# COMPARISON OF PREDICTION ALGORITHMS FOR STUDENT PERFORMANCE PREDICTION

**A MASTER'S THESIS**

**IN**

**SOFTWARE ENGINEERING**

**ATILIM UNIVERSITY**

**BY**

**AMADOU BAH**

**JUNE 2018**

# COMPARISON OF PREDICTION ALGORITHMS FOR STUDENT PERFORMANCE PREDICTION

**A THESIS SUBMITTED TO**

**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**OF**

**ATILIM UNIVERSITY**

**BY**

**AMADOU BAH**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF**

**MASTER OF SCIENCE**

**IN**

**THE DEPARTMENT OF SOFTWARE ENGINEERING**

**JUNE 2018**

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____
Prof. Dr. Ali KARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____
Prof. Dr. Ali YAZICI
Head of Department

This is to certify that we have read the thesis "Comparison of Prediction Algorithms for Student Performance Prediction" submitted by "Amadou Bah" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

| | |
|---|---|
| _____ | _____ |
| Prof. Dr. Ali YAZICI | Asst. Prof. Dr. Ziya KARAKAYA |
| Co-Supervisor | Supervisor |

Examining Committee Members

Asst. Prof. Dr. Ziya KARAKAYA _____

Assoc. Prof. Dr. Murat KOYUNCU _____

Assoc. Prof. Dr. Murat ÖZBAYOĞLU _____

Date: June 28, 2018

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Amadou Bah

Signature:

To My Family

## ABSTRACT

## COMPARISON OF PREDICTION ALGORITHMS FOR STUDENT PERFORMANCE PREDICTION

Bah, Amadou

M.S., Software Engineering Department

Supervisor: Asst.Prof.Dr. Ziya Karakaya

Co-Supervisor: Prof.Dr. Ali Yazici

June 2018, 78 pages

This thesis investigates the application of six machine learning algorithms to student performance prediction, using datasets made up of only students information available at the Atilim University administrative systems. In addition, these learning algorithms were compared using four measures: Accuracy, Precision, Recall and F-measure. The study also investigates whether the number of courses predicted together is directly or inversely proportional to the performance of the classifiers used. A measure of the effects of data preprocessing as well as Correlation based Feature Selection (CFS) on the learning algorithms was also conducted, respectively. The algorithms used are: Naive Bayes, Logistic Regression, Multilayer Perceptron, SMO (based on Support Vector Machines), IBk (K-Nearest Neighbor) and J48 (C4.5 Decision Tree).

Naïve Bayes and IBk proved to be the best among the compared algorithms. The results also show that as the number of courses being predicted together increases, the prediction performance decreases. Data preprocessing and CFS are also found to generally improve the performance of the machine learning algorithms.



Keywords: Machine Learning, Student Performance Prediction, Data Preprocessing, Correlation-based Feature Selection.

# ÖZ

## ÖĞRENCİ PERFORMANSININ ÖNGÖRÜLMESİ İÇİN TAHMİN ALGORİTMALARININ KARŞILAŞTIRILMASI

Bah, Amadou
Yüksek Lisans, Yazılım Mühendisliği Bölümü
Tez Yöneticisi: Asst.Prof.Dr. Ziya Karakaya
Ortak Tez Yöneticisi: Prof.Dr. Ali Yazici

Haziran 2018, 78 sayfa

Bu tez, Atılım Üniversitesi Bilgi Sistemlerinde barındırılan öğrenci bilgilerinin oluşturduğu veri kümelerini kullanarak altı farklı makine öğrenmesi algoritmalarının öğrenci performansı tahminine uygulanmasını incelemektedir. Bu öğrenme algoritmaları şu dört ölçü kullanılarak karşılaştırılmıştır: Doğruluk, Kesinlik, Geri Çağırma ve F-ölçüsü. Çalışmada aynı anda çok sayıda ders başarısı tahmininin, kullanılan sınıflandırıcıların performansıyla doğrudan veya ters orantılı olup olmadığına da bakılmıştır. Ayrıca; veri ön işlemenin yanı sıra, Korelasyon temelli Özellik Seçimi (CFS)'nin öğrenme algoritmaları üzerindeki etkilerinin ölçümü gerçekleştirilmiştir. Kullanılan algoritmalar şunlardır: Naif Bayes, Lojistik Regresyon, Çok Katmanlı Perceptron, SMO (Destek Vektör Makineleri), IBk (K-En Yakın Komşu) ve J48 (C4.5 Karar Ağacı).

Naïve Bayes ve IBk, karşılaştırılan algoritmalar arasında en iyi sonuçlar vermiştir. Sonuçlar, birlikte tahmin edilen derslerin sayısı arttıkça tahmin performansının azaldığını da göstermektedir. Veri ön işleme ve CFS'nin, genellikle makine öğrenimi algoritmalarının performansını artırdığı görülmüştür.

Anahtar Kelimeler: Makine Öğrenimi, Öğrenci Performansı Tahmini, Veri Ön İşlemleri, Korelasyona Dayalı Özellik Seçimi.

## ACKNOWLEDGMENTS

I express my sincere appreciations to my supervisor Asst. Prof. Dr. Ziya Karakaya for his guidance and insight throughout the research. His high standards for academic deliverables and honest critiques brought out the best in me.

Thanks also go to my co-supervisor Prof. Dr. Ali Yazici who has been my adviser from day one. His guidance and encourage is deeply appreciated.

I would also like to thank my thesis committee members – Assoc. Prof. Dr. Murat KOYUNCU and Assoc. Prof. Dr. Murat ÖZBAYOĞLU – for their valuable suggestions and comments.

I must express my sincere gratitude to my family who have always been there for me unconditionally. I offer my sincere thanks to them, especially to my parents for showing me what family really means. I'll forever be grateful.

I am deeply grateful to all those who have contributed to the success of my academic career in one way or the other. Your financial, moral and service support would forever be remembered. THANK YOU ALL.

**TABLE OF CONTENTS**

## LIST OF TABLES

**TABLE**

## LIST OF FIGURES

**FIGURES**

## LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| ANN | - | Artificial Neural Network |
| CFS | - | Correlation-based Feature Selection |
| CGPA | - | Cummulative Grade Point Average |
| CMPE225 | - | Object-Oriented Programming |
| CMP251 | - | Discrete Computational Structures |
| CSCL | - | Computer-supported Collaborative Learning |
| DPP | - | Data Preprocessing |
| EDM | - | Educational Data Mining |
| FP | - | False Positive |
| GP | - | Grade Point |
| GPA | - | Grade Point Average |
| HCI | - | Human-Computer Interaction |
| IBk | - | Instance Based learner (K-Nearest Neighbour) |
| IRS | - | Intelligent Recommender System |
| J48 | - | C4.5 (Decision Tree) |
| K-NN | - | K-Nearest Neighbor |
| ML | - | Machine Learning |
| MLP | - | Multilayer Perceptron |

| | | |
|---|---|---|
| MSE | - | Mean Square Error |
| N | - | Negative (class) |
| P | - | Positive (class) |
| SE | - | Software Engineering |
| SE221 | - | Software Requirements Engineering (course) |
| SMO | - | Sequential Minimal Optimization. Trains Support Vector Classifier |
| SVM | - | Support Vector Machines |
| TP | - | True Positive |

# CHAPTER 1

# INTRODUCTION

Education has proven to be very crucial for the continuous development of humankind. This has led to the rapid awareness towards it over the years. Nowadays, governments and households do their best to make sure their people are educated. This has led to basic education being compulsory in certain parts of the world. Generally, achievements in education are used as a measure of one's knowledge as well as their likelihood to be successful.

Education nowadays is easier to access than it has ever been. However, Students performance widely varies in learning institutions. This can be attributed to cognition level, motivational levels and environmental factors [8]. Students' performance in school is of utmost importance to the involved parties, ranging from direct stake holders like the students themselves, teachers, learning institutions, to indirect stake holders like students' guardians, sponsors, education ministries and states at large. All these parties play their role into ensuring students succeed in school. The major challenges of higher education is the underperformance of students and low student retention rate [17]. This problem can be resolved by predicting students' performance and making timely interventions [15]. Instructors are able to predict intuitively which students would complete with flying colors. However, this approach is subjective, in a limited scale, usually considers only previous grades and in-term performance, and it is usually only possible after the instructor knows the student [14, 22].

Data mining can do a far better job in the prediction task, not to mention the reliability and large scale it can provide.In the past decade, data mining has grown to be a contemporary tool for developing knowledge management systems [26]. Data mining can be categorized into descriptive and predictive data mining. Descriptive data mining makes use of association rule mining, clustering, etc. to discover patterns that culminate into better decision making, while predictive mining builds models using rule set, decision tree, neural nets, support vectors, etc., to predict class of new dataset [17]. Many researchers have been working on predictions in various backgrounds and academic areas like MBA, nursing and computer background students [9]. These researches have applied data mining techniques in many sectors – from finance, medical, marketing, to health care. However data mining hasn't attracted much attention in education until recently [20]. Lately, data mining has been vastly used in education [3]. It is among the most popular students' performance analyzing techniques [10]. The application of data mining in education is called Educational Data Mining (EDM).

EDM is an emerging interdisciplinary research area that deals with the development of methods to explore data in an educational context. It uses computational approaches to analyze data in order to investigate educational questions [3]. This research area includes domains like performance prediction, student modeling, analysis and visualization of student data, recommendation systems, student learning curve analysis, etc. [3, 5, 16]. EDM engineers try to find out which factors contribute to students' performance, and build models that make best use of these factors to predict performance of students in school using Machine Learning algorithms.

A basic definition of Machine Learning is: "Machine learning is programming computers to optimize a performance criterion using example data or past experience." [1]. In other words, just like humans, the machine learns from experience and this knowledge helps it react accordingly on future encounters. For example, if a machine is taught from past data that 90% of the times when a student studies for 3 hours per day, they would pass their exams. Then when new data of a student who studies 3 hours per day is given to the machine, it can, on its own, guess (predict) that the student will pass their exams with 90%

accuracy. A machine learning model is built and learning is the process of optimizing parameters of the model using training data or past experience. The model maybe predictive, to make predictions in the future, or descriptive to acquire knowledge from data, or both. This thesis will focus on predictive analyses. Predictive analysis make use of machine learning that figures patterns from past data and uses it to guess the future with high confidence [27]. It has been used in many applications like understanding user behavior, analyzing population trends, or fraud detection [4]. Machine learning is increasingly being used in student performance prediction [8]. The main purpose of predictive analytics in students' performances is to identify weak students and timely help them improve.

## 1.1 Aim of Study

Most studies in the literature only focus on performance predictions in a particular course or the overall performance of a graduating student. The predictions are either related to a single course or the GPA (all courses) of a student. However, not many studies were conducted in predicting students' performance in more than one course together. This thesis aims at predicting student performance using only Atilim University academic data (no survey data). Another central objective of this thesis is to compare the performance of six prediction algorithms. These algorithms are: Naïve Bayes, Logistic Regression, Multilayer Perceptron, Support Vector Machines (SMO), K-Nearest Neighbor (IBk) and Decision Trees (J48). These algorithms are compared in predicting students performance in different courses.

Among the challenges faced in Machine Learning is Feature engineering. It is the modification of data for more suitability in machine learning [4]. Koutina and Kermanidis in their paper [7] state that feature selection and resampling techniques give better accuracy than applying only machine learning algorithms without these techniques. Pojon [4] claims that feature engineering provide better improvements than method selection. Just like feature engineering, feature selection can be categorized under the data preprocessing stage. This thesis also aims at investigating the effectiveness of data

preprocessing, as well as Correlation based Feature Selection (CFS) on machine learning algorithms, respectively. Two different experimental scenarios will be used in this study. All the above mentioned objectives would fall under the first experimental scenario.

This thesis also aims at investigating the relationship between the number of courses being predicted together and the performances results of the algorithms. This warranted the second experimental scenario. This scenario aims at using three courses – Software Requirements Engineering, Object-Oriented Programming and Discrete Computational Structures – to observe the performance progress as the number of courses is increased from 1, 2 to 3 in a given prediction.

Datasets in both scenarios in this study contain features that are made up of software engineering and mathematics course information in the preceding semesters together with few engineered features and the class feature – indicating whether a student passed or failed the course. For the first scenario, programming and mathematics courses in the first, second and third semesters together with GPA, gender, scholarship and other engineered features were used to predict whether a student would pass or fail the Data Structures course in the fourth semester. For the second scenario, since the courses that constitute it are third semester courses, programming and mathematics courses in the first and second semesters were used as predictors for all the 3 courses.

Weka – a machine learning suite used for data analyses – was used for building machine learning models and evaluating them. Default settings of these classifiers were used except for IBk (K-Nearest Neighbour) in which for every experimental setting, K value between 1 and 20 that produces the best results were selected. These algorithms were evaluated using four different Machine Learning measures: Accuracy, Precision, Recall and F-measure.

## 1.2 Research Questions

Considering the objectives of this study, the following questions were formulated:

1. Are predictive models good at predicting students' performance?
2. Which is better, predictions of a single course at a time or predicting different courses together?
3. Among the six algorithms compared in this study, which is the best prediction algorithm in the student performance prediction problem?
4. What are the most useful features in predicting students performance in the given context?
5. Is data preprocessing effective in student performance predictions?
6. Is Correlation based Feature Selection effective in student performance predictions?

## 1.3 Significance of study

Prediction of students' performance has gained a lot of interest in education [3, 23]. Currently, it is of high priority to most academic institutions of higher learning [9]. Thus comes the need for automating the process. When automated, students' performance prediction can decrease teachers' duty in assessments [14], be helpful in allocating scholarships [7], help poor performing students in a timely fashion as well as motivate excelling students [7, 24, 25].

Having recommendation systems that predicts performance of students in each individual course requires huge resources. Being able to predict all courses in a given semester together is a huge gain. This thesis shows a way of achieving this but also discorvered that the number of courses being predicted together is inversely proportional to the performance of machine learning algorithms.

Students' academic performance centers on diverse factors such as personal, socio-economic, psychological and other environmental variables [12]. Most researches use survey as well as other data to acquire educational background and demographic information [9, 18]. Some studies [18] use Grade Point (GP) of fundamental courses to predict final Grade Point Average of students. Survey and enrollment data was used by [11] to predict students' academic performance. In general, most student performance

predictions use survey data or data from an online learning platform. This thesis builds six different student performance prediction models with only Atilim University administrative academic records. Apart from age and gender, no demographic information nor psychometric data was used. This is very crucial in building prediction or recommendation systems that would require no additional information from students.

Identifying features that highly correlate with the class feature is very important in building prediction models. Studies have been conducted to identity such features or factors that contribute to better predictions of student's achievements in school [7]. Among the most significant factors are: sex, age, pre-university education, financial support, father's level of education and whether the student's residence is in the institution's area [28, 29, 30, 31]. However, in this thesis, only sex (gender), age and financial support (Scholarship) features were available in the datasets used. The thesis identifies the most essential features in the given experimental setting by applying Correlation based Feature Selection. This study is therefore, very significant since it identifies which features are most important when only academic records of offline or in-class learning environment are used. This is very important in building prediction models as well as identifying which courses have a greater correlation with the predicted course.

### 1.4 Thesis Structure

The rest of the thesis is organized as follows: Chapter 2 gives a background information of the methods and evaluation techniques used in this study. Chapter 3 discusses the literature review. Chapter 4 explains the research methodology used in this thesis. Chapter 5 explains how the data was prepared before given to the classifiers. Chapter 6 describes the implementation and presents the results acquired in the experiment. Chapter 7 evaluates the results. And finally, in Chapter 8 is the discussions and conclusion.

# CHAPTER 2

# BACKGROUND INFORMATION

## 2.1 Selected Methods

### 2.1.1 Decision Trees

Decision trees are graph structures that are built by creating nodes for each potential decision that partitions the data into fully or close to homogenous parts [4]. Decision trees apply a process called recursive partitioning that recursively partition the dataset in appropriate values to form a tree structure [4].

C4.5 is a popular algorithm used to build decision trees. It was developed by Ross Quinlan. It is an extension of his previous algorithm called ID3 [8, 11]. It uses a divide and conquer method to build a tree [2, 11]. C4.5 uses a gain ratio as attribute selection criteria. The attribute with the highest gain ratio is selected as splitting attribute [8]. The implementation of C4.5 in weka is called J48. In this thesis, J48 in weka was used as the decision tree classifier.

### 2.1.2 Naïve Bayes

Naïve Bayes is a simple machine learning algorithm that uses the Bayesian Theorem for classification. The Bayesian Theorem states that the probability of event $C$ given $X$ is equal to the probability of the event $X$ given $C$ multiplied by the probability of $C$ upon probability of $X$:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \quad 2.1$$

Where $C$ and $X$ are different events, $P(C)$ and $P(X)$ are the probability of $C$ and $X$ occurring respectively. $P(C|X)$ is the conditional probability of $C$ occurring given event $X$ [4] . Naïve Bayes has been found to perform better than some more complex learners on different problem sets [2]. Naïve Bayes assumes that the predicting features are independent of each other. That is it ignores the possible correlations among features.

The application of Naïve Bayes is shown below where the data in table 2.1 is used to build a model that classifies a student having a GPA of 2 in CPME113 and 4 in SE112 course.

**Table 2.1** Sample data for Naïve Bayes.

| CMPE113 | SE112 | Pass_CMPE226 |
|---|---|---|
| 3 | 1 | 0 |
| 2 | 3 | 1 |
| 3 | 4 | 1 |
| 2 | 1 | 0 |
| 1 | 4 | 0 |

Considering Table 2.1, given a student that has GPA of 2 in CPME113 and 4 in SE112 course, Naïve Bayes classifiers this student as someone who will passes or fails CMPE226 by using the Bayesian Theorem to calculate the probability of the student passing and the probability of the student failing respectively, then compares the results and classifiers the students as the class that has the highest probability. The below steps explains the approach of Naïve Bayes.

Step 1.

