AN INVESTIGATION OF MULTI-LABEL CLASSIFICATION TECHNIQUES FOR PREDICTING HIV DRUG RESISTANCE IN RESOURCE-LIMITED SETTINGS

by

Pascal Brandt

Submitted in fulfilment of the academic requirements
for the degree of Master of Science in the
School of Mathematics, Statistics
and Computer Science

University of KwaZulu-Natal
Durban, South Africa
February 2014

As the candidate's supervisor I have approved this dissertation for submission.

Signed: _____ Name: _____ Date: _____

Abstract

South Africa has one of the highest HIV infection rates in the world with more than 5.6 million infected people and consequently has the largest antiretroviral treatment program with more than 1.5 million people on treatment. The development of drug resistance is a major factor impeding the efficacy of antiretroviral treatment. While genotype resistance testing (GRT) is the standard method to determine resistance, access to these tests is limited in resource-limited settings. This research investigates the efficacy of multi-label machine learning techniques at predicting HIV drug resistance from routine treatment and laboratory data. Six techniques, namely, binary relevance, HOMER, MLkNN, predictive clustering trees (PCT), RAkEL and ensemble of classifier chains (ECC) have been tested and evaluated on data from medical records of patients enrolled in an HIV treatment failure clinic in rural KwaZulu-Natal in South Africa. The performance is measured using five scalar evaluation measures and receiver operating characteristic (*ROC*) curves. The techniques were found to provide useful predictive information in most cases. The PCT and ECC techniques perform best and have true positive prediction rates of 97% and 98% respectively for specific drugs. The ECC method also achieved an AUC value of 0.83, which is comparable to the current state of the art. All models have been validated using 10 fold cross validation and show increased performance when additional data is added. In order to make use of these techniques in the field, a tool is presented that may, with small modifications, be integrated into public HIV treatment programs in South Africa and could assist clinicians to identify patients with a high probability of drug resistance.

Declaration 1 – Plagiarism

I, Pascal Brandt, declare that:

(i) The research reported in this dissertation, except where otherwise indicated, is my original research.

(ii) This dissertation has not been submitted for any degree or examination at any other university.

(iii) This dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

(iv) This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written but the general information attributed to them has been referenced.

b) where their exact words have been used, their writing has been placed inside quotation marks and referenced.

(v) This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Candidate: Pascal Brandt

Signature: _____

Declaration 2 – Publications

P. Brandt, D. Moodley, A. W. Pillay, C. J. Seebregts, and T. de Oliveira, "An Investigation of Classification Algorithms for Predicting HIV Drug Resistance Without Genotype Resistance Testing," in *Foundations of Health Information Engineering and Systems*, Vol. 8315 of *Lecture Notes in Computer Science*, J. Gibbons and W. MacCaull, eds., (Spinger Berlin Heidelberg, 2014), pp. 236–253.

Candidate: Pascal Brandt

Signature: _____

Acknowledgments

First, I would like to thank my wife Jodi, whose love and support made this work possible.

To all my supervisors and mentors, thank you for your guidance and your insights. Deshen, thank you for your consistently excellent advice and helping me learn the lessons that I needed to learn. Thank you Anban for keeping me on track and making sure I regularly reflected on what I was doing.

Thank you Chris for all the opportunities you have given me and for your invaluable advice on a wide range of topics. Thank you Tulio for including me in your research team and providing me with access to your work and your enthusiasm.

I would also like to thank the Rockefeller Foundation and International Development Research Centre for the bursary that enabled me to conduct this research (grant numbers 2010 THS 347 and 106452-001 respectively).

The study cohort and drug resistance analysis of the data was approved by the Biomedical Research Ethics Committee of the University of KwaZulu-Natal (BF052/010) and the Provincial Health Research Committee of the KwaZulu-Natal Department of Health (HRKM176/10). All patients received informed consents and the data was anonymised before storage in the research databases.

Preface

The research described in this dissertation was carried out in the School of Mathematics, Statistics and Computer Science at the University of KwaZulu-Natal from July 2011 to January 2014 under the supervision of Dr. Deshendran Moodley, Mr. Anban Pillay, Dr. Christopher Seebregts and Prof. Tulio de Oliveira.

Contents

Abstract ii

Declarations iii

Acknowledgments v

Preface vi

List of Figures x

List of Tables xi

Nomenclature xii

1 Introduction 1
- 1.1 Aims and Objectives . . . 3
- 1.2 Tools and Approach . . . 4
- 1.3 Contributions . . . 5
- 1.4 Structure of Dissertation . . . 6

2 Literature Review 7
- 2.1 Multi-Label Classification . . . 7
  - 2.1.1 Problem Transformation Methods . . . 8
  - 2.1.2 Algorithm Adaptation Methods . . . 12
  - 2.1.3 Ensemble Methods . . . 14
  - 2.1.4 Summary . . . 15
- 2.2 Evaluation Measures . . . 16
  - 2.2.1 Performance Evaluation Measures . . . 17
  - 2.2.2 Predictive Model Validation . . . 23
  - 2.2.3 Dataset Description Measures . . . 24
- 2.3 Applications . . . 25
  - 2.3.1 Classification Objective . . . 26
  - 2.3.2 Training Dataset . . . 27
  - 2.3.3 Evaluation Measures . . . 30
- 2.4 Summary . . . 30

3 Methods **32**
3.1 Multi-Label Classification Techniques . . . 32
3.1.1 Problem Transformation Methods . . . 33
3.1.2 Algorithm Adaptation Methods . . . 39
3.1.3 Ensemble Methods . . . 41
3.2 Evaluation . . . 44
3.2.1 Confusion Matrix . . . 44
3.2.2 Receiver Operating Characteristic . . . 45
3.2.3 Higher Order Measures . . . 45
3.3 Validation Techniques . . . 46
3.3.1 Cross Validation . . . 46
3.3.2 Stratification . . . 47
3.3.3 Dataset Validation . . . 47
3.4 Summary . . . 48

4 Experimental Design and Implementation **49**
4.1 Data . . . 49
4.1.1 Clinical Dataset . . . 49
4.1.2 Training Dataset Development . . . 52
4.1.3 Technique Selection . . . 53
4.1.4 Simplifying Assumptions . . . 54
4.1.5 Summary . . . 55
4.2 Processing Pipeline . . . 55
4.2.1 Dataset Construction . . . 55
4.2.2 Experiment Loop . . . 59
4.2.3 Summary . . . 63
4.3 Software Tools . . . 63
4.3.1 Existing Applications . . . 63
4.3.2 CuRE . . . 65
4.3.3 Stratifier . . . 70
4.3.4 Custom Scripts . . . 72
4.4 Model Parameterisation and Implementation . . . 75
4.4.1 Binary Relevance (BR) . . . 75
4.4.2 HOMER . . . 76
4.4.3 MLkNN . . . 77
4.4.4 Predictive Clustering Trees (PCT) . . . 77
4.4.5 RAkEL . . . 78
4.4.6 Ensemble of Classifier Chains (ECC) . . . 78
4.5 Summary . . . 78

5 Results **80**
5.1 Dataset Analysis . . . 80
5.1.1 Dataset 1 . . . 81
5.1.2 Dataset 2 . . . 83
5.1.3 Dataset Analysis Summary . . . 84

5.2 Scalar Evaluation Measures . . . 86
5.2.1 Dataset 1 . . . 86
5.2.2 Dataset 2 . . . 92
5.2.3 Comparison . . . 97
5.2.4 Scalar Evaluation Measure Summary . . . 103
5.3 ROC Analysis . . . 106
5.3.1 Dataset 1 . . . 107
5.3.2 Dataset 2 . . . 111
5.3.3 Comparison . . . 115
5.3.4 ROC Analysis Summary . . . 119
5.4 Results Summary . . . 120

6 Discussion **122**
6.1 Dataset . . . 122
6.2 Experimental Results . . . 126
6.3 Limitations . . . 133
6.4 Potential Application . . . 135

7 Conclusion and Future Work **137**
7.1 Future Work . . . 138

References **142**

List of Figures

2.1 Multi-label classification methods [21] . . . 9
2.2 Example transformation by the *label power set* problem transformation method . . . 10
2.3 Example transformation by the *binary relevance* problem transformation method . . . 10
2.4 Single label confusion matrix . . . 17

3.1 Binary relevance classifier . . . 34
3.2 A simple hierarchy constructed by HOMER for a hypothetical problem with 8 labels . . . 38
3.3 Mechanism of operation of a classifier chain (CC) [36] . . . 43

4.1 Multi-label classification methods used in this study (leaf nodes) . . . 54
4.2 Dataset construction phase of the processing pipeline . . . 56
4.3 Cross validation phase of the experiment loop . . . 59
4.4 Results processing phase of the experiment loop . . . 60
4.5 Processing pipeline . . . 64
4.6 Architecture of the custom developed *CuRE* tool . . . 66

5.1 *ROC* curves for dataset 1 for efavirens, didanosine, emtricitabine and delavirdine . . . 108
5.2 *ROC* curves for dataset 1 for stavudine, nevirapine, tenofovir and etravirine . . . 109
5.3 *ROC* curves for dataset 1 for abacavir, lamivudine and zidovudine . . . 110
5.4 *ROC* curves for dataset 2 for efavirens, didanosine, emtricitabine and delavirdine . . . 112
5.5 *ROC* curves for dataset 2 for stavudine, nevirapine, tenofovir and etravirine . . . 113
5.6 *ROC* curves for dataset 2 for abacavir, lamivudine and zidovudine . . . 114
5.7 *ROC* curves for both datasets for efavirens, didanosine, emtricitabine and delavirdine . . . 116
5.8 *ROC* curves for both datasets for stavudine, nevirapine, tenofovir and etravirine . . . 117
5.9 *ROC* curves for both datasets for abacavir, lamivudine and zidovudine . . . 118

List of Tables

2.1 Single label confusion matrix-based evaluation measures [58–60] 19
2.2 Multi-label example-based evaluation measures [58–60] 20

4.1 Summary of features used to construct the predictive models 51

5.1 A summary of feature completeness for dataset 1 82
5.2 Resistance and susceptibility counts for each drug for dataset 1 82
5.3 Number of patients resistant to multiple drugs for dataset 1 83
5.4 A summary of feature completeness for dataset 2 84
5.5 Resistance and susceptibility counts for each drug for dataset 2 85
5.6 Number of patients resistant to multiple drugs for dataset 2 85
5.7 Scalar evaluation measures for the problem transformation methods for dataset 1 87
5.8 Scalar evaluation measures for the algorithm adaptation methods for dataset 1 89
5.9 Scalar evaluation measures for the ensemble methods for dataset 1 91
5.10 Scalar evaluation measures for the problem transformation methods for dataset 2 93
5.11 Scalar evaluation measures for the algorithm adaptation methods for dataset 2 95
5.12 Scalar evaluation measures for the ensemble methods for dataset 2 96
5.13 Problem transformation methods evaluation measure differences 98
5.14 Algorithm adaptation methods evaluation measure differences 101
5.15 Ensemble methods evaluation measure differences 102

Nomenclature

AIDS	Acquired immune deficiency syndrome
ARV	Antiretroviral drug
AUC	Area under the *ROC* curve
CD4 count	A measure of the number of T-helper cells in the blood and an indicator of immune system health
CuRE	The custom developed tool to extract data from RegaDB and build the feature vector that represents each patient in the dataset
$\mathscr{D}$	A multi-label training dataset comprised of tuples of the form $(\mathbf{x}, Y)$
DP	Discriminant power
drug resistance	Resistance to a drug implies that the drug will not be effective if taken by the patient
$\mathscr{D}_T$	A multi-label testing dataset comprised of tuples of the form $(\mathbf{y}, Z)$
$\mathscr{E}_D$	The Euclidian distance function
example	A single data record
feature	A single observed or calculated data element associated with an example
FPR	False positive rate
GRT	Genotype resistance test

HAART	Highly active antiretroviral therapy
HIV	Human immunodeficiency virus
λ_i	An individual label
L	The set of labels in a classification problem
label	The category or class to which an example belongs
m	The number of training examples in a training dataset
MCC	The Matthews correlation coefficient
n	The total number of distinct labels in a classification problem
p	The number of features in the feature vector
RegaDB	The clinical information management system used to capture the data used in this study
ROC	Receiver operating characteristic
s	The number of unseen test examples in a testing dataset
τ	Threshold used to determine class membership when a prediction is a probability rather than an absolute value
TCE	Treatment change episode
testing	The process of using a predictive model to generate predictions for previously unseen data
TNR	True negative rate
TPR	True positive rate
training	The process of constructing a predictive model using labelled data
viral load	A measure of how many copies of the HI virus are present in the cells of a patient
$\mathbf{x}$	A feature vector used for training. Also $\mathbf{x}_i$ when referring to an individual member of a training dataset
$\boldsymbol{y}$	An unseen example. Also $\boldsymbol{y}_i$ when referring to an individual member of a test dataset
Y	The set of labels associated with a training example. Also Y_i when referring to the set of labels associated with an individual member of a training dataset
Z	The set of labels predicted by a classifier for an unseen example. Also Z_i when referring to the predictions made for an individual member of a test dataset

Chapter 1

Introduction

Since the identification of the human immunodeficiency virus (HIV) over 20 years ago, HIV/AIDS has become a global epidemic. There are an estimated 36 million people currently living with HIV and 20 million people have died because of it. The region of the world most affected by the epidemic is sub-Saharan Africa, with an estimated 25.3 million infections, representing 69% of the global HIV burden [1]. In 2011, 92% of pregnant women living with HIV resided in sub-Saharan Africa[1] and an estimated 270 000 people died of acquired immune deficiency syndrome (AIDS), which is caused by HIV[2]. South Africa has one of the highest HIV infection rates in the world with more than 5.6 million infected people. Consequently, South Africa has the largest antiretroviral treatment program in the world with more than 1.5 million people on treatment [2].

HIV attacks important cells in the human immune system and if left untreated, infected patients become more susceptible to opportunistic infections. The recommended treatment for HIV/AIDS is a combination of three antiretroviral (ARV) drugs from two or more different drug groups and is known as highly active antiretroviral therapy (HAART). The different drug groups inhibit the virus in different ways and multiple

[1] http://www.unaids.org/en/media/unaids/contentassets/documents/epidemiology/2012/gr2012/2012_FS_regional_ssa_en.pdf

[2] http://www.unaids.org/en/regionscountries/countries/southafrica/

drugs are used to ensure that the viral load (number of copies of the virus circulating in the patient) is adequately suppressed.

HIV treatment failure is deemed to be occurring when the antiretroviral drugs are no longer keeping the viral load under control [3]. The use of HAART has proven to be effective in controlling the progression of HIV and prolonging the patient's survival, but the efficacy of treatment is compromised by the development of resistance to the antiretroviral drugs. The virus has an extremely high replication rate and the process of replication is error prone. This means that as the virus replicates, small mutations occur. If these mutations emerge in the viral proteins that are targeted by the antiretroviral drugs, the drugs may no longer inhibit viral replication [4], resulting in increased viral loads. This process, known as *selective pressure*, is how drug resistance evolves. The most common method currently used to identify drug resistance (and to which drugs a patient is resistant) is the *genotype resistance test* (GRT) [5]. Once a patient's resistance profile has been determined, a drug resistance algorithm (such as Rega [6] or Stanford [5]) is used to interpret the results so that a new treatment regimen, consisting of drugs to which the patient is susceptible[3], can be designed.

In resource-limited settings like South Africa, GRT for all patients is not possible due to resource limitations such as lack of testing laboratories, skilled staff who are able to perform genotype resistance tests and funding for reagents needed for testing. While some studies conclude that including GRT in treatment guidelines may be cost neutral [7], the current South African HIV treatment guidelines do not include GRT for every patient. It is therefore necessary to seek alternative low cost methods to determine drug resistance. In order to optimise the use of GRT, the test should only be performed on patients that have a high risk of resistance. A mechanism is therefore needed to identify and triage patients based on available information so that the best use can be made of this scarce resource.

Machine learning techniques, such as *classification*, have the potential to be used to predict HIV drug resistance without the need for expensive equipment or specialist care providers. Classification is the task of using statistical models to identify which category a new observation belongs to. These models are

[3]If a patient is susceptible to a drug it means that the drug will successfully aid in reducing the viral load.

trained using data for which the category is known, and the elements that make up each unit of training data are referred to as *features*. Traditional classification categorises an unseen feature vector into a single category, while *multi-label classification* techniques are able to associate multiple category labels with new observations.

Classification techniques have been widely used to investigate problems related to HIV. In most cases the predictive models have been used to predict therapy outcome and the training datasets have included GRT data. In the South African context GRT information will not be available, so predictive models must be designed that use only the available clinical data. Models trained on this type of data have shown encouraging results in recent studies [8].

Since HIV in South Africa is treated using a limited number of drugs, and each patient may develop resistance to one of more of these drugs, using multi-label classification to predict drug resistance is appropriate. The set of drugs corresponds with the categories and the medical records of patients that have received GRT are used as the training data. Multi-label classification models trained in this way could be used to predict which sets of drugs a previously unseen patient is resistant to, without the need for GRT.

Studies have shown that HIV decision support systems can be effective and useful to clinicians [9, 10]. The predictive models examined in this study could therefore be used to help care providers decide whether or not to refer a patient for GRT, or select an optimal therapy if the regimen is to be changed. Additionally, without significant modification, it should be possible to integrate the models produced into the HIV management system to act as a decisions support and passive alerting service. This would help identify high risk patients without the need for them to actively seek care.

1.1 Aims and Objectives

The aim of this study is to investigate the efficacy of using multi-label classification techniques trained using routine clinical data to predict the probability of HIV drug resistance.

The objectives of this research are:

- Analysis and construction of a feature vector containing all relevant data elements.
- Selection and application of appropriate multi-label classification techniques and testing the models on a representative real-world clinical dataset.
- Evaluation of the performance of the models using suitable evaluation measures.
- Design and implementation of a robust experimental environment.

1.2 Tools and Approach

The FIRST[4] study, together with the Southern African Treatment and Research Network (SATuRN[5]), have established a successful HIV Treatment Failure Clinic (HIV-TFC) at the Africa Centre for Health and Population Studies[6]. The HIV-TFC consists of dedicated HIV specialists with the ability to perform GRT and a data management system for clinical, laboratory, pharmacy and specialist diagnostic information. Patients failing treatment are referred to the HIV-TFC, where their patient record is collated and presented to the specialists to assist them with decision making regarding future treatment, with the possibility of requesting a genotype resistance test.

The HIV-TFC patient data is used as the example real-world data in this study. This data, used for the clinical management of patients, is stored in a clinical information system called RegaDB [11]. The RegaDB system will be examined, domain experts from the HIV-TFC will be consulted, and the literature will be reviewed in order to determine the most complete and relevant set of features for training the predictive models.

The literature on multi-label classification will be used to identify the most appropriate and representative set of techniques to use. Existing implementations of the multi-label training and testing algorithms will

[4] Free State Initiative on Resistance, Surveillance and Treatment

[5] http://www.bioafrica.net/saturn/

[6] http://www.africacentre.ac.za/

be used where possible, such as those contained in the MEKA[7] and Mulan [12] extensions to the Weka [13] machine learning toolkit. The models will be validated using widely accepting techniques such as cross validation.

Performance of the models will be evaluated using evaluation measures that have been shown to be good indicators of classification success. Accepted *confusion matrix*-based scalar evaluation measures will be used in addition to graphical evaluation measures such as *receiver operating characteristic* (*ROC*) curves. Suitable statistical analysis tools such as R [14] will be used to generate and analyse the evaluation measures.

An experimental environment will be established that will allow for the systematic testing and evaluation of the multi-label classification techniques. This environment will enable all experiments to be easily reproducible in order to reduce the risk of human error. All custom development of scripts and stand-alone applications will be done in a way to reduce the barrier to implementation in the field.

1.3 Contributions

The primary contribution of this work is the application and evaluation of seven multi-label classification techniques trained without GRT data. Each technique has been trained, tested and validated using 10 fold cross validation on two distinct real-world datasets. The predictive performance of the models has been evaluated using five different scalar evaluation measures, as well as receiver operating characteristic (*ROC*) curve analysis.

In addition, a harmonised feature vector used for training and testing the predictive models has been developed. This was done through extensive examination of the literature and consultation with numerous domain experts. The complete feature vector consists of 62 data elements and is constructed using a custom tool that integrates with the clinical management tool used for HIV treatment in South Africa.

[7] http://meka.sourceforge.net/

An experimental framework has been developed using design decisions taken specifically in order to reduce the barrier to implementation in the field. The potential impact of the experimental results have been discussed, recommendations have been made, and further avenues of investigation have been suggested.

1.4 Structure of Dissertation

A review of related work is presented in chapter 2. The multi-label techniques, evaluation and validation measures are discussed in chapter 3. Chapter 4 describes the design and implementation of the experimental environment. Results are presented in chapter 5 and discussed in depth in chapter 6. The dissertation concludes with chapter 7, which summarises the key findings and discusses avenues for future work.

Chapter 2

Literature Review

The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience [15]. This experience is usually in the form of data collected through observation. When models are built using annotated data, a process known as *training*, it is referred to as *supervised learning*. These models are used to *annotate*, *label*, *classify* or *categorise* new, previously unseen data. This task is known as *classification*.

The data used to train classification models are known as *training examples*. Each training example is made up of a set of data elements, called *features*, and the label. The features are the data that describe the example, while the label is the name of category that the example is in. When examples have more than just a single label associated with them, the classification task is known as *multi-label classification*, which is the focus of this research.

2.1 Multi-Label Classification

A large amount of research has been done in the area of *single label* classification [16–18], where algorithms associate training examples with a single label λ_i from a set of disjoint labels $L = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ [19]. If

there are only two labels ($n = 2$), the problem is referred to as *binary* classification and if there are more than two labels ($n > 2$), it is referred to as *multiclass* classification [20, 21].

In *multi-label* (sometimes referred to as *multi-topic* or *multidimensional*) classification problems, training examples are associated with a set of labels Y such that $Y \subseteq L$. In multi-label classification, labels are not assumed to be mutually exclusive and multiple labels may be associated with a single training example. Simply put, each training example can be a member of more than one class [21].

Historically, multi-label classification has primarily been applied to text categorisation and medical diagnosis [20, 22]. For example, a news article about the South African movie *District 9* could be categorized into the *movies*, *news* and *south africa* categories. Similarly, a patient may be suffering from both *arthritis* and *dementia* at the same time. More recently, the use of multi-label classification has become increasingly popular for a variety of problem types [19], such as image and video annotation [23, 24], genomics [25, 26] and music categorisation [27, 28].

Various methods exist for multi-label classification. Early categorisation of multi-label classification methods contained only two categories, namely *problem transformation* methods and *algorithm adaptation* methods [19, 20]. More recently, a third category was introduced, namely multi-label *ensemble* methods [21], as show in figure 2.1. The multi-label classification methods within each of these categories are reviewed in the sections that follow.

2.1.1 Problem Transformation Methods

Problem transformation methods are those methods that transform a multi-label classification problem into one or more single label classification problems. Any one of the many existing single label classification techniques can then be applied to the problem [19–21].

The most rudimentary problem transformation method is the *ignore* method [19], which simply removes all multi-label examples from a dataset. Another simple problem transformation is the *copy* transformation

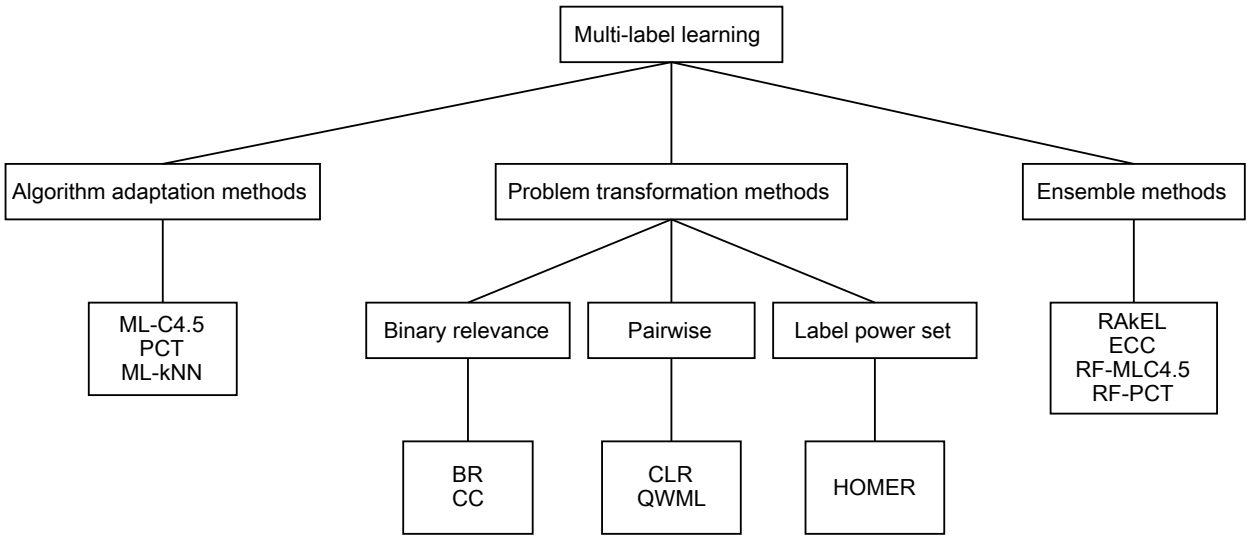


Figure 2.1. Multi-label classification methods [21].

[19]. This transformation replaces each multi-label example $(\boldsymbol{x}_i, Y_i)$ with $|Y_i|$ examples of the form $(\boldsymbol{x}_i, \lambda_j)$, for every $\lambda_j \in Y_i$, where $\boldsymbol{x}_i$ denotes the feature vector and Y_i is the set of labels associated with $\boldsymbol{x}_i$. A variation of the copy transformation, the *copy-weight* transformation, gives each generated example a weight of $\frac{1}{|Y_i|}$. This weight is then taken into consideration by training algorithms that support weighted labels.

Another group of problem transformation methods, known as the *select* group of methods, transform the multi-label dataset by replacing each set of labels, Y_i, with just one of its members $\lambda_j \in Y_i$, resulting in a single label dataset. The *select-max* method chooses λ_j to be the most common label in the dataset. The *select-min* method chooses the least common label in the dataset as λ_j. The final *select* method, *select-random*, chooses λ_j at random [19].

Another relatively simple problem transformation method is the *label power set* method [20]. An example label power set transformation is shown in figure 2.2. This method combines distinct label sets into single labels. The new set of possible labels is then $L_{PS} = P(L)$, where $P(L)$ is the power set of L. Some modifications have been proposed in order for transformation to combat the problem of large label sets. For example, the *pruned problem transformation* method (PPT) [29] only adds label power set elements to the transformed dataset if the corresponding distinct label set actually occurs in the dataset.

Features	Labels
$\mathbf{x_1}$	$\{\lambda_1, \lambda_3, \lambda_4\}$
$\mathbf{x_2}$	$\{\lambda_2, \lambda_3\}$
$\mathbf{x_3}$	$\{\lambda_5\}$
$\mathbf{x_4}$	$\{\lambda_1, \lambda_3, \lambda_4\}$
$\mathbf{x_5}$	$\{\lambda_2, \lambda_3, \lambda_5\}$

(a) Example multi-label dataset.

Features	Labels
$\mathbf{x_1}$	L_1
$\mathbf{x_2}$	L_2
$\mathbf{x_3}$	L_3
$\mathbf{x_4}$	L_1
$\mathbf{x_5}$	L_4

(b) Resulting singe-label dataset.

Figure 2.2. Example transformation by the label power set problem transformation method. Figure (a) shows an example multi-label dataset. Figure (b) shows the resulting single label dataset when the label power set problem transformation is applied. It can be seen that each distinct label set is mapped to exactly one label in the single label dataset.

Features	Labels
$\mathbf{x_1}$	$\{\lambda_1, \lambda_3\}$
$\mathbf{x_2}$	$\{\lambda_2, \lambda_3\}$
$\mathbf{x_3}$	$\{\lambda_2\}$
$\mathbf{x_4}$	$\{\lambda_1, \lambda_2, \lambda_3\}$
$\mathbf{x_5}$	$\{\lambda_3\}$

(a) Example multi-label dataset.

Features	Labels
$\mathbf{x_1}$	$true$
$\mathbf{x_2}$	$false$
$\mathbf{x_3}$	$false$
$\mathbf{x_4}$	$true$
$\mathbf{x_5}$	$false$

(b) Binary relevance dataset for λ_1 (D_{λ_1}).

Features	Labels
$\mathbf{x_1}$	$false$
$\mathbf{x_2}$	$true$
$\mathbf{x_3}$	$true$
$\mathbf{x_4}$	$true$
$\mathbf{x_5}$	$false$

(c) Binary relevance dataset for λ_2 (D_{λ_2}).

Features	Labels
$\mathbf{x_1}$	$true$
$\mathbf{x_2}$	$true$
$\mathbf{x_3}$	$false$
$\mathbf{x_4}$	$true$
$\mathbf{x_5}$	$true$

(d) Binary relevance dataset for λ_3 (D_{λ_3}).

Figure 2.3. Example transformation by the binary relevance problem transformation method. Figure (a) shows an example multi-label dataset. Figures (b) - (d) show the resulting single label datasets generated when the binary relevance problem transformation is applied. It can be seen that each feature vector in the single label datasets is labelled according to the presence or absence of the corresponding label in the multi-label dataset.

The *Hierarchy Of Multi-label classifiERs* (HOMER) method builds on the idea of the label power set problem transformation method [30]. In an attempt to reduce the computation time required for datasets with a large number of labels, HOMER introduces the idea of *meta-labels*, which are small subsets of the label set. Instead of training all base classifiers using the entire dataset, a hierarchy is built using the meta-labels and base classifiers are trained only on labels that form part of the meta-labels in the hierarchy. This method is explained further in section 3.1.1 and illustrated in figure 3.2.

The *binary relevance* problem transformation method learns n binary classifiers, where n is the number of labels in L [20]. The original dataset is transformed into n datasets D_{λ_i} for $i = 1, 2, \ldots, n$. Each dataset contains all the original examples labelled positively if the label set of the original example contained λ_i and negatively otherwise [21]. An example of the binary relevance problem transformation method is shown in figure 2.3. The final predicted label set (Z_i) for an unseen example ($\mathbf{y}_i$) is constructed by consulting each of the n classifiers to check if the label λ_i is predicted to be associated with the example, and if so, adding it to the label set Z_i. See section 3.1.1 for further details.

The final group of problem transformation methods are the *pairwise* methods. The pairwise methods transform the dataset into $\frac{Q \cdot (Q-1)}{2}$ binary label datasets, one for every pair of labels $(\lambda_i, \lambda_j), 1 \leq i < j \leq Q$. Each dataset contains the examples that are associated with exactly one of the two corresponding labels. The label set prediction for an unseen example is constructed by invoking a binary classifier trained on each dataset and a majority voting system is used to determine which labels form part of the final set, Z_i. The *multi-label pairwise perceptron* (MLPP) algorithm [31] uses perceptrons for the binary classification task. An improved voting strategy called *quick weighted voting* [32] has been proposed to increase performance.

By far the most popular problem transformation method of multi-label classification is the binary relevance method [19, 21, 28, 33, 34]. It is very often used as baseline for comparison when evaluating other methods [27, 28, 30, 31, 34–36]. The label power set method is the next most common problem transformation method, and has been used as a basis for the development of new methods such as HOMER [30, 34, 35]. An empirical study comparing the performance of the binary relevance and label power set methods has

shown that the label power set method performs better in the vast majority of cases [20]. HOMER has been shown to perform particularly well on datasets with a large number of labels [30].

2.1.2 Algorithm Adaptation Methods

The algorithm adaptation methods modify or extend existing single label classification algorithms to deal with multi-label data. Although it has been noted that in some cases the adapted algorithms are changed in a way that makes use of one of the problem transformation methods described above [20]. Reviewed below are the algorithms modified for use with multi-label data.

ML-kNN is a modification of the well-known *k nearest neighbours* algorithm (kNN) [37]. There are many other extensions of the kNN algorithm for multi-label datasets [28, 33, 38] that differ only in how the multiple label sets are aggregated. ML-kNN uses the maximum *a posteriori* principle to predict the label set of an unseen example, which is based on *prior* and *posterior* probabilities for the frequency of each label. In general, this method has been shown to perform poorly, despite its popularity [21].

Predictive clustering trees (PCTs) [39] are able to handle several types of structured outputs. PCTs are decision trees that can be considered as a hierarchy of clusters and are constructed using a standard top-down induction algorithm. For multi-label classification the prototype function returns a vector of probabilities that an unseen example is labelled with a given label [21].

Decision tree classifiers are flowchart-like graph structures that break down complex decision making processes into a set of simpler decisions [40]. The *C4.5 decision tree* algorithm splits the dataset at each internal node of the tree using the feature that results in the most *information gain*, which is calculated as the difference in *entropy* [41, 42]. The C4.5 decision tree algorithm has been extended to handle multi-label data for an application in genomics (ML-C4.5) [25]. This was done by allowing multiple labels at the leaves of the tree. The entropy function was also modified to incorporate the relative frequency of label λ_i, $p(\lambda_i)$.

Adaptive boosting, or *AdaBoost*, is method that can be used to increase the performance of many different classification algorithms [43]. The AdaBoost algorithm has been adapted in two different ways to support multi-label data. The *AdaBoost.MH* and *AdaBoost.MR* algorithms [44] are designed to minimise Hamming Loss (see section 2.2.1 for details) and place the correct labels at the top of the ranking respectively. The AdaBoost.MH algorithm has also been combined with an algorithm for creating decision trees [45], with the purpose of generating human-readable models.

A labelled multi-label dataset indicates which labels are associated with each training example, but no information is given about which features are responsible for the presence of each label. Generative classification models attempt to infer exactly this information. A probabilistic generative model has been proposed for multi-label text classification [22], in which labels generate different words using a Bayesian approach. This is accomplished using the *EM* (expectation maximisation) algorithm and the maximum *a posteriori* principle. Similar multi-label models, called *parametric mixture models* (PMMs) have also be proposed [46].

A number of neural networks algorithms have been adapted to support multi-label data. The back-propagation multi-label learner (*BP-MLL*) is a modification of the *back-propagation* algorithm that takes multiple labels into account in the error function [47]. The *multi-class multi-label perceptron* (MMP) is a group of online algorithms that rank labels based on the perceptron algorithm [48]. The multi-label FAM (ML-FAM) algorithm is a multi-label extension of the *fuzzy ARTMAP* method, which is able to incrementally train a neural network using multidimensional inputs [49, 50].

Support vector machines (SVMs) have been adapted to support multiple labels by using a ranking loss minimisation function [51]. Some modifications to SVMs have also been proposed that allow for improved performance when they are used as the base classifier in the binary relevance problem transformation method [52]. One of the improvements adds Q new features to each training example that represent the results of a first round of classification. An unseen example is classified twice, first to calculate values for the Q new fields, then again to generate a final prediction. The other two modifications remove negative training examples based on different criteria.

