+ Code + Text

Rishikesh_2412res99

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score


data = pd.read_csv('/content/datairis.csv')


X = data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = data['Species']


label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)


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


logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)


decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)
y_pred_tree = decision_tree_model.predict(X_test)


def evaluate_model(y_true, y_pred, model_name):
    print(f"Results for {model_name}:")
    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
    print(f"Accuracy: {accuracy_score(y_true, y_pred) * 100:.2f}%")
    print(f"Precision: {precision_score(y_true, y_pred, average='weighted'):.2f}")
    print(f"Recall: {recall_score(y_true, y_pred, average='weighted'):.2f}")
    print(f"F1 Score: {f1_score(y_true, y_pred, average='weighted'):.2f}")
    print("")


evaluate_model(y_test, y_pred_logistic, "Logistic Regression")
```

```
Results for Logistic Regression:
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Accuracy: 100.00%
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

```
evaluate_model(y_test, y_pred_tree, "Decision Tree Classifier")
```

```
Results for Decision Tree Classifier:
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Accuracy: 100.00%
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```