Calculate the probability of the student passing given his GPA in the course CMPE113 is 2 and SE112 is 4

$$P(Pass = 1 \mid CMPE113 = 2,\ SE112 = 4) = \frac{P(CMPE113 = 2,\ SE112 = 4 \mid Pass = 1)\ P(Pass = 1)}{P(CMPE113 = 2,\ SE112 = 4)} \quad 2.2$$

Applying the chain rule gives

$$P(Pass = 1 \mid CMPE113 = 2,\ SE112 = 4) = \frac{P(CMPE113 = 2 \mid Pass = 1)\ P(SE112 = 4 \mid Pass = 1)\ P(Pass = 1)}{P(CMPE113 = 2,\ SE112 = 4)} \quad 2.3$$

$$= \frac{0.5\ \times 0.5}{P(GPA = 3,\ Age = 12)} = \frac{0.25}{P(GPA = 3,\ Age = 12)}$$

Step 2

Calculate the probability of the student failing given his GPA in the course CMPE113 is 2 and SE112 is 4

$$P(Pass = 0 \mid CMPE113 = 2,\ SE112 = 4) = \frac{P(CMPE113 = 2,\ SE112 = 4 \mid Pass = 0)\ P(Pass = 0)}{P(CMPE113 = 2,\ SE112 = 4)} \quad 2.4$$

Applying the chain rule gives

$$P(Pass = 0 \mid CMPE113 = 2,\ SE112 = 4) = \frac{P(CMPE113 = 2 \mid Pass = 0)\ P(SE112 = 4 \mid Pass = 0)\ P(Pass = 0)}{P(CMPE113 = 2,\ SE112 = 4)} \quad 2.5$$

$$= \frac{0.33\ \times 0.33}{P(GPA = 3,\ Age = 12)} = \frac{0.11}{P(GPA = 3,\ Age = 12)}$$

Step 3

Compare the results.

Since the denominators are the same, only the numerators can be compared which shows that $P(Pass = 1 \mid CMPE113 = 2,\ SE112 = 4)$ is greater than $P(Pass = 0 \mid CMPE113 = 2,\ SE112 = 4)$ (0.25 > 0.11), then the model classifiers the student as a passing student.

### 2.1.3 Support Vector Machines

Support Vector Machines (SVM) is a supervised Machine Learning method used to classify both linear and nonlinear data. They can also be used for numeric predictions. SVM transforms the data into a higher dimension in order to find a linear optimal separating hyperplane (decision boundary) that separate classes [19]. SVM finds the hyperplane using critical instances at the boundaries called support vectors from each class and builds a discriminant function to separate them as widely as possible [2, 19]. The support vectors found using SVM can give great insight of the learned model [19].

SVM finds the maximum marginal hyperplane for a decision boundary. Such a hyperplane is the one that gives the widest separation between the classes [19]. This, with high probability gives the minimum classification error on test data (previously unseen data).

SVM has a good generalizing ability and faster than most methods [10]. It is claimed to have the highest prediction accuracy on students' performance prediction [10].

Weka's (weka.classifiers.functions.SMO) Sequential Minimal Optimization (SMO) algorithm for training a support vector classifier was used in this study. This algorithm globally replaces missing values and transforms discrete features into binary ones. It also normalizes the attributes by default.

### 2.1.4 K-Nearest-Neighbor

K-nearest-neighbor Machine Learning method was first described in the 1950s [19]. It is widely used in pattern recognition. Nearest-neighbor classifier are lazy learners because they require test data to be always available. They compare a given test instance with training instance to identify the similarities and based on the results, classify the test

instance as the class of its neighbors. A training tuple (instance) is described by say $n$ features. So all the training tuples would be represented and stored in a point in the $n$ dimensional pattern space. When given a new tuple to classify, the learner searches for $k$-nearest neighbors of the new instance and assign the new instance the most common class among its k-nearest neighbors [19]. When $k = 1$, that is 1-nearest neighbor (1NN), the classifier considers only the closest instance to the new instance to be classified.

"Closeness" here is defined in terms of distance metric. For nominal features, if $X_1$ and $X_2$ have the same value, the difference is considered as 0, and 1 otherwise. Other sophisticated schemes can have different scores for each unique value of an attribute [19]. On the other hand if the attributes are numeric, the Euclidean distance can be applied. The Euclidean distance between two points (or instances), such as $X_1 = (x_{11}, x_{12}, \dots, x_{1n})$ and $X_2 = (x_{21}, x_{22}, \dots, x_{2n})$ is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_1 i - x_2 i)^2} \tag{2.6}$$

The equation basically squares the difference of the corresponding numeric attribute values of $X_1$ and $X_2$, and accumulates it. The square root of the total accumulated sum is then taken. As shown in the equation above, KNN intrinsically assigns equal weight to each attribute. However, this can have a negative effect because some feature values can huge (e.g income). So it makes sense to normalize all features to fall between 1 and 0 for them to have the same weight. Min-max normalization can be applied to transform a value v of a numeric Attribute A to $v'$ in the range [0, 1]

$$v' = \frac{v - min_A}{max_A - min_A} \tag{2.7}$$

Where $min_A$ and $max_A$ are the minimum and maximum values of attribute A.

In case of numeric predictions, K-NN returns the average value associated with the real-valued labels of the k-nearest neighbors [19]. In this thesis, the data was normalized to a

scale in the range of [0, 1]. In this thesis weka's implementation of K-Nearest Neighbor called IBk was used.

### 2.1.5 Logistic Regression

The predictor feature in logistic regression follows the Bernoulli distribution having an unknown probability, say $p$. So the probability of success is $p$ and that of failure or a student failing in this case would be $1 - p$. Logistic regression estimates the unknown $p$ for any given linear combination of the predictor features (independent variables). The link between the predictor feature and the Bernoulli distribution is called the logit function. Logistic regression is solely based on the logit function – natural logarithm of odds ratio. As mentioned earlier, the probability is $p$ and the opposite is $1 - p$, then the odds ratio would be $\frac{p}{1-p}$. So the logit function is the natural log of the odds ratio, i.e

$$logit(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 \tag{2.8}$$

Now we solve for p, the antilog gives

$$\frac{p}{1-p} = e^{\beta_0+\beta_1 x_1} \tag{2.9}$$

$$p = e^{\beta_0+\beta_1 x_1}(1-p)$$

$$p = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}}$$

Where p is the probability of the class of interest such as a student failing, and $\beta$ is the regression coefficient [1, 38].

### 2.1.6 Multilayer Perceptron

Multilayer Perceptron is made up of more than one layer of perceptrons. It is an Artificial Neural Network (ANN) inspired by the human brain. ANN has three kinds of layers: input

layer – gateway to the network; hidden layers – where computation is done; and the output layer that outputs the results. The perceptron is the basic processing element. A perceptron has inputs and an output. The inputs can come from another perceptron or from the gateway of the network. Each input $x_j \in R, j = 1, \dots, d$ to a perceptron is assigned a connection or synaptic weight say $w_j \in R$. The output $y$ is the simplest weighted sum of the inputs [1].

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \tag{2.10}$$

Where $w_0$ is the weight coming from a bias unit $x_0$ which is always +1. $w_0$ is the intercept that avoids overfitting and makes the model more general. A perceptron with a single layer of weights can only approximate linear functions and cannot be used for nonlinear classification (discriminant) or regression [1].

A Multilayer Perceptron (MLP) has intermediate or hidden layers between the input and output layers. MLP is able to deal with nonlinear classification and nonlinear regression. MLP is a feedforward network in which when given an input x, "activation" propagates in the forward direction for the values of intermediate units say $z_h$ be computed. Each intermediate or hidden unit is also a perceptron and applies a nonlinear sigmoid function to its own weighted sum [1]

$$z_h = sigmoid(w_h^T x) = \frac{1}{1 + \exp[-(\sum_{j=1}^{d} w_{hj} x_j + w_{h0})]}, h = 1, \dots, H \tag{2.11}$$

And the outputs $y_i$, also perceptrons, take the output of the hidden units as inputs

$$y_i = v_i^T z = \sum_{h=1}^{H} v_{ih} z_h + v_{i0} \tag{2.12}$$

Where $v_{i0}$ are bias weights of the bias units in the hidden layer

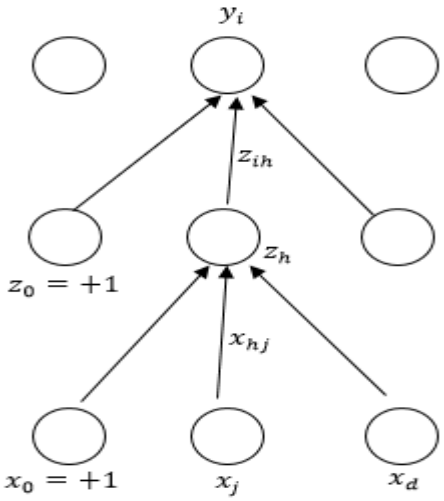



**Figure 2.1** Sample structure of multilayer perceptron. The input to the network are $x_j$, $j = 0, \dots, d$ and the hidden units with H dimensionality are $z_h, h = 1, \dots, H$. The bias of the input layer is $x_0$ and that of the hidden layer is $z_0$. The output of the units are $y_i, i = 1, \dots, K$. The weights of the first layer are $w_{hj}$ and that of the second layer are $v_{ih}$

## 2.2 Evaluation Methods

Machine learning algorithms are usually evaluated in terms of their accuracy. The accuracy of a learner is the overall rate of the correctly predicted instances or the recognition rate [19]. It is the percentage of test examples correctly classified [37]. A confusion metrics is used to present correctly classified instances to compute the accuracy. It is a table that shows the total number of instances versus the correctly and wrongly classified instanced.

**Table 2.2** Confusion matrix

| | | Predicted Class | | |
|---|---|---|---|---|
| | | Positive Prediction | Negative Prediction | Total |
| Actual Class | Positive (P) | ***TP*** | *FN* | P = TP + FN |
| | Negative (N) | *FP* | ***TN*** | N = FP + TN |
| | Total | $P' = TP + FP$ | $N' = FN + TN$ | P + N |

Table 2.2 shows a confusion metrics that comprises of two classes “Positive” and “Negative”. If the learner correctly classifies the “Positive” class, it is called True Positive (TP) and if it wrongly classifies the “Positive” class then it is called False positive (FP).

Likewise if a "Negative" class is correctly classified then it's called True Negative (TN) and otherwise called False Negative (FN). Since accuracy is the rate of correctly classified instances, then it is:

$$Accuracy = \frac{TP + TN}{P + N} \qquad 2.13$$

Accuracy seem to be a very genuine way of measuring performance of machine learning algorithm. However, it might not truly reflect how good an algorithm is. For instance, if there are 100 instances with 5% positive and the rest negative. An algorithm like ZeroR can just classify everything as the majority class and that would give it 95% accuracy. But that doesn't reflect the true capabilities of the machine. In order to show the true capabilities on both classes, especially the positive class which is usually less, the accuracy on classifying the positive class needs to be measured separately. And that is where Precision and Recall come in. Precision can be seen as: out of the total number of instances the machine classified as positive, what is the rate of the correctly identified instances. So in the above mentioned example, the Precision is 0. On the other hand Recall is the rate of the correctly classified instanced as positive out of the positive subsample. Again for the same example Recall is 0.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad 2.14$$

A better evaluation metrics is one that can combine both Precision and Recall. F-measure is 2 times the product of Precision and Recall divided by their summation. It considers both of them.

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad 2.15$$

Table 2.3 shows other measures. In this thesis, Accuracy, Precision, Recall and F-measure are the four evaluation metrics used.

**Table 2.3** Machine learning performance measures

| Measure | Formula |
|---|---|
| Accuracy, recognition rate | $\frac{TP + TN}{P + N}$ |
| Error rate, misclassification rate | $\frac{FP + FN}{P + N}$ |
| Sensitivity, true positive rate, Recall | $\frac{TP}{TP + FN} = \frac{TP}{P}$ |
| Specificity, true negative | $\frac{TN}{N}$ |
| Precision | $\frac{TP}{TP + FP}$ |
| F, harmonic measure of Precision and Recall | $F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$ |

## 2.3 Cross-Validation

When building machine learning models, the data is usually divided into training set – to train the machine – and test set – to test the capability of the machine. Given a large enough dataset $X$, it can be randomly divided into $K$ parts and each part be randomly divided into two equal training and test sets. $K$ is usually 10 or 30 [1]. The problem with this approach is that there is hardly a large enough dataset. A common alternative is cross-validation, which is the repeated use of the same data for training and testing, split differently for each iteration. This avoids overfitting – that is the machine performs very well on the small dataset, but when deployed and encounters new independent dataset, it then performs quite poorly because it wasn't prepared for that. Cross-validation gives the confidence of generalizing results to independent dataset.

In $K\text{-}fold\ cross\ validation$, dataset $X$ is randomly divided into $K$ equal parts $X_1, X_2, \dots, X_K$. One of the K parts is kept as the validation (test) set $V$ while the

remaining $K-1$ parts are used as the training set $T$ to form a training and test pair. This process is repeated $K$ times with each time choosing a different part as the validation set and the remaining as the test set. That is

$$\{T_i, V_i\}_{i=1}^{k} \quad 2.16$$

Where

$V_1 = X_1 \qquad T_1 = X_2 \cup X_3 \cup \ldots \cup X_K$

$V_2 = X_2 \qquad T_2 = X_1 \cup X_3 \cup \ldots \cup X_K$

.

.

.

$V_K = X_K \qquad T_K = X_1 \cup X_2 \cup \ldots \cup X_{K-1}$

Leave-one-out is an extreme kind of $K\text{-}fold\ cross\ validation$ where $K$ is the size of the sample. For example if there are 100 instances, Leave-one-out builds the model on 99 instances and leaves one. This is repeated 100 times to have 100 different models that are tested on the whole set. Stratified cross-validation maintains the distribution of the class in the folds. In this study $K = 10$, that is $10\text{-}fold\ cross\ validation$ was used for each of the experiments.

## 2.4 Correlation-based Feature Selection (CFS)

Genari *et al* states that a feature is relevant if it is predictive or correlated of the class, otherwise such a feature is not important for prediction [39]. A feature $X_i$ is correlated of the class C (or relevant) if it has a value $x_i$ and there exists a $c_j$ in C such that:

$$p(X_i = x_i) > 0 \quad \text{and} \quad p(C = c_j \,|X_i = x_i) \neq p(C = c_j) \quad 2.17$$

This basically means the probability of $c_i$ on the whole set shouldn't be the same as its probability when only instances where feature $X_i = x_i$ are considered. Some features

might be correlated of the class but redundant. A feature is said to be redundant if it is highly similar to another feature with respect to correlation of the class [39,40]. In other words, it is correlated of another feature.

Correlation based Feature Selection evaluates the worth of subset of features by considering their relevance in the prediction of the class and the degree of redundancy between them, then selects the most worthy subset [37].

This thesis examines the effects of CFS on learning algorithms by applying weka's CFS attribute selection filter with default settings.

# CHAPTER 3

# LITERATURE REVIEW

Murat Pojon [4] in his thesis examines the application of machine learning algorithms to predict whether a student will be successful or not. The specific focus of his research is the comparison of machine learning methods and feature engineering techniques in terms of how much they improve the prediction performance. He used 3 learning algorithms: linear regression, decision trees, and naïve Bayes classification. Feature engineering, the process of modification and selection of the features of a dataset [4], was used to improve predictions made by these learning algorithms. The author used two different datasets containing records of student information. The machine learning methods were applied to both the raw version and the feature engineered version of the datasets, to predict the student's success. The best algorithm was naïve Bayes classification for the first dataset, and decision trees for the second dataset, with 78 percent accuracy. Feature engineering was found to be more important factor in prediction performance than method selection in the data used in his study.

A study predicting the academic performance of students in a distance learning platform at the Hellenic Open University (HOU) was conducted by Kalles and C. Pierrakeas [6]. They analyzed the academic performance of the students in a year as recorded by homework assignments. Then they derived rules that explain and predict whether a student will pass or fail in the final exams. They used genetic algorithm based induction of decision trees. They had a class imbalance problem: classifiers had a poor accuracy results on the minority classes. They resolved this problem by using the resampling function in

WEKA, which oversamples the minority class and under samples the majority class. Their approach is better than the previous approaches they compared it with.

Another similar primary research was conducted at the University of Ionia [7]. They investigate some efficient machine learning technique in predicting the final grade of Informatics postgraduate students. Five academic courses were chosen, each constituting an individual dataset, and six well-known classification algorithms were experimented with. Furthermore, the datasets were enriched with demographic, in-term performance and in-class behaviour features. The small size of the datasets and the imbalance in the distribution of class values were the main research challenges of the present work. Several techniques, like resampling and feature selection, were employed to address these issues, for the first time in a performance prediction application, they claimed. Naïve Bayes and 1-Nearest Neighbor achieved the best prediction results.

Pamela Chaudhury *et al* [8] enhanced the capabilities of a performance system by investigated how different Data Preprocessing (DPP) techniques of normalization, discretization, handling of invalid entries and class imbalance problem affect classifier performance of SVM, Naïve Bayes and decision tree C4.5. They implemented these DPP techniques on student dataset. Both oversampling and under sampling technique were used to solve the problem. It was noted that oversampling enhanced the accuracy of the classifier compared to under sampling. Oversampling was achieved by SMOTE and random sample replication. Smote outperformed random sample replication and has been used by them. DPP measures improved the efficiency of all the classifiers and there was a significant reduction in error. The minority class is the class of interest in the context of student performance prediction as predicting the students who would perform poorly is vital. They achieved results with DPP measures enhancing the true positive rate for all the classifiers employed in the experiment.

There are many proposed student performance prediction models as well as feature engineering methods. One study at Universiti Teknologi MARA [18] decided to use results of early semesters as an input to predict the final CGPA of students at the end of

their studies. They present a study on Artificial Neural Network (ANN) model development in predicting academic performance of engineering students. Cumulative Grade Point Average (CGPA) was used to measure the academic achievement at semester eight. Students' results for the fundamental subjects in the first semester were used as independent variables or input predictor variables while CGPA at semester eight was used as the output or the dependent variable. Performances of the models were measured using the coefficient of Correlation R and Mean Square Error (MSE). The outcomes from the study showed that fundamental subjects at semester one and three have strong influence in the final CGPA upon graduation.

