

https://dspace.org/

An empirical comparison of supervised machine learning techniques for record linkage of data from health facilities and demographic surveillance system: a case study from rural North-East South Africa

Jezile, Vuyokazi Sharon

2018-10

https://hdl.handle.net/10539/27695

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

An Empirical Comparison of Supervised Machine Learning Techniques for Record Linkage of Data from Health Facilities and Demographic Surveillance System: A Case Study from Rural North-East South Africa

Student Name: Vuyokazi Sharon Jezile

Student Number: 0418142M

A research report submitted to the Faculty of Health Sciences, the University of the Witwatersrand in partial fulfilment of the requirements for the degree of Master of Science

In Epidemiology (Research Data Management)

Supervisor: Gideon Nimako

Co-Supervisor: Chodziwadziwa Whiteson Kabudula

University of Witwatersrand

School of Public Health

October 2018

DECLARATION

I hereby declare that this research report is project work carried out by me under the guidance of Drs Gideon Nimako and Chodziwadziwa Kabudula. I have taken care in all respects to honour the intellectual property rights of others and have acknowledged the contribution of others. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

_____

Vuyokazi Sharon Jezile

DEDICATION

I dedicate my research work to my late little sister, Mrs Zinziswa Jezile-Booi. She was my inspiration; she had so much faith in me. She was a strong-willed young woman who fought colon cancer for over six years. I thank her for always being supportive, loving, cheering me on with encouraging words, and being patient during my studies. I would also like to thank my family for standing by me throughout this journey.

ACKNOWLEDGEMENTS

I wish to acknowledge the following people and organisations:

- I am eternally grateful to God, my creator, for giving me the intellect, motivation, passion, and ability to strive for better things.
- While a completed research report bears the single name of the student, the process that leads to its completion is always accomplished through the support and help of many individuals and organisations. I would like to extend my sincere thanks to all of them.
- I would like to express my appreciation to Drs Gideon Nimako and Chodziwadziwa Kabudula for their guidance and supervision, and for providing necessary information regarding the project as well as for their support in completing the project.
- I would also like to thank the Agincourt Health Demographic Surveillance System and Bhubezi Community Health Centre for the great job they did before we could even think about this research.
- Sincere thanks to my family for their support and patience throughout the study period.
- I would like to express my gratitude towards the Wits Health Consortium – Clinical HIV Research Unit for providing financial support and time off work to enable me to undergo this master's programme at the University of the Witwatersrand, Johannesburg, South Africa.
- I would also like to extend my deepest and sincere thanks to Professors Peter Nyasulu, Cindy Firnhaber, and Prudence Ive for motivating and encouraging me to pursue the MSc in Epidemiology Research Data Management degree.

TABLE OF CONTENTS

DECLARATION ..... **i**

DEDICATION ..... **ii**

ACKNOWLEDGEMENTS ..... **iii**

LIST OF Figures ..... **vii**

LIST OF TABLES ..... **viii**

LIST OF ABBREV*I*ATIONS AND ACRONYMS ..... **ix**

ABSTRACT ..... **x**

CHAPTER 1: INTRODUCTION **1**

1.1 Introduction 1

1.2 Problem Statement 4

1.3 Research Motivation 4

1.4 Aims and Objectives 5

1.5 Outline 5

CHAPTER 2: LITERATURE REVIEW **6**

2.1 Introduction 6

2.2 Deterministic Record Linking 6

2.3 Probabilistic Record Linking 6

2.4 Blocking 7

2.5 Jaro-Winkler String Comparator 8

2.6 Weighting 9

2.7 Expectation-Maximization Classification 10

2.8 Support Vector Machine Classification 11

2.9 K-Nearest Neighbor Classification 14

2.10 Evaluation of Record Linkage Performance 17

CHAPTER 3: METHODS 19

3.1 Introduction 19

3.2 Data Sources 19

3.3 Data Pre-processing 20

3.4 Creating Training and Test Datasets 21

3.5 Experimental Setup 22

3.5.1 Expectation Maximization Method 23

3.5.2 Support Vector Machine and K-Nearest Neighbour Models 23

3.5.3 Evaluating the Record Linkage Performance 24

CHAPTER 4: RESULTS 25

4.1 Introduction 25

4.2 Description of the Data 25

4.3 Unsupervised Machine Learning Technique (EM Algorithm) Results 27

4.4 Supervised Machine Learning Technique (SVM and KNN) Results 29

4.4.1 SVM Results 29

4.4.2 KNN Results 30

4.4.3 Evaluation Results 31

CHAPTER 5: DISCUSSION 33

5.1 Introduction 33

5.1.1 SVM 34

5.1.2 KNN 34

5.1.3 EM 35

5.2 Comparison of the Techniques 35

5.2.1 SVM and KNN 36

5.2.2 SVM/KNN and EM 36

5.3 Limitations of the Study 37

CHAPTER 6: CONCLUSION AND RECOMMENDATIONS 38

6.1 Conclusion 38

6.2 Ethical Considerations 39

6.3 Future Work 39

REFERENCES 40

APPENDICES 44

Appendix I: Plagiarism Policy 44

Appendix II: Ethics Approval 45

Appendix III: Source Scripts (Python: Processing of the datasets) 46

Appendix IV: Source Scripts (Identifier Agreement) 47

Appendix V: Source Scripts (EM algorithm) 50

Appendix VI: Source Scripts (SVM and KNN Contingency tables) 54

LIST OF Figures

Figure 1: General process of record linkage (1) 7

Figure 2: Identifier agreement block 1 26

Figure 3: Identifier agreement block 2 26

Figure 4: Identifier agreement block 3 27

Figure 5: Sensitivity vs Positive Predictive Value 34

Figure 6: Performance Scores of Supervised and Unsupervised Techniques 35

LIST OF TABLES

Table 1: Illustration of weight vector calculations 9

Table 2: Illustration of SVM classification algorithm provided by Christen (2008) (15) 13

Table 3: Illustration of KNN classification algorithm (15) 15

Table 4**:** Three blocks that are not mutually exclusive 21

Table 5: Field agreement criteria 22

Table 6: Identifier Agreement Percentage by match status and blocking criteria 25

Table 7: EM Contingency table – Block 1, 2 and 3 28

Table 8: SVM Contingency table - Block 1, 2 and 3 30

Table 9: KNN Contingency table - Block 1, 2 and 3 30

Table 10: F-score Performance(45) 31

Table 11: Accuracy statistics SVM and KNN record linkage techniques 32

LIST OF ABBREV*I*ATIONS AND ACRONYMS

AIDS	Acquired Immune Deficiency Syndrome
ART	Antiretroviral Treatment
BCHC	Bhubezi Community Health Centre
CLR	Common Language Runtime
CM	Confusion Matrix
EM	Expectation-Maximization
FN	False Negative
FP	False Positive
HIV	Human Immunodeficiency Virus
HDSS	Health and Demographic Surveillance System
JW	Jaro-Winkler
KNN	K-nearest Neighbor
MRC	Medical Research Council
NAN	Not a number
NN	Nearest Neighbor
NPV	Negative Predictive Value
PPV	Positive Predictive Value
RAM	Random Access Memory
SQL	Structured Query Language
SVM	Support Vector Machine
TB	Tuberculosis
TN	True Negative
TP	True Positive
W_M	Matched Weight Vector
W_N	Unclassified Weight Vector

ABSTRACT

Record linkage of electronic patient records based on conventional personal identifiers is the most widely used method of integrating information from different sources. The record linkage of Health Demographic Surveillance System data and hospital records offer an opportunity for researchers to improve the quality of clinical research data and to discover new insights from clinical research data repositories. Record linkage algorithms are usually applied to different data sources with the aim of increasing the amount of information available to answer research questions that could not be originally answered from individual databases. Most record linkage systems do not compare results (using sensitivity and specificity criteria) from different algorithms before ascertaining a match, possible match, or non-match. There is minimal research done in comparing these different algorithms using health-related datasets. This study seeks to fill this gap by implementing and comparing the K-nearest neighbor classification (also known as K-nearest neighbor algorithm) and Support Vector Machine with deterministic and probabilistic record-linking techniques. The empirical evaluation performed on a linkage between the core dataset from The Agincourt Health and Demographic Surveillance System and electronic medical records from Bhubezi Community Health Centre. This work serves as a basis for other Health and Demographic Surveillance System sites in linking their Demographic Surveillance System census datasets with external data repositories. Match rates and error rates for the three strategies are compared, and discussions of their similarities and differences, strengths, and weaknesses are presented. The results showed that the supervised machine learning techniques performed better compared to unsupervised techniques. Support Vector Machine (SVM) and K Nearest Neighbor (KNN) techniques had a high sensitivity, specificity and PPV for all the three blocks respectively. However, when looking at the number of true matched records, KNN performed better when compared to Support Vector Machine in all the three blocks. The true matches for K-nearest neighbor were 1 995, 1 727, and 911 respectively with sensitivity ranging from 90%, 84.68% to 96.16%. The positive predictive value ranged from 93.33%, 83.62% to 70.23% respectively. The f-score measure for SVM and KNN is ranging from 99.97% to 99.98%, however for EM its ranging from 87% to 91%. The results clearly show the supervised machine learning techniques perform very well compared to the unsupervised techniques.

CHAPTER 1: INTRODUCTION

1.1 Introduction

Record linkage is a process of combining information pertaining to the same individual from different data sources. The main aim of performing record linkage is usually to increase the amount of information available to answer research questions that could not be answered from individual data sources (1). For many organisations and research institutions, record linkage is beneficial for enriching and cleaning data.

In recent years, record-linking techniques have increasingly been used in linking data from databases containing health-related information (2-5). Record linkage of cohort data with health services administrative datasets has been routine in several developed countries, including the United Kingdom (6, 7), Canada (8, 9), and Sweden (10) for many years. For example, Morin and colleagues (11) measured the burden of medication in older adults near their end of life. This was done by assessing the decedents' characteristics at the time of death through record linkage with the National Patient Register, the Social Services Register, and the Swedish Education Register (11). Another example is the study that was conducted by Beauchamp and colleagues(2). The study considered validation of de-identified record linkage to ascertain hospital admissions in a cohort study (2). This study yielded high-quality record linkage of cohort data with limited identifiers. In recent years, there have also been some examples of studies using the record linkage technique conducted in less developed countries. For instance, Kabudula and colleagues (12, 13) used record linkage to link mortality data from a Heath and Demographic Surveillance System (HDSS) and the National Civil Registration System in South Africa (13). Another study used record linkage to enrich the utility of biobanks (a collection of biomedical samples with medical, genetic, and/or genealogic data), for example, in assessing the effect of exposures on health outcomes, by linking them to certain registries (1).

Record linkage procedures essentially involve forming record pairs from two databases, say $\boldsymbol{A}$ and $\boldsymbol{B}$, by matching/comparing selected common variables in $\boldsymbol{A}$ and $\boldsymbol{B}$. The comparison results are stored in a vector of scores based on some metrics, representing the extent of agreement of the two records in the comparison fields/variables. The record pairs are then classified as relating to the same individual or not using two broad

approaches: deterministic and probabilistic techniques. These two techniques are widely used in algorithms implemented in record linkage packages.

Deterministic record linkage is the simplest of all record linkage methods. In the deterministic technique, records are linked using unique identifiers, if available in the two data sources. Classification of records from two datasets as belonging to the same person is based on exact agreement of some identifying personal information such as fingerprints, a social security or national identification number, or a set of conventional personal identifiers (e.g. a combination of first name, last name, and date of birth) (12). In contrast to deterministic record linkage methods, probabilistic record linkage methods classify pairs of records from two data sources as belonging to the same individual based on the statistical probability that common identifiers drawn from the two data sources belong to the same individual (12). Probabilistic record linkage is more reliable when unique identifiers are not available(12). In record linkage literature, probabilistic record linkage techniques are divided into two classes. One class is based on the Fellegi-Sunter framework (1969) and classifies record pairs as matches and non-matches based solely on the observed data without using a training set (12). The other class is based on machine learning techniques.

The machine learning methods for record linkage are categorised into two main groups: supervised and unsupervised learning (14). When a training set is available and used, the method is supervised; otherwise, the method is unsupervised. In a supervised learning problem, each training data sample would consist of both the input values and the associated output value $\boldsymbol{y}$, being either 1-agreement or 0-disagreement. With such training data information this classification problem can be solved easily. The goal of unsupervised learning methods is to process or extract information with the knowledge of the input data only, the output $\mathbf{y}$ is not known or is assumed to be not known. The unsupervised learning techniques can be very interesting and helpful for data preprocessing. For an unsupervised learning problem, the outputs are unknown, that is, the training data consists only of the input data without any information about the associated classes. This study assessed the performance of two supervised machine learning methods, namely, Support Vector Machine (SVM) and K-nearest neighbor (KNN) Classification.

Previous work has shown that SVM is one of the most popular supervised machine learning that has been employed successfully for record pair classification (15). Support Vector Machines (SVMs) are a complex

but powerful classification technique. Classification is based on which side of the dividing line new data falls. When working on numerical data, SVMs classify by finding the dividing line (formally called a maximum-margin hyperplane) that separates data most clearly. This can get very complex even for technical people, though once the SVM is constructed it is simple to use. While most commonly used for classification problems, SVMs have been extended to regression tasks in recent years. SVMs are often used with very complex data for applications such as recognizing handwriting, facial expression classification, and classifying images(16). The advantages of using SVM is that, they are very fast at classifying new data. Secondly, there is no need to go through training data for new classifications. SVMs can work for a mixture of categorical and numerical data and are highly accurate. SVMs are robust to high dimensionality, meaning they can work well even with a large number of features. SVMs can be a highly effective classifier, but you may not be able to figure out why it makes those classifications, typically requires a very large dataset. Other methods, like decision trees, can still give interesting output for small data sets; this may not be the case with a SVMs(17). The kernel trick is real strength of SVM, with an appropriate kernel function; we can solve any complex problem(18). SVM's are very good when we have no idea on the data. SVM scales relatively well to high dimensional data. However, the disadvantages of using SVMs are the time it takes to choose the right kernel function, difficult to understand and interpret the final model, variable weights and individual impact and long training time for large datasets. KNN can handle complex numerical functions while still being easy to interpret. However, KNN can be very slow for large datasets, require a lot of space and are computationally expensive when there are millions of variables(19). However very useful if data is difficult/expensive to collect. It is easy to determine which neighbors are used for the final predictions. The scaling process can reveal which variables are unimportant for making predictions and thus can be thrown out. KNN is robust to noisy training data and very effective if the training data is large(20). SVM and KNN can achieve better classification results for record linkage than other unsupervised approaches (15). According to Winkler, the EM algorithm provides very accurate estimates of *m*- and *u*-probabilities in situations where the amount of typographical error in the identifiers is minimal, but performs poorly when the identifiers contain numerous typographical errors(21).

