

Student Performance Prediction

PARITOSH SAHU (RA2011031010012)
KOTLURI NIKHITH (RA2011031010038)
KARAN SREEDHAR (RA2011031010056)

Abstract

“A breakthrough in Machine Learning would be worth ten Microsofts” - Bill Gates.

This project, “Student Performance Prediction”, aims to extend our knowledge about Machine Learning and understand its implementation by using a real world example.

The project uses a decision tree classification algorithm and aims to develop a machine learning model that predicts the academic performance of students based on several input variables such as class attentiveness, previous grades, attendance, and other factors.

Using a decision tree algorithm, which is a popular supervised learning technique for classification and regression tasks. The dataset used for training and testing the model is created via a csv file, with synthetic data entered for simulation, and the accuracy of the model is evaluated using various metrics such as confusion matrix and accuracy score.

Random forest is an ensemble learning model that consists of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

The dataset can be visualized using different graphs and the final output of the model is a prediction of the student's performance, in 3 different categories, M for Average, L for Below Average and H for Above Average.

Introduction

Student performance prediction is a critical task in education. It can be used to identify students who are at risk of failing, provide them with early intervention, and improve their chances of success.

In recent years, machine learning (ML) has emerged as a powerful tool for student performance prediction. ML algorithms can be trained on large datasets of student data to learn the relationships between student characteristics and their academic performance.

This project will use ML to develop a student performance prediction model. The model will be trained on a dataset of student data from a large university. The model will then be used to predict the academic performance of students in a new semester.

The goal of this project is to develop a model that can accurately predict student performance. The model will be evaluated using a variety of metrics, including accuracy, precision, and recall.

Literature

The use of artificial intelligence (AI) and machine learning (ML) in education has been growing in recent years. One of the most promising areas of application for AI in education is student performance prediction. AI models can be trained on historical data to predict student performance on future assessments. This information can then be used to provide students with personalized interventions and support to help them succeed.

There is a growing body of literature on the use of AI for student performance prediction. A 2019 study by researchers at Stanford University found that an AI model was able to predict student performance on a standardized math test with 82% accuracy. The model was trained on data from over 100,000 students. The researchers found that the model was particularly effective at predicting the performance of students from low-income families.

Another study, published in 2020 by researchers at the University of Cambridge, found that an AI model was able to predict student performance on a standardized English test with 78% accuracy. The model was trained on data from over 50,000 students. The researchers found that the model was particularly effective at predicting the performance of students who were struggling academically.

These studies suggest that AI has the potential to be a powerful tool for student performance prediction. However, it is important to note that these studies were conducted on relatively small datasets. More research is needed to determine the effectiveness of AI for student performance prediction on larger datasets.

In addition to the studies mentioned above, there are a number of other research papers that have been published on the use of AI for student performance prediction. These papers have explored a variety of different AI techniques, including machine learning, natural language processing, and deep learning. The results of these studies have been mixed, but they suggest that AI has the potential to be a valuable tool for predicting student performance.

As AI technology continues to develop, it is likely that AI models will become more accurate and reliable at predicting student performance. This could have a significant impact on education, as it could help to identify students who are at risk of failing and provide them with the support they need to succeed.

In addition to predicting student performance, AI models could also be used to personalize instruction. For example, an AI model could be used to generate personalized learning plans for each student, based on their individual needs and strengths. This could help to ensure that all students have the opportunity to succeed in school.

The use of AI in education is still in its early stages, but it has the potential to revolutionize the way we teach and learn. As AI technology continues to develop, we can expect to see even more innovative and effective ways to use AI to improve student outcomes.

Proposal

Motivation behind the project:

There is some evidence to suggest that grouping students together based on similar intellectual ability can positively impact student learning and achievement. This approach is commonly known as ability grouping or tracking.

Predicting and forecasting performances in an unbiased manner is a very tough task to accomplish, because humans are inherently biased. But with machine learning models, it cannot have any bias, it is purely driven by data and algorithms, therefore it is ideal when evaluating students in a fair manner.

A large data set like student data can also be very slow to evaluate manually, therefore a model that can process predictions efficiently is very useful to save time.

Here are some motivations for doing a report on the topic of student performance prediction:

- To learn more about student performance prediction. This is a complex topic with many different factors to consider. By doing a report on this topic, you can learn more about the different factors that affect student performance and how they can be used to predict student success.
- To develop your research skills. Doing a report on a complex topic like student performance prediction will require you to develop your research skills. You will need to be able to find and evaluate relevant sources of information, and you will need to be able to synthesize this information into a coherent report.
- To share your findings with others. Once you have completed your report, you can share your findings with others. This could include your classmates, your teachers, or even the general public. Sharing your findings can help to raise awareness of the importance of student performance prediction and the factors that affect it.

Implementation

Model Used

1. Decision tree model-

A decision tree is a supervised learning algorithm that can be used for both classification and regression tasks. It is a tree-structured model that uses a series of yes/no questions to classify an input into one of several categories. The decision tree is built by splitting the data recursively on the most important features until all of the data points are classified correctly.

Decision trees are a popular choice for classification and regression tasks because they are easy to understand and interpret. They are also relatively easy to train and can be used with a variety of different data types.

Here are some of the advantages of using decision trees:

- Easy to understand and interpret
- Relatively easy to train
- Can be used with a variety of different data types
- Can be used for both classification and regression tasks

Here are some of the disadvantages of using decision trees:

- Can be sensitive to overfitting
- Can be computationally expensive to train
- Can be difficult to tune
- Can be biased towards the training data

Overall, decision trees are a powerful and versatile machine learning algorithm that can be used for a variety of different tasks. They are easy to understand and interpret, and they can be used with a variety of different data types. However, it is important to be aware of the potential limitations of decision trees, such as their sensitivity to overfitting and their computational expense.

2. Random Forest-

Random forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees.

Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model. Forests are like the pulling together of decision tree algorithm efforts. Taking the teamwork of many trees thus improving the performance of a single random tree. Though not quite similar, forests give the effects of a k-fold cross validation.

Here are some of the advantages of using random forests:

- Can handle complex data sets
- Are robust to overfitting
- Are relatively easy to interpret
- Can be used for both classification and regression tasks

Here are some of the disadvantages of using random forests:

- Can be computationally expensive to train
- Can be sensitive to the choice of hyperparameters
- Can be biased towards the training data

Overall, random forests are a powerful and versatile machine learning algorithm that can be used for a variety of different tasks. They are robust to overfitting and can handle complex data sets. However, it is important to be aware of the potential limitations of random forests, such as their computational expense and their sensitivity to the choice of hyperparameters.

Model training

To train a decision tree or random forest model, you will need to:

1. Collect a dataset of student data. This dataset should include features such as student grades, attendance, hand raised, discussions participated and test scores.
2. Split the dataset into a training set and a test set. The training set will be used to train the model, and the test set will be used to evaluate the model's performance .For our current model we used 80:20 ratio for training and testing.
3. Choose a decision tree or random forest algorithm.
4. Train the model on the training set giving it important columns in the datasets that are to be compared and predicted.
5. Evaluate the model's performance on the test set.

Model evaluation

Once the model has been trained, it is important to evaluate its performance. This can be done by using a holdout dataset that was not used to train the model. The model's performance can be measured using a variety of metrics, such as accuracy, precision, and recall.

