MSc Research Report

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

SPAM MAIL DETECTION USING DATA MINING CLASSIFIERS

by

Sabelo Khalani

Supervisor

Ms. Y. Chhana

A Research Report submitted to the Faculty of Science, University of the Witwatersrand, in partial fulfilment of the requirements for the degree of Master of Science

September 23, 2021

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deep and sincere gratitude to the University of the Witwatersrand for giving me an opportunity to fulfill my dream. To the School of Statistics and Actuarial Science for giving me the opportunity to write a Masters research report. Most of all, I am fully indebted to my supervisor Ms. Yoko Chhana for providing me an invaluable guidance throughout the research report, her understanding, patience, great work ethics, encouragement and pushing me further than I thought I could go. It was a great privilege and honour to study under her guidance. To my family and friends for helping me to survive all the stresses since I started the research report.

Declaration

I declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

Signature of candidate

23rd September 2021

List of Acronyms

NN	Neural Networks
RF	Random Forest
SVM	Support Vector Machines
NB	Naïve Bayes
LDA	Linear Discriminant Analysis
ROC	Receiver Operating Curve
AUC	Area Under the Curve

Chapter 1

Introduction

1.1 Introduction

Kumar, Poonkuzhali and Sudhakar (2012) state that email is the quickest and most cost-effective medium of communication, allowing internet users to easily transfer information from anywhere in the world in a fraction of second. Rathi and Pareek (2013) and Basavaraju and Prabhakar (2010) state that if the email address and sender's identity are both anonymous, and the communication is sent to a group of receivers who are not interested in receiving it, then the email is characterised as spam mail. According to Basavaraju and Prabhakar (2010), email is subject to spam because of its widespread use and low cost.

According to Kumar *et al.* (2012) "telemarketers are able to access personal information from service providers and at times service providers sell the information to telemarketers. Hence a user's personal information could get accessed by others without the users permission. Telemarketers use the personal information to send bulk emails to recipients without gaining their permission to do so". As a result, the recipient's mailbox gets bombarded with emails, resulting in spam email.

1.2 Significance of the study

The importance of spam mail detection has become more essential than ever (Kiwanuka, Alqatawna, Amin, Paul and Faris, 2019). According to Cook, Manderson, Hartnett and Scanlan (2005), email spam is a big problem that causes financial harm to businesses and negatively impacts email users. Manually deleting spam mails from an inbox is inefficient and if the user is at work then it is also costing their employer money in lost productivity. Spam mail has also been known to cause network outages. Kelin (2001) states that money spent on internet access, time spent downloading, reading and deleting the spam are examples of expenses imposed on the recipient.

1.3 Aims and Objectives

The aim of this research report is to classify spam mail from nonspam mail. The email subject line is used to detect spam emails. Data mining techniques such as Random Forest (RF), Support Vector Machines (SVM), Neural Networks (NN), and Naïve Bayes (NB), as well as Linear Discriminant Analysis (LDA), a classical statistics classification methodology, are used to classify spam mail. The proposed spam mail detection techniques have acquired prominence in the literature. The goal of this study is to compare the performance of various methods in classifying spam from nonspam under the following conditions:

1. Different dataset sizes.
2. Different training data:test data splits.
3. Different feature sizes.

The above conditions are explored under different levels of spam versus nonspam in the datasets. On the basis of the performance of different methods, the findings are compared and recommendations are given.

1.4 Limitations of the study

1.4.1 Adequacy of the sample size

Chapter 3, page 10, line 7 describes the sample size that was used in the study as being 3092, however, experiment 1 (section 4.1) only considered the case where spam<nonspam, since there were 1035 files and the study required higher sample sizes such as 2500.

1.4.2 Financial resources

The use of SAS University Edition as an additional software for data analysis was preferred, however, in order to use the software one needs to be connected to the internet at all times. Data or WiFi is expensive and sometimes the network coverage is poor, hence this limited the use of this software.

Chapter 2

Literature Review

Individuals and businesses are becoming increasingly reliant on email to network and communicate information and intelligence as information and web applications evolve. However, spam is a blight of email communication (Yu and Xu, 2007).

Youn and McLeod (2007) used the following data mining categorisation algorithms for spam mail classification: Neural Networks, Support Vector Machines, Naïve Bayes and Decision Trees (J48 - which creates a binary tree). The effect of altering the quantity of datasets on the proposed data mining classification approaches was studied namely: 1000, 2000, 3000, 4000 and 4500. The experiment was performed with 55 features. Feature size refers to quantity of important words chosen from a text. The study also looked at how varied feature sizes affected the performance of the approaches namely 10, 20, 30, 40, 50 and 55. The research discovered that Decision Trees and Naïve Bayes performed well over a wide range of data sets and email feature sizes.

Like Youn and McLeod (2007), Yu and Xu (2007) used similar data mining classification techniques namely Naïve Bayes, Neural Networks, Support Vector Machines for spam mail detection and evaluated their performance, but also included another technique called Relevance Vector Machine. The difference with this study is that it looked at the use of training and testing data splits and it had much larger feature sizes. The following training and testing dataset splits were used to evaluate the suggested data mining grouping techniques: 20:80, 30:70, 40:60, 50:50, 60:40 and 70:30, respectively to test if this has any effect on the technique's performance. The experiment was performed with 100 features extracted from LINGER (LINGER is a sofware that converts the emails into a suitable format). The study also looked at how varied feature sizes affected the performance of the approaches namely: 60, 70, 80, 90, 100, 110, 120, 130 and 140. The research found that the Support Vector Machine and Relevance Vector Machine approaches consistently outperformed the Naïve Bayes classifier across multiple training and testing dataset splits and feature sizes. The study also discovered that the performance of the Neural Networks technique is affected by the amount of the training dataset, and

that employing Neural Networks as a spam detection strategy is ineffective. Youn and McLeod (2007), on the other hand, did not divide the data into a training and test dataset, and their research indicated that Naïve Bayes outperformed Support Vector Machines.

Like Youn and McLeod (2007) and Yu and Xu (2007), Tang, Krasser, He, Yang and Alperovitch (2008) used Support Vector Machines (SVM) as one of the methods in their study. Tang *et al.* (2008) also looked at Random Forest (RF). Since the dataset was huge (records of about half a million) and the spam rate was higher than nonspam, a sample from the dataset was analysed. The dataset was then separated into three equal-sized subsets with a 2:1 ratio between the spam and nonspam. The training dataset was divided into two subsets, while the testing dataset was divided into one subset. The experiment was repeated to find a classifier that is accurate, quick to process, and consumes less computing capacity. The research found that the Random Forest is more accurate than Support Vector Machines in categorising spam, although it takes longer to process and utilises more computational memory. The modelling parameters namely, γ (width of a feature) and C (the regulating parameter that is utilised to trade-off model complexity and training accuracy) were fixed. As a result, when compared to Random Forest, the Support Vector Machines technique achieved the same accuracy outcomes in a more efficient manner.

Halees (2009) used the following techniques for spam mail identification: Maximum Entropy (ME), Decision Trees (DT), Naïve Bayes, Support Vector Machines, k-Nearest Neighbour (kNN) and Neural Networks (NN). Mutual Information (MI), which is the amount of information that one random variable contains about another, was computed for each word in the email message for feature selection and features with the highest MI score were selected for analyses. Ten-fold cross validation procedure was employed. The inbox messages were divided into ten segments. Nine of them were utilised for training and one was used for testing. Support Vector Machines were shown to be the most effective in the study. This is consistent with Yu and Xu (2007) and Halees (2009) who reported comparable outcomes in their studies with Support Vector Machines being more superior in performance than other spam mail data mining classifiers such as Naïve Bayes, when using the training and test dataset.

Lakshmi and Radha (2010) applied similar methods to those applied by Youn and McLeod (2007) namely, Neural Networks and Naïve Bayes, however, they included Linear Discriminant Analysis (LDA) as a possible spam mail detection technique. The data was additionally separated into a training and a test dataset using ten-fold cross validation, which was not done by Youn and McLeod (2007). Two software tools namely Weka and Rapid Miner were used to compare the results. The study found that Neural Networks performed the best, followed by Linear Discriminant Analysis.

Awad and ELseuofi (2011) used similar methods namely, Support Vector Machines, Neural Networks and Naïve Bayes to the study done by Youn and McLeod (2007) but also included k-Nearest Neighbour, Artificial Immune System (which is frequently used to distinguish between elements that are beneficial and those that are toxic), and Rough Sets (which classifies email messages into three groups namely spam mail, unsure mail and nonspam mail) as a possible spam mail detection techniques. This study also differed from Youn and McLeod (2007) in that the data was split into two datasets: one for training and one for testing. The experiment is carried out using the most commonly used words in spam email resulting in 100 of them being used as features. Awad and ELseuofi (2011), like Youn and McLeod, got similar accuracy performance findings, demonstrating that Naïve Bayes is superior than other approaches.

Like Tang *et al.* (2008), Kumar, Poonkuzhali and Sudhakar (2012) employed Support Vector Machine (SVM) and Random Forest (RF) in their study and also looked at several other classification methods such as k-Nearest Neighbour, Neural Networks, Naïve Bayes, Decision Trees (C4.5 and ID3), Cart (C-RT, CS-MC4 and CS-CRT), Regression Techniques(PLS-DA), Linear Discriminant Analysis (PLS-LDA) and Log Regression TRIRLS. The dataset contained fifty-seven continuous element attributes and one discrete target attribute. Before feature selection, feature reduction was done using the following techniques namely, Correspondence Analysis, Canonical Discriminant Analysis and Principal Component Analysis. For finding relevant features, the dataset was subjected to Fisher filtering, ReliefF, Runs filtering, and Step disc feature selection procedures. According to the study, Fisher and Runs filtering performed better classification for several classifiers. Tang *et al.* (2008) and Kumar *et al.* (2012) obtained similar accuracy findings, demonstrating that the Random Forest (RF) method outperformed other methods.

Rathi and Pareek (2013) employed similar methods namely, Support Vector Machines, Random Forest, Naïve Bayes, Decision Trees and Cart to the study done by Kumar *et al.* (2012) and further incorporated Bayesian Networks, Random Tree and Function Tree spam mail classification techniques. The main objective of using many classifiers was to determine the best classifier that outperforms the others in terms of accuracy. They employed Best-First algorithm (which selects n best features for modeling a given dataset) for feature selection. Kumar *et al.* (2012) and Rathi and Pareek (2013) obtained similar accuracy performance findings demonstrating that the Random Forest (RF) method is superior than other techniques.

Bibi, Latif, Khalid, Ahmed, Shabir and Shahryar (2020) employed similar methods namely, Support Vector Machines and Naïve Bayes. A dataset with 960 emails is used for the modeling, training and testing. The selected dataset contains two categories of document: nonspam and spam. In this study, about 260 emails were extracted randomly to build the testing dataset for the clas-

sifier. The other 700 emails were used as the training dataset to train the classifier and 1557 attributes were selected. Word frequency was chosen as the text feature. Naïve Bayes performed better than Support Vector Machines.

Table 2.1 summarises the methods used by author and timeline to highlight and provide a quick overview of the literature.

Table 2.1: Literature summary table (method used overtime).

Methods used by Authors									
Method	Youn (2007)	Yu (2007)	Tang (2008)	Halees (2009)	Lakshmi (2010)	Awad (2011)	Kumar (2012)	Rathi (2013)	Bibi (2020)
NN	✓	✓		✓	✓	✓	✓		
RF			✓				✓	✓	
SVM	✓	✓	✓	✓		✓	✓	✓	✓
LDA					✓		✓		
NB	✓	✓		✓	✓	✓	✓	✓	✓
Decision Trees	✓			✓			✓	✓	
RVM		✓							
ME				✓					
KNN				✓		✓	✓		
Artificial Immune System						✓			
Rough Sets						✓			
Cart							✓	✓	
Regression Techniques							✓		
Log Regression TRIRLS							✓		
Bayesian Net								✓	
Random Tree								✓	
Function Tree								✓	
Methods recommended by author	NB, DT	SVM, RVM	SVM similar to RF	SVM	NN, LDA	NB	RF	RF	NB

The literature summary table points out that Support Vector Machines, Naïve Bayes, Neural Networks and Random Forest were the most common methods

for separating spam from nonspam mail with each being recommended as the best method by a third of the papers reviewed under various conditions namely: varying sizes of training versus test datasets, use of multiple training datasets and the use of one test dataset as opposed to one training dataset and one test dataset. Support Vector Machines (SVM) were consistently proven to be superior in classifying spam and nonspam messages under these changing conditions.

Chapter 3

Methodology

Random Forest, Support Vector Machine, Naïve Bayes and Neural Networks are the proposed data mining classification algorithms whilst Linear Discriminant Analysis is part of classical statistics.

According to Meira and Zaki (2014), Data mining is the process of extracting intuitive and unusual patterns, as well as descriptive, intelligible and predictive models from data. Data Classification, Regression and Time Series Analysis are examples of predictive models. Association Rules, Summarisation, Clustering and Sequence Discovery are examples of descriptive models. Han, Kamber and Pei (2011: 11) state that data mining is a "knowledge discovery process" which involves sequentially "cleaning the data, data integration, data selection, data transformation, pattern discovery, pattern evaluation and knowledge presentation".

The performance of the methods is assessed by using a confusion matrix, thus, the performance is measured in terms of accuracy, precision, recall, hitrate and F1 score. According to Rahmad, Suryanto and Ramli (2020) "there are four terms in the confusion matrix that describe the classification of performance measurement results, namely True Negative (tn), False Positive (fp), True Positive (tp), and False Negative (fn)".

Table 3.1: Confusion Matrix.

		True Values	
		True	**False**
Prediction	**True**	tp (correct result)	fp (unexpected result)
	False	fn (absent result)	tn (correct absence of result)

$$\begin{cases} tp = \text{true positive} = \text{emails that have been accurately categorised as spam} \\ tn = \text{true negative} = \text{emails that have been accurately identified as nonspam} \\ fp = \text{false positive} = \text{emails that have been incorrectly labelled as nonspam} \\ fn = \text{false negative} = \text{emails that have been incorrectly labelled as spam} \end{cases}$$

Shboul, Hakh, Faris, Aljarah and Alsawalqah (2016) explain that accuracy is equal to the number of items classified correctly divided by the total number of items.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.1}$$

Precision is defined as the proportion of estimated legitimate email that was correctly classified,thus, it is the ratio of true positives to predicted positives.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{3.2}$$

Recall is the percentage of estimated nonspam in the email data consisting of spam and nonspam,thus, it is the ratio of true positives over all positives.

$$\text{Recall (Sensitivity)} \ = \frac{tp}{tp + fn} \tag{3.3}$$

and the Hitrate is defined as the percentage of estimated spam in the email data consisting of spam and nonspam.

$$\text{HitRate} = \frac{tn}{tn + fn} \tag{3.4}$$

Bibi *et al.*(2020) explain that F1 score/measure is the harmonic mean of recall and precision,thus, F1 score conveys the balance between the precision and the recall. F1 score is used to find the optimal results of recall and precison. The F1 score formula is defined as follows;

$$\text{F1 score} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{3.5}$$

"Discarding a legitimate email is of greater concern to most users than classifying a spam message as legitimate email. This means that a high precision is particularly important" Yu and Xu (2007). According to Shams and Mercer (2013) "a higher spam recall rate shows that the probability of incorrectly classifying spam email as nonspam email is lower". The above mentioned measures of accuracy, precision, recall, hitrate and F1 score will be applied to each of the classification methods that will be discussed in the following sections. Furthermore, the Receiver Operating Curve will be used to visualise the effectiveness of the classifier. According to Jukic, Keco and Kevric (2015) "ROC can provide an extensive estimation of a classifier's effectiveness:

$$\text{ROC} = \frac{P(x|\text{positive})}{P(x|\text{negative})} \tag{3.6}$$

where $P(x|C)$ implies the likelihood that a sample belongs to the class C. In other words, ROC represents a function of the classifier's sensitivity/recall and specificity values". According to Basavaraju and Prabhakar (2010), Specificity is defined as

$$\text{Specificity} = \frac{tn}{tn + fp} \tag{3.7}$$

The Receiver Operating Characteristics (ROC) curve depicts how the recall versus precision relationship changes as the model's spam detection threshold is changed. The higher the area under the curve (AUC),thus, the further the graph is from the 45° straight line, the higher the performance of the model.

The analyses are based on the publicly available dataset called CEAS2008. The dataset contains 140 000 emails some of which are spam and others ham (also known as nonspam). A sample of 3 092 emails is selected from the 140 000 emails. Analysis in this study is done using R package. Emails are imported into R, the email dataset is then preprocessed by converting the text to lowercase, removing all punctuation from the corpus, removing all English stopwords and stemming of the attributes. The main goal of the preprocessing technique is to build a dictionary of the words. Each word is referred to an index which is a integer number, and then the occurrence of each word in the document is counted to build a feature vector with word counts. Term frequency is then chosen as the text feature. Raschka (2014) defines term frequency as the number of times a specific term t (thus, token or phrase) appears in a document d. The computed score of a feature is used for determining the rank of the features and top ranked features are used for model preparation.

3.1 Neural Networks (NN)

An architecture of connected neurons is known as a neural network. The input layer, hidden layer and output layer are the three layers of neurons that make up the Neural Network. The input layer is where the network's input variables are entered. To incorporate non-linearity into the Neural Network, the input variables are altered in the hidden layer using various functions. The value of the target variable is predicted in the output layer.

Tretyakov (2004) explains that "each neuron in the network takes input values $x_1, x_2, \ldots, x_k$ and calculates its output value o by using the formula

$$o = \phi(\sum_{i=1}^{k} w_i x_i + b). \tag{3.8}$$

where w_i and b are weights and the bias of the neuron respectively and ϕ is a certain non-linear function where $\phi(x)$ is either $\frac{1}{1+e^{\alpha x}}$ or $\tanh(x)$".

3.1.1 Advantages and Disadvantages

The advantages and disadvantages of Neural Networks can be summarised by the following points (Mohamed, 2017):
Advantages:

(i) Can be used to solve linear and nonlinear programming problems.

(ii) The ability to learn from provided examples makes them powerful and flexible; allowing neural networks to learn without having to be re-programmed.

(iii) Has been successful for solving many classification, clustering and regression problems.

Disadvantages:

(i) The success of the model depends on the quantity of the data.

(ii) Lack of clear guidelines for which neural networks architecture is optimal because the process involves trial and error.

