

AMERICAN UNIVERSITY OF BEIRUT

STUDENT INTERVENTION SYSTEM USING
MACHINE LEARNING

by
MISSAK YESSAI BOYAJIAN

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
January 2019

AMERICAN UNIVERSITY OF BEIRUT

STUDENT INTERVENTION SYSTEM USING
MACHINE LEARNING

by
MISSAK YESSAI BOYAJIAN

Approved by:



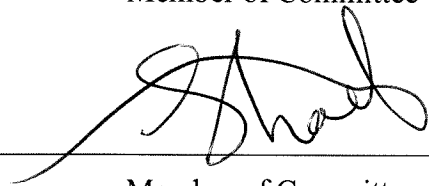
Dr. Fatima Abu Salem, Associate Professor
Computer Science

Advisor



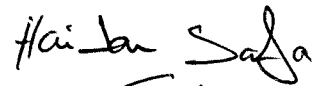
Dr. Mohammad Jaber, Assistant Professor
Computer Science

Member of Committee



Dr. Shady Elbassuoni, Assistant Professor
Computer Science

Member of Committee



Dr. Haidar Safa, Professor
Computer Science

Member of Committee

Date of thesis defense: January 2019

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Boyajian Missak Yessai
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after : **One ---- year from the date of submission of my thesis, dissertation, or project.**
Two ---- years from the date of submission of my thesis, dissertation, or project.
Three ~~---~~ years from the date of submission of my thesis, dissertation, or project.

Missak

03/05/2019

Signature

Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

ACKNOWLEDGEMENTS

This study would have never been possible without the continuous guidance of my advisor Dr. Fatima Abu Salem. She constantly provided direction and helped me to overcome any obstacle. I am very thankful for all her efforts.

I would like to thank my committee members, Dr. Mohammad Jaber, Dr. Shady Elbassuoni and Dr. Haidar Safa who gave me the opportunity and the necessary knowledge to complete the thesis.

Sincere thanks goes to my family and friends who always supported me during my best and worst times.

Last but not least, I would like to thank the faculty members of American University of Beirut for providing me with all the resources needed. Without them, I would not have been able to continue the thesis.

AN ABSTRACT OF THE THESIS OF

Missak Yessai Boyajian for Master of Science
Major: Computer Science

Title: Student Intervention System Using Machine Learning

In most universities, advisors guide students in their course selection, and warn those who might be at risk of being dismissed or placed on probation. The large number of students makes it difficult for universities to identify those at risk, as it would get very time consuming and inaccurate. Hence, there is a need for a system that can recognize these students at the end of each semester. In this work, we build a semester-based automated intervention system using machine learning that identifies students who are at risk and suggests upcoming courses, providing them with a more personalized approach. We applied supervised learning such as logistic regression, neural networks, and AdaBoost to predict three outcomes: 1) risk of dismissal, 2) risk of probation, and 3) time needed to graduate. Then we applied reinforcement learning (RL) using value iteration technique to create an optimal policy that will recommend courses to students with the goal of keeping them safe. We applied different evaluation methods, such as ROC curve and F-measure, to compare the performance of the supervised learning algorithms, and proposed a new approach to measure the effectiveness of the recommendation system. The dataset was provided by the American University of Beirut and contained a sample of 30,000 student records.

CONTENTS

ACKNOWLEDGEMENTS	v
ABSTRACT	vi
1 INTRODUCTION	2
1.1 Background and Related Works	2
1.2 Thesis Structure	6
2 EVALUATING MODEL PERFORMANCE	8
2.1 Measuring Performance for Regression	8
2.2 Measuring Performance for Classification	10
2.3 Measuring Performance Techniques	11
3 IMPROVING MODEL PERFORMANCE	15
3.1 Automatic Parameter Tuning	15
3.2 Understanding Ensembles	16
4 IMBALANCED DATA	18
5 DATA PROCESSING	21
5.1 Data Extraction	21
5.2 Normalization Methods	21
5.3 Data Exploration	23
6 MACHINE LEARNING - FIRST ATTEMPT	26
6.1 Performance Results	28
6.2 Normalization	32
6.3 Analysis	36
7 MACHINE LEARNING - SECOND ATTEMPT	37
7.1 Performance Results	39
7.2 Analysis	45

8	MACHINE LEARNING - BEST ATTEMPT	47
8.1	Performance Results	51
8.2	Analysis	57
8.3	Risk Estimates	63
8.3.1	Evaluating Risk Estimates	70
8.4	Feature Importance	76
9	APPLYING REINFORCEMENT LEARNING	79
9.1	Reinforcement vs Supervised Learning	79
9.2	Elements of RL	80
9.3	Markov Decision Process Characteristics	86
9.4	Policy Construction	87
9.4.1	Applying Value Iteration	88
9.5	Policy Evaluation	89
9.6	Handling New States	93
9.7	Test Sample	94
9.8	Policy Overview	96
9.8.1	Course Recommender	96
9.8.2	N-th Recommendation	99
9.9	Summary	100
10	CONCLUSIONS AND FUTURE WORK	103
10.1	Conclusion	103
10.2	Future Work	104
A	ABBREVIATIONS	105
	REFERENCES	107

CHAPTER 1

INTRODUCTION

Advisors are assigned to newly enrolled university students to assist them in registering their initial courses. They provide them with guidance in choosing courses and intervene when a critical situation such as a poor academic performance or probation are in sight. They usually speak from their experience and it is their job to assess a student's ability and recommend suitable courses based on performance. However, universities are composed of students with varying backgrounds and academics; this alone complicates the already challenging job of an advisor. Moreover, dealing with a large number of students is both tedious and time consuming. One has to track each student's performance and guide them, making sure they are on the correct path.

An automated student intervention system can play a critical role in aiding advisors and providing them with a more personalized approach. The system can significantly reduce the workload, allowing an advisor to focus more on crucial tasks. Having a system that can store academic history, monitor, and analyze a student's pattern will further enhance the efficiency and precision of an advisor. Coupled with an early risk prediction feature, it detects students that have a likelihood to perform poorly and deploys preemptive measures.

1.1 Background and Related Works

Several studies deal with predicting students' academic outcomes and course recommendations. Each uses different features, algorithms, evaluation methods,

and data.

Predictions

Sergi, Eloi, and Laura[1] applied machine learning techniques to predict two outcomes: 1) student dropout and 2) course grades for every student. For the first part, they applied and compared five algorithms: Logistic Regression, Gaussian Naive Bayes, Support Vector Machines, Random Forest and Adaptive Boosting. They produced best F1 scores of 82%, 76% and 61% for the degrees in computer science, law, and mathematics respectively. For the second part, they applied Collaborative Filtering Recommendation System, Linear Regression and Support Vector Regression, and achieved an MAE of 1.21, 1.32 and 1.34 respectively.

Everaldo et al.[2] used an incremental approach “to select and prioritize students who may be at risk of not graduating from high school on time.” They first developed predictive models that can identify students at risk, and then urgency score models that can rank students who are most likely to go off track. They mainly focused on schools with limited resources, and on students who belonged in the lower rank categories. Therefore, their goal was to “focus on the right students, at the right time, and with the right message.”

Nicolae-Bogdan Sara et al.[3] based their work on 72,598 Danish high-school pupils and applied machine learning techniques to predict dropouts. Random forest produced the best results, with an accuracy of 93.5% and an AUC of 0.965.

The major weakness in most of these works is that they perform the predictions once before the student enters a college or university, and do not take into account that they may improve or get worse over time. The same student who had very

good grades at school may perform much worse than another student who had average grades. There can be many reasons for this, such as personal issues, financial issues, and level of commitment to their majors, etc. Therefore, there is a need for a system that warns students after each semester.

Recommendations

Recommendation systems have been widely developed for various industries such as media streaming (YouTube[4], Netflix[5]), shopping websites (Amazon[6]), and online course services (edX[7]). For example, in 2006, Netflix announced a data mining competition[8] also known as ‘Netflix Prize’ which offered \$1 million award for the winner. Their focus was to enhance the movie recommendation system and the user experience with the goal of bringing more revenue into the company.

At academic level, many studies exist that deal with course recommendation strategies. Fábio Oliveira[9], for instance, used recommendation algorithms to help students who are pursuing their bachelor’s degree choose their master’s courses. The system used the knowledge of the grades for bachelor’s courses and predicted the grades for the master’s courses. The author used a dataset that contained the academic results of bachelor’s and master’s programs for the past twenty years.

Konstantin and Alexander[10] proposed a content-based method that recommends study materials to students who took a certain course. The model maps each course into its corresponding materials such as textbooks, online articles, or websites and then analyzes the student’s performance for each topic. Based on the analysis, the model will understand the knowledge gap and determine the required materials need for a student to close that gap. The authors worked with an online

university that offered a variety of courses. In the first semester, they worked with 692 students and split them into three groups. The first group received custom recommendations provided by the model after their first quiz. In the second group, all students received the same recommendations. The third group did not receive recommendations at all. At the end of the experiment, they found that all students liked the recommended materials, but that the model, as far as grades were concerned, worked best for mediocre students and was less effective for excellent and good students.

Sanjog Ray, Anuj Sharma[11] used a collaborative filtering approach to develop an elective course recommendation system. They proposed an approach that provides students with an accurate prediction of the grade they may attain if they choose a particular course, which will be helpful when they decide on elective courses. They performed the experiment on 255 students of 25 subjects, and were able to achieve an MAE of 0.35.

Most of these methods either recommend courses that the student has a preference for or is likely to perform well. They are suitable for online course providers, since the student is at leisure to choose any course at any time. The drawback with these methods appear when applying to students perusing a bachelor's degree because they are forced to register most of the required courses and have limited flexibility. They need to follow a certain path in order to graduate.

In this paper, we proposed a different kind of approach for course recommendations, which is based on reinforcement learning concepts rather than recommender systems, to help students pick their future courses. The system has knowledge of the students' past records, and based on that, it will suggest the best set of

courses a student should register for, in order to graduate safely without encountering probation or dismissal.

1.2 Thesis Structure

The rest of this thesis is structured as follows.

- Chapter 2: We describe some of the evaluating model performance methods such as F-measure, mean absolute error (MAE), and area under the curve (AUC) that we used to evaluate our predictive models.
- Chapter 3: We explain some of the evaluating model performance methods such as parameter tuning, bagging, and boosting.
- Chapter 4: We explain what is imbalanced dataset, how they occur, and describe some of the main techniques used to handle it.
- Chapter 5: Before beginning the machine learning process, it was important to understand the nature of the data. In this chapter, we describe our data extraction process and analyze the dataset received from the American University of Beirut.
- Chapters 6, 7, 8: We use machine learning methods, such as Logistic Regression, Random Forest, Naive Bayes, etc., to predict three outcomes: 1) dismissal, 2) probation, and 3) graduation time. Chapter 6 presents the initial features generated for the first attempt. In Chapters 7 and 8, we present the incremental machine learning progress that led to better performance.
- Chapter 9: We build a course recommendation system for the undergraduate students. We use reinforcement learning approaches to build a model/policy

that recommends set of courses at the end of each semester. The primary focus was on students at risk, but the policy can be applied to all.

- Chapter 10: We conclude our thesis and discuss future implementations.

CHAPTER 2

EVALUATING MODEL PERFORMANCE

This chapter will discuss some of the main methods[12] used to evaluate machine learning algorithms and their differences.

2.1 Measuring Performance for Regression

There are several metrics used to measure the performance of regression algorithms.

- Mean absolute error (MAE): It is one of the simple metrics used in machine learning and statistic which measures the average of the residuals of every instance.

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |y_t - y'_t|$$

where y_t = actual value and y'_t = predicted value.

MAE can be easily interpreted and treats all the data including the outliers with the same attention. However, it is not much useful for applications that tend to give more priority to the outliers and large errors.

- Mean absolute error (MSE): It is similar to MAE, but instead of using the absolute value of the difference, it squares them. We cannot directly compare MAE to MSE since the latter will produce a higher value most of the time.

However, one of the main differences between the two is that MSE will penalize the large errors unlike the MAE.

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n (y_t - y'_t)^2$$

- Root mean square error (RMSE): It is the most commonly used metric for regression tasks. Since MSE squares the residuals, the units will not match the actual output values and it will be more difficult to interpret the results. RMSE simply solves this issue by taking the square root of MSE.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - y'_t)^2}$$

- Mean absolute percentage error (MAPE): It is similar to MAE, but it converts all the error rates into percentages. The key advantage of MAPE is that it is easier to interpret the results since some researchers find percentages easier to conceptualize.

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{y_t - y'_t}{y_t} \right|$$

Some of the drawbacks of MAPE is that it will be undefined for output having values of 0. In addition, when the predicted value is lower than the actual one, it will have a lower result compared to the one having higher value with the same amount. For instance MAPE is lower for $y_t = 4$ and $y'_t = 2$ compared to $y_t = 4$ and $y'_t = 6$, even though both cases have the same residual.

In this paper, we present both RMSE and MAE, but use the latter to compare the performance of our regression algorithms. We wanted each residual to contribute with the same proportion to the final error and were not interested in penalizing the large errors more than the smaller ones.

2.2 Measuring Performance for Classification

The simplest way to measure the performance for classification algorithms is to calculate the accuracy. It simply divides the number of correct predictions made by the total number of predictions. Higher accuracy implies better learning. However, for some cases, this method is not applicable. For instance, when a dataset contains 0.01% of people having cancer, the algorithm that predicts only *no-cancer* will have a 99.99% accuracy which is considerably high. To handle this issue, there exists measurements other than accuracy such as Precision, Recall, or F-measure that focus more on the class distribution.

Confusion Matrices

The confusion matrices categorize the predictions and the actual data for each class and represent them in a table. For example, if we have two classes, then we can have a 2×2 matrix.

The relationship between positive class and negative class predictions can be represented as a 2×2 confusion matrix(Figure 2.1) that tabulates whether predictions fall into one of four categories:

- True Positive (TP): Correctly classified as the class of interest

		<u>Predicted Class</u>	
		<u>P</u>	<u>N</u>
<u>Actual Class</u>	<u>P</u>	True Positives(TP)	False Negatives(FN)
	<u>N</u>	False Positives(FP)	True Negatives(TN)

Figure 2.1: Confusion Matrix Example

- True Negative (TN): Correctly classified as not the class of interest
- False Positive (FP): Incorrectly classified as the class of interest
- False Negative (FN): Incorrectly classified as not the class of interest

The proportions are easily calculated from the confusion matrix.

2.3 Measuring Performance Techniques

We can formalize accuracy and error rate as:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{error rate} = 1 - \text{accuracy}$$

Sensitivity/Recall

The sensitivity/recall also known as the true positive rate, divides[13] the number of correctly classified positive instances with total number of positive instances.

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Specificity/Precision

The specificity/precision of a model is the same as the sensitivity, but it measures the proportion of negative examples that were correctly classified.

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

The F-measure

F-measure, also known as F1 score, represents a balanced measure between precision and recall and combines them using the harmonic mean.

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Recall} + \text{Precision}}$$

It ranges from 0 (poor) to 1 (excellent).

ROC Curves

The ROC (Receiver Operating Characteristic) curves plots the true positive rate (sensitivity) against the false positive rate ($1 - \text{specificity}$) at various threshold. A good threshold would be the one with highest sensitivity and lowest false positive rate (FPR).

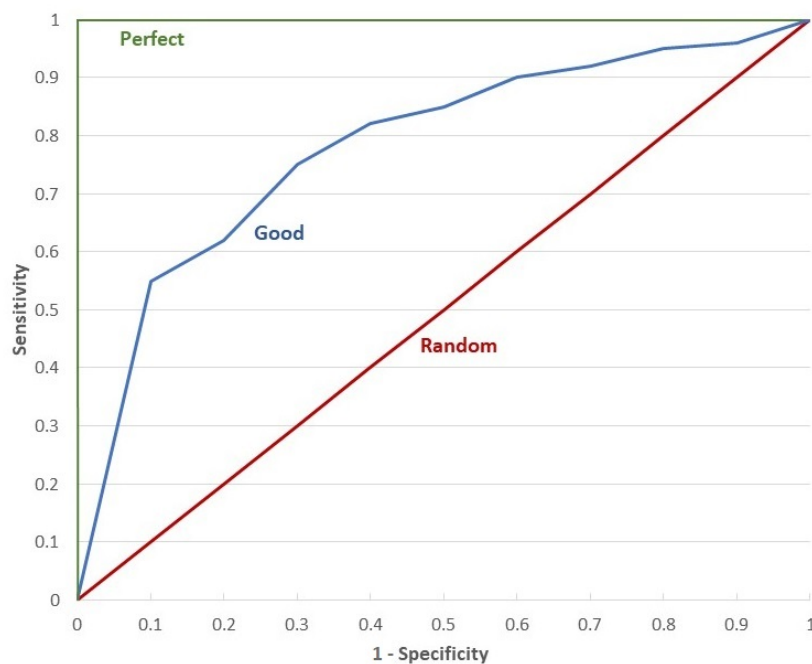


Figure 2.2: ROC Curve Example

The red line (Figure 2.2) represents a poor classifier that cannot differentiate between TPR and FPR and produces almost the same values for all the thresholds. The green line represents a perfect classifier which has 100% TPR and 0% FPR. However, such results unlikely occur in real scenarios. Most of the classifiers will produce a curve closer to the blue one.

To measure the performance of the curve, area under the ROC (AUC) is used. It ranges from 0.5 to 1. Higher AUC implies better performance as described below.

- 0.9–1.0 (perfect)
- 0.8–0.9 (good)
- 0.7–0.8 (acceptable)
- 0.6–0.7 (poor)
- 0.5–0.6 (no discrimination)

CHAPTER 3

IMPROVING MODEL PERFORMANCE

This section introduces set of techniques for improving the performance of machine learners such as parameter tuning, combining models into groups, and some cutting-edge techniques for getting the maximum level of performance. Each technique is explained briefly below.

3.1 Automatic Parameter Tuning

Every machine learning model contains many adjustable parameters. It is important to compare all possible parameter values to find the best combination. However, doing this task manually can be time consuming and almost impossible for models with many parameters. To solve this issue, there are many tools for automatic parameter tuning that find the optimal solution. However, certain questions need to be taken into consideration before tuning.

1. What type of machine learning model should be trained on the data?
2. What are the main parameters that need to be tuned?
3. Which evaluation method should be used?

Depending on the nature of the data, half of the models could be eliminated because the task may be a classification or a regression. As for the remaining models, it is better to test some of the best algorithms available and evaluate their results.

For the second question, parameters depend on the machine learning model. It is better to have a broad understanding of the importance of each parameter in order to specify which ones should be used in the tuning process.

The third question also depends on the nature of the data. The comparisons can be based on accuracy, sensitivity, specificity, or ROC curve. In some cases, the method can even be customized to a certain application.

3.2 Understanding Ensembles

Ensembles combine several single models to produce a better model. Ensembles offer a number of advantages over single models.

