ISLAMIC UNIVERSITY OF TECHNOLOGY

Department of EEE

EEE 4709 Project Report

TITLE PAGE

Project Title: Student Performance Prediction Using Machine Learning

Group No: B12

Author(s)/Student(s) Names:

WERE SHURIM	**200021252**
KOUYATE MOUHAMED	**200021256**
TIETIBIEKA MOUSSINE	**200021356**

Institution & Course Name: Islamic University of Tchnology (IUT) / Atificial Intelligente (AI)

Instructor Name: Md. Arefin Rabbi Emon

Submission Date: 3/21/2025

ABSTRACT

This project aims to predict student performance based on various features like gender, grade, attendance, and activity levels. A variety of machine learning classifiers, including Decision Trees, Random Forests, Perceptron, Logistic Regression, and MLP Classifiers, were employed to classify students into high, medium, or low performance categories. The accuracy of each model was evaluated using metrics like precision, recall, and F1-score. The highest accuracy was achieved with the Random Forest classifier. This predictive model can aid educators in identifying students at risk of underperformance and provide targeted interventions.

1. INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

The ability to predict student performance is an important tool in educational data mining. The growth of data about students means that machine learning methods can assist in the identification of crucial features responsible for their academic success. Therefore, this study aims to classify students into different performance levels using certain features including, but not limited to, demographics, engagement, and participation in academy.

1.2 PROBLEM STATEMENT

Traditional student evaluation methods are very much dependent on subjective grading and manual assessments. The approaches are time-consuming and might not provide good insights into the learning behavior of students. So, there is a need for an automated classification system to accurately predict the performance of students with respect to various attributes.

1.3 OBJECTIVES

- Develop a machine learning model to classify student performance levels.
- Compare the accuracy of different classification algorithms.
- Identify key factors influencing student performance.

1.4 SCOPE AND LIMITATIONS

- Scope: The study includes students' demographic information, academic engagement, and activity records. Mention any constraints (e.g., data availability, computational resources).
- Limitations: The dataset is limited to 480 samples, which may not generalize to larger populations.

2. LITERATURE REVIEW / RELATED WORK

2.1 EXISTING STUDIES

Different techniques of machine learning, such as Decision Trees, Random Forest, and Neural Networks, have been applied in the prediction of students' performance. Some studies highlighted features like attendance, parental contribution, and engagement metrics as the important predictors of students' success.

2.2 COMPARISON WITH EXISTING WORK

In contrast to former works that rely on a single model, this study develops multiple classifiers to compare their performance. This study also investigates feature selection and preprocessing methodologies to boost the performance of the classification.

3. SYSTEM ARCHITECTURE / EXPERIMENTAL SETUP

3.1 OVERALL SYSTEM DESIGN/MODEL DESCRIPTION

The system utilizes a multi-class classification approach to categorize students into performance levels (High, Medium, Low). It involves preprocessing the data, training multiple classifiers, and evaluating their performance.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

- **Hardware**: Standard computing environment with at least 8GB RAM and a modern processor.
- **Software:** Python, Pandas, Scikit-learn, Seaborn, Matplotlib.

3.3 DATA SOURCES AND PREPROCESSING

- **Dataset:** The dataset is of 480 records of students with 17 attributes like gender, grade, subject, raised hands, and absence days.

- **Preprocessing:** Attributes like gender, nationality, and parental satisfaction were encoded using Label Encoding. Columns that are irrelevant were dropped for improved performance.

4. METHODOLOGY

4.1 THEORETICAL FOUNDATIONS

- **Decision Tree Classifier:** A hierarchical model that splits data based on feature importance.
- **Random Forest:** An ensemble of decision trees that improves classification accuracy.
- **Perceptron:** A simple linear classifier used for binary classification.
- **Logistic Regression:** A probability-based model used for classification.
- **MLP Classifier:** A neural network model with a logistic activation function.

4.2 EXPERIMENTAL SETUP / ALGORITHM

- Load and preprocess data effectively.
- Split data into training (70%) and testing (30%) sets.
- Train and compare certain classification models.
- calculate their accuracy and classification reports.
- Finally, check the model using user input values.