The most widely used algorithm adaptation methods are *predictive clustering trees* and *ML-kNN* [21, 28, 33, 38, 39]. This is most likely due to the fact that these methods are generic enough to be applied to wide range of problems. In contrast, the C4.5 decision tree method and the generative models described above have been developed for use on a limited set of problems. Predictive clustering trees have been shown to perform well based on a wide range of performance evaluation measures [21].

2.1.3 Ensemble Methods

In contrast to other types of classification techniques that construct only one classifier to solve a problem, *ensemble methods* construct a set of classifiers to solve the same problem. This is also sometimes referred to as *committee-based learning* [53]. Sometimes ensemble members are trained to predict all labels and in other cases members are trained to predict only a subset of labels. In all cases, the final prediction is formed by combining the output of ensemble members in some way.

The *RAndom k-labELsets* (RAkEL) method [35] constructs an ensemble by considering small random subsets of labels called k-labelsets. The algorithm then learns m label power set classifiers, where m is a user parameter that specifies the number of members in the ensemble. Each ensemble member is trained to classify just one k-labelset. For classification of an unseen example, each of the m ensemble members provides a binary prediction for each label λ_i in the k-labelset that it was trained to predict. RAkEL then calculates the average prediction for each label and the label is predicted as positive if the average is above a user configured threshold τ. This method is described in more detail in section 3.1.3.

Ensemble of Classifier Chains (ECC) [34] is an ensemble method that uses *classifier chains* (CC) as a base classifier. Classifier chains work similarly to the binary relevance method in the sense that a classifier is constructed for each label. The difference is that with CC, each successive classifier in the chain appends a new feature to the feature vector of the example currently under consideration, representing the prediction of the respective classifier. This augmented feature vector is then used as the input for the following classifier. An example of this process is illustrated in figure 3.3. The ensemble is established by constructing m

classifier chains, each predicting the labels in a random order, where m is a user parameter. This is to ensure that a particular label ordering does not influence the performance of the method. To classify an unseen example, the same voting scheme is used as for RAkEL above.

Random forest [54] ensembles construct multiple decision trees and use a majority or probability distribution voting method to predict the labels associated with an unseen example. The *random forests of predictive clustering trees* (RF-PCT) [55, 56] method uses PCTs as the trees in the forest while *random forests of ML-C4.5* (RFML-C.45) [21] uses ML-C4.5 trees. Variation of the trees in the forest is created by using bagging (a type of model averaging) [57], as well as using different feature sets at different nodes in the trees.

The most popular multi-label ensemble technique is ECC, which has been shown to perform well across a wide range of performance measures [21, 34]. RAkEL has been shown to consistently perform better than the popular problem transformation methods [35]. Random forests are popular for traditional single label classification problems, but for multi-label classification they have been shown to perform well only on a limited number of datasets [21].

Related Tasks

Multi-label ranking (MLR) is a task similar to multi-classification that outputs an ordering of labels for an unseen example. The order is indicative of the relevance of each label to the particular example [19]. In some multi-label datasets, labels are related in a hierarchical structure. There exist multi-label classification and ranking algorithms that take label hierarchies into account [26]. Label ranking and hierarchies are beyond the scope of this study.

2.1.4 Summary

The multi-label methods in each of the three multi-label classification problem solution groups (problem transformation methods, algorithm adaptation methods and ensemble methods) have been reviewed. The

popular problem transformation methods were identified and their performance compared. Predictive clustering trees and MLkNN were identified as the most popular algorithm adaptation methods. Both methods are applicable across a broad range on problems. Finally, ECC and RAkEL were identified as two popular and well-performing multi-label ensemble methods.

2.2 Evaluation Measures

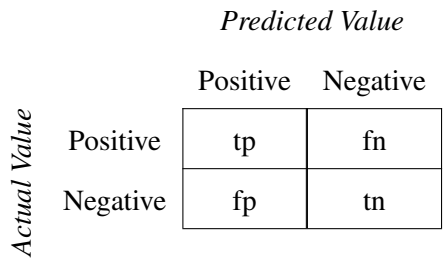
There are many different evaluation measures that can be used to gauge the performance of a classification algorithm [19, 21, 58–60]. Not all evaluation measures are useful in all cases, since some of them can be influenced by the nature of the underlying data. An evaluation measure is said to be *invariant* to a specific dataset characteristic if information provided by the evaluation measure does not change if the dataset characteristic changes [60]. Measures that are not invariant to specific characteristics cannot be reliably used if the dataset does not exhibit those characteristics.

One study identified 24 classification evaluation measures and investigated their invariance to certain characteristics of the underlying dataset [60]. It was demonstrated that if labels are distributed evenly in a dataset, simple accuracy can be used with a reasonable amount of confidence, where simple accuracy is defined in equation 2.1.

$$Simple\ accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{2.1}$$

However, if 90% of the dataset is labelled as *true* (and the rest *false*), a classifier that always predicts positive would have a simple accuracy measure of 90%, which clearly is not indicative of the strength of the algorithm. It is therefore evident that using simple accuracy to measure classification performance on a dataset in which classes are not evenly distributed will not provide meaningful information.

Due to the fact that the class distribution in the dataset used in this study is highly imbalanced (i.e. the proportion of patients with drug resistance is very high), only the evaluation measures that are invariant to



tp - True Positives, **fn** - False Negatives, **fp** - False Positives, **tn** - True Negatives

Figure 2.4. Single label confusion matrix.

class imbalance are of interest. Three types of evaluation measures are reviewed below. In section 2.2.1 classification performance evaluation measures are reviewed, which are statistics calculated based on the output of a classification technique and are used to to determine the predictive power of the technique. Methods used for validation of classification techniques are then reviewed. These methods are used in order to ensure that the performance evaluation measures can be relied upon. Lastly, measures that are used to describe datasets are reviewed in section 2.2.3.

2.2.1 Performance Evaluation Measures

When selecting a model for a classification task, it is desirable to choose the model with the best performance. Defining what is meant by good performance and measuring this is a non-trivial task. The most common evaluation measures used in the literature to assess the performance of classification tasks are discussed below. For the purposes of this discussion, the training dataset is defined as $\mathscr{D} = \{(\mathbf{x}_i, Y_i)\ for\ i = 1, \ldots, m\}$, where $\mathbf{x}_i$ is a feature vector and Y_i is the true set of labels associated with $\mathbf{x}_i$. The testing dataset and predictions produced by a classification technique are denoted by $\mathscr{D}_T = \{(\mathbf{y}_i, Z_i)\ for\ i = 1, \ldots, s\}$, where Z_i is the set of labels predicted for each previously unseen test example $\mathbf{y}_i$.

Confusion Matrix-Based Measures

The *confusion matrix* (sometimes referred to as a *contingency table*) is a tabulation of the performance of a classification technique with respect to a specific label. Figure 2.4 shows an example confusion matrix, in which *tp*, *fn*, *fp* and *tn* are place holders for the numeric values (total numbers of each type of result) that would be generated during a testing or cross validation run.

Table 2.1 defines the most common confusion matrix-based evaluation measures used in the literature [59–61]. The definitions are given for the case of single label (binary) classification. These measures can be used without modification for multiclass and multi-label classification by considering the performance of the classifier on only a single label.

All the measures in table 2.1 have been generalised for the case of multiclass classification. This is done by calculating the mean over each of the labels λ_i. The mean can be calculated by *macro-averaging* (denoted by M) or by *micro-averaging* (denoted by μ). Macro-averaging takes the simple mean of the measure calculated for each label, as shown in equation 2.2.

$$\mathrm{TPR}_M = \frac{\sum_{i=1}^{Q} \frac{tp}{tp+fn}}{Q} \tag{2.2}$$

In contrast to equation 2.2, micro-averaging takes the sum of each component of the measure before calculating the performance measure itself. The corresponding micro-averaged TPR performance measure is shown in equation 2.3.

$$\mathrm{TPR}_\mu = \frac{\sum_{i=1}^{Q} tp_i}{\sum_{i=1}^{Q} tp_i + fn_i} \tag{2.3}$$

An important difference between the two measures is that the macro-averaged measure effectively assigns all classes an equal weight, while micro-averaged measures are influenced more by the bigger classes. Both measures can be used as an indication of the performance of a classification technique across all labels.

Table 2.1. Single label confusion matrix-based evaluation measures [58–60].

Measure	Equation	Description
Accuracy	$\frac{tp+tn}{tp+fn+fp+tn}$	The percentage of labels that a classifier correctly predicts. Equivalent to equation 2.1.
True Positive Rate (TPR), *recall* or *sensitivity*	$\frac{tp}{tp+fn}$	A measure of how good a classifier is at identifying positive examples.
True Negative Rate (TNR) or $specificity$	$\frac{tn}{tn+fp}$	A measure of how good a classifier is at identifying negative examples.
Precision	$\frac{tp}{tp+fp}$	Indication of how many of the positive predictions were made correctly.
F-Score	$\frac{(\beta^2+1)tp}{(\beta^2+1)tp+\beta^2 fn+fp}$	Relationship between actual positive labels and predicted positive labels. F-score is evenly balanced when $\beta = 1$, favours precision when $\beta > 1$ and TPR otherwise [59].

The macro-averaged measure is only useful if the classes are distributed evenly, while the micro-averaged measure can be used even in the presence of class imbalance.

Table 2.2 summarises the common multi-label confusion matrix-based evaluation measures defined in the literature [19, 60]. The *exact match ratio* is the strictest performance feature, since only predictions in which all labels are predicted correctly are considered an exact match. Hamming Loss, related to Hamming distance [62], is an average measure of how well the predicted label sets match with the true labels.

Receiver Operating Characteristic

Many classification algorithms, such as SVMs and naive Bayes, output a probability of class membership rather than a simple binary *positive* or *negative* value. Thus, to determine whether a prediction should be interpreted as indicative of class membership or not, a threshold (τ) is usually applied. Threshold selection strategies have been studied [63], but are beyond the scope of this review.

Table 2.2. Multi-label example-based evaluation measures [58–60].

Measure	Equation	Description
Exact Match Ratio	$\frac{1}{m}\sum_{i=1}^{m} I(Z_i = Y_i)$	The percentage of label sets that were predicted 100% correctly (i.e. each predicted label was correct and every relevant label was predicted).
Hamming Loss	$\frac{1}{QM}\sum_{i=1}^{M}\sum_{j=1}^{Q} I(Z_j[i] = Y_j[i])$	The average symmetric difference between the predicted label set Z_i and the actual label set Y_i.
F_1 Score	$\frac{1}{m}\sum_{i=1}^{m} \frac{2\lvert Y_i \cap Z_i\rvert}{\lvert Y_i\rvert + \lvert Z_i\rvert}$	Average F Score, with $\beta = 1$, over all labels.
Accuracy	$\frac{1}{m}\sum_{i=1}^{m} \frac{\lvert Y_i \cup Z_i\rvert}{\lvert Y_i + Z_i\rvert}$	Average simple accuracy over all labels.

I is the *indicator function* defined as $I(true) = 1$ and $I(false) = 0$

Receiver operating characteristic (*ROC*) graphs are a way to visualize the performance of a classification technique. ROC graphs are generated by varying τ and plotting true positive rate (TPR) versus false positive rate (FPR, where $FPR = 1 - TNR$) in two dimensions [61]. By varying τ from 0 to 1 in regular intervals and for each interval calculating TPR and FPR, and plotting the (FPR, TPR) tuple, a *ROC* curve is constructed iteratively. If, for each value of τ, multiple tests are performed, as in the case of cross validation, then an average of the performance measure must be calculated. Although various averaging strategies exist [61], the most common is vertical averaging, which in the case of an (FPR, TPR) *ROC* curve means calculating the average true positive rate.

The *area under the ROC curve* (AUC) is a scalar measure that summarises a ROC curve. AUC will always be a value between 0 and 1, since it is a portion of the unit square. This is because each axis of the ROC curve represents a rate. AUC is widely used [8, 26, 27, 36, 64–67]. An important property of *ROC* curves (and hence *AUC*) as an evaluation measure is that they are invariant to changes in class distribution, therefore provide a robust method for evaluating classifier performance.

Multi-Label Evaluation Measures

In single label classification problems, each prediction made by a predictive model is either 100% correct or 100% incorrect. This is because only one label is being considered and the label is either associated with the example or it is not – there are no partially correct classifications. In multi-label classification the situation is different. For example, let the set of labels predicted to be associated with an unseen test example $\boldsymbol{y}_i$ be Z_i. Some members of the predicted label set Z_i may indeed be associated with $\boldsymbol{y}_i$, while others may not. This concept of partially correct predictions makes the evaluation of multi-label predictive models less straightforward than single label ones.

Since the task of label ranking is not considered in this study, there are two groups of classification evaluation measures that apply to multi-label classification. These are the *example-based* and *label-based* methods. Example-based measures examine the relationship between the full set of predicted labels (Z_i) and the true data labels (Y_i) for each unseen test example ($\mathbf{y}_i$). All labels in Z_i are used in the calculation of example-based measures, which implies that each prediction made is considered as a whole. In contrast, label-based evaluation measures are calculated using the individual members of the Z_i set. Consequently, these measures consider the predictive power of a classification technique at the individual label level.

The example-based evaluation measures only apply to multi-label classification problems and are outlined in table 2.2 [19, 60]. These measures assess the performance of a classifier over all labels in the predicted label set and the averages are taken over all test examples. The example-based evaluation measures can all be averaged on a macro (M) or micro (μ) level as described above.

Both label-based and example-based methods have their uses. If it is sufficient to know how a classifier performs across all labels on a dataset, then the example-based measures can be used. If it is important to know exactly how well a classification technique performs for an individual label, then the label-based measures should be used.

Higher Order Measures

Some composite higher order classification evaluation measures exist that provide further performance assessments. Many of these measures combine confusion matrix-based or ROC measures with statistical theory [58]. Researchers have defined *quadratic 'distance'*, which is associated with a negative log-likelihood approach for a Gaussian model. L^P distance has also been defined as in equation 2.4.

$$L^P(Z_i, Y_i) = \left[\sum_{j=1}^{Q} |Z_i[j] - Y_i[j]|^p \right]^{\frac{1}{p}} \tag{2.4}$$

Note that if $p = 1$ the definition is the same as Hamming Loss as defined in table 2.2.

The notion of *correlation* is used widely in statistics to determine the relationship between two or more variables. A form of the Pearson correlation coefficient [68] has been used in the context of machine learning and is referred to as the *Matthews correlation coefficient* [58, 69, 70]. The Matthews correlation coefficient (*MCC*) is defined in equation 2.5 in terms of the quantities in the confusion matrix. The measure is always a value between -1 and 1 and provides a comparison with respect to a random baseline. When $MCC = 0$, performance is considered to be no better than a random performance. A value of +1 indicates good performance (good correlation between the predicted labels and the actual labels) and a value of -1 indicates poor performance. One problem with the *MCC* measure is that it is not always defined, for example, if $tp + fp = 0$ (no positive predictions are made).

$$\text{MCC} = \frac{tp \cdot fn - fp \cdot fn}{\sqrt{(tp + fp) \cdot (tp + fn) \cdot (tn + fp) \cdot (tn + fn)}} \tag{2.5}$$

Another statistical measure that has been used to gauge classifier performance is *discriminant power* (*DP*) [59, 71]. This measure attempts to avoid the tradeoffs between sensitivity and specificity by applying a logistic odds transformation. *DP* is defined in terms of the confusion matrix quantities and the standard deviation of the logistic normal distribution curve ($\frac{\sqrt{3}}{\pi}$) in equation 2.6.

$$\mathrm{DP} = \frac{\sqrt{3}}{\pi} ln \left(\frac{tp}{fp} \cdot \frac{tn}{fn} \right) \tag{2.6}$$

A classifier is a poor discriminant if $DP < 1$, limited if $DP < 2$, fair if $DP < 3$ and good otherwise.

2.2.2 Predictive Model Validation

Model validation is the task of assessing whether or not a model will perform well on independent test data [17]. This is important because it gives the assurance that the model will perform well in general and is not just over-fitting to a specific dataset. The most simple and widely used method of model validation is cross-validation [8, 27, 28, 36, 47, 64–67, 72–75].

There are two main types of cross validation, *leave one out* cross validation and *k-fold* cross validation [76]. *Leave one out* cross validation builds a model using all of the training data except one example and uses the omitted example to test. This is done for each example in the training set. For large training sets this can become infeasible due to long training times, so *k-fold* cross validation is normally used. In *k-fold* cross validation, the dataset is divided into k equally sized subsets. The model is then trained using $k-1$ of the subsets and testing is done using the omitted subset and this is repeated k times, each time leaving out a different subset. Performance measures are then averaged over the k cross validation cycles.

The value of k in *k-fold* is chosen in order to reduce both *bias* and *variance*. Bias can be introduced into a predictive model if it is trained using a subset of the dataset that is not representative of the dataset as a whole. This is because the selected training dataset could lack training examples of a specific type – such as example associated with a particular combination of labels, say Y_i. The resultant predictive model would then never be able to make a prediction of Y_i for an unseen training example (even though the label set Y_i exists in the complete dataset). It can be said that this model is biased against Y_i [76]. It has been shown that in most cases a value of $k = 10$ is optimal for model validation [16, 17, 76].

An issue that often arises in real-world problems is *class imbalance* in the data [77,78]. Class imbalance in a dataset exists when the proportion of examples labelled with a particular label λ_i is much bigger (or smaller) than the number of examples not labelled with λ_i [77]. This can lead to certain performance measures (such as simple accuracy as defined in equation 2.1) providing unreliable information. Class imbalance can also affect model validation if training and test sets are not representative of independent external data. To compensate for this, *stratification* is often used to construct cross validation data subsets (referred to as *folds*) that are representative in terms of label distribution. Stratification has been shown to improve cross validation in terms of both bias and variance [76].

Standard stratification techniques do not apply to multi-label datasets, since examples are associated with more than one label, which makes label distribution more difficult to represent and calculate. Two algorithms have been proposed for the stratification of multi-label data. The *Labelsets-based stratification* algorithm [79] attempts to strictly maintain the distribution of all labels across each of the k cross validation subsets. *Iterative stratification* [79] proposes a relaxed version of the labelsets-based algorithm in which only infrequently occurring labels are strictly distributed across the k subsets.

2.2.3 Dataset Description Measures

Two dataset description measures are defined for describing multi-label datasets [19, 21], namely, *label density* and *label cardinality*. The label cardinality of a dataset is the average number of labels associated with each example and is defined in equation 2.7 [19].

$$\text{Label Cardinality} = \frac{1}{m}\sum_{i=1}^{m}|Y_i| \tag{2.7}$$

The label density of a dataset (defined in equation 2.8) is the average number of labels associated with each example divided by the number of unique labels [19].

$$\text{Label Density} = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i|}{n} \tag{2.8}$$

Label cardinality is independent of the number of unique labels (n) and is used to characterise the number of labels associated with the average example, while label density takes the number of labels in the domain into consideration.

Another important dataset description measure is the label distribution [77,78,80,81]. This is calculated by considering the simple occurrence counts (frequencies) of each label in the dataset and is important because if label distributions are not balanced, then certain evaluation measures may not provide meaningful information.

The final dataset evaluation measure to consider is feature completeness [18]. It could be the case that not every training or test example has a value associated with each of its features. Some algorithms do not support empty feature values, so it may be necessary to either remove incomplete features or add placeholder values, depending on the classification algorithm being used.

2.3 Applications

Statistical learning methods have been applied to a wide range of problems, from predicting the stock market to identifying handwritten lettering from a digitised image [17]. However, in this review the discussion is limited to the application of classification techniques to the problem of predicting HIV drug resistance and therapy outcome. These studies are relevant because they train statistical models using the same type of data used in this study, although often with the addition of viral genotype data. Both single label and multi-label classification studies are considered, since multi-label techniques have not yet been widely applied to the relevant problems.

The review of the relevant applications is organised into three sections. The first subsection discusses the applications based on the objective of the classification task, while the second subsection reviews the applications based on the data used for training the predictive models. The measures used to validate and evaluate the performance of the applied techniques are discussed in the final subsection.

2.3.1 Classification Objective

Predictive models trained on HIV-related clinical data are built to predict either *therapy outcome* (also referred to as *response to therapy*) or they are trained to predict HIV drug resistance. Therapy outcome is measured in practice by observing a patient's *viral load*. The viral load is a measure of how many copies of the HI virus are present in the cells of a patient. A low post-therapy viral load measurement implies a good response to therapy, while a high viral load is considered a bad therapy outcome. Since viral load is a continuous variable as opposed to a categorical one, the task of predicting therapy outcome is one of *regression* rather than of classification. Regression is the exercise of estimating the value of a continuous variable.

HIV drug resistance is detected by examining the genome sequence of the virus. Therefore, methods that try to predict HIV drug resistance use classification labels that correspond to mutations in the genome of the HI virus that are known to cause resistance to antiretroviral drugs. It follows that predicting drug resistance in this way is a classification task and not regression.

Antiretroviral Treatment Decision Support

Machine learning models for HIV treatment have often been applied to the problem of predicting therapy outcome and the related task of *therapy selection* [8, 64–67, 73, 75, 82]. Therapy selection (sometimes called *therapy optimisation*) is the task of choosing the drug regimen for a patient in order to improve the therapy outcome.

Models that predict response to therapy by producing a viral load prediction can be used to support the selection of a new therapy by using the model to predict the outcome of each possible therapy. Thus, models constructed to predict response to therapy can be used for therapy optimisation without modification.

HIV Drug Resistance Prediction

Only three studies apply classification techniques in order to directly predict resistance to antiretroviral drugs [36, 72, 74]. Antiretroviral drugs (ARVs) are the medication used to treat HIV infection. Resistance to a drug implies that the drug will not be effective if taken by the patient. All of these studies (each briefly discussed below) use viral genotype data as input and produce genotype mutation predictions as output.

Artificial neural networks have been used to predict drug resistance using only genomic sequence data [72]. Cross validation was used and the accuracy was defined as the ratio of correct classifications to the total number of classification (see equation 2.1). The best model had an accuracy of 68%.

A meta-classifier called a *super learner* was used on viral genotype data consisting of 740 training examples to select an optimal classifier from a set of candidate classifiers [74]. Susceptibility to one class of antiretroviral drugs (protease inhibitors) was predicted. The *super learner* was validated using 10 fold cross validation and evaluated using the concept of *risk*. This study defined risk as the difference between the *expected value* of the estimated parameter and the actual value.

The binary relevance and ensemble of classifier chains multi-label classification techniques (described in section 2.1 above) were applied to a dataset consisting of 600 genomic sequences. The classifiers were trained to predict resistance to six drugs from one drug group (nucleoside analogues). ECC performed best and achieved a mean AUC that ranged between 0.792 and 0.897 across the different drugs. 10 fold cross validation was used to validate the models.

2.3.2 Training Dataset

The discussion below is organised into two sections based on the type of data used for training the predictive models. First the applications that include viral genome sequences as part of the training data are considered, then the applications that omit viral genome data are reviewed.

In practice, the two most important clinical measures used to evaluate the condition of a patient infected with HIV are *viral load* (defined in section 2.3.1) and CD4 count. CD4 count is a measure of the number of T-helper cells in the blood and is an indicator of immune system health. These measures are therefore the most commonly used training features along with viral genome sequences. Viral genome sequences contain information about virus mutations that result in drug resistance.

Genomic Data Included

Most studies include genomic data, since this type of data are seen to provide the most amount of information about HIV drug resistance [36, 64, 65, 67, 72–75, 82]. Some studies train their classifiers using data elements known as *treatment change episodes* (TCEs) [8, 64–66, 73, 75, 82]. A TCE describes the cluster of data around the point at which a patient stops one antiretrovial regimen and starts a new one, i.e. a treatment change [82, 83]. The TCE concept was introduced in 2003 by the RDI research group[1] in an attempt to standardise the format of clinical data used for training classification models, and has been adopted by some authors outside of RDI. A TCE data element includes therapy (drug regimen), viral load and CD4 count measurements at specific times before and after the treatment change, as well as viral genotype information.

Three regression techniques (*generative discriminative engine*, *mixed effects engine* and *evolutionary engine*) were applied to a dataset consisting of about 3000 TCEs for the purpose of predicting therapy outcome [64]. These techniques were applied both individually as well as in various combinations. The best accuracy obtained on a test set was 75.1% and the highest *AUC* was 0.804.

Artificial neural networks (ANNs) have been used to predict response to antiretroviral therapy [82] using a dataset consisting of 1154 TCEs. Three-layer neural networks were constructed and the output of the models (prediction) was follow-up viral load and the evaluation measure was correlation (r^2). Three 'committees' of 10 ANNs were tested, with the best model achieving a correlation of 0.69. The r^2 correlation is always in the range $[0; 1]$, so a value of 0.69 represents above average performance.

[1] http://www.hivrdi.org/

Three regression techniques were applied to the problem of optimizing therapy selection [67]. Models that were trained using viral genotype and clinical, as well as therapy data obtained an AUC of 0.77. When only viral genotype data was used, the AUC achieved dropped to 0.75. It was also shown that a combination of models performed better than any single model.

A study that used 3143 TCEs combined with patient demographic, treatment history and viral genotype data found that purely data-driven models are comparable to state of the art rule-based systems [65]. Logistic regression, decision trees, random forests and rule bases were used to predict response to antiretroviral therapy. Simple accuracy as well as AUC were used as evaluation measures, along with 10 fold cross validation. The highest accuracy value obtained was 75.63% and the best AUC achieved was 0.761.

The performance of three models (ANN, SVM and random forest) at predicting response to antiretroviral therapy was compared [75]. Data from 1204 treatment change episodes were used that included genotype, clinical and treatment data. The predicted variable was follow-up viral load. Correlation as well as absolute difference between the predicted and actual viral load were the performance measures used. The highest correlations attained were 0.546 for ANN, 0.751 for random forest and 0.720 for SVM. The study concluded that a combination of models results in the best performance.

Genomic Data Excluded

In developing countries patients do no always have access to viral genotype data. This could be due to financial constraints or lack of resources such as laboratories and the highly skilled staff required to perform the tests. In these scenarios only clinical data such a CD4 counts and viral load measurements are available for managing patients infected with HIV. Two recent studies [8, 66] begin to address the problem of making predictions related to HIV treatment without using genomic data.

Random forests were used to predict therapy outcome for the purposes of new therapy selection [66]. Two datasets were used, one with 2831 TCEs (dataset 1) and one with 2579 TCEs (dataset 2). Additional

clinical, demographic, treatment history and GRT data were included in the dataset. Two models were constructed for each dataset, one that included the GRT data and one that omitted it. The model with GRT achieved an AUC 0.77 in dataset 1 and 0.834 on dataset 2. Models constructed without GRT data achieve comparable results, with an AUC of 0.757 and 0.821 achieved on dataset 1 and 2 respectively, which is indicative of good performance.

Response to antiretroviral therapy was modelled in a study that used 14891 TCEs that did not include viral genotype data [8]. Models were evaluated using cross validation as well as independent test TCEs from external datasets. AUC was used as the main evaluation measure, along with simple accuracy, sensitivity and specificity. The mean AUC achieved using cross validation was 0.77. The mean values for accuracy, sensitivity and specificity were 72%, 72% and 71% respectively, which indicates significant predictive power.

While most applications of classification algorithms to the problem of predicting HIV drug resistance and therapy outcome use genomic data for training, more recent studies are beginning to investigate training models using only clinical data. This is to address the reality of the situation in developing countries, where genomic data is not always available. It has been shown that using clinical data only, predictive models can achieve accuracy comparable to models trained using genomic data.

2.3.3 Evaluation Measures

By far the most common performance evaluation measure used in the above applications is the area under the receiver operating characteristic curve (AUC) [8, 26, 27, 36, 64–67]. This is most likely due to the robustness of the measure across different datasets. The most common validation technique used was k fold cross validation, with $k = 10$ the most common value for k [8, 27, 28, 36, 47, 64–67, 72–75].

2.4 Summary

Multi-label classification techniques and applications applied to the problem of predicting HIV drug resistance and therapy outcome prediction have been reviewed. The most important classification techniques

from each of the three primary solution groups (*problem transformation*, *algorithm adaptation* and *ensemble* methods) have been described.

It has been shown that the task of assessing model performance is not trivial and that there are many available evaluation measures to do so. Many of the performance measures are in some way derived from the *confusion matrix*, which enumerates the correct and incorrect predictions produced by a classification technique.

It has been illustrated that many investigations have been done into the task of applying supervised machine learning techniques to the problem of HIV drug resistance and treatment decision support. The techniques make use of a variety of classification techniques, but are mainly trained on the same type of data. The evaluation and validation methods used are also largely aligned.

Very few studies have investigated the application of classification techniques trained only using clinical data to the problem of predicting HIV drug resistance. In the remainder of this study, the best practices extracted from the reviewed literature will be used to apply multi-label classification techniques to exactly this problem.

Chapter 3

Methods

This chapter describes the multi-label classification methods used in this study, as well as the measures employed to evaluate their performance. Also described are the validation mechanisms used to ensure the reliability of these evaluations. The selected methods and mechanisms have been largely informed by previous work as described in chapter 2.

3.1 Multi-Label Classification Techniques

Six multi-label classification techniques were used to train seven different predictive models using the training data (described in section 4.1.2). The goal of this process was to develop classification models capable of predicting which drugs a patient is resistant to without the presence of a genotype resistance test result. The six techniques were selected after a review of the literature (presented in chapter 2) and are a representative sample of the multi-label classification solution space (see figure 2.1). Some theoretical background for each technique is given below.

3.1.1 Problem Transformation Methods

Binary Relevance (BR)

The first multi-label classification technique tested was binary relevance.

The binary relevance (BR) multi-label classification technique transforms the nature of the problem from multi-label to single label (binary) [20]. That is, for each label in the multi-label dataset a binary classifier is trained. To classify an unseen example, each of the binary classifiers is consulted for a prediction for the label it was trained to identify. To compose the final multi-label prediction, the predictions from each of the binary classifiers are simply concatenated. This mechanism is illustrated in figure 3.1.

Due to the fact that this technique is independent of any specific binary classification algorithm, it is possible to use any such algorithm to train the binary classification models. The two different algorithms that were selected as the so-called *base classifiers* are discussed in the next two sections.

Support Vector Machine (BR-SVM) Support vector machines (SVMs) are classification models that are commonly used in pattern recognition problems [84, 85]. SVMs map the feature space into a much higher dimensional space using a technique referred to as the *kernel trick*, which enables this mapping to be done implicitly (without have to actually calculate the mapping).

If the (now single label, since the binary relevance transformation has been applied) dataset has n examples, each with p features, it can be represented as shown below:

$$\mathscr{D} = \{(\mathbf{x}_1, \lambda_1), (\mathbf{x}_2, \lambda_2), \ldots, (\mathbf{x}_n, \lambda_n)\} \text{ where } \mathbf{x}_i \in \mathbb{R}^p$$

where the λ_i are the binary label values for each example ($\mathbf{x}_i$).

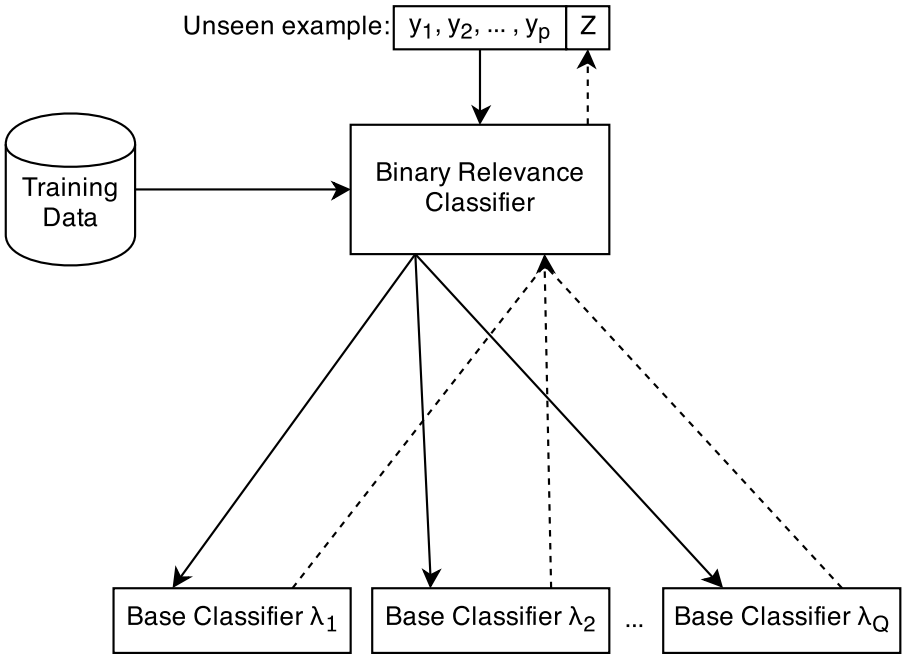


Figure 3.1. Binary relevance classifier. The solid arrows indicate input data flow while the dashed arrows indicate output data flow. The y_i represent the features and the Z represents the set of labels to be predicted.

SVMs construct a maximum margin hyperplane in the higher dimensional space by solving the following optimisation problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } \lambda_i(\mathbf{w}^T\Phi(\mathbf{x}_i)+b) \geq 1-\xi_i \text{ with } \xi_i \geq 0 \tag{3.1}$$

where $\mathbf{w}$ is the normal vector to the hyperplane, $C > 0$ is the cost parameter of the error term and b is the hyperplane offset.

Given the optimisation problem described in 3.1, the kernel function K is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$$

A number of kernel functions have been proposed, but it has been suggested [86] that the radial basis function (RBF) kernel is a reasonable choice. The RBF kernel function is defined in equation 3.2.