- Better generalization, less chance of overfitting
- Use of distributed architecture to parallelize an ensemble to improve performance
- Ensembles perform better as the data contain more complex real world scenarios

Bagging

Bagging is one of the first ensemble methods that was widely used in the industry. First, it generates a number of training datasets by sampling repetitively with replacement. These datasets are then used to generate a set of learning algorithms. Finally, The models' predictions are combined using voting (for classification) or averaging (for numeric prediction). Bagging performs well when used with sensitive learners such as decision trees.

Boosting

Boosting is another popular ensemble-based method. It takes a number of classifiers, each with an error rate less than 50% and boosts them until they attain the performance of strong learners. Beginning from an unweighted dataset, the first classifier attempts to model the outcome. Examples that the classifier predicted correctly will be less likely to appear in the training dataset for the following classifier; conversely, the difficult examples will appear more frequently. As additional rounds of weak learners are added, they are trained on data with successively more difficult examples. The process continues until the desired overall error is reached or performance no longer improves. At that point, each classifier's vote is weighted according to its accuracy on the training data on which it was built.

Random Forests

Random Forest is another type of ensemble that focuses only on decision trees. It creates an ensemble of many trees that each consider random features at each split and uses voting to combine their decisions. The algorithm performs well on most problems and can handle noisy data, but it is very difficult to interpret or visualize the model itself.

CHAPTER 4

IMBALANCED DATA

Imbalanced data occur when the class distribution within a dataset is significantly imbalanced. For instance, in this paper, only $\sim 9\%$ of the students were dismissed, and it was crucial to detect them. Machine learning algorithms usually try to achieve the best accuracy without taking into account the distribution of classes. Consequently, they will treat the minority classes as noise and ignore them. There are various ways to solve class imbalance problems.

Random Undersampling

Undersampling randomly eliminates most of the majority classes until both classes balance out. This will reduce the storage, which leads to a better execution time, but it will discard potentially useful information. This method is not used widely in the market.

Random Oversampling

Oversampling randomly replicates the minority classes until both classes balance out. This method outperforms undersampling because there is no information loss, but on the other hand, it may cause overfitting.

Synthetic Minority Oversampling Technique

This method was introduced by Nitesh V. et al.[14] in 2002 in order to overcome overfitting. It is nearly similar to the Random Oversampling technique, but instead of replicating the minority instances, it creates new synthetic ones. It works by introducing synthetic examples by selecting and merging features of the k -nearest neighbors of minority classes.

Bagging Based

Bagging is used to overcome overfitting. Bagging generates n different training samples with replacements. Each sample is trained with different machine learning algorithms, and at the end, the predictions are aggregated. This works well if all algorithms produce good results. If one of the algorithms produces poor results, it may affect the overall performance.

AdaBoost(Adaptive Boosting)

AdaBoost was created by Yoav Freund and Robert E. Schapire[15] and is based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules. AdaBoost initially gives equal weight to all training records. At each round, it runs the machine learning algorithms, increases the weight of the incorrectly classified records, and decreases the correct ones. This method is used to focus more on the incorrect data. The loop ends when the error measure is very low. However, we need an algorithm that produces an accuracy of above 50%; otherwise, we could get a better result with a random

prediction. On the other hand, one of the disadvantages of using AdaBoost is that it is very sensitive to noise.

CHAPTER 5

DATA PROCESSING

The data were provided from the American University of Beirut; they contained students' academic school records and their university courses and grades. We were only interested in the students who were majoring in computer science and filtered out the rest.

5.1 Data Extraction

First, the original excel file containing the students' records were imported into the database. Then the database was read into the program. Finally, we queried only the students who have entered university as CMPS major, graduated as CMPS major, or have taken CMPS 200 and CMPS 212 respectively.

For the summer courses, the grades were accumulated to the next fall or to the previous spring term if the student had taken fewer than 12 credits in that fall/spring term.

5.2 Normalization Methods

Normalization is a data preprocessing technique that rescales the features to a common range. Since some of the classification models use distance functions to calculate the difference between two points, they will often give more importance to the features that have larger values. For instance, SAT Math scores of students have maximum value of 800 while their university major averages have maximum

value of 100. Hence, we ran the algorithms on both normalized and non-normalized data.

Types of Normalization

There are many types of normalization. In this paper, we use the min-max method.

Min-max normalization

It is one of the simplest method which scales the numerical values of the features into values between 0 and 1 or -1 and 1 .

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where x is an original value and x' is the normalized value.

Mean normalization

It is close to the min-max implementation, but, instead of subtracting $\min(x)$ from x in the numerator, we subtract the $\text{average}(x)$.

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

5.3 Data Exploration

To have a more discrete understanding of the nature of the data, we executed some queries to get results such as:

1. GPA distribution of graduated students(Table 5.1).
2. Distribution of dismissed students(Table 5.2).
3. Distribution of students being on probation(Figure 5.1).
4. Distribution of students' residency period(Figure 5.2).

The results are shown in the following tables/figures.

GPA	Count
≥ 90	11
85–90	67
80–85	208
75–80	321
70–75	316
65–70	23

Table 5.1: GPA Distribution

Dismissed	Count
Yes	98
No	1010

Table 5.2: Dismissal Distribution

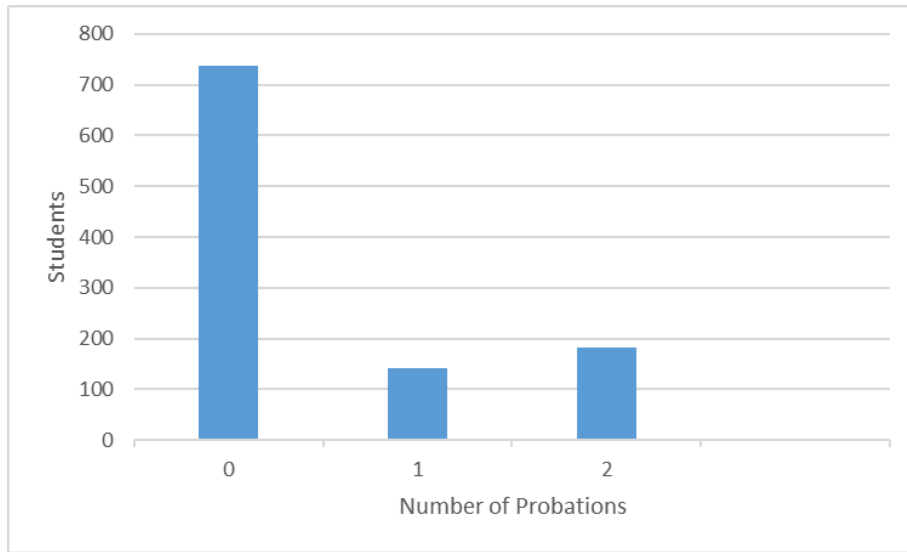


Figure 5.1: Probation Distribution

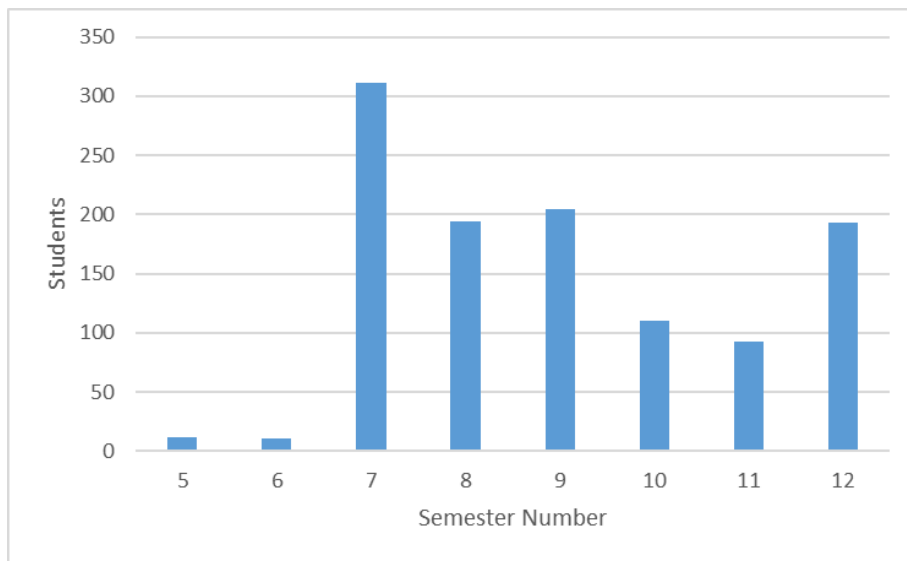


Figure 5.2: Graduation Time Distribution

We noticed that most of the students had a GPA between 75 and 80 (Table 5.1) and graduated after seven semesters (Figure 5.2). In addition to that, Table 5.2 and Figure 5.1 showed that there was an imbalanced distribution of classes.

For instance, only $\sim 9\%$ of the students were dismissed in the whole dataset and it was very crucial to detect them.

CHAPTER 6

MACHINE LEARNING - FIRST ATTEMPT

This chapter represents the initial process of feature creation and analysis. Chapters 7 and 8 will show some of the progress we made in terms of machine learning, feature creation, etc. that produced better performance.

Each algorithm was applied to students' second, third, and fourth semester. For each semester, the algorithms were applied to predict three outcomes: probation, dismissal, and graduation time(number of semesters needed to graduate).

For the first step, the following features were produced.

- **StudenAvg1**: 11th Grade School Average
- **StudenAvgYear2**: 12th Grade School Average
- **SATVerbal**: SAT Verbal Grade
- **SATMath**: SAT Math Grade
- **MajorAverage (+TotalMajorAverage)**: Average Grade of the Major Courses
- **ElectiveAverage (+TotalElectiveAverage)**: Average Grade of the Elective Courses
- **MajorCourseTaken (+TotalMajorCourseTaken)**: Total Major Courses Taken

- **ElectiveCourseTaken (+TotalElectiveCourseTaken):** Total Elective Courses Taken
- **MajorGrades (+TotalMajorGrades):** Total Major Grades
- **ElectiveGrades (+TotalElectiveGrades):** Total Elective Grades
- **MajorCredits (+TotalMajorCredits):** Total Major Credits
- **MajorPassed (+TotalMajorPassed):** Total Number of Major Courses Passed
- **ElectivePassed (+TotalElectivePassed):** Total Number of Elective Courses Passed
- **MajorFailed (+TotalMajorFailed):** Total Number of Major Courses Failed
- **ElectiveFailed (+TotalElectiveFailed):** Total Number of Elective Courses Failed
- **MajorWithdrawal (+TotalMajorWithdrawal):** Total Number of Major Courses Withdrawn
- **ElectiveWithdrawal (+TotalElectiveWithdrawal):** Total Number of Elective Courses Withdrawn
- **SemesterNumber:** The Current Number of Semester the Student is Enrolled
- **Probation:** Student Received Probation or Not - Current Semester

- **ProbationConsecutive:** The Number of Consecutive Probation Received at the end of the Term
- **ProbationNumber:** Total Number of Probation Received at the end of the Term
- **EventualProbation:** Student will Receive Probation or Not
- **Dismissal:** Student was Dismissed or Not

6.1 Performance Results

The test data contained 309 students in which 38 of them were dismissed. The following outcomes were produced for each semester.

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

A = Accuracy

F1 = F-measure

For instance, TP are those students who were dismissed, and the algorithm was able to predict it correctly. Similarly, FP are those students who were not dismissed, but the algorithm predicted it as dismissed.

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	12	260	11	26	0.88	0.39
Random Forest	7	270	1	31	0.89	0.30
Naive Bayes	24	238	33	14	0.84	0.50
Neural Network	16	259	12	22	88.99	0.48
AdaboostM1	18	259	12	20	89.64	0.43

Table 6.1: Semester 2 - Prediction: Dismissal

Naive Bayes produced the best performance in terms of predicting dismissal of the students. It produced the highest F-measure by having a value of 0.50 (Table 6.1). Out of 38, it was able to predict 24 correctly. The concern lies on the 14 students that the algorithm did not detect.

Out of the 14 :

- 6 were dismissed after 8 terms
- 1 majorless student dismissed after 11 terms
- 2 dismissed after 10 terms
- 2 dismissed after 12 terms but still graduated
- 2 dismissed but graduated with Business Degree
- 1 entered ENV, changed to CMPS, got dismissed but still graduated

The goal was to increase the detection of the dismissed students by maintaining a good F-measure.

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	30	185	16	77	0.69	0.39
Random Forest	31	185	16	76	0.70	0.34
Naive Bayes	41	156	45	66	0.63	0.42
Neural Network	44	147	44	73	0.62	0.43
AdaboostM1	24	188	13	83	0.68	0.33

Table 6.2: Semester 2 - Prediction: Probation

Algorithm	CRL	MAE	RMSE
M5P	0.689	1.1972	1.4529
Linear Regression	0.6858	1.2136	1.4556
Simple Linear Regression	0.6591	1.4219	1.6148
Random Forest	0.6877	1.1925	1.4593

Table 6.3: Semester 2 - Prediction: Graduation Time

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	17	257	14	21	88.67	0.49
Random Forest	10	255	16	28	85.76	0.31
Naive Bayes	25	237	34	13	84.79	0.51
Neural Network	11	265	6	27	89.32	0.40
AdaboostM1	10	266	5	28	89.32	0.37

Table 6.4: Semester 3 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	32	193	15	68	73.05	0.43
Random Forest	37	186	22	63	72.22	0.45
Naive Bayes	40	175	33	60	69.81	0.46
Neural Network	30	191	17	70	69.48	0.43
Naive Bayes	35	179	29	65	69.81	0.46

Table 6.5: Semester 3 - Prediction: Probation

Algorithm	CRL	MAE	RMSE
M5P	0.7264	1.0565	1.3583
Linear Regression	0.7251	1.0635	1.3609
Simple Linear Regression	0.6329	1.3514	1.5916
Random Forest	0.7494	1.0335	1.3179

Table 6.6: Semester 3 - Prediction: Graduation Time

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	17	258	13	21	88.99	0.50
Random Forest	4	271	0	34	88.99	0.11
Naive Bayes	27	240	31	11	86.41	0.56
Neural Network	11	261	10	27	88.02	0.37
AdaboostM1	13	265	6	25	89.96	0.45

Table 6.7: Semester 4 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	39	186	36	48	72.82	0.48
Random Forest	32	188	34	55	71.20	0.42
Naive Bayes	50	148	74	37	64.08	0.47
Neural Network	41	173	49	46	69.26	0.46
AdaboostM1	32	182	40	55	69.26	0.40

Table 6.8: Semester 4 - Prediction: Probation

Algorithm	CRL	MAE	RMSE
M5P	0.7559	0.9288	1.3252
Linear Regression	0.7462	0.9843	1.4162
Simple Linear Regression	0.6365	1.2726	1.5362
Random Forest	0.785	0.9103	1.3278

Table 6.9: Semester 4 - Prediction: Graduation Time

6.2 Normalization

Normalization was required in our application, since students had many numerical features such as their previous school grades, SAT scores, etc. Some of the features ranged from 0 to 100 (Major Average) and others from 0 to 800 (SAT).

We used the min-max normalization method that scaled the numerical fields in the range of 0 to 1. Consequently, we applied machine learning to the normalized data. The results are shown in the following tables.

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	22	243	28	16	85.76	0.50
Random Forest	10	266	5	28	89.32	0.37
Naive Bayes	27	236	35	11	85.11	0.54
Neural Network	16	258	13	22	88.67	0.48
AdaboostM1	14	261	10	24	88.99	0.45

Table 6.10: Semester 2 - Prediction: Dismissal - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	30	175	26	77	66.56	0.37
Random Forest	41	118	73	46	57.19	0.41
Naive Bayes	39	160	41	68	64.64	0.42
Neural Network	40	165	36	67	66.56	0.44
AdaboostM1	47	130	71	60	57.47	0.42

Table 6.11: Semester 2 - Prediction: Probation - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.689	1.1952	1.4529
Linear Regression	0.6858	1.2006	1.4556
Simple Linear Regression	0.6591	1.4219	1.6148
Random Forest	0.6877	1.1825	1.4593

Table 6.12: Semester 2 - Prediction: Graduation Time - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	17	257	14	21	88.67	0.49
Random Forest	3	269	2	35	88.02	0.13
Naive Bayes	27	237	34	11	85.44	0.54
Neural Network	11	265	6	27	89.32	0.40
AdaboostM1	10	266	5	28	89.32	0.37

Table 6.13: Semester 3 - Prediction: Dismissal - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	32	190	19	68	71.84	0.42
Random Forest	32	189	20	68	71.52	0.42
Naive Bayes	42	166	43	58	67.31	0.45
Neural Network	38	181	28	62	70.87	0.46
AdaboostM1	21	196	13	79	70.08	0.46

Table 6.14: Semester 3 - Prediction: Probation - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.689	1.1952	1.4529
Linear Regression	0.6858	1.2006	1.4556
Simple Linear Regression	0.6591	1.4219	1.6148
Random Forest	0.6877	1.1825	1.4593

Table 6.15: Semester 3 - Prediction: Graduation Time - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	10	260	11	28	87.37	0.33
Random Forest	6	271	0	32	89.64	0.28
Naive Bayes	21	240	31	17	84.47	0.46
Neural Network	12	263	8	26	88.99	0.41
AdaboostM1	12	266	5	26	89.96	0.43

Table 6.16: Semester 4 - Prediction: Dismissal - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	34	201	21	53	76.05	0.47
Random Forest	33	196	26	54	74.11	0.45
Naive Bayes	45	155	67	34	66.45	0.47
Neural Network	12	243	28	26	82.52	0.31
AdaboostM1	39	178	44	48	70.23	0.46

Table 6.17: Semester 4 - Prediction: Probation - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.7791	0.9356	1.2424
Linear Regression	0.7281	1.028	1.3558
Simple Linear Regression	0.7653	1.0301	1.3185
M5P	0.7764	0.9109	1.2214

Table 6.18: Semester 4 - Prediction: Graduation Time - Normalized

6.3 Analysis

We notice that the algorithms produced better performance after each term, since they were able to learn more about students as they take more courses. For instance, Naive Bayes had the best F-measure for predicting *Dismissal* having the value increase from 0.54 (Table 6.1) to 0.57 (Table 6.7). Another interesting point to notice is that, for some cases, normalization produced better results. For instance, Random Forest which had MAE of 1.185 (Table 6.12) for *Semester 2 - Normalized* produced slightly better result compared to the *Semester 2 - Non-normalized* version (Table 6.3) which had MAE of 1.1925.

CHAPTER 7

MACHINE LEARNING - SECOND ATTEMPT

In the previous chapter, the courses were split into *electives* and *majors*. In this chapter, we divide them into six categories:

1. Major-Programming: CMPS 200, CMPS 212, CMPS 255, CMPS 258, and MATH 218.
2. Major-Theory: CMPS/MATH 211, CMPS 256, CMPS 257, MATH 201, and STAT 230.
3. Major-Systems: CMPS 253, CMPS 272, and CMPS 277.
4. Elective-Easy: CMPS 230, CMPS 278, CMPS 284, CMPS 289, and CMPS 299.
5. Elective-Challenging: CMPS/MATH 251, CMPS 274, CMPS 281, CMPS 285, CMPS 286, CMPS 287, CMPS 297, and any MATH course other than 218 and 201 or any STAT course other than 230.
6. Undefined: Rest of the Courses.