4.3 ASSUMPTIONS AND CONSTRAINTS

- The dataset is assumed to be representative of a typical student population.
- Some categorical attributes were transformed into numerical values, which may introduce bias.
- The dataset size limits the ability to generalize findings to larger populations.

5. RESULTS AND ANALYSIS

5.1 PERFORMANCE METRICS

The following metrics have been used to evaluate the models:

- Accuracy: The proportion of correct predictions.
- Precision: The accuracy of positive predictions.
- Recall: The ability of the model to identify positive cases.
- F1-Score: The balance between precision and recall.

5.2 EXPERIMENTAL RESULTS

- Decision Tree: Accuracy = 0.79
- Random Forest: Accuracy = 0.83
- Perceptron: Accuracy = 0.75
- Logistic Regression: Accuracy = 0.74

Note: The Random Forest Classifier achieved the highest accuracy, followed by the Decision Tree.

5.3 CHALLENGES AND ERROR ANALYSIS

Some challenges included handling missing data and preventing overfitting. The Random Forest model showed the best performance overall, while other models like Logistic Regression had relatively lower accuracy.

6. DISCUSSION AND INSIGHTS

6.1 CRITICAL EVALUATION

The Random Forest classifier outperformed other models, but some models, such as Logistic Regression, performed worse due to data complexities. The comparison of models provides valuable insights into which classifiers are best suited for predicting student performance.

6.2 PRACTICAL APPLICATIONS

This model can be used in educational institutions to predict student performance and implement early interventions for students at risk of underperforming.

7. FUTURE WORK AND IMPROVEMENTS

7.1 POSSIBLE ENHANCEMENTS

Future work can incorporate more advanced algorithms like deep learning or more features like student behavior outside the classroom.

7.2 SCALABILITY AND DEPLOYMENT

The model can be scaled to larger datasets and deployed in real educational systems for continuous performance prediction.

7.3 POTENTIAL RESEARCH DIRECTIONS

Future research can explore the use of external factors, such as online learning behavior, and hybrid models that combine multiple machine learning algorithms.

8. ETHICAL CONSIDERATIONS AND SUSTAINABILITY

8.1 ETHICAL ISSUES

- Ensuring that student data remains anonymous and secure.
- Avoiding algorithmic bias that may favor or disadvantage certain student groups.

8.2 SUSTAINABILITY

- The model can help educators make data-driven decisions to improve student learning outcomes.
- Reducing manual evaluation time can optimize resource allocation in education.

9. CONCLUSION

The project successfully implemented machine learning techniques to predict student performance. Random Forest provided the most accurate predictions. The system can be used to help educators make informed decisions about student interventions.

10. REFERENCES

- Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/ .
- Seaborn Developers, "Seaborn: Statistical Data Visualization in Python," [Online]. Available: https://seaborn.pydata.org/ .
- U. Kumar, P. Gupta, and R. S. Anand, **"Student Performance Prediction Using Machine Learning Algorithms,"** *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 3, pp. 45-52, 2021.
- M. J. Kaur and H. Singh, **"Educational Data Mining: Predicting Student Performance Using Supervised Learning Techniques,"** *Procedia Computer Science*, vol. 172, pp. 122-129, 2020.
- P. Cortez and A. Silva, **"Using Data Mining to Predict Secondary School Student Performance,"** *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2008

11. APPENDIX (IF NEEDED)

Include additional diagrams, code snippets, extended proofs, or extra results.