3.2 Random Forest (RF)

The Random Forest model is part of the ensemble model family. These models incorporate predictions from a number of different models to provide a single consensus prediction. Thus, predictions from k models, $M_1, M_2, \ldots, M_k$ are combined to create M_*.

According to Han *et al.* (2011: 319), "a given dataset D is used to create k training sets, $D_1, D_2, \ldots, D_k$, where $D_i (1 \leq i \leq k-1)$ is used to generate classifier M_i. For each iteration, $i (i = 1, 2, \ldots, k)$, a training set D_i of D tuples is sampled with replacement from D. The CART methodology is used to grow the trees". The trees are allowed to reach their full potential and are not trimmed.

3.2.1 Advantages and Disadvantages

The advantages and disadvantages of Random Forest can be summarised by the following points (Prajwala, 2015):
Advantages:

(i) Provides accurate predictions for many types of applications.

(ii) Can measure the importance of each feature with respect to the training dataset.

(iii) Pairwise proximity between samples can be measured by the training dataset.

Disadvantages:

(i) For data including categorical variables with different number of levels, random forests are biased in favour of those attributes with more levels.

(ii) If the data contain groups of correlated features of similar relevance for the output, then smaller groups are favoured over larger groups.

3.3 Support Vector Machines (SVM)

SVM is a classification approach that uses a linear classifier to divide data into two groups.

Consider a training dataset T_r with n samples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where x_i is a feature vector in a d-dimensional feature space R^d and $y_i \in \{-1, +1\}$ is the corresponding class label, $1 \leq i \leq n$, the task is to find a classifier with a decision function $f(x, \theta)$ such that $y = f(x, \theta)$, where y is the class label for x, θ is a vector of unknown parameters in the function.

Tang *et al.* (2008) and Islam, Chowdhury and Zhou (2005), explain that "Support Vector Machine (SVM) is a modelling technique that finds an optimal hyperplane with the maximal margin to separate two classes, which requires that the following optimisation problem be solved.

maximise:

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j). \tag{3.9}$$

subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{3.10}$$

for $0 \leq \alpha_i \leq$ C, i =1, 2,. . . ,n". The α_i are the weights, which is attached to the training sample x_i. If $\alpha_i > 0$, x_i is called a support vector. C (regulation parameter) is a compromise between model complexity and model precision. K is a kernel function that determines how similar two samples are.

3.3.1 Advantages and Disadvantages

The advantages and disadvantages of Support Vector Machines can be summarised by the following points (Mohamed, 2017):
Advantages:

(i) Training the model is relatively easy.

(ii) Can deal with both continuous and categorical data.

(iii) The trade-off between the model complexity and the error can be controlled easily.

Disadvantages:

(i) Difficulty to interpret unless the features are interpretable.

(ii) Can be computationally expensive and it needs a good kernel function.

(iii) Lack of transparency in results because it is a non-parametric method.

3.4 Naïve Bayes (NB)

Naïve (NB) requires the collection of emails to be pre-classified as spam or nonspam after a large number of emails are classified according to the words that are present and absent in each email. When the recipient receives a new email, the probability that a new email is spam or nonspam will be computed based on the training dataset. The NB method makes independent assumptions about the influence of a feature's value on a specific class in comparison to other features. NB uses Bayes' Theorem to perform classification.

Yu and Xu (2007) describe that a given feature vector $\vec{x} = \{x_1, x_2, x_3, \ldots, x_n\}$ of an email, as the values of attributes $X_1, \ldots, X_n$ and n denotes the number of attributes in the email folder. Let C be the set to be estimated, thus, $C \in \{spam, non\text{-}spam\}$. It applies Bayes Rule to calculate the conditional probabilities of $P(C_i|X)$. "Given the inputs, $P(C_i|X)$ denotes the probability that, example X belongs to class C_i:

$$P(C_i|X) = \frac{P(C_i) \times P(X|Ci)}{P(X)} \tag{3.11}$$

P(C_i) is the probability of observing class i. $P(X|C_i)$ denotes the probability of observing the example, given class C_i. $P(X)$ is the probability of the input, which is independent of the classes" Yu and Xu (2007) and Androutsopoulos *et al.* (2000).

"Provided that we extract the following keywords from an e-mail: Benefits (0.85) million (0.15), current (0.1) reap (0.12) need (0.2). A value of 0.85 for benefits indicates 85% of previously seen emails that included that word were ultimately classified as spam, with the remaining 15% classified as legitimate email. To calculate the overall probability (P) of an e-mail being spam:

$$\begin{aligned} P &= \frac{x_1 \cdot x_2 \cdots x_n}{x_1 \cdot x_2 \cdots x_n + (1-x_1) \cdot (1-x_2) \cdots (1-x_n)} \\ &= \frac{0.85 \cdot 0.15 \cdot 0.1 \cdot 0.12 \cdot 0.2}{0.85 \cdot 0.15 \cdot 0.1 \cdot 0.12 \cdot 0.2 + (1-0.85) \cdot (1-0.15) \cdot (1-0.1) \cdot (1-0.12) \cdot (1-0.2)} \\ &= 0.00377 \end{aligned}$$

This value indicates that it is unlikely that the email message is spam; however, the ultimate classification decision would depend on the decision boundary set by the filter" Yu and Xu (2007)

Table 3.2: A simple example of Naïve Bayes classification by Yu and Xu (2007)

3.4.1 Advantages and Disadvantages

The advantages and disadvantages of Naïve Bayes can be summarised by the following (Keogh, 2006):
Advantages:

(i) Fast to train (single scan). Fast to classify.

(ii) Not sensitive to irrelevant features.

(iii) Handles real and discrete data.

Disadvantages:

(i) Assumes independence of features.

3.5 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is a frequentist approach to statistical inference. The objective is to use training samples with known classes to classify samples with unknown classes.

Let $x_1 = \{x_1^1,\ldots,x_{l1}^1\}$ and $x_2 = \{x_1^2,\ldots,x_{l2}^2\}$ be samples from two different classes and assume that $X = X_1 \cup X_2 = \{X_1,\ldots,X_1\}$. According to Wang, Youssef and Elhakeem (2006), the linear discriminant is given by a vector W that maximises the following;

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{3.12}$$

where

$$S_B = (m_1 - m_2)(m_1 - m_2)^T, \tag{3.13}$$

$$S_W = \sum_{i=1,2} \sum_{x \in x_l} (x - m_i)(x - m_i)^T \tag{3.14}$$

are the between and within class scatter matrices respectively, m_i is the mean of the class and w is a vector. The idea behind maximising $J(W)$ is to find a direction which maximises the estimated class means while minimising the classes variance in this direction. After the linear transformation matrix W has been found, the dataset can be transformed by

$$y = W^T X. \tag{3.15}$$

3.5.1 Advantages and Disadvantages

The advantages and disadvantages of Linear Discriminant Analysis can be summarised by the following points (Zhang and Chow, 2012):
Advantages:

(i) Tends to be more effective and efficient for pattern classification and image recognition.

(ii) Focuses on achieving maximum class discrimination for classification.

Disadvantages:

(i) Requires intra-class scatter to be nonsingular.

(ii) Requires an implicit assumption that each class has a unimodal Gaussian distribution

(iii) Rank limitation on the inter-class scatters.

Chapter 4

Analysis and results

The performance of data mining techniques (SVM, RF, NN, NB) as well as classical statistics technique (LDA) in classifying spam and nonspam mail was assessed in the following experiments:

(i) Experiment 1: Investigate the effect of having different dataset sizes (feature size is fixed at 140 features).

(ii) Experiment 2: Investigate the effect of using a training dataset and a test dataset - various splits are considered (feature size is fixed at 140 features).

(iii) Experiment 3: Investigate the effect of using different feature sizes

Experiments 2 and 3 are considered for the case of spam>nonspam, spam<nonspam and spam=nonspam. Experiment 1 is considered for spam<nonspam, since there are 1035 files and the study requires higher sample sizes such as 3000.

The method's accuracy, precision, recall, hitrate, and F1 score are all used to evaluate their performance. Accuracy is equal to the number of objects successfully classified divided by the total number of objects. Precision is a confirmation metric of the legitimate emails that were correctly classified. Recall calculates the proportion of estimated nonspam in the email data consisting of spam and nonspam. Hit rate is the percentage of estimated spam in email data that includes both spam and nonspam. F1 score is the harmonic mean of recall and precision. Keeping nonspam email safe, is indicated by high precision. High Recall shows that the class is effectively perceived, this means that the classifier is able to filter every single piece of spam out of the inbox Bibi *et al.* (2020). ROC curve is also used to visualise the performance of the methods.

4.1 Experiment 1: Different Sample Sizes

4.1.1 The study looks at the performance of the various techniques across different sample sizes. The training and test dataset ratio is fixed at 80%:20%, while feature size is fixed at 140 features and spam <nonspam

Table 4.1.1: Accuracy, precision, recall and hit rate classification results based on various sample sizes.

Measure	**Sample Size**	**NN**	**RF**	**SVM**	**LDA**	**NB**
Accuracy		**87.00%**	83.00%	84.00%	86.00%	85.00%
Precision		**98.15%**	98.00%	98.04%	98.11%	98.08%
Recall	500	**81.54%**	75.38%	76.92%	80.00%	78.46%
Hitrate		**73.91%**	68.00%	69.39%	72.34%	70.83%
F1 score		**89.08%**	85.21%	86.21%	88.13%	87.18 %
Accuracy		**87.00%**	84.00%	85.00%	85.50%	84.00%
Precision		**97.27%**	97.12%	97.17%	97.20%	97.12%
Recall	1000	**82.31%**	77.69%	79.23%	80.00%	77.69%
Hitrate		**74.44%**	69.79%	71.28%	72.04%	69.79%
F1 score		**89.17%**	86.33%	87.29%	87.77%	86.33%
Accuracy		**88.33%**	85.00%	85.67%	85.00%	86.75%
Precision		**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
Recall	1500	**82.32%**	77.27%	78.28%	77.27%	78.79%
Hitrate		**74.45%**	69.39%	70.34%	69.39%	70.83%
F1 score		**90.30%**	87.18%	87.82%	87.18%	88.14%
Accuracy		**87.75%**	86.25%	87.25%	87.50%	86.75%
Precision		98.22%	**99.07%**	98.21%	98.21%	98.63%
Recall	2000	**83.08%**	80.08%	82.33%	82.71%	81.20%
Hitrate		**74.29%**	71.35%	73.45%	73.86%	72.38%
F1 score		**90.02%**	88.57%	89.57%	89.80%	89.07%
Accuracy		**89.00%**	88.00%	88.40%	88.60%	87.00%
Precision		**99.28%**	98.91%	99.27%	**99.28%**	98.53%
Recall	2500	**83.89%**	82.67%	82.98%	83.28%	81.46%
Hitrate		**76.13%**	74.67%	75.11%	75.45%	73.25%
F1 score		**90.94%**	90.06%	90.40%	90.58%	89.19%
Accuracy		87.00%	87.50%	**88.33%**	88.17%	86.83%
Precision		99.08%	**99.69%**	98.52%	99.10%	98.48%
Recall	3000	81.16%	81.41%	**83.67%**	82.91%	81.41%
Hitrate		72.63%	73.09%	**75.19%**	74.53%	72.69%
F1 score		89.23%	89.63%	**90.49%**	90.28%	89.14%

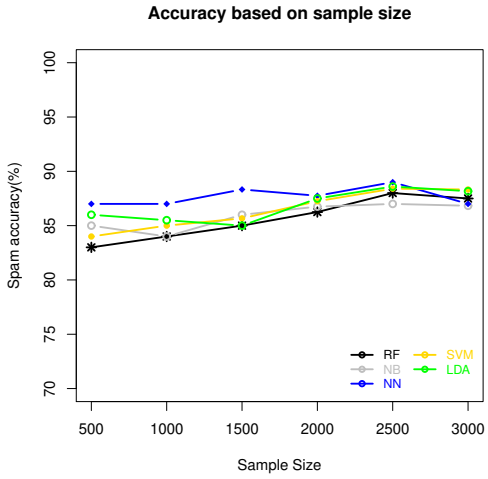

Fig. 4.1.1.1: Spam accuracy based on sample size.

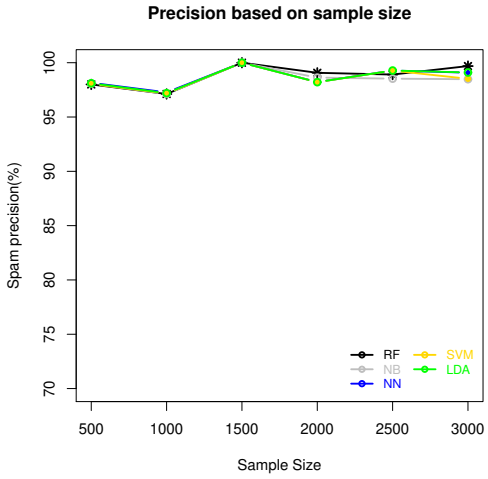

Fig. 4.1.1.2: Spam precision based on sample size.

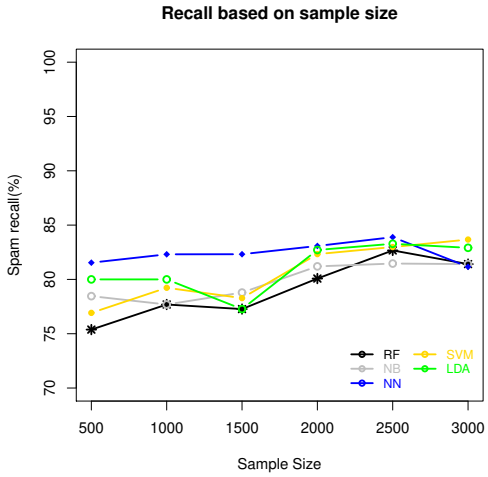

Fig. 4.1.1.3: Spam recall based on sample size.

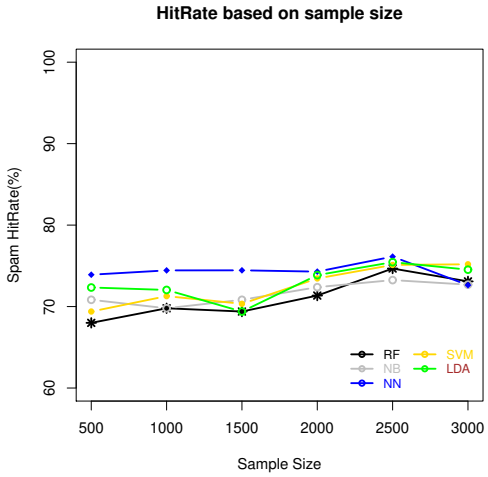

Fig. 4.1.1.4: Spam Hitrate based on sample size.

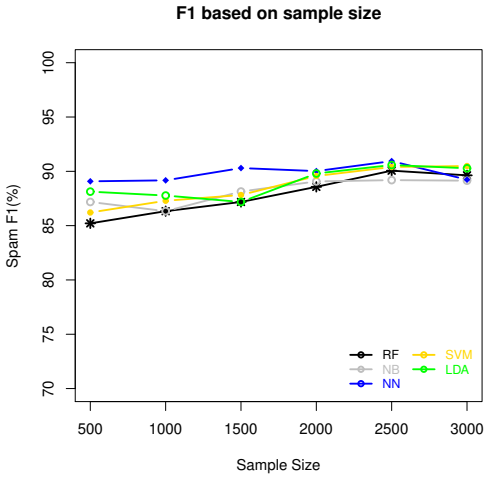

Fig. 4.1.1.5: F1 score based on sample size.

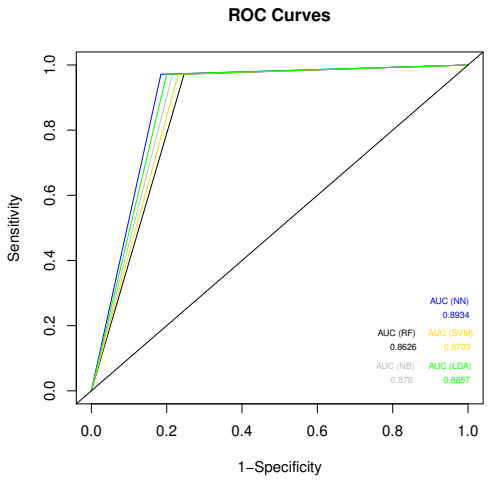

Fig. 4.1.1.6: ROC Curve - sample size 500.

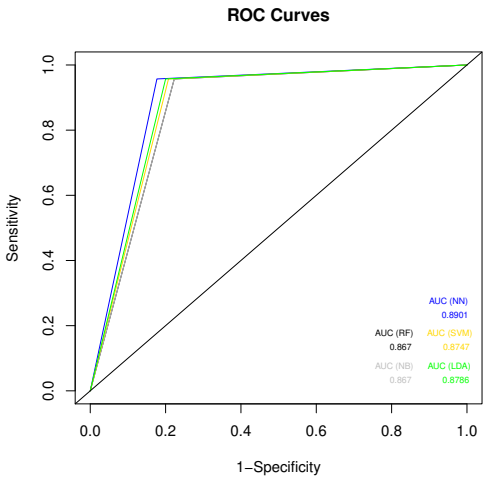

Fig. 4.1.1.7: ROC Curve - sample size 1000.

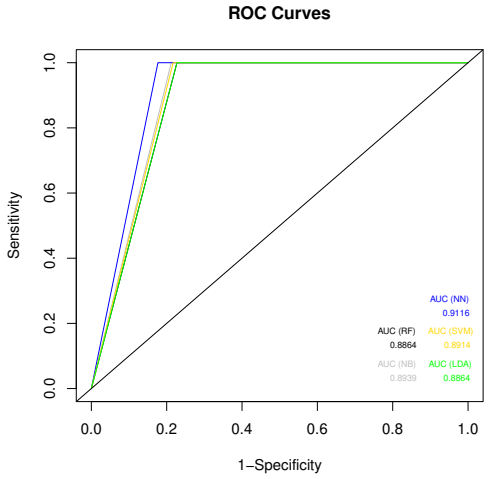

Fig. 4.1.1.8: ROC Curve - sample size 1500.

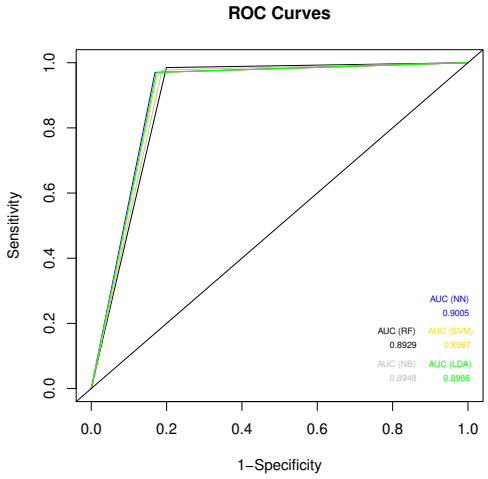

Fig. 4.1.1.9: ROC Curve - sample size 2000.

