PREDICTING ACADEMIC PERFORMANCE OF POTENTIAL ELECTRICAL ENGINEERING MAJORS

A Thesis

by

SANJHANA SUNDARARAJ

August 2017

Major Subject: Electrical Engineering

ABSTRACT

Data Analytics for education is fast growing into an important part of higher learning institutions, which helps to improve student success rate and decision-making with regards to teaching methods, course selection, and student retention.

The undergraduate program at Texas A&M University requires students to take up a general engineering program during their freshman and sophomore years. During the course of this period, student academic performance, abilities and participation is assessed. As per the Entry-to-a-Major policy, departments place the students in the best possible major based on their displayed capacities and in alignment with their goals. Our focus is on the Electrical Engineering department and the success rate of students with aspirations and background in this major. An approach to improve student retention rate is to predict beforehand the performance of students in specific course disciplines based on the information that is mined from their previous records. Based on the outcome, decisions can be made in advance regarding their further enrollment in the area and need for specific attention in certain aspects to get students up to the benchmark.

In this thesis, we put together a set attributes related to students in the general program and with an electrical engineering aligned background. The analysis centers around building a method that explains the joint influence of attributes on our target variable and comparison of prediction performances between our models. The prime tools used are Supervised classification and Ensemble learning methods. We also develop a metric-based learning framework suitable for our application that enables competitive accuracy results and efficient pattern recognition from the underlying data.

I dedicate this thesis to my grandparents and parents for their unconditional love and support through this entire journey.

ACKNOWLEDGMENTS

I thank all my committee members for taking the time and effort to conduct my defense. My biggest gratitude goes to my advisor Dr. Tie Liu and co-advisor Dr. Xiaoning Qian for their constant support, motivation and expert guidance through every phase of this study. The way they have directed me from start to finish with their valuable inputs has been a great experience to learn from and I sincerely appreciate it. I also thank Qingyu Wu, undergraduate student in the Department of Electrical and Computer Engineering who has diligently learned and worked together during the course of this study. I thank Lucia Sui, phD candidate and Nagaraj Janakiraman, phD candidate in the ECEN department for their valuable insights. I would like to thank all my Professors through the course of my Master of Science program. I have had the honour to take part in such productive courses, projects and discussions with them and they have given me irreplaceable knowledge and skills that will stand by me greatly as I move forward. Sincere thanks to the ECEN department and advisors who have always been there to clarify my queries and help me in every way. I sincerely thank the university and department for taking me into the M.S. graduate program in the Department of Electrical and Computer Engineering. A special thanks to Dr. Tie Liu for giving me the opportunity to work with him and for his time and consideration throughout this period. This graduate school exposure has been an unforgettable experience and a solid platform to begin my career with and I am really grateful to every one who has contributed to this phase. Ultimately, I thank my family and friends back home and my greatest faith of all, The Almighty for never ending blessings and support.

CONTRIBUTORS AND FUNDING SOURCES

I am extremely grateful and would like to show my appreciation to the contributions of my advisor and Chair, Dr.Tie Liu, Co-chair Dr.Xiaoning Qian and my committee members Dr.Ulisses Braga Neto and Dr.Xia Hu for their valuable inputs and guidance in a progressive direction. Special thanks to the Electrical Engineering department for providing accessibility to the student records to compile the data for this research and DeeAnn for her massive help in extracting the required data to be used in the analysis. The research though carried out without external financial support, received tremendous support from my advisor Dr.Tie Liu and his team of students as well as the ECEN department of Texas A&M University for which I would like to express my sincere gratitude.

NOMENCLATURE

TAMU	Texas A&M University
ETAM	Entry-to-a-Major
EE	Electrical Engineering
ECEN	Electrical and Computer Engineering
NB	Naive Bayes
DT	Decision Tree
CART	Classification and Regression Tree
RF	Random Forest
OOB	Out-of-Bag
SVM	Support Vector Machine
CV	Cross Validation
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
PRC	Precision Recall Curve
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Overview

1.1.1 ETAM Policy

The Entry-to-a-Major policy is the system followed by the College of Engineering for the admission of undergraduate students to a major engineering program. This decision is taken during the transition of students from sophomore to junior year. Students are part of a general engineering program in their freshman and sophomore years and take up basic engineering courses and pre-requisites that align with their major of interest. Every student is eligible to take part in two cycles of ETAM applications at the end of his freshman year and during the course of his sophomore year to give himself a good chance of getting into the major of his preference. Departments place students in the highest possible major based on the evaluation of their academic performance and abilities shown thus far. Students who do not get into a major at the end of their second cycle are given specific attention by their advisors and faculty.

1.1.2 Motivation

Students in the general engineering program are provided with the chance to take part in two ETAM cycles to get into their favoured major. There is a window of opportunity to improve students' chances of making it into their first-choice major in the period between first and second cycles of ETAM applications. Tentative knowledge about a student's potential performance in important pre-requisite courses related to their major choice beforehand can enable advisors, instructors as well as students themselves to give more time and focus on the aspects in which they are not yet at a level expected of them especially with important deciding courses to be taken in the second year of the program in between the cycles. This is where predictive analytics plays a key role in improving student success rates into their choice of major.

1.2 Problem Statement

The main focus of our analysis is the prediction of student performance in the ECEN pre-requisite engineering courses. This would provide a metric for both the admissions body as well as the students to decide on pursuing an Electrical Engineering major. An important pre-requisite course in sophomore year for higher-level courses in the EE major is ECEN 214, and the student performance in this course based on the acquired student historical records and related attributes is the target of our prediction modeling.

1.3 Approach

A natural starting point for this problem was to first model the data without considering any attribute interactions. Based on the results, we were able to build suitable models which capture the hypothesis of feature interaction to influence the target variable. The important features, which together have a significant impact on our target were identified. The next step was to develop a metric-based framework. Learning an appropriate similarity metric from our student data and utilizing this for our prediction task gave competitive results and enabled efficient interpretation of the patterns that was present in the underlying data.

2. LITERATURE REVIEW

2.1 Impact of Analytics on Education

Generally, the role of data analytics in improving education standards and student success is a significant one as highlighted by some interesting statistics from past analysis. Research by Kimberly E. Arnold and Matthew D. Pistilli have revealed that the implementation of a student success system called Course Signals which allows faculty to work with students based on predictive results boosted up grades and student retention rates. Within various courses, was a strong increase in the higher grades (As and Bs) ranging from 2.23 to 13.84 percentage, decrease in Cs ranging from 1.84 to 9.38 percentage and a decrease in Ds and Fs ranging from 0.59 to 9.40 percentage. Thus follows an increase in student retention rate into the next year of the program. Bryan Beaudoin [1] developed a model that identifies 'at-risk' students. The model had a 74% recall meaning 3 out of 4 students who do not achieve retention were marked as 'at-risk' students by his method. Such information known beforehand will enable faculty to give appropriate attention to such students. These figures thus show the profound impact of analytics and predictive modeling on education and provide encouragement to work on a systematic approach that would better assist our students to potentially being admitted to the EE major program.

2.2 Student attributes

There is a wide pool of academic, social and demographic attributes related to a student that influence his or her academic performance. This is an age old research and the earliest studies by Spady (1970) seemed to pick on academic potential, grade performance and social factors as those with the most impact on a student. Research by Tinto showed that a student's pre-entry attributes (high school performance, test scores, family background) , goals, social integration and academic factors (course performances, grade point average, teaching facilities) are the factors which influence his success rate. In the 2000's Tinto proposed that academic support and accessibility to teaching services improve student performance. Studies by Wyckoff and Habley [2] showed positive student-faculty interaction greatly influence student retention. A summary of these analysis make it apparent that the various factors mentioned are largely interconnected and come down to academic potential and performance which enables faculty to identify areas where students need more interaction and thus enable them to improve to the required standards.

2.3 Machine Learning in Education

This is a more recent application (2000's) of machine learning. Among the early research is the one by Corbett and Anderson (2000) who implemented a knowledge tracing model using a Bayesian framework. This model estimated the probability a student is capable of a certain skill based on his or her previous attempts with an accuracy of up to 48 percent. There are plenty of commercial applications of user experience modeling but its application in education is a new trend. Kardan and Conati (2010) proposed a modeling which uses learner interaction with the system and classify new learners with similar goals and abilities. Thus we can observe that classification and clustering techniques can be used to profile or categorize students with respect to their relevant attributes and this falls under the machine learning framework. [3]

2.4 Predictive Analytics

Predictive analytics is an association of data mining, pattern analysis, statistics and machine learning to discover associations in the data and determine in advance various values which can be inferred from the data. We study in literature about how predictive analytics has evolved and what are the more recent developments in this domain. Different forms of predictive analytics has its applications from the early 90's. Predictive analytics was used to decode German messages during the world wars, Kerrison Predictor automates targeting missiles against enemy planes, weather forecasting models, predictive modeling in FICO to predict credit risk scores and fraud detection to name a few. Later, predictive analytics evolved into modeling using machine learning algorithms and specifically into classification and regression problems. The more recent developments have been in the fields of natural language processing, anticipatory analytics in the medical field and information retrieval applications.

2.5 Feature Selection

Feature selection is an important step in modeling as it brings down dimensionality and complexity issues and improves associative learning for prediction of the target variable. This method eliminates irrelevant or redundant features which do not impact the output variable and hinder predictive accuracy. Existing literature propose a number of methods for selection of the optimal feature set for the modeling process.The early feature selection methods involved an exhaustive search of the feature set, generating different combinations of subsets of features and evaluating model performance with subsets of features to determine the best choice of features for the model. The early methods involved random search methods. This method starts of with a random subset of features and the generation of successive subsets were all completely random. This is known as the Las Vegas algorithm with a quadratic search complexity. The feature selection methods further developed into more specific procedures that can be classified into filter and wrapper feature selection methods. Filter methods are based on various statistical relationships that can be inferred from the data and are independent of the predictive algorithm to be learned from the data. These measures involve statistics like correlation of the independent variables with the dependent variable, variance filter approaches which for instance eliminate features with very low variance on the basis that they do not carry much information and missing values ratio of the features. On the other hand wrapper methods are based on the performance of the learning algorithm with the selected feature set. This is a more computationally prohibitive method and can be classified into a forward and backward method. The forward feature construction method involves evaluating performance of the model starting from individual features alone and recursively adding to the feature set. The backward elimination procedure works with the entire feature set initially and recursively eliminates those features which do not contribute to model accuracy. The choice of feature selection method is totally dependent on the data to be modeled and the associations present in the data.

2.6 Ensemble Learning

Supervised Learning is a subset of machine learning which involves predictive modeling of the data with knowledge of the true class or labels against which model performance or accuracy can be evaluated. Important issues with this kind of learning methods is the bias-variance tradeoff, complexity and dimensionality. Supervised learning algorithms have been extended to a family of classifiers termed an ensemble to primairly overcome bias-variance issues. The need for ensemble learning methods is deeply related to the bias-variance tradeoff in classifier methods. Early research regarding this started with the Error Correcting Ouput Coding Method by Dietterich and Bakiri in 1995. There have been varying but no constant definition of bias-variance relationship over the years of research. Kohavi and Wolpert in 1996 measured bias and variance of a model in terms of squared error. Dietterich and Kong defined bias and variance such that the probability of misclassification is a sum of the bias and variance measures of the model. Heskes defined bias-variance decomposition for a model with Kullback-Leibler loss functions. The concept of bias-variance tradeoff has evolved over the years and the need for ensembles was further worked on and formulated into specific methods such as Bagging by Breiman in 1996 and Boosting by Freund and Schapire in 1996. These methods specified various parameters and loss functions for construction of an ensemble of classifiers which gives a reasonable bias-variance tradeoff, ranking of feature importance to bring down dimensionality and redundancy issues and improve the overall predictive performance.

