A

Assesment Report

on

"Student Performance Prediction"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Name of discipline

CSE(AIML)

By

Sneha Sahu (202401100400188)

Under the supervision of

"Abhishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2025

INTRODUCTION

Problem Statement:

The aim of this project is to develop a machine learning model that can predict whether a student will **pass** or **fail** based on factors such as **attendance**, **previous academic scores**, and **study habits**.

Context:

In education, predicting a student's academic success is crucial for timely interventions and improving learning outcomes. By understanding the factors that influence student performance, educators can identify at-risk students early and provide support to help them succeed.

Key Features:

- **Attendance**: Regular attendance is often correlated with better academic performance, as students who attend classes regularly tend to grasp the material more effectively.
- **Previous Scores**: Past academic performance, such as previous exam scores or GPA, serves as a strong indicator of future success.
- **Study Habits**: The amount of time dedicated to studying, and whether students engage in consistent study routines, impacts their ability to retain information and perform well in exams.

Objective:

The goal is to use machine learning techniques to classify students into two categories: **pass** or **fail**, based on these factors. By doing so, educational institutions can take proactive measures to support students who are at risk of failing, improving overall academic outcomes.

Importance:

This prediction model can assist in early identification of students who may need additional academic support, enabling educators to intervene with tailored resources, tutoring, or counseling. It also helps schools allocate resources more effectively, ensuring that at-risk students receive the help they need to succeed.

METHODOLOGY

To build an accurate and reliable model for predicting student performance, the following step-by-step methodology is followed:

---

1. Data Collection

The dataset contains student information, including:

- **Attendance**
- **Previous academic scores (GPA)**
- **Study habits (e.g., study hours per week)**
- **Parental support**
- **Participation in extracurricular activities**
- **Grade class (used to label pass/fail)**

---

2. Data Preprocessing

- **Handling Missing Values**: Checked and handled any null or missing entries in the dataset.
- **Feature Selection**: Selected key features that influence academic performance.
- **Label Encoding**: Converted `GradeClass` into binary labels: `Pass = 0`, `Fail = 1`.
- **Feature Scaling**: Standardized numerical features using `StandardScaler` for better model performance.

---

3. Data Splitting

- The dataset is split into **training** and **testing** sets using an 80:20 ratio to evaluate the model on unseen data.

---

4. Model Building

- A **Random Forest Classifier** is used due to its ability to handle mixed feature types and reduce overfitting.

- The model is trained on the scaled training data to learn patterns that distinguish pass/fail classes.

---

5. Model Evaluation

- Evaluated the model on test data using

✓ **Accuracy Score**
✓ **Classification Report** (Precision, Recall, F1-score)
✓ **Confusion Matrix** (visualized using a heatmap)

- Analyzed **feature importance** to identify which factors most influence predictions.

---

6. Prediction

- Created a sample student profile and passed it through the model to predict whether the student would pass or fail.

---

7. Visualization

- Plotted:

✓ Target distribution (`Pass` vs `Fail`)
✓ Confusion Matrix
✓ Feature importance chart using Seaborn for better interpretation

CODE

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate

# Load the dataset
data = pd.read_csv('/8. Student Performance Prediction.csv')

# Data Exploration
print("\n" + "="*40)
print("Dataset Exploration")
print("="*40)
print(f"\nDataset Shape: {data.shape}")
print("\nFirst Few Rows:")
print(data.head())
```

```
print("\nColumn Information:")
print(data.info())
print("\nSummary Statistics:")
print(data.describe())

# Check for missing values
print("\nMissing Values in Dataset:")
print(data.isnull().sum())

# Visualize the distribution of the target variable (Pass/Fail)
plt.figure(figsize=(6, 4))
sns.countplot(x='GradeClass', data=data, palette="Set2")
plt.title('Distribution of Pass/Fail Classes', fontsize=16)
plt.xlabel('Grade Class', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()

