Part B- Data Visualization Lab Exercises

Experiment 1: Using Python, create your own having columns plant name, sunlight exposure, plant height and answer the following questions:

a. Is there a relationship between the number of hours of sunlight exposure and the height of the plants?

b. Visualize the relationship between sunlight exposure and plant height using a scatterplot.

c. Calculate the correlation coefficient between sunlight exposure and plant height. Is the correlation positive or negative? Is it strong or weak?

d. Based on the correlation coefficient, can we conclude that there is a significant association between sunlight exposure and plant growth rate?

```
import matplotlib.pyplot as plt

import pandas as pd


# reading a file already created using excel

df = pd.read_csv('plants.csv')

df
```

```
#Or you can Create a sample dataset directly

data = {

    'plant_name': ['Fern', 'Cactus', 'Bamboo', 'Rose',
'Tulip', 'Daisy', 'Sunflower', 'Lily', 'Orchid', 'Maple'],

    'sunlight_exposure': [5, 10, 12, 8, 6, 7, 14, 11, 9, 5],
# hours of sunlight

    'plant_height': [30, 150, 200, 60, 50, 40, 180, 70, 40,
20]  # height in cm

}


df1 = pd.DataFrame(data)


#Reducing the dataframe to two columns

df = df[['sunlight_exposure', 'plant_height']]

df.head()



#Visualize the relationship between sunlight exposure and
plant height using a scatterplot

plt.scatter(df['sunlight_exposure'], df['plant_height'],
color="black", marker="*")

plt.title('Relationship between sunlight exposure and plant
height')
```

```
plt.xlabel('Sunlight Exposure')

plt.ylabel('Plant Height')

plt.grid(True)

plt.show()
```

#III. Calculate the correlation coefficient between sunlight exposure and plant height. Is the correlation positive or negative? Is it strong or weak?

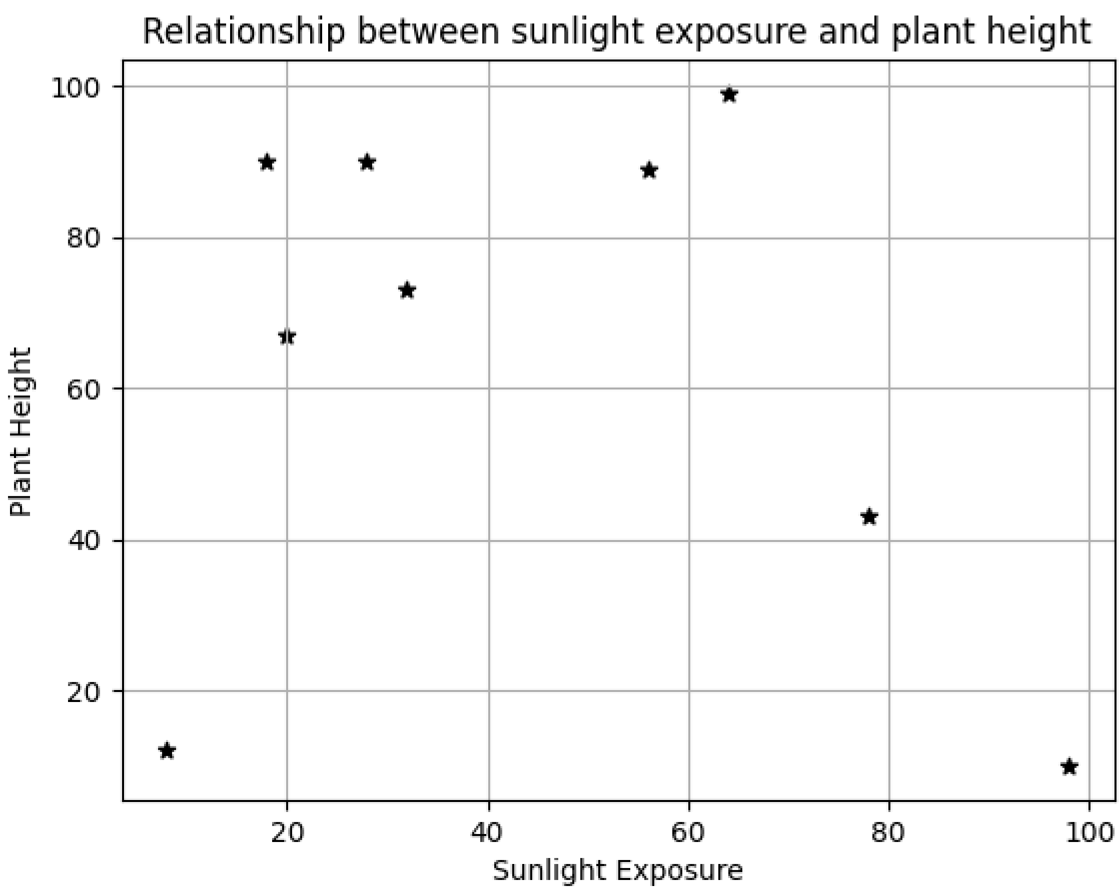
```
correlation =
df['sunlight_exposure'].corr(df['plant_height'])

print(f"Correlation between sunlight exposure and plant
height: {correlation}")
```

#d.Based on the correlation coefficient, can we conclude that there is a significant association between sunlight exposure and plant growth rate?

```
threshold = 0.7

if abs(correlation) >= threshold:

    print("There is a significant association between
sunlight exposure and plant growth rate.")

else:

    print("There is no significant association between
sunlight exposure and plant growth rate.")
```

#I. Is there a relationship between the number of hours of sunlight exposure and the height of the plants?

#No, based on the correlation coefficient, it is conclusive that there is no correlation between the number of hours of sunlight exposure and the plant height.



Correlation between sunlight exposure and plant height: -0.2411875043974829

There is no significant association between sunlight exposure and plant growth rate.