2.7 Metric Learning

Metric learning is a prominent means of realization of many machine learning algorithms right from the start of their implementations. An appropriate similarity or distance metric is learned for the data and this has its utilization in realizing various learning algorithms, both supervised and unsupervised. The k-Nearest Neighbor algorithm brought about by Cover and Hart in 1967 was one of the first algorithms to make use of a distance metric to predict the label of the target variable. This was followed by an unsupervised counterpart which is also based on distance metrics called k-Means algorithm brought about by Lloyd in 1982. Further research brought about the application of Mahalanobis distance metric for nearest neighbor algorithms. In 2011, metric learning was used in link prediction in dynamic networks where links were predicted based on the features and local neighborhood around the endpoints which required a distance metric of estimation. Metric learning gained prominence in 2012 from the work of Eric P. Xing and Pengtao Xie who applied it for multi-modal applications. Multi-modal metric learning found its importance in applications such as recommendation systems, data clustering and information retrieval. Applications of metric based learning was also a prominent research in computer vision and bioinformatics. This further extended to the emergence of kernel learning which is nonparametric and implicity learns relationships or similarities among the datapoints. The emergence of kernel learning has found its way into several machine learning applications such as support vector machines and pattern analysis.

2.8 Performance Criteria

There are several machine learning algorithms but over the years of development general characteristics of learning and performance evaluation criteria have been established. The general idea is to split the available data into training, validation and test data parts. The training and validation parts are used in the learning of the algorithm and construction of the model while the test data is used to evaluate the model performance on novel data that it has never observed beforehand in its learning. There are various performance measures of the model depending on whether it is a classification or regression problem. These measures include mean square error, absolute error and cross validation techniques for regression problems and prediction error, precision, recall, F-measure (weighted average of precision and recall of the model), confusion matrix, ROC (sensitivity vs specificity curves) and AUC curves for classification problems. The initial work by C Shang in 1996 involved developing an absolute error based algorithm for evaluating communication channel equalisers adapting single-layer perceptron models. Research by Paolo Sonego in 2008 shows the application of ROC curve estimates to evaluate performance in biomedical application models and classification of biological sequences and 3D structures. Recently, precision, recall and F-measures have found their application in evaluating models in text classification for sentiment analysis and social media analytics.

3. ATTRIBUTE ANALYSIS

An important part of building predictive models from data is identifying a suitable set of attributes which characterize our goal of prediction. By attribute or feature selection, we seek to eliminate those attributes which do not contribute to the precision of our model and do not have a significant impact on the outcome of our target variable. Removing redundant features also helps with generalization issues which can occur with enormous number of non-representative attributes. With regards to features in our model, the first step is to obtain a suitable subset of predictors from the original dataset which are representative of our target of prediction and improve model precision. The next step is to study these features to check for possible interactions among them which jointly influence the dependent variable.

3.1 Correlation and Interaction

There are some aspects to consider in feature selection and it may not always be straightforward to arrive at an optimal feature set. Correlation and interaction are two terms which express some form of relationship between variables and it is important that we do not get mixed up with these terms in our feature selection process. The correlation coefficient of the features with the target can be viewed in Table 3.1. We want to select features that are highly correlated with our target variable but within our set of features we may want to eliminate those which are highly correlated with each other and may lead to redundancy. However it is possible that there exists a feature which does not have a high correlation with the target or with another feature in the set, but along with one of the correlated features may have a joint influence on the prediction. So direct elimination of such a feature may lead to loss of information. This represents a feature interaction and as we observe correlation is not the same as interaction and is not indicative of interactions among variables which affect the model.

3.2 Feature Selection Methods

Feature selection is an important part of learning algorithms. It resolves dimensionality issues and improves prediction performance of the model by eliminating irrelevant and redundant features from the predictor set. A good feature set consists of variables which are predictive in nature. In other words, feature variables should be strongly correlated with the target of the prediction model but should be ideally uncorrelated with each other. This is because features which are highly correlated with each other may tend to carry the same information which could lead to redundancy in predictor data. Likewise features which have no correlation with the target may not carry any relevant information for the target prediction. However as discussed previously, features cannot be just eliminated in terms of correlation with the target as presence of interaction terms is also something to consider in the modeling. Feature selection methods can be classified into filter and wrapper methods. Filter methods are based on various statistical relationships that can be inferred from the data and are independent of the predictive algorithm to be learned from the data. These measures involve statistics like correlation [7] of the independent variables with the dependent variable, variance filters and missing values ratio of the features. Wrapper methods are based on the performance of the learning algorithm with the selected feature set. This is computationally more complex than filter methods and consist of forward and backward feature set construction. Forward feature construction involves evaluating performance of the model starting from individual features alone and recursively adding to the feature set while backward elimination procedure works with the entire feature set initially and eliminates those features which do not contribute to model accuracy. These methods are more feasible in the scenarios when the number of features in the original set is not too high as they are exhaustive or greedy methods [4]. Filter methods are adopted to eliminate sparse and low variance (less information) or irrelevant features which do not have any positive impact on prediction performance. [5]

3.3 Feature Set

We are presented with a set of attributes relevant to students under study. These include pre-entry variables such as high school, high school grade point average, test scores and demographic information and academic factors of the students in the Texas A&M engineering program such as their performance in courses ranging from the generic engineering courses to pre-requisite courses that are relevant to their major of interest. On observation we analyze that the pre-entry variables are extremely sparse, show negligible variance across student records or have negligible correlation with the target variable. As we discussed previously, though correlation is not a complete indication of importance as it does not consider presence of feature interactions, it must be noted the former two conditions are not ideal for the modeling. A low variance filter approach enables us to prune out irrelevant features to our problem. Intuitively, we can understand that high school performance and test scores do not show any interesting variance and do not carry much information that are contributive to our model. The sparse data here is quite similar and at a fairly high standard to gain entry into the TAMU engineering program. The focus of the feature set is on the academic factors of students at Texas A&M which influence further decision making.

3.4 Model Variables

The objective of our analysis is the prediction of student performance in the ECEN pre-requisite engineering courses during the freshman and sophomore years.An important pre-requisite course in sophomore year for higher-level courses in the EE major, ECEN214 is a metric for further admission decisions.The student performance in this course based on the mined student records is the target of our prediction modeling and the dependent variable of the model.

The independent variables or predictors as they are called for the model is a set of ten Texas A&M courses taken up by students in the semesters leading up to their ETAM applications and impart the basis they need to take up the higher courses in the major.

The predictors are listed below and an initial study to get a heuristic idea about their association with the dependent variable is done. The correlation of individual predictors with the target variable is measured. The grade distributions in each of the ten independent variables can be viewed in Figure 5.1 and the value of their correlation coefficients with the target variable can be viewed in Figure 5.2

Predictors	Correlation Coefficient with ECEN214
ENGR 111	0.21
ENGR 112	0.19
CSCE 121	0.42
MATH 151	0.51
MATH 152	0.45
MATH 251	0.49
PHYS 208	0.51
PHYS 218	0.39
CHEM 107	0.45
CHEM 117	0.46

Table 3.1: Independent Variables and Association with the Target

4. CLASSIFICATION PROBLEM

The problem at our disposal is essentially a classification problem. The explanatory variables are categorical in nature and can assume any of five letter grades namely 'A', 'B', 'C', 'D', 'F'. Likewise the target variable, ECEN214 is also categorical and outcomes are expected to be classified into one of the above mentioned letter grades. As we model the classification problem, we also analyze the potential presence of interaction among the categorical features that influence classification results. The applications we consider for this problem are supervised learning methods and ensemble learning methods.

4.1 Supervised Learning

Supervised learning is a category of machine learning where labeled data is available for modeling. The training data is a set of (X,Y) where X= $(x_1,x_2...,x_m)$, m $\big($m=10 for our problem$\big)$ is the number of explanatory variables or features. Y is the target variable, ECEN214 in our case which is the labels to compare the accuracy of our predictions against. In supervised learning, our algorithm tries to learn an appropriate function Y=f(X) which when used with novel incoming data points is able to predict the correct Y label as closely as possible. The difference with unsupervised learning is that we do not have these labels available during learning of the model on training data and hence require other metrics to evaluate the model rather than prediction error. [8]

4.1.1 Factors

There are certain key factors that go along with supervised learning methods that have an important role in the performance of these algorithms. This includes bias-variance tradeoff, curse of dimensionality and complexity. The curse of dimensionality [9] occurs when there are a large number of irrelevant or redundant features to the target variable and large number of model features or coefficients could lead to variance issues in the model. This could lead to the model being extremely sensitive to training data noise and variations which brings down the generalization of the model. Hence as discussed previously, feature selection plays a huge part in resolving this issue and penalizing large number of model coefficients which increase model complexity and variance.

4.1.1.1 Bias-Variance Tradeoff

This is one of the most important factors which determines the performance of a machine learning model. The error due to bias occurs from over simplification or of the model for easy learning of the prediction function.This error can be defined as the difference between expected model prediction and true values. When models are simplified or generalized to reduce the range of this prediction error, the bias increases. The error due to variance occurs from complexity of the model such that it mimics or memorizes the variations in the training dataset and is sensitive to the noise in the training data. This leads to over fitting issues and poor generalization of the models. Ideally we want models to have low bias and low variance. The optimal model requires a tradeoff between minimizing the bias and variance.

4.2 Independence Assumption

The initial set up where we are yet to learn about potential relationships is to model the attributes with an independence assumption. The aim is to check performance with the hypothesis of class-conditional independence of explanatory variables. This assumption models the data with the basis that the predictors are independent of each other statistically given the class [6]. In other words, the joint distribution of features given the class can be factorized into the product of their marginal distributions. Let us consider the data in the form of (X,Y) where X=(x_1,x_2,...x_m) are the input features and Y is the target variable which can take on c classes. According to the class-conditional independence of the features

$$P(x_1, x_2...x_m|Y = c) = P(x_1|Y = c)P(x_2|Y = c)....P(x_n|Y = c) \tag{4.1}$$

P(x_i | Y=c) represents the marginal distribution of each of the features. In our case, we have 10 feature variables and without the assumption of independence, the number of feature combinations to evaluate the likelihood of each class would be exponential. This is drastically reduced when independence assumption comes into play and hence resolves curse of dimensionality. Although the independence assumption has high possibilities of not holding true for data, this estimation gives us a simplified analysis of the problem and helps infer potential associations in features.

4.2.1 Multinomial Naive Bayes

Naive Bayes is a simple, probabilistic classifier which we can use as the baseline model for our problem. The setting is a multiclass classification problem. Naive Bayes is set on the conditional independence assumption. Each of the features contribute independently to the outcome class and hence this model does not consider any feature interactions that could be existent in the data.

