Student Performance Prediction

A MINI PROJECT REPORT

Submitted by

Praneeth Vanaparthi [RA2111003010678]
Shaik Sohel Pasha [RA2111003010669]
Aalap Sangvikar [RA2111003010690]

Under the guidance of

Dr. G. Ramya

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE & ENGINEERING
of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"Student Performance Prediction"** is the bona fide work of **Praneeth Vanaparthi (RA2111003010678), Shaik Sohel Pasha (RA2111003010669), Aalap Sangvikar (RA2111003010690).** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. G. Ramya
Assistant Professor
Department of Computing Technologies

ABSTRACT

Student performance prediction is a field of research that aims to develop accurate models to predict the academic performance of students. With the growing emphasis on data-driven decision-making in education, predictive analytics has gained significant attention as a tool to improve student outcomes. This abstract provides an overview of the current state of student performance prediction research, highlighting key approaches, methods, and challenges. Various approaches have been employed to predict student performance, including statistical methods, machine learning algorithms, and data mining techniques. These approaches utilize a wide range of features, such as demographic information, past academic performance, attendance, and social factors, to develop predictive models. Machine learning algorithms, in particular, have shown promise in accurately predicting student performance by leveraging large amounts of data and identifying complex patterns. Moreover, the predictive models developed for student performance prediction have a wide range of applications, including identifying at-risk students who may need additional support, optimizing curriculum design, and improving educational policies. These models can also assist educators in early intervention efforts to prevent academic failure and dropout. Furthermore, student performance prediction has been used in higher education institutions for admissions decisions, course recommendations, and personalized learning. Despite the progress made in student performance prediction, there are challenges that need to be addressed. One key challenge is the ethical use of student data, including issues related to privacy, bias, and fairness. Additionally, the interpretability and explainability of predictive models are critical for gaining the trust of stakeholders, including students, parents, and educators. Another challenge is the need for robust and reliable data, as the accuracy and reliability of predictions depend on the quality of input data. In conclusion, student performance prediction is a rapidly growing field with significant potential to improve educational outcomes. The use of machine learning algorithms and predictive analytics has shown promise in accurately predicting student performance, enabling early intervention efforts and personalized learning. However, ethical considerations, interpretability, and data quality remain important challenges that need to be addressed in future research to ensure responsible and effective use of predictive models in education.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Student performance prediction is an important area of research and application in the field of education. It involves using various techniques, such as data analysis, machine learning, and statistical modeling, to forecast or estimate the academic performance of students. By understanding and predicting student performance, educators and administrators can make informed decisions about curriculum development, instructional strategies, and intervention programs, ultimately leading to improved educational outcomes.

1.1 Importance of student performance prediction

Predicting student performance is crucial for identifying students who may be at risk of falling behind academically. By accurately forecasting student performance, educators can proactively intervene and provide additional support to students who may be struggling. This can include personalized tutoring, targeted interventions, or adjustments to instructional strategies, all of which can help students improve their academic performance and achieve better outcomes. Student performance prediction can also aid in identifying students who may excel academically, allowing educators to provide challenging and enriching opportunities to foster their talents and potential. Additionally, student performance prediction can inform educational policy and decision-making at the institutional, district, or national level, helping administrators allocate resources effectively and design evidence-based interventions to improve overall student success.

1.2 Techniques for Student Performance Prediction

There are various techniques that can be used for student performance prediction. One commonly used approach is data analysis, where historical data on student performance, such as grades, attendance, and demographic information, are analyzed to identify patterns and trends. This data can be used to develop predictive models using statistical techniques, such as regression analysis, decision trees, or machine learning algorithms, to forecast future performance. Machine learning algorithms, such as support vector machines, random forests, or neural networks, can be particularly effective in analyzing large and complex datasets to make accurate predictions. Another approach is using predictive analytics, where data on student behavior, engagement, and motivation are analyzed to predict their academic performance. These techniques can be used in combination with other data sources, such as student feedback, surveys, or assessments, to provide a holistic view of student performance and enable early detection of students who may be at risk of falling behind or excelling academically.

CHAPTER II
LITERATURE SURVEY

INTRODUTION

Predicting student performance in educational settings has gained significant attention in recent years due to its potential for improving educational outcomes and guiding educational interventions. With the increasing availability of educational data and the advancement of data analytics techniques, researchers and educators have been exploring various approaches to predict student performance, identify students at risk of underperforming, and develop interventions to support their learning. In this literature survey, we will review some of the key studies related to student performance prediction in educational settings.

2.1 Predictive Modeling Techniques:

Several predictive modeling techniques have been applied to student performance prediction, including machine learning algorithms, statistical techniques, and data mining approaches.

2.1.1 Decision Trees:

Decision trees are tree-like structures that represent decisions or decisions based on certain conditions. Decision tree-based algorithms, such as C4.5 and Random Forest, have been applied to predict student performance. These algorithms can handle both categorical and continuous variables and are capable of handling complex interactions between features.

2.1.2 Logistic Regression:

Logistic regression is a statistical technique used to predict the probability of an event occurring. In the context of student performance prediction, logistic regression can be used to predict the likelihood of a student passing or failing a course based on various features such as prior academic performance, demographic information, and engagement in class activities.

2.1.3 Support Vector Machines (SVM):

SVM is a popular machine learning algorithm used for classification tasks. SVM can be used for student performance prediction by training a model with labeled data that includes features such as attendance, engagement, and prior academic performance. SVM aims to find a hyperplane that best separates the data points into different classes, such as passing or failing a course.

2.1.4 Gradient Boosting:

Gradient boosting is an ensemble method that combines multiple weak models to create a more accurate predictive model. It can be used for student performance prediction by sequentially adding weak models that are trained on the errors of the previous models. Gradient boosting algorithms, such as XGBoost and AdaBoost, have been applied to predict student performance based on various features.

2.1.5 Neural Networks:

Deep learning techniques, such as neural networks, have also been used for student performance prediction. Neural networks are capable of capturing complex patterns in data and can be trained on large amounts of data to make accurate predictions. They can be used to predict student performance based on features such as attendance, engagement, and academic records.

2.1.6 Time Series Analysis:

Time series analysis is a technique used to analyze data that changes over time. In the context of student performance prediction, time series analysis can be used to analyze patterns and trends in student performance data over time, such as academic performance trends across different semesters or academic years. Time series models, such as ARIMA and LSTM, can be used to forecast future student performance based on historical data

2.2 Feature Selection and Feature Engineering:

Feature selection and feature engineering are critical steps in building accurate student performance prediction models. Researchers have explored various approaches to identify the most relevant features that impact student performance.

2.2.1 Univariate Feature Selection:

This method involves selecting features based on their individual statistical significance or performance in isolation. Common techniques include statistical tests such as chi-square test, t-test, or ANOVA to select features that have a significant impact on student performance.

2.2.2 Recursive Feature Elimination (RFE):

RFE is an iterative technique that involves training a model multiple times and recursively eliminating less important features based on their importance or contribution to the model's performance. This technique helps in selecting the most important features based on their relevance to the prediction task.

2.2.3 Feature Importance from Tree-based Models:

Some predictive models, such as decision trees and random forests, provide feature importance scores that reflect the contribution of each feature in the model's performance. These scores can be used to rank and select the most important features for student performance prediction.

Feature Engineering:

Feature engineering involves creating new features or transforming existing features to better represent the underlying patterns and relationships in the data. This can lead to improved predictive performance and more accurate student performance predictions. Some common techniques for feature engineering in the context of student performance prediction .

2.2.4 Creating Aggregate Features:

This involves aggregating data at a higher level, such as calculating average or sum of features over a certain period, to capture trends or patterns that may not be apparent at the individual level. For example, aggregating weekly attendance data into monthly attendance percentages can provide a more meaningful feature for predicting student performance.

2.2.5 Creating Interaction Features:

Interaction features are created by combining two or more existing features to capture potential interactions or synergies between them. For example, combining features such as study hours and prior academic performance to create an interaction feature that represents the interaction between study habits and prior performance.

2.2.6 Normalizing or Scaling Features:

Scaling or normalizing features can help in bringing features to a similar scale, which is important for models that are sensitive to the scale of the features, such as logistic regression or support vector machines. Common techniques include min-max scaling, z-score normalization, or log transformation.