Shahiri et al. [10] conducted a systematical literature review for the purpose of improving students' achievements. They provide an overview on the data mining techniques that have been used to predict students' performance with a focus on how the prediction algorithm can be used to identify the most important attributes in students' data. They find the most often used attributes to be: cumulative grade point average (CGPA), internal assessment, demographic information, and external assessments, listed in order of importance. Some other attributes often used are extra-curricular activities, high school background, and social interaction network. Psychometric factor (student interest, study behavior, engage time, and family support) is also another but rarely used feature. Among the three tasks – classification, regression and categorization – used in predictive modeling for educational data mining, their study found out that classification method is the most commonly used. Under the classification techniques, Neural Network and Decision Tree are the two methods mostly used for predicting students' performance. Naïve Bayes, K-Nearest Neighbor and Support Vector Machine are also among the top used methods.

A study conducted by Osmanbegović and Suljić [11] used survey as well as enrollment data from the University of Tuzla to compare various methods and techniques of data mining in predicting student's success. They used Chi-square test, One R-test, Info Gain test and Gain Ratio test to assess the input variables. Each test was recorded using the following metrics: Attribute (name of the attribute), Merit (measure of goodness), Merit dev (deviation, i.e. measure of goodness deviation), Rank (average position occupied by

attribute), Rank and dev (deviation, deviation takes attribute's position). Different algorithms were applied and the results averaged to form the final result of attribute ranking. They found out "GPA" to be the most vital attribute followed by "entrance exam", "study material" and "average weekly hours dedicated to studying", while "number of household members", distance of residence from the faculty" and "sex" had the least impact on predictions. With WEKA they predict a two class problem – passed or failed – using three supervised data mining algorithms. The performance of these algorithms was evaluated based on their predictive accuracy, ease of learning and user friendly features. Naïve Bayes classifier did better than decision tree and neural network methods with regards to their predictions capabilities. The authors state that a good classifier model has to be both accurate and understandable for stakeholders in the education sector.

Prediction of students' performance in senior secondary schools in a district in Malaysia was conducted by Ramesh et al [12] with the objective of identifying factors influencing the performance of students in final examinations. In their investigation, they used a survey cum experimental methodology as their data source. Results from their hypothesis testing reveals that type of school does not influence the performance of the student in their research context while parents' occupation does. Multi Layer Perceptron algorithm is the best with accuracy of 72%.

Some studies predict the overall performance of students (CGPA) as opposed to this thesis which predicts student performance in one course and in three different courses. Goga et al [13] designed a framework of Intelligent Recommender System (IRS) that relies on background factors to predict students' first year overall academic performance as well as recommends suitable actions for improvement. The authors employed a multi-dimensional methodological approach. they performed an in-depth interview technique that identified appropriate students' background factors like father's and mother's educational qualification, family size, parent's occupations, etc. Ten classification trees – Random Forest, Random Tree, J48, Decision stump, REPTree, JRip, OneR, ZeroR, PART, Decision Table – and a multilayer perceptron (Artificial Neural Network) were

used in generating models using WEKA based on students' enrollment records from Babcock University. Accuracy level, confusion matrices and model building's speed were used as benchmarks in comparing the generated models. They then designed and built a framework of IRS using Java and MySQL, based on the outcome of their study. According to their models comparison, Random Tree outperformed all the other classifiers with an accuracy of 99.908%. Random Forest, REPTree, J48, JRip, PART, Decision Table and MLP also did well with the lowest accuracy being 96.78%.

In the pursuit of a more usable prediction model and model representation, a study synthesizes learning analytics approaches, Educational Data Mining (EDM) and Human-Computer Interaction (HCI) theory using data from a collaborative geometry problem solving environment: Virtual Math Teams with Geogebra (VMTwG) [14]. They constructed 6 variables, Subject, Rules, Tools, Division of Labor, Community, and Object, by the operationalization of activity theory to holistically quantify students' participation in the CSCL (Computer-supported Collaborative Learning) course. This analysis of variables prior to the application of a model distinguished their approach from the previous ones (feature selection, Ad-hoc guesswork, etc.). Their approach diminishes data dimensionality and systematically contextualizes data in a semantic background. The underlying modeling technique for their system is Genetic Programming (GP).

The thesis herein also investigates the effects of Correlation based Feature Selection in the student performance prediction context. Mark A. Hall [37] in his doctoral thesis approached the problem of feature selection for machine learning using a correlation based approach. He evaluated CFS using natural and artificial datasets. C4.5, IBk Naïve Bayes were used. CFS was able to swiftly identify relevant features on the artificial data as long as the relevance of these features did not highly depend on other features. CFS eliminated more than half of the features on the natural domains' data. His results showed that CFS improved accuracy in most cases. It however, degraded performance of learners in cases where features that were highly predictive of few instance were eliminated.

# CHAPTER 4

# RESEARCH METHODOLOGY

## 4.1 Research Method

This thesis has two central objectives: 1) predicting student performance using only school administrative data and 2) comparing prediction algorithms. Upon surveying the literature, weak areas and gaps in the literature was realized. This led to expanding the scope of the research to investigating the relationship between the number of courses being predicted together and the performance of machine learning algorithms. The effects of data preprocessing and Correlation based Feature Selection were also examined respectively. Hence experimental research methodology is used in this thesis.

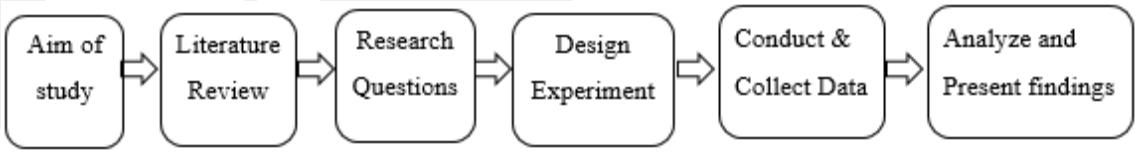



**Figure 4.1** Experimental Design Process

The research was approached using these six experimental research steps. After identifying the problem, or the aim of the study, the literature was reviewed to gather insight into the subject area as well as identify lacking areas in the literature. The research questions given in section 1.2 were then formulated.

This led to the design of the experiment. Section 4.2 explains the experimental setup used in the thesis. For conducting the experiment, the weka data analytical tool was used. Weka is an open source software with lots of machine learning algorithms implemented. Statistical data were collected, analyzed, and presented for evaluation purposes. The study was then concluded with the findings.

## 4.2 Experimental Setup

Two different experimental scenarios were used in this studies. The first scenario compares the algorithms as well as measures the effects of data preprocessing and Correlation-based Feature Selection (CFS), respectively, on the performance of the algorithms. On each dataset, feature engineering was applied to use mathematics and Software Engineering (SE) courses in preceding semesters as input variables. The first scenario's predicted course – Data Structures – is a fourth semester course and therefore all mathematics and SE courses in the preceding first, second and third semesters were used. The second scenario's datasets target class courses are made up of 3 third-semester courses - Software Requirements Engineering (SE221), Object-Oriented Programming (CMPE225) and Discrete Computational Structures (CMPE251). Hence courses in the first and second semesters were used as predictors.

Since this study also aims at investigating the effects of CFS and Data Preprocessing in the first scenario, the first scenario's data was used in four different forms: 1) raw data without attribute selection; 2) raw data with attribute selection; 3) preprocessed data without attribute selection; 4) preprocessed data with attribute selection. This resulted into 4 datasets used in the first scenario.

The study also investigates the correlation effects between the performance of machine learning algorithms and the number of courses being predicted together and hence the second scenario. The three courses in the second scenario were predicted together in all possible combinations and as a result, 7 different datasets were generated: SE221, CMPE225, CMPE251, SE221-CMPE225, SE221-CMPE251, CMPE225-CMPE251 and

SE221-CMPE225-CMPE251. Experimental scenarios one and two together forms 11 different datasets used in this thesis. Figure 4.2 shows the experimental setup.

The raw data was preprocessed – Filling missing values, Normalizing and discretizing - to give the “Preprocessed” dataset form.

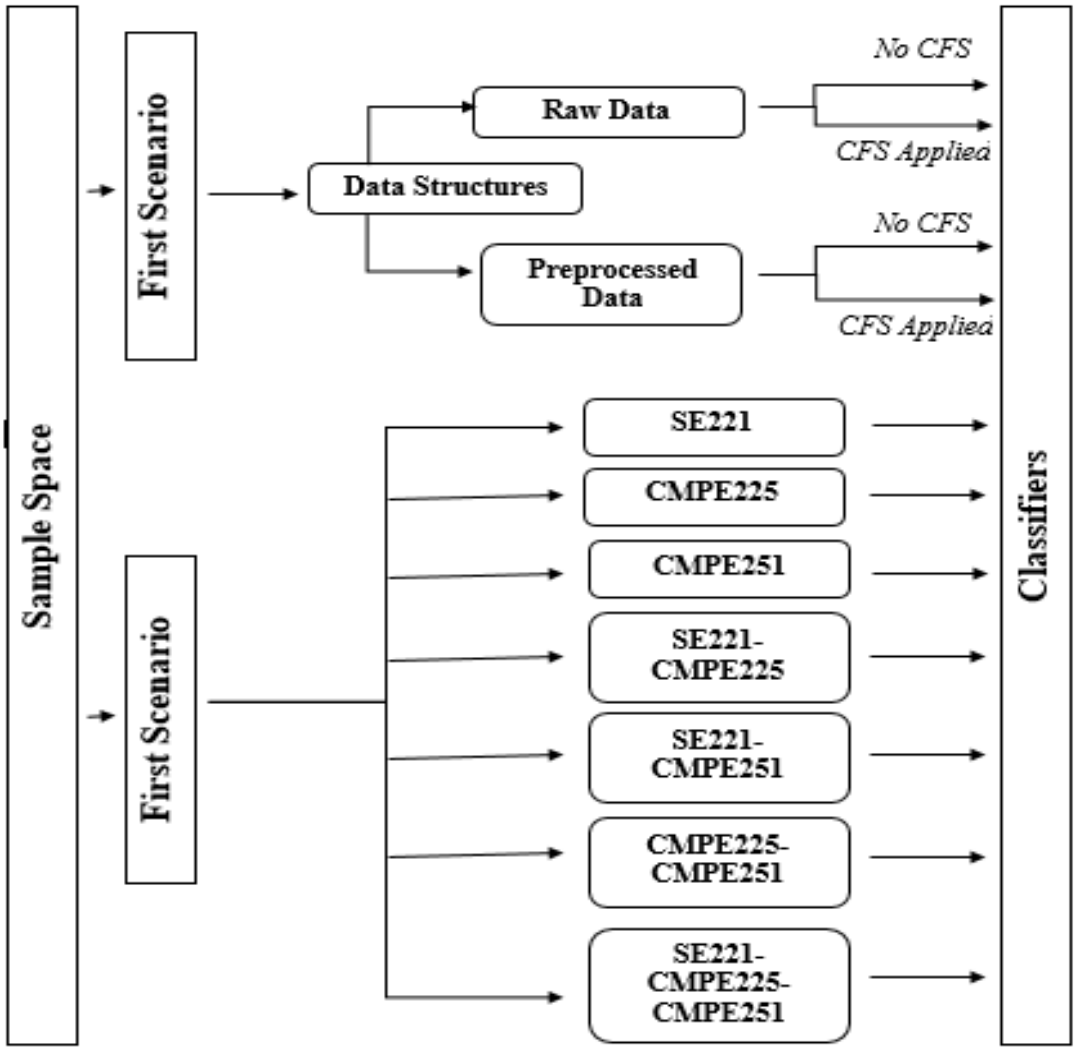



**Figure 4.2** Experimental Setup

Every algorithm has unique configuration depending on the type of algorithm in weka 3.8.1. In this study, for every algorithm being compared, the default configurations were

used except for K-nearest neighbor; the value of k from 1 to 20 that yielded the most optimal result was used.

10-fold cross validation was used when building the models. This divides the data into 10 folds and build 10 different models in which for every model, 9 folds are combined to train the model and one fold (different for every iteration) used to test it.

# CHAPTER 5

# DATA PREPROCESSING

Data collected through surveys and other various methods are susceptible to errors. Data preprocessing (DPP) makes sure the data is fully furnished to ensure high accuracy in predictions. The accuracy of learners hugely depend on the data [4]. In this case, the data is less prone to errors because it's collected from the university database system. Nonetheless, the data still needed some cleaning and had missing values. Additionally, since the data is only collected from the university systems, most of the features needed to be engineered. The following subsections describe the data and the preprocessing techniques applied.

## 5.1 Data Description

The data collected for this thesis comprises 185 Atilim University undergraduate Software Engineering students' data. Out of this data, 11 Mathematics and software engineering courses were used in this thesis. There are two experimental scenarios used in this study. The first scenario predicts a fourth-semester course using all mathematics and software engineering courses in preceding semesters 1, 2 and 3. The predicted course in the first scenario is data structures and its dataset was made up of 76 instances (unique students and their data).

Another objective of this study is to investigate the relationship between the number of courses being predicted and the performance of the prediction algorithms. Thus three courses were used in the second scenario predictions. These courses are third-semester

which were predicted again using mathematics and software engineering courses but only in the first and second semesters as predictors or independent variables. The third-semester courses predicted in this scenario are: Software Requirements Engineering (SE221), Object-Oriented Programming (CMPE225) and Discrete Computational Structures (CMPE251). All possible combination of these three courses gave 7 datasets for this scenario: SE221, CMPE225, CMPE251, SE221-CMPE225, SE221-CMPE251, CMPE225-CMPE251 and SE221-CMPE225-CMPE251. Since all students in this scenario took all the three courses, these 7 datasets all have the same predictors or independent variables. In other words they have the same exact data except for their class features. For example, suppose a student passed the courses SE221 and CMPE225 but failed CMPE251, this student will have the same exact instance in SE221, CMPE225 datasets, and the same with CMPE251 dataset except for the target class since s/he failed this course. Table 5.1 shows the courses used as predictors (independent variables) and target class (courses predicted) in both experiments – the Sem column refers to semester.

Some of these courses have prerequisites which need to be taken before the course in question is taken. For example CMPE225 is a prerequisite of CMPE226. That means all students who did CMPE226 must have already done CMPE225. Figure 5.1 shows this relationship with regards to only the courses used in this study. That is to say any course not in the diagram doesn't have a prerequisite but there is a possibility of it being a prerequisite of another course not used in the study. The arrows in the diagram point from the prerequisite course to the course in question.

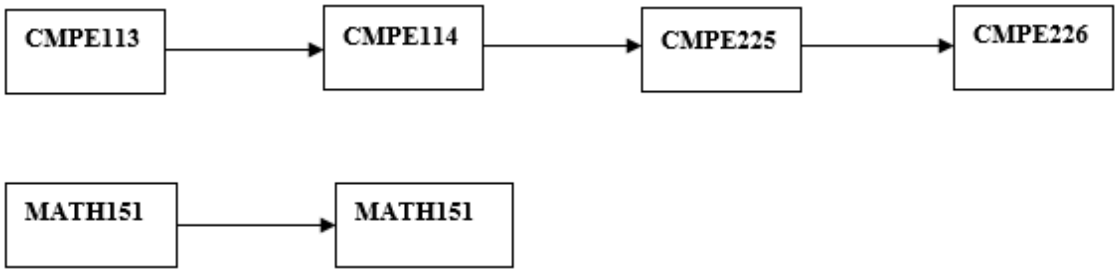



**Figure 5.1** Prerequisite diagram of courses used in this study.

**Table 5.1** Course Descriptions

| Sem. | Course Code | Course Name | Course Content |
|---|---|---|---|
| 1 | CMPE113 | Computer Programming I | Algorithm development, fundamental elements of the C language, selection statements, iteration statements, standard library functions, user-defined functions, parameter passing, application programs in a laboratory environment using the C language. |
| | CMPE109 | Fundamentals of Computing | Engineering fundamentals, computer engineering as a profession, career opportunities, professional organizations for computer engineering, ethical issues in computing; hardware components of a computer system; data representation and machine language instructions; coordinating internal activities of a computer using operating systems; networking |
| | MATH151 | Calculus I | Preliminaries, limits and continuity, differentiation, applications of derivatives, L`Hopital's Rule, integration, applications of integrals, integrals and transcendental functions, integration techniques and improper integrals. |
| 2 | SE112 | Introduction to Software Engineering | Computer Software and its types, Software Engineering discipline and its fundamental concepts, Software Process models, Requirements Engineering concepts, system modeling, architectural design, design and implementation, software testing, software evolution and maintenance, project management, quality and configuration management. |
| | CMPE114 | Computer Programming II | Pointers, dynamic memory management, parameter passing, arrays, strings, structures, file processing; application programs in a laboratory environment using the C language. |
| | MATH152 | Calculus II | Sequences, infinite series, vectors in the plane and polar coordinates, vectors and motions in space, multivariable functions and their derivatives, multiple integrals: double integrals, areas, double integrals in polar form, triple integrals in rectangular, cylindrical and spherical coordinates. |
| 3 | MATH275 | Linear Algebra | Linear equations and matrices, real vector spaces, inner product spaces, linear transformations and matrices, determinants, eigenvalues and eigenvectors. |
| | SE221 | Software Requirements Engineering | Basics of software requirement, requirements from the customer's perspective, applications in requirements engineering, the role of requirements analyst, forming product vision and project scope, understanding customer and user requirements, documenting |

| | | | |
|---|---|---|---|
| | | | requirements, risk reduction through prototyping, setting requirement priorities, validating |
| | CMPE225 | Object-Oriented Programming | Data types, expressions and statements, functions and scope rules, class definitions, inheritance, polymorphism, name overloading, templates, exception handling; input/output; object oriented principles using the UML and C++ programming language. |
| | CMPE251 | Discrete Computational Structures | Basic mathematical objects of computational mathematics: sets, sequences, relations, functions, and partitions; deductive mathematical logic proof techniques; discrete number systems; induction and recursion; graphs and sub-graphs; trees; planarity of graphs; covering problems; path problems; directed graphs; combinatorics. |
| 4 | CMPE226 | Data Structures | Stacks, recursion, queues; creation and destruction of dynamic variables, serial linked lists, circular lists, doubly linked lists, circular doubly linked lists; sorting and searching algorithms, space and time considerations, binary trees, binary search trees, tree traversal algorithms, binary tree sorting algorithms, hashing. |

Predictions of a single course is easy since it's just to say whether the student will pass or fail. But when two courses need to be predicted at the same time, the classes need to be generated in a way that would reflect grades of all courses being predicted. Such datasets can't have a dichotomous class feature since more courses need to be predicted at the same time. For that reason the class values are a two digit permutation of 0 and 1 representing a student's grade in the two courses in question. This approach yielded 4 classes for datasets in this group. "00", "01", "10", and "11". For example, the class value of "01" of dataset SE221-CMPE225 means the student failed (0) SE221 and passed (1) CMPE225. Likewise a value of "00" means they failed both and "11" means they passed both. Following the same approach for the dataset consisting of all the three courses, the following 8 classes were generated: "000", "001", "010", "011", "100", "101", "110", and "111". Tables 5.2, 5.3 and 5.4 show the classes used in the second scenario as well as the class distributions for each dataset.

