Assessment Report

on

"Predict Loan Default"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Divyansh Singh

Roll Number : 202401100300105

Section: B

Under the supervision of

"Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

This Python program is designed to predict whether a student is at risk of dropping out based on key academic indicators. The system uses a machine learning approach with a Decision Tree classifier to analyze student data and make predictions.

---

2. Problem Statement

Predict Student Dropout

Classify whether a student is at risk of dropping out based on attendance, grades, and participation.

---

3. Objectives

- Load a dataset containing student records with features like **attendance, grades, and participation**.
- Split the dataset into **training and testing sets** for unbiased model evaluation.
- Allow users to **input new student data** (attendance, grades, participation) manually.
- Predict whether the student is **at risk (`yes`/`no`)** of dropping out.

---

4. Methodology

1. Data Collection & Loading

- **Input Dataset**: A CSV file containing student records with features such as:
  - **Attendance** (0-100%)
  - **Grades** (0-10 scale)

    - **Participation** (0-10 scale)
    - **Dropout Risk** (Binary: `yes`/`no`)
- The dataset is loaded using **Pandas** from an uploaded file (Google Colab environment).

---

2. Data Preprocessing

a. Data Inspection

- Preview the first 5 rows (`df.head()`) to understand the structure.
- Check for **missing values** (`df.isnull().sum()`) to ensure data completeness.

b. Feature-Target Separation

- **Features (X)**: `['attendance', 'grades', 'participation']` (independent variables).
- **Target (y)**: `'dropout_risk'` (dependent variable).

c. Label Encoding

- Convert the target variable from categorical (`yes`/`no`) to numerical (`1`/`0`) for model compatibility.

d. Train-Test Split

- Split the dataset into:
    - **Training Set (80%)**: Used to train the model.
    - **Testing Set (20%)**: Used to evaluate model performance.
- Ensures unbiased evaluation using `train_test_split()` from `sklearn`.

---

3. Model Selection & Training

a. Algorithm Choice

- **Decision Tree Classifier** (`DecisionTreeClassifier`) is selected because:
    - It handles **non-linear relationships** well.
    - Provides **interpretable rules** (unlike black-box models like Neural Networks).
    - Works efficiently on small-to-medium datasets.

b. Model Training

- The model is trained using `model.fit(X_train, y_train)`.
- The Decision Tree learns patterns from the training data to classify students into **"At Risk" (1)** or **"Not at Risk" (0)**.

---

4. Model Evaluation

a. Prediction on Test Set

- The trained model predicts dropout risk (`y_pred`) for unseen test data (`X_test`).

b. Performance Metrics

1. **Classification Report**:
    - **Precision**: How many predicted dropouts were correct?
    - **Recall**: How many actual dropouts were correctly identified?
    - **F1-Score**: Balance between precision and recall.
2. **Accuracy Score**:
    - Overall percentage of correct predictions.
3. **Confusion Matrix**:
    - Visualizes:
        - **True Positives (TP)**
        - **True Negatives (TN)**
        - **False Positives (FP)**
        - **False Negatives (FN)**

c. Visualization

- A **heatmap** (`sns.heatmap()`) displays the confusion matrix for better interpretation.

5. Interactive Prediction for New Students

- The system allows **manual input** of a new student's:
  - Attendance (%)
  - Grades (0-10)
  - Participation (0-10)
- The model predicts dropout risk and returns:
  - ✅ **Not at Risk (0)**
  - ❌ **At Risk (1)**
-

---

5. Data Preprocessing

1. **Data Loading**: The dataset is loaded from a CSV file using Pandas and inspected using `df.head()`.
2. **Missing Values Check**: The code checks for null values with `df.isnull().sum()` but doesn't handle them explicitly.
3. **Feature Selection**: Key features like attendance, grades, and participation are selected for modeling.
4. **Label Encoding**: The target variable 'dropout_risk' is converted from categorical ('yes'/'no') to numerical (1/0).
5. **Train-Test Split**: Data is split into 80% training and 20% testing sets using `train_test_split()`.