2.2.7 Handling Missing Data:

Missing data can impact the accuracy of predictive models. Different techniques such as imputation, deletion, or using advanced methods like multiple imputation or k-nearest neighbor imputation can be employed to handle missing data effectively and ensure that the feature set used for prediction is complete.

2.3 Evaluation metrics

Evaluation metrics are crucial for assessing the performance and accuracy of predictive models for student performance prediction projects. Here are some commonly used evaluation metrics

2.3.1 Classification Metrics:

Accuracy: Measures the proportion of correctly predicted instances to the total number of instances. It is commonly used for balanced datasets.

Precision: Measures the proportion of true positive predictions to the total predicted positive instances, indicating the model's ability to correctly predict positive instances.

Recall: Measures the proportion of true positive predictions to the total actual positive instances, indicating the model's ability to identify all the positive instances.

F1-Score: The harmonic mean of precision and recall, providing a balanced measure of both precision and recall.

Area Under the Receiver Operating Characteristic (ROC) Curve and Precision-Recall (PR) Curve: ROC curve displays the true positive rate against the false positive rate, while PR curve displays precision against recall. AUC-ROC and AUC-PR values close to 1 indicate excellent predictive performance.

2.3.2 Regression Metrics:

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE): MSE measures the average of the squared differences between predicted and actual values, while RMSE is the square root of MSE, providing a measure of the average absolute prediction error.

R-squared (R2) Score: Measures the proportion of the total variance in the target variable explained by the predictive model. R2 score ranges from 0 to 1, with higher values indicating a better fit of the model to the data.

2.3.3 Cross-validation:

Cross-validation techniques, such as k-fold cross-validation or stratified k-fold cross-validation, can provide more reliable and robust evaluation results by training and evaluating the model on different subsets of data.

2.3.4 Cohort Analysis:

Comparing the performance of different student cohorts or groups, such as gender or ethnic groups, can provide insights into the model's performance for different subpopulations.

2.3.5 Predictive Power:

Assessing the predictive power of the model by measuring how well it performs in predicting future student performance or outcomes.

2.4 factors effecting student performance

There are several factors that can affect student performance in the context of a student performance prediction project. These factors can be categorized into different categories, including

2.4.1 Demographic Factors:

Demographic factors such as gender, age, ethnicity, socioeconomic status, and cultural background can impact student performance. For example, research has shown that gender-based differences may exist in academic performance, with males and females sometimes displaying different patterns of achievement in different subjects or at different educational levels. Similarly, socioeconomic status and cultural background can influence students' access to educational resources and opportunities, which can in turn impact their performance.

2.4.2 Academic Factors:

Academic factors such as prior academic achievement, attendance, engagement in classroom activities, study habits, and motivation can significantly impact student performance. Students with a strong academic foundation, good attendance, active engagement in classroom activities, effective study habits, and high motivation are more likely to perform well academically.

2.4.3 Psychological Factors:

Psychological factors, including cognitive abilities, intelligence, learning style, self-efficacy, self-regulation, and emotional well-being, can also impact student performance. For example, students with higher cognitive abilities, effective self-regulation skills, and positive emotional well-being are more likely to perform well academically.

2.4.4 Environmental Factors:

Environmental factors, such as the quality of the school, classroom environment, availability of educational resources, and support from teachers, peers, and parents, can also impact student performance. A conducive learning environment with adequate resources, supportive teachers, and engaged peers and parents can positively impact student performance.

2.4.5 Personal Factors:

Personal factors, such as students' interests, aspirations, and goals, can also impact their performance. When students are motivated by their personal interests and have clear goals and aspirations, they are more likely to perform well academically.

2.4.6 Technology Factors:

With the increasing integration of technology in education, technology-related factors, such as access to computers, internet, e-learning platforms, and digital literacy, can also impact student performance. Students who have access to technology resources and possess digital literacy skills may have an advantage in their academic performance.

2.5 Applications and impacts of student performance prediction

Applications and impacts of student performance prediction in the context of a student performance prediction project can be wide-ranging and impactful. Here are some potential applications and impacts of using predictive models for student performance prediction.

2.5.1 Early Warning Systems:

Predictive models can be used to identify students who may be at risk of academic underperformance or dropout. By analyzing various factors, such as past academic performance, attendance, engagement, and behavior, predictive models can identify students who may require additional support or interventions. Early warning systems based on predictive models can help educators and administrators intervene early and implement targeted interventions to improve student outcomes.

2.5.2 Personalized Learning:

Predictive models can help in tailoring education to individual students' needs. By analyzing students' learning styles, interests, and performance patterns, predictive models can recommend personalized learning paths, content, and resources. This can help students learn at their own pace, address their learning gaps, and optimize their academic performance.

2.5.3 Resource Allocation:

Predictive models can aid in optimizing the allocation of educational resources. By analyzing student performance data, predictive models can identify areas where additional resources, such as teaching staff, tutoring programs, or learning materials, may be needed. This can help educational institutions efficiently allocate their resources and interventions where they are most needed, thus maximizing the impact of available resources.

2.5.4 Curriculum Design:

Predictive models can inform curriculum design and instructional strategies. By analyzing student performance data, predictive models can identify areas of the curriculum that may be challenging for students, areas where students excel, and areas that require improvement. This information can guide curriculum designers and educators in developing instructional strategies, content, and assessments that better align with students' needs, leading to improved student performance.

2.5.5 Policy Making:

Predictive models can inform educational policy making. By analyzing large-scale student performance data, predictive models can provide insights into the effectiveness of educational policies and interventions. This can help policymakers make data-driven decisions and implement evidence-based policies to improve overall student performance and educational outcomes.

2.5.6 Parental Engagement:

Predictive models can also involve parents in the educational process. By analyzing student performance data, predictive models can generate reports and alerts that can be shared with parents, informing them about their child's performance, strengths, weaknesses, and areas that require additional support. This can promote parental engagement, enable timely interventions, and support parents in actively participating in their child's academic journey.

CHAPTER III
SYSTEM ARCHITECTURE AND DESIGN

A system architecture for a "student performance prediction" project would typically involve multiple components working together to collect, process, and analyze data to generate predictions. Here's a high-level overview of a typical system architecture

3.1 Data Collection:

This component involves collecting relevant data related to students' performance, such as past academic records, attendance, engagement, behavior, and other relevant factors. Data can be collected from various sources, such as student information systems, learning management systems, attendance records, surveys, and other relevant databases.

3.1.1 Identify Relevant Data:

The first step in data collection is to identify the types of data that are relevant for predicting student performance. This may include academic records, attendance, engagement, behavior, demographic information, socioeconomic status, learning style, study habits, and other factors that may impact student performance. The data should be aligned with the specific objectives of the project and should be sufficient to capture the relevant factors that can influence student performance.

3.1.2 Data Cleaning and Pre-processing:

Collected data may require cleaning and pre-processing to handle missing values, outliers, inconsistent data, and other data quality issues. Data should be cleaned, transformed, and prepared for analysis to ensure its accuracy and reliability. Data pre-processing may also involve data normalization, feature extraction, and feature engineering to derive meaningful insights from the data.

3.1.3 Data Integration:

In some cases, data from multiple sources may need to be integrated to create a comprehensive dataset for analysis. Data integration may involve merging datasets, resolving data inconsistencies, and handling data from different sources with varying formats and structures.

3.1.4 Data Storage:

Data should be stored securely and appropriately to ensure data integrity, confidentiality, and availability. Proper data storage practices, such as data backup, encryption, and access controls, should be implemented to protect the data from unauthorized access and data breaches.

3.1.5 Data Documentation:

Proper documentation of the collected data, including data sources, data quality, data transformation, and data labeling, is important for transparency, reproducibility, and future reference. Documentation should include metadata,

data dictionaries, and any other relevant information that can help in understanding and utilizing the data in the future.

3.2 Data Pre-processing:

Data pre-processing is an important step in the development of a "student performance prediction" system. It involves cleaning, transforming, and preparing the collected data to ensure its accuracy, reliability, and suitability for analysis. Here are some common data pre-processing techniques that can be applied in a student performance prediction.