The training data is a set of (X,Y) where X is the set of features and Y is the true labels of the target variable. The model is set against the backdrop of predicting which of the classes to which an input vector could belong to. The estimation of posterior probability of generating a class given the feature set is based on the likelihood and priors. We define the notations of parameters that the model considers. [10]

The prior is given by P(X). The likelihood is given by the conditional probablity of the data given the class. This is denoted by P(X/Y). The posterior probability is given by the probability of the class of the output variable given the input data. This is denoted by P(Y/X). Joint probability distribution of the input features X and the target Y is given by

$$P(Y = c, X) = P(Y = c, x_1, x_2, .., x_m)$$

$$P(Y = c, X) = P(Y = c)xP(X|Y = c)$$

$$P(Y = c, X) = P(Y = c)P(x_1|Y = c)P(x_2|Y = c)....P(x_m|Y = c)$$

The above relation is based on the class conditional independence assumption of the feature variables. According to the Bayes theorem, a relationship can be derived between the likelihood, posterior and priors. The posterior distribution is given by

$$P(Y = c|X) = \frac{P(Y = c, x_1, x_2, .., x_m)}{P(X)}$$
$$P(Y = c|X) = \frac{P(x_1, x_2, .., x_m|Y = c)P(Y = c)}{P(X)}$$
$$P(Y = c|X) = \frac{P(x_1|Y)P(x_2|Y)....P(x_m|Y)P(Y = c)}{P(X)}$$
$$P(Y = c|X) = \frac{\prod_{i=1}^{m} P(x_i|Y)P(Y = c)}{P(X)}$$

Naive Bayes holds with categorical features conforming them as Multinomial distributions where the model is based on class counts. In other words, the Multinomial naive bayes is a specific instance of naive bayes which is used with discrete counts. The priors can be modeled with equally likely probabilities or based on the class count values. A parameter to consider is when the class count is equal to zero. There is a likelihood that a particular class may not occur at all which means it has a zero probability. So the laplacian parameter adds one to each of the classes including zero-count ones. In this case the laplacian smoothing term gives a small non-zero probability to such classes so that the posterior probabilities do not abruptly fall to zero. Thus in the case when P(x_i / Y) is zero, by laplacian smoothing [11],

$$P(x_i = x|Y = c) = (1 + \text{count with } x_i = x, Y = c)/(\text{no. of values x can take}$$
$$+\text{count with } Y = c)$$

The experimental results of the Naive Bayes model with the hold-out and cross-validation methods can be viewed in Table 5.1 and Table 5.2 respectively. The class-wise performance can be studied from the confusion matrix and the metrics in Table 5.3.

4.3 Hierarchical Models

Hierarchical models are multilevel models which capture co-existence of predictor variables that lead to an outcome of the target variable [12]. In other words, they capture the mutual dependence present in the data. Hierarchical models cluster the data into different degrees of homogeneous groups at each level such that the terminal nodes contain samples of the same category. A target variable depends on a combination of feature variables and contributions of individual feature variables is termed as the main effect. The target can also depend on combination of feature variables in addition to the main effect which is known as the interaction terms.The number of possible interaction combinations which can exist is of exponential order and hence interactions should only be considered in the context of the target variable in the classification problem. In order to limit the exponentially growing interaction set, certain constraints are imposed in the construction of hierarchical models. For instance, an interaction term is relevant only when its individual variables are predictive in nature i.e. only interaction terms whose variables have non-zero coefficients in the main effect are considered as a significant interaction which impacts the target of prediction. These constraints can be visualized for a two-way or pairwise interaction as follows.

Let us consider the input in the form of (X,Y) where X=(x_1,x_2,....,x_m) are the m predictor variables and Y is the target variable. The target variable Y can be expressed as

$$Y = \beta_0 + \sum_j \beta_j x_j + 1/2 \sum_j \theta_{jk} x_j x_k + \epsilon$$

where $\sum_j \beta_j x_j$ represents the main effect, $\sum_j \theta_{jk} x_j x_k$ represents the interaction term and ϵ represents some form of irreducible noise term in the data. According to strong and weak hierarchical constraints, the two way interaction is only considered relevant with the following conditions.

$$\text{Strong hierarchical constraint } \theta_{jk} \neq 0 \iff \beta_j \neq 0 \text{ and } \beta_k \neq 0$$

$$\text{Weak hierarchical constraint } \theta_{jk} \neq 0 \iff \beta_j \neq 0 \text{ or } \beta_k \neq 0$$

4.3.1 Decision Tree

Decision tree is a hierarchical model which can be used to predict potential outcomes of the target variable. Decision trees can either be classification or regression trees. In our case, the input to the decision tree is discrete or finite values and hence our task is to model an appropriate classification tree. Decision trees also enable interpretability, in that the conjunction of features which lead to a class of the target variable can be visualized in the form of a nested or hierarchical structure. [13]

Decision trees are binary trees constructed by recursive binary splitting of the feature space. The internal nodes represent a particular split of a t predictor variable and the leaf nodes represent a class of the output variable. The branches in conjunction from root to leaf node represent a group of decision rules which categorize the target variable. A decision tree can be interpreted as a group of classification rules by following the paths from the root to the leaf nodes. The algorithm follows a top-down approach to stratify the feature space. It can also be viewed as a divide and conquer approach of splitting the input space recursively into smaller subspaces until the results can be aggregated to arrive at a particular outcome of the target variable.

The input to a decision tree is in the form of (X,Y) where X=(x_1,x_2,...,x_m) is the set of m predictor variables and Y is the target variable which can assume any of c classes. The algorithm procedure derives the condition distribution of Y given X through a pathway of nodes and branches. This involves construction of a maximum tree and then pruning it down to the right size in order to overcome high variance and over fitting issues. The stopping criterion for the construction of the maximum tree is when homogeneity of the split cannot be improved any further and the leaf nodes have the least possible count of instances of the training data.The best split at each node can be measured in terms of certain impurity metrics based on the information gain and Gini index.

The construction of the tree with choice of a node split by measuring the information is based on how much uncertainty about the class of the dependent variable is decreased from knowledge of the feature value. In other words, the choice of split is the attribute which leads to a maximum decrease in entropy. The procedure based on this metric is as follows.

The entropy of the target variable is first computed. The input space is then split on the different attributes and the resulting entropy of each branch is computed. The difference in the entropy before and after each split is calculated and the difference represents a decrease in entropy of the information gain. The attribute which gives the highest information gain is chosen as the best split for that node and the process is repeated recursively until the stopping criterion is reached.

Step 1: Calculating entropy of target H(Y).

Step 2: Calculating the decrease in entropy of target due to knowledge of a feature X_i. For a particular class c_i of the feature variable X_i , information gain is computed and averaged over all classes of the input variable.

$$I[Y, x_i = c] = H[Y] - H[Y|x_i = c]$$
$$I[Y, x_i] = P(x_i = c)I[Y, x_i = c]$$

Step 3: The feature x_i which gives the maximum I[Y,x_i] is the decision rule at that node.

The CART algorithm constructs the splits based on the impurity criterion called Gini index which is our basis for the construction of maximum tree. The Gini index which is a generalization of binomial variance is a measure of the impurity of a data split and the attribute with the least Gini index is chosen as the best split. The leaf nodes can either be chosen to contain the class labels of the result of classification or probability scores of the class likely to represent the decision rules at that leaf node. [4] A full size decision tree without pruning for our data problem can be viewed in Figure 5.3.

4.3.2 Pruning

The construction of the maximum tree involves recursive splitting until the leaf nodes contain a very small set of homogenous values. The large number of model features and coefficients and splits in the tree could lead to over fitting issues and high variance with respect to the learning data. There could be several parts or branches of the classification tree that have weak predictive power. A complex tree model could lead to over fitting while a very simple model with one split like a decision stump could be over simplified and not capture the underlying structural information in the data. Hence an optimal size of the tree should be chosen from cross-validation such that the accuracy of the classification model is not affected. This is accomplished by first constructing the complete tree model and then pruning down to an optimal size of the tree depending on cross-validation results such that the tree is generalized but also captures the underlying information patterns. The trend of misclassification rate for different levels of pruning in our data problem can be viewed in Figure 5.4 and an optimal level is selected. The pruned decision tree which gives the optimal accuracy in a decision tree model can be viewed in Figure 5.5.

The experimental results of the decision tree model with the hold-out and cross-validation methods can be viewed in Table 5.4 and Table 5.5 respectively. The class-wise performance can be studied from the confusion matrix and the metrics in Table 5.6.

4.3.3 Ensemble Learning

Ensemble learning is a method by which multiple models are strategically constructed and aggregated to give the best possible prediction results. The performance of ensemble learning for an application depends on the choice of classifiers and the parameters specified for the model. The classifiers should have a performance better than random guessing for a new value and the errors made by the classifiers should be diverse or uncorrelated. There also exists the potential of combining multiple types of classifiers based on its performance for the particular application. The method of aggregation for prediction results in classification is a form of voting. [15]

Ensemble learning methods improve the statistical, computational and representational aspects of the prediction problem than individual models. Statistically the ensemble method comes up with the best hypothesis for the input space that gives a prediction of the target variable. Computationally, it attempts to find the best generalization of the output variable. It also tries to find the combination of classification rules which best represent the data by voting from different learning models.

Ensemble learning methods are of two types namely Bagging and Boosting methods. The difference between bagging and boosting is the method of construction of the models. Bagging works with different subsets of data for each construction which can be done in parallel while boosting is a sequential or iterative procedure that works with the entire training data but with different parameters in each construction. The learning methods of our application are random forest which is a specific instance of bagging and adaptive boosting techniques.

4.3.3.1 Bagging

Bootstrap Aggregating otherwise known as Bagging is an ensemble learning method that involves construction of multiple models to derive the most accurate prediction results. Bagging combines results from models based randomly sampled subsets of the training data to improve prediction accuracy when compared to an individual model. This method involves creating subsets of training data by randomly sampling with replacement from the original data. Our choice of classifier being the decision tree is constructed on each of the subsets of data and in the case of classification a voting scheme is used to aggregate the results from multiple classifiers. [16]

The algorithm can be summarised as follows.
The input data is in the form of (X,Y) where X=(x_1 , x_2,...x_m) are the predictors and Y is the target variable which can take on c classes where c=(c_1,c_2 .. c_n).
Step1: A base learner is chosen and the no. of classifiers to be constructed is specified.
Step2: The proportion of samples to be randomly sampled is specified and the sampling is

done with replacement for each classifier.
Step3: For each classifier a hypothesis or set of decision rules is constructed. Thus from all classifiers, an ensemble of decision rules is now constructed to test on new, incoming data.
Step4: When a test sample whose label is to be predicted is evaluated on the model, the class picked by the majority of classifier rules is assigned as the label of the test data sample.

4.3.3.2 Random Forest

Random Forest is a specific form of bagging and in many ways an improvement from traditional bagging methods. Traditional bagging methods with decision tree as classifier involves a top-down greedy approach where the input feature which gives the most homogeneous splits are chosen and the process is repeated. So there exists a possibility of structurally similar trees and correlation between multiple models constructed on the training data. On the other hand, random forest has an additional parameter to conventional bagging methods in that apart from a random subset of training data, a random subset of predictors is chosen to pick the best split at each node which reduces the likelihood of correlation between the forest of trees. The characteristics which improve performance of random forest classifier is uncorrelation between trees and the strength of each individual classifier tree. The individual trees are not pruned and are grown to the smallest possible homogeneous group of values at the leaf nodes. [17]

The parameters to be considered in a random forest model are the number of trees to be constructed and the size of the subset of predictors to be chosen for each split. The number of trees to be constructed in the model is a critical parameter in order to avoid high variance and consequently overfitting issues. The size of the subset of predictors for classification problems is usually set around the square root of the size of the entire predictor set. [18]

The model involves construction of specified number of decision trees. For each tree, the training data is obtained by random sampling with replacement of N samples where N is the number of samples in the original data. A constant number m less than the number of input variables M is specified such that at each node m variables is randomly chosen and the best split on these nodes is chosen as the node decision. The randomness with replacement of the sampling ensures a portion of samples approximately around one third the original samples are left out of the training data for the construction of each tree in the forest. These left out samples form the Out-of-Bag set and can be used for validation purposes for their respective trees. Random forest does not require a separate test dataset or separate validation procedures. The mechanism runs an internal cross validation method. Since for each tree, a set of samples are left out as the OOB set, these samples can be run down the classification tree to test for the OOB error estimate. This procedure is carried out for different sample sets on each of the constructed classification trees which is analogous to cross-validation.The variable importance or ranking of features in the context of target variable can be inferred from the random forest construction in terms of decrease in accuracy due to permutation of the feature and contribution to homogeneous splits measured in terms of Gini index.

