3. STUDY Of Classifiers with respect to Statistical parameters:

Aim: TO Evaluate and compare performance of classifiers with repect to stastical Problem

Objectives:

- Load and preposes iris dataset
- Apply KNN, Decision tree and SVM classifiers
- Evaluate each model using accuracy, Precision, and, F1score
- comparee the problem of the classifiers using tabular format
- to draw observations and conclude which classifier performs best on dataset.

Pseudocode - KNN.

- Input: Training data (X_train, Y_train), Test data (X_test), k value
- calculate distance from all training points
- select the k nearest neighbours
- count class labels among neighbors
- Assign the most Frequent Lable

Decision Tree

- Input : Training data (X_train, y_train)
- Recursive tree building
  - If data is pure or stopping condition met:
    - Return leaf with class label
  - Else
    - Choose best feature
    - Split dataset
    - Recursively build subtrees
    - Use the tree to classify test data
    - Output : Predicted labels

SVM:

- Input : Training data (X_train, y_train)
- Train a model:
  - Find optimal hyperplane that seperates classes
  - Use kernal trick if non-linear data
- For each test instance:
  - Predict using the trained model
- Output : Predicted labels

classifier	Accuracy	Precision	Recall	F1-score
KNN	91.2	89.5	90.1	89.8
Decision Tree	88.7	86.4	87.2	86.8
SVM	93.1	91.8	92.5	92.1

OBSERVATIONS

* SVM consistently delivers the highest performance accross all metrics

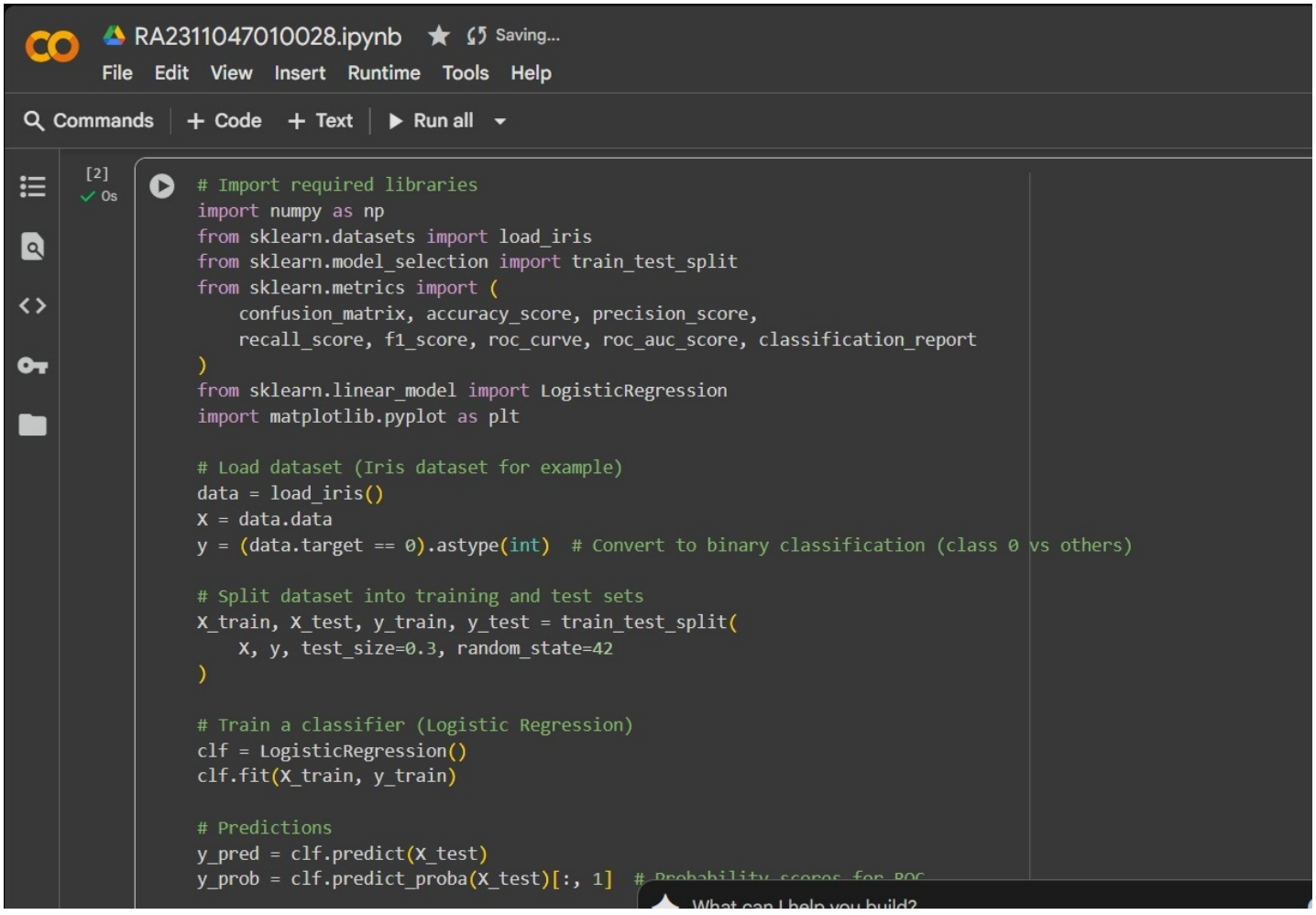
* KNN perfoms well but is influenced by the value k and the scaling of features

