www.kiet.edu
Delhi-NCR, Ghaziabad

KIET
GROUP OF INSTITUTIONS
Connecting Life with Learning

Assessment Report

On

“STUDENT PERFORMANCE PREDICTION”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY
DEGREE

session 2024-25
in

Bachelor of Technology (CSE - AI)
by

SHOURYA GARG (202401100300238)

Under the supervision of

MR. ABHISHEK SHUKLA

KIET GROUP OF INSTITUTIONS, GHAZIABAD

Affiliated to

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW
(FORMERLY UPTU)

18 APRIL, 2025

TABLE OF CONTENTS

INTRODUCTION

The problem at hand involves predicting student performance based on various factors such as attendance, previous scores, and study habits. In the current education landscape, understanding and predicting student outcomes is crucial for improving teaching methods, personalizing learning experiences, and providing timely support to students who may be struggling.

Predicting whether a student will pass or fail based on certain features can help educational institutions intervene early, offering additional resources to those at risk of failing and ensuring that high-performing students continue to thrive. This task is an excellent application of machine learning, specifically classification algorithms, which can analyze historical data and generate predictions based on patterns identified in the data.

In this project, we focus on using data such as:

- **Attendance:** The amount of time the student spends attending classes or lectures.
- **Previous Scores:** The student's past academic performance.
- **Study Habits:** How regularly and effectively the student studies.

These features are expected to have a significant influence on the student's overall performance. By leveraging machine learning techniques, specifically classification models, we aim to build a predictive model that can determine whether a student will pass or fail, based on these features.

METHODOLOGY

Step 1: Data Collection
The data used for this project is collected from a CSV file containing information about students, including their attendance, previous scores, study habits, and performance outcomes (pass or fail). Each student is represented by a row, and each column contains features or attributes relevant to their academic behavior.

Step 2: Data Inspection and Cleaning

- Before proceeding with model training, we inspect the dataset to ensure the data is clean, relevant, and usable:
- We check for missing or null values and handle them accordingly (either by removing rows or filling in missing values).
- We check for any inconsistent data entries or outliers that may distort model predictions.
- We also examine the columns to understand which features are most relevant for predicting the pass/fail outcome.

Step 3: Feature Engineering
Based on the dataset, we select a set of features that are believed to influence the student's performance:
Attendance: How often the student attends classes.
Previous Scores: The student's prior academic performance or grades.
Study Habits: Includes factors like the amount of time the student spends on weekly study sessions, and whether they participate in extracurricular activities.
The target variable is Pass/Fail, derived from the GradeClass column, where values 0, 1, 2 represent students who passed, and values 3, 4 represent students who failed.

Step 4: Data Preprocessing
This step involves the following:
Encoding: The target variable (Pass/Fail) is converted to numerical values (0 for Fail and 1 for Pass) for use in machine learning models.
Feature Scaling: Feature values are standardized using StandardScaler to bring all features onto a common scale, ensuring that no feature disproportionately influences the model due to its scale.
Splitting Data: The dataset is divided into two parts: a training set (80% of the data) to train the model, and a test set (20% of the data) to evaluate the model's performance.

Step 5: Model Selection
For this task, we use a Random Forest Classifier, which is an ensemble learning method known for its robustness and ability to handle both regression and classification problems. The Random Forest model works by constructing multiple decision trees during training and outputting the class that is the majority vote of all the trees. This approach reduces overfitting and improves model accuracy.

Step 6: Model Training
The model is trained on the training set, learning the relationships between the features (attendance, previous scores, study habits) and the target variable (Pass/Fail). We configure the Random Forest classifier and fit it to the training data to create a predictive model.

Step 7: Model Prediction
Once the model is trained, we use it to predict the Pass/Fail status for the students in the test set (data not seen by the model during training). The model generates predictions based on the learned relationships between features and target labels.

Step 8: Model Evaluation
To evaluate the performance of the trained model, we use the following metrics:
Accuracy Score: The proportion of correctly predicted labels (Pass/Fail) to the total number of predictions made.
Classification Report: Provides detailed performance metrics like precision, recall, and F1-score for each class (Pass/Fail).
Confusion Matrix: A matrix that shows the true positives, true negatives, false positives, and false negatives, helping us understand the performance of the classification model in detail.

