Part B:

Data Visualization using Python (`matplotlib, pandas, scikit-learn and statsmodels`)

Prepared by **Mohit Nair (1MS22IS079)**

Exercises

Exercise 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 scatter plot.
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?

Exercise 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.

Exercise 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.

Exercise 4: In a retail Exercise, 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.

Exercise 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.

Exercise 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 a visualization chart, plot the data before and after grouping. Also, use the Elbow Method to determine the optimal number of clusters.

Exercise 7: Compare the effectiveness of two teaching methods, A and B, in helping students pass a test. Analyse the proportions of passing students, calculate confidence intervals for the difference in proportions, conduct significance tests, and evaluate the area under the ROC curve for predictive accuracy.

Repository Credits

All code mentioned in this manual is hosted at the link:
https://github.com/themohitnair/DVLab
All datasets used in this code can also be found at the same link.

Exercise 1

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

1. Is there a relationship between the number of hours of sunlight exposure and the height of the plants?
2. Visualize the relationship between sunlight exposure and plant height using a scatterplot.
3. Calculate the correlation coefficient between sunlight exposure and plant height. Is the correlation positive or - negative? Is it strong or weak?
4. Based on the correlation coefficient, can we conclude that there is a significant association between sunlight exposure and plant growth rate?

In [49]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [50]:
```
data = {
    'plant_name': ["Tomato", "Lemon", "Capsicum", "Mulberry", "Persimmon", "Passion
Fruit"],
    'sunlight_exposure': [20, 56, 18, 98, 34, 95 ],
    'plant_height': [67, 89, 12, 101, 45, 121]
}

df = pd.DataFrame(data)
df.head()
```

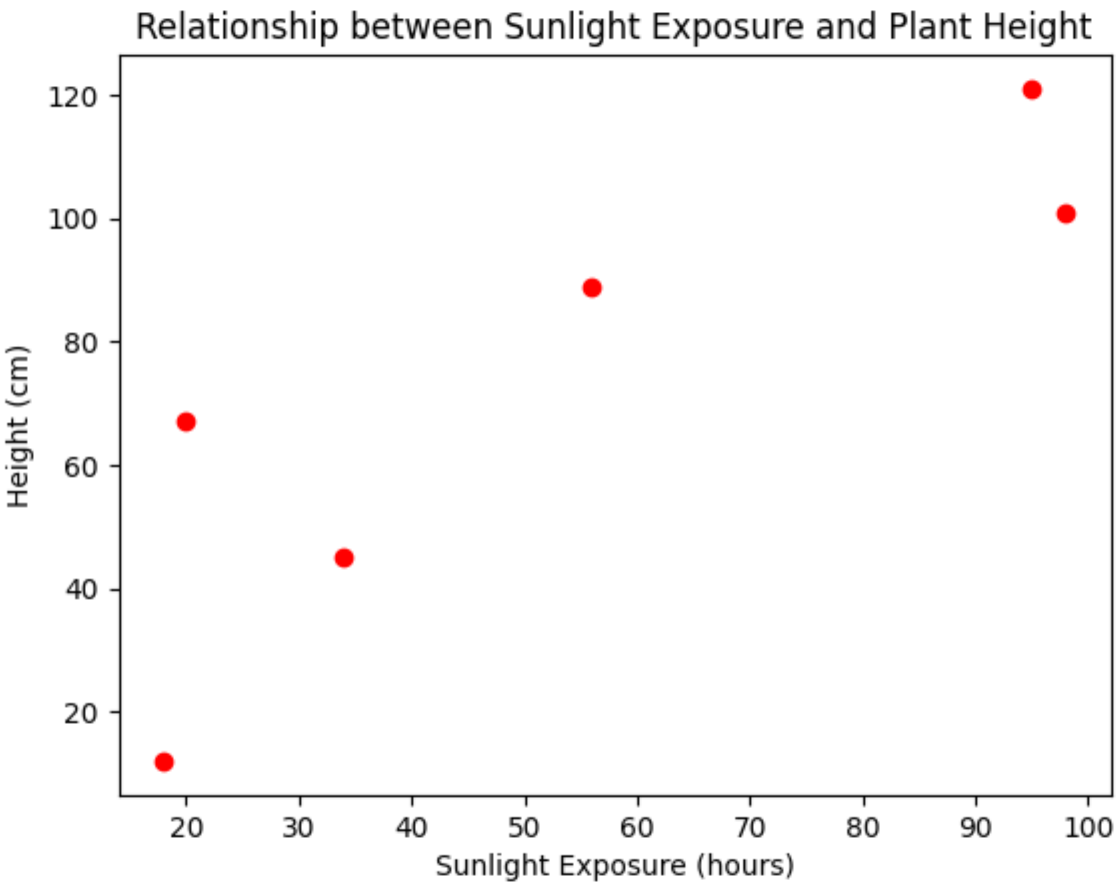
Out[50]:

	plant_name	sunlight_exposure	plant_height
0	Tomato	20	67
1	Lemon	56	89
2	Capsicum	18	12
3	Mulberry	98	101
4	Persimmon	34	45

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

In [51]:
```
plt.scatter(df['sunlight_exposure'], df['plant_height'], color="r")
plt.title("Relationship between Sunlight Exposure and Plant Height")
plt.xlabel("Sunlight Exposure (hours)")
plt.ylabel("Height (cm)")
```

Out[51]:
```
Text(0, 0.5, 'Height (cm)')
```



In [52]:
```
reduced_df = df[['sunlight_exposure', 'plant_height']]
reduced_df.corr()
```

Out[52]:

	sunlight_exposure	plant_height
sunlight_exposure	1.00000	0.86669
plant_height	0.86669	1.00000

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

In [53]:
```
corr_coeff = reduced_df['sunlight_exposure'].corr(df['plant_height'])
print(f"Correlation co-efficient: {corr_coeff}")

if corr_coeff < 0:
    sign = "negative"
elif corr_coeff > 0:
    sign = "positive"
else:
    sign = "neither"
print(f"The correlation coefficient is {sign}.")

strength = "strong" if abs(corr_coeff) > 0.5 else "weak"
print(f"The correlation is {strength}.")
```

```
Correlation co-efficient: 0.8666898574354881
The correlation coefficient is positive.
The correlation is strong.
```

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

In [54]:
```
if abs(corr_coeff) > 0:
    print(f"Yes, there is a {strength} {sign} linear relationship between Sunlight
Exposure and Plant Height.")
else:
    print("There is no relationship between Sunlight Exposure and Plant Height.")
```

```
Yes, there is a strong positive linear relationship between Sunlight Exposure and Plant H
eight.
```

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

In [55]:
```
if strength == "strong":
    print("Yes, we can conclude that there is significant association between Sunlight
Exposure and Plant Height.")
elif strength == "weak":
    print("The association between Sunlight Exposure and Plant Height is not
significant.")
elif sign == "neither":
    print("There is no association between Sunlight Exposure and Plant Height.")
```

```
Yes, we can conclude that there is significant association between Sunlight Exposure and
Plant Height.
```

Exercise 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.

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

In [61]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [62]:
```
df = pd.read_csv("solar_efficiency_temp.csv")
df.head()
```

Out[62]:

	temperature	efficiency
0	27.440675	65.188987
1	35.759468	87.633611
2	30.138169	72.520823
3	27.244159	71.431708
4	21.182740	64.327393

In [63]:
```
X = df[['temperature']]
y = df['efficiency']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
```

