M. DOOST

ATILIM UNIVERSITY 2023

PREDICTING STUDENTS' ACADEMIC PERFORMANCE USING MACHINE LEARNING TECHNIQUES

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

MIRWAIS DOOST

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

SEPTEMBER 2023

PREDICTING STUDENTS' ACADEMIC PERFORMANCE
USING MACHINE LEARNING TECHNIQUES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

MIRWAIS DOOST

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

SEPTEMBER 2023

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

____

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Software Engineering, Atılım University.**

____

Prof. Dr. Ali Yazıcı
Head of Department

This is to certify that we have read the thesis PREDICTING STUDENTS' ACADEMIC PERFORMANCE USING MACHINE LEARNING TECHNIQUES submitted by MIRWAIS DOOST and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

____

Assoc. Prof. Dr. Cansu Çiğdem EKİN
Supervisor

Examining Committee Members:

Asst. Prof. Dr. Güzin TÜRKMEN
Computer Eng. Department, Atilim University ___________________________

Assoc. Prof. Dr. Cansu Çiğdem EKİN
Computer Eng. Department, Atilim University ___________________________

Assoc. Prof. Dr. Elif POLAT HOPCAN
Department of Computer Education and
Instructional Technologies, Istanbul University ___________________________

Date: 11.09.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Mirwais Doost

Signature:

ABSTRACT

PREDICTING STUDENT ACADEMIC PERFORANCE USING MACHINE LEARNING TECHNIQUES

Doost, Mirwais

MSc., Department of Software Engineering

Supervisor: Assoc. Prof. Dr. Cansu Çiğdem EKİN

September 2023, 94 pages

Recently education sectors have the most attraction from people all around the world and this make it more valuable for whom wants to invest in this sector and make income. Therefore, there are too much efforts to make this domain more stable. Students are the largest stakeholders in this area, in this reason they need more attention in educational settings. All universities are trying to make their better quality for achieving their students' satisfaction. Because the quality of education is depended on success rate of students and capability of the institute for retaining its students, predicting student's performance is a way to identify the students who are at risk of failure, so management can make decision for improving students' performance. These analyzes can be done EDM (Educational Data Mining) a subset of ML (Machine Learning) that is able to discover very large datasets for producing valuable results. The main purpose of this study is to predict students' academic performance using most accurate data mining algorithms and determine the factors which influence the performance of computer engineering students in undergraduate level. The student academic performances were analyzed in three different aspects as Final Grades, Study Duration, and Next Term Course Grade. Our results shows that SVM (Support Vector Machine) and DT (Decision Tree) are the two best ML algorithms and also, we determined that only Final Course grades are the most valuable factors in prediction.

Keywords: Machine Learning, Data Mining, Educational Data Mining, Students' Success Rate.

ÖZ

MAKİNE ÖĞRENME TEKNİKLERİNİ KULLANARAK ÖĞRENCİNİN AKADEMİK PERFORMANSININ TAHMİN EDİLMESİ

Doost, Mirwais
Yüksek Lisans, Yazılım Mühendisliği Bölümü
Danışman: Assoc. Prof. Dr. Cansu Çiğdem EKİN

Eylül 2023, 94 sayfa

Son dönemde eğitim sektörü, dünya genelinde insanların en fazla ilgisini çeken sektörlerden biri haline gelmiş ve bu, bu sektöre yatırım yapmak ve gelir elde etmek isteyenler için daha değerli hale gelmiştir. Bu nedenle, bu alanı daha istikrarlı hale getirmek için büyük çaba harcanmaktadır. Öğrenciler, bu alandaki en büyük paydaşlardır ve bu nedenle eğitimde daha fazla dikkat gerektirirler. Tüm üniversiteler, öğrencilerinin memnuniyetini sağlamak ve eğitim kalitesini artırmak için çaba sarf etmektedir. Çünkü eğitim kalitesi, öğrencilerin başarı oranı ve kurumun öğrencilerini elinde tutma yeteneğine bağlıdır. Öğrenci performansını tahmin etmek, başarısızlık riski taşıyan öğrencileri tanımlamanın bir yolu olduğu için yönetim, öğrenci performansını artırmak için kararlar alabilir. Bu analizler, Eğitim Veri Madenciliği (EDM) olarak adlandırılan, sonuçlar üretmek için çok büyük veri kümelerini keşfedebilen Makine Öğrenimi (ML) alt kümesi aracılığıyla gerçekleştirilebilir. Bu çalışmanın ana amacı, en uygun veri madenciliği algoritmalarını kullanarak öğrenci akademik performansını tahmin etmek ve lisans düzeyinde bilgisayar mühendisliği öğrencilerinin performansını etkileyen faktörleri belirlemektir. Öğrenci akademik performansı, Final Notu, Çalışma Süresi ve Bir Sonraki Dönem Ders Notu olmak üzere üç farklı açıdan analiz edilmiştir. Sonuçlarımız, Destek Vektör Makinesi (SVM) ve Karar Ağacı (DT) gibi iki en iyi ML algoritmasının olduğunu göstermektedir ve ayrıca sadece Final Notunun tahminde en değerli faktör olduğunu göstermiştir.

Anahtar Kelimeler: Makine Öğrenimi, Veri Madenciliği, Eğitim Veri Madenciliği, Öğrenci Başarı Oranı

To

My Family,

Sometimes I forgot to tell you how much I love you. Thank you for being in my life. You are special to me. Big Hug.

ACKNOWLEDGMENTS

I express my sincere gratitude and thanks to my advisor Assoc. Prof. Dr. *Cansu Çiğdem EKİN* that supported me and encouraged me to complete this study. Without her assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. I would like to thank you very much for your support and understanding over this time.

I would also thank all my family members, who have been giving me support and motivation throughout this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

ML	:	Machine Learning
DM	:	Data Mining
EDM	:	Educational Data Mining
DT	:	Decision Tree
NV	:	Naïve Bayes
SVM	:	Support Vector Machine
LR	:	Linear Regression
RF	:	Random Forest
RMSE	:	Rout Mean Squared Error
MAE	:	Mean Absolut Error
KDD	:	Knowledge Discovery in Database

CHAPTER 1

INTRODUCTION

Storing large amounts of data both public and private by several companies in computer systems, has forced the development of new technologies for data analysis and management. Data mining with different techniques comes out for the purpose of knowledge discovery in this domain. While, these techniques can be used in different sectors, such as educational environment which originates from the traditional data analysis methods and have the ability to process and analyze large amounts of data. In the context of education, educational data mining is a method which is used for analyzing students' information stored in the databases to increase the performance of students and academic quality [13].

Higher Education which includes diploma, undergraduate, master programs, doctoral, and professional programs is considered after education. For keeping both quality of academic and students' performance, universities trying to increase their abilities to analyze the students' information for predicting various outcomes generated from the students using different data mining techniques [14].

The goal of all academic institutions is to increase the student retention. Students' consequences are important for student, academic and administrative staffs. Therefore, performance monitoring plays a vital rule to control this condition and helps students, teachers, and administrator to make better decisions in this aspect. These decisions can be made by use of new programming skills and use of AI (Artificial Intelligence) to find and generate the factors which have significant impacts on education [15], [16].

In educational environment predicting students' performance is essential for improving the student abilities. And students' performance is based on some factors like social, personal, psychological, and other environmental variables. Data mining techniques can help the educational domain to discover these factors and other hidden factors and making relationships between large amounts of data for decision making [17].

Since the use of DM is vary in different contexts and EDM is the subset of DM, it has also different usages in educational environment. So, applying EDM in the context of computer engineering in Atilim University help us to predict the performance of students like success rate, GPA, single course grade, and graduation time for making additional academic decisions.

1.1. Statement of the Problem

According to the many researches and investigations in the context of Data Mining on educational environments in the related works (Chapter 2), it states that still there are many different factors like gender, marital status, and home address which affecting the students' performances such as success rate, single course grade, and graduation time. Therefore, based on the result of these researches the following considerations were recognized:

- As we stated before, there are many factors which affect the students' performances and we are supposed to find different factors for different aspects in students' performances. For this reason, we will try to extract meaningful patterns

- While there are two types of output variables (continuous, and discrete) used in our study, we should consider both regression and classification algorithms.

- In order to consider these algorithms, we may compare the different techniques in both algorithms to find the most accurate results.

1.2. Aim of the Study

The main purpose of this study is to predict the students' academic performance including GPA, course grade, and graduation time and find the factors which have significant impact on students' performance using different data mining algorithms and comparing the algorithms to find the most accurate technique that suites our model. Shortly, the main research purpose of this study is explained as below:

- Finding the factors which influence the performance of computer engineering students in undergraduate level.
- Using different data mining techniques to predict the students' academic performance for further management decision making.
- Comparing different data mining algorithms for creating our model.
- Selecting the most accurate data mining algorithm and create our model.

1.3. Significance of the Study

This study can contribute to the literature review on predicting students' academic performance by highlighting the current situation in the field and identifying the research gaps related to the use of ML techniques in this area. The result of this study can help us to:

- ✓ Identify those students who require special attention
- ✓ Minimize the students' failure rate
- ✓ Take acceptable action for upcoming semester examination.
- ✓ And to determine the most effective machine learning methods.

1.4. Research Questions

According to analysis method, the study will answer the following questions:

1. Is it possible to predict four years GPA using course grade success in first 4 terms?
2. Can we determine students' study duration at an early stage (first 2 years) of the degree program using final grades?
3. Can we specify 3rd or 4th year students' course success by determining the relation between first two years courses and next two years courses?
    a. Can we predict course success for the course named "Computer Programming II (CMPE114)" using previous course grade in programming courses (CMPE113)?
    b. Can we predict course success for the course named "Object-Oriented

Programming (CMPE225)" using previous course grade in programming courses (CMPE114)?

c. Can we predict course success for the course named "Data Structures (CMPE226)" using previous course grade in programming courses (CMPE225)?

1.5. The Structure of this thesis' chapters

- Chapter 1: Introduction.
- Chapter 2: Related Works
- Chapter 3: Background of the Study.
- Chapter 4: Methodology.
- Chapter 5: Results.
- Chapter 6: Discussion and Conclusions
- Chapter 7: Software Implementation.

CHAPTER 2

RELATED WORKS:

2.1. Introduction:

Data mining (DM) which is often called Knowledge Discovery in Database (KDD) is a process which is based on computer information system. DM is dedicated for finding huge data repositories, generating information and discovering the knowledge, while the main target in the term of mining is knowledge. DM is used to find the patterns, create the information of hidden relationships, estimate unknown items, and clarify so many kinds of findings which are not easy to produce by computer-based information system.

DM has different usages in different applications such as higher education, crime, healthcare, and accounting that higher education is one of the most valuable applications which researchers have done several researches to discover various patterns in this area. In this study we also concentrate on educational data mining environment and conduct our research in this domain.

2.2. Educational Data Mining (EDM):

It is a modern DM application purposes for knowledge discovery, recommendation, prediction, and decisions-making. It examines the students' learning behaviors in different environments by exploring, analyzing and uncovering the hidden patterns and information of the data and develops methods for better understanding of students with their learning settings.

According to some viewpoints of different researchers, the works in educational data mining is classified as follows and the first category is the most popular methods in educational settings.

- Prediction
    - Classification

    - Regression
    - Density estimation
- Clustering
- Relationship mining
    - Association rule mining
    - Correlation mining
    - Sequential pattern mining
    - Causal data mining
- Distillation of data for human judgment
- Discovery with models

2.3. Key Applications of EDM:

Researchers study different area of educational data mining, including computer supported collaborative learning, individual learning from educational software, and the factors which have direct impacts on students' pass or failure. Across these areas, some of the key applications in educational data mining is defined as below:

Improvement of student models: these models consist of students' characteristics, knowledge, and other information which improve student learning.

Informing domain models: a model that formally describes the domain of instructions based on learning items, concepts, and skills. The reflection of domain model on human cognition like knowledge and problem-solving skills is important to enable student models for accurately predicting knowledge and skills.

Creating reports and alerts for instructors, students, and other stakeholders: using of EDM makes it easier to build teacher tools for tracking their students' progress, making best decisions for their students and advise them for better progressions. It will be applied as will on students and other stakeholders [22], [23], [24], [25].

2.4. Related works on predicting students’ Final Grades:

Quan Hung Do and Jeng-Fung Chen (2013) identified a neuro-fuzzy approach for classifying students into different groups [1] using previous result and other related factors as input variables. They compared the NFC approach with many other classification approaches including SVM, Naïve Bayes, neural network and decision tree classifiers. Therefore, the results identified that the NFC model is the most accurate approach for classifying the students into different groups.

Muluken Alemu Yehuala (2015) created a predictive model for success or failure of undergraduate students at Debre Markos University [2]. In this research Waikato Environment for Knowledge Analysis (WEKA) an open-source software is used as a tool for data preprocessing to generate the most related attributes for predicting the success or failure of students. The results generated by WEKA was based on six top attributes, which are: Sex, Higher Education Entrance Certificate Examination result, number of courses given in a semester, College, Department, and Number of students in a Class. Therefore, based on this research the findings shows that it increases the level of student success and also it can help students to decide about their field of study before enrolling in specific field of study.

Pedro Strecht et al. did a comparative study on Classification and Regression algorithms for modelling students’ academic performance [3]. In this study they used age, sex, marital status, nationality, displaced (whether the student lived outside the Proto district), scholarship, special needs, type of admission, type of student (regular, mobility, extraordinary), status of student (ordinary, employed, athlete …), years of enrolment, delayed courses, type of dedication (full-time, part-time), and debt situation as input variables and they used the approval as target variable for classification algorithm and final grade for regression algorithm. In this model k-Nearest Neighbors (kNN), Random Forest (RF), AdaBoost (AB), Classification and Regression Trees (CART), Support Vector Machines, Naive Bayes (NB) algorithms are used for classification and Ordinary Least Squares (OLS), SVM, CART, kNN, Random Forest, and AdaBoost.R2 (AB.R2) are used for regression. For evaluating the models, they used F1 score and RMSE (Root Mean Squared Error). As there were not enough evidences to determine the significant different between these two analyses, they found that more features like academic goals, personal interests, time management skills, sport activities, and sleep habits can help the analyses to compare the algorithms better.

S. B. Kotsiantis (2011) presented a case study for predicting students’ marks [4] and also developed DSS (Decision Support System) prototype for tutors. In this paper six regression techniques namely Model Trees, Neural Networks, Linear regression, Locally weighted linear regression, Support Vector Machines, and M5rules are used to compare and choose one as the most accurate technique to predict the students’ marks and to be usable in educational supporting tool for tutors. After evaluating the models which using different techniques with different evaluation algorithms namely Mean absolute error, Root mean squared error, Relative absolute error, and root relative squared error, the results identified that M5reules is the most accurate technique for predicting students’ marks and constructing the DSS for tutors.

Mustafa Yağcı (2022) created a model using the most effective machine learning methods to predict the students’ final exam grades and students at high risk of failure with their midterm exam grades as source data [5]. In this study RF (Random Forest), NN (Nearest Neighbors), LR (Logistic Regression), SVM (Support Vector Machine), NB (Naïve Bayes), and kNN (k Nearest Neighbors) machine learning algorithms were used to deploy the model for prediction of students’ final exam grades while the tenfold cross validation is used to evaluate the DM model. In this model midterm, faculty, and department data are used as input variables and final is used as target variable. As a result, students’ midterm exam grades are known as important predictors for predicting students’ final grades. And RF, NN, and SVM are chosen as the most accurate data mining algorithms for this purpose.

Yulei Pang et al. (2017) proposed a model based on SVM (Support Vector Machines) [6] a machine learning algorithm to predict students’ graduations outcomes. They selected students’ demographic information, GPA, and earned credit as most affectable features on predicting students on time graduation, but after their analyses they realized that more features are needed to better prediction of students’ graduation outcomes.

C. Anuradha1 and T. Velmurugan presented an analysis on final year results of under graduate students [18]. They compared different classification algorithms for predicting students' final year results. For this purpose, they used many common demographics and some academic information of students as input variables. After analysis the results showed that Naïve Bayes and Bayes Net classifiers have the highest accuracy rate than other classifiers and can be used for classifying students based on their performances.

Nguyen Thai Nghe1et al. (2007) did a case study for comparing Decision Tree and Bayesian Network algorithms [19] to predict students' academic performance and in specific CGPA for undergraduate and post graduate levels using 2nd year data. They classified students into groups of (fail, fair, good, and very good). The accuracy of classification methods was evaluated by Cross-Validation with 10 folds. The results indicated that Decision Tree classifier is performing much better than Bayesian Network for predicting students CGPA.

Raheela Asif et al (2017). used classification and clustering data mining methods [21] for three objectives respectively, predicting students' performance using marks only, discovering the courses which resulting the good or poor performance of degree program, and determining students' study progression over four academic years. The results showed that they can easily predict students' graduation performance using pre-university marks, and first- and second-year courses marks only, with examining decision trees they could find four courses as good or poor performance indicators for degree program, and finally for study progression they grouped students into high-performing students who achieved high marks during the four years and low-achieving students who acquired low marks during the four years.

2.5. Related works on predicting student graduation time:

Nurafifah Mohammad Suhaimi et al. (2019) studied the impact of different types of factors with different prediction methods for predicting students' graduations time [7]. They investigated some important factors including Gender, GPA, and Marital status related to students, in this investigation the results showed that Gender and Marital status are effective on student performance. While, males study style is different from female and single students are more active than married students. For finding the best model for this purpose, they examined different machine learning methods including DT (Decision Trees), NB (Naive Bayes), NN (Nural Network) and SVM (Support Vector Machines). Finally, the results indicated that NN and SVM are the best methods for predicting students' graduation time.

A research wase done by Liuli Huang et al. (2017) to predict the study duration of university students [8]. Beside students' demographic, pre-college factors, GPA, and.., three more factors: students' in-state and out-of-state address, college major, and student learning type (tutoring and supplemental instructions) which have more impact on graduation time are studied in this paper for accurately prediction of students' graduation time. The results indicated that students with in-state home address are more probably to graduate on time than students with out-of-state home addresses. College major did not have significant impact on graduation time, and finally, the students who participated in learning support resources are most likely to graduate on time than who did not participated.

Mulki Indana Zulfa et al. (2019) purposed a model using SVM algorithm to build a classification model and determine which variable has the most influence on graduation on time [9]. As a result, the semester achievement index becomes the main factor determining whether students will graduate on time or not.

Ida Bagus Adisimakrisna Peling et al. (2017) evaluated other factors which have significant impact on students' graduation time [10]. NB (Naive Bayes) algorithm is used to classify student into grouping of choice graduated first choice, second choice, and do not pass. In this evaluation the features for predicting students' graduation time are entrance path, number of credits. While, entrance path are programs which give the students the eligibility to admit for a bachelor or master program. The results indicated that entrance path had an important impact on graduation time and Naïve Bayes with accuracy of 86.16% and error of 13.84% was determined as a good classifier for predicting study period.

In a study by Masna Wati et al the performance of two ML algorithms Naïve Bayes Classifier and Tree C4.5 is evaluated based on accuracy and precision percentage for both algorithms [14]. In this study they used different factors such as gender, age, culture (some cultural habits which force students to adapt the higher education environment, these habits can be derived by the place of birth), high school status, high school major, first four semesters' GPA, participation levels (it can be active if participated in an organization while taking the study and can be passive if never participated in any organization while taking the study), and graduate time for predicting students' academic performance. Therefore, the results of this study shows that Naïve Bayes Classifier is better with average accuracy of 69.10 % and average precision of 62.30 % while Tree C4.5 has the average accuracy of 62.91 % and average precision of 58.82 %. In this study, age, student's cultural habit, and student participation level are known as factors that have some impact on student graduation time.

2.6. Related works on predicting students' future course grade:

Agoritsa Polyzou and George Karypis (2016) tried to develop methods for predicting the students' courses which are not taken by them yet [11]. In this study, sparse linear models and low-rank matrix factorizations are used for this purpose. They did the analysis in two separated parts one for student specific regression and another for course specific regression. As a result, the course-specific regression which uses the previous course grades was selected as more accurate method for predicting students' future course grades.

Mack Sweeney et al. (2015) developed a system to predict the next term course grade of students using historical grade data [12]. They experimented the CF (Collaborative Filtering) and MF (Matrix Factorization) methods for finding the most accurate model for predicting future courses' grades. Finally, they determined that MF model is the most appropriate method for this purpose while it can be used for new and existing courses.

Amirah Mohamed Shahiria et al (2015). proposed a systematical literature review for predicting students' performance using data mining techniques [20]. They focused on how the data mining techniques can help to determine the most important attributes in students' data and which technique is the most accurate for this purpose. In this research, Cumulative Grade Point Average (CGPA), internal assessment classified as assignment mark, quizzes, lab work, class test, and attendance, students' demographic (gender, age, family background, and disability), external assessment (the mark for a specific subject obtained in final exam), and psychometric factors (student interest, study behavior, engage time, and family support) are investigated as factors which influence students' performance. They also compared Neural Network, Decision Tree, Support Vector Machine, K-Nearest Nighbor, and Naïve Bayes data mining algorithms to find the most accurate methods for predicting students' performance. The results indicated that Neural Network with (98%) accuracy and Decision Tree with (91%) accuracy are the most accurate and suitable data mining algorithms, and CGPA, internal assessment, and external assessment which is the obtained mark in final examination are the most significant factors for predicting students' performance.

2.7. Summary:

There are many researches conducted on educational data mining which are using different ML algorithms for finding various factors to predict students' academic performance. As a result, most of the above studies related to the prediction of students' academic performance which includes predicting final grades (CGPA), study duration, and specific course grade prediction examined different ML algorithms for these aspects. So, the analysis results show that some factors such as gender, marital status, home address, learning type, and midterm exam scores have the significant influence on predicting students' academic performance. Five ML algorithms respectively SVM (Support Victor Machin), NB (Naïve Bayes), DT (Decision Tree), LR (Linear Regression), and RF (Random Forest) are known as the most suitable algorithms for the prediction of students' academic performance.

CHAPTER 3

BACKGROUND OF THE STUDY

3.1. Data Analysis

In this case, after collection of the dataset and applying some manual processing on it for predicting the students' academic performance we have examined some ML techniques and chose the most suitable algorithm for this purpose. For our purpose, we should process both continuous and discrete data to predict the students' academic performance. Thus, we are going to use classification and regression techniques in this study. In the section 3.2 the selected ML learning algorithms are described, and in the section 3.3 the performance metrics are explained.

- **Continuous data:** specific values which are countable.
- **Discrete data:** infinite number of fractional values between any two values which are measurable.
- **Regression:** the output values are based on input features.
- **Classification:** separates one class into positive and negative values.

3.2. Popular ML Algorithms in Prediction of Student Academic Performance

Based on our researches the three popular ML techniques (DT, NB, SVM) were considered to analyze the students' academic performance which are explained more as below:

3.2.1. DT (Decision Tree):

DT is a powerful and popular algorithm that can be used for both classification and Regression problems [7]. It follows the structure of a tree as it splits the independent inputs in similar parts and generates the corresponding output [1]. DT is structure is very simple because its processes are converted in a set of IF-THEN rules [20] and

DT is one of the ML techniques which is preferred to use in more papers in more papers of this study to its simplicity and good performance [19], [21]. Figure I, shows the structure of DT algorithm.