We generated only three features for each category:

1. Average
2. Total Passes
3. Total Failed

Feature Name	Description
SATVerbal	SAT Verbal Grade Over 800
SATMathematics	SAT Mathematics Grade Over 800
StudAvg	11th Grade Average
StudAvgYr2	12th Grade Average
MajorProgAverage	Cumulative Major-Programming Average
MajorProgPassed	Total Number of Major-Programming Courses Passed
MajorProgFailed	Total Number of Major-Programming Courses Failed
MajorTheoryAverage	Cumulative Major-Theory Average
MajorTheoryPassed	Total Number of Major-Theory Courses Passed
MajorTheoryFailed	Total Number of Major-Theory Courses Failed
MajorSystemsAverage	Cumulative Major-Theory Average
MajorSystemsPassed	Total Number of Major-Systems Courses Passed
MajorSystemsFailed	Total Number of Major-Systems Courses Failed
ElectiveEasyAverage	Cumulative Elective-Easy Average
ElectiveEasyPassed	Total Number of Elective-Easy Courses Passed
ElectiveEasyFailed	Total Number of Elective-Easy Courses Failed
ElectiveChallenging	Cumulative Elective-Challenging Average
ElectiveChallenging	Total Number of Elective-Challenging Courses Passed
ElectiveChallenging	Total Number of Elective-Challenging Courses Failed
UndefinedAverage	Cumulative Undefined Average
UndefinedPassed	Total Number of Undefined Courses Passed
UndefinedFailed	Total Number of Undefined Courses Failed
ProbationNumber	Total Number of Probation Received
ProbationConsecutive	Consecutive Number of Probation Received
Graduation Time	Time Needed to Graduate
Eventual Probation	Student Received Probation or Not
Dismissed	Student was Dismissed or Not

Table 7.1: List of Features

7.1 Performance Results

Similar to Chapter 6, we ran machine learning algorithms on the new dataset and analyzed the performance.

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	10	260	10	28	87.66	0.34
Random Forest	3	270	0	35	88.63	0.15
Naive Bayes	22	243	27	16	86.03	0.51
Neural Network	7	265	5	31	88.31	0.27
AdaboostM1	17	261	9	21	90.25	0.53

Table 7.2: Semester 2 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	11	260	10	27	87.98	0.37
Random Forest	3	270	0	35	88.63	0.15
Naive Bayes	25	238	32	13	85.38	0.53
Neural Network	7	265	5	31	88.31	0.27
AdaboostM1	17	261	9	21	90.25	0.53

Table 7.3: Semester 2 - Prediction: Dismissal - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.6908	1.1528	1.4355
Linear Regression	0.7056	1.1269	1.4039
Simple Linear Regression	0.621	1.4022	1.6147
Random Forest	0.7041	1.1389	1.4186

Table 7.4: Semester 2 - Prediction: Graduation Time

Algorithm	CRL	MAE	RMSE
M5P	0.6980	1.1980	1.4519
Linear Regression	0.6959	1.1826	1.4439
Simple Linear Regression	0.6490	1.2712	1.5845
Random Forest	0.7181	1.2292	1.4068

Table 7.5: Semester 2 - Prediction: Graduation Time - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	33	182	19	73	69.13	0.41
Random Forest	34	167	34	72	65.47	0.39
Naive Bayes	38	170	31	68	67.75	0.43
Neural Network	39	159	42	67	64.50	0.42
AdaboostM1	35	161	40	71	63.84	0.39

Table 7.6: Semester 2 - Prediction: Probation

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	38	155	46	68	62.87	0.40
Random Forest	38	156	45	68	63.19	0.40
Naive Bayes	42	151	50	64	62.87	0.42
Neural Network	16	152	49	80	56.57	0.20
AdaboostM1	47	138	63	59	60.26	0.43

Table 7.7: Semester 2 - Prediction: Probation - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	11	256	14	27	86.68	0.35
Random Forest	6	268	2	32	88.96	0.26
Naive Bayes	28	232	38	12	84.42	0.54
Neural Network	5	265	5	33	87.66	0.21
AdaboostM1	10	264	6	28	88.96	0.37

Table 7.8: Semester 3 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	14	249	21	24	85.38	0.45
Random Forest	6	268	2	32	88.96	0.26
Naive Bayes	26	226	44	12	81.82	0.48
Neural Network	6	265	5	32	87.66	0.25
AdaboostM1	19	254	16	19	88.63	0.52

Table 7.9: Semester 3 - Prediction: Dismissal - Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	29	189	19	71	70.77	0.39
Random Forest	28	178	30	52	71.53	0.41
Naive Bayes	45	164	44	55	67.86	0.47
Neural Network	35	174	34	65	67.85	0.41
AdaboostM1	35	179	29	65	69.48	0.42

Table 7.10: Semester 3 - Prediction: Probation

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	33	178	30	67	68.50	0.40
Random Forest	30	185	23	70	69.81	0.39
Naive Bayes	42	146	62	38	65.53	0.46
Neural Network	33	176	32	67	67.85	0.40
AdaboostM1	29	186	22	71	69.81	0.38

Table 7.11: Semester 3 - Prediction: Probation - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.7214	1.015	1.3674
Linear Regression	0.7306	0.9958	1.3522
Simple Linear Regression	0.5753	1.3764	1.6303
Random Forest	0.7502	0.9823	1.3117

Table 7.12: Semester 3 - Prediction: Graduation Time

Algorithm	CRL	MAE	RMSE
M5P	0.7186	1.0182	1.3773
Linear Regression	0.7166	1.028	1.3991
Simple Linear Regression	0.5753	1.3764	1.6303
Random Forest	0.7426	0.9901	1.3257

Table 7.13: Semester 3 - Prediction: Graduation Time Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	15	259	11	23	88.96	0.46
Random Forest	2	268	2	36	87.66	0.09
Naive Bayes	28	240	30	10	87.01	0.58
Neural Network	14	262	8	24	89.61	0.47
AdaboostM1	15	258	12	23	88.63	0.46

Table 7.14: Semester 4 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	0	269	1	38	87.33	0
Random Forest	6	268	2	32	88.96	0.26
Naive Bayes	38	32	238	0	22.72	0.24
Neural Network	0	270	0	38	87.66	0
AdaboostM1	7	266	4	31	88.63	0.284

Table 7.15: Semester 4 - Prediction: Dismissal - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.7223	0.9371	1.3836
Linear Regression	0.759	0.9072	1.2895
Simple Linear Regression	0.5748	1.3102	1.6169
Random Forest	0.7613	0.9199	1.258

Table 7.16: Semester 4 - Prediction: Graduation Time

Algorithm	CRL	MAE	RMSE
M5P	0.7517	0.9198	1.3016
Linear Regression	0.7474	0.9585	1.3118
Simple Linear Regression	0.5748	1.3102	1.6169
Random Forest	0.7757	0.8825	1.2499

Table 7.17: Semester 4 - Prediction: Graduation Time Normalized

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	10	208	13	77	70.77	0.18
Random Forest	19	212	9	68	75.00	0.32
Naive Bayes	42	171	50	45	69.16	0.47
Neural Network	28	190	31	59	70.77	0.38
AdaboostM1	32	204	17	55	76.62	0.47

Table 7.18: Semester 4 - Prediction: Probation

Algorithm	TP	TN	FP	FN	A	F1
Logistic Regression	18	187	34	69	66.55	0.26
Random Forest	22	214	7	65	76.62	0.38
Naive Bayes	36	167	54	51	65.91	0.41
Neural Network	26	190	31	61	70.12	0.36
AdaboostM1	23	196	25	64	71.10	0.34

Table 7.19: Semester 4 - Prediction: Probation - Normalized

7.2 Analysis

We compared the best **MAE** and **F-measure** with the previous attempt which we described in Chapter 6 and presented them in a table. The datasets are named as *outcome - semester number*(Table 7.20).

Dataset	ATT1	ATT2
Dismissal 2	0.54	0.53
Dismissal 3	0.54	0.54
Dismissal 4	0.56	0.58
Probation 2	0.44	0.43
Probation 3	0.46	0.47
Probation 4	0.41	0.47
Graduation Time 2	1.1825	1.1269
Graduation Time 3	1.003	0.99
Graduation Time 4	0.91	0.8825

Table 7.20: Performance Comparison of Attempt 1 and 2

In some of the datasets such as *Graduation Time 3*, the current attempt had better results, but, in other cases such as *Dismissal 2*, the previous attempt was slightly better.

CHAPTER 8

MACHINE LEARNING - BEST ATTEMPT

One of the ways to improve the obtained results was to make some changes concerning the features and generate different kind of features that might help the learning methods to produce better results. Hence, we decided to split the features related to the CMPS courses into three categories:

1. Major-Programming
2. Major-Theory
3. Electives

The Major-Programming category consisted of the following courses: CMPS 200, CMPS 212, CMPS 255, CMPS 258, MATH 218, CMPS 230, CMPS 278, CMPS 284, CMPS 289, CMPS 299, CMPS 253, CMPS 272, and CMPS 277.

The Major-Theory category consisted of the following courses: CMPS/MATH 211, CMPS 256, CMPS 257, MATH 201, STAT 230, CMPS/MATH 251, CMPS 274, CMPS 281, CMPS 285, CMPS 286, CMPS 287, CMPS 297, and any MATH course other than 218 and 201 or any STAT course other than 230.

Each category had seven features:

1. Average

2. Total Passed
3. Total Failed
4. Total Course Taken
5. Total Withdrawal
6. Total Grades
7. Total Credits

We added a numerical feature named as ‘FreshmanSemesters’ which defines the number of semesters a student has spent as freshman before entering his/her major.

We also added a feature that represents the rank of the school that the student came from, as such ranking can add much knowledge to the learning process. Students coming from a respectable private school usually have a better chance to succeed than students coming from an unpopular school. We categorized each of the schools into one of the following categories in the diagram below and ranked them.

School Category	Rank
Lebanon Private	7
North America Public	6
GCC Private	5
Lebanon Public	4
MENA Private	3
MENA Public	2
South Africa Private	1
Other	0

Table 8.1: School Category Ranks

Finally, the table below shows the total list of features generated for each dataset.

Feature Name	Description
SatVerbal	SAT Verbal Grade Over 800
SATMathematics	SAT Mathematics Grade Over 800
StudAvg	11th Grade Average
StudAvgYr2	12th Grade Average
SchoolRank	Rank of the school. Values between 0 and 8
FreshmanSemesters	Number of Freshman Semesters attended
MajorAverage	Cumulative Major Average
MajorPassed	Cumulative Number of Major Courses Passed
MajorFailed	Cumulative Number of Major Courses Failed
MajorCourseTaken	Cumulative Number of Major Courses Taken
MajorWithdrawal	Cumulative Number of Major Courses Withdrawn
MajorGrades	Cumulative Total Major Grades
MajorCredits	Cumulative Total Major Credits Taken
TheoryAverage	Cumulative Theory Average
TheoryPassed	Cumulative Number of Theory Courses Passed
TheoryFailed	Cumulative Number of Theory Courses Failed
TheoryCourseTaken	Cumulative Number of Theory Courses Taken
TheoryWithdrawal	Cumulative Number of Theory Courses Withdrawn
TheoryGrades	Cumulative Total Theory Grades
TheoryCredits	Cumulative Total Theory Credits Taken
ElectiveAverage	Cumulative Elective Average
ElectivePassed	Cumulative Number of Elective Courses Passed
ElectiveFailed	Cumulative Number of Elective Courses Failed
ElectiveCourseTaken	Cumulative Number of Elective Courses Taken
ElectiveWithdrawal	Cumulative Number of Elective Courses Withdrawn
ElectiveGrades	Cumulative Total Elective Grades
ElectiveCredits	Cumulative Total Elective Credits Taken
ProbationNumber	Total Number of Probation Received
ProbationConsecutive	Consecutive Number of Probation Received
Graduation Time	Time Needed to Graduate
Eventual Probation	Student Received Probation or Not
Dismissed	Student was Dismissed or Not

Table 8.2: Final list of features

8.1 Performance Results

The training was applied with k-fold cross-validation having $k = 10$ and on both normalized (min-max) and non-normalized data. For the dismissal prediction, data were resampled by 300% using the SMOTE algorithm, since we had an imbalanced distribution of outputs.

To find the optimal hyperparameters for the learning algorithms, we used the grid search approach.

- Logistic Regression: **Ridge:** 1.0E-8.
- Neural Network (Multilayer Perceptron): **Training Time(epoch):** 500, **Hidden Layers(epoch):** 1, **Learning rate:** 0.3, **Momentum:** 0.2
- Random Forest: **NumIteration:** 100, **NumExecutionSlots:** 100
- AdaBoostM1: **Classifier:** Logistic **NumExecutionSlots:** 100
- Linear Regreesion: **Ridge:** 1.0E-8

The results are shown below.

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	23	258	12	15	91.23	0.65	0.60
Random Forest	10	264	6	28	88.96	0.62	0.26
Naive Bayes	33	219	51	5	81.81	0.39	0.86
Neural Network	15	256	14	23	87.98	0.51	0.39
AdaboostM1 (Naive)	20	243	27	18	85.38	0.42	0.52

Table 8.3: Semester 2 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	22	258	12	16	90.90	0.65	0.58
Random Forest	15	257	13	23	88.31	0.53	0.39
Naive Bayes	25	239	31	13	85.71	0.44	0.66
Neural Network	14	242	28	23	83.44	0.35	0.40
AdaboostM1 (Naive)	22	251	19	16	88.63	0.53	0.58

Table 8.4: Semester 2 - Prediction: Dismissal - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.6742	1.2078	1.524
Linear Regression	0.6853	1.1826	1.4944
Simple Linear Regression	0.53	1.4762	1.7787
Random Forest	0.6958	1.164	1.48

Table 8.5: Semester 2 - Prediction: Graduation Semester

Algorithm	CRL	MAE	RMSE
M5P	0.6546	1.1834	1.5884
Linear Regression	0.6973	1.1481	1.5282
Simple Linear Regression	0.53	1.3185	1.7954
Random Forest	0.7031	1.1466	1.4764

Table 8.6: Semester 2 - Prediction: Graduation Semester - Normalized

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	24	185	24	75	67.85	0.5	0.24
Random Forest	27	190	19	72	70.45	0.58	0.27
Naive Bayes	43	155	54	56	64.28	0.44	0.43
Neural Network	29	176	33	70	66.55	0.46	0.29
AdaboostM1	43	155	54	56	64.28	0.44	0.43

Table 8.7: Semester 2 - Prediction: Probation

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	29	181	28	70	68.18	0.51	0.29
Random Forest	35	178	31	64	69.15	0.53	0.35
Naive Bayes	48	139	70	51	60.71	0.41	0.48
Neural Network	36	172	37	63	67.53	0.49	0.36
AdaboostM1	48	139	70	51	60.71	0.41	0.48

Table 8.8: Semester 2 - Prediction: Probation - Normalized

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	22	244	26	16	86.36	0.45	0.57
Random Forest	15	257	13	23	88.31	0.53	0.39
Naive Bayes	28	228	42	10	83.11	0.40	0.73
Neural Network	23	237	33	15	84.41	0.41	0.60
AdaboostM1(Naive)	20	235	35	18	82.79	0.36	0.52

Table 8.9: Semester 3 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	27	240	30	11	86.68	0.47	0.71
Random Forest	19	252	18	19	87.98	0.51	0.50
Naive Bayes	27	233	37	11	84.41	0.42	0.71
Neural Network	24	243	27	14	86.68	0.47	0.63
AdaboostM1(Naive)	19	236	34	19	82.79	0.24	0.50

Table 8.10: Semester 3 - Prediction: Dismissal - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.7184	1.0935	1.4214
Linear Regression	0.7184	1.0935	1.4214
Simple Linear Regression	0.5332	1.408	1.7485
Random Forest	0.7494	0.9989	1.3601

Table 8.11: Semester 3 - Prediction: Graduation Semester

Algorithm	CRL	MAE	RMSE
M5P	0.7122	1.1346	1.4952
Linear Regression	0.7122	1.1346	1.4952
Simple Linear Regression	0.5332	1.3552	1.8011
Random Forest	0.7258	1.0445	1.4256

Table 8.12: Semester 3 - Prediction: Graduation Semester - Normalized

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	17	213	9	69	74.67	0.65	0.20
Random Forest	17	212	10	69	74.35	0.63	0.20
Naive Bayes	63	128	94	23	62.01	0.40	0.73
Neural Network	30	196	26	56	73.37	0.53	0.34
AdaboostM1	35	175	47	51	68.18	0.42	0.40

Table 8.13: Semester 3 - Prediction: Probation

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	2	219	3	84	71.75	0.40	0.02
Random Forest	27	200	22	59	73.70	0.55	0.31
Naive Bayes	70	88	134	16	51.29	0.34	0.81
Neural Network	43	165	57	43	67.53	0.43	0.50
AdaboostM1	11	210	12	75	71.75	0.47	0.13

Table 8.14: Semester 3 - Prediction: Probation - Normalized

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	27	242	28	11	87.33	0.49	0.71
Random Forest	18	255	15	20	88.63	0.54	0.47
Naive Bayes	28	230	40	10	83.76	0.41	0.73
Neural Network	23	244	26	15	86.68	0.46	0.60
AdaboostM1	25	236	34	13	84.74	0.42	0.65

Table 8.15: Semester 4 - Prediction: Dismissal

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	26	244	26	12	87.66	0.50	0.68
Random Forest	20	253	17	18	88.63	0.54	0.52
Naive Bayes	32	227	43	6	84.09	0.42	0.84
Neural Network	21	241	29	17	85.06	0.42	0.55
AdaboostM1	23	235	35	15	83.76	0.39	0.60

Table 8.16: Semester 4 - Prediction: Dismissal - Normalized

Algorithm	CRL	MAE	RMSE
M5P	0.7641	0.9527	1.3184
Linear Regression	0.7643	0.9506	1.3179
Simple Linear Regression	0.5594	1.3286	1.7053
Random Forest	0.7834	0.8828	1.2759

Table 8.17: Semester 4 - Prediction: Graduation Semester

Algorithm	CRL	MAE	RMSE
M5P	0.7619	0.9335	1.3473
Linear Regression	0.7599	0.9342	1.3436
Simple Linear Regression	0.5594	1.2972	1.7356
Random Forest	0.7696	0.9016	1.3137

Table 8.18: Semester 4 - Prediction: Graduation Semester - Normalized

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	3	233	6	66	76.62	0.33	0.04
Random Forest	3	237	2	66	77.92	0.60	0.04
Naive Bayes	56	130	109	13	60.38	0.34	0.81
Neural Network	23	221	18	46	79.22	0.56	0.33
AdaboostM1	25	202	37	44	73.70	0.40	0.36

Table 8.19: Semester 4 - Prediction: Probation

Algorithm	TP	TN	FP	FN	A	P	RC
Logistic Regression	0	239	0	69	77.59	0	0
Random Forest	3	236	3	66	77.59	0.50	0.04
Naive Bayes	55	131	108	14	60.38	0.33	0.79
Neural Network	15	222	17	54	76.94	0.47	0.22
AdaboostM1	17	214	25	52	75	0.40	0.40

Table 8.20: Semester 4 - Prediction: Probation - Normalized

8.2 Analysis

We evaluated and analyzed the performance of various machine learning models for each outcome.

