**DSA0415 -  Fundamentals of Data Science - Lab Questions**

1. **Scenario:** You are working on a project that involves analyzing student performance data for a class of 10 students. The data is stored in a NumPy array named student\_scores, where each row represents a student and each column represents a different subject. The subjects are arranged in the following order: Math, Science, English, and History. Your task is to calculate the average score for each subject and identify the subject with the highest average score.

**Question:** How would you use NumPy arrays to calculate the average score for each subject and determine the subject with the highest average score? Assume 4x4 matrix that stores marks of each student in given order.

**Program :**

**import numpy as np**

**student\_scores=np.array([**

**[90,88,91,92],**

**[46,88,80,90],**

**[77,88,90,78],**

**[78,88,87,89]])**

**avg=np.mean(student\_scores,axis=0)**

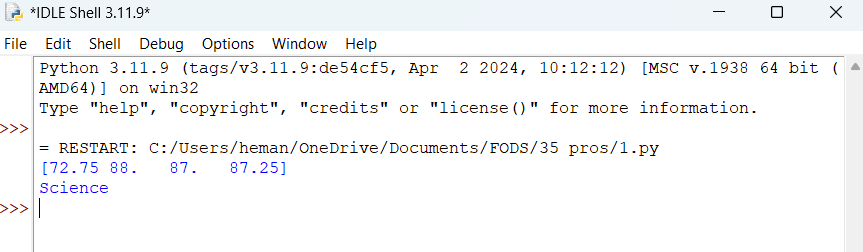
**high=np.argmax(avg)**

**highest= ['Math', 'Science', 'English', 'History'][high]**

**print(avg)**

**print(highest)**

**OUTPUT :**

****

2. **Scenario:** You are a data analyst working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a NumPy array.

**Question:** How would you find the average price of all the products sold in the past month? Assume 3x3 matrix with each row representing the sales for a different product

**PROGRAM:**

**import numpy as np**

**sales\_data = np.array([**

**[100, 120, 150],**

**[80, 90, 110],**

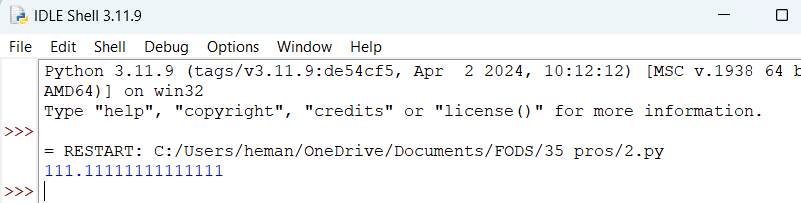
**[130, 100, 120]**

**])**

**average\_price = np.mean(sales\_data)**

**print( average\_price)**

**OUTPUT :**

****

3. **Scenario:** You are working on a project that involves analyzing a dataset containing information about houses in a neighborhood. The dataset is stored in a CSV file, and you have imported it into a NumPy array named house\_data. Each row of the array represents a house, and the columns contain various features such as the number of bedrooms, square footage, and sale price.

**Question:** Using NumPy arrays and operations, how would you find the average sale price of houses with more than four bedrooms in the neighborhood?

**PROGRAM :**

**import numpy as np**

**import pandas as pd**

**house\_data = np.array([**

**[4, 2000, 350000],**

**[3, 1800, 300000],**

**[5, 2200, 400000],**

**[4, 2400, 380000],**

**[6, 2800, 450000]**

**])**

**bedroom\_column = 0**

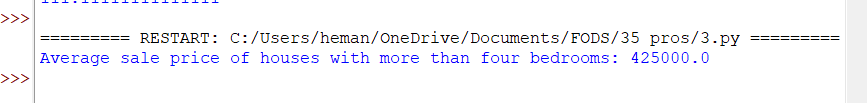
**sale\_price\_column = -1**

**houses\_more\_than\_four\_bedrooms = house\_data[house\_data[:, bedroom\_column] > 4, sale\_price\_column]**

**average\_sale\_price = np.mean(houses\_more\_than\_four\_bedrooms)**

**print("Average sale price of houses with more than four bedrooms:", average\_sale\_price)**

**OUTPUT :**

****

4. **Scenario:** You are working on a project that involves analyzing the sales performance of a company over the past four quarters. The quarterly sales data is stored in a NumPy array named sales\_data, where each element represents the sales amount for a specific quarter. Your task is to calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter.

**Question:** Using NumPy arrays and arithmetic operations calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter?

**PROGRAM :**

**import numpy as np**

**data=np.array([10000,20000,25000,30000])**

**total=np.sum(data)**

**print(total)**

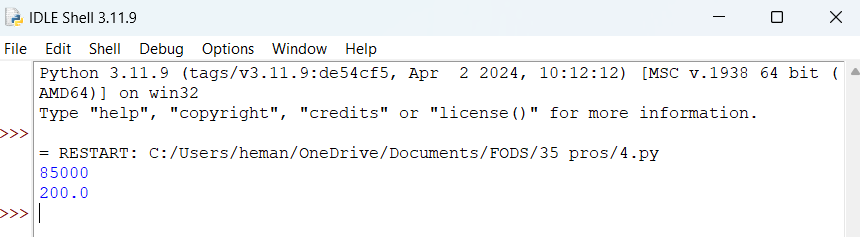
**q1=data[0]**

**q4=data[-1]**

**per=((q4-q1)/q1)\*100**

**print(per)**

**OUTPUT :**

****

5. **Scenario:** You are a data analyst working for a car manufacturing company. As part of your analysis, you have a dataset containing information about the fuel efficiency of different car models. The dataset is stored in a NumPy array named fuel\_efficiency, where each element represents the fuel efficiency (in miles per gallon) of a specific car model. Your task is to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models.

**Question:** How would you use NumPy arrays and arithmetic operations to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models?

**PROGRAM :**

**import numpy as np**

**fuel\_efficiency=np.array([20,30,40,23])**

**avg=np.mean(fuel\_efficiency)**

**print(avg)**

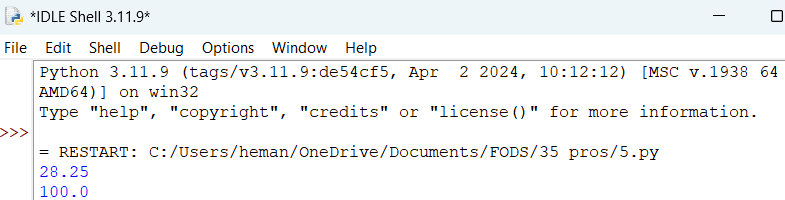
**car1=fuel\_efficiency[0]**

**car2=fuel\_efficiency[2]**

**per=((car2-car1)/car1)\*100**

**print(per)**

**OUTPUT :**

****

6. **Scenario:** You are a cashier at a grocery store and need to calculate the total cost of a customer's purchase, including applicable discounts and taxes. You have the item prices and quantities in separate lists, and the discount and tax rates are given as percentages. Your task is to calculate the total cost for the customer.

**Question:** Use arithmetic operations to calculate the total cost of a customer's purchase, including discounts and taxes, given the item prices, quantities, discount rate, and tax rate?

**PROGRAM**

**items\_price=[6,5,10,2]**

**quantites=[3,2,4,1]**

**discount=10**

**tax=8**

**total=sum(price\*quantity for price,quantity in zip(items\_price,quantites))**

**discount\_amt=(discount/100)\*total**

**total\_after\_dis=total-discount\_amt**

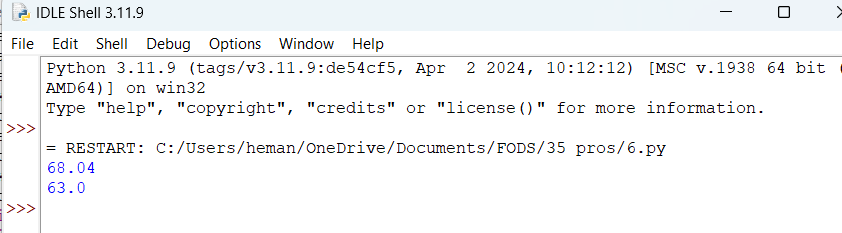
**tax\_amt=(tax/100)\*total\_after\_dis**

**total\_after\_tax=total\_after\_dis+tax\_amt**

**print(total\_after\_tax)**

**print(total\_after\_dis)**

**OUTPUT**

****

7. **Scenario:** You are working as a data analyst for an e-commerce company. You have been given a dataset containing information about customer orders, stored in a Pandas DataFrame named order\_data. The DataFrame has columns for customer ID, order date, product name, and order quantity. Your task is to analyze the data and answer specific questions about the orders.

**Question:** Using Pandas DataFrame operations, how would you find the following information from the order\_data DataFrame:

1. The total number of orders made by each customer.
2. The average order quantity for each product.
3. The earliest and latest order dates in the dataset.

**PROGRAM :**

**import pandas as pd**

**order\_data = pd.DataFrame({**

**'CustomerID': [1, 2, 1, 3, 2],**

**'OrderDate': ['2022-01-01', '2022-01-02', '2022-01-01', '2022-01-03', '2022-01-02'],**

**'ProductName': ['ProductA', 'ProductB', 'ProductA', 'ProductC', 'ProductB'],**

**'OrderQuantity': [3, 5, 2, 1, 4]**

**})**

**total=order\_data.groupby('CustomerID')['OrderDate'].count()**

**avg=order\_data.groupby('ProductName')['OrderQuantity'].mean()**

**earliest=order\_data['OrderDate'].min()**

**latest=order\_data['OrderDate'].max()**

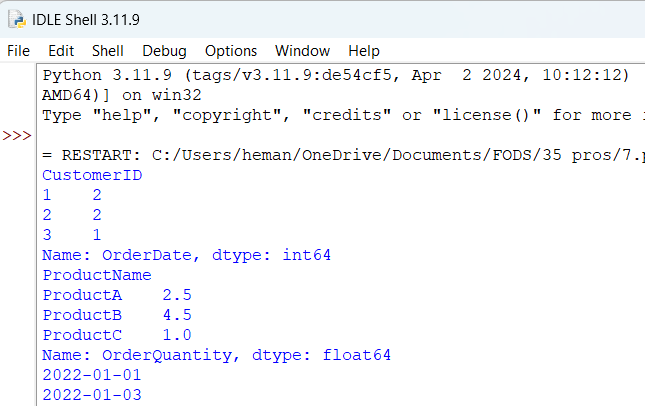
**print(total)**

**print(avg)**

**print(earliest)**

**print(latest)**

**OUTPUT :**

****

8. **Scenario:** You are a data scientist working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a Pandas data frame.

**Question:** How would you find the top 5 products that have been sold the most in the past month?

**PROGRAM :**

**import pandas as pd**

**sales\_data = pd.DataFrame({**

**'ProductID': [1, 2, 1, 3, 2, 3, 4, 5, 4, 5],**

**'ProductName': ['ProductA', 'ProductB', 'ProductA', 'ProductC', 'ProductB', 'ProductC', 'ProductD', 'ProductE', 'ProductD', 'ProductE'],**

**'QuantitySold': [10, 15, 8, 20, 12, 18, 5, 25, 6, 22]**

**})**

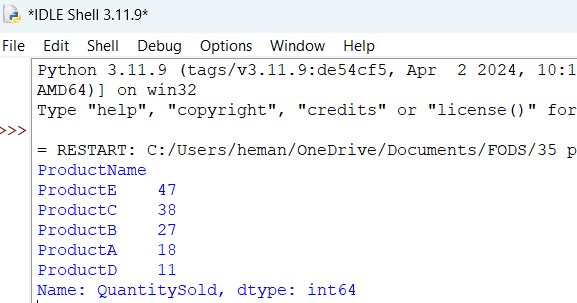
**product\_Sales=sales\_data.groupby('ProductName')['QuantitySold'].sum()**

**sorted\_sales=product\_Sales.sort\_values(ascending=False)**

**five=sorted\_sales.head(5)**

**print(five)**

**OUTPUT :**

****

9. **Scenario:** You work for a real estate agency and have been given a dataset containing information about properties for sale. The dataset is stored in a Pandas DataFrame named property\_data. The DataFrame has columns for property ID, location, number of bedrooms, area in square feet, and listing price. Your task is to analyze the data and answer specific questions about the properties.

**Question:** Using Pandas DataFrame operations, how would you find the following information from the property\_data DataFrame:

1. The average listing price of properties in each location.
2. The number of properties with more than four bedrooms.
3. The property with the largest area.

