhance its predictive capabilities. Assuming that there are sufficient ratings from a user u, the algorithm used the unbiased estimator as given by equation 3.4 below:

$$\hat{\theta}_u = \frac{\Sigma_i r_{ui} \bar{x}_{ui}}{\Sigma_i \bar{x}_{ui}^2} \tag{3.4}$$

where each summation is performed over all the items rated by u. Given the sparsity of the data, $\hat{\theta}_u$ was adjusted to avoid over-fitting.

- **User-based (user-user) Similarity Algorithm**

  User-user similarity was computed in a similar way to item-item similarity using *The Adjusted Cosine Similarity algorithm.* For a list of m users:

  $$U = \{u_1, u_2, ..., u_m\}$$

  and a list of n items

  $$L = \{i_1, i_2, ..., i_n\},$$

  each user u_i has a list of items

  $$I_{ui}$$

  which he has previously used and rated. And, each user, u_i was defined as a product of his attributes:

  $$u = \{p_1, p_2, ..., p_n\}.$$

  The process calculated the Vector similarity of the two users, predicted their ratings and calculated the Euclidean Distance between the two users. It combined both CBF in the form of the user profile and CF in the form of item ratings.

  The *user-based* algorithm isdefined as:

```
1        For each item in Items, $I_i$\\
2        For each customer \emph{U} who purchased $I_n$ \\
3        For each item $I_i$ purchased by customer U \\
4        Record that a customer purchased $I_i$ and $I_n$ \\
5        For each item In \\
6        Compute the similarity between Ii and $I_n$
7
```

  This algorithm is based on Pazzani's work Pazzani [1999], and was developed by Sarwar *et al.* Sarwar *et al.* [2002], [Sarwar *et al.*, 2001], [Sarwar *et al.*, 2000] and also by Vozalis and Margaritis [Vozalis and Margaritis, 2007], [Vozalis and Margaritis, 2006], [Vozalis and Margaritis, 2004]. The algorithm executes the following steps:

  a. Construct the demographic matrix.

  b. Preprocess the data: normalise the data and eliminate gaps.

  d. Execute the Adjusted Cosine Similarity or other neighbourhood formation algorithm.

e. Use vector similarity to calculate demographic correlations.

f. Apply the Enhanced Correlation equation 3.7 on the ratings-based correlations and on the demographic correlations.

g. Execute algorithm the algorithm described in 4.12 to generate a rating prediction.

- **Item-based (item-item) Similarity Algorithm**

The selected adjusted Cosine Correlation Coefficient algorithm performed the following steps to calculate item-item similarity:

1. Retrieve from the dataset all items rated by user u.
2. Compute the similarity of each retrieved item to item j and keep a set of k most similar items:

$$k = \{s_1, s_2, ..., s_k\}$$

The similarity, $sim(i, j)$, between two items i and j was computed by first isolating the users who had rated these items and then applying the adjusted Cosine similarity algorithm. The pseudocode for this algorithm is given by algorithm 4:

Procedure 4 Item-item Similarity

procedure ITEM SIMILARITY ALGORITHM
R = rated items;
C = customers who rated items in R
For each $i \in R$
For each $c \in C$
For each item i_k purchased by c
Compute Similarity

Many algorithms can be used to compute the similarity between two items, but in this text we to use the adjusted cosine measure described earlier, in which each vector corresponds to an item's attributes rather than a customer, and the vector's dimensions correspond to customers who have purchased that item.

Model-based Collaborative Filtering Algorithms

Memory-based algorithms have several advantages, as already indicated in section 2.4.2, such as ease of implementation, scalability, use of item ratings without needing information about item content, and the ability to incorporate new data. However, there are disadvantages too, such as their dependence on user ratings, their inability to work effectively when presented with sparse data, the associated time and memory problems that occur when the number of users and ratings increase and finally, the *cold start* problem Model-based algorithms are used to avoid these problems. In a model-based collaborative filtering environment, a predictive model of user ratings is developed and used to generate item recommendations. There is a wide variety of model-based collaborative filtering algorithms. These include singular value decomposition, principal component analysis and different data mining techniques such as classification, clustering and association rule mining algorithms. For convenience, models developed using data mining techniques will be discussed separately.

- **Singular Value Decomposition (SVD) Model**

  SVD is one of the several model-based algorithms that are used for collaborative filtering. The algorithm determines user and item features, which it uses to predict whether a user u will like a particular item i. In this research, the SVD algorithm was applied to a user-item-rating matrix similar to the matrix shown in Table 3.8. The algorithm is well-suited for operation on numeric data. The version used in this research is based on the work of different authors, such as Sarwar and others Sarwar *et al.* [2002], [Sarwar *et al.*, 2001], [Sarwar *et al.*, 2000] and Vozalis and Margaritis [Vozalis and Margaritis, 2007], [Vozalis and Margaritis, 2006], [Vozalis and Margaritis, 2004]. The algorithm was implemented as follows:

  a. Construct *user-item* rating vectors for m users and n items. That is, for each user $u \in U$, where $U = \{u_1, u_2, ..., u_m\}$ and for each item $i \in I$, where $I = \{i_1, i_2, ..., i_n\}$, construct feature vectors like $u_1 = \{p_1, p_2, ..., p_k\}$ and $i_1 = \{a_1, a_2, ..., a_k\}$, where p_k and a_k are user profile features and item attributes respectively. At the end of this step a *user-feature* matrix (F) was constructed, similar to the matrix shown in *Figure 3.12*.

  b. Using the procedure outlined in Section 3.6.3, pre-process the matrix F to produce the normalised feature matrix F_{norm}, which does not contain any empty slots.

  c. Compute the SVD of the normalised feature matrix to get

  $$F_{norm} = U_F.S_F.V_F^T$$

  Through dimension reduction only the k biggest singular values of S_F were kept, resulting in a reduced matrix F_k, such that.

$$F_k = U_k.S_k.V_k^T$$

F_k contained the n reduced demographic vectors, each consisting of k pseudo-attributes. The matrix dot product, $\sqrt{S_k} \bullet V^T$ represents n items with the help of those k-*rank* pseudo-attributes, in the k dimensional space [Vozalis and Margaritis, 2006]. Position $r_{i,j}$, in matrix F_k, denotes the rating of user u_i on item i_j. The value $\sqrt{S_k} \bullet V^T$ shows the feature meta-ratings, $mr_{i,j}$, given by the k pseudo-users on the n items.

d. Using the Adjusted Cosine Similarity algorithm (Equation 3.5), determine the similarity between the active item, i_j, and a random item, i_f. The Adjusted Cosine Similarity algorithm was implemented using the code shown in Appendix B.3.

$$sim_{(j,f)} = adj\theta_{(j,f)} = \frac{\Sigma_{i=1}^{k} mr_{i,j}.mr_{i,f}}{(\Sigma_{i=1}^{k} mr_{i,j})^2 (\Sigma_{i=1}^{k} mr_{i,f})^2} \tag{3.5}$$

where $mr_{i,j}$ represents the meta rating on the j^th item j by the i^th user i.

e. Determine the Cosine Correlation between the active item, i_a, and each of the members (i_j) of its neighbourhood, by computing their corresponding vector similarities as follows:

$$sim(\vec{i_a}, \vec{i_j}) = \frac{\vec{i_a} * \vec{i_j}}{\left|\vec{i_a}\right| * \left|\vec{i_j}\right|} \tag{3.6}$$

f. Calculate the Enhanced Correlation, $EC_{a,j}$ for every pair of the form (i_a, i_j), where i_a is the active item and i_j is a member of its neighbourhood. The $EC_{a,j}$ is expressed as a linear regression function, as follows:

$$EC_{a,j} = \alpha \times rc_{a,j} + \beta \times dc_{a,j} + \gamma \times (rc_{a,j} \times dc_{a,j}) \tag{3.7}$$

where, $rc_{a,j}$ and $dc_{a,j}$ are the ratings-based and the demographic correlation between active item i_a and neighbourhood member i_j, while α, β, γ are flags or weights representing the participation of each of the three components. Since item recommendation has to be based more on item similarity than user ratings, feature resemblance should have dominated, but in practice, user ratings dominated.

g. Using Equation 4.12, predict the ratings of user u_q on the active item i_a.

$$Pred_{q,a} = \frac{\Sigma_{k=1}^{h} EC_{a,k} \times (rr_{q,a} + \bar{r}_q)}{\Sigma_{k=1}^{h} |EC_{a,k}|} \tag{3.8}$$

where, $rr_{q,a}$ are the ratings of user q from the reduced user-item matrix F_k, and $\bar{r}_q$, the average ratings subtracted from F during preprocessing have to be added back.

Model-based Data Mining Algorithms for Recommender Systems

Different data mining algorithms were used to support the recommendation task of the prototype hybrid recommender system that was developed as part of this research. Different types of data mining techniques were used to perform, within the context of recommendation, such tasks as classification, prediction, clustering and association rule mining.

- **Classification Models**

  Different classification techniques were used in our hybrid recommender system. These included decision tree and Bayesian Network Classifiers. In this research classifiers were used, amongst others, to determine the status of the PLP Group's customers and to use this information to understand the behaviour of these customers.

  - **Decision Tree**

    Similarly, the J48 and J48graft algorithms together with the regression classifier were used as decision-based algorithms focused on predicting user retention, customer churn, and user ratings.

  - **Bayesian Network Classifier Model**

    The Bayesian Network Classifier algorithm is a classification algorithm. The Bayesian Network Classifier algorithm was used in this research to build a probability model by combining observed and recorded evidence with the real-world knowledge to establish the likelihood of occurrences. The WEKA toolkit includes the Tree Augmented Naïve Bayes (TAN) and Markov Blanket networks amongst its set of classification algorithms. Using these algorithms we were able to identify the relationships between certain customer attributes and customer retention. This information made it possible to know which user has cancelled a particular service or is maintaining their subscription to it. The Bayesian Network algorithm was chosen for two reasons: it is efficient in terms of avoiding overfitting and relationships are represented clearly.

- **Cluster Models**

Clustering techniques were used, among other things, to identify groupings and clusters in the PLP Group subscribers dataset of people with similar churn, usage and rating characteristics. In this study clustering was performed as an intermediate step before similarity, association and prediction techniques were performed. The *SimpleKMeans* algorithm is one of several clustering algorithms implemented in the WEKA toolkit. It was used in this study to group items and users into k clusters, with k representing the number of arbitrary clusters. Figure 3.16 illustrates WEKA's execution of the *SimpleKMeans* algorithm, in which a total of 24583 PLP Group customer records were used, of which 70% were used for training, and the remainder for testing. Nine customer attributes were involved in the experiment.

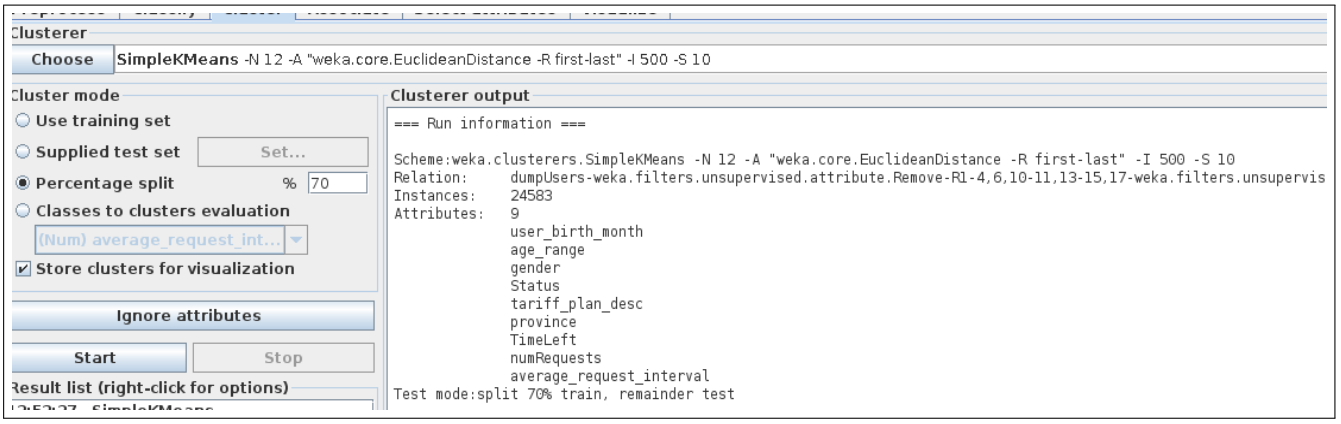


FIGURE 3.16: The SimpleKMeans in WEKA

A top-level description of the steps performed by this algorithm is as follows:

* Choose the number of clusters.In the image above 12 clusters were chosen
* Determine the initial partition and the initial mean vectors for each cluster.
* Adjust number of clusters or iterations until no relocations are possible

The K-Means algorithm process executes as shown in Figure 3.17.

The minimum distance to the centroid was used as the criterion for clustering. The algorithm was applied on user and item datasets. To improve clustering several iterations of the algorithm were executed. The K-Means algorithm was chosen because of its efficiency and economy when cosidering computing resources. It was used to establish whether there were any identifiable user clusters in the PLP data. The dataset used contained about 25000 records and 70% of them were used for training with the rest used for testing.

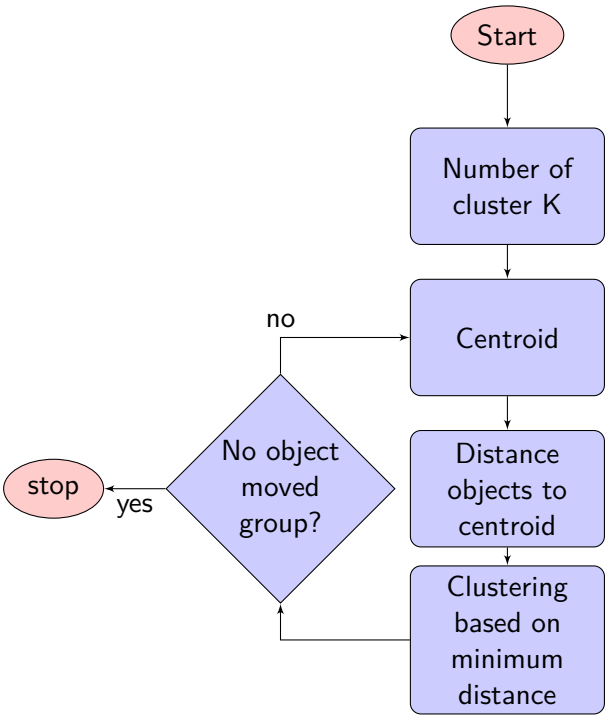


FIGURE 3.17: The K-Means Algorithm

This K-means was used to cluster customers into tree major profitable groups of customer: high user, medium user, and low user. Once these lucrative groupings have been identified, other techniques like association mining and classification were applied on each of the clusters. In this study, the K-means algorithm is applied on the data from existing customers, which includes variables from customer profiles and usage data tables. About 25000 the existing customer's data was used to build the customer behaviour analysis cluster model. The utilization of K-means for customer clustering set $k = 3$ as it was meant to cluster customers into 3 usage groups.

The Euclidean distance framework used in this algorithm used the following equations:

- *vector-to-vector* distance. For two user vectors, x and y each of them with H attributes, their Euclidean Distance was calculated as shown in equation (3.9)

$$d(y,x) = \sqrt{\frac{1}{H}\sum_{h=1}^{H}(y_h - x_h)^2} \tag{3.9}$$

- *vector-to-set* distance was calculated by using the distances between the vector y and each of the N members of the set U:

$$d(y,U) = \sqrt{\frac{1}{N}\sum_{x\in X} d^2(y,x)} \tag{3.10}$$

Table 3.15 below shows the input variables for each user vector in the clustering algorithm.

Feature Name	Description
AgeRange	1,<25; 2, 27-50; 3, 50+
BirthMonth	1, January;...;12,December
Gender	1, Male; 2, Female
Status	1, Live; 2, Cancelled; 3, Suspended
Province	1, Eastern Cape; ...; 9, Western Cape;
NumberOfRequests	>10
AverageRequestInterval (in days)	0 - 693

TABLE 3.15: Variables used in Clustering

The *k-means algorithm* is the most commonly used model for clustering large data sets. However, its biggest limitation is that it is only applicable to numeric data values. This makes it irrelevant in the real world where data is mainly categorical. The GetMore *user-feature* dataset described in Table 3.15 above included a combination of both numeric and categorical data elements. The pure *k-means* algorithm is not meaningfully applicable to nominal data due to several reasons. Categorical data is discrete and does not have an innate origin. A Euclidean distance of such data is not meaningful. While it is possible to discretize continuous data to make it nominal to avoid ordering it, numerically coding categorical data does not have a reverse effect. Numerically coded categorical data cannot be naturally ordered. Therefore, extensions and variations of the *k-means* algorithm, [Gniazdowski and Grabowski, 2016; Huang, 1998], were considered for this research.

The approach used in this study involved numerically coding nominal data [Gniazdowski and Grabowski, 2016]. Nominal variables that had only two possible values like *gender* were encoded by using the *"1-of-N-1"* encoding. *M* was represented by "1" while *F* was replaced by a "0" in the encoding process. All other attributes which had 3 or more possible values like *status* and *birthMonth* were coded using the *1-of-N* encoding.

The nominal data is first sorted in order of type before cardinality or the frequency of each value is used to compute its rank based on its ordinal position in the sorted list as shown in table 3.16 below:

Jan			Feb			June			Sep		
Position	**Value**	**Rank**	**Position**	**Value**	**Rank**	**Position**	**Value**	**Rank**	**Position**	**Value**	**Rank**
1	Jan	2	1	Feb	2	1	June	1	1	Sep	3
2	Jan	2	2	Feb	2				2	Sep	3
3	Jan	2	3	Feb	2				3	Sep	3
									4	Sep	3
									5	Sep	3

TABLE 3.16: Numerically Coding User Birth Months

Rank R presented in Table 3.16 is calculated by the formula in equation 3.11 below :

$$R = \frac{n+1}{2} \tag{3.11}$$

However, as can be seen from Table 3.16 above, values of equal frequencies tend to have the same rank (R). Both January and February have a rank of 2. To mitigate this anomaly, the above method needs to be modified using Equation 3.12 below:

$$R_j = R \cdot \sqrt[k]{-1} = R \cdot e^{i\theta} = R \cdot (\cos\theta + i\sin\theta) \tag{3.12}$$

R is rank as calculated by Equation 3.11 above, while j is the j^{th} subset *(j=0,1...,k-1)*. Similarly, $i = \sqrt{-1}$ and $\theta = \frac{2\pi J}{k}$. Birth months, in the Getmore data, were ranked as follows:

Month	***n***	***R***
January	6442	3222
February	6001	3001
March	6496	3448
April	6174	3087
May	6250	3125
June	6407	3204
July	6078	3039
August	6211	3106
September	6649	3325
October	6052	3026
November	5701	2851
December	6324	3162

TABLE 3.17: Ranking Months Using Equation 3.11

Applied to a sample Getmore dataset shown in Table 3.18 below.

Age	BirthMonth	Gender	Province	No. of Requests	Avg Request Interval	Status
30	January	1	Gauteng	14	23	live
30	February	2	Gauteng	32	12	live
34	February	1	Western Cape	12	32	canc
34	March	2	Western Cape	11	14	canc
34	March	1	Gauteng	64	7	live
35	March	2	Western Cape	9	79	canc
35	March	1	Gauteng	6	54	canc
35	April	2	Western Cape	13	36	live
35	April	1	Gauteng	23	9	susp

TABLE 3.18: Sample Getmore Data For Clustering

Some of the attributes like *Age, No.of Requests*, and *Avg. Request Interval* were already numbers so they did not need any coding. *Gender*, because it had only two possible values, was encoded using the "1-of-N-1" encoding with *male* users represented by a "1" and *female* users represented by a "2". The rest of the attributes, however, needed complex ranking as using Equation 3.12. Table 3.19 below shows the same sample as in Table 3.18 above after it had been numerically coded.

Age	BirthMonth	Gender	Province	No. of Requests	Avg Request Interval	Status
30	2	1	3.5	14	23	live
30	-2	2	3.5	32	12	live
34	-2	1	2.5	12	32	canc
34	2.5	2	2.5	11	14	canc
34	2.5	1	3.5	64	7	live
35	2.5	2	2.5	9	79	canc
35	2.5	1	3.5	6	54	canc
35	2	2	2.5	13	36	live
35	2	1	3.5	23	9	susp

TABLE 3.19: Coded Getmore Data For Clustering

The dataset was split in the ratio of 70-30 for training and testing in this experiment. The results showed that the user's birth month had little influence on the cluster determination of users, while age and gender had a very high influence. Old males were the most lucrative customers while young men were more likely to cancel the product early and had very few, far apart transactions.

- **Association Rule Mining (ARM)**

  Association rule-based recommender systems are deployed in a number of areas. In recommender systems, association rules are used to express relationships between items. The algorithm used in this study searches and selects the top k best rules, obtained by increasing the support threshold during the search [Sharma *et al.*, 2012]. The algorithm combines the support and confidence metrics into a single measure called predictive accuracy [Sharma *et al.*, 2012], which is used to generate Apriori association rules. The methodology used in this category recommends items based on usage data as well as item ratings data and customer-related profile data.

  As already explained in section 4.7 above, given lists of requests where each list is a set of items (itemset) requested by a single user, an association rule implies the form $X \Longleftrightarrow Y$ where X and Y are itemsets. The *support* for the association rule $X \Longleftrightarrow Y$ is the percentage of request lists that contain both itemsets X and Y in their requests. The *confidence* for the rule $X \Longleftrightarrow Y$ is the percentage of lists of requests that contain itemset Y among the lists that contain Y. While the *support* tells us how useful as a prediction indicator a rule is, the *confidence* gives us a degree of certainty. In this study Association Rule Mining was used for customer segmentation. The association rule mining algorithm used in this study was the *PredictiveApriori Algorithm* [Tobias, 2001]. This algorithm searches and selects the top 'k' best rules, obtained by increasing the support threshold during the search [Sharma *et al.*, 2012]. The algorithm combines the support and confidence metrics into a single measure called predictive accuracy [Karthikeyan and Thangaraju, 2014; Sharma *et al.*, 2012], which is used to generate Apriori association rules. The predictive accuracy metric is coded as follows:

Procedure 5 Predictive Accuracy algorithm

procedure DETERMINING ACCURACY
  Let D be a data file with **n** number of items
  if [x $\rightarrow$ **y] then**
    the predictive accuracy of **[x** $\rightarrow$ **y]**
    $c($**[x** $\rightarrow$ **y]**$)$ = **P[n]**
    satisfies **y**|**n** satisfies **x**
    where distribution of **r** is governed by the static process **P**
    Predictive Accuracy is the conditional probability of
    x $\rightarrow n$ and $y \rightarrow n$

[Karthikeyan and Thangaraju, 2014; Sharma *et al.*, 2012]

In this study association rule mining was used to create user profile after user clustering; computing item similarity and finding the frequent itemset.

In the previous section we used the K-means clustering algorithm to group customers into clusters with shared usage characteristics. In this section Association Rule Mining was used to identify rules that were used to create customer profiles in each cluster. After the clusters and the associated statistical summarized data had been generated by the K-means algorithm, the users were sub-divided into three major usage groups. The association rule mining algorithm is then used to create and validate the user profiles. The variables selected by the attribute selection class in section B.2 were used for both clustering and Association Rule Mining. In the clustering results shown in figure **??** above, cluster 2 has the most profitable users with over 30 requests each and mainly still active users. We used users from this cluster to extract association rules and the top 5 rules are given below in table 3.20:

RuleID	AssociationRule	Support	Confidence
1	Usage=High← Age =54;Sex=1;Province=3;	23.2%	89%
2	Usage=High← Age =61; Sex=male	12.4%	87%
3	Usage=High← Sex=Male; Age=75;	9.8%	93%
4	Usage=High← Age=68, Avg-Request-Interval=60;	7.2%	84%
5	Usage=High←,Age =59; Sex=Male; Avg-Request-Interval=120	6.6%	89%
...	...	...	...

TABLE 3.20: Customer Profile Rules

Algorithm parameters were set so as to identify association rules that had at least 80% confidence and 5% support imposed on the PredictiveApriori association rule miner. The cluster profile of cluster 2 is given above in the form of association rules, where each rule represents a user profile that was dominant or most strongly associated with the users in that cluster. For segmentation purposes, we grouped users with shared behavioural characteristics. From this clustering, marketers can create more accurate campaigns towards each target section of customers for cross-selling and up-selling and promoting consumption [Agrawal and Srikant, 1994; Farajian and Mohammadi, 2010].

In this study Association Rule Mining was also used with adjusted cosine similarity to compute item similarity. Association rules capture the relationships among items based on their patterns of co-occurrence across different transactions. In this section, we have a set of n unique items represented by the item vector: $I = \{i_1, i_2, i_3, \ldots, i_n\}$ and a set of m user transactions represented by the vector: $T = \{t_1, t_2, t_3, \ldots, t_m\}$, where each $t_i \in T$ is a non-empty subset

of I. Given that that a user transaction $t_i \in T$, is not empty, we can represent it as a vector: $\vec{t} = \{i_1^t, i_2^t, \ldots, i_n^t\}$ where, $i_n^t = \begin{cases} 1, \text{if} \quad item_i \in t \\ 0, \text{otherwise} \end{cases}$

To cluster transactions we used the cosine similarity distance measure. With any two transactions, t_1 and t_2, we defined similarity $sim(t_1, t_2)$ as the normalised cosine of the angle between the two transaction vectors. We expressed $sim(t_1, t_2)$ in terms of the size of the respective vectors as follows:

$sim(t_1, t_2) = \frac{|t_1 \cap t_2|}{\sqrt{|t_1|.|t_2|}}$

Transaction clustering of this sort does not scale well with large datasets because, unlike K-Means, it does not prescribe the number of clusters to generate. It results in a set $C = \{c_1, c_2, c_3, \ldots, c_k\}$ where each $c_i \subset T$, a set of user transactions.

In this study, we also used Association Rule Mining to make recommendations using the frequent itemsets. The algorithm for using frequent itemsets to make recommendations is outlined the algorithm 6 below:

Procedure 6 Generating Recommendations Using Frequent Itemsets

procedure RECOMMENDATION USING FREQUENT ITEMSETS

Input: active transaction t_a; fixed transaction size s

Minimum support threshold σ

Minimum confidence α

Recommend $\Longleftarrow \emptyset$

for **do**each frequent itemset I of size $|t_a| + 1$ such that $t_a \subset I$ *do*

if s **then**upport(I) $\geq \sigma$ **then**

let c = confidence($t_a \Rightarrow \{u\}$)

if **then**$c \geq \alpha$**then**

u.rec_score $\Longleftarrow c * (u, t_a)$

$Recommend \Longleftarrow Recommend \cup \{u\}$

Content-based Item Filtering

As was explained in Section 2.7.2 in Chapter 2, the CBF approach used in this study focused on the content features of both users and items. Item content features were compared using the utility algorithm as embodied in equation 2.3, and using a *user-item* matrix where users who rated an item are treated as its attributes. A CBF system recommends items based on the correlation between the content features of the items and the user's preferences.

- *Item Attributes Utility Approach* In the utility approach, each item belongs to one or more item types. The item types, as shown in figure 2.3, are organised as lists which can easily be treated as vectors. Each string value was assigned a numeric value for ease of computation. Values used with the utility function were obtained from table 2.2. The values was re-written with the *item type* attributes spread out in a matrix as shown in table 3.21:

item_id	item_category	marketing	information	car hire	financial services	legal services	...
3253	20	0	0	0	0	0	...
3304	20	0	1	0	0	0	...
3805	22	0	0	1	0	0	...
4801	21	1	1	0	1	0	...
5068	22	0	1	0	0	1	...
...	...	...	...	...	...	...	...

TABLE 3.21: Item Content Matrix

The matrix was normalised using the formula discussed in section 3.6.3. For the purposes of normalisation, zeros were ignored because they were simply boolean values which meant the item did not belong to that group of items. With zeros ignored, the matrix would look like this:

item_id	item_category	marketing	information	car hire	financial services	legal services	...
3253	20	*	*	*	*	*	...
3304	20	*	1	*	*	*	...
3805	22	*	*	1	*	*	...
4801	21	1	1	*	1	*	...
5068	22	*	1	*	*	1	...
...	...	...	...	...	...	...	...

TABLE 3.22: Item Content Matrix with Zeros Removed

The normalisation process excluded the item_category column because that was used as a weight when recommendations were made. Items belonging to the same category had a higher recommendation weighting than items belonging to different recommendations. The vector for item *4801* is $\vec{4801} = \{1, 1, 1\}$ while the vector for attribute *information* is $\vec{information} = \{1, 1, 1\}$ After normalisation, the rows and columns of table 3.22 added up to 1. Except for the category column.

