DAY 6 LAB EXPERIMENTS

Name: Rakshitha V B

Reg.no: 192324004

Dept: B.Tech AI & DS

Scenario: You are a researcher working in a medical lab, investigating the effectiveness of a new treatment for a specific disease. You have collected data from a clinical trial with two groups: a control group receiving a placebo, and a treatment group receiving the new drug.Your goal is to analyze the data using hypothesis testing and calculate the p-value to determine if the new treatment has a statistically significant effect compared to the placebo. You will use the matplotlib library to visualize the data and the p-value.

Solution:

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import ttest_ind

control_group = np.array([52, 55, 50, 48, 54, 53, 51, 49])

treatment_group = np.array([60, 62, 58, 61, 63, 59, 60, 64])

t_stat, p_value = ttest_ind(treatment_group, control_group)

print("T-statistic:", t_stat)

print("P-value:", p_value)

plt.boxplot([control_group, treatment_group],tick_labels=['Control (Placebo)', 'Treatment (Drug)'])

plt.ylabel("Disease Severity Reduction")

plt.title("Clinical Trial Results Comparison")

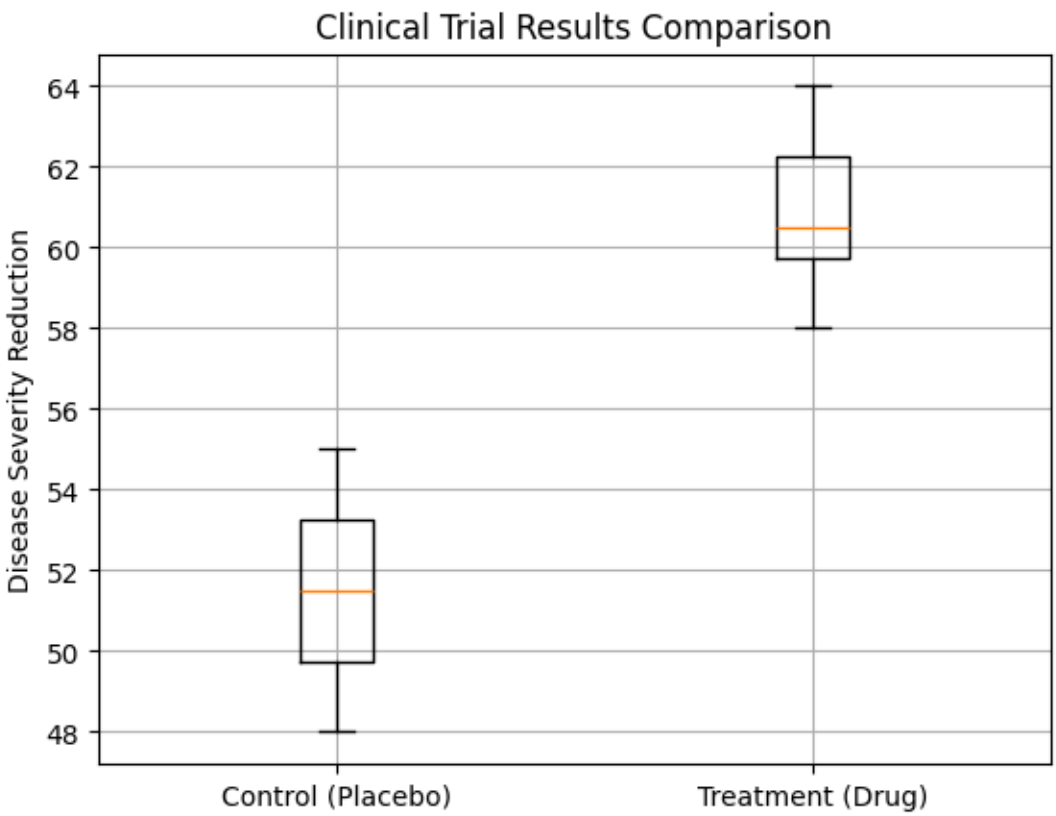
plt.grid(True)

plt.show()

```
control_group = np.array([52, 55, 50, 48, 54, 53, 51, 49])
treatment_group = np.array([60, 62, 58, 61, 63, 59, 60, 64])

t_stat, p_value = ttest_ind(treatment_group, control_group)
print("T-statistic:", t_stat)
print("P-value:", p_value)
```

```
T-statistic: 8.333333333333334
P-value: 8.487739214599852e-07
```


Clinical Trial Results Comparison
Disease Severity Reduction
64
62
60
58
56
54
52
50
48
Control (Placebo)
Treatment (Drug)

Question: K-Nearest Neighbors (KNN) Classifier

You are working on a classification problem to predict whether a patient has a certain medical condition or not based on their symptoms. You have collected a dataset of patients with labelled data (0 for no condition, 1 for the condition) and various symptom features.