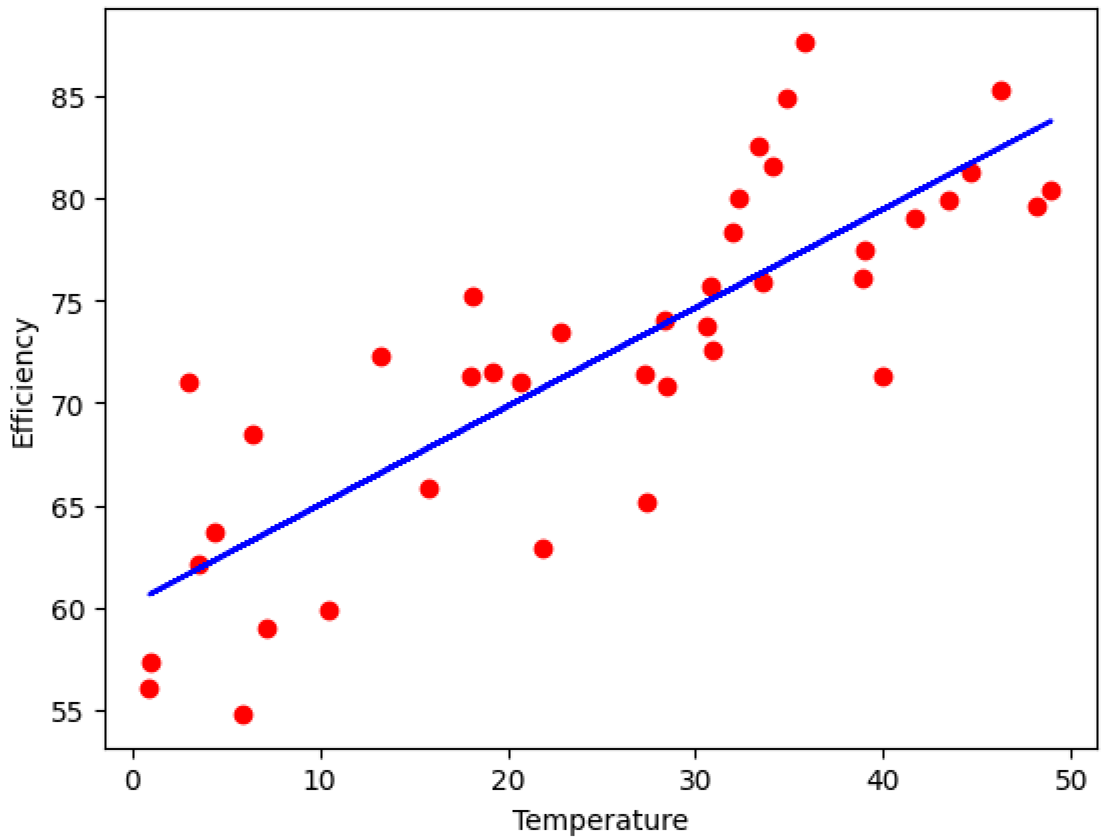
Experiment 2: In a solar panel efficiency study, researchers want to investigate the relationship between the temperature and the efficiency of solar panels. They collected data on the temperature (in Celsius) and the corresponding efficiency (in percentage) of solar panels over a period of time. The dataset contains measurements from 50 different days.

a. **Using Simple Linear Regression, can you develop a model to predict the efficiency of solar panels based on the temperature?**
b. **Perform an F-test to determine whether temperature significantly predicts the efficiency of solar panels.**
c. **Conduct a t-test to assess the significance of the regression coefficient for temperature.**

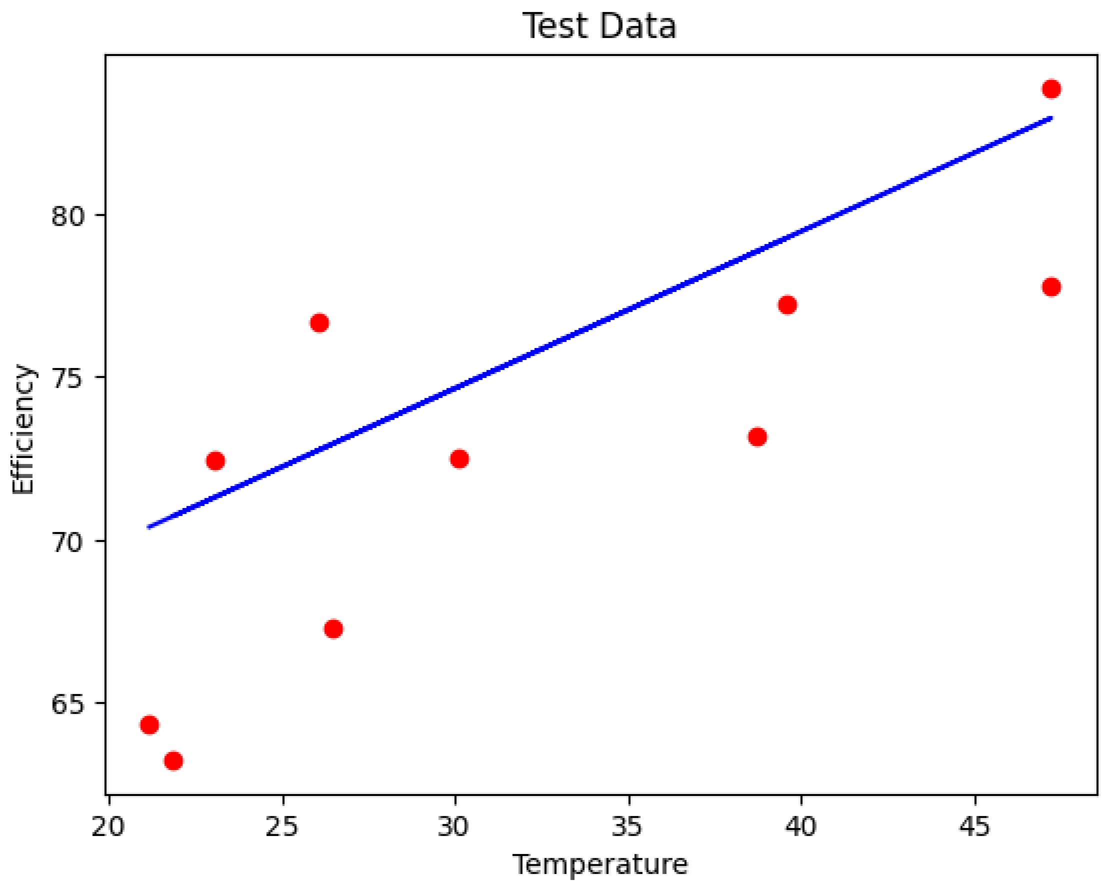
```
import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression


df = pd.read_csv('solar_efficiency_temp.csv')

df.head()

#extracting only required columns

df.iloc[:,[1,2]]

# Splitting data into Training and Testing.

X = df[['Temperature']]

y = df['Efficiency']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state = 0)


#Training the model

model = LinearRegression()

model = model.fit(X_train, y_train)


#plotting the train data

plt.scatter(X_train, y_train, color = 'red')

plt.plot(X_train, model.predict(X_train), color = 'blue')

plt.xlabel('Temperature')

plt.ylabel('Efficiency')

plt.show()


#plotting the test data

plt.scatter(X_test, y_test, color = 'red')

plt.plot(X_test, model.predict(X_test), color = 'blue')

plt.xlabel('Temperature')

plt.ylabel('Efficiency')

plt.title('Test Data')

plt.show()
```

```
#F and T test
import statsmodels.api as sm
import pandas as pd

X = df[['Temperature']]
Y = df['Efficiency']

X = sm.add_constant(X)
model = sm.OLS(Y, X).fit()


f_stat = model.fvalue
f_p_value = model.f_pvalue

t_stat = model.tvalues['Temperature']
t_p_value = model.pvalues['Temperature']

print(f"F-statistic: {f_stat:.2f}")
 print(f"t-statistic for temperature: {t_stat:.2f}")
```