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2) \text{ with } \gamma > 0 \tag{3.2}$$

Naive Bayes (BR-NB) A naive Bayes classifier models the probability of the class variable using the simplifying assumption that each feature in the feature vector is independent [87]. The probabilistic model developed for generating the predictions is derived from Bayes' theorem, which is defined in equation 3.3.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.3}$$

The term $P(A|B)$ is the *posterior* probability and can be interpreted as the probability of proposition A given that proposition B is true. The term $P(A)$ is the *prior* probability of A, or the probability of proposition A without having any information about B. In terms of binary classification the theorem can be rewritten as follows.

$$P(C = \lambda_j|\mathbf{x}_i = \mathbf{y}) = \frac{P(\mathbf{x}_i = \mathbf{y}|C = \lambda_j)P(C = \lambda_j)}{P(\mathbf{x}_i = \mathbf{y})} \tag{3.4}$$

Equation 3.4 can be interpreted as the probability that the class (C) will have the value λ_j given that the feature vector ($\mathbf{x}$) has the values $\mathbf{y}$. Expanding this to explicitly include each individual feature gives the following (omitting the variable assignments):

$$P(\lambda_j|y_1,y_2,\ldots,y_p) = \frac{P(y_1,y_2,\ldots,y_p|\lambda_j)P(\lambda_j)}{P(y_1,y_2,\ldots,y_p)}$$

The y_i are given, which makes $P(y_i,y_2,\ldots,y_p)$ a constant (labelled F). Further, observing that the numerator is equivalent to the joint probability $P(\lambda_j,y_i,y_2,\ldots,y_p)$ and applying the chain rule it can be seen that [87–89]:

$$\begin{aligned}
P(\lambda_j|y_i,y_2,\ldots,y_p) &= \frac{1}{F}P(\lambda_j,y_i,y_2,\ldots,y_p) \\
&= \frac{1}{F}P(\lambda_j)P(y_1,y_2,\ldots,y_p|\lambda_j) \\
&= \frac{1}{F}P(\lambda_j)P(y_1|\lambda_j)P(y_2,\ldots,y_p|\lambda_j,y_1) \\
&= \frac{1}{F}P(\lambda_j)P(y_1|\lambda_j)P(y_2|\lambda_j,y_1)P(y_3,\ldots,y_p|\lambda_j,y_1,y_2) \\
&= \frac{1}{F}P(\lambda_j)P(y_1|\lambda_j)P(y_2|\lambda_j,y_1)P(y_3|\lambda_j,y_1,y_2)P(y_4,\ldots,y_p|\lambda_j,y_1,y_2,y_3)
\end{aligned} \tag{3.5}$$

At this point the "naive" independence assumption is made. In the naive Bayes classification model, it is assumed that $P(y_i|\lambda_j,y_k) = P(y_i|\lambda_j)$, or more generally $P(y_i|\lambda_j,y_k,\ldots,y_{k+l}) = P(y_i|\lambda_j)$. Taking this into account, equation 3.5 simplifies to:

$$\begin{aligned}
P(\lambda_j|y_i,y_2,\ldots,y_P) &= \frac{1}{F}P(\lambda_j,y_i,y_2,\ldots,y_p) \\
&= \frac{1}{F}P(\lambda_j)P(y_i|\lambda_j)P(y_2|\lambda_j)\cdots P(y_p|\lambda_j) \\
&= \frac{1}{F}P(\lambda_j)\prod_{i=1}^{p}P(y_i|\lambda_j)
\end{aligned} \tag{3.6}$$

The naive Bayes classifier is built using equation 3.6 along with a decision rule. Intuitively, the classifier tries to find the label (λ_i) that has the highest probability given the feature vector ($\mathbf{y}$). This is know as the

maximum a posteriori probability (MAP) hypothesis [87] and can be expressed formally as:

$$\mathrm{NB}_{class}(\mathbf{y}) = \underset{j}{\operatorname{argmax}} \frac{1}{F} P(\lambda_j) \prod_{i=1}^{p} P(y_i|\lambda_j) \tag{3.7}$$

One of the efficiency problems present in the binary relevance method is that all base classifiers must be trained with every single training example. This issue is addressed by the following technique.

HOMER

The second multi-label classification technique considered in this research is referred to as the Hierarchy Of Multi-label classifiERs (HOMER) method [30]. HOMER constructs a hierarchy of multi-label classifiers as depicted in figure 3.2. Each node in the hierarchy (H_i) only considers a subset of the label set ($L_{H_i} \subseteq L$). The primary goal of architecting the technique in this way is to increase training and testing efficiency. The mechanism of this optimisation is explained below.

HOMER attempts to overcome the inefficiency of the binary relevance method by defining the concept of a *meta-label* (μ_{H_i}) as follows:

$$\mu_{H_i} \equiv \bigvee \lambda_j,\ \lambda_j \in L_{H_i} \tag{3.8}$$

That is, a training example can be considered as being labelled with μ_{H_i} if it possesses at least one of the labels in L_{H_i}. As an example, consider figure 3.2 and the training example shown below.

$$\mathbf{x}_i = (\{x_1, x_2, \ldots, x_p\}, \{\lambda_1, \lambda_2, \lambda_3\}) \tag{3.9}$$

The HOMER technique would label the training example in equation 3.9 with the meta-label μ_{H_1} in figure 3.2 since is it labelled with λ_1 and λ_2. During training, only the examples that are labelled with μ_{H_i} are passed down the hierarchy to H_i to be given as training input to the base classifiers at the leaf nodes below H_i. In this study, naive Bayes classifiers (described in some detail in section 3.1.1 above) are used as the base classifiers.

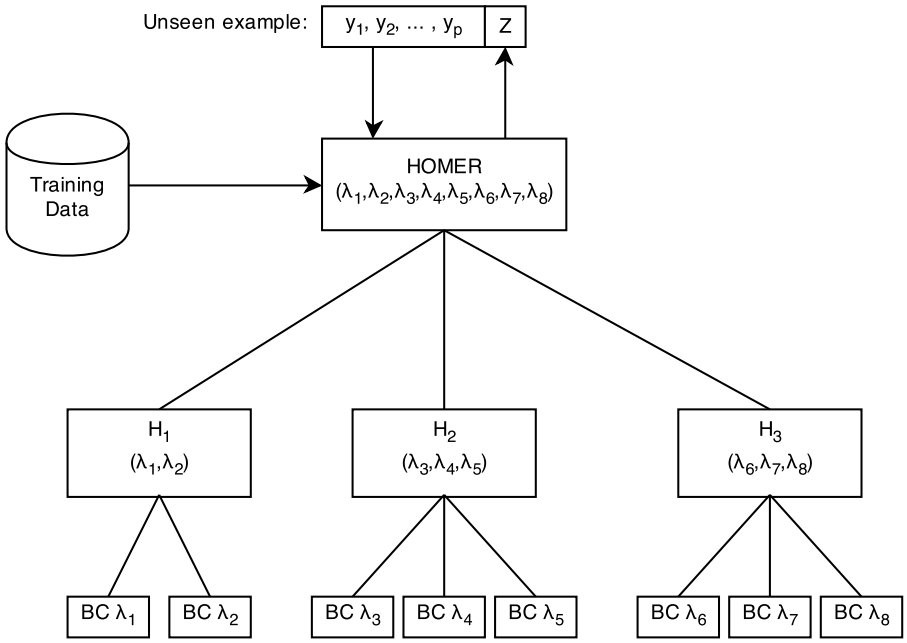


Figure 3.2. A simple hierarchy constructed by HOMER for a hypothetical problem with 8 labels. The H_i are the nodes in the hierarchy and the meta-labels (μ_{H_i}) are shown in parentheses for each node. The y_i represent the features and the Z represents the set of labels to be predicted. The *BC* λ_i are the base classifiers for each label.

To generate a label set prediction for an unseen example, the example is first passed to the root node of the hierarchy (*HOMER* in figure 3.2). Then, the example is recursively passed to each child node (H_i) using a depth-first traversal. The concatenation of the binary predictions produced by each base classifier is then taken to be the final predicted label set.

One problem that needs to be solved when constructing a classifier using the HOMER technique is how to create the disjoint label sets L_{H_i}. HOMER solves this problem by trying to distribute the labels as *evenly* as possible into k subsets such that labels belonging to the same subset are as *similar* as possible. HOMER achieves this by extending the *balanced clustering* algorithm [90] and introducing a novel clustering algorithm called *balanced k means* [30], which clusters the labels with an explicit constraint on the size of each cluster. These clusters are then used as the meta-labels.

3.1.2 Algorithm Adaptation Methods

MLkNN

The first of the algorithm adaptation methods explored in this work was the Multi-Label k Nearest Neighbours (MLkNN) technique [37]. Briefly, MLkNN is based on the traditional k nearest neighbours method and works by first identifying the nearest k neighbours of an unseen example and makes a prediction based on the labels associated with these neighbours.

The initial step of the MLkNN algorithm is to calculate the prior and posterior probabilities for each label. The prior probabilities are calculated as follows:

$$P(\lambda_j) = \frac{s \cdot \sum_{i=1}^{n} I_{\lambda_j}(\mathbf{x}_i)}{2s+n} \bigvee \lambda_j \in L \tag{3.10}$$

where $I_{\lambda_j}(\mathbf{x}_i)$ is the indicator function that returns 1 if $\mathbf{x}_i$ is labelled with λ_j and 0 otherwise and s is the smoothing factor. Intuitively, the posterior probabilities are calculated for each label (λ_j) by considering,

for each training example ($\mathbf{x}_i$), how many of its k nearest neighbours are labelled with λ_j (or not) if $\mathbf{x}_i$ is labelled with λ_j (or not). The nearest neighbours are found using a predefined distance function.

To predict the label set of an unseen example, the MLkNN algorithm first finds the k nearest neighbours of an unseen example $\mathbf{y}$, which is defined as $N_k(\mathbf{y})$. MLkNN uses the predefined distance function to calculate the distance between the unseen example ($\mathbf{y}$) and the examples in the training set ($\mathbf{x}_i$).

The predicted label set is determined using the maximum *a posteriori* probability principle (as defined in equation 3.7). Specifically, the predicted label set Z_i for an unseen example $\mathbf{y}_i$ is generated by applying the following for each label λ_j:

$$\lambda_j \in Zi = \underset{k \in 0,1}{\operatorname{argmax}} \frac{P(\lambda_j = k)P(\mathbf{y}_i|\lambda_j = k)}{P(\mathbf{y}_i)} \tag{3.11}$$

where $\lambda_k = 1$ indicates that the label is present and $\lambda_k = 0$ indicates that the label is not present.

Predictive Clustering Trees (PCT)

The fifth technique examined in this work is the predictive clustering tree (PCT) algorithm [39]. PCTs make use of first order logical decision trees [39] to cluster all training examples at nodes in the tree. The trees constructed by the PCT algorithm also have a *decision point* at each non-leaf node. This decision point can be thought of as describing the examples clustered at all nodes below that point in the tree.

The PCT method builds the clustering tree using a technique known as top-down induction [91]. The goal of this clustering is to establish clusters that minimise the *intra*-cluster distance and maximise *inter*-cluster distance. The resulting tree can also be referred to as a classification tree, since it can be used for generating predictions for unseen examples (as is the case in this study).

The final classification tree produced has a vector of probabilities at each leaf node. This vector contains a value for each possible label, indicating the chance that examples clustered at that specific leaf node are

associated with the respective label. Along with the classification model, the PCT technique produces a human readable decision tree that can be examined to follow the exact decision process used to make the classification decisions.

To classify an unseen example, the decision tree is followed from the root node down. At each node the learnt decision rule is consulted and the traversal is continued down a branch based on the result. This process continues until a leaf node is reached. The probability vector at this leaf node is then used to predict the label set.

3.1.3 Ensemble Methods

RAkEL

The first ensemble method tested was RAndom k-labELsets (RAkEL) [35]. This method constructs each member of an ensemble using a small random subset of the labels. Then, the label power set (LP) problem transformation method (see section 2.1.1) is used to train a single label classifier for each member of the power set of each of the subsets in the ensemble. One of the goals of this design approach is to take label correlations into account, but at the same time avoid the main problem of the LP method, which is that when label sets are large, the LP method becomes computationally intractable [35].

The RAkEL method defines the concept of a k-labelset as the set $L_k \subseteq L$ where $|L_k| = k$ and $L = \{\lambda_1, \lambda_2, \ldots, \lambda_Q\}$ is the set of all labels. The set of all distinct k-labelsets on L is denoted by L^k. The RAkEL algorithm constructs an ensemble of m classifiers by randomly selecting a k-labelset, L_{ki} from L^k and learning an LP classifier on the power set of labels in L_{ki}. This process is executed m times to produce m classifiers. These classifiers are denoted by h_i where $i = 1 \ldots m$.

For the classification of an unseen sample, $\mathbf{y}$, each classifier (h_i) provides a binary decision for each label, λ_j, in its k-labelset. RAkEL then calculates the average decision for each label and returns *true* (the unseen example should possess the label) if the average is greater than some threshold, t, or *false* otherwise.

Ensemble of Classifier Chains (ECC)

The Ensemble of Classifier Chains (ECC) technique [34] was the second ensemble method and final method investigated. The ECC technique builds on the binary relevance idea by using an ensemble of classifier chains (CC) to extend the feature space. This is achieved by adding a binary feature to represent the predictions of all previous classifiers, thereby forming a chain of classifiers. The goal of this modification is to model label correlations while keeping the computational complexity of the problem tractable [34].

As in the binary relevance method, classifier chains are constructed by training a classifier for each label, but the classifiers are trained in a specific order and the output of each classifier is used as an additional feature for training the next classifier (forming a chain). The top half of figure 3.3 illustrates this mechanism for a hypothetical example with six features and five labels. In the figure it can be seen that the first classifier is trained with only the example features, while the next classifier includes the first label as one of the training features. This continues for each subsequent classifier in the chain. The naive Bayes classifier is used as the base classier in this study.

To classify an unseen example using a classifier chain, the classifiers in the chain are consulted in the same order in which they were trained. The first classifier produces a prediction for the first label and this prediction is then used as a supplementary feature for the next classifier. This process is illustrated in the bottom half of figure 3.3.

The ECC technique builds an ensemble by training m classifier chains $C_1, C_2, \ldots, C_m$. Each C_i is trained using a random label ordering and well as using a random subset of $\mathscr{D}$ (the example space). To classify an unseen example using ECC, each C_i is consulted for a prediction for each label. The ECC algorithm then sums the predictions made for each label and if the sum for a particular label is above a certain threshold t, then the label is taken to be associated with the unseen example [21].

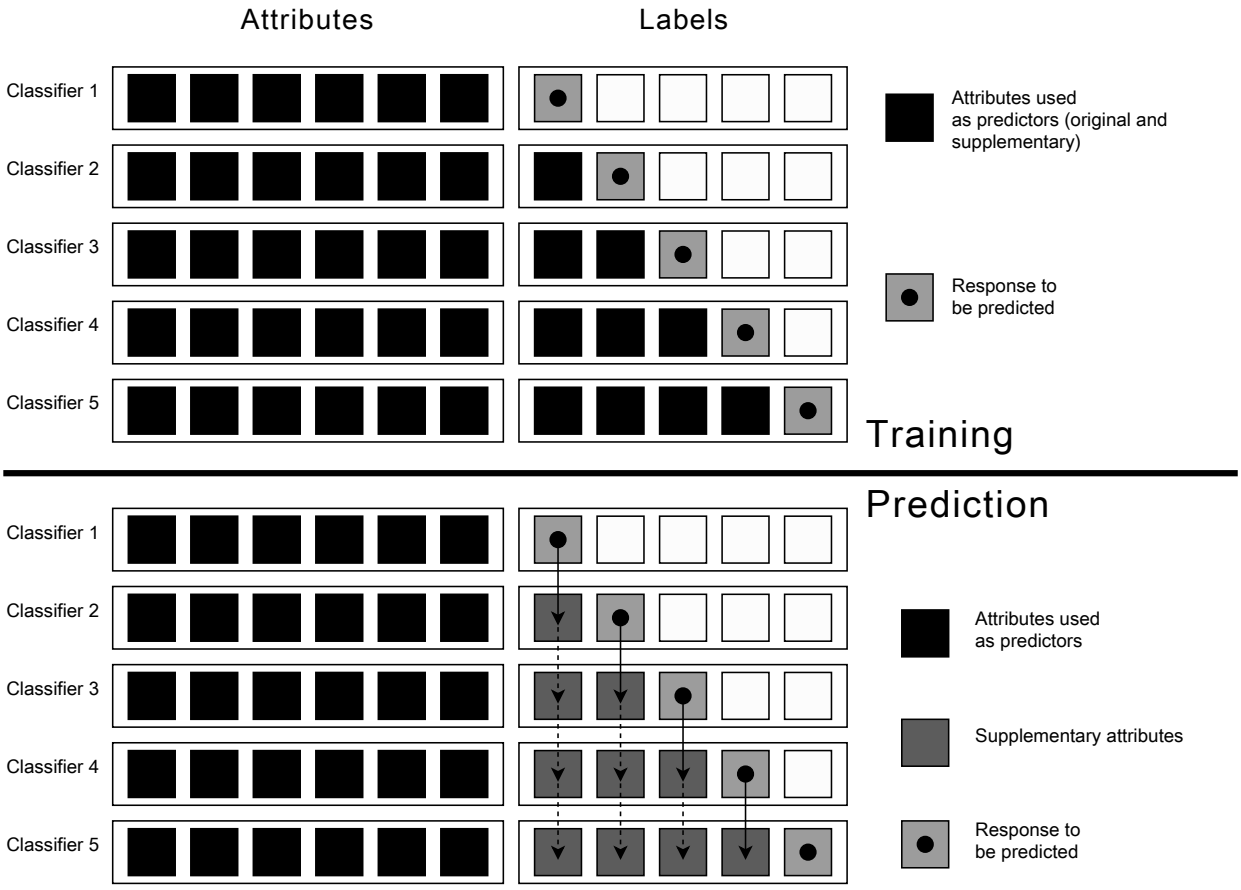


Figure 3.3. Mechanism of operation of a classifier chain (CC) [36].

3.2 Evaluation

In order to evaluate the results of the experimental methods described in section 3.1 above, it was necessary to select relevant evaluation measures from the many possible options described in section 2.2. Since the dataset used in this study is highly imbalanced, it was necessary to choose evaluation metrics that take into account label imbalance. Further, as discussed in section 4.1.1, it is important to know exactly which drugs a patient is resistant to, so it makes sense to evaluate the techniques on how well they perform at predicting resistance to specific drugs. Therefore, exclusive use was made of label-based evaluation measures as defined in section 2.2.1.

The measures described below have been selected as a representative sample of those used in the literature. The group of measures should be examined collectively in order to get the full picture of the performance of a particular technique.

3.2.1 Confusion Matrix

Most metrics are calculated from the confusion matrix generated by a classification or cross validation run (illustrated in figure 2.4). It has already been demonstrated that many measures (such as simple accuracy as defined in equation 2.1) are not invariant to an imbalanced class distribution. This means that class balance can have an effect on the value of the evaluation measure. It is therefore import to avoid using such measures in this study.

The two confusion matrix-based metrics that have been selected as evaluation measures in this study, namely *true positive rate* (TPR) and *true negative rate* (TNR) have been shown to be invariant to an imbalanced class distribution [60, 77]. These two evaluation measures are defined in equations 3.12 and 3.13 below:

$$\text{TPR} = \frac{tp}{tp + fn} \tag{3.12}$$

$$\text{TNR} = \frac{tn}{tn + fp} \tag{3.13}$$

where tp is the number of true positives predicted, tn the number of true negatives, fp the false positives and fn the false negatives.

3.2.2 Receiver Operating Characteristic

Receiver operating characteristic (ROC) curves are used as an additional evaluation measure, since it is common practice to do so and they have been shown to work well in the presence of class imbalance [61,77]. ROC curves depict relative tradeoffs between benefits (true positives) and costs (false negatives). The curves are generated by varying a threshold parameter (τ) and plotting the number of true positives versus the number of false positives for each threshold value.

Area Under Curve (AUC)

Area under the receiver operating characteristic curve (AUC) is a single scalar value that represents expected ROC performance [61]. Note that ROC curves can only be plotted for classifiers that produce a probability estimate to which thresholding can be applied and hence no such graphs are plotted for the BR-SVM and PCT methods.

3.2.3 Higher Order Measures

Matthews correlation coefficient (MCC) has been selected as a supplementary evaluation measure. The MCC is a measure of the correlation between the observed and predicted labels. Its value is in the range $[-1;1]$ with 1 indicating perfect prediction, 0 indicating no better than random prediction and -1 indicating total disagreement between prediction and observation [58]. It is defined in equation 2.5.

The final evaluation measure used in this study was discriminant power (*DP*), which is a measure that summarises sensitivity and specificity and is an indication of how well a classifier distinguishes between positive and negative examples [59]. It is defined in equation 2.6. A classifier is a poor discriminant if $DP < 1$, limited if $DP < 2$, fair if $DP < 3$ and good otherwise.

3.3 Validation Techniques

It is not sufficient to draw conclusions regarding the performance of a technique by simply training and testing a model once and calculating values for the selected set of evaluation measures. This is due to fact that overfitting[1] could have occurred, or a one-off ordering for a model with a randomisation component could have resulted in uncharacteristically high (or low) performance. It is therefore necessary to put additional control mechanisms in place to ensure that the measures of performance are reliable. These validation techniques are discussed in the following subsections.

3.3.1 Cross Validation

Cross validation is a popular technique used to validate the observed performance of a classifier. The technique works by splitting the training dataset into a number of folds. Each fold contains roughly the same amount of training examples. The model is then trained using the data from all but one of the folds. Testing is then done using the data that was left out of training. This process is repeated as many times are there are folds, each time training by leaving out a different fold and then testing with the left out fold. The performance is then taken as the average of the performance of these training/testing cycles.

[1] The result of overfitting is that a technique shows good performance on a particular dataset, but does not perform well in general.

3.3.2 Stratification

In order for the performance of each cross validation cycle to be representative of the performance if the full dataset was used for training, it is necessary to construct each fold so that the label distribution in the fold is the same as for the complete dataset [76, 79]. In the case of binary classification, the task of stratifying the dataset is simple, since there is only one target label whose distribution must be maintained. However, in multi-label datasets there are multiple distributions that must be maintained.

In this study the iterative stratification algorithm [79] is used to generate the stratified folds, since it is less strict than labelsets-based stratification and the datasets are too small for strict stratification rules to apply. That is, there are not enough examples labelled with specific label combinations for training and test folds to be strictly representative of the full dataset. The iterative stratification algorithm allows the best effort approach to constructing cross validation folds to be used.

3.3.3 Dataset Validation

The final and arguably most significant way that model validation has been performed in this study is by examining performance of the models under dataset evolution. A snapshot of the same dataset has been taken at two different points in time, once in 2011 and once in 2013. The decision was taken to run the same experiments on both datasets and compare the results.

The rationale behind this approach was to simulate model performance in the field over time. To truly examine how the models would perform in the field, it is necessary not only to test on a single, static dataset, but to test how the model performance is affected by the accumulation of data as clinical management of patients continues.

3.4 Summary

Theoretical background information and descriptions of algorithms for the selected six multi-label classification techniques have been given. The six evaluation measures used to measure the performance of the classification techniques used were also described. Finally, the details of the three control mechanisms put in place to ensure the validity of any evaluations have been given.

Chapter 4

Experimental Design and Implementation

This section describes the experimental design decisions made during the course of this study. Also discussed are the implementation details, including the experimental pipeline and the architectures and technologies of the software tools used.

4.1 Data

4.1.1 Clinical Dataset

The datasets used in this study form part of the Southern African Treatment and Resistance Network (SATuRN[1]) data repository. The data are the clinical records of 500 patients who have attended the Africa Centre[2] treatment failure clinic (TFC). On enrolment at the TFC, patients' medical files are collated and curated by the SATuRN research team members. Further, patients attending the TFC clinic each consult with a social worker who collects additional data that would not normally be collected. The result is a very detailed,

[1] http://www.bioafrica.net/saturn/index.php

[2] http://www.africacentre.ac.za/

relatively clean and comprehensive medical record for each patient that includes additional socio-economic information.

Table 4.1 summarises the dataset attributes. Patients in the datasets also have a viral isolate (used to perform GRT) associated with their clinical record, which is used to determine their resistance profile[3]. This resistance profile is then used to construct the label sets associated with each training example. The features in the *demographic*, *clinical*, *adherence* and *other* rows in table 4.1 exist in the clinical information system, while the features in the *derived* row are calculated during the training dataset development phase by the *CuRE* tool (described in sections 4.1.2 and 4.3.2 respectively).

The training data is anonymised and ethics approval has been obtained to analyse the data (as indicated in the acknowledgements section in the preamble). Some filter has been performed based on clinical criteria. To be included in the dataset, patients must be at least 16 years old, as pediatric HIV cases are treated quite differently from adult cases. Furthermore, patients must have had at least one genotype resistance test result on file, in order to generate the label set from their resistance profile (if any).

In order to evaluate how the performance of the selected classification techniques changes as the underlying dataset changes, two datasets have been used. The first dataset (labelled dataset 1) represents all the data collected for patients who attended the TFC up to June 2011. The second dataset (dataset 2) contains the data collected up to May 2013. The two datasets are snapshots of the same clinical information system at two different points in time and thus represent a real-world data evolution scenario.

Patients are resistant to between zero and 11 drugs (see tables 5.3 and 5.6). If it is known that a patient is resistant to a particular drug, then the clinical care provider will switch the patient to another drug to which the patient is susceptible[4]. This process is known as therapy optimisation and is an important step in the management of patients living with HIV.

[3] In this case, the resistance profile is essentially just a description of which drugs the patient is resistant and susceptible to.

[4] Susceptibility to a drug is the opposite of resistance and implies that the drug will be effective for that patient.

Table 4.1. Summary of features used to construct the predictive models.

Category	**Features**	**Types**	**Count**
Demographic	Age, Weight, Geographic Location, Ethnicity, Gender, Province, Country of Origin	Numeric, Categorical	7
Clinical	Drug Exposure (Tenofovir, Lopinavir/r, Atazanavir, Zidovudine, Ritonavir, Efavirens, Abacavir, Nevirapine, Raltegravir, Stavudine, Didanosine, Lamivudine), Recent Blood HB, Recent Blood ALT, Recent Blood Creatinine Clearance, Other Drug Exposure (three features), Tuberculosis Therapy (Prior, During, Post), HTLV-1 Status, HBV Status	Numeric, Categorical	23
Adherence	Treatment Break, Patient Estimated Adherence, Missed, Buddy, Remember, Counselling, Side Effects, Worst Stop, Disclosure, Names, Stop	Categorical	11
Other	Transmission Group, Other Co-morbidities, Partner On Treatment, Exposure to Single Dose NVP, Identified Virological Failure Reason, Traditional Medicine, Alcohol Consumption, TB Treatment Starting Soon, Diarrhoea or Vomiting	Categorical	9
Derived	Baseline Viral Load, Time on Failing Regimen, Drug Exposure Count, Pre-Resistance Testing Viral Load, Median Viral Load, Recent CD4 Count Gradient, Post-Treatment CD4 Count, Baseline CD4 Gradient, Pre-Resistance Testing CD4 Count, Mean Viral Load, Pre-Resistance Testing Immunological Failure, Virus Ever Suppressed	Numeric, Categorical	12
Total			**62**

4.1.2 Training Dataset Development

In order to transform the data used for clinical management of patients into a form that is useful for testing the multi-label classification techniques, a two step process is followed. First, it is necessary to determine which features should be constructed from the available dataset attributes. Following this, it is necessary to determine exactly how the feature vector (also referred to as an *example*) should be built from the data.

Feature Selection

The choice of features is influenced by two factors. First, the features commonly used in previous studies are used (as discussed in section 2.3). These features include baseline viral load, CD4 count, treatment regimen and others (see table 4.1). The second, and arguably more valuable process, is consultation with domain experts from the Africa Centre treatment failure clinic. The domain experts were consulted during numerous visits to the Africa Centre, during which meetings were held with researchers, infectious disease specialists as well as social workers. Over a period of a number of months, using an iterative feedback process, the final list of features was developed to represent the harmonised feature set considered to be important by the consulted experts.

While the elements of the feature vector represent the clinical and social data captured in the medical record for a patient, the labels represent the resistance profile of that patient. There are a total of 11 possible labels that can potentially be associated with each feature vector. Resistance to a drug is determined using the HIVDB 6.0.5 algorithm [5] and a patient is considered to be resistant to a drug if the algorithm returns a susceptibility value of ≤ 0.5.

Due to the importance of knowing exactly which set of drugs a patient is resistant to, formulation of the problem as a multi-label classification problem is natural, since a multi-label predictive model is able to associate an unseen test example with a set of labels (see section 2.1). The training dataset thus consists of feature vectors built using the medical records and the associated label sets comprise the drugs to which at least one patient has shown resistance.

Feature Vector Development

Before the clinical data can be used to train the predictive models, the data must first be transformed into a format that can be used by the relevant software. The data in the clinical information system is stored in a relational database and must be converted into a flat file in ARFF format. This ARFF file can then be used as input by the classification techniques.

In order to do this data manipulation in a way that reduces the possibility of human error, a custom tool has been developed (described in section 4.3.2) that can generate the training dataset in the required format. This tool enables the user to select any or all of the desired dataset features and will generate a file in one of two formats. One can be used by the classification techniques for experimentation and one is in human readable format that can be used for manual inspection and analysis of the data.

4.1.3 Technique Selection

There are three solution groups for solving the problem of multi-label classification (as described in section 2.1). Problem transformation (PT) methods divide the problem into a set of multiple single label classification problems and combine the result to form the multi-label prediction. Algorithm adaptation (AA) methods construct specialised algorithms, often by modifying existing single label prediction algorithms, to produce a multi-label prediction. Ensemble methods (EM) are developed on top of common problem transformation or algorithm adaptation methods [21].

In order to test the hypothesis put forward in section 1.1 and investigate the performance of multi-label classification techniques in general, two popular methods from each of the main solution groups were selected as indicated in figure 4.1.

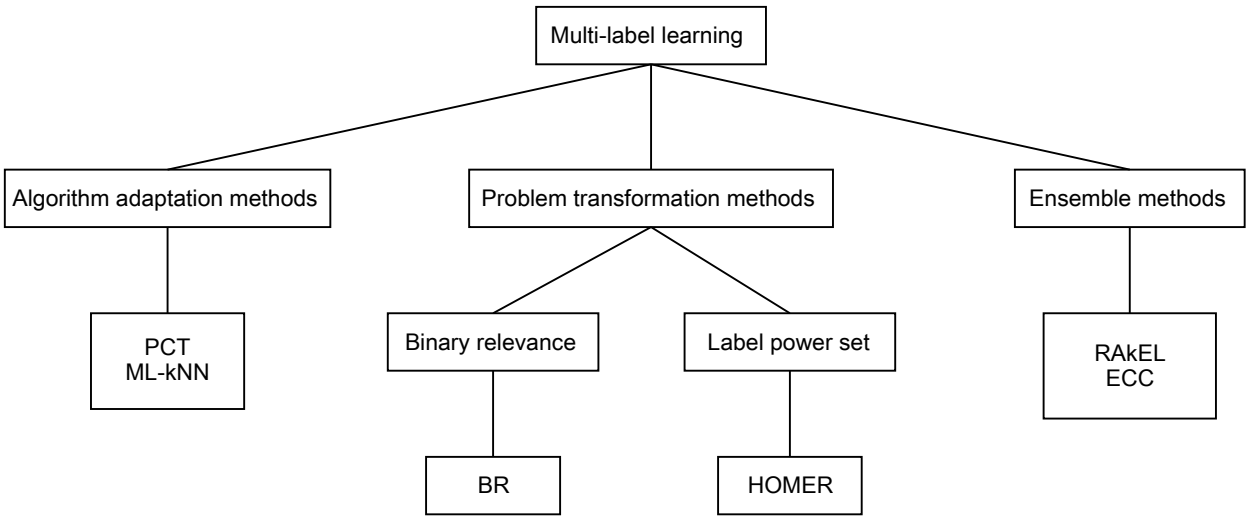


Figure 4.1. Multi-label classification methods used in this study (leaf nodes).

4.1.4 Simplifying Assumptions

To reduce complexity and make the classification problem more tractable, some simplifying assumptions have been made. The first important simplifying assumption is the assumption of independence between the set of features as well as the set of labels (drugs). It may be the case that some features are bound to one or more others by some kind of dependence relationship, but this is not explicitly modelled in the current formulation of the problem.

Further, it is known from the clinical literature that resistance to one drug often implies resistance to another, known as cross-resistance [4]. This is due to the fact that the drugs are members of just a few distinct classes that have similar mechanisms of action. These complicating realities have been abstracted away by the simplifying assumption that resistance to each drug is independent of any other.

An additional simplifying assumption is made regarding the determination of the drug resistance profile for each patient. The resistance profiles used in this study are determined using only one algorithm (described in section 4.1.2). In reality, the Africa Centre treatment failure clinic uses the cumulative score of three different algorithms and considers 5 degrees of resistance per drug (as opposed to just *resistant* or *susceptible*).

The final simplifying assumption made is that the result of the genotype resistance test is considered to be the absolute truth. However, this is not the case and there are in fact many limitations to GRT, such as the fact that it can only detect majority quasispecies within a virus population and can also not detect virus "reservoirs", which are mutations archived in the cells [3]. Some studies report that GRT is only 60%–65% predictive of response to ART [8].

4.1.5 Summary

A preliminary examination of the clinical database was performed and experts in the field of HIV patient care were consulted in order to determine which data features should be used to train the predictive models considered in this study. The approach to constructing the training dataset and selection of the classification techniques to be tested has been discussed and problem design considerations and simplifying assumptions were illuminated.

4.2 Processing Pipeline

In order to establish an experimental environment that was both robust and repeatable, the decision was taken to develop an experiment pipeline that is (mostly) automated. This was achieved using a combination of Python[5] and Bash[6] "glue" code in conjunction with the machine learning libraries and evaluation tools used for experimentation.

4.2.1 Dataset Construction

The dataset construction phase of the processing pipeline (discussed in section 4.3.2 and illustrated in figure 4.2) is executed only once per clinical dataset. It has therefore been executed exactly twice during this study – once for dataset 1 and once for dataset 2.

[5] http://www.python.org/

[6] http://www.gnu.org/software/bash/

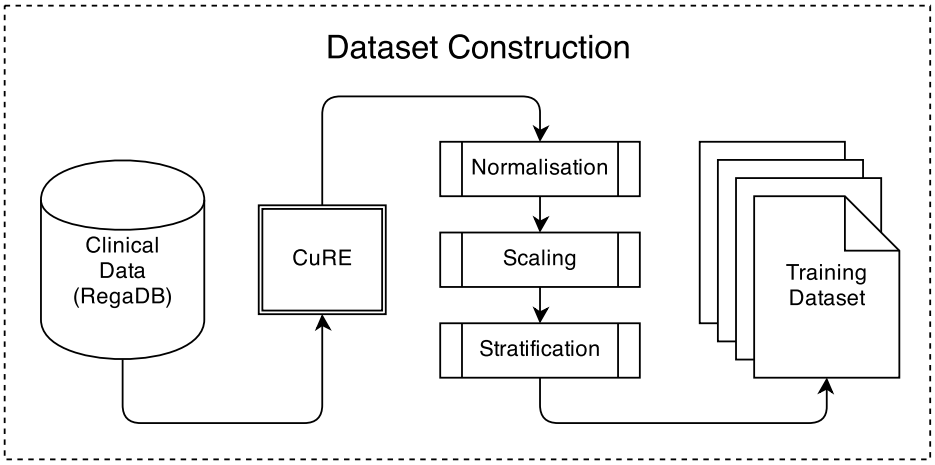


Figure 4.2. Dataset construction phase of the processing pipeline.