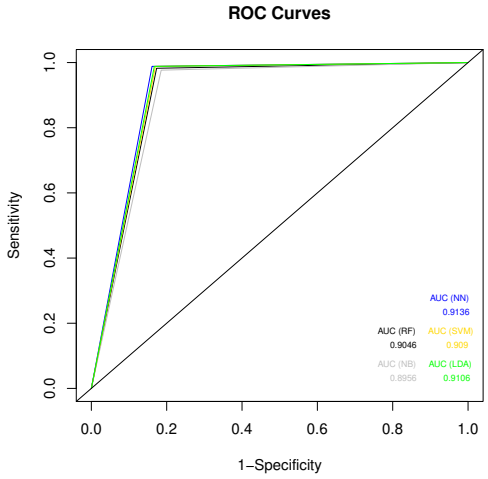

Fig. 4.1.1.10: ROC Curve - sample size 2500.

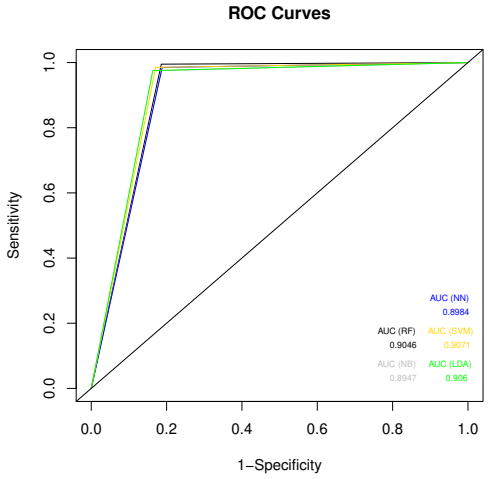

Fig. 4.1.1.11: ROC Curve - sample size 3000.

A few observations can be made from this experiment, sample size has an impact on the spam mail classification. As the sample increases, accuracy increases on average. Neural Networks performed the best in terms of accuracy, hitrate and F1 score for sample sizes 500, 1000, 1500, 2000 and 2500 as shown in Table 4.1.1 and Figures 4.1.1.1, 4.1.1.4 and 4.1.1.5 above. Support Vector Machines performed the best in terms of accuracy, recall, hitrate and F1 score for sample size 3000 as shown in Figures 4.1.1.1, 4.1.1.3, 4.1.1.4 and 4.1.1.5 above. All five methods performed similarly in terms of precision for sample size 1500 as shown in Figure 4.1.1.2 above. Random Forest is leading in terms of precision for larger sample sizes of 2000 and 3000. Linear Discriminant Analysis performed the same as Neural Networks in terms of precision for sample size 2500. In the ROC curve, Neural Networks for sample sizes 500, 1000, 1500, 2000 and 2500 is higher than the other methods. Hence Neu-

ral Networks performed the best as shown in Figures 4.1.1.6, 4.1.1.7, 4.1.1.8, 4.1.1.9 and 4.1.1.11 above. Support Vector Machines performed the best for sample size 3000 as shown in Figure 4.1.1.1 above.

4.2 Experiment 2: Training vs Test dataset split ratio

4.2.1 The study looks at the performance of the various techniques across the various training and test data split ratios for the case of spam>nonspam data[1035 vs 500]

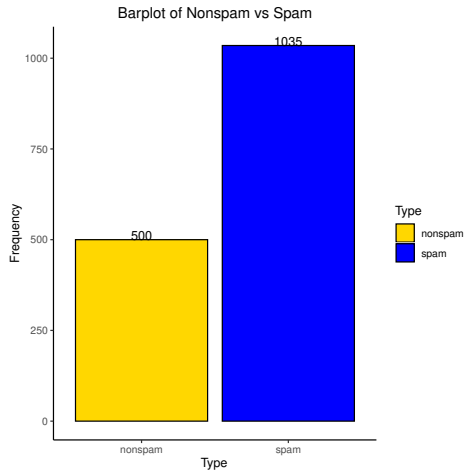

Fig. 4.2.1: Spam data greater than nonspam data.

Table 4.2.1: Accuracy, precision, recall and hit rate classification results based on various training and test data splits for case: spam >nonspam.

Measure	Ratio	NN	RF	SVM	LDA	NB
Accuracy		**93.49%**	93.30%	93.30%	92.65%	89.58%
Precision		96.98%	97.28%	97.28%	97.21%	**99.17%**
Recall	30%:70%	**82.57%**	81.71%	81.71%	79.71%	68.57%
Hitrate		**92.15%**	91.81%	91.81%	90.99%	86.79%
F1 score		**89.20%**	88.82%	88.82%	87.59%	81.08%
Accuracy		**92.62%**	92.40%	92.29%	92.29%	92.40%
Precision		96.77%	97.52%	97.51%	97.51%	**99.15%**
Recall	40%:60%	**80.00%**	78.67%	78.33%	78.33%	77.33%
Hitrate		**91.08%**	90.57%	90.44%	90.44%	90.10%
F1 score		**87.59%**	87.09%	86.87%	86.87%	86.89%
Accuracy		**92.35%**	**92.35%**	91.86%	91.53%	92.02%
Precision		96.93%	96.93%	96.88%	96.84%	**98.71%**
Recall	60%:40%	**79.00%**	**79.00%**	77.50%	76.50%	76.50%
Hitrate		**90.69%**	**90.69%**	90.09%	89.69%	89.76%
F1 score		**87.05%**	**87.05%**	86.11%	85.48%	86.20%
Accuracy		**91.97%**	91.54%	90.89%	89.59%	**91.97%**
Precision		97.48%	97.44%	97.37%	96.36%	**99.13%**
Recall	70%:30%	**77.33%**	76.00%	74.00%	70.67%	76.00%
Hitrate		**90.06%**	89.53%	88.76%	87.46%	89.60%
F1 score		**86.24%**	85.39%	84.09%	81.54%	86.04%

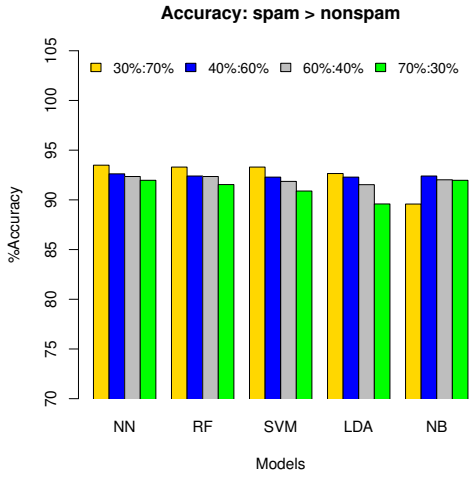

Fig. 4.2.1.1: Accuracy(spam>nonspam) based on split ratio.

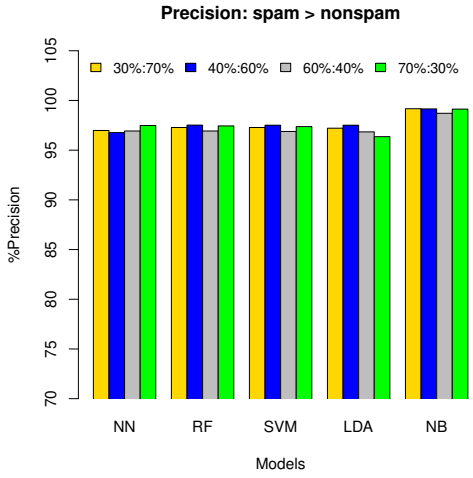

Fig. 4.2.1.2: Precision(spam>nonspam) based on split ratio.

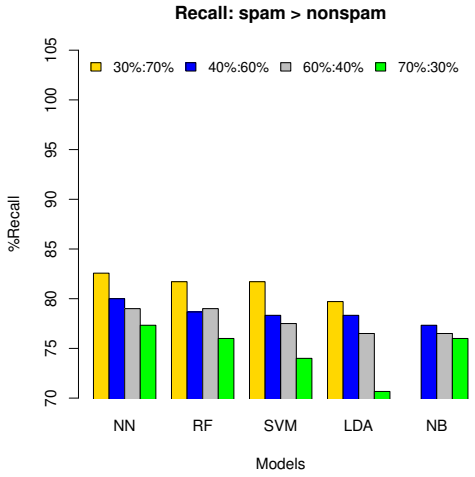

Fig. 4.2.1.3: Recall(spam>nonspam) based on split ratio.

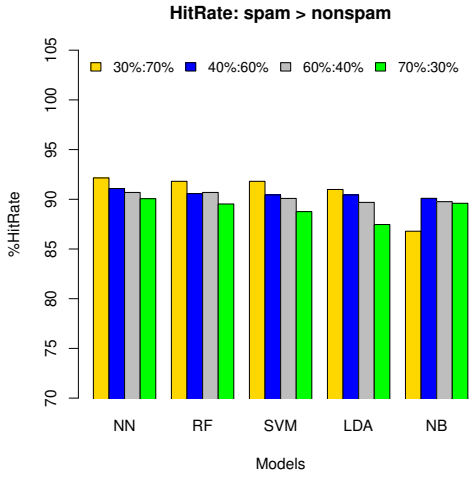

Fig. 4.2.1.4: Hitrate(spam>nonspam) based on split ratio.

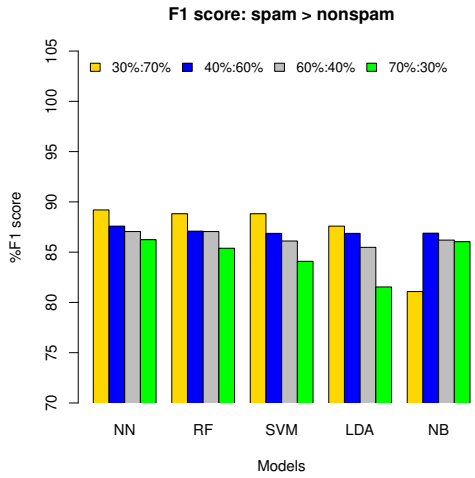

Fig. 4.2.1.5: F1 score(spam>nonspam) based on split ratio.

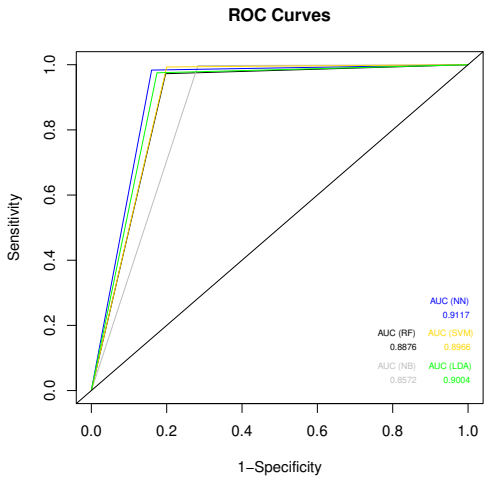

Fig. 4.2.1.6: ROC Curve - split ratio 30%:70%.

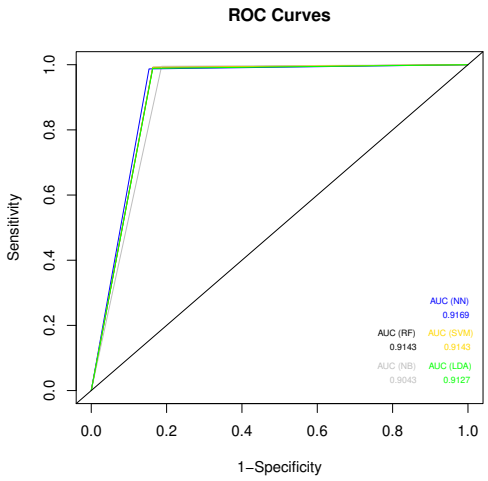

Fig. 4.2.1.7: ROC Curve - split ratio 40%:60%.

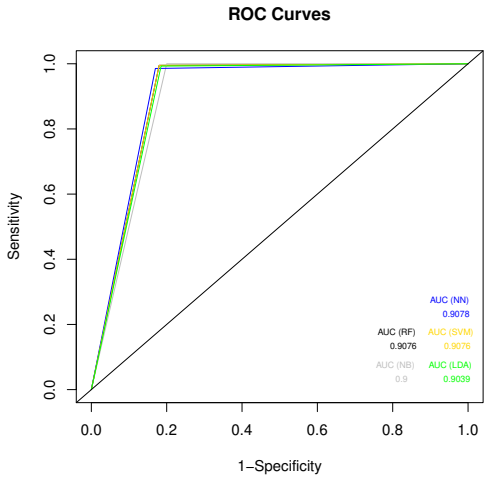

Fig. 4.2.1.8: ROC Curve - split ratio 60%:30%.

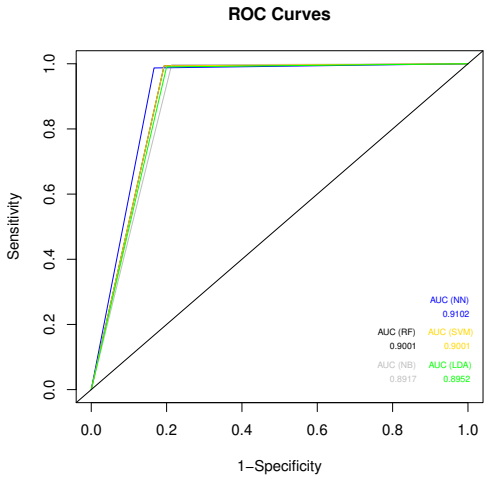

Fig. 4.2.1.9: ROC Curve - split ratio 70%:30%.

A few observations can be made from this experiment when spam exceeds nonspam. It is evident from the study that the performance of the models can also be influenced by the training and test dataset split ratios. As shown in Table 4.2.1 and Figures 4.2.1.1, 4.2.1.3, 4.2.1.4 and 4.2.1.5 Neural Networks performed the best in terms of accuracy, recall, hitrate and F1 score across all split ratios, and performed similarly with Random Forest for split ratio 60%:40%. Naïve Bayes performed the best in terms of precision across all training and test dataset split ratios as shown in Table 4.2.1 and Figure 4.2.1.2 above. The ROC curves also show that Neural Networks performed the best.

4.2.2 The study looks at the performance of the various techniques across the various training and test data split ratios for the case of spam<nonspam data[1035 vs 2057]

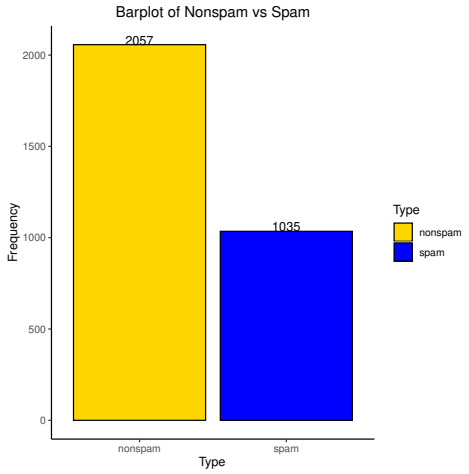

Fig. 4.2.2: Spam data less than nonspam data.

Table 4.2.2: Accuracy, precision, recall and hit rate classification results based on various training and test data splits for case: spam <nonspam.

Measure	Ratio	NN	RF	SVM	LDA	NB
Accuracy		**86.93%**	86.10%	86.00%	85.73%	86.00%
Precision		98.25%	**99.05%**	98.55%	98.21%	98.38%
Recall	30%:70%	**81.81%**	79.86%	80.14%	80.00%	74.57%
Hitrate		**72.88%**	71.12%	71.23%	70.97%	71.31%
F1 score		**89.28%**	88.43%	88.40%	88.17%	84.84%
Accuracy		**87.39%**	85.44%	86.63%	86.52%	86.85%
Precision		98.54%	**99.18%**	98.91%	98.52%	98.34%
Recall	40%:60%	**82.25%**	78.77%	80.79%	80.96%	74.57%
Hitrate		**73.45%**	70.06%	72.02%	72.06%	72.68%
F1 score		**89.66%**	87.80%	88.94%	88.88%	84.82%
Accuracy		**87.31%**	85.13%	86.58%	86.82%	86.26%
Precision		98.12%	**98.63%**	98.10%	97.83%	98.09%
Recall	60%:40%	**82.50%**	78.74%	81.41%	82.02%	81.85%
Hitrate		**73.58%**	69.83%	72.38%	72.94%	71.86%
F1 score		**89.63%**	87.57%	88.98%	89.23%	89.24%
Accuracy		**87.28%**	85.34%	85.99%	86.53%	85.56%
Precision		97.89%	97.49%	**98.03%**	97.49%	97.63%
Recall	70%:30%	80.28%	**81.60%**	80.55%	80.92%	80.23%
Hitrate		**73.71%**	72.68%	71.50%	72.68%	71.02%
F1 score		88.21%	**88.84%**	88.43%	88.44%	88.08%

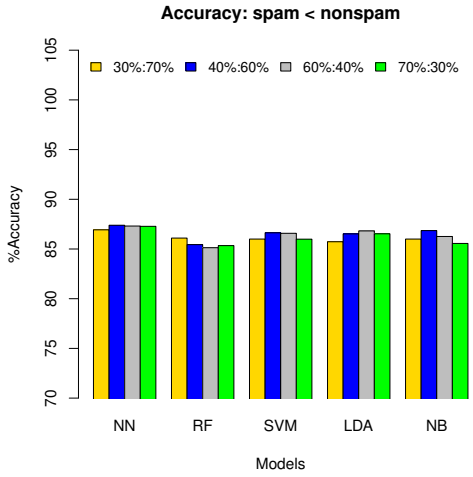

Fig. 4.2.2.1: Accuracy(spam<nonspam) based on split ratio.

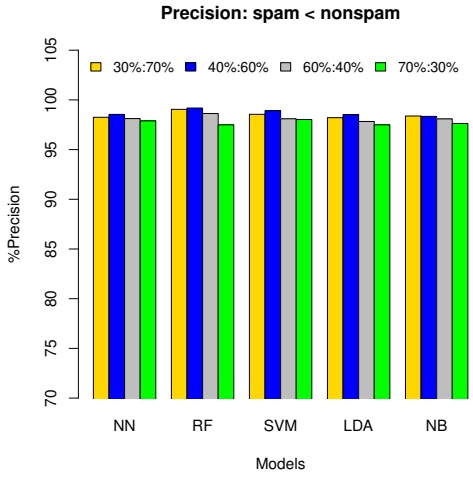

Fig. 4.2.2.2: Precision(spam<nonspam) based on split ratio.