The experimental results of the random forest model with the hold-out and cross-validation methods can be viewed in Table 5.8 and Table 5.9 respectively. Table 5.7 is a sample result with a random choice of parameters which do not give the optimal accuracy results. The class-wise performance can be studied from the confusion matrix and the metrics in Table 5.10.

4.3.3.3 Boosting

Boosting is an ensemble method similar in part to bagging but the training data for each iteration is generated based on the results of the previous iteration. The objective of this ensemble is to aggregate a set of weak learners into a strong classification.The weak learners are learned iteratively and a weighted sum of the output of the weak learners is combined to result in the final classifier.This iteration procedure is repeated until a par-

ticular misclassification rate is reached and no further improvements can be made[19]. Different boosting techniques differ in their method of weighting the weak learners and aggregating the outcomes.The generalized boosting algorithm is as follows.
The input data is in the form of (X,Y) where X=(x_1,x_2,....x_n) are the input predictors and Y is the target variable which takes on any of k classes.
Step1: A base learner is specified which constructs a function approximation based on the data tuples.
Step2: A reweighting scheme is chosen for the data points based on the results from the base learner and the reweighted samples are passed as input to the base learner for the next iteration.
Step3: A weighted aggregate of the constructed functions combine to give a strong classifier with improved prediction accuracy from the individual learners.

The boosting technique of choice is Adaptive Boosting which is distinguished from other methods by the weighting scheme it follows from every iteration [20]. The adaboost technique initially specified for binary classification based on an exponential loss function and this can be extended to muticlass classification. The extension is termed as SAMME (Stagewise Additive Modeling using Multiclass Exponential Loss Function) .The adaboost algorithm can be summarised as follows.
Step1: Specify the number of iterations for the boosting process to be repeated. Let this be denoted as N iterations.
Step2: A weak classifier is specified which takes in X,Y and weights of the samples as input. Initially all samples are given equal weights w_i=1/N where N is the number of samples in the training data. C(i) = train(X,Y,w_i)
Step3: The classifier is learned on the samples and prediction is carried out. Let the predicted labels be denoted as yhat.

yhat = predict(C(i),X)

Step4: The error of the classification is computed as

$$\text{e(i)} = \sum_{1}^{n} w_i \times \text{I}(c_i \neq \text{yhat}) \,/\, \sum_{1}^{n} w_i$$

Step5: The coefficient alpha is computed equivalent to the log odds ratio.

$$\alpha(\text{i}) = \log\ (\ (1\text{-}e^i)\ /\ e^i\ )$$

Step6: The samples are reweighted such that the misclassified points have a higher weight. Misclassified points are up weighted by a factor of exponential of alpha and renormalized.

$$w_i = w_i\ (\ \alpha^m\ .\ \text{I}(C^i(\text{x}) \neq \text{c}\ ))$$

where C^i(x) is the classifier output at the ith iteration and c is the true class. Thus when the classifier output is not equal to the true class, that is a misclassification occurs, the exponential term is retained and the misclassified samples are reweighted in the upward direction.

Step7: The final classifier is an aggregation of the learned weak classifiers and their coefficients alpha.

$$\text{C(x)} = \underset{c}{\text{argmax}} \sum_{i=1}^{N} \alpha^i\ .\ \text{I}(C^i(\text{x}) = \text{c}\ )$$

The important criterion in this binary classification is that the weak classifier should have a performance better than random guessing of 1/2. In other words for the boosting to or reweighting of samples in the right direction for successive iterations, the error term e(i) should be less than 1/2 for alpha to be a positive term and misclassified points to be weighted in the increasing order. Else alpha gets a negative value and the weighting procedure follows the opposite order.

The same theory can be extended to a multiclass classification with slight variations. [21] The SAMME algorithm is summarized as follows:

Step1: The initial steps are which involve equal weighting of sample, defining a weak classifier, prediction and computation of the error term are the same as the two class adaboost.

Step2: The difference lies in the computation of the coefficient alpha.

$$\alpha(\text{i}) = \log\ (\ (1\text{-}e^i)\ /\ e^i\ ) + \log(\text{k-1})$$

where k is the number of classes in the multiclass classification.The second term included is in relation to maintaining the direction of boosting. In the two class method we required the error term to be less than random guessing of 1/2 for alpha to be positive and weighting to be done correctly. The threshold 1/2 is too rigid a metric for the case where multiple K classes exist. Hence the inclusion of this term now requires the classifier to perform better than random guessing which is 1/K in this case of K classes for boosting to follow the right direction.

Step3: The reweighting of samples with boosted weights for misclassified points is given by

$$w_i = w_i \text{ (} \alpha^m \text{ . I(} C^i \text{(x)} \neq \text{c))}$$

Step4: The final classifier is given by the aggregation from the iterations as

$$\text{C(x)} = \underset{c}{\text{argmax}} \sum_{i=1}^{N} \alpha^i \text{ . I(} C^i \text{(x) = c)}$$

A key parameter to consider in the boosting procedure is the termination point. There could be instances when the training error is almost negligible but the validation or testing error is still relatively high. Hence an optimal choice of number of iterations is to be specified after which there can be no further improvements in prediction accuracy.

The experimental results of the Adaboost model with the hold-out and cross-validation methods can be viewed in Table 5.11 and Table 5.12 respectively. The class-wise performance can be studied from the confusion matrix and the metrics in Table 5.13.

4.4 Metric Learning

Classification performance can be greatly influenced by definition of an appropriate metric that captures the relationship between data in the input space. Metric based learning is a means to encode higher order interactions and model the data based on important factors observed from it. A similarity or distance metric between pair of input points in the feature space should be able to define minimum distance between similar points and maximum distances between dissimilar points. This method optimizes a loss function which defines distance measures between similar class of points and those between different class of points.

We define an appropriate similarity metric for our application using a support vector approach for the classification problem.

4.4.1 Support Vector Approach

The support vector approach to a classification problem involves categorizing a sample into one of two categories separated by a distinct gap or margin. The basic support vector machine is a non-probabilistic linear classifier which assigns class predictions to incoming data samples based on which side of the margin they fall in the feature space. Data points are plotted in a m-dimensional space where m represents the number of predictors and the hyperplane which separates different categories of data is to be defined. The optimal choice of separating hyperplane is the one which has the largest possible gap between the opposite categories of data points. Hence the support vector approach to classification can also be termed as a maximum margin hyperplane classifier. This optimization problem depends on the linear separability of data in the input space. The support vector approach can be summarized as follows. [23]

The input data to learn the algorithm is in the form of (X,Y) where X=(x_1,x_2.,..,x_m) are the input feature variables and Y is the target variable for prediction. For the base case, let us assume that Y takes one of two classes. The input space can be visualized as m-dimensional datapoints and the objective is to find the maximum margin hyperplane in this input space which separates the two classes of the target variable. The optimal hyperplane can be defined as the one satisfying

$$\text{wx-b=0}$$

where w is the normal to the hyperplane.The key feature in this approach is the linear separability of data and hence we can define margins on either side of the hyperplane such that points from either side do not fall inside this margin and the distance from the hyperplane is maximized. The margins can be defined as

$$wx-b=1$$

$$wx-b=-1$$

The classification is such that points which lie above wx-b $\geq$ 1 belong to a class and the points which lie below wx-b $\leq$ -1 belong to the other class thus obtaining a classification between the two groups.

This approach originally based on classification between two classes can be extended to define a multiclass classification problem. The multiclass problem can be construced as a group of two class problems. There are different approaches to do this such as 'one-vs-one' and 'one-vs-all' [24]. Let us assume Y takes any of c classes.The one-vs-all approach involves formulation of c support vector classifiers. The first support vector machine considers the first class as the positive label and the other classes as the opposite category. Likewise the cth support vector machine considers class c points as the positive label and the other classes as those which lie on the other side of the hyperplane thus resembling multiple two class SVM constructions.The one-vs-one approach considers a pair of classes for each support vector machine construction. Hence if there exists c classes, there exists construction of c(c-1)/2 support vector machines each involving a pair of classes and the results are aggregated for the final classification of multiple classes. The choice of approach for our mutliclass classification problem is the one-vs-all approach. Thus if the output variable Y takes n classes, n binary classifiers are constructed which categorizes one class from the rest. This is combined to determine multiclass decision based on the maximal output. Each classifier returns a signed value which is a confidence value of the sample x belonging to the positive class in that classifier. Higher distance away from the hyperplane gives a larger value. The sample x is labeled with the class with maximum confidence value among all the classes.

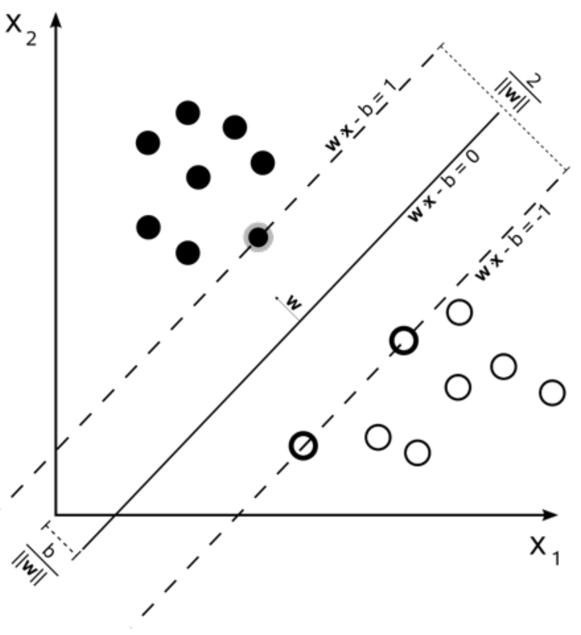

Figure 4.1: SVM with Support Vectors - Adapted from [23]

The SVM optimization problem involves maximizing the margin of separation between opposite classes and this is where a similarity or distance metric between pairs of datapoints comes into play and metric based learning can be applied with a support vector classification approach. The objective is to maximize the margin L between the categories of samples. The critical samples for the optimization are those called as support vectors which lie on the margins and only these samples are assigned a non-zero weight coefficient. The support vectors are the points on the margins which are highlighted in Figure 4.1. This can be defined as follows.

$$\max L = \sum_i a_i - 1/2 \sum a_i\, a_j\, y_i\, y_j\, (\, x_i \,.\, x_j)$$

where, $\sum_i a_i = 0$ and $a_i \geq 0$.

a_i is non-zero only for the support vectors. Consider the case when two similar support vectors x_i and x_j, i $\neq$ j have the same sign of labels y_i and y_j. Note that x here only for the support vector formulation represents m-dimensional data points and not the features.

In this case the margin L will be decreased based on the second term in the right hand side of the above equation. The critical case to maximize the margin is when similar samples have opposite sign of labels. Consider x_i and x_j are similar points but have opposite labels y_i and y_j. Hence from the above equation we observe that this is the case which enables maximization of the margin L.