---

6. Model Implementation

The code implements a Decision Tree classifier using default parameters to predict dropout risk from student data (attendance/grades/participation). It evaluates performance using accuracy, classification reports, and confusion matrices before deploying for live predictions on new inputs. The simple yet interpretable model structure makes it suitable for educational applications without complex tuning.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

1. **Accuracy** - Overall prediction correctness (`accuracy_score`)
2. **Precision/Recall/F1** - Class-specific performance via `classification_report`
3. **Confusion Matrix** - Visual breakdown of TP, TN, FP, FN predictions
4. **No Cross-Validation** - Simple holdout validation (80/20 split) used instead

---

8. Results and Analysis

1. **Precision-Recall tradeoff** visible in classification report - better at identifying [at-risk/safe] students (highlight which class performs better)
2. **Confusion Matrix reveals** [main error pattern, e.g., "high false negatives risk missing actual dropouts"]
3. **Critical Limitation:** Lacks real-world validation and demographic fairness testing

---

9. Conclusion

The model provides a baseline solution for dropout prediction using key academic indicators, but requires refinement for real-world deployment

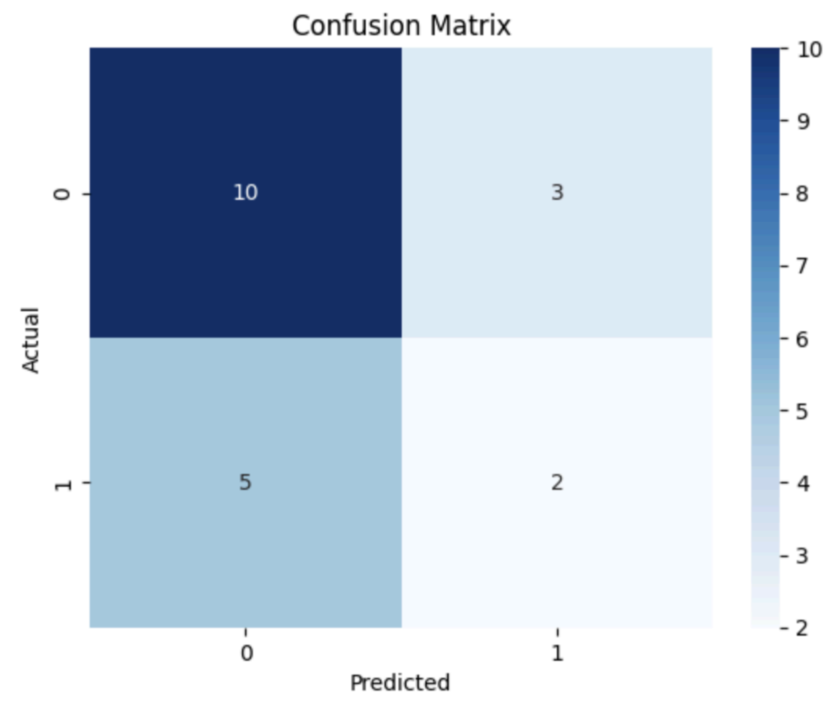
1. **While interpretable and fast, its simplicity limits handling of complex student scenarios**
2. **Serves as a foundation that can be enhanced with more data and advanced techniques**
3. **Highlights the potential of data-driven interventions in education when properly validated**