Efficiency
Temperature
85
80
75
70
65
60
55
0
10
20
30
40
50



F-statistic: 91.59

t-statistic for temperature: 9.57

Experiment 3: Given the dataset of 30 students' study hours and exam scores, how would you build a linear regression model to predict exam scores? Describe the steps you would take to diagnose the regression model, including checking assumptions, identifying outliers, and handling influential points. Finally, evaluate the model's performance and discuss any insights gained.

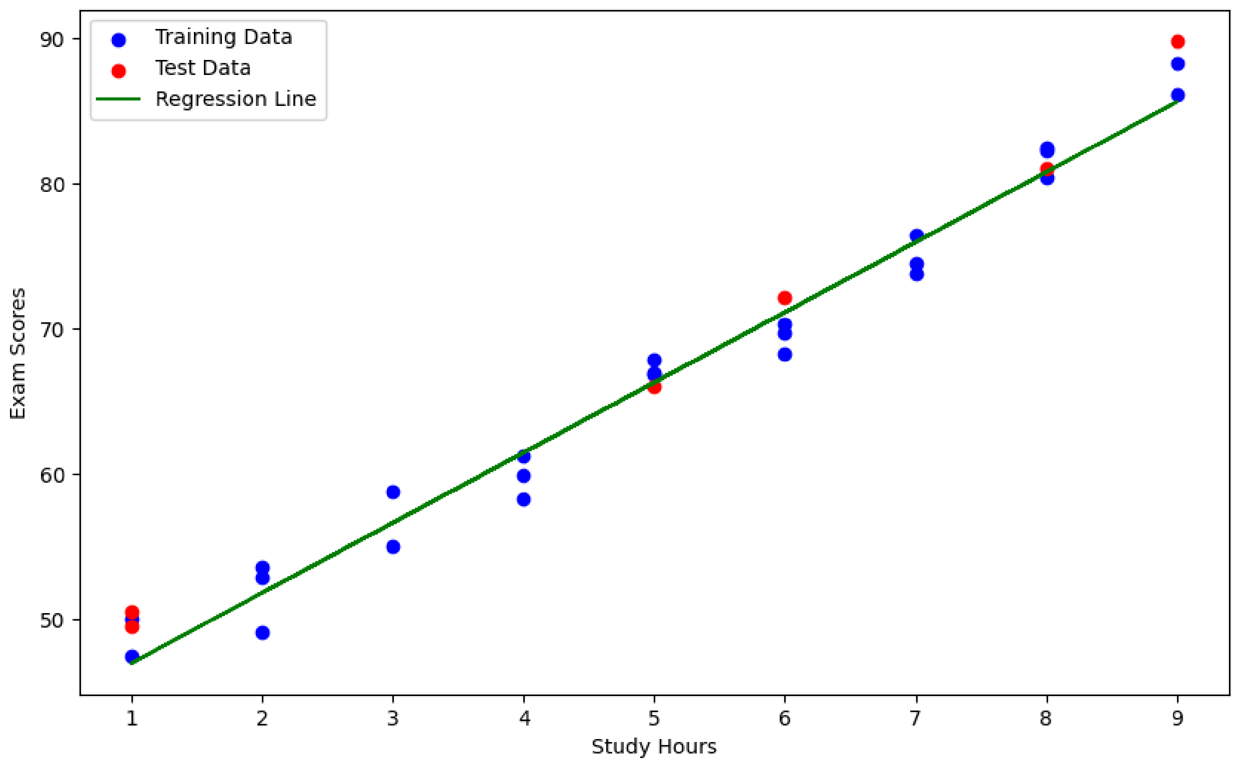
```
import pandas as pd
```

```
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score


df = pd.read_csv('student_data.csv')

df
```

#splitting data and training the model

```
X = df[['StudyHours']]

y = df['ExamScore']

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

model = LinearRegression()

model.fit(X_train, y_train)


y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

plt.figure(figsize=(10, 6))

plt.scatter(X_train['StudyHours'], y_train, label='Training Data', color='blue')
```

```
plt.scatter(X_test['StudyHours'], y_test, label='Test Data',
color='red')

plt.plot(X_train,   model.predict(X_train),   color='green',
label='Regression Line')

plt.xlabel('Study Hours')

plt.ylabel('Exam Scores')

plt.legend()

plt.show()

print("Mean Squared Error:", mse)

print("R-squared:", r2)
```