The datasets in both the first and second scenarios included other software engineering and mathematics courses but these were used as predictors (independent variables) to

predict the target class (dependent variable), which is whether a student will fail or pass the given course or courses.

**Table 5.2** Single-course datasets' classes and class distributions

| Class | SE2221 Class Count | CMPE225 Class Count | CMPE251 Class Count | Class Description |
|---|---|---|---|---|
| 0 | 15 | 14 | 19 | Fail |
| 1 | 74 | 75 | 70 | Pass |

**Table 5.3** Double-course datasets' classes and class distributions

| Class | SE2221-CMPE225 Class Count | SE221-CMPE225 Class Count | CMPE225-CMPE251 Class Count | Class Description |
|---|---|---|---|---|
| 00 | 6 | 8 | 12 | Failed both |
| 01 | 9 | 7 | 2 | Failed first & passed second |
| 10 | 8 | 11 | 7 | Passed first & failed second |
| 11 | 66 | 63 | 68 | Passed both courses |

**Table 5.4** Three-course datasets' classes and class distributions

| Class | Class count | Description |
|---|---|---|
| 000 | 6 | Failed all three |
| 001 | 0 | Passed only CMPE251 |
| 010 | 2 | Passed only CMPE225 |
| 011 | 7 | Failed only SE221 |
| 100 | 6 | Passed only SE221 |
| 101 | 2 | Failed only CMPE225 |
| 110 | 5 | Failed only CMPE251 |
| 111 | 61 | Passed all three |

The data was cleaned to remove irrelevant features and instances. It was first cleaned to remove non-credit and preparatory courses. Then feature engineering was applied to generate new features like NoCoursesInSemester (Number of Courses a student is taking in the same semester the course at hand is being taken), MATH151 (grade in calculus I), etc. Filtering was then applied to extract only the relevant courses to this study.

Students' grades were in the letter grade form making up 9 classes. But they were further discretized from 9 to 2 classes ("Pass" and "Fail"). The pass mark is taken as DD (60%

mark). That is those who got DD grade or above were considered as passing students while less than DD is considered as a poor grade. The split into two classes in the first scenario's dataset yielded 22 Fail instances and 54 Pass instances. The same split was applied to each course in the second scenario. These classes were then combined depending on number of courses to predict together as shown in Tables 5.2 – 5.4.

The class imbalance problem as shown in the numbers was compensated with the enrichment of the data with feature engineering as well as feature selection. The following subsections elaborates on each of these steps and how the final data was arrived at.

This thesis also investigates the effects of data preprocessing. Each dataset in the first scenario was experimented on with and without data preprocessing (i.e. filling in missing values, normalization and discretization) resulting to two different datasets in the first scenario. For each of these two datasets, Correlation-based Feature Selection was applied to investigate the effect of CFS on the different algorithms studied herein. This again resulted to 4 different datasets in the first scenario. Figure 4.2 shows how the datasets were derived.

Each instance in both scenarios is made up of students' age, gender, prerequisite course details, performance details and some engineered features. Tables 5.5 and 5.6 show the set of features used after the preprocessing stage.

## 5.2 Data Transformation

The data received was in the Turkish language and had to be translated for readability. Features and their values were all translated to English. Spaces and special characters were also removed from nominal values of some features like the CourseCode.

Apart from the grades, there was also another value merging applied. Over the years, some courses had undergone course code changes. Such courses were identified and their codes merged to have the most recent course code.

There are two sets of courses: one for the courses whose grades are considered as predictors (independent variable) – used as features (e.g CMPE113); and the other set is made up of instances of courses to be predicted (e.g data structure course).

For the first set – the courses whose grades were used as input – their letter grades were converted to the corresponding number grades as shown in Table 5.7 in section 5.8. For the second set, students who acquired DD or above were considered as "Pass" students while those with less as "Fail" students. These sets were then merged to have the first set as predicting features and the second set as the target (dependent) variable. This was done for all datasets used in this study.

### 5.3 Data Cleaning

The data had some records that were not needed and were deleted in the feature engineering stage. Likewise there were some features that needed to be engineered before any data reduction. That being the case, feature engineering was applied both before and after the data cleaning stage. After the first engineering stage, all the preparatory courses were deleted; that is ENG101. This was due to the fact that the course had no credit hours and no program related course is taken while in the prep school, nonetheless these records were crucial for the engineered feature NoPrepSchoolSemesters. Other non-credit courses like SE399 (Summer Practice I) were also deleted. Courses with credit hours too would sometimes be given non-conventional letter grades like I (Incomplete), NA (Not Attended), etc. Records with such course grades were also removed from the dataset.

The data also had duplicate course records of students. These duplicates were as a result of students repeating a course after having an undesired grade. However, this study aimed at predicting students' performance in their first attempt of the prediction courses in the datasets. Thus only first-attempt records of these courses were selected. On the other hand, last records of courses that were used as predictors were sampled. For example, if CMPE225 is being used to predict CMPE226, the last repeated record of a student for CMPE225 course is considered since that is their final achievement in that course.

Conversely, the first attempt's record of a student for CMPE226 is considered since the objective is to predict students' performance on CMPE226 on their first attempt.

### 5.4 Feature Engineering

Since apart from age and gender there was no demographic, in-term or psychometric data (e.g. student interest, study behavior, engage time, and family support) available, new features needed to be scavenged from the available data to give better predictions. This process is called feature engineer.

As mentioned in the data cleaning stage, feature engineering was performed both before and after the data cleaning stage.

Students at Atilim University are usually conditionally admitted until they pass through the preparatory School. The preparatory school offers an English course (ENG101) which every student is required to pass before being unconditionally admitted. Some students just do the exams and proceed to their program studies while others spend from 1 to 4 semesters in the Preparatory school. Knowing how long a student spent studying English can contribute to the performance prediction since lectures are provided in English. This necessitated the extraction of the number of semesters spent in the preparatory school (NoPrepSchoolSemesters) before deleting these course.

After the cleaning phase, the second feature engineering was applied. Intuitively, a student with few courses in a semester is more likely to perform better than a student doing many courses. Thus the **NoCoursesInSemester** feature – the number of courses a student is taking while doing a particular course.

The data received had GPA and CGPA of students after taking a course. These two features are very crucial since they can differentiate performing and underperforming students. But while predicting, we want data available before a student takes the course, not after. Hence PrevSemesterGPA (Previous Semester GPA) and PrevSemesterCGPA (previous semester CGPA) were engineered to instead be the GPA and CGPA.

**Table 5.5** Features used in second dataset.

| Feature | Value | Description |
|---|---|---|
| Scholarship | None, OSYM50, OSYM100, KYK | The scholarship a student is benefitting from. If no scholarship then None is used. There are 12 different values |
| Gender | M, F | Student's gender (M for male and F for female) |
| Age | Numeric | The age of the student |
| NoPrepSchoolSemesters | Numeric | The number of semesters a student spent in preparatory school |
| Semester | Autumn, Spring, Summer | The semester the course was taken |
| NoCoursesInSemester | Numeric | The number of courses in the semester a student took the course |
| PrevSemesterGPA | Numeric | Students Grade Point Average in semester before this course |
| PrevSemesterCGPA | Numeric | Students Cumulative Grade Point Average in preceding semester |
| CMPE113_NoTimesTaken | Numeric | Number of times Computer Programming I was taken |
| CMPE109_NoTimesTaken | Numeric | Number of times Fundamental of Computing was taken |
| MATH151_NoTimesTaken | Numeric | Number of times Calculus I was taken |
| SE112_NoTimesTaken | Numeric | No. times Intro. to Software Eng. was taken |
| CMPE114_NoTimesTaken | Numeric | No. times Computer Programming II was taken |
| MATH152_NoTimesTaken | Numeric | Number of times Calculus II was repeated |
| CMPE113 | Numeric | Grade achieved in course (GPA range 0-4) |
| CMPE109 | Numeric | Grade achieved in course |
| MATH151 | Numeric | Grade achieved in course |
| SE112 | Numeric | Grade achieved in course |
| CMPE114 | Numeric | Grade achieved in course |
| MATH152 | Numeric | Grade achieved in course |
| Grade | Numeric | Grade achieved in course |

**Table 5.6** Features in addition to Table 5.5 formed the first scenario's dataset.

| Feature | Value | Description |
|---|---|---|
| MATH275_NoTimesTaken | Numeric | Number of times Linear Algebra course was repeated |
| SE221_NoTimesTaken | Numeric | Number of times Software Requirements Engineering was repeated |
| CMPE225_NoTimesTaken | Numeric | Number of times Object-Oriented Programming was repeated |
| CMPE251_NoTimesTaken | Numeric | Number of times Discrete Computational Structures was taken |
| MATH275 | Numeric | Grade achieved in course (GPA range 0-4) |
| SE221 | Numeric | Grade achieved in course |
| CMPE225 | Numeric | Grade achieved in course |
| CMPE251 | Numeric | Grade achieved in course |

As mentioned earlier in the data cleaning phase at section 5.3, latest grades of courses were used as input to the learners. However, a student who got AA in a prerequisite course in their first attempt of the course is more likely to pass the course being predicted than a student who got AA in the prerequisite course in their 3rd attempt. And this gave the need to not only capture grades of previous courses, but also how many times they were taken. And that gave features names like CMPE113_NoTimesTaken.

## 5.5 Removing the class imbalance problem

Student data is mostly, if not at all times unbalanced. Class-imbalanced data is when data of say two classes has little representation of the main class of interest (the positive class). If one is trying to predict whether a patient has cancer, there will most likely be fewer records of the positive class (has cancer) than the negative class (no cancer) because there are fewer cancer patients in the world. Likewise when predicting students' performance, the class of interest is the "Fail" class that identifies failing student.

Traditional classification algorithms assume that the weights of the positive and negative classes are equal and build models based on this assumption [19]. In order for a classifier to have the same efficacy in predicting the positive class as it has in predicting the negative, the two classes need to be seemingly balanced and this is achieved by undersampling or oversampling[8]. Oversampling and undersampling are among the general approaches that are used to improve accuracy of imbalanced data. These approaches balance the class distribution before the data is given to classifiers. Undersampling is when the majority (negative) class instances are reduced to match the number of positive class instances. Conversely, oversampling is when the positive class is increased to balance the class distribution.

Another approach is feature selection [33, 34, 35] which finds the best possible combination of features that gives the best results. This thesis maintained the original distribution of instances and enriched the data with new engineered features then applied CFS to minimize the effects of class imbalance.

Recall (true positive rate), Precision and F-measure are additional metrics used in the thesis to make sure the imbalance biasness is taken into consideration when comparing performance.

### 5.6 Missing values.

Even though the data is not survey data, it still had missing values that needed to be dealt with to improve the learners' accuracy. In the datasets used herein, only numeric features had missing values. Weka 3.8.1 filters were used to replace missing values (weka.filters.unsupervised.attribute.ReplaceMissingValues) with the default settings. Weka replaces numeric missing values with the mean.

### 5.7 Normalization

The scale of predictor course grades were from 0 to 4 while number of attempts in a course was more than 4. All numeric features were converted or normalized to a scale of 0 to 1.

Normalization ensures all features are treated equally by having the same weight as opposed to other features having more weight when that doesn't necessarily translate to being better predictors. Normalization is much needed in IBk which calculate distances from nearest neighbor using weights of features.

In this study, weka's normalize (weka.filters.unsupervised.attribute.Normalize) filter was used with default settings to convert all feature scales to a scale of [0,1].

### 5.8 Discretization of student data

Noise in data can be defined as a random error or variance in a variable [19]. One of the techniques of data smoothing or removing noise from data is Binning. Binning smooths sorted data value by checking values around it. Eventually, the sorted values are divided into a number of bins to form discrete (nominal) data. Hence the name data discretization.

An example of discretization is the letter grades. These grades are deduced from the numerical percentage score of students that lies between 0 and 100. Discretization allows these scores to be replaced with interval labels AA, BA, BB, CB, CC, DC, DD, FD and FF.

**Table 5.7** Atilim University Grading scheme with the translation of used classes

| % Score | Grade | Letter Grade | Class |
|---|---|---|---|
| 90 – 100 | 4 | AA | Pass |
| 85 – 89 | 3.5 | BA | Pass |
| 80 – 84 | 3 | BB | Pass |
| 75 – 79 | 2.5 | CB | Pass |
| 70 – 74 | 2 | CC | Pass |
| 65 – 69 | 1.5 | DC | Pass |
| 60 – 64 | 1 | DD | Pass |
| 50 – 59 | 0.5 | FD | Fail |
| 0 – 49 | 0 | FF | Fail |

In this study, the scores are further discretized to 2 classes (Pass and Fail) instead of the conventional 9 classes (AA, BA, etc)

Table 5.7 shows the Atilim University grading scheme with the corresponding classes assigned to each grade. The numeric "Grade" column was used for the predictors instead of the discrete "Letter Grade" column. On the other hand, the "Class" column was used as the classes in the data. In the second scenario, Pass is changed to 1 and Fail changed to 0. Refer to Tables 5.2, 5.3 and 5.4 for all possible classes in the second scenario.

## 5.9 Feature Selection

Feature selection is very crucial in enhancing machine learning methods [32]. Correlation-based Feature Selection (CFS) was the feature selection approach used in this thesis. Since this thesis also aims to investigate the effects of feature selection, CFS was applied on both the raw and preprocessed forms of the first scenario's dataset. CFS evaluates the possible subsets of the features by taking into consideration the individual predictive ability of each feature and the degree of redundancy between them. In other words, it finds features that highly correlate with the class and not much correlation between it and other features. CFS is earlier explained in detail in section 2.4.

# CHAPTER 6

# IMPLEMENTATION AND RESULTS

This study aims at comparing 6 prediction algorithms on a student performance prediction. The algorithms used on the datasets are Naïve Bayes, Logistic Regression, MLP, SMO, IBk and J48. Two experimental scenarios were conducted: 1) Predicts Data Structures grades and 2) Predicts Third semester Software Engineering courses' grades.

Weka – Waikato Environment for Knowledge Analysis – was used as the analytical tool in this study. Weka is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License. Detailed setup of the experiment was already talked about in section 4.2

Accuracy is the first measure used to compare the algorithms. But given a two-class unbalanced data – say 95% of the data is one class – an algorithm can get 95% accuracy by simply classifying the whole data as the majority class. ZeroR is such an algorithm that does exactly that. This shows accuracy alone doesn't determine the real capability of learners. Hence Precision, Recall and F-measure are also used. In the above example, Precision, Recall and F-measure would all be zero (showing bad learner) even though accuracy is 95%. In this study, ZeroR in weka is used to achieve a baseline accuracy on datasets. The first scenario's dataset – Data Structures prediction set – has 22 "Fail" instances and 54 "Pass" instances, and a baseline of 71.0526% on accuracy and 0 for the remaining measures. The second scenario's datasets – Software Requirements

Engineering, Object-Oriented Programming, and Discrete Computational Structures prediction sets – have numerous classes. Tables 5.2, 5.3 and 5.4 show their classes and class distributions.

In dichotomous feature classification, the class of interest is taken as the positive class. In the first scenario, the "Fail" class is the positive class because wrongly classifying a failing student is more serious than classifying a passing student as a failing one, and therefore, it is the class of interest. Thus True Positive in confusion matrixes refers to number of students correctly classified as "Fail". Likewise wherever Positive is mentioned, it refers to the "Fail" class. In the second scenario, since there are more classes, the weighted average measure was considered instead of a predefined Positive class.

## 6.1 First Scenario: Data Structures

### 6.1.1 Raw Data

This data is the initial dataset without data preprocessing. It was acquired by feature engineering more than 90% of the features. The only features that were not engineered are Scholarship, Gender and Semester. This dataset was made up of 76 data structures performance records. 71% of students passed the course and that was the accuracy baseline. Confusion matrix for each algorithm can be found in Table 6.1 with the actual values of classes being the horizontal records and the predicted values being the columns.

All algorithms performed better than the baseline accuracy with the highest being J48, 85.5%. The least performing algorithm was 1-nearest neighbor making an accuracy of 72.3684. Naïve Bayes produced better results in predicting failing students with a True Positive rate (or Recall) of 0.818. That is 81.8% of failed students were correctly classified which is a huge improvement compared to the baseline of zero for Recall.