# Preprocessing
print("\n" + "="*40)
print("Data Preprocessing")
print("="*40)
# Convert GradeClass to binary (0 = pass, 1 = fail)
data['Pass_Fail'] = data['GradeClass'].apply(lambda x: 0 if x <= 2 else 1)
```

```
# Select relevant features
features = ['StudyTimeWeekly', 'Absences', 'GPA', 'ParentalSupport',
'Extracurricular']
X = data[features]
y = data['Pass_Fail']

# Check if any features are missing or invalid
missing_features = [col for col in features if col not in data.columns]
if missing_features:
    print(f"\nWarning: The following features are missing from the dataset:
{missing_features}")
else:
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    # Feature Scaling
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Model Training
    print("\n" + "="*40)
```

```
print("Model Training & Evaluation")
print("="*40)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Model Evaluation
y_pred = model.predict(X_test_scaled)

print(f"\nAccuracy: {accuracy_score(y_test, y_pred):.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm, index=['Pass', 'Fail'], columns=['Pass', 'Fail'])

# Improved confusion matrix plot with annotations
plt.figure(figsize=(6, 4))
sns.heatmap(cm_df, annot=True, fmt='d', cmap="Blues", cbar=False,
linewidths=0.5)
plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Actual', fontsize=12)
plt.xlabel('Predicted', fontsize=12)
```

```
plt.show()

# Feature Importance
feature_importance = pd.DataFrame({
    'Feature': features,
    'Importance': model.feature_importances_
}).sort_values('Importance', ascending=False)

print("\nFeature Importance:")
print(tabulate(feature_importance, headers='keys', tablefmt='pretty',
showindex=False))

# Plot feature importance with more appealing styling
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance,
palette="viridis")
plt.title('Feature Importance', fontsize=16)
plt.xlabel('Importance', fontsize=12)
plt.ylabel('Feature', fontsize=12)
plt.show()

# Example Prediction
print("\n" + "="*40)
```

```python
print("Example Prediction")
print("="*40)
sample_student = np.array([[15, 5, 3.5, 3, 1]])  # StudyTime, Absences, GPA, ParentalSupport, Extracurricular
sample_student_scaled = scaler.transform(sample_student)
prediction = model.predict(sample_student_scaled)

print("\nSample Features (StudyTime, Absences, GPA, ParentalSupport, Extracurricular):")
print(sample_student)
print(f"Predicted Class: {'Fail' if prediction[0] == 1 else 'Pass'}")
```

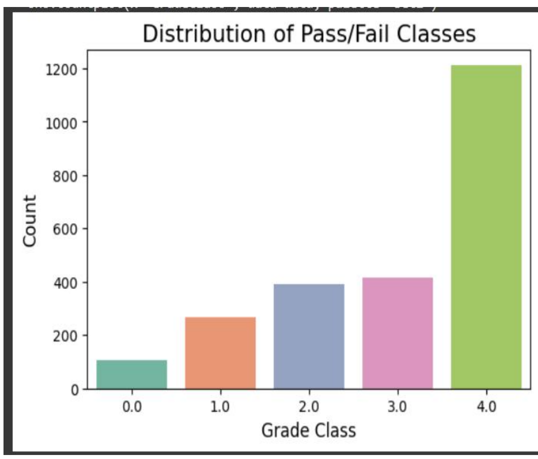
OUTPUT

```
=====================================
Dataset Exploration
=====================================

Dataset Shape: (2392, 15)

First Few Rows:
   StudentID  Age  Gender  Ethnicity  ParentalEducation  StudyTimeWeekly  \
0       1001   17       1          0                  2        19.833723
1       1002   18       0          0                  1        15.408756
2       1003   15       0          2                  3         4.210570
3       1004   17       1          0                  3        10.028829
4       1005   17       1          0                  2         4.672495