The RegaDB application [11] (including source code) has been downloaded from the RegaDB website and the application installed into the test environment. The Africa Centre database was then restored in order to establish an exact clone of the production system.

RegaDB Setup

RegaDB is a Java web application that runs inside the Tomcat Servlet container using the Postgres database management system for storage. The Spring MVC web framework[7] is used to serve content and JWT[8] is the user interface library. Hibernate[9] is used as the persistence framework. The RegaDB application is represented by the cylinder labelled *Clinical Data* in figure 4.2.

Data Extraction

The initial step of the dataset construction phase of the processing pipeline is to invoke the custom developed tool called CuRE (shown as the square box in figure 4.2). The tool was developed in order to extract data

[7] http://spring.io/guides/gs/serving-web-content/, **MVC** - *Model/View/Controller*

[8] http://www.webtoolkit.eu/jwt

[9] http://www.hibernate.org/

from the RegaDB database in the appropriate format and in a configurable and repeatable way. The design and implementation of the tool is discussed in detail in section 4.3.2. The tool is invoked via the web interface provided and results in an ARFF file or spreadsheet containing the data in tabular format being downloaded onto the filesystem of the user.

Dataset Preparation

As illustrated in figure 4.2, there are three final steps that follow data extraction by the CuRE application. These three processes comprise the dataset preparation phase and have been shown to be an important part of the classification procedure [86]. Each individual process is discussed below.

Normalisation Before the multi-label classification algorithms are used to train the models, the dataset is normalised. This involves converting categorical features to numeric values. This is necessary because some classification methods (such as support vector machines) only support feature vectors that consist only of numeric values.

As an example, if there is a categorical feature that could have any one of the values $\{red, green, blue\}$, this would be mapped to three numeric features. The value of the three numeric features would be $\{1,0,0\}$ if the original feature value was *red*, $\{0,1,0\}$ for *green* and $\{0,0,1\}$ for *blue*.

The Weka `Explorer` graphical tool was used to generate the normalised dataset. The `Explorer` tool is bundled with the Weka application and allows the user to import an ARFF file and apply *filters*. Filters are used to apply various modifications to dataset files, and in this case, the `NominalToBinary` filter was used to transform categorical features into multiple features with values in $\{0,1\}$ (i.e. binary features) as described in the above example.

The clinical data used in this study only contained numeric and categorical values, but if other data types, such as image, video or audio data existed, then the normalisation step would also include mapping these data to numeric values. The conversion of the categorical features resulted in the feature count increasing from 62 to 125.

Scaling The second data preprocessing step involved in constructing the training dataset is scaling. Scaling entails mapping all numeric values to the same range (in this case $[-1;1]$). There are two primary reasons for doing this. First, having all features in the same range eliminates the possibility of features with greater numeric values dominating in the training process and thus having a larger influence during the classification step. Secondly, using a range of $[-1;1]$ helps prevent possible calculation errors associated with computation, such as integer overflow in the case of calculating the product of two very large numbers.

Like normalisation, scaling was done using the Weka `Explorer` graphical tool. The dataset filter that was applied in this case was the `Normalisation` filter. The `Normalisation` filter takes two parameters, the *scaling factor* and the *translation value*, both of which are numeric. The scaling factor specifies the numeric range to which the features should be mapped and the translation value specifies the starting point of the numeric range as an offset from zero. For example, if a scaling factor of 1.0 and a translation value of -1.0 were specified, then all features would be mapped to the range $[-1;0]$ by the Weka `Normalisation` filter. In this study a scaling factor of 2.0 and translation value of -1.0 was used, resulting in all numeric features being mapped to the range $[-1;1]$.

Stratification The last data preparation step that must be performed before the dataset can be used for the evaluation of the multi-label classification techniques is stratification. Stratification in this context is the process of generating the dataset folds used for cross validation while maintaining the label distributions of the complete dataset. This is done for the purposes of mitigating the problem of sampling bias and reducing the variance of the evaluation results obtained.

Since the dataset used in this study is small (by machine learning standards), the Mulan implementation of the iterative stratification algorithm [79] was used to generate the 10 folds to be used for cross validation. A small Java utility, discussed in section 4.3.3 was developed to execute this task.

4.2.2 Experiment Loop

After the data has been extracted and the dataset prepared, the experiments can be conducted. The experiment loop phase of the processing pipeline comprises of three steps, namely, cross validation, results processing and evaluation measure calculation. A single execution of the experiment loop is used to generate the performance analysis results for each of the seven multi-label classification experiments conducted.

In order to reduce the possibility of human error, the experiment loop phase of the processing pipeline is almost entirely automated. As a result, the experiments are easily repeatable, which helps increase confidence in the results. Due to the ease of conducting the experiments, each experiment is performed numerous times and the results compared each time to ensure that no errors have occurred.

Cross Validation

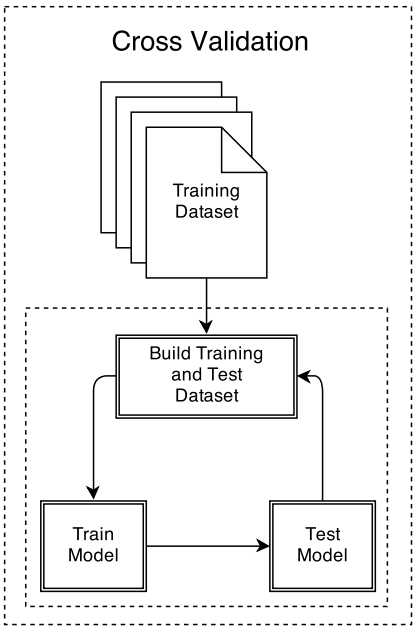


Figure 4.3. Cross validation phase of the experiment loop.

The cross validation process loop, illustrated in figure 4.3, is nested inside the experiment loop, since cross validation is performed as part of each experiment. This study makes use of 10 fold cross validation, which means that the cross validation loop is executed exactly 10 times per experiment. In other words, training and testing are performed 10 times for each classification technique, each time using the different training and test datasets constructed during the stratification phase described in section 4.2.1. The scripts developed to automate this process are discussed in section 4.3.4.

Results Processing

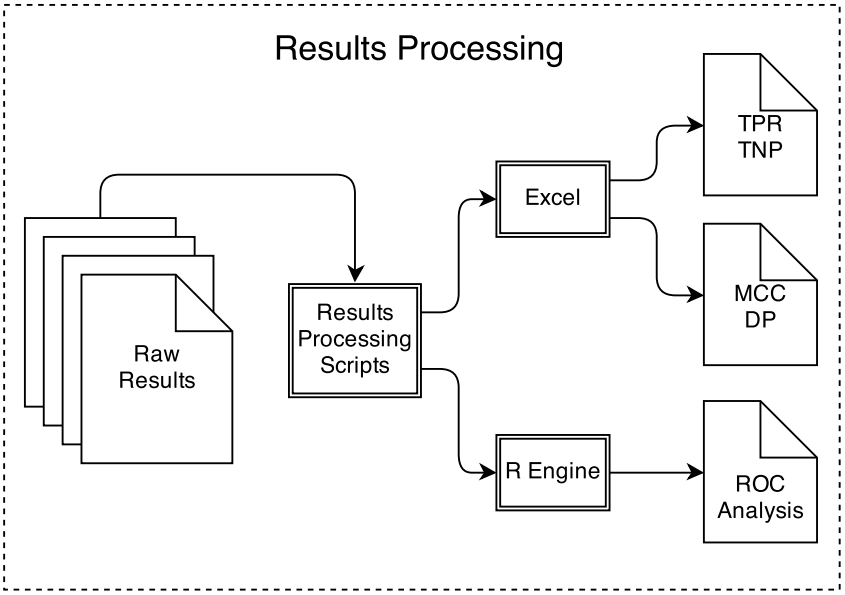


Figure 4.4. Results processing phase of the experiment loop.

The results processing step of the experiment loop is illustrated in figure 4.4. The result of the cross validation loop is always a set of raw results from each experiment. These results are in the form of plain text files that contain the output of the classifications of the test examples. Every method except PCT produces the list of label set predictions (Z_i) that were made for each test example ($\mathbf{y}_i$). An example of this output is shown in listing 4.1. Note that the predictions produced are probabilities that the given label

```
[1,1,1,1,1,1,1,1,1,1,0]:[0.00,0.63,0.72,0.02,0.84,0.00,0.99,0.00,0.91,0.72,1.07]
[1,1,1,1,1,1,1,1,1,1,1]:[1.00,0.98,1.00,1.00,0.65,1.00,0.59,1.00,0.97,1.00,0.85]
[1,1,1,1,1,1,1,1,1,1,1]:[0.29,0.35,0.96,0.48,0.12,0.29,0.09,0.29,0.10,0.96,0.08]
[1,1,1,1,1,1,1,1,1,1,1]:[0.01,0.75,0.75,0.08,0.26,0.01,0.67,0.07,0.27,0.75,0.07]
[1,1,1,1,1,1,0,1,1,1,1]:[1.00,0.93,0.99,0.79,0.82,1.00,0.66,0.99,0.77,0.99,0.67]
[1,1,1,1,1,1,0,1,1,1,1]:[0.52,0.94,0.99,0.79,0.91,0.52,0.78,0.76,0.86,0.99,0.86]
[1,1,1,1,0,1,0,1,1,1,0]:[0.23,0.29,0.99,0.42,0.41,0.23,0.10,0.33,0.14,0.99,0.26]
[1,1,1,1,1,1,0,1,1,1,1]:[1.00,0.02,0.99,1.00,0.99,1.00,0.00,0.99,0.63,0.99,0.99]
[1,1,1,1,1,1,0,1,1,1,1]:[0.20,0.44,0.97,0.40,0.11,0.20,0.07,0.16,0.08,0.97,0.05]
[1,1,1,1,1,1,0,1,1,1,1]:[0.80,0.88,0.99,0.64,0.44,0.80,0.37,0.91,0.58,0.99,0.27]
```

Listing 4.1. Example output from the MEKA and Mulan libraries. The values left of the colon represent the actual labels (1 implies that the label is associated with the example) and the values to the right are the predicted probabilities that the labels are associated with the example. The order of the labels is taken to be the same as the input file.

```
feature: stavudine
 REAL\PRED | 1 |  0 |
 ----------------------
         1 | 4 | 12 | 16
         0 | 3 | 31 | 34
 ----------------------
           | 7 | 43 | 50
 Accuracy: 0.7
```

Listing 4.2. Example confusion matrix from Clus output. This confusion matrix enumerates the prediction results for a single label (stavudine) for a single training and test cycle. The accuracy shown is simple accuracy as defined in equation 2.1.

is associated with the example. Further, the label ordering is the same as the ordering given in the input (training and test) dataset files.

Instead of a list of label set predictions, the PCT method outputs the learnt tree and decision rules along with the confusion matrices for each of the labels. An example of a confusion matrix that is output by the Clus tool [39] (used for experimenting with the PCT method) is shown in listing 4.2. This confusion matrix is for the *stavudine* label and has been generated using the real and predicted values from just one training and test cycle, in this case from dataset 2, which is why there have been a total of 50 predictions made (a single fold in 10 fold cross validation of a dataset with 500 examples consists of 50 examples).

Results Processing Scripts In order to generate the evaluation measures from the raw results in a reliable and repeatable manner, a set of results processing scripts were developed in Python[10]. Specifically, three scripts were written – two for processing the output from the MEKA and Mulan libraries (shown in listing 4.1) and one for processing the results from Clus (shown in listing 4.2). The purpose of these scripts is to transform the raw results obtained from the seven experiments into a format that can be used as input by the tools that have been used to calculate the evaluation measures. Details of the scripts are given in section 4.3.4.

Evaluation Measure Calculation

The final step in the experiment loop is the calculation of the evaluation measures. This step was fully automated and executed at least once per multi-label classification experiment. The process involves using the output of the results processing scripts as input to the evaluation measure calculation tools. The two tools that were used to calculate the final evaluation measure values were Microsoft® Excel and the R engine. R Studio[11] was used as the IDE for working with the R engine.

Microsoft® Excel The Excel spreadsheet application developed by Microsoft® is a popular data manipulation tool. In this study, the tool is used to calculate the values for the confusion matrix-based and higher order evaluation measures. A simple template spreadsheet has been created and preprogrammed with the formulae required to calculate the four scalar evaluation measures. To calculate the values for an individual experiment, all that is necessary is to import the cumulative confusion matrices generated by the results processing scripts into the template spreadsheet. The values for TPR, TNR, MCC and DP are instantly automatically calculated for each label.

[10] Script source provided on disc accompanying dissertation.

[11] http://www.rstudio.com/

R Engine The popular statistical analysis tool R was used to perform *ROC* analysis. This analysis resulted in two outputs – the *ROC* curves and the *AUC* value associated with each curve. The results processing scripts have been used to transform the outputs generated by the multi-label classification experiments into R commands. In other words, R scripts are automatically generated from the results processing scripts and all that is required to generate the *ROC* curves and *AUC* values is to execute these scripts using the R engine. The scripts execute commands provided by the ROCR package [92]. Vertical averaging is used to calculate the average true positive rate of the cross validation cycles [61].

4.2.3 Summary

The complete experimental pipeline has been described. The three phases of the pipeline, namely dataset construction, cross validation and results processing have been discussed in detail. The pipeline has been shown to be almost completely automated, which reduces the risk of human error and therefore increases the reliability of the results. Figure 4.5 illustrates the complete processing pipeline, showing each step of the process at a high level.

4.3 Software Tools

A number of applications were used during the course of the investigation conducted in this study. Many existing tools and libraries were used, but a significant amount of custom development was done, specifically for the purposes of data extraction and experimental pipeline automation.

4.3.1 Existing Applications

The patient data that is used for clinical management is stored in RegaDB, an open source patient-centric clinical data management system that stores data related to HIV treatment [11]. The data is stored longitudinally in a relational database (in this case using the Postgres[12] database management system). This

[12] http://www.postgresql.org/

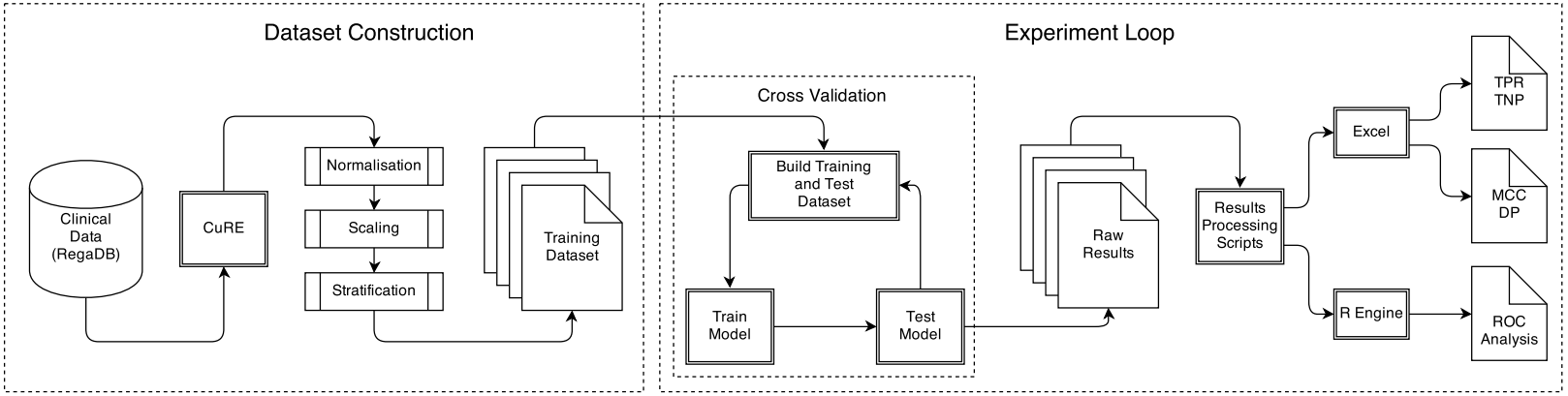


Figure 4.5. Processing pipeline. The dataset construction involves using the custom developed tool to extract the longitudinal data from the RegaDB relational database and applying the normalisation, scaling and stratification processes to produce a dataset ready for cross validation. The experiment loop involves executing the cross validation cycles, processing the raw results and using the Excel template and R engine to produce the evaluation measures.

application is the core tool used to manage the data associated with patients that attend the Africa Centre treatment failure clinic and thus any tool used to enable this research needs to be able to integrate with RegaDB.

Many libraries and tools exist to facilitate machine learning research, but not many of those include implementations of the multi-label classification techniques being investigated in this study. Furthermore, in order to design a system that could potentially be retrofitted into the existing treatment failure management processes (and therefore have an impact in the field), the selected tools should use technologies compatible with RegaDB. The Java machine learning toolkit *Weka* [13] was selected for use along with its multi-label learning extensions – MEKA and Mulan, since it uses the same technology stack and can therefore easily run alongside and interoperate with RegaDB. The Clus tool was made use of for the investigation of the PCT technique, as it is also developed in Java.

The tools selected to calculate the evaluation measures used to assess the performance of the classification techniques are Microsoft® Excel and the R engine [14]. Excel is a commonly used spreadsheet application with an intuitive user interface as well as the ability to perform the required statistical analysis in order to generate the values for TPR, TNR DP and MCC. In order to perform the *ROC* analysis, a more sophisticated tool was required. The R engine, with the addition of the ROCR module, allows the user to generate the *ROC* curves for visual performance analysis and can also generate the scalar AUC values.

4.3.2 CuRE

A custom tool called *CuRE* (*Cu*stom *R*egaDB *E*xporter) was developed[13] to extract data from the RegaDB relational database and build each of the features in the feature vector and produce a flat file in a format compatible with the machine learning libraries and tools used. The design of the tool is illustrated in figure 4.6. The paragraphs below discuss each component.

[13] Source code available online at https://github.com/psbrandt/cure

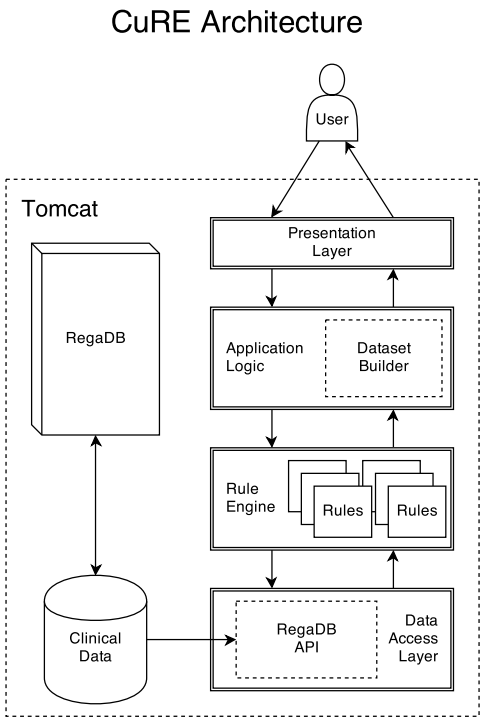


Figure 4.6. Architecture of the custom developed *CuRE* tool.

Architecture

CuRE has been designed in a way that allows retrofitting into an existing clinical management technical environment that uses RegaDB. The reason for this design is to increase the potential usability (and thus impact) of the tool. If the tool required infrastructure and expertise not already in place, this would increase the barrier to implementation.

The tool is built using the same software technologies and frameworks as RegaDB (Java, Spring and Hibernate), which means that it can be installed into the same Servlet container (Tomcat). This implies that there are no additional skills or dependencies required (over and above those required for RegaDB itself) in order to use the tool. User interaction with the custom tool is very similar to using RegaDB. The user interface is also browser-based and the tool URL is almost identical to the RegaDB URL – only the suffix is different.

Presentation Layer

The user interface of the custom tool is written in HTML, CSS and JavaScript, using the popular Bootstrap[14] frontend framework. The user is presented with a list of 62 check boxes, one for each possible feature available for use in the feature vector. The user may select any or all of the checkboxes, thereby configuring which features are going to be included in the dataset, and then choose to generate a dataset export in either ARFF or Excel format.

Application Logic

This layer contains most of the business logic of the application. It handles the processing of web requests, constructing the list of features to be generated (as configured using the user interface) as well as the generation of the dataset in a flat format from the relational database.

In order to build the dataset, the dataset builder uses Weka's internal `Instances` data structure. For each training example a Weka `Instance` object is constructed and added to the set of `Instances`. Each `Instance` has a set of features (called `Attributes` in the Weka nomenclature), each of which are constructed by the rule engine (see below).

The export functionality of the dataset builder consists of two Java Servlets, one for each output format. The Servlet that generates the (human readable) Excel export makes use of the Apache POI[15] library to generate the Microsoft® Excel document. The ARFF file generation Servlet uses Weka's internal ARFF file construction mechanism.

[14] http://getbootstrap.com/

[15] http://poi.apache.org/

Rule Engine

The rule engine provides a framework at the API level to build the rules that specify how to construct each individual feature of the training examples. The rule engine has been built using the *factory method* design pattern. The rule factory and rules are used to construct the features of the training examples in preparation for the dataset builder to generate the export file.

Rule Factory The rule factory defines an abstract rule class (as shown in listing 4.3) that rules need to extend in order for the rule engine to generate the associated features. Each rule has a `setAttributeValue` method that is used to set the value of a specific feature for a specific patient. This is done by extracting the required data via the data access layer (described below) and, if necessary, performing any additional calculations before setting the value of the feature.

Rules There are four types of rules that extend the abstract class provided by the rule factory. Each feature in the training examples is built using exactly one implementation of one of the four types of rules.

Simple Numeric A simple numeric rule sets the feature by simply reading a numeric value from the RegaDB database and setting the feature to that value. For example, *patient age* would be a simple numeric rule.

Simple Categorical Simple categorical rules are used for copying categorical values from the RegaDB database into the training example features. An example of such a feature would be the *yes/no* feature *partner on treatment*.

```java
/**
 * Abstract rule
 */
@Component
public abstract class Rule implements ApplicationContextAware {
    protected Logger logger = LoggerFactory.getLogger(this.getClass());
    protected ApplicationContext context;

    protected RuleService ruleService = null;

    public Rule(RuleService rs) {
        ruleService = rs;
    }

    // Method that must be implemented for a rule to be instantiated
    public abstract void setAttributeValue(Attribute attribute, Instance instance,
                                           Patient patient);

    public void setApplicationContext(ApplicationContext context) throws
        BeansException {
        this.context = context;
    }
}
```

Listing 4.3. The rule abstract class that much be extended for each non-trivial patient feature.

Custom All derived features are built using custom rules. This is due to the fact that there is always custom processing that needs to be done when constructing a derived feature. For example, the rule that constructs the *recent CD4 gradient* feature finds all the CD4 counts for a patient in the last 12 months, then fits a line through these measurements (using time as the x-axis and the CD4 value as the y-axis) and sets the value of the feature the gradient of this line.

It is possible to implement any logic that can be expressed in Java in a custom rule implementation. Any amount of patient data can be extracted using the RegaDB API and any operations can be applied to these data as long as the result is either a categorical or numeric feature value.

Label The label features are also built using rules. The label rules extract the resistance profile from the database (as determined by the HIVDB 6.0.5 algorithm [5]) and set the feature values accordingly.

Data Access Layer

Accessing the data in the RegaDB database is achieved using the RegaDB data API. More specifically, the data access component within RegaDB was extracted and used in the CuRE tool in order to retrieve data from the relational database in the exact same manner as RegaDB does internally. It is important to note that data is only read and never written. This is done to maintain data integrity of the clinical data. The role of the data access layer is to convert the data objects used internally by RegaDB into objects that can be manipulated by CuRE.

RegaDB API On examination of the RegaDB source code, it became evident that the application is built using a modular architecture and thus consists of several components. One of these components is the persistence module, which contains the Hibernate mapping classes used to work with the data in the application layer. This module was reused in the custom tool to access the RegaDB data.

An additional utility module from RegaDB was also reused, which enables data access at a higher level. For example, the utility modules enable accessing all clinical measurements from a patient without having to access data from all the individual tables. This is analogous to using the RegaDB service layer.

4.3.3 Stratifier

In addition to CuRE, a small Java utility, `IterativelyStratifyMultiLabelDataSet`[16], was written in order to accomplish the task of generating the stratified folds used for cross validation. The utility makes use of the Mulan library in order to generate the 10 folds, but takes the algorithm one step further in order to construct actual training and test dataset files using the folds.

The code to perform the stratified dataset file construction is shown in listing 4.4. This code generates 10 pairs of files (named as specified in the caption of listing 4.4), corresponding to the training and test datasets for each cross validation fold.

[16] Source code provided on disc accompanying dissertation.

```java
/**
 * Create files to use for cross validation based on the folds
 */
public void createCrossValidationFiles(MultiLabelInstances[] folds) {

    // Create numbered folds
    for(int i = 0; i < folds.length; i++) {
        System.out.println("Generating files for fold " + i);

        // The testing instances for this fold
        Instances test = new Instances(folds[i].getDataSet());

        // The training instances for this fold
        Instances train = null;

        for(int j = 0; j < folds.length; j++) {

            // Ignore test examples when creating the training dataset
            if(j != i) {
                if(train == null) {
                    // Initialize the training dataset
                    train = new Instances(folds[j].getDataSet());
                } else {
                    // Add all other non-test instances to dataset
                    Instances foldDatset = folds[j].getDataSet();

                    for(int k = 0; k < foldDatset.size(); k++) {
                        train.add(foldDatset.get(k));
                    }
                }
            }
        }

        // Create the ARFF files
        writeFile(test, i, true);
        writeFile(train, i, false);
    }
}
```

Listing 4.4. The method in the `IterativelyStratifyMultiLabelDataSet` utility that constructs the training and test datasets to be used in cross validation. The `writeFile` method creates the dataset files on the filesystem with meaningful names. For example `dataset1_fold_X_test.arff` and `dataset1_fold_X_train.arff` where `X` is the fold number and is in the range $[1;10]$.

4.3.4 Custom Scripts

The following paragraphs describe the numerous scripts developed in order to fully automate the experimental pipeline.

Cross Validation

A script was written in Python[17] (shown in listing 4.5) that automates the execution of cross validation for each experiment. Listing 4.6 shows an example of how the cross validation process can be invoked from the command line. Note the `XX` placeholders, which are replaced with the fold numbers during execution. Also note that any of the multi-label classification techniques can be executed by the script, since the execution command is passed to the script as a command line parameter. The output of executing the cross validation script in this way will be ten files with names of the form `BR_NB_fold_XX.out.txt` (using naive Bayes as an example), but in each case the `XX` placeholders will be replaced by a folder number. These are the files containing the raw results of the experiment.

Something to note is that it is not necessary to train and test the model in separate steps. This is due to the fact that the machine learning libraries and tools used for experimentation in this study are designed to train and test models in the same thread of execution (although, importantly, they also support saving models for later use). This can be seen from the example cross validation execution shown in listing 4.6. Another example of this is shown in listing 4.7, which shows the configuration required for the Clus application to train and test a predictive clustering tree.

Results Processing

Scripts were written in Python for processing the raw results[19]. Separate scripts were written for producing the scalar evaluation measures for Clus and for MEKA/Mulan. An additional script was written for producing the R commands necessary to generate the *ROC* curves and *AUC* values.

[17] Source code provided on disc accompanying dissertation.

```python
#!/usr/bin/python
import os
import sys

# Display help if not enough parameters are supplied
if len(sys.argv) != 2:
    print "Usage: ./cross_validate.py command_line_with_XX_as_place_holder"
    exit(1)

# Initialise the command
command = sys.argv[1]

# Generate and execute the command for each fold
for i in range(10):

    # Replace XX with fold number
    fold_command = command.replace("XX", str(i))

    # Run command
    os.system(fold_command)
```

Listing 4.5. Cross validator script. This script will run the given command once for each cross validation fold, each time modifying the training and test set filenames using the correct fold number.

Confusion Matrix-Based and Higher Order Measures The Clus results processing script is used to the generate the two confusion matrix-based and two higher order evaluation measures for the PCT method. Since the PCT technique does not produce probabilistic predictions, it is not possible to generate *ROC* curves (and hence *AUC*) for this method. The output required in order to generate the four evaluation measures for each label is the *cumulative confusion matrix* (the sum total of the confusion matrices for each of the ten cross validation folds) for each label. The script generates this matrix for each label by parsing the output file generated by each each training and test cycle and building an in-memory confusion matrix for each

```
./cross_validate.py "java -cp lib/*:lib/*.jar weka.classifiers.multilabel.BR \
-t dataset1_fold_XX_train.arff -T dataset1_fold_XX_test.arff \
-f BR_NB_fold_XX.out.txt -W weka.classifiers.bayes.NaiveBayes"
```

Listing 4.6. Example execution of cross validation for an experiment.

```
[Data]
File = dataset1_fold_1_train.arff
TestSet = dataset1_fold_1_test.arff

[features]
Target = 126-136
Clustering = 126-136
Descriptive = 1-125

[Tree]
FTest = 0.1
Heuristic = VarianceReduction
ConvertToRules = Leaves
PruningMethod = None

[Ensemble]
EnsembleMethod = RForest
```

Listing 4.7. Clus configuration file for a single fold (in this case fold 1).

label for every fold. Finally, simple matrix addition is performed and the output is echoed to the command line (which could be redirected to a file at the user's discretion).

The confusion matrix-based and higher order measures are calculated in much the same way for the MEKA and Mulan library outputs as they are for the Clus outputs. Only one additional step needs to be performed, which is the calculation of the confusion matrix for each label for each fold. This is necessary because the MEKA and Mulan outputs do not contain the confusion matrices, but rather the actual predictions made. The confusion matrix is calculated by using an in-memory matrix, parsing each prediction made and simply incrementing the value of the appropriate cell in the matrix depending on the outcome of the prediction.

ROC Analysis In order to generate the *ROC* curves and *AUC* values, the results from the methods that support the calculation of these evaluation measures (BR-NB, HOMER, MLkNN, RAkEL and ECC) must be transformed into a format that can be used as input by the *ROC* analysis tool. Note that SVMs do not produce probabilistic predictions and thus *ROC* analysis cannot be performed for the BR-SVM technique.

The R engine (using the ROCR package) has been used to generate the *ROC* curves and calculate the *AUC* values. Therefore, the scripts that have been developed transform the raw results into commands that can be understood by the R engine. This involves adding the predictions to the data structure that is required by the ROCR package and generating the R commands to do the analysis, plot the curves and produce the *AUC* values.

4.4 Model Parameterisation and Implementation

Before the machine learning techniques can be used to train and test the associated predictive models, it is first necessary to set the values for any parameters that may be required. The process is referred to as model parameterisation and is discussed below for each of the techniques investigated. Also discussed are the implementations of each technique used.

4.4.1 Binary Relevance (BR)

The binary relevance method itself does not require any parameterisation, but the base classifiers used in this study (SVM and naive Bayes) do require parameter values.

Support Vector Machine (BR-SVM)

The *radial basis function* (RBF) SVM kernel was selected because it is able to model non-linear relationships. Before the RBF kernel can be used, it is first necessary to select optimal values for the parameters C and γ in equation 3.2 on page 35. The recommended method to select optimal parameters is to first perform a coarse-grained search across the entire parameter space to identify an area of interest and then do a fine-grained search of the identified area [21, 86], which is precisely the process that was followed.

A broad grid search was performed over the values $2^{-15}, 2^{-13}, \ldots, 2^{3}$ for γ and $2^{-5}, 2^{-3}, \ldots, 2^{15}$ for C as recommended. A smaller area that gave good results was scrutinized more closely, performing 10 fold cross validation for each (C, γ) tuple (using simple accuracy as defined in equation 2.1) and it was determined that the best RBF kernel parameters are ($C = 4.5$, $\gamma = 0.0$).

The Weka wrapper implementation of LibSVM [93,94] was used along with the binary relevance implementation provided by MEKA. The BR-SVM classification technique was invoked by the custom developed cross validation script in a manner very similar to the invocation shown in listing 4.5.

Naive Bayes (BR-NB)

The only parameterisation step required before the naive Bayes classifier can be used to generate predictions is to specify how the probability distribution of the continuous features are modeled. In this study the normal (Gaussian) distribution as shown in equation 4.1 is used.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.1}$$

The Weka implementation of the naive Bayes classifier has been used and the BR-NB technique was invoked in the same way as BR-SVM – using the cross validation Python script shown in listing 4.5.

4.4.2 HOMER

In order to use the HOMER technique, a base classifier, a value for k and a clustering method must be selected. The naive Bayes classifier was selected as the base classifier and the balanced k means clustering algorithm has been used with a value of $k = 3$. The Mulan implementation of the HOMER algorithm has been used and the cross validation Python script was again used to invoke the technique from the command line.

4.4.3 MLkNN

The two parameters that need to be selected when using the MLkNN technique are the number of nearest neighbours to consider (k) as well as the smoothing parameter (s). In this study, $k = 10$ was selected as the number of neighbours, since this has previously been shown to be a reasonable choice [37]. A value of 1 was selected for s, which results in so-called *Laplace smoothing*.

Further, the Euclidian distance function has been selected for calculating the nearest neighbours. Given the unseen example $\mathbf{y} = (y_1, y_2, \ldots, y_p)$, the Euclidian distance function is defined as

$$\mathscr{E}_D = \sqrt{(x_{i1} - y_1)^2 + (x_{i2} - y_2)^2 + \cdots + (x_{iP} - y_p)^2} \tag{4.2}$$

where $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip})$ is potentially one of the k nearest neighbours.

The Mulan implementation of the MLkNN algorithm has been used and cross validation is performed using the Python script show in listing 4.5 on page 73.

4.4.4 Predictive Clustering Trees (PCT)

Two important factors to consider when constructing the trees are the cluster splitting and stopping criteria. The split method used in this investigation is the *variance reduction* heuristic. The PCT technique makes use of the *f-test* as the stopping criterion.

The Clus[18] implementation of PCT is used and cross validation is initiated using a simple shell script[19] that executes the ten configuration files (one for each of the cross validation folds) sequentially.

[18] http://dtai.cs.kuleuven.be/clus/

[19] Source code provided on disc accompanying dissertation.

4.4.5 RAkEL

The number of ensemble members (m), the size of the label sets (k) as well as the threshold (t) are required user parameters for using the RAkEL technique. The values used in this study are $m = 10$ and $k = \frac{|L|}{2}$. Since there are 11 labels in the dataset (i.e. $|L| = 11$), and k must be a natural number, $k = 5$ is used. As recommended, the threshold value used is $t = 0.5$ [35].

Since the RAkEL technique is independent of the underlying base classifier, any existing classifier may be used. In this study, the Weka implementation of the naive Bayes classifier is used. The RAkEL implementation in Java provided by the Mulan library is used and cross validation is invoked using the developed Python script.

