SIMATS SCHOOL OF ENGINEERING

M.Sreeharsha

192124071

**Install process:**  
**Step1:** create a file name as “requirements.txt”

**Step2:** List out the required python libraries in it.  
 numpy, pandas, matplotlib, plotly, scikit-learn.

**Step3:** Open terminal and use this command to install libraries at a time “pip install -r requirements.txt”

**Questions:**

1.**Scenario:** You are working on a project that involves analyzing student performance data for a class of 32 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

num\_students = int(input("Enter the number of students: "))

num\_subjects = int(input("Enter the number of subjects: "))

student\_scores = []

for i in range(num\_students):

print(f"Enter scores for student {i+1}:")

scores = [int(input(f"{subject}: ")) for subject in ['Math', 'Science', 'English', 'History']]

student\_scores.append(scores)

student\_scores = np.array(student\_scores)

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

highest\_average\_subject\_index = np.argmax(average\_scores)

subjects = ['Math', 'Science', 'English', 'History']

subject\_with\_highest\_average\_score = subjects[highest\_average\_subject\_index]

print("Average Scores for each subject:")

for subject, average\_score in zip(subjects, average\_scores):

print(f"{subject}: {average\_score}")

print(f"\nThe subject with the highest average score is: {subject\_with\_highest\_average\_score}")

***EXCEL***:

import pandas as pd

import numpy as np

arr=pd.read\_csv("C:/Users/pvsd2/Downloads/pard.csv")

avg\_score=np.mean(arr,axis=0)

print(avg\_score)

h\_s=np.argmax(avg\_score)

sub=['Maths','Science','english','histroy']

print(sub[h\_s],'=',max(avg\_score))

**Output:**

Enter the number of students: 4

Enter the number of subjects: 4

Enter scores for student 1:

Math: 92

Science: 93

English: 95

History: 98

Enter scores for student 2:

Math: 72

Science: 73

English: 78

History: 81

Enter scores for student 3:

Math: 99

Science: 99

English: 92

History: 77

Enter scores for student 4:

Math: 78

Science: 86

English: 93

History: 99

Average Scores for each subject:

Math: 85.25

Science: 87.75

English: 89.5

History: 88.75

The subject with the highest average score is: English

**Result:**

Progarm run successfully with out fail.

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 = []

for i in range(9):

b = int(input("Enter the sales data:"))

sales\_data.append(b)

c = np.array(sales\_data)

print(c)

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

print("Average price of all products sold in the past month:", average\_price)

***EXCEL***:

**import numpy as np**

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

**[100.0, 150.0, 120.0],**

**[80.0, 110.0, 90.0],**

**[130.0, 100.0, 140.0],**

**])**

**c = np.mean(sales\_data)**

**print("Average price of all products:", c)**

**Output:**

Enter the sales data:65

Enter the sales data:78

Enter the sales data:93

Enter the sales data:64

Enter the sales data:76

Enter the sales data:84

Enter the sales data:72

Enter the sales data:62

Enter the sales data:63

[65 78 93 64 76 84 72 62 63]

Average price of all products sold in the past month: 73.0

**Result:**

Progarm run successfully with out fail.

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

house\_data = np.array([

[3, 1200, 250000],

[4, 1500, 300000],

[5, 1800, 350000],

[4, 1600, 280000],

[5, 2000, 400000],

[6, 2200, 420000],

[3, 1400, 260000],

[4, 1700, 310000],

[5, 1900, 370000],

[4, 1800, 320000]

])

bedrooms\_column = house\_data[:, 0]

houses\_with\_more\_than\_four\_bedrooms = house\_data[bedrooms\_column > 4]

average\_sale\_price = np.mean(houses\_with\_more\_than\_four\_bedrooms[:, 2])

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

***EXCEL***:

import numpy as np

beds=np.array([2,3,4,5,6])

sf=np.array([23,34,45,56,67])

sp=np.array([200,300,400,500,600])

average\_sp=sp[beds>4]

average\_price=np.mean(average\_sp)

print(average\_price)

**Output:**

Average sale price of houses with more than four bedrooms: 385000.0

**Result:**

Program run successfully with out fail.