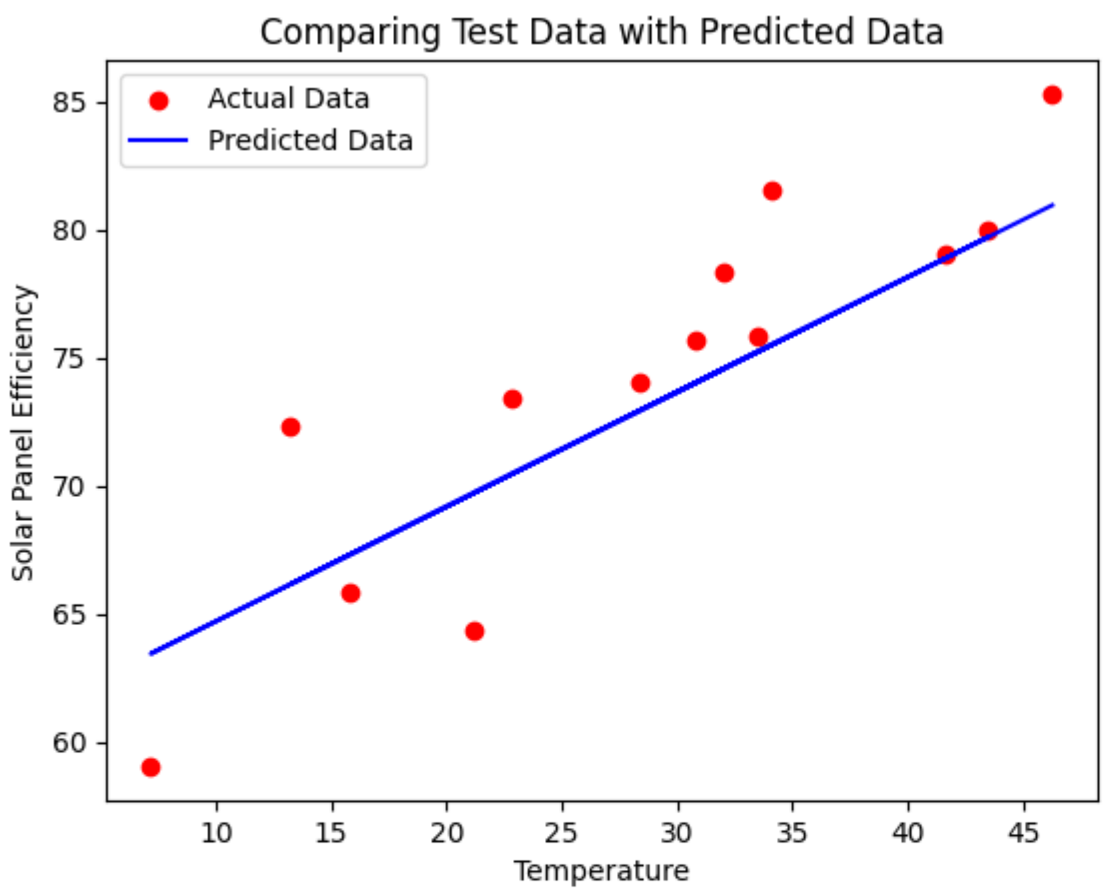
In [64]:
```
model = LinearRegression()

model.fit(X_train, y_train)
```

Out[64]:
```
▾ LinearRegression  i ?

LinearRegression()
```

In [65]:
```
y_pred = model.predict(X_test)
```

In [66]:
```
plt.title("Comparing Test Data with Predicted Data")
plt.xlabel("Temperature")
plt.ylabel("Solar Panel Efficiency")
plt.scatter(X_test, y_test, color="r", label="Actual Data")
plt.plot(X_test, y_pred, color="b", label="Predicted Data")
plt.legend()
```

Out[66]: <matplotlib.legend.Legend at 0x718ecb75ae90>



In [67]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error = {mse}\nr^2 = {r2}")
```

```
Mean Squared Error = 13.184913541739215
r^2 = 0.7385465712906308
```

2. Perform an F-test to determine whether temperature significantly predicts the efficiency of solar panels.
3. Conduct a t-test to assess the significance of the regression coefficient for temperature.

In [68]:
```
import statsmodels.api as sm
```

In [69]:
```
X = sm.add_constant(df[['temperature']])
Y = df['efficiency']
```

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

In [70]:

```
t_statistic = model.tvalues['temperature']
p_value_t = model.pvalues['temperature']

f_statistic = model.fvalue
p_value_f = model.f_pvalue

print(f"F-statistic = {f_statistic}\nt-statistic = {t_statistic}")

if p_value_t < 0.05:
    print("The regression coefficient for temperature is statistically significant.")
else:
    print("The regression coefficient for temperature is NOT statistically significant.")

if p_value_t < 0.05:
    print("The temperature significantly predicts the efficiency of solar panels.")
else:
    print("The temperature does NOT significantly predicts the efficiency of solar panels.")
```

```
F-statistic = 91.58938851225089
t-statistic = 9.570234506648786
The regression coefficient for temperature is statistically significant.
The temperature significantly predicts the efficiency of solar panels.
```

Exercise 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.

In [71]:

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [72]:

```
df = pd.read_csv("student_data.csv")
df.head()
```

Out[72]:

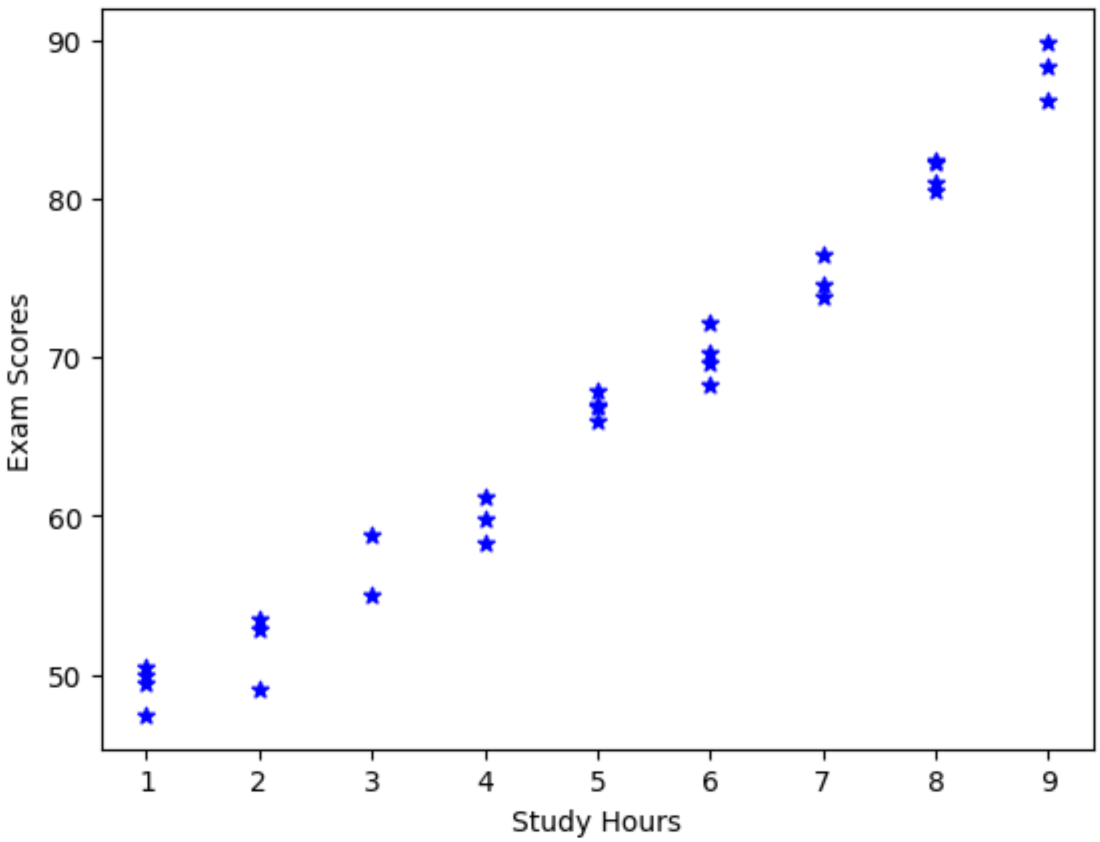
	StudyHours	ExamScore
0	5	66.938936
1	3	58.791081
2	7	73.818557
3	4	59.844898
4	6	69.690213

In [73]:

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

plt.scatter(X, y, color="b", marker="*")
plt.xlabel("Study Hours")
plt.ylabel("Exam Scores")
```