Mean Squared Error: 6.353087473532352

R-squared: 0.9710368865999279

Experiment 4: In a retail experiment, we want to understand how advertising expenditure, store location, and competition affect sales revenue. Using synthetic data, implement multiple linear regression in Python to analyse these factors. Interpret the coefficients, perform an F-test to assess overall model significance, and conduct t-tests to evaluate the significance of individual coefficients.

```
# Importing necessary libraries

import pandas as pd

import numpy as np

from sklearn import linear_model

import matplotlib.pyplot as plt


# Loading the dataset

df = pd.read_csv('/content/sales (part A).csv')

df.head()  # Displaying first few rows


# Training a linear regression model on multiple features

reg = linear_model.LinearRegression()
```

```
reg.fit(df[['AdvertisingExpenditure', 'StoreLocation',
'Competition']], df.SalesRevenue)

reg.coef_  # Model coefficients

reg.intercept_  # Model intercept


# Analyzing the relationship between AdvertisingExpenditure
and SalesRevenue

x1 = df[['AdvertisingExpenditure']]

y1 = df[['SalesRevenue']]


# Splitting data into training and testing sets

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x1, y1,
test_size=0.3)


# Training and visualizing the model

from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(x_train, y_train)


# Printing the coefficients and intercept after training
```

#note if you standarise the data before finding intercept and coefficient the value will be between 0 and 1

```
print(f"Coefficients for AdvertisingExpenditure model: {model.coef_}")

print(f"Intercept for AdvertisingExpenditure model: {model.intercept_}")

plt.scatter(x_train, y_train, color='red')  # Training data

plt.scatter(x_test, y_test, color='blue')  # Testing data

plt.plot(x_train, model.predict(x_train), color='green')  # Regression line

plt.xlabel('AdvertisingExpenditure')

plt.ylabel('SalesRevenue')

plt.show()
```

 # Repeating analysis for StoreLocation and Competition

 # StoreLocation

```
x2 = df[['StoreLocation']]

x_train, x_test, y_train, y_test = train_test_split(x2, y1, test_size=0.3)

model.fit(x_train, y_train)
```

 # Printing the coefficients and intercept for StoreLocation
```
model print(f"Coefficients for StoreLocation model: {model.coef_}")
```

```
print(f"Intercept for StoreLocation model:
{model.intercept_}")

plt.scatter(x_train, y_train, color='red')

plt.scatter(x_test, y_test, color='blue')

plt.plot(x_train, model.predict(x_train), color='green')

plt.xlabel('StoreLocation')

plt.ylabel('SalesRevenue')

plt.show()


# Competition

x3 = df[['Competition']]

x_train, x_test, y_train, y_test = train_test_split(x3, y1,
test_size=0.3)

model.fit(x_train, y_train)

# Printing the coefficients and intercept for Competition
model print(f"Coefficients for Competition model:
{model.coef_}")

print(f"Intercept for Competition model:
{model.intercept_}")

plt.scatter(x_train, y_train, color='red')

plt.scatter(x_test, y_test, color='blue')

plt.plot(x_train, model.predict(x_train), color='green')
```

```
plt.xlabel('Competition')

plt.ylabel('SalesRevenue')

plt.show()
```

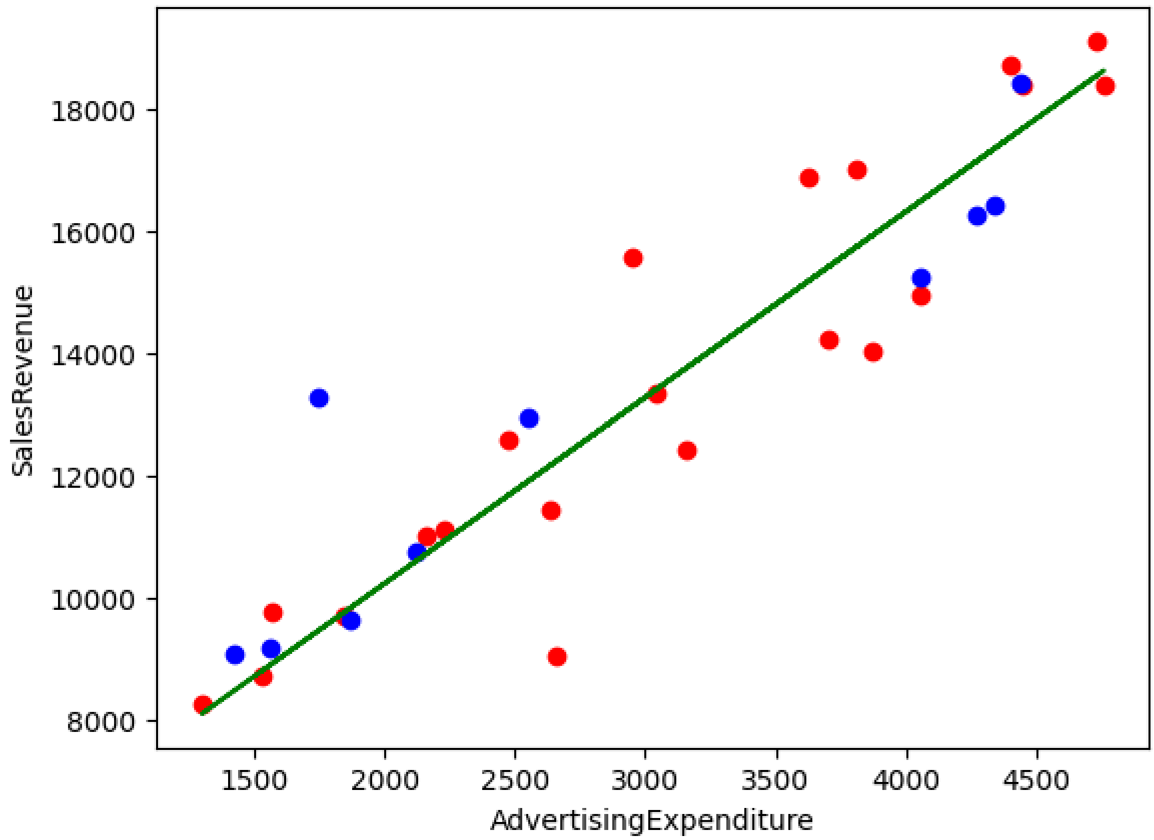
#F test

```
import statsmodels.api as sm

import pandas as pd


X = df[['AdvertisingExpenditure', 'Competition',
'StoreLocation']]

Y = df['SalesRevenue']


X = sm.add_constant(X)


model = sm.OLS(Y, X).fit()


f_stat = model.fvalue


t_stat_advertising = model.tvalues['AdvertisingExpenditure']

t_stat_competition = model.tvalues['Competition']
```

```
t_stat_location = model.tvalues['StoreLocation']


print(f"F-statistic: {f_stat:.2f}")

print(f"t-statistic for AdvertisingExpenditure:
{t_stat_advertising:.2f}")

print(f"t-statistic for Competition:
{t_stat_competition:.2f}")

print(f"t-statistic for StoreLocation:
{t_stat_location:.2f}")
```