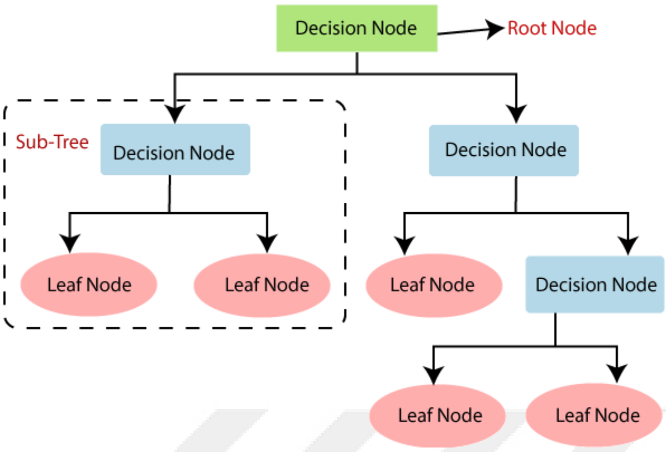


Figure 3.1 Illustration of the structure of the DT classifier

$$E(S) = \sum_{i=1}^{\infty} -p_i log_2 \, p_i \tag{1}$$

S: the dataset
c: category of data
pi: Percentage of class i
in a node of state S

3.2.2. NB (Naïve Bayes):

Naïve Bayes is one of the popular classifiers in Bayesian classifiers [18] which is based on Bayes theorem. NB only consider the general features of a variable in a class for prediction process instead of assuming the specific features for each class [17]. According its high accuracy rate in the prediction results, it is considered as an efficient algorithm for very large quantities. Figure II shows the structure of NB algorithm.

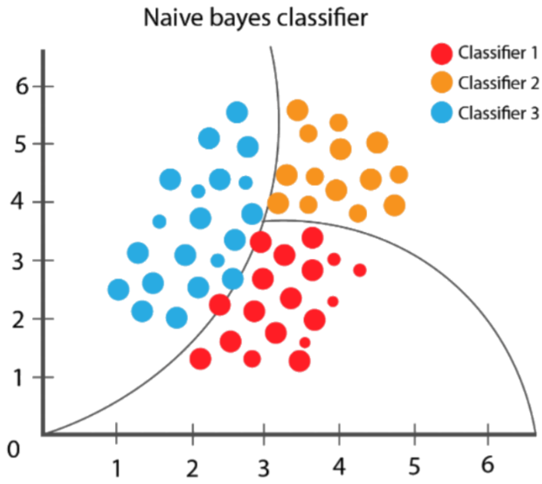


Figure 3.2 Illustration of the structure of the NB classifier

$$P(B|A) = \frac{P(B|A)\,P(A)}{P(B)} \tag{2}$$

A: dependent variable
B: independent

3.2.3. SVM (Support Vector Machine):

Support Vector Machine is one of the famous ML algorithms used to solve both classification and regression problems [6]. This algorithm builds a hyper-plane with the purpose of maximizing the distance between the nearest data points and hyper-plane. SVM is suitable for small datasets as it takes a long time for training the large datasets [7]. For this study, more papers used SVM algorithm to predict the students' academic performance [5], [9]. Figure III illustrates the structure of SVM algorithm.

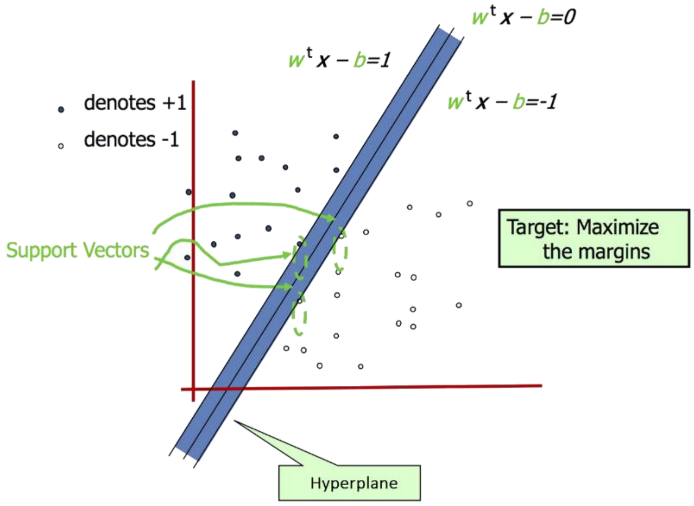


Figure 3.3 Structure of SVM algorithm

Marginal Distance: (3)

$$W * X - b = 0$$

W: the distance from hyperplane to any points
X: the data points
b: the intercept

3.2.4. LR (Linear Regression):

Linear Regression is known as a simplest statistical technique for finding the best-fitting linear relationship between the target values and its predictors [4]. Hence it deals with continuous data, it is not recommended to solve the classification problems which are using discrete data.

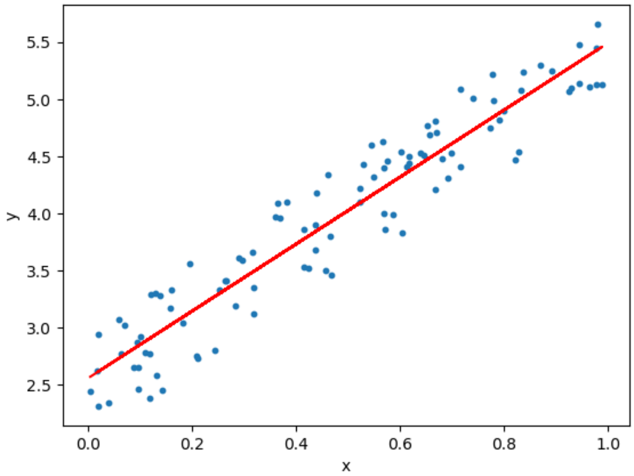


Figure 3.4 Structure of Linear Regression algorithm

$$Y_i = f(X_i, \beta) + e_i \tag{4}$$

Y_i = dependent variable
f = function
X_i = independent variable
β = unknown parameters
e_i = error terms

3.2.5. RF (Random Forest):

Random Forest is one of the most powerful ML algorithms which can be used to solve both classification ad regression problems [26]. This algorithm produces many decision trees with random attribute values. For taking the final result, Random Forest get the majority value from results generated by decision trees. The level of accuracy for this algorithm is related on the number of produced trees.

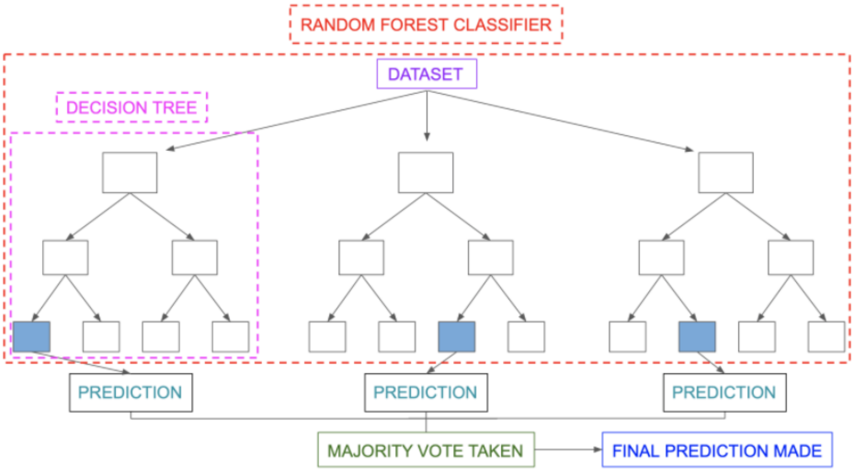


Figure 3.5 Structure of Random Forest Algorithm

3.3. Performance Metrics

In this section, all the evaluation metrics which are used in this study and applied in "CHAPTER 5" are explained in detail.

3.3.1. Confusion Matrix:

Confusion Matrix is used to measure the accuracy and precession of a data mining algorithm which can be illustrated by confusion matrix [14]. It shows the information number of correct and incorrect prediction in positive and negative instance and for this purpose it uses True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) which are explained as below. In this study we used confusion matrix for our classification model for predicting students' graduation time.

- **True Positive (TP):** True Positive (TP) is a metric which shows the correctly true classes predicted.

- **False Positive (FP):** False Positive (FP) measures the wrong true classes which are actually false.

- **True Negative (TN):** True Negative (TN) shows the negative classes as negative predicted.

- **False Negative (FN):** False Negative (FN) is the wrong prediction of negative classes which are actually true.

3.3.2. Accuracy:

It is the proportion of the correct predictions (TP + TN) to the total number of instances (TP + TN + FP + FN) and it is suitable only for classification models [5]. it takes a value between 0 to 1 while the highest rate shows the better accuracy of the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

3.3.3. Precision:

Precision is used only in classification models and it is the proportion of the number of positive instances that are correctly classified to the total number of instances that are predicted positive [5]. it takes a value between 0 to 1 while the highest rate shows the better accuracy of the model.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

3.3.4. Recall:

Recall is also usable for only classification models and it is the proportion of the correctly classified number of positive instances to the number of all instances whose actual class is positive [5]. It is also known as true positive rate. it takes a value between 0 to 1 while the highest rate shows the better accuracy of the model.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

3.3.5. F1 score:

F1 score is used only in classification models. It is the combination of the precision and recall score of the model. This metric computes the number of correct predictions made by the model across the entire dataset. it takes a value between 0 to 1 while the highest rate shows the better accuracy of the model.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

3.3.6. RMSE (Root Meas Squar Error):

RMSE focuses on the detection of big errors than detecting small errors in the model and it is suitable for both classification and regression models [12]. it takes a value between 0 to 1 while the lowest rate shows the better accuracy of the model.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{g}_i - g_i)^2}{N}} \tag{9}$$

3.3.7. MAE (Mean Absolute Error):

MAE is also used to measure the accuracy of both classification and regression models. MAE takes the absolute difference between actual values and predicted values by average it across the dataset [12]. it takes a value between 0 to 1 while the lowest rate shows the better accuracy of the model.

$$MAE = \frac{\sum_{i=1}^{N} |\hat{g}_i - g_i|}{N} \tag{10}$$

3.3.8. $\mathbf{R^2}$ score:

$\mathbf{R^2}$ is suitable for both classification and regression models. It shows the goodness of fit of a model. It takes a value between 0 to 1 while it indicates 1 it means that the model is perfectly fitted.

$$\mathbf{R^2} = 1 - \frac{RSS}{TSS} \tag{11}$$

RSS = sum of squares of residuals
TSS = total sum of squares

CHAPTER 4

METHODOLOGY

4.1. Introduction

The goal of this thesis is to find out the usage of ML techniques in the educational area specific for computer engineering in Atilim University for predicting students' academic performance. This chapter explains the steps in research process for answering the main research questions which are formed in 'Introduction' part of this thesis. So, the research questions in this study will be answered through the main five ML techniques which are described in 'Background of the Study Chapter' are chosen as the most suitable algorithms for this case and these algorithms will be examined in the 'Results Chapter'.

- DT (Decision Tree)
- SVM (Support Vector Machine)
- NB (Naïve Bayes)
- LR (Linear Regression)
- RF (Random Forest)

4.2. Design of the Study

The main steps for research process of this study are illustrated in Figure 4.1:

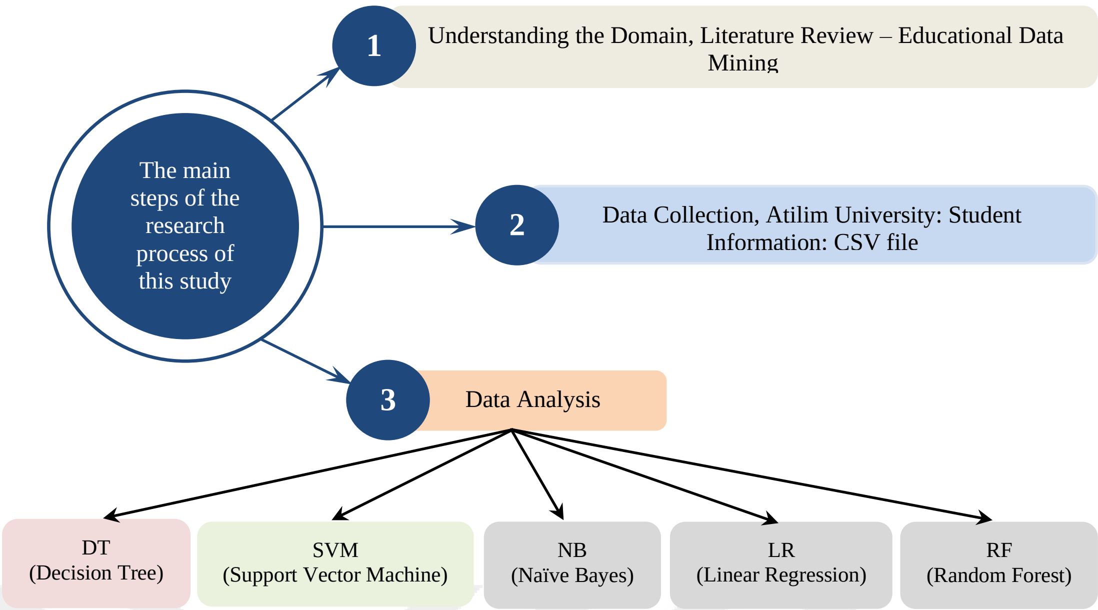


Figure 4.1 The main steps of research process

4.2.1. Understanding the Domain

At the beginning, a big effort on the literature wase done to define the pathways used to describe EDM in the literature review and evaluating the techniques and their results used for analysis regarding data mining in educational context. Thereafter, by the consulting of a domain expert proper keywords were determined for extracting the related articles within the literature as follows: educational data mining, data mining in higher education, and predicting undergraduate students' academic performance.

4.2.2. Data Collection/Data Extraction

For this case, the data is prepared by computer center in Atilim University. It consists the complete transcript data of Computer Engineering students which are demographic and functional data.

Demographic Data such as: gender, age, nationality, and residency

Functional Data such as: GPA, CGPA, scholarship, and registration reason

In the prepared dataset, there are number of 175 graduated students from 2012 to 2021 which includes all the elective and mandatory courses for each semester that for this study we considered to use only the mandatory courses for analyzing the data, the reason for this consideration is unidentifiability of number of elective courses which are going to be chose by students and it may cause pure performance of our model. There are three types of curriculums for the selected range in the dataset that is illustrated in the Table 4.1. The number of mandatory courses is varied from curriculum to curriculum. So, for this purpose we decided to choose the mandatory courses which are used among all curriculums.

Table 4.1 shows mandatory courses for three curriculums from 2013 - 2022

FALL 2013 - 2014	
1st Semester	
ENG101	English For Academic Purposes I
PHYS101	General Physics I
CMPE109	Fundamentals of Computing
CHE105	General Chemistry
CMPE113	Computer Programming I
MATH151	Calculus I
2nd Semester	
ENG102	English for Academic Purposes II
PHYS102	General Physics II
MATH152	Calculus II
CMPE114	Computer Programming II
CMPE134	Fundamentals of Electronic Components
3rd Semester	
ENG201	English for Academic Purposes III
EE235	Digital Electronics
MATH275	Linear Algebra
EE203	Digital Circuits and Systems
CMPE251	Discrete Computational Structures
CMPE225	Object-Oriented Programming
4th Semester	
MATH276	Differential Equations
IE220	Probability and Statistics
ENG204	Report Writing Skills
CMPE226	Data Structures
CMPE236	Introduction to Microprocessors and Microcontrollers
5th Semester	
CMPE323	Algorithms
CMPE331	Computer Architecture and Organization
CMPE341	Database Design and Management
CMPE399	Summer Practice I
TURK401	Turkish Language I

CMPE325	Study of Programming Languages
6th Semester	
IE305	Engineering Economy
CMPE326	Formal Languages and Automata
MATH380	Numerical Methods for Engineers
TURK402	Turkish Language II
SE346	Software Engineering
CMPE334	Computer Networks
7th Semester	
CMPE493	Project Orientation
CMPE431	Operating Systems
CMPE499	Summer Practice II
HIST101	Principles of Atatürk and History of Turkish Revolution I
8th Semester	
CMPE494	Senior Project
HIST102	Principles of Atatürk and History of Turkish Revolution II
FALL 2016 - 2017	
1st Semester	
MATH151	Calculus I
PHYS101	General Physics I
CMPE109	Fundamentals of Computing
CMPE113	Computer Programming I
HIST111	Principles of Atatürk and History of Turkish Revolution I (in English)
CHE105	General Chemistry
ENG101	English For Academic Purposes I
2nd Semester	
HIST112	Principles of Atatürk and History of Turkish Revolution II (in English)
CMPE134	Fundamentals of Electronic Components
CMPE114	Computer Programming II
HIST221	History of Civilization
ENG102	English for Academic Purposes II
PHYS102	General Physics II
MATH152	Calculus II
3rd Semester	
ENG201	English for Academic Purposes III
MATH275	Linear Algebra
EE203	Digital Circuits and Systems
CMPE251	Discrete Computational Structures
CMPE225	Object-Oriented Programming
4th Semester	
MATH276	Differential Equations
CMPE236	Introduction to Microprocessors and Microcontrollers

CMPE226	Data Structures
IE220	Probability and Statistics
ENG202	English for Academic Purposes IV
5th Semester	
CMPE399	Summer Practice I
ENG301	English for Occupational Purposes I
CMPE331	Computer Architecture and Organization
CMPE341	Database Design and Management
CMPE323	Algorithms
CMPE325	Study of Programming Languages
6th Semester	
SE346	Software Engineering
ENG302	English for Occupational Purposes II
CMPE326	Formal Languages and Automata
CMPE334	Computer Networks
MATH380	Numerical Methods for Engineers
7th Semester	
CMPE499	Summer Practice II
CMPE493	Project Orientation
TURK401	Turkish Language I
8t Semester	
TURK402	Turkish Language II
CMPE431	Operating Systems
CMPE494	Senior Project
FALL 2021 - 2022	
1st Semester	
MATH151	Calculus I
ENG101	English For Academic Purposes I
CMPE113	Computer Programming I
HIST111	Principles of Atatürk and History of Turkish Revolution I (in English)
CMPE109	Fundamentals of Computing
CHE105	General Chemistry
KRY111	Career Planning
PHYS101	General Physics I
2nd Semester	
ENG102	English for Academic Purposes II
HIST112	Principles of Atatürk and History of Turkish Revolution II (in English)
HIST221	History of Civilization
MATH152	Calculus II
CMPE134	Fundamentals of Electronic Components
PHYS102	General Physics II
CMPE114	Computer Programming II
3rd Semester	
CMPE225	Object-Oriented Programming

MATH275	Linear Algebra
CMPE251	Discrete Computational Structures
ENG201	English for Academic Purposes III
EE203	Digital Circuits and Systems
4th Semester	
MATH276	Differential Equations
ENG202	English for Academic Purposes IV
IE220	Probability and Statistics
CMPE226	Data Structures
CMPE236	Introduction to Microprocessors and Microcontrollers
5th Semester	
CMPE323	Algorithms
CMPE325	Study of Programming Languages
CMPE399	Summer Practice I
CMPE341	Database Design and Management
ENG301	English for Occupational Purposes I
CMPE331	Computer Architecture and Organization
6th Semester	
MATH380	Numerical Methods for Engineers
SE346	Software Engineering
ENG302	English for Occupational Purposes II
CMPE326	Formal Languages and Automata
CMPE334	Computer Networks
7th Semester	
CMPE493	Project Orientation
TURK401	Turkish Language I
CMPE499	Summer Practice II
8th Semester	
CMPE431	Operating Systems
CMPE494	Senior Project
TURK402	Turkish Language II

4.2.2.1. Attribute Definition for the Selected Dataset

The initial dataset has a number of columns which are belong to students' personal information and students' academic information that only some of the columns would be selected for this purpose. All the attributes are explained in Table 4.2.

Table 4.2 Describes the dataset attributes

Column Name	Data Type	Description
STD_ID	text	A unique value for each student given by the university
COURSE NAME	text	A specific name for each

		course
COURSE CODE	text	Code to identify any course instead of its name
COURSE LETTER GRADE	text	The grade which will be taken by any student and identify the level of student according to specific course ()
COURSE_CATEGORY	text	All courses are categorized into Elective and Mandatory (S and Z)
MUFREDAT	text	
SECTION	text	Initializes the course level
SECTIONOGRCOUNT	numeric	Number of students in each section
SEASON ID	numeric	A specific ID for each academic season
SEASON NAME	text	Name of the academic season (Fall, Spring, Summer)
GPA	numeric	The final mark for specific semester by any student
CGPA	numeric	The average of current semester marks with the previous semester's marks
DIPLOMA_NOT	numeric	The average of all semester's marks
GENDER	text	It identifies the sex of students (Male, and Female)
YEAR	numeric	Student birth year
NATIONALITY	text	Student nationality
CITY_CODE	numeric	A specific code for each city where students come from
REGISTRATION YEAR	text	The students' education start time
GRADUATION YEAR	text	The students' education end time
REGISTER_REASON	text	defines the normal students or transferred students
SCHOLARSHIP	text	Any given discount for any students
OSYMPUANI	numeric	

4.2.2.2. Attribute Selection for the Purpose of This Study

According to our researches we decided to choose some main attributes for answer those research questions which are mentioned in the “CHAPTER 1” and these attributes are listed in Table 4.3.

Table 4.3 Extracted columns from the dataset for predicting students’ academic performance

Column Name	Data Type	Description
STD_ID	text	A unique value for each student given by the university
COURSE CODE	text	code to identify any course instead of its name
COURSE LETTER GRADE	text	The grade which will be taken by any student and identify the level of student according to specific course ()
COURSE_CATEGORY	text	All courses are categorized into Elective and Mandatory (S and Z)
GPA	numeric	The final mark for specific semester by any student
DIPLOMA_NOT	numeric	The average of all semesters’ marks
GENDER	text	It identifies the sex of students (Male, and Female)
YEAR	numeric	Student birth year
CITY_CODE	numeric	A specific code for each city where students come from
SCHOLARSHIP	text	Amount of discount consider for a student
REGISTRATIO_REASON	text	The entrance path (with exam or without exam)

4.2.2.3. Applying Data Aggregation on the Selected Dataset

According to the need of this study we started to do some preprocessing on the existing dataset to prepare the data for analyzing step. For the initial step of dataset cleaning some manual preprocessing steps are done using Microsoft Excel as below:

1. First of all, all the column names and column data in the dataset were translated into English.
2. Because the existing [STUDENT_ID] is encrypted, so we decided to separate these data and get the unique part and store in a new column and named it [STD_ID].
3. According to the rule of higher educations and Atilim University rules we also separated the academic year and academic season from [SEASON ID] column and stored them separately in two new columns [SEASON_ID] and [SEASON_YEAR]. Though, we followed the below structure for this calculation in Table 4.4.

