Student Performance Prediction using Decision Tree

A machine learning project based on Student Performance Prediction System

1. Objective

The objective of this project is to predict student academic performance using a Decision Tree classifier based on selected demographic and academic attributes. The model aims to identify students who are at risk of poor academic performance and provide interpretable decision rules that explain the factors influencing student outcomes.

2. Problem Statement

Educational institutions often face challenges in identifying students who may underperform academically. Traditional manual analysis of student records is time-consuming and may be subjective. By applying machine learning techniques, particularly Decision Tree classification, institutions can automate performance prediction and gain transparent insights into how factors such as study time, attendance, parental education, and internet access affect academic success.

3. Dataset Description

The project uses the Student Performance Dataset obtained from the UCI Machine Learning Repository. The dataset contains academic and demographic information of secondary school students.

4. Selected Features (as per problem statement):

- Study time (studytime)
- Attendance (absences)
- Previous grades (G1, G2)
- Parental education (Medu, Fedu)
- Internet access (internet)

Target Variable:

- Performance (derived from final grade G3)
    - 1 → Good Performance (G3 ≥ 10)
    - 0 → Poor Performance (G3 < 10)

5. Methodology

The following steps were followed to build the student performance prediction system:

1. Load the dataset using Pandas
2. Preprocess the data and select required features
3. Handle categorical variables and missing values
4. Split the dataset into training and testing sets
5. Train a Decision Tree classifier using entropy
6. Prune the tree to prevent overfitting
7. Visualize the decision tree structure
8. Evaluate the model using standard performance metrics

6. Algorithm Used: Decision Tree Classifier

A Decision Tree classifier is a supervised machine learning algorithm that splits data into subsets based on feature values. The model uses entropy (information

gain) as the splitting criterion, which aligns with the theoretical concepts covered in the syllabus. Decision Trees are highly interpretable and suitable for classification tasks involving structured data.

7. Tools and Technologies Used

- Python – Programming language
- Pandas – Data handling and preprocessing
- Scikit-learn – Machine learning model and evaluation
- Matplotlib – Visualization of decision tree and confusion matrix

8. Implementation

This section explains the implementation in a step-by-step manner. Each code block is presented along with its purpose for better understanding.

8.1 Import Required Libraries

The necessary Python libraries are imported for data handling, model building, evaluation, and visualization.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

The student performance dataset is loaded from a CSV file using Pandas.

```
data = pd.read_csv("/home/sampad/student/student-mat.csv", sep=';')
print(data.head())
```

8.3 Create Target Variable

The final grade (G3) is converted into a binary target variable called Performance.

```
data['Performance'] = data['G3'].apply(lambda x: 1 if x >= 10 else 0)
```

8.4 Feature Selection

Only the features specified in the problem statement are selected for model training.

```
features = ['studytime', 'absences', 'G1', 'G2', 'Medu', 'Fedu', 'internet']
X = data[features].copy()
y = data['Performance']
X.loc[:, 'internet'] = X['internet'].map({'yes': 1, 'no': 0})
X = X.infer_objects(copy=False)
X.fillna(X.mean(), inplace=True)
```

8.5 Handle Categorical Variables

The categorical feature **internet** is encoded into numerical form.

```
X.loc[:, 'internet'] = X['internet'].map({'yes': 1, 'no': 0})
X = X.infer_objects(copy=False)
```

8.6 Split Dataset into Training and Testing Sets

The dataset is divided into training and testing subsets.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

8.7 Train the Decision Tree Classifier

A Decision Tree classifier is trained using entropy as the splitting criterion. Tree depth is limited to prevent overfitting.

```
dt = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=4,
    random_state=42
)
dt.fit(X_train, y_train)
```

8.8 Model Prediction

Predictions are generated using the trained model.

```
y_pred = dt.predict(X_test)
```