There has not been comparative research on the performance of different record-linking techniques using data from rural African settings. This study assessed the performance of different record linkage

techniques using data from an HDSS and a healthcare facility in a rural South African setting. The healthcare facility used in this study was Bhubezi Community Health Care (BCHC). The primary focus is on the empirical evaluation of machine learning techniques. We identified the most commonly used machine learning algorithms for classification are Expectation Maximization (EM) ,Support Vector Machines and K-Nearest Neighbor(17). We have weighed the advantages and disadvantages of the available methods in the context of the project and have chosen the method that best fits the timeline, chosen resources, and the research question.

1.2 Problem Statement

Data from Health and Demographic Surveillance Systems are critical for understanding population and health and their dynamics and determinants in settings where there are limited resources (such as good health statistics registries)(4, 12, 15). Nevertheless, the utility of data from most HDSSs is usually limited due to lack of integration with other administrative data, including those originating from health facilities (22). Previous work by Kabudula and others (12) demonstrated the feasibility of integrating information from an HDSS and health facilities using record linkage approaches. However, this work only considered probabilistic record linkage methods based on the Fellegi-Sunter framework. In their work, the best fully automated record linkage scenario produced a sensitivity of 83.60% and a positive predictive value (PPV) of 95.10%. It is not known whether the inclusion of supervised machine learning approaches in the record linkage process could improve the matching statistics. Previous work has shown that nearest neighbor and SVM approaches can achieve better classification results for record linkage than other unsupervised approaches (15).

1.3 Research Motivation

This study was motivated by the previous record linkage conducted by Kabudula and colleagues using data from the Agincourt HDSS (12). Their work demonstrated that it was feasible to integrate information from an HDSS and health facilities using a combination of deterministic record linkage approaches and probabilistic record linkage approaches based on the Fellegi-Sunter framework(12). In this study, the focus was on the evaluation of the performance of other record-linking techniques in linking HDSS data and clinical

records. In this study, the performance of nearest neighbor and SVM record linkage approaches are compared to the probabilistic record linking approaches based on the Fellegi-Sunter framework.

1.4 Aims and Objectives

This study aimed to empirically evaluate the performance of various record-linking techniques in linking HDSS data and clinical records. The specific objectives include:

i. To fit EM, SVM and KNN models using core dataset from Agincourt HDSS and electronic records from BCHC
ii. To implement the EM,SVM and KNN classifications and evaluate their performance in record linkage of HDSS and BCHC
iii. To compare the performance of SVM and KNN techniques to that of the EM record linkage technique based on the Fellegi-Sunter framework for record linkage of HDSS and BCHC

1.5 Outline

This report is organised into six chapters. Following this introductory chapter (i.e. Chapter 1), Chapter 2 presents the literature review of the algorithms assessed in this study: EM, SVM and KNN; Chapter 3 describes the techniques used in this study, and further describes the experimental setup; Chapter 4 presents the results of record linkage using EM, SVM, and KNN; Chapter 5 discusses the findings of the study and provides an overview of how this study can be scaled to other HDSS settings; and finally, Chapter 6 details some ethical considerations and provides a conclusion and future directions.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The previous chapter provided an introduction and background to the study. This chapter reviews literature on key concepts in record linkage. The work done by various record linkage techniques are also reviewed.

2.2 Deterministic Record Linking

Deterministic record linkage assumes error-free identifying fields and links records that exactly match on these identifying fields (23). The matched dataset can be referred to as the gold standard which is usually used in supervised learning(12, 24, 25). This algorithm determines whether record pairs agree or disagree, based on the set of identifiers. The match status can be assessed in a single step or multiple steps. In the single step, all records are compared at once using a full set of identifiers. In multi-steps, records are matched in a series of progressively less-restrictive steps in which record pairs that do not meet the first round of match criteria are passed on to the second round of matching; otherwise, it is classified as a non-match. These two approaches can also be called exact deterministic (requiring an exact match on all identifiers) and approximately or interactive deterministic (requiring an exact match on one of the several rounds of matching but not all possible identifiers) matches respectively (24).

2.3 Probabilistic Record Linking

Probabilistic record linkage offers a way of integrating data from different sources where there are no shared unique identifiers. Probabilistic record linking uses the Fellegi-Sunter model. The Fellegi- Sunter model provides a record linkage classification framework based on statistical and probabilistic principles where record pairs can be designated as matches, possible matches, or non-matches, based on a calculation of linkage scores and the application of decision rules (24). The method designates records belonging to the same entity by comparing record pairs, checking for immediate conflict (not the same person), calculating similarity scores, and weighing the scores by frequency (26, 27). One of the drawbacks of the probabilistic record linkage model is its ability to handle only binary or categorical comparison vector attributes. One way to overcome this disadvantage is to use other available machine learning approaches.

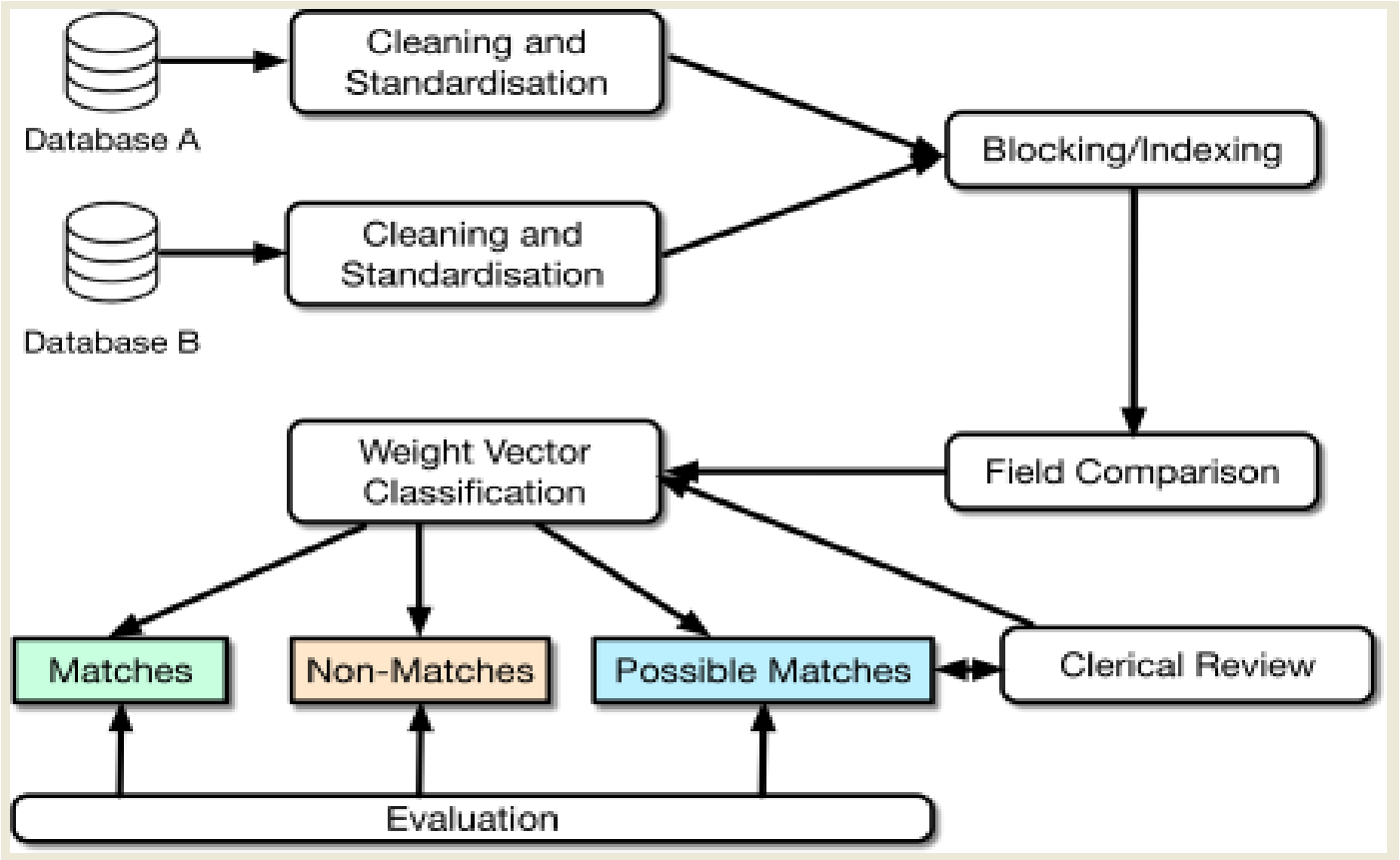


Figure 1: General process of record linkage (1)

2.4 Blocking

If there are two datasets, file A and file B, where file A comprises 200 records and file B comprises 2 000 records, a Cartesian product that may make up all possible record pairs (A*B) can be used, with 200*2 000 = 200 000 possible matches. Each record pair may be either a possible true match or a possible true non-match. However, when dealing with huge datasets, computing this is highly impractical. Therefore, it is important to reduce the comparison space by restricting it to only those record pairs that meet certain basic criteria. This is referred to as blocking.

Blocking is a processing step that matches all the records between datasets with fast and accurate methods to generate candidate match pairs. Candidate match pairs generated are small to minimise the number of detailed comparisons. The advantage of blocking is that it reduces the computing time involved in record linkage. Blocking works to functionally separate an enormous dataset into a small dataset of individuals with at least one common characteristic, such as month of birth or village of residence (24). All the record pairs that do not meet the matching criteria specified in the blocking phase are automatically classified as non-

matches and removed. Blocking is a processing step that matches all the records between datasets with fast and accurate methods to generate candidate match pairs. The advantage of blocking is that it reduces the computing time involved in record linkage. Data partitioning of record linkage helps in facilitating the efficient and effective management of big data. For example, the number of matched pairs to be considered may be limited to only those matched pairs that agree on a month of birth and/or place of residence. Blocking can influence linkage successes, and researchers are encouraged to report the specific steps of blocking strategy (24). Candidate matches generated should be small to minimise the number of detailed comparisons.

Since only record pairs in the candidate set are examined in detail during record linkage, a multi-pass approach is used in most blocking techniques to ensure that the candidate set does not leave out possible true matches (28). Examples of how to use multiple passes on databases to obtain candidate record pairs for record linkages are provided by Michelson and Knoblock (28). Each of the multiple passes corresponds with a different sort ordering of the files, and pairs are only considered in a sliding window of fixed size (29-32). After the pairs are blocked, advanced and intensive computational methods are used for comparing the pairs (5). This approach generates candidate matches using different attributes and methods across independent runs. However, the effectiveness of the multi-pass approach depends on which attributes are chosen and the methods used (1, 21).

2.5 Jaro-Winkler String Comparator

String comparison metrics are used to accommodate typographical errors on string fields in record linkage. One of the most commonly used methods is the Jaro-Winkler (JW) metric. The JW string comparator is used to measure the similarity between two strings (33) and is particularly well suited for personal names. The JW string comparator returns values between 0 (complete disagreement) and 1 (exact agreement) as a measure of similarity between two strings (33). The equation used for JW string comparator is expressed as

$$\boldsymbol{dj} = \begin{cases} \mathbf{0} \\ \frac{\mathbf{1}}{\mathbf{3}}\left(\frac{\boldsymbol{m}}{|\boldsymbol{s1}|} + \frac{\boldsymbol{m}}{|\boldsymbol{s2}|} + \frac{\boldsymbol{m}-\boldsymbol{t}}{\boldsymbol{m}}\right) \end{cases} \boldsymbol{if}\ \boldsymbol{m} = \mathbf{0} \qquad \text{Equation 1}$$

where $\boldsymbol{dj}$ is the JW score, m is the number of matching characters, t is half the number of transpositions, and the two characters from string1 (S_1) and string2 (S_2) respectively are considered matching probability (m) only if they are the same. Each character of S_1 is compared with all its matching characters in S_2.

2.6 Weighting

Record linkage employs weight given each record compared. Furthermore, likelihood ratio is assessed using the weight assigned to agreement or disagreement on each identifier. This is done by comparing the probability that true matches agree on the identifier (*m*-probability) to the probability that false matches randomly agree on the identifier (*u*-probability)(34). The *m*-probability can be estimated based on values reported in published linkage literature or by taking a random sample of pairs from the comparison space, assigning match status via manual review, and calculating the probability that two records agree on a particular identifier when they are true matches(34). The *u*-probability can be calculated by observing the probability that two records agree on a particular identifier merely by chance; for example, the u-probability for a month of birth is 1/12 or 0.083. An agreement weight is computed by dividing the *m*-probability by the *u*-probability and taking the $\log_2$ of the quotient, this is when two records agree on an identifier. For example, if the probability that true matches agree on the month of birth is 97% and the probability that false matches randomly agree on the month of birth is 8.3% (1/12), then the agreement weight for the month of birth would be calculated as ($\log_2$[.97/.083]) or 3.54(34). Weighting involves giving a score to each record pair that is compared during record linkage.

Table 1: Illustration of weight vector calculations

V1:	**Kristine**	**South**	**55**	**West**	**Road**
V2:	**Kristina**	**South**	**55**	**West**	**Rd**
V3:	**Ric**	**O' Conner**	**9**	**North**	**Straat**
V4:	**Ricardo**	**Conner**	**10**	**Nothy**	**Strat**

WV(V1, V2): [0.9, 1.0, 1.0, 1.0, 0.9]

WV(V2, V3): [0.0, 0.0, 0.0, 0.0, 0.0]

WV(V2, V4): [0.0, 0.0, 0.5, 0.0, 0.0]

WV(V3, V4): [0.7, 0.3, 0.5, 0.7, 0.9]

A weight vector ($\boldsymbol{WV}$) contains the matching weights calculated for each compared record pair. Using weight vectors, candidate pairs are then classified into possible matches, non-matches, and matches (1). Table 1 shows how the weight vectors are considered. For example, 0.9 in the first element of (V1, V2) is the JW score of Kristine and Kristina. One (1) in the second element of (V1, V2) uses the agreement status recorded as 1 for "same" and 0 for "difference". Once the weights, full and partial, for each identifier have been calculated, the linkage score for each matched pair is equal to the sum of the weights across all linkage identifiers(34).