Table 4.4 Illustrates the academic season name with their respective ID

ACADEMIC YEAR FOR (2015-2016)		
YEAR	**SEASON**	**SESON ID**
2015	FALL	1
2016	SPRING	2
2016	SUMMER	3

4. At the first we calculated the academic year from [SEASON ID] but later on we recognized that all of the years in this column are not constant. Thus, we chose the [SEASON NAME] column for getting the academic year and so the data was fully constant for calculating the academic year.
5. All the data in [CITY_CODE] column is converted in city name and stored in a new column named [CITY].
6. The same for [SEASON ID] column we also separated [REGISTRATION YEAR] into [REGISTRATION_YEAR] and [REGISTRATION_SEASON_ID] and stored them into two new columns.
7. In this step we calculated the [YEAR] for finding the semester number which student is currently enrolled. For this calculation we deducted the [REGISTRATION_YEAR] from [SEASON_YEAR] and stored the result in new column [YEAR].
8. Then we calculated the [SEASON] again for finding the semester number which student is currently enrolled. For this calculation we divided [SEAON_ID] by [REGISTRATION_SEAON_ID] and stored the result in

new column [SEASON]

9. Now that we have calculated [YEAR] and [SEASON], we can simply find the semester number which student is currently enrolled, by multiplying [YEAR] to 2 and add [SEASON] to the result and store it in new column [SEMESTER].
10. Age of the student is also found by deducting the [YEAR] which is belong to the student's birth year from [GRADUATION YEAR] and stored it in new column [AGE].
11. The COURSE CATEGORY for course named "Genel Kimya" in column [COURSE NAME] has been changed from S (Elective) to Z (Mandatory).
12. All courses related to COMPE in [COURSE_CODE] column are converted in CMPE.

After making the above changes on the dataset, some new attributes are added and the selected attributes for this study are redefined which are shown in Table 4.5:

Table 4.5 shows the new column names for the selected attributes

Column Name	**Data Type**	**Description**
STD_ID	text	A unique value for each student given by the university
COURSE CODE	text	code to identify any course instead of its name
COURSE LETTER GRADE	text	The grade which will be taken by any student and identify the level of student according to specific course ()
COURSE_CATEGORY	text	All courses are categorized into Elective and Mandatory (S and Z)
GPA	numeric	The final mark for specific semester by any student
DIPLOMA_NOT	numeric	The average of all semesters' marks
GENDER	text	It identifies the sex of students (Male, and Female)
AGE	numeric	Student birth year
CITY	text	Name of the city based on CITY_CODE
SEMESTER	numeric	The current semester number

4.2.3. Data Preprocessing

In this study python with Jupiter Notebook is chosen for data preprocessing. Therefore, for applying the selected ML algorithms on the dataset to pursue our target for predicting students' academic performance, we should consider the preparation of three separated datasets using the existing dataset and one dataset from the course curriculum as following:

- One dataset for preparing student information which will contain ["STD_ID", "AGE", "CITY"].
- Another dataset for predicting both students' Final Grades and students' next term course grades which will contain ["STD_ID", "DIPLOMA_NOT", "COURSE_CODE", "COURSE_LETTER_GRADE"].
- Another for predicting students' graduation time which will contain ["STD_ID", "SEMESTER", "GPA", "DIPLOMA_NOT", "DURATION"].
- And another dataset that is extracted from course curriculum is necessary for recommending related course of a selected course and predicting the related course grade. Below are the steps for preparing the datasets:

1. At the first step all ML libraries for the purpose of preprocessing of the dataset is imported. As shown in Figure 4.2:

Import the desired libraries for preprocessing the dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

Figure 4.2 Python libraries for data preprocessing

2. Next the data is filtered by Z (mandatory courses) in column [COURSE_CATEGORY], thus we need only the mandatory courses which are taken by students.
3. Then all the data is again filtered by column [SEMESTER] and chose only

the data between semesters 1 to 4.

4. Removing all whitespaces from the course codes in the column [COURSE_CODE] to make all course codes in the same form, because we need to categorize the data by [COURSE_CODE] for next purposes. And it is illustrated in Figure 4.3:

Import the dataset and perform some initial precesses to make the dataset ready for next steps

In [2]:
```
ds = pd.read_csv("DATASET.csv")
ds=ds[ds.COURSE_CATEGORY=='Z']
ds=ds[ds.SEMESTER<=4]
ds=ds[ds.SEMESTER>=1]
ds['COURSE_CODE'] = ds['COURSE_CODE'].apply(lambda x: x.replace(' ', ''))
ds
```

Out[2]:

	ENCRYPTED_ID	STD_ID	COURSE_NAME	COURSE_CODE	COURSE_LETTER_GRADE	COURSE_CATEGORY	MUFREDAT	SECTION	SECTIONOGRC
0	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Bilgisayar A?larõ	CMPE334	CC	Z	2016-2017 MŸfredat	CMPE334-SEC-01	
1	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Yaz Stajõ I	CMPE399	S	Z	2016-2017 MŸfredat	CMPE399-SEC-01	
2	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Proje Oryantasyon	CMPE493	CC	Z	2016-2017 MŸfredat	CMPE493-SEC-01	
4	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Yazõlõm MŸhendisli?i	SE346	BB	Z	2016-2017 MŸfredat	SE346-SEC-01	
5	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	?? Ya?amõ i□in ?ngilizce II	ENG302	AA	Z	2016-2017 MŸfredat	ENG302-SEC-01	

Figure 4.3 Shows the dataset with only students from first to forth semester

4.2.3.1. Preparing Student Information Dataset

1. Removing all the other columns and keeping only ["STD_ID", "AGE", "CITY"] columns for preparing the students information dataset. As shown in Figure 4.4:

Keep only the columns which are neccessary for student information

In [6]:
```python
df = ds[["STD_ID", "AGE", "CITY", "GENDER", "SCHOLARSHIP", "REGISTRATION_REASON"]]
df
```

Out[6]:

	STD_ID	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON
0	F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
1	F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
2	F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
4	F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
5	F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
...	...	...	...	...	...	...
10190	F677	34	Ankara	Male	NaN	DGS
10191	F677	34	Ankara	Male	NaN	DGS
10205	F678	32	Ankara	Male	NaN	DGS
10208	F678	32	Ankara	Male	NaN	DGS
10212	F678	32	Ankara	Male	NaN	DGS

2304 rows × 6 columns

Figure 4.4 Shows the specific columns for student information

2. Next, the data will be grouped by [STD_ID] to keep only the unique number of students. As shown in Figure 4.5:

Group the dataset by student ID to keep only the unique students

In [7]:
```python
gk=df.groupby(['STD_ID'])
gk=gk.first()
gk
```

Out[7]:

STD_ID	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON
F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
F427	37	Ankara	Female	%25 ÖSYM İndirimi	ÖSS
F468	26	Ankara	Male	%50 ÖSYM İndirimi	ÖSS
F46D	25	Ankara	Female	%50 ÖSYM İndirimi	ÖSS
F477	22	Ankara	Male	%50 ÖSYM İndirimi	ÖSS
...	...	...	...	...	...
F671	30	Ankara	Male	None	DGS
F672	43	Ankara	Male	None	DGS
F676	31	Ankara	Female	None	DGS
F677	34	Ankara	Male	None	DGS
F678	32	Ankara	Male	None	DGS

175 rows × 5 columns

Figure 4.5 Illustrates the dataset which is grouped by student ID

3. Then a new dataset named STD_INFO will be created and exported in .CSV format. As shown in Figure 4.6:

Store the dataset in a new csv file name STD_INFO.csv

In [8]:
```
dt=gk
dt.sort_values(by=['STD_ID'])
dt

dt.to_csv('STD_INFO.csv')

df_INFO = pd.read_csv("STD_INFO.csv")

df_INFO
```

Out[8]:

	STD_ID	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON
0	F369	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme
1	F427	37	Ankara	Female	%25 ÖSYM İndirimi	ÖSS
2	F468	26	Ankara	Male	%50 ÖSYM İndirimi	ÖSS
3	F46D	25	Ankara	Female	%50 ÖSYM İndirimi	ÖSS
4	F477	22	Ankara	Male	%50 ÖSYM İndirimi	ÖSS
...	...	...	...	...	...	...
170	F671	30	Ankara	Male	NaN	DGS
171	F672	43	Ankara	Male	NaN	DGS
172	F676	31	Ankara	Female	NaN	DGS
173	F677	34	Ankara	Male	NaN	DGS
174	F678	32	Ankara	Male	NaN	DGS

175 rows × 6 columns

Figure 4.6 Dataset for student information

4.2.3.2. Preparing Student Course Dataset

1. Removing all other columns and keeping only ["STD_ID", "DIPLOMA_NOT", "COURSE_CODE", "COURSE_LETTER_GRADE"] columns for preparing student course dataset. As shown in Figure 4.7:

Keep only the columns which are neccessary for finale grades prediction

In [3]:
```
df = ds[["STD_ID", "DIPLOMA_NOT", "COURSE_CODE", "COURSE_LETTER_GRADE"]]

df
```

Out[3]:

	STD_ID	DIPLOMA_NOT	COURSE_CODE	COURSE_LETTER_GRADE
0	F369	2.86	CMPE334	CC
1	F369	2.86	CMPE399	S
2	F369	2.86	CMPE493	CC
4	F369	2.86	SE346	BB
5	F369	2.86	ENG302	AA
...	...	...	...	...
10190	F677	2.07	MATH275	DC
10191	F677	2.07	PHYS102	DD
10205	F678	2.58	ENG102	DC
10208	F678	2.58	PHYS101	FF
10212	F678	2.58	PHYS101	FF

2304 rows × 4 columns

Figure 4.7 The specific columns for student's courses with their corresponding grades

2. Next, the data will be grouped by ['STD_ID', "DIPLOMA_NOT", "COURSE_CODE"] to get the course grade letter separately for each course

using unique students. As shown in Figure 4.8:

Group the dataset by student ID to keep the unique courses for each unique students

In [4]:
```
gk=df.groupby(['STD_ID', "DIPLOMA_NOT", 'COURSE_CODE'])
gk=gk.first()
gk
```

Out[4]:

			COURSE_LETTER_GRADE
STD_ID	**DIPLOMA_NOT**	**COURSE_CODE**	
F369	**2.86**	**CMPE323**	BB
		CMPE325	CB
		CMPE326	CC
		CMPE331	BB
		CMPE334	CC
...	**...**	**...**	...
F677	**2.07**	**MATH275**	FD
		PHYS101	CC
		PHYS102	FD
F678	**2.58**	**ENG102**	DC
		PHYS101	FF

2163 rows × 1 columns

Figure 4.8 Illustrates the dataset which is grouped by student ID and course code

3. Now that we have our student course dataset, we will convert it from vertical form into horizontal form to prepare for the prediction process. Then a new dataset named STD_COURSE will be created and exported in .CSV format. As shown in Figure 4.9:

Reshape the dataset using unstack() function and store it in a new csv file name STD_COURSE.csv

In [5]:
```
dt=gk.unstack(level=2)
dt.columns = dt.columns.droplevel(0)
dt.sort_values(by=['STD_ID'])
dt

dt.to_csv('STD_COURSE.csv')

df_COURSE = pd.read_csv("STD_COURSE.csv")
df_COURSE = df_COURSE.rename({"DIPLOMA_NOT": "CGPA"}, axis=1)

df_COURSE = df_COURSE.drop(['STD_ID'], axis=1)
df_COURSE = df_COURSE[["ENG101","PHYS101","CMPE109","CEAC105","CMPE113","MATH151","ENG102",
                      "PHYS102","MATH152","CMPE114","CMPE134","ENG201","EE235","MATH275",
                      "EE203","CMPE251","CMPE225","MATH276","IE220","ENG204","CMPE226","CMPE236","CGPA"]]
df_COURSE
```

Out[5]:

	ENG101	PHYS101	CMPE109	CEAC105	CMPE113	MATH151	ENG102	PHYS102	MATH152	CMPE114	...	MATH275	EE203	CMPE251	CMPE225	M.
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1	BA	NaN	NaN	BB	NaN	NaN	NaN	NaN	NaN	BA	...	NaN	BA	NaN	AA	
2	DC	BB	BA	DC	AA	DD	CB	CB	DC	AA	...	BA	AA	BA	AA	
3	AA	AA	BA	CB	BA	AA	AA	AA	AA	AA	...	AA	AA	AA	AA	
4	AA	AA	BA	AA	BA	AA	AA	AA	AA	AA	...	BA	AA	BA	BB	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
170	NaN	CC	NaN	NaN	NaN	NaN	CB	NaN	FF	NaN	...	NaN	NaN	NaN	NaN	
171	NaN	NaN	NaN	NaN	NaN	NaN	CC	CC	DD	NaN	...	FD	NaN	NaN	NaN	
172	NaN	FD	NaN	NaN	NaN	FF	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
173	NaN	CC	NaN	NaN	NaN	FD	NaN	FD	FD	NaN	...	FD	NaN	NaN	NaN	
174	NaN	FF	NaN	NaN	NaN	NaN	DC	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	

175 rows x 23 columns

Figure 4.9 The student’s course with the horizontal mode after grouping

4. Then, we will combine STD_INFO.csv which is created in the previous steps with the STD_COURSE.csv to make it completely ready for applying the ML algorithms and export a new dataset named df_STD_COURSE in .CSV format. As shown in Figure 4.10:

Combine the two datasets STD_INFO and STD_COURSE and store it in new csv file name df_STD_COURSE and prepare for final grades prediction

In [9]:
```python
df_STD_COURSE = pd.concat([df_INFO, df_COURSE], axis=1)
df_STD_COURSE.to_csv('df_STD_COURSE.csv')
df_STD_COURSE = pd.read_csv("df_STD_COURSE.csv")
df_STD_COURSE = df_STD_COURSE.drop(df_STD_COURSE.columns[0], axis=1)
df_STD_COURSE = df_STD_COURSE.drop(['STD_ID'], axis=1)
df_STD_COURSE['SCHOLARSHIP'] = df_STD_COURSE['SCHOLARSHIP'].astype(str)

df_STD_COURSE
```

Out[9]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG101	PHYS101	CMPE109	CEAC105	CMPE113	...	MATH275	EE203	CMPE2
0	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
1	37	Ankara	Female	%25 ÖSYM İndirimi	ÖSS	BA	NaN	NaN	BB	NaN	...	NaN	BA	Na
2	26	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	DC	BB	BA	DC	AA	...	BA	AA	E
3	25	Ankara	Female	%50 ÖSYM İndirimi	ÖSS	AA	AA	BA	CB	BA	...	AA	AA	A
4	22	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	AA	AA	BA	AA	BA	...	BA	AA	E
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
170	30	Ankara	Male	nan	DGS	NaN	CC	NaN	NaN	NaN	...	NaN	NaN	Na
171	43	Ankara	Male	nan	DGS	NaN	NaN	NaN	NaN	NaN	...	FD	NaN	Na
172	31	Ankara	Female	nan	DGS	NaN	FD	NaN	NaN	NaN	...	NaN	NaN	Na
173	34	Ankara	Male	nan	DGS	NaN	CC	NaN	NaN	NaN	...	FD	NaN	Na
174	32	Ankara	Male	nan	DGS	NaN	FF	NaN	NaN	NaN	...	NaN	NaN	Na

175 rows × 28 columns

Figure 4.10 The complete dataset for students' courses

4.2.3.3. Preparing Student GPA Dataset

1. Removing all other columns and keeping only ["STD_ID", "SEMESTER", "GPA", "DIPLOMA_NOT", "DURATION"] columns for preparing student GPA dataset. As shown in Figure 4.11:

Keep only the columns which are neccessary for predicting graduation duration

In [3]:
```python
df = ds[["STD_ID", "SEMESTER", "GPA", "DURATION"]]

df
```

Out[3]:

	STD_ID	SEMESTER	GPA	DURATION
59	F468	1	2.25	3
60	F468	1	2.25	3
61	F468	1	2.25	3
62	F468	1	2.25	3
63	F468	1	2.25	3
...	...	...	...	...
10190	F677	4	1.19	5
10191	F677	4	1.19	5
10205	F678	4	1.14	5
10208	F678	4	1.14	5
10212	F678	3	0.21	5

2253 rows × 4 columns

Figure 4.11 Shows the specific columns for student semester with its desired

2. Thereafter, the data will be grouped by ["STD_ID", "DIPLOMA_NOT", "DURATION", "SEMESTER"] to get the GPA separately for each course using unique students. As shown in Figure 4.12:

Group the dataset by student ID to keep each unique semesters with its corresponding GPA

In [4]:

```
gk=df.groupby(['STD_ID', 'DURATION', 'SEMESTER'])
gk=gk.first()
gk
```

Out[4]:

STD_ID	DURATION	SEMESTER	GPA
F468	3	1	2.25
		2	2.69
		3	3.36
		4	3.37
F46D	4	1	3.61
...	...	...	...
F677	5	2	0.93
		3	0.69
		4	1.19
F678	5	3	0.21
		4	1.14

427 rows × 1 columns

Figure 4.12 Grouping the dataset by student ID and semester

3. Now that we have our student GPA dataset, we will convert it from vertical form into horizontal form to prepare for the prediction process, and store into a new file name STD_GPA. As shown in Figure 4.13:

Reshape the dataset using unstack() function and store it in a new csv file name STD_GPA.csv

In [5]:
```
dt=gk.unstack(level=2)
dt.columns = dt.columns.droplevel(1)

dt.to_csv('STD_GPA.csv')

df_GPA = pd.read_csv("STD_GPA.csv")
df_GPA.sort_values(by=['STD_ID'])
df_GPA
```

Out[5]:

	STD_ID	DURATION	GPA	GPA.1	GPA.2	GPA.3
0	F468	3	2.25	2.69	3.36	3.37
1	F46D	4	3.61	3.95	4.00	4.00
2	F477	4	3.76	3.59	3.19	3.20
3	F478	4	1.59	1.83	1.95	3.12
4	F47F	4	NaN	3.84	3.68	3.46
...	...	...	...	...	...	...
167	F671	4	NaN	NaN	1.73	1.82
168	F672	4	NaN	NaN	1.68	1.21
169	F676	5	NaN	NaN	0.86	1.21
170	F677	5	1.08	0.93	0.69	1.19
171	F678	5	NaN	NaN	0.21	1.14

172 rows × 6 columns

Figure 4.13 The student’s GPA dataset with the horizontal mode after grouping

4. Rename all GPA columns to their corresponding semesters. As shown in Figure 4.14:

Rename the columns in STD-GPA dataset

In [6]:
```
df_GPA = df_GPA.rename({"GPA": "SEMESTER_1"}, axis=1)
df_GPA = df_GPA.rename({"GPA.1": "SEMESTER_2"}, axis=1)
df_GPA = df_GPA.rename({"GPA.2": "SEMESTER_3"}, axis=1)
df_GPA = df_GPA.rename({"GPA.3": "SEMESTER_4"}, axis=1)
new_cols = ["STD_ID","SEMESTER_1","SEMESTER_2","SEMESTER_3","SEMESTER_4","DURATION"]
df_GPA=df_GPA[new_cols]
df_GPA = df_GPA.drop(["STD_ID"], axis=1)
df_GPA
```

Out[6]:

	SEMESTER_1	SEMESTER_2	SEMESTER_3	SEMESTER_4	DURATION
0	2.25	2.69	3.36	3.37	3
1	3.61	3.95	4.00	4.00	4
2	3.76	3.59	3.19	3.20	4
3	1.59	1.83	1.95	3.12	4
4	NaN	3.84	3.68	3.46	4
...	...	...	...	...	...
167	NaN	NaN	1.73	1.82	4
168	NaN	NaN	1.68	1.21	4
169	NaN	NaN	0.86	1.21	5
170	1.08	0.93	0.69	1.19	5
171	NaN	NaN	0.21	1.14	5

172 rows × 5 columns

Figure 4.14 shows the student GPA dataset after renaming some columns

5. Then, we will combine STD_INFO.csv which is created in the previous steps with the STD_GPA.csv to make it completely ready for applying the ML algorithms and export a new dataset named df_STD in .CSV format. As shown in Figure 4.15:

Combine the two datasets STD_INFO and STD_GPA and store it in new csv file name df_STD

and prepare for study duration prediction

In [10]:
```
df_STD = pd.concat([df_INFO, df_GPA], axis=1)
df_STD.to_csv('df_STD.csv')
df_STD = pd.read_csv("df_STD.csv")
df_STD = df_STD.drop(df_STD.columns[0], axis=1)
df_STD = df_STD.drop(['STD_ID'], axis=1)
df_STD['SCHOLARSHIP'] = df_STD['SCHOLARSHIP'].astype(str)

df_STD
```

Out[10]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	SEMESTER_1	SEMESTER_2	SEMESTER_3	SEMESTER_4	DURATION
0	26	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	2.25	2.69	3.36	3.37	3
1	25	Ankara	Female	%50 ÖSYM İndirimi	ÖSS	3.61	3.95	4.00	4.00	4
2	22	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	3.76	3.59	3.19	3.20	4
3	23	Ankara	Female	%50 ÖSYM İndirimi	ÖSS	1.59	1.83	1.95	3.12	4
4	23	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	NaN	3.84	3.68	3.46	4
...	...	...	...	...	...	...	...	...	...	...
167	30	Ankara	Male	nan	DGS	NaN	NaN	1.73	1.82	4
168	43	Ankara	Male	nan	DGS	NaN	NaN	1.68	1.21	4
169	31	Ankara	Female	nan	DGS	NaN	NaN	0.86	1.21	5
170	34	Ankara	Male	nan	DGS	1.08	0.93	0.69	1.19	5
171	32	Ankara	Male	nan	DGS	NaN	NaN	0.21	1.14	5

172 rows × 10 columns

Figure 4.15 The complete dataset for student GPA with its desired semester and student Study Duration

After processing the above steps, we prepared two main datasets for applying this study purposes as: df_STD_COURSE.csv is for predicting students' final grades and next term course grades, and df_STD_GPA.csv is for predicting students' graduation time. As shown in Figure 4.16:

4.2.3.4. Preparing Course Dataset for Recommendation Process

1. At first, we will load all desired ML libraries for our purpose.

Load the corresponding ML libraries for Recommendation and Prediction purpose

In [1]:
```
import pandas as pd
import neattext.functions as nfx
from sklearn.preprocessing import LabelEncoder
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity,linear_kernel
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
```

Figure 4.16 python libraries for data preprocessing and recommendation process

2. Loading the dataset which is gotten from the 2013-2014 course curriculum using only mandatory courses. As shown in Figure 4.17:

Load our dataset

```
In [2]: df = pd.read_csv("COURSE_DATASET.csv")
        df.head()
```

Out[2]:

	COURSE_CODE	COURSE_TITLE	SEMESTER	T	P	ECTS
0	ENG101	English For Academic Purposes I	1	4	0	3.5
1	PHYS101	General Physics I	1	3	2	6.0
2	CMPE109	Fundamentals of Computing	1	2	1	2.5
3	CEAC105	General Chemistry	1	3	2	5.0
4	CMPE113	Computer Programming I	1	2	2	4.0

Figure 4.17 Mandatory courses for 2013-2014 curriculum

3. We need this dataset for recommending the most similar course to the previous courses. For this purpose, we choose the COURSE_TITLE. As shown in Figure 4.18:

Select COURSE_TITLE for performing the recommendation proccess

```
In [3]: df['COURSE_TITLE']

Out[3]: 0                      English For Academic Purposes I
        1                                    General Physics I
        2                            Fundamentals of Computing
        3                                    General Chemistry
        4                               Computer Programming I
        5                                           Calculus I
        6                     English for Academic Purposes II
        7                                   General Physics II
        8                                          Calculus II
        9                              Computer Programming II
        10                Fundamentals of Electronic Components
        11                   English for Academic Purposes III
        12                                 Digital Electronics
        13                                     Linear  Algebra
        14                         Digital Circuits and Systems
        15                    Discrete Computational Structures
        16                          Object-Oriented Programming
        17                              Differential Equations
        18                          Probability and Statistics
        19                               Report Writing Skills
        20                                     Data Structures
        21     Introduction to Microprocessors and Microcontr...
```

Figure 4.18 Course title for text processing

4. For being ready for the further process we should first clean our text by removing stop words and special characters. As shown in Figure 4.19:

Clean Text: stopwords, special characters

```
In [4]:  df['CLEAN_COURSE_TITLE'] = df['COURSE_TITLE'].apply(nfx.remove_stopwords)
         df['CLEAN_COURSE_TITLE'] = df['CLEAN_COURSE_TITLE'].apply(nfx.remove_special_characters)
         df[['COURSE_TITLE','CLEAN_COURSE_TITLE']]
```

Out[4]:

	COURSE_TITLE	CLEAN_COURSE_TITLE
0	English For Academic Purposes I	English Academic Purposes
1	General Physics I	General Physics
2	Fundamentals of Computing	Fundamentals Computing
3	General Chemistry	General Chemistry
4	Computer Programming I	Computer Programming
5	Calculus I	Calculus
6	English for Academic Purposes II	English Academic Purposes II
7	General Physics II	General Physics II
8	Calculus II	Calculus II
9	Computer Programming II	Computer Programming II
10	Fundamentals of Electronic Components	Fundamentals Electronic Components
11	English for Academic Purposes III	English Academic Purposes III
12	Digital Electronics	Digital Electronics
13	Linear Algebra	Linear Algebra
14	Digital Circuits and Systems	Digital Circuits Systems

Figure 4.19 COURSE_TITLE vs CLEAN_COURSE_TITLE

5. We do vectorization for extracting distinct features out of our text. As shown in Figure 4.20:

Vectorize our Text using countvectorizer

```
In [5]:  count_vect = CountVectorizer()
         cv_mat = count_vect.fit_transform(df['CLEAN_COURSE_TITLE'])
```

Figure 4.20 Vectorized text in CLEAN_COURSE_TITLE

6. Getting the sparsity of text. As shown in Figure 4.21:

Sparsity Matrix

```
In [6]:  cv_mat

Out[6]:  <40x65 sparse matrix of type '<class 'numpy.int64'>'
                 with 104 stored elements in Compressed Sparse Row format>
```

Figure 4.21 Sparsity matrix

7. Getting the density of text. As shown in Figure 4.22:

Density Matrix

```
In [7]:  cv_mat.todense()

Out[7]:  matrix([[1, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 1, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 1, 0]], dtype=int64)
```

Figure 4.22 Density matrix

8. We used Cosine Similarity metric for finding the similar features in the text. As shown in Figure 4.23:

Cosine Similarity Matrix

```
In [8]:  cosine_sim_mat = cosine_similarity(cv_mat)
         cosine_sim_mat

Out[8]: array([[1.        , 0.        , 0.        , ..., 0.        , 0.        ,
                0.        ],
               [0.        , 1.        , 0.        , ..., 0.        , 0.        ,
                0.        ],
               [0.        , 0.        , 1.        , ..., 0.        , 0.        ,
                0.        ],
               ...,
               [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
                0.91287093],
               [0.        , 0.        , 0.        , ..., 0.        , 1.        ,
                0.        ],
               [0.        , 0.        , 0.        , ..., 0.91287093, 0.        ,
                1.        ]])
```

Figure 4.23 Cosine similarity matrix

9. Now for performing the recommendation, we will specify the course title and their respective indices. As shown in Figure 4.24:

Get Course ID/Index in dataset

```
In [9]:  course_indices = pd.Series(df.index,index=df['COURSE_TITLE']).drop_duplicates()
         course_indices

Out[9]: COURSE_TITLE
        English For Academic Purposes I                 0
        General Physics I                               1
        Fundamentals of Computing                       2
        General Chemistry                               3
        Computer Programming I                          4
        Calculus I                                      5
        English for Academic Purposes II                6
        General Physics II                              7
        Calculus II                                     8
        Computer Programming II                         9
        Fundamentals of Electronic Components          10
        English for Academic Purposes III              11
        Digital Electronics                            12
        Linear  Algebra                                13
        Digital Circuits and Systems                   14
        Discrete Computational Structures              15
        Object-Oriented Programming                    16
        Differential Equations                         17
```

Figure 4.24 Course title with the indexes

10. In this step an optional course is selected for further processes. As shown in Figure 4.25:

Index for a specific course

```
In [10]:  title = 'Computer Programming II'
          idx = course_indices[title]
          idx

Out[10]: 9
```

Figure 4.25 Index for the selected course

11. Then we will find the similar courses among the dataset using the selected course. As shown in Figure 4.26:

The similarity scores per cosine score

```
In [11]: scores = list(enumerate(cosine_sim_mat[idx]))
         scores

Out[11]: [(0, 0.0),
          (1, 0.0),
          (2, 0.0),
          (3, 0.0),
          (4, 0.816496580927726),
          (5, 0.0),
          (6, 0.2886751345948129),
          (7, 0.3333333333333334),
          (8, 0.408248290463863),
          (9, 1.0000000000000002),
          (10, 0.0),
          (11, 0.0),
          (12, 0.0),
          (13, 0.0),
          (14, 0.0),
          (15, 0.0),
          (16, 0.408248290463863),
```

Figure 4.26 Similarity scores for the selected course

12. After that we sort the results to bring the similar courses at the beginning of the list and then remove the similar course which is belong to selected course itself. As shown in Figure 4.27:

Sort the similarity scores per cosine score and omit the First Value/itself

```
In [12]: sorted_scores = sorted(scores,key=lambda x:x[1],reverse=True)
         sorted_scores[1:]

Out[12]: [(4, 0.816496580927726),
          (8, 0.408248290463863),
          (16, 0.408248290463863),
          (33, 0.408248290463863),
          (7, 0.3333333333333334),
          (23, 0.3333333333333334),
          (27, 0.3333333333333334),
          (31, 0.3333333333333334),
          (36, 0.3333333333333334),
          (6, 0.2886751345948129),
          (39, 0.2357022603955159),
          (0, 0.0),
          (1, 0.0),
          (2, 0.0),
```

Figure 4.27 Sorted similarity scores

13. Getting the indexes for the related courses. As shown in Figure 4.28:

Related Courses Indices

```
In [13]: selected_course_indices = [i[0] for i in sorted_scores[1:]]
         selected_course_indices

Out[13]: [4,
          8,
          16,
          33,
          7,
          23,
          27,
          31,
          36,
          6,
          39,
          0,
```

Figure 4.28 Indexes for the related courses

14. Getting the similarity scores for the related courses. As shown in Figure 4.29:

Related Courses Scores

```
In [14]: selected_course_scores = [i[1] for i in sorted_scores[1:]]
         selected_course_scores
```

```
Out[14]: [0.816496580927726,
          0.408248290463863,
          0.408248290463863,
          0.408248290463863,
          0.3333333333333334,
          0.3333333333333334,
          0.3333333333333334,
          0.3333333333333334,
          0.3333333333333334,
          0.2886751345948129,
          0.2357022603955159,
          0.0,
          0.0,
```

Figure 4.29 Similarity scores for the related courses

15. Displaying the similar courses in a data frame. As shown in Figure 4.30:

Create a dataframe for most similar courses with the course title, and semester using selected course indices

```
In [15]: recommended_result = df[['COURSE_CODE','COURSE_TITLE','SEMESTER']].iloc[selected_course_indices]
         rec_df = pd.DataFrame(recommended_result)
         rec_df.head()
```

Out[15]:

	COURSE_CODE	COURSE_TITLE	SEMESTER
4	CMPE113	Computer Programming I	1
8	MATH152	Calculus II	2
16	CMPE225	Object-Oriented Programming	3
33	CMPE334	Computer Networks	6
7	PHYS102	General Physics II	2

Figure 4.30 Data frame of similar courses

16. Adding the similarity scores to the similar courses data frame. As shown in Figure 4.31:

combining the similarity scores with the similar courses dataframe

```
In [16]: rec_df['similarity_scores'] = selected_course_scores
         rec_df.head()
```

Out[16]:

	COURSE_CODE	COURSE_TITLE	SEMESTER	similarity_scores
4	CMPE113	Computer Programming I	1	0.816497
8	MATH152	Calculus II	2	0.408248
16	CMPE225	Object-Oriented Programming	3	0.408248
33	CMPE334	Computer Networks	6	0.408248
7	PHYS102	General Physics II	2	0.333333

Figure 4.31 Similarity course data frame with similarity scores

17. We only need one course related to the selected course and use it for further processes. As shown in Figure 4.32:

Select the most similar course related to selected course

```
In [17]: rec_df.head(1)
```

Out[17]:

	COURSE_CODE	COURSE_TITLE	SEMESTER	similarity_scores
4	CMPE113	Computer Programming I	1	0.816497

Figure 4.32 The most similar course

18. Getting the related semester for selected course. As shown in Figure 4.33:

Find the related semester for selected course

```
In [18]:    selected_semester = df[df.COURSE_TITLE == title]
            selected_semester = selected_semester['SEMESTER']
            selected_semester = int(selected_semester.iloc[0])
            selected_semester

Out[18]: 2
```

Figure 4.33 Related semester for selected course

19. Getting the course code related to selected course. As shown in Figure 4.34:

Get the course_code for selected course

```
In [19]:    selected_course_code = df[df.COURSE_TITLE == title]
            selected_course_code = selected_course_code['COURSE_CODE']
            selected_course_code = selected_course_code.iloc[0]
            selected_course_code

Out[19]: 'CMPE114'
```

Figure 4.34 Course code for selected course

20. The related courses which are belong to the current or previous semester of selected course are removed and only the courses which are belong to the next semester of selected course are kept. As shown in Figure 4.35:

Select only the courses which are related to the next semester of selected course

prevent choosing the previous coursres related to selected course

```
In [20]: rec_df = rec_df[rec_df.SEMESTER > selected_semester]
         rec_df
```

Out[20]:

	COURSE_CODE	COURSE_TITLE	SEMESTER	similarity_scores
16	CMPE225	Object-Oriented Programming	3	0.408248
33	CMPE334	Computer Networks	6	0.408248
23	CMPE331	Computer Architecture and Organization	5	0.333333
27	CMPE325	Study of Programming Languages	5	0.333333
31	TURK402	Turkish Language II	6	0.333333
36	CMPE499	Summer Practice II	7	0.333333
39	HIST102	Principles of Atatürk and History of Turkish R...	8	0.235702
11	ENG201	English for Academic Purposes III	3	0.000000
12	EE235	Digital Electronics	3	0.000000
13	MATH275	Linear Algebra	3	0.000000

Figure 4.35 Next term courses related to the selected course

21. We need to get the related course code for prediction process. As shown in Figure 4.36:

Get the related course code

```
In [21]:  course_code = rec_df.iloc[0, 0]
          course_code

Out[21]:  'CMPE225'
```

Figure 4.36 Related course code

22. We need the related course title for showing in our prediction result. As shown in Figure 4.37:

Get the related course title

```
In [22]:  course_title = rec_df.iloc[0, 1]
          course_title

Out[22]:  'Object-Oriented Programming'
```

Figure 4.37 Related course title

23. We need the related course semester for showing in our prediction result. As shown in Figure 4.38:

Get the related course semester

```
In [23]:  semester = rec_df.iloc[0, 2]
          semester

Out[23]:  3
```

Figure 4.38 Related course semester

24. In this step we are trying to prepare the dataset for the prediction process. First of all, we should merge student_course and STF_INFO datasets and keep only the columns which are necessary for predicting next term course grade. As shown in Figure 4.39:

Now it is time for predicting the related course grade

First of all we should merge student_course and STF_INFO datasets and keep only the columns which are necessary for predicting next term course grade

In [49]:
```
df_COURSE = pd.read_csv("STD_COURSE.csv")
df_COURSE = df_COURSE[[selected_course_code, course_code]]

df_INFO = pd.read_csv("STD_INFO.csv")

df_COURSE = pd.concat([df_INFO, df_COURSE], axis=1)
df_COURSE.to_csv('df_STD_ALL_COURSES.csv')
df_COURSE = pd.read_csv("df_STD_ALL_COURSES.csv")
df_COURSE = df_COURSE.drop(df_COURSE.columns[0], axis=1)
df_COURSE = df_COURSE.drop(['STD_ID'], axis=1)

df_COURSE['SCHOLARSHIP'] = df_COURSE['SCHOLARSHIP'].astype(str)


df_COURSE
```

Out[49]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG102	ENG201
0	26	OTHERS	Female	ÖSYM Burslu	DGS Ek Yerleştirme	NaN	NaN
1	37	Ankara	Female	%25 ÖSYM İndirimi	ÖSS	NaN	AA
2	26	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	CB	CB
3	25	Ankara	Female	%50 ÖSYM İndirimi	ÖSS	AA	AA
4	22	Ankara	Male	%50 ÖSYM İndirimi	ÖSS	AA	BA
...	...	...	...	...	...	...	...
170	30	Ankara	Male	nan	DGS	CB	NaN
171	43	Ankara	Male	nan	DGS	CC	NaN
172	31	Ankara	Female	nan	DGS	NaN	NaN
173	34	Ankara	Male	nan	DGS	NaN	NaN
174	32	Ankara	Male	nan	DGS	DC	NaN

175 rows × 7 columns

Figure 4.39 The prepared dataset for predicting next term course grade

4.3. ML Model Implementation and Fitting

After preparing the datasets in the previous section (Data Preprocessing) we have few main steps to create our ML model and prepare it for the implementation.

4.3.1. Clean the Dataset for Predicting Study Duration

After preparing the dataset by performing some preprocesses in “Data Preprocessing” section, there are few steps to clean the dataset and prepare it for prediction process.

1. At the first, we will convert the data in [CITY] column into in-state and out-of-state for this study "Ankara" is converted into in-state and the other cities are converted into out-of-state. As shown in Figure 4.40:

Clean the city column by converting the data into in-state and out-state

In [60]:

```python
def clean_city(x):
    if 'Ankara' in x:
        return 'in-state'
    return 'out-of-state'

df_STD['CITY'] = df_STD['CITY'].apply(clean_city)
df_STD
```

Out[60]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	SEMESTER_1	SEMESTER_2	SEMESTER_3	SEMESTER_4	DURATION
0	26	out-of-state	Female	ÖSYM Burslu	DGS Ek Yerleştirme	2.73	2.64	4.00	2.75	2
1	37	in-state	Female	%25 ÖSYM İndirimi	ÖSS	2.90	3.37	4.00	3.81	2
2	26	in-state	Male	%50 ÖSYM İndirimi	ÖSS	2.25	2.69	3.36	3.37	3
3	25	in-state	Female	%50 ÖSYM İndirimi	ÖSS	3.61	3.95	4.00	4.00	4
4	22	in-state	Male	%50 ÖSYM İndirimi	ÖSS	3.76	3.59	3.19	3.20	4
...	...	...	...	...	...	...	...	...	...	...
170	30	in-state	Male	nan	DGS	NaN	NaN	1.73	1.82	4
171	43	in-state	Male	nan	DGS	NaN	NaN	1.68	1.21	4
172	31	in-state	Female	nan	DGS	NaN	NaN	0.86	1.21	5
173	34	in-state	Male	nan	DGS	1.08	0.93	0.69	1.19	5
174	32	in-state	Male	nan	DGS	NaN	NaN	0.21	1.14	5

175 rows × 10 columns

Figure 4.40 Clean the city in dataset

2. Then the data in [SCHOLARSHIP] column is converted into with_scholarship and without_scholarship as “%25 ÖSYM İndirimi, %50 ÖSYM İndirimi, and ÖSYM Burslu” are changed to with_scholarship and “Atilim Burssuz” is changed to without_scholarship. As shown in Figure 4.41:

Clean the scholarship column by converting the data into with_scholarship and without_scholarship

In [61]:

```
def clean_scholarship(x):
    if '%25 ÖSYM İndirimi' in x:
        return 'with_scholarship'
    if '%50 ÖSYM İndirimi' in x:
        return 'with_scholarship'
    if 'ÖSYM Burslu' in x:
        return 'with_scholarship'
    else:
        return 'without_scholarship'
df_STD['SCHOLARSHIP'] = df_STD['SCHOLARSHIP'].apply(clean_scholarship)
df_STD
```

Out[61]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	SEMESTER_1	SEMESTER_2	SEMESTER_3	SEMESTER_4	DURATION
0	26	out-of-state	Female	with_scholarship	DGS Ek Yerleştirme	2.73	2.64	4.00	2.75	2
1	37	in-state	Female	with_scholarship	ÖSS	2.90	3.37	4.00	3.81	2
2	26	in-state	Male	with_scholarship	ÖSS	2.25	2.69	3.36	3.37	3
3	25	in-state	Female	with_scholarship	ÖSS	3.61	3.95	4.00	4.00	4
4	22	in-state	Male	with_scholarship	ÖSS	3.76	3.59	3.19	3.20	4
...	...	...	...	...	...	...	...	...	...	...
170	30	in-state	Male	without_scholarship	DGS	NaN	NaN	1.73	1.82	4
171	43	in-state	Male	without_scholarship	DGS	NaN	NaN	1.68	1.21	4
172	31	in-state	Female	without_scholarship	DGS	NaN	NaN	0.86	1.21	5
173	34	in-state	Male	without_scholarship	DGS	1.08	0.93	0.69	1.19	5
174	32	in-state	Male	without_scholarship	DGS	NaN	NaN	0.21	1.14	5

175 rows × 10 columns

Figure 4.41 Clean the scholarship in dataset

3. Then the data in [REGISTRATION _REASON] column is converted into with_exam and without_exam as “DGS Ek Yerleştirme, and DGS” are changed to without_exam and “ÖSS, and ÖSS Ek Kontenjan” are change to with_exam. As shown in Figure 4.42:

Clean the registration_reason column by converting the data into with_exam and without_exam

In [13]:

```
def clean_registration_reason(x):
    if 'DGS Ek Yerleştirme' in x:
        return 'without_exam'
    if 'DGS' in x:
        return 'without_exam'
    else:
        return 'with_exam'
df_STD['REGISTRATION_REASON'] = df_STD['REGISTRATION_REASON'].apply(clean_registration_reason)
df_STD
```

Out[13]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	SEMESTER_1	SEMESTER_2	SEMESTER_3	SEMESTER_4	DURATION
0	26	out-of-state	Female	with_scholarship	without_exam	2.73	2.64	4.00	2.75	2
1	37	in-state	Female	with_scholarship	with_exam	2.90	3.37	4.00	3.81	2
2	26	in-state	Male	with_scholarship	with_exam	2.25	2.69	3.36	3.37	3
3	25	in-state	Female	with_scholarship	with_exam	3.61	3.95	4.00	4.00	4
4	22	in-state	Male	with_scholarship	with_exam	3.76	3.59	3.19	3.20	4
...	...	...	...	...	...	...	...	...	...	...
170	30	in-state	Male	without_scholarship	without_exam	NaN	NaN	1.73	1.82	4
171	43	in-state	Male	without_scholarship	without_exam	NaN	NaN	1.68	1.21	4
172	31	in-state	Female	without_scholarship	without_exam	NaN	NaN	0.86	1.21	5
173	34	in-state	Male	without_scholarship	without_exam	1.08	0.93	0.69	1.19	5
174	32	in-state	Male	without_scholarship	without_exam	NaN	NaN	0.21	1.14	5

175 rows × 10 columns

Figure 4.42 Clean the registration_reason in dataset

4. Thereafter, the whole data in [CITY, SCHOLARSHIP, and REGISTRATION_REASON] columns are encoded. As shown in Figure 4.43:

Encode city, scholarship, and registration_reason columns

```
In [64]:
CITY = LabelEncoder()
GENDER=LabelEncoder()
SCHOLARSHIP=LabelEncoder()
REGISTRATION_REASON=LabelEncoder()

df_STD['CITY'] = CITY.fit_transform(df_STD['CITY'])
df_STD['GENDER'] = GENDER.fit_transform(df_STD['GENDER'])
df_STD['SCHOLARSHIP'] = SCHOLARSHIP.fit_transform(df_STD['SCHOLARSHIP'])
df_STD['REGISTRATION_REASON'] = REGISTRATION_REASON.fit_transform(df_STD['REGISTRATION_REASON'])

df_STD
```

Out[64]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	SEMESTER_1	SEMESTER_2	SEMESTER_3	SEMESTER_4	DURATION
0	26	1	0	0	1	2.73	2.64	4.00	2.75	2
1	37	0	0	0	0	2.90	3.37	4.00	3.81	2
2	26	0	1	0	0	2.25	2.69	3.36	3.37	3
3	25	0	0	0	0	3.61	3.95	4.00	4.00	4
4	22	0	1	0	0	3.76	3.59	3.19	3.20	4
...	...	...	...	...	...	...	...	...	...	...
170	30	0	1	1	1	NaN	NaN	1.73	1.82	4
171	43	0	1	1	1	NaN	NaN	1.68	1.21	4
172	31	0	0	1	1	NaN	NaN	0.86	1.21	5
173	34	0	1	1	1	1.08	0.93	0.69	1.19	5
174	32	0	1	1	1	NaN	NaN	0.21	1.14	5

175 rows × 10 columns

Figure 4.43 Encode all cleaned labels in dataset

5. Then clean the dataset out of anu null values. As shown in Figure 4.44:

Check the entire dataset for any null values

```
In [52]:
df_STD = df_STD.replace(np.nan, 0)
df_STD.isnull().sum()

Out[52]:
AGE                    0
CITY                   0
GENDER                 0
SCHOLARSHIP            0
REGISTRATION_REASON    0
SEMESTER_1             0
SEMESTER_2             0
SEMESTER_3             0
SEMESTER_4             0
DURATION               0
dtype: int64
```

Figure 4.44 Removing null values from STD_GPA dataset

6. Split the dataset into a set of predictor variables and predictable variable. As shown in Figure 4.45:

Divide the dataset in independent variables and dependent variable X, y

```
In [13]:
X = df_STD.drop("DURATION", axis=1)
y = df_STD["DURATION"]
```

Figure 4.45 Splitting the STD_GPA dataset in X, and y

7. Split the dataset in test and train sets then choose a ML algorithm and apply it

on the selected dataset. As shown in Figure 4.46:

Split the dataset in test and train sets and then select and fit the ML algorithm

```
In [14]:  from sklearn.svm import SVR
          regressor = SVR()
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
          regressor.fit(X_train, y_train)

Out[14]:  ▾ SVR
          SVR()
```

Figure 4.46 ML algorithm for predicting Study Duration

8. Use MAE (Mean Absolute Error) to validate the algorithm. As shown in Figure 4.47:

Evaluate the ML algorithm usin MAE (Mea Absolute Erro)

```
In [15]:  from sklearn.metrics import  mean_absolute_error
          import numpy as np
          y_pred = regressor.predict(X)
          error = np.sqrt(mean_absolute_error(y, y_pred))
          print(error)

          0.6590762290831833
```

Figure 4.47 Evaluating ML algorithm

9. Create a set of predictor variables for predicting the graduation time. As shown in Figure 4.48:

Create an array of predictor labls for predicting the graduation time

```
In [19]:  X = np.array([[26,"out-of-state","Male","with_scholarship","with_exam",2.73, 2.64, 4, 2.75]])
          X

Out[19]:  array([['26', 'out-of-state', 'Male', 'with_scholarship', 'with_exam',
                  '2.73', '2.64', '4', '2.75']], dtype='<U32')
```

Figure 4.48 A set of predictor variables for predicting graduation time

10. Encode the city label to be prepared for fitting the model. As shown in Figure 4.49:

Encode the city, scholarship, and registration_reason lables in array

```
In [57]:  X[:, 1] = CITY.transform(X[:,1])
          X[:, 2] = GENDER.transform(X[:, 2])
          X[:, 3] = SCHOLARSHIP.transform(X[:, 3])
          X[:, 4] = REGISTRATION_REASON.transform(X[:, 4])
          X = X.astype(float)
          X

Out[57]:  array([[26.  ,  1.  ,  1.  ,  0.  ,  0.  ,  2.73,  2.64,  4.  ,  2.75]])
```

Figure 4.49 Encoding the city in predictor variables

11. Predict the graduation time using the prepared variables by selected ML algorithm. As shown in Figure 4.50:

Perform the predection by predictor array using our model

In [18]:
```
y_pred = regressor.predict(X)
y_pred = round(y_pred[0])
y_pred
```

Out[18]: 4

Figure 4.50 Graduation prediction result

4.3.2. Clean the Dataset for Predicting Final Grades

After preparing the dataset by performing some preprocesses in “Data Preprocessing” section, there are few steps to clean the dataset and prepare it for prediction process.