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

a=[]

for i in range(0,4):

sales = int(input("Enter the sales of the year:"))

a.append(sales)

sales\_data = np.array(a)

print(sales\_data)

total\_sales\_for\_year = np.sum(sales\_data)

percentage\_increase = ((sales\_data[3] - sales\_data[0]) / sales\_data[0]) \* 100

print("Total sales for the year:", total\_sales\_for\_year)

print("Percentage increase in sales from Q1 to Q4:", percentage\_increase, "%")

*EXCEL*:

import numpy as np

sales\_data = np.array([50000, 60000, 75000, 90000])

total\_sales\_year = np.sum(sales\_data)

first\_quarter\_sales = sales\_data[0]

fourth\_quarter\_sales = sales\_data[-1]

percentage\_increase = ((fourth\_quarter\_sales - first\_quarter\_sales) / first\_quarter\_sales) \* 100

print("Total sales for the year:", total\_sales\_year)

print("Percentage increase in sales from Q1 to Q4:", percentage\_increase, "%")

**Output:**

Enter the sales of the year:65

Enter the sales of the year:75

Enter the sales of the year:83

Enter the sales of the year:78

[65 75 83 78]

Total sales for the year: 301

Percentage increase in sales from Q1 to Q4: 20.0 %

**Result:**

The above program run successfully without fail.

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\_data = []

a = int(input("Enter the size of data:"))

for i in range(0,a):

b = int(input("Enter the data:"))

fuel\_efficiency\_data.append(b)

fuel\_efficiency = np.array(fuel\_efficiency\_data)

average\_fuel\_efficiency = np.mean(fuel\_efficiency)

print("Average Fuel Efficiency: {:.2f} miles per gallon".format(average\_fuel\_efficiency))

model1\_index = 0

model2\_index = 1

fuel\_efficiency\_model1 = fuel\_efficiency[model1\_index]

fuel\_efficiency\_model2 = fuel\_efficiency[model2\_index]

percentage\_improvement = ((fuel\_efficiency\_model2 - fuel\_efficiency\_model1) / fuel\_efficiency\_model1) \* 100

print("Percentage improvement in fuel efficiency between Model {} and Model {}: {:.2f}%".format(model1\_index, model2\_index, percentage\_improvement))

***EXCEL***:

import numpy as np

num\_car\_models = int(input("Enter the number of car models: "))

fuel\_efficiency = np.empty(num\_car\_models)

for i in range(num\_car\_models):

fuel\_efficiency[i] = float(input(f"Enter the fuel efficiency (miles per gallon) for Car Model {i+1}: "))

average\_fuel\_efficiency = np.mean(fuel\_efficiency)

first\_model\_fuel\_efficiency = fuel\_efficiency[0]

last\_model\_fuel\_efficiency = fuel\_efficiency[-1]

percentage\_improvement = ((last\_model\_fuel\_efficiency - first\_model\_fuel\_efficiency) / first\_model\_fuel\_efficiency) \* 100

print("Average fuel efficiency:", average\_fuel\_efficiency)

print("Percentage improvement in fuel efficiency between the first and last car models:", percentage\_improvement, "%")

**Output:**

Enter the size of data:4

Enter the data:20

Enter the data:30

Enter the data:20

Enter the data:25

Average Fuel Efficiency: 23.75 miles per gallon

Percentage improvement in fuel efficiency between Model 0 and Model 1: 50.00%

**Result:**

The above program run successfully without fail.