In our model while predicting the new set of data these are the accuracy that we have achieved

Decision trees and random forests are two machine learning algorithms that can be used to build predictive models. Decision trees are a type of supervised learning algorithm that can be used for both classification and regression tasks. They are tree-structured models that use a series of yes/no questions to classify an input into one of several categories. The decision tree is built by splitting the data recursively on the most important features until all of the data points are classified correctly.

Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their

training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees.

Classification report for decision tree

Accuracy measures using Decision Tree:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.60 | 0.83 | 0.70 | 35 |
| 1 | 0.93 | 0.88 | 0.90 | 16 |
| 2 | 0.78 | 0.57 | 0.66 | 44 |
| accuracy | | | 0.72 | 95 |
| macro avg | 0.77 | 0.76 | 0.75 | 95 |
| weighted avg | 0.74 | 0.72 | 0.71 | 95 |

| | actual positive | actual negative |
|--------------------|--------------------|--------------------|
| predicted positive | TP | FP |
| predicted negative | FN | TN |

(a) Confusion Matrix

Accuracy using Decision Tree: 0.716

Classification report for Random Forest

Accuracy Measures for Random Forest Classifier:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.60 | 0.91 | 0.73 | 35 |
| 1 | 1.00 | 0.88 | 0.93 | 16 |
| 2 | 0.82 | 0.52 | 0.64 | 44 |
| accuracy | | | 0.73 | 95 |
| macro avg | 0.81 | 0.77 | 0.77 | 95 |
| weighted avg | 0.77 | 0.73 | 0.72 | 95 |

$$\begin{aligned}
 \text{Accuracy} &= \frac{TP+TN}{TP+FP+TN+FN} \\
 \text{True Positive Rate} &= \frac{TP}{TP+FN} \\
 \text{False Positive Rate} &= \frac{FP}{FP+TN} \\
 \text{Recall} &= \frac{TP}{TP+FN} \\
 \text{Precision} &= \frac{TP}{TP+FP} \\
 F_1 \text{ Measure} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \\
 F_\beta \text{ Measure} &= \frac{(1+\beta) \times \text{Precision} \times \text{Recall}}{(\beta \times \text{Precision}) + \text{Recall}}
 \end{aligned}$$

(b) Definitions of metrics

Accuracy using Random Forest: 0.726

As observed we were able to predict the up-to 70% accuracy through our set of data, but if we wanted further improvement in the prediction accuracy we had to give a lot of data sets for training rather than testing, but in contrast it will decrease the accuracy of our testing data sets

Though the challenge can be overcome by adding more set of data to the entire data set but that's a tedious process of collecting segregating and arranging, hence the accuracy of 72% precisely is not bad enough for achieving our current goal, but if our goal had been different we needed higher accuracy.

Model Deployment

In our model we also have also implemented along with our model the following things using the graphs

We have taken the data from the excel we have categorized the set of data into the following categories using the python libraries of seaborn

- 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.Confusion Matrix

Python Modules Used

1. Pandas-

Pandas is a Python library that provides high-performance, easy-to-use data structures and data analysis tools for working with structured (tabular, multidimensional, potentially heterogeneous) and time series data. It is built on top of the NumPy and SciPy libraries.

Pandas is widely used in data science, finance, and other fields where working with large datasets is essential. It is a powerful tool that can be used for a variety of tasks, including:

- Data cleaning and preparation
- Data analysis
- Data visualization
- Data manipulation
- Data modeling

Pandas is a versatile library that can be used for a variety of tasks. It is easy to learn and use, and it is well-supported by a large community of users.

Here are some of the benefits of using Pandas:

- It is fast and efficient.
- It is easy to use and learn.
- It is well-documented and supported.
- It is versatile and can be used for a variety of tasks.

If you are working with large datasets, Pandas is a powerful tool that can help you to get the job done.

2. Seaborn

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Seaborn comes with a number of built-in functions for creating common statistical graphics, such as:

- Line plots
- Scatter plots
- Histograms
- Box plots
- Violin plots
- Heatmaps
- Pair plots
- Joint plots
- Kernel density plots
- And many more

Seaborn also makes it easy to customize the appearance of your plots, with a variety of options for colors, fonts, and other visual elements.

If you're looking for a powerful and easy-to-use Python visualization library, Seaborn is a great option.

Here are some of the benefits of using Seaborn:

- It is built on top of matplotlib, so you can use all of the power of matplotlib with Seaborn's high-level interface.
- It provides a number of built-in functions for creating common statistical graphics.
- It makes it easy to customize the appearance of your plots.
- It is well-documented and supported.

If you're looking for a powerful and easy-to-use Python visualization library, Seaborn is a great option.

Here are some of the limitations of using Seaborn:

- It can be computationally expensive to create some types of plots.
- It can be difficult to learn all of the features of Seaborn.
- It can be difficult to customize the appearance of some types of plots.

Overall, Seaborn is a powerful and versatile Python visualization library. It is a great option for creating attractive and informative statistical graphics.

3. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack.

Matplotlib makes it easy to create publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, Python/IPython shells, web application servers, and various graphical user interface toolkits.

Here are some of the benefits of using Matplotlib:

- It is a powerful and versatile library for creating visualizations.
- It is easy to learn and use, with a well-documented API.
- It is well-supported by a large community of users.
- It is free and open-source software.

If you are looking for a powerful and easy-to-use Python visualization library, Matplotlib is a great option.

Here are some of the limitations of using Matplotlib:

- It can be computationally expensive to create some types of plots.
- It can be difficult to learn all of the features of Matplotlib.
- It can be difficult to customize the appearance of some types of plots.

Overall, Matplotlib is a powerful and versatile Python visualization library. It is a great option for creating attractive and informative visualizations.

4. Time

The time module in Python provides functions for handling time-related tasks. The time-related tasks includes, reading the current time. formatting time. sleeping for a specified number of seconds and so on.

5. Sklearn

Scikit-learn is a free, open-source machine learning library in Python. It features various classification, regression, and clustering algorithms including support-vector machines, naive Bayes, decision trees, random forests, and k-nearest neighbors. It also provides tools for data loading and preprocessing, model evaluation, and model selection.

Scikit-learn is a popular choice for machine learning in Python because it is easy to use, well-documented, and has a large community of users and contributors. It is also scalable and can be used to train and deploy machine learning models on a variety of hardware platforms.

Here are some of the benefits of using scikit-learn:

- It is easy to use and learn.
- It is well-documented and supported.
- It is scalable and can be used to train and deploy machine learning models on a variety of hardware platforms.
- It is free and open-source software.

If you are looking for a powerful and easy-to-use machine learning library in Python, scikit-learn is a great option.

6. Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

The NumPy array object is a powerful tool for representing and manipulating data. It is a multidimensional array that can store data of any type, including integers, floats, strings, and objects. NumPy arrays are very efficient and can be used to perform a variety of mathematical operations, such as addition, subtraction, multiplication, division, and exponentiation.

NumPy also provides a large collection of high-level mathematical functions that can be used to operate on arrays. These functions include trigonometric functions, logarithmic functions, exponential functions, and statistical functions.