4.4.2 Linear Separability

The support vector approach that has been described can be realized in the input space if and only if there exists linear separability between the categories of data. Thus a hyperplane and margin can be defined between the separable categories of data and margin distance between the nearest points maximised. There is a huge probability that the linear separability condition does not hold for real-world data as is our case and hence the maximum margin approach cannot be implemented in the input space. Hence a means should be defined such that linear separability can be achieved in the data which can be viewed in Figure 4.2. This can be achieved by mapping the data points to a higher dimensional space such that linear separability of data is achievable in the transformed space. A mapping function ϕ is defined such that the trannsformed data points given by ϕ(x) in the higher dimensional dot product space shows linear separability between different classes of the data. Thus the problem to be solved in the transformed space with data points mapped with the function ϕ can be defined as

$$\text{max L} = \sum_i a_i - 1/2 \sum a_i \, a_j \, y_i \, y_j \, (\phi(x_i).\phi(x_j))$$

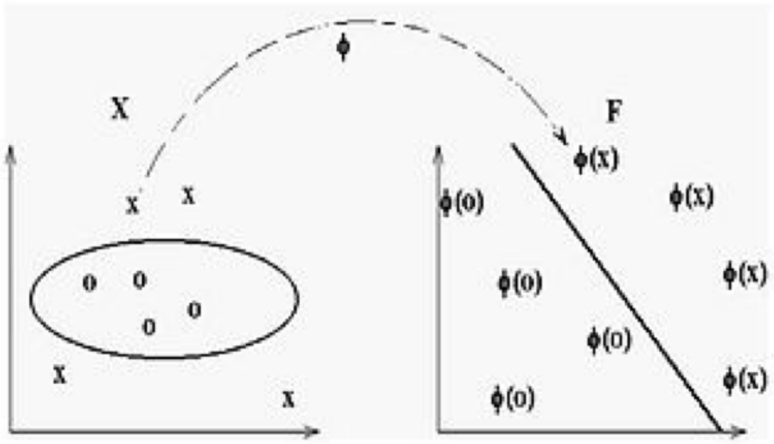

Figure 4.2: Mapping function - Adapted from [26]

4.4.3 Kernel Trick

The explicit computation of the mapping of data points to a higher dimensional space can be prohibitive. The explicit mapping can be avoided by a method called as Kernel trick. This method involves the substitution of a kernel metric between pairs of data points in the input feature space which is equivalent to the dot product of transformed data points in the higher dimensional space without having to explicitly compute the mapping coordinates. Thus the dot product of non-linearly separable data points can be computed in the input feature space with the kernel metric without explicit mapping to the higher dimensional dot product space [25]. Hence the problem to be solved transforms as follows. [27]

$$\max \mathrm{L} = \sum_i a_i \text{ - } 1/2 \sum a_i\ a_j\ y_i\ y_j\ (\phi(\ x_i).\phi(x_j))$$
$$\max \mathrm{L} = \sum_i a_i \text{ - } 1/2 \sum a_i\ a_j\ y_i\ y_j\ \mathrm{K}(\ x_i,x_j\ )$$

where,

$$\mathrm{K}(x_i,x_j) = \phi(x_i).\phi(x_j)$$

4.4.4 Kernel Conditions

Mercer's theorem gives a set of conditions for the kernel metric to represent a proper inner product [27]. We know that for a mapping function ϕ : X -> Z

$$K(x_i,x_j) = < \phi(x_i),\phi(x_j) >_Z$$

$$K(x_i,x_j) = \phi(x_i) \,.\, \phi(x_j)$$

where Z is an inner product space. The condition is that the pairwise inner products is equivalent to a positive-definite kernel matrix. In other words for any h(x) with finite L_2 norm, that is

$$\int h(x)^2 \text{ dx is finite}$$

The kernel function of the pair of samples which is equivalent to the dot product of their mapped or transformed points in the space Z holds if and only if

For real valued K(x_i,x_j)

$$\int K(x_i,x_j)\ h(x_i)\ h(x_j)\ dx_i\ dx_j \geq 0$$

For discrete scenarios, the kernel matrix should satisfy

$$\sum K(x_i,x_j)\ h(x_i)\ h(x_j)\ dx_i\ dx_j \geq 0$$

Thus the kernel matrix should be positve semi-definite. The kernel matrix is called the gram matrix which satisfies the positive semi-definite condition i.e. K^TK is positive semi-definite. If there are N data points, the Gram matrix is defined as

$$K \in R^{NxN} \text{ and } K_{ij} = K(x_i,x_j) = \phi(x_i) \,.\, \phi(x_j)$$

4.4.5 Kernel Definition

The input data at our disposal is categorical in nature and the objective is to be able to define our kernel metric with categorical predictors rather than re-encoding them as numeric values. The common scheme is to encode c distinct categorical levels of the input variables into c-1 numeric predictors called as dummy variables and then learn them using a metric.

Kernels provide a metric for pattern analysis and likewise prediction of similar pairs of variables. According to the support vector machine approach, the critical case of support vectors is when similar pairs of variables have opposite classes and the margin is to be maximized between them. So defining a similarity metric between pairs of students acts as a suitable distance metric to classify between different grade labels of the students. We define a form of kernel which computes the overlapping grades between a pair of students and provides an exponential weighting for the average number of course-wise similar grades between the students.

Our kernel metric for this application can be defined as follows. Consider every pair of data points denoted as x and y. Let m be the dimension of the data point which is the number of feature variables (m=10 in this case). A data point x is of the form x= (x_1, x_2,...,x_m) where x_i is a feature value of the data point. The kernel or similarity metric between a pair of data samples is given by

$$\text{K(x,y)} = \phi\text{(x)}.\phi\text{(y)}$$
$$\text{K(x,y)} = \exp\left(\frac{\gamma}{m} \sum_{i=1}^{m} \mathbb{1}\ (x_i = y_i)\right)$$

where m is the number of features (10) , γ is a tuning parameter which is constant and x and y are a pair of data points. The kernel matrix or Gram matrix M is a nxn matrix where n is the number of data points and M_{ij} represents the kernel metric between ith and jth data points.

4.4.6 Mapping Intuition

The function of the kernel trick is to avoid explicit computing of the mapping function ϕ. We attempt to formulate how the mapping function ϕ equivalent to our kernel metric maps the data points to achieve linear separability in a higher dimensional space.
Consider the input X which consists of N data points each with m dimensions or feature variables. Thus a data point x_i is given by x_i=(x_i^1 , x_i^2,.....,x_i^m). Let us take a pair of points x_1 , $x_2 \in X^m$.

$$K(x_1,x_2) = \phi(x_1).\phi(x_2)$$

The mapping ϕ can be defined as $\phi : X_m$ -> Z_n where n < m and

$$n = \text{no. of values } x_i \text{ can take} \times m$$

In our case, the dimension of the transformed space is 5×10 because X can take one of the 5 grades A,B,C,D,F and the number of predictor variables is 10.

The mapping of a point to the mentioned dimension can be viewed as a one-hot encoding scheme. For example the following grades have a mapping to a higher dimensional vector as follows.

A -> [0, 0, 0, 0, 1]

B -> [0, 0, 0, 1, 0]

C -> [0, 0, 1, 0, 0]

D -> [0, 1, 0, 0, 0]

F -> [1, 0, 0, 0, 0]

Thus to examine this mapping in a feature vector, let us for example take two data points with 4 features x_1 = [3,2,1,0] amd x_2 = [3,1,0,0]. Thus their mapping can be defined as follows

x_1=[3,2,1,0] , $\phi(x_1)$ = [[0,0,0,1,0] , [0,0,1,0,0] , [0,1,0,0,0] , [1,0,0,0,0]]

x_2=[3,1,0,0] , $\phi(x_2)$ = [[0,0,0,1,0] , [0,1,0,0,0] , [1,0,0,0,0] , [1,0,0,0,0]]

The dot product of the transformed points is the average sum of the overlapping encoded feature vectors. This is equivalent to our defined kernel in the input space which gives an exponential weighting to the similarity or overlap measure between the feature values between every pair of data points.

The experimental results of the kernel-based support vector classification model with the hold-out and cross-validation methods can be viewed in Table 5.14 and Table 5.15 respectively. The class-wise performance can be studied from the confusion matrix and the metrics in Table 5.16.

5. EXPERIMENTAL RESULTS

5.1 Performance Assessment

The problem to be modeled is essentially a classification problem and we define certain performance assessment criteria to evaluate the performance of each applied algorithm and derive inferences from it. The evaluation metrics for our models are listed as follows

5.1.1 Prediction Error

We tackle the classification problem using supervised learning methods due to the availability of true class labels of the available data. Hence it is possible to compute the correct and incorrect class predictions for the samples of data by comparing prediction results with the true labels and hence determine the misclassification error as the proportion of incorrectly classified samples. Conversely, the prediction accuracy is a measure of the proportion of samples that have been correctly classified by the models.

5.1.2 Hold-out and Cross-Validation Accuracy

We determine the model performance using two types of validation methods. One being the hold-out accuracy and the other being cross-validation accuracy. The hold-out procedure consists of splitting the data into two sets namely the training and the testing set. The two sets of data are disjoint or mutually exclusive of each other and commonly the training set is the larger of the two sets. Hence, in this method the classifier is learned or trained on the training set without any exposure to the test set. The trained model is then applied to the test set and outputs predictions for each of the test sample classes which is compared with the true labels. Cross validation on the other hand involves splitting data into folds and training the classifier on data except a particular fold which is left for validation. This procedure is repeated across all folds and the final accuracy is representative of the average performance accuracy over all folds. We are interested to see how hold-out and cross validation accuracies of our chosen models compare against each other. [29]

The per class statistics for classification performance of the model is visualized using the following criteria.

5.1.3 Confusion Matrix

The confusion matrix is a means to visualize the per class prediction performance of the chosen models. The confusion matrix is of the form (true class, predicted class) where the rows represent true class and columns represented predicted class. Hence we can infer that correctly classified points are grouped corresponding to the classes in the diagonal entries of the confusion matrix.

$$\begin{array}{cccccc} F & D & C & B & A & \\ 0 & 0 & 0 & 0 & 0 & F \\ 0 & 0 & 0 & 0 & 0 & D \\ 0 & 0 & 0 & 0 & 0 & C \\ 0 & 0 & 0 & 0 & 0 & B \\ 0 & 0 & 0 & 0 & 0 & A \end{array}$$

A sample at M_{ij} where i=j indicates the true class and predicted class are the same thus representing an accurate classification (diagonal positions). A sample which goes into M_{ij} where i $\neq$ j indicates that the true class i and predicted class j are not the same thus representing a misclassification.

5.1.4 Precision and Recall

Precision and recall are metrics which give us a picture about model performance and can be evaluated from the confusion matrix. Precision is a measure of how many of the predicted values in a class are correctly classified or actually part of the true labels of the class. Hence it is a measure of positive prediction by the model. Recall on the other hand is a measure of the amount of information correctly retrieved or in other words the number of samples correctly predicted. Models can be optimized based on a measure

which combines or balances both precision and recall. This is called F-measure which is a weighted average of the precision and recall of the model. Precision and recall can be computed from the confusion matrix to determine model performance. In a binary sense, the confusion matrix portrays the number of true positives, true negatives, false positives and false negatives.
In the binary case, the confusion matrix is given by

$$\begin{matrix} TP & FN \\ FP & TN \end{matrix}$$

The precision and recall of the classifier can be computed from these values by the following formulations.