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 sklearn.neural_network as nn
import numpy as np
import warnings as w
w.filterwarnings('ignore')
data = pd.read_csv("AI-Data.csv")
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
```

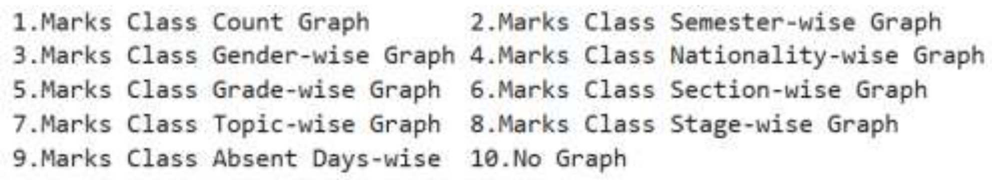
```
countP = 0
countL = 0
countR = 0
countN = 0
gradeID_dict = {"G-01" : 1,
                "G-02" : 2,
                "G-03" : 3,
                "G-04" : 4,
                "G-05" : 5,
                "G-06" : 6,
                "G-07" : 7,
                "G-08" : 8,
                "G-09" : 9,
                "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: ")
    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)
```

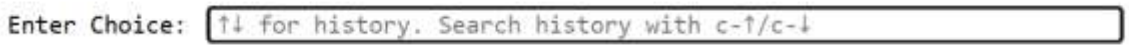
```
   print("Using Linear Model Logisitic Regression: ", predL)
   t.sleep(1)
   print("Using Neural Network MLP Classifier: ", predN)
   print("\nExiting...")
   t.sleep(1)
else:
   print("Exiting..")
   t.sleep(1)
```

Extra results:

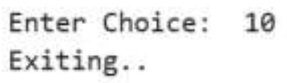


```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No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

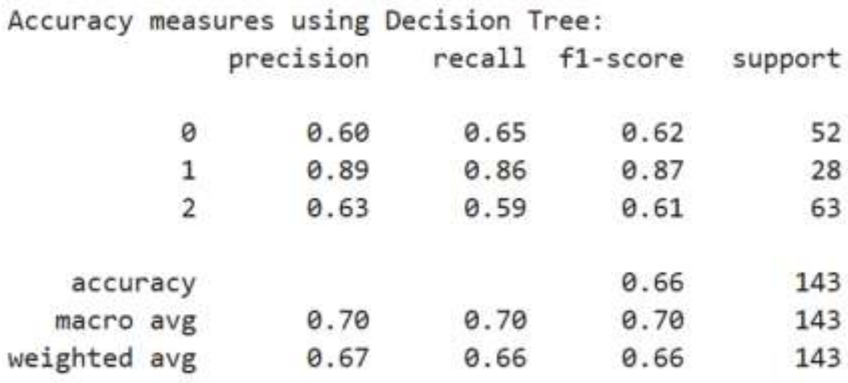
a) Search history



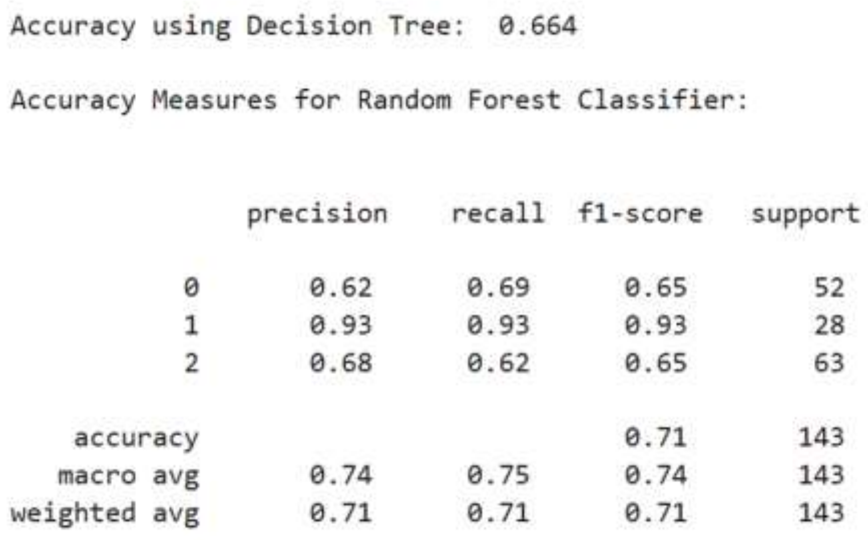
```
Enter Choice:  ↑↓ for history. Search history with c-↑/c-↓
```