3.2.1 Handling Missing Values:

Missing values are a common issue in real-world datasets, and they can affect the accuracy and reliability of predictive models. Missing values can be handled by various methods, such as imputation, deletion, or estimation. Imputation involves filling in the missing values with estimated values based on statistical methods or machine learning algorithms. Deletion involves removing the rows or columns with missing values, but it should be done carefully to avoid losing important information. Estimation involves using techniques like regression or machine learning to predict missing values based on other available data

3.2.2 Handling Outliers:

Outliers are data points that deviate significantly from the rest of the data and can adversely impact the performance of predictive models. Outliers can be handled by various methods, such as filtering, transformation, or imputation. Filtering involves removing or replacing outlier values based on predefined thresholds or statistical methods. Transformation involves applying mathematical or statistical transformations to the data to reduce the impact of outliers. Imputation involves replacing outlier values with estimated values based on statistical methods or machine learning algorithms.

3.2.3 Data Normalization:

Data normalization is the process of scaling the data to a common range to avoid bias towards certain features due to their magnitude differences. Common normalization techniques include min-max scaling, z-score normalization, and log transformation.

3.3 Machine Learning Models:

There are several machine learning models that can be used for "student performance prediction" based on the type of data available, the specific problem being addressed, and the desired outcome. Here are some commonly used machine learning models for student performance prediction.

3.3.1 Linear Regression:

Linear regression is a simple and interpretable model that can be used for predicting numerical values, such as student performance scores, based on one or more input features. It assumes a linear relationship between the input features and the target variable, and estimates the coefficients of the linear equation using least squares optimization.

3.3.2 Logistic Regression:

Logistic regression is a binary classification model that can be used for predicting binary outcomes, such as whether a student will pass or fail a course, based on input features. It estimates the probabilities of class membership using a logistic function and can be extended to handle multi-class classification using techniques such as one-vs-rest or softmax.

3.3.3 Decision Trees:

Decision trees are versatile models that can be used for both classification and regression tasks. They partition the data based on the input features and make decisions at each node based on the feature values to predict the target variable. Decision trees can be visualized and are interpretable, but they are prone to overfitting.

3.3.4 Artificial Neural Networks (ANN):

ANN is a versatile and complex model that can be used for a wide range of tasks, including classification, regression, and time-series prediction. ANN consists of multiple layers of interconnected nodes (neurons) and uses activation functions to introduce non-linearity into the model. Deep learning architectures, such as Convolutional Neural Networks (CNN) for image data or Recurrent Neural Networks (RNN) for sequential data, can also be used for more complex student performance prediction tasks.

3.4 Model Evaluation:

Model evaluation is an important step in the machine learning pipeline for "student performance prediction" as it helps to assess the performance and generalization ability of the models. Here are some commonly used evaluation metrics

3.4.1 Mean Squared Error (MSE):

MSE is a regression metric that measures the average squared difference between the predicted and actual values of the target variable. Lower values of MSE indicate better performance.

3.4.2 Root Mean Squared Error (RMSE):

RMSE is a variant of MSE that takes the square root of the MSE to obtain a metric in the same units as the target variable. RMSE is a commonly used metric for regression tasks, including "student performance prediction".

3.4.3 R-squared (R2):

R2 is a regression metric that measures the proportion of the variance in the target variable that is explained by the model. R2 ranges from 0 to 1, with higher values indicating better performance.

3.4.4 F1 Score:

F1 Score is a harmonic mean of precision and recall that balances between precision and recall. F1 Score is a commonly used metric for imbalanced datasets where both precision and recall are important.

3.4.5 Receiver Operating Characteristic (ROC) Curve:

ROC curve is a graphical representation of the trade-off between the true positive rate (TPR) and false positive rate (FPR) for different thresholds in binary classification.

3.5 Model Deployment:

Model deployment is an important step in the machine learning pipeline for "student performance prediction" as it allows the model to be used for real-world applications

3.5.1 Web Application:

One of the most common ways to deploy a machine learning model is through a web application. The model can be integrated into a web application using web frameworks such as Flask, Django, or Node.js. Users can interact with the web application by providing input data, and the model will generate predictions based on the input data.

3.5.2 REST API:

Another approach to deploy a machine learning model is through a REST API. The model can be deployed on a cloud-based platform such as AWS or Azure, and the REST API can be used to send requests to the model and receive responses. This approach is commonly used when the model needs to be accessed by multiple applications or services.

3.5.3 Mobile Application:

Machine learning models can also be deployed in mobile applications. The model can be integrated into the mobile application using mobile development frameworks such as React Native or Flutter. Users can interact with the mobile application by providing input data, and the model will generate predictions based on the input data.

3.6 User Interface:

The user interface (UI) is an important component of any machine learning application, including "student performance prediction", as it allows users to interact with the model and understand the predictions.

3.6.1 Input Data:

The UI should provide a clear and intuitive way for users to input data for "student performance prediction". This could include input fields for demographic data, academic history, and other relevant features.

3.6.2 Model Output:

The UI should display the predictions generated by the model in a clear and understandable format. This could include a summary of the predicted performance, as well as visualizations and charts to help users understand the factors that contribute to the prediction.

3.6.3 User Feedback:

The UI should provide a way for users to provide feedback on the predictions generated by the model. This could include the ability to flag incorrect predictions, provide additional information, or request additional analysis.

3.6.4 Accessibility:

The UI should be designed with accessibility in mind, ensuring that it can be used by people with disabilities such as visual or motor impairments. This could include features such as voice commands, screen readers, and high-contrast color schemes.

3.6.5 User Experience (UX):

The UI should be designed with the user experience in mind, ensuring that it is easy to use and understand. This could include features such as clear labeling, intuitive navigation, and responsive design for different screen sizes and devices.

3.7 Monitoring and Maintenance

Monitoring and maintenance are critical aspects of any machine learning system, including "student performance prediction". Here are some considerations for monitoring and maintaining the system:

3.7.1 Data Quality:

The accuracy and reliability of the predictions generated by the model are dependent on the quality of the input data. Therefore, it is important to regularly monitor the quality of the input data and ensure that any issues are addressed promptly.

3.7.2 Model Performance:

The performance of the machine learning model should be regularly monitored to ensure that it continues to generate accurate and reliable predictions. This could include monitoring metrics such as accuracy, precision, recall, and F1 score.

3.7.3 Security:

The machine learning system should be designed with security in mind to prevent unauthorized access to sensitive data or model parameters. This could include implementing access controls, encrypting data, and using secure communication protocols.

3.7.4 Scalability:

As the number of users and amount of data increase, it may be necessary to scale the machine learning system to ensure that it continues to perform efficiently. This could include deploying the system on a cloud-based platform that can automatically scale resources as needed.

3.7.5 Documentation:

It is important to maintain up-to-date documentation for the machine learning system, including information on the data sources, model architecture, and deployment process. This can help ensure that the system is maintainable and can be easily updated or modified as needed.

3.8 Data Security and Privacy:

Data security and privacy are essential considerations when it comes to "student performance prediction.

3.8.1 Data Encryption:

All student data should be encrypted to prevent unauthorized access. Encryption involves converting data into a code that can only be accessed by authorized parties with the decryption key.

3.8.2 Access Control:

Access to student data should be restricted to authorized personnel only. This can be achieved by implementing role-based access control, which assigns specific roles and permissions to different individuals based on their job responsibilities.

3.8.3 Secure Storage:

All student data should be stored in secure databases and servers that are protected by firewalls and other security measures.

3.8.4 Regular Audits:

Regular audits should be conducted to ensure that data security and privacy policies are being followed.

3.8.5 Privacy Policies:

A clear and concise privacy policy should be provided to all stakeholders. The policy should clearly state how student data will be collected, processed, and used.

3.8.6 Data Breach Response Plan:

Develop a data breach response plan that outlines the steps to be taken in the event of a security breach. The plan should include procedures for notifying affected parties, investigating the breach, and preventing future incidents.