The Cosine Similarity measure was then used to calculate the utility of any two item normalised vectors. The result of this algorithm was used to recommend totally new items that had not yet been rated by anyone.

- *Item-Rating Data Similarity* Item similarity using rating information is only interested in whether the item in question was used by similar users or not. Figure 3.18 below shows how the rating information is kept in the database. The picture shows a sample of users who rated item *3*.

```
mysql> select * from item_preferences order by itemID limit 10;
+----------+--------+------------+---------------------+
| userID   | itemID | preference | date_ts             |
+----------+--------+------------+---------------------+
| 16213338 |      3 |          2 | 2014-02-03 00:00:00 |
| 16294579 |      3 |          2 | 2014-02-25 00:00:00 |
| 16315845 |      3 |          2 | 2014-02-12 00:00:00 |
| 16503808 |      3 |          2 | 2014-01-16 00:00:00 |
| 16581390 |      3 |          2 | 2014-03-04 00:00:00 |
| 16631027 |      3 |          2 | 2014-03-05 00:00:00 |
| 16639335 |      3 |          2 | 2014-02-14 00:00:00 |
| 16667589 |      3 |          2 | 2014-03-05 00:00:00 |
| 16718734 |      3 |          2 | 2014-01-20 00:00:00 |
| 16760167 |      3 |          2 | 2014-03-12 00:00:00 |
+----------+--------+------------+---------------------+
10 rows in set (0,92 sec)
```

FIGURE 3.18: Users who rated the same item

For the purpose of computing item similarity, users who used an item were presented as the item's attributes. In the vector space model, an item I is represented as an m-dimensional vector, where each dimension corresponds to a distinct attribute and m is the total number of attributes all the items have as shown in table 3.21 above, but with users who used the item added as additional item attributes. The item vector is written as $w_i t_i$, where w_i is the weight of the attribute t_i that indicates its importance. If an item I does not contain the attribute t_i then the weight w_i is zero. The attribute Selection implementation code is given in Appendix B.2. In this CBF system, we used information retrieval-related technology, *tf*iIf* (term-frequency (tf) times inverse Item frequency(tIf)), which is the value of a variable associated with an attribute is a real number that represents its importance or relevance. The *tf * iIf* weight, *w(t,I)*, of an attribute t in item I is a function of the frequency of t in the item (tf_t, I), the number of items that share the attribute (If_t) and the number of items in the collection *(N)*. The weight of an attribute can be formally established by equation 3.13 below:

$$w(t,I) = \frac{tf_{t,I} log(\frac{N}{If_t})}{\sqrt{\sum_t (tf_{t,I})^2 log(\frac{N}{If_{t_i}})^2}} \tag{3.13}$$

Where I is the item, t is the item attribute, f the frequency of the attribute and N the total number of items as already explained. The weights are normalised by cosine normalisation so they range between 0 and 1 and for the vectors to be equal in length. The cosine similarity measure was then used to determine the similarity of items: $sim(I_i, I_j) = \frac{\sum_k w_{ki}.w_{kj}}{\sqrt{\sum_k w_{ki}}^2 . \sqrt{\sum_k w_{kj}}^2}$

Estimations of a user's interest score in a given item were deduced by computing the cosine similarity of the active item with other items that the user has liked.

3.7 Adaptability of the Prototype Hybrid Recommender System

The model and prototype build in this study used data collected from a value-added service provided by a telecommunications company in South Africa to try to retain and attract new customers. Many organisations in various industries also use a similar mix of value-added services to create a fulfilling relationship with their customers.

3.7.1 Examples of Value-Added Services in other Industries

Value-added services are now used in banking, retail, health and beauty, and so on. Most major banks encourage their clients to purchase at designated shops in order to get more from their transactions. First National Bank (FNB) has the *eBucks* programme which gives customers a certain percentage of their expenses back. These eBucks can be used to purchase goods and services from various service providers. ABSA (ABSA Rewards) and Standard Bank (uCount Rewards) offer a similar service to their respective customers.

In the retail industry, grocers like Pick n Pay, Woolworth and Spar give their customers loyalty cards which enable them to earn loyalty points. However, the main benefit of this study to the retail industry is the fact that it can help these organisations to recommend goods and services they offer to both new and existing customers. Most organisation already use Association-rule Mining to arrange goods in their supermarkets, but few analyse customer content features deep enough to be able to recommend goods to new customers.

Health and wellness organisations like BankMed, Discovery and Clicks use a combination of both loyalty points and value-added services like *Vitality* and *Bounce* to retain customers.

3.7.2 Adapting the Prototype Hybrid Recommender System

The methodology used to develop this recommender system revolves around a universally applicable framework in the data mining space, CRISP-DM. For every context that this recommender system needs to be applied to, the process needs to proceed through the six stages of CRISP-DM. It is important to understand the business, the data understanding, preparation, modelling, evaluation and deployment. The recommender system built in this study attempted to address all the known limitations of recommender systems which are *new-user* and *new-item* problems. Where user content features are not available, usage data can be used and vice-versa. For new products, those that have never been rated by anyone before, content features are used. Products that satisfy a similar need or want are classified together and recommended as alternative products. For example, honey can be recommended as both a substitute of sugar or jam and as a complement of tea leaves.

3.8 Chapter Summary

This section focused on the research method which was cemented on the CRISP-DM standard. The section went through all the stages of the standard indicating how each step was applied in the research process. Most of the discussion focused on the data understanding, data preparation and modelling phases as that is where the gist of this text is. However, the rationale for the research had to be extracted from the business case which is given under *Section* 3.2.

Various modelling techniques and algorithms under the different data mining techniques were identified, discussed and applied to the dataset. Most recommender systems in literature combine collaborative and content-based filtering techniques to improve rating prediction in varying degrees. However, our approach is special in that we unify user-item ratings and content features in a single matrix which iare used by K-NN techniques to cluster users and items before computing similarity. Moreover, our research gives equal attention to the use of both user features and item attributes. To ameliorate the computational overload and memory usage of our system, we used matrix factorisation (SVD) for dimensionality reduction. Although the resulting low-dimensional matrices are just an estimation of the original matrix they are less sparse but accurate and they helped reveal hidden user-item relations.

The recommender system built as part of this research was described in section 3.5. We described its architecture and its process flow. The Chapter concluded by an analysis of the various challenges that faced this research. Finally, we looked at the various fields and areas where this system can be applied and how it could be applied.

The next Chapter will examine the results of the modelling phase and evaluate the various models using the metrics already discussed in this Chapter.

Chapter 4

Results

4.1 Introduction

Chapter 3 provided a detailed introduction and discussion of the CRISP-DM process model, the standard methodology that was used to guide this research through its different phases. Apart from a discussion of some of the reasons for conducting this research, the chapter also contains a design and a detailed description of the prototype hybrid recommender system that was developed and implemented as part of this research. The system had to be able to predict user preferences and improve user satisfaction with the GetMore programme which was hoped would increase sales, usage and user loyalty for the PLP Group and Cell C . This chapter presents the results of the experiments that were performed to answer the main research question, which was stated in Chapter 1 as follows:

Can an effective hybrid recommender system be designed and implemented that accurately predicts the rating of Cellphone Contract Customers and recommend specific services to individual PLP Group clients based on the preferences of similar clients?

In the following sections, we discuss the sub-questions that were posed to answer the main question. In each case, and to remind the reader, a sub-question is re-stated, followed by a description of the tasks that were performed in order to answer it. The results obtained from the experiments are then presented and analyzed. Section 4.2 discusses how the PLP Group customers' latent traits were used to determine the similarity between items (products/services) and between users (customers). Section 4.3 discusses the user-based collaborative filtering approach that was adopted to identify similar users. In Section 4.4 results are presented on the

traits and attributes shared by frequent PLP Group subscribers. In the product and service-based business environment, customers are the most important component. Companies will go to great lengths in order to avoid customer attrition. Section 4.5 focusses on this topic and looks at the customer information that could possibly be used to avoid contract cancellations by PLP Group customers. Section 4.2 looks at subscribers' latent traits and their possible use in rating the services provided by the PLP Group. Association rule mining techniques are increasingly being used in different types of recommender systems. In Section 4.7 we discuss their suitability in predicting subscriber preferences. Section 4.4.2.1 discusses the topic of item similarity from the content-based and collaborative filtering perspectives, and in Section 4.11 the chapter is concluded by summarising the main points.

4.2 Extracting Latent Traits

In Chapter 2 collaborative filtering was described as the most successful approach for building recommender systems. The technique is widely used in commercial recommender systems. Its main drawback is the naive assumption it makes about there being sufficient historical user or item data for estimating similarity between items or users. However, the sparse rating matrix in Figure 3.8, prepared using our own data, indicated that there was not enough user-item information. To deal with this weakness on the part of collaborative filtering algorithms, we decided to extract and use the latent features of users and items to identify hidden relationships between users and items.

Singular Value Decomposition (SVD), which described in detail in section 2.7.1.1, can be used for both dimensional reduction and for the extraction of latent features. The features were incorporated into the recommendation process in order to improve the quality of the recommendations and to ameliorate the data sparsity problem. The purpose of latent feature extraction models is to map users and items in a common latent space by representing user and item features in the same vector space.

The following question was posed to investigate latent traits as determinants of similarity:

> *What are the latent traits of the PLP Group's subscribers? How can these traits be used to calculate user similarity?*

Instead of predicting unknown ratings, latent feature models (lfm) help in computing missing feature information. Not all the features in our dataset were used in the experiments that were

performed in order to answer this question. A latent feature selection technique was used to identify and remove features that did not contribute any useful information in predicting the class variable. Two types of features were identified and eliminated: *irrelevant* and *redundant* features. The former are neither negatively nor positively correlated to the class attribute and the latter are a group of features which are highly correlated with one or more other features. Only good features were retained. These are features that correlate highly with the class variable, and which do not correlate with each other.

This idea was formalised by Ghiselli [1964], as follows:

$$Merit_s = \frac{k\bar{r_{cf}}}{\sqrt{k + k(k-1)\bar{r_{ff}}}} \tag{4.1}$$

where

$Merit_s$ is the "merit" of a feature subset S containing k features,

$\bar{r_{cf}}$ is the average correlation of the feature to the class attribute and

$\bar{r_{ff}}$ is the average correlation between any two features.

Figure 4.1 below reveals the correlation between items and user-age, and between items and user-provinces. The data used to produce the images was made up of about 25000 users and about 500 out of 1500 items.

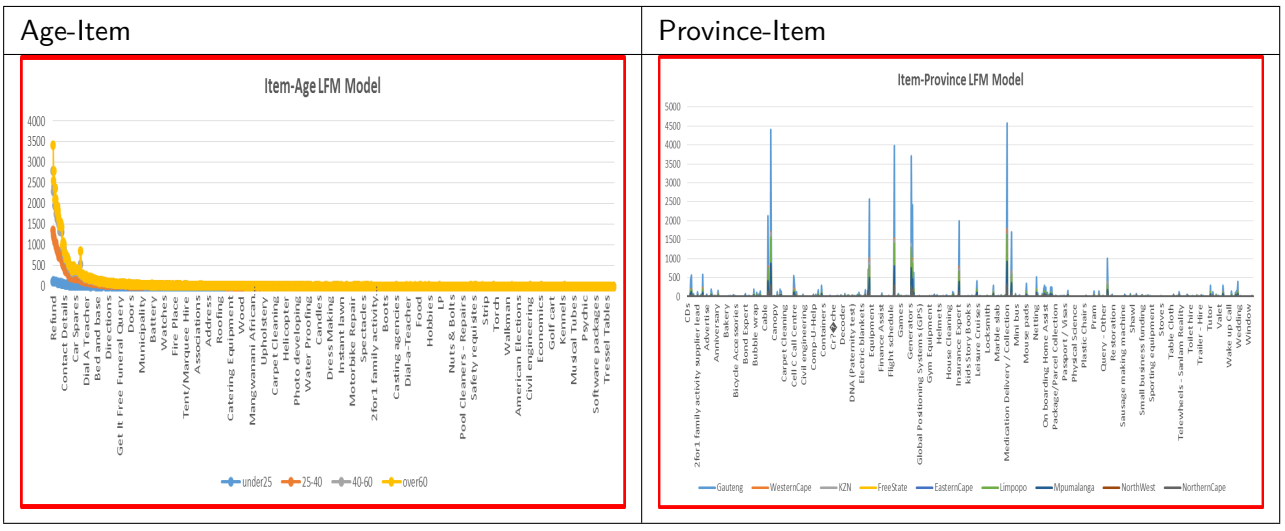


TABLE 4.1: Correlation Between User and Item Features

		features			
		f_{k1}	f_{k2}	...	f_{kp}
Users	u_1	0	1	...	0
	u_2	1	0	...	1
	u_3	1	0	...	1
	u_4	0	1	...	0
	...	...	...	...	...
	u_m	0	1	...	1

(A) User Feature Matrix (U)

		Item Attributes			
		a_{j1}	a_{j2}	...	a_{jq}
Items	i_1	0	1	...	0
	i_2	1	0	...	1
	i_3	1	0	...	0
	i_4	0	1	...	0
	...	...	...	...	...
	i_n	0	1	...	1

(B) Item Feature Matrix (I)

		Items			
		i_1	i_2	...	i_n
Users	u_1	0	1	...	0
	u_2	1	0	...	1
	u_3	1	0	...	0
	u_4	0	1	...	0
	...	...	...	...	...
	u_n	0	1	...	1

(C) Rating Matrix (R)

TABLE 4.2: User, Item and Rating Matrices

The images above use absolute values to depict a potential relationship between user age and certain items and between items and user provinces. From the image on the left we can deduce that all age-groups claim refunds from the service, a sign that they had been charged after they had cancelled their subscription. The 60+ age group dominates the dataset. Users under the age of 40 showed a high interest in the items like *dial-a-teacher, car spares, and so on* The graph on the right shows the dominance of Gauteng Province in the dataset. These results corroborate those from the age-item graph in the dominance of medical-related items needed mainly by the 60+ age-group. The type of items requested and displayed in the graphs also tell a lot about the class of people who can afford to use the service. The users are people with a stable and regular income who can afford medication, holidays and cars with canopies.

Although most conclusions drawn from Figure 4.1 above are common analysis knowledge, these results are based on the SVD approach to retrieving latent feature relations. This image proves that our measurement results and the real world knowledge are consistent and confirm the trustworthiness of the SVD approach.

Extracting Latent Features Using SVD

As already indicated above, SVD can be used for both dimensionality reduction and for the extraction of latent features. In this section, we explain how SVD was used to extract latent user-item features. Two matrices, U and I, were used to extract latent features. The matrix U represents all users and I is the matrix of all possible items. Table 4.2 below shows the 3 matrices combined to form a user-item matrix *(X)* used in SVD.

In X, user and item features are combined to form a big feature matrix:

$$X = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

SVD does not seek to predict missing ratings, but features. Items rated by a user form a part of the user features in this instance while users who rated an item become a part of the composition of the item-features. Performing SVD weighed heavily on the computing resources. To ameliorate this challenge we used a subset of 25000 from the dataset. The following Java Code shows the implementation of SVD:

```
1  public class SVD {
2     public static void main(String[] args) throws IOException {
3             double [][] data = new double [90000][1500];
4             String scan;
5             FileReader file = new
      FileReader("~/BitBucket/School/Data/Matrices/combinedMatrices/features.csv");
6             ...
7     Matrix C= new Matrix(data);
8    X.norm1();
9       Matrix B = C.getMatrix(0, 25000,2,500);
10     Matrix X = B.times(B.transpose());
11  // compute the singular values decomposition
12       System.out.println("X = U S V^T");
13       SingularValueDecomposition s = X.svd();
14       Matrix U = s.getU();
15       Matrix S = s.getS();
16       Matrix V = s.getV();
17       System.out.println("rank = " + s.rank());
18       ...
```

In the above code, X, which is a sub-matrix of C is decomposed into three matrices U, V while

and S which consists of the singular values in the formula:$X = USV^T$, where the singular vectors contained in U describe the latent user features and the right singular values in V' represent the hidden item latent features.

$$U = \begin{bmatrix} 0.22 & -0.11 & 0.29 & 0.41 & -0.11 & -0.34 & 0.52 & -0.06 & -0.41 \\ 0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11 \\ 0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49 \\ 0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01 \\ 0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27 \\ 0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\ 0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\ 0.30 & 0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17 \\ 0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58 \\ 0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23 \\ 0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.23 \\ 0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18 \end{bmatrix}$$

$$S = \begin{bmatrix} 3.34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.54 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.64 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.56 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.36 \end{bmatrix}$$

$$V' = \begin{bmatrix} 0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06 \\ 0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24 \\ 0.46 & -0.03 & 0.21 & 0.04 & 0.38 & 0.72 & -0.24 & 0.01 & 0.02 \\ 0.54 & -0.23 & 0.57 & 0.27 & -0.21 & -0.37 & 0.26 & -0.02 & -0.08 \\ 0.28 & 0.11 & -0.51 & 0.15 & 0.33 & 0.03 & 0.67 & -0.06 & -0.26 \\ 0.00 & 0.19 & 0.10 & 0.02 & 0.39 & -0.30 & -0.34 & 0.45 & -0.62 \\ 0.01 & 0.44 & 0.19 & 0.02 & 0.35 & -0.21 & -0.15 & -0.76 & 0.02 \\ 0.02 & 0.62 & 0.25 & 0.01 & 0.15 & 0.00 & 0.25 & 0.45 & 0.52 \\ 0.08 & 0.53 & 0.08 & -0.03 & -0.60 & 0.36 & -0.04 & -0.07 & -0.45 \end{bmatrix}$$

Simple arithmetic can confirm that multiplying the 3 matrices U,S,V, using only two singular values gives us $X \approx \hat{X} \approx U_2 S_2 V_2'$.

$$U_2 = \begin{bmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & 0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{bmatrix}$$

(A) U_2

$$S_2 = \begin{bmatrix} 3.34 & 0 \\ 0 & 2.54 \end{bmatrix}$$

(B) S_2

$$V_2' = \begin{bmatrix} 0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06 \\ 0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24 \end{bmatrix}$$

(C) V_2'

TABLE 4.3: $X = U_2.S_2.V_2'$

In this case, $\hat{X}$, is not an exact match of X:

$$\hat{X} = \begin{bmatrix} 0.16 & 0.40 & 0.38 & 0.47 & 0.18 & -0.05 & -0.12 & -0.16 & -0.09 \\ 0.14 & 0.37 & 0.33 & 0.40 & 0.16 & -0.03 & -0.07 & -0.10 & -0.04 \\ 0.15 & 0.51 & 0.36 & 0.41 & 0.24 & 0.02 & 0.06 & 0.09 & 0.12 \\ 0.26 & 0.84 & 0.61 & 0.70 & 0.39 & 0.03 & 0.08 & 0.12 & 0.19 \\ 0.45 & 1.23 & 1.05 & 1.27 & 0.56 & -0.07 & -0.15 & -0.21 & -0.05 \\ 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\ 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\ 0.22 & 0.55 & 0.51 & 0.63 & 0.24 & -0.07 & -0.14 & -0.20 & -0.11 \\ 0.10 & 0.53 & 0.23 & 0.21 & 0.27 & 0.14 & 0.31 & 0.44 & 0.42 \\ -0.06 & 0.23 & -0.14 & -0.27 & 0.14 & 0.24 & 0.55 & 0.77 & 0.66 \\ -0.06 & 0.34 & -0.15 & -0.30 & 0.20 & 0.31 & 0.69 & 0.98 & 0.85 \\ -0.041 & 0.25 & -0.10 & -0.21 & 0.15 & 0.22 & 0.50 & 0.71 & 0.62 \end{bmatrix}$$

The more dense matrix reduced with SVD, $\hat{X}$, gets closer and closer to X as more and more singular values are kept. As explained in Section 3.6.4, the number of the largest singular values kept is referred to as the rank k of matrix S. Root Mean Squared Error (RMSE)(4.6.1) was used to establish these optimal value of k and the results are displayed in Figure 4.1 below:

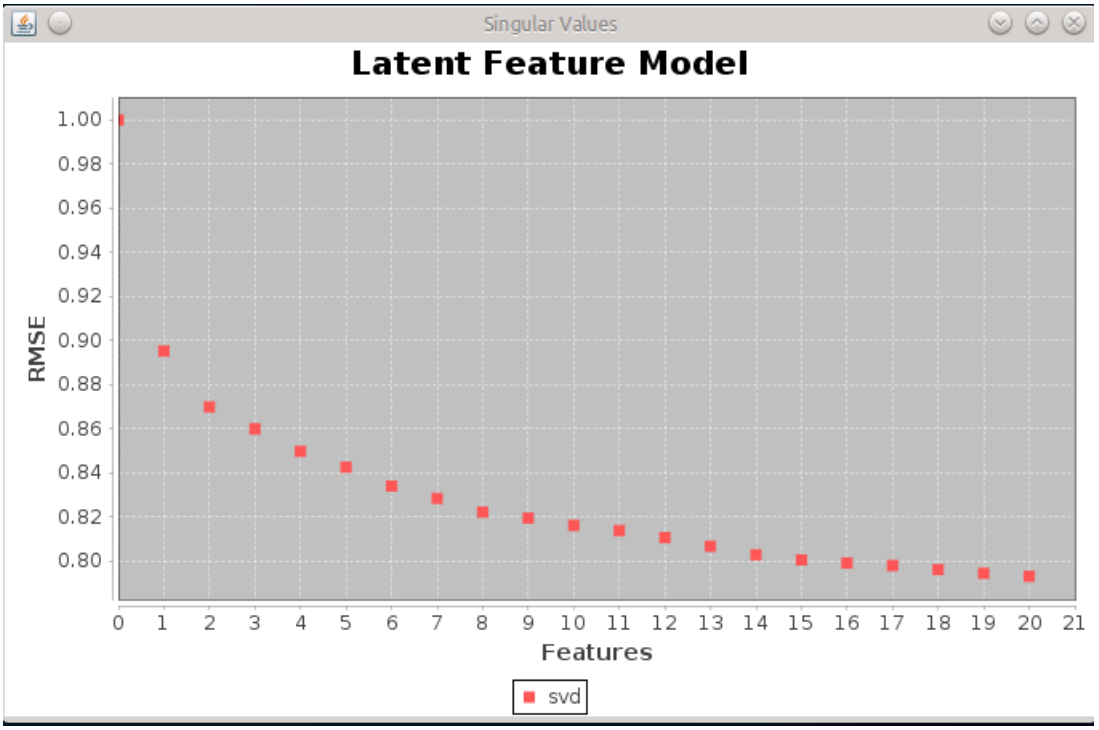


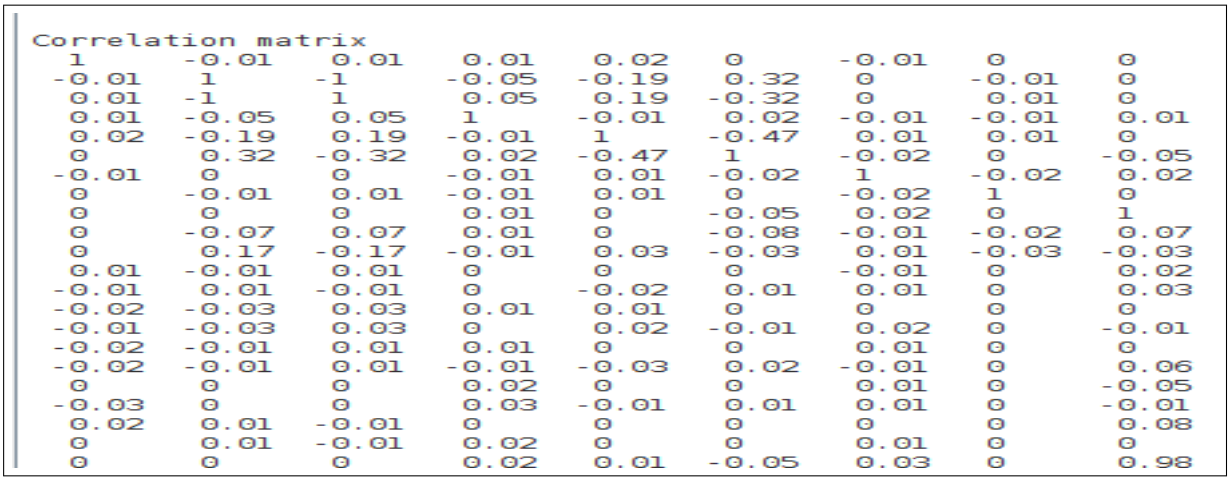
FIGURE 4.1: Latent Feature Model Performance

The results indicated that the model performed better with more latent attributes. The value of k has to be large enough to capture all the latent relationships while avoiding over-fitting.

Extracting Latent Features Using Principal Components Analysis

Although principal component analysis has proven to be an extremely popular technique for dimensionality reduction, like SVD, it has applications in many areas which include data compression, image analysis and feature extraction. In PCA data is considered as consisting of underlying latent relationships and noise. The PCA method reveals relationships between objects and facilitates similarity computations.

In this study,a user-item feature matrix consisting of 25000 records, read from a comma-delimited text file, was used to perform PCA. The first task in PCA is to compute a correlation matrix C as shown in Figure 4.2



Correlation matrix

1	-0.01	0.01	0.01	0.02	0	-0.01	0	0
-0.01	1	-1	-0.05	-0.19	0.32	0	-0.01	0
0.01	-1	1	0.05	0.19	-0.32	0	0.01	0
0.01	-0.05	0.05	1	-0.01	0.02	-0.01	-0.01	0.01
0.02	-0.19	0.19	-0.01	1	-0.47	0.01	0.01	0
0	0.32	-0.32	0.02	-0.47	1	-0.02	0	-0.05
-0.01	0	0	-0.01	0.01	-0.02	1	-0.02	0.02
0	-0.01	0.01	-0.01	0.01	0	-0.02	1	0
0	0	0	0.01	0	-0.05	0.02	0	1
0	-0.07	0.07	0.01	0	-0.08	-0.01	-0.02	0.07
0	0.17	-0.17	-0.01	0.03	-0.03	0.01	-0.03	-0.03
0.01	-0.01	0.01	0	0	0	-0.01	0	0.02
-0.01	0.01	-0.01	0	-0.02	0.01	0.01	0	0.03
-0.02	-0.03	0.03	0.01	0.01	0	0	0	0
-0.01	-0.03	0.03	0	0.02	-0.01	0.02	0	-0.01
-0.02	-0.01	0.01	0.01	0	0	0.01	0	0
-0.02	-0.01	0.01	-0.01	-0.03	0.02	-0.01	0	0.06
0	0	0	0.02	0	0	0.01	0	-0.05
-0.03	0	0	0.03	-0.01	0.01	0.01	0	-0.01
0.02	0.01	-0.01	0	0	0	0	0	0.08
0	0.01	-0.01	0.02	0	0	0.01	0	0
0	0	0	0.02	0.01	-0.05	0.03	0	0.98

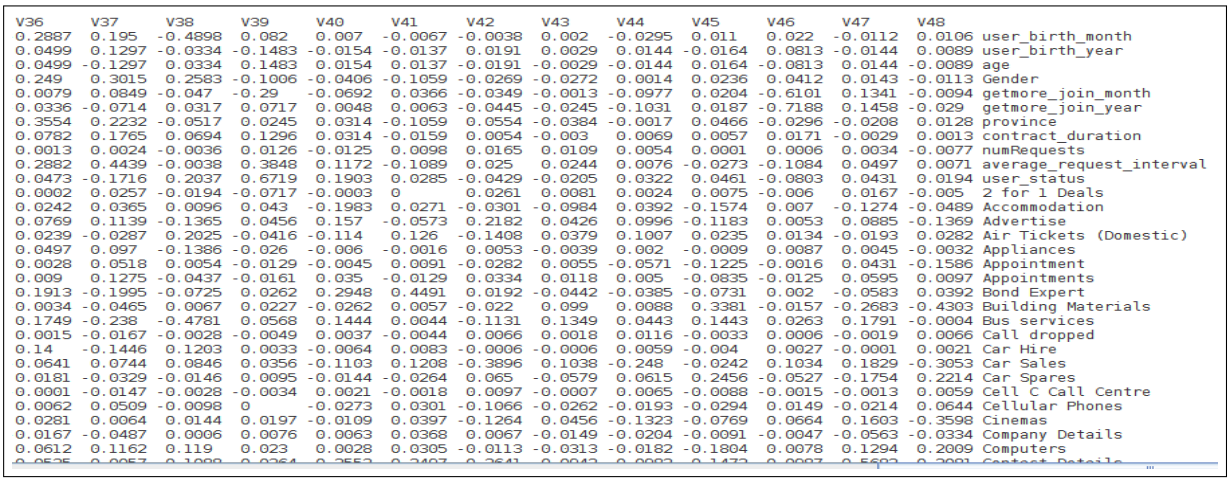
FIGURE 4.2: Correlation Matrix

We then solve for the roots of the correlation matrix 4.2and derive eigenvalues where each eigenvalue satisfies $|C - \lambda I| = 0$, where I is the identity of C. In order to be able to validate our results we extracted a sample matrix S from our dataset using equation 4.2:

$$S = \frac{1}{n-1} \sum_{t=1}^{n} (X_x - \bar{x})(X_x - \bar{x})^1 \tag{4.2}$$

where $\frac{1}{n-1}$ represents the sample size we used while the other part represented the normalising algorithm.