**PROGRAM :**

**import pandas as pd**

**property\_data = pd.DataFrame({**

**'property\_ID': [1, 2, 3, 4],**

**'location': ['A', 'B', 'A', 'B'],**

**'noofbedrooms': [5, 4, 6, 4],**

**'areasqft': [1200, 1000, 1300, 1400],**

**'listingprice': [50000, 35000, 55000, 40000]**

**})**

**avg\_listing\_price\_per\_location = property\_data.groupby('location')['listingprice'].mean()**

**print("Average listing price of properties in each location:")**

**print(avg\_listing\_price\_per\_location)**

**num\_properties\_more\_than\_four\_bedrooms = property\_data[property\_data['noofbedrooms'] > 4].shape[0]**

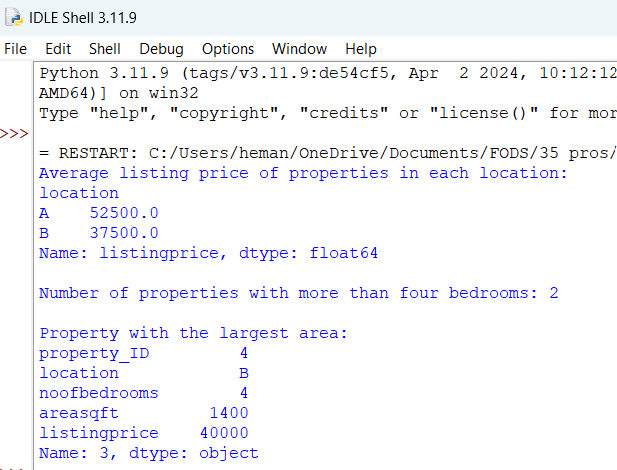
**print("\nNumber of properties with more than four bedrooms:", num\_properties\_more\_than\_four\_bedrooms)**

**property\_with\_largest\_area = property\_data.loc[property\_data['areasqft'].idxmax()]**

**print("\nProperty with the largest area:")**

**print(property\_with\_largest\_area)**

**OUTPUT :**

****

10. **Scenario:** You are working on a data visualization project and need to create basic plots using Matplotlib. You have a dataset containing the monthly sales data for a company, including the month and corresponding sales values. Your task is to develop a Python program that generates line plots and bar plots to visualize the sales data.

**Question:**

1. How would you develop a Python program to create a line plot of the monthly sales data?

2: How would you develop a Python program to create a bar plot of the monthly sales data?

**PROGRAM :**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**sales\_data = pd.DataFrame({**

**'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'],**

**'Sales': [10000, 12000, 8000, 15000, 13000, 16000]**

**})**

**plt.plot(sales\_data['Month'], sales\_data['Sales'], marker='o', color='b', label='Monthly Sales')**

**plt.title('Monthly Sales Data')**

**plt.xlabel('Month')**

**plt.ylabel('Sales')**

**plt.show()**

**plt.bar(sales\_data['Month'], sales\_data['Sales'], color='c', label='Monthly Sales')**

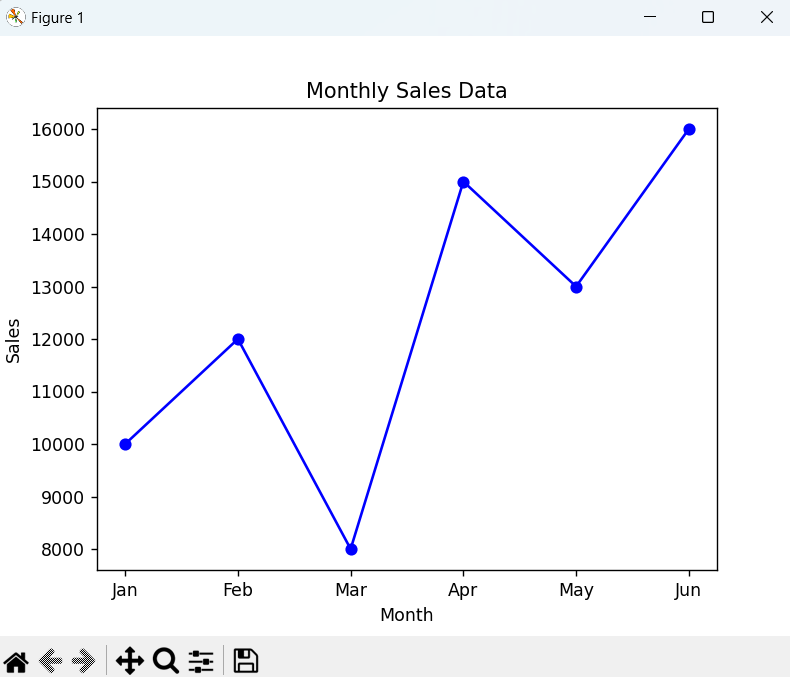
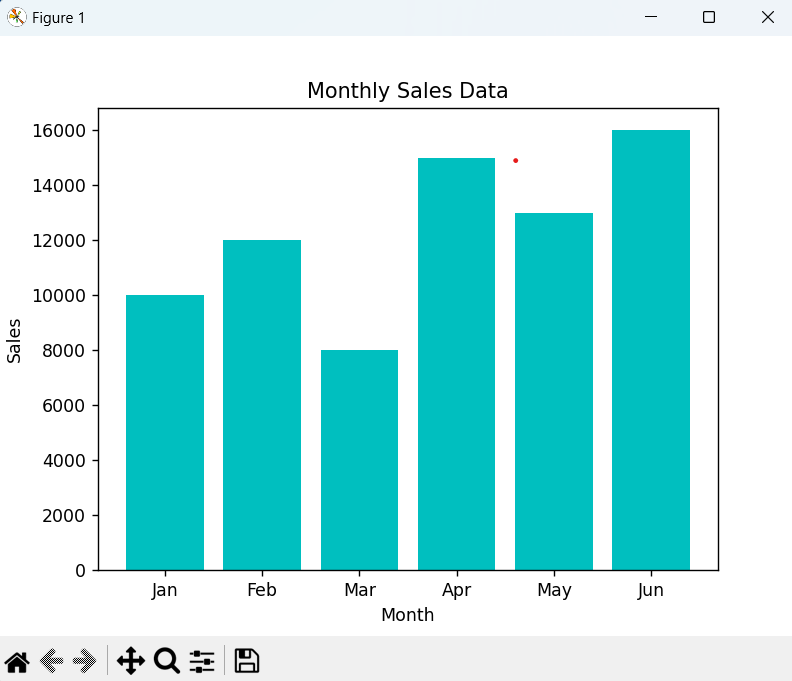
**plt.title('Monthly Sales Data')**

**plt.xlabel('Month')**

**plt.ylabel('Sales')**

**plt.show()**

**OUTPUT :**

 ****

11. **Scenario:** You are a data scientist working for a company that sells products online. You have been tasked with creating a simple plot to show the sales of a product over time.

**Question:**

1. Write code to create a simple line plot in Python using Matplotlib to predict sales happened in a month?

2. Write code to create a scatter plot in Python using Matplotlib to predict sales happened in a month?

3. Develop a Python program to create a bar plot of the monthly sales data.

**PROGRAM:**

import matplotlib.pyplot as plt

import pandas as pd

sales\_data = pd.DataFrame({

'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'],

'Sales': [10000, 12000, 8000, 15000, 13000, 16000]

})

plt.plot(sales\_data['Month'], sales\_data['Sales'], marker='o', color='b', label='Monthly Sales')

plt.title('Monthly Sales Data - Line Plot')

plt.xlabel('Month')

plt.ylabel('Sales')

plt.show()

plt.scatter(sales\_data['Month'], sales\_data['Sales'], color='r', label='Monthly Sales')

plt.title('Monthly Sales Data - Scatter Plot')

plt.xlabel('Month')

plt.ylabel('Sales')

plt.show()

plt.bar(sales\_data['Month'], sales\_data['Sales'], color='g', label='Monthly Sales')

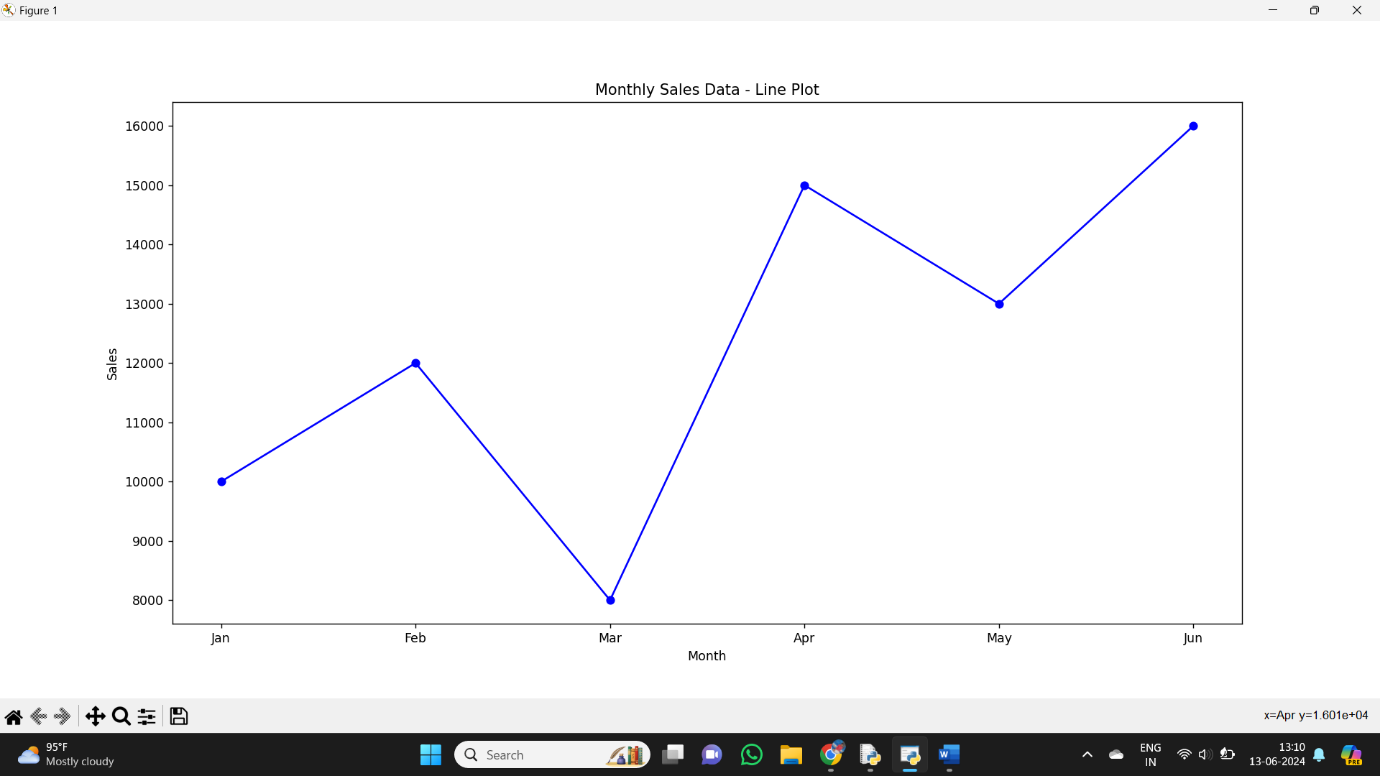
plt.title('Monthly Sales Data - Bar Plot')

plt.xlabel('Month')

plt.ylabel('Sales')

plt.show()

**OUTPUT:**

****

12. **Scenario:** You are working on a data analysis project that involves analyzing the monthly temperature and rainfall data for a city. You have a dataset containing the monthly temperature and rainfall values for each month of a year. Your task is to develop a Python program that generates line plots and scatter plots to visualize the temperature and rainfall data.

**Question:**

1. Develop a Python program to create a line plot of the monthly temperature data.

2: Develop a Python program to create a scatter plot of the monthly rainfall data.