Step 9: Result Presentation
The results are then presented in a neat table format, showing the first 20 students with their predicted Pass/Fail status. This allows for easy interpretation and visualization of the model's output.

Step 10: Saving Results
The final predictions (Pass/Fail) for all students are saved to a CSV file, which can be later used for analysis, reporting, or further modeling.

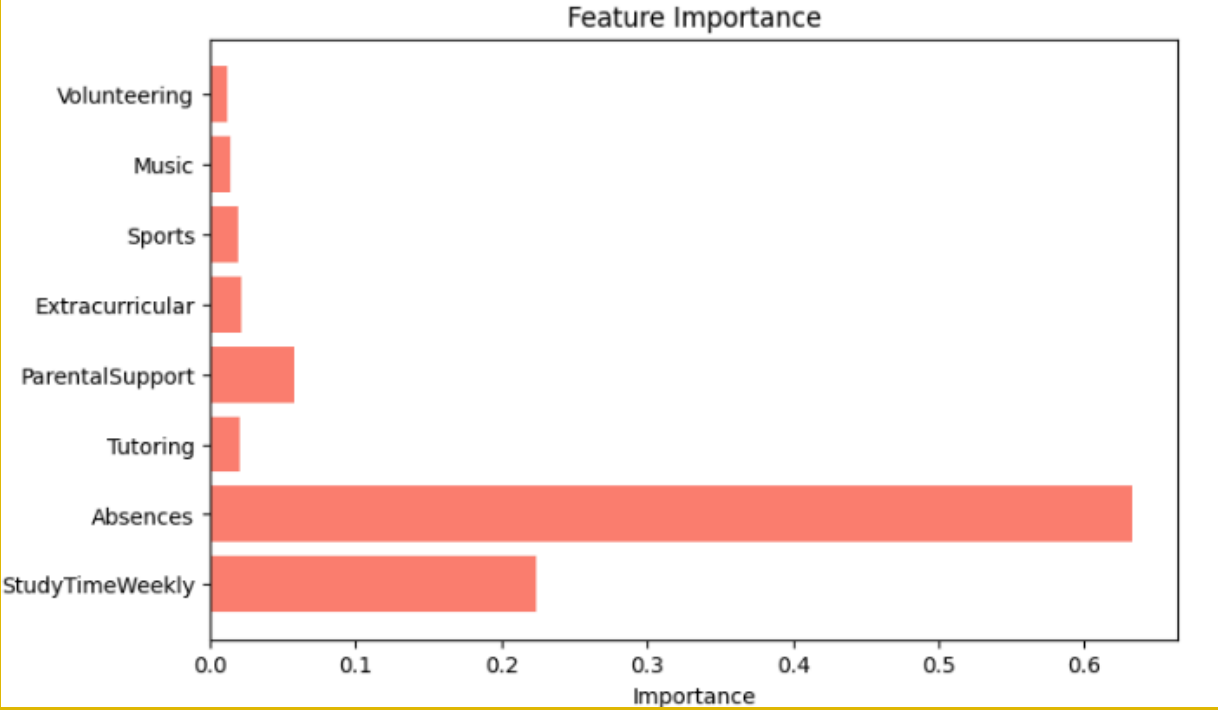
CODE

```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix

# Step 2: Mount Drive to load the dataset
from google.colab import drive
drive.mount('/content/drive')

# Step 3: Load the Dataset
file_path = "/content/drive/MyDrive/8. Student Performance Prediction.csv"
df = pd.read_csv(file_path)

# Step 4: Show columns and first few rows to inspect the dataset
print("Columns in the dataset:", df.columns.tolist())
print("\nFirst 5 rows of the dataset:\n", df.head())

# Step 5: Create a Pass/Fail column based on GradeClass
df['Pass/Fail'] = df['GradeClass'].apply(lambda x: 'Pass' if x in [0, 1, 2] else 'Fail')

# Step 6: Select relevant features for prediction
features = ['StudyTimeWeekly', 'Absences', 'Tutoring', 'ParentalSupport',
        'Extracurricular', 'Sports', 'Music', 'Volunteering']
X = df[features]  # Features (inputs)
y = df['Pass/Fail']  # Target variable

# Step 7: Encode target variable (Pass/Fail) to numerical values
y_encoded = y.map({'Pass': 1, 'Fail': 0})
```

```
# Step 8: Scale the features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 9: Split the data into training and testing sets (80% training, 20%
testing)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded,
test_size=0.2, random_state=42)

# Step 10: Train the Random Forest Model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 11: Make predictions using the test set
y_pred = model.predict(X_test)

# Step 12: Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy Score:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Step 13: Show predictions for the first 20 students (Pass/Fail predictions)
results_df = pd.DataFrame({
   'Student_ID': df['StudentID'][:20].values,  # Assuming there's a StudentID
column
   'Predicted': y_pred[:20],  # Only show predictions for the first 20 students
   'Predicted_Label': pd.Series(y_pred[:20]).map({1: 'Pass', 0: 'Fail'})
})

# Step 14: Organize the output into a table for easy viewing
print("\nPass/Fail Predictions for the first 20 students:")

# Display the results as a neat table
print("\n", results_df.to_string(index=False))

# Step 15: Save the predictions to a CSV file
full_results_path = "/content/drive/MyDrive/student_predictions.csv"
results_df.to_csv(full_results_path, index=False)
print("\nPredictions saved to CSV:", full_results_path)
```