Out[73]:

```
Text(0, 0.5, 'Exam Scores')
```



In [74]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42,
test_size=0.25)
```

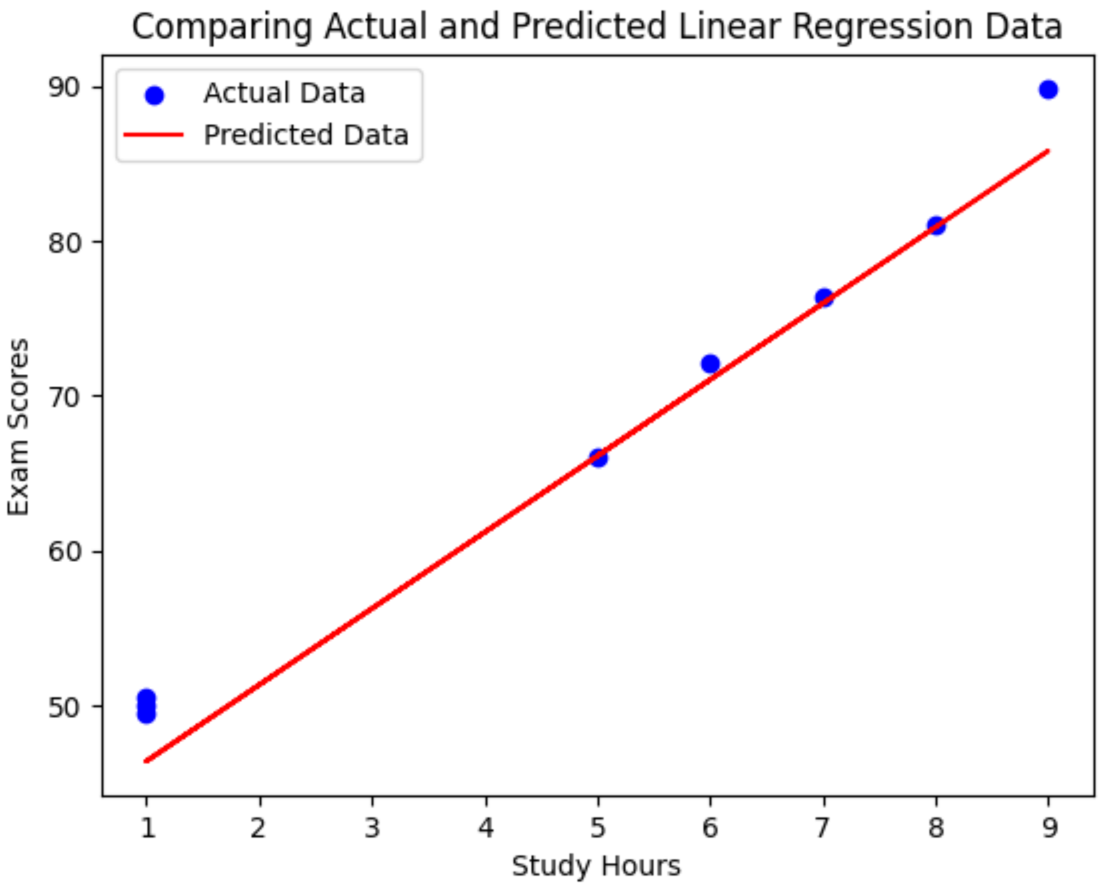
In [75]:
```
model = LinearRegression()

model.fit(X_train, y_train)
```

Out[75]:
```
▾ LinearRegression  i ?

LinearRegression()
```

In [76]:
```
y_pred = model.predict(X_test)
```

In [77]:
```
plt.title("Comparing Actual and Predicted Linear Regression Data")
plt.xlabel("Study Hours")
plt.ylabel("Exam Scores")
plt.scatter(X_test, y_test, color="b", label="Actual Data")
plt.plot(X_test, y_pred, color="r", label="Predicted Data")
plt.legend()
```

Out[77]:
```
<matplotlib.legend.Legend at 0x7cb01a7ae0d0>
```



In [78]:

```
r2 = r2_score(y_pred, y_test)
mse = mean_squared_error(y_pred, y_test)

print(f"Mean Squared Error = {mse}\nr^2 = {r2}")
```

```
Mean Squared Error = 7.148419001716639
r^2 = 0.9696208656224866
```

Exercise 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.

In [31]:
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [32]:
```
df = pd.read_csv("sales.csv")
df.head()
```

Out[32]:

	AdvertisingExpenditure	StoreLocation	Competition	SalesRevenue
0	4269	1	1.509	16259
1	4441	1	1.285	18432
2	1866	0	1.018	9630
3	3871	0	1.116	14029
4	4760	1	1.015	18392

In [33]:
```
X = df[["AdvertisingExpenditure", "Competition", "StoreLocation"]]
Y = df["SalesRevenue"]
```

In [34]:
```
model = LinearRegression()

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=42)

model.fit(X_train, Y_train)
```

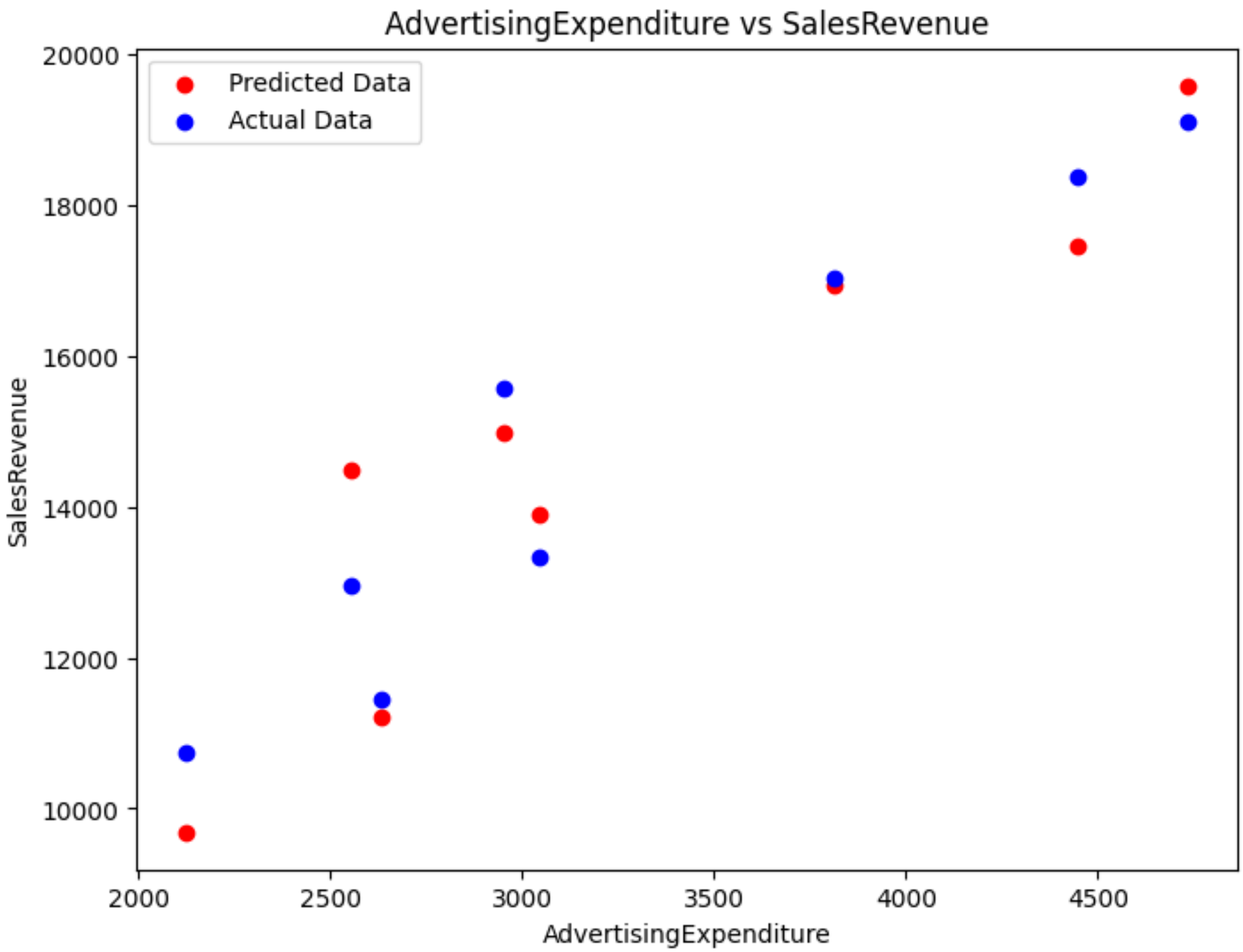
Out[34]:
```
▾ LinearRegression  i ?