Figure 4.3 below shows the PCA model results for the top 15 eigenvalues.



V36	V37	V38	V39	V40	V41	V42	V43	V44	V45	V46	V47	V48	
0.2887	0.195	-0.4898	0.082	0.007	-0.0067	-0.0038	0.002	-0.0295	0.011	0.022	-0.0112	0.0106	user_birth_month
0.0499	0.1297	-0.0334	-0.1483	-0.0154	-0.0137	0.0191	0.0029	0.0144	-0.0164	0.0813	-0.0144	0.0089	user_birth_year
0.0499	-0.1297	0.0334	0.1483	0.0154	0.0137	-0.0191	-0.0029	-0.0144	0.0164	-0.0813	0.0144	-0.0089	age
0.249	0.3015	0.2583	-0.1006	-0.0406	-0.1059	-0.0269	-0.0272	0.0014	0.0236	0.0412	0.0143	-0.0113	Gender
0.0079	0.0849	-0.047	-0.29	-0.0692	0.0366	-0.0349	-0.0013	-0.0977	0.0204	-0.6101	0.1341	-0.0094	getmore_join_month
0.0336	-0.0714	0.0317	0.0717	0.0048	0.0063	-0.0445	-0.0245	-0.1031	0.0187	-0.7188	0.1458	-0.029	getmore_join_year
0.3554	0.2232	-0.0517	0.0245	0.0314	-0.1059	0.0554	-0.0384	-0.0017	0.0466	-0.0296	-0.0208	0.0128	province
0.0782	0.1765	0.0694	0.1296	0.0314	-0.0159	0.0054	-0.003	0.0069	0.0057	0.0171	-0.0029	0.0013	contract_duration
0.0013	0.0024	-0.0036	0.0126	-0.0125	0.0098	0.0165	0.0109	0.0054	0.0001	0.0006	0.0034	-0.0077	numRequests
0.2882	0.4439	-0.0038	0.3848	0.1172	-0.1089	0.025	0.0244	0.0076	-0.0273	-0.1084	0.0497	0.0071	average_request_interval
0.0473	-0.1716	0.2037	0.6719	0.1903	0.0285	-0.0429	-0.0205	0.0322	0.0461	-0.0803	0.0431	0.0194	user_status
0.0002	0.0257	-0.0194	-0.0717	-0.0003	0	0.0261	0.0081	0.0024	0.0075	-0.006	0.0167	-0.005	2 for 1 Deals
0.0242	0.0365	0.0096	0.043	-0.1983	0.0271	-0.0301	-0.0984	0.0392	-0.1574	0.007	-0.1274	-0.0489	Accommodation
0.0769	0.1139	-0.1365	0.0456	0.157	-0.0573	0.2182	0.0426	0.0996	-0.1183	0.0053	0.0885	-0.1369	Advertise
0.0239	-0.0287	0.2025	-0.0416	-0.114	0.126	-0.1408	0.0379	0.1007	0.0235	0.0134	-0.0193	0.0282	Air Tickets (Domestic)
0.0497	0.097	-0.1386	-0.026	-0.006	-0.0016	0.0053	-0.0039	0.002	-0.0009	0.0087	0.0045	-0.0032	Appliances
0.0028	0.0518	0.0054	-0.0129	-0.0045	0.0091	-0.0282	0.0055	-0.0571	-0.1225	-0.0016	0.0431	-0.1586	Appointment
0.009	0.1275	-0.0437	-0.0161	0.035	-0.0129	0.0334	0.0118	0.005	-0.0835	-0.0125	0.0595	0.0097	Appointments
0.1913	-0.1995	-0.0725	0.0262	0.2948	0.4491	0.0192	-0.0442	-0.0385	-0.0731	0.002	-0.0583	0.0392	Bond Expert
0.0034	-0.0465	0.0067	0.0227	-0.0262	0.0057	-0.022	0.099	0.0088	0.3381	-0.0157	-0.2683	-0.4303	Building Materials
0.1749	-0.238	-0.4781	0.0568	0.1444	0.0044	-0.1131	0.1349	0.0443	0.1443	0.0263	0.1791	-0.0004	Bus services
0.0015	-0.0167	-0.0028	-0.0049	0.0037	-0.0044	0.0066	0.0018	0.0116	-0.0033	0.0006	-0.0019	0.0066	Call dropped
0.14	-0.1446	0.1203	0.0033	-0.0064	0.0083	-0.0006	-0.0006	0.0059	-0.004	0.0027	-0.0001	0.0021	Car Hire
0.0641	0.0744	0.0846	0.0356	-0.1103	0.1208	-0.3896	0.1038	-0.248	-0.0242	0.1034	0.1829	-0.3053	Car Sales
0.0181	-0.0329	-0.0146	0.0095	-0.0144	-0.0264	0.065	-0.0579	0.0615	0.2456	-0.0527	-0.1754	0.2214	Car Spares
0.0001	-0.0147	-0.0028	-0.0034	0.0021	-0.0018	0.0097	-0.0007	0.0065	-0.0088	-0.0015	-0.0013	0.0059	Cell C Call Centre
0.0062	0.0509	-0.0098	0	-0.0273	0.0301	-0.1066	-0.0262	-0.0193	-0.0294	0.0149	-0.0214	0.0644	Cellular Phones
0.0281	0.0064	0.0144	0.0197	-0.0109	0.0397	-0.1264	0.0456	-0.1323	-0.0769	0.0664	0.1603	-0.3598	Cinemas
0.0167	-0.0487	0.0006	0.0076	0.0063	0.0368	0.0067	-0.0149	-0.0204	-0.0091	-0.0047	-0.0563	-0.0334	Company Details
0.0612	0.1162	0.119	0.023	0.0028	0.0305	-0.0113	-0.0313	-0.0182	-0.1804	0.0078	0.1294	0.2009	Computers

FIGURE 4.3: Sample PCA Variables

Table 4.4 below attempts to explain the results depicted in figure 4.3 above.

	Principal Component			
Variable	**v1**	**v2**	**v3**	**v4**
user_birth_month	0.0013	0.0068	**0.5683**	0.0313
user_birth_year	-0.0021	**-0.5676**	0.0043	-0.0067
age	0.0085	**0.0557**	**0.6195**	0.0007
gender	0.0085	**0.0557**	**0.6195**	0.0007
getmore_join_date	-0.0942	0.0882	-0.2263	-0.0011
getmore_join_year	0.106	-0.4121	-0.0062	0.0295
...	...	...	...	...

TABLE 4.4: Explaining Principal Components

The interpretation of the principal components in table 4.4 above is based on finding the variables that are most strongly correlated with each component, that is, which of these numbers (written in bold) are large enough in magnitude, the farthest from zero in either positive or negative direction. Determining which numbers are large or small is an arbitrary subjective decision.In this study, the correlation value had to be at least 0.5 to be relevant.

In the sample above, the first principal component (v1) does not correlate with any attribute while the second principal component correlates with 3 attributes. It negatively correlates with user_birth_year while it positively correlates with age and gender. Age and gender seem to have the same significance for all variables. *Age* and *Gender* vary directly and at equal angles in the table above which means that one of them was redundant in the given data snapshot.

Examining the eigenvalues helped us to determine how many principal components should be considered.

eigenvalue	proportion	cumulative	
8.92683	0.12573	0.12573	-0.331Member called in for-0.329numRequests-0.327Call dropped-0.327Enquiry about service-0.323Cell C Call Centre...
5.58707	0.07869	0.20442	-0.383Directions-0.354Appointment-0.346Cellular Phones+0.322Appointments-0.315Car Spares...
3.54579	0.04994	0.25436	0.477Municipality+0.32 Small business funding+0.318Training-0.304Legal Wellbeing-0.262Paint...
2.7256	0.03839	0.29275	-0.395Small business funding-0.355Contact Details-0.351Training-0.286Legal Wellbeing-0.278Company Details...
2.53366	0.03569	0.32844	0.42 School+0.365Property-0.346Weather+0.341Wake up Call-0.252Paint...
2.35228	0.03313	0.36157	-0.561user_birth_year+0.561age-0.361getmore_join_year+0.276getmore_join_month-0.139user_status...
2.19884	0.03097	0.39254	-0.365Accommodation-0.3Dial A Teacher-0.269Tyre-0.263Employment Agency-0.246Weather...
1.8962	0.02671	0.41924	0.377Dial A Teacher+0.366Accommodation-0.342Building Materials-0.308Tyre+0.222Employment Agency...
1.58055	0.02226	0.4415	0.622Cinemas-0.617Car Sales-0.219Weather-0.21Property-0.142Bus services...
1.49585	0.02107	0.46257	-0.529Employment Agency+0.425Dial A Teacher+0.388Mathematics-0.316Accommodation-0.229Weather...
1.30496	0.01838	0.48095	-0.549getmore_join_month-0.465user_status+0.463getmore_join_year+0.212age-0.212user_birth_year...
1.18498	0.01669	0.49764	0.671Computers-0.403Service+0.244user_birth_month+0.23 Building Materials-0.219Appointments...
1.14039	0.01606	0.5137	0.516Mathematics+0.357Languages-0.332Advertise-0.249Bus services+0.233Air Tickets (Domestic)...
1.12294	0.01582	0.52952	-0.451Advertise+0.437average_request_interval-0.409Computers+0.22 user_birth_month-0.217Service...
1.11148	0.01565	0.54518	0.51 TV+0.508Air Tickets (Domestic)+0.273province+0.269Exchange rate+0.262Bus services...
1.09277	0.01539	0.56057	-0.398user_birth_month-0.339Washing machine+0.319Follow up on Cancellation+0.313average_request_interval-0.239Gender...
1.07481	0.01514	0.5757	-0.498Advertise+0.428Service+0.245Appointments+0.236Bus services-0.226Games...
1.05937	0.01492	0.59062	0.481province-0.415Exchange rate-0.298contract_duration+0.229Gender-0.192Tax Expert...
1.04215	0.01468	0.6053	0.448Get It Membership Cancellation+0.403Finance Expert-0.319Follow up on Cancellation-0.223Service-0.221Gender...
1.03111	0.01452	0.61983	0.533Finance Expert+0.38 Gender-0.301Get It Membership Cancellation+0.283Tax Expert+0.28 Washing machine...
1.01521	0.0143	0.63412	0.457Car Hire+0.306contract_duration-0.278Get It Membership Cancellation-0.263Appliances-0.259On boarding Classifieds...
1.01066	0.01423	0.64836	0.465Car Hire-0.389Appliances-0.322Onboarding+0.265Tax Expert+0.243Get It Membership Cancellation...
1.00362	0.01414	0.66249	0.55 2 for 1 Deals-0.491Movie Vouchers-0.357Games-0.276Personal loans+0.273Car Hire...

FIGURE 4.4: Eigenvalues and the Proportion of Variation

Added together the eigenvalues in figure 4.4 above we get the total variance of approximately 47.144. The proportion column indicates the ratio of the variation in the variable that can be explained by the first eigenvalue (only about 12.5% in the example).

Next, we explain the computation of the principal component scores. For example, the first principal component was computed using the elements of the first eigenvector: $user_birth_month = -0.331 \times MemberCalledinforInfo - 0.329 \times numRequests \times \ldots$

The coefficients above are too small to be used to explain the user_birth_month attribute. In this case PCA, like SVD, gives us the latitude to compare item and user features directly against each other in the same feature space and to establish latent feature relationships.

4.2.1 Calculating Similarity using Latent Factors

SVD replaces the N dimensions in the original feature matrix by $k < N$ best surrogates which can be used to estimate the original ones. This leaves us with reasonably fewer values on which to perform our similarity computation. Using the matrix, $\hat{X}$ produced by SVD with k=2 above, the latent features of two users a and b are indicated by highlighted rows.

$$\hat{X} = \begin{bmatrix} 0.16 & 0.40 & 0.38 & 0.47 & 0.18 & -0.05 & -0.12 & -0.16 & -0.09 \\ 0.14 & 0.37 & 0.33 & 0.40 & 0.16 & -0.03 & -0.07 & -0.10 & -0.04 \\ 0.15 & 0.51 & 0.36 & 0.41 & 0.24 & 0.02 & 0.06 & 0.09 & 0.12 \\ 0.26 & 0.84 & 0.61 & 0.70 & 0.39 & 0.03 & 0.08 & 0.12 & 0.19 \\ 0.45 & 1.23 & 1.05 & 1.27 & 0.56 & -0.07 & -0.15 & -0.21 & -0.05 \\ 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\ 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\ 0.22 & 0.55 & 0.51 & 0.63 & 0.24 & -0.07 & -0.14 & -0.20 & -0.11 \\ 0.10 & 0.53 & 0.23 & 0.21 & 0.27 & 0.14 & 0.31 & 0.44 & 0.42 \\ -0.06 & 0.23 & -0.14 & -0.27 & 0.14 & 0.24 & 0.55 & 0.77 & 0.66 \\ -0.06 & 0.34 & -0.15 & -0.30 & 0.20 & 0.31 & 0.69 & 0.98 & 0.85 \\ -0.041 & 0.25 & -0.10 & -0.21 & 0.15 & 0.22 & 0.50 & 0.71 & 0.62 \end{bmatrix}$$

There are several approaches used to compute similarity $\text{sim}(u_x, u_y)$ between users in collaborative recommender systems in the literature. The most common of these approaches calculate the similarity between two users based on their ratings on the items that both users have rated. The two most popular approaches to calculating similarity are the correlation- and cosine-based algorithms. In this section, we used the Cosine Similarity algorithm (4.3) to compute user-user similarity using latent features.

$$sim(x, y) = cos(\theta) = \frac{vectorX.vectorY}{|vectorX| \,.\, |vectorY|} \tag{4.3}$$

The highlighted users give us two vectors: $u_x = \{0.16, 0.40, 0.38, 0.47, 0.18, -0.05, -0.12, -0.16, -0.09\}$
$u_y = \{0.14, 0.37, 0.33, 0.40, 0.16, -0.03, -0.07, -0.10, -0.04\}$

Cosine similarity simply calculates the angle between the vectors *x,y*, which in this case is **0.005143**. The active user $u_x \in \hat{X}$ is compared against user u_y who can be any user in $\hat{X}$ and a certain number of users most similar to u_x is returned and used to generate recommendations for the active user.

4.3 Collaborative Filtering User-User Similarity

Useful information can be obtained from the PLP Group's dataset about its customers, including their subscription to different products and/or services. A group of customers may share similar preferences for some products and/or services, which may then be recommended to a customer with similar characteristics to members of the group. The following question was posed to look into this issue, and to investigate different approaches for discovering similar users.

Which statistical or data mining technique is most suitable for discovering PLP Group subscribers whose preferences are similar to those of a client for whom a service is being recommended?

To answer this question different approaches were used to determine the similarity between the active user and other users. An Association Rule Mining algorithm, similarity measures and a clustering algorithm were used. In the field of recommendation systems Association Rule Mining can be used to discover relationships or correlations between items. A user who prefers items that are preferred by other users is recommended other items which these users prefer. With suitable similarity measure the similarity between any two users can be determined by using the ratings of all the users. Clustering methods identify users who share similar item preferences, and the cluster to which the active user belongs is used to recommend items.

4.3.1 Using Association Rules (user-user Similarity)

The similarity between users was determined based on the similarity of their preferred items. Similar items were discovered by mining the data for frequent itemsets. An itemset is a grouping of items into a single unit, and a frequent itemset is one in which the support is equal to or greater than some specified threshold value. To calculate user similarity we had to list user transactions as vectors and calculate the similarity of the vectors. We had a set of m user transactions where active user's u_a transactions were represented by a vector: $T_a = \{t_1, t_2, \ldots, t_n\}$ where t_i is an item for which user u_a has previously rated.

To perform the required experiment a matrix consisting of 25000 users (as records) and 500 items (as variables) was used. The data was divided into a training dataset and a test dataset. For training, 70% of the records were used and for testing, the remaining 30% of the records were used.

The transaction matrix U, shown in equation 4.4 below

$$X = \begin{bmatrix} * & i_1 & i_2 & i_3 & i_4 & i_5 & i_6 & i_8 & i_9 & i_{10} \\ u_1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ u_2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ u_3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ u_4 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ u_5 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ u_6 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ u_7 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ u_8 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ u_9 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ u_{10} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ u_{11} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ u_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ u_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.4}$$

The similarity between any two users is greatly influenced by the items the two users have in common. SVD was used to reduce dimensionality and limit the number of variables available for analysis. With each user treated as a vector of items, the following equation was used to calculate similarity:

$sim(u_1, u_2) = \frac{|t_1 \cap t_2|}{\sqrt{|t_1|.|t_2|}}$

In this case, $t_1 = \{1, 0, 0, 1, 0, 0, 0, 0, 0\}$ and $t_2 = \{1, 0, 1, 0, 0, 0, 0, 0, 0\}$, as per the example given in equation 4.4. we expressed $sim(u_1, u_2)$ in terms of the size of the respective transaction vectors as shown above. However, this example is simplistic in that it uses the data before it is normalised. Only item 1 is common in both vectors. In a large dataset like the one used in this study an item had many occurrences.

4.3.2 Memory-Based Collaborative Filtering

Clustering users according to rating data involved using both explicit and implicit ratings. Implicit ratings were obtained by a simple count of the number of times a user requested a specific item. The similarity techniques employed in this subsection included Adjusted Cosine Similarity, Pearson Correlation Similarity, and Euclidean Distance (Simple K-Means) Algorithm.

As already explained in Section 2.7.5, CF recommendation proceeded through three basic steps. Initially, a neighbourhood of users that are most similar to the user requesting recommendations (the active user a_i) is constructed using the line of code below:

```
public static GenericUserSimilarity.UserUserSimilarity
    constructUserSimilarityFromResultSet(ResultSet rs){
                        try {
                                long user1 = rs.getInt(USER_ID_1);
                                long user2 = rs.getInt(USER_ID_2);
                                double rel = rs.getDouble(SIMILARITY);
                                GenericUserSimilarity.UserUserSimilarity similarity =
    new GenericUserSimilarity.UserUserSimilarity(user1, user2, rel);
                                return similarity;
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                        return null;
                }
```

The process of computing similarity has already been outlined in Section 2.7.5. Then, predictions were made of the ratings the active user would give to items new to him. This prediction was based on the ratings given to the items by the active user's neighbours determined in the first step.Finally, the a recommendation list based on the predicted ratings is constructed. The implementation of the final step in Java resembled the following:

```
public class MyRecommenderBuilder implements RecommenderBuilder {
        public Recommender buildRecommender(DataModel model) throws TasteException {
                UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
                UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1,
    similarity, model);
                return new GenericUserBasedRecommender(model, neighborhood,
    similarity);}
                }
```

The next subsection explains the results obtained from each of the three measures of similarity listed above.

Pearson Correlation Similarity

A common, but harder to compute formula for the Pearson correlation coefficient r is given below:

$$sim(xy) = r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2}\sqrt{\sum(y-\bar{y})^2}} \tag{4.5}$$

where x and y are user rating datasets for the respective users. The Java implementation of the Pearson Correlation algorithm in the Mahout Package used to build this recommender system considers the algorithm to be a part of the nearest neighbourhood algorithms.

```
1
2  public class UserBasedRecommender implements Recommender {
3   public UserBasedRecommender(DataModel userDataModel) throws TasteException {
4                     UserSimilarity userSimilarity = new
       PearsonCorrelationSimilarity(userDataModel);
5                  userSimilarity.setPreferenceInferrer(new
       AveragingPreferenceInferrer(userDataModel));
6                  UserNeighborhood neighborhood =
7                            new NearestNUserNeighborhood(3, userSimilarity,
       userDataModel);
8
9                  recommender = new CachingRecommender(new
       GenericUserBasedRecommender(userDataModel, neighborhood, userSimilarity)); }
10                 }
```

This algorithm was applied on a dataset of 67000 users who had rated a total of 1200 items. The dataset did not include any feature items.Table 4.5 shows a sample of the data used to calculate collaborative user filtering on a rating scale of 1 to 10.

To predict missing ratings we used rating extrapolation where available ratings by a user were used to predict missing ones. In this study we predicted all missing ratings by using the user average ($\bar{r}_u$) and item average ($\bar{r}_i$) values, together with the overall matrix average ($\bar{r}$) value. First, we used linear regression (LR) to identify the parameters that return the best possible estimation of our known ratings:

$$r(ui) = \alpha\bar{r} + \beta\bar{r}_u + \gamma\bar{r}_i + \varepsilon_{ui} \tag{4.6}$$

The identified normalisation parameters were then used to calculate the unknown user ratings:

$$\hat{r}(ui) = \alpha\bar{r} + \beta\bar{r}_u + \gamma\bar{r}_i \tag{4.7}$$

As described in section 2.7.3.5, *r(ui)* represents the known rating of user u on item i. In order to find an estimated value for the above of equation, LR tries to minimise the sum of the squared error residuals $\sum_{i=1}^{n} \varepsilon_i^2$.

Table 4.5 gives a small view of the rating matrix before it was normalised. The examples that follow use row data for simplicity.

	2-4-1 Deals	Legal Assist	Doctor Booking	Finance Assist	Medical Assist	Car Spares
u_1	3	7	4	9	9	7
u_2	7	5	5	3	8	8
u_3	7	5	5	0	8	4
u_4	5	6	8	5	9	8
u_5	5	8	8	8	10	9
u_6	7	7	8	4	7	8

TABLE 4.5: Ratings given to six items by selected users

With the data presented in table 4.5 we can create a feature vector for each user: $x_1 = (3, 7, 4, 9, 9, 7)^T$ $x_6 = (7, 7, 8, 4, 7, 8)^T$,where x_1 and x_6 correspond to users u_1 and u_6 respectively.

Substituting the values of x_1 and x_6 above in the Pearson correlation Coefficient equation 4.5, we get the following:

	x_1	x_6	$\bar{x_1}$	$\bar{x_6}$	$x_1 - \bar{x}$	$x_6 - \bar{x_6}$	$(x_1 - \bar{x})^2$	$(x_6 - \bar{x_6})^2$	$(x_1 - \bar{x})(x_6 - \bar{x_6})$
	3	7			-3.5	0.17	12.25	0.0289	-0.595
	7	7			0.5	0.17	0.25	0.0289	0.085
	4	8			-2.5	1.17	6.25	1.3689	-2.925
	9	4			2.5	-2.83	6.25	8.0089	-7.075
	9	7			2.5	0.17	6.25	0.0289	0.425
	7	8			0.5	1.17	0.25	1.3689	0.585
Total	39	41	6.5	6.83	-2.5	0.02	31.25	10.8334	-9.5

TABLE 4.6: Pearson Correlation Table of Values

The various variables above give us the following:

$$sim(x_1 x_6) = r = \frac{9.5}{\sqrt{31.25^2}\sqrt{10.8334^2}} \approx 0.03$$

The same procedure was carried out for all vectors and the top-n most similar users to the active user were chosen for the purposes of recommendation. Table 4.7 below is a matrix representation of user similarity. It gives an example of the similarity between the six users given in Table 4.5 above.

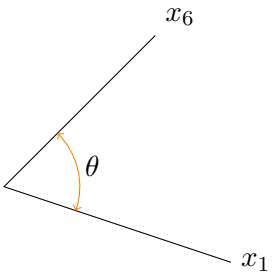
	Users						
Users	u_1	u_2	u_3	u_4	u_5	u_6	...
u_1	1	-0.1195	-0.3574	0.2082	0.7619	0.03	...
u_2	-0.1195	1	0.7535	0.5806	0.0321		...
u_3	-0.3574	0.7535	1	0.4514			...
u_4	0.2082	0.5806	0.4514	1		0.6179	...
u_5	0.7619	0.0321			1	0.0812	...
u_6	0.03			0.6179	0.0812	1	...
...	...	...	...	...	...	...	...

TABLE 4.7: Pearson's correlation Coefficient Similarity for Selected Users

Users are represented by the userID attribute from the user dataset. From this data, a neighbourhood of users is formed from which recommendations for the active user are made. In table 4.7 above, *user1* and *user2* are significantly correlated (0.7619). The same applies to user u_2 and u_3, but not for users u_1 and u_3 who have a negative correlation which implies non-similarity.

Adjusted Cosine Similarity

Cosine Similarity is the measure of calculating the difference of angle between two rating or feature vectors as shown in the diagram below.



Since the length of the vectors did not matter we had to normalise each vector to a unit vector by using multiplicative normalisation discussed in section 3.6.3, and then calculated the inner product of two vectors. The underlying principle in this approach is similar to the one discussed in the Pearson's correlation approach except that in the cosine-based approach the users are considered as vectors $\vec{a_i}$ and $\vec{a_j}$ in an m-dimensional space, where $m = |B_{ij}|$. $|B_{ij}|$ represents items rated by both user i and user j. The vectors, therefore, represent the rating preferences for the items that were rated by both users. The similarity is then calculated as the angle between those two vectors.

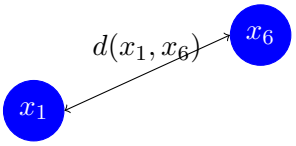
However, Cosine similarity also enables us to use content features as already shown in section 4.2.1 above. According to the Cosine Similarity algorithm, the two users in our example,

x_1 and x_2 are more similar than what the Pearson Correlation algorithm made them appear to be: $sim(x_1, x_6) = \frac{\vec{x_1}.\vec{x_6}}{|\vec{x_1}|.|\vec{x_6}|} \approx 0.108$

After we had implemented the cosine similarity algorithm without interpolating the missing ratings first, due to the data sparsity problem, if a given user has no users among potential candidates of nearest neighbours, who rated the same item, all cosine similarity scores became 0, and we could not establish which users were the most similar to the active user. To ameliorate this problem, we used equation 4.7 to estimate missing ratings.

4.3.2.1 Distance-based User Similarity

The Euclidean Distance is one of the simplest yet powerful ways to determine the similarity score between any two users. Euclidean Distance is the ordinary straight line distance between two points in Euclidean Space:



In 2-dimensional space, the distance is just between two objects on an xy Cartesian plane space, and we just extend this concept to use for our 90000-dimensional space matrix to calculate the length of the attributes. However, the users should have numerical attributes which explain why we needed rating data.Since rating data varied in scale and user-bias we had to normalise it to rationalise the rating discrepancies. Finally, we applied the Euclidean distance formula as stated in equation 4.8:

$$d(x_1, x_6) = \sqrt{\sum_{i=1}^{n}(x_1 - x_6)^2} \tag{4.8}$$

We subtracted each rating on an item given by user x_6 from the rating of the same item given by the user x_1 and added them in quadrature. The result was the ”distance” between the two users . The shorter the distance, the more similar the users were.