2.7 Expectation-Maximization Classification

The EM algorithm is a commonly used probabilistic algorithm for finding maximum likelihood estimates of unknown parameters.The Expectation-Maximization (EM) algorithm is a general iterative algorithm for maximum likelihood estimation in incomplete data problems (34-37). The key idea of the EM algorithm is that "missing data" are not the missing observations but functions, such as sufficient statistics, of the observations that appear in the complete data log likelihood. The EM algorithm mainly involves two steps. The first step is called the expectation-step ("E step"), whereas the second step is called the maximization-step ("M step") (34-37). In the E step, the conditional expectation of the "missing data", given the observed data parameters and the current estimate of the model parameters, is found. The M step performs the maximum likelihood estimation of the parameters as if there were no missing values, which implies that it is no different from normal maximum likelihood methods. In record linkage, the EM algorithm is used to estimate the values needed in weight computation for each field in the context of the Fellegi-Sunter probabilistic record linkage model (34-37). In each case, expectation maximization provides a simple, easy-to-implement and efficient tool for learning parameters of a model. The parameters with multiple iterations of the E step and M step can be defined as indicated in Equations 2 - 6 below to establish estimates for the proportion of true links (p) and identifier agreement rates (m_k and u_k), also known as probability mass functions (36). As in (36), for the E step, the unknown values for g_j are estimated using ($g_m(y^j)$,$g_u(y^j)$), where(31,32,33,34,35,36):

$$g_m(y^j) = \frac{p\prod_{k=1}^{n} m_k\, y_k^j (1-m_k)^{\,1-y_k^j}}{p\prod_{k=1}^{n} m_k y_k^j (1-m_k)^{\,1-y_k^j} + (1-p)\prod_{k=1}^{n} u_k\; y_k^j (1-u_k\;)^{\,1-y_k^j}}$$ Equation 2

$$g_u(y^j)\frac{p\prod_{k=1}^{n} u_k\, y_k^j (1-u_k)^{\,1-y_k^j}}{p\prod_{k=1}^{n} u_k y_k^j (1-u_k)^{\,1-y_k^j} + (1-p)\prod_{k=1}^{n} m_k\; y_k^j (1-m_k\;)^{\,1-y_k^j}}$$ Equation 3

Also as shown in (36), the M step yields estimates of m, u and p parameters using the following equations:

$$p = \frac{\sum_{j=1}^{N} g_m(y^j)}{N}$$ Equation 4

$$m_k = \frac{\sum_{j=1}^{N} y_k^j \cdot g_m(y^j)}{\sum_{j=1}^{N} g_m(y^j)}$$ Equation 5

$$u_k = \frac{\sum_{j=1}^{N} y_k^j \cdot g_u(y^j)}{\sum_{j=1}^{N} g_u(y^j)}$$ Equation 6

The advantage of the general EM-algorithm discussed in this section is that there are no additional assumptions made, which are potentially violated. Winkler (21) proposed a slightly different way of estimating the *m*- and *μ*-marginal probability mass functions based on information on the distribution of attributes in the populations. His method proposes an approach to deal with the lack of knowledge about the distribution of attributes in the population of true links(21). Both the *m*-probability mass function and the *u*-probability mass function play an important role in the framework (34).

2.8 Support Vector Machine Classification

SVMs are one of the supervised learning techniques used for classification and regression analysis. The critical elements for supervised learning techniques are training and test datasets. Training datasets are data used in various areas of information science to discover potential predictive relationships. On the other hand, test datasets are data used to assess the strength and utility of the predictive relationships.

A supervised learning technique can be called a classifier when its goal is to build a classification model (14). Various authors have demonstrated that in machine learning, SVM classification approach stands out from the others with its better classification performance and yields very good results compared to other methods (21, 30, 38). As mentioned above, the real strength of SVM is kernel function, therefore SVM classifier is a kernel-based supervised learning algorithm that classifies the data into two or more classes. classifier. A Kernel function is a mapping procedure done to the training dataset to improve its similarity to a linearly separable dataset. The function of mapping is to increase the dimensionality of the dataset and it is done efficiently using a kernel function. Different SVM algorithms use different types of kernel functions, some of the commonly used kernel functions are linear, Gaussian radial basis function (RBF), nonlinear, quadratic, Multilayer Perceptron kernel, and Polynomial kernel. In this research work, linear and RBF kernel functions were used(17, 39). The linear kernel function and RBF kernel functions were used due to their dissimilar characteristics. The linear kernel function performs well with linearly separable dataset and the RBF kernel function performs well with non-linear data set. The linear kernel function takes less time to train the SVM compared to the RBF kernel function. The linear kernel function performs well with linearly separable data set and the RBF kernel function performs well with non-linear dataset. The linear kernel function is less prone to over fitting compared to the RBF kernel function. The performance of the SVM classifier relies on the choice of the regularization parameter C that is known as box constraint and the kernel parameter that is also known as the scaling factor. Together they are known as the hyperplane parameter. The SVM classifier SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. The mathematical representation of linear kernel function:

$$K(X_i; X_j) = X_i^T X_j) \qquad \text{Equation 7}$$

SVM is particularly designed for binary classification. During the training phase, SVM builds a model, maps the decision boundary for each class, and specifies the hyperplane that separates the different classes. Increasing the distance between the classes by increasing the hyperplane margin helps increase the classification accuracy. SVM can be used to effectively perform nonlinear classification. This is why SVMs are also called as large margin classifiers since they create a large margin between your data points and the decision boundary (or the hyper-plane). The most important training instances are the ones that are

making up the boundary. The models are generally much more complex then the distance based classifiers, but consider more information when classifying the target instance and will generally have much better accuracy. However, the major disadvantage for SVM is its relatively complex training and classifying of algorithms, as well as the high time and memory consumption during both the training and classification stages. Weight vectors were sorted according to their distances, showing resemblances and differences respectively (15, 40). An illustration of the SVM classification algorithm, as provided by Christen (2008) (15), is presented in Table 2.

Table 2: Illustration of SVM classification algorithm provided by Christen (2008) (15)

Input	**Output**
- Complete weight vector set: W	- Weight vectors classified as matches: Z_M
- Seed training examples match set: W_M	- Weight vectors classified as non-matches: Z_N
- Seed training examples non-match set: W_N	1: T_M := $W_M$and T_N:= W_N
- Increment percentage: ip	2: W_U := W \ ($W_M \cup W_N$)
- Total training percentage: tp	3: svm0 := train svm(T_M, T_N)
	4: i := 0
	5: while (|T_M | + |T_N |) < (|W| $*$ tp/100) do:
	6: X_M,, X_N := svm classify (svmi, W_U)
	7: Sort XM and X_N according to distance from svmi decision boundary (hyperplane)
	8: Y_M := |X_M, | $*$ (ip/100) vectors from X_M furthest away from decision boundary
	9: Y_N := |X_N | $*$ (ip/100) vectors from X_N furthest away from decision boundary
	10: T_M:= $T_M \cup Y_M$
	11: T_N := $T_N \cup Y_N$
	12: i := i + 1

Input	Output
	13: svmi := train svm(T_M, T_N) 14: W_U := W_U \ (Y_M ∪ Y_N) 15: end while 16: X_M, X_N:= svm classify(svmi, W_U) 17: Z_M := T_M ∪ X_M and Z_N := T_N ∪ X_N

2.9 K-Nearest Neighbor Classification

The K-nearest neighbor is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g. distance functions). K is the number of nearest neighbours to be considered; if K is too small, the method is susceptible to noise in the data and if it is too large, the decision is smeared out, covering too great an area of instance space. KNN was already used in statistical estimation and pattern recognition at the beginning of the 1970s as a non-parametric technique (15, 41). KNN is a non-parametric and instant-based, lazy learning algorithm, meaning that it does not make any assumptions on the underlying data distribution. Non-parametric means it makes no explicit assumptions about the functional form, avoiding the dangers of mismodelling of the underlying distribution of the data. Instance-based learning means that the algorithm does not explicitly learn a model. Instead, it chooses to memorise the training instances which are subsequently used as “knowledge” for the prediction phase. Concretely, this means that only when a query to the database is made (i.e. when one asks it to predict a label, given an input) the algorithm uses the training instances to spit out an answer.

KNN does not use the training data points to do any generalisation. Training examples can be selected using two main approaches, either a nearest-based or distance threshold approach. In the nearest-based approach, similarities and dissimilarities vectors are determined by sorting weight vectors according to their distances. However, previous research has proven that the nearest neighbor-based approach generally outperforms other classifications by its simplicity. KNN uses the threshold approach based on selection because it allows explicit specification of the number of weight vectors to be included into matched and non-matched weights (15, 42). The major disadvantage of KNN is that it uses all features in distance computation

and causes the method to be computationally intensive, especially when the size of the training set grows. In addition, the accuracy of KNN is severely degraded by the level of noisy features, especially when the number of attributes grows. The K-nearest neighbor is very efficient at training time, since training simply consists of storing the training set. However, testing is much slower, since it can potentially involve measuring the distance between the test point and every training point, which can make K-nearest neighbor impractical for large data sets.

In the KNN algorithm, the classes of its K -nearest neighbors, determine the classification of a new test feature vector. Here, the KNN algorithm was implemented using Euclidean distance metrics to locate the nearest neighbor [23]. The Euclidean distance metrics $d(x, y)$ between two points x and y is calculated using the Eq. (2). Where N is the number of features such that x = {x1, x2, x3…xN} k and y = {y1, y2, y3…yN}. The number of neighbors (i.e., k) used to classify the new test vector was varied in the range of 1 to 10, and its effects on the classification performance were determined in the form of classification accuracy with standard deviation.

$$d(x, y) = \sum_{i=1}^{N} \sqrt{x_i^2 - y_i^2} \qquad \text{Equation 8}$$

Evaluation of SVM and KNN classifiers is done using the weight vector classification of unmatched ($\boldsymbol{W_N}$) and matching ($\boldsymbol{W_M}$) records, using certain parameters such as sensitivity, positive predictive value, specificity, and (F-score). Table 30 shows the K-nearest neighbor classification algorithm, as provided by Christen (2008) (15).

Table 3: Illustration of KNN classification algorithm (15)

Input	Output
- Complete weight vector set: W - Seed training examples match set: W_M	- Weight vectors classified as matches: Z_M - Weight vectors classified as non-matches: Z_N

Input	Output
- Seed training examples non-match set: W_N - Distance function: dist - Number of nearest neighbors to consider: k	1: Z_M: = WM and Z_N: = W_N 2: W_T := ($W_M \cup W_N$) and : W_U= W \ W_T 3: Initialise empty heap H 4: := [], N := [], U := [] 5: for ($w_m \in W_M$) do: 6: M[w_m] := {(k + 1) nearest $w_u \in W_U$, sorted according to dist(w_m, w_u)}(12, 43) 7: end for 8: for ($w_n \in W_N$) do: 9: N[w_n] := {(k + 1) nearest $w_u \in W_U$, sorted according to dist(wn, w_u)} 10: end for 11: for ($w_u \in W_U$) do: 12: U[w_u] := {(k + 1) nearest $w_t \in W_T$, sorted according to dist(w_u, w_t)} 13: s := P $w_t \in$U[w_u][1:k] dist(w_u, w_t) 14: Insert (s, w_u) into H 15: end for 16: while (W_U 6= ∅) do: 17: (s, w_t) := first element in H 18: W_U := W_U − w_t 19: if (U[wt] contains more weight vectors from ZM than ZN) then: 20: Z_M:= Z_M+ w_t 21: else: 22: Z_N := Z_N + w_t 23: end if 24: X_u := ∪$w_u \in$U[wt](M[w_u] ∪ N[w_u]) 25: for (∈ $w_u X_u$) do: 26: d := dist(w_t, w_u)

Input	Output
	27: if (w_u is one of (k + 1) nearest to w_t) then: 28: Update w_t in $U[w_u]$ with d 29: s := P $w_v \in U[w_u]$[1:k] dist(w_u, w_v) 30: Update (s, w_u) in H 31: end if 32: end for 33: if ($w_t \in Z_M$) then: 34: M[w_t]: = {(k + 1) nearest $w_u \in X_u$, sorted according to dist(w_t, w_u)} 35: else: 36: U[w_t] := {(k + 1) nearest wu $\in X_u$, sorted according to dist(w_t, w_u)} 37: end if 38: end while

2.10 Evaluation of Record Linkage Performance

The record linkage performance is measured by a number of indicators described below. Firstly are the familiar indicators: specificity and sensitivity. Other indicators are Positive Predictive Value (PPV), Negative Predictive Value (NPV) and F-Score. The formulas for these measures are as follows:

$$\textbf{Sensitivity} = \frac{\textbf{True Positive (TP)}}{(\textbf{True Positive (TP)} + \textbf{False Negative(FN)})} \qquad \text{Equation 9}$$

True positive matches are all the records that match successfully, and false matches are all the records that do not match. However, there may be records that do not match due to discrepancies (false negatives); those that match in error are known as false positive (44). Sensitivity is the capacity of the linkage process to recognise the true matches, while specificity is the capacity of the linkage process to recognise non-

matches. Sensitivity is also known as recall. PPV represents the proportion of matched pairs classified by the algorithm as matches that are true matches; it is also known as a precision measure.

$$\textbf{Positive Predictive Value} = \frac{\textbf{True Positive (TP)}}{(\textbf{True Positive(TP)} + \textbf{False Positive(FP)})} \quad \text{Equation 10}$$

NPV represents the proportion of matched pairs classified by the algorithm as non-matches that are true non-matched.

$$\textbf{Specificity} = \frac{\textbf{True Negative (TN)}}{(\textbf{False Positive(FP)} + \textbf{True Negative(TN)})} \quad \text{Equation 11}$$

$$\textbf{Negative Predictive Value} = \frac{\textbf{True Negative}}{(\textbf{True Negative(TN)} + \textbf{False Negative(FN)})} \quad \text{Equation 7}$$

The F-score, also known as an accuracy measure or F-measure, is the harmonic mean between precision and recall. Therefore, F-score can be interpreted as a weighted average of recall and precision, where F-score reaches its best value at 1 and worst score at 0. The formula for F-score is:

$$\mathbf{F} = \mathbf{2} * \left(\frac{\textbf{precision} * \textbf{recall}}{\textbf{precision} + \textbf{recall}}\right) \quad \text{Equation 13}$$

CHAPTER 3: METHODS

3.1 Introduction

The preceding chapter presented a review of the literature relevant to this study. This chapter focuses on the methods used in the study. Specifically, the chapter describes the study population and the deterministic and probabilistic methods used to perform the record linkage of data from the HDSS and BCHC records. This research involved a number of steps in order to achieve the objective of comparing the performance of SVM and nearest neighbor techniques to that of the EM record linkage technique based on the Fellegi-Sunter framework for record linkage of HDSS and BCHC. First, record pairs were blocked using year of birth, gender, village and age. Second, comparisons were made between the records in a block and the agreement status of the fields was determined. Thirdly, the record pairs in each block were split into 50% training and 50% test sets that were used to implement EM, SVM, KNN methods to see how well they all perform. The splitting algorithm guarantees that the training and test data sets are non-overlapping and the class proportions remain the same in the three block. Finally, the performance of each method was evaluated using sensitivity, PPV, NPV and the F-score measure.

3.2 Data Sources

The data used in this study come from two sources. The first datasets come from the Agincourt HDSS, which has conducted longitudinal population surveillance in the Agincourt area in rural north-east of South Africa since 1992. At the start of the population surveillance in 1992, the Agincourt HDSS compromised of 57 600 people in 8 900 households in 20 villages(4). By 2006, the Agincourt HDSS population had increased to about 70 000 people in 11 700 households. By mid-2011, the population under surveillance comprised some 90 000 people residing in 16 000 households, in 27 villages(4).