LinearRegression()
```

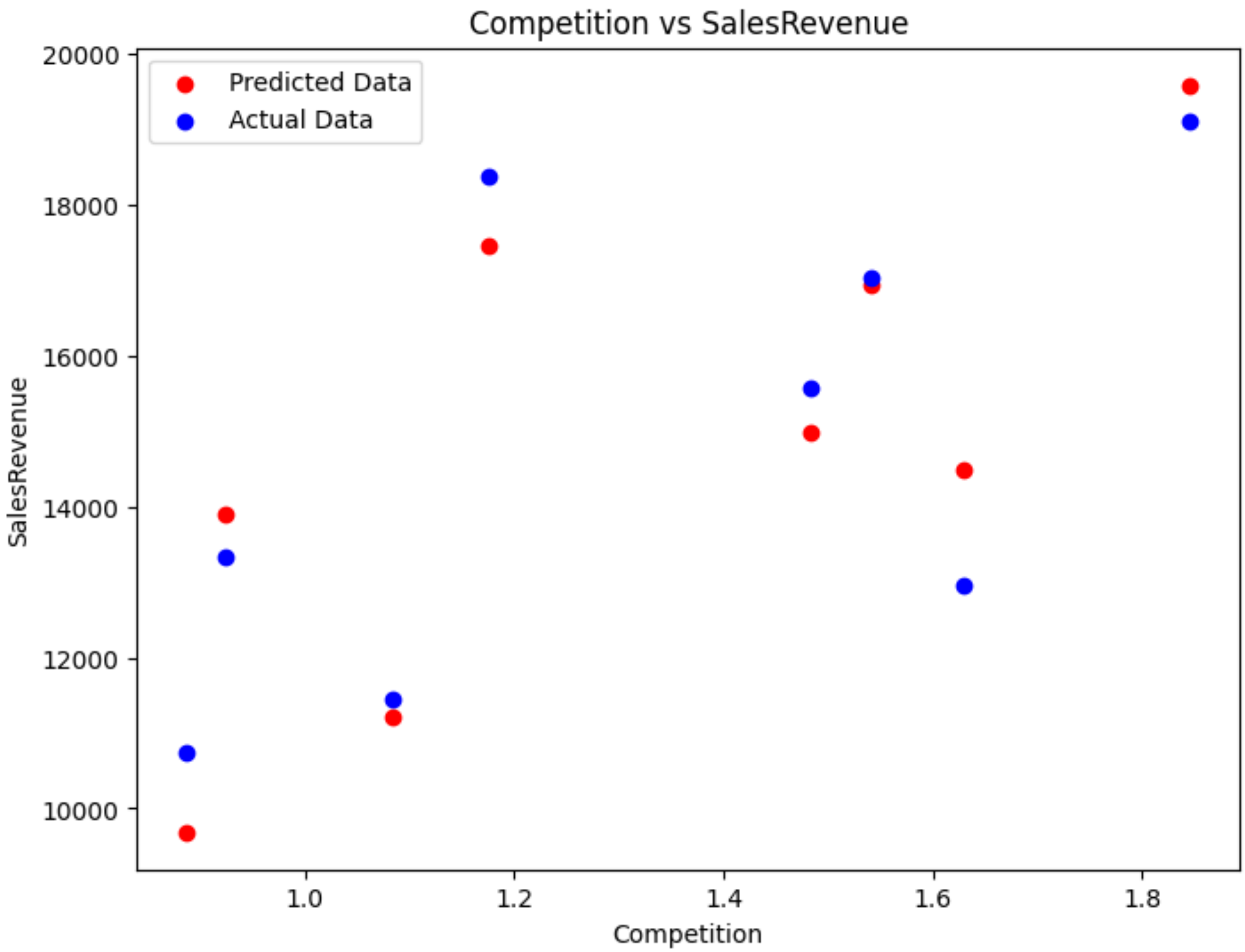
In [35]:
```
coefficients = model.coef_
intercept = model.intercept_
print(f"Coefficient = {coefficients}\nIntercept = {intercept}")
```

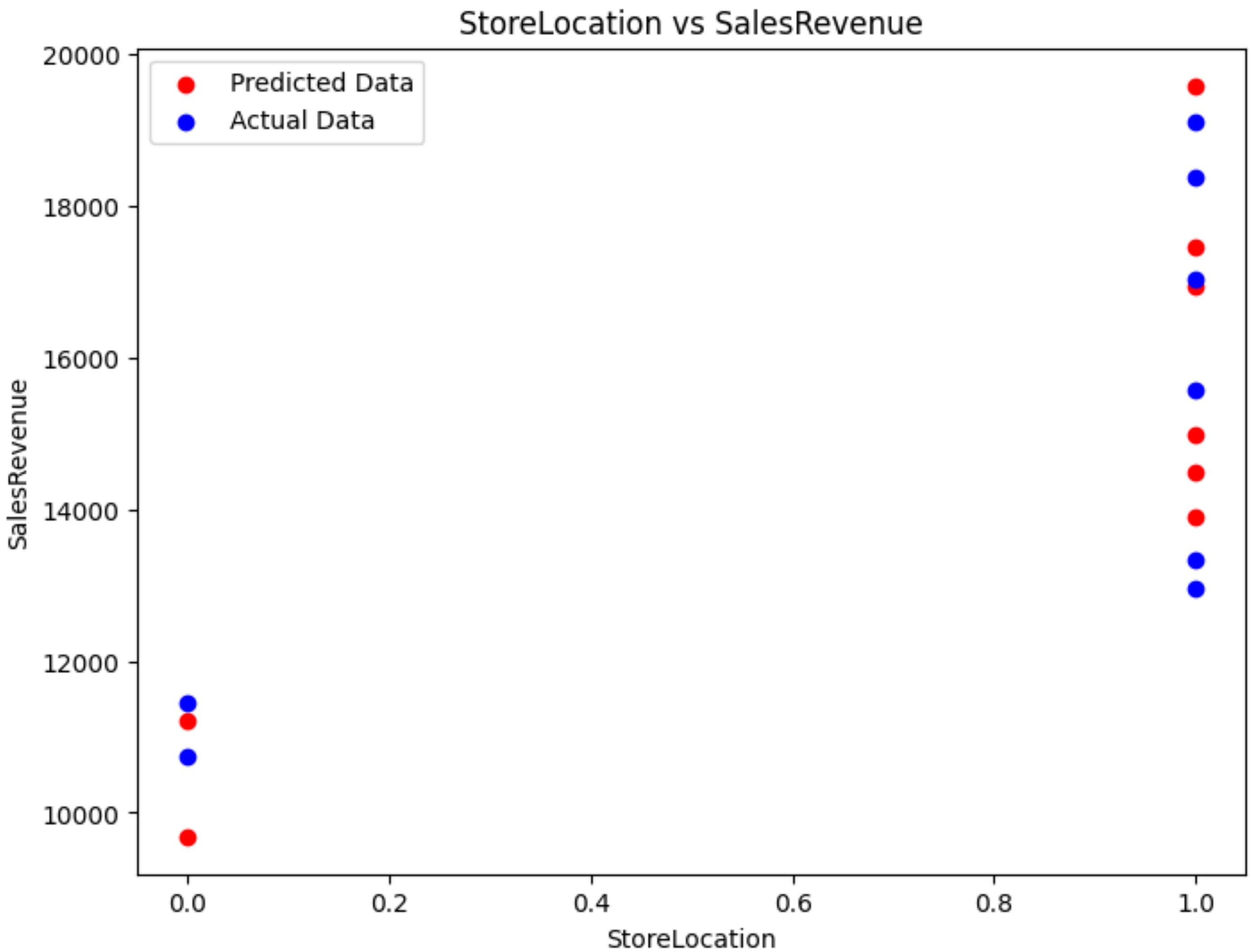
```
Coefficient = [2.11493691e+00 2.27274333e+03 2.19396228e+03]
Intercept = 3176.7913667305384
```

In [36]:
```
Y_pred = model.predict(X_test)
```

In [42]:
```
predictors = ["AdvertisingExpenditure", "Competition", "StoreLocation"]

for predictor in predictors:
    plt.figure(figsize=(8, 6))
    plt.title(f"{predictor} vs SalesRevenue")
    plt.xlabel(predictor)
    plt.ylabel("SalesRevenue")
    plt.scatter(X_test[predictor], Y_pred, color="r", label="Predicted Data")
    plt.scatter(X_test[predictor], Y_test, color="b", label="Actual Data")
    plt.legend()
```




Competition vs SalesRevenue
Predicted Data
Actual Data
SalesRevenue
Competition
20000
18000
16000
14000
12000
10000
1.0
1.2
1.4
1.6
1.8



In [38]:
```
import statsmodels.api as sm
```

In [39]:
```
X_with_const = sm.add_constant(X)
```

In [40]:
```
model = sm.OLS(Y, X_with_const).fit()

for predictor in predictors:
    t_statistic = model.tvalues[predictor]
    p_value_t = model.pvalues[predictor]

    print(f"t-statistic for {predictor} = {t_statistic}")

    if p_value_t < 0.05:
        print(f"{predictor} is a statistically significant predictor of SalesRevenue.")
    else:
        print(f"{predictor} is NOT a statistically significant predictor of
SalesRevenue.")
```