CHAPTER IV
METHODOLOGY

Student performance prediction is a complex task that requires a well-defined methodology to ensure accurate results. The first step in this methodology is to define the problem and clearly state the objectives of the project. Once the problem is defined, data collection and preprocessing can begin. Relevant data can be collected from various sources and preprocessed by cleaning it, removing any duplicates, and handling any missing values. The next step is to identify the relevant features or variables that can potentially impact student performance. This feature engineering involves transforming and selecting the most relevant features for the prediction model. Exploratory data analysis can then be conducted to gain insights into the data, identify patterns and trends, and visualize the data using graphs and charts. Once the data is prepared, an appropriate machine learning model can be selected and trained on the data. The model's performance can then be evaluated and fine-tuned until its accuracy is satisfactory. Finally, the model can be deployed in a production environment for use in predicting student performance. It is important to monitor the model's performance over time and retrain or fine-tune the model as necessary to ensure its continued accuracy and relevance.

To ensure the accuracy of the student performance prediction model, it is crucial to carefully select and preprocess the data. The data should be relevant and up-to-date, and any errors or inconsistencies should be corrected. Feature selection is also important as it helps to identify the most influential factors that contribute to student performance. Exploratory data analysis can then be used to visualize and understand the data better, which can help to refine the model and improve its accuracy.

Selecting an appropriate machine learning model is also critical to the success of the project. Different models may perform better depending on the problem statement, available data, and objectives of the project. It is important to evaluate the performance of the model using appropriate metrics such as accuracy, precision, recall, and F1 score. The model can then be fine-tuned by adjusting its hyperparameters and validated on a separate validation set to ensure its performance is satisfactory.

Once the model is trained and validated, it can be deployed in a production environment and used to predict student performance. However, it is important to monitor the model's performance over time and fine-tune it as necessary to ensure its continued accuracy and relevance. This could involve retraining the model on new data or adjusting its hyperparameters to better fit the data.

4.1 Define the problem:

Clearly define the problem statement and objectives of the project. For example, the objective could be to predict student grades based on various academic and non-academic factors.

4.2 Data collection and preprocessing:

Collect relevant data from various sources such as academic records, attendance records, and student surveys. Preprocess the data by cleaning it, removing any duplicates, and handling any missing values.

4.3 Feature engineering:

Identify the relevant features or variables that can potentially impact student performance. These could include factors such as socio-economic status, previous academic performance, and attendance records. Feature engineering involves transforming and selecting the most relevant features for the prediction model.

4.4 Exploratory data analysis (EDA):

Conduct EDA to gain insights into the data, identify patterns and trends, and visualize the data using graphs and charts.

4.5 Machine learning model selection:

Select an appropriate machine learning model based on the problem statement, available data, and the objective of the project. Examples of machine learning models that can be used for student performance prediction include linear regression, decision trees, and neural networks

4.6 Model training and testing:

Split the data into training and testing sets, train the selected machine learning model on the training set, and test its performance on the testing set. Evaluate the performance of the model using appropriate metrics such as accuracy, precision, recall, and F1 score.

4.7 Model tuning and validation:

Fine-tune the model by adjusting its hyperparameters, and validate the model's performance on a separate validation set. This step involves iteratively adjusting the model until its performance is satisfactory.

4.8 Model deployment:

Once the model is trained and validated, it can be deployed in a production environment for use in predicting student performance.

4.9 Model monitoring and maintenance:

Monitor the performance of the model over time, and retrain or fine-tune the model as necessary to ensure its continued accuracy and relevance.

In summary, a well-defined methodology is crucial for building an accurate student performance prediction model. This methodology involves defining the problem, collecting and preprocessing the data, feature selection, exploratory data analysis, machine learning model selection, training and testing the model, fine-tuning and validation, model deployment, and model monitoring and maintenance. By following this methodology, we can build an accurate model that can help to predict student performance and identify the factors that contribute to academic success.

CHAPTER V
CODING AND TESTING

```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import time as t
import sklearn.utils as u
import sklearn.preprocessing as pp
import sklearn.tree as tr
import sklearn.ensemble as es
import sklearn.metrics as m
import sklearn.linear_model as lm
import sklearn.neural_network as nn
import numpy as np
#import random as rnd
import warnings as w
w.filterwarnings('ignore')
ch = 0
while(ch != 10):
   print("1.Marks Class Count Graph\t2.Marks Class Semester-wise
Graph\n3.Marks Class Gender-wise Graph\t4.Marks Class Nationality-
wise Graph\n5.Marks Class Grade-wise Graph\t6.Marks Class Section-
wise Graph\n7.Marks Class Topic-wise Graph\t8.Marks Class Stage-wise
Graph\n9.Marks Class Absent Days-wise\t10.No Graph\n")
   ch = int(input("Enter Choice: "))
   if (ch == 1):
      print("Loading Graph... \n")
      t.sleep(1)
      print("\tMarks Class Count Graph")
      axes = sb.countplot(x='Class', data=data, order=['L', 'M', 'H'])
      plt.show()
   elif (ch == 2):
      print("Loading Graph... \n")
      t.sleep(1)
      print("\tMarks Class Semester-wise Graph")
      fig, axesarr = plt.subplots(1, figsize=(10, 6))
      sb.countplot(x='Semester', hue='Class', data=data, hue_order=['L',
'M', 'H'], axes=axesarr)
```

```
        plt.show()
    elif (ch == 3):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Gender-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='gender', hue='Class', data=data, order=['M', 'F'],
hue_order=['L', 'M', 'H'], axes=axesarr)
        plt.show()
    elif (ch == 4):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Nationality-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='NationalITy', hue='Class', data=data, hue_order=['L',
'M', 'H'], axes=axesarr)
        plt.show()
    elif (ch == 5):
        print("Loading Graph: \n")
        t.sleep(1)
        print("\tMarks Class Grade-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='GradeID', hue='Class', data=data, order=['G-02', 'G-
04', 'G-05', 'G-06', 'G-07', 'G-08', 'G-09', 'G-10', 'G-11', 'G-12'], hue_order
= ['L', 'M', 'H'], axes=axesarr)
        plt.show()
    elif (ch ==6):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Section-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='SectionID', hue='Class', data=data, hue_order = ['L',
'M', 'H'], axes=axesarr)
        plt.show()
    elif (ch == 7):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Topic-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='Topic', hue='Class', data=data, hue_order = ['L', 'M',
'H'], axes=axesarr)
        plt.show()
    elif (ch == 8):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Stage-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='StageID', hue='Class', data=data, hue_order = ['L',
'M', 'H'], axes=axesarr)
        plt.show()
    elif (ch == 9):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Absent Days-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='StudentAbsenceDays', hue='Class', data=data,
hue_order = ['L', 'M', 'H'], axes=axesarr)
        plt.show()
if(ch == 10):
    print("Exiting..\n")
    t.sleep(1)
#cor = data.corr()
#print(cor)
data = data.drop("gender", axis=1)
data = data.drop("StageID", axis=1)
data = data.drop("GradeID", axis=1)
data = data.drop("NationalITy", axis=1)
data = data.drop("PlaceofBirth", axis=1)
data = data.drop("SectionID", axis=1)
data = data.drop("Topic", axis=1)
data = data.drop("Semester", axis=1)
data = data.drop("Relation", axis=1)
data = data.drop("ParentschoolSatisfaction", axis=1)
data = data.drop("ParentAnsweringSurvey", axis=1)
#data = data.drop("VisITedResources", axis=1)
data = data.drop("AnnouncementsView", axis=1)
u.shuffle(data)
countD = 0
countP = 0
```