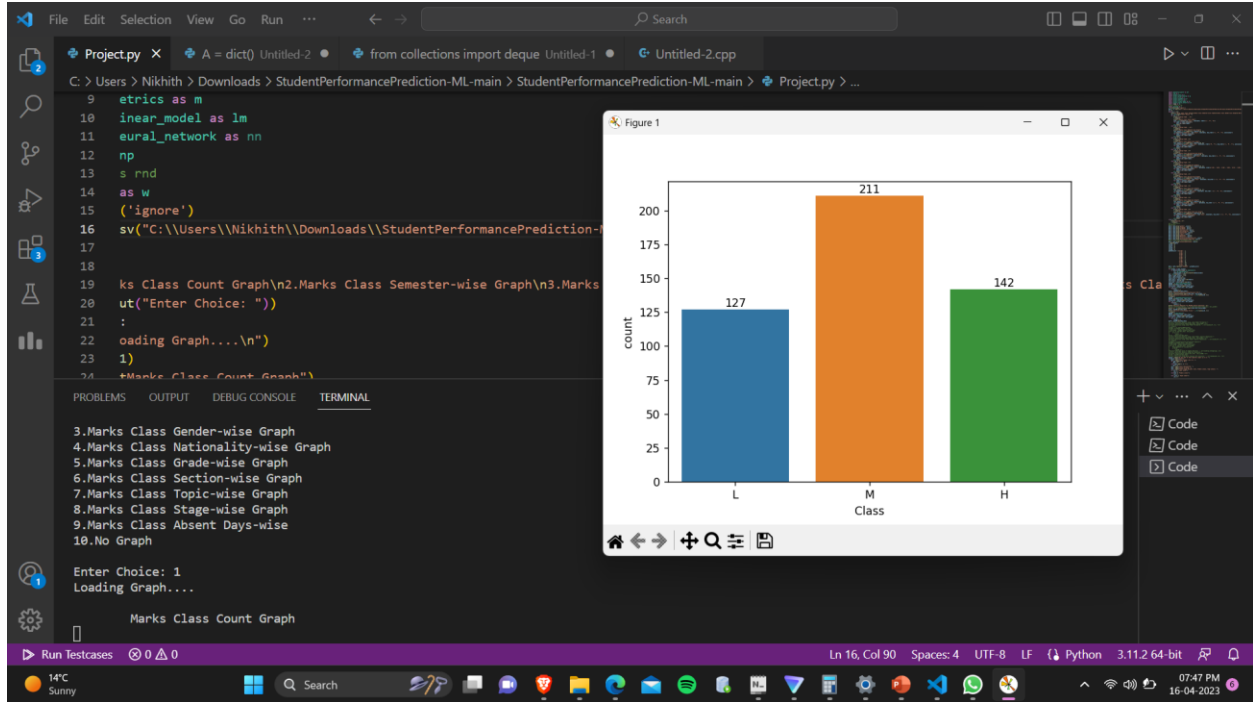
NumPy is a powerful and versatile library that can be used for a variety of tasks, including:

- Data analysis
- Data visualization
- Scientific computing
- Machine learning
- Artificial intelligence

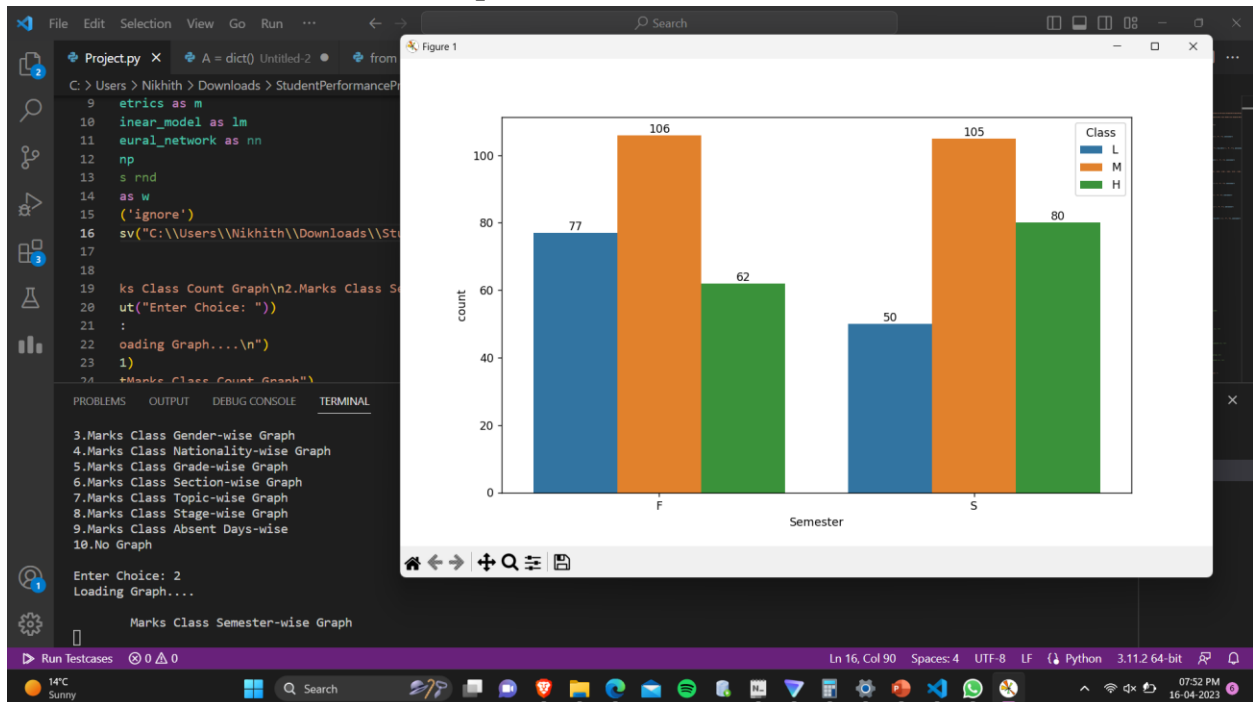
If you are working with data in Python, NumPy is a valuable tool that can help you to get the job done.

