11 write a python program to implement support vector classifier on diabetes.csv dataset and print confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix

# Load the dataset
data = pd.read_csv("diabetes.csv")

# Separate features and target variable
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the SVC model
model = SVC()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Print the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

12 write a python program to implement linear regression on salary.csv dataset and print mean absolute error.plot a graph year of experience vs salary

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('salary_data.csv')

# Extract features (X) and target variable (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean absolute error
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)
```

```
# Plot years of experience vs. salary
plt.scatter(X, y, color='blue', label='Actual data')
plt.plot(X, model.predict(X), color='red', label='Linear regression line')
plt.title('Years of Experience vs. Salary')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.show()
```

13write a python program to implement a classification algorithm on HeartDisease1.csv dataset and print acccuracy score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv('heart.csv')

# Split features (X) and target variable (y)
X = data.drop(columns=['target'])
y = data['target']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Random Forest Classifier
classifier = RandomForestClassifier(random_state=42)
```

```
classifier.fit(X_train, y_train)


# Make predictions
y_pred = classifier.predict(X_test)


# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy Score:", accuracy)
```

14 write a python program to implement a classification algorithm on HeartDisease1.csv dataset and print classification report

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report


# Load the dataset
data = pd.read_csv('heart.csv')


# Split features (X) and target variable (y)
X = data.drop(columns=['target'])
y = data['target']


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Initialize and train the Random Forest Classifier
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_train, y_train)


# Make predictions
```

```
y_pred = classifier.predict(X_test)


# Print classification report

print("Classification Report:")

print(classification_report(y_test, y_pred))
```

15 write a python program to implement a classification algorithm on HeartDisease1.csv dataset and print confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('heart.csv')

# Split features (X) and target variable (y)
X = data.drop(columns=['target'])
y = data['target']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Random Forest Classifier
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Make predictions
y_pred = classifier.predict(X_test)

# Print confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

16 write a python program to implement a knn classifier on HeartDisease1.csv dataset and print accuracy score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv('heart.csv')

# Split features (X) and target variable (y)
X = data.drop(columns=['target'])
y = data['target']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the KNN Classifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy Score:", accuracy)
```

17write a python program to implement logistic regression on HeartDisease1.csv dataset and print confusion matrix

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix

import seaborn as sns

import matplotlib.pyplot as plt
```

```
# Load the dataset
data = pd.read_csv('heart.csv')

# Split features (X) and target variable (y)
X = data.drop(columns=['target'])
y = data['target']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Make predictions
y_pred = logreg.predict(X_test)

# Print confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

18 write a python program to display the confusion matrix in excel without using any predefined function by using the dataset confusion_matrix_example

```
import matplotlib.pyplot as plt

import numpy

from sklearn import metrics


actual = numpy.random.binomial(1,.9,size = 1000)

predicted = numpy.random.binomial(1,.9,size = 1000)


confusion_matrix = metrics.confusion_matrix(actual, predicted)


cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [0, 1])


cm_display.plot()

plt.show()
```

19 write a python program to display the confusion matrix in a matrix format without using any predefined function by using the dataset confusion_matrix_example

```
# Define the confusion matrix values

true_positive = 42

false_positive = 8

false_negative = 18

true_negative = 32


# Create a matrix for the confusion matrix

confusion_matrix = [[true_positive, false_positive], [false_negative, true_negative]]


# Display the confusion matrix

print("Confusion Matrix:")

for row in confusion_matrix:
    print(row)
```

20 write a python program to implement a regression algorithm on Advertising.csv and print any one error

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Load the dataset

data = pd.read_csv('Advertising.csv')


# Extract features (X) and target variable (y)

X = data[['TV', 'Radio', 'Newspaper']]

y = data['Sales']


# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Initialize and train the Linear Regression model

model = LinearRegression()

model.fit(X_train, y_train)


# Make predictions

y_pred = model.predict(X_test)


# Calculate mean squared error (MSE) as an example of regression error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
```

21 write a python program to implement a regression algorithm on Advertising.csv and print mean absolute error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

# Load the dataset
data = pd.read_csv('Advertising.csv')

# Split features (X) and target variable (y)
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean absolute error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
```

22write a python program to implement a regression algorithm on Advertising.csv and print mean squared error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the dataset
data = pd.read_csv('Advertising.csv')

# Split features (X) and target variable (y)
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean squared error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
```

23 write a python program to implement a regression algorithm on Advertising.csv and print root mean squared error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error


# Load the dataset
data = pd.read_csv('Advertising.csv')


# Split features (X) and target variable (y)
X = data[['TV', 'Radio', 'Newspaper']]
```

```
y = data['Sales']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean squared error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate root mean squared error (RMSE)
rmse = mse ** 0.5
print("Root Mean Squared Error (RMSE):", rmse)
```

24 write a python program to implement adaboost classifier on social.csv dataset and print classification report

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import classification_report

# Load the dataset
data = pd.read_csv("social.csv")

# Splitting the dataset into features and target variable
X = data.drop('Estimated Salary', axis=1)
```

```
y = data['Estimated Salary']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initializing AdaBoost classifier
ada_clf = AdaBoostClassifier(n_estimators=50, random_state=42)

# Training the classifier
ada_clf.fit(X_train, y_train)

# Predicting the test set results
y_pred = ada_clf.predict(X_test)

# Printing classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

25 write a python program to implement adaboost classifier on social.csv dataset and print accuracy score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv("social.csv")

# Split the data into features and target
X = data.drop(columns=['Estimated Salary'])
y = data['Estimated Salary']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the AdaBoost classifier
ada_clf = AdaBoostClassifier(n_estimators=50, random_state=42)