Graduation Time

Random Forest produced the best MAE for all semesters. It achieved the best MAE of 0.8828 as shown in Table 8.17. This implies that the algorithm was mistaking approximately by an average of one semester which is quite acceptable.

Mistake by one semester means that the algorithm expected the student to graduate in eight or six semesters whereas the actual number would have been seven.

Dismissed

To compare the learning models, we plot the ROC and the F-measure for each semester.

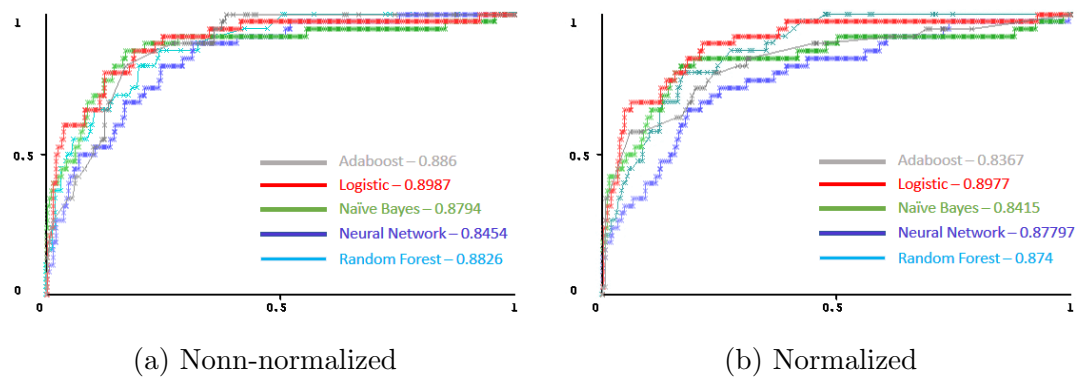


Figure 8.1: ROC - Semester 2

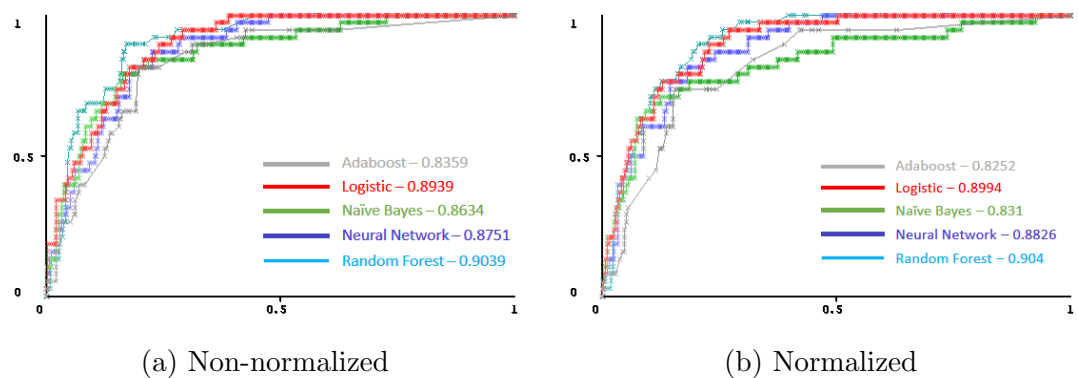


Figure 8.2: ROC - Semester 3

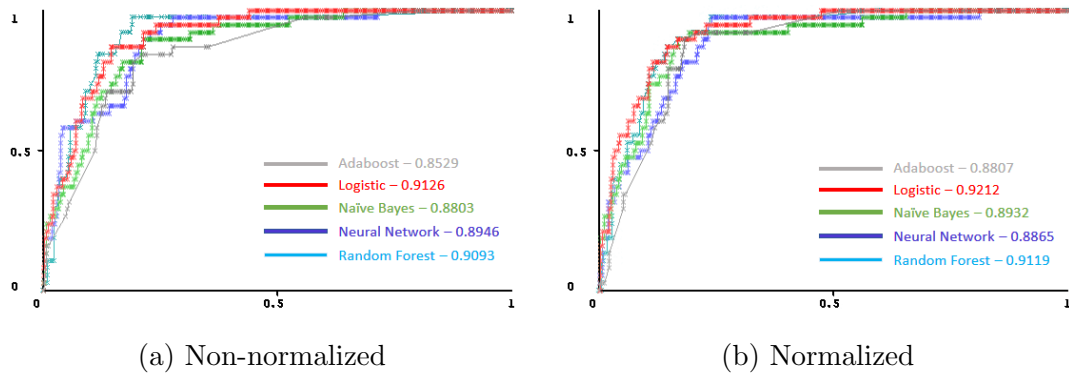


Figure 8.3: ROC - Semester 4

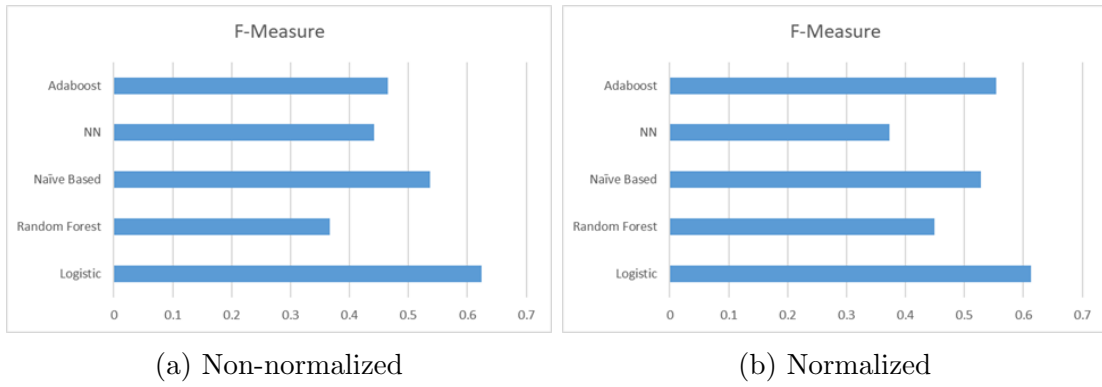


Figure 8.4: F-measure - Semester 2

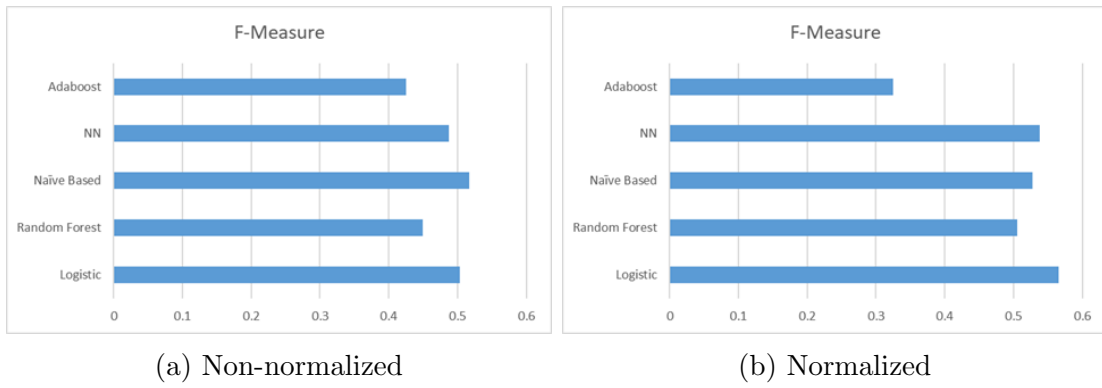


Figure 8.5: F-measure - Semester 3

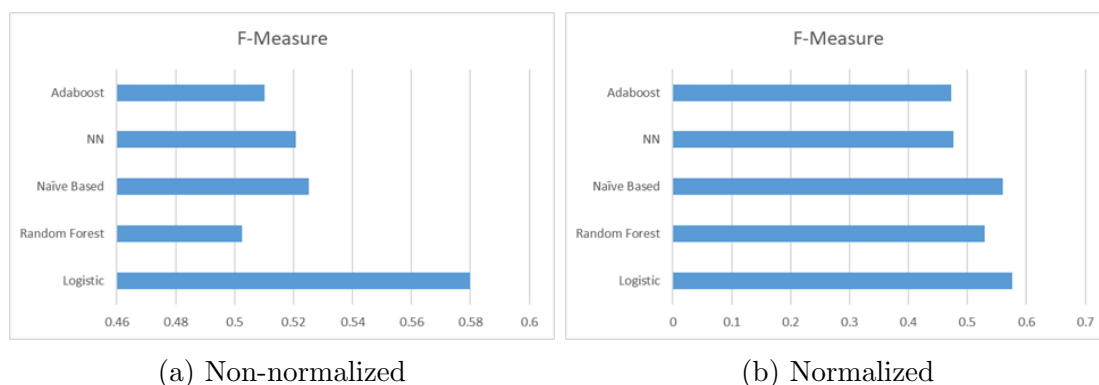


Figure 8.6: F-measure - Semester 4

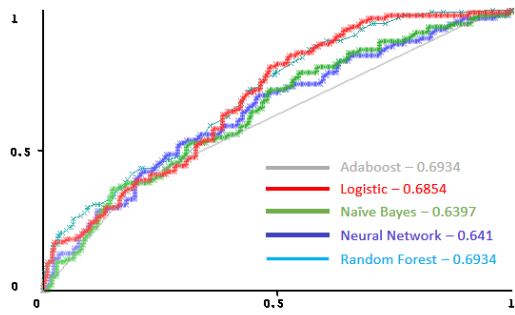
All the algorithms produced better AUC after each semester, since they were able to learn more about the student performance as the semester progresses. For semesters 2 and 4, Logistic Regression produced the best AUC, having values of 0.8987 (Figure 8.1a) and 0.9212 (Figure 8.3b) respectively. For semester 3, Random Forest gave the best result and performed slightly better than Logistic Regression. It provided an AUC of 0.904, as shown in Figure 8.2a.

For the F-measures, Logistic Regression gave the best results for all semesters. It had a value of 0.624 (Figure 8.4a) for semester 2, 0.565 (Figure 8.5b) for semester 3, and 0.58 for semester 4 (Figure 8.6a)

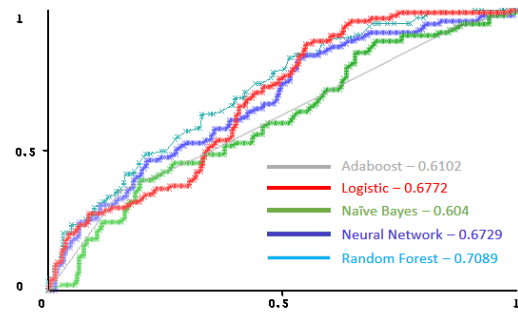
Overall, Logistic Regression performed best in terms of predicting the dismissal outcome.

Probation

Similar to dismissal, we plot the ROC curve and the F-measure for each semester.

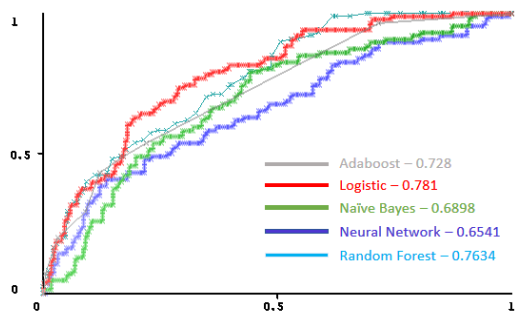


(a) Non-normalized

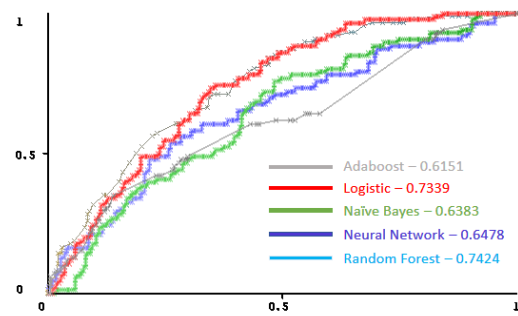


(b) Normalized

Figure 8.7: ROC - Semester 2

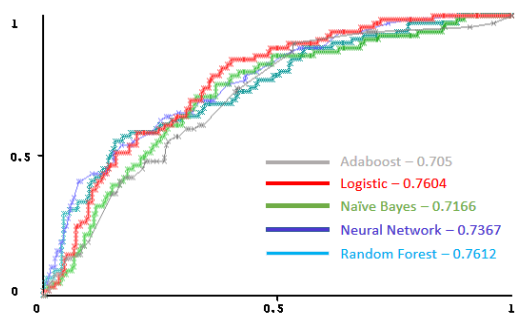


(a) Non-normalized

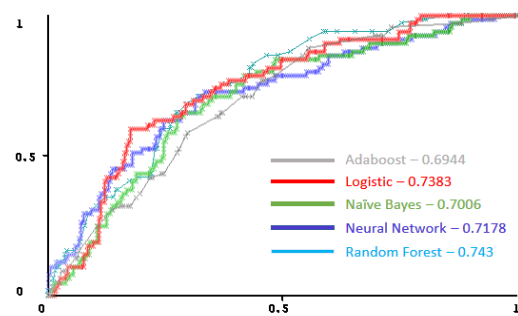


(b) Normalized

Figure 8.8: ROC - Semester 3

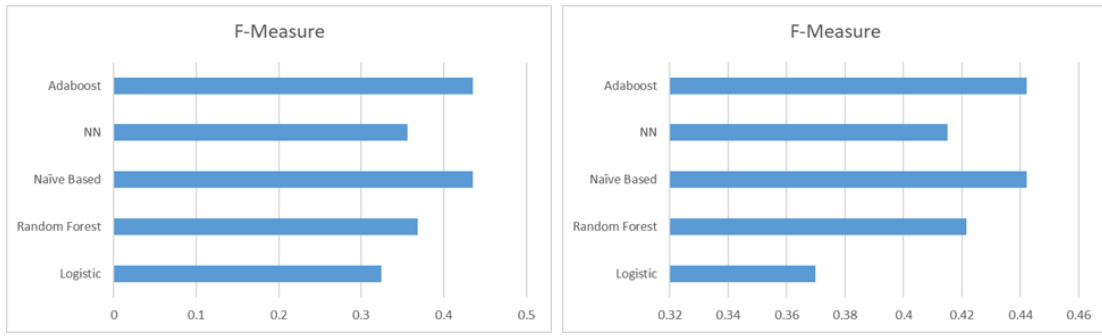


(a) Non-normalized



(b) Normalized

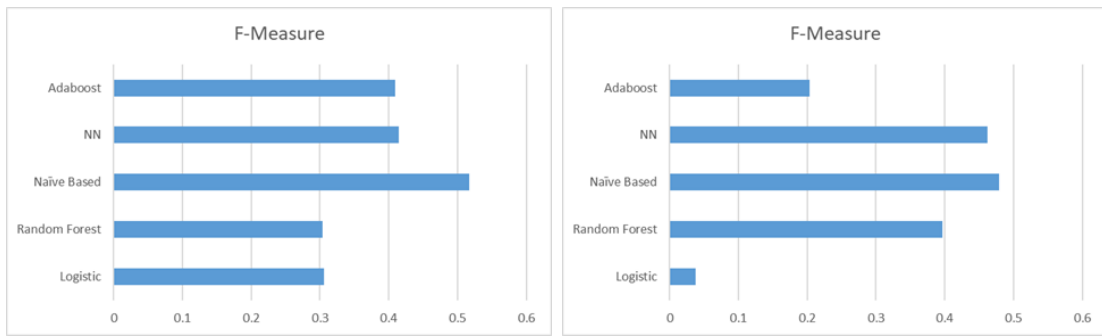
Figure 8.9: ROC - Semester 4



(a) Non-normalized

(b) Normalized

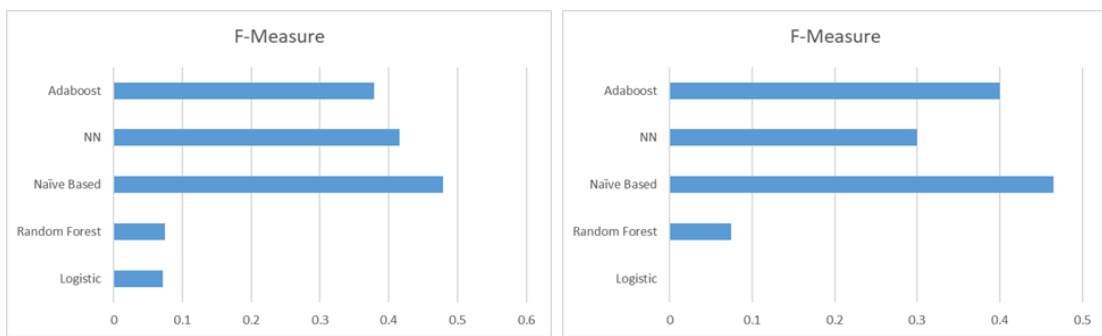
Figure 8.10: F-measure - Semester 2



(a) Non-normalized

(b) Normalized

Figure 8.11: F-measure - Semester 3



(a) Non-normalized

(b) Normalized

Figure 8.12: F-measure - Semester 4

For semesters 2 and 4, Random Forest produced the best AUC, having a value of 0.70 (Figure 8.7b) and 0.7612 (Figure 8.9a) respectively. For semester 3, Logistic Regression performed the best having an AUC of 0.78 (Figure 8.8a).

Regarding the F-measures, Naive Bayes gave the best performance for all semesters. It incrementally improved after each semester. It had a value of 0.44 (Figure 8.4a) for semester 2, 0.47 (Figure 8.5b) for semester 3, and 0.48 for semester 4 (Figure 8.6a).

Although Logistic Regression and Random Forest produced high AUC, they performed poorly in terms of F-Measure. The main reason for this is the having of imbalanced distribution of classes. AUC was high because of the large amount of negative samples, while the F-Measure was low because of the poor precision.

8.3 Risk Estimates

The faculty may decide to assist limited number of students every year. Hence, it is necessary to provide them with a list of students ordered by their risk or probability distribution produced by the classification algorithms. For instance, the faculty may want to assist 10 students who have the highest risk scores for which they will need to select the 10 students from top of the list.

In order to evaluate the performance of the algorithms in terms of ranking the students, we generated empirical risk curve[2] for each algorithm which is defined as the proportion of true positives to the total number of students in that group.

1. For each algorithm, we sorted students by their probability estimates in descending order. Top of the list represents students with the highest risk of receiving dismissal/probation.

2. We divided the list into 10 groups, and computed the mean empirical list for each one of them.
3. We plotted the results into a graph.

The algorithm that produces good ranking will have a monotonically increasing graph. On the contrary, having a monotonically decreasing graph implies that students having lower risk scores are more likely to get dismissal/probation. The results are shown below.