4.4.6 Ensemble of Classifier Chains (ECC)

The parameters necessary to apply the ECC method are the number of models (m), the threshold (t) as well as the base classifier to use. In this study a naive Bayes base classifier is used and the number of models in the ensemble is chosen to be $m = 10$. The default threshold of $t = 0.5$ is used to determine how many votes are required for a label to be predicted as present. A value of 0.5 means that at least half of the ensemble members have to vote for a label for it to be considered as being associated with the unseen example.

The Mulan implementation of ECC is used, but a small modification had to be made to the MEKA source code enabling the ECC algorithm to be invoked from the command line. Cross validation is invoked using the Python script as with the methods discussed above.

4.5 Summary

The approach taken to formulate the problem in a way that could be systematically investigated has been discussed in detail. The preliminary data analysis results were used to inform the decisions regarding the

techniques to be investigated as well as the selection of evaluation measures that would produce reliable performance analysis results. A review of the literature and numerous consultations with domain experts resulted in the development of a set of features that can be considered the relevant information used for the management of patients failing HIV therapy. This relevant information has been used as input data to train the classification models.

The tools used for experimentation have been discussed and the details given for all custom developed components. The parameterisation of the multi-label classification techniques was also described and the mechanisms used to invoke the specific software tools have been mentioned. The design and implementation of a robust experimental pipeline was detailed and the importance of being able to achieve repeatable results free from human error and dataset bias was emphasised.

Chapter 5

Results

The results of the experiments conducted using the methods described previously are given below. The results are presented in three main subsections. First, the dataset analysis is presented, followed by the scalar evaluation measures. The scalar measures include TPR, TNR, MCC, DP and AUC. The ROC analysis results are presented in the final subsection. In each of the three subsections, results are first presented for dataset 1 and then for dataset 2. A comparison and summary of the results from the two datasets is then given.

5.1 Dataset Analysis

The following subsections describe the two datasets used in this study. These two datasets represent the contents of the same system at two different points in time, roughly two years apart. The first dataset (referred to as dataset 1) is a snapshot of the clinical information system (RegaDB) used at the Africa Centre for managing patients failing antiretroviral therapy taken in mid 2011. The second dataset (dataset 2) is a snapshot of the clinical information system system taken in mid 2013.

5.1.1 Dataset 1

The first dataset comprised of patients that were seen by clinical care providers at the Africa Centre treatment failure clinic. Data was used from all patients that were captured in the system since the clinic's inception in early 2009 until June 2011. The number of patients in dataset 1 is 252 and these patients have a mean age of 37.49 years. There are 190 female patients in the dataset, comprising 75.4% of the total.

The final training dataset consists of 62 features per patient (as summarised in table 4.1 on page 51). Feature completeness is defined as:

$$Feature\ Completeness\ =\ \frac{Number\ of\ patients\ with\ a\ value\ for\ the\ feature}{Total\ number\ of\ patients} \tag{5.1}$$

where having a value for a feature means one of two things. For a so-called *simple* feature (one that is used directly from the clinical information system, such as *age* or *gender*), it means that a value exists in the clinical information system for that patient. For a derived feature such as *recent CD4 count gradient* or *baseline viral load*, it means that values exist in the clinical information system for all features that are used in the calculation of the derived feature.

The average feature completeness is 76.84% and 24 features (38.71%) were 100% complete. Twelve features (19.35%) were between 90% and 100% complete, and eight features (12.90%) were over 78% (but less than 90%) complete. 14 features (22.58%) were between 22% and 78% complete, and four features (6.45%) were less than 10% complete. The feature completeness for dataset 1 is summarised in table 5.1.

Table 5.2 shows, for each drug, how many patients are resistant and how many are susceptible. It can therefore be inferred from the table that resistance in dataset 1 is more common than susceptibility. Tenofovir, zidovudine and stavudine are the only drugs where susceptibility dominates over resistance, while didanosine and abacavir are relatively well-balanced. Patients are overwhelmingly more resistant than susceptible to the other six drugs (efavirens, emtricitabine, delavirdine, nevirapine).

Table 5.1. A summary of feature completeness for dataset 1.

Number of Features	Percentage Complete
24 (38.71%)	100%
12 (19.35%)	90% – 99.99%
8 (12.90%)	78% – 89.99%
14 (22.53%)	22% – 77.99%
4 (6.45%)	0% – 9.99%

Average feature completeness is 76.84%

Table 5.2. Resistance and susceptibility counts for each drug for dataset 1.

	efavirens	didanosine	emtricitabine	delavirdine
Resistant	229 (90.87%)	103 (40.87%)	217 (86.11%)	226 (89.68%)
Susceptible	23 (9.13%)	149 (59.13%)	35 (13.89%)	26 (10.32%)
	stavudine	**nevirapine**	**tenofovir**	**etravirine**
Resistant	93 (36.90%)	229 (90.87%)	61 (24.21%)	196 (77.78%)
Susceptible	159 (63.10%)	23 (9.13%)	191 (75.79%)	56 (22.22%)
	abacavir	**lamivudine**	**zidovudine**	
Resistant	123 (48.81%)	217 (86.11%)	77 (30.56%)	
Susceptible	129 (51.19%)	35 (13.89%)	175 (69.44%)	

Table 5.3. Number of patients resistant to multiple drugs for dataset 1.

Number of Drugs	0	1	2	3	4	5	6	7	8	9	10	11
Number of Patients	19	0	3	11	5	18	72	5	11	38	45	25
Percent (%)	7.54	0.00	1.19	4.37	1.98	7.14	28.57	1.98	4.37	15.08	17.86	9.92

There are 19 patients (7.54%) that show resistance to no drugs, which means that 92.46% of patients show resistance to at least one drug. Most commonly (in 28.57% of cases) patients are resistant to 6 drugs. There are 45 patients (17.86%) that are resistant to 10 drugs and 38 patients (15.08%) resistant to 9 drugs. A summary of how many drugs patients are resistant to is given in table 5.3.

The label cardinality (defined in equation 2.7) is 7.03, which can be interpreted to mean that the average patient is resistant to about 7 drugs. The label density (defined in equation 2.8) is 0.64, which is the proportion of the total number of drugs that the average patient is resistant to.

5.1.2 Dataset 2

The second dataset has a total of 500 patients. The additional 248 patients were added to the clinical information system between June 2011 and May 2013. Of the 500 patients, 362 (72.4%) are female and the mean age is 37.17 years.

The average feature completeness (as defined in equation 5.1) for dataset 2 is 78.29%. There are 22 features (35.48%) that are 100% complete and 20 (32.26%) that are between 81% and 99% complete. The majority (67.74%) of features in dataset 2 are greater than 81% complete. 10 features are between 45% and 79% complete and 10 features are less than 10% complete. Table 5.4 summarises the feature completeness for dataset 2.

The resistance and susceptibility counts for each drugs are given table 5.5. As in dataset 1, tenofovir, zidovudine and stavudine are the only drugs to which patients are overwhelmingly more susceptible than re-

Table 5.4. A summary of feature completeness for dataset 2.

Number of Features	Percentage Complete
22 (35.48%)	100%
20 (32.26%)	81% – 99.99%
10 (16.13%)	45% – 79.99%
6 (9.68%)	13% – 29.99%
4 (6.45%)	0% – 9.99%

The average feature completeness is 78.29%

sistant, while didanosine and abacavir are about evenly distributed. Patients are considerably more resistant than susceptible to efavirens, emtricitabine, delavirdine, nevirapine, etravirine and lamivudine.

The number of patients in dataset 2 that show no resistance to any drugs is 52 (10%). There are 143 patients (28.60%) resistant to 6 drugs and 77 (15.40%) and 78 (15.60%) patients resistant to 9 and 10 drugs respectively. This information is summarised in table 5.6. The label cardinality for dataset 2 is 6.83 and the label density is 0.62. Thus the average patient is resistant to 62% of the drugs, or 6.83 drugs.

5.1.3 Dataset Analysis Summary

The datasets used in this study to evaluate the multi-label classification techniques have been described. There are roughly twice as many patients in the second dataset, but the average age of the patients in dataset 1 and dataset 2 are almost identical (37.49 and 37.17 years respectively). About three quarters of the patients in each dataset are female.

In both datasets, 42 out of the 62 features (67.74%) are more than 80% complete. Dataset 1 has 24 features that are 100% complete compared to 22 in dataset 2. The average feature completeness of dataset 2

Table 5.5. Resistance and susceptibility counts for each drug for dataset 2.

	efavirens	**didanosine**	**emtricitabine**	**delavirdine**
Resistant	440 (88.00%)	203 (40.60%)	420 (84.00%)	435 (87.00%)
Susceptible	60 (12.00%)	297 (59.40%)	80 (16.00%)	65 (13.00%)
	stavudine	**nevirapine**	**tenofovir**	**etravirine**
Resistant	175 (35.00%)	440 (88.00%)	122 (24.40%)	382 (76.40%)
Susceptible	325 (65.00%)	60 (12.00%)	378 (75.60%)	118 (23.60%)
	abacavir	**lamivudine**	**zidovudine**	
Resistant	241 (48.20%)	420 (84.00%)	137 (27.40%)	
Susceptible	259 (51.80%)	80 (16.00%)	363 (72.60%)	

Table 5.6. Number of patients resistant to multiple drugs for dataset 2.

Number of Drugs	0	1	2	3	4	5	6	7	8	9	10	11
Number of Patients	52	0	5	15	11	32	143	9	32	77	78	46
Percent (%)	10.40	0.00	1.00	3.00	2.20	6.40	28.60	1.80	6.40	15.40	15.60	9.20

was 78.29%, which is slightly higher than dataset 1 at 76.84%. Both datasets only have 4 features less than 10% complete.

The resistance and susceptibility counts for dataset 1 and dataset 2 are very similar. In both datasets the same three drugs (tenofovir, zidovudine and stavudine) are the only drugs to which more patients are susceptible than resistant. The proportion of patients resistant to each number of drugs in both datasets is about the same. The biggest difference is that the number of patients not resistant to any drugs is 3% higher in dataset 2 than in dataset 1.

5.2 Scalar Evaluation Measures

Two sets of experiments were conducted, one for each dataset. The results are presented separately for each multi-label classification solution group. The results of the problem transformation methods are presented first, followed by the algorithm adaptation methods and then the ensemble methods. A comparison of the results obtained from the two datasets is then given, followed by a summary.

The scalar evaluation measures consist of TPR, TNR, MCC, DP and AUC, since these are the numeric values calculated from the raw output of the application of each multi-label classification technique. The five measures were calculated for each label, technique and dataset. The evaluation measures are presented below, first for dataset 1 and then for dataset 2.

5.2.1 Dataset 1

The scalar evaluation measures are given in the tables below for dataset 1. The values for each evaluation measure were generated as part of the processing pipeline (see section 4.2). More specifically, the generation of the numeric values from the raw experimental results was automated using the results processing scripts described in section 4.2.2, as well as the Excel template (see section 4.2.2).

Table 5.7. Scalar evaluation measures for the problem transformation methods for dataset 1. The highest value for each evaluation measure is shown in bold.

	BR-SVM				**BR-NB**					**HOMER**				
	TPR	**TNR**	**MCC**	**DP**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**
efavirens	1.00	0.00	-	-	0.86	0.26	0.10	0.43	0.54	0.89	0.13	0.02	0.09	0.51
didanosine	0.27	0.80	0.08	0.22	0.54	0.62	0.16	0.36	0.62	0.56	0.59	0.15	0.34	0.62
emtricitabine	1.00	0.00	-	-	**0.93**	0.20	0.15	0.63	0.61	**0.94**	0.17	0.16	0.70	0.57
delavirdine	1.00	0.00	-	-	0.83	0.23	0.05	0.22	0.56	0.84	0.19	0.02	0.11	0.51
stavudine	0.22	0.85	0.08	0.24	0.38	0.70	0.08	0.18	0.58	0.38	0.81	0.20	0.50	0.60
nevirapine	1.00	0.00	-	-	0.86	0.26	0.10	0.43	0.54	0.89	0.13	0.02	0.09	0.51
tenofovir	0.26	0.94	**0.27**	**0.92**	0.43	**0.86**	**0.31**	**0.85**	**0.65**	0.44	**0.84**	**0.29**	**0.80**	**0.66**
etravirine	**0.99**	0.02	0.03	0.31	0.88	0.16	0.05	0.17	0.60	0.52	0.52	0.03	0.08	0.50
abacavir	0.50	0.61	0.12	0.26	0.54	0.64	0.18	0.41	0.63	0.63	0.57	0.19	0.43	**0.66**
lamivudine	1.00	0.00	-	-	**0.93**	0.20	0.15	0.63	0.61	**0.94**	0.17	0.16	0.70	0.57
zidovudine	0.05	**0.96**	0.03	0.15	0.38	0.79	0.17	0.45	0.63	0.38	0.82	0.22	0.57	0.62
mean	0.66	0.38	0.10	0.35	0.69	0.45	0.14	0.43	0.60	0.67	0.45	0.13	0.40	0.57
std dev	0.40	0.44	0.09	0.28	0.23	0.27	0.07	0.21	0.04	0.23	0.30	0.10	0.28	0.06

BR-SVM - Binary relevance with support vector machine base classifier, **BR-NB** - Binary relevance with naive Bayes base classifier, **HOMER** - Hierarchy of multi-label classifiers, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

Problem Transformation Methods

The scalar evaluation measures are given for the problem transformation methods for dataset 1 in table 5.7. When $TPR = 1.0$ and $TNR = 0.0$ it is referred to as a *blanket positive prediction*. When a classification technique produces blanket positive predictions, it does not provide us with any useful predictive information. In table 5.7 the label with the best value for each evaluation measure is shown in bold (excluding blanket positive predictions).

It can be seen that the BR-SVM method produces blanket positive predictions for five labels (efavirens, emtricitabine, delavirdine, nevirapine and lamivudine). These five labels correspond to the five most imbalanced labels in dataset 1 ($> 86\%$ of patients are resistant to the corresponding drug). For all labels except tenofovir the method does not discriminate well, as seen by the low *DP* values in table 5.7. Overall, the BR-SVM method performs poorly, with only the performance of predicting tenofovir moderately above the performance of a random classifier based on the *MCC* value.

In table 5.7 it can be seen that switching the base classifier to naive Bayes drastically improves performance. There are no blanket positive predictions and the mean *DP* value increased by over 20%. The mean TPR and TNR values also increase. BR-NB had an *MCC* value of just over 0.3 for tenofovir, meaning that it is significantly better than a random classifier at predicting, in this case, absence of resistance to this drug. The mean *AUC* of all the drugs for the BR-NB method are 10% above that of a random classifier (0.5) at 0.6.

HOMER performed slightly worse than the BR-NB method on dataset 1, with the mean of each measure less than BR-NB. The exception was the mean TNR value, which did not change. HOMER outperforms BR-NB at predicting cases of lamivudine resistance, with a TPR of 0.94. The *MCC* value for the zidovudine indicates that HOMER performs better than a random classifier for this label. The mean *AUC* over all the labels for HOMER is 0.57, which is only 7% better than a random classifier.

Table 5.8. Scalar evaluation measures for the algorithm adaptation methods for dataset 1. The highest value for each evaluation measure is shown in bold.

	MLkNN					**PCT**			
	TPR	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**
efavirens	1.00	0.00	-	-	0.44	**0.97**	0.35	**0.39**	**1.56**
didanosine	0.08	0.83	-0.14	-0.51	0.45	0.47	0.78	0.26	0.62
emtricitabine	1.00	0.00	-	-	0.50	0.94	0.31	0.30	1.09
delavirdine	1.00	0.00	-0.02	-	0.52	**0.97**	0.31	0.36	1.45
stavudine	0.04	0.97	0.03	0.18	0.48	0.31	0.77	0.09	0.22
nevirapine	1.00	0.00	-	-	0.44	**0.97**	0.35	**0.39**	**1.56**
tenofovir	0.00	0.98	-0.06	-	0.52	0.28	**0.87**	0.18	0.55
etravirine	1.00	0.00	-	-	0.47	0.91	0.23	0.18	0.60
abacavir	**0.39**	0.63	0.02	0.04	**0.54**	0.56	0.70	0.26	0.60
lamivudine	1.00	0.00	-	-	0.50	0.94	0.31	0.30	1.09
zidovudine	0.01	**0.99**	**0.04**	**0.46**	0.53	0.19	0.83	0.04	0.11
mean	0.59	0.40	-0.02	0.04	0.49	0.68	0.53	0.25	0.86
std dev	0.48	0.47	0.07	0.41	0.04	0.32	0.26	0.12	0.52

MLkNN - Multi-label *k*-nearest neighbours, **PCT** - Predictive clustering trees, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

Algorithm Adaptation Methods

The scalar evaluation measures for the algorithm adaptation methods for dataset 1 are given in table 5.8. The best value for each evaluation measure is shown in bold (not including blanket positive predictions). MLkNN appears to be the worst performing of all the methods on dataset 1. It has the lowest mean value for each evaluation measure and blanket positive predictions for six labels (efavirens, emtricitabine, delavirdine, nevirapine, etravirine and lamivudine). Further, the mean AUC is 0.49, which indicates worse than random performance. This is confirmed by multiple MCC and DP values less than zero.

The PCT method does not suffer from blanket positive predictions and has the highest mean DP, MCC and TNR values for dataset 1. It has three MCC values above 0.35 and numerous DP values above 1. These values indicate that PCT is substantially better than a random classifier at predicting resistance to efavirens, delavirdine and nevirapine. PCT performs especially well relative to the other methods for efavirens and nevirapine with a TPR of 0.97 and DP value of 1.56 in both cases. Unlike the other methods, it does not appear to be a good predictor for tenofovir.

Ensemble Methods

Table 5.9 gives the scalar evaluation measures for the ensemble methods for dataset 1. The highest values for each measure are shown in bold. The RAkEL method suffers from no blanket positive predictions, but does have MCC and DP values below zero for two drugs, indicating very poor performance in these cases. RAkEL appears at first glance to perform relatively well for lamivudine, with a DP value of 1.06. However, the TNR for this drug is only 0.11, indicating that the method is not good at detecting negative examples. The RAkEL method has an average AUC of 0.58, which puts it between BR-NB and HOMER in terms of this measure for dataset 1.

The ECC method performs worse than a random classifier (based on the MCC values) for efavirens, nevirapine and etravirine, but has relatively high DP values (greater than 1) for emtricitabine, tenofovir and

Table 5.9. Scalar evaluation measures for the ensemble methods for dataset 1. The highest value for each evaluation measure is shown in bold.

	RAkEL					**ECC**				
	TPR	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**
efavirens	0.92	0.22	0.13	0.62	0.59	1.00	0.00	-0.02	-	0.68
didanosine	0.52	0.58	0.10	0.23	0.58	0.41	0.73	0.15	0.35	0.64
emtricitabine	0.93	0.20	0.16	0.67	0.60	1.00	0.03	0.09	1.02	0.62
delavirdine	0.96	0.04	0.01	0.05	0.60	1.00	0.04	0.19	-	0.69
stavudine	0.49	0.62	0.11	0.26	0.54	0.25	0.82	0.09	0.24	0.54
nevirapine	0.91	0.22	0.12	0.56	0.60	1.00	0.00	-0.02	-	0.61
tenofovir	0.38	**0.88**	**0.28**	0.82	**0.66**	0.39	**0.93**	**0.38**	**1.16**	**0.72**
etravirine	0.85	0.11	-0.05	-0.20	0.50	0.97	0.02	-0.02	-0.20	0.56
abacavir	0.42	0.74	0.18	0.42	0.60	0.59	0.70	0.28	0.65	0.68
lamivudine	**0.98**	0.11	0.19	**1.06**	0.60	**1.00**	0.03	0.09	1.02	0.60
zidovudine	0.26	0.82	0.09	0.27	0.55	0.26	0.89	0.19	0.58	0.63
mean	0.69	0.41	0.12	0.43	0.58	0.71	0.38	0.13	0.60	0.63
std dev	0.28	0.32	0.09	0.36	0.04	0.33	0.42	0.13	0.46	0.06

RAkEL - Random *k*-labelsets, **ECC** - Ensemble of classifier chains, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

lamivudine. Of the latter three, only tenofovir also shows an *MCC* value significantly greater than zero (0.38). The average *AUC* for the ECC method is 0.63 with an *AUC* value of 0.72 for tenofovir, making it the best performer for this drug and in terms of the *AUC* evaluation measure for dataset 1.

5.2.2 Dataset 2

The scalar evaluation measures are given for dataset 2 in tables 5.10 – 5.12 below. These measures were automatically generated by the results processing scripts and Excel spreadsheet template as part of the processing pipeline (see section 4.2). The values were generated in exactly the same way as for dataset 1, the only difference being the underlying clinical dataset used for training and cross validation.

Problem Transformation Methods

The problem transformation scalar evaluation measures automatically generated for dataset 2 by the processing pipelines are given in table 5.10. The highest value for each evaluation measure is shown in bold. A value of $TPR = 1.0$ is not considered a high value if it represents a blanket positive prediction (i.e. $TNR = 0.0$ for the same label).

The BR-SVM technique produces blanket positive predictions for the same five labels as it does for dataset 1 (efavirens, emtricitabine, delavirdine, nevirapine and lamivudine). These five labels are the five most imbalanced in the dataset. For each label, more than 84% of patients are resistant to the corresponding drug (as shown in table 5.5). The only label for which BR-SVM seems to show reasonable performance is tenofovir. The *DP* value of 1.34 indicates that BR-SVM is fairly predictive for this label, which is confirmed by the *TNR* value of 0.94.

As in dataset 1, it can be seen that using a naive Bayes base classifier for the binary relevance problem transformation technique significantly increases performance (shown in 5.10). There are no longer any

Table 5.10. Scalar evaluation measures for the problem transformation methods for dataset 2. The highest value for each evaluation measure is shown in bold.

	BR-SVM				**BR-NB**					**HOMER**				
	TPR	**TNR**	**MCC**	**DP**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**
efavirens	1.00	0.00	-	-	0.47	**0.85**	0.21	0.88	**0.75**	0.61	0.73	0.22	0.80	0.67
didanosine	0.43	0.84	0.30	0.77	0.70	0.52	0.22	0.53	0.65	0.42	0.75	0.18	0.44	0.58
emtricitabine	1.00	0.00	-	-	**0.92**	0.28	0.22	0.79	0.72	**0.71**	0.68	**0.29**	**0.89**	**0.73**
delavirdine	1.00	0.00	-	-	0.43	0.80	0.16	0.61	0.72	0.51	0.71	0.14	0.50	0.57
stavudine	0.14	0.91	0.08	0.29	0.31	0.77	0.08	0.22	0.59	0.38	0.71	0.10	0.24	0.55
nevirapine	1.00	0.00	-	-	0.47	**0.85**	0.21	0.88	**0.75**	0.61	0.73	0.22	0.80	0.67
tenofovir	0.43	0.94	**0.44**	**1.34**	0.62	0.76	**0.35**	**0.93**	0.73	0.42	**0.83**	0.25	0.68	0.60
etravirine	**1.00**	0.02	0.08	1.04	0.64	0.63	0.23	0.61	0.69	0.43	0.79	0.19	0.57	0.57
abacavir	0.59	0.71	0.30	0.69	0.52	0.68	0.20	0.47	0.64	0.50	0.68	0.18	0.41	0.60
lamivudine	1.00	0.00	-	-	**0.92**	0.28	0.22	0.79	0.72	**0.71**	0.68	**0.29**	**0.89**	**0.73**
zidovudine	0.03	**0.98**	0.02	0.16	0.31	0.83	0.16	0.44	0.65	0.39	0.77	0.16	0.43	0.61
mean	0.69	0.40	0.20	0.72	0.57	0.66	0.21	0.65	0.69	0.52	0.73	0.20	0.60	0.63
std dev	0.38	0.46	0.17	0.45	0.21	0.21	0.07	0.22	0.05	0.12	0.05	0.06	0.22	0.06

BR-SVM - Binary relevance with support vector machine base classifier, **BR-NB** - Binary relevance with naive Bayes base classifier, **HOMER** - Hierarchy of multi-label classifiers, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

blanket positive predictions and the TPR values are as high as 0.92 (for emtricitabine and lamivudine). BR-NB has AUC values of 0.75 for both efavirens and nevirapine, which indicates markedly better predictive power than a random classifier (0.5).

HOMER performs best among the problem transformation methods at predicting negative values (which corresponds with showing susceptibility to a drug). This is indicated by the high mean TNR value (0.73) produced by the HOMER technique. The technique also produces a very low standard deviation for TNR (only 0.05). HOMER perform best for lamivudine and emtricitabine, since for these labels it produces the highest TPR, MCC, DP and AUC values.

Algorithm Adaptation Methods

Table 5.11 shows the scalar evaluation measures for the MLkNN and PCT problem transformation techniques. The MLkNN method performs poorly, with the MCC (0.03) and AUC (0.55) values indicating that this technique performs only a fraction better than a random classifier. MLkNN produces blanket positive predictions for emtricitabine and lamivudine, which implies that the technique has no predictive power for these labels. It may seem that with a DP value of 1.11 for efavirens that MLkNN could be useful in this case, but it can be seen from the low TNR value (0.02) that this is not true.

As for dataset 1, the PCT multi-label classification technique produces no blanket positive predictions. PCT has very high TPR values (0.93) for efavirens, delavirdine and nevirapine, indicating that the technique is very good at predicting cases of resistance to these drugs. Further, PCT can correctly predict susceptibility to tenofovir in 90% of cases (according to the TNR value). The reasonably high MCC and DP values for tenofovir confirms that PCT performs best for this drug. PCT also has the highest mean TPR value (0.72) out of the problem transformation and algorithm adaptation methods, meaning that, on average, it can correctly predict resistance to a drug in 72% of cases.

Table 5.11. Scalar evaluation measures for the algorithm adaptation methods for dataset 2. The highest value for each evaluation measure is shown in bold.

	MLkNN					**PCT**			
	TPR	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**
efavirens	**1.00**	0.02	**0.07**	**1.11**	0.58	**0.93**	0.28	0.24	0.95
didanosine	0.06	0.93	-0.01	-0.03	0.49	0.55	0.74	0.30	0.69
emtricitabine	1.00	0.00	-	-	**0.63**	0.92	0.30	0.25	0.87
delavirdine	**1.00**	0.02	**0.07**	1.06	0.58	**0.93**	0.26	0.22	0.86
stavudine	0.01	0.99	0.03	0.34	0.46	0.38	0.78	0.17	0.43
nevirapine	**1.00**	0.02	**0.07**	**1.11**	0.58	**0.93**	0.28	0.24	0.95
tenofovir	0.01	**1.00**	0.04	0.63	0.53	0.45	**0.90**	**0.38**	**1.08**
etravirine	0.99	0.02	0.01	0.14	0.58	0.91	0.25	0.19	0.63
abacavir	0.27	0.67	-0.06	-0.14	0.52	0.62	0.71	0.33	0.77
lamivudine	1.00	0.00	-	-	**0.63**	0.92	0.30	0.25	0.87
zidovudine	0.01	0.99	0.00	-0.07	0.46	0.34	0.82	0.16	0.45
mean	0.58	0.42	0.03	0.46	0.55	0.72	0.51	0.25	0.78
std dev	0.49	0.48	0.04	0.53	0.06	0.25	0.27	0.07	0.21

MLkNN - Multi-label *k*-nearest neighbours, **PCT** - Predictive clustering trees, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

Table 5.12. Scalar evaluation measures for the ensemble methods for dataset 2. The highest value for each evaluation measure is shown in bold.

	RAkEL					**ECC**				
	TPR	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**
efavirens	0.68	0.67	0.23	0.80	**0.72**	0.97	0.28	0.34	1.37	**0.83**
didanosine	0.41	0.84	0.28	0.72	0.65	0.52	0.78	0.31	0.75	0.71
emtricitabine	0.80	0.46	0.23	0.70	0.65	**0.98**	0.24	0.34	**1.40**	0.77
delavirdine	0.74	0.63	0.27	0.87	0.70	0.96	0.32	0.37	1.39	0.80
stavudine	0.61	0.60	0.20	0.47	0.62	0.34	0.83	0.19	0.50	0.65
nevirapine	0.67	0.68	0.24	0.81	**0.72**	0.97	0.27	0.31	1.29	0.82
tenofovir	0.42	**0.92**	**0.39**	**1.15**	0.70	0.50	**0.91**	**0.44**	1.26	0.81
etravirine	0.65	0.49	0.13	0.33	0.59	0.95	0.25	0.27	0.95	0.71
abacavir	0.33	0.85	0.20	0.54	0.60	0.59	0.74	0.33	0.78	0.71
lamivudine	**0.90**	0.30	0.22	0.76	0.66	0.97	0.28	0.34	1.32	0.76
zidovudine	0.25	0.83	0.09	0.26	0.57	0.28	0.88	0.19	0.57	0.68
mean	0.59	0.66	0.23	0.67	0.65	0.73	0.52	0.31	1.05	0.75
std dev	0.21	0.19	0.08	0.26	0.05	0.28	0.29	0.07	0.35	0.06

RAkEL - Random *k*-labelsets, **ECC** - Ensemble of classifier chains, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

Ensemble Methods

The scalar evaluation measures are given for the RAkEL and ECC ensemble methods for dataset 2 in table 5.12. The highest value for each measure is shown in bold. As for dataset 1, the RAkEL method produces no blanket positive predictions, which means that it can provide predictive information for every label. The highest values for TNR, MCC and DP produced by the RAkEL technique are all for tenofovir. This means that RAkEL performs best at predicting this label, which is confirmed by the fact that the AUC value is also high (0.70).

The ECC technique trained and validated using dataset 2 shows the best results out of all the experiments performed. It has the highest mean values for the TPR, MCC, DP and AUC evaluation measures across all experiments on both datasets. ECC has a particular high value (and low standard deviation) for the AUC measure for the efavirens label (0.83) and has a TPR value of 0.97 for the same label. This means that ECC correctly detects resistance to efavirens in 97% of cases for dataset 2. The ECC technique also has a TPR value above 0.95 for six different labels (efavirens, emtricitabine, delavirdine, nevirapine, etravirine and lamivudine). For the same 6 labels, ECC has an AUC of over 0.71 in each case.

5.2.3 Comparison

Presented below is a comparison between the scalar evaluation measures generated from the outputs of the experiments on each dataset. The tables are presented in a similar format as in sections 5.2.1 and 5.2.2 above. The difference is that the numeric values presented in tables 5.13 – 5.15 represent the difference between the evaluation measures produced in dataset 1 and 2.

The numeric values seen in the evaluation measure difference tables are calculated by subtracting the evaluation measures produced for dataset 1 from the measures produced for dataset 2. For example, the value for the efavirens label in the TNR column under the BR-NB method in table 5.13 below is 0.59, which is calculated by taking the TNR value for efavirens for BR-NB for dataset 2 (0.85 as shown in table 5.10)

Table 5.13. Problem transformation methods evaluation measure differences. The values are generated by subtracting the values generated for dataset 1 from the corresponding values for dataset 2. The absolute values are shown and bold values indicate improved performance on dataset 2, while grey values imply that performance was better on dataset 1.

	BR-SVM				**BR-NB**					**HOMER**				
	TPR	**TNR**	**MCC**	**DP**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**
efavirens	0.00	0.00	-	-	0.39	**0.59**	**0.11**	**0.45**	**0.21**	0.28	**0.60**	**0.21**	**0.72**	**0.16**
didanosine	**0.16**	**0.04**	**0.22**	**0.56**	**0.16**	0.10	**0.07**	**0.17**	**0.03**	0.14	**0.16**	**0.03**	**0.10**	0.04
emtricitabine	0.00	0.00	-	-	0.01	**0.08**	**0.07**	**0.16**	**0.11**	0.24	**0.50**	**0.14**	**0.19**	**0.16**
delavirdine	0.00	0.00	-	-	0.40	**0.57**	**0.11**	**0.39**	**0.16**	0.33	**0.52**	**0.12**	**0.39**	**0.06**
stavudine	0.08	**0.06**	0.00	**0.05**	0.07	**0.07**	**0.01**	**0.04**	**0.01**	**0.01**	0.09	0.10	0.26	0.04
nevirapine	0.00	0.00	-	-	0.39	**0.59**	**0.11**	**0.45**	**0.21**	0.28	**0.60**	**0.21**	**0.72**	**0.16**
tenofovir	**0.16**	0.00	**0.17**	**0.43**	**0.20**	0.10	**0.05**	**0.07**	**0.08**	0.02	0.01	0.04	0.12	0.06
etravirine	**0.01**	0.00	**0.05**	**0.72**	0.24	**0.47**	**0.18**	**0.43**	**0.10**	0.09	**0.27**	**0.16**	**0.49**	**0.07**
abacavir	**0.08**	**0.10**	**0.18**	**0.43**	0.03	**0.05**	**0.02**	**0.06**	**0.01**	0.13	**0.11**	0.01	0.02	0.06
lamivudine	0.00	0.00	-	-	0.01	**0.08**	**0.07**	**0.16**	**0.11**	0.24	**0.50**	**0.14**	**0.19**	**0.16**
zidovudine	0.02	**0.02**	0.01	**0.01**	0.06	**0.04**	0.02	0.01	**0.02**	**0.01**	0.05	0.05	0.14	0.01
mean	**0.03**	**0.02**	**0.10**	**0.37**	0.12	**0.21**	**0.07**	**0.22**	**0.09**	0.15	**0.28**	**0.07**	**0.20**	**0.06**
std dev	**0.02**	0.02	0.08	0.17	**0.02**	**0.06**	0.00	0.01	0.01	**0.11**	**0.25**	**0.04**	**0.06**	0.00

BR-SVM - Binary relevance with support vector machine base classifier, **BR-NB** - Binary relevance with naive Bayes base classifier, **HOMER** - Hierarchy of multi-label classifiers, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

and subtracting the corresponding value produced for dataset 1 (0.26 shown in table 5.7). Only the absolute values of the differences are shown in the tables and a positive difference (i.e. the technique performs better on dataset 2) is shown in bold, while negative differences (the technique performs better on dataset 1) are shown in grey. Note that a reduced standard deviation is considered an improvement, so the standard deviation values shown in bold indicate that the technique had a lower standard deviation on dataset 2.

Problem Transformation Methods

Table 5.13 shows the evaluation measure differences for the problem transformation techniques. There is no difference for the six labels that BR-SVM produces blanket positive predictions for. Most of the differences indicate that BR-SVM performs better on dataset 2. This is confirmed by the fact that the mean value for every evaluation measure has improved for dataset 2. The most notable increase for the BR-SVM technique is the *DP* value for etravirine, which increased from 0.31 to 1.04 – an increase of over 300%. The value for all evaluation measures increased for both didanosine and abacavir, which implies on dataset 2 BR-SVM performs better in all cases for these drugs.

