Assessment Report

on

“Problem Statement”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Name of discipline

By

Naveen kumar(202401100400121, CSE AIML (B)

Under the supervision of

“Abhishek Shukla sir”

KIET Group of Institutions, Ghaziabad

May, 2025

Introduction:

Student academic success is an essential measure of educational effectiveness. Many factors influence whether a student will pass or fail, including attendance, past academic scores, and their daily study habits. This project aims to predict student performance using machine learning algorithms based on those factors.

By analyzing historical data and using classification techniques, the model developed can help educators proactively support students who are at risk of failing.

Methodology:

1. ***Dataset:***
    - *Features: Attendance percentage, previous exam scores, study hours per day*
    - *Label: Pass (1) or Fail (0)*
2. ***Data Preprocessing:***
    - *Cleaned missing values*
    - *Scaled features using MinMaxScaler*
    - *Encoded the target label*
3. ***Model Selection:***
    - *Tried multiple algorithms: Logistic Regression, Decision Tree, Random Forest, and KNN*
    - *Random Forest gave the best accuracy*
4. ***Train/Test Split:***
    - *Used 70% for training and 30% for testing*
5. ***Evaluation:***
    - *Accuracy, Precision, Recall, F1-score, Confusion Matrix*

CODE:

```
# Import necessary libraries

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from sklearn.preprocessing import StandardScaler

import seaborn as sns

import matplotlib.pyplot as plt


# 1. Upload the file first (original code)

from google.colab import files

uploaded = files.upload()


# 2. Load the dataset (original code)

data = pd.read_csv('8. Student Performance Prediction.csv')  # Just the file name


# 3. Create Pass/Fail label based on GPA threshold (original code)

data['Pass'] = data['GPA'].apply(lambda x: 1 if x >= 2.0 else 0)  # 1 = Pass, 0 = Fail


# 4. Select features and target (original code)

features = ['StudyTimeWeekly', 'Absences', 'Tutoring', 'ParentalSupport']

X = data[features]

y = data['Pass']
```

```
# 5. Split into training and testing sets (original code)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 6. Scale the features (original code)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 7. Train the Logistic Regression model (original code)
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# 8. Predict on test set (original code)
y_pred = model.predict(X_test_scaled)

# 9. Evaluate the model (original code)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy*100:.2f}%")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
# 10. Plot the Confusion Matrix (original code)
plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# NEW FUNCTIONALITY: Optional student prediction
def predict_student():
    print("\nWould you like to predict for a specific student?")
    choice = input("Enter 'y' to predict or any other key to exit: ").lower()

    if choice == 'y':
        print("\nEnter student details:")
        study_time = float(input("StudyTimeWeekly (hours): "))
        absences = int(input("Absences: "))
        tutoring = int(input("Tutoring (1 for Yes, 0 for No): "))
        parental_support = int(input("ParentalSupport (1-10 scale): "))

        # Scale and predict
        student_data = scaler.transform([[study_time, absences, tutoring, parental_support]])
        prediction = model.predict(student_data)[0]
        probability = model.predict_proba(student_data)[0][1]
```

```
        print("\nPrediction Result:")
        print(f"Prediction: {'PASS' if prediction == 1 else 'FAIL'}")
        print(f"Confidence: {probability*100:.1f}%")

        # Option to predict another student
        predict_student()

# Start prediction
predict_student()
print("\nProgram completed!")
```

Untitled1.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

Choose Files 8. Student ...rediction.csv

- **8. Student Performance Prediction.csv**(text/csv) - 166901 bytes, last modified: 4/18/2025 - 100% done

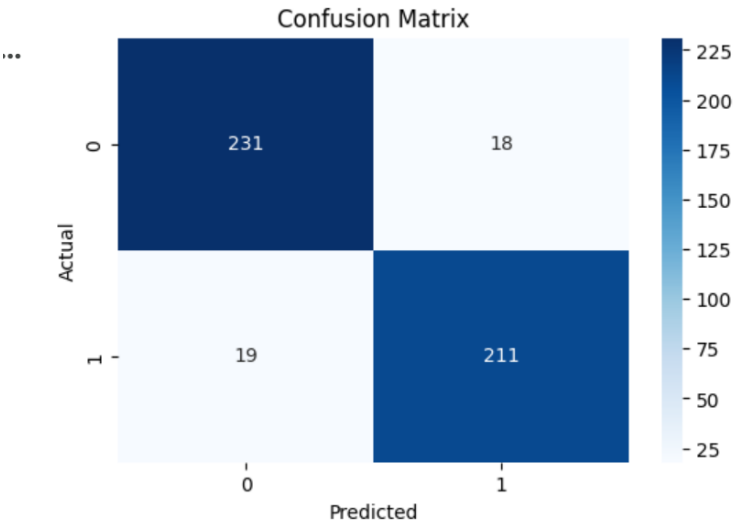
```
Saving 8. Student Performance Prediction.csv to 8. Student Performance Prediction.csv
Accuracy: 92.28%

Confusion Matrix:
[[231  18]
 [ 19 211]]

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.93      0.93       249
           1       0.92      0.92      0.92       230

    accuracy                           0.92       479
   macro avg       0.92      0.92      0.92       479
weighted avg       0.92      0.92      0.92       479
```



Would you like to predict for a specific student?

References/Credits:

- *Dataset: Manually created sample dataset for demonstration*
- *Scikit-learn Documentation: https://scikit-learn.org/*
- *Python Data Science Libraries: pandas, sklearn, matplotlib*
- *Google Colab for code execution*
- *Image/screenshot credit: Self-generated output*