   Absences  Tutoring  ParentalSupport  Extracurricular  Sports  Music  \
0         7         1                2                0       0      1
1         0         0                1                0       0      0
2        26         0                2                0       0      0
3        14         0                3                1       0      0
4        17         1                3                0       0      0

   Volunteering       GPA  GradeClass
0             0  2.929196         2.0
1             0  3.042915         1.0
2             0  0.112602         4.0
3             0  2.054218         3.0
4             0  1.288061         4.0
```

```
Column Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2392 entries, 0 to 2391
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   StudentID          2392 non-null   int64
 1   Age                2392 non-null   int64
 2   Gender             2392 non-null   int64
 3   Ethnicity          2392 non-null   int64
 4   ParentalEducation  2392 non-null   int64
 5   StudyTimeWeekly    2392 non-null   float64
 6   Absences           2392 non-null   int64
 7   Tutoring           2392 non-null   int64
 8   ParentalSupport    2392 non-null   int64
 9   Extracurricular    2392 non-null   int64
 10  Sports             2392 non-null   int64
 11  Music              2392 non-null   int64
 12  Volunteering       2392 non-null   int64
 13  GPA                2392 non-null   float64
 14  GradeClass         2392 non-null   float64
dtypes: float64(3), int64(12)
memory usage: 280.4 KB
None
```

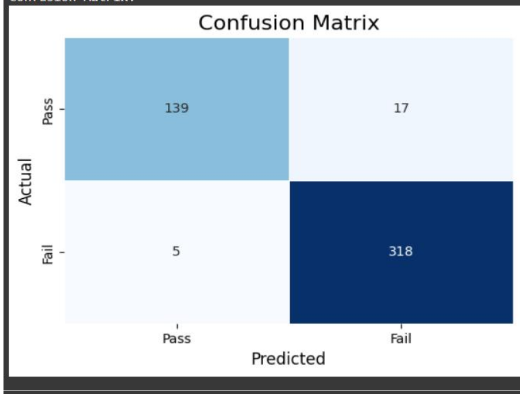
```
Summary Statistics:
         StudentID          Age       Gender    Ethnicity  ParentalEducation  \
count  2392.000000  2392.000000  2392.000000  2392.000000        2392.000000
mean   2196.500000    16.468645     0.510870     0.877508           1.746237
std     690.655244     1.123798     0.499986     1.028476           1.000411
min    1001.000000    15.000000     0.000000     0.000000           0.000000
25%    1598.750000    15.000000     0.000000     0.000000           1.000000
50%    2196.500000    16.000000     1.000000     0.000000           2.000000
75%    2794.250000    17.000000     1.000000     2.000000           2.000000
max    3392.000000    18.000000     1.000000     3.000000           4.000000

       StudyTimeWeekly     Absences     Tutoring  ParentalSupport  \
count      2392.000000  2392.000000  2392.000000      2392.000000
mean          9.771992    14.541388     0.301421         2.122074
std           5.652774     8.467417     0.458971         1.122813
min           0.001057     0.000000     0.000000         0.000000
25%           5.043079     7.000000     0.000000         1.000000
50%           9.705363    15.000000     0.000000         2.000000
75%          14.408410    22.000000     1.000000         3.000000
max          19.978094    29.000000     1.000000         4.000000

       Extracurricular       Sports        Music  Volunteering          GPA  \
count      2392.000000  2392.000000  2392.000000   2392.000000  2392.000000
mean          0.383361     0.303512     0.196906      0.157191     1.906186
std           0.486307     0.459870     0.397744      0.364057     0.915156
min           0.000000     0.000000     0.000000      0.000000     0.000000
25%           0.000000     0.000000     0.000000      0.000000     1.174803
50%           0.000000     0.000000     0.000000      0.000000     1.893393
75%           1.000000     1.000000     0.000000      0.000000     2.622216
max           1.000000     1.000000     1.000000      1.000000     4.000000