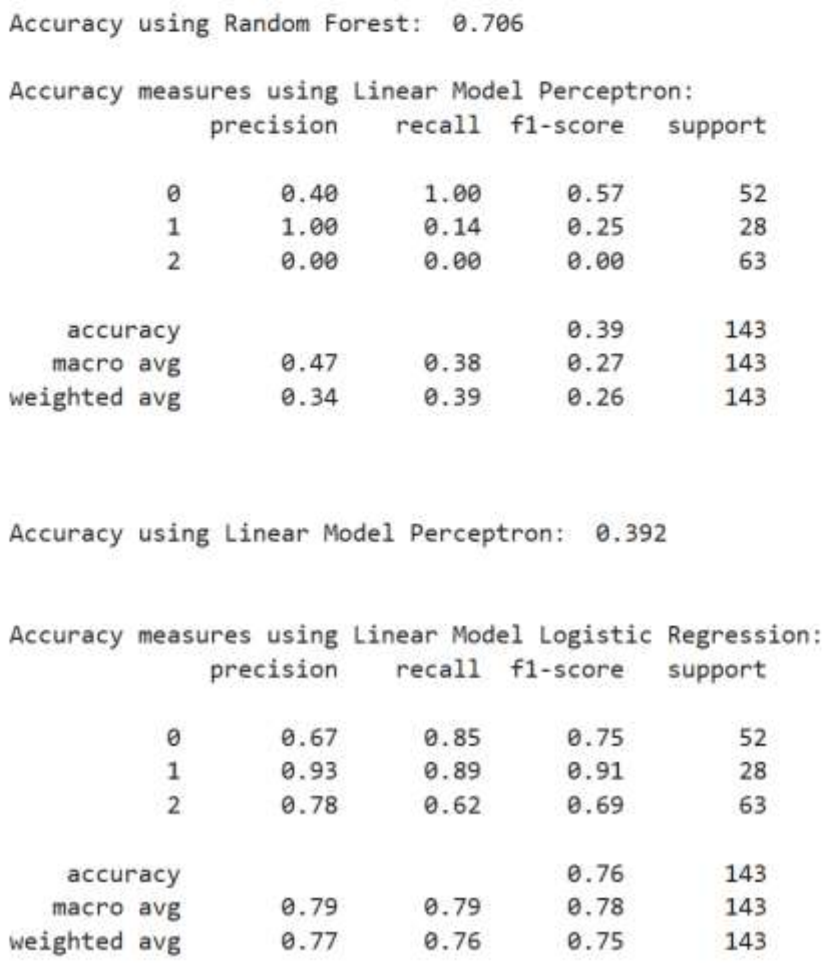
```
Enter Choice:  10
Exiting..
```



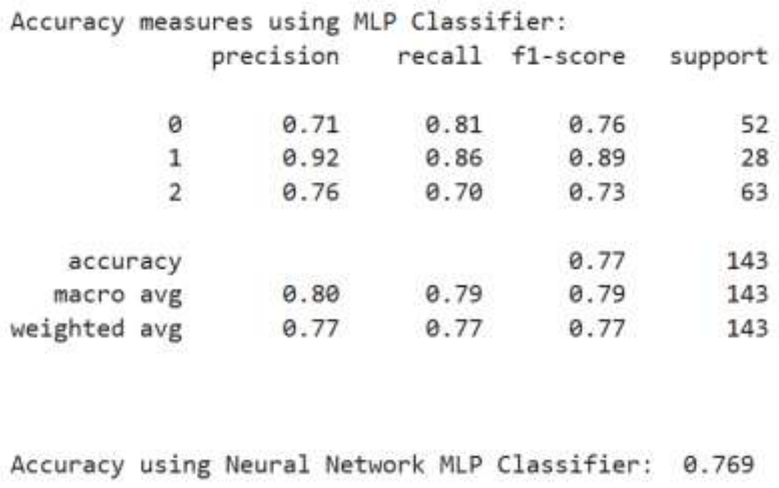
```
Accuracy measures using Decision Tree:
              precision    recall  f1-score   support

           0       0.60      0.65      0.62        52
           1       0.89      0.86      0.87        28
           2       0.63      0.59      0.61        63

    accuracy                           0.66       143
   macro avg       0.70      0.70      0.70       143
weighted avg       0.67      0.66      0.66       143
```



```
Accuracy using Decision Tree:  0.664

Accuracy Measures for Random Forest Classifier:


              precision    recall  f1-score   support

           0       0.62      0.69      0.65        52
           1       0.93      0.93      0.93        28
           2       0.68      0.62      0.65        63

    accuracy                           0.71       143
   macro avg       0.74      0.75      0.74       143
weighted avg       0.71      0.71      0.71       143
```