8.9 Model Evaluation

The model is evaluated using Accuracy, Precision, Recall, F1-score, and Confusion Matrix.

```
print("\nAccuracy:", accuracy)
print("\nPrecision:", precision)
print("\nRecall:", recall)
print("\nF1-Score:", f1)
print("\nConfusion Matrix:\n", conf_matrix)
```

8.10 Visualize Confusion Matrix

The confusion matrix is visualized for better understanding of classification performance.

```
disp = ConfusionMatrixDisplay(
    confusion_matrix=confusion_matrix(y_test, y_pred),
    display_labels=['Poor', 'Good']
)

disp.plot(cmap='Blues')
plt.title("Confusion Matrix - Student Performance Prediction")
plt.show()
```

8.11 Visualize Decision Tree Structure

The trained Decision Tree is visualized to interpret classification rules.

```
plt.figure(figsize=(20, 10))
plot_tree(
    dt,
    feature_names=features,
    class_names=['Poor', 'Good'],
    filled=True
)
plt.title("Decision Tree for Student Performance Prediction")
plt.show()
```

9. Code

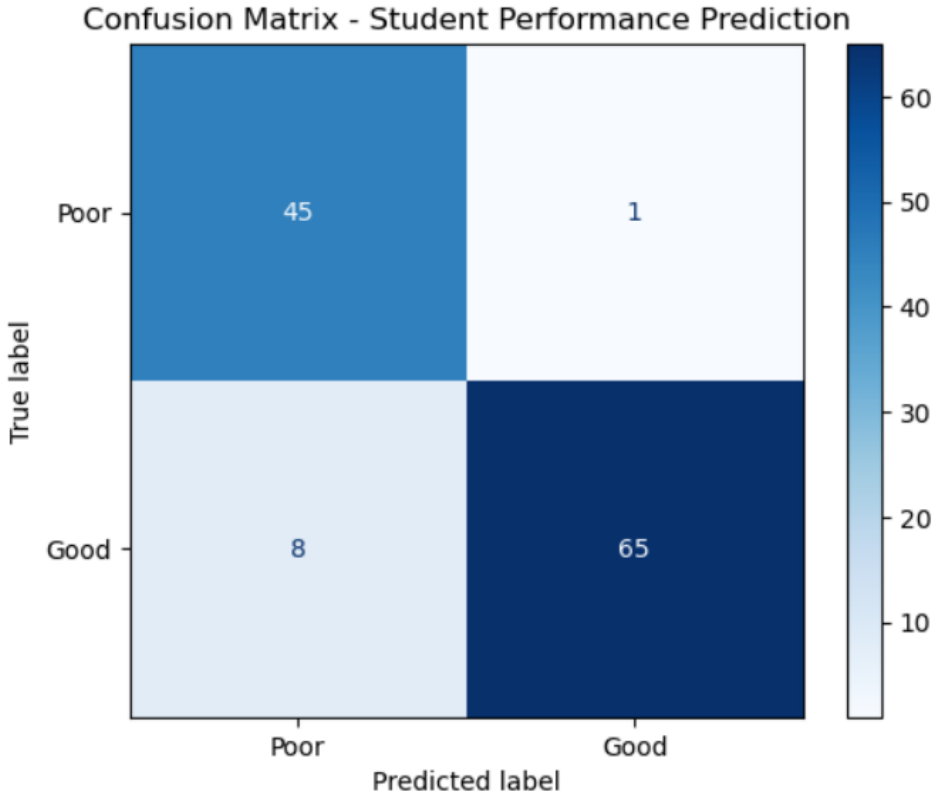
[9]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

data = pd.read_csv("/home/sampad/student/student-mat.csv", sep=';')
print(data.head())

data['Performance'] = data['G3'].apply(lambda x: 1 if x >= 10 else 0)

features = ['studytime', 'absences', 'G1', 'G2', 'Medu', 'Fedu', 'internet']
X = data[features].copy()
y = data['Performance']
X.loc[:, 'internet'] = X['internet'].map({'yes': 1, 'no': 0})
X = X.infer_objects(copy=False)
X.fillna(X.mean(), inplace=True)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

dt = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=4,
    random_state=42
)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