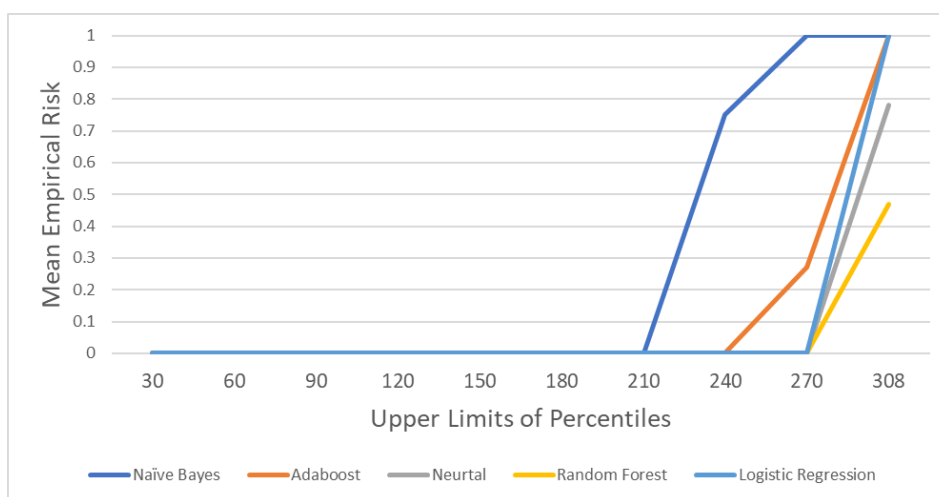


Figure 8.13: Risk Curves: Dismissal - Semester 2

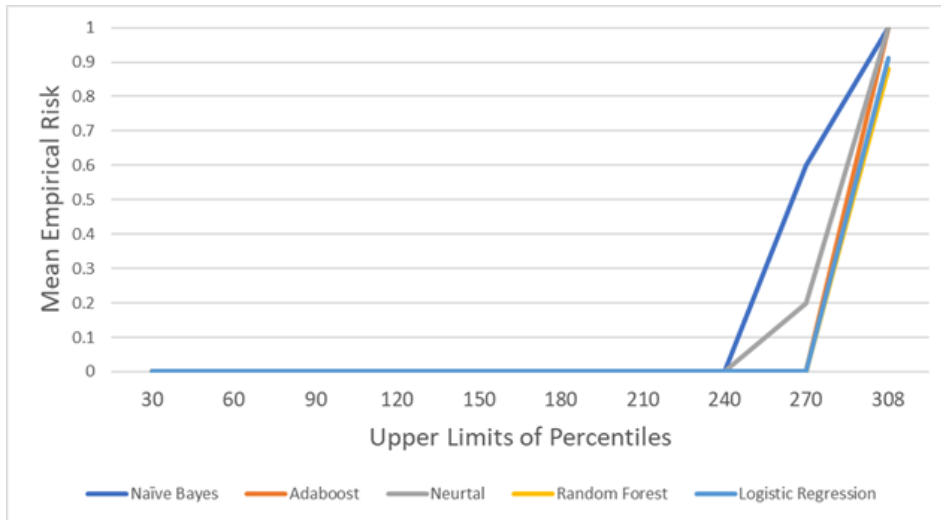


Figure 8.14: Risk Curves: Dismissal - Semester 2 - Normalized

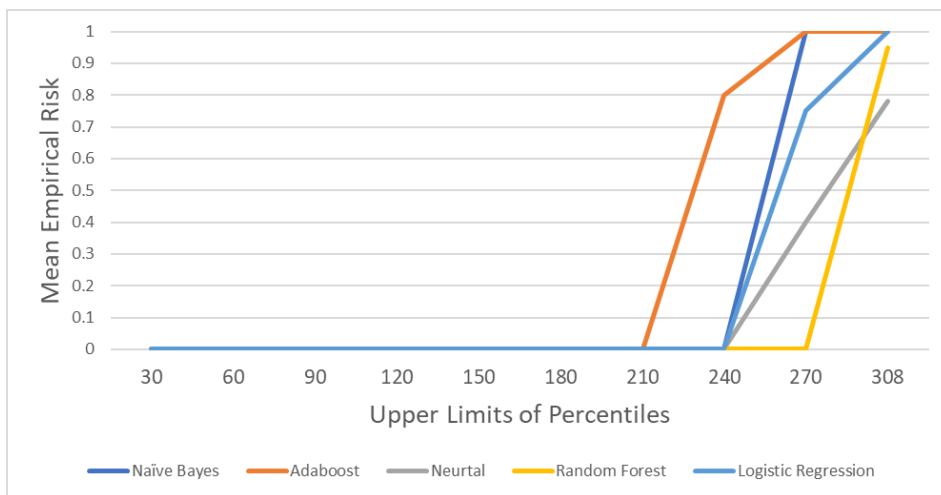


Figure 8.15: Risk Curves: Dismissal - Semester 3

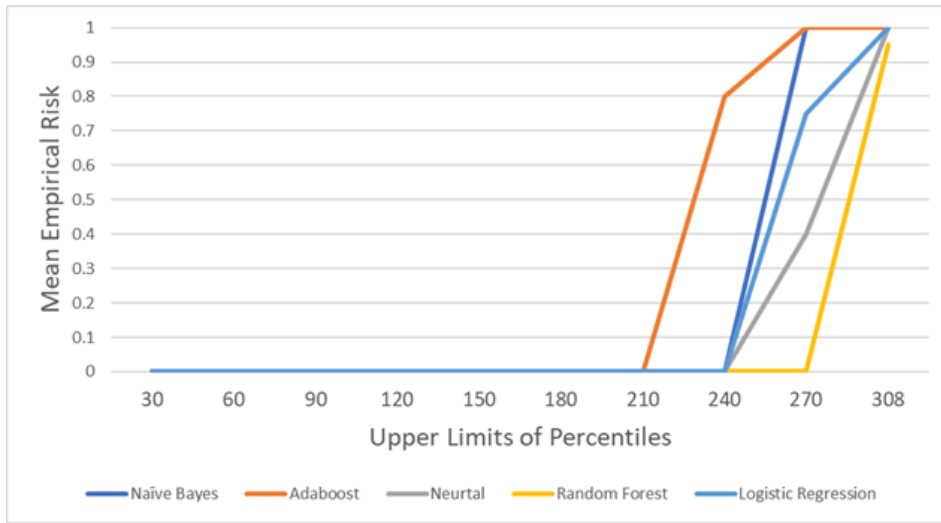


Figure 8.16: Risk Curves: Dismissal - Semester 3 - Normalized

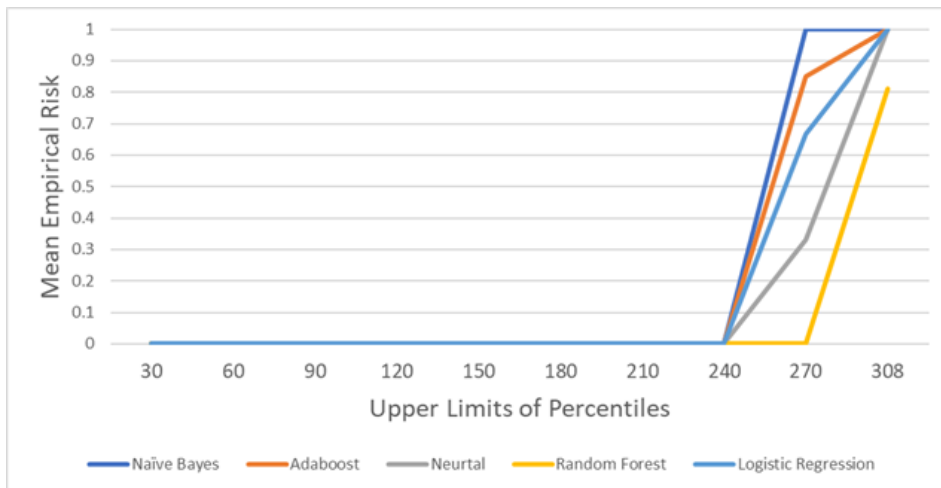


Figure 8.17: Risk Curves: Dismissal - Semester 4

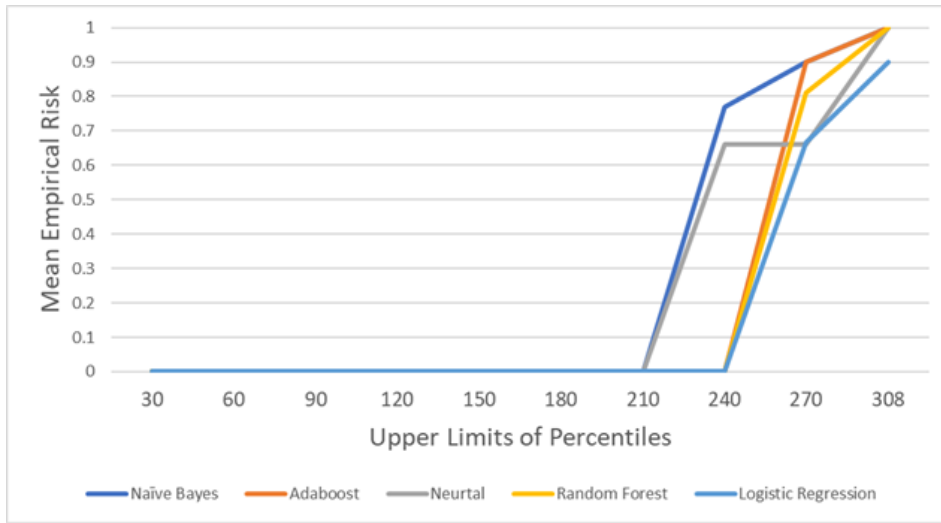


Figure 8.18: Risk Curves: Dismissal - Semester 4 - Normalized

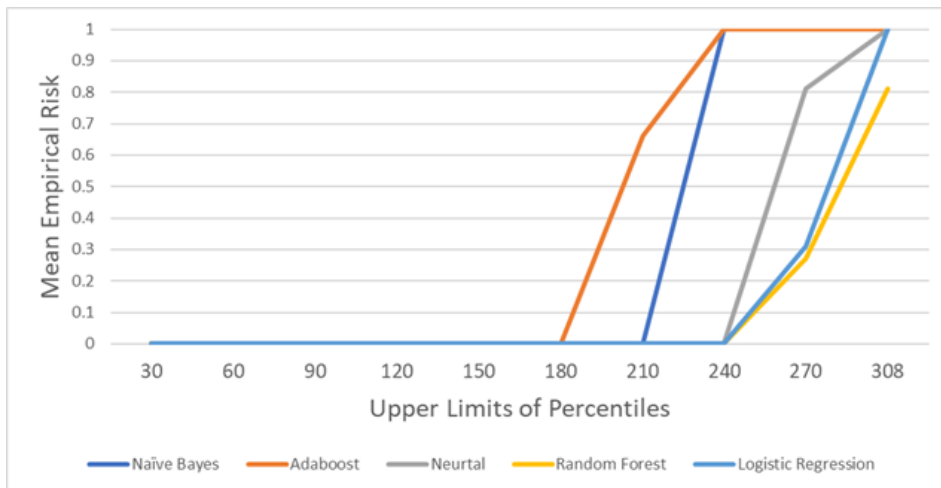


Figure 8.19: Risk Curves: Probation - Semester 2

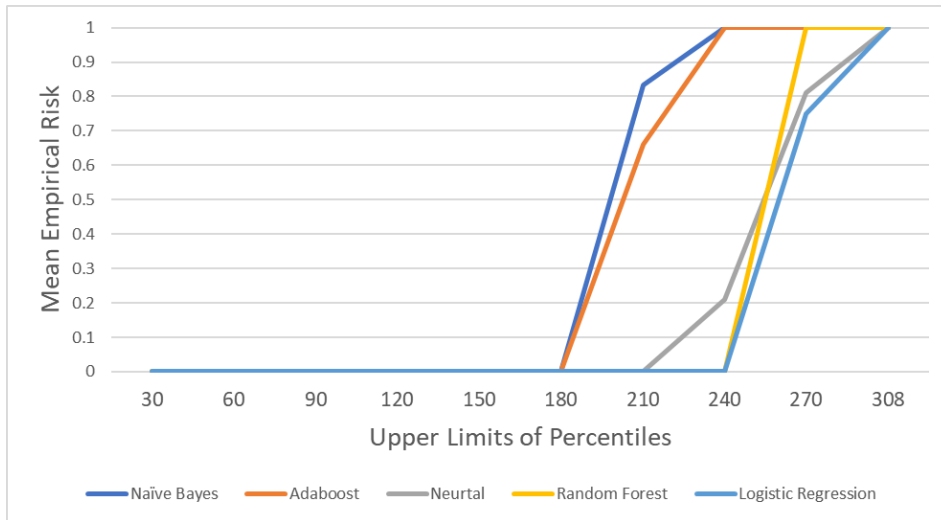


Figure 8.20: Risk Curves: Probation - Semester 2 - Normalized

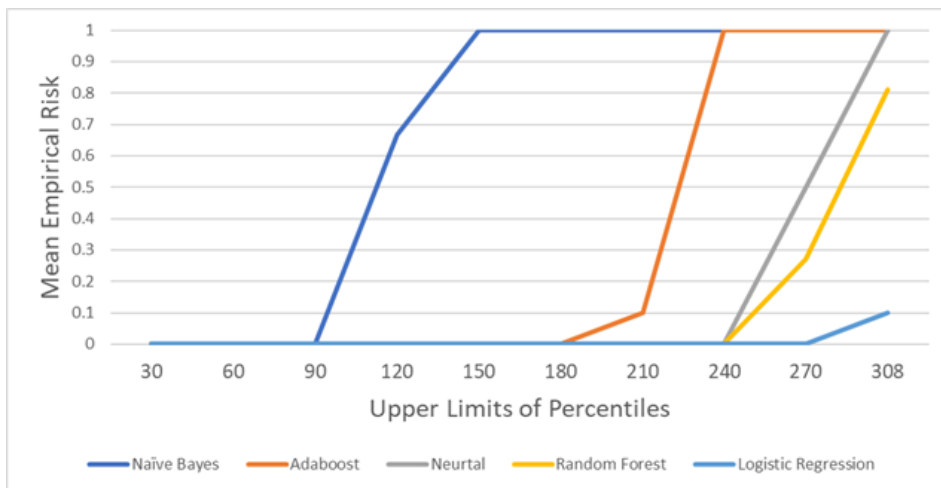


Figure 8.21: Risk Curves: Probation - Semester 3

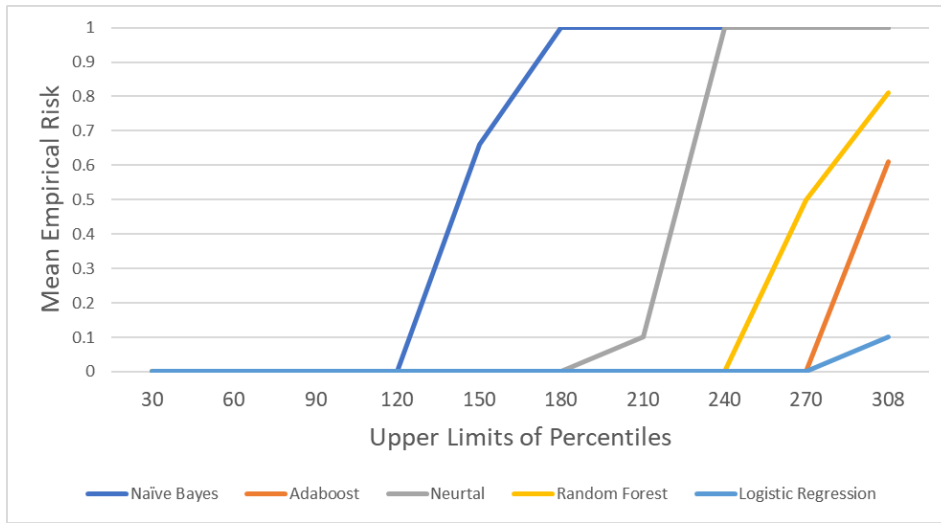


Figure 8.22: Risk Curves: Probation - Semester 3 - Normalized

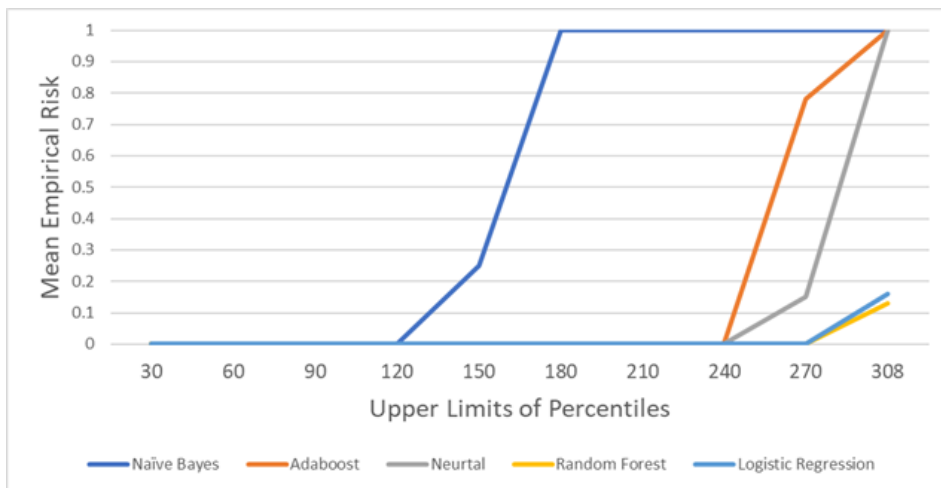


Figure 8.23: Risk Curves: Probation - Semester 4

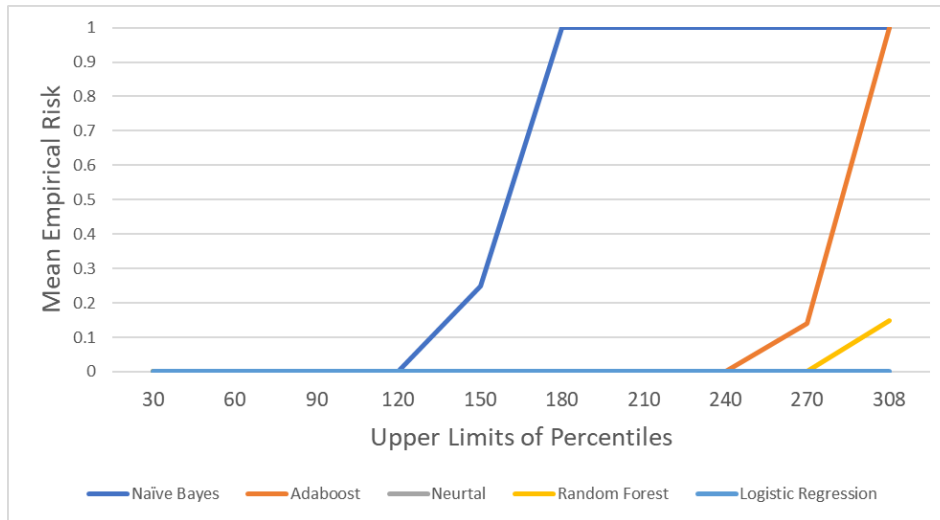


Figure 8.24: Risk Curves: Probation - Semester 4 - Normalized

The figures showed that all of the algorithms produced monotonically increasing curves which implied that they all had good risk estimates.