**Table 6.1** Confusion matrix of Naïve Bayes, Logistic Regression, MLP, SMO, IBk and J48 on the first raw dataset.

| | **Predicted Failed** | **Predicted Passed** |
|---|---|---|
| **Naïve Bayes** | | |
| Actual Failed | 18 | 4 |
| Actual Passed | 9 | 45 |
| **Logistic Regression** | | |
| Actual Failed | 11 | 11 |
| Actual Passed | 15 | 39 |
| **Multilayer Perceptron** | | |
| Actual Failed | 14 | 8 |
| Actual Passed | 6 | 49 |
| **SMO** | | |
| Actual Failed | 13 | 9 |
| Actual Passed | 6 | 48 |
| **IBk** | | |
| Actual Failed | 8 | 14 |
| Actual Passed | 7 | 47 |
| **J48** | | |
| Actual Failed | 14 | 8 |
| Actual Passed | 3 | 51 |

### 6.1.2 Preprocessed Data

The missing values in the dataset were filled in with the modes of the features, for discrete features and mean of the features for numeric features. All numeric features were then normalized to the scale "[0,1]". The normalized form was then discretized.

The raw data of the first scenario's dataset was preprocessed and the performance of the learners measured. Out of the 28 predictors used in the first scenario's dataset, only 8 predictors were not converted into a one-value feature by the preprocessing stage.

Data preprocessing improved all algorithms in terms of their Accuracy and F-measure. Naïve Bayes performed best on the preprocessed data. Table 6.2 shows the confusion

matrix of all the 6 algorithms with “Fail” being the positive class and “Pass”, the negative class.

**Table 6.2** Confusion matrix of Naïve Bayes, Logistic Regression, MLP, SMO, IBk and J48 on the first preprocessed dataset.

| | **Predicted Failed** | **Predicted Passed** |
|---|---|---|
| **Naïve Bayes** | | |
| Actual Failed | 19 | 3 |
| Actual Passed | 4 | 50 |
| **Logistic Regression** | | |
| Actual Failed | 12 | 10 |
| Actual Passed | 6 | 48 |
| **Multilayer Perceptron** | | |
| Actual Failed | 13 | 9 |
| Actual Passed | 5 | 49 |
| **SMO** | | |
| Actual Failed | 15 | 7 |
| Actual Passed | 3 | 51 |
| **IBk** | | |
| Actual Failed | 16 | 6 |
| Actual Passed | 2 | 52 |
| **J48** | | |
| Actual Failed | 15 | 7 |
| Actual Passed | 4 | 50 |

### 6.1.3 Correlation-based Feature Selection on Raw Data

CFS finds features that have high correlation with the class and little or no correlation with other features. The results achieved in each case are presented in this section and the following section.

**Table 6.3** Confusion matrix of Naïve Bayes, Logistic Regression, MLP, SMO, IBk and J48 on CFS applied raw first scenario's dataset.

| | Predicted Failed | Predicted Passed |
|---|---|---|
| **Naïve Bayes** | | |
| Actual Failed | 18 | 4 |
| Actual Passed | 6 | 48 |
| **Logistic Regression** | | |
| Actual Failed | 12 | 10 |
| Actual Passed | 7 | 47 |
| **Multilayer Perceptron** | | |
| Actual Failed | 15 | 7 |
| Actual Passed | 8 | 46 |
| **SMO** | | |
| Actual Failed | 13 | 9 |
| Actual Passed | 5 | 49 |
| **IBk** | | |
| Actual Failed | 13 | 9 |
| Actual Passed | 4 | 50 |
| **J48** | | |
| Actual Failed | 15 | 7 |
| Actual Passed | 4 | 50 |

Weka's CFS was applied on the raw dataset and 7 predictors that produced the most optimal result were selected. These predictors are: Scholarship, PrevSemesterGPA, CMPE114_NoTimesTaken, CMPE251_NoTimesTaken, CMPE113, CMPE109 and CMPE251. Table 6.3 shows the confusion matrix of the algorithms on the first scenario's dataset with CFS applied.

### 6.1.4 Correlation-based Feature Selection on Preprocessed Data

CFS on the preprocessed form of the first scenario's dataset yielded only 2 features; PrevSemesterGPA and CMPE114_NoTimesTaken. All the six algorithms produced the same results. This can be attributed to the little data used, the very few features and the preprocessing applied that converted most features to one-value features.

**Table 6.4** Confusion matrix of all six algorithms (Naïve Bayes, Logistic Regression, Multilayer Perceptron, SMO, IBk and J48) on first preprocessed dataset with feature selection.

| | **Predicted Failed** | **Predicted Passed** |
|---|---|---|
| **Actual Failed** | 15 | 7 |
| **Actual Passed** | 1 | 53 |

## 6.2 Second Scenario

The second scenario measures the difference in performance of the machine learning algorithms on datasets made up of one, two to three courses. In other words, this experimental scenario used all six algorithms to build models on these individual courses, a combination of any of the two courses, and a combination of all the three courses. This resulted into seven datasets and hence seven different experiments.

Three third semester courses were used in this scenario. These courses are: Software Requirements Engineering (SE221); Object-Oriented Programming (CMPE225); and Discrete Computational Structures (CMPE251). For every dataset in this scenario, each student has only one record. That is one record of a student is made up of preceding semesters' grades that predicts the grades of the student in all three courses. And just like the first scenario, mathematics and software engineering courses in preceding semesters were used as inputs (predictors).

The datasets used in this scenario is made up of 89 students. All these students took all the three courses with some passing in a/some course(s) and failing in the other(s), while other students passed or failed all three courses. These 89 student records were reused for

every dataset with the difference being the target class. The distribution of the target class varies in every dataset as some students passed in other courses and failed in others. The resulting datasets are grouped into three groups: Individual course datasets; two courses datasets; and three courses dataset. The individual course datasets are datasets made up of only one course's classes: (1) SE221, (2) CMPE225, and (3) CMPE251. The two courses datasets are datasets made up of exactly two courses' classes: (1) SE221-CMPE225, (2) SE221-CMPE251, and (3) CMPE225-CMPE251. The third group is a dataset containing all three courses' classes: SE221-CMPE225-CMPE251. This gives a total of 7 datasets for this scenario. Since this scenario aims at investigating the proportional (directly/inversely) relationship between number of courses and prediction performance, it is important to note that all 7 datasets contain the same data (predictor/independent variables) except for their target classes. Tables 5.2, 5.3 and 5.4 in chapter 5 show the classes used in the second scenario.

Unlike the first scenario, there is no particular class of interest (Positive class) in this scenario. This is due to the fact that some datasets have more than two classes and most if not all classes are equally important. As a result, the weighted average of Precision, Recall, and F-measure of learners were recorded as opposed to that of a supposedly Positive class. Weka calculates weighted average as follows:

$$\frac{\sum_{i=1}^{n}(C_i \times \beta_i)}{\sum_{i=1}^{n} C_i} \qquad 6.1$$

Where $C$ is the number of instances in a class, $\beta$ is the resultant measure of the class (e.g Precision, Recall, F-measure, etc) and $n$ is the number of classes in the dataset.

The following subsections show the results of the learners on each dataset in the three groups.

### 6.2.1 One-Course Prediction

This group contains three datasets in which each of them has only one course's classes: SE221, CMPE225 and CMPE251. Each of these datasets have the same 89 instances but

with different course classes and class distributions. The dataset SE221 has 74 instances in the "1" (pass) class and 15 instances in the "0" (fail) class. CMPE225 has 75 in the "1" class and 14 in the "0" class. And finally CMPE251 has 70 instances of "1" and 19 instances of "0". Tables 6.5, 6.6 and 6.7 show the performance measure of the 6 algorithms on the three datasets respectively.

**Table 6.5** Performance results of algorithms on SE221 course dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 77.5281 | 0.799 | 0.775 | 0.785 |
| Logistic | 69.6629 | 0.746 | 0.697 | 0.718 |
| Multilayer Perceptron | 79.7753 | 0.775 | 0.798 | 0.785 |
| SMO | 83.1461 | 0.796 | 0.831 | 0.8 |
| IBk | 84.2697 | 0.816 | 0.843 | 0.818 |
| J48 | 78.6517 | 0.769 | 0.787 | 0.777 |

**Table 6.6** Performance results of algorithms on CMPE225 course dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 89.8876 | 0.896 | 0.899 | 0.897 |
| Logistic | 85.3933 | 0.867 | 0.854 | 0.86 |
| Multilayer Perceptron | 89.8876 | 0.896 | 0.899 | 0.897 |
| SMO | 91.0112 | 0.906 | 0.91 | 0.907 |
| IBk | 97.7528 | 0.978 | 0.978 | 0.977 |
| J48 | 85.3933 | 0.83 | 0.854 | 0.833 |

**Table 6.7** Performance results of algorithms on CMPE251 course dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 78.6517 | 0.809 | 0.787 | 0.795 |
| Logistic | 78.6517 | 0.799 | 0.787 | 0.792 |
| Multilayer Perceptron | 82.0225 | 0.828 | 0.82 | 0.823 |
| SMO | 79.7753 | 0.777 | 0.798 | 0.783 |
| IBk | 84.2697 | 0.833 | 0.843 | 0.836 |
| J48 | 85.3933 | 0.845 | 0.854 | 0.845 |

### 6.2.2 Two-Courses Predictions

This group too has three datasets but each dataset is a combination of exactly two out of the three courses' classes, thus the datasets: SE221-CMPE225, SE221-CMPE251 and

CMPE225-CMPE251. Refer to Table 5.3 in chapter 5 for the class distributions of these datasets.

**Table 6.8** Performance results of algorithms on SE221 and CMPE225 courses dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 70.7865 | 0.731 | 0.708 | 0.717 |
| Logistic | 61.7978 | 0.716 | 0.618 | 0.658 |
| Multilayer Perceptron | 69.6629 | 0.69 | 0.697 | 0.693 |
| SMO | 76.4045 | 0.716 | 0.764 | 0.733 |
| IBk | 80.8989 | 0.779 | 0.809 | 0.781 |
| J48 | 68.5393 | 0.679 | 0.685 | 0.682 |

**Table 6.9** Performance results of algorithms on SE221 and CMPE251 courses dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 67.4157 | 0.712 | 0.674 | 0.681 |
| Logistic | 55.0562 | 0.626 | 0.551 | 0.582 |
| Multilayer Perceptron | 68.5393 | 0.679 | 0.685 | 0.681 |
| SMO | 71.9101 | 0.643 | 0.719 | 0.676 |
| IBk | 76.4045 | 0.691 | 0.764 | 0.705 |
| J48 | 66.2921 | 0.626 | 0.663 | 0.644 |

**Table 6.10** Performance results of algorithms on CMPE225 and CMPE251 courses dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 73.0337 | 0.782 | 0.73 | 0.752 |
| Logistic | 73.0337 | 0.751 | 0.73 | 0.737 |
| Multilayer Perceptron | 86.5169 | 0.848 | 0.865 | 0.852 |
| SMO | 79.7753 | 0.764 | 0.798 | 0.777 |
| IBk | 85.3933 | 0.826 | 0.854 | 0.833 |
| J48 | 78.6517 | 0.76 | 0.787 | 0.773 |

Tables 6.8, 6.9 and 6.10 show the performances of the machine learning algorithms on two-course datasets.

### 6.2.3 Three-Courses Predictions

This group contains only one dataset as all the three courses' classes are combined in the same dataset. The class distribution of this dataset can be found in chapter 5 on Tables 5.2, 5.3 and 5.4

**Table 6.11** Performance results of algorithms on SE221, CMPE225 and CMPE251 courses dataset

| Algorithms | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Naïve Bayes | 64.0449 | 0.672 | 0.64 | 0.654 |
| Logistic | 57.3034 | 0.669 | 0.573 | 0.611 |
| Multilayer Perceptron | 65.1685 | 0.658 | 0.652 | 0.652 |
| SMO | 69.6629 | 0.686 | 0.697 | 0.659 |
| IBk | 70.7865 | 0.667 | 0.708 | 0.686 |
| J48 | 62.9213 | 0.582 | 0.629 | 0.604 |

# CHAPTER 7

# EVALUATION

## 7.1 First Scenario: Data Structures

### 7.1.1 Method Comparison

Decision trees produced the highest accuracy for the raw form of the first scenario's dataset, however Naïve Bayes was better at identifying failing students (Recall or True Positive rate) and had the highest F-measure. All six algorithms had an accuracy of at least 80% except Logistic Regression and K-nearest Neighbor which also had an F-measure of less than 0.5. Logistic Regression was the only algorithm that couldn't have an accuracy better than the baseline accuracy of 71.0526 %.

**Table 7.1** Method comparison for first raw dataset.

| **Method** | **Accuracy (%)** | **Precision** | **Recall** | **F-measure** |
|---|---|---|---|---|
| Naïve Bayes | 82.8947 | 0.667 | 0.818 | 0.735 |
| Logistic Regression | 65.7895 | 0.423 | 0.500 | 0.458 |
| Multilayer Perceptron | 81.5789 | 0.700 | 0.636 | 0.667 |
| SMO | 80.2632 | 0.684 | 0.591 | 0.634 |
| Nearest Neighbor | 72.3684 | 0.533 | 0.364 | 0.432 |
| Decision Tree | 85.5263 | 0.824 | 0.636 | 0.718 |

Nonetheless it is 50% (0.5 Recall) accurate at predicting failing students and better than nearest neighbor in this aspect. The order of performance according to accuracy is as

follow starting from the best. J48, Naïve Bayes, MLP, SMO, K-NN, and Logistic Regression. F-measure ranks them as follows starting from the best: Naïve Bayes, J48, MLP, SMO, Logistic Regress, and K-NN. Tables 7.1 and 7.2 show the results comparison of these algorithms on the raw and preprocessed data, respectively, for the first scenario's dataset.

**Table 7.2** Method comparison for first preprocessed dataset.

| **Method** | **Accuracy (%)** | **Precision** | **Recall** | **F-measure** |
|---|---|---|---|---|
| Naïve Bayes | 90.7895 | 0.826 | 0.864 | 0.844 |
| Logistic Regression | 78.9474 | 0.667 | 0.545 | 0.600 |
| Multilayer Perceptron | 81.5789 | 0.722 | 0.591 | 0.650 |
| SMO | 86.8421 | 0.833 | 0.682 | 0.750 |
| Nearest Neighbor | 89.4737 | 0.889 | 0.727 | 0.800 |
| Decision Tree | 85.5263 | 0.789 | 0.682 | 0.732 |

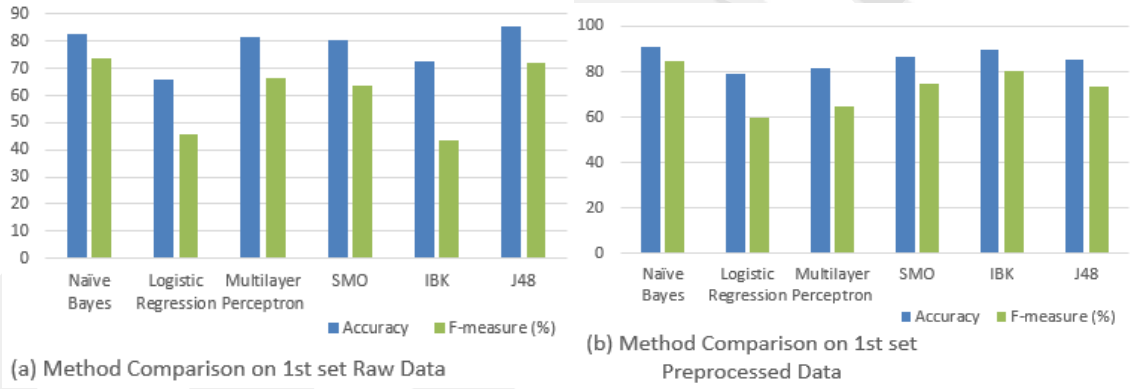



**Figure 7.1** Method comparison on first scenario's dataset (a) results from raw form (b) results from preprocessed form.

Unlike on the preprocessed data, the Accuracy order of the learners on the raw data corresponds to their F-measures. When comparing the F-measures, J48 was second after Naïve Bayes on the raw dataset and was only better than MLP and Logistic Regression on the preprocessed set. SMO on the raw data was only better than Logistic Regression and K-Nearest Neighbor but was third in line on the preprocessed dataset coming after Naïve Bayes and K-Nearest Neighbor. The order of performance among these algorithms is not

the same on the raw and preprocessed data. But still, on average, Naïve Bayes outperformed the other algorithms and Logistic Regression was not as good as the rest. Data Preprocessing on the first scenario's dataset improved generally improved the learners. The next section evaluates it in detail.

### 7.1.2 Data Preprocessing Improvement

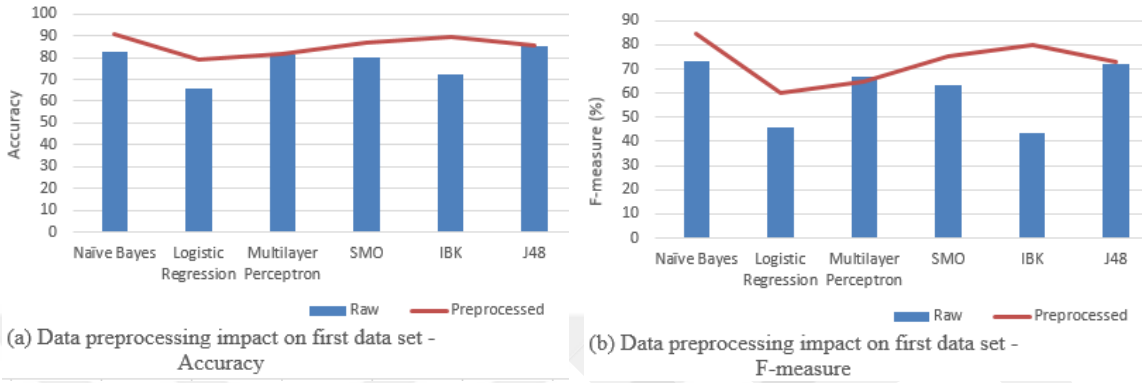
The performance of the learners improved or at least stayed the same after preprocessing. There was no decrease in accuracy. There was, however, a decrease in precision of J48, and MLP saw a decrease in both Recall and F-measure. Data Preprocessing shows a great improvement in IBk. It showed a difference of 17% in Accuracy and 37% in F-measure. Table 7.3 shows the difference in the measurement metrics used. That is the score in the raw dataset from the score in the preprocessed dataset. For example Naïve Bayes has a 7.8948% difference when preprocessing was applied. That is achieved by subtracting Accuracy on raw data from Accuracy on preprocessed data (90.7895% - 82.8947% = 7.8948%).