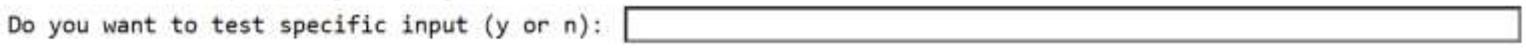


```
Accuracy using Random Forest:  0.706

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
```

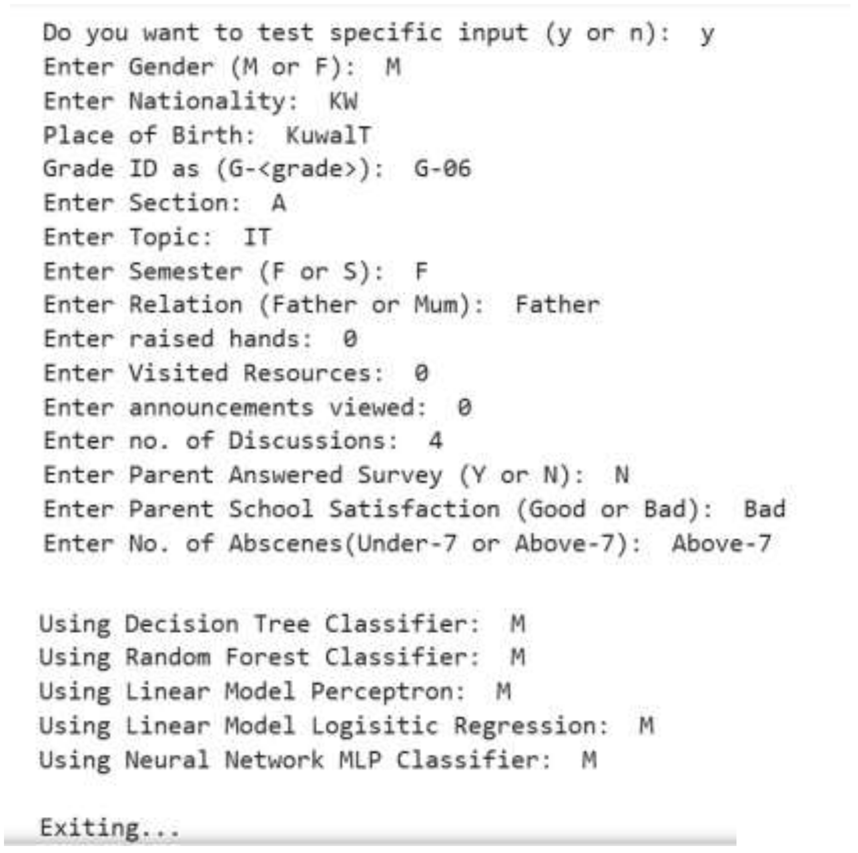


```
Accuracy measures using MLP Classifier:
              precision    recall  f1-score   support

           0       0.71      0.81      0.76        52
           1       0.92      0.86      0.89        28
           2       0.76      0.70      0.73        63

    accuracy                           0.77       143
   macro avg       0.80      0.79      0.79       143
weighted avg       0.77      0.77      0.77       143


Accuracy using Neural Network MLP Classifier:  0.769
```

b) **Test specific input**



```
Do you want to test specific input (y or n):
```