The second dataset consists of records of about 10 000 individuals that sought Human Immunodeficiency Virus/Acquired Immune Deficiency Syndrome (HIV/AIDS) care and treatment at any time between 1 January 2007 and 31 December 2013 from BCHC. The BCHC, set up as a public–private partnership, was opened in the Agincourt study site in 2007 (45). The BCHC is a one-stop clinic for basic health care, tuberculosis

(TB), and HIV/AIDS care. It provides vital antiretroviral treatment so that people living with HIV/AIDS can manage their illness and reclaim a better quality of life (45).

The variables common to both data sources that were used for record linkage are: the 13-digit South African National Identity Number (a unique 13-digit number assigned to South African citizens), first name, other first name, surname, other surname, sex, day of birth, month of birth, year of birth, household member first name, household member surname, telephone number, and village.

3.3 Data Pre-processing

Real-world data is generally incomplete, inconsistent, and noisy. For instance, demographic information often contains blank spaces, typographical and data entry errors, misspellings, and outliers. Sometimes an individual's information changes over time, with changes in an individual's life circumstances, such as marriage or moving, thus leading to changes in one's name or address (34). Therefore, data cleaning is very important before attempting any record linkage process. Examples of data cleaning activities include filling in missing values, correcting inconsistencies and identifying outliers, and smoothing out noisy data.

Data standardisation is the first step to ensure that data is able to be shared across different data sources. Ideally, such standardisation should be performed during data entry(34). If for some reason this is not possible, a comprehensive back-end process is necessary to eliminate any inconsistencies in the data. In this study, the first step that was done prior to data linkage was pre-processing of data to guarantee that all variables conformed to the same format. Three datasets were extracted using a comma delimited text file from a Microsoft SQL server database. The data standardisation process involved changing and removing unwanted characters and tokens in all the datasets. This was applied to string information such as names, surnames, and addresses as well as numerical information. We removed all the special characters such as dots, slashes, and minus characters. The main reason for performing the aforementioned procedures was to make the information more easily comparable and also readable in the Python programming language that was used to carry out the record linkage experiments. This process reduced the number of typographical variations in all the strings. All values that appeared not to be a number (NAN) were replaced with blanks. This allowed the text file to be easily read in Python and manipulated using the pandas package.

3.4 Creating Training and Test Datasets

The creation of training and test datasets was done as follows. First, a dataset of true matched record pairs (Dataset $\boldsymbol{A}$) was created by linking the individuals in the dataset from BCHC to the individuals in the Agincourt HDSS dataset using phone number and the 13-digit South African Identity Number. Second, another dataset of record pairs (Dataset $\boldsymbol{B}$) was created using the records from BCHC in Dataset $\boldsymbol{A}$ and records from Agincourt HDSS. This dataset consisted of both matched and unmatched record pairs. Finally, a third dataset (Dataset $\boldsymbol{C}$) was created by adding the match status to each record pair in Dataset $\boldsymbol{B}$. This dataset contained 12,540,123 record pairs.

Field comparison of all the possible record pairs from the records in the HDSS and BCHC datasets would be computationally very intensive. Therefore, in creating Dataset $\boldsymbol{B}$ and subsequently Dataset $\boldsymbol{C}$, blocking techniques were used to reduce computational time, by restricting the comparison space to blocks or pockets of record pairs, where one or more variables match exactly. Since blocking may decrease the sensitivity of record linkage if blocking variables are measured with error, three blocking schemes that were not mutually exclusive were employed. The first scheme (block 1) used exact match on year of birth and gender. The second scheme (block 2) used exact match on gender, village, and <10 years age difference. Gender was chosen as a good blocking variable to be used in combination with others as it was believed that not many records were likely to be wrongly classified or to have missing values in this field. The last block (block 3) used exact match on first letters of the first name and surname and <10 years age difference. The datasets generated by these three blocking schemes are described in Table 4. Creation of all the datasets was conducted using SQL scripts.

Table 4**:** Three blocks that are not mutually exclusive

Block	**# from Bhubezi Clinic**	**# from HDSS**	**# of Record Pairs**	**Matched record pairs**	**Non-matched record pairs**
1	4 377	154 165	3 997 123	3 965	3 993 158
2	4 174	148 252	6 119 937	3 795	6 116 142
3	4 369	136 438	2 423 063	2 891	2 420 172

In creating the datasets of record pairs, fields were considered to agree using the criteria given in Table 5.

Table 5: Field agreement criteria

HDSS Variables	BCHC Variables	Agreement criteria
First name	First name	JW-Score >=0.8
Surname	Surname	JW-Score >=0.8
Village	Village	Exact
Household member first name	Household member first name	JW-Score >=0.8
Household member surname	Household member surname	JW-Score >=0.8
Day of birth	Day of birth	Absolute Difference = 0
Month of birth	Month of birth	Absolute Difference = 0
Year of birth	Year of birth	Absolute Difference = 0
Sex	Sex	Exact
National ID Number	National ID Number	Exact
Telephone Number	Telephone Number	exact

The dataset from the three blocks was exported from SQL Server database to a comma delimited data file. The comma delimited data file was processed in Python to partition each block into 50% training set and 50% test set, in order to train the SVM and KNN algorithms (see processing script in appendix III). The variables included in the data file were: (name, surname, householdmember, sex, day of birth, month of birth and year of birth, and village). Name was created by finding the maximum score of all scores on names, surname was created by finding the maximum all scores on surnames, and household member was created by finding the maximum all scores on household members.

3.5 Experimental Setup

Three probabilistic record linkage experiments were implemented in this study. The first experiment was based on the Fellegi-Sunter model and used the EM algorithm to estimate the *m* and *u* probabilities and the proportion of true matches among all possible record pair combinations. The second experiment used SVM and the third one used K-nearest neighbor classification. The EM, SVM and K-nearest neighbor classification techniques were all implemented in Python version 3.6.

3.5.1 Expectation Maximization Method

For the experiment using the EM algorithm, the first step was to determine which set to use between the training set and the test set described above. In order to be able to compare the supervised and unsupervised learning techniques, the 50% test set that was used to predict the scores for SVM and KNN is the same sample that was used for the experiment using the EM algorithm. Second, we recoded the record scores for name, surname, householdmember, using a threshold of a JW score ≥ 0.8 to depict a match and a JW score < 0.8 to depict a non-match. In the case of a match a score of 1 was assigned and a score of 0 was assigned in the case of a non-match. The rest of the variables, day of birth, month of birth, year difference, gender and village were already labelled as '1' or '0' - corresponding to match or non-match as indicated. Third step was to set the link/match prevalence, (π) at 0.1. The match prevalence is the probability of a randomly selected pair of records to be a true link. The next step was to maximise the m_k , u_k and p parameters, followed by ensuring that both the identifier agreement rates m_k and u_k can only take values [0, 1]. This step is required to define the dictionary with the starting values for both identifier rates marginal probability mass function (see implementation example in appendix IV).

3.5.2 Support Vector Machine and K-Nearest Neighbour Models

The theory and algorithm of SVM and KNN have been described in various literature, and several sets of SVM and KNN scripts are freely available on the Internet. The version of SVM scripts used in this study are presented in Appendix II to IX. The implementations of both KNN and SVM algorithms in the python language are based on descriptions of the algorithms presented in the sections 2.8 and 2.9 above. The most widely used library for implementing machine-learning algorithms in Python is scikit-learn. The class used for SVM classification in scikit-learn, with the following parameters svm.SVC ,C=1.0, kernel='linear', C is the regularization parameter of the error term; linear is the kernel type used.The value of the box constraint C for the soft margin was set to 1 for both linear.

The “IsMatch” variable in the datasets was used to determine the number of matches and non-matches in both the training and test sets using the slicing method. All the matched and non-matched records were joined using concatenation method. All this was done prior to building the SVM classifier. The SVM classifier was built using the training set only. The training process continued until the model achieved a desired level of accuracy on the training data.

Following fitting, the model was then used to predict new values using the test set. The risk of overfitting is was determined and less in both SVM and KNN. KNN adds the complication of having to choose the best value for K(46), while the challenge for SVM’s complication is to choose a good kernel function. For example, if K is too small and the dataset is large, we run the risk of overfitting. Therefore, K needs to be large enough to avoid overfitting, but small enough to avoid oversimplifying the distribution.

Once the prediction was done, the confusion matrix was created for each block using the prediction and the “IsMatch” variable from the test dataset (see appendix VIII- X). This was repeated for KNN classifier. For KNN classifier, 3 nearest neighbors were considered, therefore K= 3(47).

3.5.3 Evaluating the Record Linkage Performance

The performace evaluation of SVM and KNN algorithms was done using contingency tables also known as confusion matrices. The process of obtaining the performance indicators for SVM and KNN, a confusion matrix table were created using the true labels and prediction data. This table was used to describe the performance of the classification models on a set of test data for which the true values are known. In evaluation the perfomace of the SVM and KNN algorithms, the following steps were followed: (i) the 50% training set was used to train and fit the model; (ii) the test set was used to predict labels of new observations; and (iii) true labels and prediction labels were used to generate the confusion matrix. The confusion matrix was used to determine TP, FP, TN, and FN values. To obtain the performance indicators for EM was slightly different as we used the test set only. The test set was used, m, u and p parameters were calculated, following we join the configurations designated as matches and non-matches by the EM algorithm to the

labelled pairwise comparison dataset. We also obtained performance statistics such as sensitivity, PPV, specificity, NPV and the well-known F-score (see appendix VIII- X).

CHAPTER 4: RESULTS

4.1 Introduction

This chapter reports the results of the record linkage experiments executed in this study using the datasets of record from Agincourt HDSS and BCHC.

4.2 Description of the Data

Table 6 presents the percentage of record pairs with agreement on each of the identifiers used in the record linkage experiments. The identifier agreement percentages are also shown in Figures 2, 3, and 4. The percentages are presented by match status and blocking criteria. In block 1, the name achieved the linkage of 98.9%, surname achieved the linkage of 91.7%, household member achieved the identifier agreement percentage of 87.2% and birth day variable achieved the linkage of 92.5%. In block 2 name identifier agreement percentage is 95.6%, surname 90.9% and gender 89.4%. In block 3 the name identifier agreement percentage is 97.9%, surname 97.3%, household member is 98.5% and for gender 89.99%. Village, identifier agreement percentage is 29.30% and 30% retrospectively.

Table 6: Identifier Agreement Percentage by match status and blocking criteria

	Identifier Agreement Percentage					
	Block1		Block 2		Block3	
Variables	% in matches	% in non matches	% in matches	% in non matches	% in matches	% in non matches
Name	98.9	1.8	95.6	1.8	97.9	9.5
Surname	91.7	4.3	90.9	4.9	97.27	15.4
Gender	xx	xx	89.4	3.7	89.9	3.6
Household member	87.2	3.7	xx	xx	98.5	54.7
Birth Day	92.5	8.8	90.8	8.8	91.3	8.8
Birth Month	18	7.4	93.8	4.8	94.6	4.9
Birth Year	xx	xx	xx	xx	85.6	7.3
Village	29.6	1.97	30.4	2.1	30.1	2

xx: not required per blocking scheme.

Among all the eight identifier fields in table 6, name, and surname were the variables that had the highest percentages of agreement in the matched record pairs across all the blocking schemes. Followed by birth day variables that had the highest percentages of agreement in the matched record pairs across all the blocking schemes. Village, birth year, and birth month were variables that had the least percentages of

agreement in the matched record pairs in all three blocks. The identifier agreement plots in Table 6 and Figure 2 for blocks 1, 2 and 3 were generated using the python script in appendix IV, respectively.