**PROGRAM:**

import matplotlib.pyplot as plt

import pandas as pd

weather\_data = pd.DataFrame({

'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],

'Temperature': [10, 12, 15, 18, 22, 25, 27, 26, 23, 20, 15, 12],

'Rainfall': [50, 40, 30, 20, 10, 5, 8, 12, 15, 25, 35, 45]

})

plt.plot(weather\_data['Month'], weather\_data['Temperature'], marker='o', color='b', label='Temperature')

plt.title('Monthly Temperature Data - Line Plot')

plt.xlabel('Month')

plt.ylabel('Temperature (°C)')

plt.show()

plt.scatter(weather\_data['Month'], weather\_data['Rainfall'], color='r', label='Rainfall')

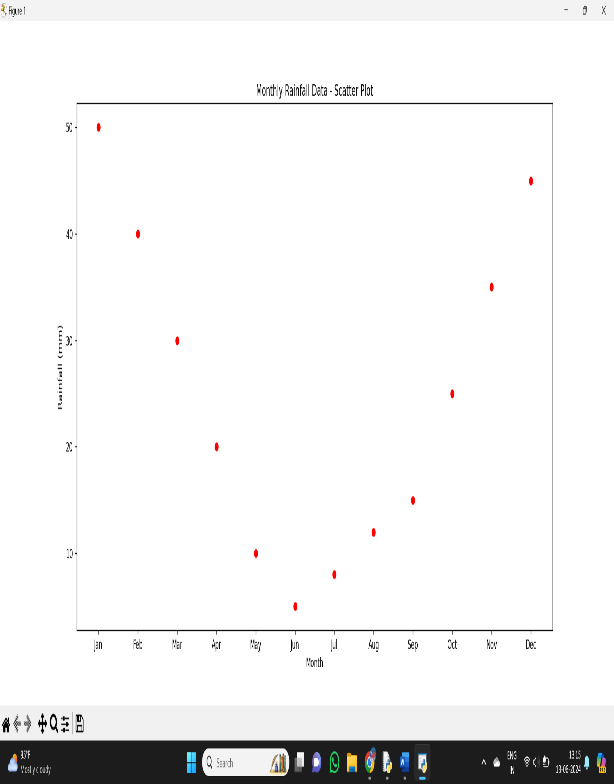
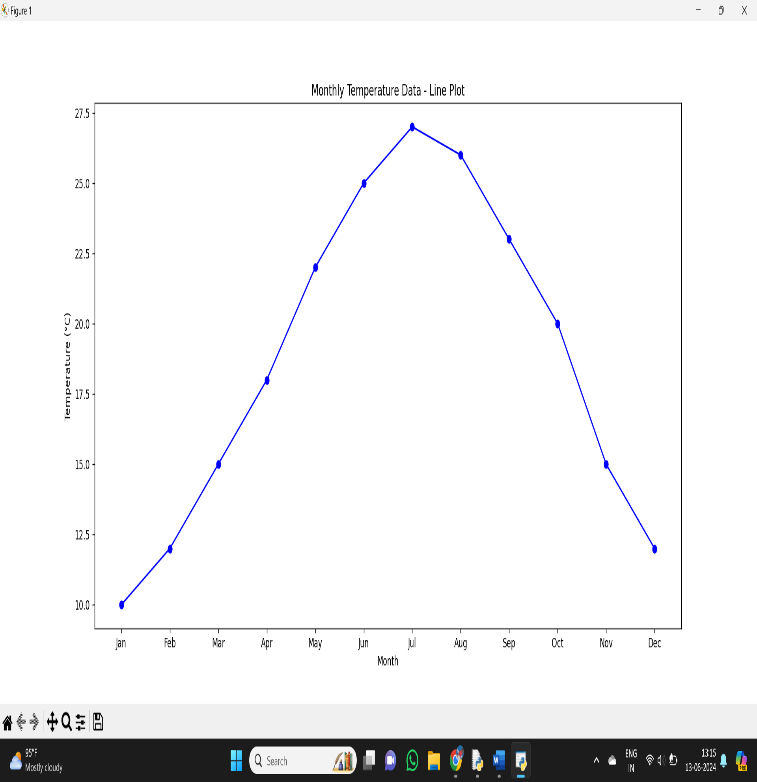
plt.title('Monthly Rainfall Data - Scatter Plot')

plt.xlabel('Month')

plt.ylabel('Rainfall (mm)')

plt.show()

**OUTPUT:**

****

13.**Scenario**: You are a data analyst working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a Pandas data frame.

**Question:** Develop a code in python to find the frequency distribution of the ages of the customers who have made a purchase in the past month.

**PROGRAM:**

import pandas as pd

sales\_data = pd.DataFrame({

    'CustomerID': [1, 2, 3, 4, 5],

    'Name': ['Customer1', 'Customer2', 'Customer3', 'Customer4', 'Customer5'],

    'Age': [25, 30, 28, 35, 40],

    'PurchaseAmount': [100, 150, 120, 200, 180]

})

age\_frequency = sales\_data['Age'].value\_counts().sort\_index()

print("Frequency distribution of customer ages:")

print(age\_frequency)

**OUTPUT:**

****

14. **Scenario:** You are a data analyst working for a social media platform. As part of your analysis, you have a dataset containing user interaction data, including the number of likes received by each post. Your task is to develop a Python program that calculates the frequency distribution of likes among the posts.

**Question:** Develop a Python program to calculate the frequency distribution of likes among the posts?

import pandas as pd

interaction\_data = pd.DataFrame({

    'PostID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

    'Likes': [50, 30, 20, 50, 10, 30, 40, 50, 20, 30]

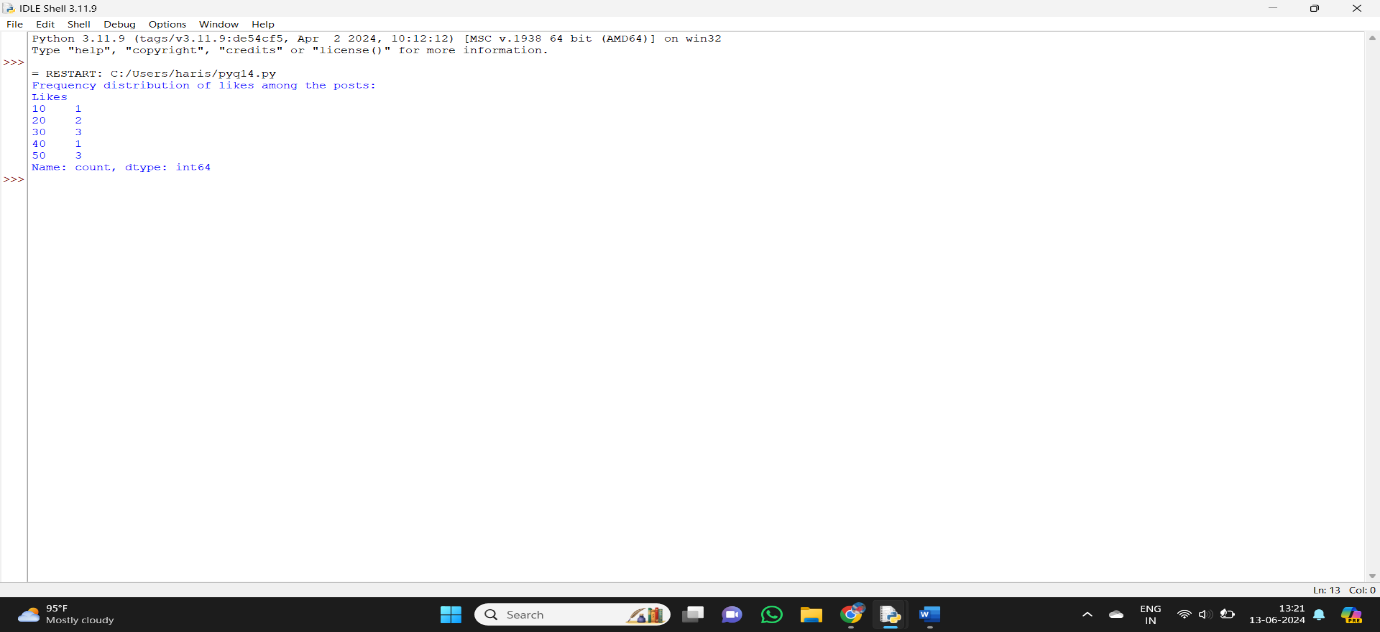
})

likes\_frequency = interaction\_data['Likes'].value\_counts().sort\_index()

print("Frequency distribution of likes among the posts:")

print(likes\_frequency)

**OUTPUT:**

****

15. **Scenario:** You are working on a project that involves analyzing customer reviews for a product. You have a dataset containing customer reviews, and your task is to develop a Python program that calculates the frequency distribution of words in the reviews.

**Question:** Develop a Python program to calculate the frequency distribution of words in the customer reviews dataset?

import pandas as pd

data = {'customerID': [1, 2, 3, 4, 5],

        'reviews': ['good','bad','good','ok','bad']}

df = pd.DataFrame(data)

distribution = df['reviews'].value\_counts().sort\_index()

print("review ")

print(distribution)

**OUTPUT:**

****

**16.** Scenario: You are a data analyst working for a marketing research company. Your team

has collected a large dataset containing customer feedback from various social media

platforms. The dataset consists of thousands of text entries, and your task is to develop a

Python program to analyze the frequency distribution of words in this dataset. Your program

should be able to perform the following tasks:

Load the dataset from a CSV file (data.csv) containing a single column named

&quot;feedback&quot; with each row representing a customer comment.

Preprocess the text data by removing punctuation, converting all text to lowercase,

and eliminating any stop words (common words like &quot;the,&quot; &quot;and,&quot; &quot;is,&quot; etc. that don&#39;t

carry significant meaning).

Calculate the frequency distribution of words in the preprocessed dataset.

Display the top N most frequent words and their corresponding frequencies, where N

is provided as user input.

Plot a bar graph to visualize the top N most frequent words and their frequencies.

Question: Create a Python program that fulfills these requirements and helps your team gain

insights from the customer feedback data.

import pandas as pd

import string

import nltk

from nltk.corpus import stopwords

from collections import Counter

import matplotlib.pyplot as plt

nltk.download('stopwords')

data = {

'feedback': [

'This product is great! I love it.',

'Terrible customer service, will not buy again.',

'Amazing quality, very satisfied with the purchase.',

'The product was okay, but the delivery was slow.',

'I will definitely recommend this to my friends.',

'Not worth the money. Very disappointed.'

]

}

df = pd.DataFrame(data)

df = pd.read\_csv('data.csv')

def preprocess\_text(text):

text = text.translate(str.maketrans('', '', string.punctuation))

text = text.lower()

stop\_words = set(stopwords.words('english'))

text = ' '.join(word for word in text.split() if word not in stop\_words)

return text

df['cleaned\_feedback'] = df['feedback'].apply(preprocess\_text)

all\_words = ' '.join(df['cleaned\_feedback']).split()

word\_freq = Counter(all\_words)

def display\_top\_n\_words(word\_freq, N):

most\_common\_words = word\_freq.most\_common(N)

for word, freq in most\_common\_words:

print(f'{word}: {freq}')

def plot\_top\_n\_words(word\_freq, N):

most\_common\_words = word\_freq.most\_common(N)

words, freqs = zip(\*most\_common\_words)

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

plt.bar(words, freqs)

plt.xlabel('Words')

plt.ylabel('Frequencies')

plt.title(f'Top {N} Most Frequent Words')

plt.xticks(rotation=45)

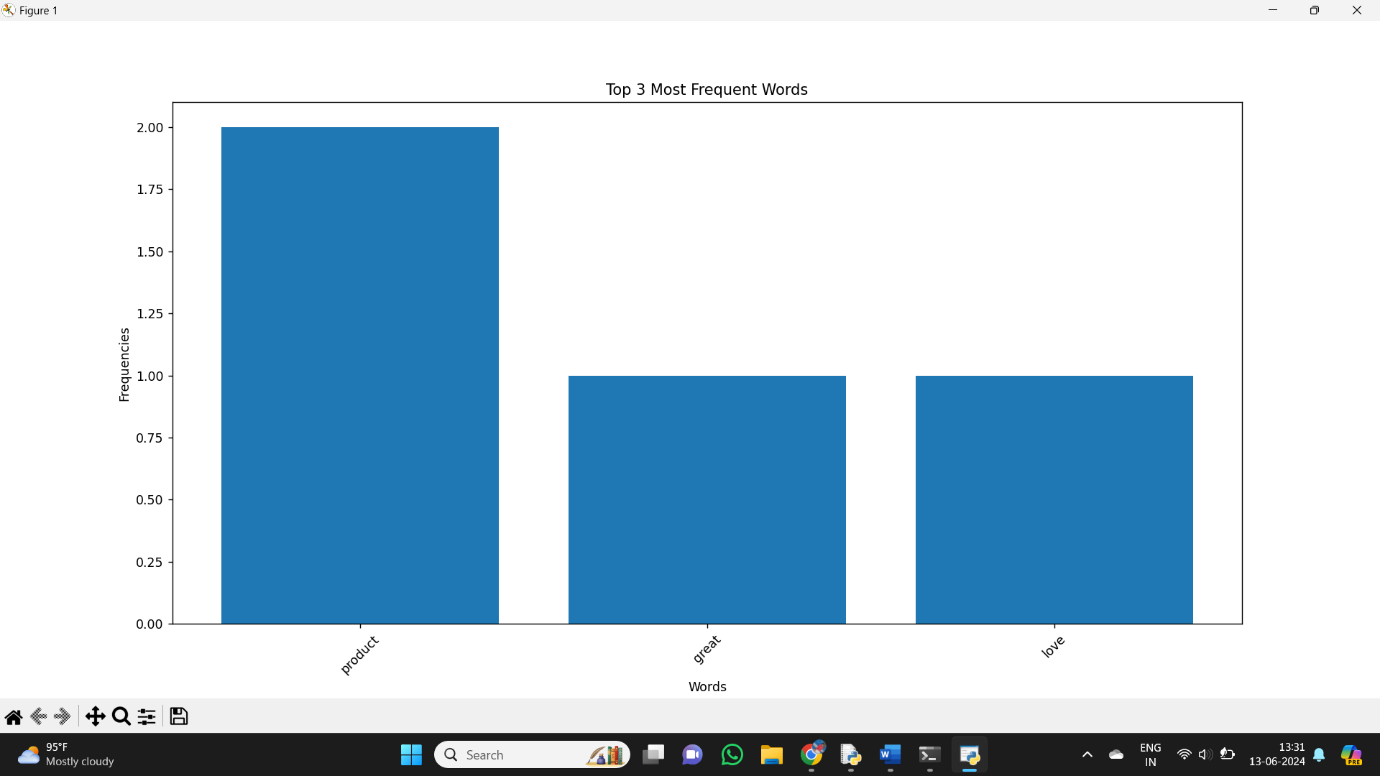
plt.show()

N = int(input("Enter the number of top frequent words to display: "))

display\_top\_n\_words(word\_freq, N)

plot\_top\_n\_words(word\_freq, N)

**OUTPUT:**



**17.** Suppose a hospital tested the age and body fat data for 18 randomly selected adults with

the following result.