4.3.3 Cluster-based User Similarity

The K-means algorithm clustered users into k mutually exclusive clusters like in the example in Figure 4.8

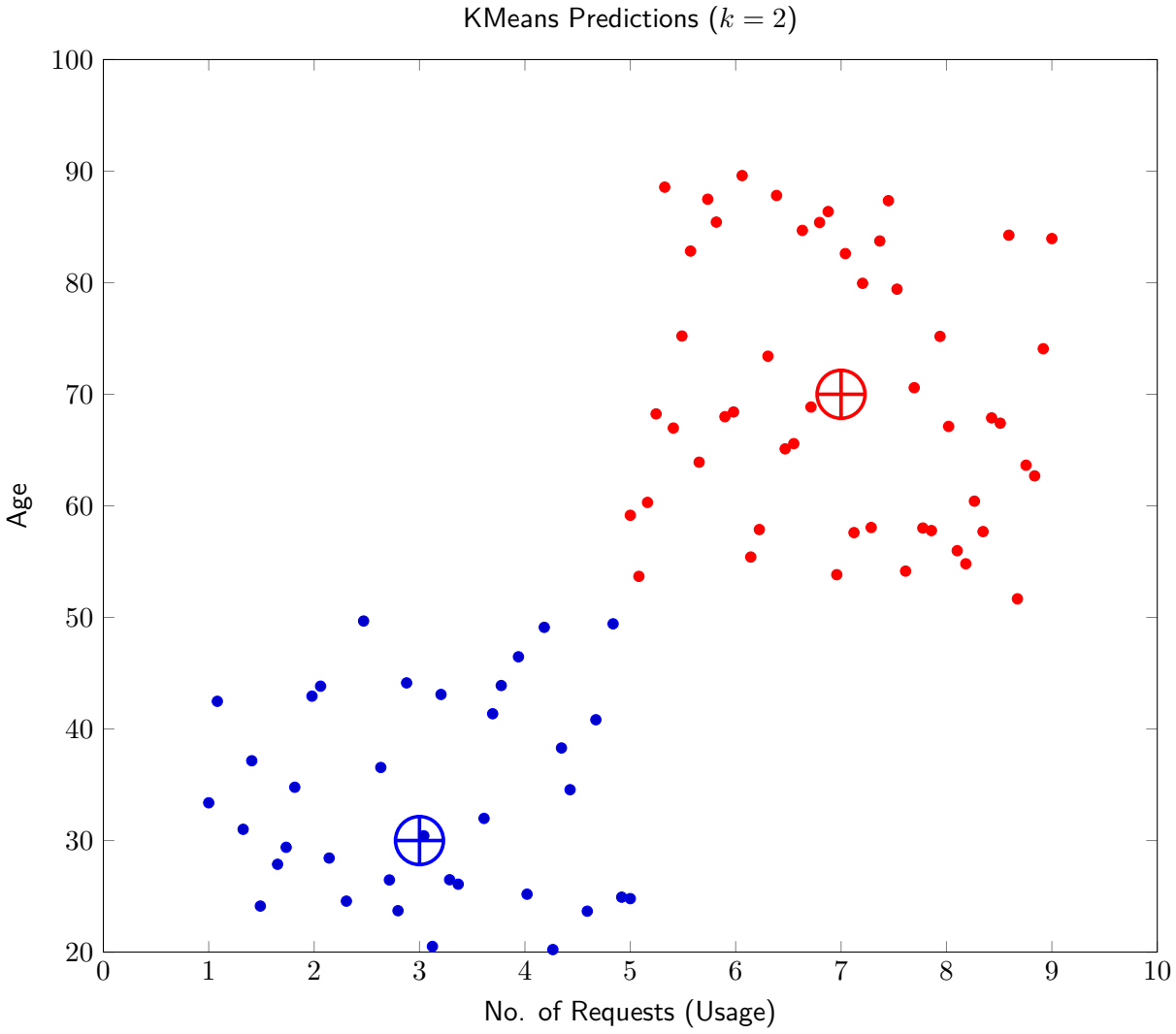


FIGURE 4.5: The K-Means Clustering Technique

The symbols $\bigoplus$ mark the centroids of each cluster where the number of clusters (k) is 2. The algorithm itself is formally defined by the following equation 4.9

$$\Sigma_{i=1}^{k} \Sigma_{x_j \in S_j} d(x_j, \mu_i) \tag{4.9}$$

where, S_i is the i^{th} cluster (i = 1,2,3,...,k). μ_i is the i^{th} centroid of all the points in cluster S_i. d is simply a distance function.

The K-Means algorithm performed very poorly due to the inconsistency and sparsity of the rating regime. Performance deteriorated exponentially with the increase in the number of classes. The results of a k=5 clustering example displayed in Figure 4.8 below indicate that almost 65% of the instances were incorrectly clustered.

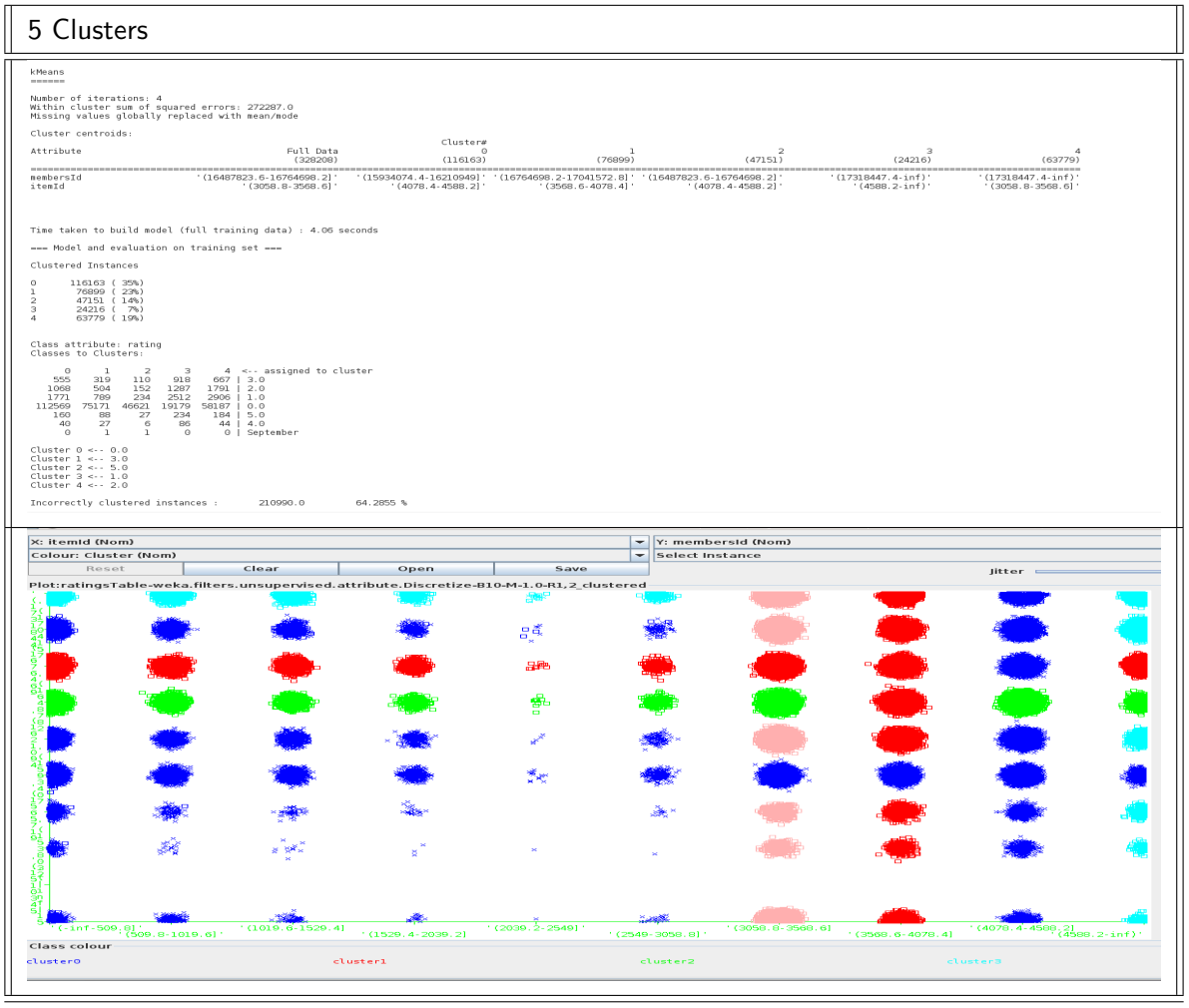


TABLE 4.8: K-Means Clustering Results

Using a dataset of 328208 records the algorithm performed even more poorly as more clusters were added by increasing the value of k. The intention was to improve recommendations by selecting a user's neighbours from his own user cluster.

4.3.4 User Similarity with Matrix Factorization

In this study, we used a version of SVD which combined both user and item features in the same vector space. The use of SVD gave us three distinct advantages:

1. Ability to combine user data with item data in the same space and compute underlying feature relationships;

2. Ability to use both rating data and binary data collected from usage figures. Most users did not stay long enough to rate an item, but their continued use/request for an item implied favour for it.

3. Ability to reduce dimensionality and leave our data manageable. However, SVD weighed heavily on our computing resources to the extent that we were forced to use just a small subset of our dataset with 25000 records and 500 features.

In our approach, we had ratings and features from about m users and n items, arranged in an $m \times n$ matrix R. 70% of the data was used for training the model while 30% was used for validation. However, we faced three challenges:

- The number of users and items was too large, with the number of users far much larger than the number of items on a scale of almost 100:1.

- A large proportion of the matrix was unknown. About 90% of it was blank. It was sparse in the sense that it contained unknown values as opposed to zeros.

- Finally, the amount of observed data varied significantly among users and among items. This was caused by the fact that the users joined the GetMore service at different times and items got added at different intervals. While the *user-join-date* was available in the dataset, the no date was provided as to when an item had been added.

SVD, when used only with known ratings in a sparse matrix, is prone to over-fitting. In this study, we tried to impute the missing values by using known values as already explained equation 4.6 and 4.7 above. Our approach involved constructing a *user-item* matrix made of combined feature and rating vectors for m users and n items. That is, for each user $u \in U$, where $U = \{u_1, u_2, ..., u_m\}$ and for each item $i \in I$, where $I = \{i_1, i_2, ..., i_n\}$. Each user and item had distinct content vectors constructed from features like $u_1 = \{p_1, p_2, ..., p_k\}$ and $i_1 = \{a_1, a_2, ..., a_k\}$, where p_k and a_k are user profile features and item attributes respectively. The different feature and rating matrices had their missing values interpolated and then normalised (see section 3.6.3) before SVD was applied on them to create a united matrix.

At the end of this step a user-feature matrix (F) was constructed, similar to the matrix shown in *Figure* 3.12 The rest of the process resembled the procedure carried out in section sec:latentsvd above. We computed the similarity between users by using Adjusted Cosine Similarity (see section 4.3.2). The enhanced correlation algorithm (equation 3.7) was used to filter out bias caused by the dominance of feature attributes over rating data.

The results from the experiments above are summarised in the table 4.20 below:

Algorithm	RMSE	MAE	TP Rate	FP Rate	F-Measure	ROC	Accuracy
Pearson	0.45	0.23	0.67	0.25	0.67	0.71	67.13
Cosine	0.40	0.26	0.69	0.31	0.67	0.85	68.67
Euclidean Distance	0.59	0.35	0.47	0.37	0.38	0.62	47.02
Cluster-Based	0.95	0.74	0.36	0.64	0.72	0.51	36.20
Cosine with SVD	0.36	0.21	0.78	0.22	0.23	0.89	78.06

TABLE 4.9: Collaborative Filtering Usage Accuracy Matrix

The adjusted Cosine Similarity algorithm with SVD performed the best.

4.4 Content-based Filtering User-user Recommendation

The section sought to identify the major features of PLP users who extensively used the service. A closer look at this group of users would help inform the creation of targeted marketing groups while also challenging the service provider to make the service more palatable to other groups. The major questions answered in this section were:

> *What are the common traits and attributes of frequent users of the PLP Group's services? How can these characteristics be used in recommending services to similar clients?*

This question targeted the PLP group users who had already used the service. The dataset used in this section contained 67300 users who had rated at least one item. For all the classification algorithms we used 10 Folds cross-validation.

The classification data mining technique (see Section 2.7.3) was used to answer the first part of the question while the second part requires us to use the results of classification to recommend items to users.

4.4.1 Classification of Users by Usage

Classification involves predicting the value of a class attribute based on the values of other attributes referred as the predicting attributes or predictors. The attributes selected as predictors,

in this case, were selected during the data preparation phase using the Regression Classifier (see table 2.5). The following attributes were selected for use in this section:

Attribute Name	**Description**
age	The age of the user in integers
gender	Gender of the user: 1 male, 2 female
join_year	The year the user joined the service
province	The province of the user
avg_request_interval	The average time between the user's requests
number_of_requests	The total number of requests used as Class attribute
user_status	Indication if the user is still a member or not

TABLE 4.10: Attributes used in this Classification Exercise

First of all, users were classified according to usage. Users who had requested less than 5 item were considered low users while those who had requested between 5 and 10 items were grouped under medium users. Anything above 10 was considered high usage. However, this preliminary classification presented us with two challenges:

- The first major challenge we faced was that this classification according to usage did not factor in the length of time the user had used the service. Users who started using the service over 10 years earlier were to be equally compared with users who had just joined the service.
- Imbalanced data which arises because learning algorithms tended to be biased towards most frequent classes while ignoring minority ones. This would result in a faulty classification.

The first problem was addressed by factoring in the average interval between requests. Profitable users are those who used the service more than others and more frequently. We expressed the number of requests logged as a ratio of the interval between requests. The second problem was caused by the imbalances in the preliminary classification data as shown in table 4.11:

	Low Users	Medium Users	High Users
Absolute Numbers	52956	13218	1127
Percentage	78.69%	19.64%	1.67%

TABLE 4.11: Preliminary User Classifications

Our data in table 4.11 above presents a clear imbalance since its distribution skewed towards low users. One of the most used methods to learn from imbalanced data involves re-sampling

the data, either by deliberately over-sampling the under-represented classes or under-sampling the over-represented ones, until all the classes are equally represented.

Since the data in the sample was so imbalanced, we could not use classification as an accuracy measure,but the geometric mean of accuracy scores per class *(g-mean)*, defined by the equation 4.10 below:

$$g - mean = \sqrt[n]{\Pi_{i=1}^{n} \frac{x_i}{n_i}} \tag{4.10}$$

where x_i is the number items in class i that have been correctly classified while n_i is the total number of items in i.

Faced with this challenge, in this study we had to use random under-sampling which involved removing randomly chosen instances of over-represented classes in the dataset until all classes had the same number of records of 1127, and then we used the geometric mean to measure the quality of the inferred classifiers. Now we had used three sets of 10-fold data files as our dataset:

- The original imbalanced data which was biased towards low users with numeric continuous attributes;
- The original imbalanced data which was biased towards low users with categorical discrete attributes;
- The re-balanced dataset obtained using deliberate under-sampling.

We applied K-NN, NaiveBayes, C4.5 (J48),Apriori for Classification (AprioriC) and the Regression Classifier algorithms to the dataset. We also tried to use the same parameters for the algorithms of the same type with 10-fold cross-validation.

Table 4.12 summarises selected results from our classification activities.

Algorithm	**Original Numeric Data**		**Original Nominal**		**Rebalanced**	
	Correctly Classified (%)	G-Mean	Correctly Classified (%)	G-Mean	Correctly Classified (%)	G-Mean
Naive Bayes	61.17	13.23	67.23	4.46	51.32	14.44
C4.5 (J48)	68.37	43.29	74.22	19.87	45.23	8.17
AprioriC	59.02	0.00	60.82	0.00	62.64	0.00
LR	60.84	0.00	72.63	0.00	60.84	0.00
K-NN	60.43	11.31	60.79	6.19	52.80	9.76

TABLE 4.12: Classification Algorithms Results

The results above show that it was much easier to get a high accuracy rate when the data was imbalanced. This is understandable considering the proportion of low users to high users (50:1), but when all the classes had been re-sampled and had a balanced number of instances it became more difficult to achieve a good accuracy rate. Linear Regression was an exception returning similar results for the numerical data as it did for the re-balanced data.

In a rare fashion, the AprioriC algorithm Tunc and Dag [2006] performed better with the re-balanced data. The Confusion Matrix below provides more details on the performance of the AprioriC algorithm on the re-balanced data:

a	b	c	<---- classified as
589	420	47	a = High
37	472	24	b = Medium
501	0235	1056	c = Low

TABLE 4.13: Confusion Matrix of the AprioriC

Surprisingly, the largest number of absolute values correctly classified was for the low users who also happened to dominate the full dataset.

Having proved that we could satisfactorily classify users according to usage, we extracted the 1127 *High* users from dataset and tried to establish the most influential attributes that may be used to predict usage.

4.4.2 Predicting User Usage

The feature matrix we created for the purpose of predicting user usage from content features was quite dense. The whole dataset of 1127 users was used in the experiments with 10-fold validation. We still had to normalise the data to rationalise the rating scales. As can be seen from table 4.10, some attributes are years, while others are gender with values of either 1 or 2. To establish the dominant attributes, we performed a few experiments using a couple of algorithms from the ones listed below:

- Decision Tree Algorithms
    1. C4.5 Algorithm
- Statistical Meta-Algorithms
    1. Logistic Regression

Since the output attribute, *user_status*, was discrete, using linear regression would have been meaningless. Categorical input attributes have already been numerically coded in Chapter 3 using the complex ranking modelled by equation 3.12. Logistic Regression used took the form of $y = f(x)$, where, y is a categorical variable and x is a set of both continuous and numerically coded categorical variables. Three models were built to predict the three user statuses: *live, cancelled,* and *suspended*. Live members were members who are subscribed to the GetMore programme while cancelled ones had opted to unsubscribe from the programme. Suspended members were those who had failed to meet their cellphone contractual payments or who were in arrears in their payments. At any given time a user could be *live* or not, *cancelled* or not, *suspended* or not

For this exercise, we used a dataset with 12500 records and six input attributes. We used *R* programming to manipulate the data. We loaded the data into the workspace using the *read.csv()* command:

```
training.data.nom <- read.csv('lognom.csv',header=T,na.strings=c(""))
```

Figure 4.6 below shows a summary of both the input and output variables after they had been loaded onto the *R* workspace:

```
> training.data.nom <- read.csv('lognom.csv',header=T,na.strings=c(""))
> summary(training.data.nom)
    birthMonth         age          gender              province
 September: 6649   42     : 2250   22  :    1   Gauteng     : 7557
 March    : 6495   32     : 2184   45  :    1   Western Cape: 2906
 January  : 6442   39     : 2152   F   :31125   KZN         : 2758
 June     : 6407   40     : 2120   M   :44023   Limpopo     : 2636
 December : 6324   29     : 2118   NA's:    2   Eastern Cape: 1584
 (Other)  :42833   30     : 2096                (Other)     : 2761
 NA's     :    2   (Other):62232                NA's        :54950
  numRequests       avgRequestInterval  status
 Min.   :   0.000   Min.   :  0.0      CANC:39258
 1st Qu.:   1.000   1st Qu.:  0.0      LIVE:25989
 Median :   2.000   Median : 10.0      M   :    2
 Mean   :   4.112   Mean   : 36.1      SUSP: 9901
 3rd Qu.:   5.000   3rd Qu.: 50.0      NA's:    2
 Max.   :1151.000   Max.   :693.0
```

FIGURE 4.6: Summary of the raw variables

Age had the most number of missing values:

```
> sapply(training.data.nom,function(x) sum(is.na(x)))
        user_birth_month                      age                   Gender
```

```
                     0                    113                      0
              province             numRequests average_request_interval
                     0                      0                      0
           user_status
                     0
>
```

A casual glance at the unique summaries below shows that the *user_birth_month* data needs to be cleaned. We should not have 13 months:

```
> sapply(training.data.nom, function(x) length(unique(x)))
       user_birth_month                    age                 Gender
                     13                     74                      2
              province             numRequests average_request_interval
                     10                    128                   8620
           user_status
                      3
>
```

We went on to split the data into a *Training* and a *Testing* dataset on a ratio of *20000:4500*:

```
>
>
> train <- training.data[1:20000,]
> test <- training.data[20001:24500,]
>
```

We then built our first model for the *Live* users by calling the *glim()* function in *R*:

```
model <- glm(user_status ~.,family=binomial(link='logit'),data=train)
```

We viewed the results of the *Live* model by using the *summary()* function:

```
> summary(model)
```

```
Call:
glm(formula = user_status ~ ., family = binomial(link =‘‘logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4659  -0.7564  -0.6383  -0.5098   2.1103

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)               5.7386545  0.0880759 -31.094  < 2e-16 ***
user_birth_month          0.0099115  0.0048787   2.032   0.0422 *
age                       2.0334922  0.0012789  26.189  < 2e-16 ***
Gender                   -1.0420588  0.0342905  -1.227   0.2200
province                 -0.0124995  0.0077657  -1.610   0.1075
numRequests              -0.0004031  0.0010128  -0.398   0.6906
average_request_interval  0.0023826  0.0003460   6.887 5.71e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22078  on 19892  degrees of freedom
Residual deviance: 21302  on 19886  degrees of freedom
  (107 observations deleted due to missingness)
AIC: 21316

Number of Fisher Scoring iterations: 5

>
```

Analysing the results above shows that of all the significant variables, a unit increase in age increases the log odds by 2.03 while being female reduces the log odds by 1.04.

We then used the McFadden R^2 index to assess the model fit:

```
> library(pscl)
> pR2(model)
```

```
          llh       llhNull            G2      McFadden          r2ML
       -422.06508  -523.110162  491.016097    0.460714          0.331122
         r2CU
       0.58946
>
```

The above steps were carried out for all 3 datasets: *live data, cancelled data*, and *suspended data*. After evaluating the fit of the model we had to assess how good the model was when predicting *user status (y)* on a new dataset. The output probabilities of the prediction testing were in the form $Pr(y = 1|X)$ with a decision boundary of 0.5. If $Pr(y = 1|X) > 0.5$ then *y=1* (user status is live) otherwise *y=0*. For prediction accuracy testing we used our *test* data:

```
>
> training.results <- predict(model,newdata=test,type='response')
> training.results <- ifelse(training.results > 0.5,1,0)
>
```

We then calculated the Classification error of the model and subtracted it from one to get the accuracy ratio:

```
> classificationError <- mean(training.results != test$user_status)
> print(paste('Accuracy',1-classificationError))
[''Accuracy 0.535468"
>
```

The 0.54 accuracy on the test dataset is a good result. However, it is dependent on the manual *20000:4500* split of the data that we made earlier , therefore if we want a more accurate prediction score, we have to do some cross-validation such as *10-fold* cross validation.

We then plotted a ROC curve to calculate the AUC as a performance measure. The ROC curve was plotted using the following call to the *R* programme using test data:

```
library(ROCR)
p <- predict(model, newdata=test, type=''response")
pr <- prediction(p, test$user_status)
prf <- performance(pr, measure = ''tpr", x.measure = ''fpr")
plot(prf)
```

The resulting curve is illustrated in figure 4.7 below:

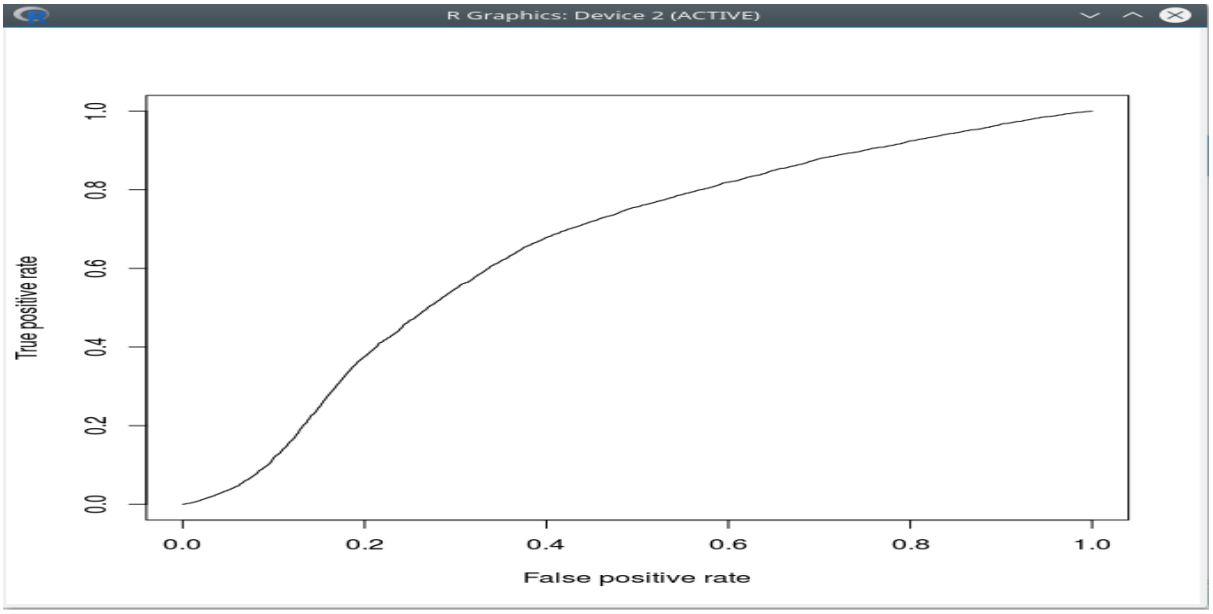


FIGURE 4.7: Area under the ROC Curve

To get the exact AUC in *R* we called:

```
>
> auc <- performance(pr, measure = ‘‘auc")
> auc <- auc@y.values[[1]]
> auc
[1] 0.6557673
```

To create our *K-Means* clustering example for this study, we used a subset of the dataset. We clustered the dataset using two variables, age and number of requests:

```
> dat = getMoreData[,c(3,4)]
> plot(dat, main = "Age vs Number of Requests", pch =20, cex =2)
>
```

The results are plotted in figure 4.8 below:

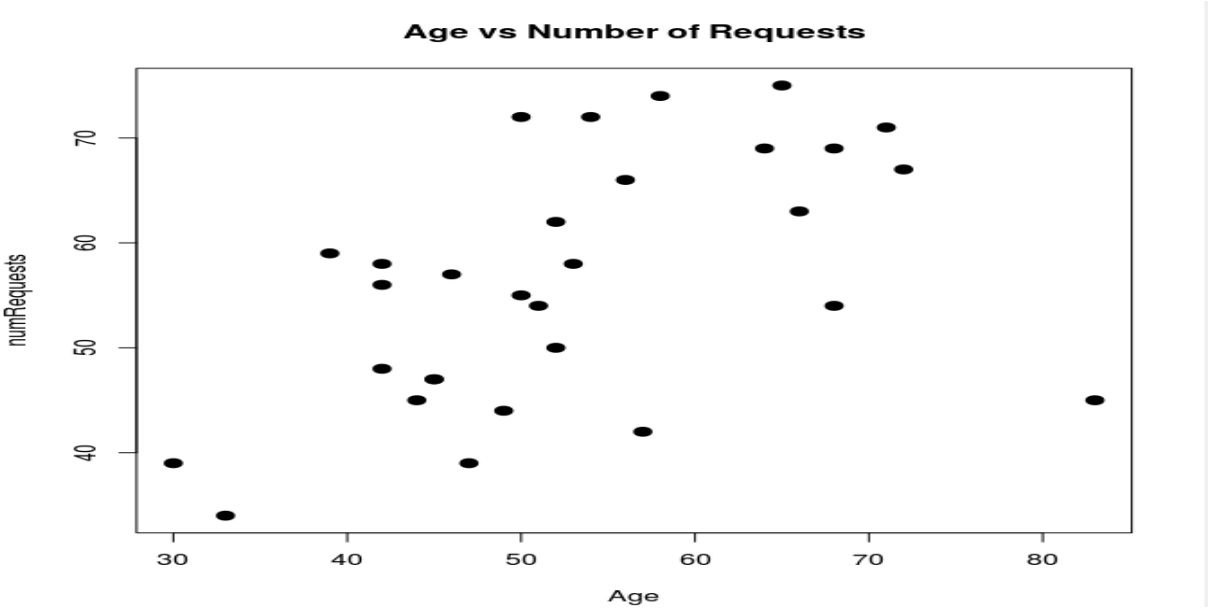


FIGURE 4.8: Age vs Number of Requests

Although this is just a small subset of the dataset, but we can see a positive correlation between age and the number of requests logged. After clustering the data using two clusters and user_status as the colour the results are illustrated in figure 4.9.