8.3.1 Evaluating Risk Estimates

In order to better evaluate the risk estimates, we introduced two evaluation metrics known as *Top K Recall* and *top K Precision* curves which provide precision and recall values at different values of threshold K. For instance, the faculty might be interested only with students that belong to the top 10 of the list(ranking).

The results are shown in the following figures.

Dismissal

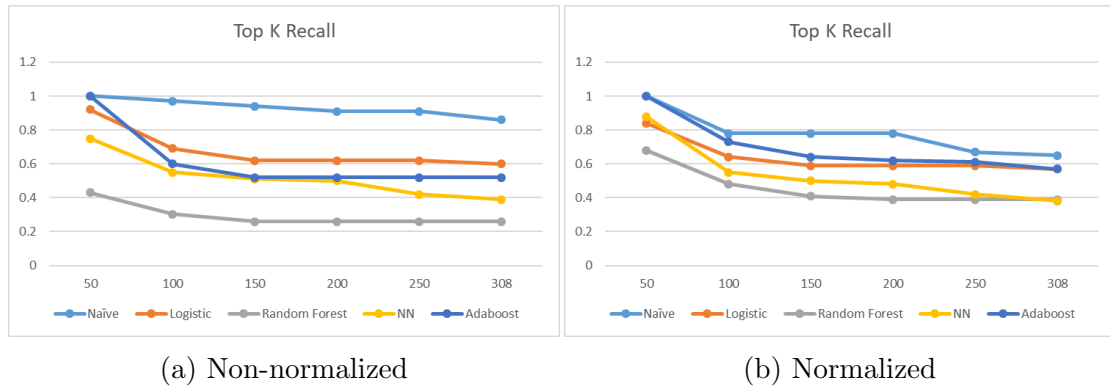


Figure 8.25: Top K - Recall - Semester 2

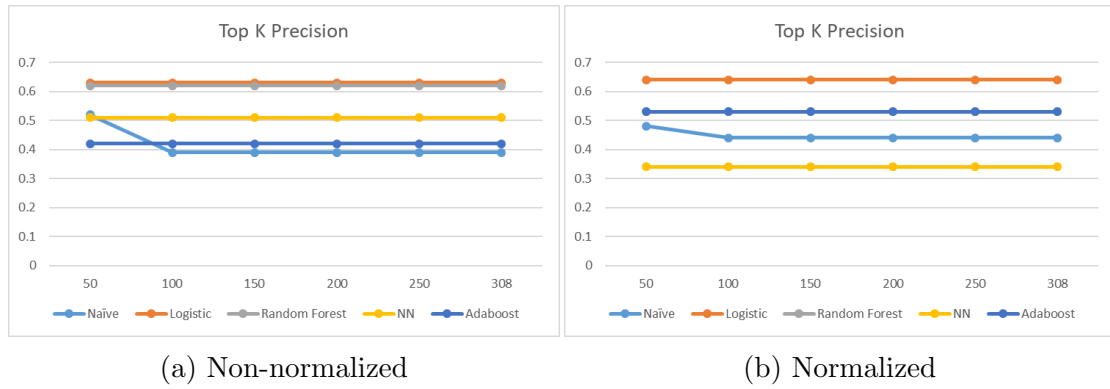
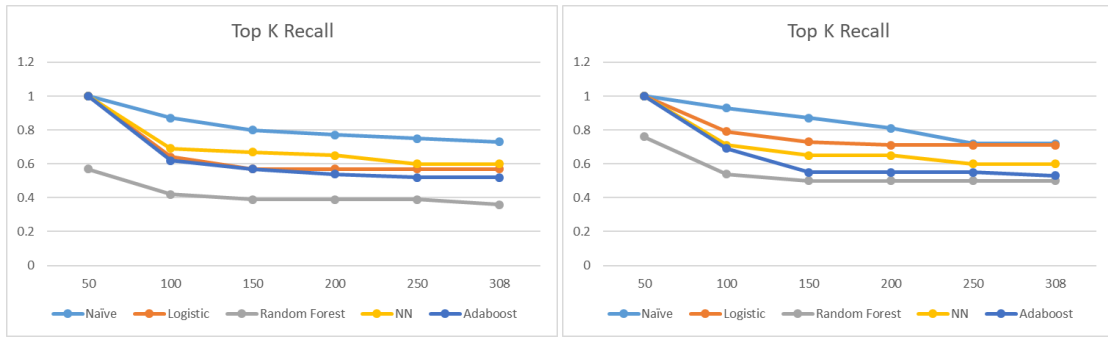


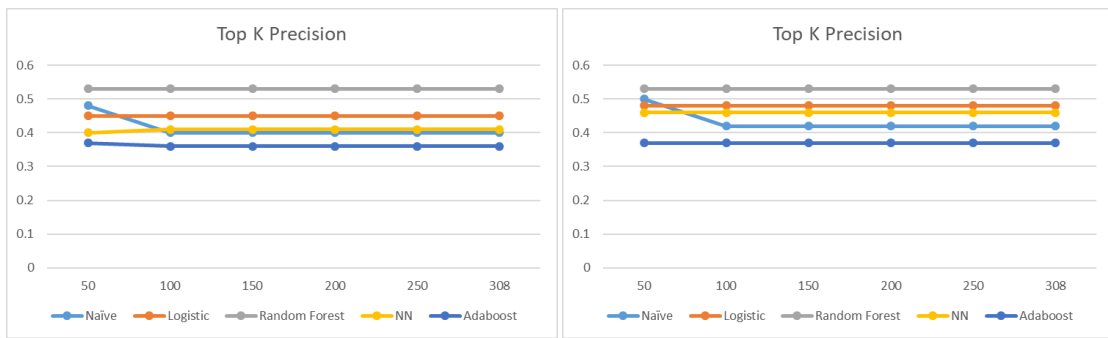
Figure 8.26: Top K - Precision - Semester 2



(a) Non-normalized

(b) Normalized

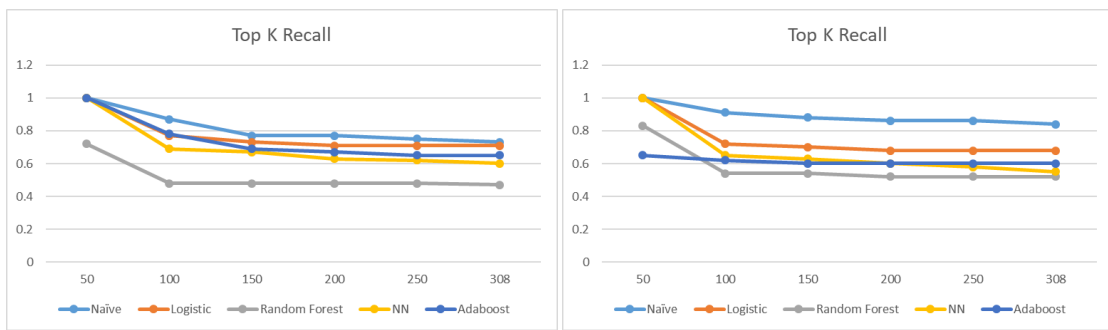
Figure 8.27: Top K - Recall - Semester 3



(a) Non-normalized

(b) Normalized

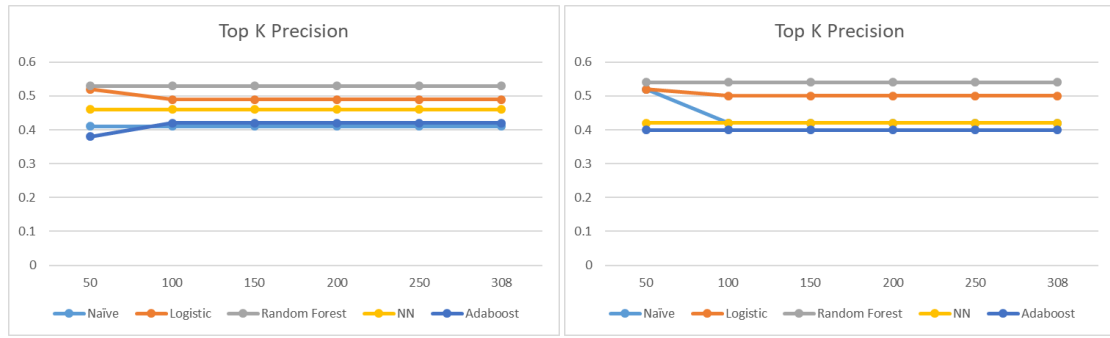
Figure 8.28: Top K - Precision - Semester 3



(a) Non-normalized

(b) Normalized

Figure 8.29: Top K - Recall - Semester 4



(a) Non-normalized

(b) Normalized

Figure 8.30: Top K - Precision - Semester 4

Figures 8.25, 8.27, and 8.29 illustrate the recall at top K curves for Semesters 2, 3, and 4 respectively. It can be seen that for all the values of K, Naive Bayes outperformed its counterparts for all semesters. Logistic Regression came second in most of the figures, coming third only in Figures 8.25(b) and 8.27(a) to AdaBoost and Neural Network respectively.

Precision at top K showed more of flat graphs because of the imbalance distribution of classes. Random Forest outperformed all the other algorithms for all the values of K except for Semester 2 (Figure 8.26), coming second to Logistic Regression. The latter gave the second best performance in terms of precision for Semesters 3 and 4 (Figures 8.28 and 8.30).

It can be inferred that Logistic Regression produced a better balance (F-measure) between precision and recall for different values at K.

Probation

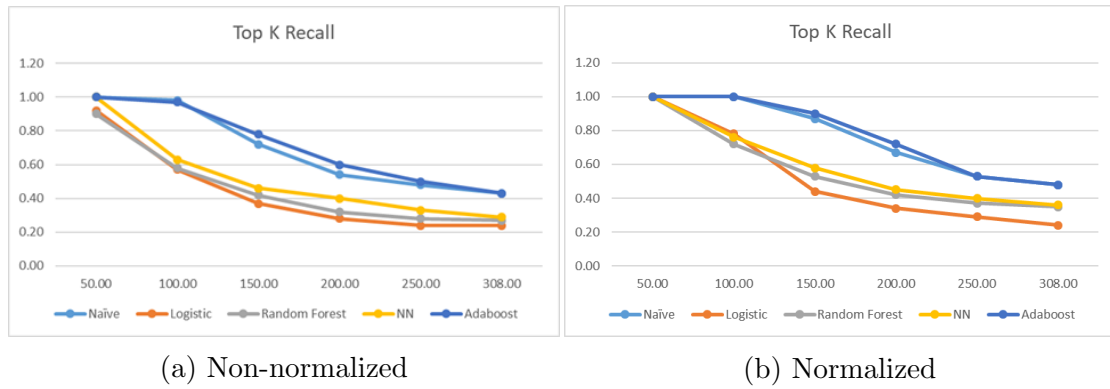


Figure 8.31: Top K - Recall - Semester 2

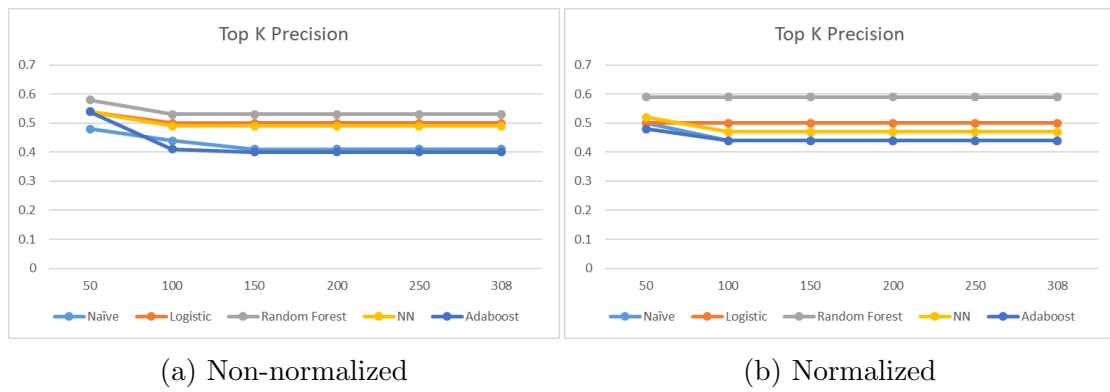
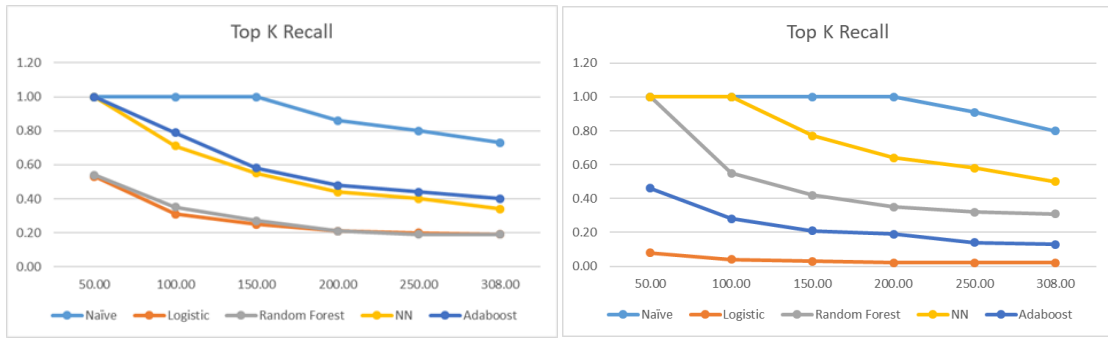


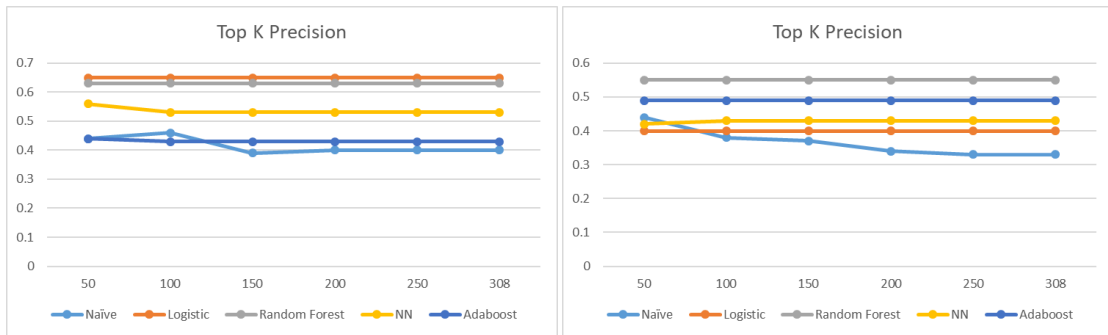
Figure 8.32: Top K - Precision - Semester 2



(a) Non-normalized

(b) Normalized

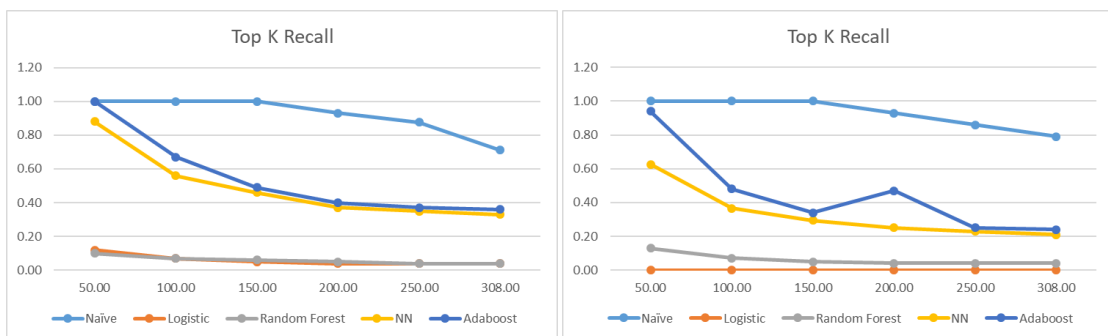
Figure 8.33: Top K - Recall - Semester 3



(a) Non-normalized

(b) Normalized

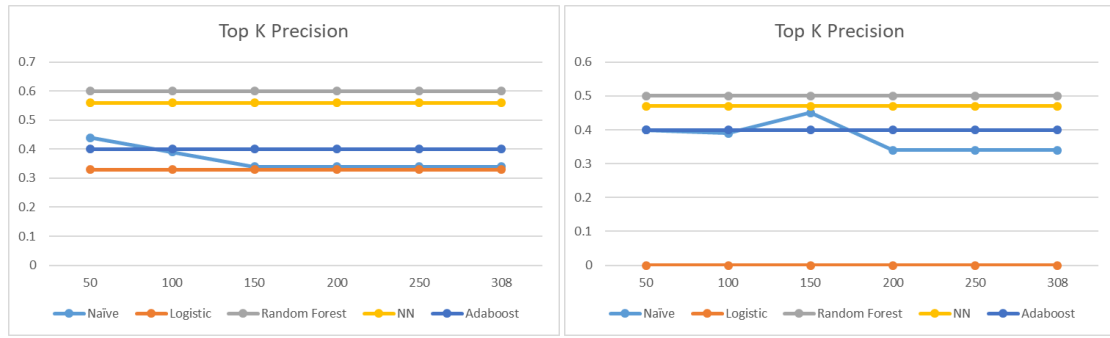
Figure 8.34: Top K - Precision - Semester 3



(a) Non-normalized

(b) Normalized

Figure 8.35: Top K - Recall - Semester 4



(a) Non-normalized

(b) Normalized

Figure 8.36: Top K - Precision - Semester 4

Naive Bayes outperformed its counterparts in terms of top K recall for all semesters except for semester 2 (Figure 8.31) where AdaBoost performed slightly better. It also produced the best balance between precision and recall for different values of K. Random Forest, on the other hand, performed the best for the top K precision but performed very poor with respect to the top K recall. The rest of the algorithms performed poorly for all semesters.

8.4 Feature Importance

In order to understand which factors contributed the most to the predictions, we used the Information Gain (IG)[16] approach which works independently of the algorithms, to rank the features according to their level of importance.

Rank	Sem2	Sem3	Sem4
1	Theory - F	Probation Consecutive	Elective - Avg
2	Elective - F	Programming - F	Theory - Avg
3	Programming - F	Number of Probation	Programming - F
4	Theory - P	Elective - A	Elective - F
5	Theory - W	Elective - F	Theory - P

Figure 8.37: Feature Ranking - Dismissal

Rank	Sem2	Sem3	Sem4
1	Number of Probation	Number of Probation	Elective - Avg
2	Probation Consecutive	Elective - Avg	Theory - Avg
3	Probation_2	Theory - Avg	Programming - F
4	Elective - Avg	Programming - Avg	Elective - F
5	Theory - Avg	Probation Consecutive	Theory - P

Figure 8.38: Feature Ranking - Probation

It can be inferred from Figure 8.37 that *Elective Failed* and *Programming Failed* were highly ranked across all the semesters in terms of contributing the most for getting into dismissal. *Theory Passed* also made it into the top 5 except for semester 3.