**Table 7.3** Data Preprocessing Improvements on First Scenario's Dataset.

| Algorithms | Accuracy Difference (%) | Precision Difference | Recall Difference | F-measure Difference |
|---|---|---|---|---|
| Naïve Bayes | 7.8948 | 0.159 | 0.046 | 0.109 |
| Logistic Regression | 13.1579 | 0.244 | 0.045 | 0.142 |
| Multilayer Perceptron | 0 | 0.022 | -0.045 | -0.017 |
| SMO | 6.5789 | 0.149 | 0.091 | 0.116 |
| IBk | 17.1053 | 0.356 | 0.363 | 0.368 |
| J48 | 0 | -0.035 | 0.046 | 0.014 |

Figure 7.2 shows the impact of data preprocessing on the data. 7.2 (a) shows the Accuracy of the algorithms on the raw data as columns and the accuracy on the preprocessed as line. It clearly shows the overall improvement on the data. Figure 7.2 (b) shows the same

improvement in F-measure. F-measure values are converted to percentage for a better presentation.




(a) Data preprocessing impact on first data set - Accuracy

(b) Data preprocessing impact on first data set - F-measure

**Figure 7.2** Data preprocessing improvement on Accuracy and F-measure of learners on first scenario's dataset.

### 7.1.3 Correlation based Feature Selection Improvement

Apart from Multilayer Perceptron, CFS had a positive impact on the first scenario's dataset, especially on Logistic Regression which shows about 12% difference in accuracy.

**Table 7.4** Correlation based Feature Selection Improvements on First Raw Dataset.

| Algorithms | Accuracy Difference (%) | Precision Difference | Recall Difference | F-measure Difference |
|---|---|---|---|---|
| Naïve Bayes | 3.9474 | 0.083 | 0 | 0.048 |
| Logistic Regression | 11.8421 | 0.209 | 0.045 | 0.127 |
| Multilayer Perceptron | -1.3157 | -0.048 | 0.046 | 0 |
| SMO | 1.3157 | 0.038 | 0 | 0.016 |
| IBk | 10.5263 | 0.232 | 0.227 | 0.235 |
| J48 | 0 | -0.035 | 0.046 | 0.014 |

Tables 7.4 and 7.5 show the difference in the unfiltered and CFS applied on the raw and preprocessed forms of the first scenario's dataset. That is the measure in CFS applied data minus measure in data without CFS. For example, Naïve Bayes had 82.8947%

accuracy in the raw form of the first scenario's dataset and 86.8421% in the CFS applied data of the same dataset.

**Table 7.5** Correlation based Feature Selection Improvements on First Preprocessed Dataset.

| Algorithms | Accuracy Difference (%) | Precision Difference | Recall Difference | F-measure Difference |
|---|---|---|---|---|
| Naïve Bayes | -1.3158 | 0.112 | -0.182 | -0.055 |
| Logistic Regression | 10.5263 | 0.271 | 0.137 | 0.189 |
| Multilayer Perceptron | 7.8948 | 0.216 | 0.091 | 0.139 |
| SMO | 2.6316 | 0.105 | 0 | 0.039 |
| IBk | 0 | 0.049 | -0.045 | -0.011 |
| J48 | 3.9474 | 0.149 | 0 | 0.057 |

So the difference is accuracy is the difference $86.8421\% - 82.8947\%$ which is 3.9474% as shown in the first cell of Table 7.4.

CFS on the raw form of the first scenario's dataset improved the performance of Naïve Bayes, Logistic Regression, SMO and K-NN. It decreased the Accuracy and Precision of MLP, however, F-measure remained the same. J48 showed no improvement in Accuracy and decreased in Recall but F-measure increased. Overall, CFS proved to be effective on the raw first scenario's dataset.

On the preprocessed form of the first scenario's dataset, Naïve Bayes saw rather a decline in both accuracy and f-measure. IBk showed no difference in accuracy but declined in F-measure. Figure 7.3 shows the graphical representation of the results comparing the before and aftermath of the application of CFS on the first scenario's dataset. Figure 7.3 (a) shows accuracy improvement of CFS on the raw first scenario's dataset; (b) shows the same

improvement on the preprocessed form; and (c) shows the difference in f-measure of the raw dataset as bars, and the difference on the preprocessed dataset as lines.

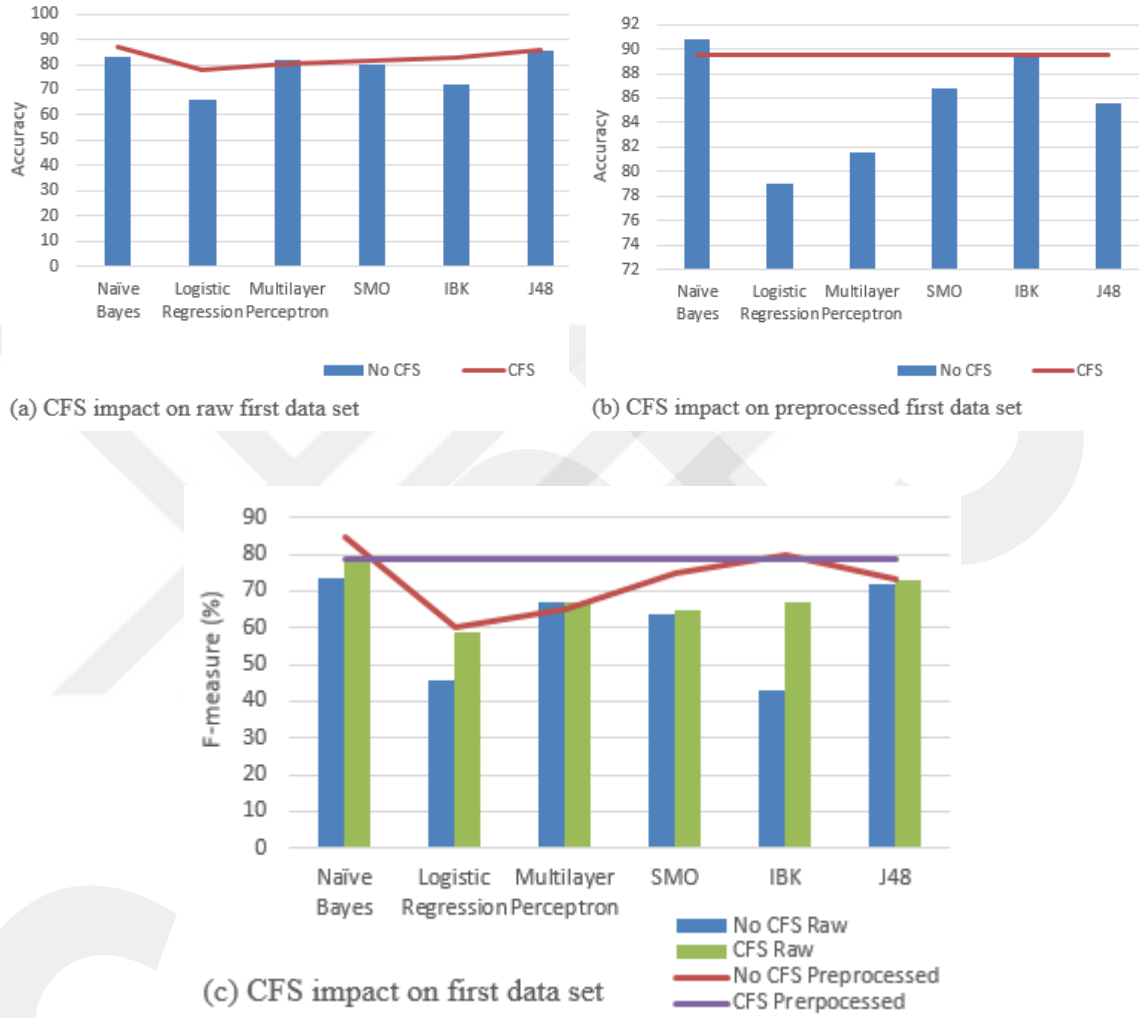



**Figure 7.3** Correlation base Feature Selection improvement on Accuracy and F-measure; (a) on raw first scenario's dataset and (b) on preprocessed first scenario's dataset.

CFS on the preprocessed data of the first scenario's dataset produced the same results for all algorithms. This can be attributed to the small dataset with only 2 independent or predicting features with each having two possible values. Method selection doesn't matter in this case since all algorithms had the same performance.

## 7.2 Second Scenario

The purpose of this scenario is to investigate the proportional relationship between the performance of machine learning algorithms on students' data and the number of courses in the dataset. That is to say whether the performance is directly or inversely proportional to the number of courses being predicted. This is in an effort to answer the second question of this thesis study which asks whether predictions focused on a course is better than predictions focused on more than one course.

To evaluate the correlation between the performance of learners and the number of courses in a dataset, the results of the three courses – SE221, CMPE225 and CMPE251 – were analyzed respectively from one individual course, two courses, to all the three courses. This gives a better presentation of the progression or decline in performance of the algorithms as the number of courses is increased. In addition, these courses were combined in every group to show the average trend of each learner as number of courses increases. The below subsections evaluate each course separately from when it's the only course to when it's combined with the other two courses.

### 7.2.1 SE221

This section follows the progress of the machine learning algorithms on datasets containing SE221 classes, starting from when it is the only course to predict, to all the three courses together. Since the progress measure in this section only deals with SE221 course contained datasets starting from the least number of courses, Table 7.6 second column shows Accuracy results of learners when applied to one-course prediction of SE221. And the third and fourth columns show the Accuracy when the learners were applied to two-course predictions of SE221 course. The last column shows Accuracy results on three-course predictions of SE221. If the table is observed closely, it can be seen that the performance decreases as the number of courses increases. However, the results can be better presented to easily reflect the change in performance as the number of courses increases.

**Table 7.6** Accuracy of SE221 included datasets

| Accuracy | SE221 | SE221-CMPE225 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 77.5281 | 70.7865 | 67.4157 | 64.0449 |
| Logistic | 69.6629 | 61.7978 | 55.0562 | 57.3034 |
| Multilayer Perceptron | 79.7753 | 69.6629 | 68.5393 | 65.1685 |
| SMO | 83.1461 | 76.4045 | 71.9101 | 69.6629 |
| IBk | 84.2697 | 80.8989 | 76.4045 | 70.7865 |
| J48 | 78.6517 | 68.5393 | 66.2921 | 62.9213 |

In order to show a better graphical presentation of the performance progression as the number of courses increase, the ratio of the first column to every other column used. That is divide every column by SE221 column. In other words, for every row, divide the whole row with SE221. For example, the first data row of Table 7.6 is Naïve Bayes which has its first column as 77.5281. So taking the ratio of this row becomes SE221/SE221 = 77.5281/77.5281 =1 for SE221. The second column becomes SE221-CMPE225/SE221=70.7865/77.5281 = 0.913. And this continues for the rest of the columns. This process is repeated for every row which resulted to Table 7.7. This ensures the first column (SE221) is always 1 and the other columns are ratios to the SE221. It better shows the change in accuracy as number of courses increases. Table 7.7 shows the results of taking the ratio of SE221 to other datasets.

**Table 7.7** Change in Accuracy as number of courses increase for SE221

| Algorithms | SE221 | SE221-CMPE225 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 1 | 0.913 | 0.8696 | 0.8261 |
| Logistic | 1 | 0.8871 | 0.7903 | 0.8226 |
| Multilayer Perceptron | 1 | 0.8732 | 0.8592 | 0.8169 |
| SMO | 1 | 0.9189 | 0.8649 | 0.8378 |
| IBk | 1 | 0.96 | 0.9067 | 0.84 |
| J48 | 1 | 0.8714 | 0.8429 | 0.8 |

Table 7.8 shows the F-measure results of the algorithms on the SE221-included datasets. The ratio approach was applied for F-measure too that resulted to Table 7.9

**Table 7.8** F-measure SE221 included datasets

| Algorithms | SE221 | SE221-CMPE225 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 0.785 | 0.717 | 0.681 | 0.654 |
| Logistic | 0.718 | 0.658 | 0.582 | 0.611 |
| Multilayer Perceptron | 0.785 | 0.693 | 0.681 | 0.652 |
| SMO | 0.8 | 0.733 | 0.676 | 0.659 |
| IBk | 0.818 | 0.781 | 0.705 | 0.686 |
| J48 | 0.777 | 0.682 | 0.644 | 0.604 |

**Table 7.9** Change in F-measure as number of courses increases for SE221

| F-measure | SE221 | SE221-CMPE225 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 1 | 0.913 | 0.868 | 0.833 |
| Logistic | 1 | 0.916 | 0.811 | 0.851 |
| Multilayer Perceptron | 1 | 0.883 | 0.868 | 0.831 |
| SMO | 1 | 0.916 | 0.845 | 0.824 |
| IBk | 1 | 0.955 | 0.862 | 0.839 |
| J48 | 1 | 0.878 | 0.829 | 0.777 |

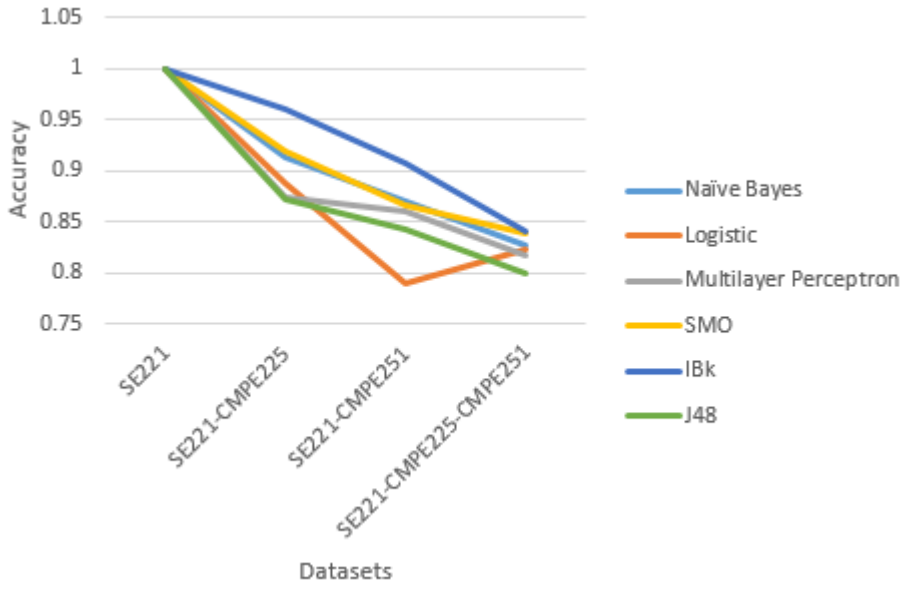



**Figure 7.4** Accuracy decline as number of courses increases for SE221

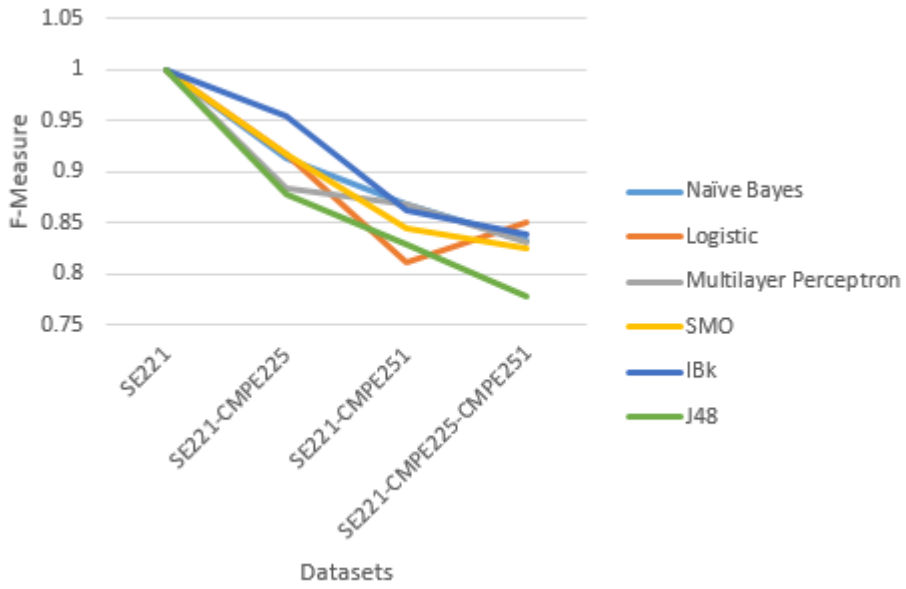



**Figure 7.5** F-measure decline as number of courses increases for SE221

The interest here is not to compare the algorithms but rather to show the change in performance as the number of courses increase. That is to say for every dataset, did the learners improve or decrease in performance. For that reason the change tables like Tables 7.7 and 7.9 are used to construct the line graphs instead of the actual performance tables – Tables 7.6 and 7.8. Figures 7.3 and 7.4 show the change in accuracy and f-measure, respectively, as the number of courses increases.

The results show that all the algorithms performed the best when predictions was made to only SE221 compared to when SE221 was combined with other courses. It also shows that only Logistic Regression did not decline in performance throughout but rather showed an improvement from two-course dataset (SE221-CMPE251) to three-course dataset. All the other algorithms showed a decline in performance as the number of courses increases with regards to SE221. Contrasting the results of Table 7.7 and 7.8 also showed SE221 had a better correlation to CMPE225 than it does to CMPE251. That is to say for two course predictions, SE221 with CMPE225 produce better results than SE221 with CMPE251.