Precision = TP / (TP+FP)

Recall = TP / (TP+FN)

This concept can be extended to the multiclass case for the precision and recall formulations.If M represents a confusion matrix for multiple classes, M being a k x k matrix where k is the number of classes

Precision = M_{ii} / $\sum_j M_{ji}$

Recall = M_{ii} / $\sum_i M_{ij}$

5.1.5 ROC Curve and AUC

Sensitivity and specificity are two measures which also capture model performance. Sensitivity which is synonymous with Recall is a measure of the proportion of positives (for each class in a multi class sense) that are correctly predicted by the model. Specificity is a measure of the proportion of negatives(in the multi class sense it refers to samples belonging to other classes when we consider a particular class) which are predicted right. These measures can be combined visually to characterize the model behavior with respect to the data. This leads us to the metrics which are ROC curve and AUC generally used with positives and negatives in a binary classification but can be extended to the multi class case with a one-vs-rest scheme. The ROC curve is a plot of the true positive rate or recall or sensitivity vs the false positive rate which is 1-specificity of the model. The accuracy of the test is characterized by separation of positives and negatives for the class under consideration. This can be represented by the area under the ROC curve. The closer this measure is to 1 the better is the performance of the model test. The baseline measure is random prediction which is 1 / the number of classes. AUC values below random prediction indicates that the model is performing worse than the baseline which is random guessing.

5.1.6 PRC Curve Area

The PRC curve is a measure of precision against recall. The recall axis or true positive rate is common with ROC curve but the other axis in this case is Precision. PRC is hypothesized to carry more information about the classification performance when there is an imbalance in class sample distribution in the dataset. This is because due to imbalance in the data, say skew towards the number of negatives, a large change in false positives affects precision and hence the PRC curve much more significantly than the ROC curve. Hence performance with respect to skewed datasets is more accurately captured with the PR Curve. In the case of a balanced dataset, the ROC and PR curve contain the same plot curves and hence the same interpretation [30].

We start with getting an idea about the data distribution and statistical relationships such as correlation with the target variable.We observe the correlation coefficient of the explanatory variables with the target variable and the degree of individual association with the target. The following is the distribution statistic of grades in each course considered for the model and the correlation coefficient of individual courses with the target variable.

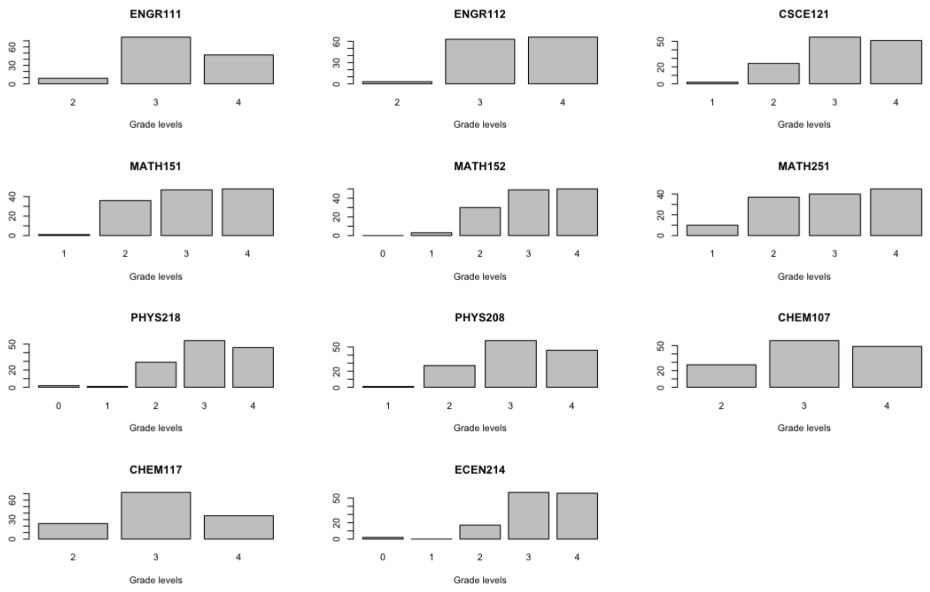

Figure 5.1: Grade Distribution Plots

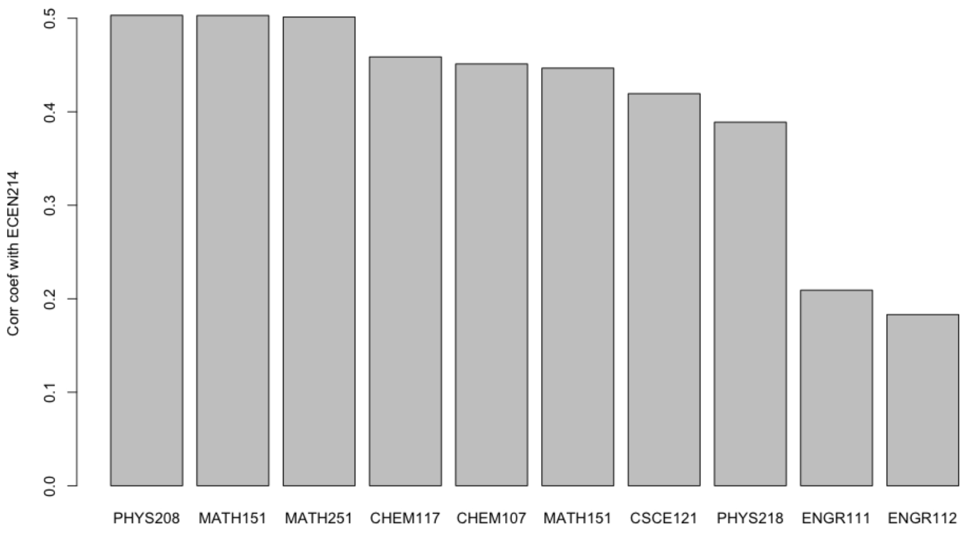

Figure 5.2: Correlation Coefficients

5.2 Naive Bayes

The Multinomial Naive Bayes model is evaluated using hold-out accuracy and 4-fold cross validation technique on the dataset. The data is categorical and has 5 levels or classes namely A(4) , B(3), C(2), D(1), F(0).The classification results can be observed as follows.

Correctly Classified Instances	62	62.6262 %
Incorrectly Classified Instances	37	37.3737 %

Table 5.1: Naive Bayes Hold-out Accuracy and Error

Correctly Classified Instances	117	64.2857 %
Incorrectly Classified Instances	65	35.7143 %

Table 5.2: Naive Bayes Cross-Validation Accuracy and Error

The confusion matrix is constructed as follows from the cross-validation prediction where diagonal entries represent correctly classified samples and the rest represent misclassifications. This enables us to visualize prediction results and related derived statistics in a class-wise manner from the data.

$$\begin{array}{cccccc}
F & D & C & B & A & \\
2 & 0 & 1 & 0 & 0 & F \\
0 & 0 & 0 & 2 & 0 & D \\
0 & 0 & 8 & 14 & 1 & C \\
0 & 0 & 12 & 43 & 19 & B \\
0 & 0 & 2 & 14 & 64 & A
\end{array}$$

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	PRC Area	Class
0.667	0.017	0.400	0.667	0.500	0.86	0.513	F
0.000	0.000	0.000	0.000	0.000	0.581	0.019	D
0.348	0.069	0.421	0.348	0.381	0.804	0.375	C
0.581	0.278	0.589	0.581	0.585	0.744	0.587	B
0.800	0.206	0.753	0.800	0.776	0.896	0.896	A

Table 5.3: Naive Bayes Prediction Statistics

5.3 Decision Tree

The decision tree is constructed by recursive binary splitting with the original predictor set over the feature space. Initially, construction of maximum size tree is done. The full size tree had a complex model of 75 node splits. The next step is to ensure the tree is not overfitting with high variance and complexity and hence cross-validation techniques enable us to discover the opitmal splits of the tree and prune down the tree to that size. The following plot indicates the best sizes to which the tree should be pruned to achieve optimal accuracy. We observe that tree size of 2-7 nodes gives reduced error performances. It is important to choose the pruning such that bias-variance tradeoff is achieved and a sample pruned decision tree is shown below.

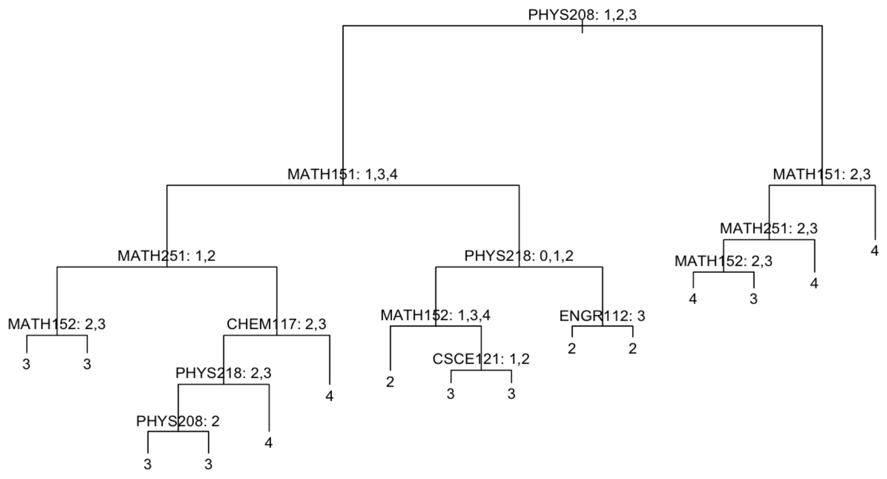

Figure 5.3: Maximum Size Decision Tree

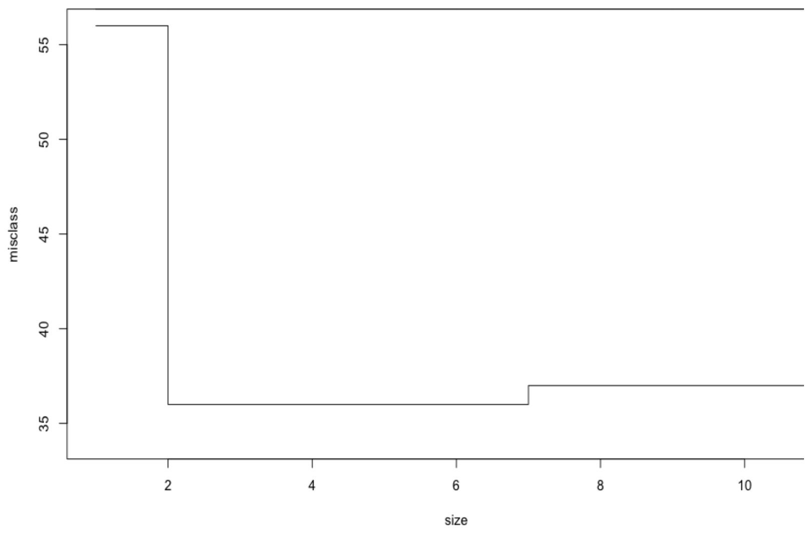

Figure 5.4: Pruning size vs Misclassification Rate

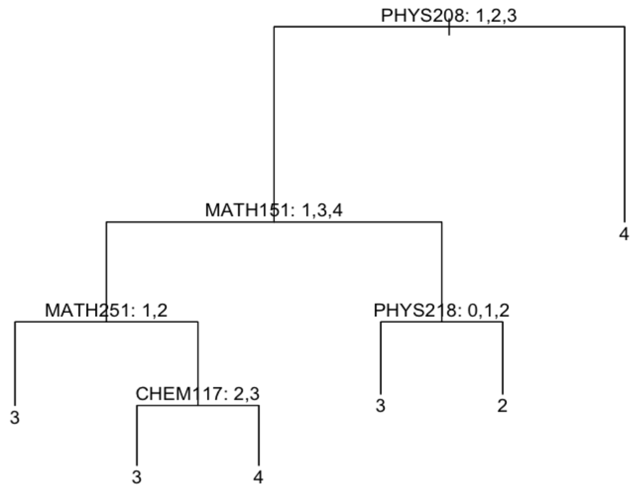

Figure 5.5: Sample Pruned Decision Tree

Correctly Classified Instances	64	64.6464 %
Incorrectly Classified Instances	35	35.3535 %