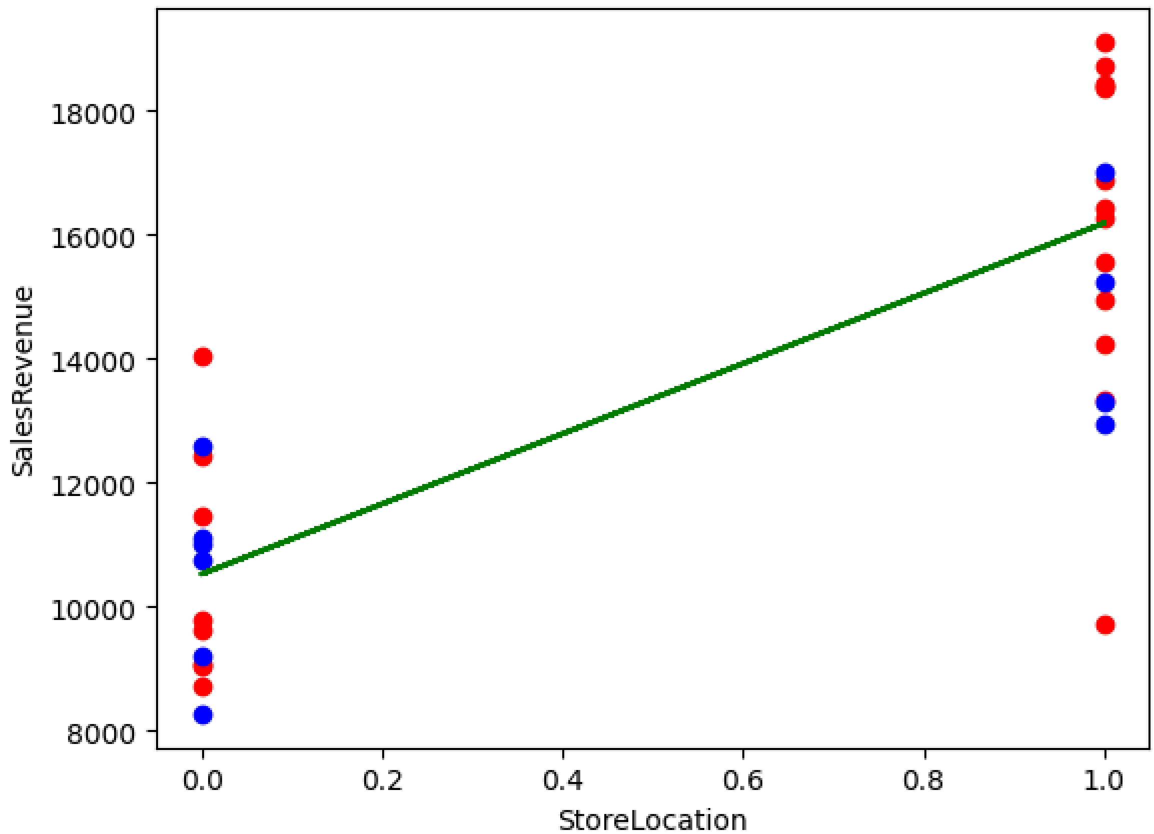
Coefficients for AdvertisingExpenditure model: [[3.03985947]]

Intercept for AdvertisingExpenditure model: [4152.65185694]