### 7.2.2 CMPE225

In this section, progress of the learners as the number of courses increase was again measured, following the CMPE225 course. Unlike the effects on SE221, CMPE225 showed a consistent decline in both accuracy and f-measure of learners as the number of courses decrease. Even though SE221 performs better with CMPE225 as mentioned in the previous section, CMPE225 on the other hand performs better with CMPE251. For that

**Table 7.10** Accuracy of CMPE225 included datasets

| Algorithms | CMPE225 | CMPE225-CMPE251 | SE221-CMPE225 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 89.8876 | 73.0337 | 70.7865 | 64.0449 |
| Logistic | 85.3933 | 73.0337 | 61.7978 | 57.3034 |
| Multilayer Perceptron | 89.8876 | 86.5169 | 69.6629 | 65.1685 |
| SMO | 91.0112 | 79.7753 | 76.4045 | 69.6629 |
| IBk | 97.7528 | 85.3933 | 80.8989 | 70.7865 |
| J48 | 85.3933 | 78.6517 | 68.5393 | 62.9213 |

**Table 7.11** F-measure of CMPE225 included datasets

| Algorithms | CMPE225 | CMPE225-CMPE251 | SE221-CMPE225 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 0.897 | 0.752 | 0.717 | 0.654 |
| Logistic | 0.86 | 0.737 | 0.658 | 0.611 |
| Multilayer Perceptron | 0.897 | 0.852 | 0.693 | 0.652 |
| SMO | 0.907 | 0.777 | 0.733 | 0.659 |
| IBk | 0.977 | 0.833 | 0.781 | 0.686 |
| J48 | 0.833 | 0.773 | 0.682 | 0.604 |

**Table 7.12** Change in Accuracy as number of courses increases for CMPE225

| Algorithms | CMPE225 | CMPE225-CMPE251 | SE221-CMPE225 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 1 | 0.8125 | 0.7875 | 0.7125 |
| Logistic | 1 | 0.8553 | 0.7237 | 0.6711 |
| Multilayer Perceptron | 1 | 0.9625 | 0.775 | 0.725 |
| SMO | 1 | 0.8765 | 0.8395 | 0.7654 |
| IBk | 1 | 0.8736 | 0.8276 | 0.7241 |
| J48 | 1 | 0.9211 | 0.8026 | 0.7368 |

**Table 7.13** Change in F-measure as number of courses increases for CMPE225

| Algorithms | CMPE225 | CMPE225-CMPE251 | SE221-CMPE225 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 1 | 0.838 | 0.799 | 0.729 |
| Logistic | 1 | 0.857 | 0.765 | 0.71 |
| Multilayer Perceptron | 1 | 0.95 | 0.773 | 0.727 |
| SMO | 1 | 0.857 | 0.808 | 0.727 |
| IBk | 1 | 0.853 | 0.799 | 0.702 |
| J48 | 1 | 0.928 | 0.819 | 0.725 |

reason the order of columns was rearranged in Tables 7.10 through 7.13 to better show the slope of performance decline.

Tables 7.10 and 7.11 show the Accuracy and F-measure of CMPE225-included datasets, respectively. Tables 7.12 and 7.13 show change in performances (refer to section 7.2.1 on how the table is computed).

Figure 7.6 and 7.7 shows a graphical representation of the change in accuracy and f-measure respectively as the number of courses increase. CMPE225 shows a more uniform decline than the other courses.

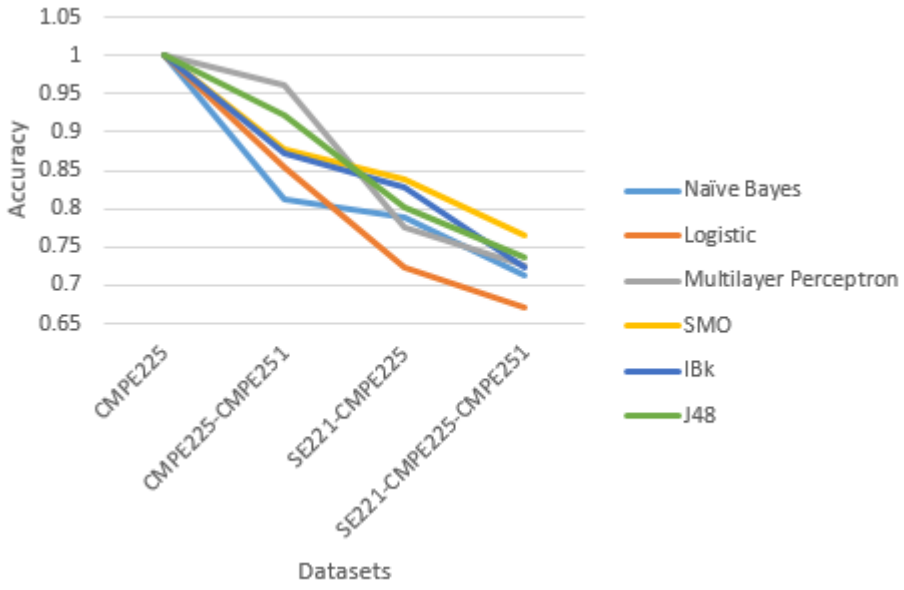



**Figure 7.6** Accuracy decline as number of courses increases for CMPE225

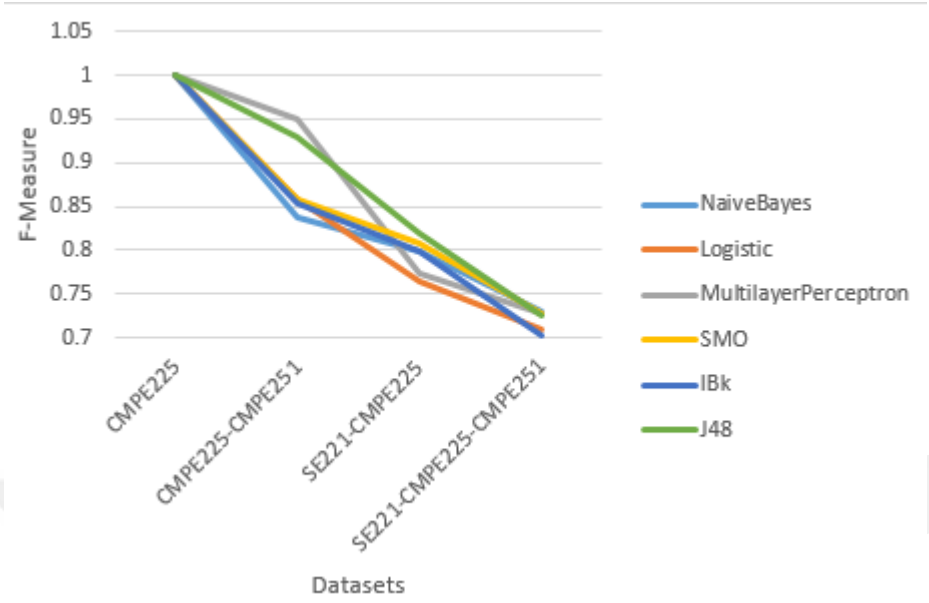



**Figure 7.7** F-measure decline as number of courses increases for CMPE225

### 7.2.3 CMPE251

CMPE251 was also tracked in terms of learners' performance progression but the results weren't as consistent as with the previous courses. As already explained in section 7.2.1, Logistic Regress shows better accuracy when in three-course predictions than two-course predictions on SE221-CMPE251. What is interesting here is that CMPE251 with CMPE225 – two-course predictions – showed better results than predictions on only CMPE251 for Multilayer Perceptron and IBk, and no difference for SMO. Up to this point one-course predictions showed better results than the rest for all algorithms. Nonetheless, more algorithms (three) show better performance in one-course predictions. Even with this discrepancy, the results show CMPE251 is more correlated with CMPE225 than it is with SE221. This suggests that students who pass CMPE251 are more likely to pass CMPE225 than they are to pass SE221. Table 7.14 and 7.15 show the accuracy and f-measure change respectively as number of courses increases.

F-measure in Table 7.15 shows CMPE251 being better than two course predictions except for Multilayer Perceptron algorithm. This difference is better shown in figures 7.8 and 7.9.

**Table 7.14** Accuracy of CMPE251 included courses

| Algorithms | CMPE251 | CMPE225-CMPE251 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 78.6517 | 73.0337 | 67.4157 | 64.0449 |
| Logistic | 78.6517 | 73.0337 | 55.0562 | 57.3034 |
| Multilayer Perceptron | 82.0225 | 86.5169 | 68.5393 | 65.1685 |
| SMO | 79.7753 | 79.7753 | 71.9101 | 69.6629 |
| IBk | 84.2697 | 85.3933 | 76.4045 | 70.7865 |
| J48 | 85.3933 | 78.6517 | 66.2921 | 62.9213 |

**Table 7.15** F-measure of CMPE251 included courses

| Algorithms | CMPE251 | CMPE225-CMPE251 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 0.795 | 0.752 | 0.681 | 0.654 |
| Logistic | 0.792 | 0.737 | 0.582 | 0.611 |
| Multilayer Perceptron | 0.823 | 0.852 | 0.681 | 0.652 |
| SMO | 0.783 | 0.777 | 0.676 | 0.659 |
| IBk | 0.836 | 0.833 | 0.705 | 0.686 |
| J48 | 0.845 | 0.773 | 0.644 | 0.604 |

**Table 7.16** Change in Accuracy as number of courses increases for CMPE251

| Algorithms | CMPE251 | CMPE225-CMPE251 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 1 | 0.9286 | 0.8571 | 0.8143 |
| Logistic | 1 | 0.9286 | 0.7 | 0.7286 |
| Multilayer Perceptron | 1 | 1.0548 | 0.8356 | 0.7945 |
| SMO | 1 | 1 | 0.9014 | 0.8732 |
| IBk | 1 | 1.0133 | 0.9067 | 0.84 |
| J48 | 1 | 0.9211 | 0.7763 | 0.7368 |

**Table 7.17** Change in F-measure as number of courses increases for CMPE251

| Algorithms | CMPE251 | CMPE225-CMPE251 | SE221-CMPE251 | SE221-CMPE225-CMPE251 |
|---|---|---|---|---|
| Naïve Bayes | 1 | 0.946 | 0.857 | 0.823 |
| Logistic | 1 | 0.931 | 0.735 | 0.771 |
| Multilayer Perceptron | 1 | 1.035 | 0.827 | 0.792 |
| SMO | 1 | 0.992 | 0.863 | 0.842 |
| IBk | 1 | 0.996 | 0.843 | 0.821 |
| J48 | 1 | 0.915 | 0.762 | 0.715 |

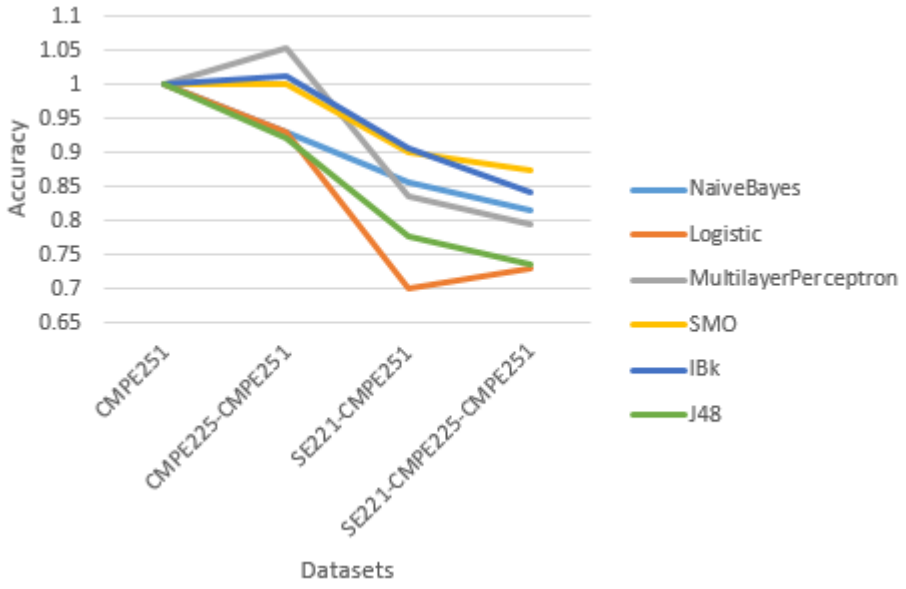



**Figure 7.8** Accuracy decline as number of courses increases for CMPE251

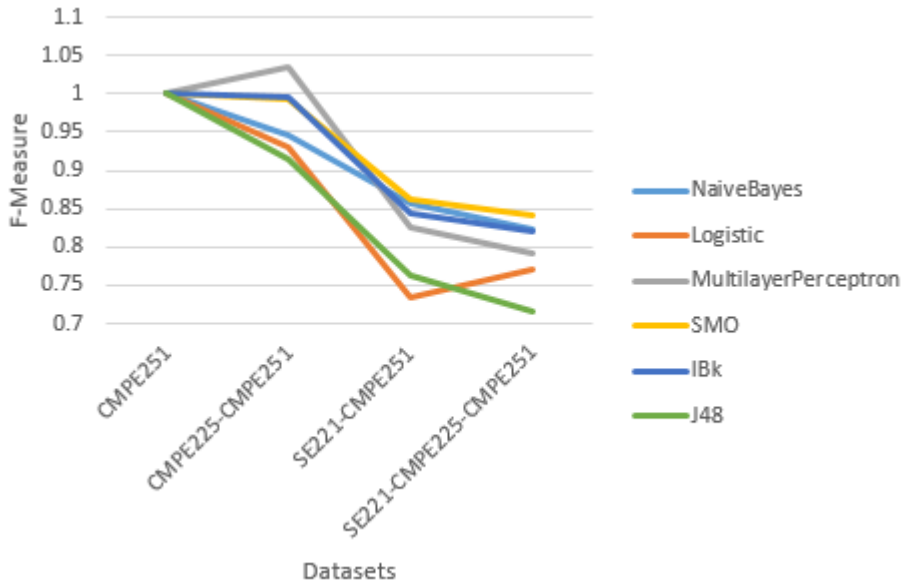



**Figure 7.9** F-measure decline as number of courses increases for CMPE251

### 7.2.4 Average Performance Differences as Number of Courses Increases

The results in each category was averaged to have a general overview of the learners' performance on one-course predictions, two-course predictions and three-course predictions. For example, the second column of Table 7.18 shows the average accuracy of

**Table 7.18** Averaged Accuracy per number of courses

| Algorithms | One-Course Average | Two-Courses Average | Three-Courses |
|---|---|---|---|
| Naïve Bayes | 82.0225 | 70.412 | 64.0449 |
| Logistic | 77.9026 | 63.2959 | 57.3034 |
| Multilayer Perceptron | 83.8951 | 74.9064 | 65.1685 |
| SMO | 84.6442 | 76.03 | 69.6629 |
| IBk | 88.7641 | 80.8989 | 70.7865 |
| J48 | 83.1461 | 71.161 | 62.9213 |

**Table 7.19** Averaged F-measure per number of courses

| Algorithms | One-Course Average | Two-Courses Average | Three-Courses |
|---|---|---|---|
| Naïve Bayes | 0.826 | 0.717 | 0.654 |
| Logistic | 0.79 | 0.659 | 0.611 |
| Multilayer Perceptron | 0.835 | 0.742 | 0.652 |
| SMO | 0.83 | 0.729 | 0.659 |
| IBk | 0.877 | 0.773 | 0.686 |
| J48 | 0.818 | 0.7 | 0.604 |

**Table 7.20** Change in Average Accuracy as number of courses increases for all three courses

| Algorithms | One-Course | Two-Courses | Three-Courses |
|---|---|---|---|
| Naïve Bayes | 1 | 0.868 | 0.792 |
| Logistic | 1 | 0.834 | 0.773 |
| Multilayer Perceptron | 1 | 0.889 | 0.781 |
| SMO | 1 | 0.878 | 0.794 |
| IBk | 1 | 0.881 | 0.782 |
| J48 | 1 | 0.856 | 0.738 |

**Table 7.21** Change in Average F-measure difference as number of courses increases for all three courses

| Algorithms | One Course | Two Courses | Three Courses |
|---|---|---|---|
| Naïve Bayes | 1 | 0.868 | 0.792 |
| Logistic | 1 | 0.834 | 0.773 |
| Multilayer Perceptron | 1 | 0.889 | 0.781 |
| SMO | 1 | 0.878 | 0.794 |
| IBk | 1 | 0.881 | 0.782 |
| J48 | 1 | 0.856 | 0.738 |

learners on SE221, CMPE225 and CMPE251 datasets. Whereas the third column shows the average accuracy on SE221-CMPE225, SE221-CMPE251 and CMPE225-CMPE251.

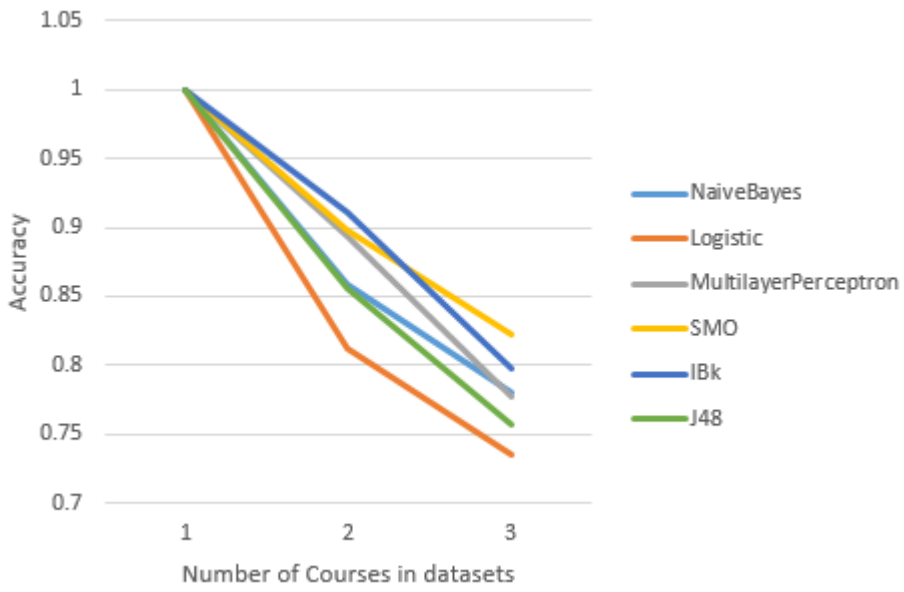



**Figure 7.10** Average Accuracy decline as number of courses increases for all three courses

Table 7.20 and 7.21 shows the normalized results of the accuracy and f-measure of Tables 7.18 and 7.19. These two tables clearly show that the number of courses being predicted is inversely proportional to the prediction performance. This is further elaborated in figures 7.10 and 7.11 which show the change of performance in line graphs. The y-axis

show the performance and the x-axis show the number of courses being predicted at the same time.