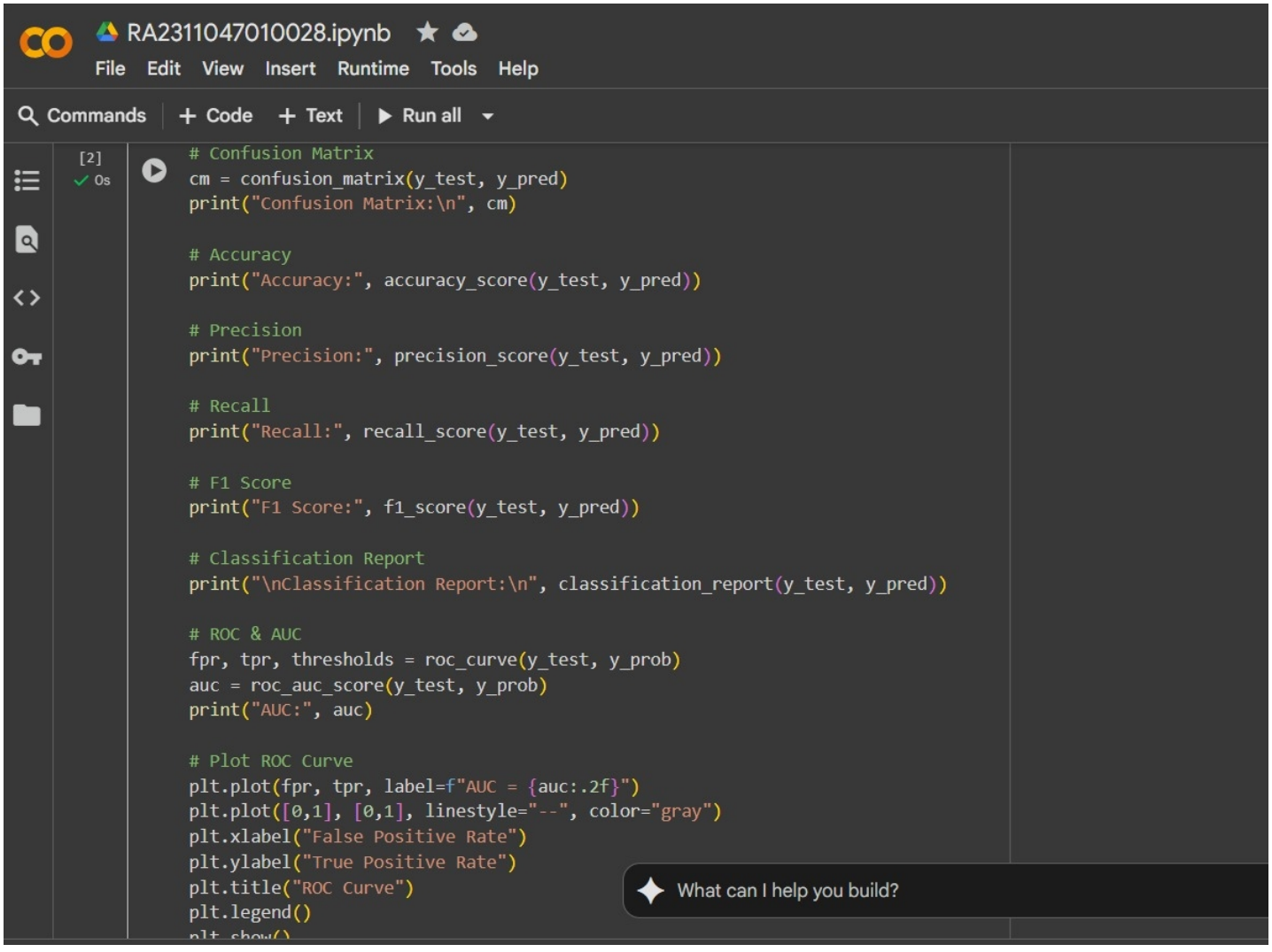
* Decision Tree Provides good results and is easy to intrept, but it may overfit on complex or noisy dataset.