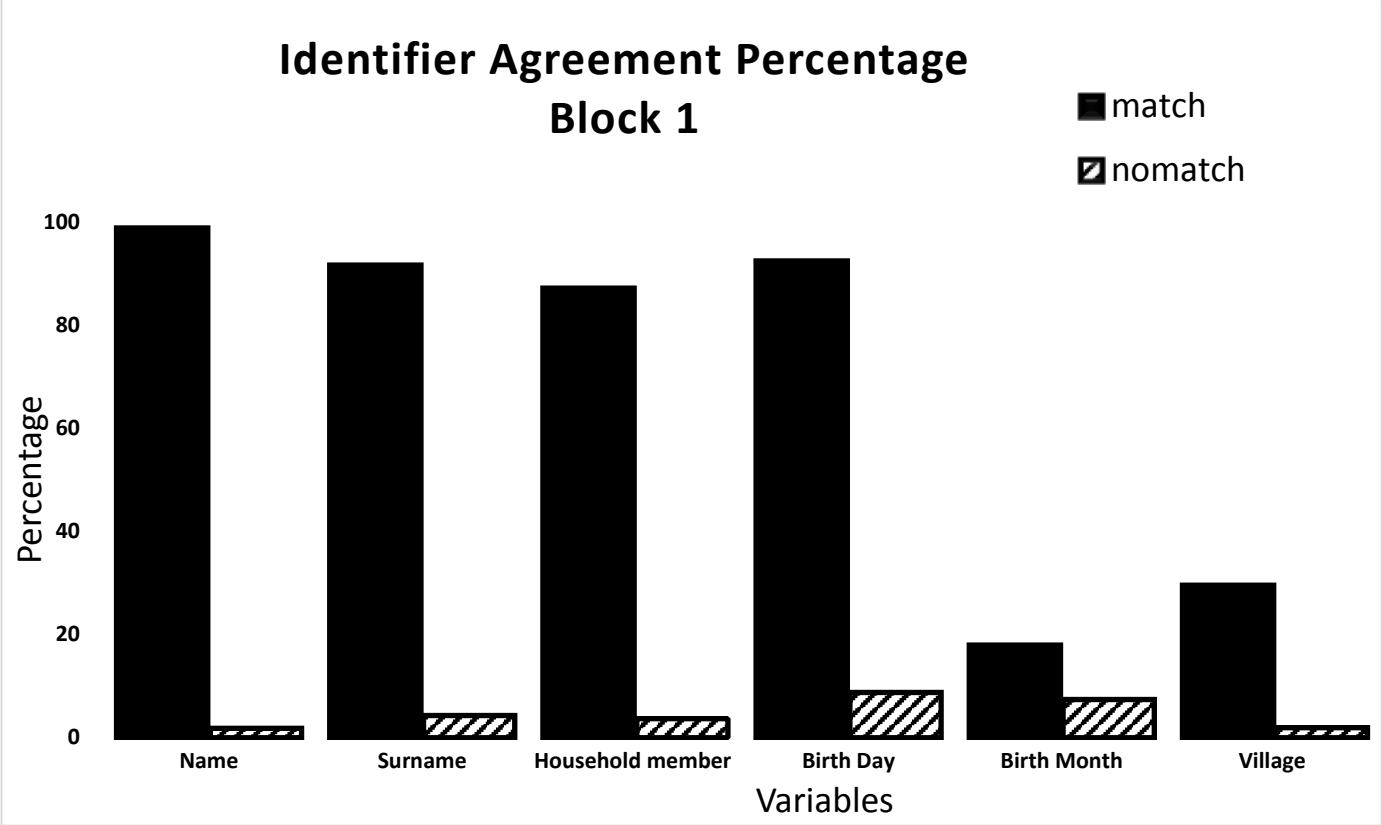


Figure 2: Identifier agreement block 1

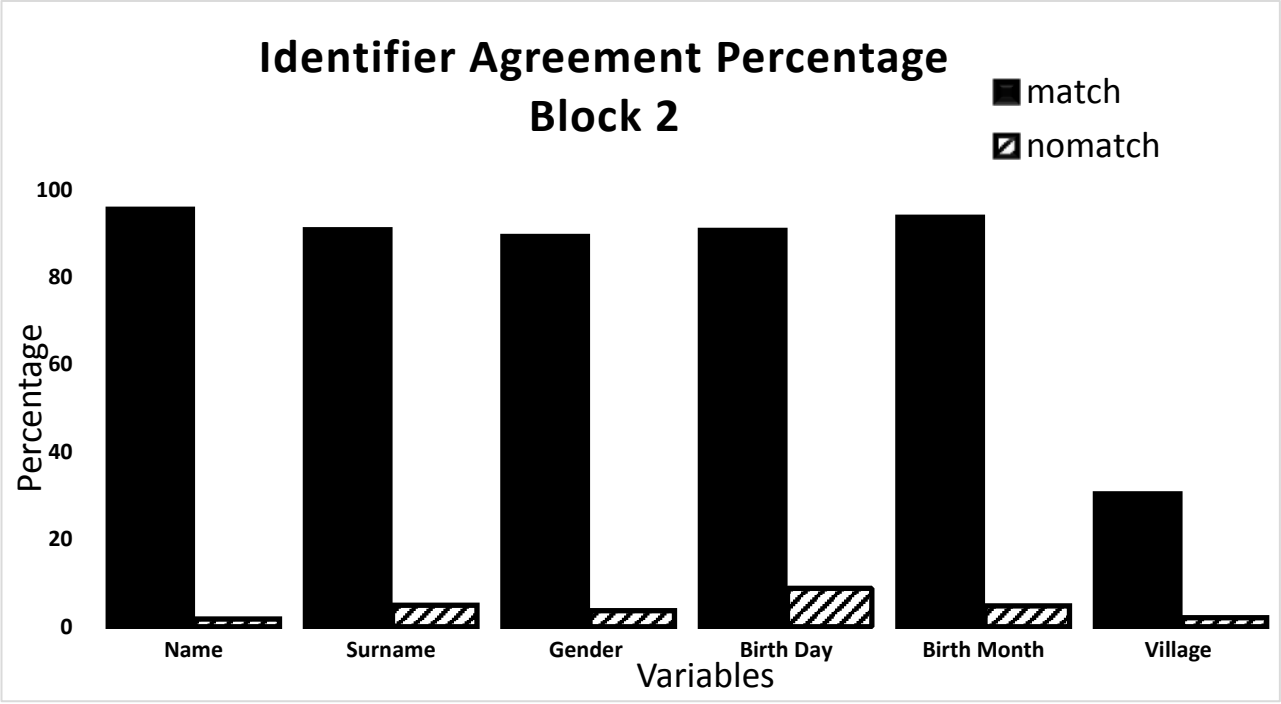


Figure 3: Identifier agreement block 2

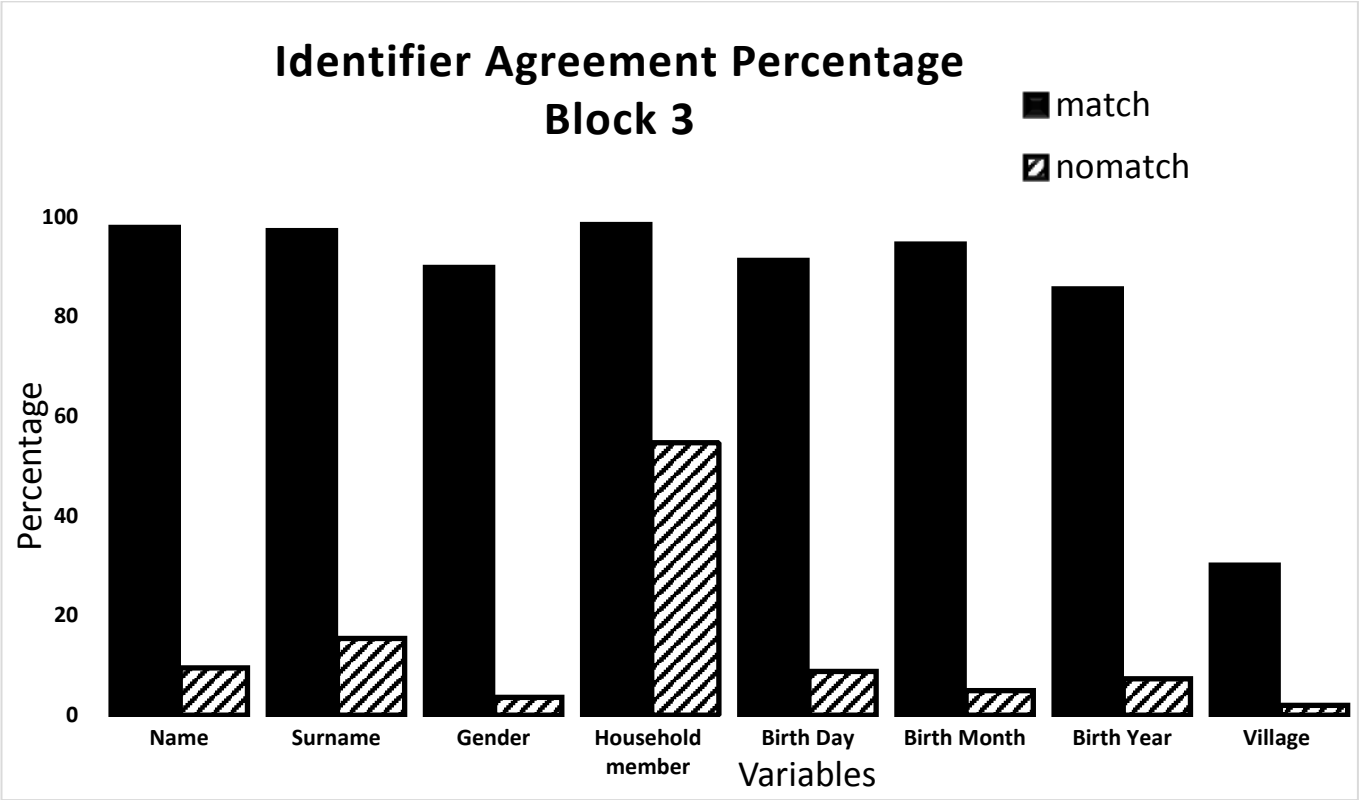


Figure 4: Identifier agreement block 3

4.3 Unsupervised Machine Learning Technique (EM Algorithm) Results

Block 1

A total of 2 016 records were regarded as true matches, where agreement on all the variables is equal to '1'. To obtain the number of false positive matches, the total number of true positive matches was subtracted from the total number of records designated as matches by the algorithm. Therefore, the number of FP in block 1 was 128. A total of 1 996 481 record pairs were true negatives in block1. They had disagreement on all the variables (agreement = '0'). The number of false negatives was obtained by finding the difference between the total number of non-matches and that of true negatives. From this calculation, it was found that 224 record pairs were false negatives in block 1. The calculated values are presented in Table 7 and the script used to generate the table is represented in appendix V.

Block 2

A total of 1 761 records were regarded as true matches, where agreement on all the variables is equal to '1'. To obtain the number of false positive matches, the total number of true positive matches was subtracted

from the total number of records designated as matches by the algorithm. Therefore, the number of FP in block 1 was 105. A total of 3 057 783 record pairs were true negatives in block 2. They had disagreement on all the variables (agreement = '0'). The number of false negatives was obtained by finding the difference between the total number of non-matches and that of true negatives. From this calculation it was found that 320 record pairs were false negatives in block 2. The calculated values are presented in Table 7 and the script used to generate the table is represented in appendix V.

Block 3

A total of 1 457 records were regarded as true matches, where agreement on all the variables is equal to '1'. To obtain the number of false positive matches, the total number of true positive matches was subtracted from the total number of records designated as matches by the algorithm. Therefore, the number of FP in block 1 was 252. A total of 1 209 646 record pairs were true negatives in block 3. They had disagreement on all the variables (agreement = '0'). The number of false negatives was obtained by finding the difference between the total number of non-matches and that of true negatives. From this calculation it was found that 177 record pairs were false negatives in block 3. The calculated values are presented in Table 7 and the script used to generate the table is represented in appendix V.

Table 7: EM Contingency table – Block 1, 2 and 3

	Block 1			**Block 2**			**Block 3**		
	match	no_match	**Total**	match	no_match	**Total**	match	no_match	**Total**
match	2016	128	**2 144**	1 761	105	**1 866**	1 457	252	**1 709**
no_match	224	1 996 194	**1 996 418**	320	3 057 783	**3 058 103**	177	1 209 646	**1 209 823**
Total	2240	1 996 322	**1 998 562**	2 081	3 057 888	**3 059 696**	1 634	1 209 904	**1 211 532**

4.4 Supervised Machine Learning Technique (SVM and KNN) Results

4.4.1 SVM Results

Block 1

The confusion matrix generated 1 948 true matched records, where the agreement score on all the variables was ≥ 0.8. The number of false positive matches was 292. The calculated number of true negative matches was 1 996 175 and of false negative matches was 147. The calculated values are presented in Table 8 and the script used to generate the table is represented in appendix VI.

Block 2

The confusion matrix generated 1 647 true matched records, where the agreement score on all the variables was ≥ 0.8. The number of false positive matches was 434. The calculated number of true negative matches was 3 057 709 and of false negative matches was 179. The calculated values are presented in Table 8 and the script used to generate the table is represented in appendix VI.

	Block 1			Block 2			Block 3		
	match	no_match	**Total**	match	no_match	**Total**	match	no_match	**Total**
match	1 948	292	**2 240**	1 647	434	**2081**	918	110	**1 028**
no_match	147	1 996 175	**1 996 322**	179	3 057 709	**3 057 888**	42	423 432	**423 474**
Total	2 095	1 996 467	**1 998 562**	1 826	3 058 143	**3 059 969**	960	423 542	**424 502**

Block 3

The confusion matrix generated 918 true matched records where the agreement score on all the variables was ≥ 0.8. The number of false positive matches was 110. The calculated number of true negative matches was 423 432 and of false negative matches was 42. The calculated values are presented in Table 8 and the script used to generate the table is represented in appendix VI.

Table 8: SVM Contingency table - Block 1, 2 and 3

4.4.2 KNN Results

Block 1

For block 1, using the KNN method, the confusion matrix generated 1 995 true matched records and 245 false positive matches. The calculated number of true negatives was 1 996 178 and 144 record pairs were false negatives. The calculated values are presented in Table 9 and the script used to generate the table is represented in appendix VI.

Block 2

For block 2, using the KNN method, the confusion matrix generated 1 727 true matched records and 354 false positive matches. The calculated number of true negatives was 3 057 706 and 182 record pairs were false negatives. The calculated values are presented in Table 9 and the script used to generate the table is represented in appendix IV.

Block 3

KNN algorithm found 911 true matched records and 117 false positive matched record pairs, generated using confusion matrix. The calculated number of true negatives was 423 432 and 42 record pairs were false negatives in block 3. The calculated values are presented in Table 9 and the script used to generate the table is represented in appendix VI.

Table 9: KNN Contingency table - Block 1, 2 and 3

	Block 1			**Block 2**			**Block 3**		
	match	no_match	**Total**	match	no_match	**Total**	match	no_match	**Total**
match	1 995	245	2 240	1 727	354	2081	911	117	1 028
no_match	144	1 996 178	1 996 322	182	3 057 706	3 057 888	42	423 432	423 474
Total	2 139	1 996 423	1 998 562	1 909	3 058 060	3 059 969	953	423 549	424 502

4.4.3 Evaluation Results

We used f- score, linkage quality as primary metrics, and sensitivity and PPV as secondary metrics to evaluate the linkage outcomes. A linkage quality measure was calculated using f-score, which was improved on an evaluation methodology by Ferrante and Boyd (48). The results with f-score ≥ 0.90 were considered ‘‘very good’’, those with 0.85 ≤ f-score < 0.90 were relatively ‘‘good’’, those with 0.80 ≤ f-score < 0.85 were ‘‘fair’’, those with 0.50 < f-measure < 0.80 were ‘‘poor’’, and f-score < 0.50 were ‘‘very poor’’. Table 11 presents the summary of f-score performance.

Table 10: F-score Performance(48)

Blocks	**F-scores**		
	SVM	**KNN**	**EM**
1	Very good	Very good	Very good
2	Very good	Very good	Good
3	Very good	Very good	Good

Using EM algorithm, record linkage based on identifiers that are routinely collected in health facilities have sensitivity ranging from 84.62 %, to 90.00% and PPV ranging from 85.25% to 94.37%. Specificity for all 3 blocks was very high, ranging from 99.98% to 99.99%, with NNV ranging from 99.98% to 99.99%. The SVM, sensitivity was ranging from 83.99 %, to 92.98% and PPV ranging from 70.23% to 99.98%. it is evident that block 1, with sensitivity of 92.98% and PPV of 99.98%, produced the best results among all three blocks.

The sensitivity for true matches increased when less information was available for linking the records compared to when full information was available, this is illustrated in table 11. In block 1 only 6 variables, were linked and the sensitivity is very high. Specificity for all 3 blocks was very high, ranging 99.92% to 99.99%, with NNV ranging from 99.98% to 99.99%. When using the KNN algorithm, the sensitivity was ranging from 84.68 %, to 95.19% and PPV ranging from 84.82% to 86.96%. Specificity for all 3 blocks was

very high, ranging 99.96% to 99.99%. The NNV was also very high across, 99.99% across all the 3 blocks.

The F-score results vary by notable small amount across all the linkage techniques, ranging from 87.16% to 99.98%. The f-score for both SVM and KNN is high in all three blocking schemes compared to the EM record linkage technique. The overall linkage quality results are presented in table 11 and on the script in appendix V and VI.