Write a Python program that allows the user to input the features of a new patient and the value of k (number of neighbors). The program should use the KNN classifier from the scikit-learn library to predict whether the patient has the medical condition or not based on the input features.

Solution:

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
X = np.array([
    [1, 1, 1, 0],
    [0, 1, 0, 0],
    [1, 1, 1, 1],
    [0, 0, 0, 0],
    [1, 0, 1, 1],
    [0, 1, 0, 1]
])
y = np.array([1, 0, 1, 0, 1, 0])
k = int(input("Enter the value of k (number of neighbors): "))
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X, y)
print("Enter patient symptoms (1 = Yes, 0 = No)")
fever = int(input("Fever: "))
cough = int(input("Cough: "))
fatigue = int(input("Fatigue: "))
headache = int(input("Headache: "))
new_patient = np.array([[fever, cough, fatigue, headache]])
prediction = knn.predict(new_patient)
if prediction[0] == 1:
    print("Prediction: Patient HAS the medical condition.")
else:
```

print("Prediction: Patient does NOT have the medical condition.")

```
print("Enter patient symptoms (1 = Yes, 0 = No)")
fever = int(input("Fever: "))
cough = int(input("Cough: "))
fatigue = int(input("Fatigue: "))
headache = int(input("Headache: "))
new_patient = np.array([[fever, cough, fatigue, headache]])
prediction = knn.predict(new_patient)
```

```
Enter patient symptoms (1 = Yes, 0 = No)
Fever: 0
Cough: 1
Fatigue: 1
Headache: 1
```

```
if prediction[0] == 1:
    print("Prediction: Patient HAS the medical condition.")
else:
    print("Prediction: Patient does NOT have the medical condition.")
```

```
Prediction: Patient HAS the medical condition.
```

Question: Evaluation Metrics for Model Performance

You have trained a machine learning model on a dataset, and now you want to evaluate its performance using various metrics.

Write a Python program that loads a dataset and trained model from scikit-learn. The program should ask the user to input the names of the features and the target variable they want to use for evaluation. The program should then calculate and display common evaluation metrics such as accuracy, precision, recall, and F1-score for the model's predictions on the test data.

Solution:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd

iris = load_iris()

X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.Series(iris.target)

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

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
```

f1 = f1_score(y_test, y_pred, average='macro')

print("Accuracy :", accuracy)

print("Precision:", precision)

print("Recall   :", recall)

print("F1-score :", f1)