        GradeClass
count  2392.000000
mean      2.983696
std       1.233908
min       0.000000
25%       2.000000
50%       4.000000
75%       4.000000
```

```
Missing Values in Dataset:
StudentID            0
Age                  0
Gender               0
Ethnicity            0
ParentalEducation    0
StudyTimeWeekly      0
Absences             0
Tutoring             0
ParentalSupport      0
Extracurricular      0
Sports               0
Music                0
Volunteering         0
GPA                  0
GradeClass           0
dtype: int64
<ipython-input-9-1086bdddd192>:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='GradeClass', data=data, palette="Set2")
```

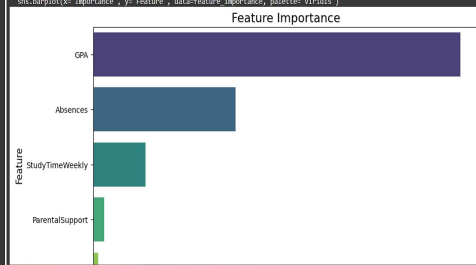



```
========================================
Data Preprocessing
========================================

========================================
Model Training & Evaluation
========================================

Accuracy: 0.95

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       156
           1       0.95      0.98      0.97       323

    accuracy                           0.95       479
   macro avg       0.96      0.94      0.95       479
weighted avg       0.95      0.95      0.95       479
```

```
Confusion Matrix:
```




```
Feature Importance:
+----------------+---------------------+
|    Feature     |     Importance      |
+----------------+---------------------+
|      GPA       |  0.6346303890927906 |
|    Absences    | 0.24634003828540066 |
| StudyTimeWeekly | 0.09087609434533615 |
| ParentalSupport | 0.019432643852083018 |
| Extracurricular | 0.008720834424389607 |
+----------------+---------------------+
<ipython-input-9-1086bdddd192>:100: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Importance', y='Feature', data=feature_importance, palette="viridis")
```




```
========================================
Example Prediction
========================================

Sample Features (StudyTime, Absences, GPA, ParentalSupport, Extracurricular):
[[15.   5.   3.5  3.   1. ]]
Predicted Class: Pass
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

REFERENCE

Student Performance Dataset

The dataset used for this project is sourced from the **UCI Machine Learning Repository**, a well-known collection of datasets for machine learning research. The dataset includes student performance data based on their study time, absences, GPA, and other factors that may contribute to whether they pass or fail.

UCI Machine Learning Repository. (n.d.). *Student Performance Dataset*. Retrieved from https://archive.ics.uci.edu/ml/datasets/Student+Performance

- **Libraries and Tools**

  Several Python libraries were used for data manipulation, modeling, and visualization in this project:

✓ **Pandas** (McKinney, 2011) was used for data manipulation and preprocessing.

✓ **NumPy** (Harris et al., 2020) helped with numerical computations.

✓ **Scikit-learn** (Pedregosa et al., 2011) was employed for machine learning model training and evaluation, including the **Random Forest Classifier** used for classification.

✓ **Matplotlib** (Hunter, 2007) and **Seaborn** (Waskom et al., 2020) were used for plotting visualizations, including the distribution of grades and confusion matrix.

- **Machine Learning Algorithm**

  The **Random Forest Classifier** (Breiman, 2001) was chosen due to its effectiveness in handling both classification tasks and large datasets, offering insights into feature importance and model performance.
- **General Machine Learning References**

  For a comprehensive understanding of machine learning techniques and Python tools, we referred to **Alpaydin's Introduction to Machine Learning** (2020) and **VanderPlas' Python Data Science Handbook** (2016).

In-Text Citation Example:

> "The **Student Performance Dataset** from the **UCI Machine Learning Repository** (UCI, n.d.) was utilized to analyze factors contributing to student success or failure.
>
> "To classify student outcomes, we employed the **Random Forest Classifier** (Breiman, 2001), which is known for its robust performance on structured data."