Question:

Calculate the mean, median and standard deviation of age and %fat using Pandas.

Draw the boxplots for age and %fat.

Draw a scatter plot and a q-q plot based on these two variables.

**PROGRAM:**

import pandas as pd

import matplotlib.pyplot as plt

import scipy.stats as stats

age = [23, 23, 27, 27, 39, 41, 47, 49, 50, 52, 54, 54, 56, 57, 58, 58, 60, 61]

percent\_fat = [9.5, 26.5, 7.8, 17.8, 31.4, 25.9, 27.4, 27.2, 31.2, 34.6, 42.5, 28.8, 33.4, 30.2, 34.1, 32.9, 41.2, 35.7]

data = pd.DataFrame({'Age': age, '%Fat': percent\_fat})

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

median\_age = data['Age'].median()

std\_age = data['Age'].std()

mean\_fat = data['%Fat'].mean()

median\_fat = data['%Fat'].median()

std\_fat = data['%Fat'].std()

print("Age: Mean =", mean\_age, "Median =", median\_age, "Standard Deviation =", std\_age)

print("%Fat: Mean =", mean\_fat, "Median =", median\_fat, "Standard Deviation =", std\_fat)

data.boxplot(column=['Age', '%Fat'])

plt.title("Boxplots for Age and %Fat")

plt.show()

plt.scatter(data['Age'], data['%Fat'])

plt.xlabel('Age')

plt.ylabel('%Fat')

plt.title('Scatter Plot')

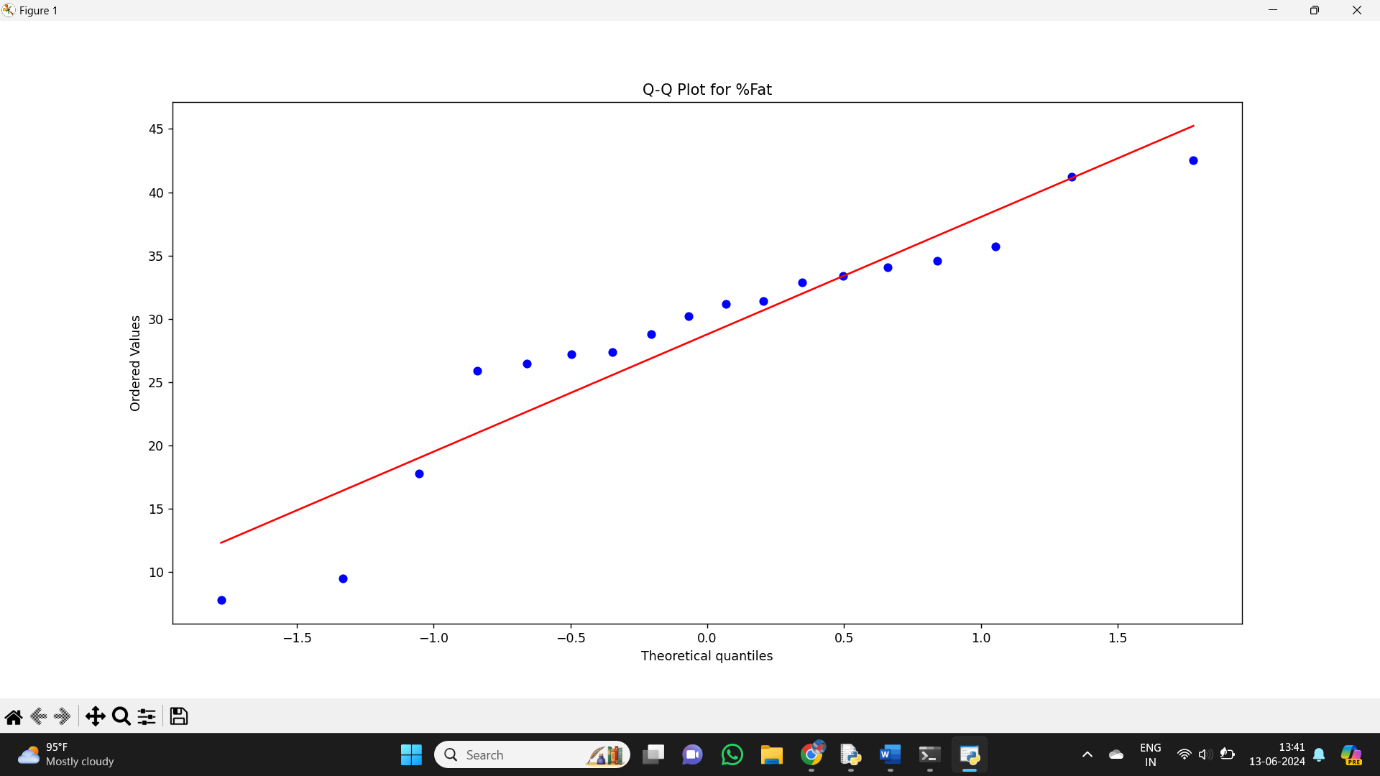
plt.show()

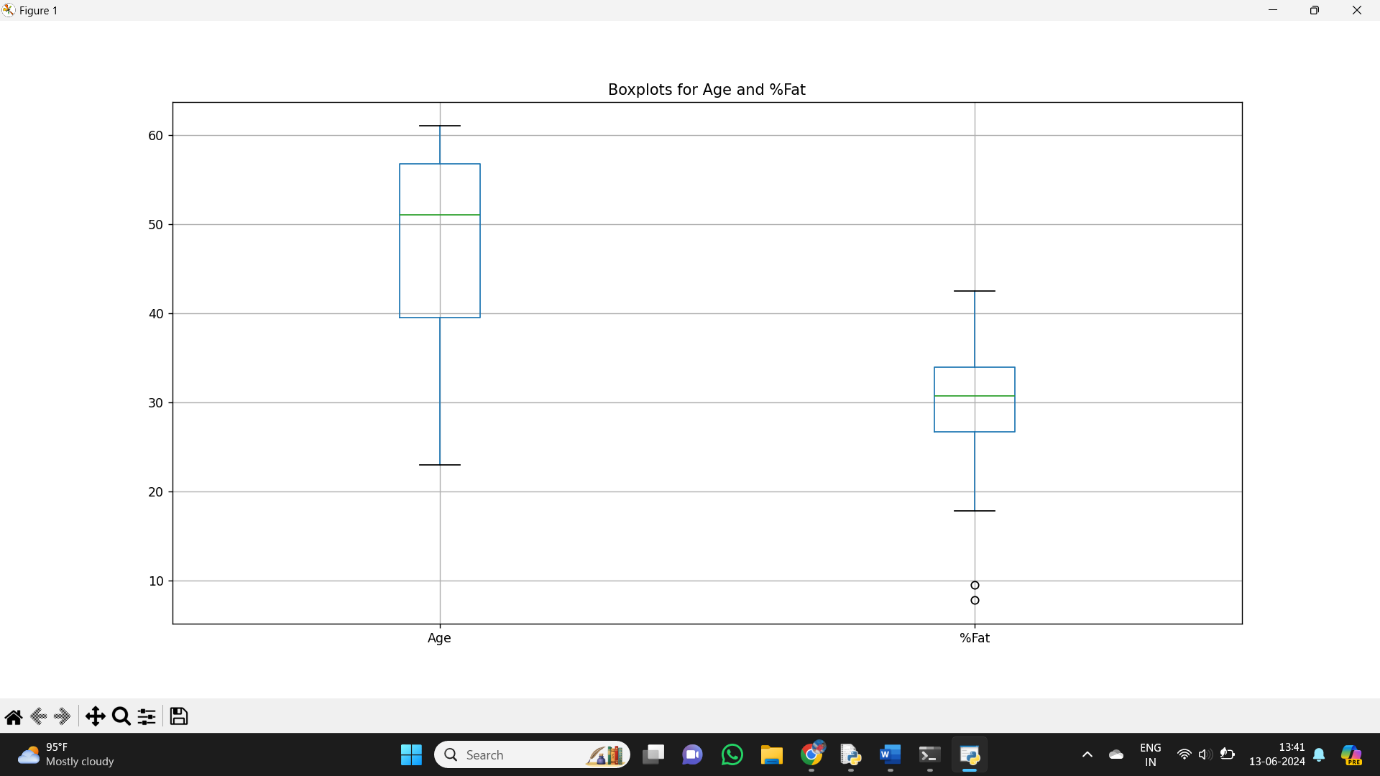
plt.figure()

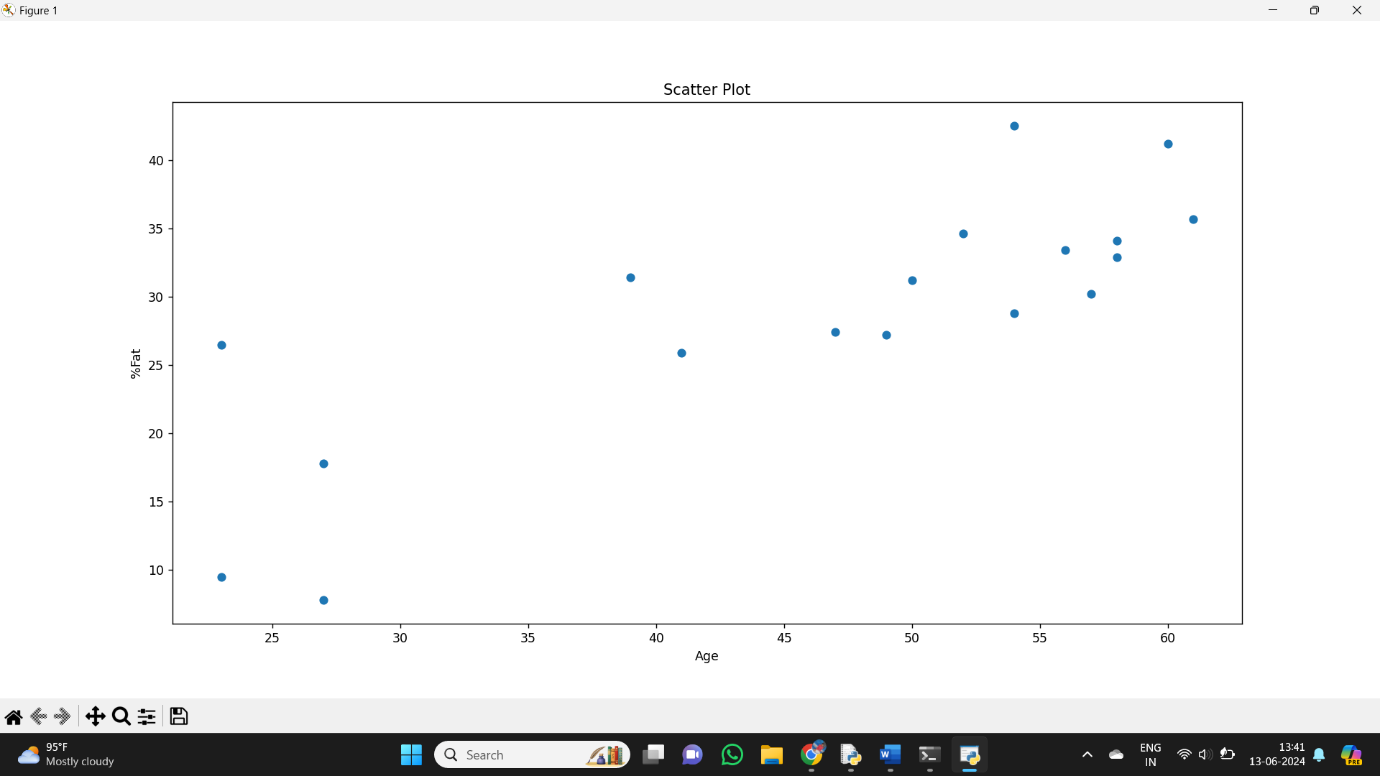
stats.probplot(data['%Fat'], dist="norm", plot=plt)

plt.title('Q-Q Plot for %Fat')

plt.show()

**OUTPUT:**

****

****

18. a. **Scenario:** You are working with a dataset representing the daily sales of a product over the past month. Calculate the variance of the daily sales to understand how much the sales figures deviate from the mean.