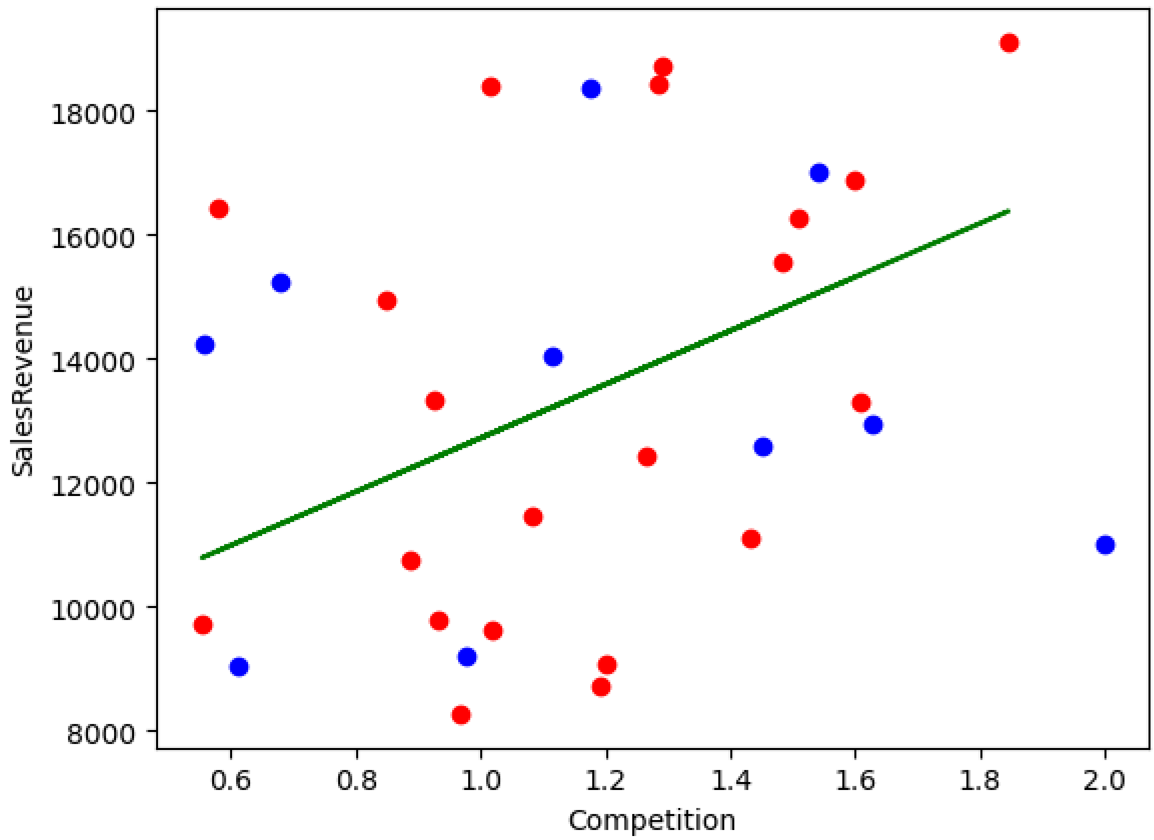
Coefficients for StoreLocation model: [[5660.75]]

Intercept for StoreLocation model: [10521.25]



Coefficients for Competition model: [[4326.56396988]]

Intercept for Competition model: [8389.76085821]



F-statistic: 177.17

t-statistic for AdvertisingExpenditure: 12.74

t-statistic for Competition: 5.35

t-statistic for StoreLocation: 4.90

Experiment 5: Given a dataset that contains information about different types of flowers (e.g., Iris dataset), perform classification using the k-Nearest Neighbors (kNN) algorithm. Evaluate the performance of the model by calculating its accuracy and visualize the results using appropriate techniques.

```
# Import necessary libraries
from sklearn.datasets import load_iris
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 Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Labels

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=50)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

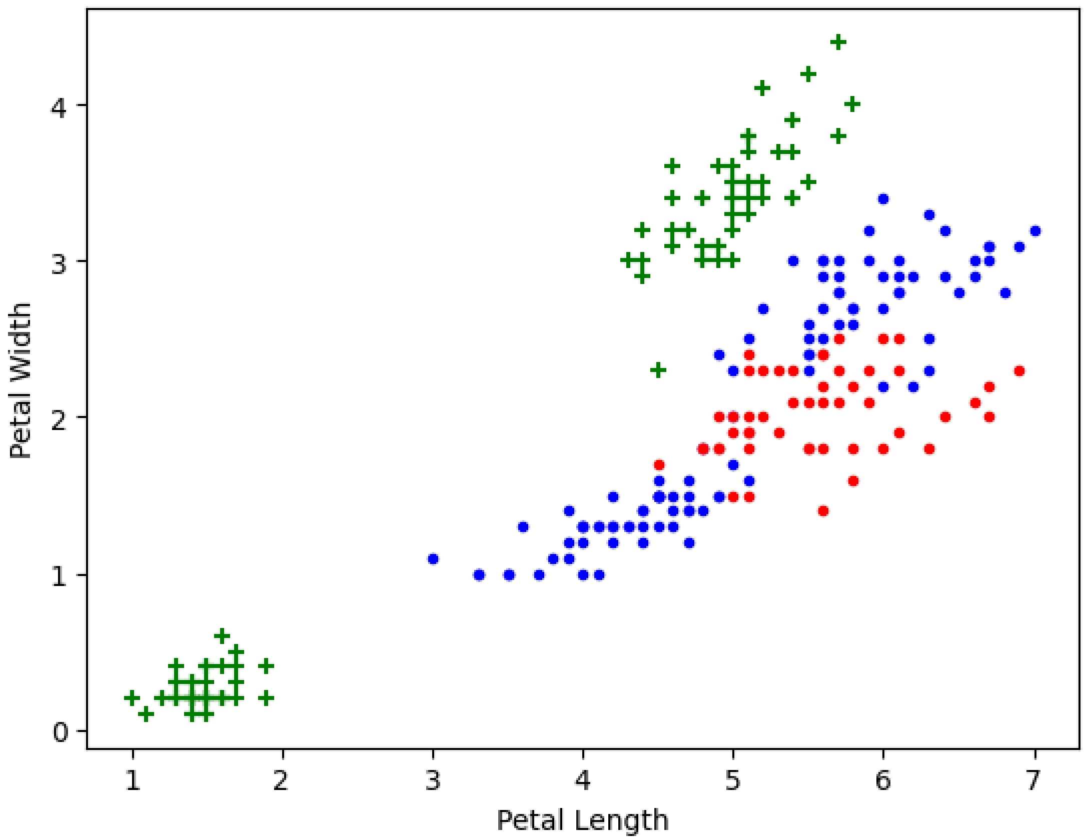
```
X_test = scaler.transform(X_test)


# Train KNN classifier

k = 5  # Number of neighbors

knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(X_train, y_train)


# Make predictions

y_pred = knn.predict(X_test)