1. First of all, we will convert the data in [CITY] column into in-state and out-of-state for this study “Ankara” is converted into in-state and the other cities are converted into out-of-state. As shown in Figure 4.51:

Clean the city column by converting the data into in-state and out-state

In [96]:
```
def clean_city(x):
    if 'Ankara' in x:
        return 'in-state'
    return 'out-of-state'

df_STD_COURSE['CITY'] = df_STD_COURSE['CITY'].apply(clean_city)
df_STD_COURSE
```

Out[96]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG101	PHYS101	CMPE109	CEAC105	CMPE113	...	MATH275	EE203	CMPE251
0	26	out-of-state	Female	ÖSYM Burslu	DGS Ek Yerleştirme	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
1	37	out-of-state	Female	%25 ÖSYM İndirimi	ÖSS	BA	NaN	NaN	BB	NaN	...	NaN	BA	NaN
2	26	out-of-state	Male	%50 ÖSYM İndirimi	ÖSS	DC	BB	BA	DC	AA	...	BA	AA	BA
3	25	out-of-state	Female	%50 ÖSYM İndirimi	ÖSS	AA	AA	BA	CB	BA	...	AA	AA	AA
4	22	out-of-state	Male	%50 ÖSYM İndirimi	ÖSS	AA	AA	BA	AA	BA	...	BA	AA	BA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	30	out-of-state	Male	nan	DGS	NaN	CC	NaN	NaN	NaN	...	NaN	NaN	NaN
171	43	out-of-state	Male	nan	DGS	NaN	NaN	NaN	NaN	NaN	...	FD	NaN	NaN
172	31	out-of-state	Female	nan	DGS	NaN	FD	NaN	NaN	NaN	...	NaN	NaN	NaN
173	34	out-of-state	Male	nan	DGS	NaN	CC	NaN	NaN	NaN	...	FD	NaN	NaN
174	32	out-of-state	Male	nan	DGS	NaN	FF	NaN	NaN	NaN	...	NaN	NaN	NaN

175 rows × 28 columns

Figure 4.51 Clean the city in dataset

2. Then the data in [SCHOLARSHIP] column is converted into with_scholarship and without_scholarship as "%25 ÖSYM İndirimi, %50 ÖSYM İndirimi, and ÖSYM Burslu" are changed to with_scholarship and "Atilim Burssuz" is changed to without_scholarship. As shown in Figure 4.52:

Clean the scholarship column by converting the data into with_scholarship and without_scholarship

In [97]:
```python
def clean_scholarship(x):
    if '%25 ÖSYM İndirimi' in x:
        return 'with_scholarship'
    if '%50 ÖSYM İndirimi' in x:
        return 'with_scholarship'
    if 'ÖSYM Burslu' in x:
        return 'with_scholarship'
    else:
        return 'without_scholarship'
df_STD_COURSE['SCHOLARSHIP'] = df_STD_COURSE['SCHOLARSHIP'].apply(clean_scholarship)
df_STD_COURSE
```

Out[97]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG101	PHYS101	CMPE109	CEAC105	CMPE113	...	MATH275	EE203	CMPE25
0	26	out-of-state	Female	with_scholarship	DGS Ek Yerleştirme	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
1	37	out-of-state	Female	with_scholarship	ÖSS	BA	NaN	NaN	BB	NaN	...	NaN	BA	Na
2	26	out-of-state	Male	with_scholarship	ÖSS	DC	BB	BA	DC	AA	...	BA	AA	B
3	25	out-of-state	Female	with_scholarship	ÖSS	AA	AA	BA	CB	BA	...	AA	AA	A
4	22	out-of-state	Male	with_scholarship	ÖSS	AA	AA	BA	AA	BA	...	BA	AA	B
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
170	30	out-of-state	Male	without_scholarship	DGS	NaN	CC	NaN	NaN	NaN	...	NaN	NaN	Na
171	43	out-of-state	Male	without_scholarship	DGS	NaN	NaN	NaN	NaN	NaN	...	FD	NaN	Na
172	31	out-of-state	Female	without_scholarship	DGS	NaN	FD	NaN	NaN	NaN	...	NaN	NaN	Na
173	34	out-of-state	Male	without_scholarship	DGS	NaN	CC	NaN	NaN	NaN	...	FD	NaN	Na
174	32	out-of-state	Male	without_scholarship	DGS	NaN	FF	NaN	NaN	NaN	...	NaN	NaN	Na

175 rows × 28 columns

Figure 4.52 Clean the scholarship in dataset

3. Then the data in [REGISTRATION _REASON] column is converted into with_exam and without_exam as "DGS Ek Yerleştirme, and DGS" are changed to without_exam and "ÖSS, and ÖSS Ek Kontenjan" are change to with_exam. As shown in Figure 4.53:

Clean the registration_reason column by converting the data into with_exam and without_exam

In [12]:
```
def clean_registration_reason(x):
    if 'DGS Ek Yerleştirme' in x:
        return 'without_exam'
    if 'DGS' in x:
        return 'without_exam'
    else:
        return 'with_exam'
df_STD_COURSE['REGISTRATION_REASON'] = df_STD_COURSE['REGISTRATION_REASON'].apply(clean_registration_reason)
df_STD_COURSE
```

Out[12]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG101	PHYS101	CMPE109	CEAC105	CMPE113	...	MATH275	EE203	CMPE25
0	26	out-of-state	Female	with_scholarship	without_exam	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
1	37	in-state	Female	with_scholarship	with_exam	BA	NaN	NaN	BB	NaN	...	NaN	BA	Na
2	26	in-state	Male	with_scholarship	with_exam	DC	BB	BA	DC	AA	...	BA	AA	B
3	25	in-state	Female	with_scholarship	with_exam	AA	AA	BA	CB	BA	...	AA	AA	A
4	22	in-state	Male	with_scholarship	with_exam	AA	AA	BA	AA	BA	...	BA	AA	B
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
170	30	in-state	Male	without_scholarship	without_exam	NaN	CC	NaN	NaN	NaN	...	NaN	NaN	Na
171	43	in-state	Male	without_scholarship	without_exam	NaN	NaN	NaN	NaN	NaN	...	FD	NaN	Na
172	31	in-state	Female	without_scholarship	without_exam	NaN	FD	NaN	NaN	NaN	...	NaN	NaN	Na
173	34	in-state	Male	without_scholarship	without_exam	NaN	CC	NaN	NaN	NaN	...	FD	NaN	Na
174	32	in-state	Male	without_scholarship	without_exam	NaN	FF	NaN	NaN	NaN	...	NaN	NaN	Na

175 rows × 28 columns

Figure 4.53 Clean the registration_reason in dataset

4. Thereafter, the whole data in [CITY, SCHOLARSHIP, and REGISTRATION_REASON] columns are encoded. As shown in Figure 4.54:

Encode city, scholarship, and registration_reason columns

In [99]:

```
CITY=LabelEncoder()
GENDER=LabelEncoder()
SCHOLARSHIP=LabelEncoder()
REGISTRATION_REASON=LabelEncoder()
ENG101=LabelEncoder()
PHYS101=LabelEncoder()
CMPE109=LabelEncoder()
CHE105=LabelEncoder()
CMPE113=LabelEncoder()
MATH151=LabelEncoder()
ENG102=LabelEncoder()
PHYS102=LabelEncoder()
MATH152=LabelEncoder()
CMPE114=LabelEncoder()
CMPE134=LabelEncoder()
ENG201=LabelEncoder()
EE235=LabelEncoder()
MATH275=LabelEncoder()
EE203=LabelEncoder()
CMPE251=LabelEncoder()
CMPE225=LabelEncoder()
MATH276=LabelEncoder()
IE220=LabelEncoder()
ENG204=LabelEncoder()
CMPE226=LabelEncoder()
CMPE236=LabelEncoder()

df_STD_COURSE['CITY'] = CITY.fit_transform(df_STD_COURSE['CITY'])
df_STD_COURSE['GENDER'] = GENDER.fit_transform(df_STD_COURSE['GENDER'])
df_STD_COURSE['SCHOLARSHIP'] = SCHOLARSHIP.fit_transform(df_STD_COURSE['SCHOLARSHIP'])
df_STD_COURSE['REGISTRATION_REASON'] = REGISTRATION_REASON.fit_transform(df_STD_COURSE['REGISTRATION_REASON'])
df_STD_COURSE['ENG101'] = ENG101.fit_transform(df_STD_COURSE['ENG101'])
df_STD_COURSE['PHYS101'] = PHYS101.fit_transform(df_STD_COURSE['PHYS101'])
df_STD_COURSE['CMPE109'] = CMPE109.fit_transform(df_STD_COURSE['CMPE109'])
df_STD_COURSE['CEAC105'] = CHE105.fit_transform(df_STD_COURSE['CEAC105'])
df_STD_COURSE['CMPE113'] = CMPE113.fit_transform(df_STD_COURSE['CMPE113'])
df_STD_COURSE['MATH151'] = MATH151.fit_transform(df_STD_COURSE['MATH151'])
df_STD_COURSE['ENG102'] = ENG102.fit_transform(df_STD_COURSE['ENG102'])
df_STD_COURSE['PHYS102'] = PHYS102.fit_transform(df_STD_COURSE['PHYS102'])
df_STD_COURSE['MATH152'] = MATH152.fit_transform(df_STD_COURSE['MATH152'])
df_STD_COURSE['CMPE114'] = CMPE114.fit_transform(df_STD_COURSE['CMPE114'])
df_STD_COURSE['CMPE134'] = CMPE134.fit_transform(df_STD_COURSE['CMPE134'])
df_STD_COURSE['ENG201'] = ENG201.fit_transform(df_STD_COURSE['ENG201'])
df_STD_COURSE['EE235'] = EE235.fit_transform(df_STD_COURSE['EE235'])
df_STD_COURSE['MATH275'] = MATH275.fit_transform(df_STD_COURSE['MATH275'])
df_STD_COURSE['EE203'] = EE203.fit_transform(df_STD_COURSE['EE203'])
df_STD_COURSE['CMPE251'] = CMPE251.fit_transform(df_STD_COURSE['CMPE251'])
df_STD_COURSE['CMPE225'] = CMPE225.fit_transform(df_STD_COURSE['CMPE225'])
df_STD_COURSE['MATH276'] = MATH276.fit_transform(df_STD_COURSE['MATH276'])
df_STD_COURSE['IE220'] = IE220.fit_transform(df_STD_COURSE['IE220'])
df_STD_COURSE['ENG204'] = ENG204.fit_transform(df_STD_COURSE['ENG204'])
df_STD_COURSE['CMPE226'] = CMPE226.fit_transform(df_STD_COURSE['CMPE226'])
df_STD_COURSE['CMPE236'] = CMPE236.fit_transform(df_STD_COURSE['CMPE236'])
df_STD_COURSE
```

Out[99]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG101	PHYS101	CMPE109	CEAC105	CMPE113	...	MATH275	EE203	CMPE251
0	26	0	0	0	1	9	9	9	9	9	...	8	8	8
1	37	0	0	0	0	1	9	9	2	9	...	8	1	8
2	26	0	1	0	0	5	2	1	5	0	...	1	0	1
3	25	0	0	0	0	0	0	1	3	1	...	0	0	0
4	22	0	1	0	0	0	0	1	0	1	...	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	30	0	1	1	1	9	4	9	9	9	...	8	8	8
171	43	0	1	1	1	9	9	9	9	9	...	7	8	8
172	31	0	0	1	1	9	7	9	9	9	...	8	8	8
173	34	0	1	1	1	9	4	9	9	9	...	7	8	8
174	32	0	1	1	1	9	8	9	9	9	...	8	8	8

175 rows × 28 columns

Figure 4.54 Encode all cleaned labels in dataset

5. Then clean the dataset out of anu null values. As shown in Figure 4.55:

Check the entire dataset for any null values

```
In [86]:  df_STD_COURSE = df_STD_COURSE.replace(np.nan, 0)
          df_STD_COURSE.isnull().sum()

Out[86]:  AGE                    0
          CITY                   0
          GENDER                 0
          SCHOLARSHIP            0
          REGISTRATION_REASON    0
          ENG101                 0
          PHYS101                0
          CMPE109                0
          CEAC105                0
          CMPE113                0
          MATH151                0
          ENG102                 0
          PHYS102                0
          MATH152                0
          CMPE114                0
          CMPE134                0
          ENG201                 0
          EE235                  0
          MATH275                0
          EE203                  0
          CMPE251                0
          CMPE225                0
          MATH276                0
          IE220                  0
          ENG204                 0
          CMPE226                0
          CMPE236                0
          CGPA                   0
          dtype: int64
```

Figure 4.55 Removing the null values in dataset

6. Split the dataset into a set of predictor variables and predictable variable. As shown in Figure 4.55:

Divide the dataset in independent variables and dependent variable X, y

```
In [12]:  X = df_STD_COURSE.drop("CGPA", axis=1)
          y = df_STD_COURSE["CGPA"]
```

Figure 4.56 Split the STD_COURSE dataset in X, and y

7. Split the dataset in test and train sets and choose a ML algorithm and apply it on the selected dataset. As shown in Figure 4.57:

Split the dataset in test and train sets and then select and fit the ML algorithm

```
In [13]:  from sklearn.svm import SVR
          regressor = SVR()
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
          regressor.fit(X_train, y_train)

Out[13]:  ▾ SVR
          SVR()
```

Figure 4.57 ML algorithm for predicting Finale Grades

8. Use MAE (Mean Absolute Error) to validate the algorithm. As shown in Figure 4.58:

Evaluate the ML algorithm usin MAE (Mea Absolute Erro)

```
In [14]:  from sklearn.metrics import  mean_absolute_error
          import numpy as np
          y_pred = regressor.predict(X)
          error = np.sqrt(mean_absolute_error(y, y_pred))
          error
Out[14]:  0.41236337386134836
```

Figure 4.58 Evaluating ML algorithm

9. Create a set of predictor variables for predicting the graduation time. As shown in Figure 4.59:

Create an array of predictor labels for predicting the CGPA

```
In [18]:  X = np.array([[26,"in-state","Male","with_scholarship","with_exam","AA","BB","AA","AA","CC","CB","CB","AA","AA","BB",
                        "CC","DD","DC","DD","AA","BB","CC","AA","AA","AA","BB","AA"]])
          X

Out[18]:  array([['26', 'in-state', 'Male', 'with_scholarship', 'with_exam', 'AA',
                  'BB', 'AA', 'AA', 'CC', 'CB', 'CB', 'AA', 'AA', 'BB', 'CC', 'DD',
                  'DC', 'DD', 'AA', 'BB', 'CC', 'AA', 'AA', 'AA', 'BB', 'AA']],
                dtype='<U16')
```

Figure 4.59 A set of predictor variables for predicting Final Grades

10. Encode the city label to be prepared for fitting the model. As shown in Figure 4.60:

Encode all the lables in array

```
In [91]:  X[:, 1] = CITY.transform(X[:, 1])
          X[:, 2] = GENDER.transform(X[:, 2])
          X[:, 3] = SCHOLARSHIP.transform(X[:, 3])
          X[:, 4] = REGISTRATION_REASON.transform(X[:, 4])
          X[:, 5] = ENG101.transform(X[:, 5])
          X[:, 6] = PHYS101.transform(X[:, 6])
          X[:, 7] = CMPE109.transform(X[:, 7])
          X[:, 8] = CHE105.transform(X[:, 8])
          X[:, 9] = CMPE113.transform(X[:, 9])
          X[:, 10] = MATH151.transform(X[:, 10])
          X[:, 11] = ENG102.transform(X[:, 11])
          X[:, 12] = PHYS102.transform(X[:, 12])
          X[:, 13] = MATH152.transform(X[:, 13])
          X[:, 14] = CMPE114.transform(X[:, 14])
          X[:, 15] = CMPE134.transform(X[:, 15])
          X[:, 16] = ENG201.transform(X[:, 16])
          X[:, 17] = EE235.transform(X[:, 17])
          X[:, 18] = MATH275.transform(X[:, 18])
          X[:, 19] = EE203.transform(X[:, 19])
          X[:, 20] = CMPE251.transform(X[:, 20])
          X[:, 21] = CMPE225.transform(X[:, 21])
          X[:, 22] = MATH276.transform(X[:, 22])
          X[:, 23] = IE220.transform(X[:, 23])
          X[:, 24] = ENG204.transform(X[:, 24])
          X[:, 25] = CMPE226.transform(X[:, 25])
          X[:, 26] = CMPE236.transform(X[:, 26])
          X = X.astype(float)
          X

Out[91]:  array([[26.,  0.,  1.,  0.,  0.,  0.,  2.,  0.,  0.,  4.,  3.,  3.,  0.,
                   0.,  2.,  4.,  5.,  5.,  6.,  0.,  2.,  4.,  0.,  0.,  0.,  2.,
                   0.]])
```

Figure 4.60 Encoding all predictor variables

11. Predict the final grades using the prepared variables by selected ML algorithm. As shown in Figure 4.61:

Perform the predection by predictor array using our model

```
In [17]: y_pred = regressor.predict(X)
         y_pred

Out[17]: array([3.41656967])
```

Figure 4.61 Final Grades prediction result

4.3.3. Clean the Dataset for Predicting Next Term Course Grades

After finding the most similar course to the selected course in "Data Preprocessing" section, there are some few steps for cleaning the dataset and preparing for the prediction process.

1. First of all, we will convert the data in [CITY] column into in-state and out-of-state for this study "Ankara" is converted into in-state and the other cities are converted into out-of-state. As shown in Figure 4.62:

Clean the city column by converting the data into in-state and out-state

```
In [50]: def clean_city(x):
             if 'Ankara' in x:
                 return 'in-state'
             return 'out-of-state'

         df_COURSE['CITY'] = df_COURSE['CITY'].apply(clean_city)
         df_COURSE

Out[50]:
```

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG102	ENG201
0	26	out-of-state	Female	ÖSYM Burslu	DGS Ek Yerleştirme	NaN	NaN
1	37	in-state	Female	%25 ÖSYM İndirimi	ÖSS	NaN	AA
2	26	in-state	Male	%50 ÖSYM İndirimi	ÖSS	CB	CB
3	25	in-state	Female	%50 ÖSYM İndirimi	ÖSS	AA	AA
4	22	in-state	Male	%50 ÖSYM İndirimi	ÖSS	AA	BA
...	...	...	...	...	...	...	...
170	30	in-state	Male	nan	DGS	CB	NaN
171	43	in-state	Male	nan	DGS	CC	NaN
172	31	in-state	Female	nan	DGS	NaN	NaN
173	34	in-state	Male	nan	DGS	NaN	NaN
174	32	in-state	Male	nan	DGS	DC	NaN

175 rows × 7 columns

Figure 4.62 Clean the city in dataset

2. Then the data in [SCHOLARSHIP] column is converted into with_scholarship and without_scholarship as "%25 ÖSYM İndirimi, %50 ÖSYM İndirimi, and ÖSYM Burslu" are changed to with_scholarship and "Atilim Burssuz" is changed to without_scholarship. As shown in Figure 4.63:

Clean the scholarship column by converting the data into with_scholarship and without_scholarship

In [51]:
```
def clean_scholarship(x):
    if '%25 ÖSYM İndirimi' in x:
        return 'with_scholarship'
    if '%50 ÖSYM İndirimi' in x:
        return 'with_scholarship'
    if 'ÖSYM Burslu' in x:
        return 'with_scholarship'
    else:
        return 'without_scholarship'
df_COURSE['SCHOLARSHIP'] = df_COURSE['SCHOLARSHIP'].apply(clean_scholarship)
df_COURSE
```

Out[51]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG102	ENG201
0	26	out-of-state	Female	with_scholarship	DGS Ek Yerleştirme	NaN	NaN
1	37	in-state	Female	with_scholarship	ÖSS	NaN	AA
2	26	in-state	Male	with_scholarship	ÖSS	CB	CB
3	25	in-state	Female	with_scholarship	ÖSS	AA	AA
4	22	in-state	Male	with_scholarship	ÖSS	AA	BA
...	...	...	...	...	...	...	...
170	30	in-state	Male	without_scholarship	DGS	CB	NaN
171	43	in-state	Male	without_scholarship	DGS	CC	NaN
172	31	in-state	Female	without_scholarship	DGS	NaN	NaN
173	34	in-state	Male	without_scholarship	DGS	NaN	NaN
174	32	in-state	Male	without_scholarship	DGS	DC	NaN

175 rows × 7 columns

Figure 4.63 Clean the scholarship in dataset

3. Then the data in [REGISTRATION _REASON] column is converted into with_exam and without_exam as “DGS Ek Yerleştirme, and DGS” are changed to without_exam and “ÖSS, and ÖSS Ek Kontenjan” are change to with_exam. As shown in Figure 4.64:

Clean the registration_reason column by converting the data into with_exam and without_exam

In [52]:
```
def clean_registration_reason(x):
    if 'DGS Ek Yerleştirme' in x:
        return 'without_exam'
    if 'DGS' in x:
        return 'without_exam'
    else:
        return 'with_exam'
df_COURSE['REGISTRATION_REASON'] = df_COURSE['REGISTRATION_REASON'].apply(clean_registration_reason)
df_COURSE
```