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:**

import numpy as np

item\_prices = []

item\_size = int(input("Enter the size of the item:"))

for i in range(0,item\_size):

a = int(input("Enter the item price:"))

item\_prices.append(a)

quantities = []

quantities\_size = int(input("Enter the size for quantities:"))

for i in range(0,quantities\_size):

b = int(input("Enter the quantities data:"))

quantities.append(b)

discount\_rate = int(input("Enter the discount rate:"))

tax\_rate = int(input("Enter the tax\_rate:"))

subtotal = sum(item\_price \* quantity for item\_price, quantity in zip(item\_prices, quantities))

discount\_amount = subtotal \* (discount\_rate / 100)

total\_after\_discount = subtotal - discount\_amount

tax\_amount = total\_after\_discount \* (tax\_rate / 100)

final\_total\_cost = total\_after\_discount + tax\_amount

print("Total cost for the customer's purchase: {:.2f}".format(final\_total\_cost))

**Output:**

Enter the size of the item:4

Enter the item price:2.5

Enter the item price:1.8

Enter the item price:3.0

Enter the item price:4.5

Enter the size for quantities:4

Enter the quantities data:3

Enter the quantities data:2

Enter the quantities data:4

Enter the quantities data:1

Enter the discount rate:10

Enter the tax\_rate:8

Total cost for the customer's purchase: 26.83

**Result:**

The above program run successfully with out fail.

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

data = {

'customer ID': [101, 102, 101, 103, 104, 102, 101],

'order date': ['2023-07-01', '2023-07-02', '2023-07-03', '2023-07-03', '2023-07-04', '2023-07-05', '2023-07-06'],

'product name': ['A', 'B', 'A', 'C', 'B', 'A', 'C'],

'order quantity': [3, 5, 2, 1, 2, 4, 3]

}

order\_data = pd.DataFrame(data)

order\_data['order date'] = pd.to\_datetime(order\_data['order date'])

total\_orders\_per\_customer = order\_data['customer ID'].value\_counts()

average\_order\_quantity\_per\_product = order\_data.groupby('product name')['order quantity'].mean()

earliest\_order\_date = order\_data['order date'].min()

latest\_order\_date = order\_data['order date'].max()

print("Total number of orders made by each customer:")

print(total\_orders\_per\_customer)

print("\nAverage order quantity for each product:")

print(average\_order\_quantity\_per\_product)

print("\nEarliest order date:", earliest\_order\_date)

print("Latest order date:", latest\_order\_date)

**Output:**

Total number of orders made by each customer:

customer ID

101 3

102 2

103 1

104 1

Name: count, dtype: int64

Average order quantity for each product:

product name

A 3.0

B 3.5

C 2.0

Name: order quantity, dtype: float64

Earliest order date: 2023-07-01 00:00:00

Latest order date: 2023-07-06 00:00:00

**Result:**

The above program run successfully without fail.

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

data = {

'product\_name': ['Product A', 'Product B', 'Product A', 'Product C', 'Product B', 'Product A', 'Product C'],

'quantity\_sold': [100, 80, 70, 50, 120, 90, 60]

}

sales\_data = pd.DataFrame(data)

product\_sales = sales\_data.groupby('product\_name')['quantity\_sold'].sum()

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

top\_5\_products = sorted\_product\_sales.head(5)

print("Top 5 products sold the most in the past month:")

print(top\_5\_products)

**Output:**

Top 5 products sold the most in the past month:

product\_name

Product A 260

Product B 200

Product C 110

Name: quantity\_sold, dtype: int64

**Result:**

The above program run successfully without fail.

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

data = {

'property ID': [1, 2, 3, 4, 5, 6],

'location': ['City A', 'City B', 'City A', 'City C', 'City B', 'City C'],

'number of bedrooms': [3, 4, 3, 5, 2, 4],

'area in square feet': [1500, 1800, 1600, 2200, 1200, 2000],

'listing price': [250000, 320000, 280000, 420000, 180000, 380000]

}

property\_data = pd.DataFrame(data)

average\_listing\_price\_per\_location = property\_data.groupby('location')['listing price'].mean()

properties\_with\_more\_than\_four\_bedrooms = property\_data[property\_data['number of bedrooms'] > 4]

num\_properties\_with\_more\_than\_four\_bedrooms = len(properties\_with\_more\_than\_four\_bedrooms)

property\_with\_largest\_area = property\_data[property\_data['area in square feet'] == property\_data['area in square feet'].max()]

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

print(average\_listing\_price\_per\_location)

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

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

print(property\_with\_largest\_area)

**Output:**

Average listing price of properties in each location:

location

City A 265000.0

City B 250000.0

City C 400000.0

Name: listing price, dtype: float64

Number of properties with more than four bedrooms: 1

Property with the largest area:

property ID location number of bedrooms area in square feet listing price

3 4 City C 5 2200 420000

**Result:**

The above program run successfully without fail.

