31**. Scenario**: You work as a data scientist for an e-commerce company that sells a wide range of products online. The company collects vast amounts of data about its customers, including their purchase history, browsing behavior, demographics, and more. The marketing team wants to understand their customer base better and improve their targeted marketing strategies. They have asked you to perform customer segmentation using clustering techniques to identify distinct groups of customers with similar characteristics.

**Question:** Your task is to use Python and clustering algorithms to segment the customers into different groups based on their behavior and characteristics. The marketing team will use these segments to tailor their marketing campaigns and promotions effectively.

**Program:**

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read\_csv('C:/Users/Hello/Desktop/fods/custmer.csv')

features = data[['Purchase History', 'Browsing Behavior', 'Demographics']]

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(features)

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42)

kmeans.fit(scaled\_features)

wcss.append(kmeans.inertia\_)

plt.figure(figsize=(8, 6))

plt.plot(range(1, 11), wcss, marker='o', linestyle='--')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS')

plt.title('Elbow Method')

plt.show()

num\_clusters = 3 # You can change this based on the plot

kmeans = KMeans(n\_clusters=num\_clusters, init='k-means++', random\_state=42)

clusters = kmeans.fit\_predict(scaled\_features)

data['Cluster'] = clusters

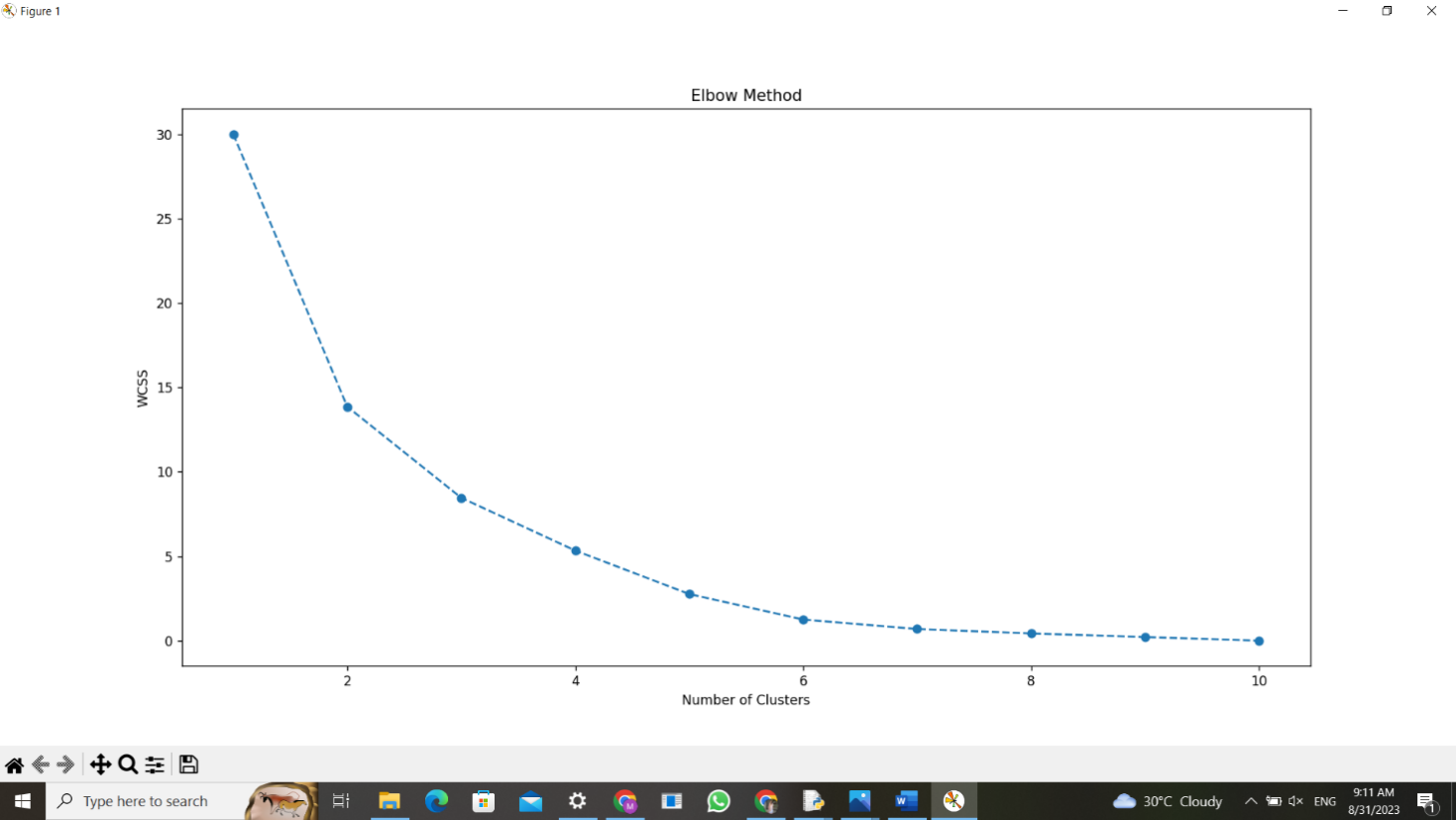
sns.pairplot(data=data, hue='Cluster', diag\_kind='kde')

plt.show()

cluster\_means = data.groupby('Cluster').mean()

print(cluster\_meaans)

**Output:**



32. **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.