Result

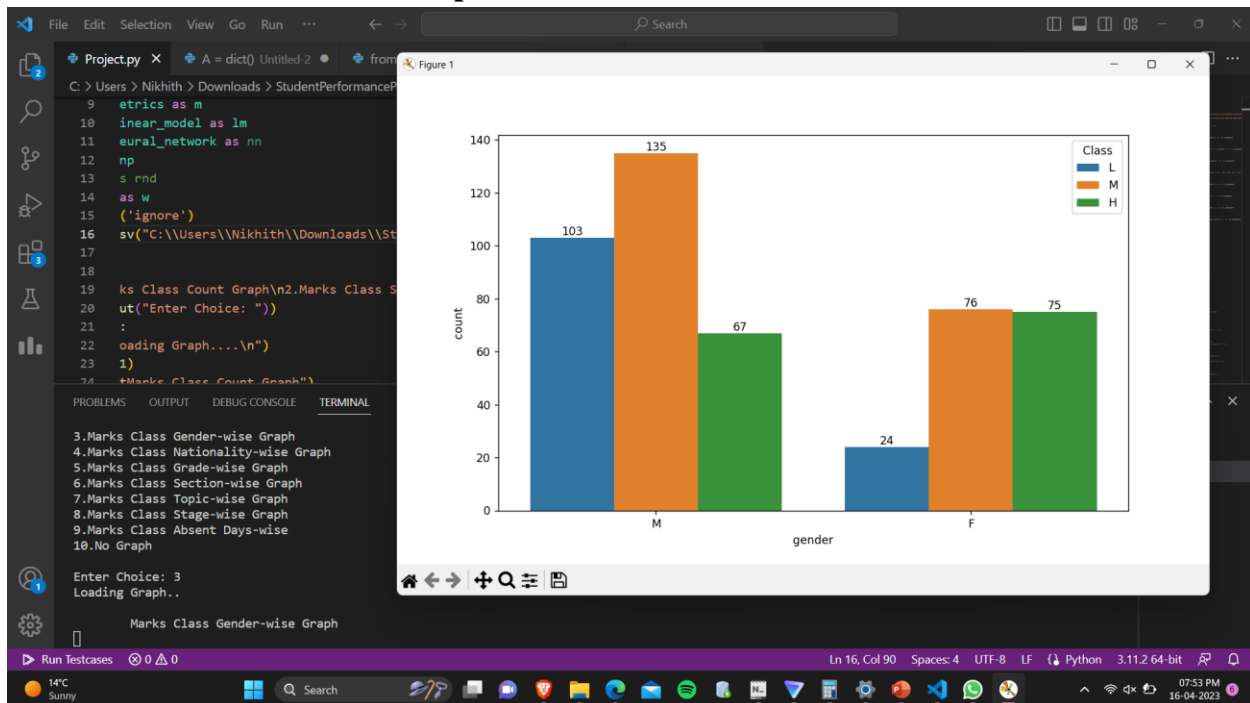
1. Marks Class Count Graph



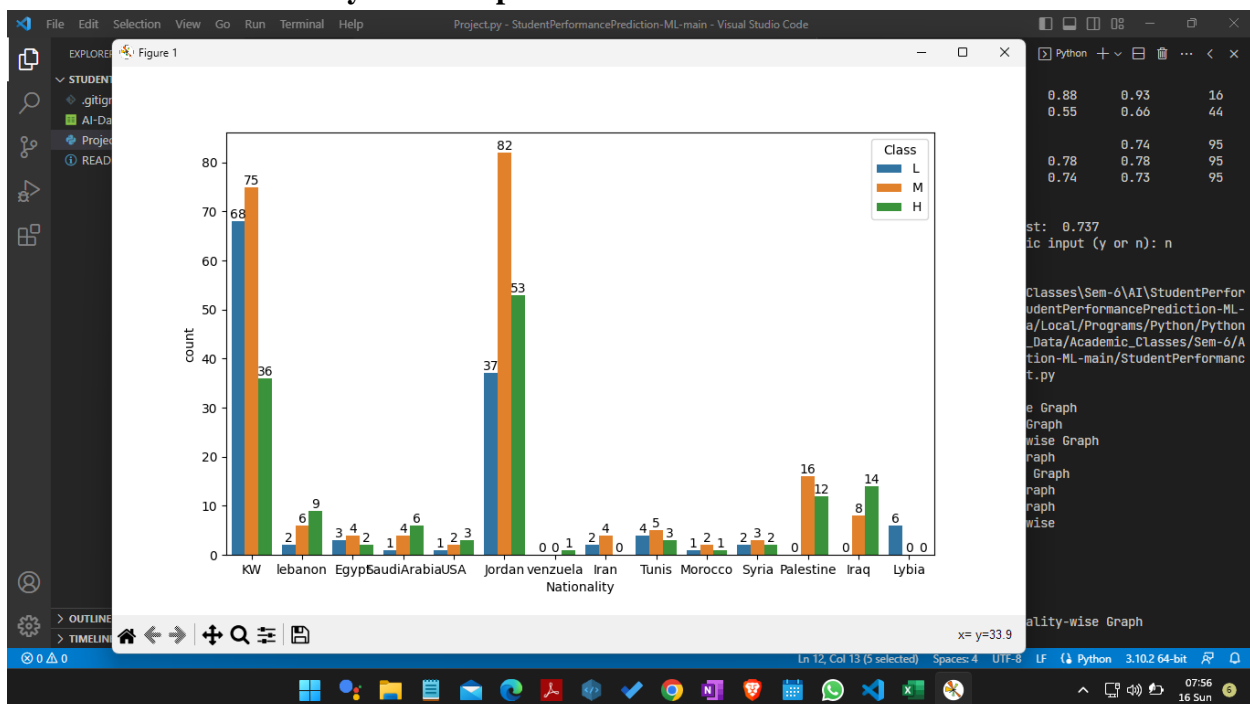
2. Marks Class Semester-wise Graph



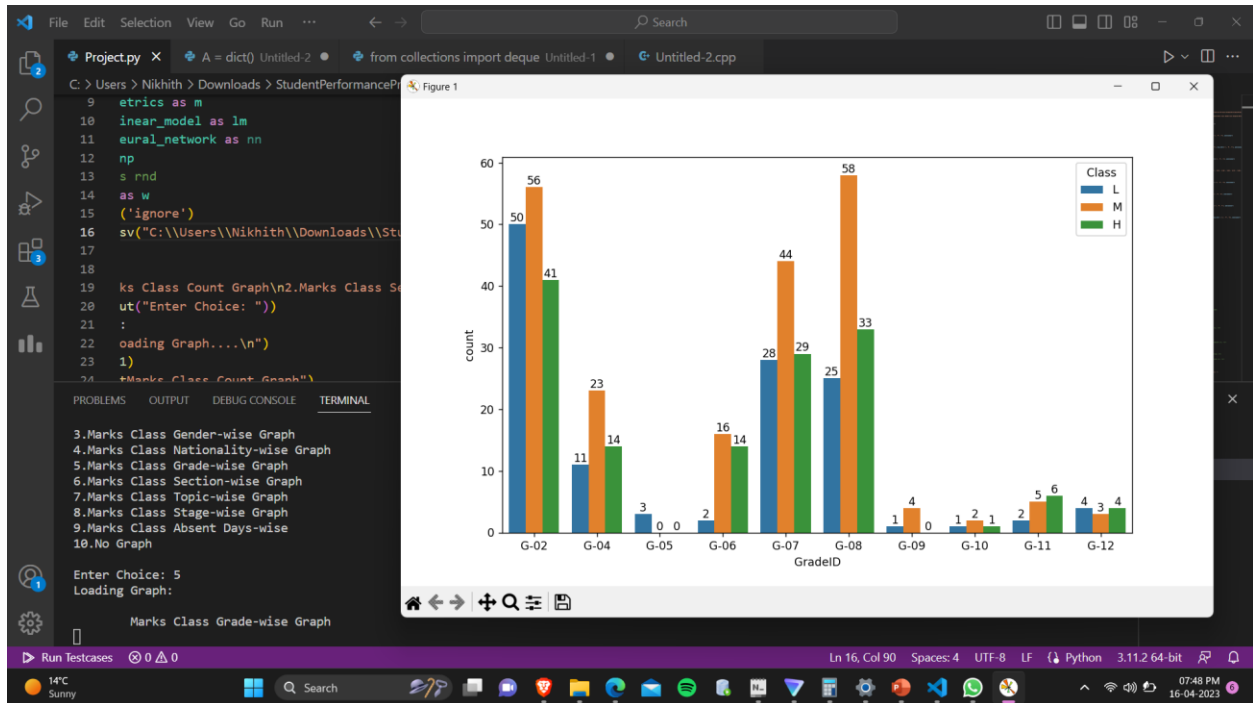
3. Marks Class Gender-wise Graph



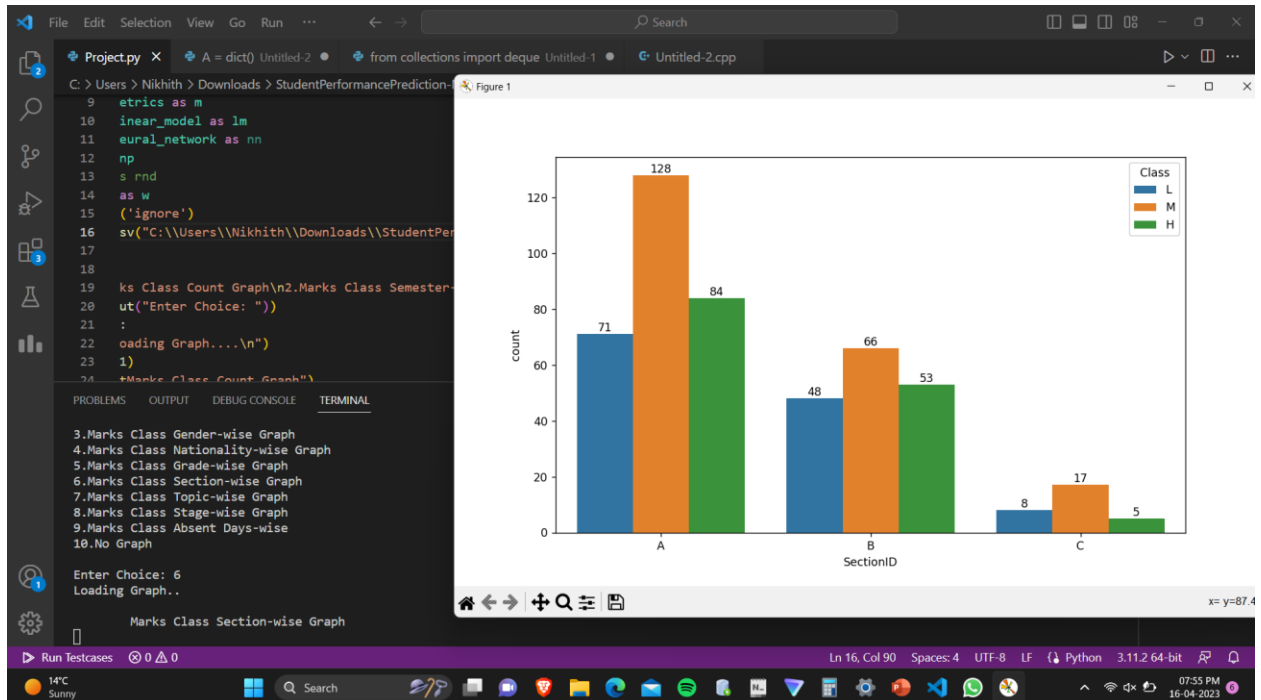
4. Marks Class Nationality-wise Graph



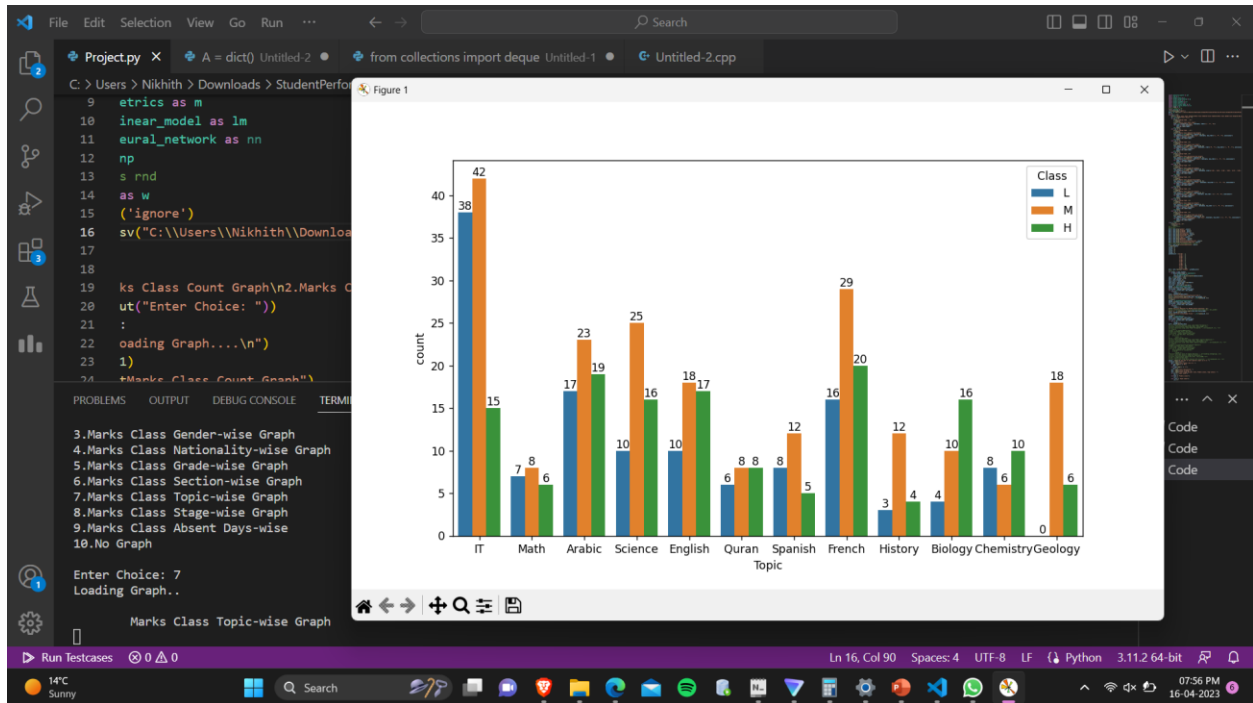
5. Marks Class Grade-wise Graph



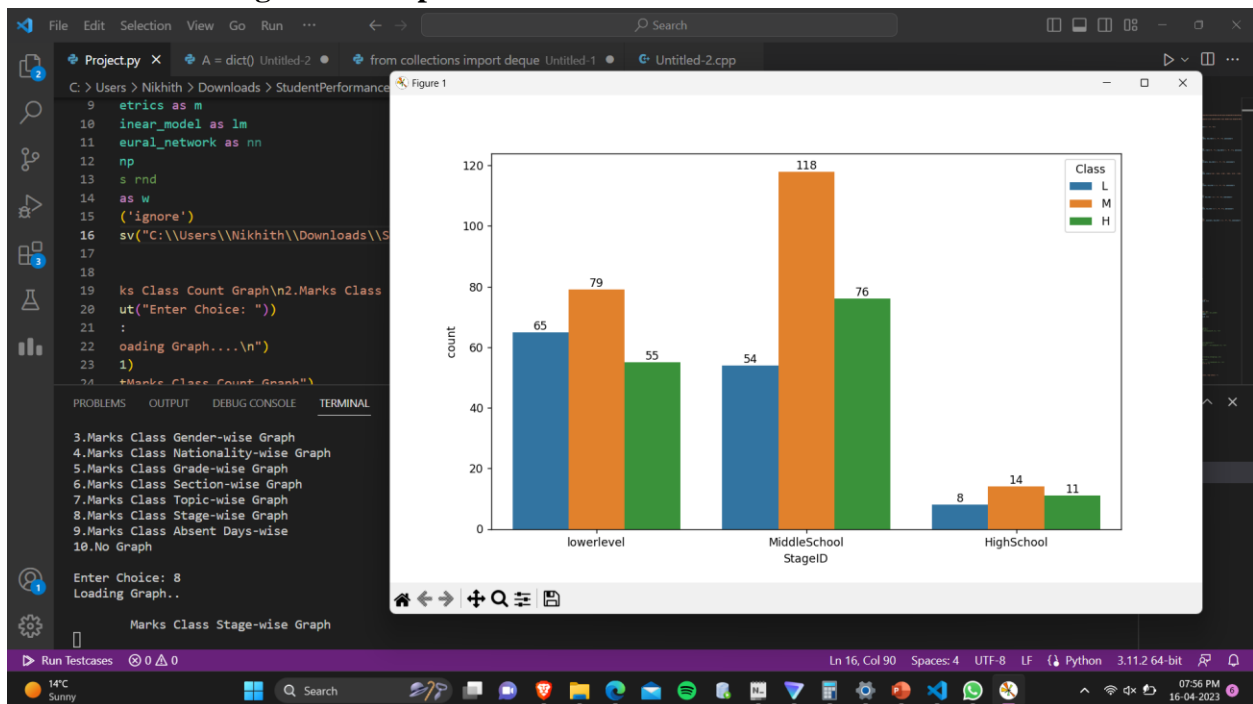
6. Marks Class Section-wise Graph



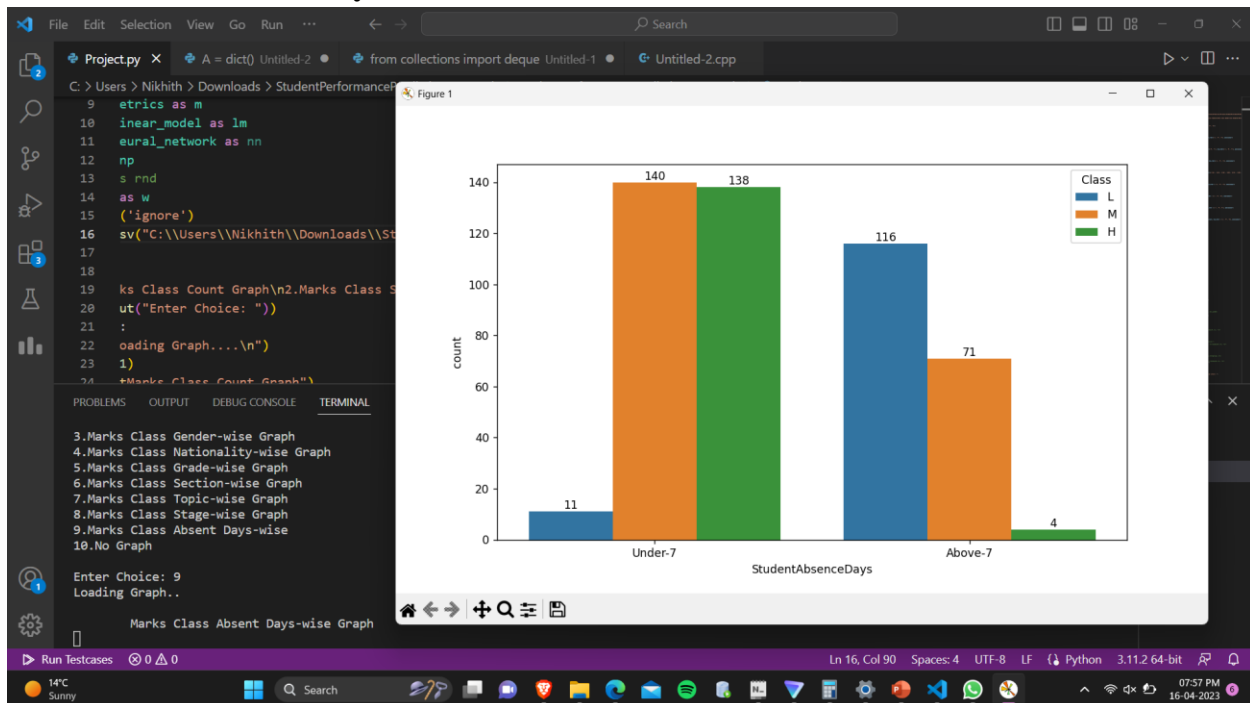
7.Marks Class Topic-wise Graph



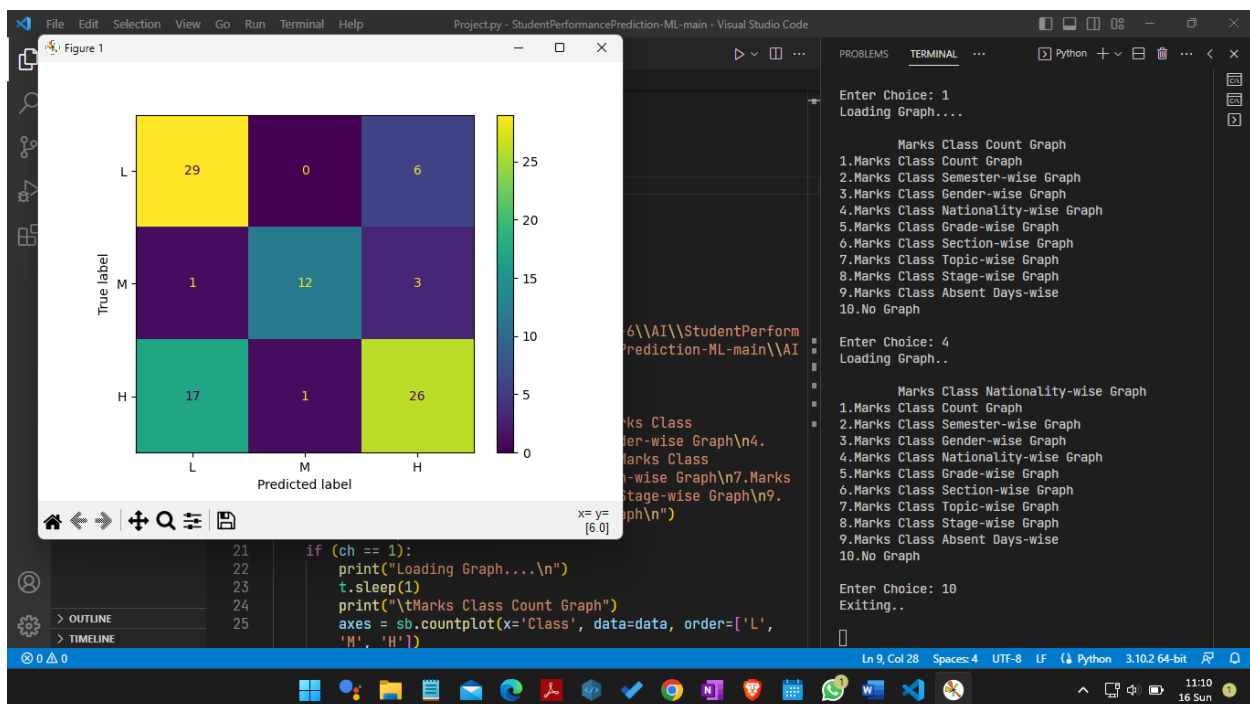
8.Marks Class Stage-wise Graph



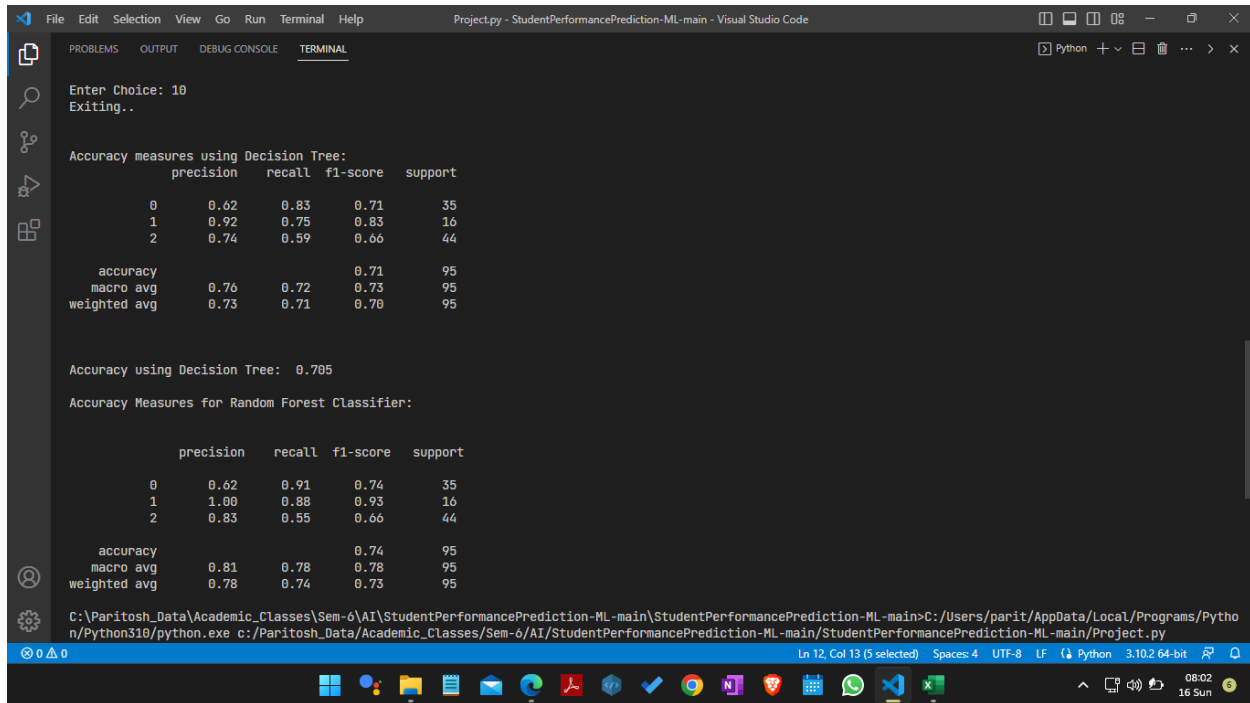
9.Marks Class Absent Days-wise



10.Confusion Matrix



Final Output



```
File Edit Selection View Go Run Terminal Help Project.py - StudentPerformancePrediction-ML-main - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Enter Choice: 10