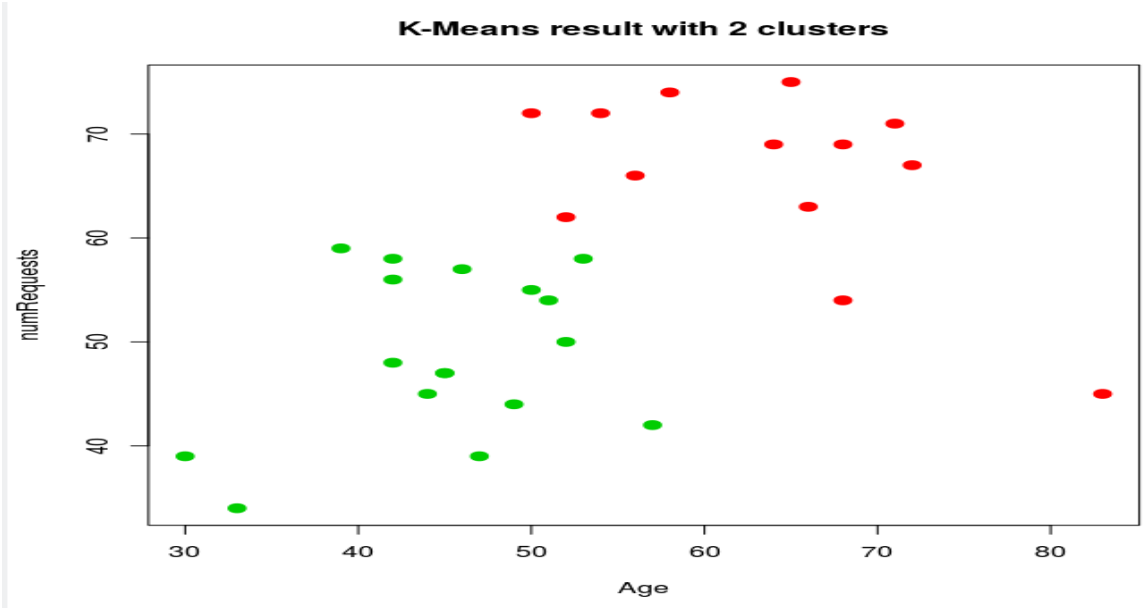


FIGURE 4.9: Age vs Number of Requests

The green dots represents the users who are no longer active.

When we used the algorithm to determine the number of clusters, we got the following cluster assignments:

```
K-means clustering with 5 clusters of sizes 4, 2, 8, 6, 8, 2

Cluster means:
       Age numRequests
1 79.50000      76.000
2 88.50000      59.500
3 38.62500      65.250
4 56.66667      79.000
5 41.50000      26.500an

Clustering vector:
 [1]  5 4 3 1 3 5 5 4 3 5 3 3 2 1 1 4 4 5 2  5 3 5 3 4 1 3 4 5

Within cluster sum of squares by cluster:
[1] 123.0406 345.1890 123.2136 232.6502  27.1030
 (between_SS / total_SS =  79.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[5] "betweenss"    "size"         "iter"         "ifault"
>
```

The models performed differently for *Cancelled* and *Suspended* users. The dataset was dominated by the cancelled users, so the modelling results for the cancelled user had a better accuracy ratio of 0.85 and an AUC of 0.89, while data for the suspended user was inconclusive.

With the C4.5 algorithm, we were forced to include the other classes: *Low users* and *Medium users*. We did this so as to produce a multi-node visualisation of a decision tree easily. To make the algorithm work better, the highly-ranked attributes like age, and join_date were manually categorised. We identified 3 major age-groups by using a frequency table with trial-and-error. These age-groups were *less than 30, 30-55* and *over 55*. Since the majority of the users had joined in 2011, we categorised them in *before2011, after2011*, and *in2011*. This enabled us to use the influential predictors as splitting attributes. C4.5, on the other hand, singled out *avg_request_interval* as the root of the decision tree followed by *age* and the *join_date*. The most significant correlation between the number of requests and the average request interval was observed where the average interval was 29 days. This was a clear indication of the dominance of month end users in the dataset.

In terms of accuracy, the C4.5 algorithm, performed marginally poorly when compared to LR. Its RMSE was 82.65% while MAE was 16%. Figure 4.14 below illustrates the area under ROC for the 3 classes of users using C4.5:

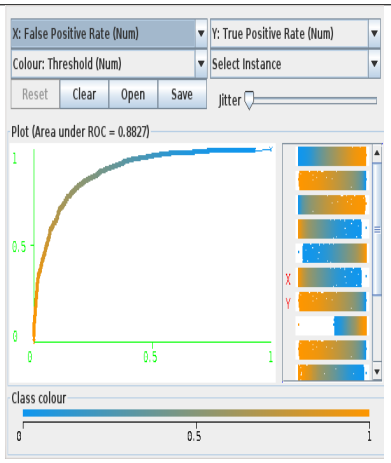
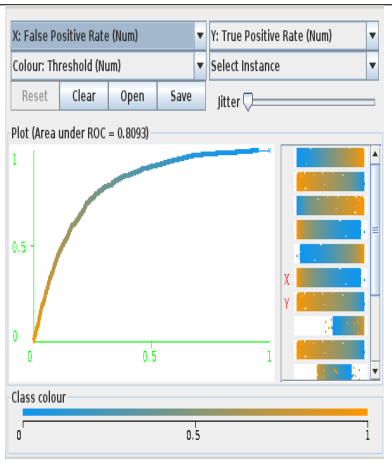
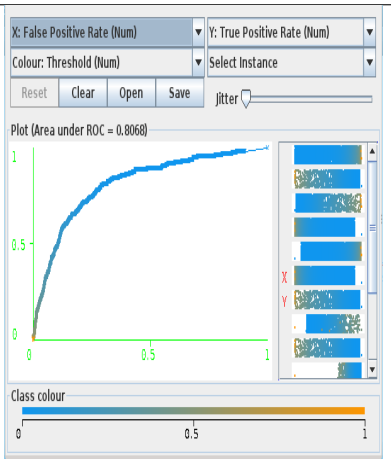
High	Medium	Low
		

TABLE 4.14: Area Under ROC for the User Classes

The area under ROC was 0.88 for the High users and about 0.81 for both the Medium and Low users.

The average Entropy (approximately 0.496) of the tree is calculated by equation 4.11, where c are the different predictors and H is a vector of those predictors.

$$E(U, H) = \sum_{c \in H} P(c)E(c) \tag{4.11}$$

The value of 0.496 meant the sample was neither perfectly distributed nor homogeneous.

From the results obtained from the algorithms highlighted above, we concluded that a user's age and length of membership were the most important attributes together with gender. Male users who were in the 50-65 age-range were the highest users followed by the 35-50 age-group where neither of the sexes dominated. The 30 and below age-group was the worst in terms of usage. These results indicated that attributes could be used to identify an active user's nearest neighbours who could be used to recommend items of interest to him.

4.4.2.1 K-NN with SVD on Content Features

In order to provide relevant predictions, we had to create a neighboughood cluster for the active user from which we had to draw items for recommendation. The decision rules derived from the

C4.5 algorithm and the weights we got from LR were used to determine the similarity of users.

Content-based filtering(CBF) used user attributes to cluster users. User attributes were treated as vectors for each user row. Singular Value Decomposition (SVD)was used to approximate the missing attribute values based on the matrix factorization ($\hat{r}_{u,i} = (U_k S_k^{\frac{1}{2}})_u.(S_k^{\frac{1}{2}} V_k^1)_i$). The accuracy and precision of the extrapolated attributes varied directly with the dimensions of the decomposed matrix.

Very short singular vectors did not provide adequate explanatory power to differentiate the appropriate users. On the other hand, too long singular vectors led to over-fitting.

The optimal accuracy performance was around $k = 10$ to 25 Throughout the study the dimension k was fixed to 20, which is the approximate mean of the best performing features.

Dimensions were not the only determinant factor in the accuracy of the recommender system. Different user attributes had varying influence on the accuracy of the system. Most of these attributes like age and gender had already been identified as the key features. The contribution of each attribute was assigned a weight. The feature matrix was then multiplied by the ascertained weight value which increased the feature values. However, when set too high, the weights tended to outweigh the rating information. Our aim was to try and establish if certain user traits were linked to certain products or usage. The Java implementation of the SVD algorithm is given in Appendix B.1. The Cosine Similarity measure (See equation 4.3) was used to calculate user similarity. For each user, we selected a set of 10 nearest neighbours and from those 10 we only selected items that the active user had not yet rated to be recommended.

The advantage at this stage was that we had very few attributes to work with after the classification activities of the previous section. We used all the 3381 users who had been sampled for classification with the dataset split 70% for training and 30% for testing.

Content-based Item Similarity

Content-based item similarity computation used content attributes to identify item similarity. The only item attribute available was item type. In the database, the item type has the following structure. The methods used to satisfy the requirements of this sub-question make predictions using available ratings given by the same user on similar items. The advantage of item-oriented methods was that the users were more familiar with items they had previously preferred than those liked by potentially like-minded users.

id	key	name	type
1	3253	Telewheels - Cell C	Telewheels, car hire, taxi, bus service
2	3304	Tax Expert	Financial Services, financial assistance, business assistance
3	3805	Web Design	IT Services, website, internet, ADSL, hosting
4	4823	Washing machine	Electrical Goods, electrical gadgets, stoves, washing machine, refrigerator
5	4827	Accommodation	Onboarding, information, enquiry, tax services

TABLE 4.15: Items Table Extract

For this subsection, we used 1107 unique items which had already been used before for training, while the remaining 500 were used for testing. To construct item-item similarity, item attributes(item types) were converted into vectors using the following piece of code extract:

```
Map<Long,ImmutableSparseVector> itemVectors = new HashMap<Long,
   ImmutableSparseVector>();
        for (Map.Entry<Long,Map<Long,Double>> entry: itemData.entrySet()) {
            MutableSparseVector vec = MutableSparseVector.create(entry.getValue());
            itemVectors.put(entry.getKey(), vec.immutable()); }
        return itemVectors;   }

```

The vectors are then mapped to the users through the user preferences table. Due to the dearth of item attributes, item similarity for existing items had to use cosine similarity based on previous users preference. A group of items sharing the similar item types was considered even more similar if it had similar ratings from similar users.

```
       Map<Long, ScoredIdListBuilder> itemSimilarities = new HashMap<Long,
   ScoredIdListBuilder>();
        for (long firstItem: items) {
            for (long secondItem: items) {
                if (firstItem == secondItem) continue;
                ImmutableSparseVector firstRatings = itemVectors.get(firstItem);
                ImmutableSparseVector secondRatings = itemVectors.get(secondItem);
                double similarity = cosineVectorSimilarity.similarity(firstRatings,
   secondRatings);
                if (similarity <= 0) continue;
                ScoredIdListBuilder currSimilarityList = null;
                if (itemSimilarities.containsKey(firstItem))
                    currSimilarityList = itemSimilarities.get(firstItem);
                else
                    currSimilarityList = ScoredIds.newListBuilder();
                currSimilarityList.add(secondItem, similarity);
                itemSimilarities.put(firstItem, currSimilarityList);       }
        }
```

The resultant item similarity table in the database included all similar items and a similarity score which ranged between 0 and 1. The most similar users are selected based on the similarity score given in table 4.16.

itemID1	itemID2	similarity
3253	3806	0.6540
3253	3309	0.9000
3253	4827	0.0822
4827	4815	0.8762
4827	4829	0.0231

TABLE 4.16: Item-item Similarity in the Database

The formal statement of the adjusted Cosine Similarity has already been detailed in sections 4.3.2 and 2.7.5.

4.4.3 CBF Recommendation

When making recommendations, the item-based recommender selected *top-k* most similar items to the items the user had preferred before. The items in table 4.16 were ranked in terms of similarity in a descending order and then the top most similar items were selected. In this study, k was set to 10.

In a similar fashion, the user-based recommender selected a neighbourhood of users who were most similar to the active user. From that group of users, it selected the most common itemset that that had the highest ratings and recommended the top-n items to the active user.

4.5 Predicting Membership Churn

Customer attrition was one of the major challenges facing the organisation. This was not limited to the number of customers who cancelled the service year-on-year, but also the short time members took in the service before deciding to cancel and the increasing number of refunds and complaints was increasing. Of the 90000 users in this dataset, about 60000 of them had cancelled the service within 12 months of joining it. Figure 4.10 below show the absolute figures of users who cancelled (red), who are live (blue) and those suspended (light blue) in each age-group.

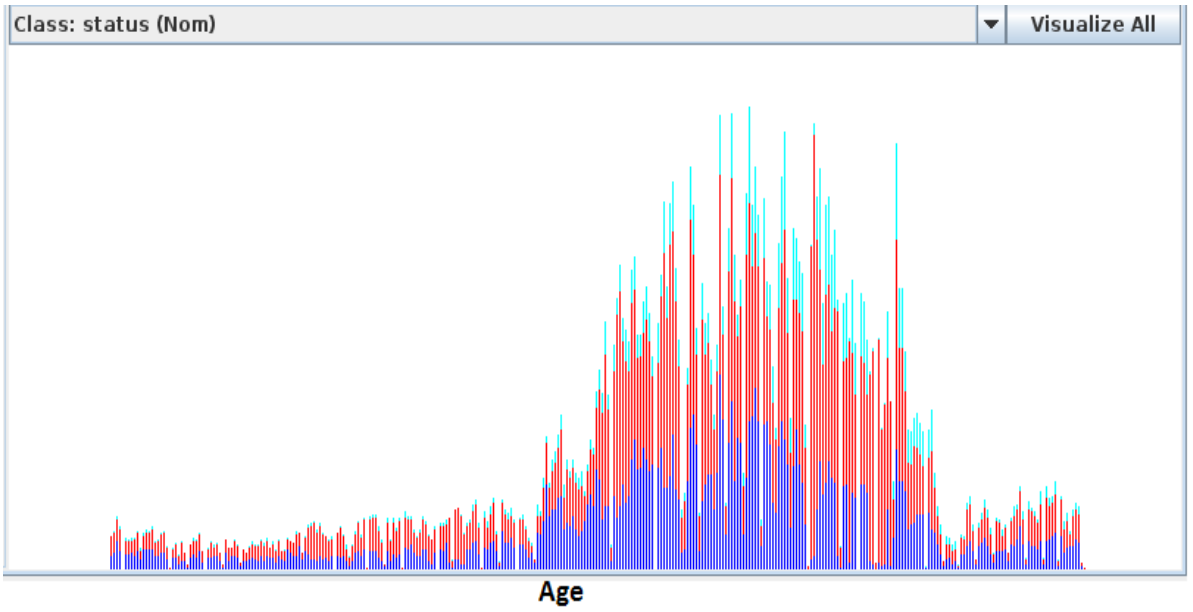


FIGURE 4.10: User Churn Against User Age

The picture was a cause for concern. system developed during this study had to enable the organisation to profile members who cancel the service in order to improve user retention.

Formally stated, the question addressed in this section was:

> *What is known about PLP clients who cancel their subscription to a particular service? Can such information be used to predict service cancellation, and lead to preemptive measures to reduce the number or likelihood of subscriber cancellations?*

This question required us to look for answers from both usage data and content attributes to predict customer loss (churn). To answer the question we had to be able to profile the type of user who cancelled then we had to identify similar users who were still using the service (Live users). Before doing anything further, we had to be sure we could safely classify users as *Live, Cancelled (Canc)* or *Suspended (Susp)* using their content attributes only. According to Figure 4.11 below, using a sample data of 24583 users, we can accurately classify 71% of the instances which is a good figure.

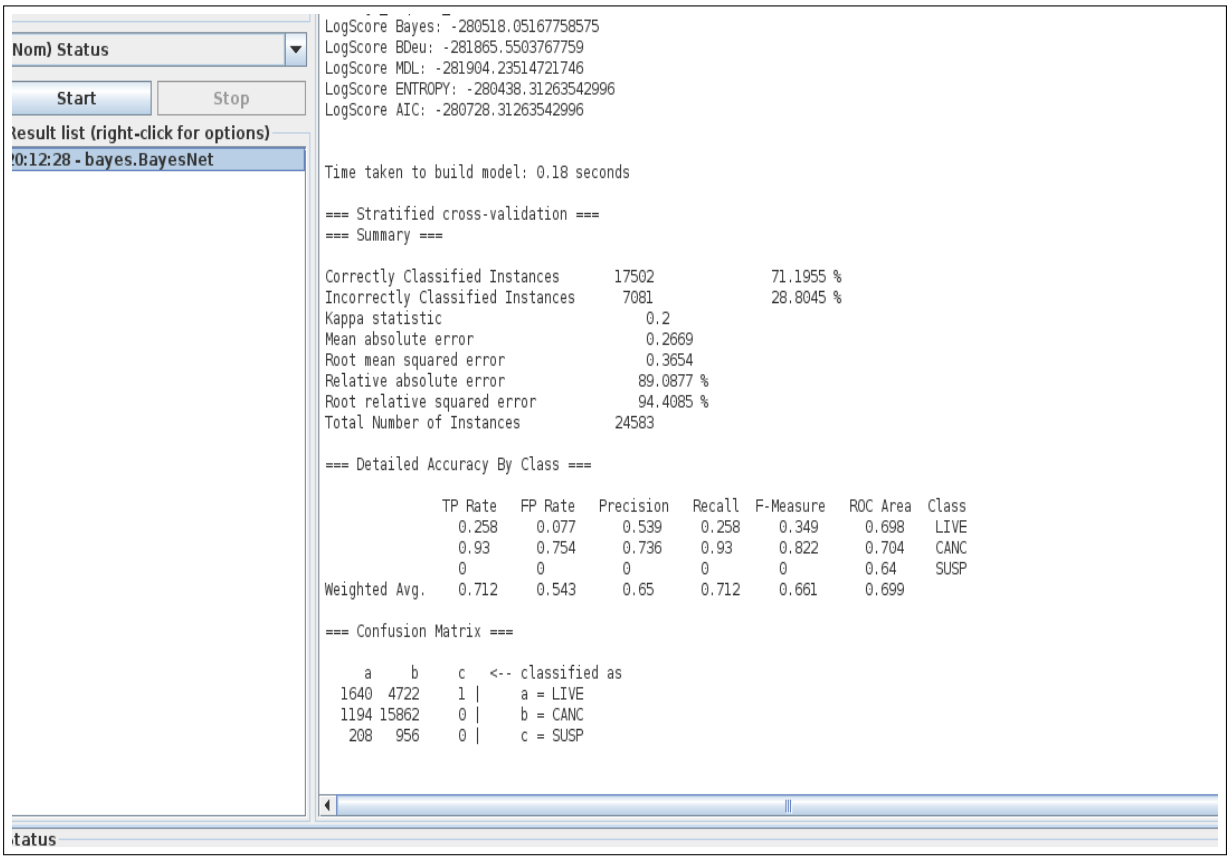


FIGURE 4.11: Bayes Network Classifier

After proving that users could be safely classified according to status using content features, we proceeded to predict user status.

4.5.1 Content Features with SVD and Linear Regression

In this exercise, we used a total 24583 re-balanced samples with 10 fold cross-validation. First, we constructed our user-feature matrix using content attributes. Then we applied SVD (see section 4.2 and fed the result to WEKA to compute feature weights using Linear Regression.

After that, we split our dataset to *Live* and *Cancelled* members. For each Live member, we computed a "likelihood-to-Cancel" score, a similarity measure with cancelled members.

The results of the Linear Regression Algorithm were as follows:

```
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===



Linear Regression Model

user_status =
     -0.7156 * age +
      0.6279 * getmore_join_year +
      0.6058 * province +
     -0.6002 * numRequests +
     -0.5859 * average_request_interval +
    364.4153

Time taken to build model: 0.35 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.6863
Mean absolute error                      0.4423
Root mean squared error                  0.5511
Relative absolute error                 64.6712 %
Root relative squared error             67.4032 %
Total Number of Instances               24583
```

We applied the weights provide by Linear Regression to the reduced user-feature matrix. A value of 1 meant the member was Live, 2 was Cancelled and 3 suspended. The sample prediction results from the algorithm were as follows:

```
=== Predictions on test data ===

-----------------------------------------
inst#,    actual,   predicted,     error

-----------------------------------------
     1      3          2.632     -0.268
     2      1          1.461      0.461
     3      3          2.577     -1.023
     4      1          1.189      0.189
     5      3          2.82      -0.18
     6      3          2.932     -0.068
     7      1          2.641      1.641
```

The `actual` variable in the results indicates the actual user status, while the error is the difference between the actual value and the predicted value. The sum of prediction errors in the above example was small. Again, the Linear Regression algorithm affirmed what we already knew that *user_status* was heavily correlated to age.

4.5.2 Churn Prediction using Rating Data

Having established that it was possible to predict user churn using content features, we had to try the same procedure with rating data. This presented us with 2 major challenges we did not face when we used content features:

- The rating matrix was too sparse; the rating fill-rate was less than 25%.
- There were too many items making the matrix too wide.

To complete missing rating data we used rating interpolation as described by equation 4.6. Known ratings were used to estimate unknown ratings. For each rating gap, the estimated value was informed by the user's known ratings, the item's ratings and the total ratings in the matrix. After all the values had been filled, normalised the matrix and then applied SVD to reduce dimensionality. The reduced matrix had 4205 instances. Like in the previous section, we applied the Bayes Network algorithm just to check if it is possible to reasonably classify the dataset according to user status using rating information. The resultant Confusion Matrix was as follows:

```
=== Confusion Matrix ===

    a    b    c   <-- classified as
 1318   19   40 |    a = 1
    5 1441  196 |    b = 2
   94   38 1053 |    c = 3
```

Out of the 4205 users, 3812 were correctly classified thus giving us an accuracy rate of over 90%. That was not right as it raised the problem of over-fitting. We took a closer look at the data to try and identify the problem. The majority of the users who cancelled the service in the sample requested nothing else except cancellation-related items. Similarly, the users who were still active requested items that had nothing to do with cancellation of service. That explained why the algorithm performed so well, but the objective had been achieved; user preferences could be used to classify users according to their status.

To predict user status we then calculated user similarity using Cosine similarity and used the similarity scores as attribute weights in predicting the user status. The status column was binarized and a user was considered to be either active or not. The following algorithm was used to predict user status:

$$pred(u,i) = \frac{\sum_{\in rateditems(u)} itemSim_{(i,j)}.r_{u,j}}{\sum_{\in rateditems(u)} \left|itemSim_{(i,j)}\right|} \tag{4.12}$$

Where, u is the active user we intend to predict his rating for item i. In this algorithm, user status was included as an item with a correlation coefficient of 0.98 to another item which was available for selection called *cancellation*

The final confusion matrix for this experiment is presented below

a	b	<---- classified as
1260	693	a = Live
156	2094	b = Canc

TABLE 4.17: Usage Churn Confusion Matrix

The algorithm had a prediction accuracy of 80% which was too high. However, with proper modifications, we could get a proper accuracy figure from the data.

We then combined the user features together with the rating data and the summary of the results is presented below:

	Correctly Classified (%)	RMSE (%)	MAE (%)	ROC		
				Live	Canc	Susp
Content Features	90	36.65	10.37	0.88	0.82	0.96
Rating Data	80	52.97	20.64	0.67	0.79	—
Combined	65	50.64	35.99	0.65	0.81	0.64

TABLE 4.18: Summary of User-Churn Results

The results showed that it was possible for the organisation to profile its users according to likelihood to cancel the service. They also showed that it was possible to predict with a degree of reliability which users were about to cancel the service by using the content features, rating data, or both.

4.6 Evaluating Prediction Algorithms

In the previous section, we attempted to prove that we could predict user ratings and classify users by a given classifier using both content features and user preferences. In this section we want to identify the best algorithm to predict user ratings. As already shown in Figure 2.1, in literature, classification is considered as a sub-class of prediction. In practice, we evaluate classification and regression algorithms when we evaluate prediction. Prediction algorithms are evaluated for:

- Accuracy
- Speed
- Robustness
- Scalability
- Interpretability

A good classifier should be able to accurately predict the class label correctly and the accuracy of the predictor measures the percentage of correctly classified instances against incorrectly classified one. Speed or computational complexity is another important evaluation metric. A good classifier should be able to produce good results with little computational cost. Robustness involves the ability to handle dirty data while scalability speaks to the efficiency of the classifier when handling larger datasets. Most classifiers present their results in the form of decision trees, induction rules or time taken to develop the model. Other options are usually left to the discretion of the user.

In this section we attempted the sub-question:

Which data mining techniques are best able to predict or determine whether subscribers will like or dislike a particular item or service offered by the PLP Group?

In the previous section, we used classification to predict user profitability and user churn. We provided data to prove that prediction was the best technique to identify similar users, items and estimate a rating that a user could give to an item.

In this section, we evaluated all the algorithms we used to predict user ratings, status and usage. For this exercise, we used a dataset of 24583 records with a 10-fold cross-validation test mode. We tried to keep all the parameters the same for the algorithms when monitoring their speed. The algorithms evaluated were C4.5 (J48), Regression Classifier, Naive Bayes Classifier and the Bayes Network Classifier.

Before we display the results we obtained, we would like to introduce the evaluation algorithms used in this experiment.

4.6.1 Evaluation Metrics

To validate if an algorithm satisfied any of the criteria laid above, we used MAE, RMSE and ROC, TP-rate and Recall to measure their performance.

Root Mean Squared Error(RMSE)

The RMSE estimates the deviation of the actual values from the regression (mean) line. To calculate the RMSE we had to calculate the error for each instance first, and then we squared the the *error* values. We then added up together the squared errors and calculated their square root.

inst#	actual	predicted	error
1	3	2.632	-0.268
2	1	1.461	0.461
3	3	2.577	-1.023
4	1	1.189	0.189
5	3	2.82	-0.18
6	3	2.932	-0.068
7	1	2.641	1.641

TABLE 4.19: Classification Predictions

Using the example given in table 4.19, RMSE would be: $RMSE = \sqrt{\frac{1}{|R|}\Sigma_{(u,i)\in R}(\hat{r}_{u,i} - r_{u,i})^2}$
$RMSE = \sqrt{-0.268^2 + 0.461^2 + (-1.023^2) + 0.189^2 + (-0.18^2) + (-0.068^2) + 1.641^2} \approx 1.336$

Mean Absolute Error(MAE)

Although the Mean Absolute Error(MAE) is commonly used to forecast error in time series analyses, we used it here because most of the items were requested sequentially. Equation 4.13 shows how the MAE is calculated [Sarwar *et al.*, 2000].

$$MAE = \frac{\Sigma_{i=1}^{n}|p_i - q_i|}{n} \tag{4.13}$$

where p is the predicted score and q is the observed score. In terms of user rating prediction, the lower the MAE the more accurately the recommender system predicts user ratings, which means that there is a negligible difference between the predicted and observed results. In the example given in table 4.19 above, the shortest way to calculate MAE would be to add all the values in the *error* column and divide them by 7, the number of records. MAE would be about 0.75. MAE was used to evaluate all the algorithms used here.

Precision and Recall

These two are self-explanatory. Precision is the ratio of relevant selected items and the number of selected items [Euzenat, 2007]. Both Precision and Recall attempt to measure the relevance of the chosen algorithm.

$$[H]Precision = \frac{|B_{rs}|}{|B_s|} = P(A,R) = \frac{|R \cap A|}{A} = \frac{correctly\ recommended\ items}{total\ recommended\ items} \tag{4.14}$$

The recall measure is defined in Equation 4.15.

$$[H]Recall = \frac{|B_{rs}|}{|B_r|} = R(A,R) = \frac{|R \cap A|}{R} = \frac{correctly\ recommended\ items}{total\ useful\ recommendations} \tag{4.15}$$

4.6.2 Evaluation Results

The summary accuracy performance values of the five algorithms used in this section of the study is given in the table below:

Algorithm	RMSE	MAE	TP Rate	Time(s)	ROC	Accuracy
C4.5 (J48)	0.65	0.39	0.67	0.42	0.71	67.13
Bayes Network	0.48	0.30	0.69	0.35	0.75	68.67
Naive Bayes Classifier	0.35	0.25	0.82	0.05	0.85	83.02
Linear Regression	0.46	0.28	0.78	0.18	0.82	78.20

TABLE 4.20: Collaborative Filtering Usage Accuracy Matrix

We used 25853 records with both user features and rating data. Missing values were computed using the algorithm explained in equation 4.7. The Matrix was then converted into a binary Matrix by the subtractive normalisation procedure and SVD was applied for dimensionality reduction. The classification algorithms were then applied to the dataset to predict usage and status and their results observed. The above metrics were common to all the algorithms. The Naive Bayes Classifier performed best followed by the Regression Classifier.

4.7 Association Rule Mining (ARM)

In recommending items to users, the organisation was also interested in finding out which item groups were requested together by different users. The classical market-basket algorithm assumes that there is a large number of items with which users fill their market baskets and our challenge is to establish those items that put together in one basket by most customers. The organisation intended to use this information to position items for recommendation.

The question we sought to answer in this section is formally stated below as:

> *Which of the products or services offered by the PLP Group should be recommended together or in combinations?*

The advantage of association rule algorithms over decision tree algorithms like J48 (C4.5) and ZeroR is that associations are multi-directional and can exist between any of the attributes. A decision tree algorithm will build rules with only a single conclusion focused on the class attribute, whereas association algorithms attempt to find multiple rules, each of which may

have a different conclusion. However, one standout disadvantage of association algorithms is that they tend to find patterns within large datasets and usually require much more time and processing power to run when compared to a decision tree algorithm.