Figure 8.38 shows the feature ranking according to the probation. *Elective Average* and *Theory Average* were highly ranked across all the three semesters.

In addition, *Number of Probation* and *Probation Consecutive* features also contributed significantly, except for semester 4.

CHAPTER 9

APPLYING REINFORCEMENT LEARNING

Reinforcement learning (RL) is being widely used in artificial intelligence, gaming, statistics, optimization, mathematics, and many more. It is different from supervised learning, which was previously used in this paper. It is the science of choosing actions or giving suggestions rather than predicting something that might happen in the future. It is the learning of what to do next, what actions to take to maximize the reward for the future. The policy must interpret actions that it has tried in the past and pick the one that yields the most reward. The reward, action, and policy are described in more detail in section 9.2.

In this chapter, we use some of the main concepts from RL and the Markov decision process to build a model/policy that can assist the advisors. It will recommend what kind of courses to take next, with the goal of protecting students from dismissal/probation.

9.1 Reinforcement vs Supervised Learning

Firstly, it is important to mention the key differences between reinforcement and supervised learning.

- The objective of RL is to find the best action leading to the maximum reward instead of applying predictions or classifications.
- The training process of RL involves determining the best policy instead of fitting the best model.

- In RL, the recommended action of a state can be dependant on others. For instance, in a chess game, the next best move of a player depends on the opponent's state. In our work, the states are independent with each other because the course recommendations of any student at any semester does not depend on other students.

In supervised learning, on the other hand, the learner will always produce the same predictions for each record after training.

9.2 Elements of RL

RL consists of five main sub-elements: states, actions, rewards, transitions, and policy.

1. **State (*s*)**: The first element of our RL is the state of the environment. It defines all current features of students during a specific time. We had two kinds of states: *Initial* and *Regular*.

Initial states contain features related to students' records before entering university. They are defined as the following.

SATV - SAT Verbal
SATM - SAT Mathematics
SA1 - Class Average - 11th Grade
SA2 - Class Average - 12th Grade
SR - School Rank

Table 9.1: List of Initial State Features

On the other hand, regular states represent the university grades of students after each semester.

SN - Semester Number (0-12)
MPA - Major Programming Average (0-100)
MPC - Number of Major Programming Courses Taken (0-20)
MTA - Major Theory Average(0-100)
MTC - Number of Major Theory Courses Taken(0-20)
EA - Elective Average(0-100)
EC - Number of Elective Courses Taken(0-20)
PN - Probation Number(0-2)

Table 9.2: List of Regular State Features

All the features have numerical types. For instance, a student may belong to the following state(table 9.3) at the end of the first semester.

SN	MPA	MPC	MTA	MTC	EA	EC	PN
1	75	1	70	1	80	2	0

Table 9.3: List of Regular State Features

Two states are identical if they have the same feature values. Initially, each state will have only one action(set of courses). However, if two states are identical, then both actions are grouped into a single state. It was important for each state to have multiple actions in order for the policy to be trained and learn which action leads to a higher reward. For instance, two students having identical feature values may have registered for different set of courses. Our goal is to build a model that can identify which action is the best for

each state. However, since we had a lack of training data, the number of identical states were quite low which meant that most of the states had only one action. Hence, the policy could not be trained properly. To overcome this issue, we categorized the average grades into seven categories: 100, 95, 90, 85, 80, 75, 70, and 60. For example, if a student got a major average of 87, the model will categorize the grade as 90. For the SAT scores, the categories included: 800, 700, 600, 500, 400, and 300.

Finally, our training data consisted of:

- 800 students
 - 75 unique initial states
 - 1,350 unique states in total
2. **Action (a)**: It represents set of courses that students should register the following semester. It is defined as an array of six integers [A, B, C, X, Y, Z]
- A: Number of major courses to take
 - B: Number of theory courses to take
 - C: Number of elective courses to take
 - X: Number of major courses to take at summer
 - Y: Number of theory courses to take at summer
 - Z: Number of elective courses to take at summer

The goal is to find a policy that maps a state to a set of the **courses to be taken (action)**. For instance, after a student completes his/her first

semester, the policy might suggest him/her to take three elective courses and one major course for the following semester.

3. **Reward (\mathcal{R}):** A reward function takes a state s as input and returns its corresponding reward $\mathcal{R}(s) \in \mathbb{R}$. However, this reward does not represent the future. It is only the reward received for reaching that state. A higher semester average will result in a higher reward. The reward functions for our states are as follow.

- $\mathcal{R}(s) = \text{Semester Average(SA)}$; If student passes the semester without probation/dismissal.
- $\mathcal{R}(s) = -1200$; If student is placed on probation.

We wanted states that represented students receiving a probation to be penalized more than the regular states(without probation). In Figure 5.2, we already observed that the maximum number of semesters needed for a student to graduate was 12. In addition, the maximum reward students can receive for each semester is 100(Semester Average). As a result, by assigning a reward of -1200(100×12), we guarantee that the cumulative(total) reward received for each of students having at least one probation will always be negative, and students with higher number of probation will always have lower cumulative rewards. For instance, dismissed students will always have lower cumulative reward than the ones with only one probation. Building the policy and calculating the cumulative rewards are explained more accurately in section 9.4.

Transitions (T): The transition system model is a higher order representation

of the students' behavior throughout their semesters. Since identical states are grouped together, each state will have many transitions. A transition can be defined as: $T(s) \rightarrow s'$. If s has a semester number (SN) of 2, then s' should have a SN of 3.

An example of transition model is shown in Figure 9.1 as mentioned below.

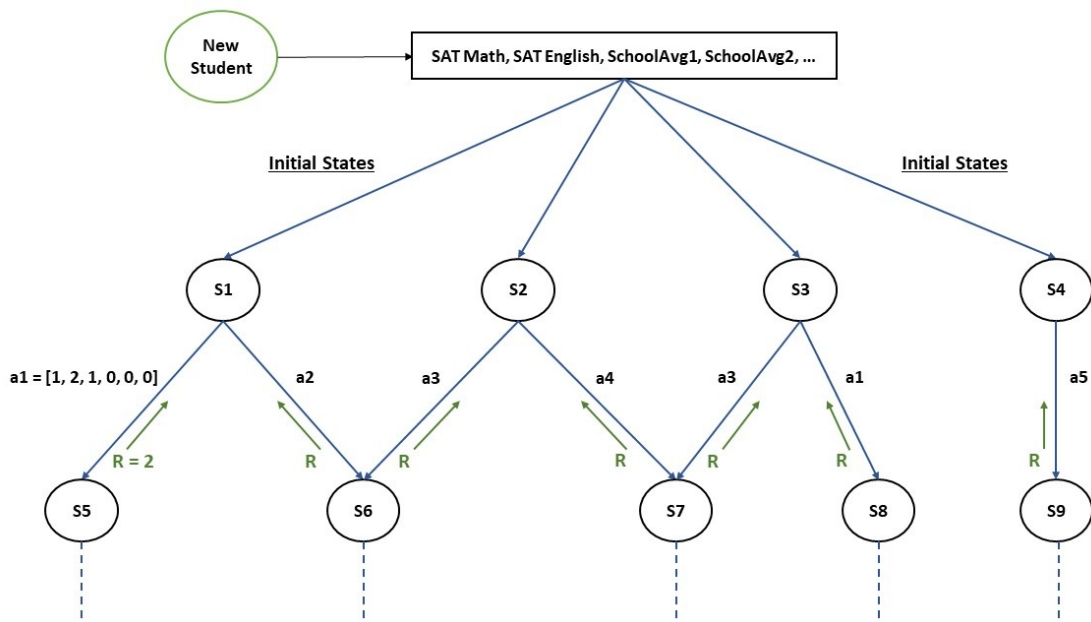


Figure 9.1: An example of Transition System

s_1 to s_4 represent the initial states (described in section 9.3) of the transition system while the rest represent the regular states. All states have future rewards (R). For instance, the reward for reaching s_5 equals 2. The process of calculating the rewards is described in section 9.4. In addition, a_1 to a_5 represent the set of actions taken by the states. We can observe that s_1 represents two students who belong to the same state. s_1 could be students coming from the same school having similar grades. One of them took action a_1 (set of courses) while the other

took a_2 which led them to two different states; s_5 and s_6 . Moreover, two different states can also lead to an identical state. For instance, s_1 taking a_2 led to the same state s_6 as s_2 taking a_3 .

Policy (π): The goal is to construct the optimal policy. It should be a function that takes a state as a parameter and returns the best action. The system should input students as state parameters and the policy should output recommended courses to each of them.

Value (V): It is defined as the future reward as opposed to the current reward. $V(s)$ returns the expected future reward of the current state s under a certain policy (π). If $V(s) > V(s')$, it means that s is more likely to graduate safely without encountering dismissal or probation. It was also important to quantify the performance of **state-action** $\mathbb{P}(s, a)$ that represents the future reward of a state s taking a certain action a . It is defined as:

$$\mathbb{P}(s, a) = \mathcal{R}(s') + \gamma V(s'), \text{ where } s \xrightarrow{a} s'$$

Consequently, the future reward of a state will be the reward of the action that has the highest value:

$$V(s) = \max_{a \in \mathcal{A}_s} \mathbb{P}(s, a)$$

However, there are some states that have no actions. We call them the *terminal states*. They represent the final semesters of which students have either graduated or been dismissed. In this case, the future reward of a terminal state represents

the reward of the state itself:

$$V(s) = \mathcal{R}(s)$$

The discount factor γ is a number between 0 and 1 that represents the importance of the future reward. Higher values give more priority for the longer terms. For example, the policy may recommend a set of courses for a particular semester knowing that the student may not perform well that semester but as a long term plan, it is the best choice and the student should graduate safely. Lower values indicate that the policy does not care about the future and suggests set of courses that guarantees the best outcome for that particular semester only. In our application, we assigned $\gamma = 1$, since our main focus was to help students to graduate(long term) safely with the highest cumulative average.

9.3 Markov Decision Process Characteristics

Our environment can be formalized as a Markov Decision Process which was first introduced by Richard Bellman[17] and is currently frequently used in RL algorithms and optimization. Some of the characteristics of MDP are as follows:

- The environment is fully observable and the policy can see the entire state. The state does not depend on other variables where the value is unknown. In our case, students are not dependent on each other, and the algorithm can observe all the grades. In contrast, a good example of a non observable environment is a poker game where the player at any state cannot observe the opponents hand.

- The future is independent of the past and history can be disregarded because our present state accumulates all the relevant information from the history such as the grades. Once a student reaches a certain state, the model does not consider what happened in the past.

It was important to understand the nature of our environment before selecting the right algorithm. One of the main techniques used to solve MDP is known as *Value Iteration* technique which is described below.

9.4 Policy Construction

The goal is to find the optimal policy π that guarantees the best selection of courses. To achieve this, we used the Value Iteration technique that improves the estimate of $V(s)$ iteratively. The algorithm initializes $V(s)$ to random values and repeatedly updates the $\mathbb{P}(s, a)$ and $V(s)$ values until they converge to the optimal values.

- V encodes the future reward and is computed: $V(s) = \max_{a \in \mathcal{A}_s} \mathbb{P}(s, a)$
- Given a performance measure \mathbb{P} , we needed to construct a policy that can maximize the cumulative reward by selecting the action with the highest reward. Hence it can be formulated as:

$$\textbf{Optimal Policy} \quad \pi(s) = \arg \max_a \{\mathbb{P}(s, a) \mid a \in \mathcal{A}(s)\}$$

where “arg” returns the index of the action with the maximum cumulative reward for input s .

9.4.1 Applying Value Iteration

The algorithm below computes the policy in a finite state-space model. It will compute the future rewards of all the actions generated by each of the states. Hence, the best action for a specific state would be the one with the highest reward.

```
1: Definition:  $\mathcal{A}_s =$  Set of actions in  $s$ 
2: Input: Initialization of  $V(s) = 0$  for all  $s \in S$ , error = -1
3: while error  $\neq 0$  do
4:   error = -1
5:   for each  $s \in S$  do
6:     if  $s$  is a terminal state then
7:        $V(s) = \mathcal{R}(s)$ 
8:     else
9:       for each  $a \in \mathcal{A}_s$  do
10:        tmp =  $\mathcal{R}(s') + V(s')$ , where  $s \xrightarrow{a} s'$ 
11:        if  $\mathbb{P}(s, a)$  is defined then
12:          error = max(error, |tmp -  $\mathbb{P}(s, a)$ |)
13:        else
14:          error = max(error, tmp)
15:        end if
16:         $\mathbb{P}(s, a) =$  tmp
17:      end for
18:       $V(s) = \max_{a \in \mathcal{A}_s} \mathbb{P}(s, a)$ 
19:    end for
20:  end while
```

9.5 Policy Evaluation

The iteration should stop when the policy converges to the optimal values[18]. Having an acyclic transition system and terminal states that produce the same reward at each iteration, the error measure should reach a value of zero after a certain iteration. It occurs when $\mathbb{P}(s, a)$ does not learn/improve anymore and $\text{tmp} - \mathbb{P}(s, a)$ starts producing a value of 0.

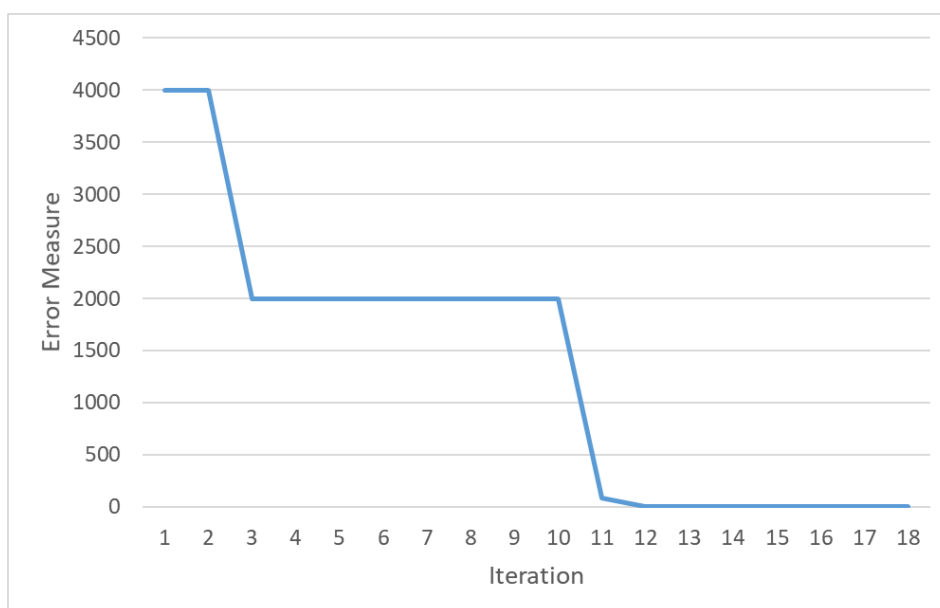


Figure 9.2: Error Measures Over Iterations

In Figure 9.2, we plotted the error measure after the end of each iteration and noticed that the policy reached its optimal state after 12 iterations as expected, given that the maximum number of semesters needed to graduate in our dataset was 12, while having the reward values back-propagate from the terminal states till the initial states.

Moreover, in order to evaluate the performance of the policy, we generated the

best trace and the worst trace for each state after semester one, and analyzed their performance after each iteration. The best traces are computed by selecting actions with the highest rewards for each state. On the contrary, the worst trace is computed by selecting actions with the lowest rewards. A path will lead either to *success* or *failure*:

- *Success* occurs when the path graduates safely without any probation. That is, according to the policy, if the student had followed the model's suggestion, then he/she might have had a better chance to graduate safely.
- *Fail* occurs when the path either graduates with a probation or gets a dismissal.

We formulated the performance as:

$$P = \frac{n_{Success}}{n_{Fail} + n_{Success}}$$

where $n_{Success}$ and n_{Fail} represent the number of success/fail paths the policy was able to generate out of the total number of students at risk. For instance, if 10 students were dismissed and the policy was only able to generate success paths for five of them, then $n_{Success} = 5$, $n_{Fail} = 5$, and P would be 50%.

In order to check if the policy was learning and improving, we plot the number success/fail for the best/worst trace after each iteration. The dataset contained 98 dismissed students and 310 students who received a probation.

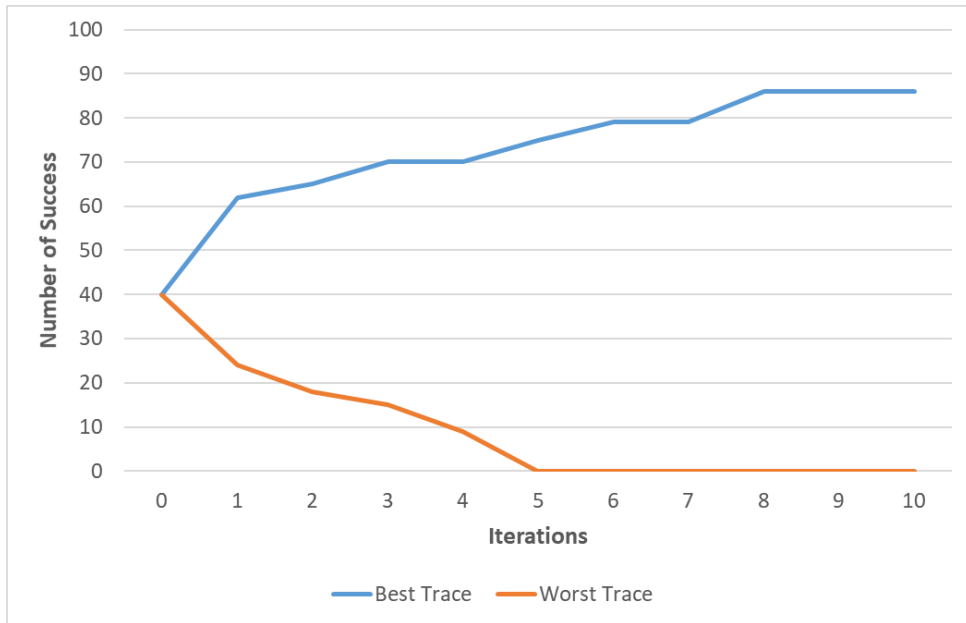


Figure 9.3: Performance Evaluation of the Best and Worst Trace: Dismissal

We can notice that, at iteration 0 (Figure 9.3), the worst trace and the best trace represented the same trace and have the same number success/fail values. The reason is because the policy still cannot differentiate the importance of the corresponding actions since $\mathbb{P}(s, a)$ for all $s \in S$ are still undefined. At iteration 1, the policy showed slight improvement having the number of success increase from 40 to 62 for the best trace, and decrease from 40 to 24 for the worst trace. This implied that, the best traces were able to carry the 62 out of 98 students into graduation without any risk.