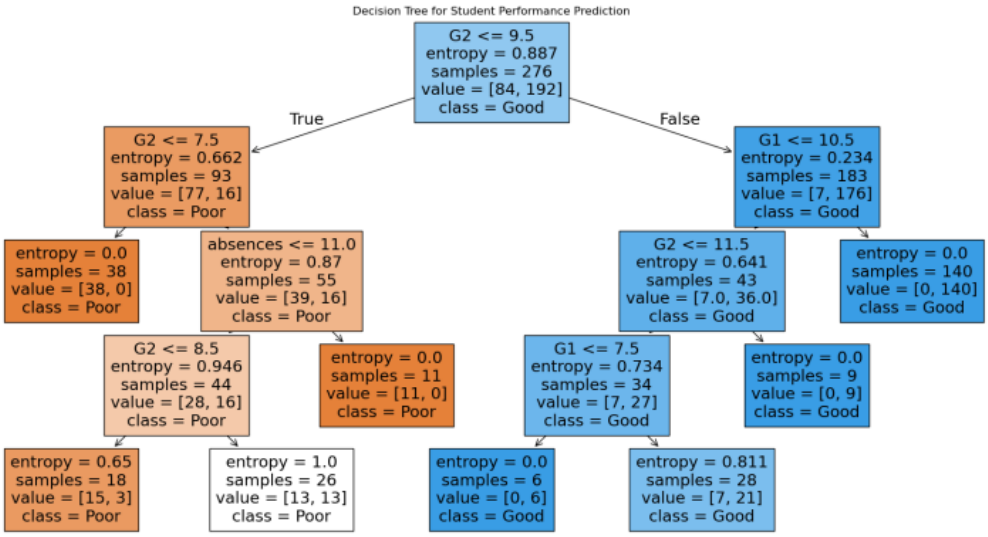
```
conf_matrix = confusion_matrix(y_test, y_pred)

print("\nAccuracy:", accuracy)
print("\nPrecision:", precision)
print("\nRecall:", recall)
print("\nF1-Score:", f1)
print("\nConfusion Matrix:\n", conf_matrix)

disp = ConfusionMatrixDisplay(
    confusion_matrix=confusion_matrix(y_test, y_pred),
    display_labels=['Poor', 'Good']
)

disp.plot(cmap='Blues')
plt.title("Confusion Matrix - Student Performance Prediction")
plt.show()

plt.figure(figsize=(20, 10))
plot_tree(
    dt,
    feature_names=features,
    class_names=['Poor', 'Good'],
    filled=True
)
plt.title("Decision Tree for Student Performance Prediction")
plt.show()
```

10. Output

```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  … \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home     other  …
2     GP   F   15       U     LE3       T     1     1  at_home     other  …
3     GP   F   15       U     GT3       T     4     2   health  services  …
4     GP   F   16       U     GT3       T     3     3    other     other  …

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        6   5   6   6
1      5        3      3     1     1      3        4   5   5   6
2      4        3      2     2     3      3       10   7   8  10
3      3        2      2     1     1      5        2  15  14  15
4      4        3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]

Accuracy: 0.9243697478991597

Precision: 0.9848484848484849

Recall: 0.8904109589041096
```

```
F1-Score: 0.935251798561151

Confusion Matrix:
 [[45  1]
 [ 8 65]]
```





11. Results and Discussion

The Decision Tree model achieved good prediction accuracy while maintaining interpretability. Previous grades and study time were identified as the most influential factors in determining student performance. The use of pruning helped reduce overfitting and improved generalization on unseen data.

12. Conclusion

This project demonstrates the effectiveness of Decision Tree classifiers in predicting student academic performance. The model provides transparent decision rules that can assist educators in identifying at-risk students early and implementing targeted interventions. The approach is simple, interpretable, and aligns well with educational data analysis requirements.