The dataset we used in this section for ARM had 124 distinct items, 4205 transactions, 496 maximum transactions, transaction size was 1753 and the average transaction size was 22.

4.7.1 Mining Frequent Itemsets

First, we created a rating Matrix made up of users and item preferences. The preferences were counts of the number of times the item has appeared in the customer's transactions. This meant that all of the 4205 customers used in the sample had only one transaction. The dataset was one big Matrix (T) of many transactions (t) . Each transaction ($t \in T$) was made up of lists of items (i) such that: $t_u = \{i_1, i_2, i_3, \ldots, i_n\}$ is the transaction for user u in the Matrix M. As already indicated in Section 3.6.4.

Mining frequent patterns was quite a challenge due to the following reasons:

- Data sparsity: Few meaningful patterns existed in the dataset such that to get any associations we were forced to use larger datasets. Algorithms like the Apriori took longer to compute and were heavy on the computing resources.
- Bundled items like *Enquiry about service; member called for info;* and so on dominated the patterns leading to service cancellation.

Some of these challenges related to the dataset that we had our disposal, while others were a result of the nature of the data flow processes in the organisation. Challenges related to the dataset included a bias caused by the dominance of cancelled members in the dataset. Cancelled members represented user dissatisfaction with the whole service which led to the dominance of cancellation-related items over other items.

Business data flow process related challenges involved double counting of service requests caused by lack of proper work-flow management. Follow-up requests for items and requests overriding previous requests were entered as new requests.

Some of the challenges faced are given in section 4.10.

For the purposes of mining association rules, the user-item matrix was binarized into zeros and ones as shown in the sample in Figure 4.12 below.

	A	B	C	D	E	F	G	H	I
1	2-4-1 Deals	Legal Assist	Doctor Booking	Finance Assist	Medical Assist	Car Spares	TV Repairs	Accommodation	Teacher-on-Call
2	0	1	0	1	0	0	1	0	0
3	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	1	1	1	0	1	0	1	0	0
6	0	0	1	0	0	0	1	0	0
7	1	0	0	0	0	1	0	0	0
8	0	1	0	0	0	0	0	0	0
9	0	1	0	0	1	0	0	0	0
10	1	0	0	1	0	0	0	0	0
11	1	1	1	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0
13	0	0	1	0	0	0	1	0	0
14	1	0	0	0	0	1	0	0	0
15	1	1	0	1	1	1	0	0	1
16	1	1	1	0	0	0	0	0	0
17	1	1	1	0	0	0	1	0	0
18	0	0	1	0	0	0	0	0	0
19	0	0	1	0	0	0	0	0	0
20	1	1	1	1	1	1	1	0	0
21	1	1	1	0	0	1	0	0	0
22	0	1	0	0	0	0	0	0	0

FIGURE 4.12: Binary Rating Matrix

The ones indicate that an item was a part of the transactions whereas zeros indicate that it was not there.

4.7.2 ARM using the Apriori Algorithm

The Apriori algorithm brought the system down several times during its implementation. We were forced to reduce the amount of data available to the algorithm to generate association rules. The revised dataset was as follows:

Transactions	Items	Association Rules	Minimum Support	Minimum Confidence
1150	50	10	90	75

TABLE 4.21: Apriori Dataset

The rules from the algorithm were in the form: Given an antecedent item(s), a confidence percentage, and consequent item, the rule will be:

> *If all the antecedent item(s) have been purchased, then we can say, with the Confidence percentage, that the consequent item(s) will also be purchased.*

A snap view of some of the top rules generated by the Apriori are given in table 4.22 below:

No.	Confidence	Antecedent(A)	Consequent(C)	Support (A)	Support (C)
1	94.8345%	2-4-1 Deals & Medical Assist	Doctor Booking	180	326
2	92.0769%	2-4-1 Deals & Legal Assist&Medical Assist	Doctor Booking	170	230
3	90.0034%	2-4-1 Deals & Legal Assist&TV Repairs	Doctor Booking	117	223
4	88.0980%	2-4-1 Deals	Doctor Booking	153	200
5	88.0043%	2-4-1 Deals & Car Spares	Doctor Booking	115	200
6	80.9878%	2-4-1 Deals & Legal Assist	Doctor Booking	162	186
7	80.9009%	2-4-1 Deals & Finance Assist	Doctor Booking	143	186

TABLE 4.22: Example Rules from the Apriori Algorithm

According to the best rule, we can say, with about 95% confidence, that users who request 2-4-1 Deals and Medical assist will also request a Doctor Booking. There were 180 transactions with the antecedent item combinations 326 transactions massively supported the consequent in the first rule. One explanation why 2-4-1 Deals dominated item requests was because it was a new product package with various items included in it.

4.7.3 ARM using the Predictive Apriori Algorithm

The Predictive Apriori algorithm was very fast and was considerably processing-friendly. Since it did not weigh heavily on the computing resources we decided to use the full dataset set aside for Association Rule Mining which had 124 distinct items, 4205 transactions, 496 maximum transactions, transaction size was 1753 and the average transaction size was 22. Sample results from the application of the Predictive Apriori Algorithm are given by figure 4.13 below:

```
PredictiveApriori
=================


Best rules found:

  1. Weather='(27.7-30.8]' 3 ==> Contact Details='(14.6-24.2]' 3    acc:(0.79997)
  2. Appointment='(-inf-20.2]' 2 ==> Contact Details='(14.6-24.2]' 2    acc:(0.74998)
  3. Banking='All' 2 ==> Legal Wellbeing='(-inf-9.9]' 2    acc:(0.74998)
  4. Bond Expert='(11.3-inf)' 2 ==> Car Spares='(-inf-8.2]' Vehicle Accessories='(6.3-7.6]' 2    acc:(0.74998)
  5. Building Materials='(19.4-21.2]' 2 ==> Cables='All' Car Sales='(-inf-7]' 2    acc:(0.74998)
  6. Building Materials='(19.4-21.2]' 2 ==> Cables='All' Car Spares='(-inf-8.2]' 2    acc:(0.74998)
  7. Building Materials='(21.2-inf)' 2 ==> 2 for 1 Deals='(-inf-34.5]' Car Spares='(24.2-27.4]' 2    acc:(0.74998)
  8. Building Materials='(21.2-inf)' 2 ==> 2 for 1 Deals='(-inf-34.5]' Ceilings='All' 2    acc:(0.74998)
  9. Cables='All' 2 ==> Building Materials='(19.4-21.2]' Car Sales='(-inf-7]' 2    acc:(0.74998)
 10. Cables='All' 2 ==> Building Materials='(19.4-21.2]' Car Spares='(-inf-8.2]' 2    acc:(0.74998)
 11. Car Spares='(24.2-27.4]' 2 ==> 2 for 1 Deals='(-inf-34.5]' Building Materials='(21.2-inf)' 2    acc:(0.74998)
 12. Car Spares='(24.2-27.4]' 2 ==> 2 for 1 Deals='(-inf-34.5]' Ceilings='All' 2    acc:(0.74998)
 13. Ceilings='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Building Materials='(21.2-inf)' 2    acc:(0.74998)
 14. Ceilings='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Car Spares='(24.2-27.4]' 2    acc:(0.74998)
 15. Chemical='All' 2 ==> Building Materials='(19.4-21.2]' Cables='All' 2    acc:(0.74998)
 16. Chemical='All' 2 ==> Building Materials='(19.4-21.2]' Car Sales='(-inf-7]' 2    acc:(0.74998)
 17. Compressor='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Building Materials='(21.2-inf)' 2    acc:(0.74998)
 18. Compressor='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Car Spares='(24.2-27.4]' 2    acc:(0.74998)
 19. Contact Details='(53-62.6]' 2 ==> Legal Wellbeing='(-inf-9.9]' School='(-inf-5.4]' 2    acc:(0.74998)
 20. Contact Details='(81.8-91.4]' 2 ==> Building Materials='(19.4-21.2]' Cables='All' 2    acc:(0.74998)
 21. Contact Details='(81.8-91.4]' 2 ==> Building Materials='(19.4-21.2]' Car Sales='(-inf-7]' 2    acc:(0.74998)
 22. Doors='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Building Materials='(21.2-inf)' 2    acc:(0.74998)
 23. Doors='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Car Spares='(24.2-27.4]' 2    acc:(0.74998)
 24. Electrical components='All' 2 ==> Building Materials='(19.4-21.2]' Cables='All' 2    acc:(0.74998)
 25. Electrical components='All' 2 ==> Building Materials='(19.4-21.2]' Car Sales='(-inf-7]' 2    acc:(0.74998)
 26. Entertainment='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Building Materials='(21.2-inf)' 2    acc:(0.74998)
 27. Entertainment='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Car Spares='(24.2-27.4]' 2    acc:(0.74998)
 28. Exchange rate='(-inf-279.1]' 2 ==> Contact Details='(14.6-24.2]' Enquiry about service='(-inf-43.7]' 2    acc:(0.74998)
 29. Exchange rate='(-inf-279.1]' 2 ==> Contact Details='(14.6-24.2]' Property='(-inf-5.9]' 2    acc:(0.74998)
 30. Flooring='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Building Materials='(21.2-inf)' 2    acc:(0.74998)
 31. Flooring='All' 2 ==> 2 for 1 Deals='(-inf-34.5]' Car Spares='(24.2-27.4]' 2    acc:(0.74998)
 32. Guttering='(7.7-inf)' 2 ==> Building Materials='(19.4-21.2]' Cables='All' 2    acc:(0.74998)
 33. Guttering='(7.7-inf)' 2 ==> Building Materials='(19.4-21.2]' Car Sales='(-inf-7]' 2    acc:(0.74998)
```

FIGURE 4.13: Part Results from Predictive Apriori

As can be seen from the snapshot, the Predictive Apriori algorithm does not provide *support* and *confidence* statistics but gives an *accuracy* value. In the first *rule*, the accuracy value was 0.79997. The results indicated an almost 80% accuracy for the rule that all requests for information about the weather forecasts were also followed by an update of the contact details. This may indicate that people who wished to visit another area on a specific date were interested in the weather conditions at that place and wanted all communications forwarded to them in that new area. Similarly, people who had an appointment needed to know how to get there.

The conclusion we could draw from these two algorithms were that the organisation could identify items that shipped together with up to 90% confidence.

4.8 Prototype Hybrid Recommender System Applications

Most of the prior work in the field of Recommender Systems is marred by the inability to address the *new-user, new-item* problem commonly called the *cold-start* problem. The issue emanates from the fact that most Recommender Systems rely on previous user preferences to make recommendations. They rely on either the candidate user's ratings for similar items or on the rating of other users, or both.

The other shortcoming of most of the good Recommender Systems out there is their cost in terms of computing resources. As a result, most of them tend to use either a subset of the dataset, or use expensive server-side infrastructure. The approach used in this study sought to,

among other things, produce accurate recommendations for both new and old users with little computing cost.

4.8.1 Cold Start Problem

In this study, we used both a user's personal and demographic attributes and usage data to determine *user-user similarity*. The use of personal and demographic attributes enabled us to predict user preferences with a high degree of accuracy in the face of missing user ratings. In this way, we did not have to rely on items previously rated by any of the users to determine user similarity. However, where rating information was available, recommendations were even more accurate. Figure 4.14 below shows items recommended by the system to a new user who has no previous usage data.

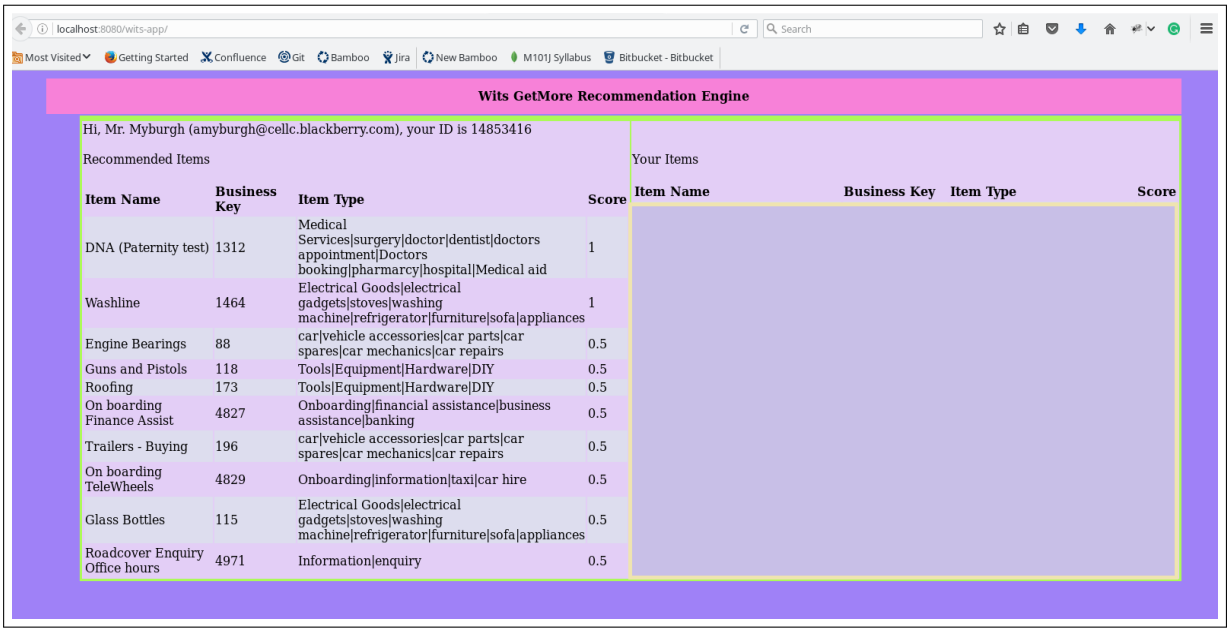


FIGURE 4.14: New User Screen

The user recommendations are based on the ratings and usage of users with similar personal and demographic attributes as the candidate user.

Similarly, items were also compared to each other based on item attributes which were not necessarily dependent on the users who had rated the items. However, the presence of usage and rating data made the item-similarity algorithms perform even better. This helped us to take care of the *new-item* problem in this study.In figure 4.14 above, *washline* is a new item that has never been rated or requested by anyone before yet it has been recommended because it shared similar attributes to the items that users who are similar to the candidate user have requested before. The screenshot only shows the top ten potential recommendations sorted based on their similarity score.

Figure 4.15 below also affirms the attributes of novelty and serendipity provided by the system developed in this study.

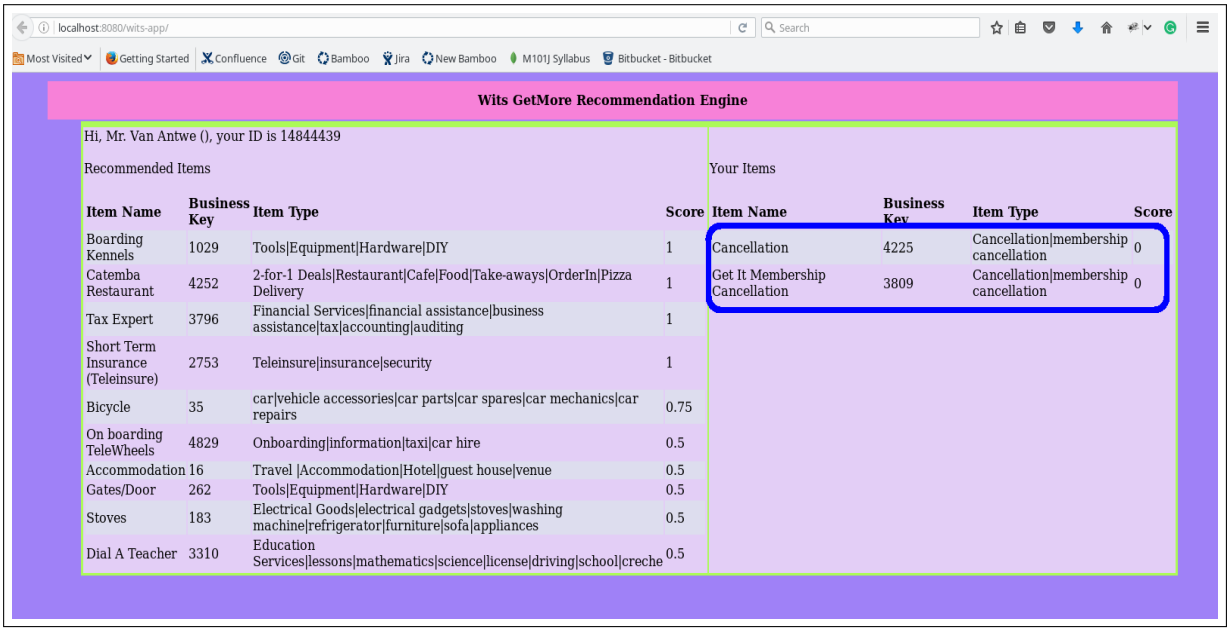


FIGURE 4.15: Recommending Items Unrelated to Rated Items

The candidate user requested to be unsubscribed from programme without even rating the service, yet the system recommends to him certain items based on personal attributes which have a match-score of up to one. User retention procedures in the organisation can then dangle such products to the user in a bid to make him change his mind.

It is this non-reliance on rating data which makes this system unique.

4.8.2 System Performance

Most recommendation algorithms steer clear of using users personal and geographic attributes due to the cost of processing this type of information. In this study, we used matrix factorisation and clustering to reduce the amount of data available for processing.

Matrix factorisation, whose results are discussed in section 4 above, helped reduce the number of attributes that had to be processed at any given time. This, coupled with the numeric coding of attributes discussed in section 3.6.4, made it easier to process the data without much computing overhead.

Clustering users, which is also discussed in the same in section 3.6.4 above, was done in order to pre-group users into clusters before any similarity computation was carried out. This move ensured that we arbitrarily cluster users into a pre-determined number of smaller clusters using the *k-means* algorithm before more resource-heavy similarity scoring algorithms were used.

4.9 Evaluating the Prototype Recommender System

The prototype hybrid recommender system developed as a part of this study was expected to be a tool that would help both the organization and its customers. The organization needed a tool that would enable it to understand the needs of its customers. An improved understanding of the customers would guide the organization in evolving the product to attract and retain more customers. The customers, on the other hand, needed a user-friendly system that would lead them to the exact items they needed when they needed them. A tool that would solve the problem of information overload and reduce the time users spent searching for the services they needed.

In this section we evaluate the system in three dimensions:

- architecture;
- functionality; and
- usability.

4.9.1 System Architecture

The application was designed to be an enterprise application using the classical client-server architecture. The application was to be hosted on a WildFly application server using a web interface and backed up by a MySQL database. When started for the first time, the application was expected to calculate all user similarity scores and cache them in memory to improve the application efficiency. The application was expected to recommend items using item similarity for all users who had used the system before and then use user similarity for new users.

Due to its design, the system was difficult to use in a single system. During testing, if a user had already preferred many items the system had a lot of data to cache and system performance degraded heavily. After returning recommendations for one heavy user we had to restart the application to force it to clear its cache before using it again. The available memory was not enough to cache results for two users.

We concluded that while the architecture was appropriate for handling the complex algorithms used by the system, the organization needed to invest heavily in infrastructure: storage hardware, processing power, and physical memory.

4.9.2 Functionality and Accuracy

The application was not tested on real users to get their feedback on the recommendations it provided. The recommendations were based on the rating predictions and similarity metrics the recommender implemented in the back end. On the front end, it displayed on the user's details, items and recommendations.

The recommender system provided some novel recommendations. However, user reaction to these items was needed to determine whether they described the recommendations as novel or just wild guesses. Without the actual user feedback, there was no way we could calculate the difference between the actual rating and the estimated rating for new recommendations which was a prerequisite for computing RMSE.

The prototype system was built to return only recommendations based on the items and user details already in the system and the user. We did not provide a functionality for a user to select an item.

4.9.3 Usability

As already stated in section 4.9.2, we never got to testing the system on real users which robbed us of an opportunity to get the actual user feedback on the system. The owners of the data did not give us access to contact the users whose data was used to build the system.

However, the system was built to give the user as much control as possible. The user had to manage his login details and the system had to provide the recommendations and handle all the other technical details.

Section 4.10 below details some of the limitations we faced during this study.

4.10 Limitations

The major limitation for this research came from the data side and from technology. Data limitations included inadequate attributes to build a customer profile and lack of supplier data for the rated items.

Technology-related limitations included resource-expensive algorithms and inability to get an accurate model.Most of the clustering algorithms expected too many arbitrarily predefined variables. In the K-Means algorithm, for example, the value of k has to be predetermined and has a huge bearing on the accuracy and efficiency of the model. Designing, developing and maintaining a recommendation system is not only expensive in both money and effort Spiegel

et al. [2009], it also required access to a quality pool information that is usually sensitive and is guarded jealously by both users and business Setten [2005].

The other major handicap is that the research adopted a 2-dimensional approach, users and items, and completely ignored the third dimension, that of the sales agent. Some sales agents are so good in their work to the effect that they can literally sell a refrigerator to an Eskimo while others are too bad to sell ice blocks in the desert. The 'who' is making sale dimension is also important

4.10.1 Data Sparsity

Users rarely stay around to provide feedback and ratings. Currently, only about 2% of the purchases are rated by the users. The services provided by PLP thrived on reducing the hassles and time users took when looking for goods and services. Getting feedback also needed to respect that.

In the case where users did not rate a product, any usage of more than once was translated into an implicit rating and was included in the computation.

4.10.2 Scalability

The total database of logged requests, users and items sit at over a million records for the past 12 months. This is an indication that memory-based algorithms needed to be highly optimised. The algorithm also needed to be robust enough to accommodate novel, innovative items which are introduced into the system. The bulk of the data is stored in the database with only the relevant items available in memory for fast recommendation

4.10.3 Collinearity

Collinearity resulted from users rating complementary items together. An example is when members of the same family rated the service of a restaurant; or when a user rated movie sequels in succession. The data preparation phase sought to eliminate over-fitting and dangers of collinearity by using normalization and pre-processing techniques.

4.10.4 Attribute Extraction

There is not much research in literature in this domain to create an appropriate attribute extraction framework. Most recommender systems seemed to focus on the web, movie or semantics recommendation. Classifying items according to attributes was difficult to implement and test without a clear tested framework to guide. This part remains an arbitrary act which may lead to developer bias.

4.10.5 General Application

The amount of personalisation required by this system lent itself to rigidity. The system tried to answer too many questions and use too many algorithms. Lack of adaptable generic recommender systems which could be applied to any domain remains a challenge to researchers in data mining. However, the results of this study are expected to apply to all other domains where there is a wide range of constantly changing items to choose from.

4.10.6 Human Computer Interaction

There is usually a trade-off between system precision and accuracy. To achieve multiple objectives, the system became heavy, clumsy and unfriendly to users. The usability of the system is overlooked in the focus on accurate and Mathematically correct algorithms. Before the system is deployed for use by the business, a user interface needs to be designed.

4.11 Chapter Summary

Our recommendation system combined collaborative and content-based filtering techniques to improve rating prediction. However, we also get to combine both user-item ratings and content features in a single unified matrix as when usage and average request interval form part of user attributes. This helped us to maximize the effectiveness of nearest neighbourhood techniques. Furthermore, this study gave emphasis to the value of both user and item features, where the significance and weighting of individual and combined features was analysed. In order to optimize the algorithms and ameliorate processing strain and memory usage of our system implementation, used matrix factorization (SVD) on the hybrid model for dimensional reduction. The resulting low-dimensional matrices were just an estimation of the original rating matrix as given by the equation, $R \approx U\text{ffl}S\text{ffl}V$, they were less sparse and revealed hidden user (U) or rather item (I) relations. Features populated by SVD had a positive impact on the explanatory

and prediction power of the decomposition since a more diverse array of features contributes to a more accurate definition of the individual users and items.The main purpose of this study was to find out which features are meaningful to user preference (rating) prediction, accompanied by an investigation into the extent to which matrix factorization could be useful to the given algorithms. On top of everything else, we were faced with the tantamount task of determining the optimal parameter settings of our constructed recommender system for the observed GetMore dataset. We also introduced and compared the various evaluation techniques, top of these being the RMSE, the MAE and the ROC curve to evaluate different variants of our hybrid solution. In Chapter 5 we explain what these results mean for both the research question and business.

Chapter 5

Discussion

5.1 Introduction

Telecommunications companies in South Africa, facing stiff competition from both within the industry and from external over-the-top service providers like WhatsApp, have had to dig deep and use the large amounts of data at their disposal to gain competitive advantage. Different telecommunications providers have come up with diverse, innovative ways to keep their customers happy and loyal. Cell C has come up with the `GetMore` service with the aims of retaining old customers and attracting new ones.

The research question we attempted to answer in this study sought to help Cell C find out what the customers needed and was broken down into more specific sub-questions for manageability. The statement of the research question was summarised as:

> *Can an effective prototype hybrid recommender system be designed, developed and implemented to accurately predict item ratings for cellphone contract customers, determine their profiles and characteristics, predict their behaviour, and recommend the PLP Group's services to individual clients based on the preferences of clients with similar interests?*

A number of sub-questions were derived from the main question to address the research question and, in the principle of breaking a big problem into smaller parts, each sub-question was tackled separately. The focus points in answering the questions were:

- Extracting latent features;
- Predicting missing ratings;

- Computing user and item similarity; and
- Building a recommendation system

In Chapter 4 we made an attempt to present the most relevant results from the study which were a culmination of several different experiments. It is needless to say there were a lot more other algorithms and combinations of algorithms that were tested, tried and experimented upon whose results were not included in the previous chapter.

5.2 Extracting Latent Features

Both SVD and PCA methods used in the study demonstrated clearly that they could be used for both dimensionality reduction and for the extraction of latent features. The results indicated positive underlying relationships existed between user characteristics and item characteristics. The results obtained from the experiments indicated that in order to get a more accurate picture of the results we needed to extract more and more latent features. They gave us the lowest RMSE value of 79% at 20 features. We could get more accurate results, but that was hindered by the limited computing resources at our disposal. For an organisation that is prepared to invest more in computing power to leverage the capabilities of Big Data technology would benefit a lot in utilising latent feature analysis in dealing with customer needs.

Due to its high computational overhead, the extraction of latent features was done offline and then the results used to an approximate rating matrix $\hat{X}$ whose vector elements were used to compute similarity online. In line with the RMSE results we obtained when extracting latent features when we calculated user similarity using the extracted features we discovered that the results became more and more accurate when we used more and more singular values in our calculations. This tallied with our earlier discovery that we needed to extract more latent features to achieve accuracy. The Cosine Similarity measure proved to be the most accurate measure of similarity when using the latent feature vectors. However, to get accurate results on any prediction and computation, we had to contend with the issue of missing ratings.

5.3 Rating Prediction

The dataset had only a 0.02% fill-rate. All metrics that relied on rating data had to be filled by predicting the unknown user ratings before any recommendations could be made. Linear Regression (LR) and the C4.5 (J48) algorithms were used to compute missing ratings. The LR algorithm was more accurate than the other algorithm because it took into account both the

rows and the columns of the rating matrix together with the overall ratings in the matrix. The RMSE value of 65.11% was far much lower than that of C4.5 which stood at 82.65%.

Rating prediction algorithms were also used to partially predict user churn and user-profitability. A thorough understanding of the profile of the user who cancelled the service would help the organization to apply pre-emptive retention measures. With an accuracy of 71% using the Bayesian Network Classifier, the organisation could safely identify well over 70% users who were candidates for cancellation.

Since rating information was so sparse in our dataset, we also used content features enhanced with SVD to predict user-churn. The LR algorithm managed to correctly classify 68.7% users. This too was a good statistic for the organisation. Prediction results using rating data were outstanding with an accuracy ratio of over 80%. However, a closer look at the Confusion Matrix and at the chosen sample of 4205 revealed that the data had been biased in such a way that all cancelled users requested cancellation-related items while those who were still active did not request any of those items. In this case, data bias led to over-fitting results.

Similarly, predicting user churn using user similarity scores had an accuracy ratio of over 80% when using either content feature or rating data, but that was toned down to 65% when both content features and rating data were combined. This indicated that there was still some bias in the use of rating data that was eliminated by using the hybrid approach.

5.4 Computing Similarity

Various algorithms were used to calculate item and user similarity for content-based, collaborative or hybrid purposes. SVD with Content Features returned the highest accuracy ratio of 78.0%. This was in tandem with what Spiegel *et al.* [2009] and others had already discovered. SVD could be used to enhance the performance of similarity algorithms, but using it weighed heavily on the computing resources. Data sparsity affected distance-based measures of similarity and the clustering technique performed badly, but the adjusted Cosine Similarity measure worked perfectly. Combining our content features with rating data helped us to ameliorate the challenges of data sparsity and address the new-user, new-item problems faced by systems using a single approach. Accurately identifying similar users would assist the organization to identify items for recommendations from the pool of users that were similar to the active user.