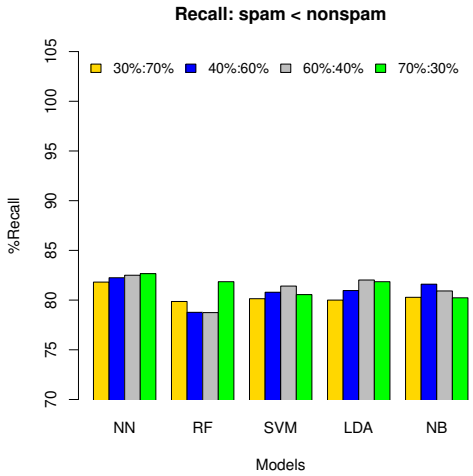

Fig. 4.2.2.3: Recall(spam<nonspam) based on split ratio.

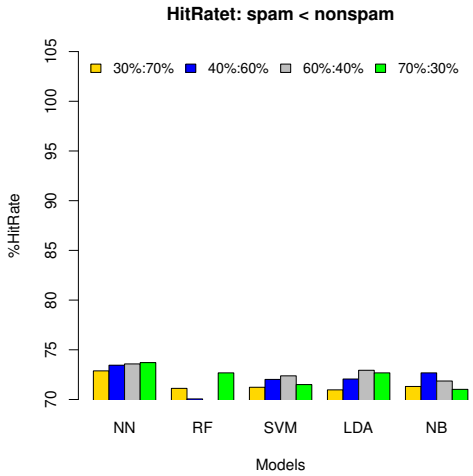

Fig. 4.2.2.4: Hitrate(spam<nonspam) based on split ratio.

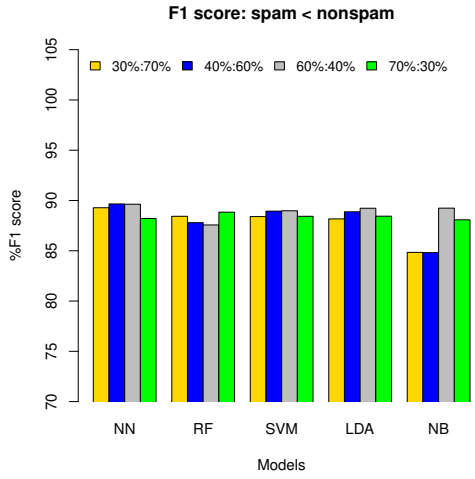

Fig. 4.2.2.5: F1 score(spam<nonspam) based on split ratio.

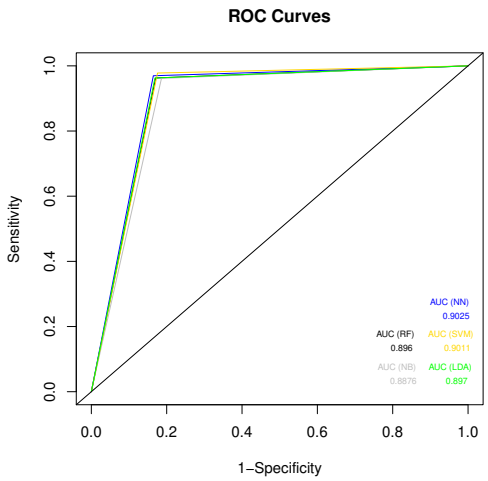

Fig. 4.2.2.6: ROC Curve - split ratio 30%:70%.

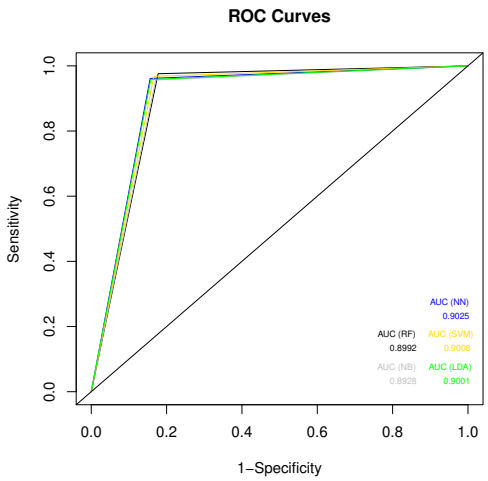

Fig. 4.2.2.7: ROC Curve - split ratio 40%:60%.

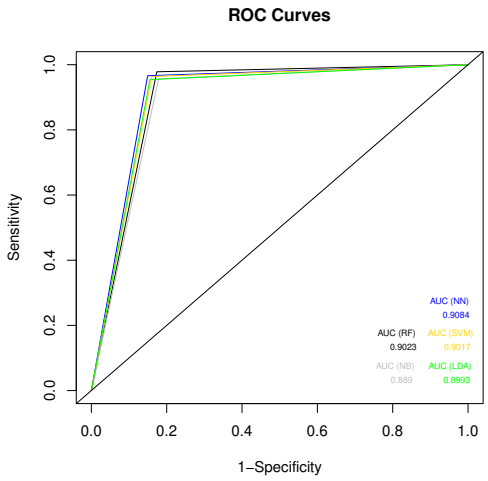

Fig. 4.2.2.8: ROC Curve - split ratio 60%:30%.

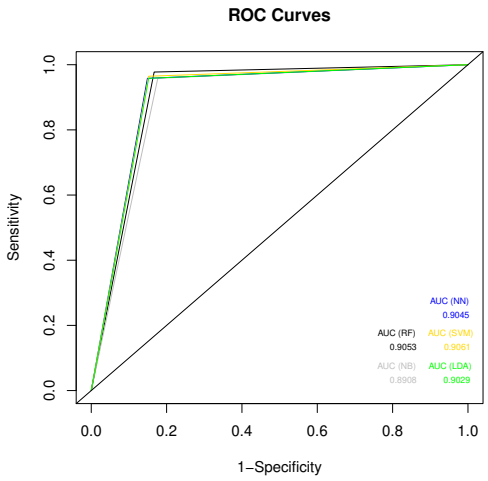

Fig. 4.2.2.9: ROC Curve - split ratio 70%:30%.

From Table 4.2.2 and Figures 4.2.2.1, 4.2.2.3, 4.2.2.4 and 4.2.2.5, Neural Networks consistently performed the best in terms of accuracy, recall, hitrate and F1 score across dataset split ratios except when the split ratio is 70%:30%, Random Forest performed the best in terms of recall and F1 score. Random Forest performed the best in terms of precision across dataset split ratios except when the split ratio is 70%:30%, Support Vector Machines performed the best in terms of precision as shown in Table 4.2.2 and Figure 4.2.2.2 above.

4.2.3 The study looks at the performance of the various techniques across the various training and test data split ratios for the case of spam=nonspam data[1035 vs 1035]

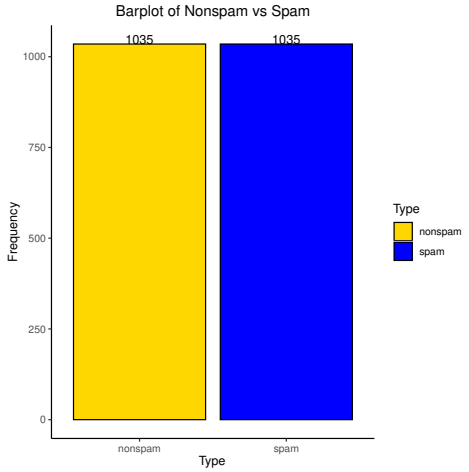

Fig. 4.2.3: Spam data equal to nonspam data.

Table 4.2.3: Accuracy, precision, recall and hit rate classification results based on various training and test data splits for case: spam =nonspam.

Measure	Ratio	NN	RF	SVM	LDA	NB
Accuracy		**90.41%**	88.69%	89.72%	90.28%	89.38%
Precision		97.72%	**98.61%**	98.16%	97.87%	98.47%
Recall	30%:70%	**82.76%**	78.48%	80.97%	82.34%	80.00%
Hitrate		**85.05%**	82.13%	83.80%	84.76%	83.16%
F1 score		**89.62%**	87.40%	88.74%	89.44%	88.28%
Accuracy		**91.38%**	89.86%	90.34%	90.50%	89.77%
Precision		97.77%	**98.43%**	97.18%	97.00%	98.24%
Recall	40%:60%	**84.70%**	81.00%	83.09%	83.57%	81.00%
Hitrate		**86.51%**	83.86%	85.23%	85.57%	83.84%
F1 score		**90.77%**	88.87%	89.58%	89.79%	88.79%
Accuracy		**91.79%**	91.55%	91.30%	91.30%	89.49%
Precision		97.53%	**98.86%**	96.98%	97.24%	97.95%
Recall	60%:40%	**85.75%**	84.06%	85.27%	85.02%	80.68%
Hitrate		**87.28%**	86.13%	86.85%	86.70%	83.57%
F1 score		**91.26%**	90.86%	90.75%	90.72%	88.48%
Accuracy		92.28%	**92.44%**	92.28%	91.80%	90.35%
Precision		97.82%	**98.53%**	97.13%	97.45%	98.08%
Recall	70%:30%	86.50%	86.17%	**87.14%**	85.85%	82.32%
Hitrate		87.90%	87.71%	**88.34%**	87.36%	84.76%
F1 score		91.81%	**91.94%**	91.86%	91.28%	89.51%

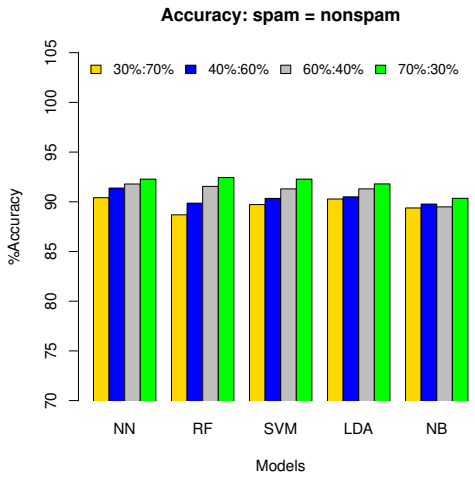

Fig. 4.2.3.1: Accuracy(spam=nonspam) based on split ratio.

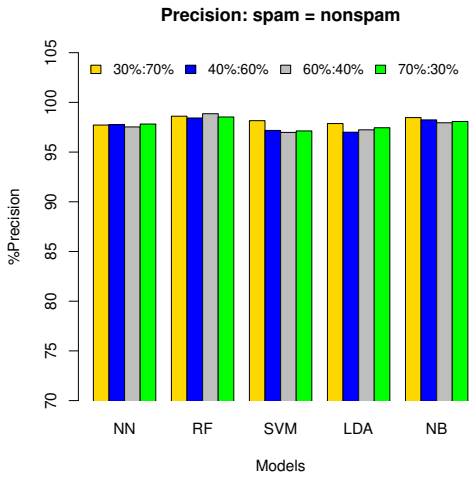

Fig. 4.2.3.2: Precision(spam=nonspam) based on split ratio.

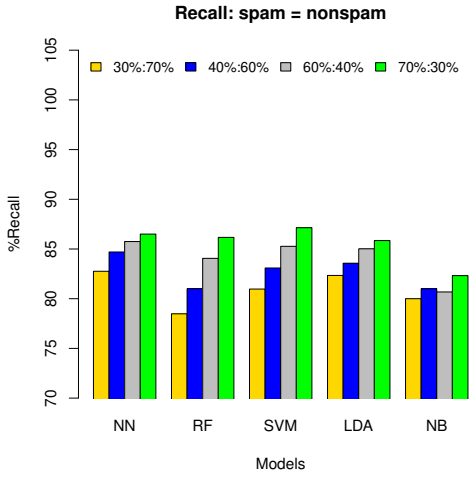

Fig. 4.2.3.3: Recall(spam=nonspam) based on split ratio.

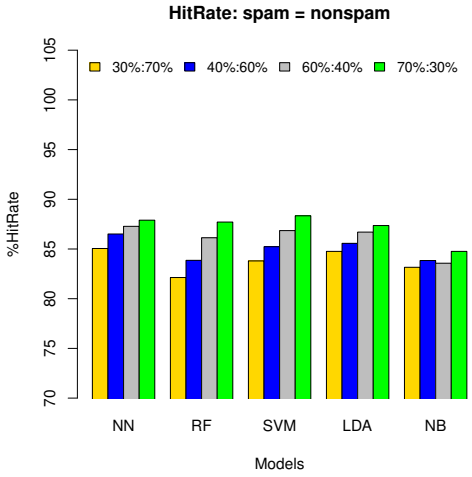

Fig. 4.2.3.4: Hitrate(spam=nonspam) based on split ratio.

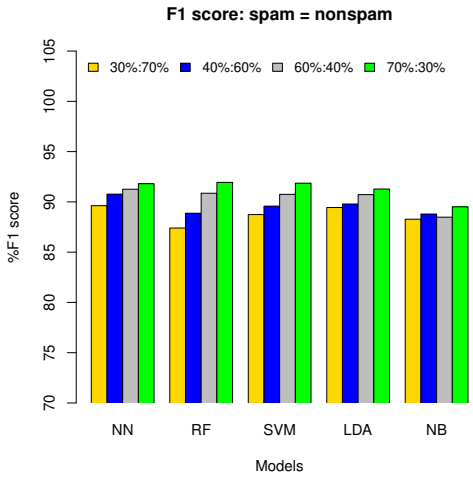

Fig. 4.2.3.5: F1 score(spam<nonspam) based on split ratio.

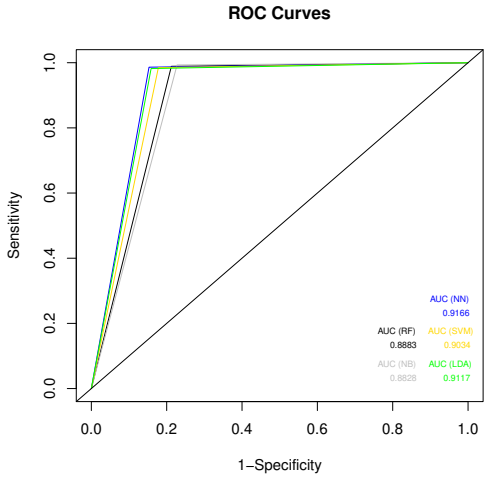

Fig. 4.2.3.6: ROC Curve - split ratio 30%:70%.

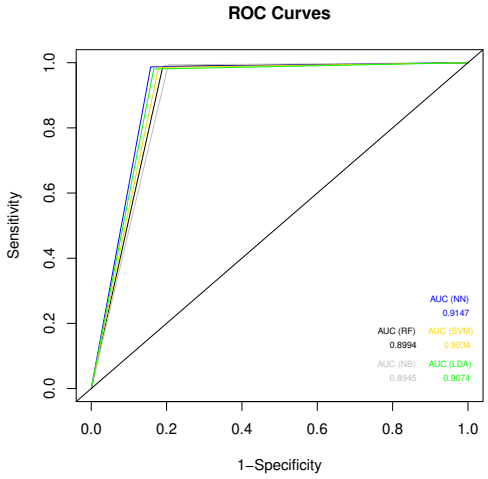

Fig. 4.2.3.7: ROC Curve - split ratio 40%:60%.

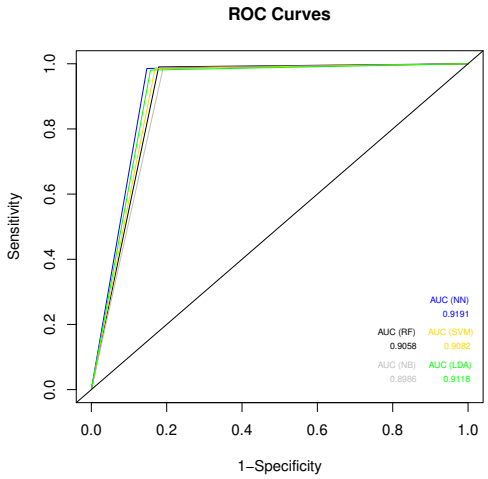

Fig. 4.2.3.8: ROC Curve - split ratio 60%:30%.

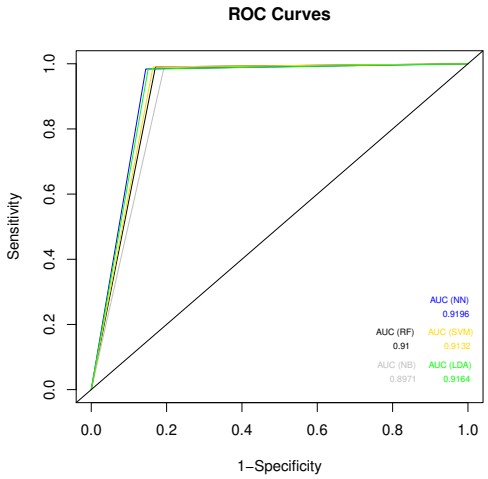

Fig. 4.2.3.9: ROC Curve - split ratio 70%:30%.

Neural Networks performed the best in terms of accuracy, recall hitrate and F1 score across all split ratios except when the split ratio is 70%:30%, Random Forest performed the best in terms of accuracy and F1 score and Support Vector Machines performed the best in terms of recall and hitrate as shown in Table 4.2.3 and Figures 4.2.3.1, 4.2.3.3, 4.2.3.4 and 4.2.3.5 above. Random Forest performed the best in terms of precision across dataset split ratios as shown in Table 4.2.3 and Figure 4.2.3.2 above.

4.3 Experiment 3: Different Feature Sizes

4.3.1 The study looks at the performance of the methods across various feature sizes for the case of spam>nonspam data[1035 vs 500].

Table 4.3.1: Accuracy, precision, recall and hit rate classification results based on various feature sizes for case: spam>nonspam.