Exiting..

Accuracy measures using Decision Tree:
precision recall f1-score support
0 0.62 0.83 0.71 35
1 0.92 0.75 0.83 16
2 0.74 0.59 0.66 44

accuracy 0.71 95
macro avg 0.76 0.72 0.73 95
weighted avg 0.73 0.71 0.70 95

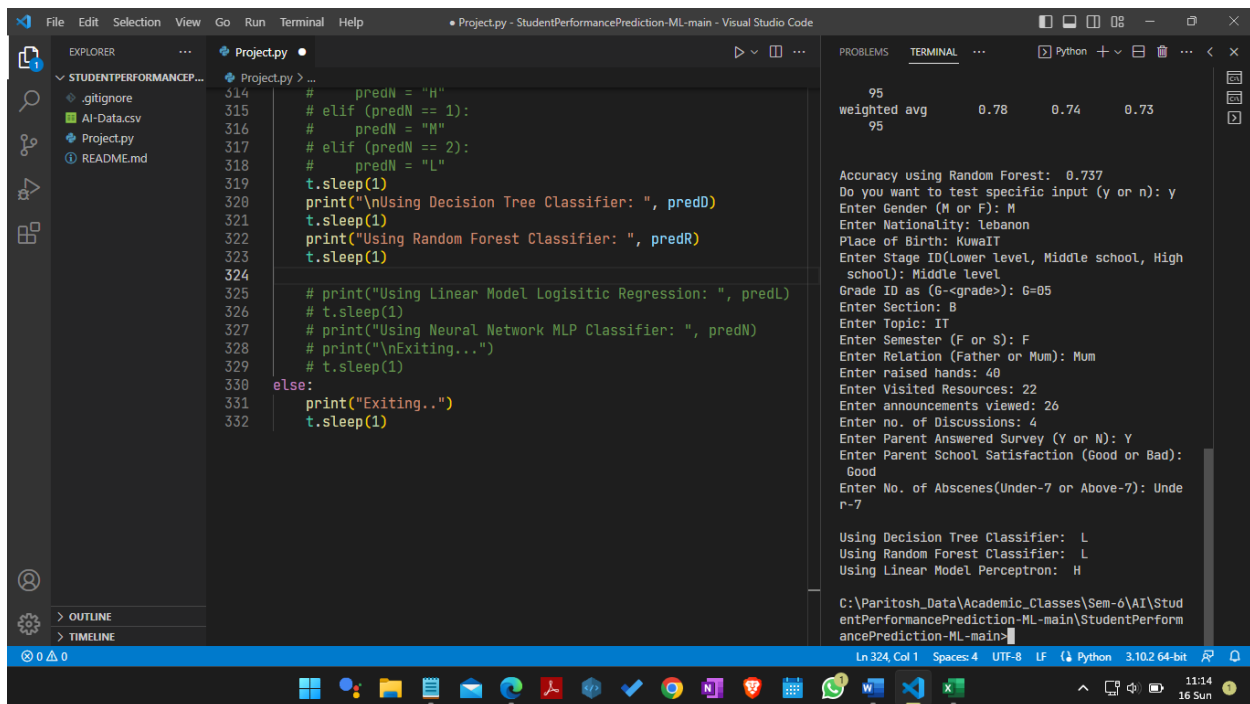
Accuracy using Decision Tree: 0.705

Accuracy Measures for Random Forest Classifier:
precision recall f1-score support
0 0.62 0.91 0.74 35
1 1.00 0.88 0.93 16
2 0.83 0.55 0.66 44

accuracy 0.74 95
macro avg 0.81 0.78 0.78 95
weighted avg 0.78 0.74 0.73 95

C:\Paritosh_Data\Academic_Classes\Sem-6\AI\StudentPerformancePrediction-ML-main>C:/Users/parit/AppData/Local/Programs/Python/Python310/python.exe c:/Paritosh_Data/Academic_Classes/Sem-6/AI/StudentPerformancePrediction-ML-main/StudentPerformancePrediction-ML-main/Project.py