Table 5.4: Decision Tree Hold-out Accuracy and Error

Correctly Classified Instances	120	65.9341 %
Incorrectly Classified Instances	62	34.0659 %

Table 5.5: Decision Tree Cross-Validation Accuracy and Error

The confusion matrix is constructed as follows from the cross-validation prediction where diagonal entries represent correctly classified samples and the rest represent misclassifications. This enables us to visualize prediction results and related derived statistics in a class-wise manner from the data.

$$\begin{array}{cccccc}
F & D & C & B & A & \\
0 & 0 & 1 & 2 & 0 & F \\
0 & 0 & 0 & 1 & 1 & D \\
0 & 0 & 8 & 11 & 4 & C \\
0 & 1 & 10 & 47 & 12 & B \\
0 & 0 & 3 & 12 & 65 & A
\end{array}$$

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	PRC Area	Class
0.000	0.022	0.000	0.000	0.000	0.808	0.144	F
0.000	0.011	0.000	0.000	0.000	0.342	0.011	D
0.348	0.088	0.364	0.348	0.356	0.820	0.371	C
0.622	0.241	0.639	0.622	0.630	0.763	0.711	B
0.813	0.167	0.793	0.813	0.802	0.885	0.846	A

Table 5.6: Decision Tree Prediction Statistics

5.4 Random Forest

The choices of parameters for the random forest model are shown below and we take a look at the performance results for different tuning of the parameters.

Base Learner : Decision Tree

No. of trees in the forest : 100

No. of cross-validation folds : 4

Predictor subset size : 3

Max depth of trees : 3

Correctly Classified Instances	139	76.3736 %
Incorrectly Classified Instances	43	23.6263 %

Table 5.7: Random Forest Cross-Validation Accuracy and Error - Trial parameters

Base Learner : Decision Tree

No. of trees in the forest : 100

No. of cross-validation folds : 6

Predictor subset size : 3

Max depth of trees : 5

Correctly Classified Instances	151	82.967 %
Incorrectly Classified Instances	31	17.033 %

Table 5.8: Random Forest Cross-Validation Accuracy and Error - Optimal Parameters

We analyze that the optimal depth parameter of the constructed base learners is 5 and determine the hold-out prediction accuracy in this setup.

Correctly Classified Instances	80	80.808 %
Incorrectly Classified Instances	19	19.1919 %

Table 5.9: Random Forest Hold-out Accuracy and Error

The confusion matrix is constructed as follows from the cross-validation prediction where diagonal entries represent correctly classified samples and the rest represent misclassifications. This enables us to visualize prediction results and related derived statistics in a class-wise manner from the data.

$$\begin{array}{cccccc}
F & D & C & B & A & \\
2 & 0 & 1 & 0 & 0 & F \\
0 & 0 & 0 & 1 & 1 & D \\
0 & 0 & 10 & 12 & 1 & C \\
0 & 0 & 1 & 59 & 14 & B \\
0 & 0 & 0 & 13 & 67 & A
\end{array}$$

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	PRC Area	Class
0.667	0.000	1.000	0.667	0.800	0.901	0.685	F
0.000	0.000	0.000	0.000	0.000	0.311	0.012	D
0.435	0.006	0.909	0.435	0.588	0.852	0.693	C
0.797	0.250	0.686	0.797	0.738	0.877	0.849	B
0.838	0.157	0.807	0.838	0.822	0.933	0.929	A

Table 5.10: Random Forest Prediction Statistics

5.4.1 Variable Importance

The feature importance can be ranked from the Random Forest construction based on certain metrics in terms of accuracy and contribution to purity or homogeneity of split [31]. These metrics are Mean Decrease Accuracy and the Gini Index and the variable importance plot for this case can be viewed in Figure 5.6. The Mean Decrease Accuracy metric refers to the drop in accuracy or the proportion of samples that are incorrectly classified by removing the feature under consideration from the model. This can be done by first learning the forest of trees and computing the cross-validated out-of-bag error estimate including the feature. Then, the feature under consideration is permuted from the predictor set and the out-of-bag error estimate with the feature permuted is computed. The level of difference in accuracies is indicative of the importance of the feature variable in the model. The Gini index represents the contribution of a feature variable to the homogeneity or purity of a split in the construction of the trees. When a feature variable contributes to purity of split and the feature is permuted from consideration for the split, it leads to an increase in impurity of split or a significant drop in mean gini coefficient. Hence the level of drop is an indicator of how important or useful the variable is in the construction of the model.

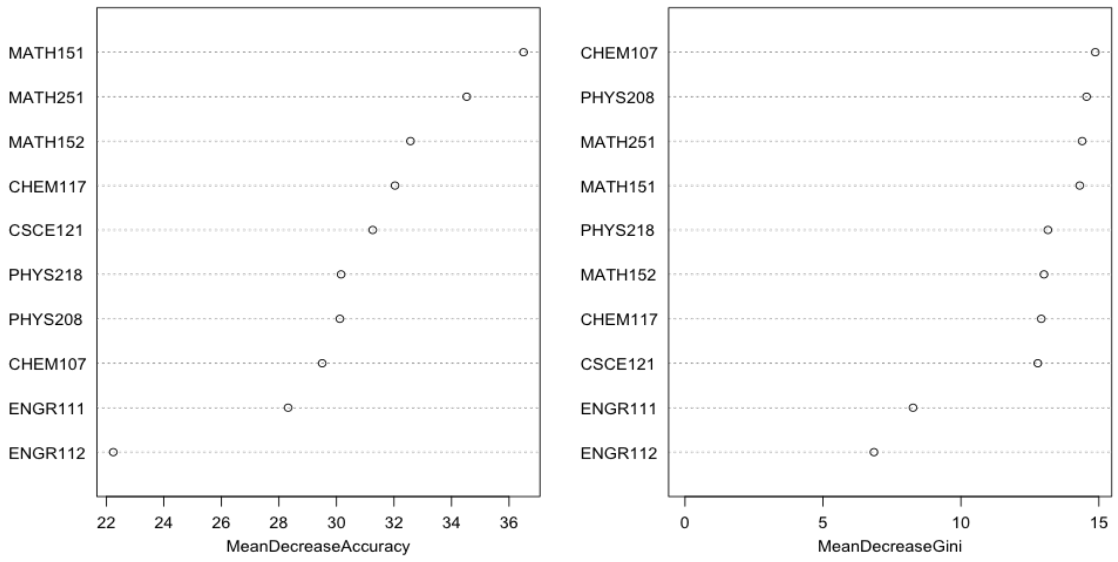

Figure 5.6: Variable Importance Plots

5.5 AdaBoost

The critical parameter is to specify when the boosting should be stopped (number of iterations). An optimal choice is made based on the least obtainable training and prediction error and adaptive boosting is terminated when no further improvements in accuracy can be made. The trend of training and testing errors for various choice of termination iteration and a non-optimal and optimal choice results can be viewed in Figure 5.7 and Figure 5.8

respectively. For instance, choice of number of iterations as 30 gave almost negligible training error but the cross validation/testing error was still relatively high. The optimal choice for no. of iterations in this case was 400 iterations with base learner as the decision tree.

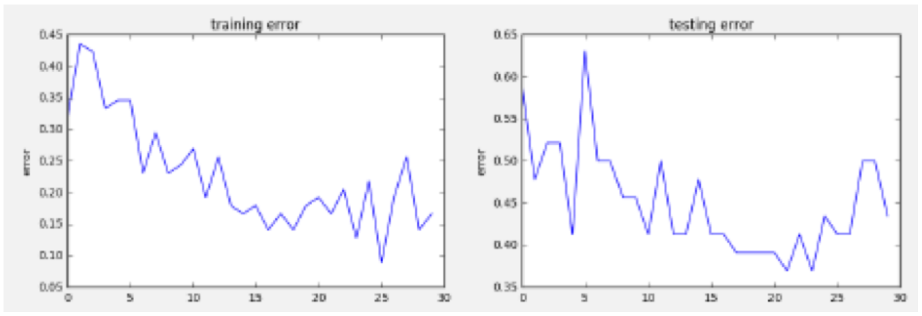

Figure 5.7: Non-optimal Iteration Number Choice

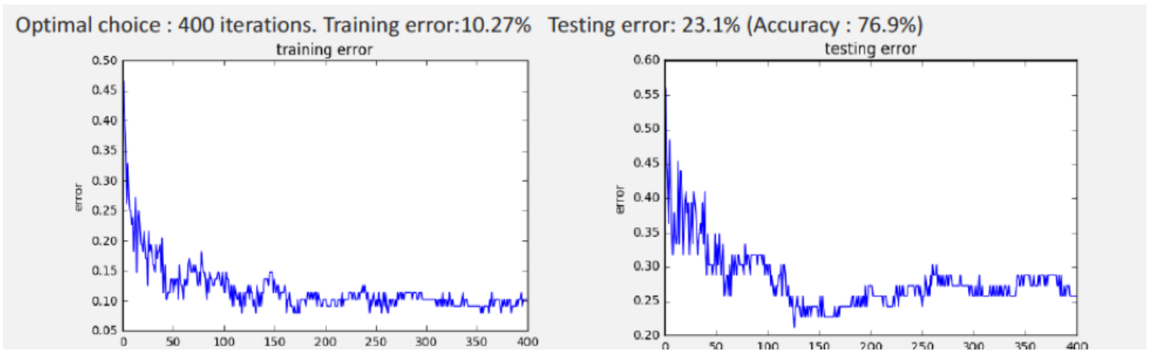

Figure 5.8: AdaBoost Error Plots

Correctly Classified Instances	75	75.7575%
Incorrectly Classified Instances	24	24.2424 %

Table 5.11: AdaBoost Hold-out Accuracy and Error

Correctly Classified Instances	140	76.9231%
Incorrectly Classified Instances	42	23.0769 %

Table 5.12: AdaBoost Cross-Validation Accuracy and Error

The confusion matrix is constructed as follows from the cross-validation prediction where diagonal entries represent correctly classified samples and the rest represent misclassifications. This enables us to visualize prediction results and related derived statistics in a class-wise manner from the data.

$$\begin{array}{cccccc}
F & D & C & B & A & \\
2 & 0 & 1 & 0 & 0 & F \\
0 & 0 & 0 & 1 & 1 & D \\
0 & 0 & 11 & 11 & 1 & C \\
0 & 3 & 4 & 60 & 7 & B \\
0 & 0 & 0 & 13 & 67 & A
\end{array}$$

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	PRC Area	Class
0.667	0.006	0.667	0.667	0.667	0.983	0.535	F
0.000	0.017	0.000	0.000	0.000	0.542	0.032	D
0.435	0.019	0.769	0.435	0.556	0.807	0.557	C
0.811	0.241	0.698	0.811	0.750	0.837	0.703	B
0.838	0.098	0.870	0.838	0.854	0.902	0.848	A