Measure	Feature	NN	RF	SVM	LDA	NB
Accuracy		**88.93%**	**88.93%**	**88.93%**	**88.93%**	88.83%
Precision		**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
Recall	45	**66.00%**	**66.00%**	**66.00%**	**66.00%**	**66.00%**
Hitrate		**85.89%**	**85.89%**	**85.89%**	**85.89%**	**85.89%**
F1 score		**79.52%**	**79.52%**	**79.52%**	**79.52%**	**79.52%**
Accuracy		**89.90%**	**89.90%**	**89.90%**	**89.90%**	89.25%
Precision		**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
Recall	60	**69.00%**	**69.00%**	**69.00%**	**69.00%**	67.00%
Hitrate		86.90%	86.25%	**86.97%**	**86.97%**	86.25%
F1 score		**81.66%**	**81.66%**	**81.66%**	**81.66%**	80.24%
Accuracy		**91.53%**	**91.53%**	**91.53%**	**91.53%**	91.21%
Precision		97.44%	97.44%	97.44%	97.44%	**100.00%**
Recall	80	**76.00%**	**76.00%**	**76.00%**	**76.00%**	73.00%
Hitrate		**89.52%**	**89.52%**	**89.52%**	**89.52%**	88.46%
F1 score		**85.39%**	**85.39%**	**85.39%**	**85.39%**	84.39%
Accuracy		91.53%	91.53%	91.53%	**91.86%**	89.58%
Precision		97.44%	97.44%	97.44%	97.44%	**100.00%**
Recall	100	76.00%	76.00%	76.00%	**77.00%**	68.00%
Hitrate		89.52%	89.52%	89.52%	**89.91%**	86.61%
F1 score		85.39%	85.39%	85.39%	**86.02%**	80.95%
Accuracy		**93.81%**	93.49%	93.49%	93.49%	89.90%
Precision		97.65%	97.62%	97.62%	97.62%	**100.00%**
Recall	120	**83.00%**	82.00%	82.00%	82.00%	69.00%
Hitrate		**92.34%**	91.93%	91.93%	91.93%	86.97%
F1 score		**89.73%**	89.13%	89.13%	89.13%	81.66%
Accuracy		93.49%	**93.81%**	**93.81%**	**93.81%**	91.53%
Precision		97.65%	97.65%	97.65%	97.65%	**98.68%**
Recall	140	**83.00%**	**83.00%**	**83.00%**	**83.00%**	75.00%
Hitrate		92.31%	**92.34%**	**92.34%**	**92.34%**	89.18%
F1 score		**89.73%**	**89.73%**	**89.73%**	**89.73%**	85.23%

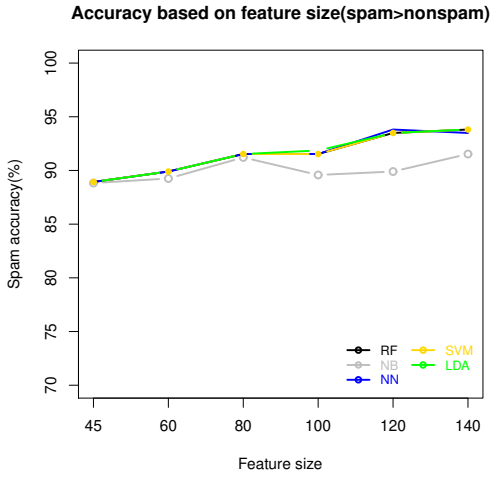

Fig. 4.3.1.1: Accuracy(spam>nonspam) based on feature size.

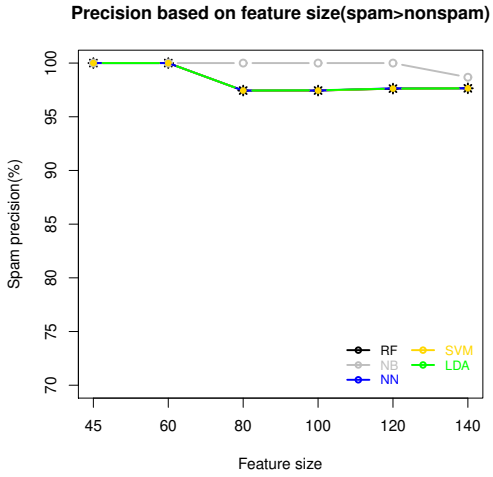

Fig. 4.3.1.2: Precision(spam>nonspam) based on feature size.

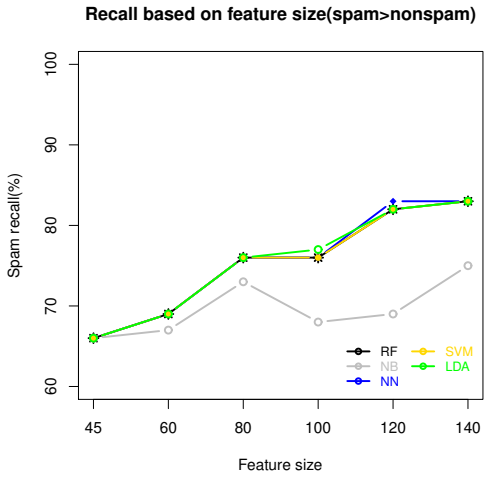

Fig. 4.3.1.3: Recall(spam>nonspam) based on feature size.

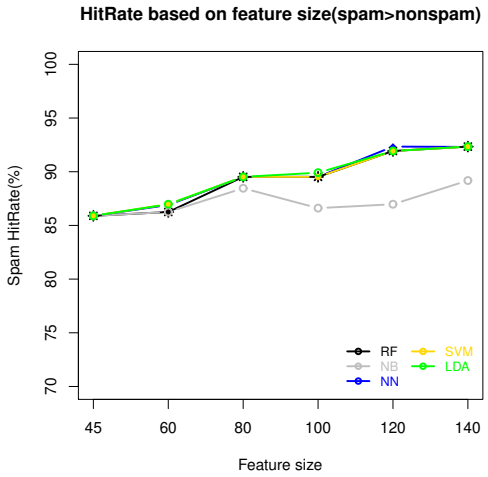

Fig. 4.3.1.4: Hitrate(spam>nonspam) based on feature size.

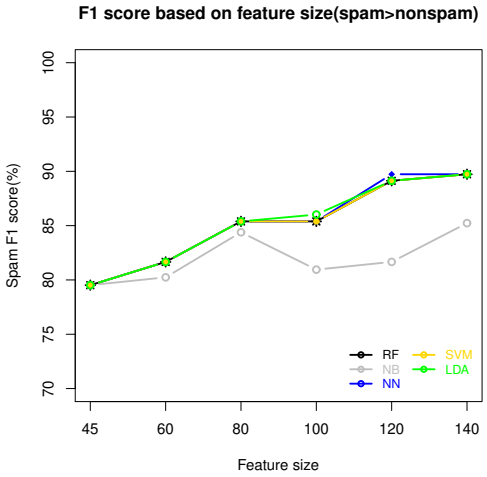

Fig. 4.3.1.5: F1 score(spam>nonspam) based on feature size.

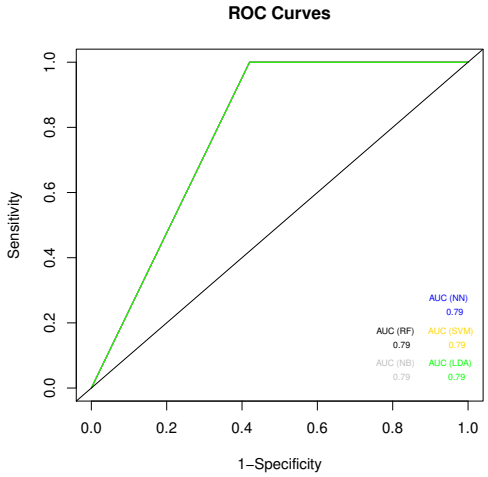

Fig. 4.3.1.6: ROC Curve - feature size 45.

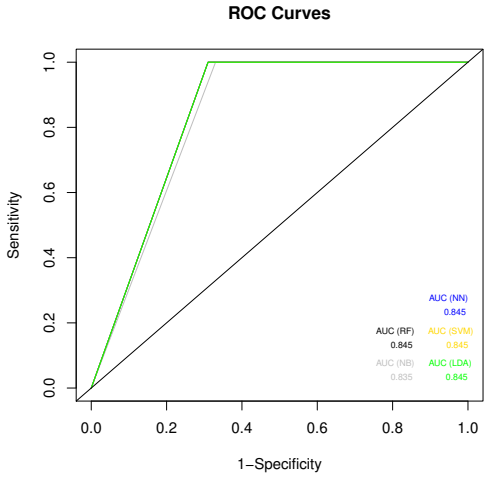

Fig. 4.3.1.7: ROC Curve - feature size 60.

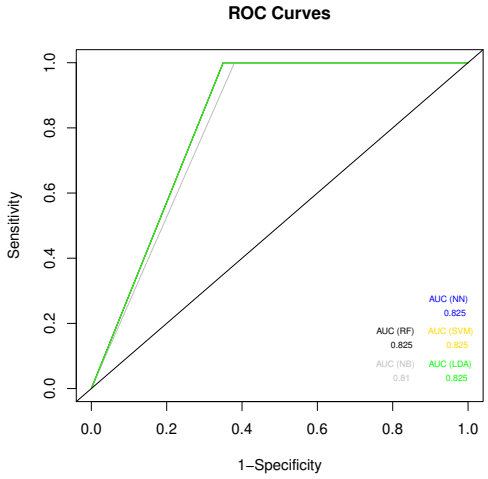

Fig. 4.3.1.8: ROC Curve - feature size 80.

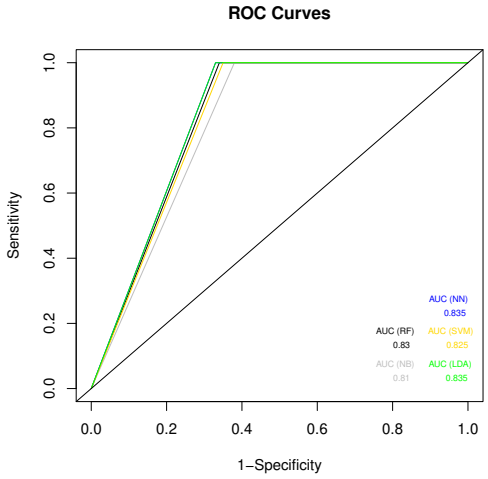

Fig. 4.3.1.9: ROC Curve - feature size 100.

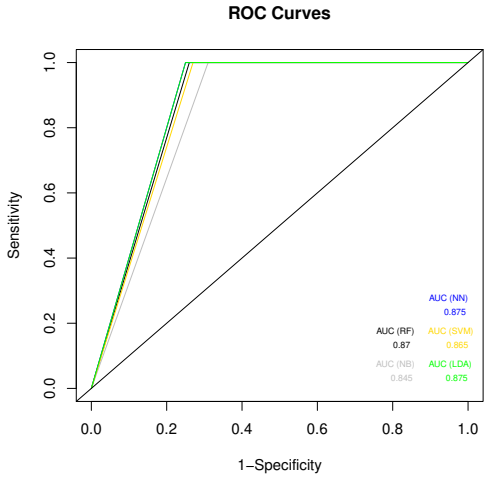

Fig. 4.3.1.10: ROC Curve - feature size 120.

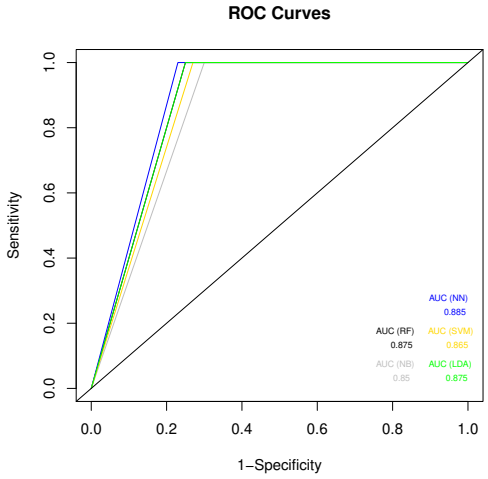

Fig. 4.3.1.11: ROC Curve - feature size 140.

As shown in Table 4.3.1 and Figure 4.3.1.1, spam mail features have an impact on the spam mail classification. As the feature size increases, the spam accuracy starts to increase gradually for all the methods. In terms of accuracy, precision, recall and F1 score, Neural Networks, Random Forest, Support Vector Machines and Linear Discriminant Analysis achieved similar results when the feature size is 45 and 60 as shown in Table 4.3.1 and Figures 4.3.1.1, 4.3.1.2, 4.3.1.3 and 4.3.1.5 above. When the feature size is 80, Neural Networks, Random Forest, Support Vector Machines and Linear Discriminant Analysis achieved similar results in terms of accuracy, recall, hitrate and F1 score as shown in Table 4.3.1 and Figures 4.3.1.1, 4.3.1.3, 4.3.1.4 and 4.3.1.5 above. Naïve Bayes performed the best in terms of precision for feature sizes 80, 100, 120 and 140 as shown in Table 4.3.1 and Figure 4.3.1.2 above. Linear Discriminant Analysis performed the best in terms of accuracy, recall, hitrate

and F1 score when the feature size is 100 as shown in Table 4.3.1 and Figures 4.3.1.1, 4.3.1.3, 4.3.1.4 and 4.3.1.5 above. Neural Networks performed the best in terms of accuracy, recall, hitrate and F1 score when the feature size is 120 as shown in Table 4.3.1 and Figures 4.3.1.1, 4.3.1.3, 4.3.1.4 and 4.3.1.5 above. From Figure 4.3.1.6, the ROC curve shows that all methods performed similarly for feature size 45.

4.3.2 The study looks at the performance of the methods across various feature sizes for the case of spam<nonspam data[1035 vs 2057].

Table 4.3.2: Accuracy, precision, recall and hit rate classification results based on various feature sizes for case: spam<nonspam.

Measure	Feature	NN	RF	SVM	LDA	NB
Accuracy		74.76%	74.92%	74.76%	74.60%	**75.89%**
Precision		73.39%	72.94%	73.39%	72.84%	**99.62%**
Recall	45	97.32%	**99.03%**	97.32%	98.54%	63.99%
Hitrate		84.93%	**93.33%**	84.93%	90.21%	58.19%
F1 score		83.68%	**84.01%**	83.68%	83.76%	77.93%
Accuracy		**80.10%**	77.83%	77.83%	77.50%	78.48%
Precision		70.32%	75.37%	75.37%	75.28%	**99.64%**
Recall	60	**99.66%**	99.03%	99.03%	98.54%	67.88%
Hitrate		**99.52%**	94.87%	94.87%	92.50%	60.95%
F1 score		82.46%	**85.60%**	**85.60%**	85.35%	80.75%
Accuracy		74.43%	79.13%	79.13%	78.80%	**81.88%**
Precision		72.55%	76.50%	76.50%	76.42%	**99.34%**
Recall	80	**99.03%**	**99.03%**	**99.03%**	98.54%	73.24%
Hitrate		92.98%	**95.35%**	**95.35%**	93.18%	65.08%
F1 score		83.75%	**86.32%**	**86.32%**	86.08%	84.32%
Accuracy		**84.30%**	83.82%	83.50%	78.48%	83.01%
Precision		98.16%	98.14%	98.13%	76.43%	**99.35%**
Recall	100	77.86%	77.13%	76.64%	**97.81%**	74.94%
Hitrate		68.84%	68.14%	67.68%	**90.22%**	66.56%
F1 score		**86.84%**	86.38%	86.06%	85.81%	85.44%
Accuracy		**86.89%**	86.08%	86.41%	**86.89%**	85.76%
Precision		97.97%	**99.39%**	97.95%	98.83%	98.79%
Recall	120	**82.00%**	79.5%	81.27%	**82.00%**	79.5%
Hitrate		**72.99%**	70.93%	72.20%	**72.99%**	70.73%
F1 score		89.28%	88.34%	88.83%	**89.63%**	88.10%
Accuracy		**89.32%**	88.19%	88.03%	89.00%	87.70%
Precision		98.59%	**99.13%**	**99.13%**	98.58%	98.55%
Recall	140	**85.16%**	82.97%	83.21%	84.67%	82.73%
Hitrate		**76.81%**	74.45%	74.54%	76.23%	73.99%
F1 score		**91.38%**	90.33%	90.48%	91.10%	89.95%

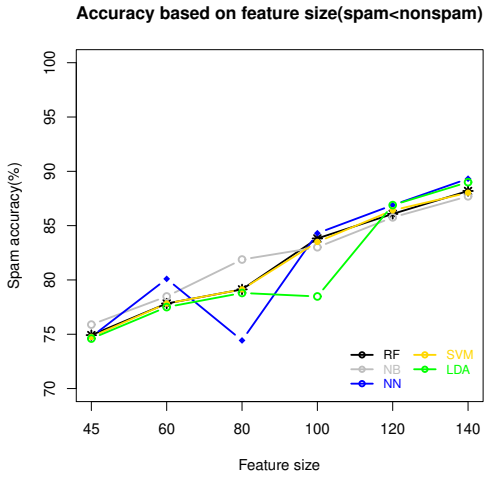

Fig. 4.3.2.1: Accuracy(spam<nonspam) based on feature size.

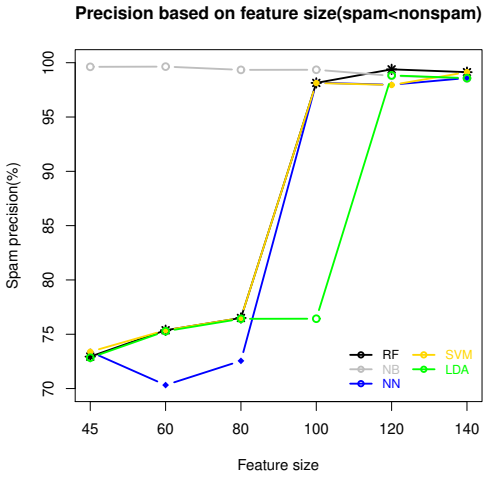

Fig. 4.3.2.2: Precision(spam<nonspam) based on feature size.

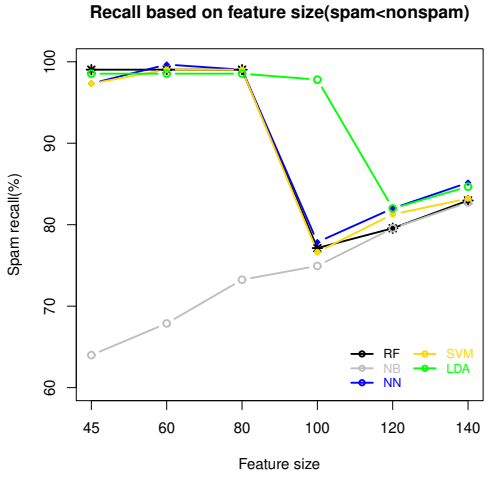

Fig. 4.3.2.3: Recall(spam<nonspam) based on feature size.