Out[52]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG102	ENG201
0	26	out-of-state	Female	with_scholarship	without_exam	NaN	NaN
1	37	in-state	Female	with_scholarship	with_exam	NaN	AA
2	26	in-state	Male	with_scholarship	with_exam	CB	CB
3	25	in-state	Female	with_scholarship	with_exam	AA	AA
4	22	in-state	Male	with_scholarship	with_exam	AA	BA
...	...	...	...	...	...	...	...
170	30	in-state	Male	without_scholarship	without_exam	CB	NaN
171	43	in-state	Male	without_scholarship	without_exam	CC	NaN
172	31	in-state	Female	without_scholarship	without_exam	NaN	NaN
173	34	in-state	Male	without_scholarship	without_exam	NaN	NaN
174	32	in-state	Male	without_scholarship	without_exam	DC	NaN

175 rows × 7 columns

Figure 4.64 Clean the registration_reason in dataset

4. Clean the dataset out any null values. As shown in Figure 4.65:

Check the dataset for any null values

```
In [43]:  df_COURSE = df_COURSE.replace(np.nan, 0)
          df_COURSE.isnull().sum()

Out[43]:  AGE                    0
          CITY                   0
          GENDER                 0
          SCHOLARSHIP            0
          REGISTRATION_REASON    0
          ENG102                 0
          ENG201                 0
          dtype: int64
```

Figure 4.65 Remove all null values in the dataset

5. Thereafter, the whole data in [CITY, SCHOLARSHIP, and REGISTRATION_REASON] columns are encoded. As shown in Figure 4.66:

Encode city, scholarship, registration_reason and both courses columns

```
In [53]:  CITY = LabelEncoder()
          GENDER = LabelEncoder()
          SCHOLARSHIP = LabelEncoder()
          REGISTRATION_REASON=LabelEncoder()
          selected_course_code_le = LabelEncoder()
          course_code_le = LabelEncoder()

          df_COURSE['CITY'] = CITY.fit_transform(df_COURSE['CITY'])
          df_COURSE['GENDER'] = GENDER.fit_transform(df_COURSE['GENDER'])
          df_COURSE['SCHOLARSHIP'] = SCHOLARSHIP.fit_transform(df_COURSE['SCHOLARSHIP'])
          df_COURSE['REGISTRATION_REASON'] = REGISTRATION_REASON.fit_transform(df_COURSE['REGISTRATION_REASON'])
          df_COURSE[course_code] = course_code_le.fit_transform(df_COURSE[course_code])
          df_COURSE[selected_course_code] = selected_course_code_le.fit_transform(df_COURSE[selected_course_code])
          df_COURSE
```

Out[53]:

	AGE	CITY	GENDER	SCHOLARSHIP	REGISTRATION_REASON	ENG102	ENG201
0	26	1	0	0	1	9	6
1	37	0	0	0	0	9	0
2	26	0	1	0	0	3	3
3	25	0	0	0	0	0	0
4	22	0	1	0	0	0	1
...	...	...	...	...	...	...	...
170	30	0	1	1	1	3	6
171	43	0	1	1	1	4	6
172	31	0	0	1	1	9	6
173	34	0	1	1	1	9	6
174	32	0	1	1	1	5	6

175 rows × 7 columns

Figure 4.66 Encode all cleaned labels in dataset

6. Then split the dataset into two parts: X (independent variables) and y (dependent variable) which is also called predictable variable. As shown in Figure 4.67:

Divid the dataset into independent and dependent variables X and y

```
In [27]:  X = df_COURSE.drop(course_code, axis=1)
          y = df_COURSE[course_code]
```

Figure 4.67 Split the dataset in X, and y for next term course grade prediction

7. Split the dataset in test and train sets and make it ready for fitting according different ML algorithms. As shown in Figure 4.68:

Split the dataset in test and train sets and then select and fit the ML algorithm

```
In [28]:  regressor = DecisionTreeRegressor(random_state=0)
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
          regressor.fit(X_train, y_train)

Out[28]:  DecisionTreeRegressor
          DecisionTreeRegressor(random_state=0)
```

Figure 4.68 Select ML algorithm for predicting next term course grade

8. Create a set of predictor labels for predicting the graduation time. As shown in Figure 4.69:

Create an array of predictor labels for predicting the next term course grade

```
In [57]:  X = np.array([[25,'in-state','Female',"with_scholarship","with_exam",'BA']])
          X[:, 1] = CITY.transform(X[:, 1])
          X[:, 2] = GENDER.transform(X[:, 2])
          X[:, 3] = SCHOLARSHIP.transform(X[:, 3])
          X[:, 4] = REGISTRATION_REASON.transform(X[:, 4])
          X[:, 5] = selected_course_code_le.transform(X[:, 5])
          # X = np.array([['BA']])
          X = X.astype(float)
          X

Out[57]:  array([[25.,  0.,  0.,  0.,  0.,  1.]])
```

Figure 4.69 Select a course to find its related course

9. Predicting the related course grade using selected course grade. As shown in Figure 4.70:

Predict the related course grade

```
In [30]:  y_pred = regressor.predict(X)
          y_pred

Out[30]:  array([4.85714286])
```

Figure 4.70 Predict the related course grade

10. Decoding the predicted value to display as a letter grade. As shown in Figure 4.71:

Decode the predicted value to make it as letter grades

```
In [31]: y_pred = course_code_le.inverse_transform([round(y_pred[0])])
         y_pred[0]

Out[31]: 'DC'
```

Figure 4.71 Decode the related course grade

11. And finally displaying the result in a data frame. As shown in Figure 4.72:

Display the result in dataframe

```
In [36]: result = pd.DataFrame({
                 "SELECTED COURSE": [title],
                 "RELATED_COURSE_CODE": [course_code],
                 "RELATED_COURSE_TITLE": [course_title],
                 "SEMESTER": [semester],
                 "ESTIMATED GRADE": [y_pred[0]],
             })

         result

Out[36]:
```

	SELECTED COURSE	RELATED_COURSE_CODE	RELATED_COURSE_TITLE	SEMESTER	ESTIMATED GRADE
0	Computer Programming II	CMPE225	Object-Oriented Programming	3	DC

Figure 4.72 Display the next term course grade prediction results in a data frame

CHAPTER 5

RESULTS:

In this study the results are based on two types of data analysis (continues, and discrete data) using different ML algorithms on the existing dataset. Therefore, DT, NB, and SVM algorithms are preferred for answering the research questions 1 through 3 as these algorithms can be used for both classification and regression models:

1. The values stored in [COURSE] and [GPA] columns of the extracted dataset are all in continuous form. So, it is impossible to classify these kind data.
2. The values stored in [DURATION] columns of the extracted dataset are all in discrete form. thus, it is possible to classify the duration and fitting the model.

The results of both classification and regression models using all ML algorithms which are chosen for this study with their evaluation report will be described as following:

5.1. Regression Model for Predicting Final Grades (RQ 1)

The evaluation report for continuous data is only available with the calculation of RMSE, MAE, and R2 score the other metrics (Accuracy, Precision, Recall, F1 score) are only suitable for classification models. confusion matrix is not available for regression models.

5.1.1. DT (Decision Tree Regressor)

Only some of the performance metrics such as RMSE, MAE, and R2 score are suitable for evaluating the regression models. So, the available performance metrics for Decision Tree Regressor are described in Table 5.1.

Table 5.1 The performance report for DT (Decision Tree Regressor) algorithm for predicting final grades

RMSE	**MAE**	**R^2 score**	**Accuracy**	**Precision**	**Recall**	**F1 score**
0.3254	0.5263	0.5068	NULL	NULL	NULL	NULL

5.1.2. SVM (Support Vector Machine Regressor)

The same as DT algorithm in regression models, for SVM some performance metrics such as accuracy, precision, recall, and F1 score are also not available. The rest of performance metrics results are described Table 5.2.

Table 5.2 The performance report for SVM (Support Vector Machine Regressor) algorithm for predicting final grades

RMSE	**MAE**	**R^2 score**	**Accuracy**	**Precision**	**Recall**	**F1 score**
0.266	0.4601	0.6705	NULL	NULL	NULL	NULL

5.1.3. LR (Linear Regression)

The same as DT algorithm in regression models, for SVM some performance metrics such as accuracy, precision, recall, and F1 score are also not available. The rest of performance metrics results are described in Table 5.3.

Table 5.3 The performance report for LR (Linear Regression) algorithm for predicting final grades

RMSE	**MAE**	**R^2 score**	**Accuracy**	**Precision**	**Recall**	**F1 score**
0.2715	0.4722	0.6568	NULL	NULL	NULL	NULL

5.1.4. RF (Random Forest)

The same as DT algorithm in regression models, for SVM some performance metrics such as accuracy, precision, recall, and F1 score are also not available. The rest of performance metrics results are described in Table 5.4.

Table 5.4 The performance report for RF (Random Forest Regressor) algorithm for predicting final grades

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
0.2753	0.4774	0.647	NULL	NULL	NULL	NULL

As a result, in our analysis for predicting final grades SVM with lowest error rates RMSE: 0.266, and MAE: 0.4601 and highest accuracy R2 score: 0.6705 is selected as the most accurate algorithm. Figure 5.1 and figure 5.2 illustrate the performance of different algorithms.

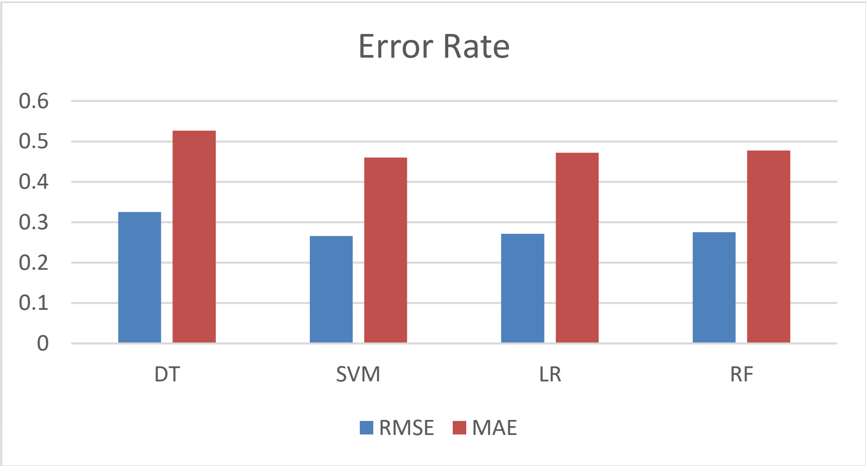


Figure 5.1 The error rates graph for all ML algorithms used for predicting final grades

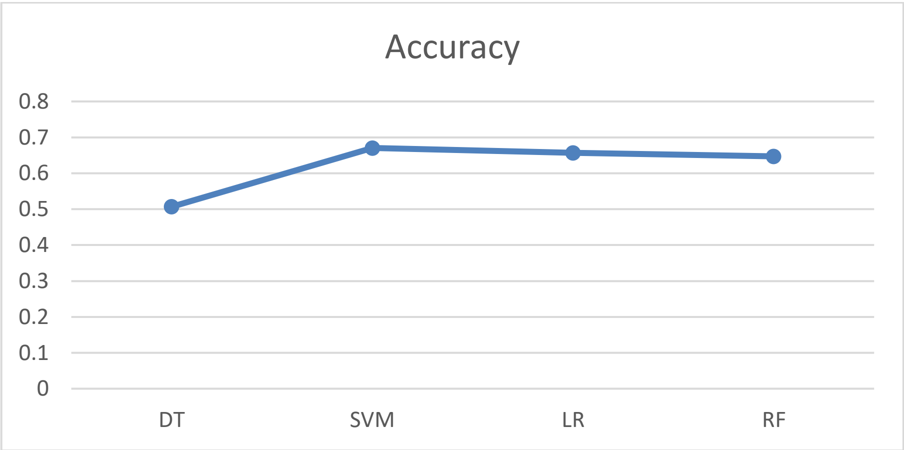


Figure 5.2 The accuracy results graph for all ML algorithms used for predicting final grades

5.2. Classification Model for Predicting Study Duration (RQ 2)

In this model all four ML algorithms evaluated by seven accuracy identifiers which are mentioned with their respective confusion matrices.

5.2.1. DT (Decision Tree Classifier)

Table 5.5 and Figure 5.3 respectively define the metrics performance and confusion matrix for Decision Tree Classifier:

Table 5.5 The performance report for DT (Decision Tree) algorithm for predicting Study Duration

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
0.7171	0.6325	0.4878	0.6571	0.3582	0.4583	0.3494

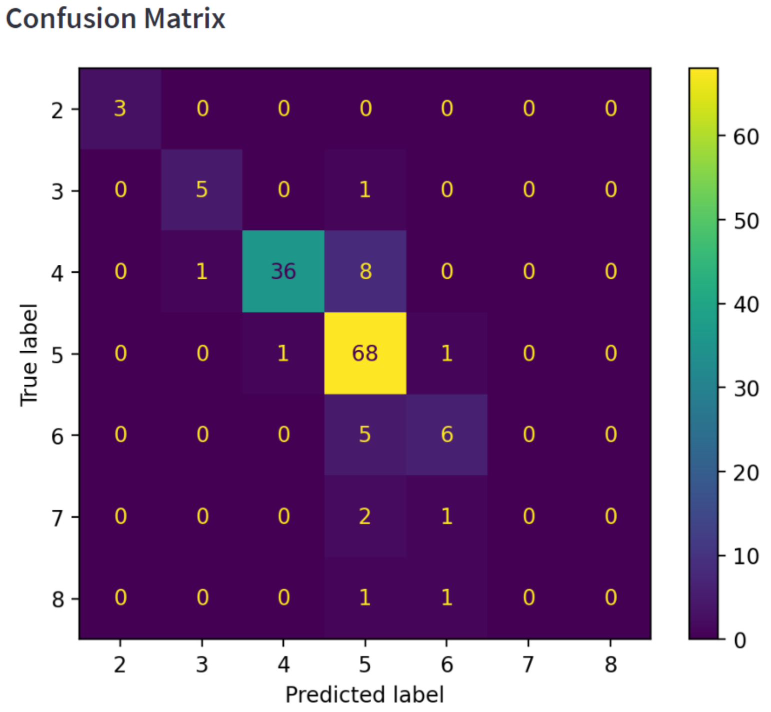


Figure 5.3 Shows the confusion matrix for DT (Decision Tree) algorithm

5.2.2. SVM (Support Vector Machine Classifier)

Table 5.6 and Figure 5.4 show the accuracy results and confusion matrix for the selected ML algorithm.

Table 5.6 The performance report for SVM (Support Vector Machine) algorithm for predicting graduation duration

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
0.7746	0.6094	0.4024	0.7143	0.4583	0.4833	0.426

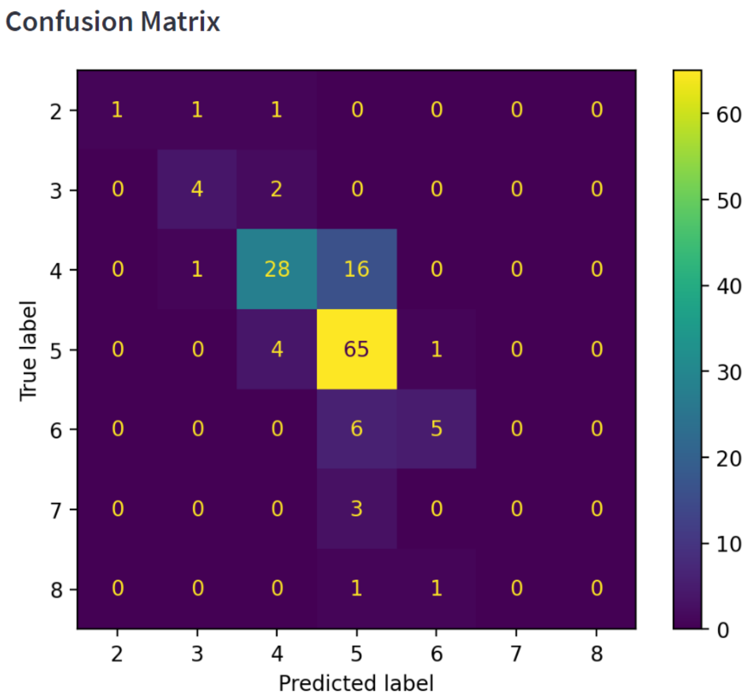


Figure 5.4 Shows the confusion matrix for SVM (Support Vector Machine) algorithm

5.2.3. NB (Naïve Bayes Classifier)

Table 5.7 and Figure 5.5 show the performance metrics results and confusion matrix for Naïve Bayes Classifier.

Table 5.7 The performance report for NB (Naïve Bayes Classifier) algorithm for predicting graduation duration

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
0.9103	0.6969	0.1748	0.6571	0.294	0.4	0.3304

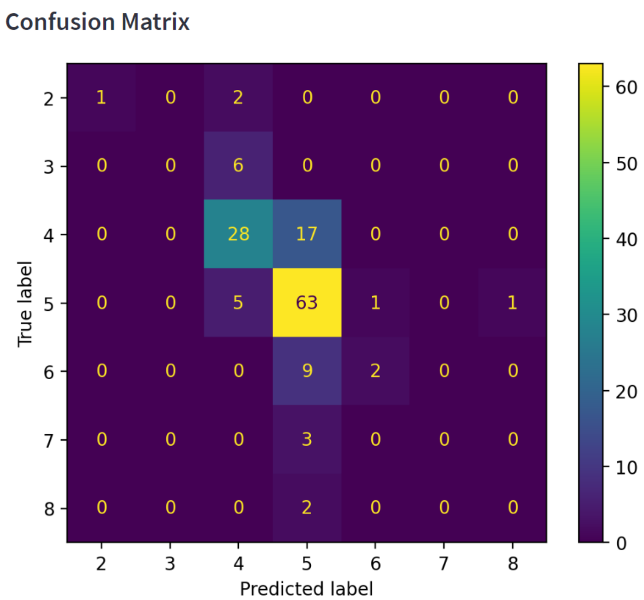


Figure 5.5 Shows the confusion matrix for NB (Naïve Bayes) algorithm

5.2.4. RF (Random Forest Classifier)

Table 5.8 and Figure 5.6 show the performance metrics results and confusion matrix for Random Forest Classifier.

Table 5.8 The performance report for RF (Random Forest Classifier) algorithm for predicting graduation duration

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
0.7559	0.6325	0.4309	0.6571	0.438	0.4583	0.3768

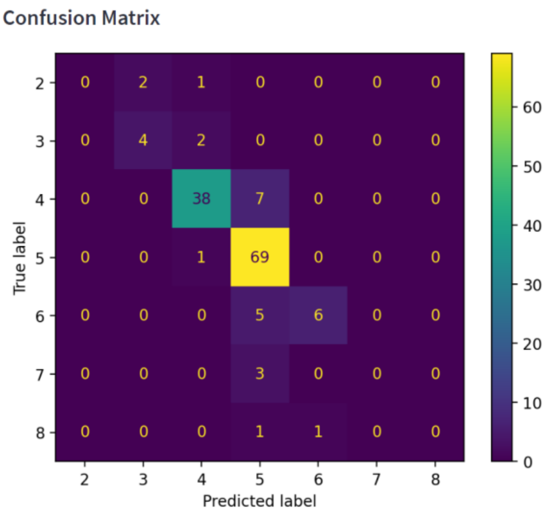


Figure 5.6 Confusion matrix for RF (Random Forest) algorithm

As a result, in our analysis for predicting study duration SVM with error rates RMSE: 0.7746, and MAE: 0.6094 and accuracies R2 score: 4024, Accuracy: 0.7143, Precision: 0.4583, Recall: 0.4833, and F1 score: 0.426 and DT with error rates RMSE: 0.7171, and MAE: 0.6325 and accuracies R2 score: 4878, Accuracy: 0.6571, Precision: 0.3582, Recall: 0.4583, and F1 score: 0.3494 are selected as the most accurate algorithms. Figure 5.7 and figure 5.8 illustrate the performance of different algorithms.

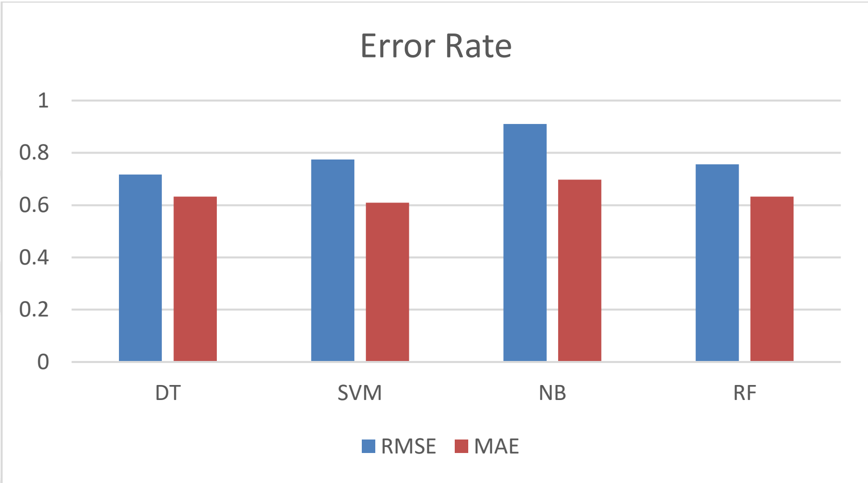


Figure 5.7 The error rates graph for all ML algorithms used for predicting study duration

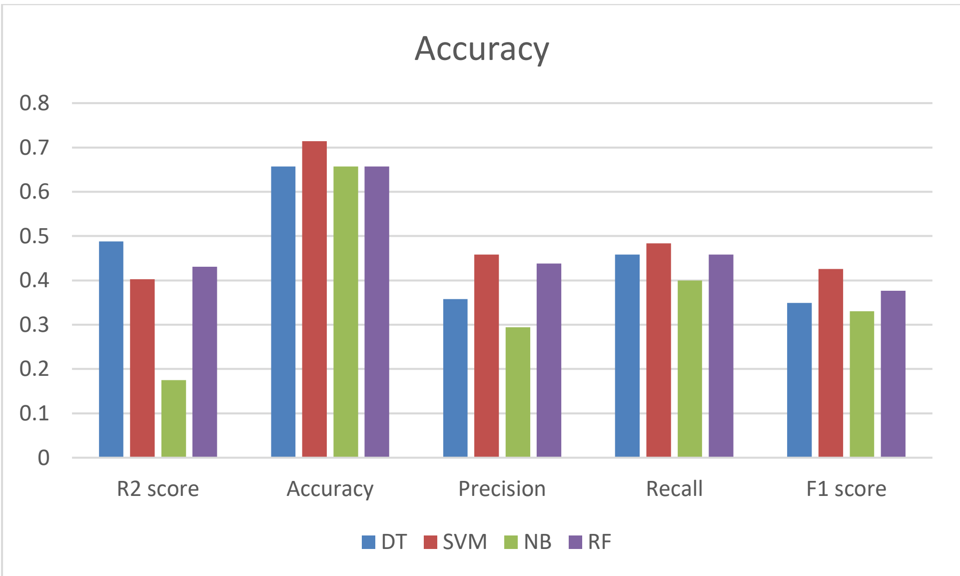


Figure 5.8 The accuracy graph for all ML algorithms used for predicting study duration

5.3. Regression Model for Predicting Next Term Course Grades (RQ 3)

For predicting the next term course grades, we decided to use regression model since it has better performance for this aspect. Thus, for evaluating our model we used RMSE, MAE, and R2 score as the other metrics are not available for regression models.