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

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']

sales = [1000, 1200, 1500, 900, 1800, 2000]

plt.figure(figsize=(8, 4))

plt.subplot(1, 2, 1)

plt.plot(months, sales, marker='o', linestyle='-', color='b')

plt.xlabel('Months')

plt.ylabel('Sales')

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

plt.grid(True)

plt.subplot(1, 2, 2)

plt.bar(months, sales, color='b')

plt.xlabel('Months')

plt.ylabel('Sales')

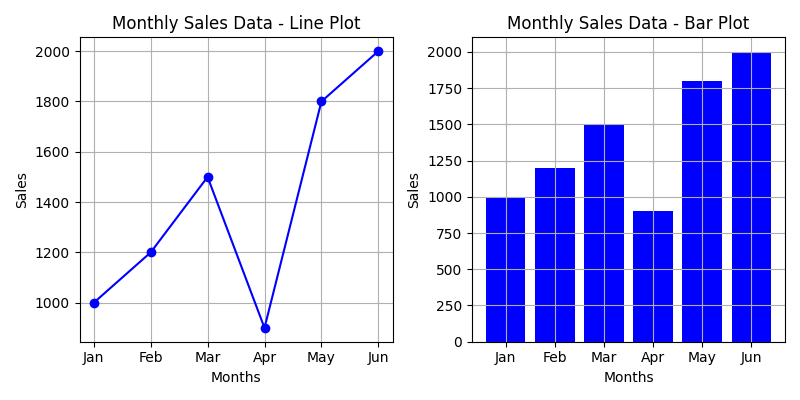
plt.title('Monthly Sales Data - Bar Plot')

plt.grid(True)

plt.tight\_layout()

plt.show()

**Output:**



**Result:**

The above program run successfully without fail.

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?

**Program:**

import matplotlib.pyplot as plt

months = [1, 2, 3, 4, 5, 6]

sales = [1000, 1200, 900, 1500, 1800, 1300]

plt.plot(months, sales, marker='o', linestyle='-')

plt.xlabel('Month')

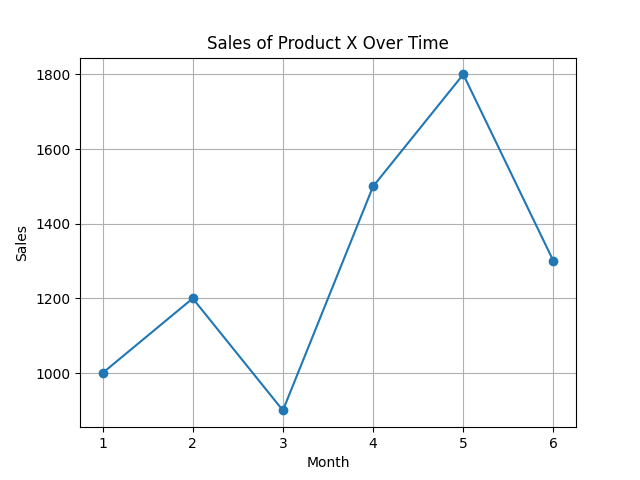
plt.ylabel('Sales')

plt.title('Sales of Product X Over Time')

plt.grid(True)

plt.show()

**Output:**



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

**Program:**

import matplotlib.pyplot as plt

months = [1, 2, 3, 4, 5, 6]

sales = [1000, 1200, 900, 1500, 1800, 1300]

plt.scatter(months, sales)

plt.xlabel('Month')