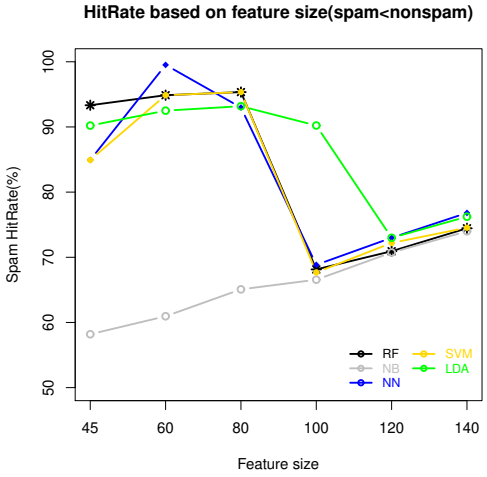

Fig. 4.3.2.4: Hitrate(spam<nonspma) based on feature size.

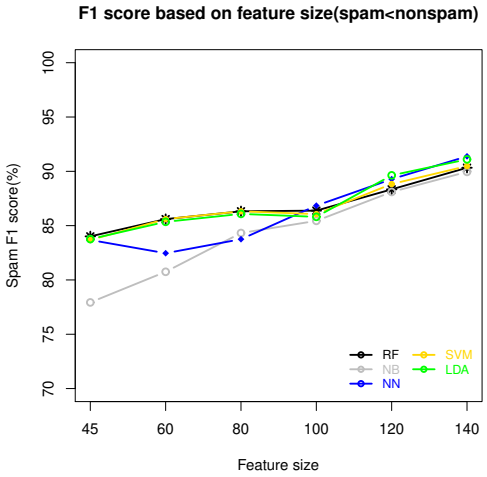

Fig. 4.3.2.5: F1 score (spam<nonspam) based on feature size.

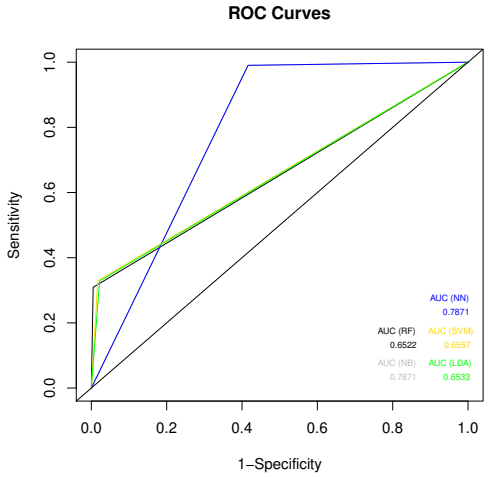

Fig. 4.3.2.6: ROC Curve - feature size 45.

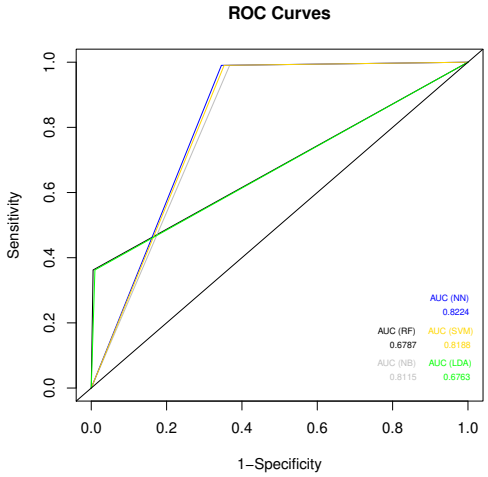

Fig. 4.3.2.7: ROC Curve - feature size 60.

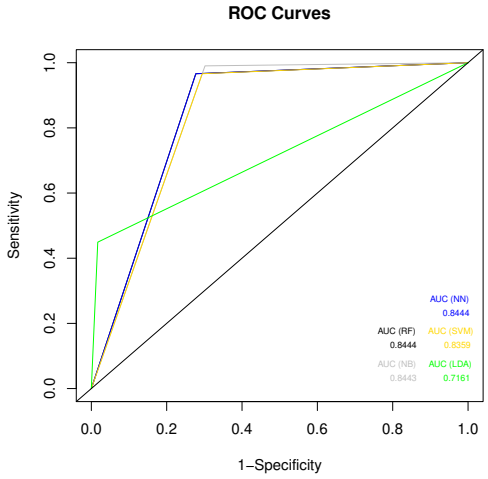

Fig. 4.3.2.8: ROC Curve - feature size 80.

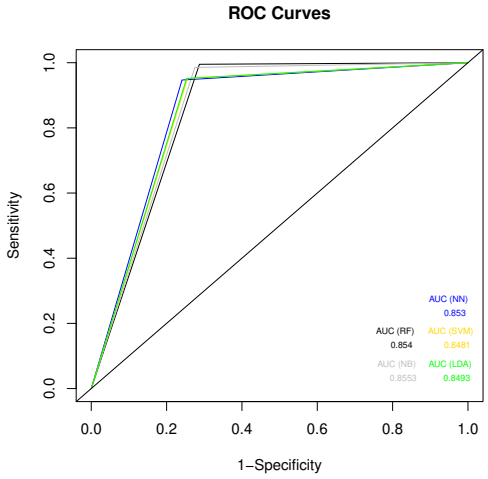

Fig. 4.3.2.9: ROC Curve - feature size 100.

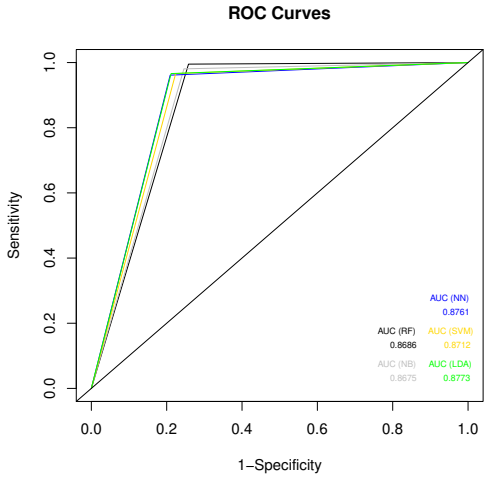

Fig. 4.3.2.10: ROC Curve - feature size 120.

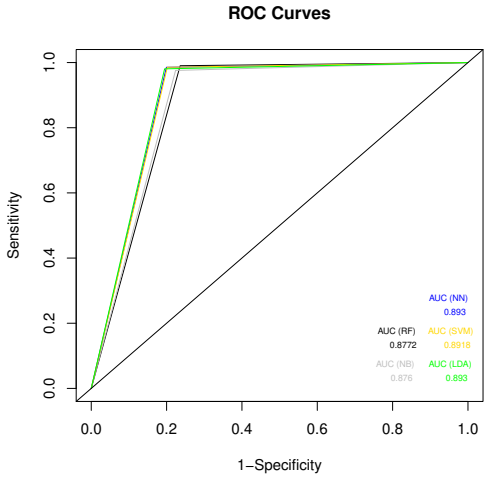

Fig. 4.3.2.11: ROC Curve - feature size 140.

Neural Networks performed the best in terms of accuracy, recall and hitrate for feature sizes 60, 120 and 140 as shown in Table 4.3.2 and Figures 4.3.2.1, 4.3.2.3 and 4.3.2.4 above. Naïve Bayes performed the best in terms of precision for feature sizes 45, 60, 80 and 100 as shown in Table 4.3.2 and Figure 4.3.2.3 above. Naïve Bayes performed the best in terms of accuracy for feature sizes 45 and 80. Linear Discriminant Analyses performed the best in terms of recall and hitrate for feature size 100, and performed the same as Neural Networks for feature 120. Random Forest performed the best in terms of recall, hitrate and F1 score for feature size 45, and performed the same as Support Vector Machines for feature size 80 as shown in Figures 4.3.2.3, 4.3.2.4 and 4.3.2.5 above.

4.3.3 The study looks at the performance of the methods across various feature sizes for the case of spam=nonspam data[1035 vs 1035]

Table 4.3.3: Accuracy, precision, recall and hit rate classification results based on various feature sizes for case: spam=nonspam.

Measure	Feature	NN	RF	SVM	LDA	NB
Accuracy		**80.43%**	78.74%	79.47%	79.71%	79.71%
Precision		97.73%	99.17%	99.19%	**99.20%**	96.83%
Recall	45	62.32%	57.97%	59.42%	59.90%	**70.49%**
Hitrate		**72.34%**	70.31%	71.03%	71.28%	70.49%
F1 score		76.11%	73.17%	74.32%	74.70%	**81.59%**
Accuracy		**81.40%**	80.19%	80.68%	81.16%	81.16%
Precision		97.79%	99.21%	99.22%	**99.24%**	96.97%
Recall	60	**64.25%**	60.87%	61.84%	62.80%	61.84%
Hitrate		**73.38%**	71.78%	72.28%	72.79%	71.99%
F1 score		**77.55%**	75.45%	76.19%	76.92%	75.52%
Accuracy		**84.30%**	82.85%	84.06%	84.06%	84.06%
Precision		**100.00%**	**100.00%**	99.30%	99.30%	97.18%
Recall	80	**68.60%**	65.70%	**68.60%**	**68.60%**	66.67%
Hitrate		**76.10%**	74.46%	76.01%	76.01%	74.63%
F1 score		**81.38%**	79.30%	81.14%	81.14%	79.08%
Accuracy		**88.16%**	86.96%	87.68%	87.44%	87.44%
Precision		**100.00%**	**100.00%**	99.37%	**100.00%**	97.32%
Recall	100	**76.33%**	73.91%	75.85%	74.88%	70.05%
Hitrate		**80.86%**	79.31%	80.47%	79.92%	76.60%
F1 score		**86.58%**	85.00%	86.03%	85.64%	81.46%
Accuracy		**90.34%**	88.89%	89.37%	88.89%	88.89%
Precision		**100.00%**	**100.00%**	**100.00%**	**100.00%**	97.52%
Recall	120	**80.68%**	77.78%	78.74%	77.78%	75.85%
Hitrate		**83.81%**	81.82%	82.47%	81.82%	80.24%
F1 score		**89.31%**	87.50%	88.11%	87.50%	85.33%
Accuracy		**92.03%**	90.82%	91.30%	**92.03%**	**92.03%**
Precision		**100.00%**	99.42%	**100.00%**	**100.00%**	97.65%
Recall	140	**84.06%**	82.13%	82.61%	**84.06%**	80.19%
Hitrate		**86.25%**	84.77%	85.19%	**86.25%**	83.20%
F1 score		**91.34%**	89.95%	90.48%	**91.34%**	88.06%

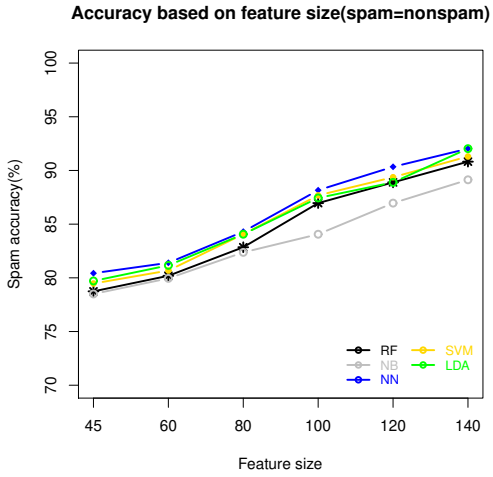

Fig. 4.3.3.1: Accuracy(spam=nonspam) based on feature size.

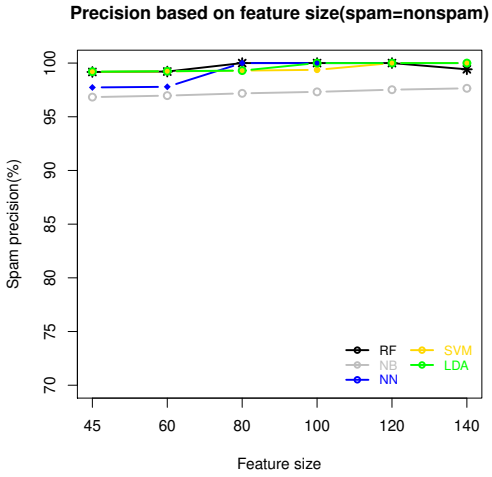

Fig. 4.3.3.2: Precision(spam=nonspam) based on feature size.

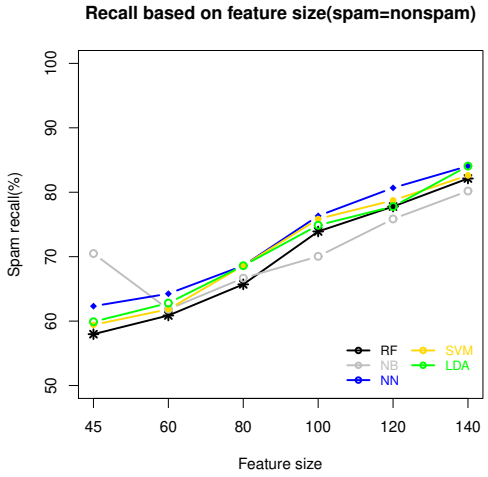

Fig. 4.3.3.3: Recall(spam=nonspam) based on feature size.

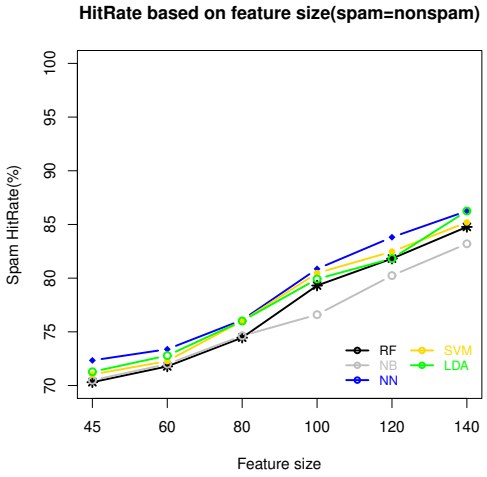

Fig. 4.3.3.4: Hitrate(spam=nonspam) based on feature size.

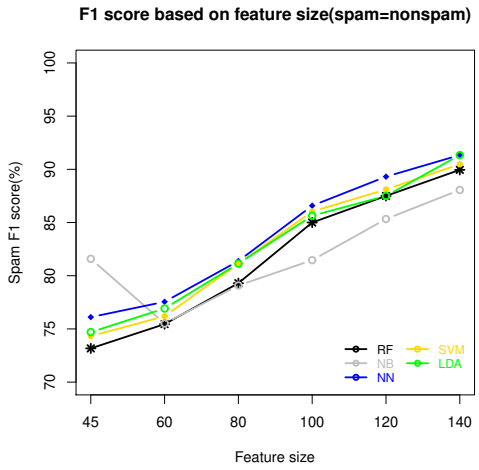

Fig. 4.3.3.5: F1 score (spam=nonspam) based on feature size.

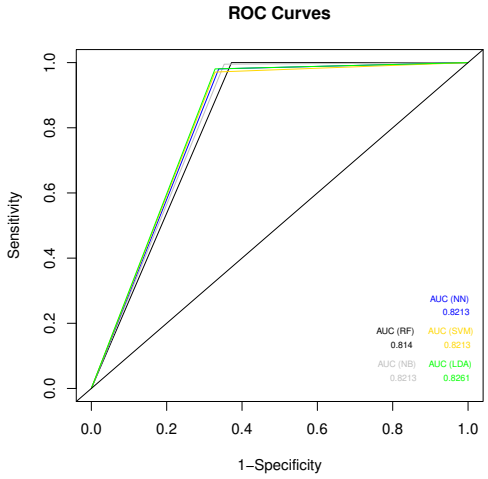

Fig. 4.3.3.6: ROC Curve - feature size 45.

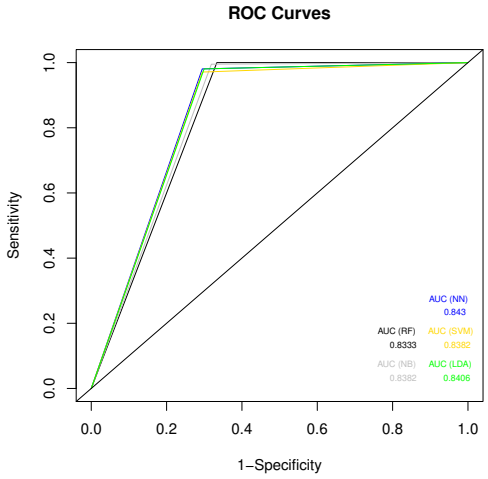

Fig. 4.3.3.7: ROC Curve - feature size 60.

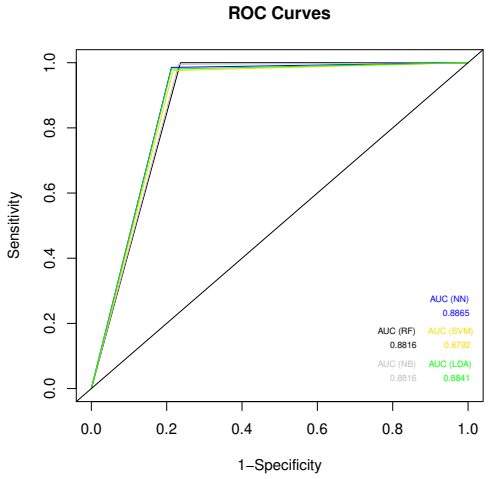

Fig. 4.3.3.8: ROC Curve - feature size 80.

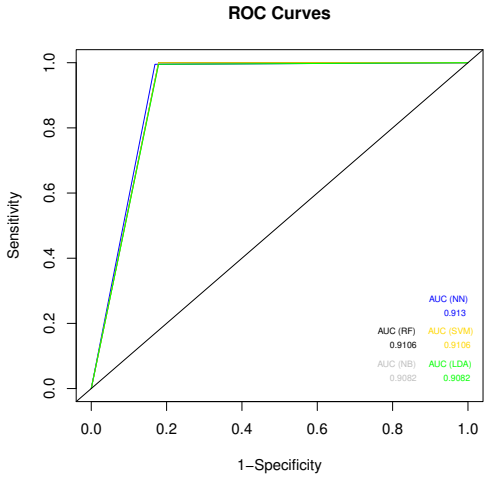

Fig. 4.3.3.9: ROC Curve - feature size 100.