5.3.1. DT (Decision Tree Regressor)

Only some of the performance metrics such as RMSE, MAE, and R2 score are suitable for evaluating the regression models. So, the available performance metrics for Decision Tree Regressor are described Table 5.9:

Table 5.9 The performance report for DT (Decision Tree Regressor) algorithm for predicting next term course grades

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
2.5602	1.4677	0.247	NULL	NULL	NULL	NULL

5.3.2. SVM (Support Vector Machine Regressor)

The same as DT algorithm in regression models, for SVM some performance metrics such as accuracy, precision, recall, and F1 score are also not available. The rest of performance metrics results are described in Table 5.10:

Table 5.10 The performance report for SVM (Support Vector Machine Regressor) algorithm for predicting next term course grades

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
2.5019	1.4658	0.281	NULL	NULL	NULL	NULL

5.3.3. LR (Linear Regression)

The same as DT algorithm in regression models, for SVM some performance metrics such as accuracy, precision, recall, and F1 score are also not available. The rest of performance metrics results are described in Table 5.11:

Table 5.11 The performance report for LR (Linear Regression) algorithm for predicting next term course grades

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
2.8409	1.4944	0.0729	NULL	NULL	NULL	NULL

5.3.4. RF (Random Forest)

The same as DT algorithm in regression models, for SVM some performance metrics such as accuracy, precision, recall, and F1 score are also not available. The rest of performance metrics results are described in Table 5.12:

Table 5.12 The performance report for RF (Random Forest Regressor) algorithm for predicting next term course grades

RMSE	MAE	R^2 score	Accuracy	Precision	Recall	F1 score
2.8302	1.5212	0.0799	NULL	NULL	NULL	NULL

As a result, in our analysis for predicting final grades SVM with lowest error rates RMSE: 2.5019, and MAE: 1.4658 and highest accuracy R2 score: 0.281 is selected as the most accurate algorithm. Figure 5.9 and figure 5.10 illustrate the performance of different algorithms.

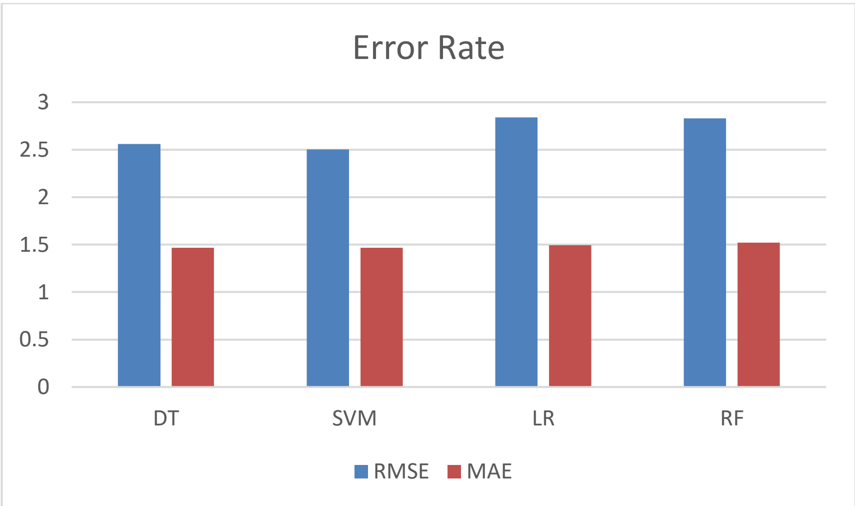


Figure 5.9 The error rates graph for all ML algorithms used for predicting next term course grades

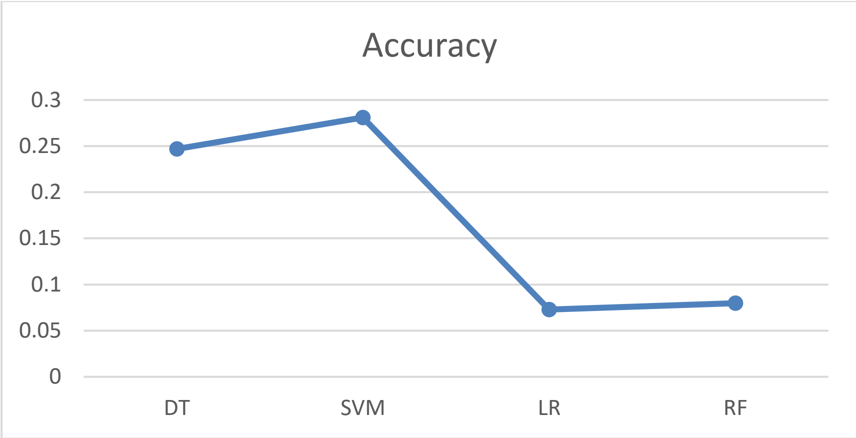


Figure 5.10 The accuracy graph for all ML algorithms used for predicting next term course grades

CHAPTER 6

DISCUSSION AND CONCLUSION:

In this chapter we are trying to explain the reasons behind the results which are obtained from predicting three different parameters respectively Final Grades, Study Duration, and Next Term Course Grades.

6.1. Final Grades

In this study we used some demographic information and the letter grades for each course for first four semesters to predict the final grades. It shows that demographic data does not have significant impact on predicting final grades. We used different ML algorithms to examine our model for finding the best algorithm for this purpose. There are different accuracy results for each algorithm since we calculated the error rate by RMSE and MAE and the accuracy with R2 score evaluation metrics. The performance results indicated that Support Vector Machine has the lowest error rate in predicting final grades. So, the Support Vector Machine algorithm with RMSE: 0.2689 and MAE: 0.4608 outperformed the other 3 algorithms (Decision Tree, Support, Linear Regression, and Random Forest).

In a similar study for predicting students' final grades [3] which students demographic data such as age, gender, marital status, nationality, displaced (whether the student lived outside the Proto district), scholarship, special needs, type of admission, type of student (regular, mobility, extraordinary), status of student (ordinary, employed, athlete …), years of enrolment, delayed courses, type of dedication (full-time, part-time), and debt situation are used as input variables and different ML algorithms including Ordinary Least Squares (OLS), SVM, CART, kNN, Random Forest, and AdaBoost.R2 (AB.R2) are used and evaluated by F1 score and RMSE. Therefore, the results shows that more features like academic goals, personal interests, time management skills, sport activities, and sleep habits can help the analyses to compare the algorithms better.

In another similar study for predicting final exam grades [5], midterm, faculty, and department data are used as input variables. The results showed that students' midterm exam grades are as the most important feature for predicting students' final grades. RF, NN, and SVM are chosen as the most accurate data mining algorithms for this purpose.

And in another study using DT algorithm [21] the results showed that it is possible to predict students' performance with using pre-university marks, and the first two years courses' marks as prediction variables.
Table 6.1 and figure 6.1 illustrates the RMSE and MAE results for predicting final grades in this study.
Support Vector Machine with R2 score: 0.6633 also has the highest accuracy across other three algorithms (Decision Tree, Linear Regression, and Random Forest). As a result, Support Vector Machine Algorithm is the best algorithm for predicting students' final grades. Table 6.2 and figure 6.2 illustrates the R2 results for predicting final grades in this study

6.2. Study Duration

For predicting study duration, we decided to use some demographic information and the GPA for first four semesters. In this purpose demographic data also did not perform any impressive impact on the result of predicting study duration. For this purpose, we chose different ML algorithm for examining our model to find the best fitted algorithm. There are different accuracy results for each algorithm since we calculated the error rate by RMSE and MAE and the accuracy with R2 score, Accuracy, Precision, Recall, and F1 score evaluation metrics. Therefore, the performance results showed that Decision Tree algorithm with lowest error rate RMSE: 0.6969 and MAE: 0.6064 made the best result and outperform the three other algorithm (Support Vector Machine, Naïve Bayes, and Random Forest) were chosen for this purpose.
In a similar study for predicting graduation time [7], some factors including GPA, Gender, and Marital status are investigated and the result indicated that gender and marital status are effective on student performance. NN (Neural Network), and SVM (Support Vector Machine) are known as the best algorithms in this study.

In another similar study [8] beside the general demographic data, three more factors: students' in-state and out-of-state address, college major, and student learning type (tutoring and supplemental instructions) are investigated and the results showed that student's home address and learning type have significant impact on predicting students' graduation time.

And in another study for predicting graduation time [9] after examining different variables, semester achievement index is chosen as the best predictor variable. In this study SVM is known as the best ML algorithm.

Table 6.3 and figure 6.3 illustrates the RMSE and MAE results for predicting study duration in this study.

Decision Tree algorithm with the highest accuracy of R2 score: 0.5163, Accuracy: 0.6857, Precision: 0.4459, Recall: 0.4667, and F1 score: 0.3687 made the best result other than three other algorithms (Support Vector Machine, Naïve Bayes, and Random Forest). Thus, in predicting students' study duration also Random Forest performed the best result and known as the best algorithm for this purpose. Table 6.4 and figure 6.4 illustrates the R2, Accuracy, Precision, Recall, and F1 score results for predicting study duration in this study.

6.3. Next Term Course Grade

In this study, for predicting next term course grades we need to consider to purposes, first finding the most similar future course for the selected previous course then applying the prediction algorithm on the extracted dataset. Therefore, from the dataset which we have extracted from course curriculum list we only chose the COURSE_TITLE attribute for the recommendation process. In this way we could successfully find the most similar course to the selected course. After that, we used different ML algorithms for examining our model for predicting next term course grades. For this purpose, the regression model is chosen and has been evaluated by RMSE, MAE, and R2 score validation metrics. The results indicates that SVM (Support Vector Machine) with the lowest error rate of RMSE: 2.4103 and MAE: 1.3587 is known as the best fitted algorithm for predicting next term course grades.

In a similar study for predicting future course grades [12] CF (Collaborative Filtering), and MF (Matrix Factorization) methods are examined. The results showed that MF model is the most appropriate method for this purpose.

The results for the accuracy of our model for predicting next term course grades also shows the SVM with the R2 score: 0.3326 as the best algorithm for this aspect. So, according to this results SVM is selected as the best ML algorithm for predicting next term course grades.

6.4. Selecting the Best Algorithm for this Study

In this study our purpose for predicting student academic performance is based on two main analysis regression and classification. Since, we decided to predict Students' final grades, study duration, and next term course grades which some of them are in continuous form and some of them are discrete form, we used different classification and regression to find the best results for our purposes. After creating and examining our model with different ML algorithms as the results indicates we selected SVM as the best algorithm for predicting final grades and next term course grades, and DT and SVM for predicting study duration.

6.5. Selecting the Best Dataset for this Study

We used two main datasets in this study which one dataset includes students' transcript data and some demographic information and another that is extracted from course curriculum list and includes only mandatory courses. For making the better analysis processing, we extracted some sub datasets from the students' transcript dataset. Therefore, both datasets playing the important roles for this study for predicting students' academic performance.

6.6. Conclusions

The main purpose of this study is to identify the students who are at the risk of failure and need special attention. This study can help both students and teachers to make better decisions for their academic aspects such as selecting the most proper courses, being ready for new upcoming courses, and motivated for continuing the education progress. Therefore, we investigated Atilim University computer engineering undergraduate students' transcription data which are taken before for the purpose of predicting their academic performance including final grades, study duration, and next term course grades. For this reason, we examined students' personal and transcription data such as Age, Gender, City, GPA, and CGPA for their direct influence on student's academic performance. Additionally, we aimed to find the factors that have most impact on the students' success rate and finding the best ML algorithms to fit our purposes. Our finding indicates that the previous course results which are obtained by students have the higher impact on the predicting student's final grades, study duration, and next term course grades. In similar research for predicting final grades which they compare different ML algorithms, they found that more features like academic goals, personal interests, time management skills, sport activities, and sleep habits can help the analyses to compare the algorithms better [3]. In another similar study for predicting final grades, that midterm exam result is selected as the best predictor variable and RF, NN, and SVM are selected as the best algorithms [5]. In this research [6] GPA and earned credit are chosen as the best features for predicting final grades. In the study of [7] for predicting graduation time, Gender and Marital Status are selected as the best features and NN, and SVM are selected as the best algorithms. In another study [8], student's home address and learning type are chosen as the best predictor variables for predicting graduation time. In this study [9] for predicting graduation time, semester achievement index is chosen as the best feature and SVM algorithm with high accuracy rate is chosen for the prediction purpose. In a similar study for predicting future course grades [12] MF is selected as the most appropriate method for this purpose. Therefore, for our study it is considered that the better previous results have direct influence on better achievement of future results and vice versa. For the comparison of input datasets, it is better to know that each dataset plays its own role while the analysis will be only done with the contribution of all datasets and one dataset cannot make any results by itself in the analysis. SVM and DT algorithms are selected as the best fitted algorithms in this study, SVM is well performed algorithm for our regression models and DT is well performed algorithm for our classification model.

6.7. Limitations

This study faced some certain limitations. First, if we had more student demographic data such as marital status, income, sport activities, and parents' education we could have better results on predicting students' academic performance. Moreover, some student academic information like, entrance exam score, midterm exam grades, assignments, and class activities could cause better results. On the other hand, after preprocessing of the raw data we only had 175 students. Thus, the size of data has significant impact on the results and if we could have a greater number of students in the dataset the results might have been better. Moreover, missing data in the dataset which are caused by some student situations are ignored such as horizontal transfer and vertical transfer. Therefore, if we could use all missing data with proper values, it could help us to have better results for this purpose.

6.8. Future Work

For predicting students' academic performance in universities, midterm exam grades might be the best predictor. In a study performed by Mustafa Yağcı [5] it is claimed that midterm exam grades are the best predictors in predicting students' final grades. Therefore, the next study will consider midterm exam grades for predicting students' academic performance.

In Raheela Asif and her colleagues' study [21], pre-university marks and first two years courses' marks are used as factors for predicting students' performance. However, we have used first two years courses marks in our study but for getting better results the next study will also consider students' pre-university marks.

In a study by Masna Wati and her colleagues [14], age, culture (some cultural habits which force students to adapt the higher education environment, these habits can be derived by the place of birth), and participation levels (it can be active if participated in an organization while taking the study and can be passive if never participated in any organization while taking the study) are selected as factors that have some impacts on the students' graduation time. Thus, as we have already used age as a feature in our study, for getting better result we will use students' culture as a predictor feature in our future work.

CHAPTER 7

SOFTWARE IMPLEMENTATION:

7.1. Introduction

In data science area, however there many tools which we can use them for analyzing and or visualizing the data, but it is a good idea if we can develop our own applications to fulfill our own requirements for better understanding the domain.

7.2. Software Implementation Tools

In this study after analyzing the data and creating our model by Jupiter Notebook using python (a programming language), for applying our created model and visualizing the data in better and simpler way we decided to develop and design a web-based application for Computer Engineering department of Atilim University which can be accessible for both students and teachers. For this case, we preferred to use streamlit library a very amazing and powerful python library that include most of the data science classes which is very simple to design the application only by calling the specific properties of this library. As an example, for creating any component in our application such as a textbox simply we can use streamlit.text_input(). By using streamlit library we won't be worry about the design of application front-end by using HTML, CSS, and java script because all of these are already included in this built-in library.

7.3. Software Functionality

This web application is developed for predicting students' academic performance for computer engineering department using the students' personal information and transcript data for first 4 semesters. The functionalities of this application are respectively described as below:

- Uploading the dataset

- Preprocesses the uploaded dataset
- Examining different ML algorithms on the prepared dataset
- Creating the model for both classification and regression problems
- Fitting the model
- Generating the performance report
- Generating the prediction result
- And visualizing the data

7.4. UI (User Interface)

In this section the usage of all parts of the application with their functionality will be described. The UI parts are explained as below:

7.4.1. Login Form

Users can use this form to enter the application by their prepared username and password. As illustrated in Figure 7.1:

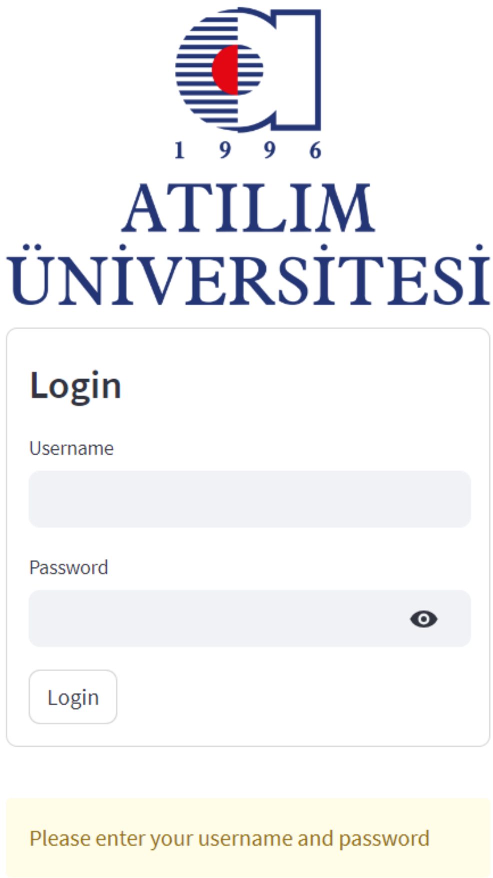


Figure 7.1 Application login form

7.4.2. General View of the Application

This is a general view of the application while the user logs in to the system which generally divided in two parts the sidebar section and main panel that will be described more in the next sections. As illustrated in Figure 7.2:

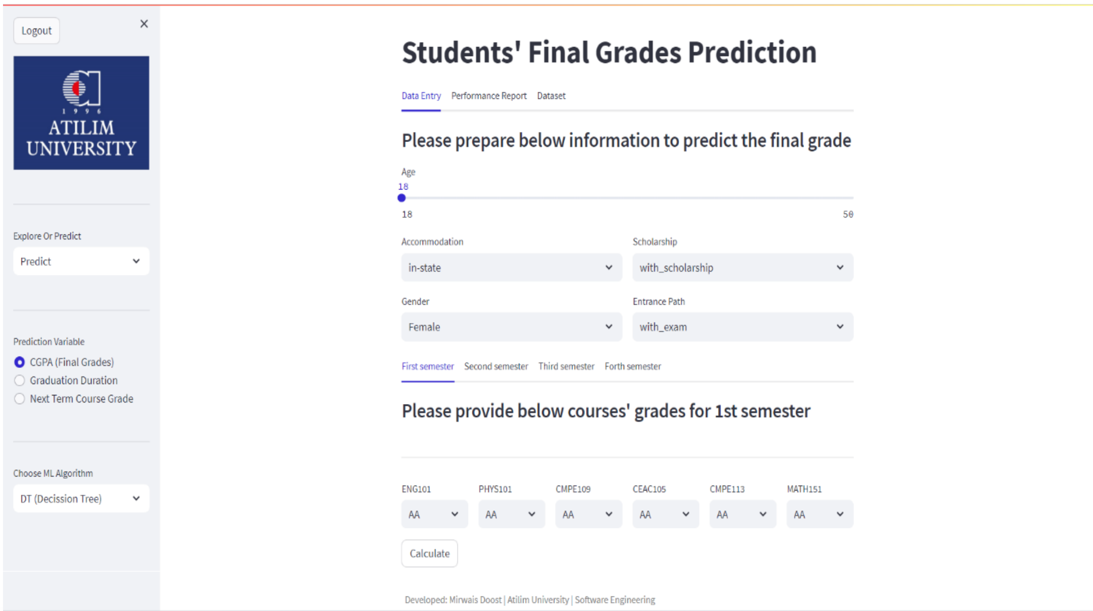


Figure 7.2 Application general view

7.4.3. Sidebar

This section includes the logout accessibility, switching the application in different parts, choosing the prediction reason, and selecting different types of ML algorithms. As illustrated in Figure 7.3:

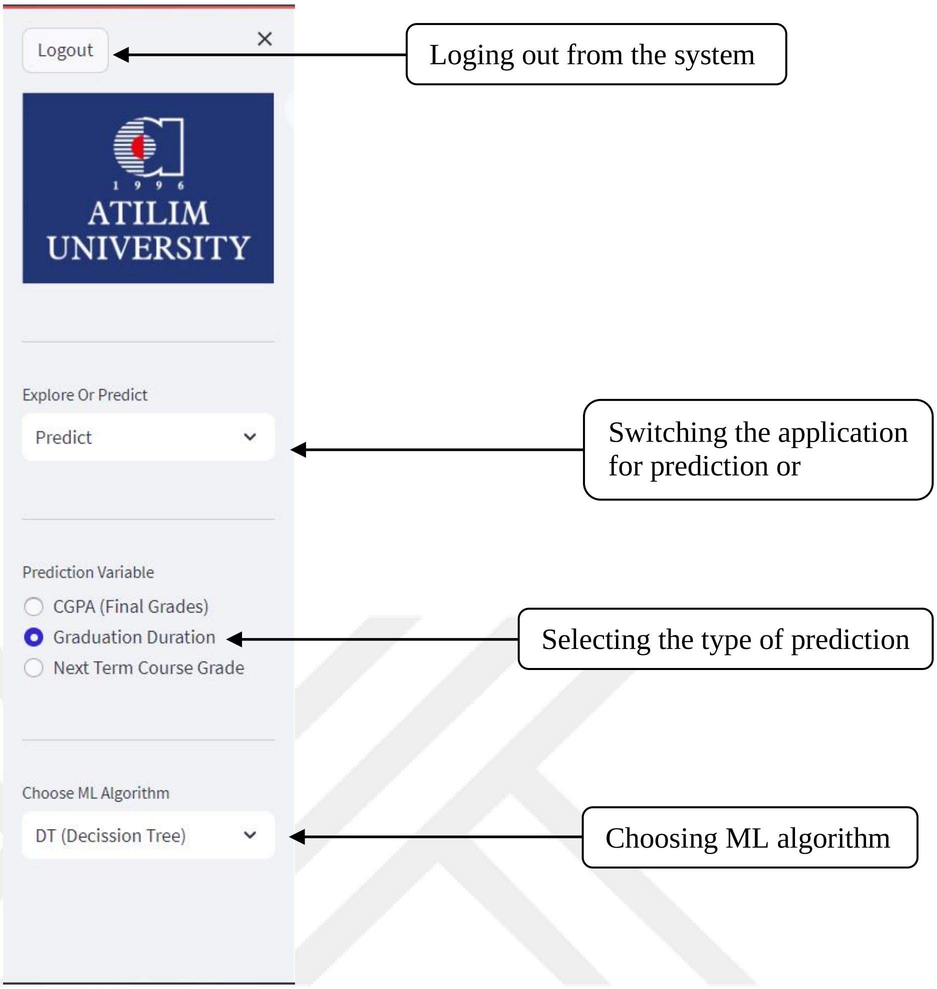


Figure 7.3 Application sidebar

7.4.4. Main Panel

In this section there are three different tabs (Data Entry, Performance Report, and Dataset) which each tab has its own usability.

7.4.5. Main Panel (Data Entry)

In this tab of main panel, the user will be requested to enter some information for doing the prediction depend on the selected prediction reason.