The evaluation measure difference for the BR-NB method shown in table 5.13 indicates improved performance overall on dataset 2. This is evident from the fact that the mean values for TNR, MCC, DP and AUC have all increased. The only mean value that did not increase was TPR, which decreased by 0.12. This implies that the BR-NB method is on average 12% worse at detecting cases of resistance. The standard deviations for each evaluation measure improved or stayed the same for TPR, TNR and MCC. Standard deviation decreased by 0.01 for both DP and AUC, meaning that the range of values obtained for these measures was distributed more widely for dataset 2 than for dataset 1.

As is the case with BR-NB, TPR for the HOMER technique has decreased for dataset 2. The mean value for the TPR evaluation measure has decreased by 0.15. The only two instances where the TPR value increased were for stavudine and zidovudine (both drugs in dataset 2 to which patients are overwhelmingly more susceptible than resistant). The evaluation measure that has increased most significantly for the

HOMER method is TNR. The TNR value has increase by 0.60 for efavirens and nevirapine and by 0.50 or more for emtricitabine, delavirdine and lamivudine. In the case of efavirens, this means that HOMER is 5 times better at predicting susceptibility to efavirens for dataset 2 than for dataset 1, which is a considerable improvement. The improvement is similar for nevirapine, emtricitabine, delavirdine and lamivudine.

Algorithm Adaptation Methods

The scalar evaluation measure differences for the algorithm adaptation techniques are shown in table 5.14. It can be seen that the TNR values for MLkNN have all either increased or stayed the same, meaning that MLkNN is better at detecting susceptibility to drugs on dataset 2 than on dataset 1. While the mean value increased for four out of the five measures, the standard deviation decreased for four out of the five. This implies that overall predictive power increased, but the range of values for the evaluation measures was spread more widely. Although not shown in table 5.14, it is important to notice that the MLkNN method only has two blanket positive predictions on dataset 2 (emtricitabine and lamivudine) compared to six on dataset 1 (efavirens, emtricitabine, delavirdine, nevirapine, etravirine and lamivudine), which is a large improvement and implies that MLkNN can provide meaningful predictive information for four more labels on dataset 2 than on dataset 1.

The values for all four evaluation measures decreased for the PCT technique for efavirens, emtricitabine, delavirdine, nevirapine and lamivudine. These are all drugs to which many more patients are resistant than susceptible to (in both datasets). For all the other labels, the evaluation measure values have increased, with the exception of the TNR value for didanosine, which has decreased by a small amount (0.03). The most significant evaluation measure improvement for dataset 2 for the PCT technique is the DP value for tenofovir, which has increased by 0.54. This represents a more than 100% improvement for that measure. Overall, considering the sum of all the improvements (2.49) and comparing this to the sum of all the decreases in performance (3.16), it appears that the PCT method performs slightly better on dataset 1 than on dataset 2.

Table 5.14. Algorithm adaptation methods evaluation measure differences. The values are generated by subtracting the values generated for dataset 1 from the corresponding values for dataset 2. The absolute values are shown and bold values indicate improved performance on dataset 2, while grey values imply that performance was better on dataset 1.

	MLkNN					**PCT**			
	TPR	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**
efavirens	0.00	**0.02**	-	-	**0.14**	0.04	0.06	0.14	0.61
didanosine	0.01	**0.11**	**0.13**	**0.48**	**0.04**	**0.08**	0.03	**0.04**	**0.07**
emtricitabine	0.00	0.00	-	-	**0.13**	0.02	0.01	0.05	0.22
delavirdine	0.00	**0.02**	**0.09**	-	**0.06**	0.04	0.05	0.13	0.59
stavudine	0.03	**0.03**	0.00	**0.16**	0.02	**0.07**	**0.01**	**0.08**	**0.21**
nevirapine	0.00	**0.02**	-	-	**0.14**	0.04	0.06	0.14	0.61
tenofovir	**0.01**	**0.01**	**0.10**	-	**0.02**	**0.17**	**0.02**	**0.20**	**0.54**
etravirine	0.01	**0.02**	-	-	**0.11**	0.00	**0.01**	**0.01**	**0.03**
abacavir	0.12	**0.04**	0.08	0.19	0.02	**0.06**	**0.02**	**0.07**	**0.17**
lamivudine	0.00	0.00	-	-	**0.13**	0.02	0.01	0.05	0.22
zidovudine	0.01	0.00	0.04	0.53	0.07	**0.14**	0.02	**0.13**	**0.34**
mean	0.01	**0.02**	**0.05**	**0.42**	**0.06**	**0.04**	0.02	0.00	0.08
std dev	0.01	0.01	**0.03**	0.12	0.02	**0.07**	0.01	**0.05**	**0.31**

MLkNN - Multi-label *k*-nearest neighbours, **PCT** - Predictive clustering trees, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

Table 5.15. Ensemble methods evaluation measure differences. The values are generated by subtracting the values generated for dataset 1 from the corresponding values for dataset 2. The absolute values are shown and bold values indicate improved performance on dataset 2, while grey values imply that performance was better on dataset 1.

	RAkEL					**ECC**				
	TPR	**TNR**	**MCC**	**DP**	**AUC**	**TPR**	**TNR**	**MCC**	**DP**	**AUC**
efavirens	0.24	**0.45**	**0.10**	**0.18**	**0.13**	0.03	**0.28**	**0.36**	-	**0.15**
didanosine	0.12	**0.26**	**0.18**	**0.49**	**0.07**	**0.11**	**0.05**	**0.17**	**0.40**	**0.07**
emtricitabine	0.13	**0.26**	**0.07**	**0.03**	**0.05**	0.02	**0.21**	**0.24**	**0.38**	**0.15**
delavirdine	**0.22**	**0.59**	**0.27**	**0.83**	**0.10**	0.04	**0.28**	**0.18**	-	**0.11**
stavudine	**0.12**	0.03	**0.08**	**0.20**	**0.08**	**0.09**	**0.01**	**0.11**	**0.27**	**0.12**
nevirapine	0.24	**0.47**	**0.12**	**0.25**	**0.12**	0.03	**0.27**	**0.33**	-	**0.21**
tenofovir	**0.04**	**0.04**	**0.11**	**0.33**	**0.04**	**0.11**	0.02	**0.06**	**0.10**	**0.09**
etravirine	0.20	**0.38**	**0.18**	**0.54**	**0.09**	0.03	**0.23**	**0.29**	**1.15**	**0.15**
abacavir	0.09	**0.10**	**0.03**	**0.12**	0.00	0.00	**0.04**	**0.05**	**0.13**	**0.03**
lamivudine	0.08	**0.19**	**0.03**	0.30	**0.06**	0.03	**0.25**	**0.25**	**0.30**	**0.16**
zidovudine	0.01	**0.01**	0.01	0.01	**0.02**	**0.02**	0.02	0.00	0.01	**0.05**
mean	0.10	**0.25**	**0.11**	**0.24**	**0.07**	**0.02**	**0.14**	**0.18**	**0.45**	**0.12**
std dev	**0.07**	**0.13**	**0.01**	**0.10**	0.01	**0.05**	**0.13**	**0.06**	**0.11**	0.00

RAkEL - Random *k*-labelsets, **ECC** - Ensemble of classifier chains, **TPR** - True positive rate, **TNR** - True negative rate, **MCC** - Matthews correlation coefficient, **DP** - Discriminative power, **AUC** - Area under the receiver operating characteristic curve

Ensemble Methods

The evaluation measure differences for the ensemble methods are given in table 5.15. As with BR-NB and HOMER, most of the TPR values have decreased in dataset 2 for RAkEL. The only labels for which the TPR values have increased are stavudine and tenofovir. Both of these labels correspond to drugs to which many more patients are susceptible than resistant. The only other drug for which this is the case is zidovudine and the TPR value for zidovudine has only decreased by a very small amount (0.01). This means that overall, for the TPR measure, RAkEL has improved for the drugs to which patients are more susceptible than resistant, and worsened for drugs to which patients are more resistant than susceptible. The values for TNR, MCC, DP and AUC have all increased on average.

The AUC value has increased for all labels for dataset 2 for the ECC technique, with the value for nevirapine increasing by the largest amount (0.21). The MCC values have also all either increased or remained the same and all but one of the DP values (zidovudine, which has only decreased by 0.01) have increased. The mean value for every evaluation measure has increased and the standard deviations have all improved (i.e. decreased) or remained the same. This means that the ECC has improved by all measures on dataset two, with an average of a 14% increase for the TNR measure and a 12% increase for AUC. Overall, the ECC technique has showed the most improvement on dataset 2, with 42 out of the total of 55 evaluation measures increasing and only 9 measures decreasing.

5.2.4 Scalar Evaluation Measure Summary

The scalar evaluation measures generated by the results processing component of the processing pipeline have been given in tables 5.7 – 5.12 above. These tables show the 363 evaluation measures generated per dataset for all of the seven multi-label experiments conducted, plus the 66 mean values and standard deviations. Thus a total of 429 measures have been presented per dataset, resulting in a total of 858 evaluation measures for all experiments conducted. An additional 429 values have been given in tables 5.13 – 5.15.

These values represent the differences between the evaluation measures generated for dataset 1 and those generated for dataset 2.

The scalar evaluation measures for dataset 1 have been presented in tables 5.7 – 5.9. These measures include the TPR, TNR, MCC, DP and AUC values (where possible) for each label for each of the seven multi-label classification models trained and evaluated using dataset 1 (BR-SVM, BR-NB, HOMER, MLkNN, PCT, RAkEL and ECC). In general the BR-SVM and MLkNN techniques did not perform particularly well on dataset 1. They both had numerous cases of blanket positive predictions and the mean TPR and TNR values were all below 0.66 for both of these methods. MLkNN had a mean TNR of 0.40, which means that on average it could only correctly predict susceptibility to a drug in 40% of cases.

The BR-NB, HOMER and PCT techniques performed slightly better than BR-SVM and MLkNN on dataset 1. These techniques produced no blanket positive predictions and have mean TPR values of 0.69, 0.67 and 0.68 respectively. This means that in more than two thirds of cases, these three methods are able to correctly predict resistance to a drug. For BR-NB and HOMER, this is confirmed by the AUC values (0.60 and 0.57 respectively), which indicate that these methods are moderately more effective than a random classifier. The same can be said about the PCT method due to the fact that it has a DP value of 0.86 – the highest of all methods for dataset 1. PCT has a mean TNR of 0.53, which makes it the best at predicting susceptibility to drugs on dataset 1.

The RAkEL method performs about as well as PCT on dataset 1 in terms of the TPR measure. The mean TPR values for these methods are 0.69 and 0.68 respectively, meaning that on average they can correctly predict resistance to a drug in 69% and 68% of cases. Both methods perform particularly well for efavirens, emtricitabine, delavirdine, nevirapine and lamivudine (the drugs to which patients are more resistant than susceptible), with TPR values as high as 0.97 in some cases. ECC performs best on dataset 1 at predicting resistance, with a mean TPR of 0.71, although this may be slightly inflated due to the fact that the ECC technique produces blanket positive predictions for two labels (efavirens and nevirapine).

For dataset 1 some techniques produce mean TPR values as high as 0.71 (in the case of ECC), but TNR values are generally low. This means that while all of the methods seem to be sensitive enough to detect

resistance better than a random classifier, most are not specific enough to also be able to detect susceptibility. Only PCT has a TNR value above 0.5 (0.53), meaning that this method is the only one that is able to predict susceptibility to a drug better than a random classifier. This, along with the fact that PCT has the highest values for the MCC (0.25) and DP (0.86), makes this technique the best performing on dataset 1 based on the scalar evaluation measures.

The BR-SVM and MLkNN methods again do not perform well on dataset 2. Both still produce blanket positive predictions, although MLkNN produces blanket positive predictions for only two labels on dataset 2, compared to 5 on dataset 1. The TPR and TNR values for these two techniques are much the same for dataset 2 as for dataset 1, with at most a 0.03 improvement (in the case of TPR for BR-SVM). BR-SVM and MLkNN are both still only moderately better than a random classifier at predicting resistance and worse than a random classifier at predicting susceptibility on dataset 2.

The performance of the BR-NB, HOMER and PCT techniques has increased on average on dataset 2. Most significantly, the mean TNR values for all of these three methods is above 0.5 for dataset 2. This means that BR-NB, HOMER and PCT are all able to predict resistance and susceptibility to drugs better than a random classifier on dataset 2. On dataset 1, only the PCT technique was able to do this. The HOMER method has a mean TNR of 0.73 on dataset 2, making it the best method at predicting susceptibility to drugs across all the experiments conducted on both datasets.

The RAkEL and ECC methods both perform better than a random classifier based on all five evaluation metrics on dataset 2. RAkEL has a mean TNR value of 0.66 making it the second best method (tied with BR-NB) at predicting susceptibility to drugs across all experiments on both datasets. The mean TPR value for the the ECC method has increased to 0.73, meaning that on dataset 2 this method is able to correctly predict resistance to drugs in 73% of cases on average. ECC also has the only mean DP value above 1.0 and the highest mean AUC (0.75).

Most methods perform better than a random classifier at predicting both resistance and susceptibility when trained and evaluated using dataset 2, with only BR-SVM and MLkNN still performing worse than a

random classifier based on the mean TNR measure. This is confirmed by the fact that the mean AUC values are all above 0.5 and the mean MCC values are all above 0.0. The ECC is the best performing technique on dataset 2, with relatively high mean values for every measure and particularly high mean AUC and DP values.

While the scalar evaluation measure values obtained by training and evaluating the multi-label classification methods using dataset 1 are encouraging, the results obtained from dataset 2 confirm that the techniques tested are able to provide meaningful predictive information regarding resistance and susceptibility to the antiretroviral drugs. The general trend of improved mean values for all the scalar evaluation measures coupled with reduced standard deviations implies that the models improve when more data is used for training. This is desirable when the underlying dataset used for training is continually growing.

5.3 ROC Analysis

The receiver operating characteristic (ROC) curves generated from the raw results produced as part of the experimental pipeline described in chapter 4 are shown in figures 5.1 – 5.6. These curves were automatically generated using the developed results processing scripts and the R engine (described in sections 4.2.2 and 4.2.2). Note that *ROC* curves can only be generated for classification techniques that produce a probability of class membership as output (as opposed to an explicit class label) and therefore no *ROC* curves could be generated for the BR-SVM and PCT methods.

Each point on a *ROC* curve is generated by varying the threshold parameter (τ) and then calculating the false positive rate and true positive rate over all the results. Since 10 fold cross validation is used in this study, the average true positive rate is taken over all 10 cross validation cycles. This is known as vertical averaging [61].

The *ROC* curves generated based on the results of the experiments on dataset 1 are shown first. A separate *ROC* curve is shown for each of the 11 labels, and all five techniques for which it is possible to

generate *ROC* curves are shown in a single image. The error bars are shown for each value of τ that has been used and the *AUC* and standard error values for each technique are shown in the legend. The *ROC* curves generated from the output of the experiments on dataset 2 are then shown, followed by *ROC* curves that illustrate the differences between the two datasets.

5.3.1 Dataset 1

The *ROC* curves generated for dataset 1 are shown in figures 5.1 – 5.3. Ideally, a *ROC* curve should be as close to the top left of the unit square as possible. Stated differently, the greater the area under the *ROC* curve (*AUC*), the better the performance of the classification technique. A random classifier would have a *ROC* curve with a gradient of one (i.e. a straight line), resulting in an *AUC* of 0.5. Having an $AUC < 0.5$ implies that the technique performs worse than a random classifier.

Most of the *ROC* curves shown in figures 5.1 – 5.3 are indicative of performance only marginally better than that of a random classifier. Notable exceptions are the curves for the MLkNN. Five of these curves (efavirens, didanosine, stavudine, nevirapine and etravirine) show performance worse than random classifier. The drug for which the *ROC* curves are closer to the upper left of the graph is tenofovir (see figure 5.2(c)). It can be seen that for this drug all of the *AUC* values are above 0.5, with ECC having the particularly high value of 0.72.

The average *AUC* for the *ROC* curves generated for dataset 1 is 0.58 and the average standard error is 0.048. This implies that on average (over all techniques and labels), the performance of the predictive models is only slightly better than a random classifier for dataset 1. That said, there is at least one technique that obtains an *AUC* of more than 0.6 for every drug, which is significantly better than the performance of a random classifier. This means that while on average the performance of the techniques may not be great, there are instances in which the predictive models do provide useful predictive information (for a subset of labels). This corresponds with what has been illustrated by the scalar evaluation metric values shown in section 5.2 above.

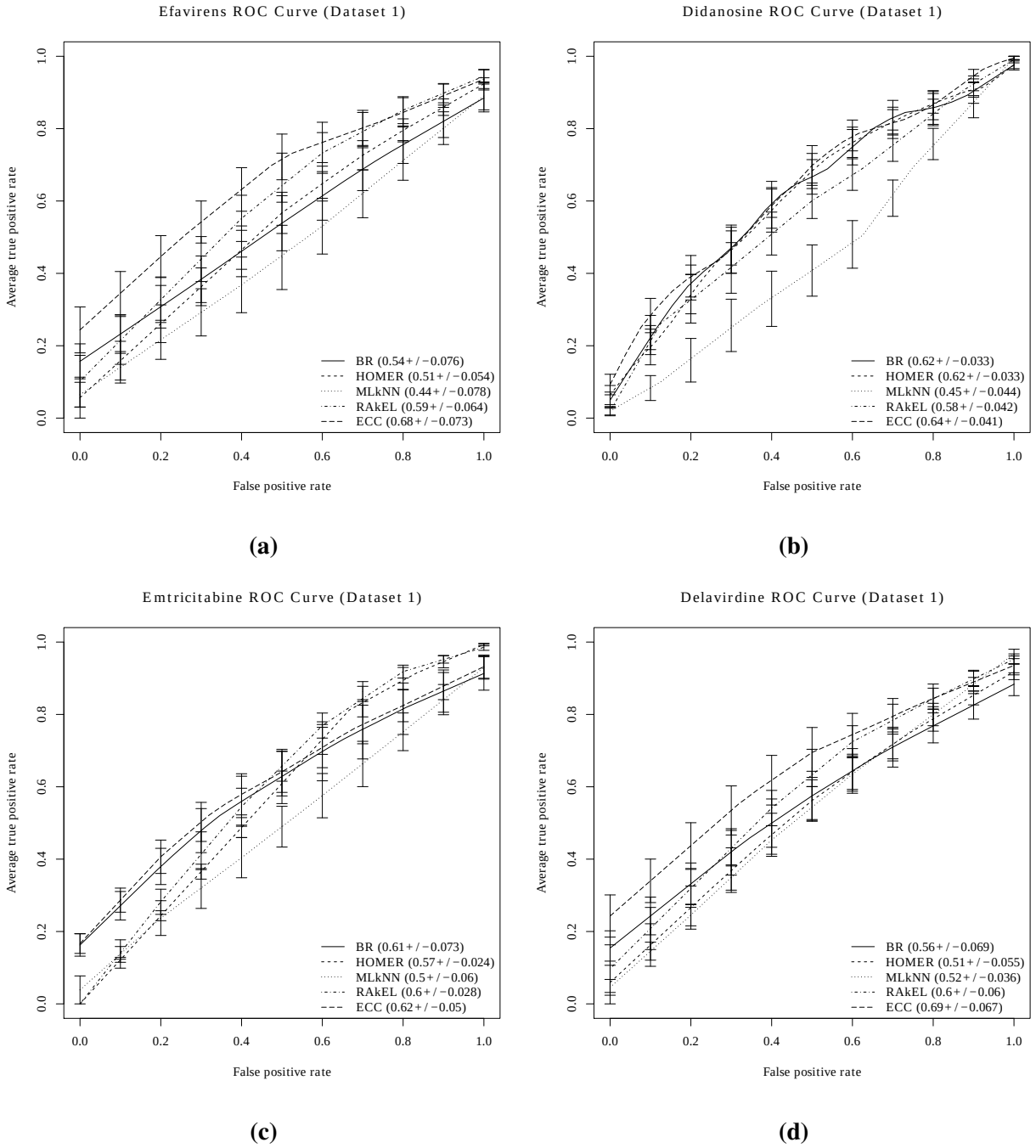


Figure 5.1. *ROC* curves for dataset 1 for efavirens, didanosine, emtricitabine and delavirdine. *AUC* and standard error are given in parentheses for each method.

ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains

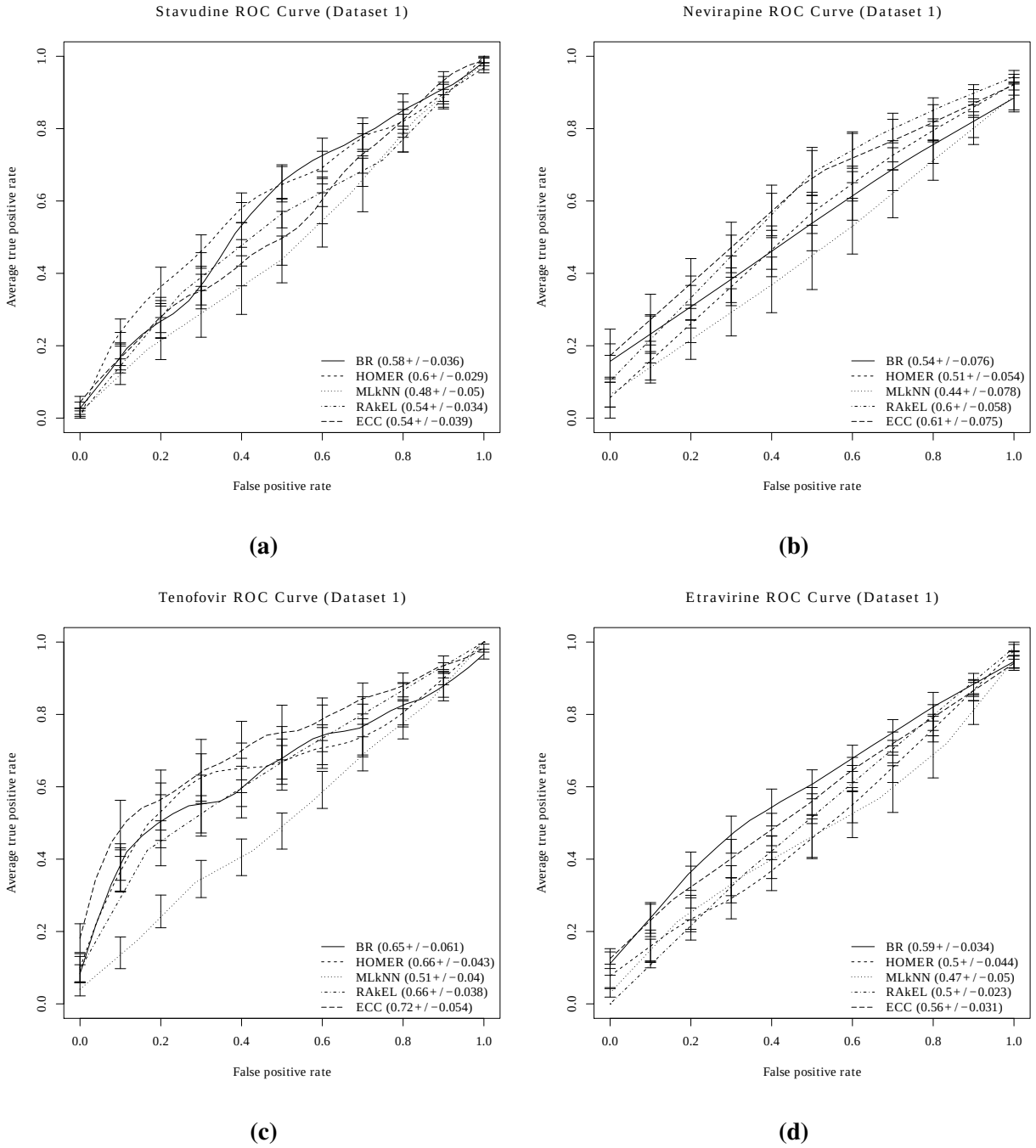
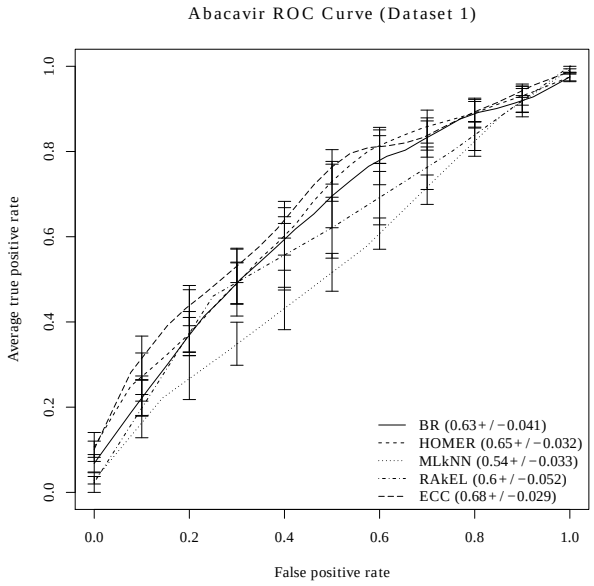
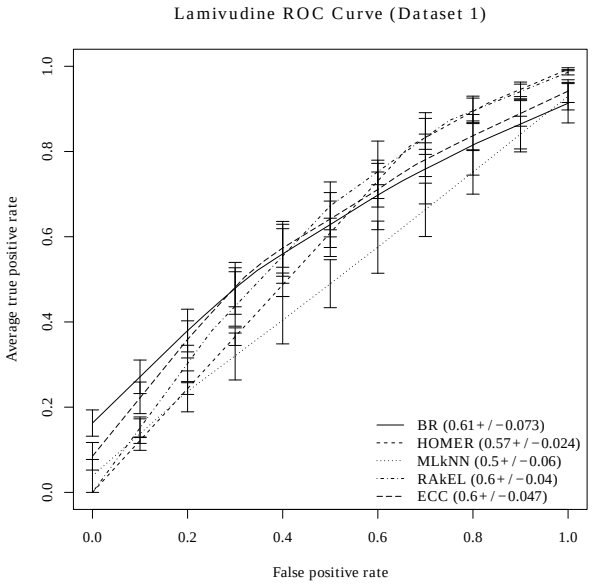


Figure 5.2. *ROC* curves for dataset 1 for stavudine, nevirapine, tenofovir and etravirine. *AUC* and standard error are given in parentheses for each method.

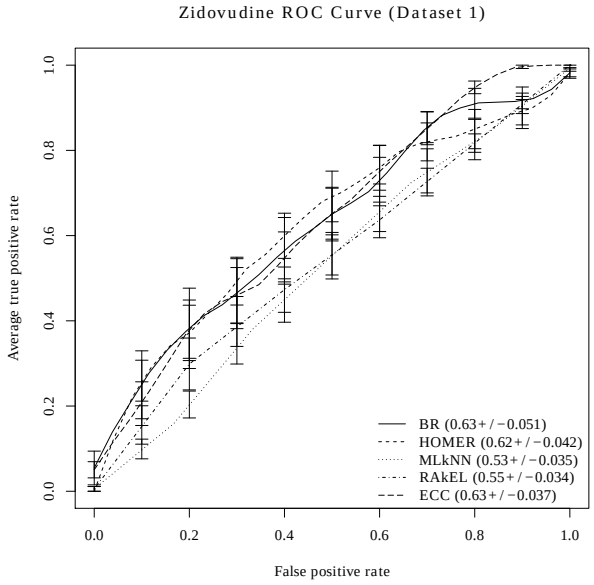
ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains



(a)



(b)



(c)

Figure 5.3. *ROC* curves for dataset 1 for abacavir, lamivudine and zidovudine. *AUC* and standard error are given in parentheses for each method.

ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains

5.3.2 Dataset 2

The *ROC* curves generated for each label for dataset 2 are shown in figures 5.4 – 5.6. The average area under the curve (*AUC*) for the 11 labels represented in the figures is 0.65 and the average standard error is 0.03. There are only three cases (didanosine, stavudine and zidovudine) where the *AUC* values is below that of a random classifier (0.5) and all three of these are produced by the MLkNN classification technique.

The *ROC* curves produced by the ECC method are the best in all cases for dataset 2. There are four labels (efavirens, delavirdine, nevirapine and tenofovir) for which the *AUC* produced by the ECC is greater than 0.8, which indicates very good performance by this method. The HOMER and MLkNN are the two worst performing methods in terms of the *ROC* curves for dataset 2, with one of these two methods having the lowest *AUC* value on each figure. Interestingly, the HOMER method is good at predicting susceptibility on dataset 2 (according to the *TNR* measure), but shows poor performance on the *ROC* curves. This is because the *ROC* curves show performance at predicting resistance (presence of a label) and not susceptibility.

Particularly good *ROC* curves are produced for the lamivudine (figure 5.3(b)) and emtricitabine (figure 5.1(c)) labels. For both of these labels, the lowest *AUC* value is 0.63 and the average is 0.7, which implies performance much greater than a random classifier for all classification techniques. Nevirapine and efavirens are the labels with the highest mean *AUC*, which is 0.71. This is despite the fact that the MLkNN method produces an *AUC* of just 0.58 for both of these labels. Removing this low value from the calculation, the BR-NB, HOMER, RAkEL and ECC methods produce an average *AUC* 0.74 for both nevirapine and efavirens, which indicates very good performance.

The average standard errors produced for each method range from 0.027 (for ECC and RAkEL) to 0.032 for MLkNN on dataset 2. This implies that MLkNN produces the widest range of average true positive values, while ECC and RAkEL produce a much narrower range. This can be interpreted to mean that the ECC and RAkEL methods produce consistently good results.

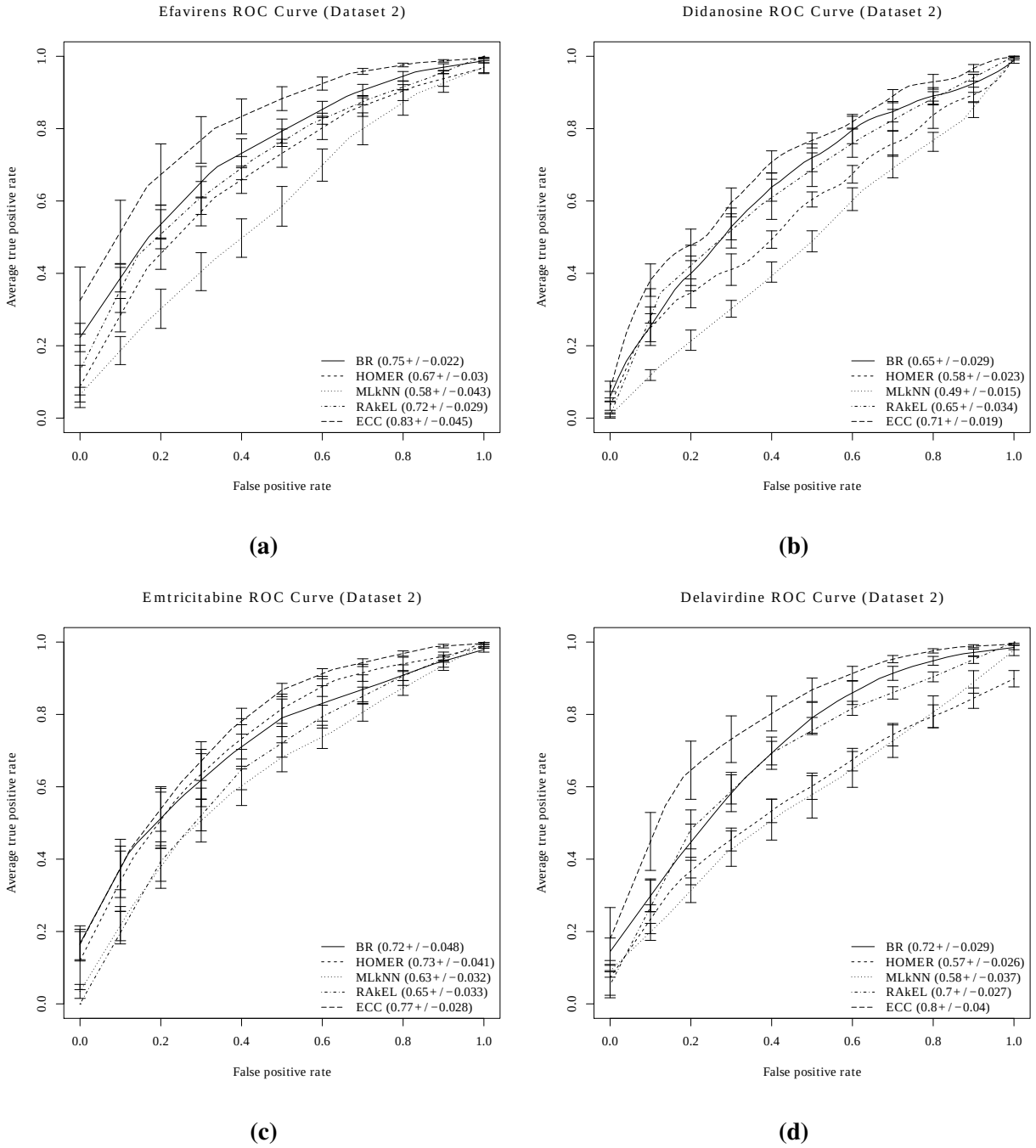
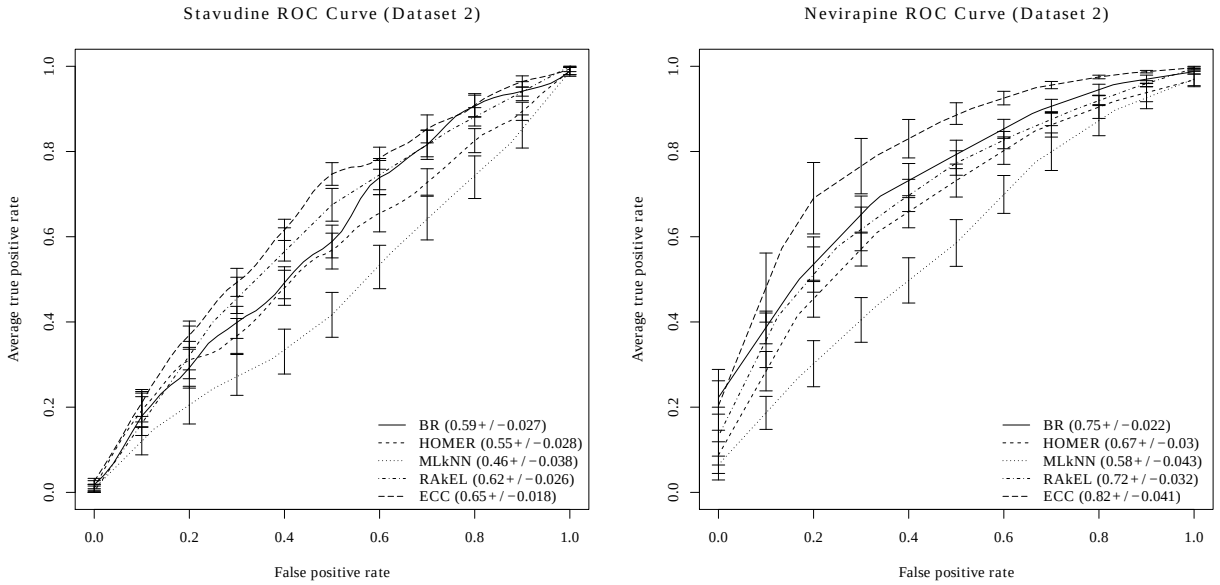


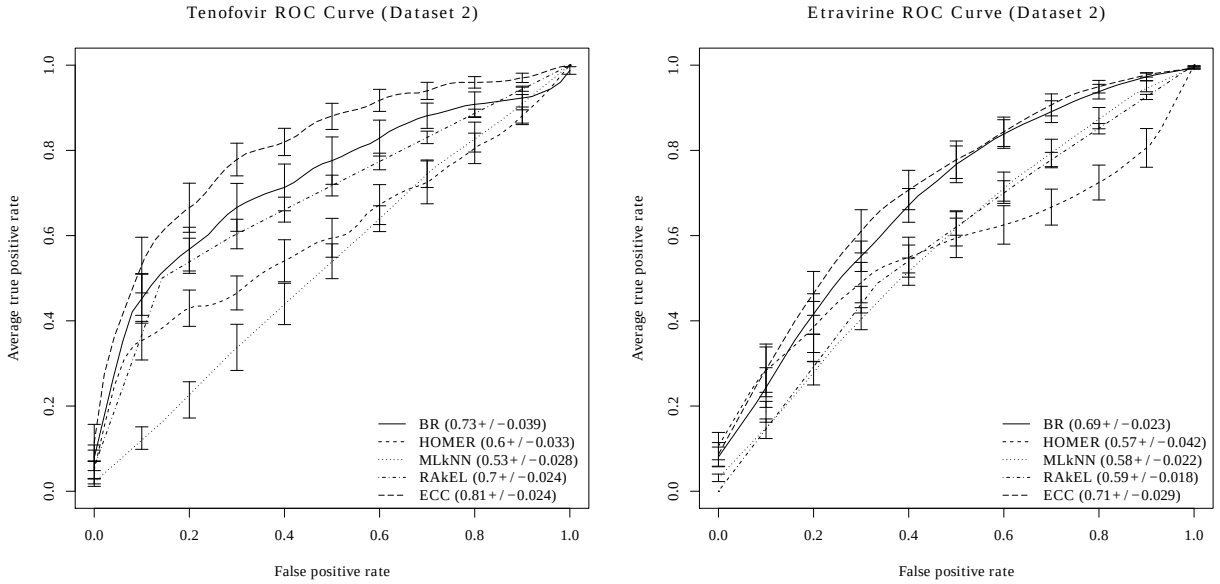
Figure 5.4. *ROC* curves for dataset 2 for efavirens, didanosine, emtricitabine and delavirdine. *AUC* and standard error are given in parentheses for each method.

ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains



(a)

(b)

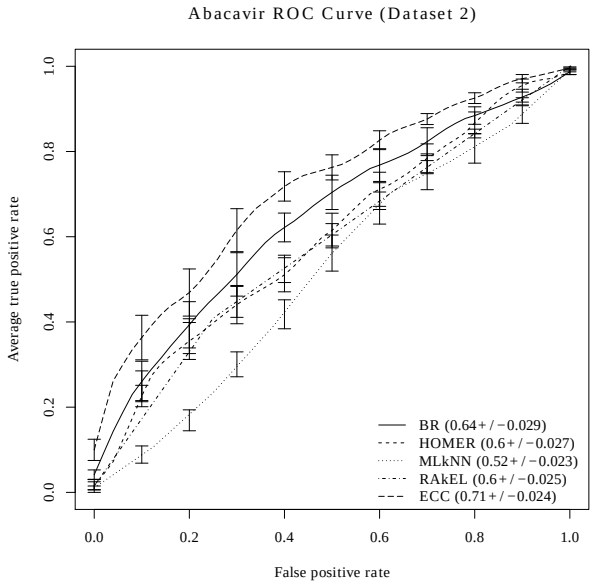


(c)

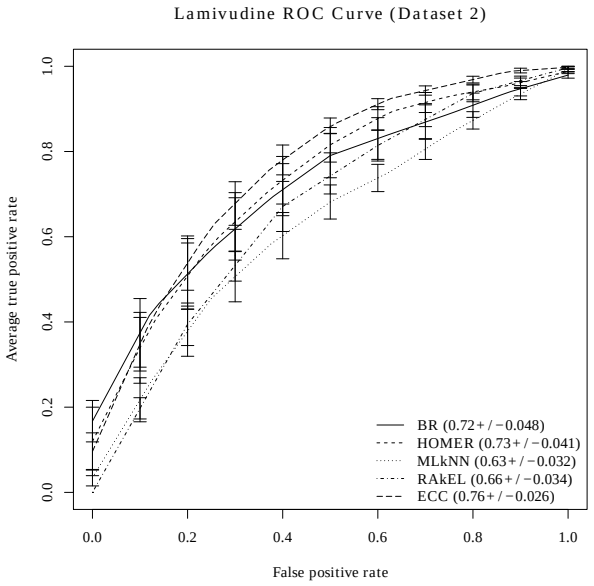
(d)

Figure 5.5. *ROC* curves for dataset 2 for stavudine, nevirapine, tenofovir and etravirine. *AUC* and standard error are given in parentheses for each method.

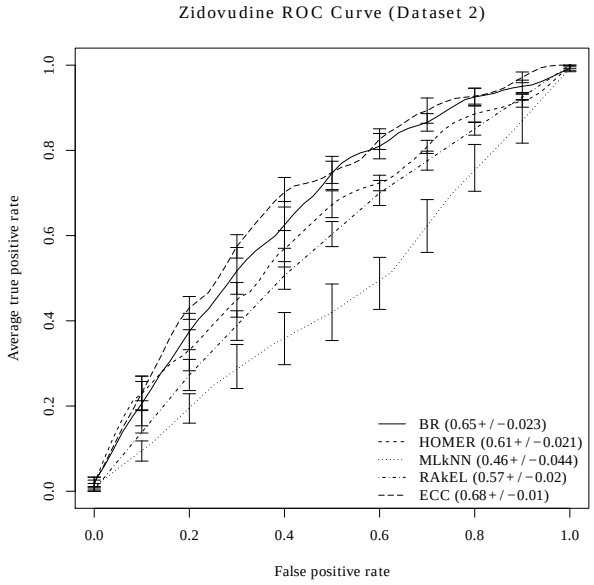
ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains



(a)



(b)



(c)

Figure 5.6. *ROC* curves for dataset 2 for abacavir, lamivudine and zidovudine. *AUC* and standard error are given in parentheses for each method.

ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains

5.3.3 Comparison

The *ROC* curves presented in figures 5.7 – 5.9 show the differences between the *ROC* curves generated for dataset 1 and the *ROC* curves generated for dataset 2. The curves were generated using a script that was developed to process the R command scripts generated by the processing pipeline. The curves generated for dataset 1 are drawn in grey and the curves generated for dataset 2 are drawn in black. The error bars have not been drawn in the interest of simplicity. The first number given in parentheses in the legend is the *AUC* for the dataset 1 curve and the second number is the *AUC* for the dataset 2 curve.

The first thing that becomes evident on examination of the *ROC* difference curves is that in almost every case the *AUC* values for the dataset 2 curves are greater than those for the dataset 1 curves. This implies that most methods performed better for every label on dataset 2 than on dataset 1 (in terms of the *ROC* curve evaluation measure). The only exceptions are the MLkNN method (which performed worse for stavudine, abacavir and zidovudine) and the HOMER method (which performed worse for didanosine, stavudine, tenofovir, abacavir and zidovudine).

All of the five techniques performed better on average for all labels except abacavir. The average *AUC* value over all techniques for this label decreased by 0.01. The biggest average *AUC* increase for a label was 0.17, which occurred for nevirapine. In addition, the average *AUC* over all methods increased by 0.16 for tenofovir. The biggest *ROC* curve improvement was an increase of the *AUC* by 0.21. This occurred for the BR-NB method for the efavirens label (shown in figure 5.7(a)) and for the ECC method for the nevirapine label (show in figure 5.8(b)). The average *AUC* increase over all labels and techniques was 0.13.

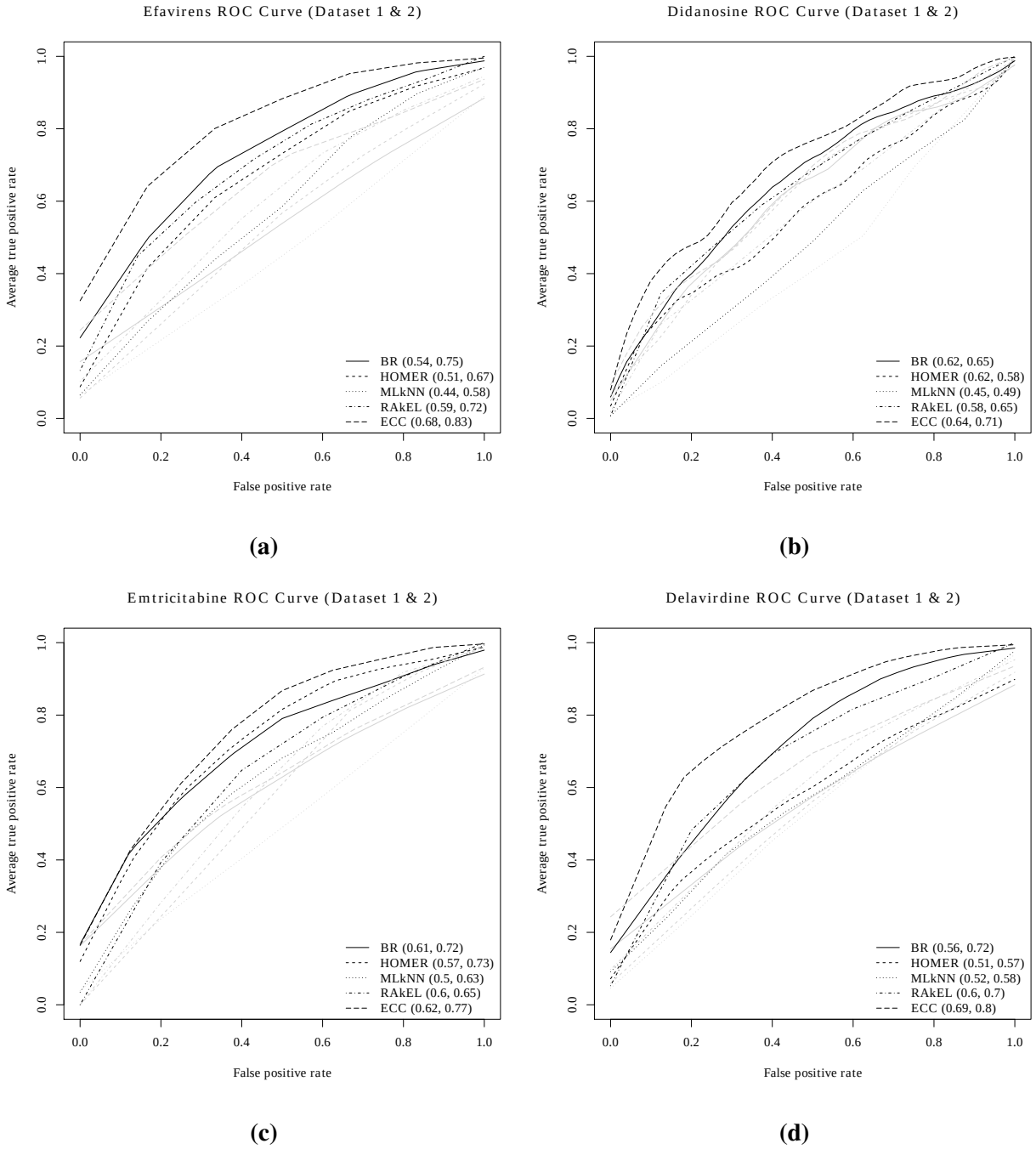


Figure 5.7. *ROC* curves for both datasets for efavirens, didanosine, emtricitabine and delavirdine. The curves for dataset 1 are shown in grey and the curves for dataset 2 are shown in black. *AUC* is given in parentheses for dataset 1 and 2 respectively.

ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label *k* nearest neighbours, **RAkEL** - Random *k*-labelsets, **ECC** - Ensemble of classifier chains

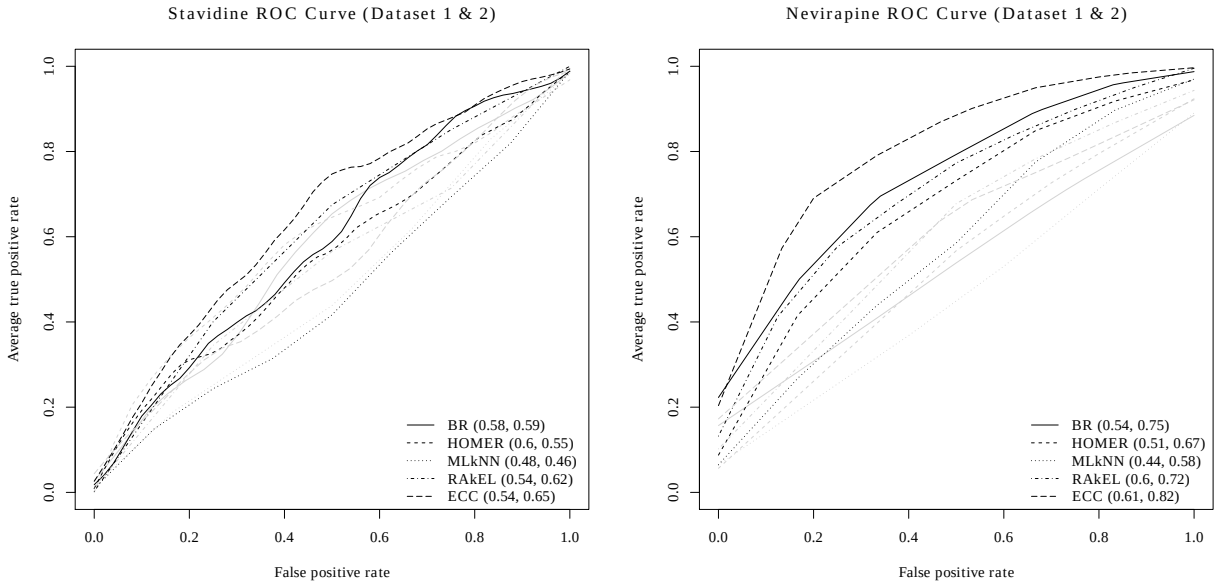
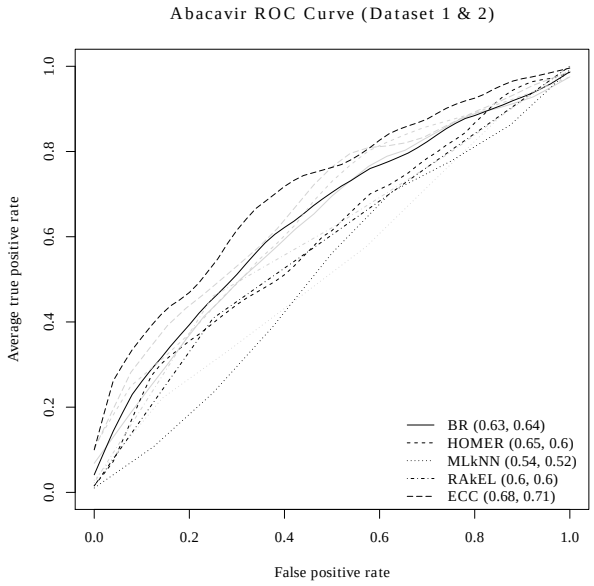
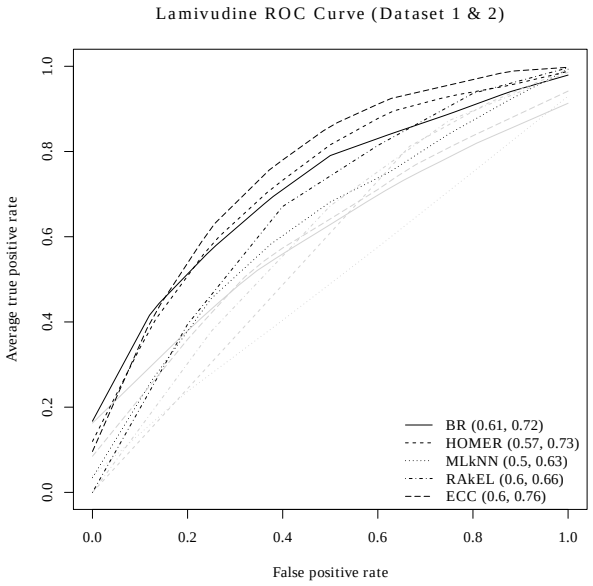


Figure 5.8. *ROC* curves for both datasets for stavudine, nevirapine, tenofovir and etravirine. The curves for dataset 1 are shown in grey and the curves for dataset 2 are shown in black. *AUC* is given in parentheses for dataset 1 and 2 respectively.

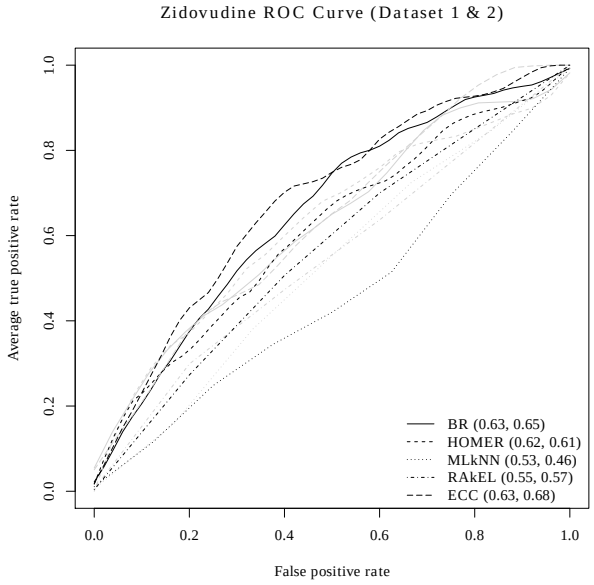
ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label k nearest neighbours, **RAkEL** - Random k-labelsets, **ECC** - Ensemble of classifier chains



(a)



(b)



(c)

Figure 5.9. *ROC* curves for both datasets for abacavir, lamivudine and zidovudine. The curves for dataset 1 are shown in grey and the curves for dataset 2 are shown in black. *AUC* is given in parentheses for dataset 1 and 2 respectively.

ROC - Receiver operating characteristic, **AUC** - Area under *ROC* curve, **BR** - Binary relevance (with naive Bayes base classifier), **HOMER** - Hierarchy of multi-label classifiers, **MLkNN** - Multi-label k nearest neighbours, **RAkEL** - Random k-labelsets, **ECC** - Ensemble of classifier chains

5.3.4 ROC Analysis Summary

The dataset 1 *ROC* curves indicate that many methods do not perform much better than a random classifier. This is due to the fact that the average *AUC* for dataset 1 is just 0.64, which is only 14% better than random classifier. However, there are some exceptions. For example, techniques that tend to perform well on the tenofovir label have an average *AUC* value of 0.72 for this label. The MLkNN technique performed the worst with a mean *AUC* of 0.49 (less than a random classifier).

The dataset 2 *ROC* curves shown an improvement in almost all cases. Every technique had an average *AUC* above 0.5, which means that each technique could provide meaningful predictive information based on the *ROC* evaluation measure. The ECC method produced particularly good *ROC* curves on dataset 2 with four *AUC* values above 0.08, the highest being 0.83 for the efavirens label. These high *AUC* values are indicative of very good performance by ECC.

The *ROC* analysis mostly agrees with the scalar evaluation measures. The advantage of the *ROC* analysis is that it gives insight into each technique regardless of the value of τ that is selected as the threshold parameter (which is 0.5 in all cases). Varying this parameter gives a sense of the robustness of a technique and considering the *AUC* value plus the the standard error together is indicative of the consistency of the technique. This is confirmed by considering the MLkNN technique on the first dataset. By all of the scalar evaluation measure, MLkNN does not perform well and precisely this is echoed by the *ROC* curves. MLkNN achieves a low average *AUC* (0.49 on dataset 1 and 0.55 on dataset 2) and high standard deviations (0.05 on dataset 1 and 0.03 dataset 2).

The ECC technique has shown the most improvement on dataset 2 over dataset 1 in terms of the *ROC* analysis. This corresponds with what the scalar evaluation measures indicate and provides us with further confidence in the technique. It can also be inferred that the ECC method is consistent across different values of τ, since it has the lowest overall standard error (0.027).

5.4 Results Summary

The dataset description, scalar evaluation measures and *ROC* curves have been presented for all experiments. These results represent the performance evaluations of the seven multi-label classification techniques on both dataset 1 and dataset 2. The tables and figures given in sections 5.1 – 5.3 correspond to the results of the 14 experiments conducted (seven techniques tested on two datasets).

Examination of the datasets revealed that the two, while very different in size, were very similar in terms of label distribution and feature completeness. Dataset 2 was roughly double the size of dataset 1, with dataset 1 having 252 patients and dataset 2 having 500. The feature completeness was over 80% in both cases, which is good for a clinical dataset. This is most likely due to the fact that these datasets were both research datasets and there is a dedicated team ensuring the data is of good quality. Further, the label distribution and the proportion of patients resistant to each number of drugs were very similar for both datasets.

The scalar evaluation measures for each dataset were first presented and discussed. A comparison of the measures was then presented. It is evident from these evaluation measures that the BR-SVM and MLkNN techniques are the worst performing on both datasets. The BR-NB, HOMER and PCT perform better, but not as well as RAkEL and ECC. On dataset 1, while some techniques were relatively good at predicting resistance, not many were able to predict susceptibility. In fact, the only method that was able to out-perform a random classifier at predicting susceptibility was PCT, with a mean TNR of 0.53.

On dataset 2, most of the techniques performed significantly better. The only techniques with mean TNR values below that of a random classifier were BR-SVM and and MLkNN. The average increase over all the scalar evaluation measures was 0.10. The mean TPR value decreased by 0.04 and the mean TNR value increased by 0.13. The reason for the decrease in TPR is most likely due to the TNR value being artificially inflated for dataset 1 and the lower TPR on dataset two probably implies better predictive power. This scenario is referred to as the accuracy paradox [95] and is discussed in section 6.2.

The *ROC* analysis presented in section 5.3 largely confirms the performance shown by the scalar evaluation measures. The *ROC* curves confirm that the MLkNN method is the worst performing and that ECC shows the biggest improvement on dataset 2. In addition, the *ROC* analysis provides insight into the consistency of the techniques over different threshold values (τ).

In general, the results obtained from the multi-label classification experiments are encouraging. The methods performed well on dataset 1 at predicting resistance, but were not able to provide much useful information with regards to susceptibility (with the exception of the PCT method). This changed when the techniques were trained and evaluated using dataset 2. In this case, all of the techniques were able to provide predictive information for both resistance and susceptibility.

Chapter 6

Discussion

This section discusses the implications of the results presented in terms of the research question. The dataset is discussed first, followed by the experimental results. The limitations of the predictive models are then described and the chapter concludes with a discussion about the potential impact of the implementation of the classification techniques in a real-world clinical environment.

6.1 Dataset

Descriptions of both datasets used have been give in section 5.1.3. The salient points are recalled below and an analysis of the specific characteristics of the datasets that may have impacted the performance of the classification techniques is given.

Overview

The two datasets were snapshots of the same clinical information system (the implementation of RegaDB used by the Africa Centre) at two different points in time, roughly two years apart. Dataset 2 (downloaded

in 2013) is about twice the size of dataset 1 (downloaded in 2011) and consisted of the medical records of 500 patients. Dataset 1 contained the clinical data from 252 patients. Other than the size difference, the datasets were very similar. The mean patient age (just over 37 years) and the proportion of females (about three quarters) in each dataset were the same. The datasets are also similar in terms of average feature completeness, with dataset 1 having a value of 76.84% and dataset 2 having a value of 78.29%. More than a third of the features in each dataset were 100% complete. The proportions of patients resistant and susceptible to each drug as well as the proportions of patients resistance to each number of drugs, were almost identical, with the biggest difference being only 3% across both of these measures.

The reason that a large proportion of patients are resistant to some drugs but susceptible to others is that the *genetic barrier* of the drugs are different. The genetic barrier of a drug is the number of mutations necessary in the viral genome before resistance is developed [3]. For example, lamivudine is known to have a genetic barrier of 1, which means that just one specific mutation is required in the viral genome for the patient to develop resistance to lamivudine. Drugs like zidovudine and stavudine have a genetic barrier greater than 1, so multiple specific mutations are required to develop resistance to these drugs. Tables 5.2 and 5.5 show that more patients are resistant to lamivudine than zidovudine or stavudine. A recent study has found that modelling the genetic barrier of the virus can be used as a predictor for treatment outcome [96].

The small number of training examples by machine learning standards (252 in dataset 1 and 500 in dataset 2) could be partly responsible for the low performance for some of the labels. Given the similarity of the datasets, the fact that the predictive models perform better on dataset 2 must be due to the fact that dataset 2 contains more training examples than dataset 1, which is exactly the behaviour we hope to see from models used for classification. In fact, this is implied in the definition of machine learning given at the beginning of chapter 2: *"machine learning is concerned with the question of how to construct computer programs that automatically improve with experience."* The important implication of this result is that as more patient data is collected over time, the predictions generated by the models will continue to improve.

Feature Vector

Since its introduction in 2007, many studies have made use of the *treatment change episode* (TCE) as the unit of data (i.e. feature vector) used to train and test classification models [5, 64, 65, 67, 75, 82, 96–98]. A TCE is centred around the time when a patient switches from one drug regimen to another. The data comprises of genotype information, treatment information from before and after the switch, as well as multiple viral load measurements taken at specific times before and after the switch. Although studies using this type of data have reported good results, there are a number of problems with using TCEs as training examples. The first issue that arises when dealing with data from an African context (such as the data used in this study) is that the treatment guidelines used in African contexts do not match those used in the European or American contexts. Therefore, it is often the case that the viral load measurements required for a specific TCE have not been taken within the limited time frame mandated by the TCE definition. Furthermore, TCEs require a genotype resistance test to have been done for each patient, which will not always be the case in South Africa and other developing countries.

TCEs are primarily used for predicting treatment outcome rather than resistance to specific drugs, which is the objective of this study. TCEs represent isolated moments in time rather than the full patient history. Patients may change treatments many times over the course of their lives and it is therefore possible to extract many TCEs from a single patient records. These TCEs are then used independently as training examples by the classification technique. There is a significant problem with this approach. Each time a patient is exposed to a drug, the virus population in that patient reacts to the drug, thereby influencing that patient's response to the current and all future treatment regimens [3, 4]. This means that a patient who has experienced many TCEs could have a very different therapy outcome to a patient who is experiencing their first TCE, even if the treatment that each is switching to and from are the same. This makes using TCEs problematic and therefore the decision has been taken to use the full patient record over TCEs. This means that there are fewer training examples, but that each training example encodes the full history of the patient.

Feature Completeness

The effects of feature completeness and possible incorrect feature values have not been examined in this study, but it is worth noting that the datasets do contain features whose values can not reliably be validated as correct. These are the features that have been self-reported by the patients and mainly relate to adherence to therapy. There are no objective or proxy measures for the adherence features, such as pharmacy refill data or medication possession ratios and since adherence is widely considered an important factor involved in the development of drug resistance [99–101], the reliability of these self-reported features may have a significant impact on the performance of the classification models. Furthermore, missing values for features have been interpreted to mean that the value for that specific feature is not available. It is possible to extract more information from missing features, for example, if a patient is missing a value for *baseline viral load* (which is a derived feature), it could be for a number of reasons. The patient could have never had a viral load measurement done, or the measurement could have not been done in the specific time frame required for it to be considered a *baseline* measurement. Thus, missing features could provide further information that could be valuable, but this information has not been exploited in this study.

Data Quality

Another factor that could influence the performance is the presence of features in the dataset that contain incomplete data. Unfortunately, missing and inaccurate data are common aspects of public health ART treatment program data in South Africa and other resource-limited countries. Patients miss clinic visits for a number of health and socio-economic reasons and it is often the case that laboratory test results are not available when needed by the clinician. Further investigation should be done on the use of feature selection techniques to ensure that the optimal subset of features is being used for training and prediction, since irrelevant and redundant features can reduce classification accuracy [102].

The clinical datasets used in this study were used for both clinical management of patients as well as for research. There is a small team of dedicated data management staff who are responsible for making sure

the datasets are as clean and complete as possible. The datasets are therefore much cleaner than the average clinical dataset. This could result in better performances from the classifiers, since they have been trained using more complete and accurate underlying data than might be case for the typical dataset used for the clinical management of patients.

Despite the fact that the datasets may be of higher quality than most datasets used for the clinical management of HIV patients, they nevertheless represent an example of a real-world dataset that might be used in the field. This gives us confidence that the performance of the predictive models are representative of the performance what we might expect if the models were implemented in a clinical environment.

6.2 Experimental Results

The multi-label classifiers were evaluated based on their ability to predict known resistance to a set of antiretroviral drugs based only on prior patient biographical and treatment history data. Comprehensive summaries of the experimental results have been presented in sections 5.2.4, 5.3.4 and 5.4, so the paragraphs below will illuminate only the most important results and then focus on discussing how the various dataset characteristics, design decisions and other factors may have had an effect on these results.

Overview

While most of the classification models were able to provide some predictive information (i.e. they performed better than a random classifier), there were some experiments where this was not the case. The most notable examples of poor performance are the blanket positive predictions produced by the BR-SVM and MLkNN methods. For dataset 1, MLkNN produces blanket positives for 6 of the 11 labels and BR-SVM produces blanket positive for 5 labels. In all cases, these labels correspond to the most imbalanced labels (many more patients are resistant than susceptible to the corresponding drugs). This indicates that the predictive models produced using the MLkNN and BR-SVM techniques were not able to detect cases where

patients were susceptible to drugs when very few such examples exist in the training datasets. The performance of the BR-SVM and MLkNN techniques on dataset 2 are similarly poor, although slightly better than on dataset 1. It follows that BR-SVM and MLkNN are not useful in a real world clinical environment if the proportions of patients resistant and susceptible to the various drugs are not well balanced.

Dataset Effects

The poor performance of BR-SVM is surprising, since SVMs are a popular technique and have been applied to many problems. However, the results obtained are consistent with previous investigations on the application of SVMs using imbalanced data, which found that SVMs performed poorly in this case [103, 104]. Furthermore, it has been shown that SVMs will favour the majority class [80], which is exactly what can be seen in tables 5.7 and 5.10. These tables illustrate the blanket positive predictions generated by the SVMs. The results obtained for MLkNN also correspond with previous studies that show that the nearest neighbours technique does not perform well on imbalanced datasets [105].

Techniques that perform well on the highly imbalanced labels (such as efavirens and emtricitabine) perform less well on the relatively balanced labels (such as didanosine and abacavir). The techniques that exhibit this behaviour are BR-NB, HOMER, RAKEL and PCT on dataset 1 and all of the techniques except BR-SVM and MLkNN on dataset 2. In general, this effect is less prominent on dataset 2 since the models produce more consistent predictions (as shown by the reduced standard deviations). The fact that the predictive models seem not to be able to simultaneously produce accurate predictions for highly imbalanced as well as relatively balanced labels supports the idea that no single model should be used to assess resistance to all drugs. The results of multiple models should be combined into an ensemble to produce the best results. Using ensembles or *committees* in this context has been studied before and was shown to increase the performance of classification models trained for predicting response to HIV therapy [75].

It is also important to note that models that perform well on imbalanced labels do so regardless of whether patients are more resistant to the corresponding drug or whether they are more susceptible. An

example of this can be seen in figure 5.10 on page 93, where it can be seen that the ECC technique performs well at predicting resistance to nevirapine, to which patients are overwhelmingly more resistant than susceptible. The same method also performs well at predicting susceptibility to tenofovir, to which patients are much more susceptible than resistant.

Classification Technique Effects

The remarkably different results obtained by the BR-SVM and BR-NB techniques demonstrate the effect that the choice of base classifier can have (see tables 5.9 and 5.12). Therefore, the comparison of the performance of different multi-label classification techniques that make use of base classifiers is only meaningful when each uses the same base classifier. In this study the same base classifier is used for all of the BR-NB, HOMER, RAkEL and ECC techniques. PCT does not require a base classifier.

The BR-NB and HOMER techniques show very similar performance, with the main difference being that the HOMER method has a much lower mean standard deviation on dataset 2. This is consistent with the literature [30]. The RAkEL and ECC techniques both also produce lower mean standard deviations on dataset 2. It can thus be suggested that the HOMER, RAkEL and ECC methods produce more consistent predictions as the training dataset grows, while the consistency of the predictions produced by BR-NB remains unchanged. Although the performance of the PCT method on dataset 2 is not improved, the mean standard deviation is still less than on dataset 1. The mean standard deviations of the BR-SVM and MLkNN methods are higher on dataset 2 than on dataset 1.

The performance of the RAkEL technique corresponds with what has been reported by previous studies [27]. In this study it can be seen that RAkEL performs better than both the binary relevance method (BR-SVM and BR-NB) and MLkNN. Furthermore, it has also been demonstrated that RAkEL does not perform as well as ECC [106], which is supported by our results.

The mean *AUC* values for the ECC method (the best performing method in terms of the *AUC* measure) are 0.63 on dataset 1 and 0.75 on dataset 2. These results are comparable to the results obtained by two

recent studies [8,36], both of which investigate the application of classification techniques to the problem of HIV treatment failure. One study [8] made use of *random forest* classifiers trained using 14891 TCEs and did not include genotype resistance test information. The TCEs were extracted from a research database containing patient records from North America, Europe, Australia and Japan. The *random forest* models obtained *AUC* values in the range $0.74 - 0.77$. The other study [36] used the ECC method and was trained on a dataset that consisted of 600 training examples that included GRT data. The training examples were extracted from a North American research database and the ECC models achieved *AUC* values in the range $0.77 - 0.91$. The models examined in this study were trained using less data, different machine learning techniques and data from a rural South African context that does not included GRT information. The fact that the results achieved are still comparable to those found in the literature is a promising outcome.

The Accuracy Paradox

Many of the techniques (BR-NB, HOMER, MLkNN, RAkEL) show a reduced mean TPR on dataset 2, which could be interpreted to mean that the techniques performed worse in general on dataset 2 than on dataset 1. However, this is not the case and the reduced TPR values are observed due to the *accuracy paradox* [95]. The accuracy paradox occurs when a predictive model demonstrates inflated performance on imbalanced datasets. An example of the accuracy paradox is given in section 2.2 using equation 2.1 (simple accuracy). Other examples of the accuracy paradox are the blanket positive predictions produced by BR-SVM and MLkNN in some cases. These blanket positives result in a TPR value of 1.0, which implies that the techniques identify positive examples in 100% of cases. However, it is known that these techniques do not provide any useful predictive information at all because the TNR value is 0.0.