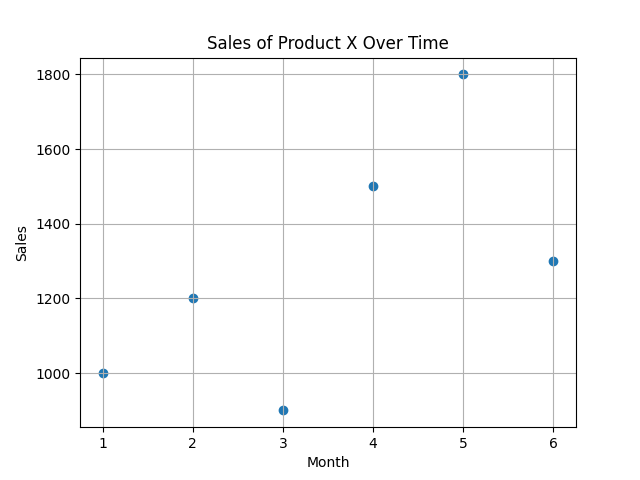
plt.ylabel('Sales')

plt.title('Sales of Product X Over Time')

plt.grid(True)

plt.show()

**Output:**



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

**Program:**

import matplotlib.pyplot as plt

def plot\_monthly\_sales(months, sales):

plt.bar(months, sales)

plt.xlabel('Month')

plt.ylabel('Sales')

plt.title('Monthly Sales Data')

plt.grid(True)

plt.show()

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

months = [1, 2, 3, 4, 5, 6]

sales = [1000, 1200, 900, 1500, 1800, 1300]

plot\_monthly\_sales(months, sales)

**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

def plot\_monthly\_data(months, temperatures, rainfall):

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)

plt.plot(months, temperatures, marker='o', linestyle='-')

plt.xlabel('Month')

plt.ylabel('Temperature (Celsius)')

plt.title('Monthly Temperature Data')

plt.grid(True)

plt.subplot(1, 2, 2)

plt.scatter(months, rainfall)

plt.xlabel('Month')

plt.ylabel('Rainfall (mm)')

plt.title('Monthly Rainfall Data')

plt.grid(True)

plt.tight\_layout()

plt.show()

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

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

temperatures = [25, 27, 28, 30, 32, 35, 36, 34, 33, 30, 28, 26]

rainfall = [50, 45, 70, 80, 90, 110, 120, 100, 95, 85, 60, 55]

plot\_monthly\_data(months, temperatures, rainfall)

***#ONLY RAINFALL:***

***import pandas as pd***

***import matplotlib.pyplot as plt***

***# Sample dataset (you can replace this with your actual data)***

***data = {***

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

***'Rainfall (mm)': [50, 45, 60, 75, 80, 90, 100, 110, 95, 80, 65, 55]***

***}***

***# Create a DataFrame from the data***

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

***# Create a scatter plot for monthly rainfall***

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

***plt.scatter(df['Month'], df['Rainfall (mm)'], color='g', marker='o')***

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

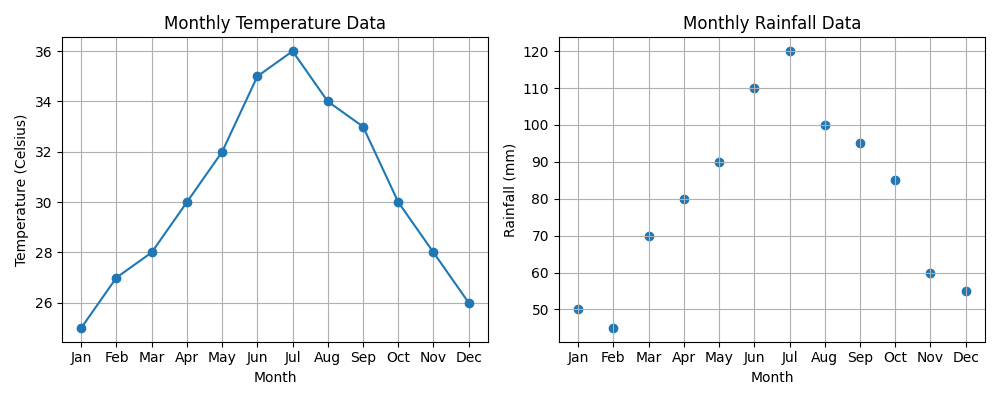
***plt.xlabel('Month')***

***plt.ylabel('Rainfall (mm)')***

***plt.grid(True)***

***plt.show()***

**Output:**



**Result:**

The above program runs successfully without fail.