5.5 Business Problem

The business problem has already been stated in Section 3.6.1. User retention was given as one of those problems. What the results in Chapter 4 have shown is that the service is not palatable

to young and old users. This statistic should be worrying to business since the majority of cellphone users in South Africa are in the 18-35 age group. The ability of the recommender system to compute user and item similarity with high levels of accuracy was an encouraging outcome for the business. With over a thousand items in the database it had never been an easy task to select items for up-sales, but the recommendation engine would be able to show the user some novel items which could be of interest to him.

5.6 Chapter Summary

Both research and business questions were answered. The study has shown that understanding the users and the items is key in retaining users while attracting new ones.Indications from the study outcomes were that usage trends could also be improved through proper recommendations.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Constructing a recommender system was a taxing and complicated work. Like all data mining project, a diverse set of skills was needed to produce an effective and efficient system. Business and data understanding required strong business and systems analysis skills while algorithm selection and optimisation needed someone who had a Mathematical or Statistical background. The final design part needed a software developer and engineer who was skilled in both Java and JSON.

This study sought to construct a unique hybrid recommender system, a novel approach to the prediction problem which combined both content-based and collaborative filtering algorithms and enhanced them with Matrix Factorisation. The intention of combining the rating and content variables was to reduce the number of parameters but increase the degree of accuracy by eliminating the deficiencies caused by lack of rating data or incomplete feature information.

The amount and diversity of goods and services available to consumers were huge and new ones kept coming up either as a response to unlimited human wants or as an inducement to get people to spend more. Users, therefore, needed assistance to choose from loads of goods and services on offer while business needed help to target consumers with goods most relevant to them in their marketing. The huge amount of information available, items and services that also made it a challenge to find out what was of interest to the user. This text focused on helping the organization and the users to sift through loads of information at their disposal and give a personalised recommendation automatically by building a system which learns and adapts its behaviour to support users in decision-making.

The system built as part of this study recommended items based on how it computes the item's utility to the user. The process calculates how relevant an item is to a user and then

recommends it. The reason there were so many algorithms used stemmed from the need to capture all humanely possible preferences. The techniques used by Recommender Systems use knowledge about the user, the behaviour of other similar users, the item, and other similar items to make a recommendation. Each of the techniques used has its own advantages and disadvantages. The only way to counter the disadvantages and build a strong recommendation system was to use an eclectic of techniques.Systems which combined different techniques as was done in this text are called a hybrid recommender system and have been proved to be slightly more accurate. The recommender system had to incorporate what the user is, his goals and the properties of the items to make an almost perfect recommendation.

Due to the size of content features of the combined matrix our model grew to an enormous size causing the system to crash often due to increased computational requirements. Singular Value Decomposition ameliorated the strain placed on the computing resources by reducing the sizes of our content vectors, but performing SVD was too heavy given the size of the dataset.

6.2 Future Work

During the study, we came across many potentially interesting and relevant research subjects, but to keep focused on the objectives of this research and to avoid scope-creep these topics had to be ignored. This section briefly highlights those research subjects for possible future work.

This study focused only on the item and the user. It assumed that the user and the item would interact directly with the recommender system, however, in many instances, the user interacts directly with a seller. This study completely ignored the impact of the sales agent when building this recommendation.

It is highly likely that more dimensions that add to user satisfaction can be identified. The recommendation algorithms used can do more to understand the user. Although the dataset contained many user attributes, a lot of information about the user was missing.

Another question that needs further study is whether the prediction method developed in this research is also applicable to other domains. One of the prevalent shortcomings of recommender systems is lack of universal applicability.

There are many other theories in other fields of study which seek to explain human behaviour and human choice. It may be useful to investigate how the Decision theory in Economics, for example, could help explain human behaviour and thus improve the quality of our recommendations.

Appendix A

Examples of Recommender Systems

A.1 Introduction

This Appendix give a few web-based recommendation systems used by some reputable organisations.

A.2 Amazon

The Amazon is an American e-commerce company. It is an electronic market for goods and services.



FIGURE A.1: The Amazon Recommendation System

In the image in Figure A.1 the recommender is recommending items based on item-item similarity.

A.3 IMDb

IMDb is a movie rental site. In the picture in figure A.2 a user is shown movies similar to the one that he liked.



FIGURE A.2: The IMDb Movie Recommendation System

A.4 Gumtree

Gumtree is another consumer-to-consumer e-Commerce website. In the image the system shows the buyer other cars he may be interested in.

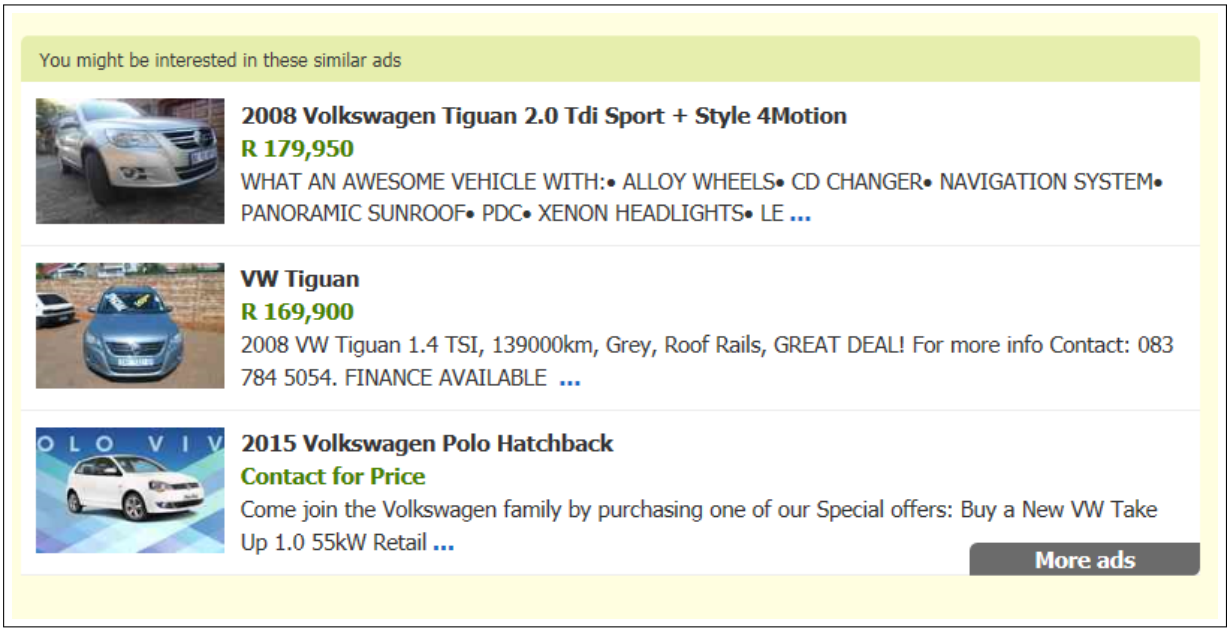


FIGURE A.3: The Gumtree Recommendation System

Appendix B

Java Code

B.1 Latent Features

```
/**
 *
 */
package main.java.za.ac.za.wits.mpume.getmore.matrixfactorization;

/**
 * @author mpume
 *
 */


import java.util.HashMap;
import java.util.Map;

import main.java.za.ac.za.wits.mpume.getmore.util.Classifier;
import main.java.za.ac.za.wits.mpume.getmore.util.FeatureException;
import main.java.za.ac.za.wits.mpume.getmore.util.SGDLearner;
import main.java.za.ac.za.wits.mpume.getmore.util.VectorUtils;
import main.java.za.ac.za.wits.mpume.getmore.util.data.Instance;
import main.java.za.ac.za.wits.mpume.getmore.util.data.Item;
import main.java.za.ac.za.wits.mpume.getmore.util.data.User;
import main.java.za.ac.za.wits.mpume.getmore.util.data.UserItem;

/**
 * Matrix factorization model for recommendation.
 * The model only uses latent features. (please note that it suffers from "cold
    start" problem )
```

```
27  *
28  */
29 public class MatrixFactorizationModel implements SGDLearner, Classifier{
30
31     // free-parameters to control to model, should be learned from cross validation
32     // TODO need to set this in config.
33     public double stepSize = 0.01;
34     public double regularizationRate = 1;
35
36     public double squareLoss = 0;
37
38     /** Item id (index) to Item object map */
39     Map<Integer, Item> items = new HashMap<Integer, Item>();
40     /** User id (index) to User object map */
41     Map<Integer, User> users = new HashMap<Integer, User>();
42
43     @Override
44     public void update(Instance instance) throws FeatureException {
45         int userIndex = ((UserItem)instance).getUserIndex();
46         int itemIndex = ((UserItem)instance).getItemIndex();
47         double rating = ((UserItem)instance).getRating();
48
49         if (!users.containsKey(userIndex)){
50             // initialize new UserMF
51             users.put(userIndex, new User());
52         }
53
54         if (!items.containsKey(itemIndex)){
55             // initialize new ItemMF.
56             items.put(itemIndex, new Item());
57         }
58
59         User user = users.get(userIndex);
60         Item item = items.get(itemIndex);
61
62         // update the weights.
63         double epsilon = VectorUtils.calInnerProduct(user.getLatentFeatures(),
64                 item.getLatentFeatures()) - rating;
65         squareLoss += epsilon * epsilon;
66
67         // update User latent feature vector.
68         // u(t) = (1-\eta * lamda)u(t-1)-\eta * epsilon * v(t-1).
69         // u(t) is the User for time t,  and v(t) is Item for time t.
```

```
70          double[] firstTerm = VectorUtils.calMultiply(1-stepSize*regularizationRate,
    user.getLatentFeatures());
71          double[] secondTerm = VectorUtils.calMultiply(stepSize * epsilon,
    item.getLatentFeatures()) ;
72          double[] updatedUserFeatures = VectorUtils.calMinus(firstTerm, secondTerm);
73
74          // update Item latent feature vector
75          // v(t) = (1-\eta * \lamda)v(t-1) - \eta * \epsilon * u(t-1)
76          firstTerm = VectorUtils.calMultiply(1-stepSize*regularizationRate,
    item.getLatentFeatures());
77          secondTerm = VectorUtils.calMultiply(stepSize * epsilon,
    user.getLatentFeatures());
78          double[] updatedItemFeatures = VectorUtils.calMinus(firstTerm, secondTerm);
79
80          user.setLatentFeatures(updatedUserFeatures);
81          item.setLatentFeatures(updatedItemFeatures);
82          users.put(userIndex, user);
83          items.put(itemIndex, item);
84      }
85
86      @Override
87      public double predict(Instance instance) {
88          User user = users.get(instance.getUserIndex());
89          Item item = items.get(instance.getItemIndex());
90
91          if (user == null || item == null){
92              System.out.println("cold start for standard matrix factorization!");
93              return -1;
94          }
95
96          double predictedValue = 0;
97          try{
98              predictedValue = VectorUtils.calInnerProduct(user.getLatentFeatures(),
99                      item.getLatentFeatures());
100         } catch(Exception e){
101             System.out.println("Cannot prodict the rating for given instance.
    Exception" +
102                     " occurs when compute the dot product of two feature vectors");
103             return -1;
104         }
105         return predictedValue;
106     }
107 }
```

Singular Value Decomposition

```
package main.java.za.ac.za.wits.mpume.getmore.util;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

import weka.core.matrix.Matrix;
import weka.core.matrix.SingularValueDecomposition;

public class SVD {
        public static void main(String[] args) throws IOException {
                double[][] data = new double[4205][136];
                String scan;
                FileReader file = new FileReader(
                                "data/user_vectors.csv");
                BufferedReader br = new BufferedReader(file);
                String first = br.readLine();
                if (null != first) {
                        System.out.println("First is: " + first);
                        String[] numberStrings = first.split(" ");
                        double[] numbers = new double[numberStrings.length];
                        for (int i = 0; i < numbers.length; i++) {
                                numbers[i] = Double.parseDouble(numberStrings[i]);
                                // System.out.println("Number " + i + " is " +
    numbers[i]);
                        }
                        while ((scan = br.readLine()) != null) {
                                 System.out.println(scan);
                        }
                }
                br.close();

                Matrix C = new Matrix(data);
                C.norm1();
                C.print(9, 6);

                int M = 9, N = 6;
                Matrix B = C.getMatrix(0, 420, 2, 135);

                B.print(1, 12);
                // Matrix B = Matrix.random(5, 3);

```

```
42                  Matrix A = B.times(B.transpose());
43
44                  // compute the singular values decomposition
45
46                  SingularValueDecomposition s = A.svd();
47                  Matrix svalues = new Matrix(s.getSingularValues(), 1);
48                  svalues.print(1000, 20);
49          }
50
51  }
```

B.2 Attribute Selection

These are not in any way near implementation specific. In practice on data access and interface classes will be needed to complement the Recommender101 API.

User Attributes

```
1
2   package weka.test.packages.weka.test;
3
4   /**
5    * @author WEKA
6    *
7    */
8
9   import weka.attributeSelection.*;
10  import weka.core.*;
11  import weka.core.converters.ConverterUtils.*;
12  import weka.classifiers.*;
13  import weka.classifiers.meta.*;
14  import weka.classifiers.trees.*;
15  import weka.filters.*;
16
17  import java.util.*;
18
19  public class AttributeSelectionClass {
20
21
22    /**
23     * uses the meta-classifier
```

```
24     */
25    protected static void useClassifier(Instances data) throws Exception {
26      System.out.println("\n1. Meta-classfier");
27      AttributeSelectedClassifier classifier = new AttributeSelectedClassifier();
28      CfsSubsetEval eval = new CfsSubsetEval();
29
30      GreedyStepwise search = new GreedyStepwise();
31      search.setSearchBackwards(true);
32      J48 base = new J48();
33      classifier.setClassifier(base);
34      classifier.setEvaluator(eval);
35      classifier.setSearch(search);
36      Evaluation evaluation = new Evaluation(data);
37      evaluation.crossValidateModel(classifier, data, 10, new Random(1));
38      System.out.println(evaluation.toSummaryString());
39    }
40
41    /**
42     * uses the filter
43     */
44    protected static void useFilter(Instances data) throws Exception {
45      System.out.println("\n2. Filter");
46      weka.filters.supervised.attribute.AttributeSelection filter = new
        weka.filters.supervised.attribute.AttributeSelection();
47      CfsSubsetEval eval = new CfsSubsetEval();
48      GreedyStepwise search = new GreedyStepwise();
49      search.setSearchBackwards(true);
50      filter.setEvaluator(eval);
51      filter.setSearch(search);
52      filter.setInputFormat(data);
53      Instances newData = Filter.useFilter(data, filter);
54      System.out.println(newData);
55    }
56
57    /**
58     * uses the low level approach
59     */
60    protected static void useLowLevel(Instances data) throws Exception {
61      System.out.println("\n3. Low-level");
62      AttributeSelection attsel = new AttributeSelection();
63      CfsSubsetEval eval = new CfsSubsetEval();
64      GreedyStepwise search = new GreedyStepwise();
65      search.setSearchBackwards(true);
66      attsel.setEvaluator(eval);
```

```
67      attsel.setSearch(search);
68      attsel.SelectAttributes(data);
69      int[] indices = attsel.selectedAttributes();
70      System.out.println("selected attribute indices (starting with 0):\n" +
        Utils.arrayToString(indices));
71    }
72
73
74    public static void main(String[] args) throws Exception {
75      // load data
76
77
78     System.out.println("\n0. Loading data");
79     DataSource source = new
        DataSource("/home/mpume/BitBucket/School/Data/Matrices/dumpUsersP.arff");
80      Instances data = source.getDataSet();
81      if (data.classIndex() == -1)
82        data.setClassIndex(data.numAttributes() - 1);
83
84      // 1. meta-classifier
85      useClassifier(data);
86      // 2. filter
87      useFilter(data);
88
89      // 3. low-level
90      useLowLevel(data);
91      // save labeled data
92      System.out.println("\n4. Saving data");
93      BufferedWriter writer = new BufferedWriter(
94                                  new
        FileWriter("/home/mpume/BitBucket/SelectedUsersAttributes.arff"));
95      writer.write(data.toString());
96      writer.newLine();
97      writer.flush();
98      writer.close();
99    }
100 }
```

B.3 Calculating Similarity

Adjusted Cosine Similarity

```
/**
 *
 */
package weka.test.packages.weka.test;

/**
 * @author mpume
 *
 */


import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.TreeMap;

import com.Ostermiller.util.CSVParser;
import com.Ostermiller.util.LabeledCSVParser;


        public class AdjustedCosineSimilarity {
                public static final int COSINE = 0;
                public static final int L2 = 1;
                public static final int L1 = 2;
                public static double calculateSimilarity(double[] vec1, double[]
    vec2, int type) {
                        double similarity = 0;
                        assert(vec1.length == vec2.length);
                        for (int i = 0; i < vec1.length; i++) {
                                switch (type) {
                                case(COSINE):
                                        similarity += vec1[i] * vec2[i];
                                        break;
                                case(L2):
                                        similarity += Math.pow(vec1[i] - vec2[i], 2);
                                        break;
                                case(L1):
```

```
43                                      similarity += Math.abs(vec1[i] - vec2[i]);
44                                      break;
45                              }
46                      }
47                      if (type == COSINE)
48                              similarity = similarity / (vectorLength(vec1) *
    vectorLength(vec2));
49                      else if (type == L2)
50                              similarity = Math.sqrt(similarity);
51                      return similarity;
52              }
53              private static double vectorLength(double[] vec) {
54                      double len = 0;
55                      for (int i = 0; i < vec.length; i++) {
56                              len += vec[i] * vec[i];
57                      }
58                      len = Math.sqrt(len);
59                      return len;
60              }
61              public static double calculateSimilarity(double[] vec1, double[]
    vec2) {
62                      return calculateSimilarity(vec1, vec2, L2);
63              }
64
65              public static int[] similarVectors(double[][] vectors, int index, int
    type) {
66                      int[] indices = new int[vectors.length - 1];
67                      int ind = 0;
68                      TreeMap<Integer, Double> unsortedSimilarities = new
    TreeMap<Integer, Double>();
69                      for (int i = 0; i < vectors.length; i++) {
70                              if (i == index) continue;
71                              unsortedSimilarities.put(i,
    calculateSimilarity(vectors[index], vectors[i], type));
72                      }
73
74                      TreeMap<Integer, Double> sortedSimilarities =
75                              new TreeMap<Integer, Double>(new
    ValueComparer(unsortedSimilarities));
76                      sortedSimilarities.putAll(unsortedSimilarities);
77                      for (Iterator<Integer> it =
    sortedSimilarities.keySet().iterator(); it.hasNext(); ) {
78                              indices[ind++] = it.next();
79                      }
```

```
80                      return indices;
81              }
82
83              public static int[] similarVectors(double[][] vectors, int index) {
84                      return similarVectors(vectors, index, COSINE);
85              }
86
87              public static int[] similarVectors(ArrayList<double[]> vectors, int
    index, int type) {
88                      int[] indices = new int[vectors.size() - 1];
89                      int ind = 0;
90                      TreeMap<Integer, Double> unsortedSimilarities = new
    TreeMap<Integer, Double>();
91                      for (int i = 0; i < vectors.size(); i++) {
92                              if (i == index) continue;
93                              unsortedSimilarities.put(i,
    calculateSimilarity(vectors.get(index), vectors.get(i), type));
94                      }
95
96                      TreeMap<Integer, Double> sortedSimilarities =
97                              new TreeMap<Integer, Double>(new
    ValueComparer(unsortedSimilarities));
98                      sortedSimilarities.putAll(unsortedSimilarities);
99                      for (Iterator<Integer> it =
    sortedSimilarities.keySet().iterator(); it.hasNext(); ) {
100                             indices[ind++] = it.next();
101                     }
102                     return indices;
103             }
104
105             public static int[] similarVectors(ArrayList<double[]> vectors, int
    index) {
106                     return similarVectors(vectors, index, COSINE);
107             }
108
109             public static void saveVectorsToFile(ArrayList<double[]> vectors,
    String filename) {
110                     FileWriter fw;
111                     double[] vector;
112                     try {
113                             fw = new FileWriter(filename);
114                             for (int i = 0; i < vectors.size(); i++) {
115                                     vector = vectors.get(i);
116                                     for (int j = 0; j < vector.length; j++) {
```

```
117                                             fw.write(Double.toString(vector[j]) +
    " ");
118                                     }
119                                     fw.write("\n");
120                             }
121                             fw.flush();
122                             fw.close();
123
124                     } catch (IOException e) {
125                             e.printStackTrace();
126                     }
127             }
128
129
130             public static ArrayList<double[]> loadVectorsFromFile() throws
    FileNotFoundException, IOException {
131                     String filename =
    "/home/mpume/BitBucket/School/Data/Matrices/UserItemMatrixCos.csv";
132
133                     BufferedReader fr;
134                     String line;
135                     String[] values;
136                     double[] vector;
137                     ArrayList<double[]> vectors = new ArrayList<double[]>();
138                     try {
139                             //BufferedReader ff = new BufferedReader(new
    FileReader("/home/mpume/BitBucket/School/Data/Matrices/UserItemMatrixNoGuest.arff"));
140
141                             fr = new BufferedReader(new FileReader(filename));
142                             line = fr.readLine();
143                             while (line != null) {
144                                     values = line.split(",");
145
146                                     vector = new double[values.length];
147                                     for (int i = 0; i < values.length; i++) {
148                                             vector[i] =
    Double.parseDouble(values[i]);
149                                     }
150                                     vectors.add(vector);
151                                     line = fr.readLine();
152                                     System.out.println(line.toString());
153                             }
154                             fr.close();
155                             return vectors;
```

```
156                         } catch (IOException e) {
157                                 e.printStackTrace();
158                         }
159                         return null;
160                 }
161
162                 private static class ValueComparer implements Comparator<Integer> {
163                         private TreeMap<Integer, Double>  _data = null;
164                         public ValueComparer (TreeMap<Integer, Double> data){
165                                 super();
166                                 _data = data;
167                         }
168
169                  public int compare(Integer o1, Integer o2) {
170                          double e1 = _data.get(o1);
171                      double e2 = _data.get(o2);
172                      if (e1 > e2) return -1;
173                      if (e1 == e2) return 0;
174                      if (e1 < e2) return 1;
175                      return 0;
176                  }
177                 }
178
179
180         }
```

References

[Adeniyi *et al.* 2016] Adeniyi D .A. , Wei Z. , and Yongquan Y . Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method. *Applied Computing and Informatics*, 12(1):90–108, 2016.

[Adomavicius and Tuzhilin 2001] Adomavicius G. and Tuzhilin A . Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, 2001.

[Adomavicius and Tuzhilin 2005] Adomavicius G. and Tuzhilin A . Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[Agrawal and Srikant 1994] Agrawal R and Srikant R . Fast algorithms for mining association rules. *Proceeding VLDB '94 Proceedings of the 20th International Conference on Very Large Databases*, pages 487–499, 1994.

[Agrawal *et al.* 2009] Agrawal M. , Karimzadehgan M. , and Zhai C . An online news recommender system for social networks. *Urbana*, 51:61801, 2009.

[Agrawal 2013] Agrawal D . A comprehensive study of data mining and application. *International Journal of Advanced Research in Computer Engineering & Technology* (IJARCET), 2(1):249–252, January 2013.

[Aksel and Birtürk 2010] Aksel F. and Birtürk A . An adaptive hybrid recommender system that learns domain dynamics. In *Int. Workshop on Handling Concept Drift in Adaptive Information Systems: Importance, Challenges and Solutions (HaCDAIS-2010) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 49–56, 2010.

[Amatriain 2014] Amatriain X . The recommender problem revisited. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 397–398. ACM, 2014.

[Antunes and Oliveira 2004] Antunes C. and Oliveira A.L . Sequential pattern mining with approximated constraints. In *Int. Conf Applied Computing*, pages 131–138, 2004.

[Antunes 2008] Antunes C . Acquiring background knowledge for intelligent tutoring systems. In *Educational Data Mining 2008: The 1st International Conference on Educational Data Mining Montréal, Québec, Canada, June 20-21, 2008 Proceedings 1*, pages 18–27, Montreal, Canada, June 2008.

[Arndt and Gersten 2001] Arndt D. and Gersten W . External data selection for data mining in direct marketing. In *Proceedings of the Sixth International Conference on Information Quality*, pages 44–61, 2001.

[Asabere 2012] Asabere N. Y . Towards a perspective of hybrid approaches and methodologies in recommender systems. *International Journal of Computer Science and Telecommunications*, 3(11):23–32, November 2012.

[Azevedo 2008] Azevedo A. Isabel R. L . Kdd, semma and crisp-dm: a parallel overview. *IADS-DM*, 2008.

[Baitharu and Pani 2013] Baitharu T.R. and Pani S. K . A survey on application of machine learning algorithms on data mining. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 3(7):17–20, December 2013.

[Baker 2005] Baker K . Singular value decomposition tutorial. 2005.

[Balabanović 1997] Balabanović M . An adaptive web page recommendation service. In *Proceedings of the first international conference on Autonomous agents*, pages 378–385. ACM, 1997.

[Balabanović 1998] Balabanović M . Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction*, 8(1-2):71–102, 1998.

[Basu *et al.* 1998] Basu C. , Hirsh H. , and Cohen W . Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[Bell and Koren 2007] Bell R. M. and Koren Y . Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 43–52. IEEE, 2007.

[Berkhin 2006] Berkhin P . A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[Berry and Linoff 2004] Berry M.J.A. and Linoff G.S . 2004. *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management.* Wiley, Indianapolis, 2nd edition.

[Bhavnani *et al.* 2008] Bhavnani A. , Chiu R. W. , Janakiram S. , and Bhatia D . The role of mobile phones in sustainable rural poverty reduction. 2008.

[Blockeel *et al.* 2002] Blockeel H. , Bruynooghe M. , Dzeroski S. , Ramon J. , and Struyf J . Hierarchical multi-classification. In *Proceedings of the ACM SIGKDD 2002 workshop on multi-relational data mining (MRDM 2002)*, pages 21–35, 2002.

[Brand 2003] Brand M . Fast online SVD revisions for lightweight recommender systems. In *Proceedings of the Third SIAM International Conference on Data Mining*, volume 112, page 37. SIAM, 2003.

[Breese *et al.* 1998] Breese J. S , Heckerman D. , and Kadie C . Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[Burke 2002] Burke R . Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[Burke 2007] Burke R . Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[Business Intelligence 2014] *Business Intelligence Defined*, August 2014. Turning Data into Actionable Information. Retrieved 03 August 2014, from `http://www.logianalytics.com/bi-encyclopedia/business-intelligence`

[Bustos 2012] *Selling Value-added Services to Mobile Customers*, July 2012. Retrieved 25 July 2012, from `http://www.getelastic.com/selling-value-added-services-to-mobile-customers`

[Cacheda *et al.* 2011] Cacheda F. , Carneiro V. , Fernández D. , and Formoso V . Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):2, 2011.

[Calvo-Flores *et al.* 2006] Calvo-Flores M. D. , Galindo E. G. , Jiménez M. C. P. , and Pineiro O. P . Predicting students' marks from moodle logs using neural network models. *Current Developments in Technology-Assisted Education*, 1:586–590, 2006.

[Camilovic 2008] Camilovic D . Data mining and CRM in telecommunications. *Serbian Journal of Management*, 3(1):61–72, February 2008.

[Carlsson and Walden 2002] Carlsson C. and Walden P . Mobile commerce: A summary of quests for value-added products and services. *Proceedings, 15th Bled Electronic Commerce Comference, eReality: Constructing the eEconomy, Bled*, pages 463–475, 2002.

[CellC GetMore 2014] *GETMORE*, August 2014. Your very own personal assistant, expert advice, great savings and 24/7 emergency services are just one call away. Let GET MORE connect you to the right people at the right time, negotiate on your behalf, make bookings and above all, save you time and money. Retrieved 03 August 2014, from `https://www.getmore247.co.za/`

[Chapman *et al.* 2000] Chapman P. , Clinton J. , Kerber R. , Khabaza T. , Reinartz T. , Shearer C. , and Wirth R . Crisp-dm 1.0 step-by-step data mining guide. 2000.

[Cheung *et al.* 2003] Cheung K. , Kwok J. T. , Law M. H. , and Tsui K . Mining customer product ratings for personalized marketing. *Decision Support Systems*, 35(2):231–243, 2003.

[Chirita *et al.* 2005] Chirita P. , Nejdl W. , and Zamfir C . Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 67–74. ACM, 2005.