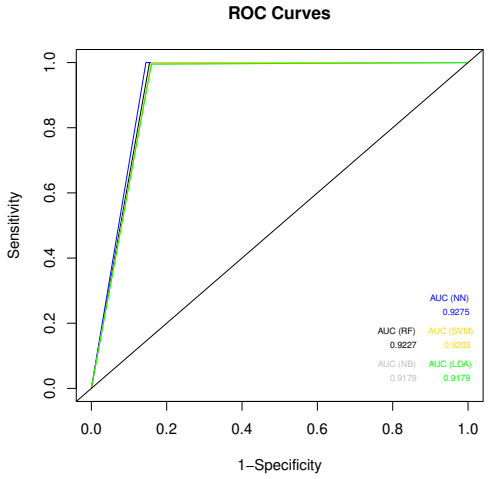

Fig. 4.3.3.10: ROC Curve - feature size 120.

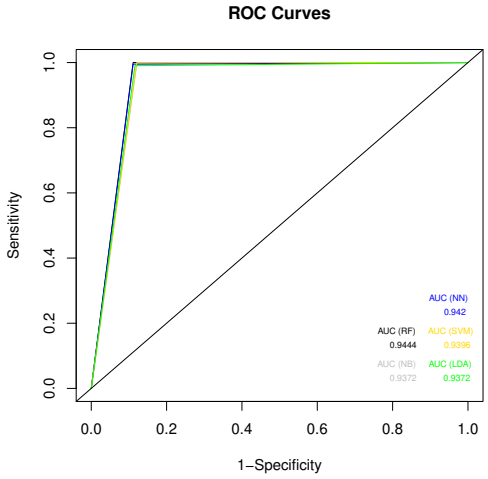

Fig. 4.3.3.11: ROC Curve - feature size 140.

Similar observations can be made in Table 4.3.3 and Figures 4.3.3.1, 4.3.3.2, 4.3.3.3, 4.3.3.4 and 4.3.3.5, Neural Networks performed better than all other methods across all feature sizes except when the feature size is 45, and 60, Linear Discriminant Analysis performed the best in terms of precision. Naïve Bayes performed the best in terms of recall and F1 score for feature size 45. On average, as the feature size increases, it appears that the performance in terms of accuracy, precision, recall, hitrate and F1 score increases.

Chapter 5

Conclusion

In this paper, four data mining classifiers including Neural Networks, Random Forest, Support Vector Machines and Naïve Bayes as well as the classical statistics technique Linear Discriminant Analysis were proposed for classifying spam and nonspam mail. Three experiments were carried out to assess the performance of these techniques by changing the sample size, changing training set size and extracted feature size. In this section the following summary tables that were obtained from the analyses section are presented.

Table 5.1: Different Sample Sizes summary results.

Experiment 1: Different Sample Sizes						
Method	500	1000	1500	2000	2500	3000
Accuracy	**NN**	**NN**	**NN**	**NN**	**NN**	SVM
Precision	NN	NN	NN,RF, SVM,NB LDA	RF	NN,LDA	RF
Recall	NN	NN	NN	NN	NN	SVM
Hitrate	NN	NN	NN	NN	NN	SVM
F1 score	NN	NN	NN	NN	NN	SVM

In experiment 1 (different sample sizes), Neural Networks performed the best in terms of accuracy, recall, hitrate and F1 score for sample sizes 500, 1000, 1500, 2000 and 2500. Random Forest performed the best in terms of precision for sample sizes 2000 and 3000. In terms of precision for sample size 1500, all methods performed the same. In terms of accuracy, recall, hitrate and F1 score for sample size 3000, Support Vector Machines performed the best. Linear Discriminant Analysis performed similarly to Neural Networks in terms of precision for sample size 2500 as shown in Table 5.1 above.

In experiment 2 (training vs test dataset split ratio), three conditions, thus, spam>nonspam, spam<nonspam and spam=nonspam were explored.

Table 5.2: Training vs Test dataset split ratio summary results.

Experiment 2: Training vs Test dataset split ratio					
Condition	Measure	30%:70%	40%:60%	60%:40%	70%:30%
spam>nonspam	Accuracy	**NN**	**NN**	**NN**,RF	NN,NB
	Precision	NB	NB	NB	NN
	Recall	NN	NN	NN,RF	NN
	Hitrate	NN	NN	NN,RF	NN
	F1 score	NN	NN	NN,RF	NN
spam<nonspam	Accuracy	**NN**	**NN**	**NN**	**NN**
	Precision	RF	RF	RF	SVM
	Recall	NN	NN	NN	RF
	Hitrate	NN	NN	NN	NN
	F1 score	NN	NN	NN	RF
spam=nonspam	Accuracy	**NN**	**NN**	**NN**	RF
	Precision	RF	RF	RF	RF
	Recall	NN	NN	NN	SVM
	Hitrate	NN	NN	NN	SVM
	F1 score	NN	NN	NN	RF

Neural Networks performed the best in terms of accuracy, recall, hitrate and F1 score under all conditions and for split ratios 30%:70%, 40%:60% and 60%:40%. Naïve Bayes performed the best in terms of precision for all split ratios for the spam>nonspam condition. Random Forest peformed the best in terms of precision for all split ratios for spam=nonspam and spam<nonspam except for split ratio 70%:30% where Support Vector Machines peformed the best. Random Forest performed the best in terms of accuracy for split ratio 70%:30% under the spam=nonspam condition. Naïve Baye performed the same as Neural Networks in terms of accuracy for split ratio 70%:30% under the spam>nonspam condition as shown in Table 5.2 above.

In experiment 3 (different feature sizes), three conditions, thus, spam>nonspam, spam<nonspam and spam=nonspam were looked at across different feature sizes chosen. It was noted that the feature size has an effect on the performance of the techniques in classifying spam and nonspam mail. Most methods performed the same for feature sizes 45, 60, 80 and 140 for the spam>nonspam condition. Naïve Bayes performed the best in terms of precision for feature sizes 45, 60, 80 and 100 for spam<nonspam. Neural Networks performed the best in terms of accuracy, hitrate and F1 score for feature sizes 45, 60, 80, 100 and 120 for spam=nonspam condition as show in Table 5.3 below.

Table 5.3: Different Feature Sizes summary results.

Experiment 3: Different Feature Sizes							
Condition	Measure	45	60	80	100	120	140
spam>nonspam	Accuracy	**NN**,RF, SVM, LDA	**NN**,RF, SVM, LDA	**NN**,RF, SVM, LDA	LDA	**NN**	RF, SVM, LDA
	Precision	NN,RF, SVM, LDA, NB	NN,RF, SVM, LDA, NB	NB	NB	NB	NB
	Recall	NN,RF, SVM, LDA, NB	NN,RF, SVM, LDA	NN,RF, SVM, LDA	LDA	NN	NN,RF, SVM, LDA
	Hitrate	NN,RF, SVM, LDA, NB	SVM, LDA	NN,RF, SVM, LDA	LDA	NN	RF, SVM, LDA
	F1 score	NN,RF, SVM, LDA, NB	NN,RF, SVM, LDA	NN,RF, SVM, LDA	LDA	NN	NN,RF, SVM, LDA
spam<nonspam	Accuracy	NB	**NN**	NB	**NN**	**NN**, LDA	**NN**
	Precision	NB	NB	NB	NB	RF	RF, SVM
	Recall	RF	NN	NN,RF, SVM	LDA	NN, LDA	NN
	Hitrate	RF	NN	RF, SVM	LDA	NN, LDA	NN
	F1 score	RF	RF,SVM	RF, SVM	NN	LDA	NN
spam=nonspam	Accuracy	**NN**	**NN**	**NN**	**NN**	**NN**	**NN**,NB, LD
	Precision	LDA	LDA	NN,RF	NN,RF, LDA	NN,RF, SVM, LDA	NN, SVM, LDA
	Recall	NB	NN	NN, SVM, LDA	NN	NN	NN, LDA
	Hitrate	NN	NN	NN	NN	NN	NN, LDA
	F1 score	NB	NN	NN	NN	NN	NN, LDA

This study illustrates that there is no single method that is considered to be

the best or optimal in classifying spam and nonspam email, however NN appears to perform well across the different experiments in terms of accuracy followed by some other machine learning techniques in comparison to LDA. It is recommended that more than one method to be used for spam mail classification.

For future work, one could use the email content-based approach which considers the full content of the email rather than the email header-based approach that was used in this study to investigate if this has an impact on the performance of the classifiers.

References

[1] Al-Shboul, A.B. Hakh, H. Faris, H. Aljarah, I. and Alsawalqah, H. Voting-based Classification for E-mail Spam Detection. *Journal of ICT Research and Applications*, vol. no 10.1, 2016, pp. 29-42.

[2] Androutsopoulos, I. Paliouras, G. Karkaletsis, V. Sakkis, G. Spyropoulos, C.D. and Stamatopoulos, P. Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach, *Institute of Informatics and Telecommunications*, 2000, pp. 1-12.

[3] Awad, W.A. and Elseuofi, S.M. Machine Learning Methods for Spam Email Classification, *International Journal of Computer Science and Information Technology*, vol. no 3.1, Feb. 2011, pp. 173-183.

[4] Basavaraju, M. and Prabhakar, R. A Novel Method of Spam Mail Detection using Text Based Clustering Approach, *International Journal of Computer Applications*, vol. no 5.4, Aug. 2010, pp. 15-24.

[5] Bibi, A. Latif, R. Khalid, S. Ahmed, W. Shabir, R.A. and Shahryar, T. Spam Mail Scanning Using Machine Learning Algorithm. JCP, vol. no 15.2, Jan. 2020, pp. 73-84.

[6] Cook, D. Hartnett, J. Manderson, K. and Scanlan, J. Catching spam before it arrives: Domain specific blacklists, *In Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, vol. no 54, Jan. 2006, pp. 193-202.

[7] Halees, A.E. Filtering Spam E-Mail from Mixed Arabic and English Messages: A Comparison of Machine Learning Techniques, *The International Arab Journal of Information Technology*, vol. no 6.1, Jan. 2009, pp. 52-56.

[8] Han, J. Kamber, M. and Pei, J. Data Mining: Concepts and Techniques 3^{rd} Edition, 2011, pp. 11-319.

[9] Islam, M.R. Chowdhury, M.U. and Zhou, W. An innovative spam filtering model based on support vector machine, *International Conference on Computational Intelligence for Modeling, Control and Automation and International Conference on Intelligence Agents, Web Technologies and Internet Commerce*, vol. no 2, Nov. 2005, pp. 348-353.

[10] Jukic, S. Azemovic, J. Keco, D. and Kevric, J. Comparison of machine learning techniques in spam e-mail classification, *Southeast Europe Journal of Soft Computing*, vol. no 4.1, Mar. 2015, pp. 32-36.

[11] Kelin, S.A. State Regulation of Unsolicited Commercial E-Mail, 2001, pp. 435

[12] Keogh, E. Naive Bayes Classifier, Nov. 2006.

[13] Kiwanuka, F.N. Alqatawna, J. Amin, A.H.M. Paul, S. and Faris, H. Towards Automated Comprehensive Feature Engineering for Spam Detection, *In Proceedings of the 5th International Conference on Information Security and Privacy*, ICISSP, 2019, pp. 429-437.

[14] Kumar, K.R. Poonkuzhali, G. and Sudhakar, P. Comparative Study on Email Spam Classifier using Data Mining Techniques, *Proceedings of the International MultiConference of Engineers and Computer Scientits*, IMECS, Vol. no 1, Mar. 2012, pp. 14-16.

[15] Lakshmi, D.R. and Radha, N. Supervised Learning Approach for Spam Classification Analysis using Data Mining Tools, *International Journal on Computer Science and Engineering*, vol. no 2.9, 2010, pp. 2760-2766.

[16] Meira, Jr.W. and Zaki, M.J. Data Mining and Analysis: Fundamental Concepts and Algorithms. *Cambridge University Press*, 2014.

[17] Mohamed, A.E. Comparative Study of Four Supervised Machine Learning Techniques for Classification, *International Journal of Applied*, vol. no 7.2, Jun. 2017, pp. 5-18.

[18] Prajwala, T.R. A Comparative Study on Decision Tree and Random Forest using R tool, *International Journal of Advanced Research in Computer and Communication Engineering*, vol. no 4.1, Jan. 2015, pp. 196-199.

[19] Rahmad, F. Suryanto, Y. and Ramli, K. Performance Comparison of Anti-Spam Technology Using Confusion Matrix Classification, *In IOP Conference Series: Materials Science and Engineering*, vol. no 879.1, July. 2020.

[20] Rathi, M. and Pareek, V. Spam Mail Detection through Data Mining: A Comparative Performance Analysis, *Modern Education and Computer Science Press*, Dec. 2013, pp. 31-39.

[21] Raschka, S. Naïve bayes and text classification i-introduction and theory, *arXiv preprint arXiv:1410.5329*, Oct. 2014, pp. 16-17.

[22] Shams, R. and Mercer, R.E. Classifying Spam Emails using Text and Readability Features, *IEEE 13th international conference on data mining*, London, 2013, pp. 657-666.

[23] Tang, Y. Krasser, S. He, Y. Yang, W. and Dmitri, A. Support Vector Machines and Random Forests Modeling for Spam Senders Behavior Analysis, *IEEE Globecom 2008-2008 IEEE Global Telecommunications Conference*, Nov. 2008, pp. 1-5.

[24] Tretyakov, K. Machine Learning Techniques in Spam Filtering, *Institute of Computer Science*, Tartu May. 2004, pp. 60-79.

[25] Wang, R. Youssef, A.M. and Ahmed K. Elhakeem, A.K. On Some Feature Selection Strategies for Spam Filter Design, *Canadian Conference on Electrical and Computer Engineering*, May. 2006, pp. 2186-2189.

[26] Youn, S. and McLeod, D. *A Comparative Study for Email Classification*, Los Angeles: University of Southern California, 2007.

[27] Yu, B. and Xu, Z.B. *A comparative study for content-based dynamic spam classification using four machine learning algorithms*, China: Xi and Jiaotong University, 2007.