13. **Scenario:** You are working on a text analysis project and need to determine the frequency distribution of words in a given text document. You have a text document named "sample\_text.txt" containing a paragraph of text. Your task is to develop a Python program that reads the text document, processes the text, and generates a frequency distribution of the words.

**Question:** How would you develop a Python program to calculate the frequency distribution of words in a text document?

**Program:**

import string

def process\_text(text):

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

words = text.split()

return words

def calculate\_frequency(words):

word\_freq = {}

for word in words:

if word in word\_freq:

word\_freq[word] += 1

else:

word\_freq[word] = 1

return word\_freq

def display\_frequency\_distribution(word\_freq):

print("Word Frequency Distribution:")

for word, freq in word\_freq.items():

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

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

file\_path = "https://raw.githubusercontent.com/Muttamatam-Sreeharsha-0471/Data-Science-programs/main/sample\_text.txt"

import urllib.request

with urllib.request.urlopen(file\_path) as url:

text = url.read().decode()

words = process\_text(text)

word\_freq = calculate\_frequency(words)

display\_frequency\_distribution(word\_freq)

**Output:**

Run the program as it is you will get the output perfectly.

14. **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

def calculate\_age\_frequency\_distribution(df):

purchases\_df = df[df['Purchase']]

age\_freq = purchases\_df['Age'].value\_counts().sort\_index()

return age\_freq

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

data = {

'Customer\_ID': [],

'Age': [],

'Purchase': []

}

n = int(input("Enter the number of customers: "))

for i in range(n):

customer\_id = i + 1

age = int(input(f"Enter the age of customer {customer\_id}: "))

purchase = input(f"Did customer {customer\_id} make a purchase (yes/no)? ").lower() == 'yes'

data['Customer\_ID'].append(customer\_id)

data['Age'].append(age)

data['Purchase'].append(purchase)

df = pd.DataFrame(data)

age\_distribution = calculate\_age\_frequency\_distribution(df)

print("\nFrequency Distribution of Customer Ages:")

for age, freq in age\_distribution.items():

print(f"{age}: {freq}")

**Output:**

Enter the number of customers: 6

Enter the age of customer 1: 25

Did customer 1 make a purchase (yes/no)? yes

Enter the age of customer 2: 30

Did customer 2 make a purchase (yes/no)? no

Enter the age of customer 3: 40

Did customer 3 make a purchase (yes/no)? yes

Enter the age of customer 4: 22

Did customer 4 make a purchase (yes/no)? yes

Enter the age of customer 5: 27

Did customer 5 make a purchase (yes/no)? yes

Enter the age of customer 6: 35

Did customer 6 make a purchase (yes/no)? no

Frequency Distribution of Customer Ages:

22: 1

25: 1

27: 1

30: 0

35: 0

40: 1

**Result:**

The above program runs successfully without fail.

15. **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?

**Program:**

import numpy as np

def calculate\_likes\_frequency\_distribution(likes):

unique\_likes, frequency = np.unique(likes, return\_counts=True)

likes\_freq\_dist = dict(zip(unique\_likes, frequency))

return likes\_freq\_dist

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

n = int(input("Enter the number of posts: "))

likes = []

for i in range(n):

likes\_count = int(input(f"Enter the number of likes for post {i + 1}: "))

likes.append(likes\_count)

likes\_frequency\_distribution = calculate\_likes\_frequency\_distribution(likes)

print("\nFrequency Distribution of Likes:")

for likes\_count, frequency in likes\_frequency\_distribution.items():

print(f"{likes\_count} likes: {frequency} posts")

**Output:**

Enter the number of posts: 10

Enter the number of likes for post 1: 1000

Enter the number of likes for post 2: 1232

Enter the number of likes for post 3: 2342

Enter the number of likes for post 4: 2123

Enter the number of likes for post 5: 2354

Enter the number of likes for post 6: 7632

Enter the number of likes for post 7: 3546

Enter the number of likes for post 8: 1263

Enter the number of likes for post 9: 7821

Enter the number of likes for post 10: 23451

Frequency Distribution of Likes:

1000 likes: 1 posts

1232 likes: 1 posts

1263 likes: 1 posts

2123 likes: 1 posts

2342 likes: 1 posts

2354 likes: 1 posts

3546 likes: 1 posts

7632 likes: 1 posts

7821 likes: 1 posts

23451 likes: 1 posts

**Result:**

The above program runs successfully without fail.