**program**

import numpy as np

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

data = {

'HouseSize': [1000, 1500, 1200, 1800, 1350],

'Price': [300000, 450000, 360000, 540000, 405000]

}

df = pd.DataFrame(data)

plt.scatter(df['HouseSize'], df['Price'])

plt.xlabel('House Size (sqft)')

plt.ylabel('Price ($)')

plt.title('House Price vs. House Size')

plt.show()

X = df[['HouseSize']]

y = df['Price']

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)

print(f'Mean Squared Error: {mse}')

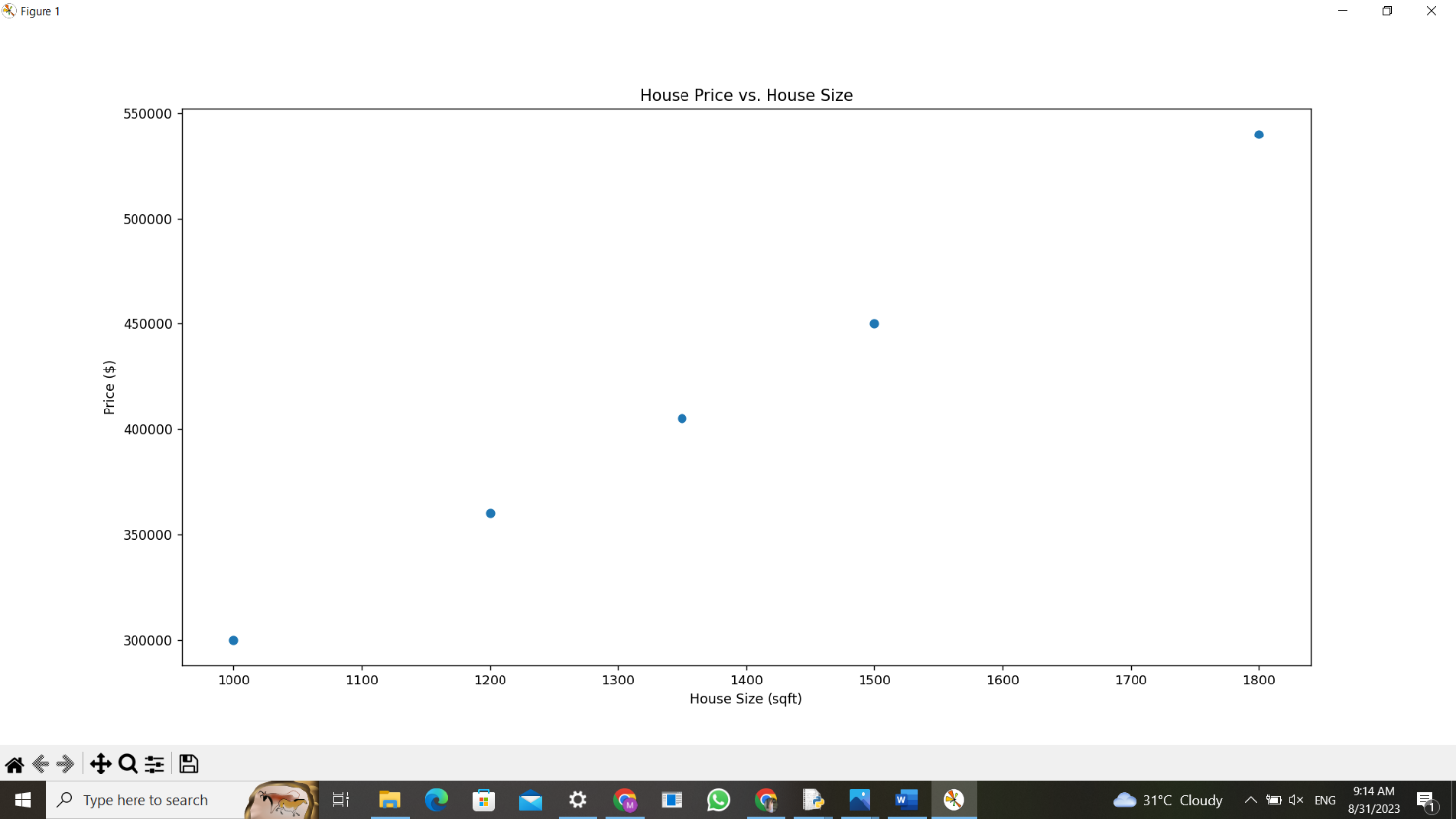
print(f'R-squared: {r2}')

new\_house\_size = 1400

predicted\_price = model.predict(np.array([[new\_house\_size]]))[0]

print(f'Predicted Price for a {new\_house\_size} sqft house: ${predicted\_price:.2f}')

**output**



33. **Scenario:** You work as a data scientist for an automobile company that sells various car models. The company has collected data on different car attributes, such as engine size, horsepower, fuel efficiency, and more, along with their corresponding prices. The marketing team wants to build a predictive model to estimate the price of cars based on their features.

**Question:** Your task is write a Python program that perform linear regression modeling to predict car prices based on a selected set of features from the dataset. Additionally, you need to evaluate the model's performance and provide insights to the marketing team to understand the most influential factors affecting car prices.

**program**

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\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Generate a synthetic dataset (replace this with loading your actual dataset)

np.random.seed(0)

n\_samples = 100

X = np.random.rand(n\_samples, 3) # Features: Engine Size, Horsepower, Fuel Efficiency

coefficients = np.array([5000, 100, -8000]) # Coefficients for the features

intercept = 10000

y = X.dot(coefficients) + intercept + np.random.normal(0, 2000, n\_samples) # Adding noise

# Create a DataFrame

data = pd.DataFrame(data=X, columns=['Engine Size', 'Horsepower', 'Fuel Efficiency'])

data['Price'] = y

# Select features and target variable

selected\_features = ['Engine Size', 'Horsepower', 'Fuel Efficiency']

X = data[selected\_features]

y = data['Price']

# 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 the linear regression model

model = LinearRegression()

# Fit the model to the training data

model.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

# Get the coefficients and intercept

coefficients = model.coef\_

intercept = model.intercept\_

print("Coefficients:", coefficients)

print("Intercept:", intercept)

# Plot actual vs. predicted prices

plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title("Actual vs. Predicted Car Prices")

plt.show()

# Display the most influential factors affecting car prices

feature\_importance = pd.Series(coefficients, index=selected\_features)

sorted\_importance = feature\_importance.abs().sort\_values(ascending=False)

print("Most Influential Factors:")

print(sorted\_importance)

**output**

Mean Squared Error: 4385860.996208939

R-squared: 0.30530368075209824

Coefficients: [ 4737.51953867 -379.84355304 -7825.76608626]