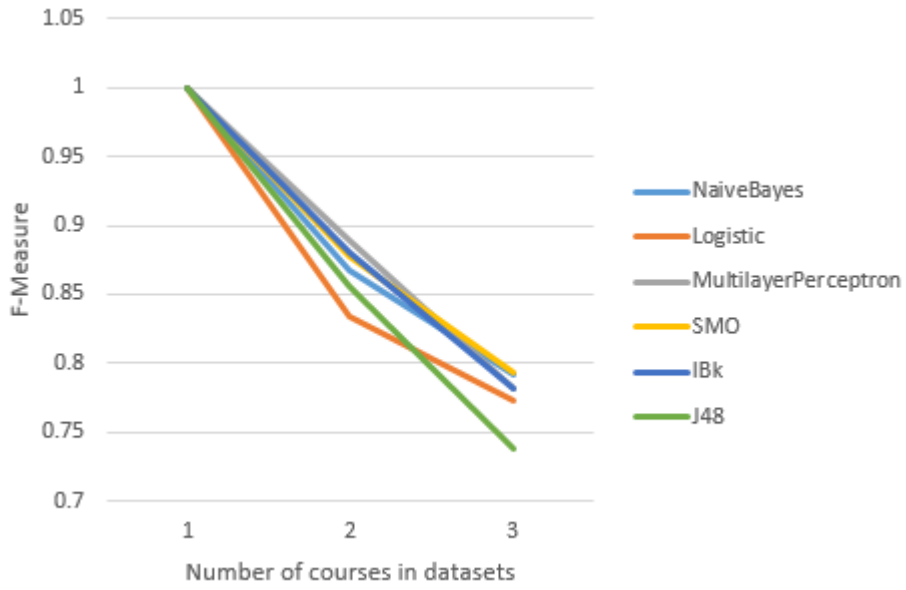



**Figure 7.11** Average F-measure difference as number of courses increases for all three courses

### 7.2.5 More Findings

By observing the performance of the machine learning algorithms on each individual course in the second scenario (Table 7.22), it can be shown that CMPE225 gave better performance followed by CMPE251 then SE221. In other words, the same exact predictors for the same students are better at predicting CMPE225 than they are for CMPE251 and SE221.

**Table 7.22** Accuracy of individual courses in second scenario

| Accuracy | SE221 | CMPE251 | CMPE225 |
|---|---|---|---|
| Naïve Bayes | 77.5281 | 78.6517 | 89.8876 |
| Logistic | 69.6629 | 78.6517 | 85.3933 |
| Multilayer Perceptron | 79.7753 | 82.0225 | 89.8876 |
| SMO | 83.1461 | 79.7753 | 91.0112 |
| IBk | 84.2697 | 84.2697 | 97.7528 |
| J48 | 78.6517 | 85.3933 | 85.3933 |

This difference in performance is as a results of the correlation between the predictors and the classes being predicted. For example, CMPE225 has the prerequisite courses CMPE113 and CMPE114 whose data is included in the datasets that is used as predictors. The same condition is also true for CMPE251. Although it does not have any prerequisite course, the content is mostly based on mathematics knowledge and the dataset contains two different previous math course grades used as predictors.

This correlation benefit reflected in the results reveals that prerequisite grades are very crucial in predicting courses. It also proves that prerequisite models of Software Engineering department is correctly designed.

# CHAPTER 8

# DISCUSSION AND CONCLUSION

The main aim of this thesis was to compare six prediction algorithms in terms of their performance on predicting student performance. This thesis also investigated the effects of data preprocessing and Correlation based Feature Selection (CFS) on the performance of machine learning algorithms. In a different scenario, the proportional relationship between number of courses being predicted at the same time and the performance of these algorithms was investigated. Four performance measure were used: Accuracy, Precision, Recall and F-measure. Naïve Bayes, Logistic Regress, MLP, SMO, IBk and J48 were compared on two different students' datasets. The first scenario's dataset was made up of predictor features (Mathematics and Software Engineering courses in first, second and third semesters together with few engineered features) and a target class feature as Data Structures grades (fourth semester course). The second scenario considers the same predictor features, but only for the first and second semesters. Its target classes are all possible combinations of three software engineering third semester courses: Software Requirements Engineering (SE221), Object-Oriented Programming (CMPE225) and Discrete Computation Structures (CMPE251). This resulted to the second experimental scenario having 7 datasets: SE221, CMPE225, CMPE251, SE221-CMPE225, SE221-CMPE251, CMPE225-CMPE251 and SE221-CMPE225-CMPE251. Models were built to determine whether a student will pass or fail using the Weka analytical tool. For each dataset, Mathematics and Software Engineering courses' grades of all the preceding semesters were used as predictors (independent variables) together with some student academic records like scholarship status, gender, and few engineered features.

## 8.1 Answers to Research Questions

This study was set out to answer some research questions. The following bulletins list these questions together with their answers and brief discussions.

1. **Are predictive models good at predicting students' performance?**

   Yes!

   The results generally show a great improvement from the baseline performance the Precision, Recall and F-measures of the machine learning algorithms were impressive.

2. **Which is better, predictions of a single course at a time or predicting different courses together?**

   Prediction of a single course at a time proved to be better.

   Experimenting on seven datasets in the second scenario revealed that the number of courses being predicted by machine learning algorithms is inversely proportional to the performance of those algorithms. That is to say as the number of courses increases, the performance of the algorithms decreases. All algorithms strictly adhered to this rule throughout when the course to begin with is CMPE225. Logistic Regression deviated when SE221 was the course considered and Multilayer Perceptron and IBk too didn't follow the rule when CMPE251 was the course in question. However, generally, this rule is mostly true and was proven when the results were averaged.

3. **Among the six algorithms compared in this study, which is the best prediction algorithm in the student performance prediction problem?**

   Naïve Bayes had the best f-measure on three of the four forms of the first scenario's dataset, for the last form – CFS applied on preprocessed form – all algorithms showed the same results. This means that Naïve Bayes is better at identifying failing students, which is the main reason for prediction. In terms of accuracy, it only fell behind J48 on the raw form of the data but had a better accuracy in the preprocessed form as well as the CFS applied forms. Even

though the second scenario's purpose wasn't to compare the learning algorithms, IBk and mostly had the best performance while Logistic Regress performed the least. Overall, Naïve Bayes had the best performance in the first scenario and IBk is the best in the second scenario.

4. **What are the most useful features in predicting students performance in the given context?**

   Correlation-based Feature Selection (CFS) when applied on the dataset of the first scenario revealed the following feathers as the most useful in the predictions:

   1. Previous semester GPA;
   2. Number of times Discrete Computation Structures was repeated;
   3. Grades acquired in: Fundamentals of Computing, Discrete Computational Structures, Computer Programming I;
   4. Scholarship of the student;
   5. Number of times Computer Programming II was repeated.

   Weka's CorrelationAttributeEval ranked them in terms of usefulness in the order of the list above.

5. **Is data preprocessing effective in student performance predictions?**

   Yes!

   Data preprocessing shows a great improvement on the machine learning algorithms, especially on IBk which shows accuracy difference of 17% (difference means results in preprocessed minus results in raw data) and an f-measure difference of 0.368. However, MLP and J48 showed no difference in Accuracy and MLP showed a decline in f-measure. IBk again on the second dataset showed better improvement when data preprocessing was applied. SMO showed a decline on accuracy but f-measure was improved.

6. **Is Correlation based Feature Selection effective in student performance predictions?**

   Yes!

   CFS on the preprocessed form of the first scenario's dataset gave the same results for all algorithms (89.4737% Accuracy, 0.938 Precision, 0.682 Recall and 0.789 F-measure). This can be attributed to the small dataset as well as the few features (2 features) available with only two possible values for each.

   Logistic Regression showed the greatest improvement in accuracy when CFS was applied on both the raw and preprocessed forms of the first scenario's dataset. J48 showed no difference in accuracy but showed little improvement in F-measure on the first scenario's dataset. MLP on the other hand showed a decline in accuracy and no difference in f-measure.

## 8.2 Conclusion

Predictive models are very good at predicting student's performance. A novel discovery is this study is that the number of courses being predicted is inversely proportional to the performance of machine learning algorithms. That is to say it is better to predict one course at a time than predicting more than one course together.

Overall, Naïve Bayes proved to be the best at correctly predicting failing students in the given experimental setup (when only administrative academic data are used). It was also confirmed that data preprocessing improves performance of machine learning algorithms. Using CFS also showed improvements in the results. It helped to identify the most useful features in the prediction as: Previous semester GPA; Number of times Discrete Computation Structures was repeated; Grades acquired in: Fundamentals of Computing, Discrete Computational Structures, Computer Programming I; Scholarship of the student; and Number of times Computer Programming II was repeated.

It was also observed that having data about prerequisite knowledge required for a course is very crucial in predicting student performance in a particular course.

In conclusion, successful predictions doesn't only depend on the machine learning algorithms but also on the domain in which predictions are made, the constituents of the data, the size of the data, available features as well as the preprocessing applied. All these need to be fine-tuned in order to achieve better results.

## 8.3 Future Work

This study had few limitations which might impact the performance of the learners. The size of data is very useful in machine learning and this experiment fell short of data. Future studies can do better if more data is acquired that is say in the thousands.

There were also some academic records features that could have had a positive impact on the predictions but couldn't be acquired. Some of these features are pre-university features like student's secondary school results or mathematics grades, student's national rank, and even in-term performance and final percentage score in predictor courses.

The novel discovery of the number of courses being inversely proportional to the performance of the learners can also be repeated with larger datasets and more courses. A mathematical model can also be created to show the inverse relationship.

On a similar but different experiment, survey data can be used to acquire demographic data and psychometric information that can also be put together with the features used herein.

# REFERENCES


[1] Alpaydin, E. "Introduction to Machine Learning, 2nd edn. Adaptive Computation and Machine Learning." (2010).

[2] Witten, Ian H., and Eibe Frank. "Data mining: Practical machine learning tools and techniques, (morgan kaufmann series in data management systems)." Morgan Kaufmann, June 104 (2005): 113.

[3] Romero, Cristóbal, and Sebastián Ventura. "Educational data mining: a review of the state of the art." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40, no. 6 (2010): 601-618.

[4] Pojon, Murat. "Using Machine Learning to Predict Student Performance." (2017).

[5] Ma, Cheng, Baofeng Yao, Fang Ge, Yurong Pan, and Youqiang Guo. "Improving Prediction of Student Performance based on Multiple Feature Selection Approaches." In Proceedings of the 2017 International Conference on E-Education, E-Business and E-Technology, pp. 36-41. ACM, 2017.

[6] Kalles, Dimitris, and Christos Pierrakeas. "Analyzing student performance in distance learning with genetic algorithms and decision trees." Applied Artificial Intelligence 20, no. 8 (2006): 655-674.

[7] Koutina, Maria, and Katia Lida Kermanidis. "Predicting postgraduate students' performance using machine learning techniques." In Artificial Intelligence Applications and Innovations, pp. 159-168. Springer, Berlin, Heidelberg, 2011.

[8] Chaudhury, Pamela, Sushruta Mishra, Hrudaya Kumar Tripathy, and Brojo Kishore. "Enhancing the capabilities of Student Result Prediction System." In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, p. 91. ACM, 2016.

[9] Arsad, Pauziah Mohd, and Norlida Buniyamin. "A neural network students' performance prediction model (NNSPPM)." In Smart Instrumentation, Measurement and Applications (ICSIMA), 2013 IEEE International Conference on, pp. 1-5. IEEE, 2013.

[10] Shahiri, Amirah Mohamed, and Wahidah Husain. "A review on predicting student's performance using data mining techniques." Procedia Computer Science 72 (2015): 414-422.

[11] Osmanbegović, Edin, and Mirza Suljić. "Data mining approach for predicting student performance." Economic Review 10, no. 1 (2012): 3-12.

[12] Ramesh, V. A. M. A. N. A. N., P. Parkavi, and K. Ramar. "Predicting student performance: a statistical and data mining approach." International journal of computer applications 63, no. 8 (2013).

[13] Goga, Maria, Shade Kuyoro, and Nicolae Goga. "A recommender for improving the student academic performance." Procedia-Social and Behavioral Sciences 180 (2015): 1481-1488.

[14] Xing, Wanli, Rui Guo, Eva Petakovic, and Sean Goggins. "Participation-based student final performance prediction model through interpretable Genetic Programming: Integrating learning analytics, educational data mining and theory." Computers in Human Behavior 47 (2015): 168-181.

[15] Huang, Shaobo, and Ning Fang. "Work in progress: Early prediction of students' academic performance in an introductory engineering course through different mathematical modeling techniques." In Frontiers in Education Conference (FIE), 2012, pp. 1-2. IEEE, 2012.

[16] Choudhary, Neha, and Ashutosh Guide Mishra. "Student Performance Measure by using Different Classification Methods of Data Mining." PhD diss., 2016.

[17] Mishra, Tripti, Dharminder Kumar, and Sangeeta Gupta. "Mining students' data for performance prediction." In Fourth International Conference on Advanced Computing & Communication Technologies, pp. 255-262. 2014.

[18] Arsad, Pauziah Mohd, and Norlida Buniyamin. "Prediction of engineering students' academic performance using Artificial Neural Network and Linear Regression: A comparison." In Engineering Education (ICEED), 2013 IEEE 5th Conference on, pp. 43-48. IEEE, 2013.

[19] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.

[20] Natek, Srečko, and Moti Zwilling. "Student data mining solution–knowledge management system related to higher education institutions." Expert systems with applications 41, no. 14 (2014): 6400-6407.

[21] de Baker, Ryan Shaun Joazeiro, Tiffany Barnes, and Joseph E. Beck. "Educational Data Mining 2008."

[22] Gunnarsson, Bjorn Levi, and Richard Alterman. "Predicting failure: a case study in co-blogging." In Proceedings of the 2nd international conference on learning analytics and knowledge, pp. 263-266. ACM, 2012.

[23] Baker, Ryan SJD, and Kalina Yacef. "The state of educational data mining in 2009: A review and future visions." JEDM| Journal of Educational Data Mining 1, no. 1 (2009): 3-17.

[24] Pokay, Patricia, and Phyllis C. Blumenfeld. "Predicting achievement early and late in the semester: The role of motivation and use of learning strategies." Journal of educational psychology 82, no. 1 (1990): 41.

[25] Imbrie, P. K., J. J. Lin, Ken Reid, and Alexander Malyscheff. "Using hybrid data to model student success in engineering with artificial neural networks." In Proceedings of the Research in Engineering Education Symposium, pp. 7-10. 2008.

[26] Jashapara, Ashok. Knowledge management: An integrated approach. Pearson Education, 2004.

[27] Nyce, Charles, and API CPCU. "Predictive analytics white paper." American Institute for CPCU. Insurance Institute of America (2007): 9-10.

[28] Kotsiantis, Sotiris, Christos Pierrakeas, and Panagiotis Pintelas. "PREDICTING STUDENTS'PERFORMANCE IN DISTANCE LEARNING USING MACHINE LEARNING TECHNIQUES." Applied Artificial Intelligence 18, no. 5 (2004): 411-426.

[29] Touron, Javier. "The determination of factors related to academic achievement in the university: implications for the selection and counselling of students." Higher Education 12, no. 4 (1983): 399-410.

[30] Lassibille, Gérard, and Lucía Navarro Gómez. "Why do higher education students drop out? Evidence from Spain." Education Economics 16, no. 1 (2008): 89-105.

[31] Herzog, Serge. "Measuring determinants of student return vs. dropout/stopout vs. transfer: A first-to-second year analysis of new freshmen." Research in higher education 46, no. 8 (2005): 883-928.

[32] Forman, George, and Ira Cohen. "Learning from little: Comparison of classifiers given little training." In European Conference on Principles of Data Mining and Knowledge Discovery, pp. 161-172. Springer, Berlin, Heidelberg, 2004.

[33] Wasikowski, Mike, and Xue-wen Chen. "Combating the small sample class imbalance problem using feature selection." IEEE Transactions on knowledge and data engineering 22, no. 10 (2010): 1388-1400.

[34] Van Hulse, Jason, Taghi M. Khoshgoftaar, and Amri Napolitano. "Experimental perspectives on learning from imbalanced data." In Proceedings of the 24th international conference on Machine learning, pp. 935-942. ACM, 2007.

[35] Barandela, Ricardo, Rosa M. Valdovinos, J. Salvador Sánchez, and Francesc J. Ferri. "The imbalanced training sample problem: Under or over sampling?." In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pp. 806-814. Springer, Berlin, Heidelberg, 2004.

[36] Guyon, Isabelle, and André Elisseeff. "An introduction to variable and feature selection." Journal of machine learning research 3, no. Mar (2003): 1157-1182.

[37] Hall, Mark Andrew. "Correlation-based feature selection for machine learning." (1999).

[38] Peng, Chao-Ying Joanne, Kuk Lida Lee, and Gary M. Ingersoll. "An introduction to logistic regression analysis and reporting." The journal of educational research 96, no. 1 (2002): 3-14.

[39] Kohavi, Ron, and George H. John. "Wrappers for feature subset selection." *Artificial intelligence* 97, no. 1-2 (1997): 273-324.

[40] Kohavi, Ron, and Dan Sommerfield. "Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology." In *KDD*, pp. 192-197. 1995.