**PROGRAM:**

import numpy as np

daily\_sales = np.array([100, 120, 90, 110, 130, 95, 105, 115, 125, 100, 110, 120, 130, 95, 105])

mean\_sales = np.mean(daily\_sales)

variance\_sales = np.var(daily\_sales)

print("Variance of Daily Sales:", variance\_sales)

**OUTPUT:**

 b. **Scenario:** In a study, you have collected data on the hours spent studying (variable x) and the corresponding exam scores (variable y) for a group of students. Calculate the covariance between study hours and exam scores to explore if there is a relationship.

**PROGRAM:**

import numpy as np

study\_hours = np.array([2, 3, 1, 4, 5, 2, 3, 4, 5, 1])

exam\_scores = np.array([65, 70, 60, 75, 80, 65, 70, 75, 80, 60])

covariance\_study\_exam = np.cov(study\_hours, exam\_scores)[0, 1]

print("Covariance between Study Hours and Exam Scores:", covariance\_study\_exam)

**OUTPUT:**

****

19. **Scenario:** You are dealing with a dataset containing the monthly expenses of different departments in a company. Use NumPy functions to efficiently calculate both the variance and covariance matrix of these expenses.

**PROGRAM:**

import numpy as np

monthly\_expenses = np.array([

    [50000, 55000, 60000, 52000, 58000],

    [40000, 42000, 38000, 41000, 45000],

    [30000, 32000, 31000, 30000, 33000]

])

variance\_expenses = np.var(monthly\_expenses, axis=1)

print("Variance of Monthly Expenses for Each Department:")

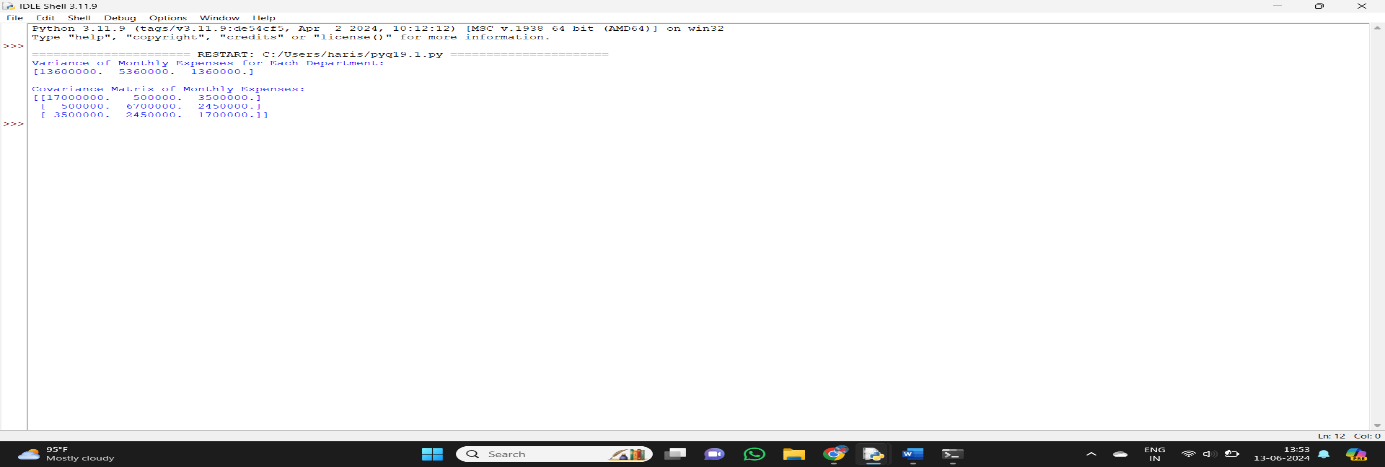
print(variance\_expenses)

covariance\_matrix\_expenses = np.cov(monthly\_expenses)

print("\nCovariance Matrix of Monthly Expenses:")

print(covariance\_matrix\_expenses)

**OUTPUT:**

****

20. a. **Scenario:** Analyzing the performance of servers, you want to determine the 25th, 50th, and 75th percentiles of response times to identify potential bottlenecks.

import numpy as np

response\_times = np.array([20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70])

percentiles\_25th = np.percentile(response\_times, 25)

percentiles\_50th = np.percentile(response\_times, 50)

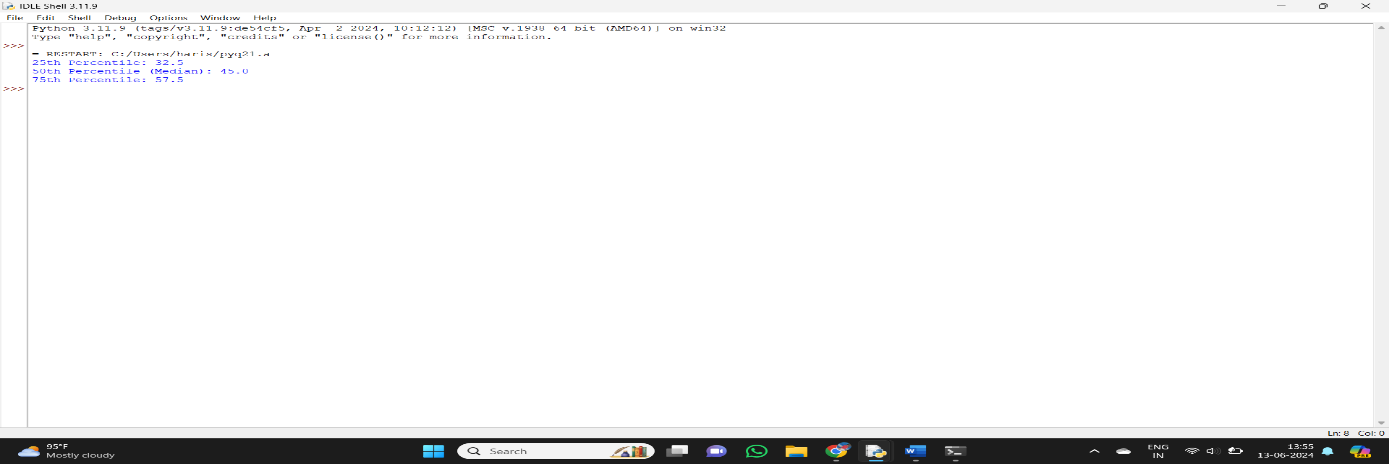
percentiles\_75th = np.percentile(response\_times, 75)

print("25th Percentile:", percentiles\_25th)

print("50th Percentile (Median):", percentiles\_50th)

print("75th Percentile:", percentiles\_75th)

**OUTPUT:**

****

b. **Scenario:** In a medical study, you have collected data on patients' recovery times after a procedure. Calculate the 10th, 50th, and 90th percentiles to understand the distribution of recovery times.

import numpy as np

recovery\_times = np.array([10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60])

percentiles\_10th = np.percentile(recovery\_times, 10)

percentiles\_50th = np.percentile(recovery\_times, 50)

percentiles\_90th = np.percentile(recovery\_times, 90)

print("10th Percentile:", percentiles\_10th)

print("50th Percentile (Median):", percentiles\_50th)

print("90th Percentile:", percentiles\_90th)

**OUTPUT:**

****

23.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.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

np.random.seed(42)

num\_customers = 100

customer\_ids = range(1, num\_customers + 1)

total\_amount\_spent = np.random.uniform(10, 1000, num\_customers)

num\_items\_purchased = np.random.randint(1, 50, num\_customers)

data = pd.DataFrame({

'customer\_id': customer\_ids,

'total\_amount\_spent': total\_amount\_spent,

'num\_items\_purchased': num\_items\_purchased

})

features = data[['total\_amount\_spent', 'num\_items\_purchased']]

scaler = StandardScaler()

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

sse = []

k\_range = range(1, 11)

for k in k\_range:

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

kmeans.fit(scaled\_features)

sse.append(kmeans.inertia\_)

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

plt.plot(k\_range, sse, marker='o')

plt.xlabel('Number of Clusters')

plt.ylabel('SSE')

plt.title('Elbow Method for Optimal k')

plt.show()

optimal\_k = 3

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

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

data['cluster'] = clusters

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

for cluster in range(optimal\_k):

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

plt.scatter(clustered\_data['total\_amount\_spent'],

clustered\_data['num\_items\_purchased'],

label=f'Cluster {cluster}')

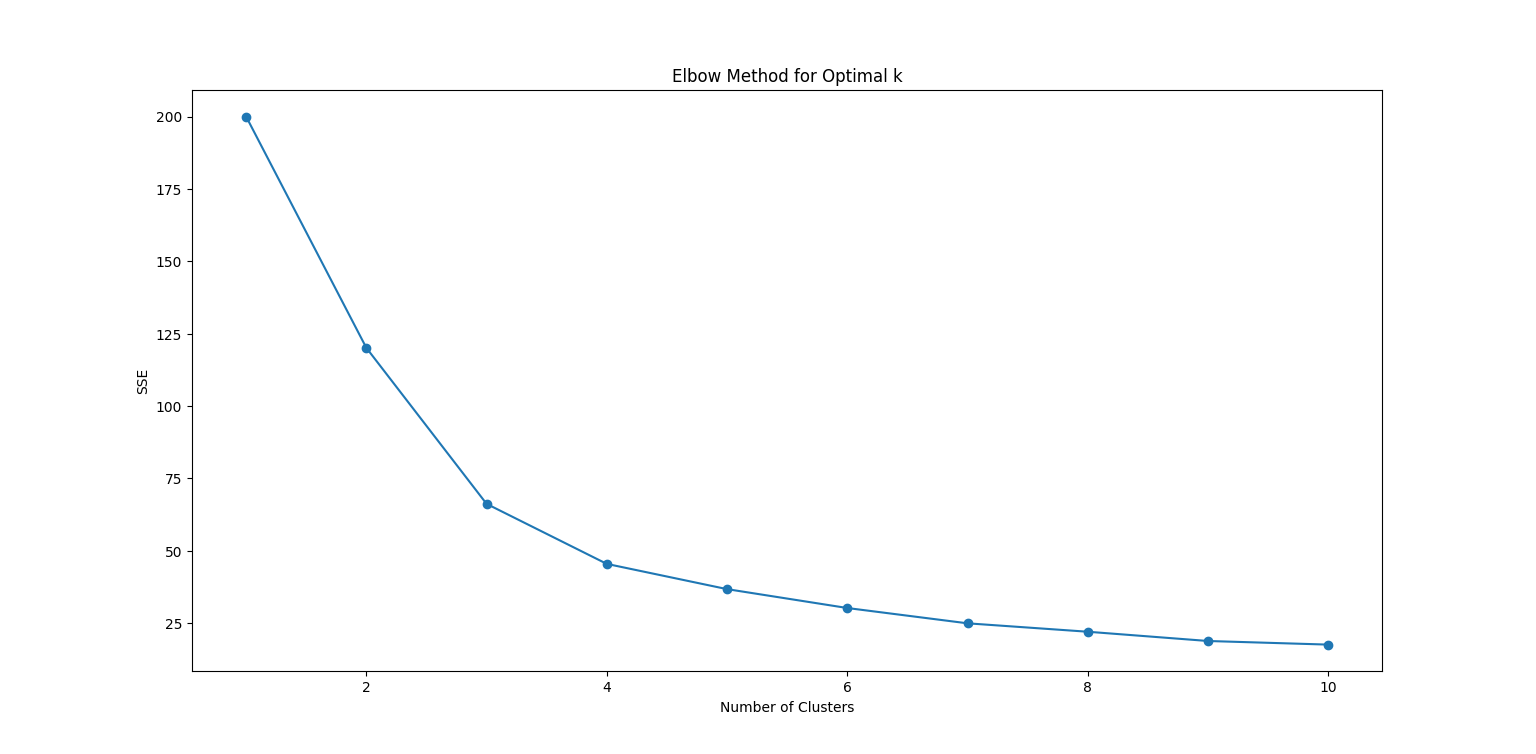
plt.xlabel('Total Amount Spent')

plt.ylabel('Number of Items Purchased')

plt.title('Customer Segments')

plt.legend()

plt.show()