Intercept: 10032.674024941847

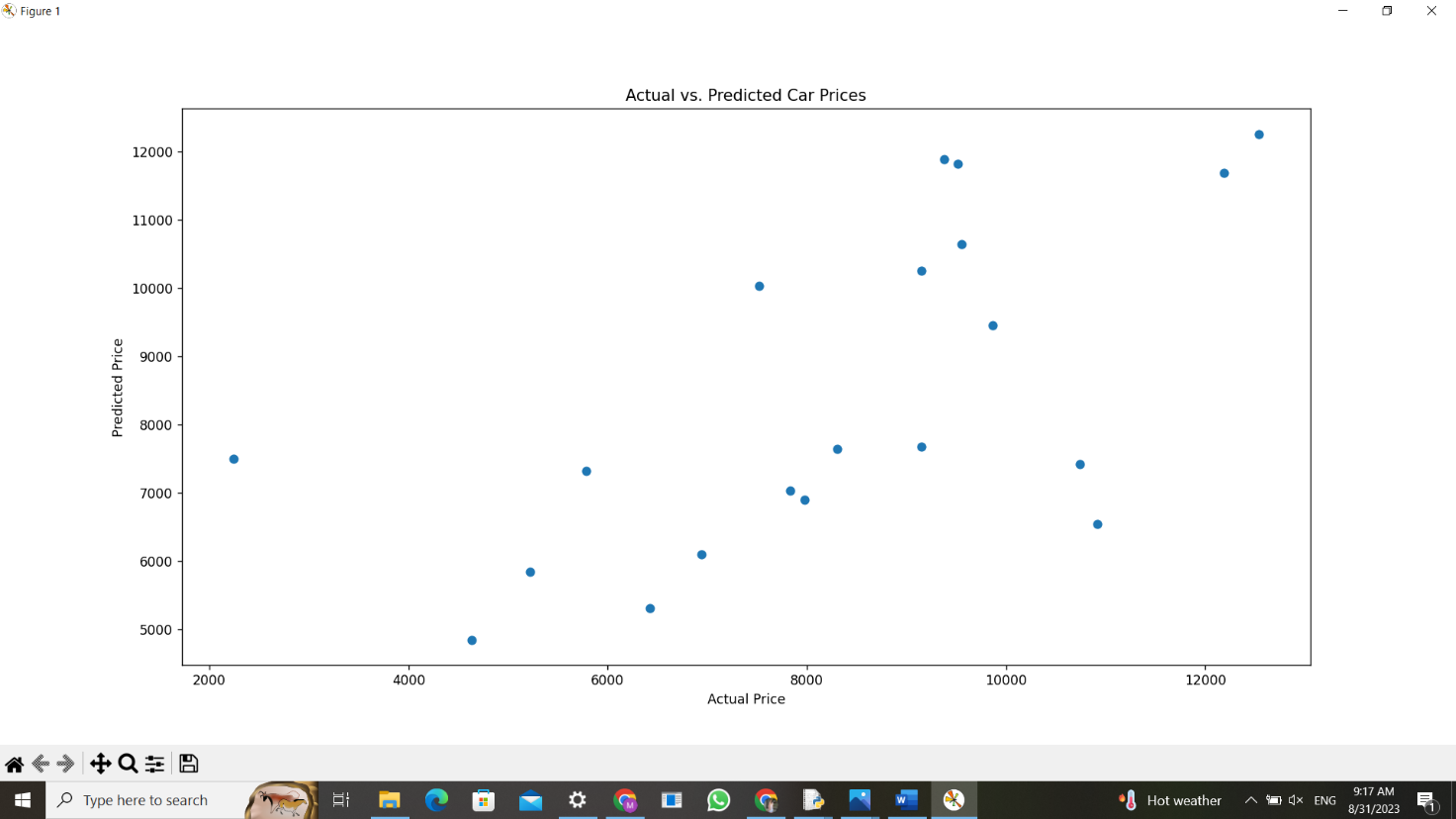
Most Influential Factors:

Fuel Efficiency 7825.766086

Engine Size 4737.519539

Horsepower 379.843553

dtype: float64



34. **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.

**program**

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\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Generate a synthetic dataset (replace this with loading your actual dataset)

np.random.seed(0)

n\_samples = 100

X = np.random.rand(n\_samples, 3) # Features: Engine Size, Horsepower, Fuel Efficiency

coefficients = np.array([5000, 100, -8000]) # Coefficients for the features

intercept = 10000

y = X.dot(coefficients) + intercept + np.random.normal(0, 2000, n\_samples) # Adding noise

# Create a DataFrame

data = pd.DataFrame(data=X, columns=['Engine Size', 'Horsepower', 'Fuel Efficiency'])

data['Price'] = y

# Select features and target variable

selected\_features = ['Engine Size', 'Horsepower', 'Fuel Efficiency']

X = data[selected\_features]

y = data['Price']

# 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 the linear regression model

model = LinearRegression()

# Fit the model to the training data

model.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

# Get the coefficients and intercept

coefficients = model.coef\_

intercept = model.intercept\_

print("Coefficients:", coefficients)

print("Intercept:", intercept)

# Plot actual vs. predicted prices

plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title("Actual vs. Predicted Car Prices")

plt.show()

# Display the most influential factors affecting car prices

feature\_importance = pd.Series(coefficients, index=selected\_features)

sorted\_importance = feature\_importance.abs().sort\_values(ascending=False)

print("Most Influential Factors:")

print(sorted\_importance)

**output**

Mean Squared Error: 4385860.996208939

R-squared: 0.30530368075209824

Coefficients: [ 4737.51953867 -379.84355304 -7825.76608626]