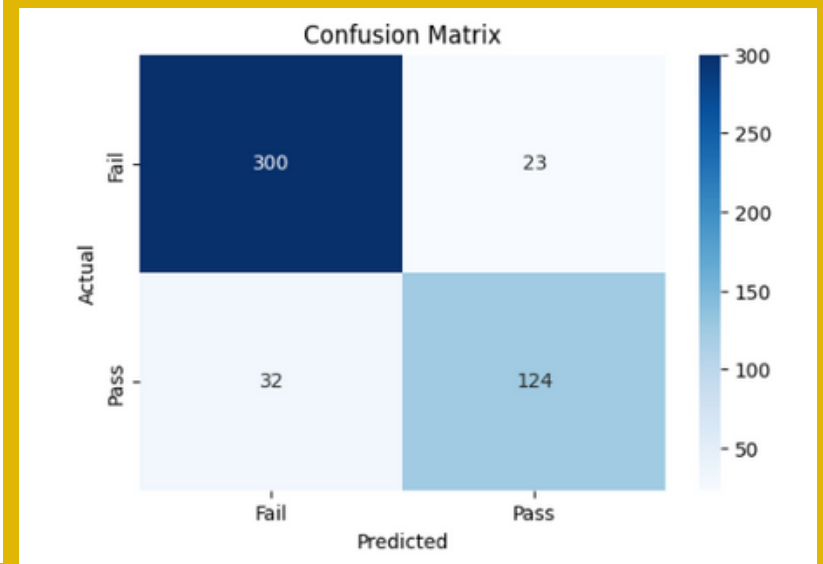
```
# Step 16: Visualizations
# Accuracy Score Bar Plot
plt.figure(figsize=(6, 4))
plt.bar(['Accuracy'], [accuracy], color='skyblue')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.show()

# Confusion Matrix Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Fail',
'Pass'], yticklabels=['Fail', 'Pass'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
# Feature Importance Bar Chart
feature_importances = model.feature_importances_
plt.figure(figsize=(8, 5))
plt.barh(features, feature_importances, color='salmon')
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.show()
```

SNAPSHOTS OF OUTPUT



Pass/Fail Predictions for the first 20 students:

Student_ID	Predicted	Predicted_Label
1001	0	Fail
1002	1	Pass
1003	0	Fail
1004	1	Pass
1005	0	Fail
1006	0	Fail
1007	0	Fail
1008	0	Fail
1009	0	Fail
1010	0	Fail
1011	1	Pass
1012	0	Fail
1013	1	Pass
1014	0	Fail
1015	0	Fail
1016	0	Fail
1017	0	Fail
1018	0	Fail
1019	0	Fail
1020	0	Fail



REFERENCES

www.canva.com

www.wikipedia.com

www.google.com