7.4.6. Main Panel (Data Entry for Predicting CGPA)

In this section, the user will be requested to enter the information for student's age, accommodation, scholarship, gender, entrance path, and the courses' grades for first four semesters to predict the student's final grade. As illustrated in Figure 7.4:

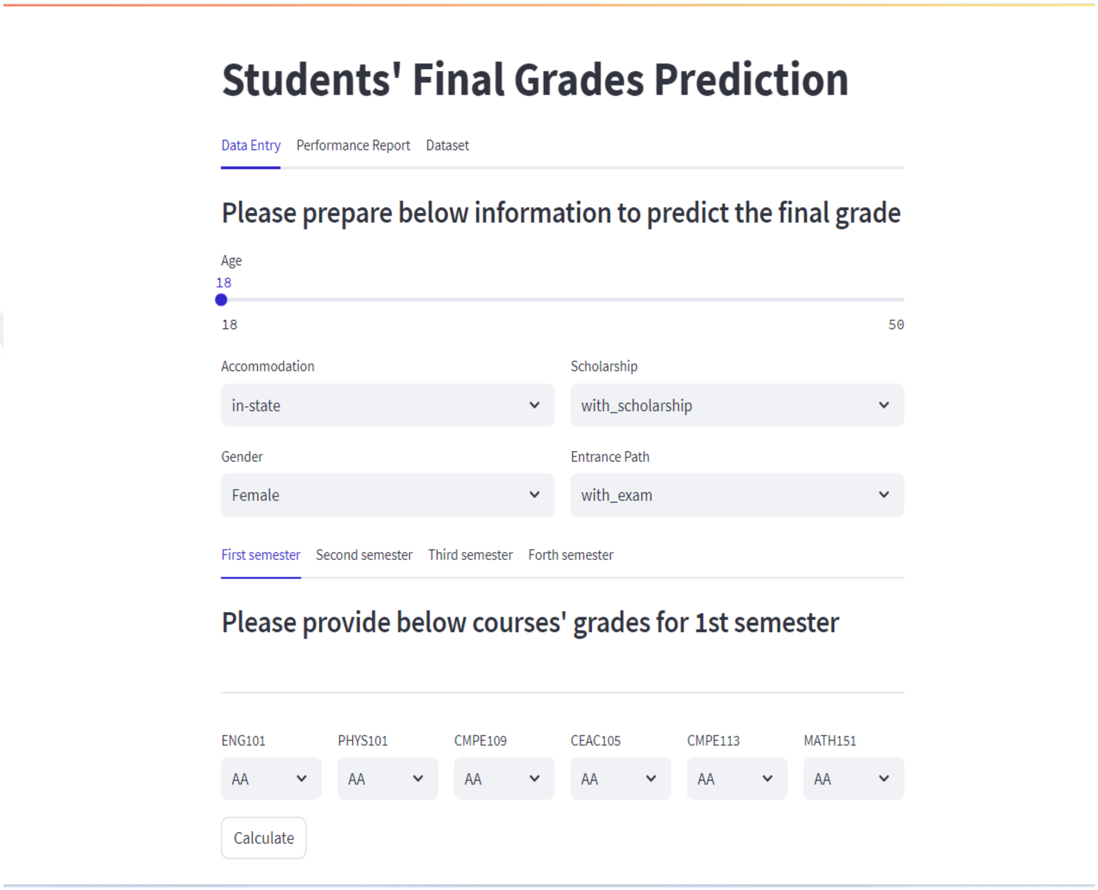


Figure 7.4 Application main panel (Data Entry for Predicting CGPA)

7.4.7. Main Panel (Data Entry for Predicting Duration)

In this section, the user will prepare the information for student's age, accommodation, scholarship, gender, entrance path, and the GPA for first four semesters to predict the student's Study Duration. As illustrated in Figure 7.5:

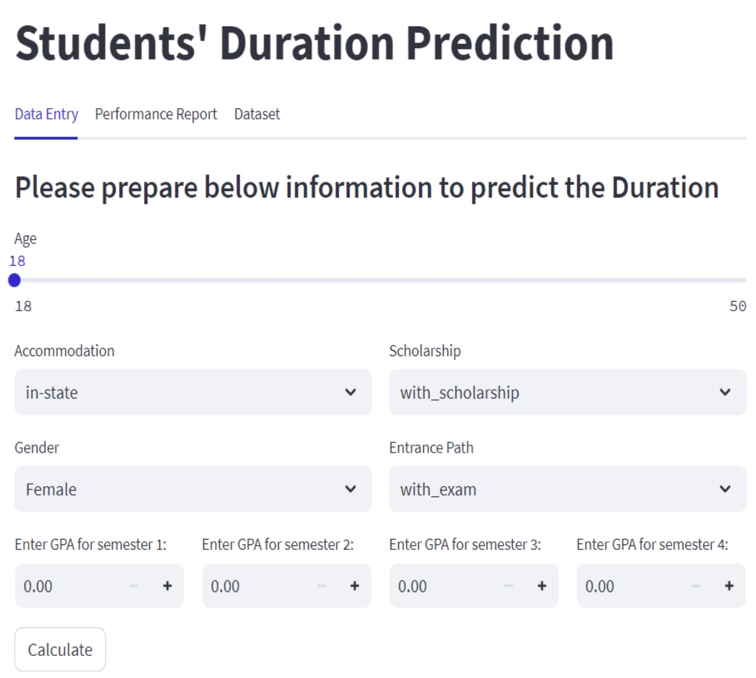


Figure 7.5 Application main panel (Data Entry for Predicting Duration)

7.4.8. Main Panel (Data Entry for Next Term Course Grade Prediction)

In this section, the user will prepare the information for student's age, accommodation, scholarship, gender, entrance path, course tile and the course grade for first four semesters to predict the student's Study Duration. As illustrated in Figure 7.6:

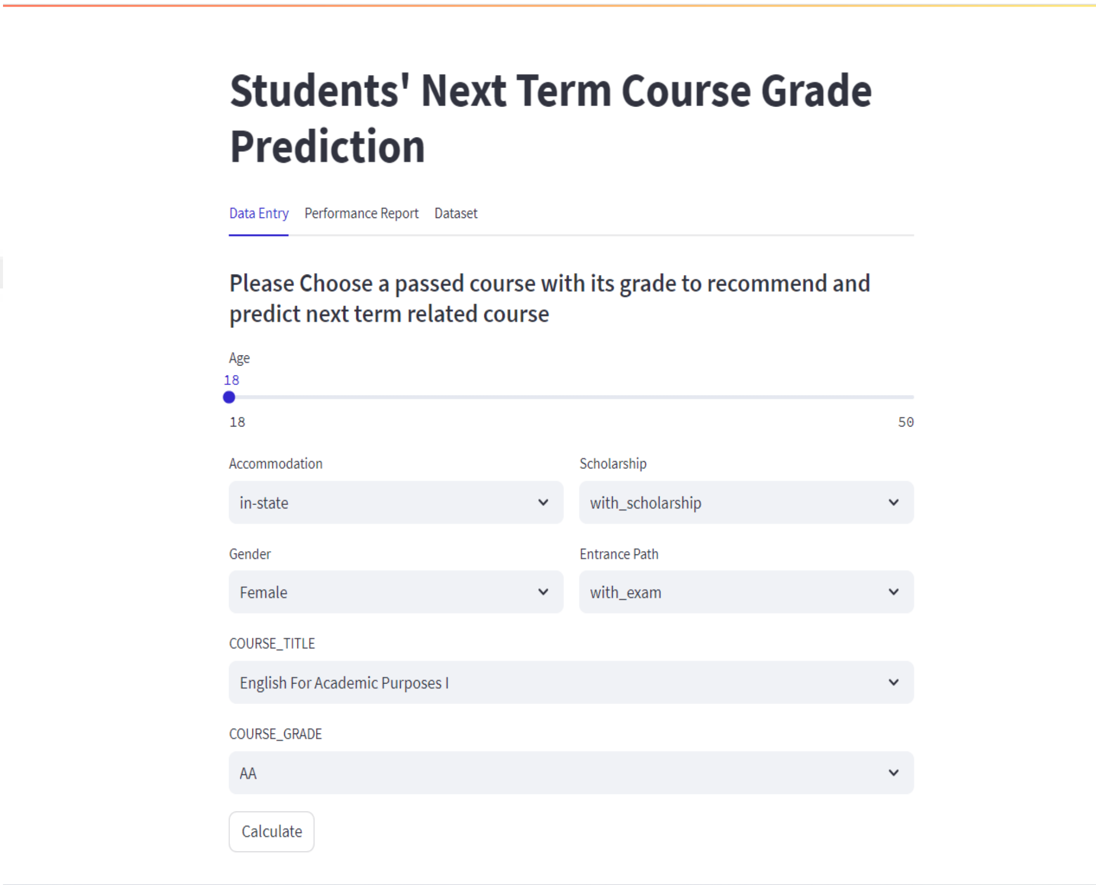


Figure 7.6 Application main panel (Data Entry for Predicting Next Term Course Grade)

7.4.9. Main Panel (Performance Report)

This tab in main panel includes the result of all performance metrics results for the selected prediction reason and selected ML algorithm. As illustrated in Figure 7.7:

Data Entry Performance Report Dataset

Report for the chosen ML algorithm

RMSE	MAE	R2 score	Accuracy	Precision	Recall	F1 score
0.6	0.2571	0.6036	0.7886	0.7886	0.7886	0.7886

Confusion Matrix

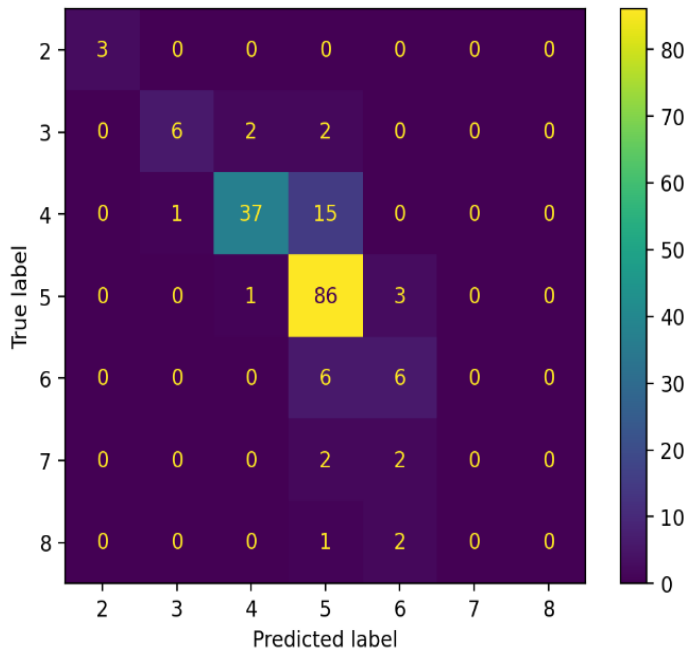
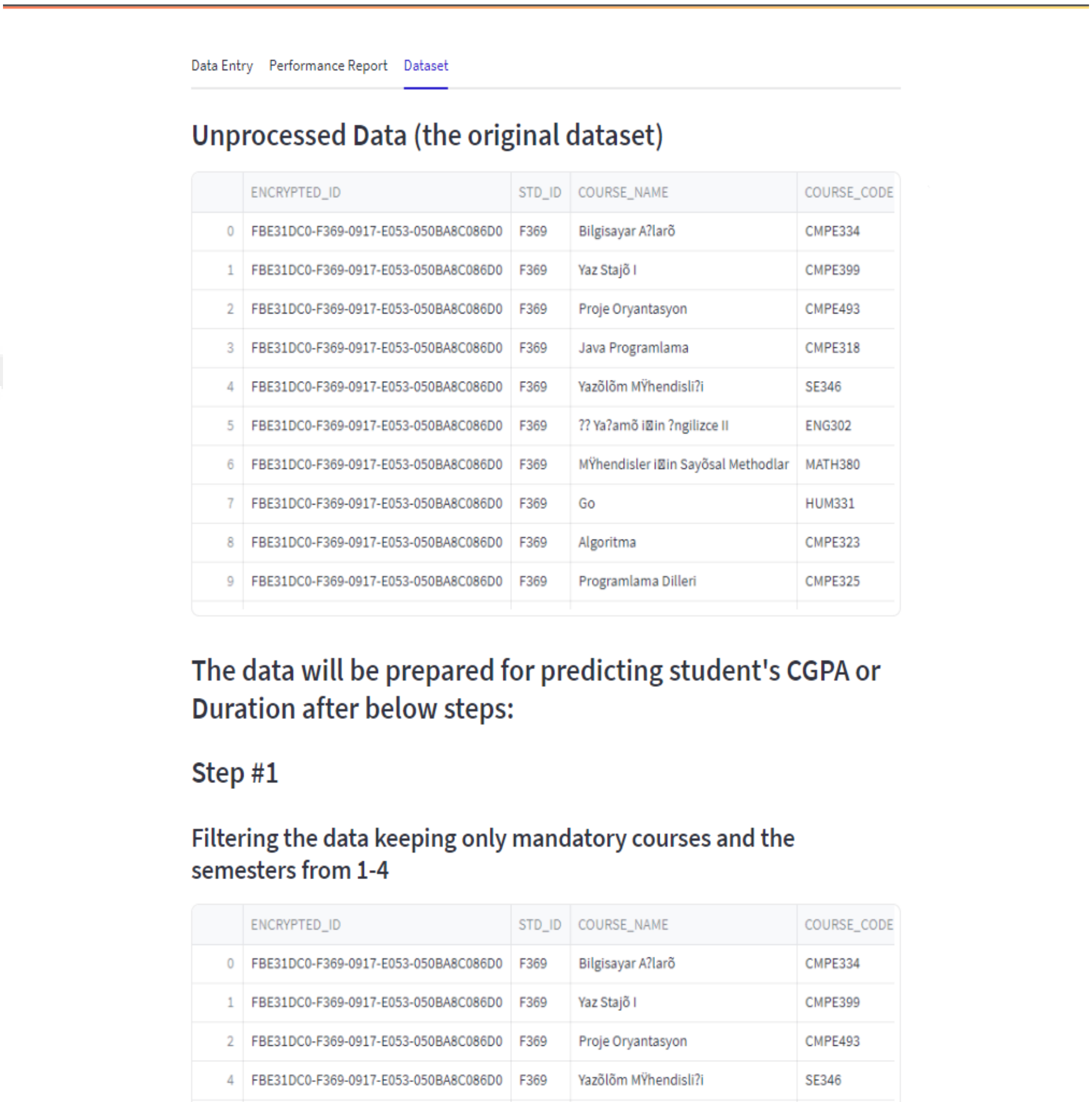


Figure 7.7 Application main panel (Performance Report)

7.4.10. Main Panel (Dataset)

In the tab all the steps for preprocessing of the dataset for different reason is described step by step. As illustrated in Figure 7.8:



Data Entry Performance Report Dataset

Unprocessed Data (the original dataset)

	ENCRYPTED_ID	STD_ID	COURSE_NAME	COURSE_CODE
0	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Bilgisayar A?larõ	CMPE334
1	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Yaz Stajõ I	CMPE399
2	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Proje Oryantasyon	CMPE493
3	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Java Programlama	CMPE318
4	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Yazõlõm MÝhendisli?i	SE346
5	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	?? Ya?amõ i☒in ?ngilizce II	ENG302
6	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	MÝhendisler i☒in Sayõsal Methodlar	MATH380
7	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Go	HUM331
8	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Algoritma	CMPE323
9	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Programlama Dilleri	CMPE325

The data will be prepared for predicting student's CGPA or Duration after below steps:

Step #1

Filtering the data keeping only mandatory courses and the semesters from 1-4

	ENCRYPTED_ID	STD_ID	COURSE_NAME	COURSE_CODE
0	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Bilgisayar A?larõ	CMPE334
1	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Yaz Stajõ I	CMPE399
2	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Proje Oryantasyon	CMPE493
4	FBE31DC0-F369-0917-E053-050BA8C086D0	F369	Yazõlõm MÝhendisli?i	SE346

Figure 7.8 Application main panel (Dataset)

7.4.11. Data Visualization Part

In this part the prepared dataset is visualized for different aspects. As illustrated in Figure 7.9:

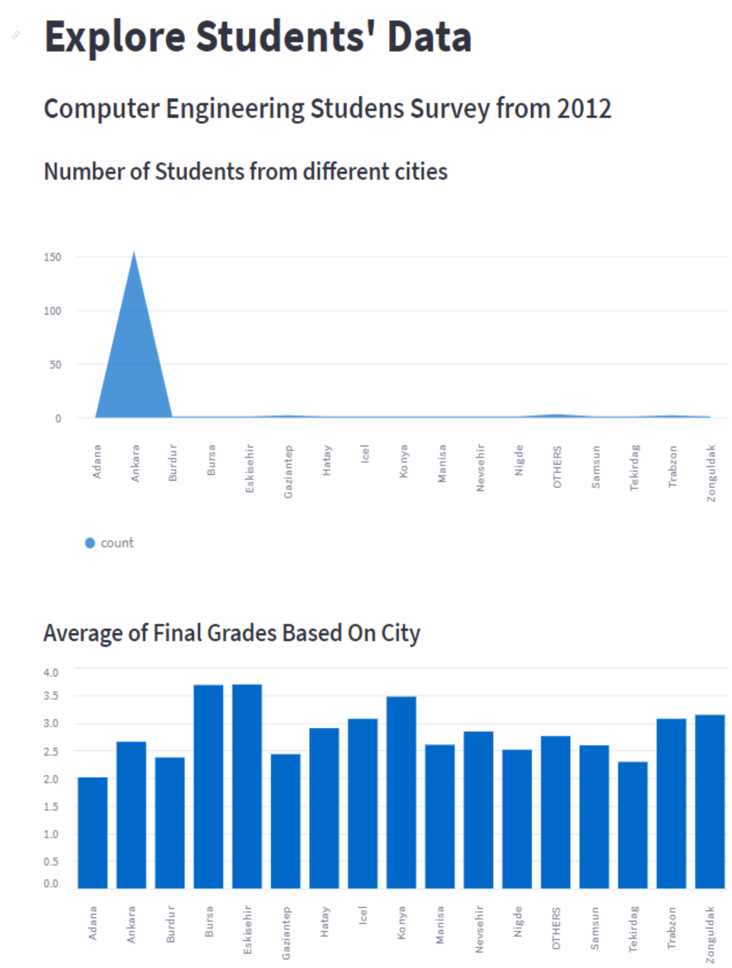


Figure 7.9 Application Data Visualization Part

REFERENCES

[1] Do, Q.H. and Chen, J.F., "A neuro-fuzzy approach in the classification of students' academic performance" *Computational intelligence and neuroscience*, 2013, pp.6-6.

[2] Yehuala, M.A., 2015. "Application of data mining techniques for student success and failure prediction (The case of Debre Markos university)" *International journal of scientific & technology research*, 4(4), pp.91-94.

[3] Strecht P, Cruz L, Soares C, Mendes-Moreira J, Abreu R. "A Comparative Study of Classification and Regression Algorithms for Modelling Students' Academic Performance" *International educational data mining society*, 2015 Jun.

[4] Kotsiantis, S.B., "Use of machine learning techniques for educational proposes: a decision support system for forecasting students' grades" *Artificial Intelligence Review*, 2012, pp.331-344.

[5] Yağcı, M., "Educational data mining: prediction of students' academic performance using machine learning algorithms" *Smart Learning Environments*, 2022, p.11.

[6] Pang Y, Judd N, O'Brien J, Ben-Avie M, "Predicting students' graduation outcomes through support vector machines" In *IEEE Frontiers in Education Conference (FIE)*, 2017, pp. 1-8.

[7] N. Suhaimi, N.M., Abdul-Rahman, S., Mutalib, S., Hamid, N.A. and Hamid, A., "Review on predicting students' graduation time using machine learning algorithms" *International journal of modern education and computer science*, 2019, pp.1-13.

[8] L. Huang, L., Roche, L.R., Kennedy, E. and Brocato, M.B., "Using an Integrated Persistence Model to Predict College Graduation" *International Journal of Higher Education*, 2017, pp.40-56.

[9] M. Zulfa, M.I., Fadli, A. and Ramadhani, Y., "Classification model for graduation on time study using data mining techniques with SVM algorithm". In *AIP Conference Proceedings*, 2019

[10] Peling, I.B.A., Arnawan, I.N., Arthawan, I.P.A. and Janardana, I.G.N., "Implementation of Data Mining To Predict Period of Students Study Using Naive Bayes Algorithm" *Int. J. Eng. Emerg. Technol*, 2017, p.53.

[11] Polyzou, A. and Karypis, G., “Grade prediction with course and student specific models” In *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference*, 2016, pp. 89-101.

[12] Sweeney, M., Lester, J. and Rangwala, H., October, “Next-term student grade prediction” In *2015 IEEE International Conference on Big Data*, 2015, pp. 970-975.

[13] Campagni, R., Merlini, D., Sprugnoli, R. and Verri, M.C., “Data mining models for student careers” *Expert Systems with Applications*, 2015, pp.5508-5521.

[14] Wati, M., Indrawan, W., Widians, J.A. and Puspitasari, N., “Data mining for predicting students' learning result” In *2017 4th International Conference on Computer Applications and Information Processing Technology*, 2017, pp. 1-4.

[15] Z Kovacic, Z., “Early prediction of student success: Mining students' enrolment data” In *Informing Science + Information Technology Education Joint Conference, Cassino, Italy*, 2010.

[16] Ogor, E.N., “Student academic performance monitoring and evaluation using data mining techniques” In *Electronics, robotics and automotive mechanics conference*, 2007, pp. 354-359.

[17] Shruthi, P. and Chaitra, B.P., (2016, Mar), “Student performance prediction in education sector using data mining” *International Journal of Advanced Research in Computer Science and Software Engineering*, [online], 6(3), pp. 212-218. Available: http://www.ijarcsse.com

[18] Anuradha, C. and Velmurugan, T., “A comparative analysis on the evaluation of classification algorithms in the prediction of students performance” *Indian Journal of Science and Technology*, 2015, pp.1-12.

[19] Nghe, N.T., Janecek, P. and Haddawy, P., “A comparative analysis of techniques for predicting academic performance”. In *2007 37th annual frontiers in education conference-global engineering: knowledge without borders, opportunities without passports*, 2007, pp. T2G-7.

[20] Shahiri, A.M., Husain, W. and Rashid, N.A., “The Third Information Systems International Conference” *Procedia Computer Science*, 2015, pp.414-422.

[21] Asif, R., Merceron, A., Ali, S.A. and Haider, N.G., “Analyzing undergraduate students' performance using educational data mining” *Computers & education*, 2017, pp.177-194.

[22] Peña-Ayala, A., “Educational data mining: A survey and a data mining-based analysis of recent works” *Expert systems with applications*, 2014, pp.1432-1462.

[23] Mohamad, S.K. and Tasir, Z., “Educational data mining: A review” *Procedia-Social and Behavioral Sciences*, 2013, pp.320-324.

[24] Baker, R.S. and Yacef, K., “The state of educational data mining in 2009: A review and future visions” *Journal of educational data mining*, 2009, pp.3-17.

[25] Romero, C. and Ventura, S., “Educational data mining: a review of the state of the art” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (applications and reviews)*, 2010, pp.601-618.

[26] Jayaprakash, S., Krishnan, S. and Jaiganesh, V., “Predicting students academic performance using an improved random forest classifier” In *2020 international conference on emerging smart computing and informatics (ESCI)*, 2020, pp. 238-243.