Intercept: 10032.674024941847

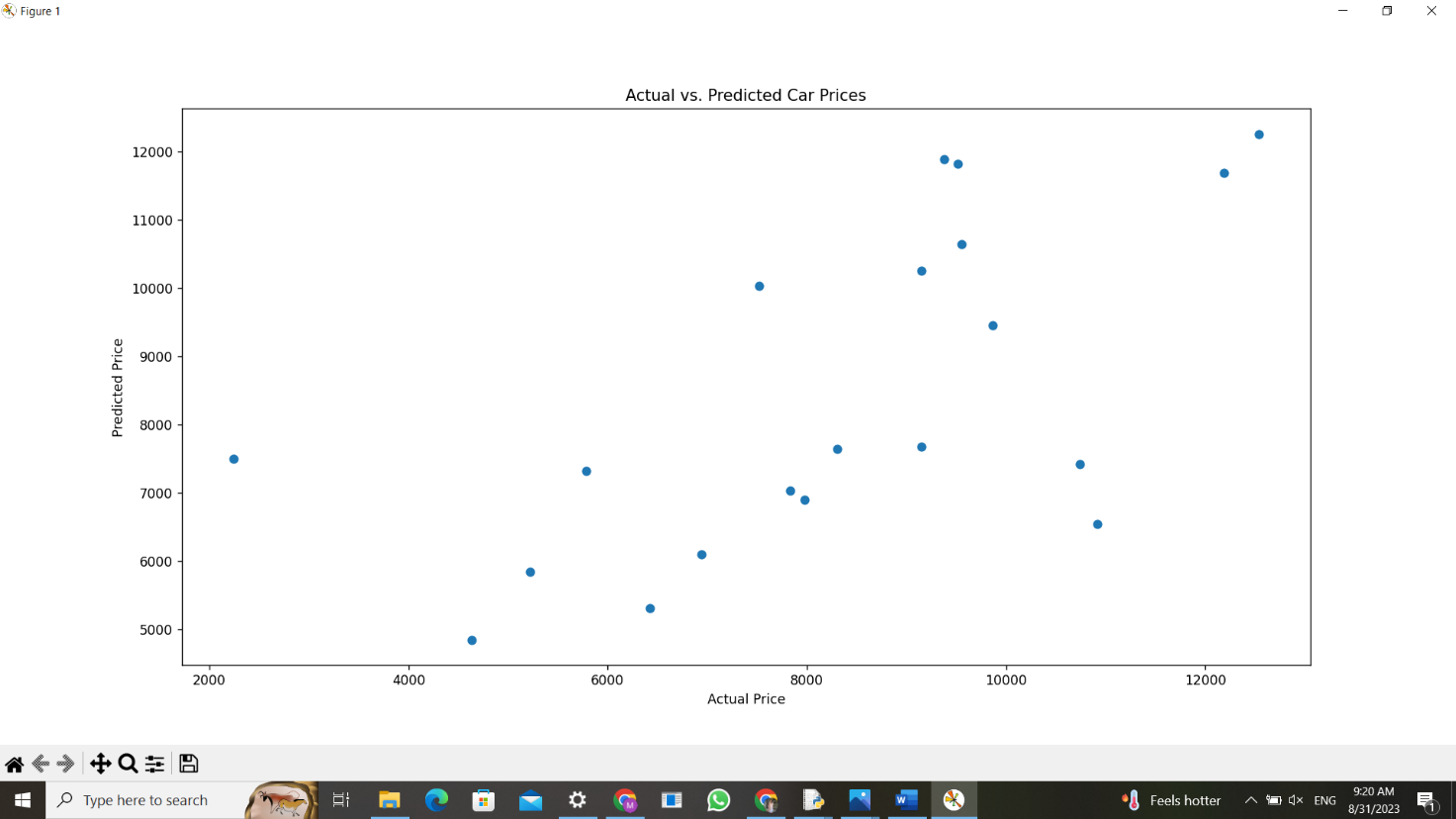
Most Influential Factors:

Fuel Efficiency 7825.766086

Engine Size 4737.519539

Horsepower 379.843553

dtype: float64



35. **Scenario:** You work as a data scientist for a retail company that operates multiple stores. The company is interested in segmenting its customers based on their purchasing behavior to better understand their preferences and tailor marketing strategies accordingly. To achieve this, your team has collected transaction data from different stores, which includes customer IDs, the total amount spent in each transaction, and the frequency of visits.

**Question:** Your task is to build a clustering model using the K-Means algorithm to group customers into distinct segments based on their spending patterns.

**program**

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

data = {

'CustomerID': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110],

'AmountSpent': [500, 100, 1200, 800, 150, 3000, 2000, 700, 900, 400],

'Frequency': [10, 5, 15, 12, 2, 20, 18, 8, 10, 6]

}

df = pd.DataFrame(data)

X = df[['AmountSpent', 'Frequency']]

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

num\_clusters = 3

kmeans = KMeans(n\_clusters=num\_clusters, random\_state=42)

kmeans.fit(X\_scaled)

df['Cluster'] = kmeans.labels\_

plt.scatter(df['AmountSpent'], df['Frequency'], c=df['Cluster'], cmap='rainbow')

plt.xlabel('Amount Spent')

plt.ylabel('Frequency')

plt.title('Customer Segmentation')

plt.show()

print(df)