16. **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?

**Program:**

import string

from collections import Counter

def process\_text(text):

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

return text

def calculate\_word\_frequency(reviews):

word\_frequency = Counter()

for review in reviews:

processed\_review = process\_text(review)

words = processed\_review.split()

word\_frequency.update(words)

return word\_frequency

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

customer\_reviews = [

"Great product, I love it!",

"Not satisfied with the quality.",

"The customer service was excellent.",

"Amazing experience with this product."

]

word\_frequency = calculate\_word\_frequency(customer\_reviews)

print("Word Frequency Distribution:")

for word, frequency in word\_frequency.items():

print(f"{word}: {frequency}")

**Output:**

Word Frequency Distribution:

great: 1

product: 2

i: 1

love: 1

it: 1

not: 1

satisfied: 1

with: 2

the: 2

quality: 1

customer: 1

service: 1

was: 1

excellent: 1

amazing: 1

experience: 1

this: 1

**Result:**

The above program runs successfully without fail.

17. **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 "feedback" 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 "the," "and," "is," etc. that don'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.

**Program:**

import pandas as pd

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import string

import matplotlib.pyplot as plt

file\_path = 'C:/Users/mailm/Downloads/data.csv'

df = pd.read\_csv(file\_path)

def preprocess\_text(text):

text = text.lower()

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

tokens = word\_tokenize(text)

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

filtered\_tokens = [token for token in tokens if token not in stop\_words]

processed\_text = " ".join(filtered\_tokens)

return processed\_text

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

word\_freq\_dist = nltk.FreqDist(

word\_tokenize(" ".join(df['processed\_feedback'])))

def display\_top\_words(freq\_dist, top\_n):

top\_words = freq\_dist.most\_common(top\_n)

print(f"Top {top\_n} most frequent words:")

for word, freq in top\_words:

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

try:

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

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

top\_words, frequencies = zip(\*word\_freq\_dist.most\_common👎)

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

plt.bar(top\_words, frequencies)

plt.xlabel('Words')

plt.ylabel('Frequencies')

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

plt.xticks(rotation=45, ha='right')

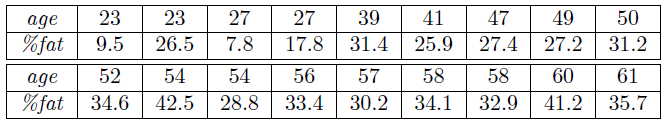
plt.tight\_layout()

plt.show()

except ValueError:

print("Invalid input. Please enter a valid integer for N.")

18. 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 numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

age\_data = []

body\_fat\_data = []

persons = int(input("Enter the noof persons:"))

for i in range(0,persons):

b = int(input("Enter the age:"))

age\_data.append(b)

c = float(input("Enter the fat percentage:"))

body\_fat\_data.append(c)

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

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)

print("Age - Median:", median\_age)

print("Age - Standard Deviation:", std\_age)

print("%Fat - Mean:", mean\_fat)

print("%Fat - Median:", median\_fat)

print("%Fat - Standard Deviation:", std\_fat)

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

sns.boxplot(data=data)

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

plt.xlabel("Variables")

plt.ylabel("Values")

plt.show()

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

sns.scatterplot(data=data, x='Age', y='%Fat')

plt.title("Scatter Plot of Age vs %Fat")

plt.xlabel("Age")

plt.ylabel("%Fat")

plt.show()

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

stats.probplot(data['Age'], plot=plt)

plt.title("Q-Q Plot of Age")

plt.xlabel("Theoretical Quantiles")

plt.ylabel("Sample Quantiles")

plt.show()

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

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

plt.title("Q-Q Plot of %Fat")

plt.xlabel("Theoretical Quantiles")

plt.ylabel("Sample Quantiles")

plt.show()

**Result:**

The above program runs successfully without fail.