Table 11: Accuracy statistics SVM and KNN record linkage techniques

Sample	**Data size**	**Statistic**	**SVM**	**KNN**	**EM**
Block 1	1 996 467	F-score	0.9998	0.9997	0.9197
		PPV	0.9998	0.8482	0.9403
		Sensitivity	0.9298	0.9000	0.9000
		Specificity	0.9998	0.9999	0.9999
		NNP	0.9999	0.9998	0.9999
Block 2	3 059 969	F-score	0.9998	0.9998	0.8923
		PPV	0.7998	0.8696	0.9437
		Sensitivity	0.9000	0.8468	0.8462
		Specificity	0.9999	0.9999	0.9999
		NNP	0.9999	0.9999	0.9999
Block 3	424 502	F-score	0.9998	0.9997	0.8716
		PPV	0.7023	0.8482	0.8525
		Sensitivity	0.8399	0.9519	0.8917
		Specificity	0.9992	0.9996	0.9998
		NNP	0.9999	0.9999	0.9999

CHAPTER 5: DISCUSSION

5.1 Introduction

This study has conducted empirical comparisons of the probabilistic record linkage technique based on the Fellegi-Sunter framework and supervised machine learning techniques in record linkage of data from an HDSS and health facility in South Africa. The record linkage techniques compared are EM, SVM, and KNN respectively. To the researcher's knowledge, this is the first study to assess record linkage using SVM and KNN techniques in linking HDSS and health care facility data in South Africa.

The main parameters that differentiated performances of the 3 techniques are the combined effect of sensitivity and PPV and/or the f-score. While most previous studies used sensitivity and PPV to evaluate linkage outcomes, we also used the f-score that found the optimal balance between sensitivity and PPV. The f-score used in our study better captured the trade-off between sensitivity and PPV than single metrics(48). Record linkage outcomes are affected by database quality, implementation of linkage and method used for performance validation. We systematically implemented and evaluated supervised and unsupervised record linkage techniques using three blocking scheme. Supervised record linkage techniques demonstrated superior performance to unsupervised record linkage in most of the blocking schemes of our study, which was consistent with the literature review.

Our study findings suggest that record linkage using supervised machine learning techniques achieved good results. However, the unsupervised method using the EM algorithm also performed very good looking at the f-score measure. We also found that name and surname yielded the highest agreement rate in HDSSs/BCHC record pairs. This can be seen in Figure 2 and table 9. This is followed by the household member name and the birth day. The village variable yielded the least agreement rate. This could be due to various reason as presented in chapter 4, table 9. The results suggest that the healthcare facility does not always routinely record characteristics such as village, another household member's first name and surname, National ID number. Hence linkage in these variables maybe not be as high as the name and surname.

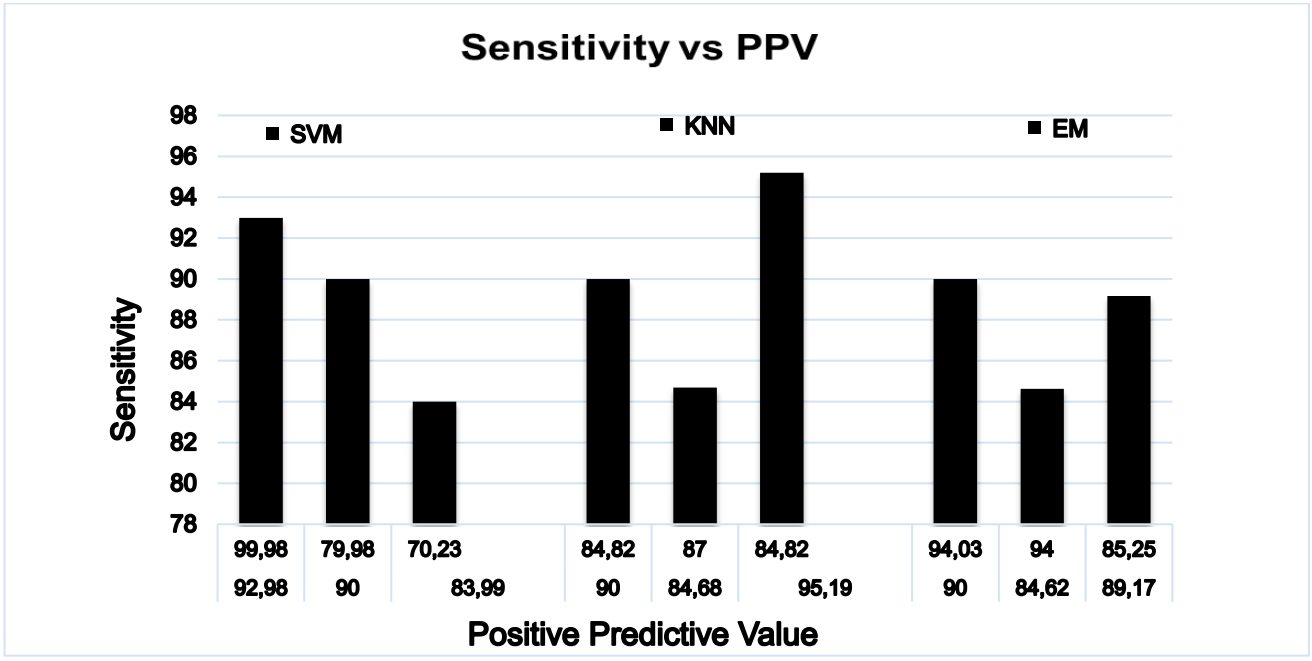


Figure 5: Sensitivity vs Positive Predictive Value

5.1.1 SVM

The classification using SVM, shows the sensitivity of 93% in block 1, 90% in block 2 and 84% in block 3. This shows that SVM has correctly classified true matches in block 1 and 2 compared to block 3 dataset. These results suggest that, you get better classification results when you have the large data size, block 1 and 2 both have the largest data size compared to block 2 and 3. 99.9% of record pairs are classified as true non-matches when using SVM model. Specificity is very high across all the 3 blocks, 99.9%. Sensitivity and specificity are inversely proportional, meaning that as the sensitivity decreases, the specificity increases. In block 1, the PPV score is 99.9 percentage of matched record pairs with true matches are identified and this suggests that SVM model is doing as good to identify true matched record pairs. However, block 2, and 3 PPV scores are at 80% and 70% respectively.

5.1.2 KNN

The classification using KNN, yielded sensitivity of 90% in block 1, 87% in block 2 and 95% in block 3. This shows that KNN correctly classified most of the true matches in all the three dataset. These results suggest that, KNN is a better classifier, and the size of the dataset does not matter. Specificity is very high across all the 3 blocks, 99.9%. Block 1 and 3 have the PPV of 85%, and block 2 with PPV of 87%.

5.1.3 EM

The classification using EM algorithm yielded sensitivity of 90% in block 1, 85% in block 2 and 89% in block 3. This shows that the EM algorithm correctly classified true matches in block 1 better compared to block 2 and 3 datasets. These results suggest that, EM is a better classifier, and the size of the dataset does not matter. Specificity is very high across all the 3 blocks, 99.9%. Block 1 and 2 have the PPV of 94%, and block 3 with PPV of 85%.

5.2 Comparison of the Techniques

Table 11 summarizes the performance of EM, SVM and KNN in all three blocking schemes. In block 1, KNN record linkage showed better performance in sensitivity, both EM and SVM record linkage showed better performance in PPV. When reviewing the results for block 2, KNN and EM record linkage showed better performance in PPV and SVM showed better performance in PPV. However, in block 3 all the record linkage techniques EM, SVM and KNN showed better performance in sensitivity. Based on a review of previous linkage studies of medical databases, there is not a specific suggested cutoff value for sensitivity or PPV. Ideally both parameters need to be optimized, however some studies suggested to sacrifice sensitivity and maintain a high PPV, because FP biases both the risk ratio and risk difference to the null while FN does not affect risk ratio.

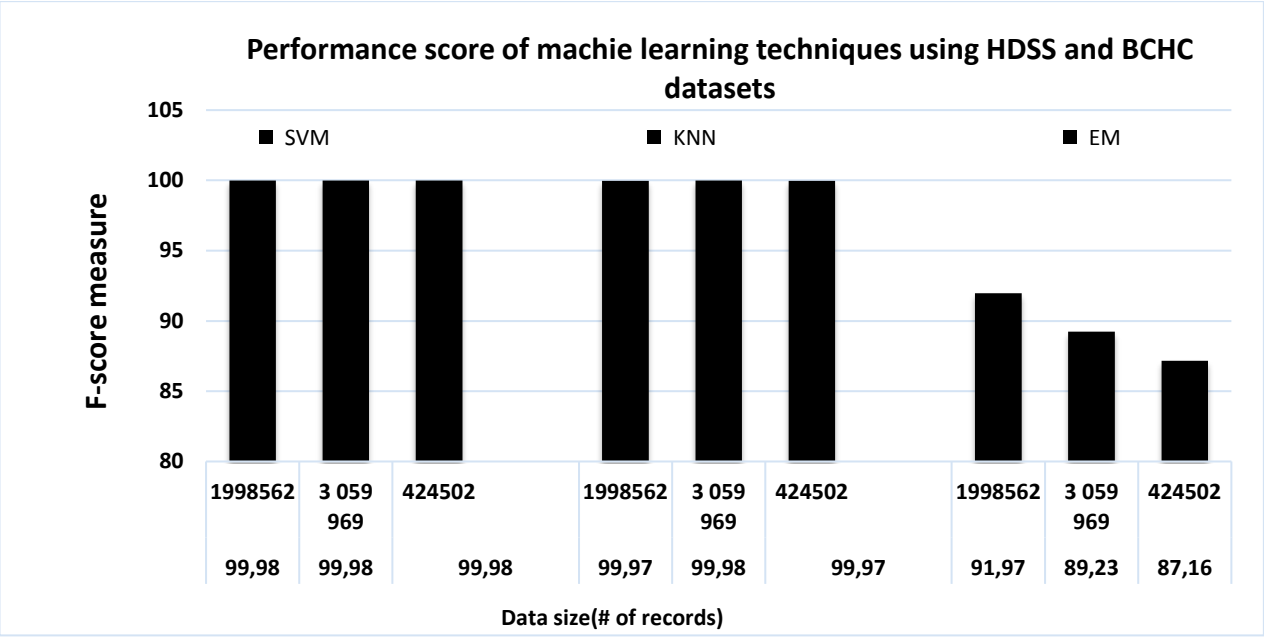


Figure 6: Performance Scores of Supervised and Unsupervised Techniques

Figure 6 shows that the *F-score* values for EM are different across the three blocks. Block 1 is showing 92% performance score, and this is the highest score compared to the other blocks. For SVM, the *F-score* value increases as the data size decreases. Block 2 has the lowest performance when compared to the other two blocks, and it shows that the other methods have approximately the same performance. The *F-score* values from KNN increase as the data size increases. Block 2 has the highest performance when compared to the other two blocks, and this shows that the other blocks have approximately the same performance.

5.2.1 SVM and KNN

KNN and SVM proved to be the best techniques in record linkage. The f-score for both techniques yielded 99.99%. This finding concur with previous research done using supervised machine learning techniques through SVM and KNN.

5.2.2 SVM/KNN and EM

The results show that the EM algorithm matched more records compared to SVM and KNN. However, this does not necessarily mean that the model performed better as it yielded sensitivity ranging from 84% to 90% compared to that from SVM, which ranged from 84% to 93%. These differences in individual metrics were not significant enough to show a difference in the f-score of three linkage techniques when the rate of missing and error was very low or low, regardless of the discriminative power and file size. The results might not be theoretically generalised to other HDSS and heath care facilities or other countries because the quality of administrative data may vary across geographical areas and institutions. Overall, the results show that all 3 models performed very well. Even though EM performed very well, it is notable that the f-score obtained from SVM and KNN were very good for all three blocking schemes compared to those produced by the EM algorithm.

When comparing the performance of the three blocking schemes, different sizes of the datasets was noted. Block 2 had over 6 million records pairs, while block 1 had over 3 million record pairs, with block 3 having more than 2 million record pairs. The first block performed best across all the machine-learning algorithms applied on this study, with F-scores of 99.98, 99.97, and 91.97 for SVM.KNN and EM respectively. The performance of SVM and KNN is very good for all the blocks, although we noted that the EM algorithm performance decreases as the size of the dataset decreases.

5.3 Limitations of the Study

The limitation of the study was the actual size of the database; the SQL query generated over 12 500 000 record pairs. As a result, it was challenging to also run all the different proportions that were generated for each block. The 8GB RAM was not enough to store, import, and run all the different proportions, however all this was achieved by running the experiments on the Agincourt Server. The matching rate was influenced by a number of limitations relating to the amount, accuracy, completeness, and consistency of information available for the linkage. The other concern was setting of thresholds and the difficulty of making final cut-off for EM decisions in computing the performance indicators. The reported precision rates across all the 3 blocks were similar (ranging from 99.98 to 99.99); however, the recall percentages were lower. In order to accurately assess for bias due to errors in linkage, characteristics of unmatched records and a measure of the quality of linkage, such as the rates of false positives and false negatives, need to be routinely measured and reported. Depending on the research question and outcomes under study, potential bias in study results needs to be assessed on a case-by-case basis.

SVMs are complex, as discussed above finding the correct kernel to use is difficult, before we had to go through all the kernels to determine which one will be more suitable. We also tried to plot the maximum margin separating hyperplane within a two-class separable dataset using the linear kernel. This was not feasible as we had six to eight variables, the computing time took more than 24 hours, not sure, if this was due to the memory of the computer or we needed to just do binary classification. Therefore, we could not plot and calculate distance to the decision boundary because the dataset was too large.

KNN, typically can not handle high dimensional data, fitting the training dataset took longer. KNN also demonstrates that the sensitivity and PPV are lower as the dataset increase. It was also noted that KNN consumes large time for training.

EM tends to be much more numerically stable when compared to other methods, while require many iterations, and higher dimensionality can dramatically slow down the E-step. Implementation of this algorithm was very fast, however the performance of this algorithm depends on the size of the dataset. As the dataset decreases so is the performance of the algorithm see figure 6.

CHAPTER 6: CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This final chapter focuses on the conclusion and ethical considerations of the study. It also provides recommendations for future research.

The need to link records has become increasingly necessary in today's world due to the enormous amount of information being collected. There are several studies, which study active learning in the context of record linkage. In this study, three classifiers, namely, EM, SVM and KNN were used for record linkage of HDSS and health facility data. The results also show that EM algorithm can be effective unsupervised learning technique for estimating parameters for record linkage. The use of the sample based EM parameters can result in improved separation of matches and non-matches in record linkage. This method managed to indicate the number of true matches versus the number of true non-matches. It was noted that the EM-based method using 50% test sets yielded high number of true matched record pairs in all 3 blocking schemes compared to SVM and KNN. However, it was also noted that f-scores were comparatively lower compared to the KNN and SVM for all three blocking schemes, 91.92%, 89.23% and 85.25% respectively. However, KNN - based method using 50% training and test sets also yielded the best results when looking at the performance scores for all three blocking schemes, also showing high PPV.

The observation made is that the K-nearest neighbor classifier typically has good predictive accuracy, looking at the number of true matched records, sensitivity, PPV and the accuracy measure, and F-scores for all three blocks. Overall, KNN is a good classifier. SVM classifier's performance is a little less more compared to KNN. However, SVM is also a good classifier.

This study has reviewed, analysed, and compared existing approaches, EM, SVM, and KNN in order to suggest the most effective approach in terms of efficiency. Most importantly, it has shown that both supervised and unsupervised machine learning techniques performed very well. While the results of this study are useful in understanding record linkage using the different techniques, the performance of EM, SVM and KNN techniques was successful.