**output**

CustomerID AmountSpent Frequency Cluster

0 101 500 10 0

1 102 100 5 2

2 103 1200 15 0

3 104 800 12 0

4 105 150 2 2

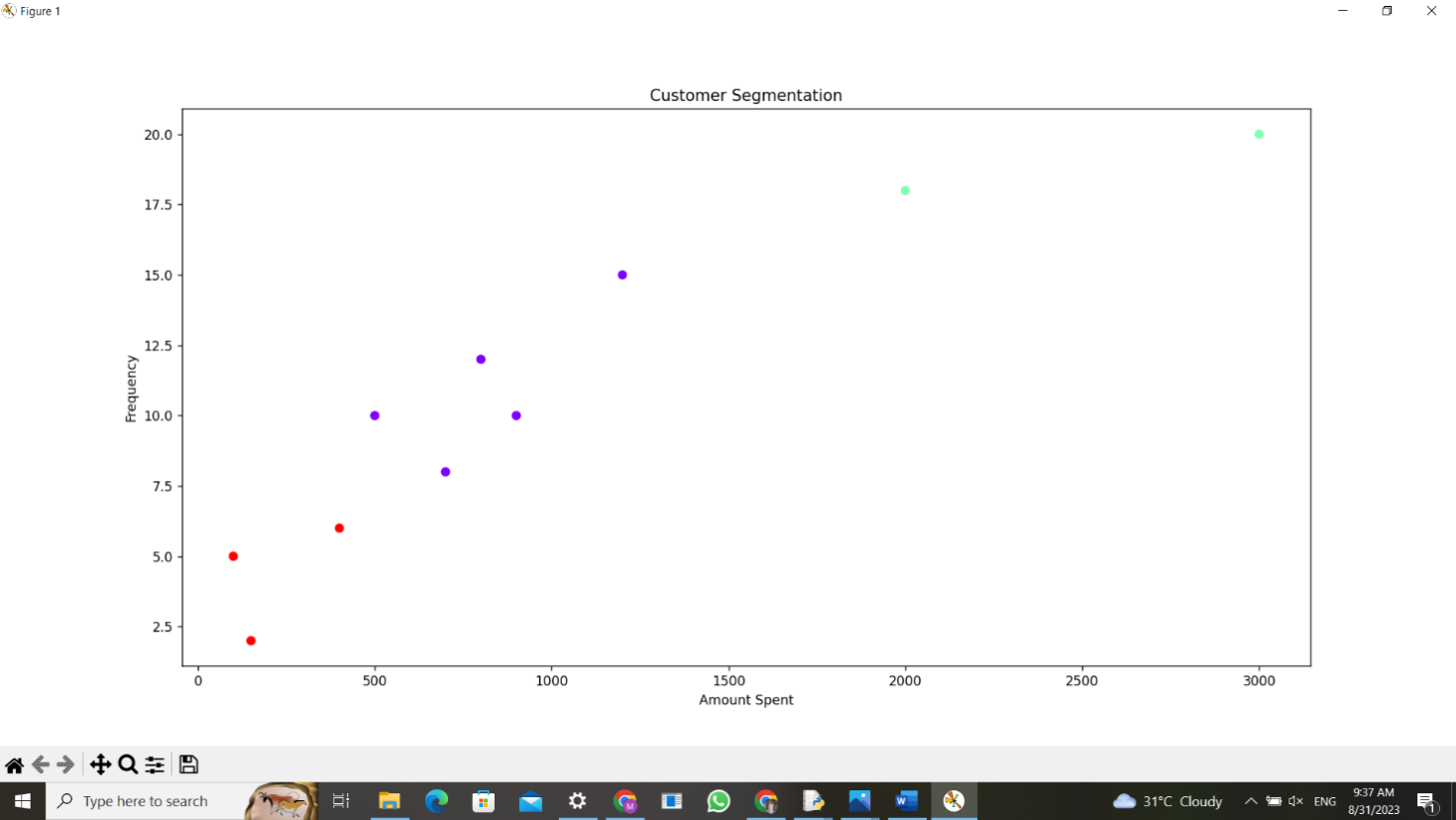
5 106 3000 20 1

6 107 2000 18 1

7 108 700 8 0

8 109 900 10 0

9 110 400 6 2



36. **Scenario:** You are a data analyst working for a finance company. Your team is interested in analyzing the variability of stock prices for a particular company over a certain period. The company's stock data includes the closing prices for each trading day of the specified period.

**Question:** Your task is to build a Python program that reads the stock data from a CSV file, calculates the variability of stock prices, and provides insights into the stock's price movements.

**Program**

import pandas as pd

import numpy as np

data = pd.read\_csv('C:/Users/Hello/Downloads/stock\_data.csv')

data['Price Change'] = data['Close Price'].diff()

mean\_change = data['Price Change'].mean()

std\_dev\_change = data['Price Change'].std()

coefficient\_of\_variation = std\_dev\_change / mean\_change

if coefficient\_of\_variation < 0.1:

insight = "The stock has relatively low variability in price."

elif coefficient\_of\_variation < 0.3:

insight = "The stock has moderate variability in price."

else:

insight = "The stock has high variability in price."

print("Mean Price Change:", mean\_change)

print("Standard Deviation of Price Change:", std\_dev\_change)

print("Coefficient of Variation:", coefficient\_of\_variation)

print("Insight:", insight)

**output**

Mean Price Change: 0.618421052631579

Standard Deviation of Price Change: 2.5476683526486923

Coefficient of Variation: 4.119633931942566

Insight: The stock has high variability in price.

37. **Scenario:** You are a data scientist working for an educational institution, and you want to explore the correlation between students' study time and their exam scores. You have collected data from a group of students, noting their study time in hours and their corresponding scores in an exam.

**Question:** Identify any potential correlation between study time and exam scores and explore various plotting functions to visualize this relationship effectively.

Program

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Generate example data (replace this with your actual data)

np.random.seed(42)

num\_students = 50

study\_time = np.random.randint(1, 6, size=num\_students)

exam\_scores = study\_time \* 10 + np.random.normal(0, 5, size=num\_students)

# Create a DataFrame from the generated data

data = pd.DataFrame({'Study Time': study\_time, 'Exam Score': exam\_scores})

# Calculate the correlation coefficient

correlation = data['Study Time'].corr(data['Exam Score'])

# Print the correlation coefficient

print(f"Correlation coefficient: {correlation:.2f}")

# Create scatter plot using Matplotlib

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

plt.scatter(data['Study Time'], data['Exam Score'])

plt.title('Study Time vs Exam Score')

plt.xlabel('Study Time (hours)')

plt.ylabel('Exam Score')

plt.grid()

plt.show()

# Create scatter plot with regression line using Seaborn

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

sns.regplot(x='Study Time', y='Exam Score', data=data)

plt.title('Study Time vs Exam Score')

plt.xlabel('Study Time (hours)')