24. 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.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

np.random.seed(42)

num\_customers = 200

customer\_ids = range(1, num\_customers + 1)

total\_amount\_spent = np.random.uniform(10, 2000, num\_customers)

frequency\_of\_visits = np.random.randint(1, 100, num\_customers)

data = pd.DataFrame({

'customer\_id': customer\_ids,

'total\_amount\_spent': total\_amount\_spent,

'frequency\_of\_visits': frequency\_of\_visits

})

features = data[['total\_amount\_spent', 'frequency\_of\_visits']]

scaler = StandardScaler()

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

sse = []

k\_range = range(1, 11)

for k in k\_range:

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

kmeans.fit(scaled\_features)

sse.append(kmeans.inertia\_)

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

plt.plot(k\_range, sse, marker='o')

plt.xlabel('Number of Clusters')

plt.ylabel('SSE')

plt.title('Elbow Method for Optimal k')

plt.show()

optimal\_k = 4

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

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

data['cluster'] = clusters

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

for cluster in range(optimal\_k):

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

plt.scatter(clustered\_data['total\_amount\_spent'],

clustered\_data['frequency\_of\_visits'],

label=f'Cluster {cluster}')

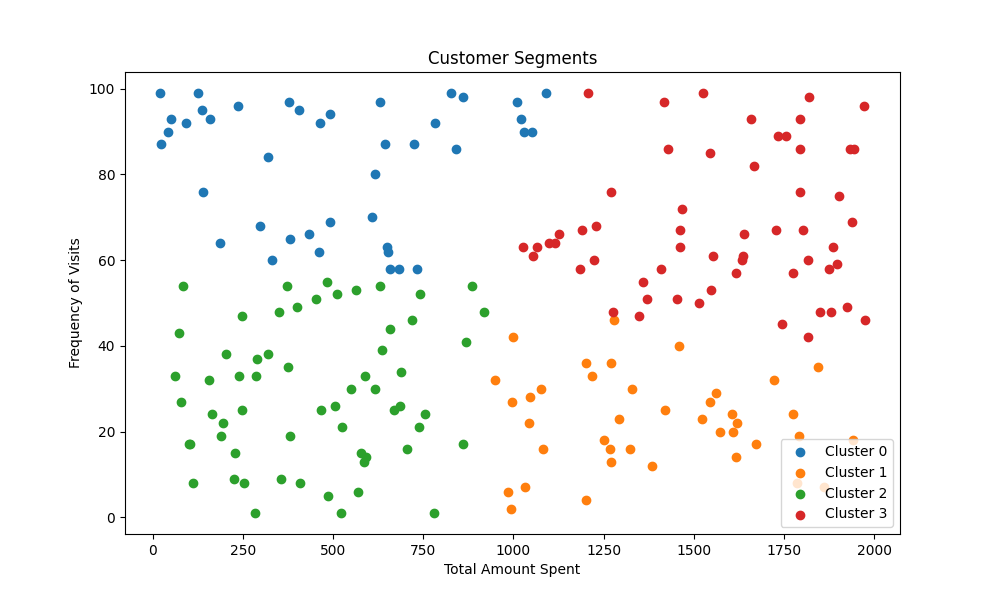
plt.xlabel('Total Amount Spent')

plt.ylabel('Frequency of Visits')

plt.title('Customer Segments')

plt.legend()

plt.show()



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 (&quot;Good&quot;) or

negatively (&quot;Bad&quot;) 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 (&quot;Good&quot; or &quot;Bad&quot;) for new patients based on their features. Evaluate the

model&#39;s performance using accuracy, precision, recall, and F1-score.Make predictions on the

test set and display the results.

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

np.random.seed(42)

num\_patients = 500

age = np.random.randint(18, 90, num\_patients)

gender = np.random.choice(['Male', 'Female'], num\_patients)

blood\_pressure = np.random.randint(80, 180, num\_patients)

cholesterol = np.random.randint(100, 300, num\_patients)

treatment\_outcome = np.random.choice(['Good', 'Bad'], num\_patients)

data = pd.DataFrame({

'age': age,

'gender': gender,

'blood\_pressure': blood\_pressure,

'cholesterol': cholesterol,

'treatment\_outcome': treatment\_outcome

})

data['gender'] = data['gender'].map({'Male': 0, 'Female': 1})

data['treatment\_outcome'] = data['treatment\_outcome'].map({'Good': 1, 'Bad': 0})

X = data[['age', 'gender', 'blood\_pressure', 'cholesterol']]

y = data['treatment\_outcome']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_scaled, y\_train)

y\_pred = knn.predict(X\_test\_scaled)

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)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

print('Confusion Matrix:')

print(conf\_matrix)

Out put

Accuracy: 0.37

Precision: 0.35

Recall: 0.38

F1 Score: 0.36

Confusion Matrix:

[[19 34]

[29 18]]

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&#39;s performance to ensure its accuracy

and reliability.

import pandas as pd

import numpy as np

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

# Generate sample real estate data

np.random.seed(42)

num\_houses = 200

house\_size = np.random.uniform(1000, 4000, num\_houses) # in square feet

num\_bedrooms = np.random.randint(2, 6, num\_houses)

location = np.random.choice(['Location1', 'Location2', 'Location3'], num\_houses)

house\_price = house\_size \* 150 + np.random.normal(0, 25000, num\_houses) # in dollars

data = pd.DataFrame({

'house\_size': house\_size,

'num\_bedrooms': num\_bedrooms,

'location': location,

'house\_price': house\_price

})

# Perform bivariate analysis

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

plt.scatter(data['house\_size'], data['house\_price'], alpha=0.5)

plt.xlabel('House Size (square feet)')

plt.ylabel('House Price (dollars)')

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

plt.show()

# Select feature and target variable

X = data[['house\_size']]

y = data['house\_price']

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Build and train the linear regression model

model = LinearRegression()

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

# Make predictions on the test set

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

# Evaluate the model's performance

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(f'Mean Absolute Error (MAE): {mae:.2f}')

print(f'Mean Squared Error (MSE): {mse:.2f}')

print(f'Root Mean Squared Error (RMSE): {rmse:.2f}')

print(f'R-squared: {r2:.2f}')

# Visualize the results with a scatter plot and the regression line

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

plt.scatter(X\_test, y\_test, alpha=0.5, label='Actual')

plt.plot(X\_test, y\_pred, color='red', linewidth=2, label='Predicted')

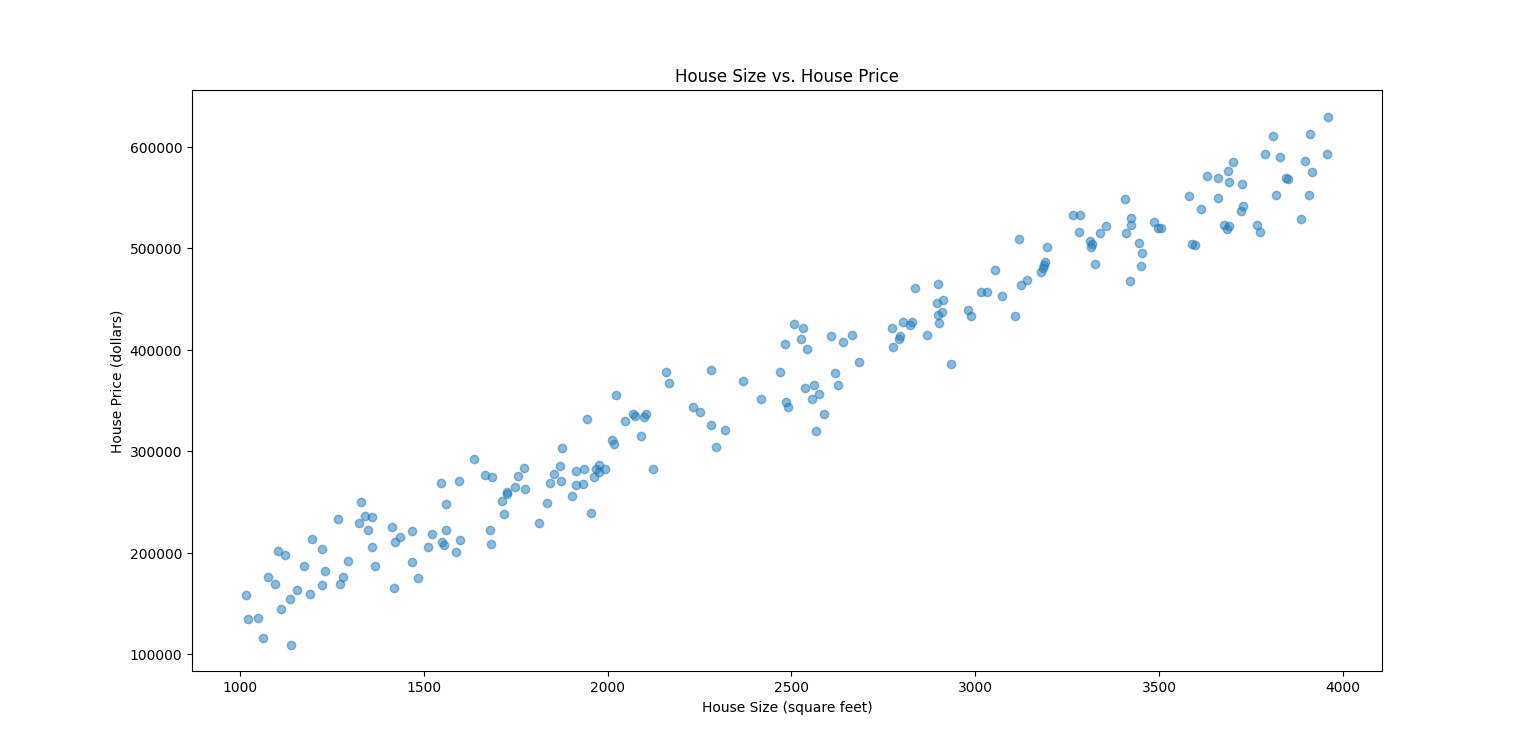
plt.xlabel('House Size (square feet)')

plt.ylabel('House Price (dollars)')

plt.title('House Size vs. House Price (with Linear Regression Line)')

plt.legend()

plt.show()



Question: Classification and Regression Trees (CART) for Car Price Prediction

You are working for a car dealership, and you want to predict the price of used cars based on

various features such as the car&#39;s mileage, age, brand, and engine type. You have collected a

dataset of used cars with their respective prices.

Write a Python program that loads the car dataset and allows the user to input the features of

a new car they want to sell. The program should use the Classification and Regression Trees

(CART) algorithm from scikit-learn to predict the price of the new car based on the input

features.

The CART algorithm will create a tree-based model that will split the data into subsets based

on the chosen features and their values, leading to a decision path that eventually predicts the

price of the car. The program should output the predicted price and display the decision path

(the sequence of conditions leading to the prediction) for the new car.

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor, export\_text

from sklearn.metrics import mean\_absolute\_error

# Sample dataset

data = {

'mileage': [15000, 30000, 45000, 60000, 75000, 90000, 105000, 120000],

'age': [1, 2, 3, 4, 5, 6, 7, 8],

'brand': ['BrandA', 'BrandB', 'BrandA', 'BrandB', 'BrandA', 'BrandB', 'BrandA', 'BrandB'],

'engine\_type': ['Type1', 'Type2', 'Type1', 'Type2', 'Type1', 'Type2', 'Type1', 'Type2'],

'price': [20000, 18000, 16000, 14000, 12000, 10000, 8000, 6000]

}

df = pd.DataFrame(data)

# Encode categorical variables

df['brand\_cat'] = df['brand'].astype('category')

df['engine\_type\_cat'] = df['engine\_type'].astype('category')

df['brand'] = df['brand\_cat'].cat.codes

df['engine\_type'] = df['engine\_type\_cat'].cat.codes

# Features and target variable

X = df[['mileage', 'age', 'brand', 'engine\_type']]

y = df['price']

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the Decision Tree Regressor

model = DecisionTreeRegressor(random\_state=42)

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

# Evaluate the model

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

mae = mean\_absolute\_error(y\_test, y\_pred)

print(f'Mean Absolute Error: {mae:.2f}')

# Function to predict price and display decision path

def predict\_price(mileage, age, brand, engine\_type):

brand\_code = df['brand\_cat'].cat.categories.get\_loc(brand)

engine\_type\_code = df['engine\_type\_cat'].cat.categories.get\_loc(engine\_type)

new\_car = np.array([[mileage, age, brand\_code, engine\_type\_code]])

predicted\_price = model.predict(new\_car)[0]

decision\_path = model.decision\_path(new\_car)

node\_indicator = decision\_path.indices[decision\_path.indptr[0]:decision\_path.indptr[1]]

print(f'Predicted Price: ${predicted\_price:.2f}')

print('Decision Path:')

print(export\_text(model, feature\_names=['mileage', 'age', 'brand', 'engine\_type'], max\_depth=len(node\_indicator)))

return predicted\_price

# Input features for a new car

new\_car\_features = {

'mileage': 50000,

'age': 3,

'brand': 'BrandA',

'engine\_type': 'Type1'

}

# Predict price for the new car

predicted\_price = predict\_price(new\_car\_features['mileage'], new\_car\_features['age'], new\_car\_features['brand'], new\_car\_features['engine\_type'])

Output

Predicted Price: $16000.00

Decision Path:

|--- mileage <= 90000.00

| |--- mileage <= 30000.00

| | |--- value: [20000.00]

| |--- mileage > 30000.00

| | |--- age <= 3.50

| | | |--- value: [16000.00]

| | |--- age > 3.50

| | | |--- mileage <= 67500.00

| | | | |--- value: [14000.00]

| | | |--- mileage > 67500.00

| | | | |--- value: [12000.00]

|--- mileage > 90000.00

| |--- age <= 7.50

| | |--- value: [8000.00]

| |--- age > 7.50

| | |--- value: [6000.00]

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.