6.2 Ethical Considerations

The study received ethical approval from the University of the Witwatersrand Human Research Ethics Committee (Clearance number: **M151169**). Privacy and confidentiality of the datasets were done by ensuring that the data was extracted and saved in a secured password-protected computer. The researcher visited Agincourt and was shown the data management and the strict data use policies at Agincourt. Dr Chodziwadziwa Kabudula has experience in electronic record linkage and has published papers regarding record linkage. The data was used only for this study and its related publications at the university and in peer review journals. The information about participants was kept private and confidential and used only for this study.

6.3 Future Work

In the literature search, it was found that in South Africa, there are few articles on record linkage. Agincourt HDSS is more active in linking the census data with data from nearby healthcare centres. However, there is substantial literature in other parts of the world exploring all the different techniques mentioned in this study. The record linkage that has been done so far shows high sensitivity of around 80% to 99%, although there were also studies with lower sensitivities. In future, other datasets such as hospital records linked with laboratory data can be explored and worked with. Documentation at provincial hospitals is limited/insufficient; thus, focus can be placed at introducing all the above record linkage techniques to make sense of the patient's health status. We also plan to evaluate the performance of the record linkage approaches in a scenario where only routinely collected identifiers are used versus the scenario where a name of another household member are used as an additional identifier for linking.

REFERENCES

1. Ariel A. Record linkage in health data: a simulation study.

2. Beauchamp A, Tonkin AM, Kelsall H, Sundararajan V, English DR, Sundaresan L, et al. Validation of de-identified record linkage to ascertain hospital admissions in a cohort study. BMC medical research methodology. 2011;11(1):42.

3. Jutte DP, Roos LL, Brownell MD. Administrative record linkage as a tool for public health research. Annual review of public health. 2011;32:91-108.

4. Kahn K, Collinson MA, Gómez-Olivé FX, Mokoena O, Twine R, Mee P, et al. Profile: Agincourt health and socio-demographic surveillance system. International journal of epidemiology. 2012;41(4):988-1001.

5. Li B, Quan H, Fong A, Lu M. Assessing record linkage between health care and Vital Statistics databases using deterministic methods. BMC health services research. 2006;6(1):48.

6. Goldacre M, Abisgold J, Seagroatt V, Yeates D. Cancer after cholecystectomy: record-linkage cohort study. British journal of cancer. 2005;92(7):1307.

7. Group WoSCPS. Computerised record linkage: compared with traditional patient follow-up methods in clinical trials and illustrated in a prospective epidemiological study. Journal of clinical epidemiology. 1995;12(48):1441-52.

8. Liu S, Wen SW. Development of record linkage of hospital discharge data for the study of neonatal readmission. Chronic Diseases and Injuries in Canada. 1999;20(2):77.

9. Smith ME, Newcombe H. Accuracies of computer versus manual linkages of routine health records. Methods Archive. 1979;18:89-97.

10. Ye W, Lagergren J, Nyren O, Ekbom A. Risk of pancreatic cancer after cholecystectomy: a cohort study in Sweden. Gut. 2001;49(5):678-81.

11. Morin L, Vetrano DL, Rizzuto D, Calderón-Larrañaga A, Fastbom J, Johnell K. Choosing Wisely? Measuring the Burden of Medications in Older Adults near the End of Life: Nationwide, Longitudinal Cohort Study. The American Journal of Medicine. 2017.

12. Kabudula CW, Clark BD, Gómez-Olivé FX, Tollman S, Menken J, Reniers G. The promise of record linkage for assessing the uptake of health services in resource constrained settings: a pilot study from South Africa. BMC medical research methodology. 2014;14(1):71.

13. Kabudula CW, Joubert JD, Tuoane-Nkhasi M, Kahn K, Rao C, Gmez-Oliv FX, et al. Evaluation of record linkage of mortality data between a health and demographic surveillance system and national civil registration system in South Africa. Population Health Metrics. 2014;12(1):23.

14. Elfeky MG, Verykios VS, Elmagarmid AK, Ghanem TM, Huwait AR. Record linkage: a machine learning approach, a toolbox, and a digital government Web service. 2003.

15. Christen P, editor Automatic record linkage using seeded nearest neighbour and support vector machine classification. Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining; 2008: ACM.

16. Palaniappan R, Sundaraj K, Sundaraj S. A comparative study of the svm and k-nn machine learning algorithms for the diagnosis of respiratory pathologies using pulmonary acoustic signals. BMC bioinformatics. 2014;15(1):223.

17. Raikwal J, Saxena K. Performance evaluation of SVM and k-nearest neighbor algorithm over medical data set. International Journal of Computer Applications. 2012;50(14).

18. Jakkula V. Tutorial on support vector machine (svm). School of EECS, Washington State University. 2006;37.

19. Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence. 2011;33(1):117-28.

20. Kotsiantis SB, Zaharakis I, Pintelas P. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering. 2007;160:3-24.

21. Winkler WE, editor Overview of record linkage and current research directions. Bureau of the Census; 2006: Citeseer.

22. Rosman DL. The feasibility of linking hospital and police road crash casualty records without names. Accident Analysis & Prevention. 1996;28(2):271-4.

23. Gu L, Baxter R, Vickers D, Rainsford C. Record Linkage: Current Practice and Future Directions.

24. Dusetzina SB, Tyree S, Meyer A-M, Meyer A, Green L, Carpenter WR. Linking data for health services research: a framework and instructional guide: Agency for Healthcare Research and Quality (US), Rockville (MD); 2014.

25. Pacheco AG, Saraceni V, Tuboi SH, Moulton LH, Chaisson RE, Cavalcante SC, et al. Validation of a hierarchical deterministic record-linkage algorithm using data from 2 different cohorts of human immunodeficiency virus-infected persons and mortality databases in Brazil. American journal of epidemiology. 2008;168(11):1326-32.

26. Fellegi IP. Record linkage and public policy—a dynamic evolution. Record Linkage Techniques. 1997:3-12.

27. Silveira DPd, Artmann E. Accuracy of probabilistic record linkage applied to health databases: systematic review. Revista de saúde pública. 2009;43(5):875-82.

28. Michelson M, Knoblock C. Learning Blocking Schemes for Record Linkage2006.

29. Alenazi SR, Ahmad K. Record Duplication Detection in Database: A Review. International Journal on Advanced Science, Engineering and Information Technology. 2016;6(6):838-45.

30. Jurczyk P, Lu JJ, Xiong L, Cragan JD, Correa A, editors. FRIL: a tool for comparative record linkage. AMIA annual symposium proceedings; 2008: American Medical Informatics Association.

31. Kumar DA, Begum TUS. A comparative study on fingerprint matching algorithms for EVM. Journal of Computer Sciences and Applications. 2013;1(4):55-60.

32. Sariyar M, Borg A, Pommerening K. Active learning strategies for the deduplication of electronic patient data using classification trees. Journal of biomedical informatics. 2012;45(5):893-900.

33. Winkler WE. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. 1990.

34. De Bruin J. Probabilistic record linkage with the Fellegi and Sunter framework: Using probabilistic record linkage to link privacy preserved police and hospital road accident records. 2015.

35. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. Journal of the royal statistical society Series B (methodological). 1977:1-38.

36. Grannis SJ, Overhage JM, Hui S, McDonald CJ, editors. Analysis of a probabilistic record linkage technique without human review. AMIA annual symposium proceedings; 2003: American Medical Informatics Association.

37. Yancey WE. Improving EM algorithm estimates for record linkage parameters.

38. Joachims T. Learning to classify text using support vector machines: Methods, theory and algorithms: Kluwer Academic Publishers; 2002.

39. Zhang X, Lu X, Shi Q, Xu X-q, Hon-chiu EL, Harris LN, et al. Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data. BMC bioinformatics. 2006;7(1):197.

40. Christen P, Churches T. Febrl-freely extensible biomedical record linkage. 2002.

41. Martin B. Instance-based learning: nearest neighbour with generalisation. 1995.

42. Dramé K, Mougin F, Diallo G. Large scale biomedical texts classification: a kNN and an ESA-based approaches. Journal of biomedical semantics. 2016;7(1):40.

43. Kabudula CW, Clark BD, Gómez-Olivé FX, Tollman S, Menken J, Reniers G. The promise of record linkage for assessing the uptake of health services in resource constrained settings: a pilot study from South Africa. BMC Medical Research Methodology. 2014;14.

44. Blakely T, Salmond C. Probabilistic record linkage and a method to calculate the positive predictive value. International journal of epidemiology. 2002;31(6):1246-52.

45. Mee P, Collinson MA, Madhavan S, Kabudula C, Gómez-Olivé FX, Kahn K, et al. Determinants of the risk of dying of HIV/AIDS in a rural South African community over the period of the decentralised roll-out of antiretroviral therapy: a longitudinal study. Global health action. 2014;7(1):24826.

46. Bhatia N. Survey of nearest neighbor techniques. arXiv preprint arXiv:10070085. 2010.

47. Li C, Zhang S, Zhang H, Pang L, Lam K, Hui C, et al. Using the K-nearest neighbor algorithm for the classification of lymph node metastasis in gastric cancer. Computational and mathematical methods in medicine. 2012;2012.

48. Ferrante A, Boyd J. A transparent and transportable methodology for evaluating Data Linkage software. Journal of biomedical informatics. 2012;45(1):165-72.

APPENDICES

Appendix I: Plagiarism Policy



PLAGIARISM DECLARATION TO BE SIGNED BY ALL HIGHER DEGREE STUDENTS

SENATE PLAGIARISM POLICY: APPENDIX ONE

I Vuyokazi Sharon Jezile (Student number: 0418142M) am a student

Registered for the degree of Msc in Epidemiology Research Data Management in the academic year 2018.

I hereby declare the following:

- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- I confirm that the work submitted for assessment for the above degree is my own unaided work except where I have explicitly indicated otherwise.
- I have followed the required conventions in referencing the thoughts and ideas of others.
- I understand that the University of the Witwatersrand may take disciplinary action against me if there is a belief that this is not my own unaided work or that I have failed to acknowledge the source of the ideas or words in my writing.

Signature: _____ Date: 26/03/2018

Appendix II: Ethics Approval

R14/49 Miss Vuyokazi Sharon Jezile

HUMAN RESEARCH ETHICS COMMITTEE (MEDICAL)

CLEARANCE CERTIFICATE NO. M151169

NAME: (Principal Investigator)	Miss Vuyokazi Sharon Jezile
DEPARTMENT:	Public Health Wits Rural Health and Health Transitions Research Unit (Agincourt)
PROJECT TITLE:	An Empirical Comparison of Supervised Machine Learning Techniques in Record Linkage of Data from Health Facilities and Demographic Surveillance System: A Case Study from Rural North East South Africa
DATE CONSIDERED:	27/11/2015
DECISION:	Approved unconditionally
CONDITIONS:	
SUPERVISOR:	Gideon Nimako
APPROVED BY:	Professor A Dhai, Co-Chairperson, HREC (Medical)
DATE OF APPROVAL:	21/2/2015

This clearance certificate is valid for 5 years from date of approval. Extension may be applied for.

DECLARATION OF INVESTIGATORS

To be completed in duplicate and **ONE COPY** returned to the Secretary in Room 10004, 10th floor, Senate House, University.

I/we fully understand the conditions under which I am/we are authorized to carry out the above-mentioned research and I/we undertake to ensure compliance with these conditions. Should any departure be contemplated, from the research protocol as approved, I/we undertake to resubmit the application to the Committee. **I agree to submit a yearly progress report**.

Principal Investigator Signature Date

PLEASE QUOTE THE PROTOCOL NUMBER IN ALL ENQUIRIES

Appendix III: Source Scripts (Python: Processing of the datasets)

Preparation/Processing of Block 1/2/3 datasets:

```
import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib import style

style.use("ggplot")

from sklearn import svm


data = pd.read_csv('BCHC_X_ADSS_With_Match_Status_Block1.txt',sep=',')

df = pd.DataFrame(data)

df['IsMatch'].value_counts()

#check data type and fix replace ','with '.'

for n in ['BName_Name','BName_OtherName1','BName_OtherName2','BOtherName_Name','BOtherName_OtherName1','BOtherName_OtherName2']:

    if df[n].dtypes == np.object:

        df[n]=df[n].str.replace(',','.').apply(pd.to_numeric)


#df['BName_Name'].hist()

#df['BName_Name'].hist(by=df['IsMatch'])

df['name']=df[['BName_Name','BName_OtherName1','BName_OtherName2','BOtherName_Name','BOtherName_OtherName1','BOtherName_OtherName2']].apply( max, axis=1 )


#check data type and fix replace ','with '.'

for n in ['BSurname_Surname','BSurname_OtherSurname1','BSurname_OtherSurname2']:

    if df[n].dtypes == np.object:

        df[n]=df[n].str.replace(',','.').apply(pd.to_numeric)


df['surname']    =df[['BSurname_Surname','BSurname_OtherSurname1','BSurname_OtherSurname2']].apply(    max, axis=1 )


#delete variables not needed for further processing

#del surnames, surnames_df, surname_max
```

```
#check data type and fix replace ','with '.'

For n in ['HHMember11','HHMember12','HHMember13','HHMember14','HHMember15','HHMember16','HHMember17','HHMember18','HHMember19','HHMember110']:

    if df[n].dtypes == np.object:

        df[n]=df[n].str.replace(',','.').apply(pd.to_numeric)

df['hhmember'] =df[['HHMember11','HHMember12','HHMember13','HHMember14','HHMember15','HHMember16','HHMember17','HHMember18','HHMember19','HHMember110']].apply( max, axis=1 )


X=df[['Block','Id','PatientNoNumeric','IsMatch','name','surname','hhmember','Gender', 'BDay', 'BMonth', 'BYearDiff', 'Village']]

X.to_csv('BCHC_X_ADSS_With_Match_Status_Block1_Ready.txt', index=False, header=True, sep=',')
```