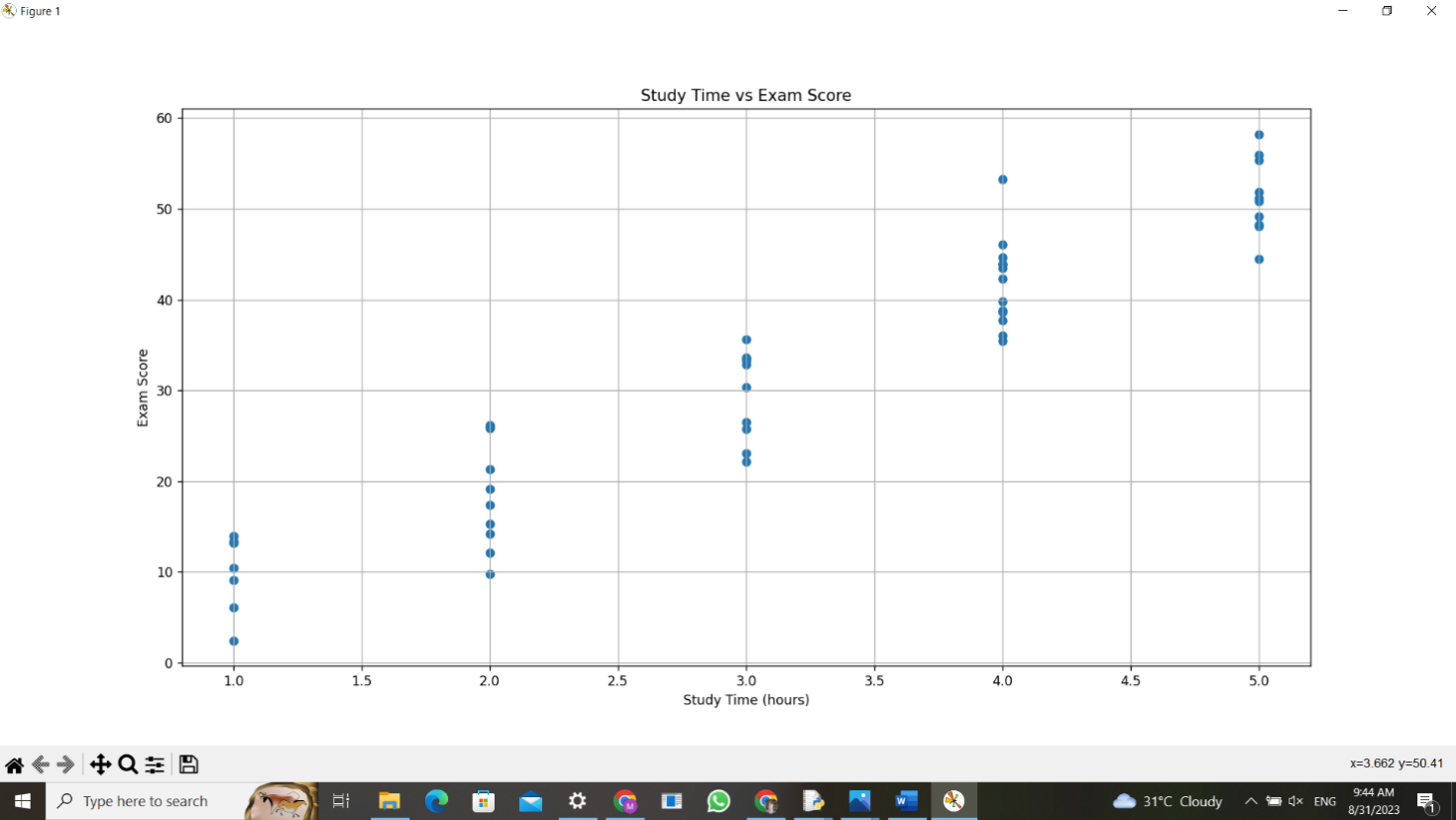
plt.ylabel('Exam Score')

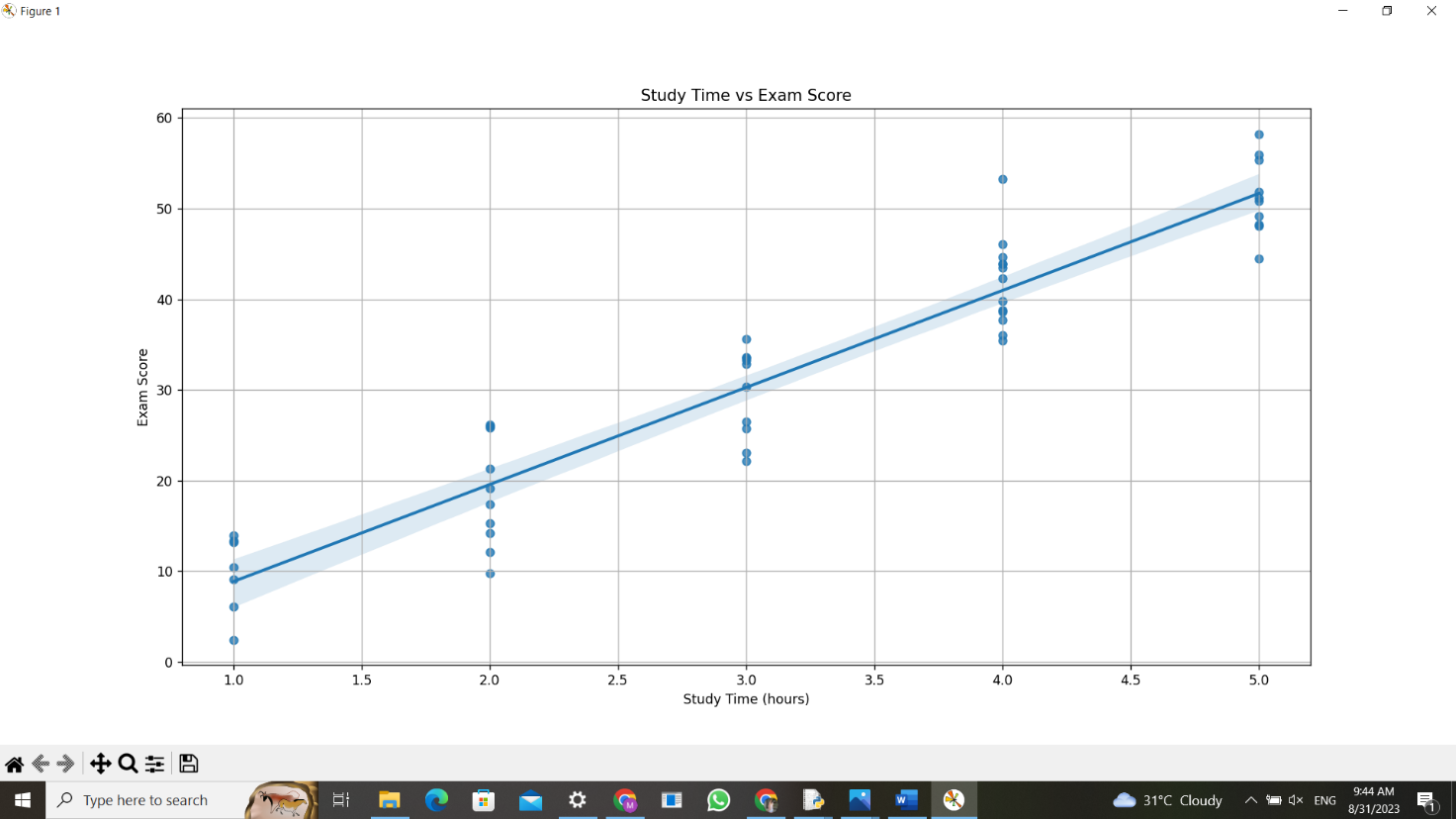
plt.grid()