```
t-statistic for AdvertisingExpenditure = 12.738460146150278
AdvertisingExpenditure is a statistically significant predictor of SalesRevenue.
t-statistic for Competition = 5.350557857468894
Competition is a statistically significant predictor of SalesRevenue.
t-statistic for StoreLocation = 4.899145856634402
StoreLocation is a statistically significant predictor of SalesRevenue.
```

In [41]:

```python
X_with_const = sm.add_constant(X[predictor])
model = sm.OLS(Y, X_with_const).fit()

f_statistic = model.fvalue
p_value_f = model.f_pvalue

print(f"F-statistic for {predictor} = {f_statistic}")

if p_value_f < 0.05:
    print(f"{predictor} is a statistically significant predictor of SalesRevenue.")
else:
    print(f"{predictor} is NOT a statistically significant predictor of SalesRevenue.")
```

```
F-statistic for StoreLocation = 44.458964675049536
StoreLocation is a statistically significant predictor of SalesRevenue.
```

Exercise 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.

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

In [2]:
```
df = pd.read_csv("iris_dataset.csv")
df.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
X = df[["sepal_length", "sepal_width", "petal_length", "petal_width"]]
Y = df["target"]
```

In [4]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=42)
```

In [5]:
```
scaler = StandardScaler()

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

In [6]:
```
encoder = LabelEncoder()

Y_train_enc = encoder.fit_transform(Y_train)
Y_test_enc = encoder.transform(Y_test)
```

In [7]:
```
knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train_scaled, Y_train_enc)
```

Out[7]:

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [8]:

```
Y_pred = knn.predict(X_test_scaled)
```

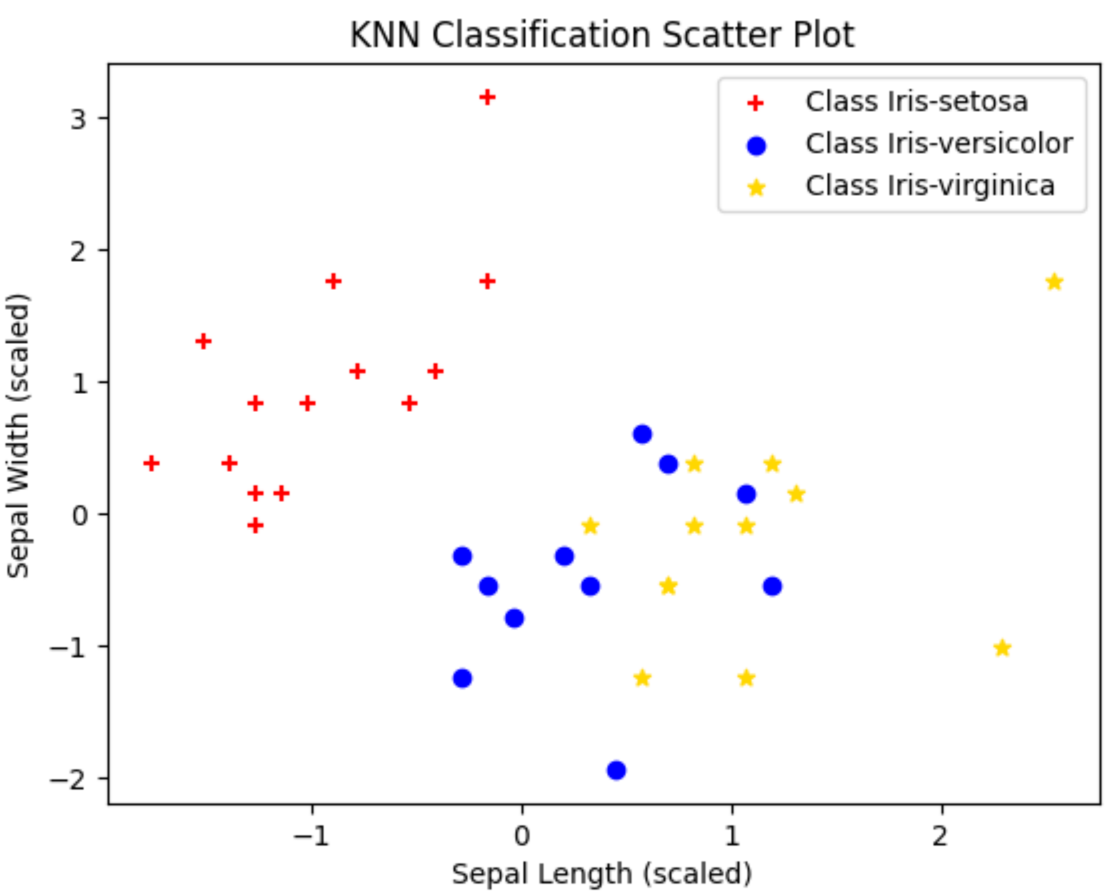
In [9]:

```
accuracy = accuracy_score(Y_test_enc, Y_pred)
print(f"The KNN Classifier is {accuracy * 100:.0f}% accurate")
```

```
The KNN Classifier is 100% accurate
```

In [10]:

```
labels = encoder.classes_

markers = ["+", "o", "*"]
colors = ["red", "blue", "gold"]

for i, label in enumerate(labels):
    class_points = (Y_pred == i)
    plt.scatter(X_test_scaled[class_points, 0], X_test_scaled[class_points, 1],
label=f'Class {label}', marker=markers[i], color=colors[i])
    plt.title("KNN Classification Scatter Plot")
    plt.xlabel("Sepal Length (scaled)")
    plt.ylabel("Sepal Width (scaled)")
    plt.legend()
```



Exercise 6

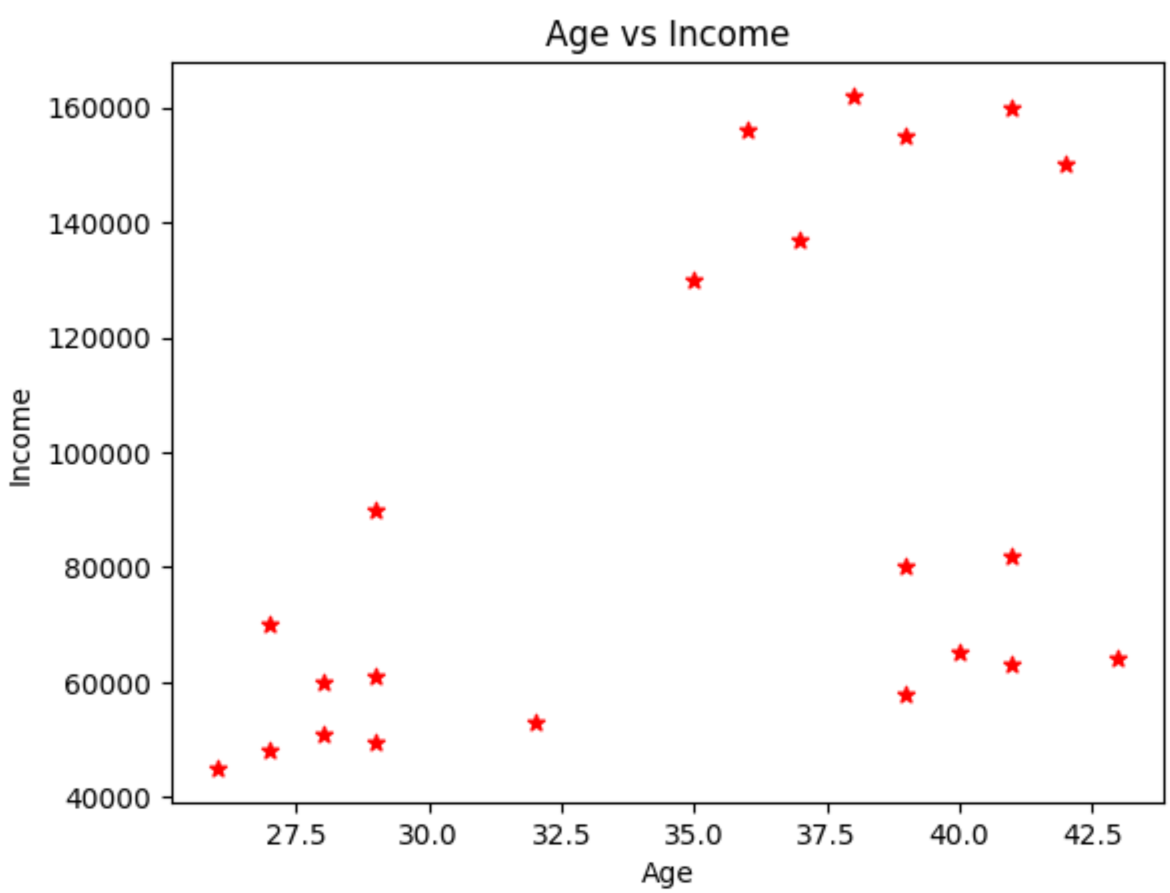
Given a dataset that contains customer information (such as Age, Income, and Spending Score), perform K-means clustering to group customers into clusters. Use visualization chart, plot the data before and after grouping. Also, use the Elbow Method to determine the optimal number of clusters.

In [23]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

In [24]:
```
df = pd.read_csv("income_clustering.csv")