In order to avoid misinterpreting the accuracy paradox as good performance (or interpreting reduced values for a single metric as overall worsened performance), it is important to consider multiple evaluation measures. This is the fundamental reason why *ROC* analysis is used (including *AUC*) in addition to four different scalar evaluation measures in this study. If TPR and TNR are considered together, it can be seen

that all techniques performed better on dataset 2 than on dataset 1. If all evaluation measures are considered, it can be seen that all but the *PCT* method have performed better on dataset 2 than on dataset 1 (although *PCT* did show improved standard deviation values on dataset 2).

Sensitivity And Specificity

The scalar evaluation measure values obtained indicate that many of the techniques (BR-NB, HOMER, PCT, RAkEL and ECC) are able to detect resistance to one or more drugs in over 95% of cases (as measured by TPR). The PCT technique is able to do so for three different drugs (efavirens, delavirdine and nevirapine) on dataset 1, while the ECC method achieves a TPR of 95% or higher for six drugs on dataset 2 (efavirens, emtricitabine, delavirdine, nevirapine, etravirine, lamivudine). However, although the TPR values are high, they are often accompanied by a low TNR value, indicating that the rate of false positives is high. This means that if the predictive models were used to prioritise patients for genotype resistance testing, a high number of patients could be tested unnecessarily. Fortunately this effect is reduced when the models are trained using dataset 2 and it is expected to further diminish as the size of the training dataset grows.

Despite this potential for suboptimal GRT prioritisation, there are still use cases where the models (still trained on relatively small datasets) could be used effectively in resource-limited settings. For example, if GRT is not available and/or if the care provider decides that a regimen change is necessary, the false positives can be interpreted as the models making conservative predictions. Clinically, it is important to try to retain patients as long as possible on first line treatment regimens since second line therapy costs as much as 2.4 times more than first line therapy and compromises treatment outcomes [7], but if the decision is made to switch regimen, conservative prediction results could be used to help design a treatment regimen to which the virus is susceptible by avoiding drugs for which the models produce TPR values close to 1.0.

Techniques like PCT, BR-NB and ECC have high values for all measures for those drugs on which they perform well. This should give us confidence that the decision support information provided is reliable and not influenced by a single outlier evaluation measure value. Conversely, we should have less confidence in

the performance of a technique for a drug if only one of the measures show good performance, for example, the MLkNN technique has a high TNR value (0.97) for stavudine, but all other metrics indicate very poor performance. This includes techniques that produce blanket positive predictions and therefore have a TPR value of 1.0. Despite this apparently high TPR value, these techniques provide no information that can be used to support decisions.

Data Dependencies

It has been shown that feature dependencies can sometimes have an effect on the performance of classification techniques [107, 108]. The simplifying assumption has been made (discussed in section 4.1.4) that all features are independent of each other. However, in reality, this is not the case since *viral load* has an effect on *CD4 count* because the virus has an impact on the human immune system (measured using *CD4 count*). This dependency (and possibly others) could have a negative impact on the performance of the classification techniques that make use of the naive Bayes classifier, which makes explicit use of the independence assumption. It has been shown that taking feature dependencies into account has the most positive effect on domains where decision tree learners (such as the PCT technique) perform significantly better than naive Bayes [107]. This is not the situation in the experiments that have been conducted, so there is not likely going to be large gain in performance by explicitly representing feature dependencies.

In this study the simplifying assumption has been made that all labels are independent of each other. We know from the literature that this is not true and this effect can even be observed in our own results [4]. For example, it can be seen in figure 5.7 on page 87 that both the BR-NB and HOMER method perform well for the emtricitabine and tenofovir labels. The drugs corresponding to these labels are both members of the same drug group (*nucleotide analogue reverse transcriptase inhibitors* or NRTIs). This effect can also be seen in figure 5.8 on page 89, where it can be seen that the PCT method performs best for the efavirens and nevirapine labels. The drugs corresponding to these two labels are both members of the *non-nucleoside reverse transcriptase inhibitor* (NNRTI) drug group. The fact that performance across related labels is consistent could mean that the classifiers do somehow implicitly encode label dependencies.

It is possible that the label independence assumption could have a negative impact on the performance of the classification techniques. Some studies have investigated the effect of explicitly modelling label dependencies and have obtained promising results [109, 110]. A very recent study conducted with the objective of predicting HIV drug resistance found that the ECC multi-label classification technique modelled label dependencies implicitly [36]. This implicit label dependency modelling can be seen in figure 3.3 on page 43, where it can be seen that each classifier in the classifier chain is trained using the labels from the previous classifiers as input. Although label dependencies have not been explicitly modelled in this study, we are still able to provide some evidence that such dependency modelling would improve performance, since the one technique used that implicitly models label dependencies is also the technique that performs best over all.

Temporal Nature of the Data

The patient data used to construct the training datasets consist of longitudinal records. This means that the information stored in the clinical information system for each patient has been captured over a period of time and each piece of information associated with a patient is also associated with a point in time. This temporal nature of the data has not been explicitly modelled in the features used for training, but some features do have an implicit temporal aspect. For example, the *baseline viral load* feature represents a viral load test performed on a patient within a certain time frame around the date on which the patient first started on antiretroviral therapy. Another example is *recent CD4 gradient*, which calculates the rate of change of the CD4 count measurements taken for a patient in the last year.

There exists another temporal aspect to the dataset, which is the fact that each patient captured in the system has been added to the database at a specific point in time. In other words, patients who were infected a long time ago were added to the system before recently infected patients. This temporal element has the potential to have an impact on our predictive models because of the way that HIV evolves. Recently infected patients may be infected with a strain of the virus that has already evolved resistance to some drugs (referred

to as *transmitted resistance*). There is no way to detect this from the clinical record and it is possible that such cases could have an impact on model performance. A solution to this problem is beyond the scope of this study, but some initial ideas on how to begin to address this issue are discussed in section 7.1.

Virtual Field Test

In addition to using the numerous scalar evaluation measures and *ROC* analysis along with 10 fold cross validation to evaluate and validate the models, two datasets have also been used. Using two datasets simulates the evolution of the dataset over time and has the effect of performing a *virtual field test*. That is, the models have been tested not only using real clinical data from the field, but the response of models to a change in the underlying dataset has also been tested. This has been achieved by using a snapshot of the clinical information system used at the Africa Centre at two different points in time.

The results of this virtual field test have been overwhelmingly positive, with all but one predictive model performing better on the dataset 2 than on than on dataset 1. The models also showed a reduced standard deviation on dataset 2, which implies that the predictions they produce become more consistent as the dataset grows. This improved performance gives us confidence not only that the models are robust under a change in the underlying data, but that they in fact improve as the dataset grows – a result that is very desirable.

6.3 Limitations

Although the results obtained in chapter 5 are promising, they must be interpreted with caution. The datasets used are very small in machine learning terms and there is a possibility that our results may not be generalisable to all patient datasets. This is further exacerbated by the fact that research datasets have been used, which are likely to be cleaner and more complete than the average dataset.

A further limiting factor is the imbalanced nature of the datasets used in this study. The data used comes from the Africa Centre treatment failure clinic and not from a general HIV treatment clinic, which would

also include patients not failing treatment. As a result, the datasets used for training the predictive models contain more patients with resistance than would typically be seen in the general population of HIV infected individuals. This could reduce the viability of applying these techniques in a general clinical setting.

While many methods have shown very good results (up to 97% true positive rate in some cases), there are many cases where the techniques are not very specific for some labels. This means that even though the techniques are able to correctly identify cases of resistance well (i.e. they are very sensitive), they are not able to detect susceptibility as well. The most extreme case of this is demonstrated by the blanket positive predictions generated by a few of the techniques. It has been demonstrated that this lack of sensitivity improves as the dataset grows, but the predictions produced by models trained on small datasets should be used with caution. One way to address this limitation is to evaluate performance using numerous evaluation measures.

The fact that feature and label dependencies have not been explicitly modelled may be an additional limitation of this research. As discussed in section 6.2 above, some features do have an impact on each other (CD4 count and viral load) and studies have shown that modelling label dependencies can have a positive effect. It is therefore possible that the predictive models could be further optimised by explicit data dependency modelling.

The final limiting factor has been touched on in section 6.2 above and has to do with the temporal nature of the underlying clinical datasets. The phase during which stratification is applied and the cross validation folds are generated does not take into account the time that each patient record was added to the dataset. It is therefore possible that examples representing very old patient records could fall into a test set, while the corresponding training set is comprised only of examples representing new patient records. This scenario is not representative of how models would be used in a real clinical setting, since the models would be trained using existing (older) data and a prediction would then be generated for the new (unseen) patient.

Despite these limitations, the models that have been trained and evaluated are still able to provide useful predictive information. In section 7.1 strategies are discussed to address each of these limitations with the

goal of improving the performance of the classification techniques and thereby increasing their potential impact in the field.

6.4 Potential Application

Resource-limited public health treatment programs are usually characterised by limited availability of GRT, second line therapy and highly skilled clinicians. In these settings, the predictive power demonstrated by the classifiers may be sufficient to facilitate and improve clinical decision making. If the primary goal is to optimise the use of GRT, then the techniques with the highest mean TNR values should be selected. This is PCT on dataset 1 (see table 5.8), while HOMER has the highest TNR on dataset 2 (see table 5.10). If the primary goal is new therapy selection, then BR-NB or ECC should be used, since these techniques have the highest mean TPR values on dataset 1 and 2 respectively (see tables 5.7 and 5.11). If it is of particular importance to know about the presence of resistance to a specific drug, then the method with the highest TNR for that drug should be selected, since this minimises false positives.

The rationale for this research was to address the issue of detecting HIV drug resistance in resource-limited settings such as South Africa. It is therefore necessary to discuss how the results of this research can be integrated into the current clinical management process being used to treat patients HIV patients. The Africa Centre works closely with the South African department of health on matters of HIV treatment and particularly HIV treatment failure. Many of the clinicians consulted during the course of this study have helped establish the drug resistance testing guidelines for South Africa [111]. Therefore, the treatment failure clinical model used by the Africa Centre as well as the use of RegaDB as the clinical information system is likely to scale up in the South African context. Since the processing pipeline has been designed using these tools, it would be easy to integrate the models examined in this study (or any models for which there is an implementation in Java) into the RegaDB system. This means that with a minimal amount of effort, it would be possible to provide the information produced by the predictive models to clinical care providers in real time.

At least one study has investigated the potential utility of using computational models as part of the HIV treatment process in Canada and Italy [9]. Twenty-three clinical care providers participated in the study, which made a web interface available that allowed the care providers to make use of a set of predictive models. The clinical care providers changed their treatment suggestions in 33% of cases based on the output of the predictive models and 99% of care providers indicated that they would use the system if it was available in practice. This study on utility, along with the results achieved by the models investigated using data from an African context supports the idea that the output of this research could have real world impact.

One further point that increases the possible impact of this research is the fact that the predictive models built in this study, except PCT, are all updateable. This means that as more data becomes available, the models can be adjusted to incorporate the newly available information without having to be rebuilt from scratch. This is advantageous in a production environment at scale, since for large datasets, rebuilding the predictive models from scratch may not be viable when predictions are needed within limited time frames. The fact that performance of the models improve with more data, and that new data can be added incrementally increases the viability of deployment in the field.

Chapter 7

Conclusion and Future Work

This work investigates and evaluates multi-label classification techniques for predicting HIV drug resistance. It was found that all models are able to provide useful predictive information for some drugs. The performance of the ECC technique is comparable to what has been found in the recent literature and can thus be considered on par with the state of the art.

Although the results of this research are promising, the claim is not made that predictive models trained using only clinical, adherence and demographic data can completely replace the need for genotype resistance testing at this stage. However, these techniques may be useful in resource-limited public health settings where decisions such as whether to remain on the same therapy, and if not, which new drugs to select, need to be made in the absence of a GRT result or a specialised clinician.

While none of the techniques stand out as a significantly good predictor for resistance to all drugs, some methods perform well on a subset of the drugs. For example, the ECC technique is good at detecting resistance to efavirens and nevirapine, achieving true positive prediction rates of 97% and *AUC* values of 0.83 and 0.82 respectively. This means that no single model should be relied on to predict resistance to all of the drugs. Instead, an ensemble of models should be constructed to take advantage of the strengths of all the models in order to produce a more accurate final prediction.

Possibly the most important result that has been obtained is evidence that the models produce more accurate and more consistent predictions as the underlying dataset grows. The advent of limited resistance testing in the public antiretroviral therapy program in South Africa and the development of national surveillance datasets will allow for the construction of much larger datasets and this could potentially drastically increase the accuracy of the predictive models [111].

In conclusion, significant evidence has been provided supporting the original hypothesis that multi-label classification techniques trained using only the demographic, routine treatment, laboratory and adherence data, can successfully predict the presence of HIV drug resistance in a patient, without the need for a genotype resistance test.

7.1 Future Work

During the course of this study, many future possible avenues of investigation have been identified. It has not been possible to investigate all of these given the scope of the project, but many of the ideas are discussed in the paragraphs that follow.

Large Datasets As larger datasets become available, it would be interesting to observe the impact on performance of the classification models. Current evidence suggests that the predictive accuracy and consistency will increase.

Feature Extraction It may be interesting to apply one or more feature extraction techniques such as *principle component analysis* [112]. These techniques have been shown to increase the performance of classification techniques. If this can be shown to be the case for these datasets, this process could be incorporated into the processing pipeline. *Clustering* techniques may also be used to identify which features are most influential in the predictions.

Real Field Test To test whether or not the models would be useful in a real clinical environment it is necessary to deploy and evaluate them in such an environment. There are a few tasks that would need to be completed before this could be done. First, the developed application would need to be integrated into RegaDB so that clinical care providers could see the predictive information within the RegaDB application. This would involve the small task of automating one or two processing pipeline steps that are not currently fully automated (i.e. the manual running of scripts). A mechanism would also have to be developed to detect when new data is added to the dataset and update the models accordingly. During the field test it would be interesting to evaluate the effect of an early warning system that makes use of the models. For example, an alert could be shown on the dashboard of a patient if the models predict that the patient is resistant to their current treatment.

Simulated Field Test If it is not possible to perform an actual field test, there is also the possibility of simulating a field test by taking the temporal nature of the underlying datasets into account. The dataset would need to be examined more closely and patient data should be incrementally added to the models in the same order in which they were added to the clinical information system. This would provide a fine-grained view of the performance of the models over multiple points in time. Currently there is only evidence of the performance increase of the models at two points in time.

Feature Augmentation There are different ways in which the training features can be constructed. For example, various techniques for modelling feature and label dependencies can be tested. Training and testing could be conducted with different subsets of the features and features could also be constructed that explicitly encode the temporal nature of the underlying data. To model label dependencies, drug groups could be used as the labels instead of using the individual drugs and the performance of the models could be evaluated using these new labels. Finally, an attempt could be made to extract more information from empty feature values and this information could be used during model training.

Ensembles In order to develop a model that performs better for a wider range of drugs, various ensembles of techniques can be investigated. Different combinations of multi-label classification techniques can be combined with the goal of producing more accurate results for more drugs.

Test More Base Classifiers There are very many single label classifiers available and it might be interesting to see how the performance of the multi-label classification techniques would change if different base classifiers were used, especially those that have been modified for use on imbalanced datasets [103, 104].

More Data Sources All of the data used for training has been extracted from the RegaDB clinical information system. It may be useful to extend the current framework to allow for data integration from multiple databases and potentially other types of data sources.

Decision Tree Validation The decision trees used by the PCT method are able to produce human readable decision trees. It would be very interesting to try to determine how well these decisions match up with the decision making process used by clinical care providers when they are deciding what course of action to take for patients who are failing HIV therapy.

Standard Deviation Per Measure Per Label It would be interesting to look at the individual standard deviations per measure for each label, since this would help to separate the effects of sample variation from those of algorithm differences.

Stratification by Length of Time Present in the Dataset In order to determine if it is better to have fewer patients with longer histories or more patients with shorter histories in the dataset, it would be useful to stratify by the length of the longitudinal records of patients. The following experiments could then be conducted:

- Train a classifier on dataset 2, but evaluate it on dataset 1.
- Reverse the above and train a classifier on dataset 1, but evaluate on dataset 2.

References

[1] P. Piot, M. Bartos, P. D. Ghys, N. Walker, and B. Schwartländer, "The Global Impact of HIV/AIDS," *Nature* **410,** 968–73 (2001).

[2] "Mid-Year Population Estimates," *Technical Report*, Statistics South Africa (2011) .

[3] T. Rossouw, T. de Oliveira, and R. J. Lessels, *HIV/TB Drug Resistance & Clinical Management Case Book* (South African Medical Research Council Press, 2013).

[4] F. Clavel and A. J. Hance, "HIV Drug Resistance," *The New England Journal of Medicine* **350,** 1023–35 (2004).

[5] T. F. Liu and R. W. Shafer, "Web Resources for HIV Type 1 Genotypic-Resistance Test Interpretation," *Clinical Infectious Diseases* **42,** 1608–18 (2006).

[6] K. Van Laethem, A. De Luca, A. Antinori, A. Cingolani, C. F. Perna, and A. Vandamme, "A Genotypic Drug Resistance Interpretation Algorithm That Significantly Predicts Therapy Response In HIV-1-Infected Patients," *Antiviral Therapy* **7,** 123–9 (2002).

[7] S. Rosen, L. Long, I. Sanne, W. S. Stevens, and M. P. Fox, "The Net Cost of Incorporating Resistance Testing into HIV/AIDS Treatment in South Africa: A Markov Model with Primary Data," *Journal of the International AIDS Society* **14,** 24 (2011).

[8] A. D. Revell *et al.*, "Computational Models Can Predict Response to HIV Therapy Without a Genotype and May Reduce Treatment Failure in Different Resource-Limited Settings," *Journal of Antimicrobial Chemotherapy* (2013).

[9] B. Larder *et al.*, "Clinical Evaluation of the Potential Utility of Computational Modeling as an HIV Treatment Selection Tool by Physicians with Considerable HIV Experience," *AIDS Patient Care and STDs* **25,** 29–36 (2011).

[10] M. Zazzi *et al.*, "Prediction of Response to Antiretroviral Therapy by Human Experts and by the EuResist Data-Driven Expert System (The EVE Study)," *HIV Medicine* **12,** 211–8 (2011).

[11] P. Libin *et al.*, "RegaDB: Community-Driven Data Management and Analysis for Infectious Diseases," *Bioinformatics* **29,** 1477–80 (2013).

[12] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "MULAN: A Java Library for Multi-Label Learning," *Journal of Machine Learning Research* **12,** 2411–14 (2011).

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations* **11,** 10–18 (2009).

[14] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013.

[15] T. M. Mitchell, *Machine Learning* (McGraw Hill, 1997).

[16] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer New York, 2006).

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (2008).

[18] I. Witten and E. Frank, *Data Mining Practical Machine Learning Tools and Techniques* (Elsevier, 2011).

[19] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining Multi-Label Data," in *Data Mining and Knowledge Discovery Handbook* (Springer, 2010), pp. 667–85.

[20] G. Tsoumakas and I. Katakis, "Multi-Label Classification: An Overview," *International Journal of Data Warehousing and Mining* **3,** 1–13 (2007).

[21] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An Extensive Experimental Comparison of Methods for Multi-Label Learning," *Pattern Recognition* **45,** 3084–3104 (2012).

[22] A. K. McCallum, "Multi-Label Text Classification with a Mixture Model Trained by EM," *Proceedings of the AAAI 99 Workshop on Text Learning* pp. 1–7 (1999).

[23] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning Multi-Label Scene Classification," *Pattern Recognition* **37,** 1757–71 (2004).

[24] G. Qi, X. Hua, Y. Rui, and J. Tang, "Correlative Multi-Label Video Annotation," *Proceedings of the 15th International Conference on Multimedia* pp. 17–26 (2007).

[25] A. Clare and R. King, "Knowledge Discovery in Multi-Label Phenotype Data," in *Principles of Data Mining and Knowledge Discovery* (Springer, 2001), pp. 42–53.

[26] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical Multi-Label Prediction of Gene Function," *Bioinformatics* **22,** 830–6 (2006).

[27] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multi-Label Classification of Music into Emotions," in *Proceedings of the 9th International Conference of Music Information Retrieval*, pp. 325–30 (2008).

[28] A. Wieczorkowska, P. Synak, and Z. W. Ras, "Multi-Label Classification of Emotions in Music," in *Intelligent Information Processing and Web Mining* (Springer, 2006), pp. 307–15.

[29] J. Read, "A Pruned Problem Transformation Method for Multi-label Classification," in *Proceedings of the 2008 New Zealand Computer Science Research Student Conference*, pp. 143–150 (2008).

[30] G. Tsoumakas, "Effective and Efficient Multilabel Classification in Domains with Large Number of Labels," *Proceedings of the 2008 Workshop on Mining Multidimensional Data)* (2008).

[31] E. L. Mencia and J. Furnkranz, "Pairwise Learning of Multilabel Classifications with Perceptrons," *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks* pp. 2899–2906 (2008).

[32] S. Park and J. Fürnkranz, "Efficient Pairwise Classification," in *Machine Learning: ECML 2007*, Vol. 4701 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2007), pp. 658–65.

[33] E. Spyromitros, G. Tsoumakas, and I. Vlahavas, "An Empirical Study of Lazy Multilabel Classification Algorithms," in *Artificial Intelligence: Theories, Models and Applications* (Springer, 2008), pp. 401–06.

[34] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-Label Classification," in *Machine Learning and Knowledge Discovery in Databases*, Vol. 5782 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2009), pp. 254–69.

[35] G. Tsoumakas and I. Vlahavas, "Random k-Labelsets: An Ensemble Method for Multilabel Classification," in *Machine Learning: ECML 2007*, Vol. 4701 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2007), pp. 406–17.

[36] D. Heider, R. Senge, W. Cheng, and E. Hüllermeier, "Multilabel Classification for Exploiting Cross-Resistance Information in HIV-1 Drug Resistance Prediction," *Bioinformatics* **29,** 1946–52 (2013).

[37] M. Zhang and Z. Zhou, "ML-KNN: A Lazy Learning Approach to Multi-Label Learning," *Pattern Recognition* **40,** 2038–48 (2007).

[38] X. Luo and A. Zincir-Heywood, "Evaluation of Two Systems on Multi-Class Multi-Label Document Classification," in *Foundations of Intelligent Systems*, Vol. 3488 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2005), pp. 161–169.

[39] H. Blockeel and L. De Raedt, "Top-Down Induction of First-Order Logical Decision Trees," *Artificial Intelligence* **101,** 285–97 (1998).

[40] S. Safavian and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *Systems, Man and Cybernetics, IEEE Transactions* **21,** 660–674 (1991).

[41] J. R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann, 1993).

[42] S. Ihara, *Information Theory for Continuous Systems* (World Scientific, 1993).

[43] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," in *Computational Learning Theory*, **55,** 23–37 (Springer, 1996).

[44] R. Schapire and Y. Singer, "BoosTexter: A Boosting-Based System for Text Categorization," *Machine Learning* **39,** 135–68 (2000).

[45] F. Comité, R. Gilleron, and M. Tommasi, "Learning Multi-Label Alternating Decision Trees from Texts and Data," in *Machine Learning and Data Mining in Pattern Recognition*, Vol. 2734 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2003), pp. 35–49.

[46] N. Ueda and K. Saito, "Parametric Mixture Models for Multi-Labeled Text," in *Advances in Neural Information Processing Systems*, pp. 721–28 (2002).

[47] M. Zhang and Z. Zhou, "Multi-Label Neural Networks with Applications to Functional Genomics and Text Categorization," *Knowledge and Data Engineering, IEEE Transactions* **18,** 1338–51 (2006).

[48] K. Crammer and Y. Singer, "A Family of Additive Online Algorithms for Category Ranking," *The Journal of Machine Learning Research* **3,** 1025–58 (2003).

[49] E. Sapozhnikova, "Multi-Label Classification With ART Neural Networks," *Second International Workshop on Knowledge Discovery and Data Mining* **2,** 144–47 (2009).

[50] G. Carpenter and S. Grossberg, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *Neural Networks, IEEE Transactions* **3,** 698–713 (1992).

[51] A. Elisseeff and J. Weston, "A Kernel Method for Multi-Labelled Classification," in *Advances in Neural Information Processing Systems*, pp. 681–87 (2001).

[52] S. Godbole and S. Sarawagi, "Discriminative Methods for Multi-Labeled Classification," in *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 22–30 (2004).

[53] Z. Zhou, *Ensemble Methods: Foundations and Algorithms* (CRC Press, 2012).

[54] L. Breiman, "Random Forests," *Machine learning* **45,** 5–32 (1999).

[55] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Ensembles of Multi-Objective Decision Trees," in *Machine Learning: ECML 2007*, Vol. 4701 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2007), pp. 624–31.

[56] D. Kocev, "Ensembles for Predicting Structured Outputs," *Ph.D. Thesis*, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia (2012) .

[57] L. Breiman, "Bagging Predictors," *Machine Learning* **24,** 123–40 (1996).

[58] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the Accuracy of Prediction Algorithms for Classification: An Overview," *Bioinformatics* **16,** 412–24 (2000).

[59] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in *AI 2006: Advances in Artificial Intelligence*, Vol. 4304 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2006), pp. 1015–21.

[60] M. Sokolova and G. Lapalme, "A Systematic Analysis of Performance Measures for Classification Tasks," *Information Processing & Management* **45,** 427–37 (2009).

[61] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters* **27,** 861–74 (2006).

[62] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell Systems Technical Journal* **29,** 147–60 (1950).

[63] R. Fan and C. Lin, "A Study on Threshold Selection for Multi-label Classification," *Department of Computer Science, National Taiwan University* pp. 1–23 (2007).

[64] A. Altmann *et al.*, "Comparison of Classifier Fusion Methods for Predicting Response to Anti HIV-1 Therapy," *PloS ONE* **3,** e3470 (2008).

[65] M. C. F. Prosperi *et al.*, "Investigation of Expert Rule Bases, Logistic Regression, and Non-Linear Machine Learning Techniques for Predicting Response to Antiretroviral Treatment," *Antiviral Therapy* **14,** 433–42 (2009).

[66] M. C. F. Prosperi *et al.*, "Antiretroviral Therapy Optimisation Without Genotype Resistance Testing: A Perspective on Treatment History Based Models," *PloS ONE* **5,** e13753 (2010).

[67] M. Rosen-Zvi *et al.*, "Selecting Anti-HIV Therapies Based on a Variety of Genomic and Clinical Factors," *Bioinformatics* **24,** 399–406 (2008).

[68] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine* **2,** 559–72 (1901).

[69] B. Matthews, "Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme," *Biochimica et Biophysica Acta (BBA)–Protein Structure* **405,** 442–51 (1975).

[70] D. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies* **2,** 37–63 (2011).

[71] D. D. Blakely, E. Z. Oddone, V. Hasselblad, D. L. Simel, and D. B. Matchar, "Noninvasive Carotid Artery Testing: A Meta-Analytic Review," *Annals of Internal Medicine* **122,** 360–67 .

[72] S. Drăghici and R. B. Potter, "Predicting HIV Drug Resistance with Neural Networks," *Bioinformatics* **19,** 98–107 (2003).

[73] M. Prosperi, G. Ulivi, and M. Zazzi, "Statistical Comparison of Machine Learning Techniques for Treatment Optimisation of Drug-Resistant HIV-1," in *Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems*, pp. 427–32 (2007).

[74] S. E. Sinisi, E. C. Polley, M. L. Petersen, S. Rhee, and M. J. van der Laan, "Super Learning: An Application to the Prediction of HIV-1 Drug Resistance," *Statistical Applications in Genetics and Molecular Biology* **6,** 7 (2007).

[75] D. Wang *et al.*, "A Comparison of Three Computational Modelling Methods for the Prediction of Virological Response to Combination HIV Therapy," *Artificial Intelligence in Medicine* **47,** 63–74 (2009).

[76] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *Proceedings of the International Joint Conference on Artificial Intelligence*, **14,** 1137–43 (1995).

[77] Z. Zhou, "Class-Imbalance Learning," in *Ensemble Methods: Foundations and Algorithms* (CRC Press, 2012), pp. 171–179.

[78] F. Provost, "Machine Learning from Imbalanced Data Sets 101," in *Proceedings of the AAAI Workshop on Imbalanced Data Sets*, (2000).

[79] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the Stratification of Multi-Label Data," in *Machine Learning and Knowledge Discovery in Databases*, Vol. 6913 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2011), pp. 145–58.

[80] H. He and E. Garcia, "Learning from Imbalanced Data," *Knowledge and Data Engineering, IEEE Transactions* **21,** 1263–84 (2009).

[81] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special Issue on Learning from Imbalanced Data Sets," *ACM SIGKDD Explorations Newsletter* **6,** 1–6 (2004).

[82] B. Larder *et al.*, "The Development of Artificial Neural Networks to Predict Virological Response to Combination HIV Therapy," *Antiviral Therapy* **12,** 15–24 (2007).

[83] D. Wang, B. A. Larder, A. Revell, R. Harrigan, and J. Montaner, "A Neural Network Model using Clinical Cohort Data Accurately Predicts Virological Response and Identifies Regimens with Increased Probability of Success in Treatment Failures," *Antiviral Therapy* **8,** U99–U99 (2003).

[84] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–52 (1992).

[85] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning* **20,** 273–97 (1995).

[86] C. Hsu, C. Chang, and C. Lin, "A Practical Guide to Support Vector Classification," *Bioinformatics* **1,** 1–16 (2010).

[87] I. Rish, "An Empirical Study of the Naive Bayes Classifier," *Proceedings of the IJCAI Workshop on Empirical Methods in Artificial Intelligence* **3,** 41–46 (2001).

[88] G. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 338–45 (1995).

[89] Wikipedia, *Naive Bayes Classifier*, http://en.wikipedia.org/wiki/Naive_Bayes_classifier, Accessed 21 September 2013.

[90] A. Banerjee and J. Ghosh, "Scalable Clustering Algorithms with Balancing Constraints," *Data Mining and Knowledge Discovery* **13,** 365–95 (2006).

[91] J. Quinlan, "Induction of Decision Trees," *Machine learning* **1,** 81–106 (1986).

[92] T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer, "ROCR: Visualizing Classifier Performance in R," *Bioinformatics* **21,** 3940–41 (2005).

[93] Y. El-Manzalawy and V. Honavar, *WLSVM: Integrating libSVM into WEKA Environment*, Software available at http://www.cs.iastate.edu/~yasser/wlsvm, 2005, Accessed 4 August 2011.

[94] C. Chang and C. Lin, "LIBSVM," *ACM Transactions on Intelligent Systems and Technology* **2,** 1–27 (2011).

[95] X. Zhu and I. Davidson, *Knowledge Discovery and Data Mining: Challenges and Realities* (IGI Global, 2007).

[96] N. Beerenwinkel *et al.*, "The Individualized Genetic Barrier Predicts Treatment Response in a Large Cohort of HIV-1 Infected Patients," *PLoS Computational Biology* **9,** e1003203 (2013).

[97] B. R. Beaumont, *Types of Health Information Systems (IS)*, http://www.floppybunny.org/robin/web/virtualclassroom/chap12/s2/systems1.pdf, 2011, Accessed 17 April 2013.

[98] S. Rhee *et al.*, "Predictive Value of HIV-1 Genotypic Resistance Test Interpretation Algorithms," *The Journal of Infectious Diseases* **200,** 453–63 (2009).

[99] G. K. Robbins, B. Daniels, H. Zheng, H. Chueh, J. B. Meigs, and K. A. Freedberg, "Predictors of Antiretroviral Treatment Failure In an Urban HIV Clinic," *Journal of Acquired Immune Deficiency Syndromes* **44,** 30–7 (2007).

[100] J. Parienti, V. Massari, D. Descamps, A. Vabret, E. Bouvet, B. Larouzé, and R. Verdon, "Predictors of Virologic Failure and Resistance in HIV-Infected Patients Treated with Nevirapine- or Efavirenz-Based Antiretroviral Therapy," *Clinical Infectious Diseases* **38,** 1311–16 (2004).

[101] P. R. Harrigan *et al.*, "Predictors of HIV Drug-Resistance Mutations in a Large Antiretroviral-Naive Cohort Initiating Triple Antiretroviral Therapy," *The Journal of Infectious Diseases* **191,** 339–47 (2005).

[102] O. Okun, "Introduction to Feature and Gene Selection," in *Feature Selection and Ensemble Methods for Bioinformatics: Algorithmic Classification and Implementations* (IGI Global, Hershey, 2011), pp. 117–122.

[103] G. Wu and E. Chang, "Class-Boundary Alignment for Imbalanced Dataset Learning," in *Proceedings of the Workshop on Learning from Imbalanced Datasets*, pp. 49–56 (2003).

[104] R. Akbani, S. Kwek, and N. Japkowicz, "Applying Support Vector Machines to Imbalanced Datasets," in *Machine Learning: ECML 2004*, Vol. 3201 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2004), pp. 39–50.

[105] I. Mani and I. Zhang, "kNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction," in *Proceedings of the Workshop on Learning from Imbalanced Datasets*, pp. 1–7 (2003).

[106] M. Tahir, J. Kittler, K. Mikolajczyk, and F. Yan, "Improving Multilabel Classification Performance by Using Ensemble of Multi-label Classifiers," in *Multiple Classifier Systems*, Vol. 5997 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2010), pp. 11–21.

[107] M. Pazzani, "Searching for Dependencies in Bayesian Classifiers," in *Learning from Data*, Vol. 112 of *Lecture Notes in Statistics* (Springer New York, 1996), pp. 239–48.

[108] A. Jakulin and I. Bratko, "Analyzing Attribute Dependencies," in *Knowledge Discovery in Databases: PKDD 2003*, Vol. 2838 of *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2003), pp. 229–240.

[109] Z. Younes, F. Abdallah, and T. Denœux, "Multi-Label Classification Algorithm Derived From k-Nearest Neighbor Rule with Label Dependencies," in *Proceedings of the 16th European Signal Processing Conference*, pp. 1–5 (2008).

[110] M. Zhang and K. Zhang, "Multi-Mabel Learning by Exploiting Label Dependency," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10 pp. 999–1008 (ACM, New York, NY, USA, 2010).

[111] F. Conradie *et al.*, "The 2012 Southern African ARV Drug Resistance Testing Guidelines by the Southern African HIV Clinicians Society," *Southern African Journal of HIV Medicine* **13,** 162–67 (2012).

[112] I. Jolliffe, *Principal Component Analysis* (Wiley Online Library, 2005).