# Calculate accuracy

accuracy = accuracy_score(y_test, y_pred) * 100


# Print accuracy in percentage

print(f"Accuracy of K-Nearest Neighbors classifier on Iris dataset: {accuracy:.2f}%")


import pandas as pd

from sklearn.datasets import load_iris

iris = load_iris()

iris.feature_names
```

```
iris.target_names

df = pd.DataFrame(iris.data,columns=iris.feature_names)

df.head()

df['target'] = iris.target

df.head(-5)

df[df.target==1].head()

df0 = df[df.target==0]

df1 = df[df.target==1]

df2 = df[df.target==2]

df2.head()


import matplotlib.pyplot as plt

plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')

plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')

plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
```

```
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')

plt.scatter(df2['petal length (cm)'], df2['petal width (cm)'],color="red",marker='.')

from sklearn.model_selection import train_test_split

X = df.drop(['target'], axis='columns')

y = df.target

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

len(X_train)

len(X_test)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

knn.score(X_test, y_test)
```

Accuracy of K-Nearest Neighbors classifier on Iris dataset: 95.56%

0.9666666666666667



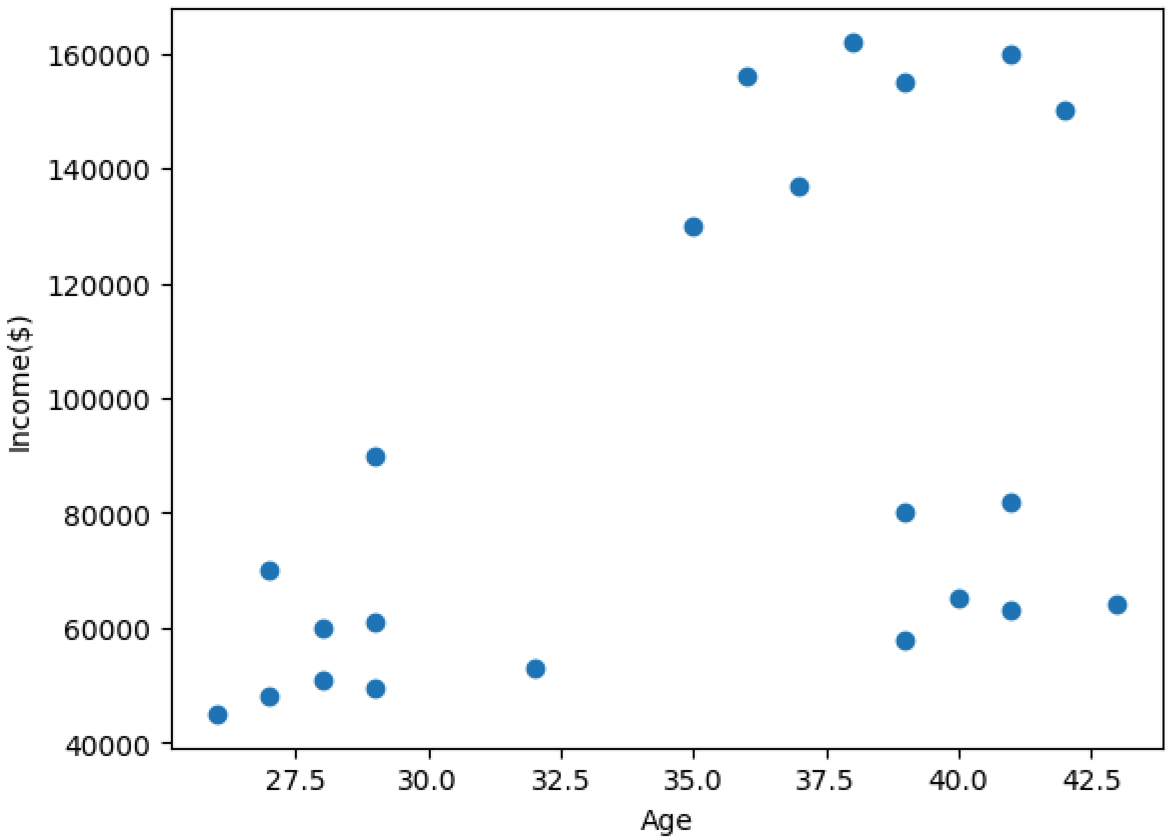
Experiment 6: Given a dataset that contains customer information (such as Age, Income, and Spending Score), perform K-means clustering to group customers into clusters. Use visualization chart, plot the data before and after grouping. Also, use the Elbow Method to determine the optimal number of clusters.

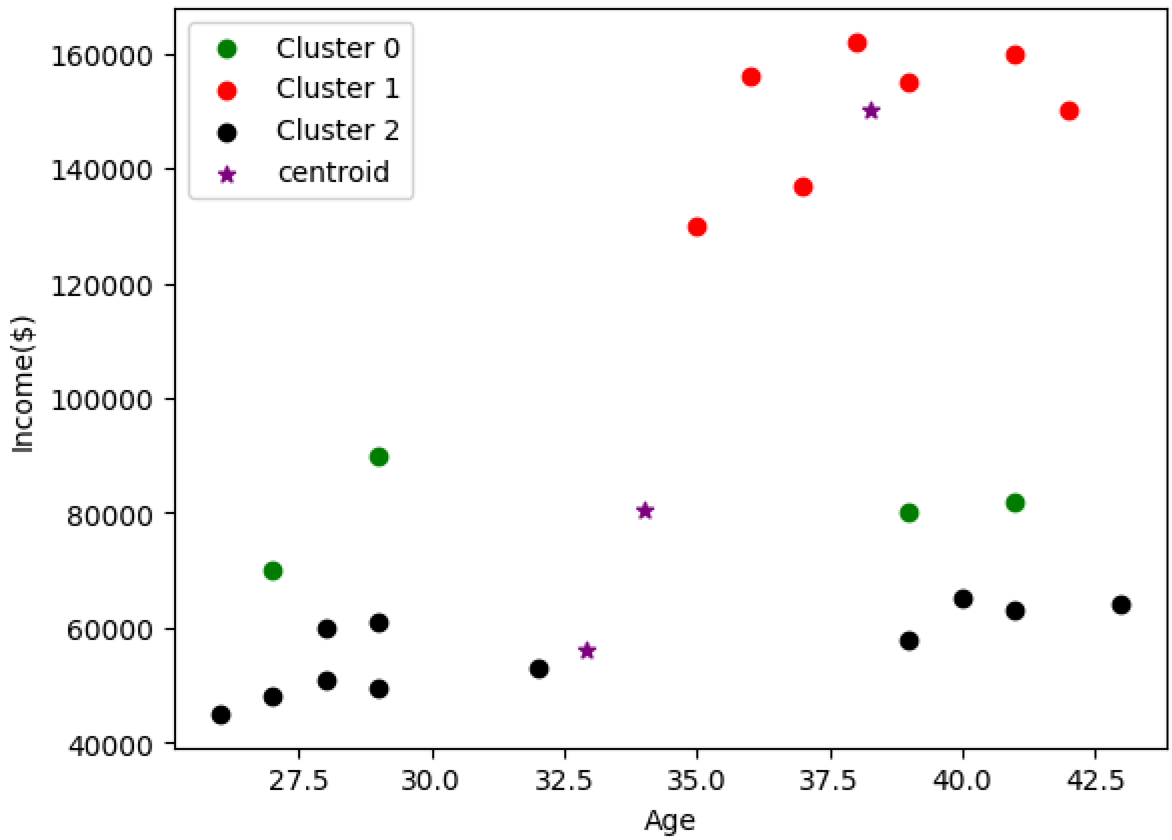
```
from sklearn.cluster import KMeans