df = df[["Age", "Income($)"]]

scaler = StandardScaler()
sc_df = scaler.fit_transform(df)
```

In [25]:
```
plt.scatter(df["Age"], df["Income($)"], color="r", marker="*")
plt.title("Age vs Income")
plt.xlabel("Age")
plt.ylabel("Income")
```

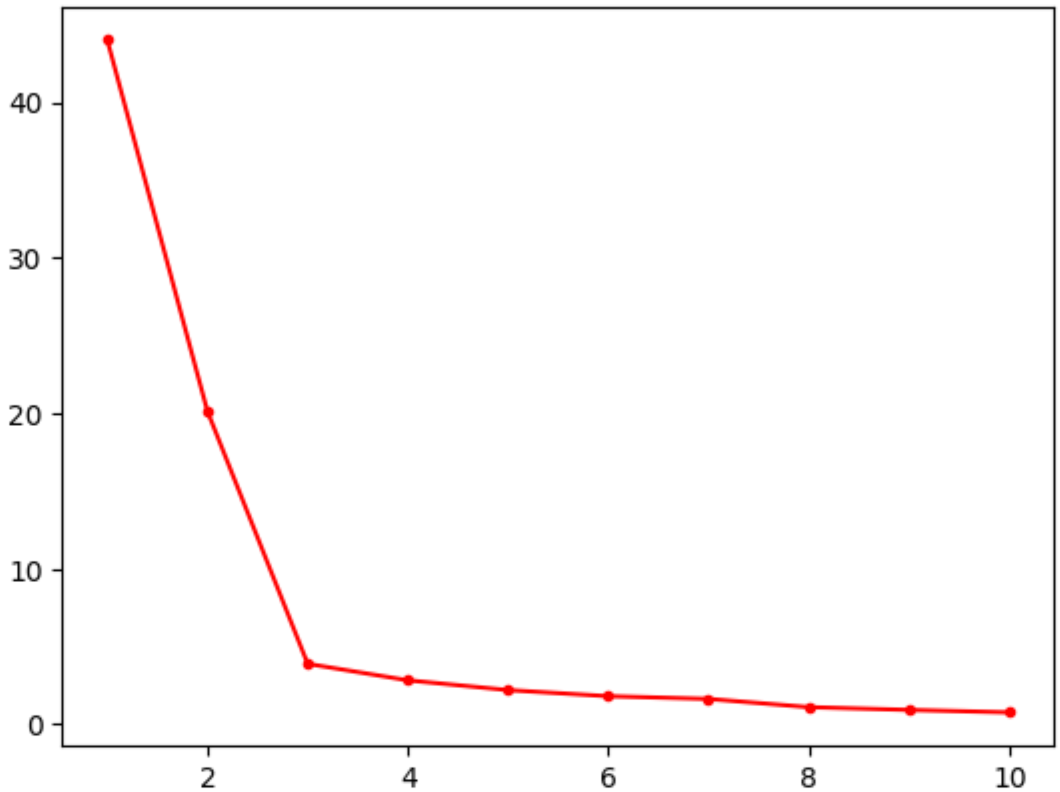
Out[25]: Text(0, 0.5, 'Income')



In [26]:
```
k_range = range(1, 11)
sse = []
for k in k_range:
    kmn = KMeans(n_clusters=k)
    kmn.fit(sc_df)
    sse.append(kmn.inertia_)
```

In [27]:
```
plt.plot(k_range, sse, color="r", marker=".")
```

Out[27]: [<matplotlib.lines.Line2D at 0x71dd15d8b890>]



In [28]:
```
kmn = KMeans(n_clusters=3)
clusters = kmn.fit_predict(sc_df)
```

In [29]:
```
df['clusters'] = clusters
df.head()
```

Out[29]:

	Age	Income($)	clusters
0	27	70000	0
1	29	90000	0
2	29	61000	0
3	28	60000	0
4	42	150000	1

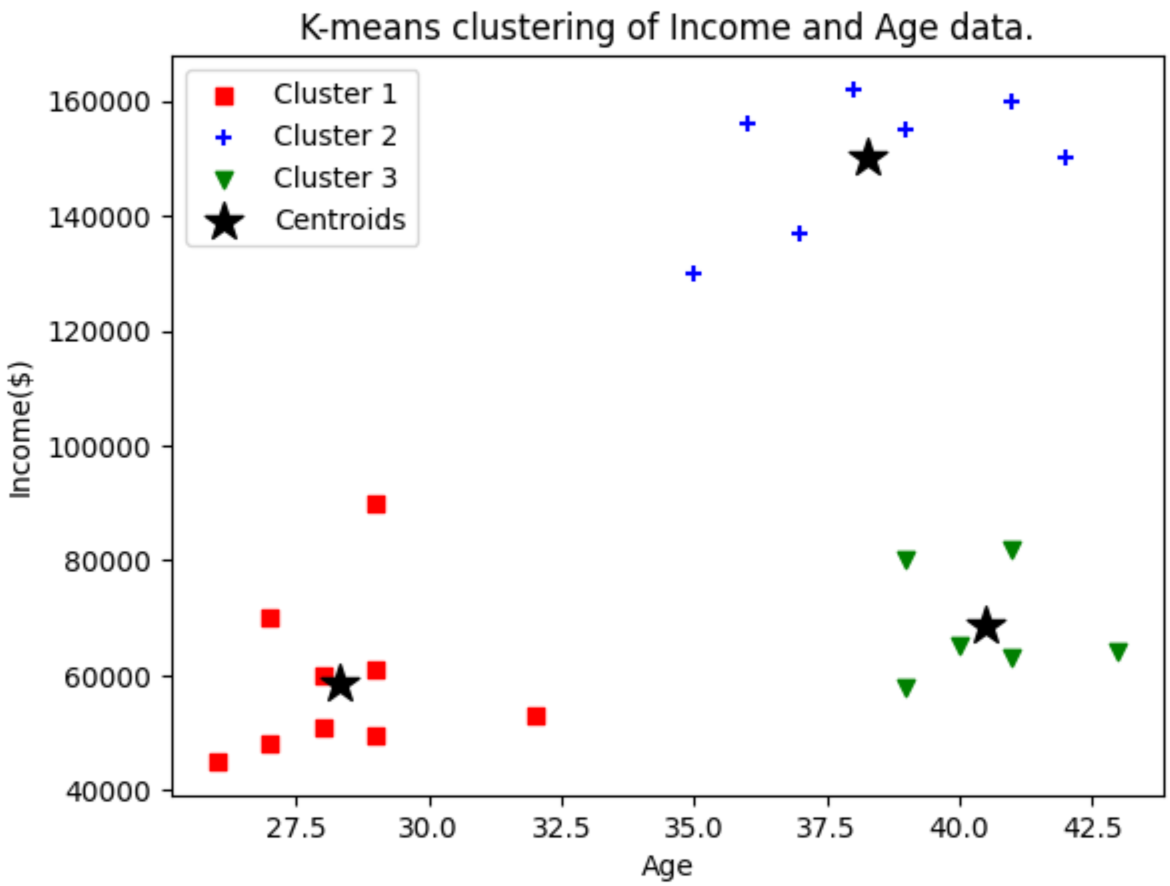
In [30]:
```
cl1 = df[df['clusters'] == 0]
cl2 = df[df['clusters'] == 1]
```