[Cios *et al.* 2007] Cios K.J. , Pedrycz W. , Swiniarski R.W. , and Kurgan L.A . 2007. *Data Mining: A Knowledge Discovery Approach.* Springer Science+Business Media, LLC, New York.

[Cisty and Bezak 2013] Cisty M. and Bezak J . The data mining ensemble approach to river flow predictions. *evaluation*, 8:213–220, 2013.

[Claypool *et al.* 1999] Claypool M. , Gokhale A. , Miranda T. , Murnikov P. , Netes D. , and Sartin M . Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[Curram and Mingers 1994] Curram S. P. and Mingers J . Neural networks, decision tree induction and discriminant analysis: An empirical comparison. *Journal of the Operational Research Society*, pages 440–450, 1994.

[Deerwester *et al.* 1990] Deerwester S. C. , Dumais S. T. , Landauer T. K. , Furnas G. W. , and Harshman R. A . Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[Deshpande and Karypis 2004] Deshpande M. and Karypis G . Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* (TOIS), 22(1):143–177, 2004.

[deVille 2006] deVille B . 2006. *Decision Trees for Business Intelligence and Data Mining: Using* SAS *Enterprise Miner.* SAS Institute, Cary, NC, USA.

[Dolnicar and Jordaan 2007] Dolnicar S. and Jordaan Y . A market-oriented approach to responsibly managing information privacy concerns in direct marketing. *Journal of Advertising*, 36(2):123–149, 2007.

[Dooms *et al.* 2011] Dooms S. , De Pessemier T. , and Martens L . An online evaluation of explicit feedback mechanisms for recommender systems. In *7th International Conference on Web Information Systems and Technologies (WEBIST-2011)*, pages 391–394. Ghent University, Department of Information technology, 2011.

[Drachsler *et al.* 2008] Drachsler H. , Hummel H. G. K. , and Koper R . Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model. *International Journal of Learning Technology*, 3(4):404–423, 2008.

[Duan *et al.* 2011] Duan L. , Street W. N. , and Xu E . Healthcare information systems: data mining methods in the creation of a clinical recommender system. *Enterprise Information Systems*, 5(2):169–181, 2011.

[Dumais 1992] Dumais S. T . Enhancing performance in Latent Semantic Indexing (LSI) retrieval. 1992.

[Eichelmann *et al.* 2011] Eichelmann T. , Fuhrmann W. , Trick U. , and Ghita B . Value-added services. 2011. Retrieved 22 July 2014, from `http://www.e-technik.orgaufsaetze_vortraegeaufsaetzeeichelmann_et_al_ita11.pdf`

[Euzenat 2007] Euzenat J . Semantic precision and recall for ontology alignment evaluation. In *IJCAI*, pages 348–353, 2007.

[Farajian and Mohammadi 2010] Farajian M. A. and Mohammadi S . Mining the banking customer behavior using clustering and association rules methods. *International Journal of Industrial Engineering*, 21(4), 2010.

[Fayyad and Smyth 1996] Fayyad,U.and Piatetsky-Shapiro G.and and Smyth P . The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 29(11):27–34, 1996.

[Fayyad *et al.* 1996] Fayyad U. , Piatetsky-Shapiro G. , and Smyth P . From data mining to knowledge discovery in databases. 1996.

[Fernandez-Luque *et al.* 2009] Fernandez-Luque L. , Karlsen R. , and Vognild L. K . Challenges and opportunities of using recommender systems for personalized health education. 2009.

[Francke and Weideman 2008] Francke E. and Weideman M . South african youth and mobile technology impact: The MXit phenomenon. *Dynamics (JBMD)*, page 81, 2008.

[Galland 2012] Galland A . *Recommender Systems.* Technical report, INRIA-Saclay, 2012., 2012.

[Gera *et al.* 2014] Gera P. K. , Basha S. A. H. , and Rao K. Sr . Analyzing education data through association rules: A case study. *International Journal of Advanced Trends in Computer Science and Engineering*, 3(1):161–166, 2014.

[Geyer-Schulz *et al.* 2001] Geyer-Schulz A. , Hahsler M. , and Jahn M . Educational and scientific recommender systems: Designing the information channels of the virtual university. 2001.

[Ghazanfar and Prugel-Bennett 2010] Ghazanfar A and Prugel-Bennett A . An improved switching hybrid recommender system using Naive Bayes Classifier and collaborative filtering. In *The 2010 IAENG International Conference on Data Mining and Applications*, volume 1, pages 1–10, Hong Kong, May 2010.

[Ghiselli 1964] Ghiselli E. E . 1964. *Theory of psychological measurement*, volume 13. McGraw-Hill New York.

[Giudici 2003] Giudici P . 2003. *Applied Data Mining: Statistical Methods for Business and Industry*. Wiley, New York.

[Glauber *et al.* 2013] Glauber R. , Loula A. , and Rocha-Junior J . A mixed hybrid recommender system for given names 25-36. In *Proceedings of the ECML PKDD Discovery Challenge - Recommending Given Names co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2013*, ECML PKDD 2013, pages 25–36, Prague, September 2013.

[Gniazdowski and Grabowski 2016] Gniazdowski Z. and Grabowski M . Numerical coding of nominal data. 2016.

[Good *et al.* 1999] Good N. , Schafer J. B. , Konstan J. A. , Borchers A. , Sarwar B. , Herlocker J. , and Riedl J . Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446, 1999.

[Gorunescu 2011] Gorunescu F . 2011. *Data Mining: Concepts,Models and Techniques*, volume 12 of *Intelligent Systems Reference Library*. Springer, Berlin.

[Gottgtroy *et al.* 2004] Gottgtroy P. , Kasabov N. , and MacDonell S . An ontology driven approach for knowledge discovery in biomedicine. 2004.

[Gower 2014] Gower S . Netflix prize and svd. 2014.

[Grivolla *et al.* 2010] Grivolla J. , Badia T. , Campo D. , Sonsona M. , and Pulido J . A hybrid recommender combining user, item and interaction data. *age*, 20:25–30, 2010.

[Gunawardana and Meek 2009] Gunawardana A. and Meek C . A unified approach to building hybrid recommender systems. *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 117–124, 2009.

[Gupta and Garg 2014] Gupta A. and Garg D . Applying data mining techniques in job recommender system for considering candidate job preferences. In *International Conference*

on Advances in Computing, Communications and Informatics (ICACCI, 2014), pages 1458–1465. IEEE, 2014.

[Han 1996] Han J . Data mining techniques. In *ACM SIGMOD Record*, volume 25, page 545. ACM, 1996.

[Han 2006] Han J . 2006. *Data Mining:Concepts and Techniques*. Morgan Kauffman, Boston, 2nd edition.

[Han 2011] Han J . 2011. *Data Mining:Concepts and Techniques*. Morgan Kauffman, Boston, 3rd edition.

[Hand *et al.* 2001] Hand D. , Mannila H. , and Smythe P . 2001. *Principles of Data Mining*. MIT Press.

[Haruechaiyasak *et al.* 2004] Haruechaiyasak C. , Shyu M. , and Chen S . A data mining framework for building a web-page recommender system. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 357–362. IEEE, 2004.

[Herlocker *et al.* 1999] Herlocker J. L. , Konstan J. A. , Borchers A. , and Riedl J . An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[Herlocker *et al.* 2004] Herlocker J. L. , Konstan J. A. , Terveen L. G. , and Riedl J. T . Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[Hsieh 2004] Hsieh N . An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert systems with applications*, 27(4):623–633, 2004.

[Huang 1998] Huang Z . Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[Hüllermeier 2005] Hüllermeier E . Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems*, 156(3):387–406, 2005.

[Inc] Inc Oracle Corporation . *MySQL Community Downloads*. http://dev.mysql.com/downloads/. Accessed January 4, 2015.

[Jackson 2002] Jackson J . Data mining: A conceptual overview. *Communications of the Association for Information Systems*, 8(1):19, 2002.

[Jannach *et al.* 2011] Jannach D. , Zanker M. , Felfernig A. , and Friedrich G . 2011. *Recommender Systems: An Introduction*. Cambridge University Press, Cambridge.

[Jantan *et al.* 2012] Jantan H. , Hamdan A. . , and Othman Z. A . Intelligent dss for talent management: a proposed architecture using knowledge discovery approach. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, pages 90–96. ACM, 2012.

[Jawaheer *et al.* 2014] Jawaheer G. , Weller P. , and Kostkova P . Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4(2):8, 2014.

[Jovanoski and Lavrač 2001] Jovanoski V. and Lavrač N . 2001. *Classification rule learning with APRIORI-C*. Springer.

[Kadav *et al.* 2003] Kadav A. , Kawale J. , and Mitra P . *Data Mining Standards*, 2003.

[Kamath *et al.* 2008] Kamath V. , Bhonsale S. , and Manjrekar P . Customer relationship management: A key success factor in services marketing (a case study of tourism (hotel) services in navi mumbai). In *Conference on Tourism in India–Challenges Ahead*, volume 15, page 17, 2008.

[Karthikeyan and Thangaraju 2014] Karthikeyan T. and Thangaraju P . Pca-nb algorithm to enhance the predictive accuracy. *Int. J. Eng. Tech*, 6(1):381–387, 2014.

[Kirkos *et al.* 2007] Kirkos E. , Spathis C. , and Manolopoulos Y . Data mining techniques for the detection of fraudulent financial statements. *Expert Systems with Applications*, 32:995–1003, 2007.

[Koren *et al.* 2009] Koren Y. , Bell R. , and Volinsky C . Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Koren 2008] Koren Y . Factorization meets the neighborhood: a multifaceted collaborative filtering model. *KDD'08*, 2008.

[Kruse *et al.* 1999] Kruse R. , Nauck D. , and Borgelt C . Data mining with fuzzy methods: status and perspectives. In *Proc. 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, 1999.

[Lai and Liaw 2008] Lai J. Z.C. and Liaw Y . Improvement of the k-means clustering filtering algorithm. *Pattern Recognition*, 41(12):3677–3681, 2008.

[Larose 2005] Larose D.T . 2005. *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, New Jersey.

[Larose 2006] Larose D.T . 2006. *Data Mining Methods and Models*. John Wiley & Sons, New Jersey.

[Larsen 2013] Larsen J. B . Content-based recommender systems. 2013.

[Lee and Chang 2013] Lee C. and Chang Y . Enhancing accuracy and performance of collaborative filtering algorithm by stochastic svd and its mapreduce implementation. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1869–1878. IEEE, 2013.

[Lee and Kerschberg 1998] Lee S. W. and Kerschberg L . A methodology and life cycle model for data mining and knowledge discovery in precision agriculture. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2882–2887. IEEE, 1998.

[Lee and Seung 2000] Lee D. D. and Seung H. S . Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2000.

[Lee *et al.* 1999] Lee W. , Stolfo S. J , and Mok K. W . A data mining framework for building intrusion detection models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 120–132. IEEE, 1999.

[Lehmann *et al.* 2008] Lehmann A. , Fuhrmann W. , Trick U. , and Ghita B . New possibilities for the provision of value-added services in SIP-based peer-to-peer networks. *Proc. of SEIN*, pages 167–176, 2008.

[Li and Kim 2003] Li Q. and Kim B.M . Clustering approach for hybrid recommender system. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03)*, pages 1–7. IEEE, 2003.

[Li *et al.* 2012] Li L. , Hsu R. , and Lee F . Review of recommender systems and their applications. *T&S Journal Publications*, 2012.

[Lin *et al.* 2002] Lin W. , Alvarez S. A. , and Ruiz C . Efficient adaptive-support association rule mining for recommender systems. *Data mining and knowledge discovery*, 6(1):83–105, 2002.

[Linden *et al.* 2003] Linden G. , Smith B. , and York J . Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[Ling and Li 1998] Ling C. X. and Li C . Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79, 1998.

[Llanes and Puldon 2008] Llanes J. J. and Puldon R. S . CRISP-DM *Methodology Applied to a Process Mining Project: A Case Study*, 2008.

[Lops *et al.* 2011] Lops P. , De Gemmis M. , and Semeraro G . Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[Ma *et al.* 1998] Ma B. , Liu W. , and Y Hsu . Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[Ma *et al.* 2007] Ma H. , King I. , and Lyu M. R . Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 39–46. ACM, 2007.

[Ma *et al.* 2011] Ma H. , Zhou D. , Liu C. , Lyu M. R. , and King I . Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[Maimon and Rokach 2009] Maimon O. and Rokach L . Introduction to knowledge discovery and data mining. In *Data mining and knowledge discovery handbook*, pages 1–15. Springer, 2009.

[Makwana *et al.* 2014] Makwana K. , Sharma N. , and Arora S . Factors influencing consumer brand switching behaviour in telecommunication industry: An empirical study. *Prestige e-Journal of Management & Reserach*, 1(1):87–96, 2014.

[Manouselis *et al.* 2011] Manouselis N. , Drachsler H. , Vuorikari R. , Hummel H. , and Koper R . Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer, 2011.

[Martinez *et al.* 2008] Martinez L. , Barranco M. J. , Pérez L. G. , and Espinilla M . A knowledge based recommender system with multigranular linguistic information. *International Journal of Computational Intelligence Systems*, 1(3):225–236, 2008.

[Martínez *et al.* 2009] Martínez A. B. B. , Arias J. J. P. , Vilas A. F. , Duque J. G. , and Nores M. L . What's on tv tonight? an efficient and effective personalized recommender system of tv programs. *Consumer Electronics, IEEE Transactions on*, 55(1):286–294, 2009.

[Mashat *et al.* 2013] Mashat A. F. , Fouad M. M. , Yu P. S. , and Gharib T. F . Discovery of association rules from university admission system data. *International Journal of Modern Education & Computer Science*, 5(4), 2013.

[Matatov *et al.* 2010] Matatov N. , Rokach L. , and Maimon O . Privacy-preserving data mining: A feature set partitioning approach. *Information Sciences*, 180(14):2696–2720, 2010.

[McGregor *et al.* 2012] McGregor C. , Catley C. , and James A . A process mining driven framework for clinical guideline improvement in critical care. In *Learning from Medical Data Streams 13th Conference on Artificial Intelligence in Medicine (LEMEDS)*, volume 765. Citeseer, 2012.

[Medhi and Dakua 2005] Medhi D. G. and Dakua J . Moviereco: A recommendation system. In *WEC (2)*, pages 70–73. Citeseer, 2005.

[Melville and Sindhwani 2010] Melville P. and Sindhwani V . Recommender systems. *Encyclopedia of Machine Learning*, pages 2–11, 2010.

[Melville *et al.* 2002] Melville P. , Mooney R. J. , and Nagarajan R . Content-boosted collaborative filtering for improved recommendations. In *in Eighteenth National Conference on Artificial Intelligence*, 2002.

[Merceron and Yacef 2008] Merceron A. and Yacef K . Interestingness measures for association rules in educational data. In *Educational Data Mining 2008: The 1st International Conference on Educational Data Mining Montréal, Québec, Canada, June 20-21, 2008 Proceedings 1*, pages 18–27, Montreal, Canada, June 2008.

[Milne and Rohm 2000] Milne G. R. and Rohm A. J . Consumer privacy and name removal across direct marketing channels: Exploring opt-in and opt-out alternatives. *Journal of Public Policy & Marketing*, 19(2):238–249, 2000.

[Minaei-Bidgoli and Punch 2003] Minaei-Bidgoli B. and Punch W . Using genetic algorithms for data mining optimization in an educational web-based system. In *Genetic and Evolutionary Computation—GECCO 2003*, pages 2252–2263. Springer, 2003.

[Mooney and Roy 1999] Mooney R. J. and Roy L . Content-based book recommending using learning for text categorization. In *IN PROCEEDINGS OF THE FIFTH ACM CONFERENCE ON DIGITAL LIBRARIES*, 1999.

[Moro *et al.* 2010] Moro S. , Cortez P. , and Laureano R.M.S . *A Data Mining Approach for Bank Telemarketing Using the rminer Package and R Tool.* Working Paper, June 2010.

[Moss and Atre 2003] Moss L. T. and Atre S . February 2003. *Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications.* Addison Wesley, New Jersey.

[Nabavi and Jafari 2013] Nabavi S and Jafari . Providing a customer churn prediction model using random forest and boosted trees techniques. *Journal of Basic and Applied Scientific Research*, 3(6):1018–1026, 2013.

[Niwa *et al.* 2006] Niwa S. , Honiden S. , and others . Web page recommender system based on folksonomy mining for itng'06 submissions. In *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*, pages 388–393. IEEE, 2006.

[Olson and Delen 2008] Olson D.L. and Delen D . 2008. *Advanced Data Mining Techniques.* Springer, Berlin.

[Otero and Sánchez 2006] Otero J. and Sánchez L . Induction of descriptive fuzzy classifiers with the logitboost algorithm. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 10(9):825–835, 2006.

[Pal 2011] Pal J. K . Usefulness and applications of data mining in extracting information from different perspectives. *Annals of Library and Information Studies*, 58(March):7–16, 2011.

[Parra *et al.* 2011] Parra D. , Karatzoglou A. , Amatriain X. , and Yavuz I . Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. *Proceedings of the CARS-2011*, 2011.

[Pattamavorakun and Pattamavorakun 2010] Pattamavorakun S. and Pattamavorakun S . *Intelligent Personalization of* T*hai M-commerce as Marketing Value-added Services Tool*, 2010.

[Pazzani and Billsus 2007] Pazzani M. J. and Billsus D . Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[Pazzani 1999] Pazzani M. J . A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[Pechenizkiy *et al.* 2008] Pechenizkiy M. , Calders T. , Vasilyeva E. , and De Bra P . Mining the student assessment data: Lessons drawn from a small scale case study. In *Educational Data Mining 2008*, 2008.

[Pei and Han 2004] Pei J. and Han J . Mining sequential patterns by pattern-growth: The prefixSpan Approach. *IEEE Transactions and Knowledge on Data Engineering*, 16(10):1–17, October 2004.

[Pennock *et al.* 2000] Pennock D. M. , Horvitz E. , Lawrence S. , and Giles C. L . Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 473–480. Morgan Kaufmann Publishers Inc., 2000.

[Phyu 2009] Phyu T. N . Survey of classification techniques in data mining. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 18–20, 2009.

[Piatetsky, G 2014] *CRISP-DM, still the top methodology for analytics, data mining, or data science projects*, October 2014. Data Mining, Analytics, Big Data, and Data Science. Retrieved 28 October 2014, from `http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-pr`

[Prangl *et al.* 2007] Prangl M. , Szkaliczki T. , and Hellwagner H . A framework for utility-based multimedia adaptation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(6):719–728, 2007.

[Quinlan 2014] Quinlan J. R . 2014. *C4.5: programs for machine learning*. Elsevier.

[Rajaraman and Ullman 2012] Rajaraman A. and Ullman J. D . 2012. *Mining of massive datasets*. Cambridge University Press.

[RedHat] RedHat . *WildFly*. http://wildfly.org/. Accessed January 4, 2015.

[Resnick *et al.* 1994] Resnick P. , Iacovou N. , Suchak M. , Bergstrom P. , and Riedl J . Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[Ricci *et al.* 2011] Ricci F. , Rokach L. , and Shapira B . 2011. *Introduction to recommender systems handbook*. Springer.

[Ricci 2010] Ricci F . Mobile recommender systems. *Information Technology & Tourism*, 12(3):205–231, 2010.

[Rombouts and Verhoef 2004] Rombouts J. and Verhoef T . *A simple hybrid movie recommender system*. Online, 2004.

[Romero *et al.* 2008] Romero C. , Ventura S. , Espejo P. G. , and Hervás C . Data mining algorithms to classify students. In *EDM*, pages 8–17, 2008.

[Rouse,M 2015] *m-commerce (mobile commerce)*, January 2015. Retrieved 18 January 2015, from `http://searchmobilecomputing.techtarget.com/definition/m-commerce`

[Rud 2001] Rud O. P . 2001. *Data mining cookbook: modeling data for marketing, risk, and customer relationship management*. John Wiley & Sons.

[Salehi 2014] Salehi M . Latent feature based recommender system for learning materials using genetic algorithm. 2014.

[Saraee *et al.* 2005] Saraee M. , Khan S. , and Yamaner S . Data mining approach to implement a recommendation system for electronic tour guides. In *Proceedings of The 2005 International Conference on E-Business, Enterprise Information Systems, E-Government, and Outsourcing, EEE 2005, Las Vegas, Nevada, USA, June 20-23, 2005*, pages 215–218. CSREA Press, 2005.

[Sarwar *et al.* 1998] Sarwar B. M. , Konstan J. A. , Borchers A. , Herlocker J. , Miller B. , and Riedl J . Using filtering agents to improve prediction quality in the grouplens research

collaborative filtering system. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 345–354. ACM, 1998.

[Sarwar *et al.* 2000] Sarwar B. , Karypis G. , Konstan J. , and Riedl J . Analysis of recommendation algorithms for e-Commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.

[Sarwar *et al.* 2001] Sarwar B. , Karypis G. , Konstan J. , and Riedl J . Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[Sarwar *et al.* 2002] Sarwar B. , Karypis G. , Konstan J. , and Riedl J . Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[Schafer *et al.* 2007] Schafer J. B. , Frankowski D. , Herlocker J. , and Sen S . Collaborative filtering recommender systems. 2007.

[Schein *et al.* 2002] Schein A. I. , Popescul A. , Ungar L. H. , and Pennock D. M . Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[Schultz *et al.* 2001] Schultz M. G. , Eskin E. , Zadok . , and Stolfo S. J . Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 38–49. IEEE, 2001.

[Setten 2005] Setten V . *Supporting People in Finding Information: Hybrid Recommender Systems and Goal-Based Structuring.* PhD thesis, Telematica Instituut, Enschede, December 2005.

[Shah *et al.* 2003] Shah H. , Undercoffer J. , and Joshi A . Fuzzy clustering for intrusion detection. In *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, volume 2, pages 1274–1278. IEEE, 2003.

[Shani *et al.* 2007] Shani G. , Meisles A. , Rokach L. , Gleizer Y. , and D Ben-Shimon . *A Stereotypes-Based Hybrid Recommender System for Media Items*, May 2007.

[Sharma and Osei-Bryson 2009] Sharma S. and Osei-Bryson K . Framework for formal implementation of the business understanding phase of data mining projects. *Expert Systems with Applications*, 36(2):4114–4124, 2009.

[Sharma *et al.* 2012] Sharma M. , Choudhary J. , and Sharma G . Evaluating the performance of apriori and predictive apriori algorithm to find new association rules based on the statistical measures of datasets. In *International Journal of Engineering Research and Technology*, volume 1. ESRSA Publications, 2012.

[Shaw *et al.* 2001] Shaw M. J. , Subramaniam C. , Tan G. W. , and Welge M. E . Knowledge management and data mining for marketing. *Decision support systems*, 31(1):127–137, 2001.

[Shearer 2000] Shearer C . The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22, 2000.

[Shen and Chuang 2009] Shen C. and Chuang H . A study on the applications of data mining techniques to enhance customer lifetime value. *WSEAS Transactions on Information Science and Applications*, 6(2):319–328, 2009.

[Shepitsen *et al.* 2008] Shepitsen A. , Gemmell J. , Mobasher B. , and Burke R . Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266. ACM, 2008.

[Shinde and Kulkarni 2011] Shinde S. K. and Kulkarni U. V . Hybrid personalized recommender system using modified fuzzy C-Means clustering algorithm. *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, 1(4):88, 2011.

[Singh *et al.* 2011] Singh S. , Sharma G. D. , and Mahendru M . Role of value added services in shaping Indian telecom industry. *Available at SSRN 1905423*, 2011.

[Sophatsathit 2013] Sophatsathit N . The use of recommender systems in decision support–a case study on used car dealers. *World Academy of Science, Engineering and Technology*, 77:318–322, May 2013.

[Spiegel *et al.* 2009] Spiegel S. , Kunegis J. , and Li F . Hydra: a hybrid recommender system [cross-linked rating and content information]. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 75–80. ACM, 2009.

[Su and Khoshgoftaar 2009] Su X. and Khoshgoftaar T. M . A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[Symeonidis *et al.* 2007] Symeonidis P , A. Nanopoulos , Papadopoulos A.N. , and Manolopoulos Y . Collaborative recommender systems: Combining effectiveness and efficiency. *Expert Systems with Applications*, pages 1–19, May 2007.

[Symeonidis 2008] Symeonidis P . Content-based dimensionality reduction for recommender systems. In *Data Analysis, Machine Learning and Applications*, pages 619–626. Springer, 2008.

[Takács *et al.* 2008a] Takács G. , Pilászy I. , Nemeth B. , and Tikk D . Investigation of various matrix factorization methods for large recommender systems. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 553–562. IEEE, 2008.

[Takacs *et al.* 2008b] Takacs G. , Pilászy I. , Németh B. , and Tikk D . Matrix factorization and neighbor based algorithms for the Netflix Prize problem. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 267–274, New York, NY, USA, 2008. ACM.

[Tan *et al.* 2006] Tan P.N. , Steinbach M. , and Kumar V . 2006. *Introduction to Data Mining: Instructor's Solution Manual*. Addison-Wesley, New Jersey.

[Tobias 2001] Tobias S . Finding association rules that trade support optimally against confidence. In *5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 424–435. Springer, 2001.

[Tran and Cohen 2000] Tran T. and Cohen R . Hybrid recommender systems for electronic commerce. In *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press*, 2000.

[Tunc and Dag 2006] Tunc B. and Dag H . Generating classification association rules with modified apriori algorithm. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Databases*, pages 384–387, 2006.

[Turney 1995] Turney Peter D . Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of artificial intelligence research*, pages 369–409, 1995.

[Ungar and Foster 1998] Ungar L. H. and Foster D. P . Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129, 1998.

[Vaidya and Clifton 2002] Vaidya J. and Clifton C . Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.

[Vodacom VoucherCloud 2014] *VoucherCloud*, August 2014. vouchercloud as a service is FREE to anyone. The app is FREE however normal data usage rates will apply when you download or access the app on your device. If you dial into the vouchercloud USSD service on *120*100# to redeem a deal, it will be free for Vodacom users. Non-Vodacom users will be charged 20 cents for 20 seconds. Retrieved 03 August 2014, from `http://www.vodacom.co.za/vodacom/services/app-store/vouchercloud`

[Vozalis and Margaritis 2004] Vozalis M. and Margaritis K. G . Collaborative filtering enhanced by demographic correlation. In *AIAI Symposium on Professional Practice in AI, of the 18th World Computer Congress*, 2004.

[Vozalis and Margaritis 2006] Vozalis M. G. and Margaritis K. G . Applying SVD on generalized item-based filtering. *International Journal of Computer Science & Applications*, 3(3):27–51, 2006.

[Vozalis and Margaritis 2007] Vozalis M. G. and Margaritis K. G . Using SVD and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, 177(15):3017–3037, 2007.

[Wang and Han 2007] Wang J. and Han J . *Frequent Closed Sequence Mining without Candidate Maintenance*, 2007.

[Wang *et al.* 2015] Wang S. , Zheng Z. , Wu Z. , Lyu M. R. , and Yang F . Reputation measurement and malicious feedback rating prevention in web service recommendation systems. *Services Computing, IEEE Transactions on*, 8(5):755–767, 2015.

[Weiss 2005] Weiss G. M . Data mining in telecommunications. In *Data Mining and Knowledge Discovery Handbook*, pages 1189–1201. Springer, 2005.

[Wirth and Hipp 2000] Wirth R. and Hipp J . CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, pages 29–30, 2000.

[Witten *et al.* 2011] Witten I.H. , Frank E. , and Hall M.A . 2011. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kauffman, Sydney.

[Woolf and Wang 2000] Woolf P. J. and Wang Y . A fuzzy logic approach to analyzing gene expression data. *Physiological Genomics*, 3(1):9–15, 2000.

[Yu *et al.* 2004] Yu K. , Schwaighofer A. , Tresp V. , Xu X. , and Kriegel H . Probabilistic memory-based collaborative filtering. *Knowledge and Data Engineering, IEEE Transactions on*, 16(1):56–69, 2004.

[Zhang and Huang 2008] Zhang B. C. and Huang J . The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, 36(4):1567–1594, 2008.

[Zhang and Zhou 2004] Zhang D. and Zhou L . Discovering golden nuggets: Data mining in financial application. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS*, 34(4):513, 2004.

[Ziegler 2005] Ziegler C . Semantic web recommender systems. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 521–521. Springer, 2005.

[Zigkolis *et al.* 2009] Zigkolis C. , Kompatsiaris Y. , and Vakali A . Information analysis in mobile social networks for added-value services. In *The W3C Workshop on the Future of Social Networking, Barcelona*, 2009.