import pandas as pd
```

```
from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

df = pd.read_csv("income_clustering.csv")

print(df.head())

plt.scatter(df.Age,df['Income($)'])

plt.xlabel('Age')

plt.ylabel('Income($)')

plt.show()

km=KMeans(n_clusters=3)

y_predicted = km.fit_predict(df[['Age','Income($)']])

y_predicted

df['cluster']=y_predicted

df.head()

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

plt.scatter(df1.Age,df1['Income($)'],color='green',
label='Cluster 0')

plt.scatter(df2.Age,df2['Income($)'],color='red',
label='Cluster 1')

plt.scatter(df3.Age,df3['Income($)'],color='black',
label='Cluster 2')
```

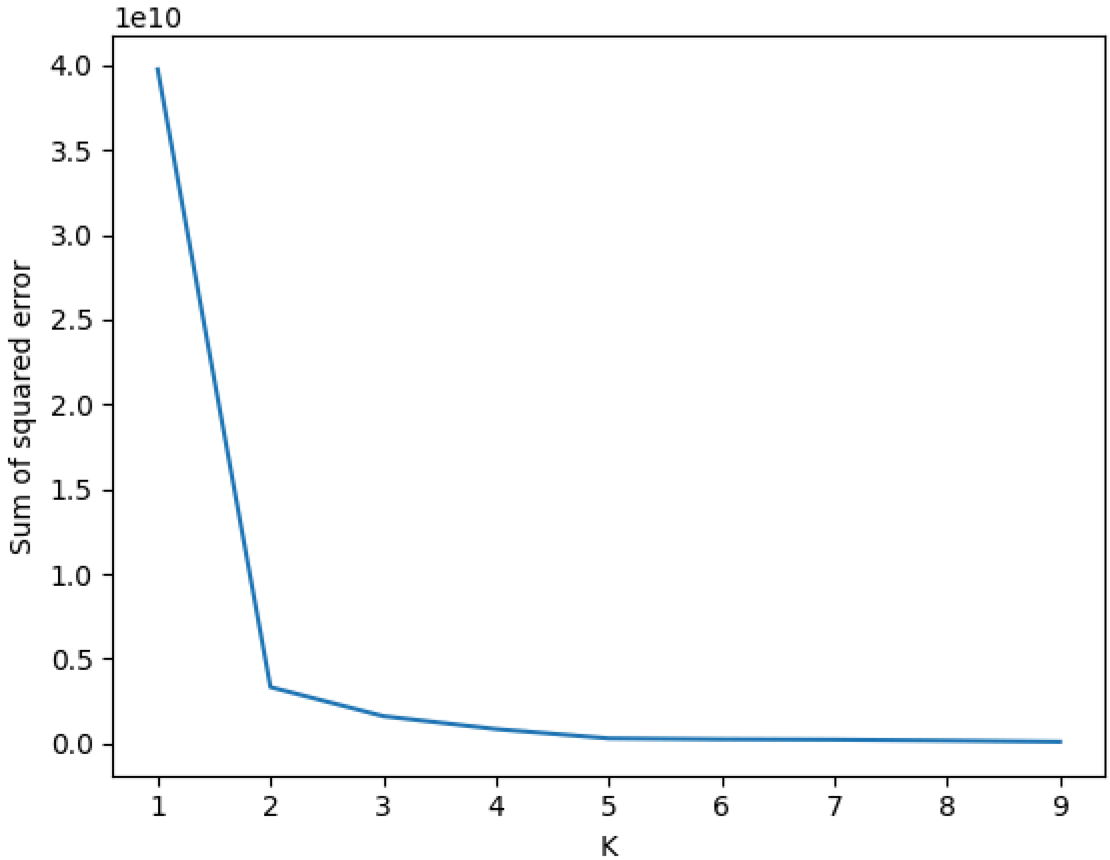
```
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1
],color='purple',marker='*',label='centroid')

plt.legend()

plt.xlabel('Age')

plt.ylabel('Income($)')

plt.show()

sse = []

k_rng = range(1,10)

for k in k_rng:

  km=KMeans(n_clusters=k)

  km.fit(df[['Age','Income($)']])

  sse.append(km.inertia_)

plt.xlabel('K')

plt.ylabel('Sum of squared error')

plt.plot(k_rng,sse)
```

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000


160000
140000
120000
100000
80000
60000
40000
Income($)
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Age



[<matplotlib.lines.Line2D at 0x7c4abd5d06a0>]



1e10
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Sum of squared error
1
2
3
4
5
6
7
8
9
K