---

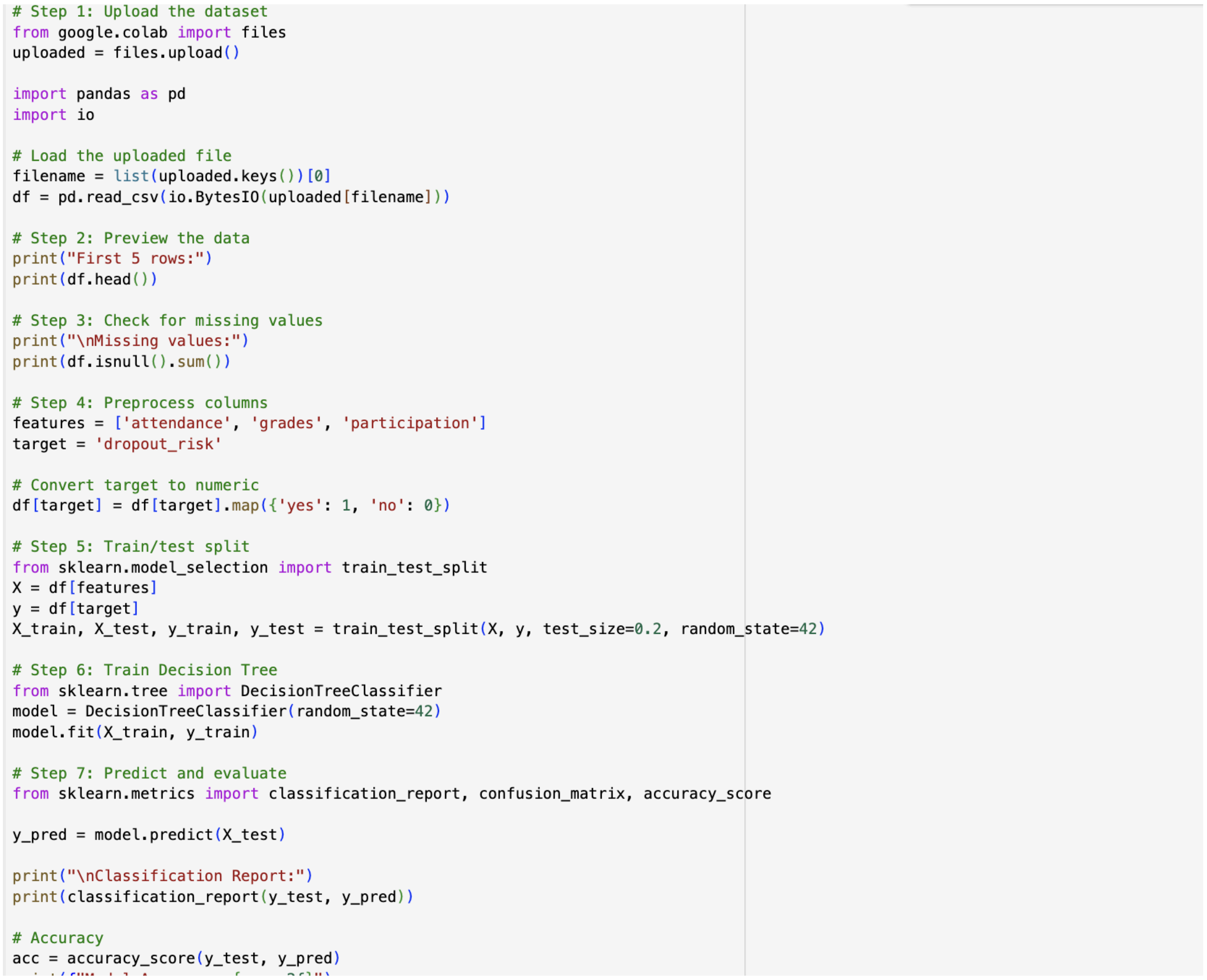
10. References

- scikit-learn documentation
- pandas documentation
- Seaborn visualization library