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Sample customer data (Replace with actual customer dataset)

data = {

'customer\_id': range(1, 101),

'age': np.random.randint(18, 70, 100),

'annual\_income': np.random.randint(20000, 100000, 100),

'spending\_score': np.random.randint(1, 100, 100)

}

df = pd.DataFrame(data)

# Feature selection

features = df[['age', 'annual\_income', 'spending\_score']]

# Scaling the features

scaler = StandardScaler()

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

# Finding the optimal number of clusters using the Elbow Method

wcss = []

for i in range(1, 11):

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

kmeans.fit(scaled\_features)

wcss.append(kmeans.inertia\_)

# Plotting the Elbow graph

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

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

plt.xlabel('Number of clusters')

plt.ylabel('WCSS (Within-Cluster Sum of Squares)')

plt.title('Elbow Method For Optimal Number of Clusters')

plt.show()

# From the Elbow graph, let's choose k=4 (as an example)

kmeans = KMeans(n\_clusters=4, random\_state=42)

df['cluster'] = kmeans.fit\_predict(scaled\_features)

# Evaluate the clusters using silhouette score

silhouette\_avg = silhouette\_score(scaled\_features, df['cluster'])

print(f'Silhouette Score: {silhouette\_avg:.2f}')

# Visualizing the clusters

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

sns.scatterplot(data=df, x='annual\_income', y='spending\_score', hue='cluster', palette='viridis')

plt.title('Customer Segments')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.legend(title='Cluster')

plt.show()

# Descriptive statistics for each cluster

cluster\_summary = df.groupby('cluster').agg({

'age': ['mean', 'std'],

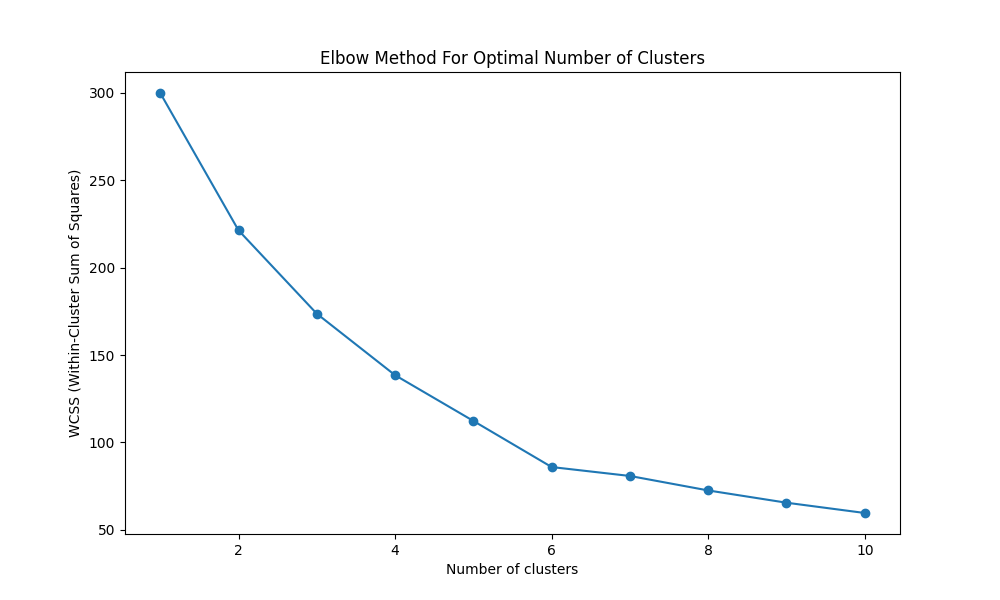
'annual\_income': ['mean', 'std'],

'spending\_score': ['mean', 'std']

}).reset\_index()

print(cluster\_summary)

Output



29. 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

labeled 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.

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Sample dataset (replace with actual dataset)

data = {

'feature1': np.random.rand(100),

'feature2': np.random.rand(100),

'feature3': np.random.rand(100),

'feature4': np.random.rand(100),

'condition': np.random.randint(0, 2, 100)

}

df = pd.DataFrame(data)

# Features and target variable

X = df.drop('condition', axis=1)

y = df['condition']

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scaling the features

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

# Training the KNN model

def train\_knn(k):

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(X\_train\_scaled, y\_train)

return knn

# Predicting for a new patient

def predict\_condition(knn\_model, new\_patient\_features):

new\_patient\_scaled = scaler.transform([new\_patient\_features])

prediction = knn\_model.predict(new\_patient\_scaled)

return prediction[0]

# Evaluating the model

def evaluate\_model(knn\_model):

y\_pred = knn\_model.predict(X\_test\_scaled)

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

report = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

print('Classification Report:')

print(report)

# User input for k and new patient features

k = int(input("Enter the value of k (number of neighbors): "))

new\_patient\_features = list(map(float, input("Enter the features of the new patient separated by space: ").split()))

# Train the model and make predictions

knn\_model = train\_knn(k)

evaluate\_model(knn\_model)

prediction = predict\_condition(knn\_model, new\_patient\_features)

condition = 'has the condition' if prediction == 1 else 'does not have the condition'

print(f'The patient {condition}.')

Output

Enter the value of k (number of neighbors): 6

Enter the features of the new patient separated by space: 88

Accuracy: 0.40

Classification Report:

precision recall f1-score support

0 0.42 0.89 0.57 9

1 0.00 0.00 0.00 11

accuracy 0.40 20

macro avg 0.21 0.44 0.29 20

weighted avg 0.19 0.40 0.26 20

30.Question : Decision Tree for Iris Flower Classification

You are analyzing the famous Iris flower dataset to classify iris flowers into three species

based on their sepal and petal dimensions. You want to use a Decision Tree classifier to

accomplish this task.

Write a Python program that loads the Iris dataset from scikit-learn, and allows the user to

input the sepal length, sepal width, petal length, and petal width of a new flower. The

program should then use the Decision Tree classifier to predict the species of the new flower.

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

from sklearn import tree

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

feature\_names = iris.feature\_names

target\_names = iris.target\_names

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the Decision Tree classifier

clf = DecisionTreeClassifier(random\_state=42)

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

# Evaluate the model

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

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

print(f'Accuracy: {accuracy:.2f}')

print('Classification Report:')

print(classification\_report(y\_test, y\_pred, target\_names=target\_names))

# Function to predict the species of a new flower

def predict\_species(sepal\_length, sepal\_width, petal\_length, petal\_width):

new\_flower = np.array([[sepal\_length, sepal\_width, petal\_length, petal\_width]])

prediction = clf.predict(new\_flower)

species = target\_names[prediction][0]

return species

# User input for the new flower's dimensions

sepal\_length = float(input("Enter sepal length (cm): "))

sepal\_width = float(input("Enter sepal width (cm): "))

petal\_length = float(input("Enter petal length (cm): "))

petal\_width = float(input("Enter petal width (cm): "))

# Predict the species of the new flower

predicted\_species = predict\_species(sepal\_length, sepal\_width, petal\_length, petal\_width)

print(f'The predicted species of the new flower is: {predicted\_species}')

# Plot the Decision Tree

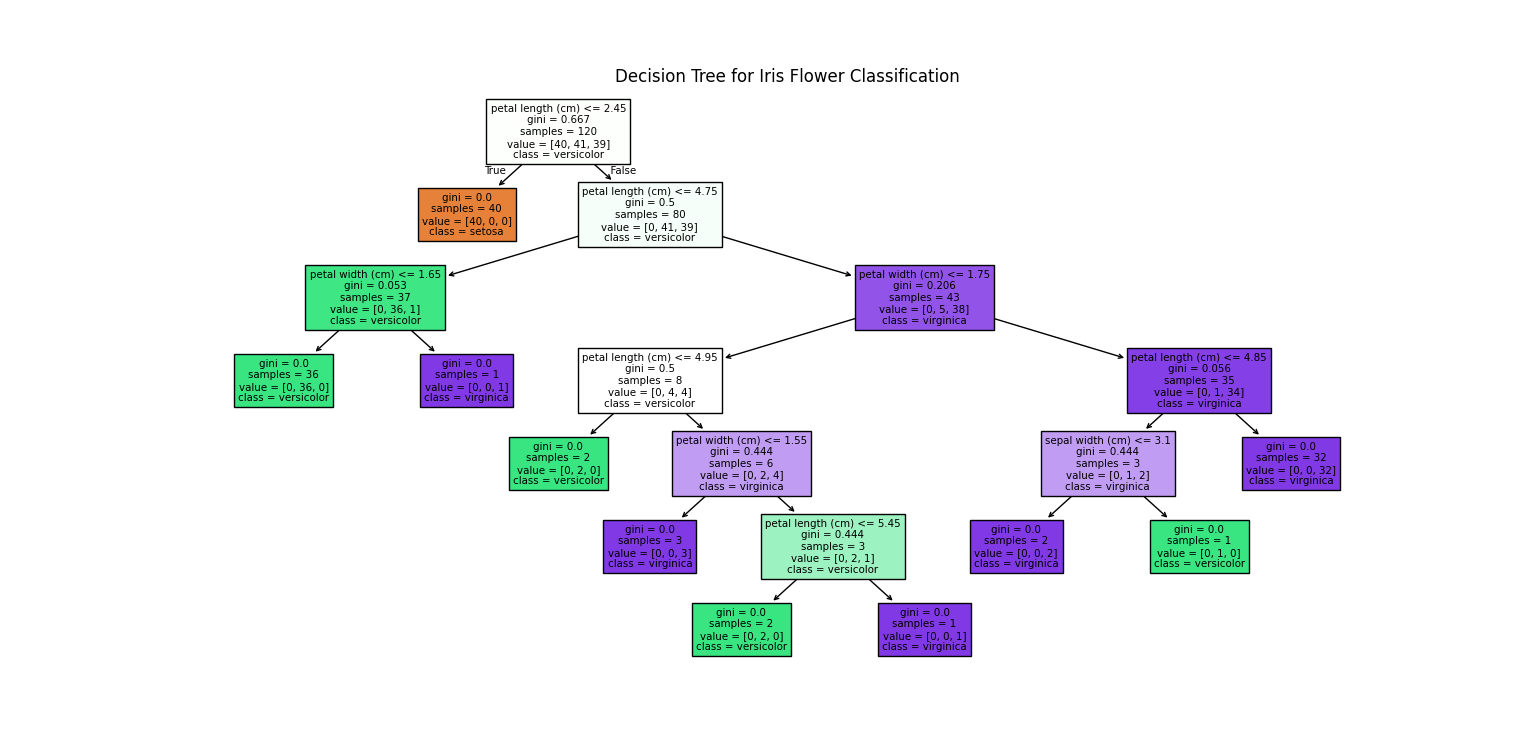
plt.figure(figsize=(12, 8))

tree.plot\_tree(clf, feature\_names=feature\_names, class\_names=target\_names, filled=True)

plt.title('Decision Tree for Iris Flower Classification')

plt.show()

Output



31.Question : Linear Regression for Housing Price Prediction

You are a real estate analyst trying to predict housing prices based on various features of the

houses, such as area, number of bedrooms, and location. You have collected a dataset of

houses with their respective prices.

Write a Python program that allows the user to input the features (area, number of bedrooms,

etc.) of a new house. The program should use linear regression from scikit-learn to predict the

price of the new house based on the input features.