Table 5.13: AdaBoost Prediction Statistics

5.6 Kernel Metric Based SVM

The defined kernel metric is suitable for categorical features without having to re-encode them as dummy values. The optimal value of tuning parameter γ for our application is 10.

$$K(x,y) = \exp\left(\frac{\gamma}{m} \sum_{i=1}^{m} \mathbb{1}\,(x_i = y_i)\right)$$

The classification results are shown as follows

Correctly Classified Instances	78	78.7879%
Incorrectly Classified Instances	21	21.2121 %

Table 5.14: Kernel SVM Hold-out Accuracy and Error

Correctly Classified Instances	154	84.6154%
Incorrectly Classified Instances	28	15.3847 %

Table 5.15: Kernel SVM Cross-Validation Accuracy and Error

The confusion matrix is constructed as follows from the cross-validation prediction where diagonal entries represent correctly classified samples and the rest represent misclassifications. This enables us to visualize prediction results and related derived statistics in a class-wise manner from the data.

$$\begin{array}{cccccc}
F & D & C & B & A & \\
2 & 0 & 1 & 0 & 0 & F \\
0 & 0 & 0 & 2 & 0 & D \\
0 & 0 & 14 & 8 & 1 & C \\
0 & 0 & 0 & 64 & 10 & B \\
0 & 0 & 0 & 6 & 74 & A
\end{array}$$

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	PRC Area	Class
0.667	0.000	1.000	0.667	0.800	0.909	0.686	F
0.000	0.000	0.000	0.000	0.000	0.269	0.011	D
0.609	0.000	1.000	0.609	0.757	0.902	0.782	C
0.878	0.139	0.813	0.878	0.844	0.918	0.895	B
0.925	0.118	0.860	0.925	0.892	0.971	0.965	A

Table 5.16: Kernel SVM Prediction Statistics

6. SUMMARY AND CONCLUSIONS

We compile a comprehensive summary of the steps taken to tackle this education analytics problem and model the data to arrive at meaningful prediction methods and competitive accuracy results. The problem is essentially a supervised classification problem and our goal is to predict student performance based on relevant attributes which gives both the department and student a better picture for their ETAM applications and decision-making. We analyze the attributes presented to us and based on preliminary filter methods arrive at a predictor set which represents their academic performances over the course of the general engineering program. From here, we take it to wrapper feature selection methods using classification algorithms to further pin-point important attributes which impact our target. We initially evaluate model performance with the assumption that feature variables are statistically independent of each other. The next step was to model the features into multi-level models signifying non-linear interactions to result in categorical predictions. The model for this purpose is the decision tree which is a hierarchical model. There is potential for the hierarchical algorithm to fit and learn the data better in order to vastly improve prediction accuracy results. This is accomplished by means of ensemble learning methods. The two types of ensembles used are Random Forest which is a specific type of bootstrap aggregating method (Bagging) and Adaptive Boosting which is an iterative boosting ensemble that follows a different algorithm to Bagging. The performance results are compared for both types. We round up our evaluation with the application of a support vector based classification. The non-linear separability in the data calls for a metric based support vector approach which involves the application of a kernel metric that captures similarity between data points and aids in the categorical classification using a maximum margin separation approach between classes. From our application we are also able to deduce the ranking of features based on their importance in influencing the target outcome. In the course of evaluation, we compare the difference in results between two types of validation methods, one being the hold-out testing method and the other being a cross-validation approach. To further breakdown the model, we look at methods like the confusion matrix and precision-recall based metrics to get a more detailed insight into the model performance.

A notable observation is the imbalance of classes in our dataset. There is a vast abundance of data points with labels 'A','B' and 'C' grades which is very high compared to the number of 'F' and 'D' labeled datapoints. Particularly the presence of 'A' and 'B' labeled data points is very high compared to the rest of the classes. Consequently we observe that per class performance can only be significantly looked at for classes 'A','B' and 'C' due to presence of sufficient number of samples and these classes have noteworthy precision-recall statistics which can be looked at. On the other hand, scarcity of the other classes makes it difficult to interpret its results. We observe that the individual performances of classes 'A' and 'B' with respect to their total number of samples has a good ratio of correctly classified to misclassified instances. The independence assumption model gave the least accuracy results among our applications with an accuracy of 64.29%. This is our baseline as we observed significant improvements in accuracy results with other models which consider feature dependencies. An interesting aspect is the difference between the accuracies in hold-out testing evaluation and cross-validation evaluation. A common trend we observe in our models is the cross-validation accuracy being slightly better than the hold-out accuracy. This is expected with the cross-validation being evaluated across multiple folds and averaging out the results. But a significant difference in accuracy which can point out the instability in cross-validation accuracy is in the kernel based support vector method. While the prediction accuracy using testing is 78.79%, we observe quite a significant jump to 84.62%accuracy in the cross-validation technique. We also note the fact that this being a multiclass classification problem requires the support vector approach to fit multiple two-class models using a 'one-vs-all' scheme. While other models, involve fitting a model which incorporates a multi-class prediction method, the kernel method comprises of multiple models to predict the classes and this along with averaging out the results could lead to an overfitting issue for this data application. Among the applied models for the prediction problem, we conclude that the best competitive prediction accuracy was obtained using a the random forest aggregation method. The choice of parameters for this setup proved to be critical as significant difference in accuracies was observed by tuning the parameters of the model. The base learner for the ensemble is the decision tree with maximum depth of tree parameter being 5. The best accuracy results was obtained in this setup and it gave a hold-out and cross-validation accuracy of 80.8% and 82.97% respectively. It is possible to deduce information about feature interactions from our best performing model, the random forest. We examine every tree in the ensemble setup and list out the top 5 combinations of pairs of features in the non-decreasing order of frequency of co-occurence in the decision paths. We observe that the feature pairs CSCE121-MATH251, PHYS208-MATH251, CSCE121-PHYS218, CSCE121-MATH152, PHYS208-CSCE121 are the most frequent co-occurring pairs of features. We can infer that a feature for instance CSCE121 which does not have the highest correlation coefficient with the target, in interaction or combination with other features makes a significant contribution to the prediction of the target outcomes thus indicative of its predictive power or influence. We also note that the models have been applied and evaluated for their performance with categorical variables instead of conventionally with numeric values and the kernel metric has also been applied to measure between categorical variables without re-encoding them to dummy numeric values based on the levels.

REFERENCES

[1] Niall Sclater, Alice Peasgood, Joel Mullan. *Learning Analytics in Higher Education*. April 2016 [Online] . Available: https://www.jisc.ac.uk/sites/default/files/learning-analytics-in-he-v2_0.pdf

[2] Demetriou, C. & Schmitz-Sciborski, A. (2011). *Integration, motivation, strengths and optimism: Retention theories past, present and future*. In R. Hayes (Ed.), Proceedings of the 7th National Symposium on Student Retention, 2011, Charleston. (pp. 300-312). Norman, OK: The University of Oklahoma.

[3] Marie Beinkowski, Mingyu Feng, Barbara Means. *Enhancing Teaching and Learning Through Educational Data Mining and Learning Analytics*, Center for Technology in Learning, October 2012

[4] R. Caruana and D. Freitag. *Greedy attribute selection in Machine Learning*, Proceedings of the Eleventh International Conference. Morgan Kaufmann, 1994.

[5] H. Almuallim and T. G. Dietterich. *Efficient algorithms for identifying relevant features*, In Proceedings of the Ninth Canadian Conference on Artificial Intelligence, pages 38âĂŞ45. Morgan Kaufmann, 1992.

[6] Jarecki, J., Meder, B., & Nelson, J. D. (2013). *The assumption of class-conditional independence in category learning*, In M. Knauff, M. Pauen, N. Sebanz, & I. Wachsmuth (Eds.), Proceedings of the 35th annual conference of the cognitive science society (pp. 2650âĂŞ2655). Austin, TX: Cognitive Science Society.

[7] Hall M. *Correlation-based feature selection for machine learning*, IPhD Thesis , 1999New ZealandDepartment of Computer Science, Waikato University.

[8] C. Donalek. *Supervised and Unsupervised Learning*, (April), 2011.

[9] Keogh, E. & Mueen. *A. Encyclopedia of Machine Learning*, (Springer, 2011).

[10] Kibriya, A.M., Frank, E., Pfahringer, B., Holmes, G. *Multinomial naive Bayes for text categorization revisited*,In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS, vol. 3339, pp. 488âĂŞ499. Springer, Heidelberg (2004)

[11] McCallum, A., Nigam, K.. *A Comparison of Event Models for Naive Bayes Text Classification*,In: AAAI/ICML-98 Workshop on Learning for Text Categorization. (1998) 41âĂŞ48

[12] A. Gelman. *Multilevel (hierarchical) modeling: What it can and cannot do*, Technometrics, 48:432âĂŞ435, 2006

[13] Rokach L, Maimon O (2008). *Data mining with decision trees: theory and applications*, World Scientific Publishing, Singapore

[14] Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification and regression trees*, Wadsworth International Group, Belmont, CA

[15] L. Rokach. *Ensemble-based classifiers*, rtif. Intell. Rev., vol. 33, pp. 1-39, 2010.

[16] Efron B, Tibshirani R (1993). *An introduction to the bootstrap*, rtif. Chapman and Hall, New York.

[17] Liaw A, Wiener M. *Classification and regression by randomForest*, Rnews 2002, 2: 18âĂŞ22.

[18] Svetnik V, Liaw A, Tong C, Wang T. *Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules*, Multiple Classier Systems, Fifth International Workshop, MCS 2004, Proceedings, 9âĂŞ11 June 2004, Cagliari, Italy. Lecture Notes in Computer Science, Springer 2004, 3077: 334âĂŞ343.

[19] R. Schapire. *The Boosting Approach to Machine Learning: An Overview*, Proc. MSRI Workshop Nonlinear Estimation and Classification, 2001.

[20] Schapire, R.E. *Explaining adaboost. In: Empirical Inference*, Festschrift in Honor of Vladimir N. Vapnik (2013)

[21] Zhu, J., Rosset, S., Zou, H., and Hastie, T. *Multi-class AdaBoost*, Technical Report 430, Department of Statistics, University of Michigan, 2006.

[22] Nam Nguyen and Yunsong Guo. *Metric Learning: A Support Vector Approach*, In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), pages 125âĂŞ136, 2008.

[23] *Support vector machine. In Wikipedia*,The Free Encyclopedia.

[24] E. Allwein, R. Schapire, Y. Singer. *Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers*, Machine Learning Research, pp. 113-141, 2000.

[25] Joachims, T. (1999). In B. SchÃűlkopf, C. Burges, & A. Smola (Eds.). *Advances in kernel methodsâĂŤsupport vector learning*, Cambridge: MIT Press.

[26] *Polynomial kernel*, In Wikipedia, The Free Encyclopedia.

[27] Hofmann, Martin. *Support Vector Machines – Kernels and the Kernel Trick*, Notes. 26 June 2006. Web. 7 Jan. 2013.

[28] Kohavi, R. (1995). *A study of cross-validation and bootstrap for accuracy estimation and model selection*, In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (pp. 1137âĂŞ1143). San Francisco, CA: Morgan Kaufmann

[29] Davis J, Goadrich M (2006). *The relationship between Precision-Recall and ROC curves*, In: Cohen W, Moore A, eds. Proceedings of the 23rd International Conference on Machine Learning, ACM Press, Pittsburgh, PA, pp 233âĂŞ240

[30] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. *Understanding variable importances in forests of randomized trees*, n Advances in Neural Information Processing Systems, pp. 431âĂŞ439, 2013.