plt.show()

**output**

Correlation coefficient: 0.95





38**. Scenario:** You work for a weather data analysis company, and your team is responsible for developing a program to calculate and analyze variability in temperature data for different cities.

**Question:** Writea python program will take in a dataset containing daily temperature readings for each city over a year and perform the following tasks:

1. Calculate the mean temperature for each city.
2. Calculate the standard deviation of temperature for each city.
3. Determine the city with the highest temperature range (difference between the highest and lowest temperatures).
4. Find the city with the most consistent temperature (the lowest standard deviation).

**Program**

import pandas as pd

data = pd.read\_csv("C:/Users/Hello/Desktop/fods/temp.csv")

# Calculate the mean temperature for each city

mean\_temperatures = data.groupby('City')['Temperature'].mean()

# Calculate the standard deviation of temperature for each city

std\_dev\_temperatures = data.groupby('City')['Temperature'].std()

# Determine the city with the highest temperature range

temperature\_ranges = data.groupby('City')['Temperature'].agg(lambda x: max(x) - min(x))

city\_with\_highest\_range = temperature\_ranges.idxmax()

highest\_range\_value = temperature\_ranges.max()

# Find the city with the most consistent temperature (lowest standard deviation)

city\_with\_lowest\_std = std\_dev\_temperatures.idxmin()

lowest\_std\_value = std\_dev\_temperatures.min()

# Print the results

print("Mean Temperatures:")

print(mean\_temperatures)

print("\nStandard Deviations of Temperatures:")

print(std\_dev\_temperatures)

print("\nCity with the Highest Temperature Range:")

print(f"{city\_with\_highest\_range} (Range: {highest\_range\_value:.2f}°C)")

print("\nCity with the Most Consistent Temperature:")

print(f"{city\_with\_lowest\_std} (Standard Deviation: {lowest\_std\_value:.2f}°C)")

**output**

Mean Temperatures:

City

Chicago 10.0

Los Angeles 23.2

New York 18.8

Name: Temperature, dtype: float64

Standard Deviations of Temperatures:

City

Chicago 1.581139

Los Angeles 1.303840

New York 2.588436

Name: Temperature, dtype: float64

City with the Highest Temperature Range:

New York (Range: 7.00°C)

City with the Most Consistent Temperature:

Los Angeles (Standard Deviation: 1.30°C)

39. **Scenario:** You work as a data scientist for a marketing agency, and one of your clients is a large e-commerce company. The company wants to understand the purchasing behavior of its customers and segment them into different groups based on their buying patterns. The e-commerce company has provided you with transaction data, including customer IDs, the total amount spent in each transaction, and the number of items purchased.

**Question:** Build a clustering model using the K-Means algorithm to group customers based on their spending and purchase behavior and visualize the clusters using scatter plots or other appropriate visualizations to gain insights into customer distribution and distinguish different segments.

**Program**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Generate example data (replace this with your actual data)

np.random.seed(42)

num\_customers = 200

total\_amounts = np.random.randint(20, 300, size=num\_customers)

num\_items = np.random.randint(1, 15, size=num\_customers)

# Create a DataFrame from the generated data

data = pd.DataFrame({'Customer ID': range(1, num\_customers + 1),

'Total Amount': total\_amounts,

'Number of Items': num\_items})

# Select features for clustering

X = data[['Total Amount', 'Number of Items']]

# Standardize the features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Choose the number of clusters

num\_clusters = 3

# Build the K-Means model

kmeans = KMeans(n\_clusters=num\_clusters, random\_state=42)

cluster\_labels = kmeans.fit\_predict(X\_scaled)

# Add the cluster labels to the DataFrame

data['Cluster'] = cluster\_labels

# Visualize the clusters using scatter plots

plt.figure(figsize=(10, 8))

colors = ['r', 'g', 'b']

for cluster in range(num\_clusters):

plt.scatter(data[data['Cluster'] == cluster]['Total Amount'],

data[data['Cluster'] == cluster]['Number of Items'],

color=colors[cluster], label=f'Cluster {cluster}')

plt.title('Customer Clusters')

plt.xlabel('Total Amount')

plt.ylabel('Number of Items')

plt.legend()

plt.show()

# Interpretation of the clusters

for cluster in range(num\_clusters):

cluster\_data = data[data['Cluster'] == cluster]

avg\_amount = cluster\_data['Total Amount'].mean()

avg\_items = cluster\_data['Number of Items'].mean()