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

# Sample housing dataset (replace with your actual dataset)

data = {

'area': [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],

'bedrooms': [3, 3, 3, 2, 2, 3, 4, 4, 2, 3],

'location': [1, 2, 3, 2, 1, 3, 2, 3, 1, 2],

'price': [245000, 312000, 279000, 308000, 199000, 219000, 405000, 324000, 220000, 300000]

}

df = pd.DataFrame(data)

# Features and target variable

X = df[['area', 'bedrooms', 'location']]

y = df['price']

# Split the data into training and test 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)

# Evaluate the model

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

mae = mean\_absolute\_error(y\_test, y\_pred)

print(f'Mean Absolute Error: {mae:.2f}')

# Function to predict the price of a new house

def predict\_price(area, bedrooms, location):

new\_house = np.array([[area, bedrooms, location]])

predicted\_price = model.predict(new\_house)

return predicted\_price[0]

# User input for the new house's features

area = float(input("Enter the area of the house (in square feet): "))

bedrooms = int(input("Enter the number of bedrooms: "))

location = int(input("Enter the location (e.g., 1 for downtown, 2 for suburbs, 3 for rural): "))

# Predict the price for the new house

predicted\_price = predict\_price(area, bedrooms, location)

print(f'The predicted price of the house is: ${predicted\_price:.2f}')

Output

Mean Absolute Error: 46607.85

Enter the area of the house (in square feet): 66

Enter the number of bedrooms: 7

Enter the location (e.g., 1 for downtown, 2 for suburbs, 3 for rural): 3

The predicted price of the house is: $-14425.73

31.**Question** : Linear Regression for Housing Price Prediction

You are a real estate analyst trying to predict housing prices based on various features of the houses, such as area, number of bedrooms, and location. You have collected a dataset of houses with their respective prices.

Write a Python program that allows the user to input the features (area, number of bedrooms, etc.) of a new house. The program should use linear regression from scikit-learn to predict the price of the new house based on the input features.

**PROGRAM :**

**import numpy as np**

**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**

**import joblib**

**data = {**

**'area': [1500, 2000, 2500, 3000, 3500],**

**'bedrooms': [3, 4, 3, 5, 4],**

**'age': [10, 5, 8, 12, 7],**

**'price': [300000, 400000, 350000, 500000, 450000]**

**}**

**df = pd.DataFrame(data)**

**X = df[['area', 'bedrooms', 'age']]**

**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)**

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

**joblib.dump(model, 'house\_price\_predictor.pkl')**

**def predict\_price(area, bedrooms, age):**

**model = joblib.load('house\_price\_predictor.pkl')**

**features = np.array([[area, bedrooms, age]])**

**predicted\_price = model.predict(features)**

**return predicted\_price[0]**

**area = float(input("Enter the area of the house in square feet: "))**

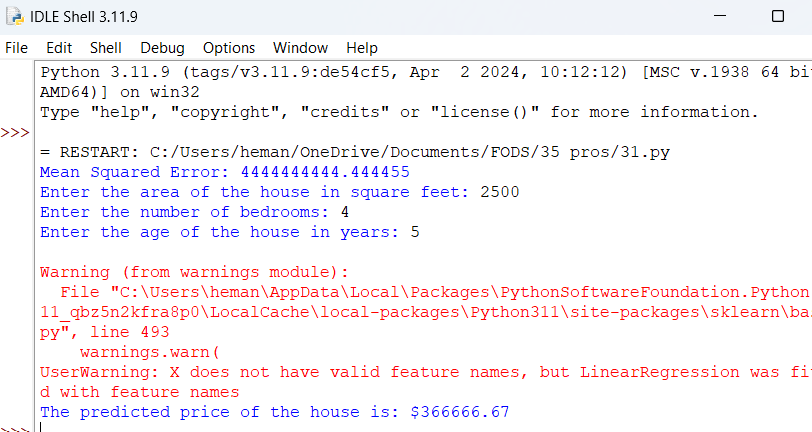
**bedrooms = int(input("Enter the number of bedrooms: "))**

**age = int(input("Enter the age of the house in years: "))**

**predicted\_price = predict\_price(area, bedrooms, age)**

**print(f"The predicted price of the house is: ${predicted\_price:.2f}")**

**OUTPUT :**

****

32.**Question:** Logistic Regression for Customer Churn Prediction

You are working for a telecommunications company, and you want to predict whether a customer will churn (leave the company) based on their usage patterns and demographic data. You have collected a dataset of past customers with their churn status (0 for not churned, 1 for churned) and various features.

Write a Python program that allows the user to input the features (e.g., usage minutes, contract duration) of a new customer. The program should use logistic regression from scikit-learn to predict whether the new customer will churn or not based on the input features.

**PROGRAM :**

**import numpy as np**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score**

**import joblib**

**data = {**

**'usage\_minutes': [200, 150, 300, 400, 250],**

**'contract\_duration': [12, 24, 12, 6, 18],**

**'age': [34, 45, 23, 36, 52],**

**'monthly\_bill': [50, 70, 40, 60, 80],**

**'churn': [0, 1, 0, 1, 0]**

**}**

**df = pd.DataFrame(data)**

**X = df[['usage\_minutes', 'contract\_duration', 'age', 'monthly\_bill']]**

**y = df['churn']**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**model = LogisticRegression()**

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

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

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print(f"Accuracy: {accuracy}")**

**joblib.dump(model, 'churn\_predictor.pkl')**

**def predict\_churn(usage\_minutes, contract\_duration, age, monthly\_bill):**

**model = joblib.load('churn\_predictor.pkl')**

**input\_data = pd.DataFrame([[usage\_minutes, contract\_duration, age, monthly\_bill]], columns=['usage\_minutes', 'contract\_duration', 'age', 'monthly\_bill'])**

**churn\_prediction = model.predict(input\_data)**

**return churn\_prediction[0]**

**usage\_minutes = float(input("Enter the usage minutes: "))**

**contract\_duration = int(input("Enter the contract duration (in months): "))**

**age = int(input("Enter the age of the customer: "))**

**monthly\_bill = float(input("Enter the monthly bill amount: "))**

**churn\_prediction = predict\_churn(usage\_minutes, contract\_duration, age, monthly\_bill)**

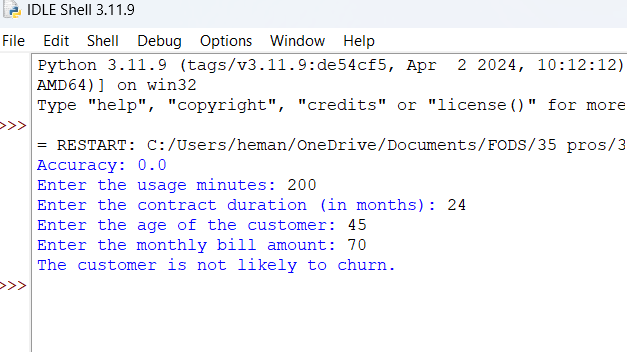
**if churn\_prediction == 1:**

**print("The customer is likely to churn.")**

**else:**

**print("The customer is not likely to churn.")**

**OUTPUT :**

****

33.**Question:** K-Means Clustering for Customer Segmentation

You are working for an e-commerce company and want to segment your customers into distinct groups based on their purchasing behavior. You have collected a dataset of customer data with various shopping-related features.

Write a Python program that allows the user to input the shopping-related features of a new customer. The program should use K-Means clustering from scikit-learn to assign the new customer to one of the existing segments based on the input features.

**PROGRAM :**

**import numpy as np**

**import pandas as pd**

**from sklearn.cluster import KMeans**

**import joblib**

**data = {**

**'annual\_income': [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70],**

**'spending\_score': [39, 81, 6, 77, 40, 76, 6, 94, 3, 72, 48, 70, 90, 32, 49, 63, 52, 39, 70, 45]**

**}**

**df = pd.DataFrame(data)**

**X = df[['annual\_income', 'spending\_score']]**

**num\_clusters = 3**

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

**kmeans.fit(X)**

**joblib.dump(kmeans, 'customer\_segmentation.pkl')**

**def predict\_segment(annual\_income, spending\_score):**

**kmeans = joblib.load('customer\_segmentation.pkl')**

**input\_data = pd.DataFrame([[annual\_income, spending\_score]], columns=['annual\_income', 'spending\_score'])**

**segment = kmeans.predict(input\_data)**

**return segment[0]**

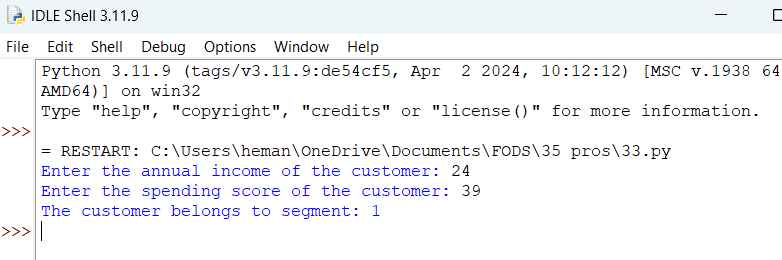
**annual\_income = float(input("Enter the annual income of the customer: "))**

**spending\_score = float(input("Enter the spending score of the customer: "))**

**customer\_segment = predict\_segment(annual\_income, spending\_score)**

**print(f"The customer belongs to segment: {customer\_segment}")**

**OUTPUT :**

****

34.**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.

**PROGRAM :**

**import pandas as pd**

**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 joblib**

**def evaluate\_model(data\_path, model\_path, feature\_names, target\_name):**

**df = pd.read\_csv(data\_path)**

**model = joblib.load(model\_path)**

**X = df[feature\_names]**

**y = df[target\_name]**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**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(f"Accuracy: {accuracy:.2f}")**

**print(f"Precision: {precision:.2f}")**

**print(f"Recall: {recall:.2f}")**

**print(f"F1-score: {f1:.2f}")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**data\_path = input("Enter the path to the dataset (CSV file): ")**

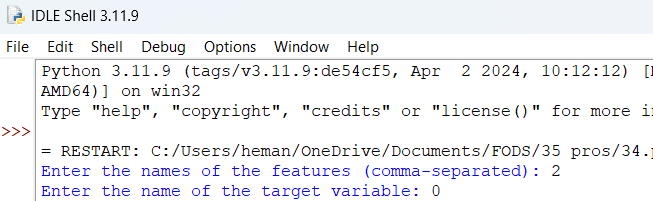
**model\_path = input("Enter the path to the trained model (joblib file): ")**

**feature\_names = input("Enter the names of the features (comma-separated): ").split(',')**

**target\_name = input("Enter the name of the target variable: ")**

**evaluate\_model(data\_path, model\_path, feature\_names, target\_name)**

**OUTPUT :**

****

35. **Question:** You are a data scientist working for an e-commerce company. The marketing team has conducted an A/B test to evaluate the effectiveness of two different website designs (A and B) in terms of conversion rate. They randomly divided the website visitors into two groups, with one group experiencing design A and the other experiencing design B. After a week of data collection, you now have the conversion rate data for both groups. You want to determine whether there is a statistically significant difference in the mean conversion rates between the two website designs.

**Question:**

"Based on the data collected from the A/B test, is there a statistically significant difference in the mean conversion rates between website design A and website design B?

**PROGRAM :**

**import numpy as np**

**import scipy.stats as stats**

**conversion\_rate\_A = np.array([0.10, 0.12, 0.09, 0.11, 0.13])**

**conversion\_rate\_B = np.array([0.15, 0.17, 0.16, 0.18, 0.14])**

**t\_statistic, p\_value = stats.ttest\_ind(conversion\_rate\_A, conversion\_rate\_B)**

**print(f"t-statistic: {t\_statistic}")**

**print(f"p-value: {p\_value}")**

**alpha = 0.05**

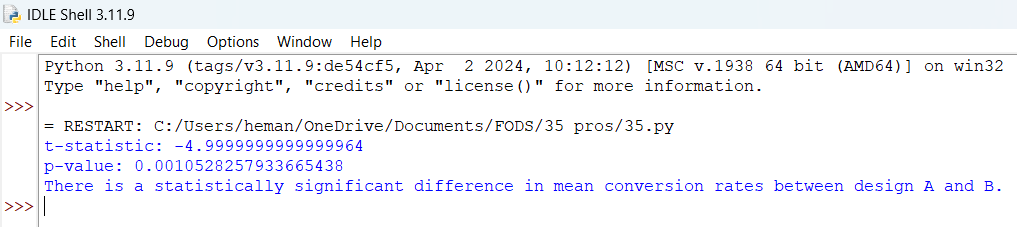
**if p\_value < alpha:**

**print("There is a statistically significant difference in mean conversion rates between design A and B.")**

**else:**

**print("There is no statistically significant difference in mean conversion rates between design A and B.")**

**OUTPUT :**

****