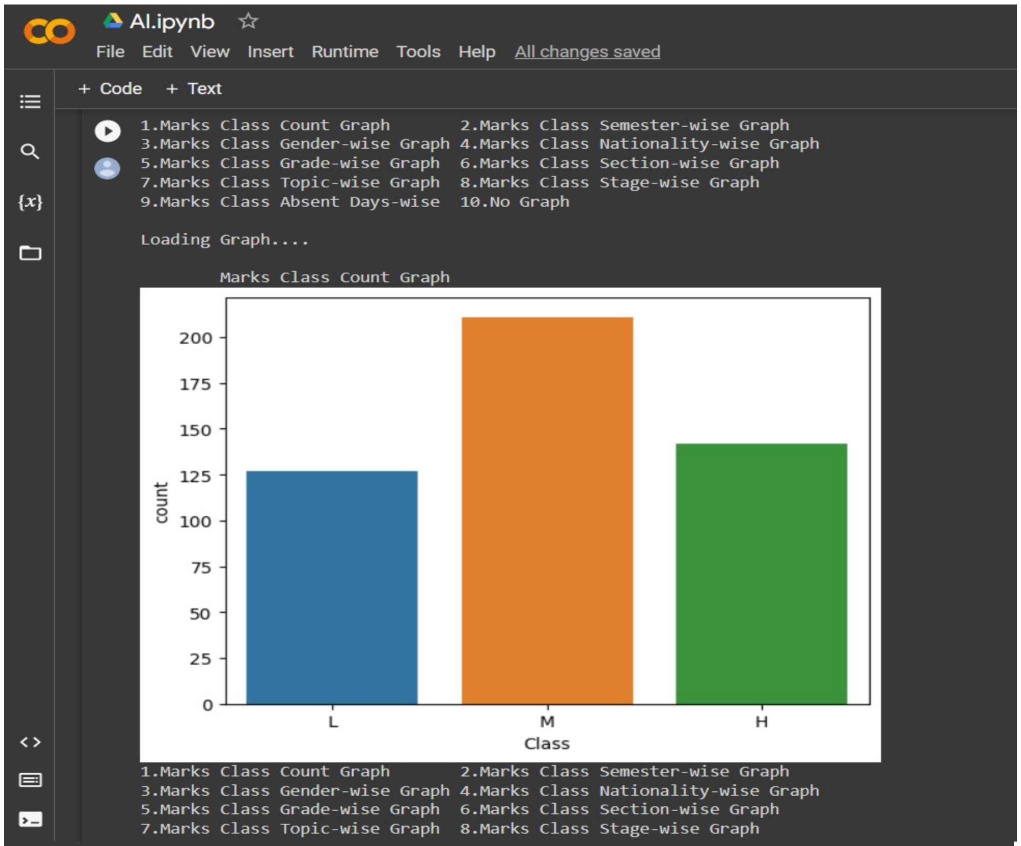
```
countL = 0
countR = 0
countN = 0
gradeID_dict = {"G-01" : 1,
                "G-02" : 2,
                "G-03" : 3,
                "G-04" : 4,
                "G-05" : 5,
                "G-06" : 6,
                "G-07" : 7,
                "G-08" : 8,
                "G-09" : 9,
                "G-10" : 10,
                "G-11" : 11,
                "G-12" : 12}
data = data.replace({"GradeID" : gradeID_dict})
#sig = []
for column in data.columns:
    if data[column].dtype == type(object):
        le = pp.LabelEncoder()
        data[column] = le.fit_transform(data[column])
ind = int(len(data) * 0.70)
feats = data.values[:, 0:4]
lbls = data.values[:,4]
feats_Train = feats[0:ind]
feats_Test = feats[(ind+1):len(feats)]
lbls_Train = lbls[0:ind]
lbls_Test = lbls[(ind+1):len(lbls)]
modelD = tr.DecisionTreeClassifier()
modelD.fit(feats_Train, lbls_Train)
lbls_predD = modelD.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predD):
    if(a==b):
        countD += 1
accD = (countD/len(lbls_Test))
print("\nAccuracy measures using Decision Tree:")
print(m.classification_report(lbls_Test, lbls_predD),"\n")
print("\nAccuracy using Decision Tree: ", str(round(accD, 3)))
t.sleep(1)
modelR = es.RandomForestClassifier()
```

```
modelR.fit(feats_Train, lbls_Train)
lbls_predR = modelR.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predR):
    if(a==b):
        countR += 1
print("\nAccuracy Measures for Random Forest Classifier: \n")
#print("\nConfusion Matrix: \n", m.confusion_matrix(lbls_Test,
lbls_predR))
print("\n", m.classification_report(lbls_Test,lbls_predR))
accR = countR/len(lbls_Test)
print("\nAccuracy using Random Forest: ", str(round(accR, 3)))
t.sleep(1)
modelP = lm.Perceptron()
modelP.fit(feats_Train, lbls_Train)
lbls_predP = modelP.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predP):
    if a == b:
        countP += 1
accP = countP/len(lbls_Test)
print("\nAccuracy measures using Linear Model Perceptron:")
print(m.classification_report(lbls_Test, lbls_predP),"\n")
print("\nAccuracy using Linear Model Perceptron: ", str(round(accP, 3)),
"\n")
t.sleep(1)
modelL = lm.LogisticRegression()
modelL.fit(feats_Train, lbls_Train)
lbls_predL = modelL.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predL):
    if a == b:
        countL += 1
accL = countL/len(lbls_Test)
print("\nAccuracy measures using Linear Model Logistic Regression:")
print(m.classification_report(lbls_Test, lbls_predL),"\n")
print("\nAccuracy using Linear Model Logistic Regression: ",
str(round(accP, 3)), "\n")
t.sleep(1)
modelN = nn.MLPClassifier(activation="logistic")
modelN.fit(feats_Train, lbls_Train)
lbls_predN = modelN.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predN):
    #sig.append(1/(1+ np.exp(-b)))
    if a==b:
        countN += 1
#print("\nAverage value of Sigmoid Function: ",
str(round(np.average(sig), 3)))
print("\nAccuracy measures using MLP Classifier:")
print(m.classification_report(lbls_Test, lbls_predN),"\n")
accN = countN/len(lbls_Test)
print("\nAccuracy using Neural Network MLP Classifier: ",
str(round(accN, 3)), "\n")
choice = input("Do you want to test specific input (y or n): ")
if(choice.lower()=="y"):
    gen = input("Enter Gender (M or F): ")
    if (gen.upper() == "M"):
      gen = 1
    elif (gen.upper() == "F"):
      gen = 0
    nat = input("Enter Nationality: ")
    pob = input("Place of Birth: ")
    sta = input("Enter Stage ID(Lower level, Middle school, High school):
")
    if (sta == "Lower level"):
        sta = 2
    elif(sta == "Middle school"):
        sta = 1
    elif (sta == "High school"):
        sta = 0
    gra = input("Grade ID as (G-<grade>): ")
    if(gra == "G-02"):
        gra = 2
    elif (gra == "G-04"):
        gra = 4
    elif (gra == "G-05"):
        gra = 5
    elif (gra == "G-06"):
        gra = 6
    elif (gra == "G-07"):
        gra = 7
    elif (gra == "G-08"):
        gra = 8
```

```
elif (gra == "G-09"):
    gra = 9
elif (gra == "G-10"):
    gra = 10
elif (gra == "G-11"):
    gra = 11
elif (gra == "G-12"):
    gra = 12
sec = input("Enter Section: ")
top = input("Enter Topic: ")
sem = input("Enter Semester (F or S): ")
if (sem.upper() == "F"):
   sem = 0
elif (sem.upper() == "S"):
   sem = 1
rel = input("Enter Relation (Father or Mum): ")
if (rel == "Father"):
   rel = 0
elif (rel == "Mum"):
   rel = 1
rai = int(input("Enter raised hands: "))
res = int(input("Enter Visited Resources: "))
ann = int(input("Enter announcements viewed: "))
dis = int(input("Enter no. of Discussions: "))
sur = input("Enter Parent Answered Survey (Y or N): ")
if (sur.upper() == "Y"):
   sur = 1
elif (sur.upper() == "N"):
   sur = 0
sat = input("Enter Parent School Satisfaction (Good or Bad): ")
if (sat == "Good"):
   sat = 1
elif (sat == "Bad"):
   sat = 0
absc = input("Enter No. of Abscenes(Under-7 or Above-7): ")
if (absc == "Under-7"):
   absc = 1
elif (absc == "Above-7"):
   absc = 0
arr = np.array([rai, res, dis, absc])
```