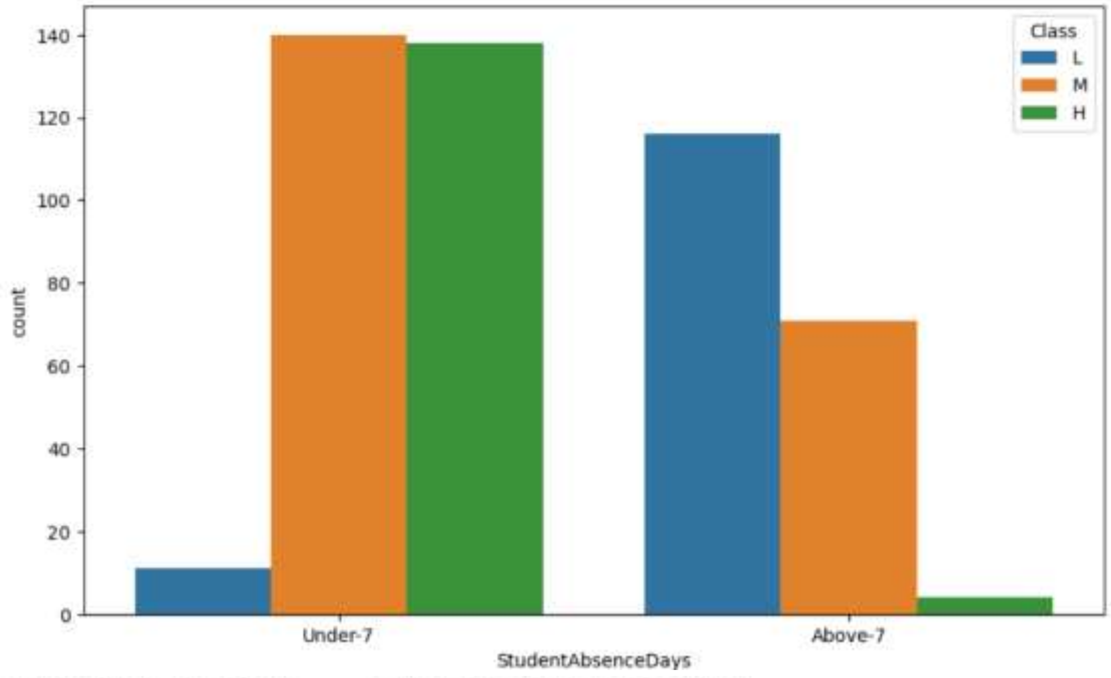


```
Do you want to test specific input (y or n):  y
Enter Gender (M or F):  M
Enter Nationality:  KW
Place of Birth:  KuwaIT
Grade ID as (G-<grade>):  G-06
Enter Section:  A
Enter Topic:  IT
Enter Semester (F or S):  F
Enter Relation (Father or Mum):  Father
Enter raised hands:  0
Enter Visited Resources:  0
Enter announcements viewed:  0
Enter no. of Discussions:  4
Enter Parent Answered Survey (Y or N):  N
Enter Parent School Satisfaction (Good or Bad):  Bad
Enter No. of Abscenes(Under-7 or Above-7):  Above-7

Using Decision Tree Classifier:  M
Using Random Forest Classifier:  M
Using Linear Model Perceptron:  M
Using Linear Model Logisitic Regression:  M
Using Neural Network MLP Classifier:  M

Exiting...
```