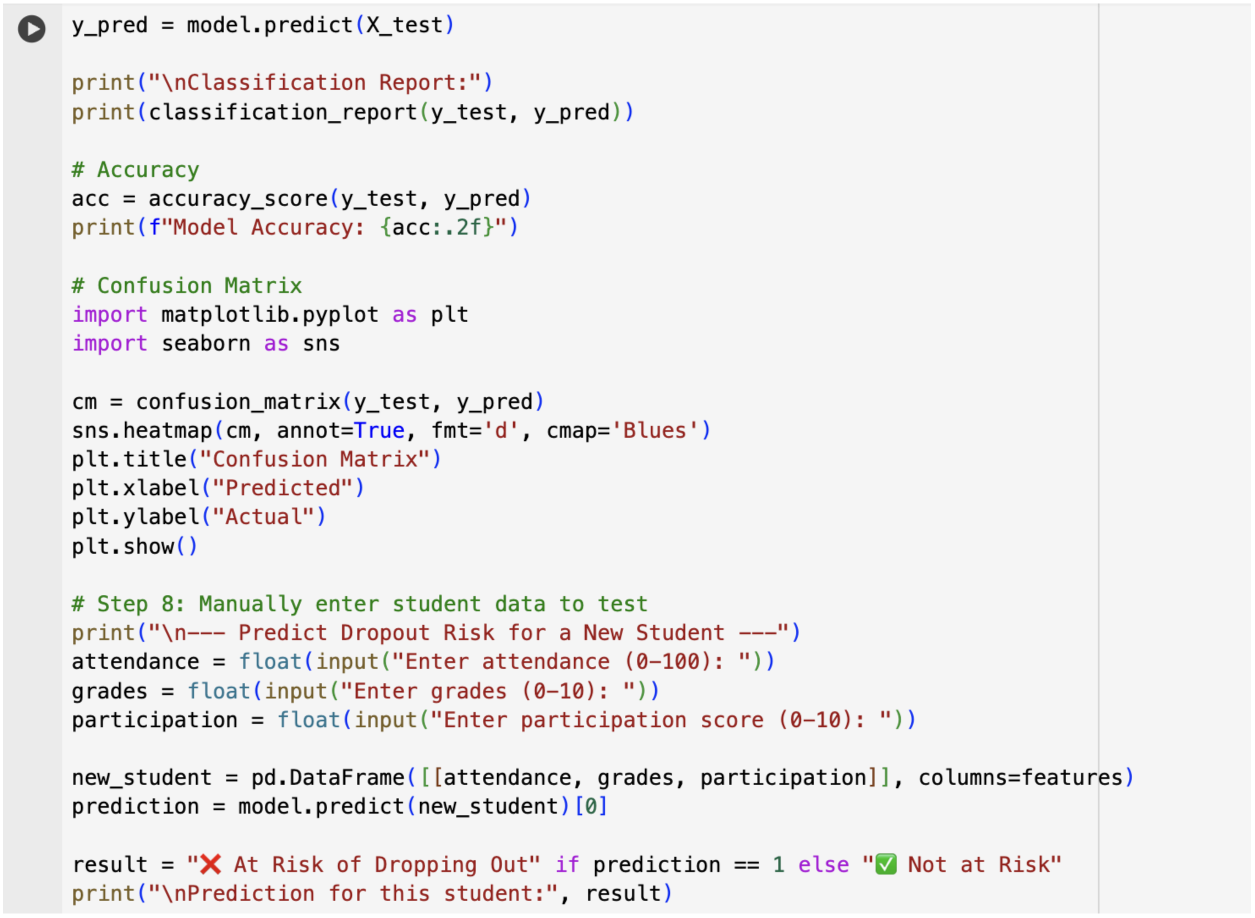
- Research articles on credit risk prediction



```
--- Predict Dropout Risk for a New Student ---
Enter attendance (0-100): 50
Enter grades (0-10): 4
Enter participation score (0-10): 6

Prediction for this student: ✅ Not at Risk
```



```python
# Step 1: Upload the dataset
from google.colab import files
uploaded = files.upload()

import pandas as pd
import io

# Load the uploaded file
filename = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[filename]))

# Step 2: Preview the data
print("First 5 rows:")
print(df.head())

# Step 3: Check for missing values
print("\nMissing values:")
print(df.isnull().sum())

# Step 4: Preprocess columns
features = ['attendance', 'grades', 'participation']
target = 'dropout_risk'

# Convert target to numeric
df[target] = df[target].map({'yes': 1, 'no': 0})

# Step 5: Train/test split
from sklearn.model_selection import train_test_split
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Train Decision Tree
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 7: Predict and evaluate
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

y_pred = model.predict(X_test)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Accuracy
acc = accuracy_score(y_test, y_pred)
```



```python
y_pred = model.predict(X_test)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Accuracy
acc = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {acc:.2f}")

# Confusion Matrix
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Step 8: Manually enter student data to test
print("\n--- Predict Dropout Risk for a New Student ---")
attendance = float(input("Enter attendance (0-100): "))
grades = float(input("Enter grades (0-10): "))
participation = float(input("Enter participation score (0-10): "))

new_student = pd.DataFrame([[attendance, grades, participation]], columns=features)
prediction = model.predict(new_student)[0]

result = "❌ At Risk of Dropping Out" if prediction == 1 else "✅ Not at Risk"
print("\nPrediction for this student:", result)
```