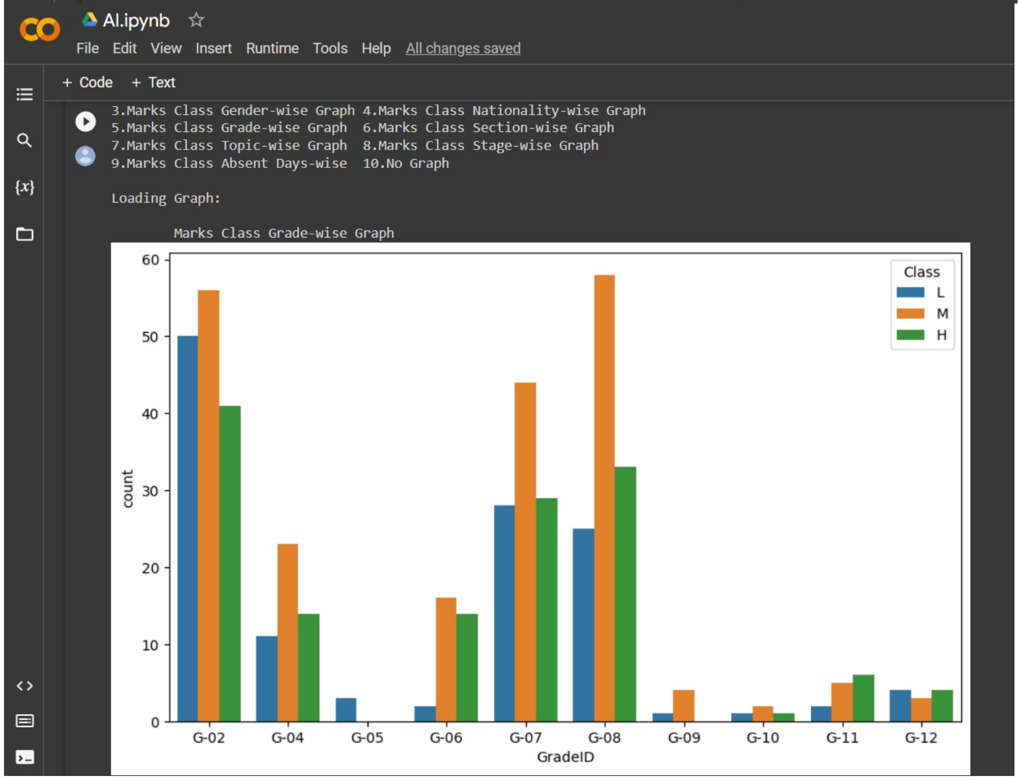
```
    #arr = np.array([gen, rnd.randint(0, 30), rnd.randint(0, 30), sta, gra,
rnd.randint(0, 30), rnd.randint(0, 30), sem, rel, rai, res, ann, dis, sur, sat,
absc])
    predD = modelD.predict(arr.reshape(1, -1))
    predR = modelR.predict(arr.reshape(1, -1))
    predP = modelP.predict(arr.reshape(1, -1))
    predL = modelL.predict(arr.reshape(1, -1))
    predN = modelN.predict(arr.reshape(1, -1))
    if (predD == 0):
        predD = "H"
    elif (predD == 1):
        predD = "M"
    elif (predD == 2):
        predD = "L"
    if (predR == 0):
        predR = "H"
    elif (predR == 1):
        predR = "M"
    elif (predR == 2):
        predR = "L"
    if (predP == 0):
        predP = "H"
    elif (predP == 1):
        predP = "M"
    elif (predP == 2):
        predP = "L"
    if (predL == 0):
        predL = "H"
    elif (predL == 1):
        predL = "M"
    elif (predL == 2):
        predL = "L"
    if (predN == 0):
        predN = "H"
    elif (predN == 1):
        predN = "M"
    elif (predN == 2):
        predN = "L"
    t.sleep(1)
    print("\nUsing Decision Tree Classifier: ", predD)
    t.sleep(1)
    print("Using Random Forest Classifier: ", predR)
    t.sleep(1)
    print("Using Linear Model Perceptron: ", predP)
    t.sleep(1)
    print("Using Linear Model Logisitic Regression: ", predL)
    t.sleep(1)
    print("Using Neural Network MLP Classifier: ", predN)
    print("\nExiting...")
    t.sleep(1)
else:
    print("Exiting..")
    t.sleep(1)
```

RESULTS

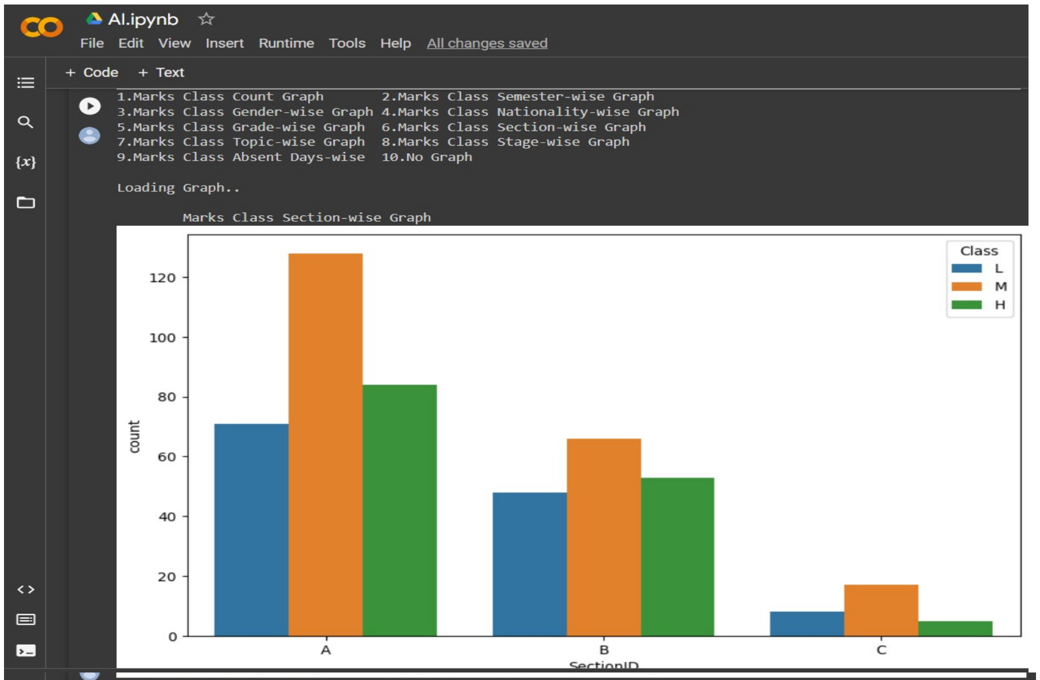


```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Loading Graph....

        Marks Class Count Graph
```

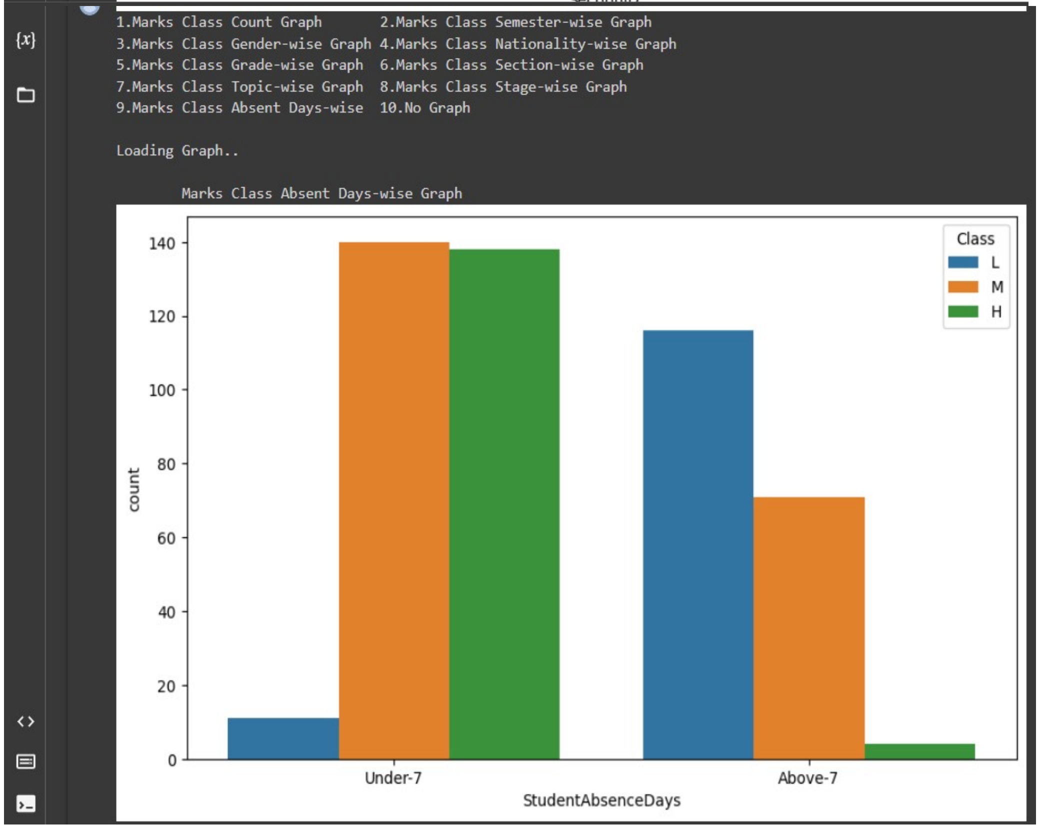
```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
```



```
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Loading Graph:

        Marks Class Grade-wise Graph
```