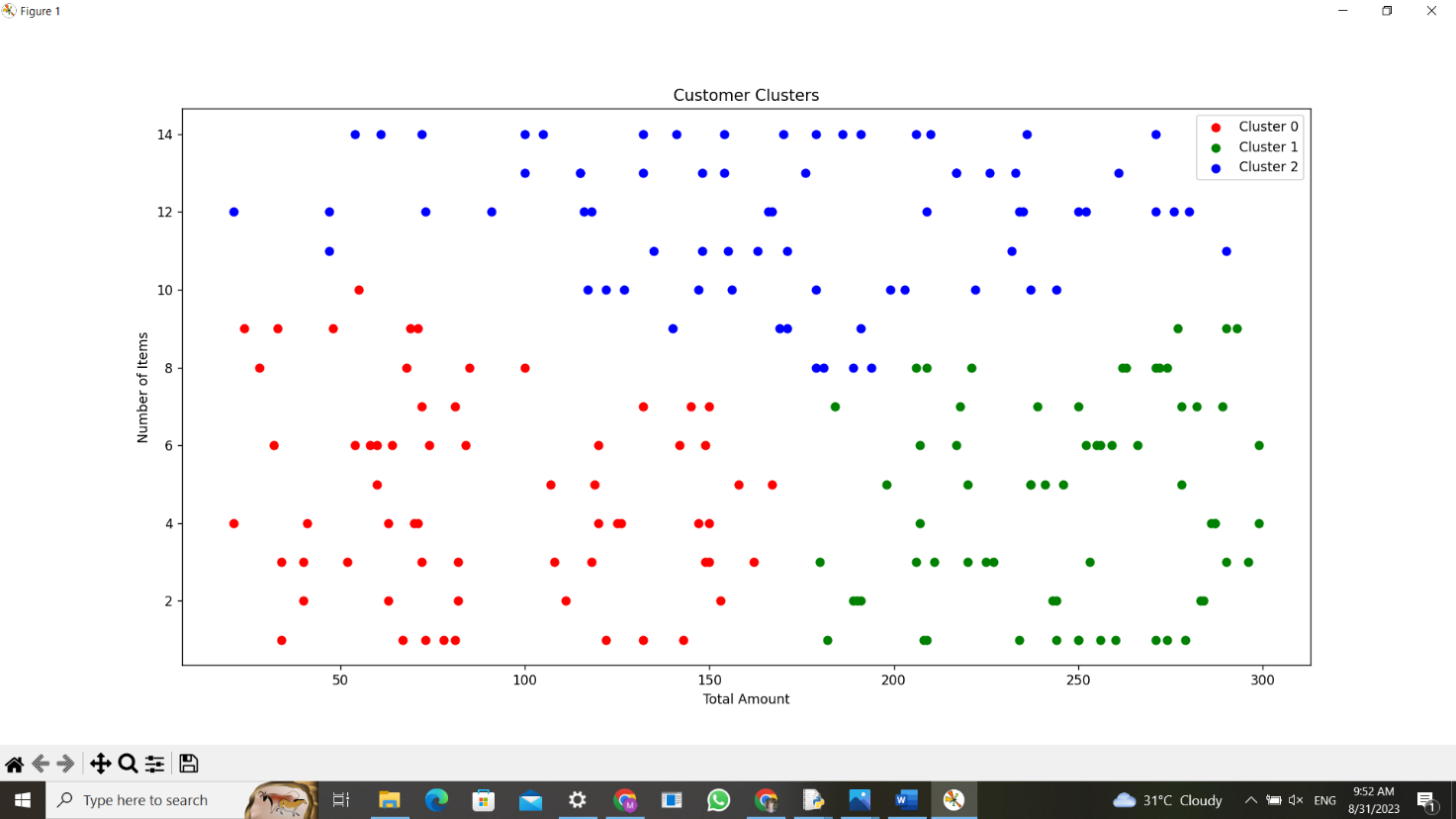
print(f"Cluster {cluster} - Average Amount: {avg\_amount:.2f}, Average Items: {avg\_items:.2f}")

**output**

Cluster 0 - Average Amount: 90.30, Average Items: 4.68

Cluster 1 - Average Amount: 246.38, Average Items: 4.48

Cluster 2 - Average Amount: 170.08, Average items:11.80



40. **Scenario:** You are a data analyst working for a sports analytics company. The company has collected data on various soccer players, including their names, ages, positions, number of goals scored, and weekly salaries. Create dataset on your own and store in a CSV file.

**Question:** Develop a Python program to read the data from the CSV file into a pandas data frame, to find the top 5 players with the highest number of goals scored and the top 5 players with the highest salaries. Also calculate the average age of players and display the names of players who are above the average age and visualize the distribution of players based on their positions using a bar chart.

**Program**

import pandas as pd

data = pd.read\_csv('C:/Users/Hello/Desktop/fods/soccer.csv')

top\_goal\_scorers = data.nlargest(5, 'Goals')

top\_earners = data.nlargest(5, 'Salary')

average\_age = data['Age'].mean()

above\_avg\_age\_players = data[data['Age'] > average\_age]

print("Top 5 Goal Scorers:")

print(top\_goal\_scorers)

print("\nTop 5 Earners:")

print(top\_earners)

print("\nAverage Age of Players:", average\_age)

print("\nPlayers Above Average Age:")

print(above\_avg\_age\_players)

import matplotlib.pyplot as plt

position\_distribution = data['Position'].value\_counts()

position\_distribution.plot(kind='bar')

plt.xlabel('Position')

plt.ylabel('Number of Players')

plt.title('Distribution of Players Based on Positions')

plt.show()

**output**

Top 5 Goal Scorers:

Name Age Position Goals Salary

6 Player G 29 Forward 30 150000

3 Player D 30 Forward 25 120000

0 Player A 25 Forward 20 100000

9 Player J 23 Forward 18 95000

4 Player E 27 Midfielder 15 90000

Top 5 Earners:

Name Age Position Goals Salary

6 Player G 29 Forward 30 150000

3 Player D 30 Forward 25 120000

0 Player A 25 Forward 20 100000

9 Player J 23 Forward 18 95000

4 Player E 27 Midfielder 15 90000

Average Age of Players: 26.5

Players Above Average Age:

Name Age Position Goals Salary

1 Player B 28 Midfielder 10 80000

3 Player D 30 Forward 25 120000

4 Player E 27 Midfielder 15 90000

6 Player G 29 Forward 30 150000

8 Player I 31 Defender 1 50000