7/14/25

Result:

Successfully Evaluated And Compared performance of all three classifier with respect to statistical Parameters.


RA2311047010028.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[2]
0s
# Import required libraries
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score, roc_curve, roc_auc_score, classification_report
)
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
# Load dataset (Iris dataset for example)
data = load_iris()
X = data.data
y = (data.target == 0).astype(int) # Convert to binary classification (class 0 vs others)
# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42
)
# Train a classifier (Logistic Regression)
clf = LogisticRegression()
clf.fit(X_train, y_train)
# Predictions
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)[:, 1]
What can I help you build?


RA2311047010028.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[2]
0s
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))
# Precision
print("Precision:", precision_score(y_test, y_pred))
# Recall
print("Recall:", recall_score(y_test, y_pred))
# F1 Score
print("F1 Score:", f1_score(y_test, y_pred))
# Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# ROC & AUC
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)
print("AUC:", auc)
# Plot ROC Curve
plt.plot(fpr, tpr, label=f"AUC = {auc:.2f}")
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
What can I help you build?

RA2311047010028.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all

[2] ✓ 0s

```
# Plot ROC Curve
plt.plot(fpr, tpr, label=f"AUC = {auc:.2f}")
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
Confusion Matrix:
 [[26  0]
 [ 0 19]]
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        19

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45

AUC: 1.0
```

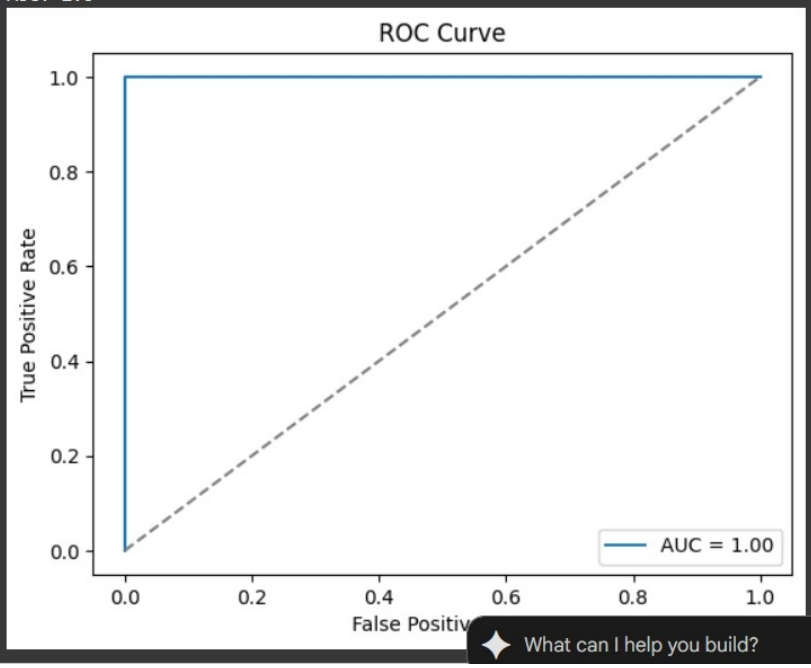
RA2311047010028.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
              accuracy                           1.00        45
             macro avg       1.00      1.00      1.00        45
          weighted avg       1.00      1.00      1.00        45

AUC: 1.0
```



What can I help you build?