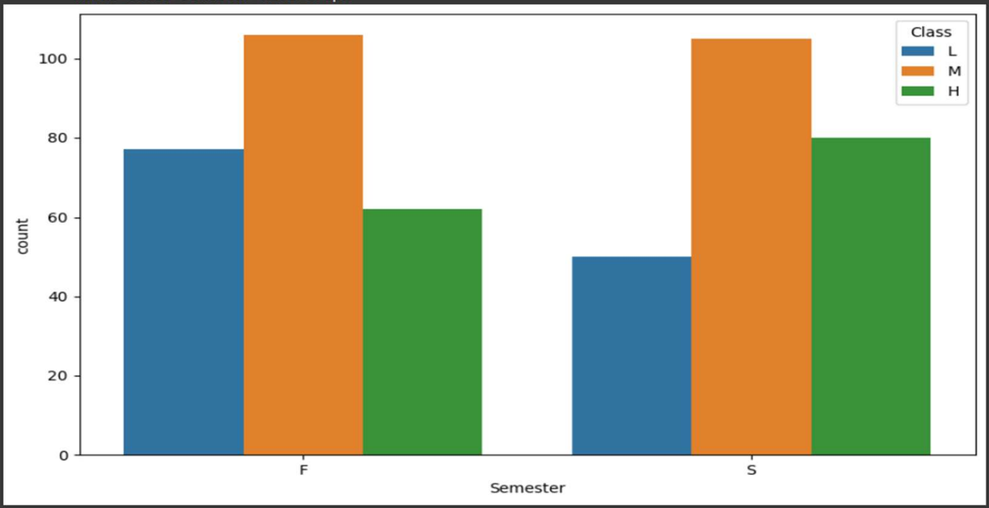


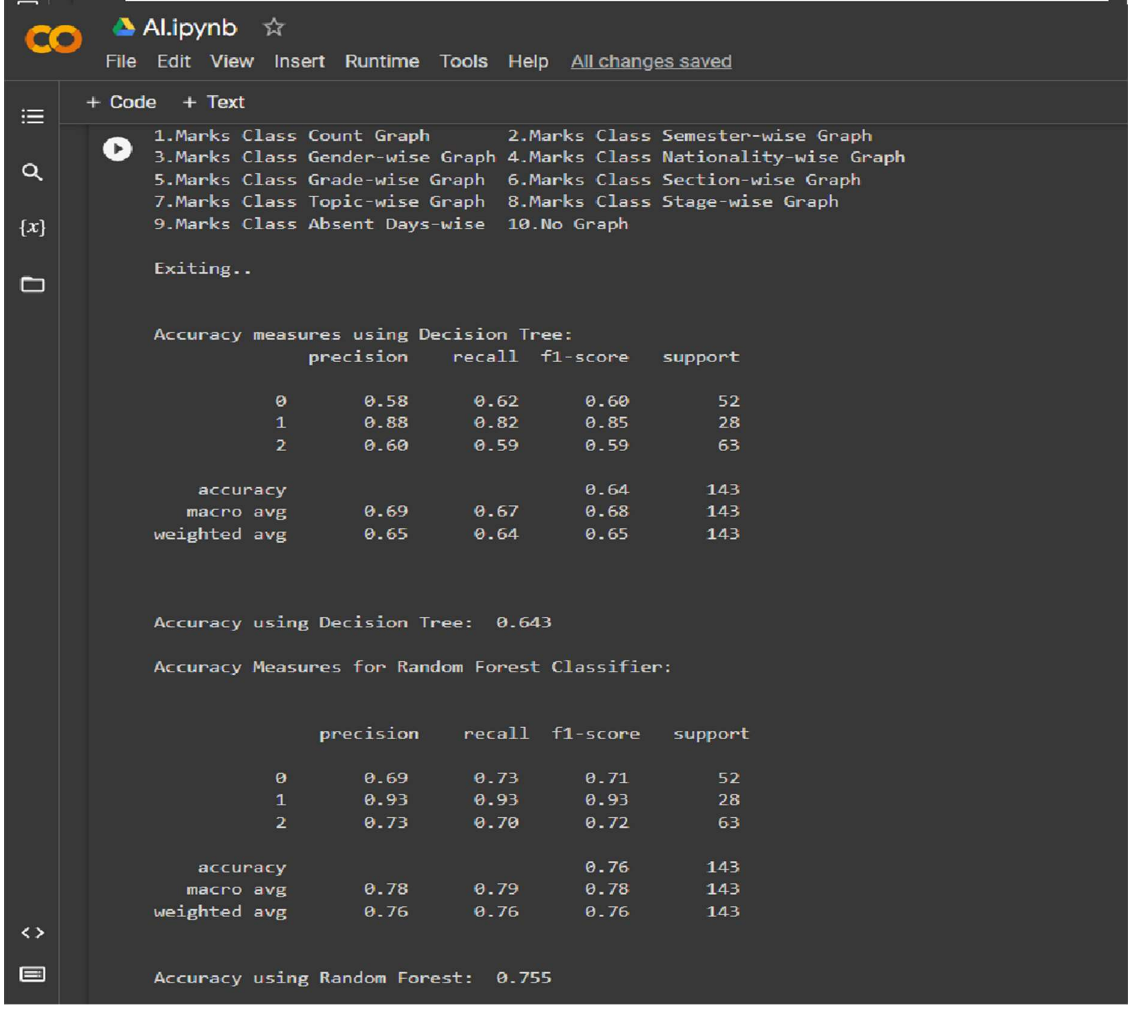
```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Loading Graph..

        Marks Class Section-wise Graph
```