# Train the AdaBoost classifier
ada_clf.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = ada_clf.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy Score:", accuracy)
```

26 write a python program to implement adaboost classifier on social.csv dataset and print confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix

# Load the dataset
data = pd.read_csv("social.csv")

# Split the data into features and target
X = data.drop(columns=['Estimated Salary'])
y = data['Estimated Salary']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the AdaBoost classifier
ada_clf = AdaBoostClassifier(n_estimators=50, random_state=42)

# Train the AdaBoost classifier
ada_clf.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = ada_clf.predict(X_test)

# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

27 write a python program to implement decision tree classifier on social.csv dataset and print accuracy score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv("social.csv")

# Split the data into features and target
X = data.drop(columns=['Estimated Salary'])
y = data['Estimated Salary']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Decision Tree classifier
tree_clf = DecisionTreeClassifier(random_state=42)

# Train the Decision Tree classifier
tree_clf.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = tree_clf.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy Score:", accuracy)
```

28 write a python program to implement support vector classifier on social.csv dataset and print precision score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import precision_score

# Load the dataset
data = pd.read_csv("social.csv")

# Split the data into features and target
X = data.drop(columns=['Estimated Salary'])
y = data['Estimated Salary']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Support Vector Classifier
svc = SVC()

# Train the Support Vector Classifier
svc.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = svc.predict(X_test)

# Calculate the precision score
precision = precision_score(y_test, y_pred)
print("Precision Score:", precision)
```

29 write a python program to implement find S-algorithm on data.csv dataset

```
import csv

# Function to load data from CSV file
def load_data(file_path):
    with open(file_path, 'r') as file:
        reader = csv.reader(file)
        data = [row for row in reader]
    return data

# Function to implement Find-S algorithm
def find_s_algorithm(data):
    num_attributes = len(data[0]) - 1  # Number of attributes (excluding the class label)
    hypothesis = ['0'] * num_attributes  # Initialize hypothesis with most specific values
    for instance in data:
        if instance[-1] == '1':  # Check if the instance belongs to positive class
            for i in range(num_attributes):
                if hypothesis[i] == '0':  # If attribute value is not already set to a specific value
                    hypothesis[i] = instance[i]
                elif hypothesis[i] != instance[i]:
                    hypothesis[i] = '?'  # If attribute value conflicts, set it as '?'

    return hypothesis

# Function to test the hypothesis
def test_hypothesis(hypothesis, test_instance):
    for i in range(len(hypothesis)):
        if hypothesis[i] != '?' and hypothesis[i] != test_instance[i]:
            return 'No'  # If any attribute value doesn't match, return 'No'
    return 'Yes'  # If all attribute values match, return 'Yes'

# Main function
def main():
    file_path = 'data.csv'
    data = load_data(file_path)

    # Print the data
    print("Data:")
    for row in data:
        print(row)

    # Implement Find-S algorithm
    hypothesis = find_s_algorithm(data)
    print("\nHypothesis:")
    print(hypothesis)

    # Test the hypothesis
    test_instance = ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
    print("\nTesting Hypothesis with instance:", test_instance)
    result = test_hypothesis(hypothesis, test_instance)
    print("Result:", result)

if __name__ == "__main__":
    main()
```

30 write a python program to implement decision tree regressor on Advertising.csv and print mean absolute error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error

# Load the dataset
data = pd.read_csv('Advertising.csv')

# Separate features (X) and target variable (y)
X = data.drop(columns=['Sales'])  # Features
y = data['Sales']  # Target variable

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize Decision Tree Regressor
```

```
model = DecisionTreeRegressor(random_state=42)


# Train the model

model.fit(X_train, y_train)


# Predict the target variable on the testing set

y_pred = model.predict(X_test)


# Calculate mean absolute error

mae = mean_absolute_error(y_test, y_pred)


print("Mean Absolute Error:", mae)
```

1 write a python program to implement a classification algorithm on diabetes.csv(hint remove column patientID) print accuracy

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv("diabetes.csv")

# Remove the patientID column
df = df.drop(columns=["patientID"])

# Split features and target variable
X = df.drop(columns=["Outcome"]) # Features
y = df["Outcome"] # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict the labels for test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

2 write a python program to implement a classification algorithm on diabetes.csv(hint remove column patientID) print classification report

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Load the dataset
df = pd.read_csv("diabetes.csv")

# Remove the patientID column
df = df.drop(columns=["patientID"])

# Split features and target variable
X = df.drop(columns=["Outcome"]) # Features
y = df["Outcome"] # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict the labels for test set
```

```
y_pred = clf.predict(X_test)

# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

3 write a python program to implement a classification algorithm on diabetes.csv(hint remove column patientID) print confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("diabetes.csv")

# Remove the patientID column
df = df.drop(columns=["patientID"])

# Split features and target variable
X = df.drop(columns=["Outcome"]) # Features
y = df["Outcome"] # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict the labels for test set
y_pred = clf.predict(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

4 write a python program to implement a logistic regression on diabetes.csv(hint:remove the column patientID)print accuracy_score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv("diabetes.csv")

# Remove the patientID column
df = df.drop(columns=["patientID"])

# Split features and target variable
X = df.drop(columns=["Outcome"])  # Features
y = df["Outcome"]  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the Logistic Regression Classifier
clf = LogisticRegression(random_state=42)
clf.fit(X_train, y_train)

# Predict the labels for test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
```

```
print("Accuracy:", accuracy)
```

5 write a python program to implement a knn classifier on diabetes.csv and print accuracy_score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv("diabetes.csv")

# Remove the patientID column
df = df.drop(columns=["patientID"])

# Split features and target variable
X = df.drop(columns=["Outcome"])  # Features
y = df["Outcome"]  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the KNN classifier
k = 5  # Number of neighbors
clf = KNeighborsClassifier(n_neighbors=k)
clf.fit(X_train, y_train)

# Predict the labels for test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```