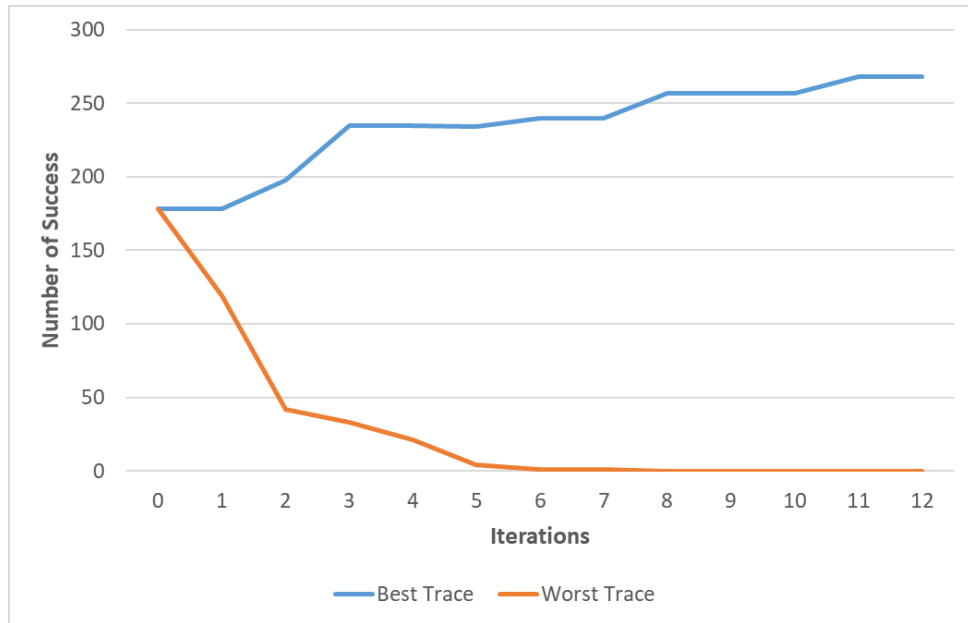


Figure 9.4: Performance Evaluation of the Best and Worst Trace: Probation

Overall, figures 9.3 and 9.4 show that the performance of the best traces were iteratively improving leading more students to success through their best traces. They were able to recommend better set of courses after each iteration. For instance, in Figure 9.3, the performance P reached an optimal value of 87% after the 8th iteration. The 13% are those students that were dismissed for whom the policy was not able to provide a successful path. One of the main reason for this is the lack of training data since the policy could not find any alternative actions that can lead to success. The policy only suggests actions by learning from the experiences of the previous students. However, throughout semesters, new states and transitions will be created and added as new students enroll into the university which will considerably improve the performance of our policy.

At the end of execution, the policy will produce a matrix table that maps state-action instances to reward values.

State/Action	A_1	A_2	A_3	A_4
S_1	400	370	366	386
S_2	385	360	410	430
S_3	390	386	398	400
S_4	420	340	379	364

Table 9.4: Table Representation of a Policy

For example, the policy will recommend action A_3 (Table 9.4) to S_2 because it has the highest reward (410) of the row. It can also be inferred that, A_2 is the worst action to take from S_2 , since it has the lowest value(360). In section 9.8, we describe this process with real examples.

9.6 Handling New States

The value iteration algorithm(section 9.4) learnt from the past students and generated a policy which mapped each state to the corresponding action leading to a maximum reward. However, the faculty may need to assist only students at risk, hence, the system can simply input states(Figure 9.5) representing those students into the policy and retrieve the recommended courses. However, in certain cases, the state may not be defined in the policy. This happens when the size of the training data is not enough to cover all the possible states. To overcome this issue, the model will first check if the state has been defined by the policy or not. If not, it will select the nearest state to the new state using Euclidean Distance(with normalization) and return its

corresponding action.

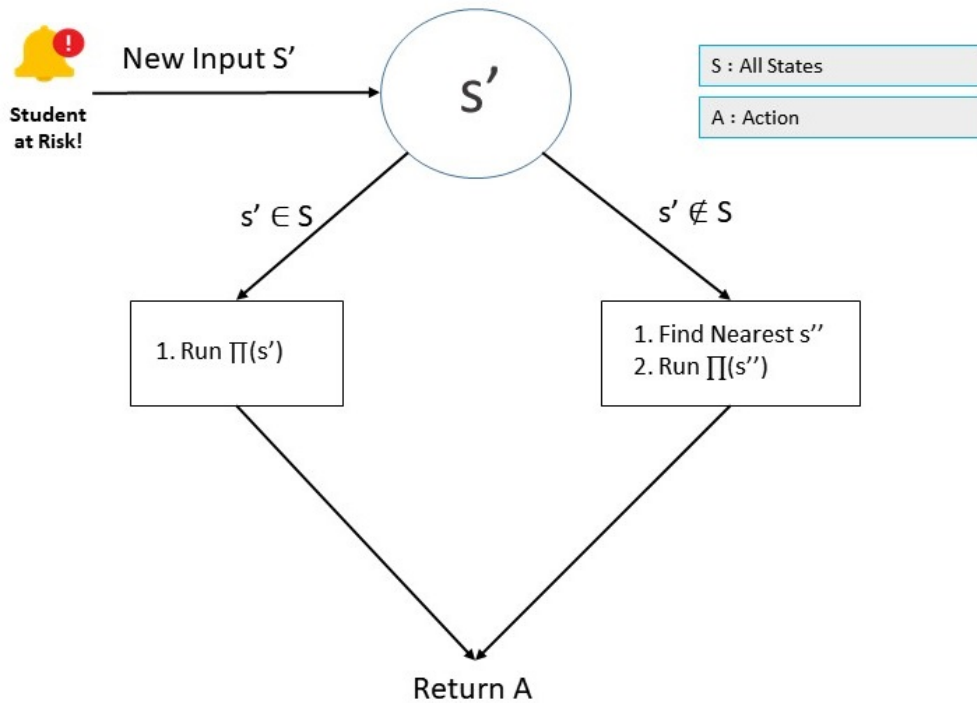


Figure 9.5: Policy Evaluation of New States

9.7 Test Sample

In this part, we provide an example of a dismissed student and describe why he/she might have gone off track according to the policy.

We selected the first student from the dataset. The student was dismissed at the end of Semester 5. In order to understand what may have caused that, we compared his/her action with the policy's recommendation.

First, at the end of the first semester, the student belonged to the following state S_1 :

Property	Value
Semester	1
Total Elective Average	40
Total Elective Courses Taken	2
Total Major Average	70
Total Major Courses Taken	1
Total Theory Average	60
Total Theory Courses Taken	2
Number of Probation	0

Table 9.5: List of State S_1 Properties

and registered for the following courses (A).

Property	Value
Number of Major Courses	1
Number of Theory Courses	0
Number of Elective Courses	3

Table 9.6: List of A Properties

To understand if the student picked the best action, we imported S_1 into the policy and analyzed the corresponding actions.

Action	Major	Theory	Elective	Reward
A_1	2	1	1	590
A_2	0	3	1	520
A_3	1	0	3	560
A_4	2	1	3	500

Table 9.7: Corresponding Actions From S_1 and Their Rewards

According to the policy, the student had chosen the second best action since

$A = A_2$ and it has the second highest reward. Hence, it can be inferred that, choosing A_1 might have resulted in a better academic position in the future.

9.8 Policy Overview

To have a general overview of what the policy consists of, we presented the mapping between state-action in a tabular form. It provided an insight of a student's performance at different stages of course registration.

9.8.1 Course Recommender

Since identical states are grouped into one (section 9.2), a state is said to be more recurring/populated if it contains more identical states than the other. We selected the top five recurring states from each semester starting from the initial one and provided the best and the worst action for each of them. Unlike the best state, the worst action of a policy can be defined as:

$$\pi(s) = \arg \min_a \{P(s, a) \mid a \in \mathcal{A}(s)\}$$

It is the action that results in the lowest future reward.

We defined actions (section 9.2) as: $N_{Major} - N_{Theory} - N_{Elective}$ and states as S_i^j where i and j represent the states' semester number and recurring rank respectively. Ranks are ordered from 1 to 5, 1 being the state with the highest recurrence for semester i . Actions containing summer courses were not highly recurring, hence we did not include any summer features in them. On the other hand, states are defined by their features which were described in section 9.2.

- **SATV**: SAT Verbal Grade

- **SATM**: STA Math Grade
- **SA1**: School Average Grade 11
- **SA2**: School Average Grade 12
- **MPA**: Cumulative Major Programming Average
- **MPC**: Cumulative Major Programming Courses Taken
- **MTA**: Cumulative Major Theory Average
- **MTC**: Cumulative Major Theory Courses Taken
- **EA**: Cumulative Elective Average
- **EC**: Cumulative Elective Average

	SATV	SATM	SA1	SA2	SR	Best A.	Worst A.
S_0^1	400	600	60	60	7	2-0-3	0-2-3
S_0^2	400	700	60	60	7	0-3-3	1-1-2
S_0^3	300	300	60	60	7	1-0-3	3-1-1
S_0^4	400	700	60	60	7	0-1-3	0-3-1
S_0^5	400	700	60	60	7	0-1-3	0-3-1

Table 9.8: Policy: Top 5 Initial States

Table 9.8 shows that students who belonged to S_0^1 before beginning their university career performed best when they registered for two major and three elective courses in their first semester. However, they performed worst when

they registered for 2 major courses and four elective courses. The same procedure was applied till semester four.

	MPA	MPC	MTA	MTC	EA	EC	Best A.	Worst A.
S_1^1	60	1	60	2	60	2	2-1-2	0-1-4
S_1^2	0	0	85	2	85	2	0-3-3	1-0-3
S_1^3	75	1	70	2	70	2	1-1-3	1-2-2
S_1^4	60	1	60	2	60	2	2-1-2	1-5-1
S_1^5	0	0	60	2	60	2	0-2-4	2-0-2

Table 9.9: Policy: Top 5 Regular States: Semester 1

	MPA	MPC	MTA	MTC	EA	EC	Best A.	Worst A.
S_2^1	60	3	60	2	60	2	1-2-3	0-2-3
S_2^2	75	3	70	2	70	2	0-2-3	1-2-2
S_2^3	60	3	75	2	75	2	0-3-2	1-2-2
S_2^4	75	3	70	2	70	2	0-3-2	1-1-4
S_2^5	85	3	75	2	75	2	1-2-2	0-2-4

Table 9.10: Policy: Top 5 Regular States: Semester 2

	MPA	MPC	MTA	MTC	EA	EC	Best A.	Worst A.
S_3^1	75	5	70	3	70	3	0-3-2	1-1-4
S_3^2	85	5	75	3	75	3	1-1-2	0-4-2
S_3^3	80	4	75	5	75	5	1-1-4	2-1-3
S_3^4	75	5	75	3	75	3	1-2-2	1-2-2
S_3^5	80	4	75	5	75	5	1-1-4	2-1-3

Table 9.11: Policy: Top 5 Regular States: Semester 3

	MPA	MPC	MTA	MTC	EA	EC	Best A.	Worst A.
S_3^1	85	4	85	3	85	3	2-0-3	1-1-3
S_3^2	75	6	75	5	75	5	0-2-4	0-3-2
S_3^3	75	7	70	5	70	5	0-2-3	0-3-1
S_3^4	75	7	70	4	70	4	0-3-1	0-2-3
S_3^5	90	8	90	5	90	5	1-0-3	1-0-4

Table 9.12: Policy: Top 5 Regular States: Semester 4

9.8.2 *N-th Recommendation*

It might occur that students cannot or are not willing to register for the recommended courses because of financial or personal reasons. For instance, some of the courses may not be available in the next semester or there might be schedule conflicts between them. It is essential to have a policy that is flexible enough to provide alternative recommendations or the next best actions that meet students' requirement. In order to achieve that, for each state, we order the actions by their reward values in descending order. For instance, state S_0^1 (Table 9.7) will have the following results (Table 9.13).

	Action	Reward	n
A_1	2-1-1	590	0
A_2	1-0-3	560	1
A_3	0-3-1	520	2
A_4	2-1-3	500	N

Table 9.13: Top 5 Actions for S_0^1

Hence, it can be implied that:

- $n = 0$ defines the best action
- $n = 1$ defines the second best action
- $n = 2$ defines the third best action
- $n = N$ defines the worst action; N being total number of actions from s .

Accordingly, the policy will find the next best action that meets the students' requirement. For example:

1. If a student decided to register for four courses, then the policy will output A_2 ($n = 1$). It will not recommend A_1 since it contains 5 courses(2-0-3)
2. If a student decided to register for six courses, then the policy will output A_3 ($n = 2$).
3. If a student decided to register for the best set of courses, then the policy will output A_1 ($n = 0$).

9.9 Summary

In this chapter, we built a course recommendation system using reinforcement learning that focuses on assisting students with the course selections. The system decides what to do next by learning the actions of the past students. We reused the same dataset generated in chapters 6, 7, and 8 for machine learning and produced a transition system consisting of states, actions, rewards, and transitions.

```

1: Algorithm: Course Recommendation Builder
2: Input: _studentsData: Student Records
3: Definition:
   _listStates //List of total states to be generated
   _stateP, _stateC //State variables
   _actionC //Action variable
4: for each _student  $\in$  _studentsData do
5:   Initialize _semesterNum  $\leftarrow$  0
6:   Initialize _totalProbation  $\leftarrow$  0
7:   Instantiate _stateP(_semesterNum, SATV, SATM, SA1, SA2, SR)
8:   _listSemesters  $\leftarrow$  All semester records registered by _student
   (Each semester record contains: courses registered, status(probation/dismissed),
   and reward value)
9:   for each _semester  $\in$  _listSemesters do
10:    _semesterNum  $\leftarrow$  _semesterNum + 1;
11:    Instantiate _stateC(_semesterNum, MPC, MTC, EC, MPA, MTA, EA)
12:    Instantiate _actionC(Set of courses registered in _semester)
13:    if status(_semester) = probation then
14:      _totalProbation  $\leftarrow$  _totalProbation + 1;
15:      Set Probation Number of _stateC  $\leftarrow$  _totalProbation
16:      Set Reward of _stateC  $\leftarrow$  -1200
17:    else
18:      Set Reward of _stateC  $\leftarrow$  Semester Average
19:    end if
20:    if _listStates contains s' identical to _stateC then

```

```

21:      $\_stateC \leftarrow s'$ 
22:   else
23:     Add  $\_stateC$  to  $\_listStates$ 
24:   end if
25:   Add Transition  $t \leftarrow (\_actionC, \_stateC)$  to  $\_stateP$ 
26:    $\_stateP \leftarrow \_stateC$ 
27: end for
28: Run ValueIteration( $\_listStates$ );

```

We formalized our environment into a Markov Decision Process and used concepts from value iteration algorithm to build a decision making policy that is able to recommend the best set of courses. In order to evaluate our algorithm, we generated and analyzed the best traces and worst traces for students at risk and noticed that the policy was improving after each iteration. In addition, the system is also flexible enough to provide alternative set of courses in case students cannot register for the best recommended courses. Finally, the recommendation system will be rebuilt after each semester as more training samples will be available to be trained on.

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

10.1 Conclusion

In this thesis, we presented a new course recommendation and an early warning system for undergraduate students using machine learning.

Our contribution included the following:

- **Semester-Based Predictions:** Rather than predicting the students' outcome only once, we proposed a system that accumulates the grades of each semester alongside their school/SAT grades. It was important to address this problem, since some students may arrive at university with good grades and background but perform poorly after each semester. Hence, it was important to have semester-based predictions that can warn a student who might be at risk at the end of each semester.
- **Course Recommendation for Undergraduate Students:** Some of the previous works approached the course recommendation system in a way that does not apply to undergraduate students. These methods work best if the student has the flexibility to choose courses from existing lists, such as those provided by online course platforms. However, with regards to a Bachelor's degree, the student has a strict set of courses and a path to follow in order to graduate safely. Hence, we approached this case by applying reinforcement

learning to build a policy(recommender) that learns from the past students and accordingly helps the new students with their upcoming courses selection.

10.2 Future Work

The work presented in this thesis opens the door to many future work and experiments which include:

1. **Freshman/Majorless:** Our work targeted students pursuing a Bachelor's degree for a specific major. However, it would be challenging to experiment on majorless and freshman students since the goal is to address as many students as possible.
2. **Software Integration:** This work should eventually be integrated with the university's database system and website. Students should receive warnings and course recommendations on their dashboard.

APPENDIX A

ABBREVIATIONS

MAE	Mean Square Error
AUC	Area Under the Curve
ROC	Receiver Operating Characteristic
RMSE	Root Mean Square Error
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
FPR	False Positive Rate
TPR	True Positive Rate
SMOTE	Synthetic Minority Oversampling Technique
AdaBoost	Adaptive Boosting
IG	Information Gain

RL	Reinforcement Learning
MDP	Markov Decision Process
SATV	SAT Verbal
SATM	SAT Mathematics
SA1	Class Average 11th Grade
SA2	Class Average 12th Grade
SR	School Rank
SN	Semester Number
MPA	Major Programming Average
MPC	Major Programming Courses Taken
MTA	Major Theory Average
MTC	Major Theory Courses Taken
EA	Elective Average
EC	Elective Courses Taken
R	Reward
PN	Probation Number

REFERENCES

- [1] Rovira S., Puertas E., and Igual L. Data-driven system to predict academic grades and dropout. *PLoS ONE*, 12, 2017.
- [2] Himabindu L., Everaldo A., Carl S., David M., Nasir B. Rayid G., and Kecia L. A machine learning framework to identify students at risk of adverse academic outcomes. 2015.
- [3] Nicolae-Bogdan S., Rasmus H., Christian I., and Stephen A. High-school dropout prediction using machine learning: A danish large-scale study. *ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium)*, 2015.
- [4] Youtube - <https://www.youtube.org/>.
- [5] Netflix -<https://www.netflix.com/>.
- [6] Amazon - <https://www.amazon.com/>.
- [7] Edx - <https://www.edx.org/>.
- [8] Carlos A. Gomez-Uribe and Neil H. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.* 6, 4, Article 13, 2015.
- [9] Fábio O. Masters' courses recommendation: Exploring collaborative filtering and singular value decomposition with student profiling, thesis, tecnico lisboa. 2014.

- [10] K. Bauman and A. Tuzhilin. Recommending learning materials to students by identifying their knowledge gaps. *RecSys Poster Proceedings*, 2014.
- [11] S. Ray and A. Sharma. A collaborative filtering based approach for recommending elective courses. *In: Dua S., Sahni S., Goyal D.P. (eds) Information Intelligence, Systems, Technology and Management. ICISTM 2011. Communications in Computer and Information Science, vol 141. Springer, Berlin, Heidelberg, 2014.*
- [12] Brett Lantz. *Machine Learning with R*. Packt Publishing, 2013.
- [13] Christopher D. Manning; Prabhakar R. Hinrich S. *Introduction to Information Retrieval*, chapter 7. Cambridge University Press, 2009.
- [14] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16, 2002.
- [15] Yoav F. and R. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, 1999.
- [16] Antony S. Thanamani B. Azhagusundari. Feature selection based on information gain. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2, 2013.
- [17] Richard E. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*. 6., 1957.

- [18] Elena P., Irina R., and Rina D. Value iteration and policy iteration algorithms for markov decision problem. 1996.