Appendix A

Extra Results

Fig. 6.1: word cloud

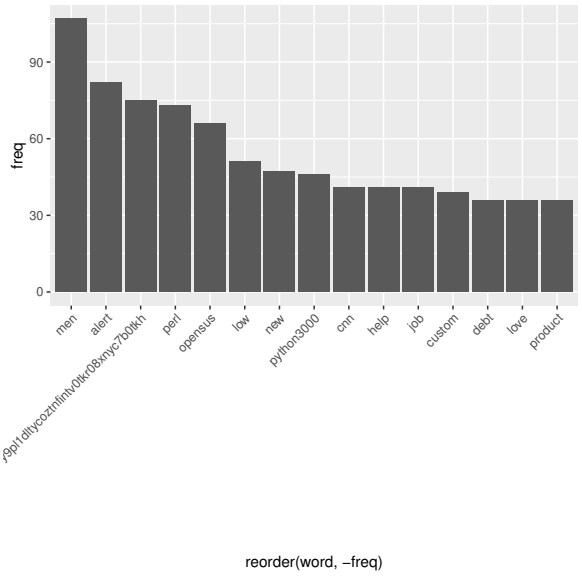

Fig. 6.2: word frequency>35

Appendix B

Code

```
#Load packages
library(readxl)
library(tidyr)
library("ggplot2")
library("lattice")
library("caret")
library("NLP")
library("tm")
library("RColorBrewer")
library("e1071")
library("wordcloud")
library("hellno")
library("randomForest")
library("ElemStatLearn")
library("recipes")
library("dplyr")
library("kernlab")
library("knitr")
library("rpart")
library("SparseM")
library("ROCR")
library("gplots")
library("naivebayes")
library("foreach")
library("iterators")
library("itertools")
library("missForest")
library("plyr")
library("tidyr")
library("gbm")
library("nnet")
library("SnowballC")
library("gmodels")
library("mda")
```

```
library("MASS")
library("stringr")
library("caTools")
library("pROC")

#Importing nonspam data files
setwd("C:/Users/Sabelo Khalani/Music/NonSpam2") #Case 1: spam>ham
setwd("C:/Users/Sabelo Khalani/Music/NonSpam") #Case 2: spam<ham
setwd("C:/Users/Sabelo Khalani/Music/NonSpam1") #Case 3: spam=ham

nonspam_files = list.files(pattern = "*.xlsx")
nonspam_files

rba = lapply(nonspam_files, function(i){
  x = read_excel(i, sheet = 1)
  x
})
rba[[1]]
nonspam = do.call("rbind.data.frame", rba) #combine the files
nrow(nonspam)

#Importing spam data files
setwd("C:/Users/Sabelo Khalani/Music/Spam")
spam_files = list.files(pattern = "*.xls")
spam_files
rba = lapply(spam_files, function(i){
  x = read_excel(i, sheet = 1)
  x
})
rba[[1]]
spam = do.call("rbind.data.frame", rba)
nrow(spam)
colnames(nonspam)= c("type","subject") #rename the columns
nrow(nonspam)
nonspam
colnames(spam)= c("type","subject") #rename the columns
nrow(spam)
spam


spam_ham = rbind(spam,nonspam) # Combine spam and nonspam folders
spam_ham = spam_ham[sample(nrow(spam_ham)),] #Shuffly
spam_ham = spam_ham[sample(nrow(spam_ham), 3000), ] #sample size
table(spam_ham$type) #Total number of spam and nonspam table

#Create a bargraph of spam vs nonspam
```

```
ggp = ggplot(count_data, aes(x= Type, y = Frequency, fill = Type))+
geom_bar(stat = "identity", color = "black") +
geom_text(aes(label = Frequency), vjust = 0) +
labs(title = "Barplot of Spam vs Ham") +
scale_fill_manual(values = c("gold","blue")) +
theme(plot.title = element_text(hjust = 0.5))
ggp
ggp +
 theme(axis.line = element_line(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank())

#Create corpus
corpus = Corpus(VectorSource(spam_ham$subject))
corpus[[1]][1]
corpus = tm_map(corpus, PlainTextDocument)
corpus = tm_map(corpus, tolower)
corpus[[1]][1]
corpus = tm_map(corpus, removePunctuation)
corpus[[1]][1]
corpus = tm_map(corpus, removeWords,
stopwords("english"))
corpus[[1]][1]
corpus = tm_map(corpus, stemDocument)
corpus[[1]][1]

frequencies = DocumentTermMatrix(corpus,
control=list(bounds = list(global = c(11, Inf)))) # Words frequency
sparse = removeSparseTerms(frequencies, 0.995)
tsparse = as.data.frame(as.matrix(sparse))
colnames(tsparse) = make.names(colnames(tsparse))
tsparse$recomended = spam_ham$type
prop.table(table(tsparse$recomended))

#Building word frequency
freq=sort(colSums(as.matrix(sparse)), decreasing=TRUE)
tail(freq, 10)
findFreqTerms(sparse, lowfreq=0) #Identifying terms that appears
                                                    frequently


#Building Word cloud
```

```
set.seed(1234)
wordcloud(words = wf$word, freq = wf$freq, min.freq = 1,
                max.words=500, random.order=FALSE, rot.per=0.35,
                colors=brewer.pal(8, "Dark2"))

#Building models
set.seed(1234)
split = sample.split(tsparse$recomended, SplitRatio = 0.8)
trainsparse = subset(tsparse, split ==TRUE)
testsparse = subset(tsparse, split ==FALSE)
trainsparse$recomended = as.factor(trainsparse$recomended)
testsparse$recomended = as.factor(testsparse$recomended)

rf_model = randomForest(recomended~., data=trainsparse)
predRF = predict(rf_model, newdata=testsparse)
table(testsparse$recomended, predRF)
confusionMatrix(predRF, testsparse$recomended)

convert_count = function(x) {
        x = ifelse(x > 0, 1,0)
        x = factor(x, levels=c(0,1), labels = c("No","Yes"))
        return(x)
}
sms_train = apply(trainsparse, MARGIN = 2, convert_count)
sms_test = apply(testsparse, MARGIN = 2, convert_count)
nb_classifier = naiveBayes(sms_train, trainsparse$recomended,
                                    laplace=1 )
nb_pred = predict(nb_classifier, newdata = sms_test,type ='class')
CrossTable(nb_pred,testsparse$recomended,
prop.chisq = F,
prop.t = F,
prop.c = F,
dnn = c('predicted','actual'))
table(nb_pred)
prop.table(table(nb_pred))
sms_test_bn_val_df = data.frame(sms_test)
sms_pred_class_df = data.frame(nb_pred)
dim(sms_test_bn_val_df)
prop.table(table(sms_pred_class_df))
table(sms_pred_class_df)
confusionMatrix(data=nb_pred, reference = testsparse$recomended,
                                        positive = "ham")


Neural.Net = nnet(recomended~., data=trainsparse, size=1)
nn_pred = predict(Neural.Net, newdata = testsparse,type ='class')
```

```
CrossTable(nn_pred, testsparse$recomended,
prop.chisq = F,
prop.t = F,
prop.c = F,
dnn = c('predicted','actual'))
NNet = confusionMatrix(as.factor(nn_pred),
                                    as.factor(testsparse$recomended))
NNet

svm.linear = svm(recomended~., data=trainsparse, scale=FALSE,
                                                  kernel='linear')
svm.linear
pred.linear = predict(svm.linear, newdata = testsparse)
linear = confusionMatrix(pred.linear, testsparse$recomended)
linear

lda_train = train(recomended~., data=trainsparse, method="lda")
pred_lda = predict(lda_train, testsparse)
confusionMatrix(pred_lda, testsparse$recomended)

#Building ROC curves
rf.ROCR = prediction(as.numeric(predRF),
                  as.numeric((testsparse$recomended)))
print(rf.AUC = as.numeric(performance(rf.ROCR,"auc")@y.values))
rf.prediction = prediction(abs(as.numeric(predRF)),
                  as.numeric(testsparse$recomended))
roc = performance(rf.prediction,"tpr","fpr")
plot(roc, colorize=T, main = "ROC Curves",
        ylab = "Sensitivity",
        xlab = "1 minus Specificity")
abline(a=0, b=1)


nb.ROCR = prediction(as.numeric(nb_pred),
                  as.numeric((testsparse$recomended)))
print(nb.AUC = as.numeric(performance(nb.ROCR,"auc")@y.values))
nb.prediction = prediction(abs(as.numeric(nb_pred)),
                  as.numeric(testsparse$recomended))
roc1 = performance(nb.prediction,"tpr","fpr")
plot(roc1, colorize=T, main = "ROC Curves",
        ylab = "Sensitivity",
        xlab = "1 minus Specificity")
abline(a=0, b=1)


nn_pred = as.factor(nn_pred)
```

```
nn.ROCR = prediction(as.numeric(nn_pred),
                     as.numeric((testsparse$recomended)))
nn.ROCR = prediction(as.numeric(nn_pred),
                     as.factor(testsparse$recomended))
print(nn.AUC = as.numeric(performance(nn.ROCR,"auc")@y.values))
nn.prediction = prediction(abs(as.numeric(nn_pred)),
                           as.numeric(testsparse$recomended))
roc2 = performance(nn.prediction,"tpr","fpr")
plot(roc2, colorize=T, main = "ROC Curves",
        ylab = "Sensitivity",
        xlab = "1 minus Specificity")
abline(a=0, b=1)


svm.ROCR = prediction(as.numeric(pred.linear),
                     as.numeric((testsparse$recomended)))
print(svm.AUC = as.numeric(performance(svm.ROCR,"auc")@y.values))
svm.prediction = prediction(abs(as.numeric(pred.linear)),
                           as.numeric(testsparse$recomended))
roc3 = performance(svm.prediction,"tpr","fpr")
plot(roc3, colorize=T, main = "ROC Curves",
        ylab = "Sensitivity",
        xlab = "1 minus Specificity")
abline(a=0, b=1)


lda.ROCR = prediction(as.numeric(pred_lda),
                           as.numeric((testsparse$recomended)))
print(lda.AUC = as.numeric(performance(lda.ROCR,"auc")@y.values))
lda.prediction = prediction(abs(as.numeric(pred_lda)),
                           as.numeric(testsparse$recomended))
roc4 = performance(lda.prediction,"tpr","fpr")
plot(roc4, colorize=T, main = "ROC Curves",
        ylab = "Sensitivity",
        xlab = "1 minus Specificity")
abline(a=0, b=1)
plot(roc4, col = "black", main = "ROC Curves",
        ylab = "Sensitivity",
        xlab = "1 minus Specificity")


plot(roc1, add = TRUE, col="grey")
plot(roc2, add = TRUE, col = "blue")
plot(roc3, add = TRUE, col = "gold")
plot(roc4, add = TRUE, col = "green")
abline(a=0, b=1)
```

```
auc = performance(rf.prediction, "auc") #Area Under Curve (AUC)
auc = unlist(slot(auc, "y.values"))
auc = round(auc, 4)
legend(.75,.2, auc, title = "AUC (RF)", cex = .6,
                          box.col = "white", text.col = "black")
auc1 = performance(nb.prediction, "auc")
auc1 = unlist(slot(auc1, "y.values"))
auc1 = round(auc1, 4)
legend(.75,.1, auc1, title = "AUC (NB)",
                 cex = .6, box.col = "white",text.col = "grey")
auc2 = performance(nn.prediction, "auc")
auc2 = unlist(slot(auc2, "y.values"))
auc2 = round(auc2, 4)
legend(.89,.3, auc2, title = "AUC (NN)",
                 cex = .6, box.col = "white",text.col = "blue")
auc3 = performance(svm.prediction, "auc")
auc3 = unlist(slot(auc3, "y.values"))
auc3 = round(auc3, 4)
legend(.89,.2, auc3, title = "AUC (SVM)",
                 cex = .6,box.col = "white",text.col = "gold")
auc4 = performance(lda.prediction, "auc")
auc4 = unlist(slot(auc4, "y.values"))
auc4 = round(auc4, 4)
legend(.89,.1, auc4, title = "AUC (LDA)",
                 cex = .6, box.col = "white",text.col = "green")
```

```
#Sample Size graphs
**Accuracy
Sample_Size = c(500,1000,1500,2000,2500,3000)
RF = c(83.00,84.00,85.00,86.25,88.00,87.50)
NB = c(85.00,84.00,86.00,86.75,87.00,86.83)
NN = c(87.00,87.00,88.33,87.75,89.00,87.00)
SVM = c(84.00,85.00,85.67,87.25,88.40,88.33)
LDA = c(86.00,85.50,85.00,87.50,88.60,88.17)

plot(RF,type = "o", lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Sample Size", ylab="Spam accuracy(%)",
                 main="Accuracy based on sample size",xpd=FALSE)
axis(1,at=1:length(Sample_Size),labels=Sample_Size)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
```

```
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)

**Precision
RF = c(98.00,97.12,100.00,99.07,98.91,99.69)
NB = c(98.08,97.12,100.00,98.63,98.53,98.48)
NN = c(98.15,97.27,100.00,98.22,99.28,99.08)
SVM = c(98.04,97.17,100.00,98.21,99.27,98.52)
LDA = c(98.11,97.20,100.00,98.21,99.28,99.10)

plot(RF,type = "o",lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Sample Size", ylab="Spam precision(%)",
                main="Precision based on sample size", xpd = FALSE)
axis(1,at=1:length(Sample_Size),labels=Sample_Size)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)

**Recall
RF = c(75.38,77.69,77.27,80.08,82.67,81.41)
NB = c(78.46,77.69,78.79,81.20,81.46,81.41)
NN = c(81.54,82.31,82.32,83.08,83.89,81.16)
SVM = c(76.92,79.23,78.28,82.33,82.98,83.67)
LDA = c(80.00,80.00,77.27,82.71,83.28,82.91)

plot(RF,type = "o",lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Sample Size", ylab="Spam recall(%)",
        main="Recall based on sample size",xpd=FALSE)
axis(1,at=1:length(Sample_Size),labels=Sample_Size)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
```

```
inset=0.01)

**HitRate
RF = c(68.00,69.79,69.39,71.35,74.67,73.09)
NB = c(70.83,69.79,70.83,72.38,73.25,72.69)
NN = c(73.91,74.44,74.45,74.29,76.13,72.63)
SVM = c(69.39,71.28,70.34,73.45,75.11,75.19)
LDA = c(72.34,72.04,69.39,73.86,75.45,74.53)

plot(RF,type = "o", lwd=2,pch=8,xaxt="n",ylim=c(60,100),col="black",
xlab="Sample Size", ylab="Spam HitRate(%)",
        main="HitRate based on sample size",xpd= FALSE)
axis(1,at=1:length(Sample_Size),labels=Sample_Size)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","brown"),
inset=0.01)
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)

**F1 score
RF = c(85.21,86.33,87.18,88.57,90.06,89.63)
NB = c(87.18,86.33,88.14,89.07,89.19,89.14)
NN = c(89.08,89.17,90.30,90.02,90.94,89.23)
SVM = c(86.21,87.29,87.82,89.57,90.40,90.49)
LDA = c(88.13,87.77,87.18,89.80,90.58,90.28)

plot(RF, type = "o", lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Sample Size",ylab="Spam F1(%)",
main="F1 based on sample size",xpd =FALSE)
axis(1,at=1:length(Sample_Size),labels=Sample_Size)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)
```

```
#Split ratio graghs (spam>nonspam, spam<nonspam and spam=nonspam)

***Accuracy
Results1 = matrix(c(93.49,92.62,92.35,91.97,
93.30,92.40,92.35,91.54,
93.30,92.29,91.86,90.89,
92.65,92.29,91.53,89.59,
89.58,92.40,92.02,91.97
), ncol=5)
rownames(Results1) = c('30%:70%','40%:60%','60%:40%','70%:30%')
colnames(Results1) = c('NN', 'RF', 'SVM', 'LDA', 'NB')
Results1
barplot(Results1,xlab='Models',
ylab='%Accuracy',ylim = c(70, 105), main='Accuracy: spam>nonspam',
col=c("gold","blue","grey","green"),beside=TRUE,
legend=c("30%:70%","40%:60%","60%:40%","70%:30%"), xpd = FALSE,
args.legend = list(bty = "n", x = "topleft", ncol=5, cex =0.9))

***Precision
Results11 = matrix(c(96.98,96.77,96.93,97.48,
97.28,97.52,96.93,97.44,
97.28,97.51,96.88,97.37,
97.21,97.51,96.84,96.36,
99.17,99.15,98.71,99.13
), ncol=5)
rownames(Results11) = c('30%:70%','40%:60%','60%:40%','70%:30%')
colnames(Results11) = c('NN', 'RF', 'SVM', 'LDA', 'NB')
Results11
barplot(Results11,xlab='Models',
ylab='%Precision',ylim = c(70, 105), main='Precision: spam>nonspam',
col=c("gold","blue","grey","green"), beside=TRUE,
legend=c("30%:70%","40%:60%","60%:40%","70%:30%"), xpd = FALSE,
args.legend = list(bty = "n", x = "topleft", ncol=5, cex =0.9))


***Recall
Results111 = matrix(c(82.57,80.00,79.00,77.33,
81.71,78.67,79.00,76.00,
81.71,78.33,77.50,74.00,
79.71,78.33,76.50,70.67,
68.57,77.33,76.50,76.00
), ncol=5)
rownames(Results111) =c('30%:70%','40%:60%','60%:40%','70%:30%')
colnames(Results111) = c('NN', 'RF', 'SVM', 'LDA', 'NB')
```

```
Results111
barplot(Results111,xlab='Models',
ylab='%Recall',ylim = c(70, 105), main='Recall: spam>nonspam',
col=c("gold","blue","grey","green"), beside=TRUE,
legend=c("30%:70%","40%:60%","60%:40%","70%:30%"), xpd = FALSE,
args.legend = list(bty = "n", x = "topleft", ncol=5, cex =0.9))


***HitRate
Results1111 = matrix(c(92.15,91.08,90.69,90.06,
91.81,90.57,90.69,89.53,
91.81,90.44,90.09,88.76,
90.99,90.44,89.69,87.46,
86.79,90.10,89.76,89.60
), ncol=5)
rownames(Results1111) = c('30%:70%','40%:60%','60%:40%','70%:30%')
colnames(Results1111) = c('NN', 'RF', 'SVM', 'LDA', 'NB')
Results1111
barplot(Results1111,xlab='Models',ylab='%HitRate',ylim = c(70, 105),
                                    main='HitRate: spam>nonspam',
col=c("gold","blue","grey","green"), beside=TRUE,
legend=c("30%:70%","40%:60%","60%:40%","70%:30%"), xpd = FALSE,
args.legend = list(bty = "n", x = "topleft", ncol=5, cex =0.9))

***F1
Results_1 = matrix(c(89.20,87.59,87.05,86.24,
88.82,87.09,87.05,85.39,
88.82,86.87,86.11,84.09,
87.59,86.87,85.48,81.54,
81.08,86.89,86.20,86.04), ncol=5)
rownames(Results_1) = c('30%:70%','40%:60%','60%:40%','70%:30%')
colnames(Results_1) = c('NN', 'RF', 'SVM', 'LDA', 'NB')
Results_1
barplot(Results_1,xlab='Models',ylab='%F1 score',ylim = c(70, 105),
                                    main='F1 score:spam>nonspam',
col=c("gold","blue","grey","green"),beside=TRUE,
legend=c("30%:70%","40%:60%","60%:40%","70%:30%"), xpd = FALSE,
args.legend = list(bty = "n", x = "topleft", ncol=5, cex =0.9))


#Feature size (spam>nonspam, spam<nonspam and spam=nonspam)
**Accuracy
features = c(45,60,80,100,120,140)
RF = c(88.93,89.90,91.53,91.53,93.49,93.81)
NB = c(88.83,89.25,91.21,89.58,89.90,91.53)
NN = c(88.93,89.90,91.53,91.53,93.81,93.49)
```

```
SVM = c(88.93,89.90,91.53,91.53,93.49,93.81)
LDA = c(88.93,89.90,91.53,91.86,93.49,93.81)

plot(RF, type = "l", lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Feature size", ylab="Spam accuracy(%)",
main="Accuracy based on feature size(spam>nonspam)",xpd=FALSE)
axis(1,at=1:length(features),labels=features)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="l",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="c",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)



**Precision
RF = c(100,100,97.44,97.44,97.62,97.65)
NB = c(100,100,100,100,100,98.68)
NN = c(100,100,97.44,97.44,97.65,97.65)
SVM = c(100,100,97.44,97.44,97.62,97.65)
LDA = c(100,100,97.44,97.47,97.62,97.65)

plot(RF, type = "o", lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Feature size", ylab="Spam precision(%)",
main="Precision based on feature size(spam>nonspam)",xpd=FALSE)
axis(1,at=1:length(features),labels=features)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="l",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="c",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)

**Recall
RF = c(66,69,76,76,82,83)
NB = c(66,67,73,68,69,75)
NN = c(66,69,76,76,83,83)
SVM = c(66,69,76,76,82,83)
LDA = c(66,69,76,77,82,83)
```

```
plot(RF, type = "o",lwd=2,pch=8,xaxt="n",ylim=c(60,100),col="black",
xlab="Feature size", ylab="Spam recall(%)",
main="Recall based on feature size(spam>nonspam)",xpd =FALSE)
axis(1,at=1:length(features),labels=features)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)

**HitRate
RF = c(85.89,86.25,89.52,89.52,91.93,92.34)
NB = c(85.89,86.25,88.46,86.61,86.97,89.18)
NN = c(85.89,86.90,89.52,89.52,92.34,92.31)
SVM = c(85.89,86.97,89.52,89.52,91.93,92.34)
LDA = c(85.89,86.97,89.52,89.91,91.93,92.34)

plot(RF, type = "o", lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Feature size", ylab="Spam HitRate(%)",
main="HitRate based on feature size(spam>nonspam)",xpd =FALSE)
axis(1,at=1:length(features),labels=features)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)

**F1
RF = c(79.52,81.66,85.39,85.39,89.13,89.73)
NB = c(79.52,80.24,84.39,80.95,81.66,85.23)
NN = c(79.52,81.66,85.39,85.39,89.73,89.73)
SVM = c(79.52,81.66,85.39,85.39,89.13,89.73)
LDA = c(79.52,81.66,85.39,86.02,89.13,89.73)

plot(RF,type = "o",lwd=2,pch=8,xaxt="n",ylim=c(70,100),col="black",
xlab="Feature size", ylab="Spam F1 score(%)",
main="F1 score based on feature size(spam>nonspam)",xpd =FALSE)
```

```
axis(1,at=1:length(features),labels=features)
lines(NB,col="grey", type="b",lwd=2)
lines(NN,col="blue",type="b",pch=18,lwd=2)
lines(SVM, col="gold",type="b",pch=20,lwd=2)
lines(LDA, col="green",type="b",lwd=2)
legend("bottomright",legend=c("RF","NB","NN","SVM","LDA"),
lty=1,lwd=2,pch=21,col=c("black","grey","blue","gold","green"),
ncol=2,bty="n",cex=0.8,
text.col=c("black","grey","blue","gold","green"),
inset=0.01)
```