Diagrams:

```
Enter Choice:  9
Loading Graph..
```

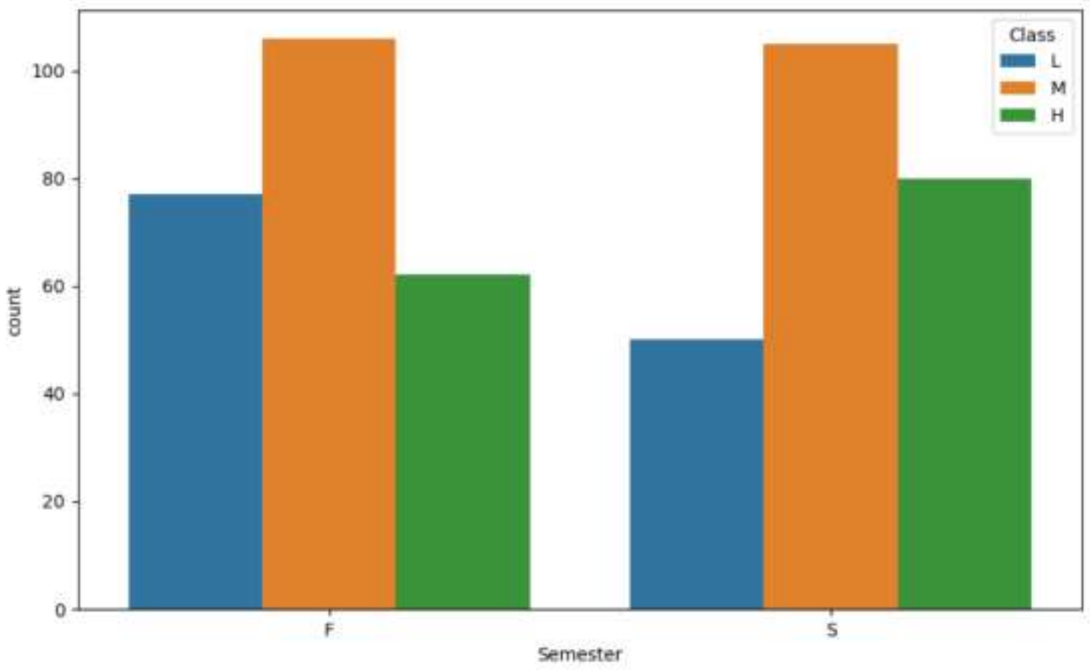
Marks Class Absent Days-wise Graph



```
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
```

```
Enter Choice:  2
Loading Graph....
```

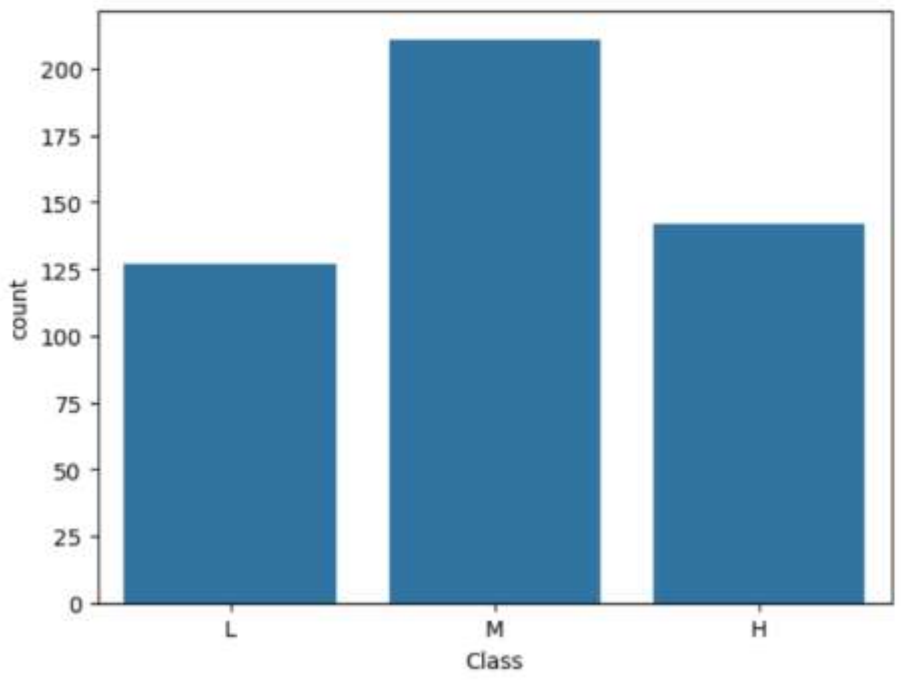
Marks Class Semester-wise Graph



```
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
```

```
Enter Choice:  1
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
9.Marks Class Absent Days-wise  10.No Graph
```