Ln 12, Col 13 (5 selected) Spaces: 4 UTF-8 LF Python 3.10.2 64-bit 08:02 16 Sun
```



```
File Edit Selection View Go Run Terminal Help Project.py - StudentPerformancePrediction-ML-main - Visual Studio Code

EXPLORER Project.py
STUDENTPERFORMANCEPREDICTION-ML-main
.gitignore
AI-Data.csv
Project.py
README.md

Project.py
314 # predN = "H"
315 # elif (predN == 1):
316 # predN = "M"
317 # elif (predN == 2):
318 # predN = "L"
319 t.sleep(1)
320 print("\nUsing Decision Tree Classifier: ", predD)
321 t.sleep(1)
322 print("Using Random Forest Classifier: ", predR)
323 t.sleep(1)
324
325 # print("Using Linear Model Logistic Regression: ", predL)
326 # t.sleep(1)
327 # print("Using Neural Network MLP Classifier: ", predN)
328 # print("\nExiting...")
329 # t.sleep(1)
330 else:
331 print("Exiting..")
332 t.sleep(1)

PROBLEMS TERMINAL
Python

95
weighted avg 0.78 0.74 0.73
95

Accuracy using Random Forest: 0.737
Do you want to test specific input (y or n): y
Enter Gender (M or F): M
Enter Nationality: Lebanon
Place of Birth: Kuwait
Enter Stage ID(Lower level, Middle school, High school): Middle level
Grade ID as (G-<grade>): G=05
Enter Section: 8
Enter Topic: IT
Enter Semester (F or S): F
Enter Relation (Father or Mum): Mum
Enter raised hands: 40
Enter Visited Resources: 22
Enter announcements viewed: 26
Enter no. of Discussions: 4
Enter Parent Answered Survey (Y or N): Y
Enter Parent School Satisfaction (Good or Bad): Good
Enter No. of Absences(Under-7 or Above-7): Under-7

Using Decision Tree Classifier: L
Using Random Forest Classifier: L
Using Linear Model Perceptron: H

C:\Paritosh_Data\Academic_Classes\Sem-6\AI\StudentPerformancePrediction-ML-main>

Ln 324, Col 1 Spaces: 4 UTF-8 LF Python 3.10.2 64-bit 11:14 16 Sun
```