Appendix IV: Source Scripts (Identifier Agreement)

```
data = pd.read_csv('BCHC_X_ADSS_With_Match_Status_Block1_Ready.txt',index_col=False, sep=',')

df = pd.DataFrame(data)

df['IsMatch'].value_counts()

#check data type and fix replace ','with '.'


X=df[['name','surname','hhmember','BDay', 'BMonth','Village']]

y_train = df[['IsMatch']]

y = y_train

clf=svm.SVC(kernel='linear',C=1.0)

clf = svm.SVC(gamma=0.001, C=100.)

clf.fit(X,y.values.ravel())


################UDENTIFIER AGREEMENT##################################################

X=df[['Block','Id','PatientNoNumeric','IsMatch','name','surname','hhmember','Gender', 'BDay', 'BMonth', 'BYearDiff', 'Village']]


matches = X[X['IsMatch']==1]

n_matches=len(matches)

#3267

matches[['name','surname','hhmember', 'BDay', 'BMonth', 'Village']].describe()
```

```
threshhold = 0.8
percentFactor= 100.0/n_matches
match_totals = [ len([i for i in matches["name"] if i >=threshhold])*percentFactor,
                 len([i for i in matches["surname"] if i >= threshhold])*percentFactor,
                 len([i for i in matches["BDay"] if i == 0])*percentFactor,
                 len([i for i in matches["BMonth"] if i == 0])*percentFactor,
                 len([i for i in matches["Village"] if i == 1])*percentFactor,
                 len([i for i in matches["hhmember"] if i >= threshhold])*percentFactor
               ]
match_totals = pd.DataFrame(match_totals)
match_totals.index = ['name','surname','BDay','BMonth','hhmember','Village']
match_totals.columns=['matches']

#non matches
nonmatches = X[X['IsMatch']==0]
nonmatches[['name','surname','BDay','BMonth','hhmember','Village']].describe()
n_nonmatches=len(nonmatches)
threshhold = 0.8
percentFactor= 100.0/n_nonmatches
nomatch_totals = [ len([i for i in nonmatches["name"] if i >=threshhold])*percentFactor,
                 len([i for i in nonmatches["surname"] if i >= threshhold])*percentFactor,
                 len([i for i in nonmatches["BDay"] if i == 0])*percentFactor,
                 len([i for i in nonmatches["BMonth"] if i == 0])*percentFactor,
                 len([i for i in nonmatches["Village"] if i == 1])*percentFactor,
                 len([i for i in nonmatches["hhmember"] if i >= threshhold])*percentFactor
               ]

nomatch_totals = pd.DataFrame(nomatch_totals)
nomatch_totals.index = ['name','surname','hhmember','BDay','BMonth', 'Village']
nomatch_totals.columns=['nonmatches']

percentage_agree = pd.concat([match_totals,nomatch_totals], axis=1, join='outer')

########################IDENTIFIER AGREEMENT BY % PLOT ################################
```

```
N=6

ind = np.arange(N)
width = 0.35       # the width of the bars

fig, ax = plt.subplots()

match_totals_list = pd.to_numeric(match_totals['matches'], errors='coerce')
rects1 = ax.bar(ind, match_totals_list, width, color='r')
#rects1 = ax.bar(ind, match_totals, width, color='r')

nomatch_total_list = pd.to_numeric(nomatch_totals['nonmatches'], errors='coerce')
rects2 = ax.bar(ind + width, nomatch_total_list, width, color='b')

#rects2 = ax.bar(ind + width, nomatch_totals, width, color='b')

# add some text for labels, title and axes ticks
ax.set_ylabel('Percentage Agreement')
ax.set_title('Identifier Agreement')
ax.set_xticks(ind + width / 2)

ax.set_xticklabels(('name', 'surname','hhmember','BDay','BMonth','Village'), rotation=35)

ax.legend((rects1[0], rects2[0]), ('match', 'nomatch'))

from matplotlib.ticker import FuncFormatter
formatter = FuncFormatter(lambda y, pos: "%d%%" % (y))
ax.yaxis.set_major_formatter(formatter)
plt.show()
########################IDENTIFIER AGREEMENT BY % PLOT################################
```

Appendix V: Source Scripts (EM algorithm)

```
# -*- coding: utf-8 -*-

"""

Created on Wed Sep 27 19:45:40 2017


@author: Vuyokazi Jezile

"""# ---------------------------------------

# EXAMPLE BINARY ASSUMPTION

# ---------------------------------------

# Comparison vectors with label 1 for agreement and 0 for disagreement


data =  pd.read_csv('BCHC_X_ADSS_With_Match_Status_Block3_TestSet.txt',index_col=False, sep=',')

#len=

################################################### Partitioning block3----------TEST SET----- 50% ##############################################


#    lines = [line for line in source]

df = pd.DataFrame(data)


# Add unique identifier to each row

df['identifier'] = df.index


#X=df[['Block','Id','PatientNoNumeric','IsMatch','name','surname','hhmember','Gender',  'BDay',  'BMonth',  'BYearDiff', 'Village']]

X=df[['name','surname','hhmember','Gender','BDay', 'BMonth', 'BYearDiff','Village']]


#check

df.iloc[10,:]


X.iloc[10,:]


## record scores for name, surname, hhmember

## 1 if above threshold; 0 otherwise
```

```
# Initiate comparison matrix and calculated weight matrix with zeros
com_matrix = pd.DataFrame(np.zeros(X.shape), columns=X.columns)
columns = X.columns.tolist()

## record scores for name, surname, hhmember
## 1 if above threshold; 0 otherwise
# Compare and fill the result into the comparison matrix
for col in ['name','surname','hhmember']:
    com_matrix[col] = (X[col] >= 0.8)

for col in ['BDay','BMonth']:
    com_matrix[col] = (X[col] == 0)

com_matrix['Village'] = (X['Village'] == 1)

com_matrix.head()

df[columns].iloc[26,:]

com_matrix.iloc[26,:]

m_start = {'name': {0: 0.1, 1: 0.9},
           'surname': {0: 0.1, 1: 0.9},
           'hhmember': {0: 0.1, 1: 0.9},
           'Gender' : {0: 0.1, 1: 0.9},
           'BDay'   : {0: 0.1, 1: 0.9},
           'BMonth' : {0: 0.1, 1: 0.9},
           'BYearDiff': {0: 0.1, 1: 0.9},
           'Village' : {0: 0.1, 1: 0.9}}



u_start = {'name': {0: 0.9, 1: 0.1},
           'surname': {0: 0.9, 1: 0.1},
```

```
        'hhmember': {0: 0.9, 1: 0.1},
        'Gender' : {0: 0.9, 1: 0.1},
        'BDay'   : {0: 0.9, 1: 0.1},
        'BMonth' : {0: 0.9, 1: 0.1},
        'BYearDiff': {0: 0.9, 1: 0.1},
        'Village' : {0: 0.9, 1: 0.1}}

# set the match prevalence
p_start = 0.1
# Start an estimation with 150 iterations
#est = estimation_new . ECMEstimate (y , m_start , u_start , p_start,max_iter=150)
est = estimation_new . ECMEstimate (com_matrix , m_start , u_start , p_start)
est.estimate()

# Print the summary
est_sum = est.summary( )

# Print the parameters m, u and p
print (est . m)
print (est . u)
print (est . p)

# ----------------------
Z = est_sum



matches =Z[Z['lambda']>=0.1]
XX= matches[['count']]
Total_matches =XX.sum()

nonmatches =Z[Z['lambda'] < 0.1]
XY= nonmatches[['count']]
Total_nomatches=XY.sum()
```

```
"""
To compute the performance indicators (Sensitivity, Specificity, PPV and NPV):
You need to join the configurations designated as matches and nonmatches by the EM algorithm to the
labelled pairwise comparison dataset
"""
#merge with com_matrix data
com_matrix_with_ident = com_matrix.copy()
com_matrix_with_ident['identifier']=df['identifier']

#join the matches configurations to the labelled (with match status) comparison dataset
match_result = pd.merge(matches, com_matrix_with_ident, how='inner', on=['name', 'surname','hhmember','Gender','BDay', 'BMonth','BYearDiff', 'Village'])
match_result_identifiers=match_result['identifier']
df_match =  pd.merge(df, pd.DataFrame(match_result_identifiers), how='inner', on=['identifier'])

M=len(df_match)
TP=df_match['IsMatch'].value_counts()[1]
FP=df_match['IsMatch'].value_counts()[0]

#join the nonmatches configurations to the labelled (with match status) comparison dataset
nonmatch_result = pd.merge(nonmatches, com_matrix_with_ident, how='inner', on=['name', 'surname','hhmember','Gender','BDay', 'BMonth','BYearDiff', 'Village'])
nonmatch_result_identifiers=nonmatch_result['identifier']
df_nonmatch =  pd.merge(df, pd.DataFrame(nonmatch_result_identifiers), how='inner', on=['identifier'])

N=len(df_nonmatch)
TN=df_nonmatch['IsMatch'].value_counts()[0]
FN=df_nonmatch['IsMatch'].value_counts()[1]

#recall
Sensitivity= TP/(TP+FN)*100
print(Sensitivity)

#precision
PPV = TP/(TP+FP)*100
```

```
print(PPV)

Specificity= TN/(FP+TN)*100
print(Specificity)

Negative_Predictive_Value=TN/(TN+FN)*100
print(Negative_Predictive_Value)

Fscore= 2*((Sensitivity*PPV)/(Sensitivity+PPV))
print(Fscore)

linkage_metrics=pd.DataFrame([Sensitivity,Specificity,PPV,Negative_Predictive_Value])

linkage_metrics.rename(
  columns={
    0 : 'value'
  },
  inplace=True
)
linkage_metrics['metric'] =['Sensitivity','Specificity','PPV','Negative Predictive Value']
```

Appendix VI: Source Scripts (SVM and KNN Contingency tables)

```
 # -*- coding: utf-8 -*-
"""
Created on Tue Jul  6 15:16:57 2017

@author: Vuyokazi Jezile
"""
#Contingency Table and Evaluation Calculations Block 1
********** LOAD AND FILTER TRAINING DATA **********

data = pd.read_csv('BCHC_X_ADSS_With_Match_Status_Block1_Ready.txt',index_col=False, sep=',')
```

```
##len=3997123

####################Partitioning block1----------TEST SET----- 50% ######################################


#    lines = [line for line in source]

df = pd.DataFrame(data)

df['IsMatch'].value_counts()


X=df[['Block','Id','PatientNoNumeric','IsMatch','name','surname','hhmember','Gender',  'BDay',  'BMonth',  'BYearDiff',
'Village']]


###matches

matches = X[X['IsMatch']==1]

n_matches=len(matches)

matches[['name','surname','hhmember','Gender', 'BDay', 'BMonth', 'BYearDiff', 'Village']].describe()


####SLICING #########

matches[:2239]

ismatch = matches[:2239]


nonmatches = X[X['IsMatch']==0]

nonmatches[['name','surname','BDay','BMonth','hhmember','Village']].describe()

n_nonmatches=len(nonmatches)

nonmatches[:1996322]

nomatch = nonmatches[:1996322]

Final_Tables = pd.concat([ismatch,nomatch])


df1 = pd.DataFrame(Final_Tables)

df1['IsMatch'].value_counts()

X1=df1[['name','surname','hhmember','BDay', 'BMonth','Village']]

#build svm classifier

y_train = df1[['IsMatch']]

y1 = y_train

clf=svm.SVC(kernel='linear',C=1.0)

#clf = svm.SVC(gamma=0.001, C=100.)
```

```
clf.fit(X1,y1.values.ravel())


#could not convert string to float: 'Village'
fscore = clf.score(X1,y1)
################################ DISTANNCE TO HYPERPLANE ########################
#use the model with all variables/features
import numpy as np
#calculate distance to decision boundery
#y = w *X + b ; distance = y / w_norm
ydf = clf.decision_function(X1)
w_norm = np.linalg.norm(clf.coef_)
dist = ydf / w_norm
plt.axis([0,1, 0, 1])
#plot histogram of distances
fig3, ax3 = plt.subplots()
plt.hist(dist, bins=100)
plt.xlabel("Distance", size=14)
plt.ylabel("Count", size=14)
plt.title('Distance from Decision Boundary', size=16)
########################Partitioning block1----------TEST SET----- ###################################
matches = X[X['IsMatch']==1]
n_matches=len(matches)
matches[['name','surname','hhmember','Gender', 'BDay', 'BMonth', 'BYearDiff', 'Village']].describe()
####SLICING #########
matches[2239:]
ismatch = matches[2239:]


nonmatches = X[X['IsMatch']==0]
nonmatches[['name','surname','BDay','BMonth','hhmember','Village']]. describe ()
n_nonmatches=len(nonmatches)


nonmatches[1996322:]
nomatch = nonmatches[1996322:]
Final_Tablest = pd.concat([ismatch,nomatch])
```

```
df2 = pd.DataFrame(Final_Tablest)
df2['IsMatch'].value_counts()
#check data type and fix replace ','with '.'
#n_matches=len(matches)
#matches [['name','surname','hhmember','Gender', 'BDay', 'BMonth', 'BYearDiff', 'Village']].describe()

Xt=df2[['name','surname','hhmember', 'BDay', 'BMonth','Village']]

### prediction using test dataset
predicted = clf.predict(Xt)
y_test = df2[['IsMatch']]
confusion_matrix(y_test, predicted)
cm =confusion_matrix(y_test, predicted)
print(cm)
# Show confusion matrix in a separate window
plt.matshow(cm, cmap=plt.cm.Blues)
plt.title('Confusion Matrix')

################################################################################
#set data labels and tick marks
target_names = ["no_match", "match"]
thresh = cm.max() / 2.
import itertools
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
        horizontalalignment="center",
        color="white" if cm[i, j] > thresh else "black")
plt.colorbar()
class_names = target_names
import numpy as np
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks, class_names)
```

```
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
#############################################################################################
TP = 1948
FN = 147
FP = 292
TN = 1996175

Sensitivity= TP/(TP+FN)*100
print(Sensitivity)

Positive_Predictive_Value = TP/(TP+FP)*100
print(Positive_Predictive_Value)

Specificity= TN/(FP+TN)*100
print(Specificity)

Negative_Predictive_Value=TN/(TN+FN)*100
print(Negative_Predictive_Value)

########################################KNN ######################################
from sklearn.neighbors import KNeighborsClassifier

# instantiate learning model (k = 3)
KNN = KNeighborsClassifier(n_neighbors=3)

# fitting the model
KNN.fit(X1,y1.values.ravel())

# predict the response
pred = KNN.predict(Xt)

# evaluate accuracy
```

```
#accuracy_score(y_test, pred)
fscore2 = KNN.score(X1,y1)
confusion_matrix(y_test, pred)

##################################################KNN#################################
cmK =confusion_matrix(y_test, pred)
print(cmK)
# Show confusion matrix in a separate window
plt.matshow(cmK, cmap=plt.cm.Blues)
plt.title('Confusion Matrix')

#############################################################################################
#set data labels and tick marks
target_names = ["no_match", "match"]
thresh = cmK.max() / 2.
import itertools
for i, j in itertools.product(range(cmK.shape[0]), range(cmK.shape[1])):
        plt.text(j, i, cmK[i, j],
        horizontalalignment="center",
        color="white" if cmK[i, j] > thresh else "black")
plt.colorbar()
class_names = target_names
import numpy as np
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks, class_names)

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

##############################KNN ################################################
TP = 2016
FN = 224
```

```
FP = 144

TN = 1996178

Sensitivity= TP/(TP+FN)*100

print(Sensitivity)

Positive_Predictive_Value = TP/(TP+FP)*100

print(Positive_Predictive_Value)

Specificity= TN/(FP+TN)*100

print(Specificity)

Negative_Predictive_Value=TN/(TN+FN)*100

print(Negative_Predictive_Value)
```