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
```

```
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1-score :", f1)
```

```
Accuracy : 1.0
Precision: 1.0
Recall   : 1.0
F1-score : 1.0
```

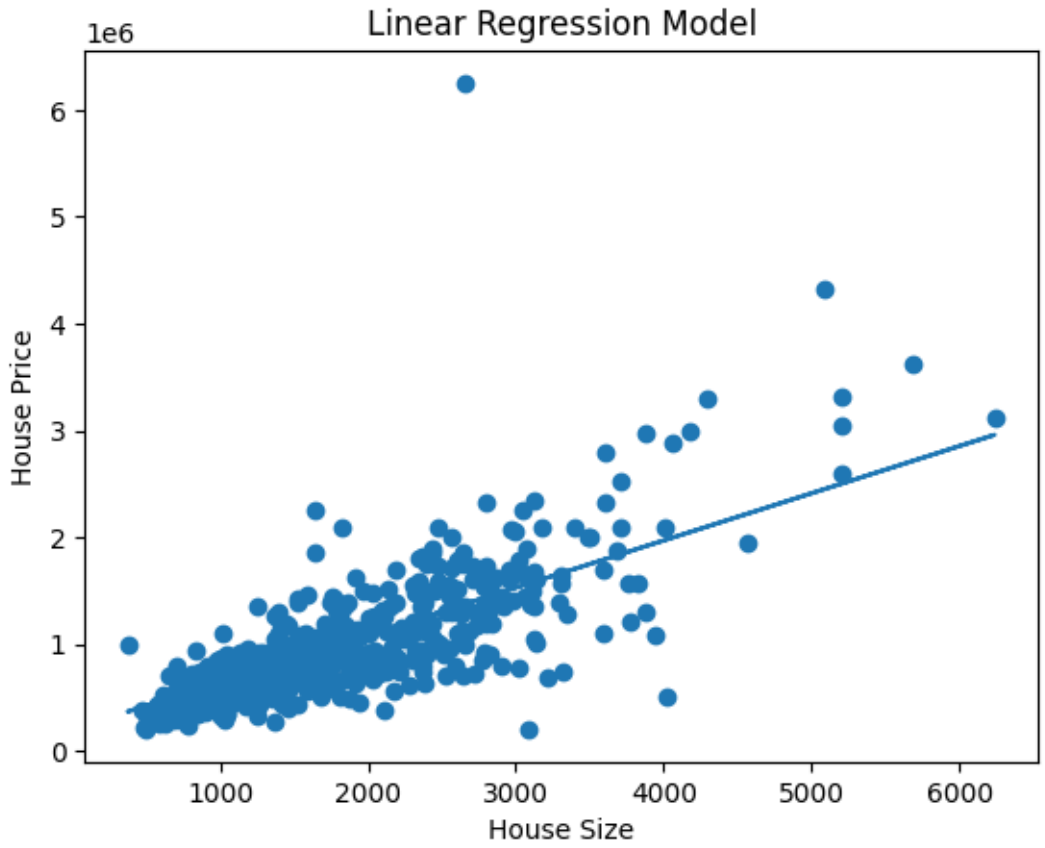
Scenario: You work as a data scientist for a real estate company. The company has collected data on various houses, including features such as the size of the house, number of bedrooms, location, and other relevant attributes. The marketing team wants to build a predictive model to estimate the price of houses based on their features. They believe that linear regression modeling can be an effective approach for this task.

Question: Your task is write a Python program to perform bivariate analysis and build a linear regression model to predict house prices based on a selected feature (e.g., house size) from the dataset. Additionally, you need to evaluate the model's performance to ensure its accuracy and reliability.

Solution:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
data = pd.read_excel("House_Prediction.xlsx")
X = data[['size']]
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred)
plt.xlabel("House Size")
plt.ylabel("House Price")
plt.title("Linear Regression Model")
plt.show()
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("Mean Absolute Error:", mae)
print("Mean Squared Error :", mse)
print("Root Mean Squared Error:", rmse)
print("R² Score:", r2)
```




```
Mean Absolute Error: 221292.7817047867
Mean Squared Error : 135323017778.13336
Root Mean Squared Error: 367862.77030726196
R² Score: 0.583112692492
```

Scenario: Suppose you are working as a data scientist for a medical research organization. Your team has collected data on patients with a certain medical condition and their treatment outcomes. The dataset includes various features such as age, gender, blood pressure, cholesterol levels, and whether the patient responded positively ("Good") or negatively ("Bad") to the treatment. The organization wants to use this model to identify potential candidates who are likely to respond positively to the treatment and improve their medical approach.

Question: Your task is to build a classification model using the KNN algorithm to predict the treatment outcome ("Good" or "Bad") for new patients based on their features. Evaluate the model's performance using accuracy, precision, recall, and F1-score.Make predictions on the test set and display the results.

Solution:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
data = pd.DataFrame({ "age": [25, 45, 35, 50, 23, 40, 60, 48, 33, 55],
   "gender": ["Male", "Female", "Female", "Male", "Female", "Male", "Male", "Female", "Male", "Female"],
   "blood_pressure": [120, 140, 130, 150, 118, 135, 160, 145, 128, 155],
   "cholesterol": [180, 220, 200, 240, 175, 210, 260, 230, 195, 250],
   "outcome": ["Good", "Bad", "Good", "Bad", "Good", "Good", "Bad", "Bad", "Good", "Bad"]
})
encoder = LabelEncoder()
data["gender"] = encoder.fit_transform(data["gender"])
data["outcome"] = encoder.fit_transform(data["outcome"])
X = data.drop("outcome", axis=1)
y = data["outcome"]
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=42)
```

```
k = 3
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X_train, y_train)
y_pred = model.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)
print("Model Performance Metrics")
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1-score :", f1)
results = pd.DataFrame({ "Actual": y_test.values, "Predicted": y_pred
})
print("\nPrediction Results on Test Data:")
print(results)
```

```
Model Performance Metrics
Accuracy : 1.0
Precision: 1.0
Recall   : 1.0
F1-score : 1.0
```

```
results = pd.DataFrame({
    "Actual": y_test.values,
    "Predicted": y_pred
})

print("\nPrediction Results on Test Data:")
print(results)
```

```
Prediction Results on Test Data:
   Actual  Predicted
0       1          1
1       0          0
2       1          1
```