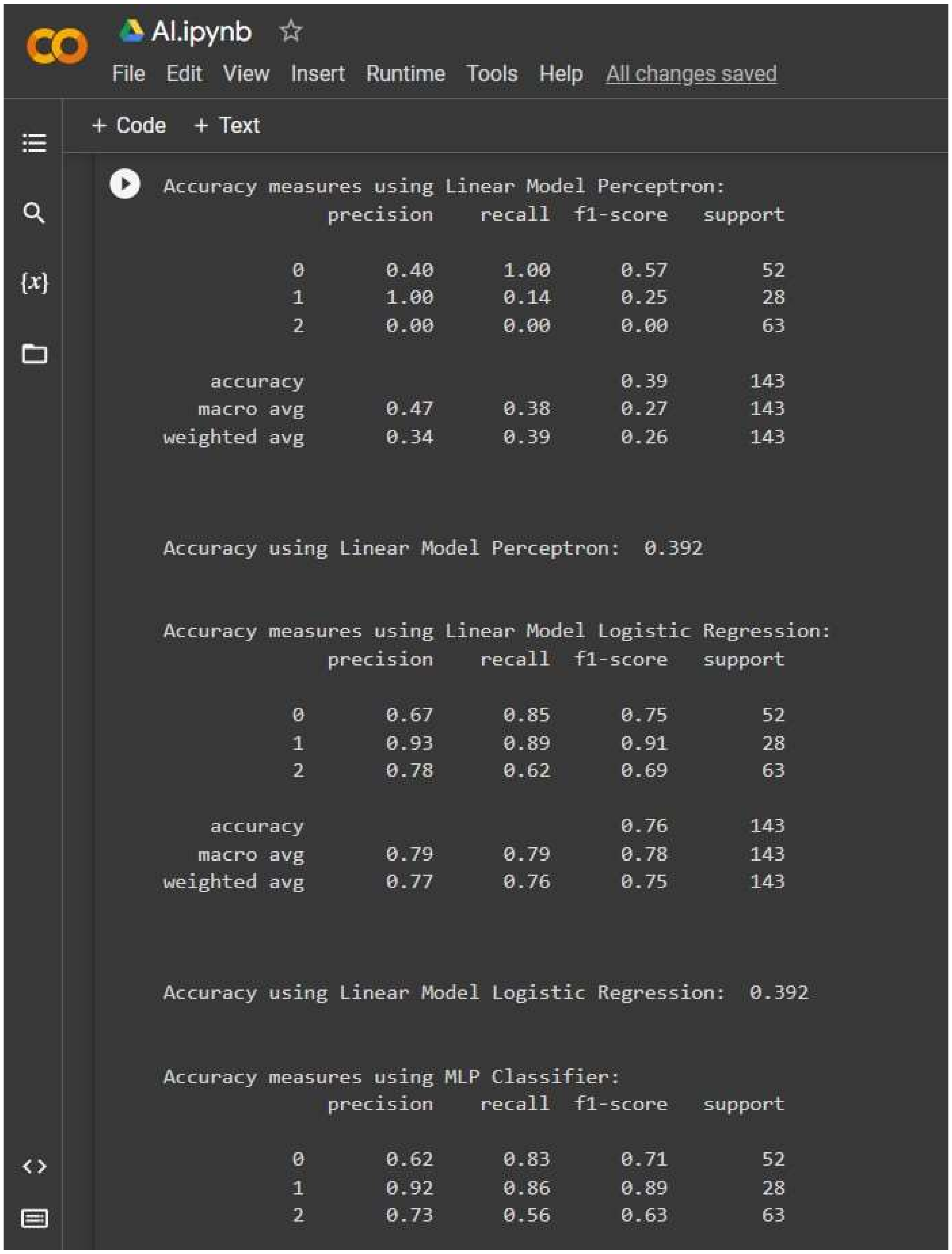
```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Loading Graph..

        Marks Class Absent Days-wise Graph
```

```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Loading Graph....

        Marks Class Semester-wise Graph
```





```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Exiting..


Accuracy measures using Decision Tree:
              precision    recall  f1-score   support

           0       0.58      0.62      0.60        52
           1       0.88      0.82      0.85        28
           2       0.60      0.59      0.59        63

    accuracy                           0.64       143
   macro avg       0.69      0.67      0.68       143
weighted avg       0.65      0.64      0.65       143



Accuracy using Decision Tree:  0.643

Accuracy Measures for Random Forest Classifier:


               precision    recall  f1-score   support

           0       0.69      0.73      0.71        52
           1       0.93      0.93      0.93        28
           2       0.73      0.70      0.72        63

    accuracy                           0.76       143
   macro avg       0.78      0.79      0.78       143
weighted avg       0.76      0.76      0.76       143


Accuracy using Random Forest:  0.755
```



```
Accuracy measures using Linear Model Perceptron:
              precision    recall  f1-score   support

           0       0.40      1.00      0.57        52
           1       1.00      0.14      0.25        28
           2       0.00      0.00      0.00        63

    accuracy                           0.39       143
   macro avg       0.47      0.38      0.27       143
weighted avg       0.34      0.39      0.26       143



Accuracy using Linear Model Perceptron:  0.392


Accuracy measures using Linear Model Logistic Regression:
              precision    recall  f1-score   support

           0       0.67      0.85      0.75        52
           1       0.93      0.89      0.91        28
           2       0.78      0.62      0.69        63

    accuracy                           0.76       143
   macro avg       0.79      0.79      0.78       143
weighted avg       0.77      0.76      0.75       143



Accuracy using Linear Model Logistic Regression:  0.392


Accuracy measures using MLP Classifier:
              precision    recall  f1-score   support

           0       0.62      0.83      0.71        52
           1       0.92      0.86      0.89        28
           2       0.73      0.56      0.63        63
```

CHAPTER VI
CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

In conclusion, a student performance prediction model is a valuable tool that can help educators and administrators identify students who may be at risk of falling behind academically. By using a well-defined methodology that involves data collection, preprocessing, feature selection, model selection, training, testing, and validation, we can build accurate models that provide valuable insights into the factors that contribute to academic success.

However, it is important to acknowledge the ethical implications of using such a model and to ensure that it is used responsibly and transparently. Care must be taken to avoid biases and to protect the privacy and security of student data.

Overall, the development of a student performance prediction model is an ongoing process that requires ongoing monitoring and fine-tuning. By continuing to explore new approaches and techniques, we can build models that are more accurate, effective, and ethical, and that ultimately help to improve student outcomes and promote academic success.

FUTURE ENHANCEMENT:

There are several ways in which a student performance prediction model can be further enhanced in the future. Some potential areas for improvement include:

6.1 Incorporating more diverse and relevant data:

While current models rely on academic and demographic data, incorporating additional data such as social media activity, extracurricular activities, and mental health indicators could provide valuable insights into factors that contribute to academic success.

6.2 Using more advanced machine learning models:

Deep learning and reinforcement learning models have shown promising results in predicting student performance, and further exploration of these models could lead to even greater accuracy.

6.3 Addressing ethical concerns:

Ensuring that the model is transparent, unbiased, and respects student privacy is critical to its success. More research and development are needed to ensure that these concerns are adequately addressed.

6.4 Developing personalized interventions:

Once a student has been identified as at-risk, the model could be used to develop personalized interventions that address their specific needs and challenges. This could include recommendations for additional resources, tutoring, or counseling services.

6.5 Improving data quality and availability:

Collecting high-quality and comprehensive data is critical to the accuracy of the model. Efforts should be made to improve data collection processes and make data more widely available to researchers and practitioners.

Overall, the development of a student performance prediction model is an ongoing process that requires ongoing innovation and collaboration between researchers, educators, and policymakers. By continually improving the model and ensuring that it is used in an ethical and responsible manner, we can help to improve student outcomes and promote academic success for all students.

REFERENCES

[1] J. Xu, K. H. Moon, and M. Van Der Schaar, “A Machine Learning Approach for Tracking and Predicting Student Performance in Degree Programs,” IEEE J. Sel. Top. Signal Process., vol. 11, no. 5, pp. 742–753, 2017.

[2] K. P. Shaleena and S. Paul, “Data mining techniques for predicting student performance,” in ICETECH 2015 - 2015 IEEE International Conference on Engineering and Technology, 2015, no. March, pp. 0–2.

[3] A. M. Shahiri, W. Husain, and N. A. Rashid, “A Review on Predicting Student’s Performance Using Data Mining Techniques,” in Procedia Computer Science, 2015.

[4] Y. Meier, J. Xu, O. Atan, and M. Van Der Schaar, “Predicting grades,” IEEE Trans. Signal Process., vol. 64, no. 4, pp. 959–972, 2016.

[5] P. Guleria, N. Thakur, and M. Sood, “Predicting student performance using decision tree classifiers and information gain,” Proc. 2014 3rd Int. Conf. Parallel, Distrib. Grid Comput. PDGC 2014, pp. 126–129, 2015.

[6] P. M. Arsad, N. Buniyamin, and J. L. A. Manan, “A neural network students’ performance
prediction model (NNSPPM),” 2013 IEEE Int. Conf. Smart Instrumentation, Meas. Appl. ICSIMA 2013, no. July 2006, pp. 26–27, 2013.

[7] K. F. Li, D. Rusk, and F. Song, “Predicting student academic performance,” Proc. - 2013 7th Int. Conf. Complex, Intelligent, Softw. Intensive Syst. CISIS 2013, pp. 27–33, 2013.

[8] G. Gray, C. McGuinness, and P. Owende, “An application of classification models to predict learner progression in tertiary education,” in Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014, 2014.

[9] N. Buniyamin, U. Bin Mat, and P. M. Arshad, “Educational data mining for prediction and classification of engineering students achievement,” 2015 IEEE 7th Int. Conf. Eng. Educ. ICEED 2015, pp. 49–53, 2016.

[10] Z. . Alharbi, J. . Cornford, L. . Dolder, and B. . De La Iglesia, “Using data mining techniques to predict students at risk of poor performance,” Proc. 2016 SAI Comput. Conf. SAI 2016, pp. 523–531, 2016.