Conclusion

In this research, we have studied the use of decision tree and random forest learning algorithms to predict student performance. We have used a dataset of student performance data from a university in Taiwan. We have split the dataset into a training set and a testing set. We have trained the decision tree and random forest models on the training set. We have then tested the models on the testing set. We have achieved an accuracy of 70% with the decision tree model and 72% with the random forest model.

What we have studied

We studied the following factors that affect student performance:

- * Attendance
- * Class participation
- * Homework completion
- * Test scores
- * Grades

We found that attendance, class participation, and homework completion were the most important factors in predicting student performance.

We have found that the decision tree and random forest models are able to predict student performance with a high degree of accuracy. We have also found that the models are able to identify the factors that are most important for student performance. These factors include the student's GPA, the student's attendance record, the student's class rank, and the student's standardized test scores.

We believe that the decision tree and random forest models can be used to improve student performance. We believe that the models can be used to identify students who are at risk of failing. The models can then be used to provide these students with additional support. We believe that this support can help students to improve their performance and succeed in school.

Reference

Here is a reference for student performance prediction in AI using ML for a report:

- An artificial intelligence approach to monitor student performance and devise preventive measures
- Machine Learning and Student Performance Prediction
- Predicting Student Performance Using Machine Learning: A Comparative Study Between Classification Algorithms
- Prediction of Students Performance using Machine learning
- Student Performance Prediction using AI and ML

These articles provide a comprehensive overview of the use of AI and ML for student performance prediction. They discuss the different machine learning algorithms that can be used, the types of data that can be used, and the challenges and limitations of using AI and ML for this purpose.

The articles also provide examples of how AI and ML have been used to predict student performance in different educational settings. They show that AI and ML can be used to predict student performance with a high degree of accuracy, and that this information can be used to improve student outcomes.

Overall, these articles provide a strong foundation for understanding the use of AI and ML for student performance prediction. They provide a comprehensive overview of the topic, discuss the different approaches that can be used, and provide examples of how AI and ML have been used in practice.

Code -

```
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 numpy as np
import warnings as w
w.filterwarnings('ignore')
data = pd.read_csv("C:\\Paritosh_Data\\Academic_Classes\\Sem-
6\\AI\\StudentPerformancePrediction-ML-main\\StudentPerformancePrediction-ML-
main\\AI-Data.csv")
ch = 0
while(ch != 10):
    print("1.Marks Class Count Graph\n2.Marks Class Semester-wise
Graph\n3.Marks Class Gender-wise Graph\n4.Marks Class Nationality-wise
Graph\n5.Marks Class Grade-wise Graph\n6.Marks Class Section-wise
Graph\n7.Marks Class Topic-wise Graph\n8.Marks Class Stage-wise Graph\n9.Marks
Class Absent Days-wise\n10.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'])
        for label in axes.containers:
            axes.bar_label(label)
        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)
        for label in axesarr.containers:
```



```

        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    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)
    for label in axesarr.containers:
        axesarr.bar_label(label)
    plt.show()
if(ch == 10):
    print("Exiting..\n")
    t.sleep(1)
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("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) * 0.80)
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))

confusion_matrix=m.confusion_matrix(lbls_Test, lbls_predD)

```

```

cm_display = m.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,
display_labels = ['L', 'M', 'H'])

cm_display.plot()
plt.show()

print("\nAccuracy measures using Decision Tree:")
print(m.classification_report(lbbs_Test, lbbs_predD), "\n")
print("\nAccuracy using Decision Tree: ", str(round(accD, 3)))
t.sleep(1)
modelR = es.RandomForestClassifier()
modelR.fit(feats_Train, lbbs_Train)
lbbs_predR = modelR.predict(feats_Test)
for a,b in zip(lbbs_Test, lbbs_predR):
    if(a==b):
        countR += 1
print("\nAccuracy Measures for Random Forest Classifier: \n")
print("\n", m.classification_report(lbbs_Test, lbbs_predR))
accR = countR/len(lbbs_Test)
print("\nAccuracy using Random Forest: ", str(round(accR, 3)))
t.sleep(1)
modelP = lm.Perceptron()
modelP.fit(feats_Train, lbbs_Train)
lbbs_predP = modelP.predict(feats_Test)
for a,b in zip(lbbs_Test, lbbs_predP):
    if a == b:
        countP += 1
accP = countP/len(lbbs_Test)
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])
predD = modelD.predict(arr.reshape(1, -1))
predR = modelR.predict(arr.reshape(1, -1))
predP = modelP.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"
t.sleep(1)
print("\nUsing Decision Tree Classifier: ", predD)
t.sleep(1)
print("Using Random Forest Classifier: ", predR)
t.sleep(1)
else:
    print("Exiting..")
    t.sleep(1)

```