```
cl3 = df[df['clusters'] == 2]

centroids = scaler.inverse_transform(kmn.cluster_centers_)
```

In [31]:

```
plt.title("K-means clustering of Income and Age data.")
plt.xlabel("Age")
plt.ylabel("Income($)")
plt.scatter(cl1['Age'], cl1['Income($)'], color="r", marker="s", label="Cluster 1")
plt.scatter(cl2['Age'], cl2['Income($)'], color="b", marker="+", label="Cluster 2")
plt.scatter(cl3['Age'], cl3['Income($)'], color="g", marker="v", label="Cluster 3")
plt.scatter(centroids[:, 0], centroids[:, 1], label="Centroids", s=200, marker="*",
color="black")
plt.legend()
```

Out[31]: <matplotlib.legend.Legend at 0x71dd15c24cd0>



Exercise 7

Compare the effectiveness of two teaching methods, A and B, in helping students pass a test. Analyse the proportions of passing students, calculate confidence intervals for the difference in proportions, conduct significance tests, and evaluate the area under the ROC curve for predictive accuracy.

In [126…
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
```

In [127…
```
df = pd.read_csv("teaching_methods.csv")
df.head()
```

Out[127…

	Method	Outcome	Study Time
0	A	Pass	10
1	B	Pass	12
2	A	Pass	8
3	A	Pass	6
4	A	Pass	9

In [128…
```
encoder = LabelEncoder()
```

In [129…
```
df["Method"] = encoder.fit_transform(df["Method"])
df["Outcome"] = encoder.fit_transform(df["Outcome"])

df.head()
```

Out[129…

	Method	Outcome	Study Time
0	0	1	10
1	1	1	12
2	0	1	8
3	0	1	6
4	0	1	9

In [130…
```
X = df[["Method", "Study Time"]]
Y = df["Outcome"]
```

In [131…
```
n_A = len(df[df["Method"] == 0])
```

```python
n_B = len(df[df["Method"] == 1])

x_A = len(df[(df["Method"] == 0) & (df["Outcome"] == 1)])
x_B = len(df[(df["Method"] == 1) & (df["Outcome"] == 1)])

p_A = x_A / n_A
p_B = x_B / n_B

print(f"Sample size of students taught by Method A: {n_A}")
print(f"Sample size of students taught by Method B: {n_B}")

print(f"Number of passing students taught by Method A: {x_A}")
print(f"Number of passing students taught by Method B: {x_B}")

print(f"Student Pass Rate with Method A: {p_A}")
print(f"Student Pass Rate with Method B: {p_B}")
```

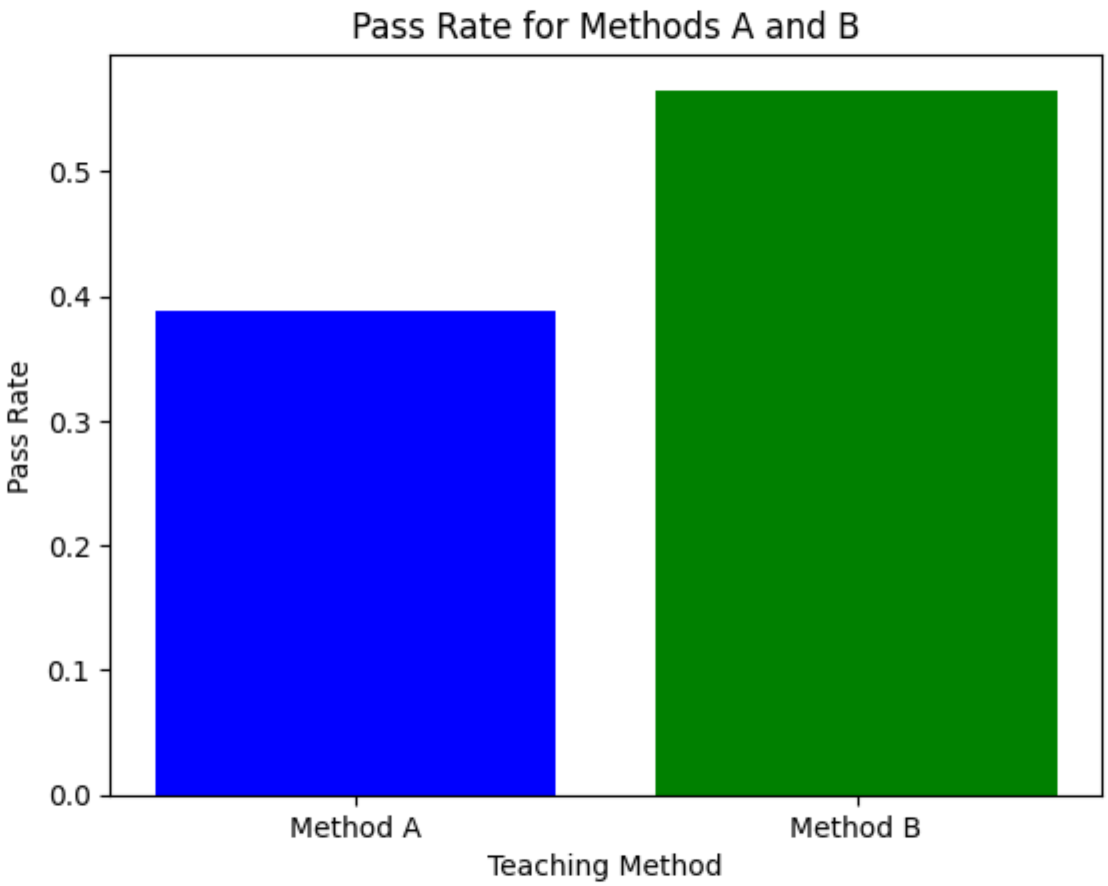
```
Sample size of students taught by Method A: 54
Sample size of students taught by Method B: 46
Number of passing students taught by Method A: 21
Number of passing students taught by Method B: 26
Student Pass Rate with Method A: 0.3888888888888889
Student Pass Rate with Method B: 0.5652173913043478
```

In [132…

```python
methods = ['Method A', 'Method B']
pass_rates = [p_A, p_B]

plt.bar(methods, pass_rates, color=['blue', 'green'])
plt.xlabel('Teaching Method')
plt.ylabel('Pass Rate')
plt.title('Pass Rate for Methods A and B')
```

Out[132…

```
Text(0.5, 1.0, 'Pass Rate for Methods A and B')
```

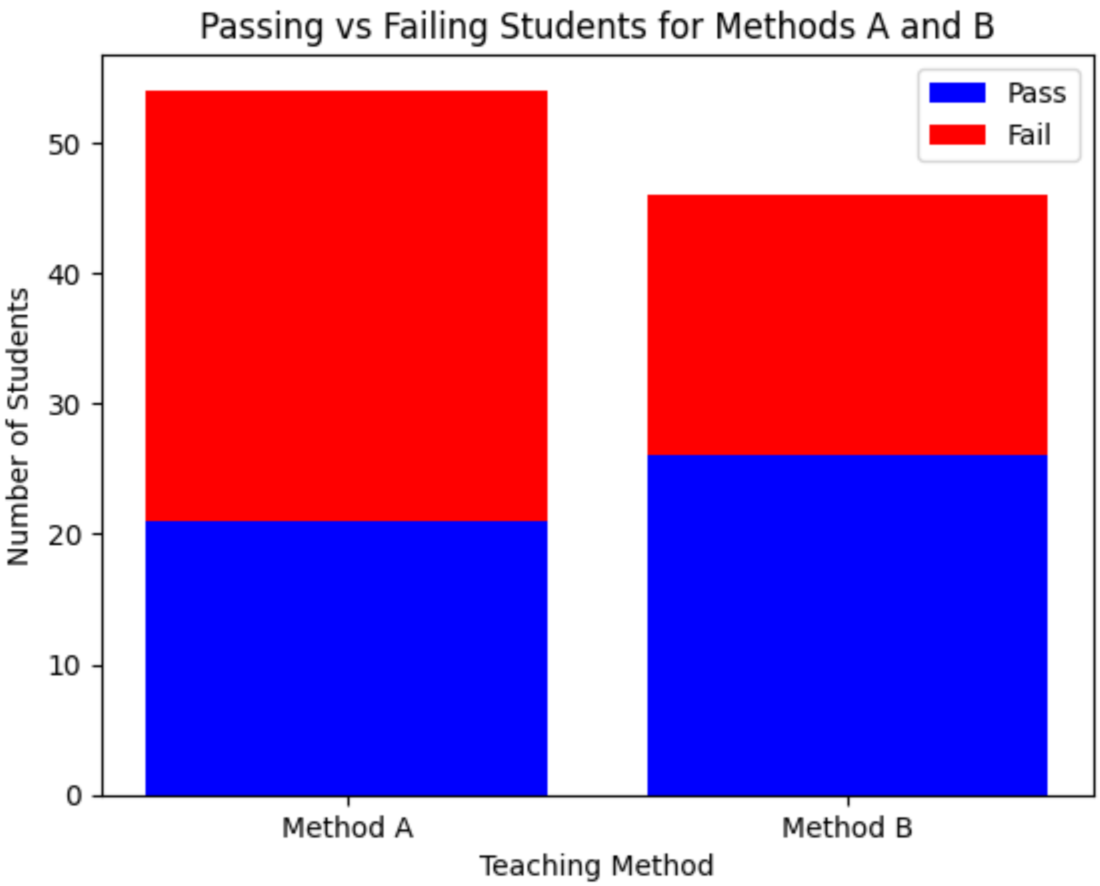


In [133…

```
pass_fail_counts_A = [x_A, n_A - x_A]
pass_fail_counts_B = [x_B, n_B - x_B]

plt.bar(methods[0], pass_fail_counts_A[0], label='Pass', color='blue')
plt.bar(methods[0], pass_fail_counts_A[1], bottom=pass_fail_counts_A[0], label='Fail',
color='red')

plt.bar(methods[1], pass_fail_counts_B[0], color='blue')
plt.bar(methods[1], pass_fail_counts_B[1], bottom=pass_fail_counts_B[0], color='red')

plt.xlabel('Teaching Method')
plt.ylabel('Number of Students')
plt.title('Passing vs Failing Students for Methods A and B')

plt.legend()
```

Out[133…

```
<matplotlib.legend.Legend at 0x7ea662c32990>
```



In [134…
```
from statsmodels.stats.proportion import proportions_ztest, confint_proportions_2indep

ci_low, ci_high = confint_proportions_2indep(x_A, n_A, x_B, n_B)

print(f"The Confidence Interval for difference in proportions is [{ci_low},
{ci_high}]")
```

```
The Confidence Interval for difference in proportions is [-0.35420954540253474, 0.0189285
411289222]
```

In [135…
```
successes = [x_A, x_B]
n_obs = [n_A, n_B]

z_statistic, p_value = proportions_ztest(successes, n_obs)

print(f"z-statistic: {z_statistic}")

if p_value < 0.05:
    print("The proportions are significantly different.")
else:
    print("The proportions are not significantly different.")
```

```
z-statistic: -1.7608057630771965
The proportions are not significantly different.
```

In [136…
```
X = df[["Method", "Study Time"]]
Y = df["Outcome"]
```

In [137…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42,
```

```
test_size=0.25)
```

In [138…

```
model = LogisticRegression()

model.fit(X_train, Y_train)
```

Out[138…

```
▾ LogisticRegression  i ?

LogisticRegression()
```

In [139…

```
Y_prob = model.predict_proba(X_test)[:, 1]

Y_pred = model.predict(X_test)
```

In [140…

```
auc = roc_auc_score(Y_test, Y_prob)

print(f"AUC: {auc:.2f}")
```

```
AUC: 1.00
```

In [141…

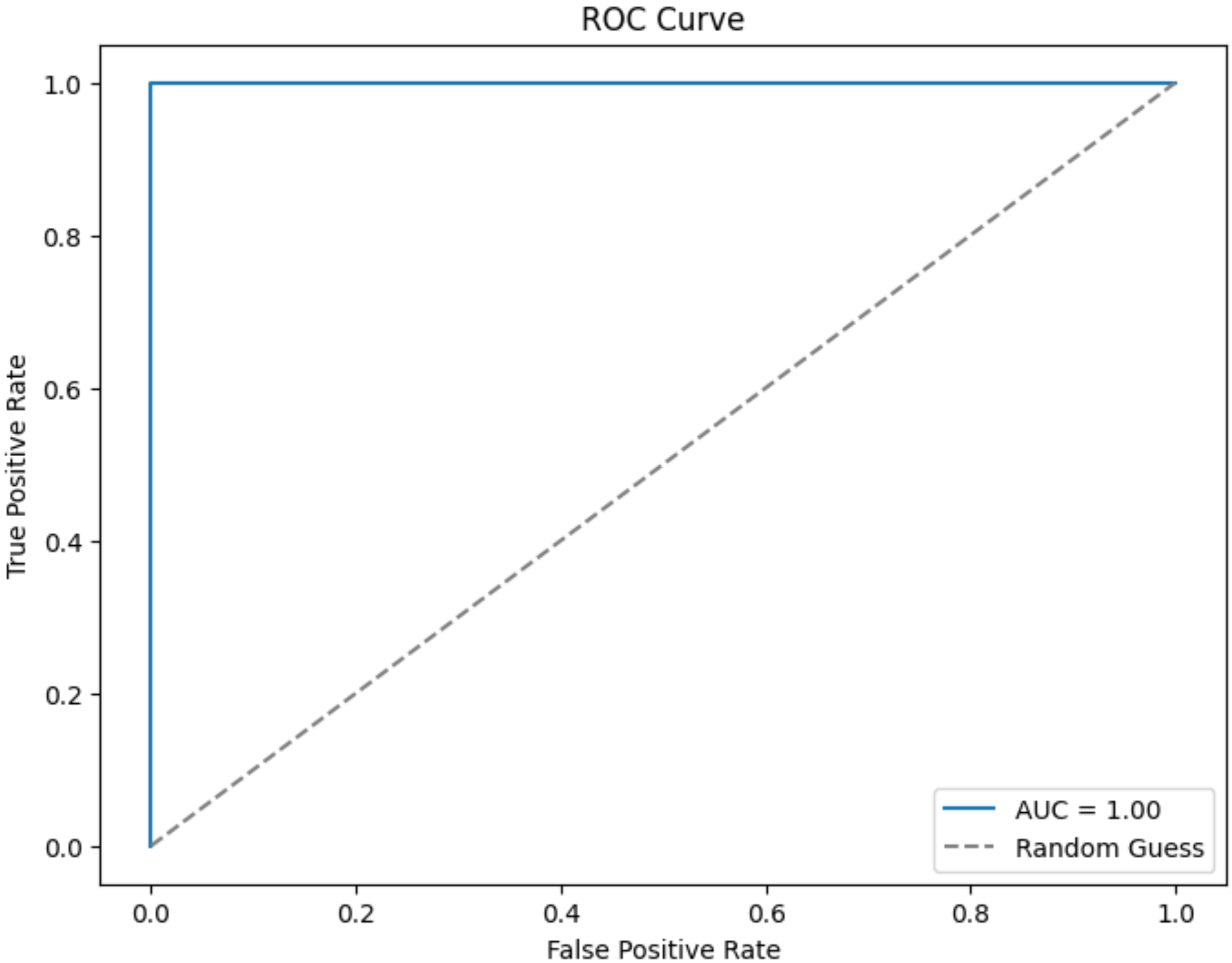
```
fpr, tpr, thresholds = roc_curve(Y_test, Y_prob)
```

In [142…

```
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"AUC = {auc:.2f}")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Random Guess")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
```

Out[142…

```
<matplotlib.legend.Legend at 0x7ea662c92e90>
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 1.00
Random Guess