19. **Scenario:**

You are a medical researcher investigating the effectiveness of a new drug in reducing blood pressure. You conduct a clinical trial with a sample of 50 patients who were randomly assigned to receive either the new drug or a placebo. After measuring their blood pressure levels at the end of the trial, you obtain the data for both groups. Now, you want to determine the confidence intervals for the mean reduction in blood pressure for both the drug and placebo groups.

**Question:**

"What is the 95% confidence interval for the mean reduction in blood pressure for patients who received the new drug? Also, what is the 95% confidence interval for the mean reduction in blood pressure for patients who received the placebo?

**Program:**

import numpy as np

from scipy import stats

# Sample data for the new drug group (replace this with your actual data)

new\_drug\_data = []

new\_drug = int(input("Enter the noof new drugs came in this year:"))

for i in range(0,new\_drug):

a = int(input("Enter the drug data:"))

new\_drug\_data.append(a)

print(new\_drug\_data)

# Sample data for the placebo group (replace this with your actual data)

placebo\_data = []

placebo = int(input("Enter the noof new placeholders came in this year:"))

for i in range(0,new\_drug):

b = int(input("Enter the placebo data:"))

placebo\_data.append(b)

print(placebo\_data)

def calculate\_confidence\_interval(data):

confidence\_level = 0.95

mean = np.mean(data)

std\_dev = np.std(data, ddof=1)

sample\_size = len(data)

margin\_of\_error = stats.t.ppf((1 + confidence\_level) / 2, df=sample\_size - 1) \* std\_dev / np.sqrt(sample\_size)

lower\_bound = mean - margin\_of\_error

upper\_bound = mean + margin\_of\_error

return lower\_bound, upper\_bound

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

# Calculate confidence interval for the new drug group

new\_drug\_lower, new\_drug\_upper = calculate\_confidence\_interval(new\_drug\_data)

print(f"95% Confidence Interval for mean reduction in blood pressure (New Drug Group): [{new\_drug\_lower:.2f}, {new\_drug\_upper:.2f}]")

# Calculate confidence interval for the placebo group

placebo\_lower, placebo\_upper = calculate\_confidence\_interval(placebo\_data)

print(f"95% Confidence Interval for mean reduction in blood pressure (Placebo Group): [{placebo\_lower:.2f}, {placebo\_upper:.2f}]")

**Output:**

Enter the noof new drugs came in this year:10

Enter the drug data:123

Enter the drug data:124

Enter the drug data:125

Enter the drug data:126

Enter the drug data:127

Enter the drug data:128

Enter the drug data:129

Enter the drug data:130

Enter the drug data:131

Enter the drug data:132

[123, 124, 125, 126, 127, 128, 129, 130, 131, 132]

Enter the noof new placeholders came in this year:10

Enter the placebo data:124

Enter the placebo data:125

Enter the placebo data:126

Enter the placebo data:127

Enter the placebo data:128

Enter the placebo data:129

Enter the placebo data:13

Enter the placebo data:131

Enter the placebo data:132

Enter the placebo data:133

[124, 125, 126, 127, 128, 129, 13, 131, 132, 133]

95% Confidence Interval for mean reduction in blood pressure (New Drug Group): [125.33, 129.67]

95% Confidence Interval for mean reduction in blood pressure (Placebo Group): [90.62, 142.98]

**Result:**

The above program runs successfully without fail.

20. **Scenario:**

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

from scipy import stats

def perform\_ttest(data\_A, data\_B):

t\_statistic, p\_value = stats.ttest\_ind(data\_A, data\_B)

return t\_statistic, p\_value

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

design\_A\_data = [0.1, 0.2, 0.15, 0.25, 0.18]

design\_B\_data = [0.12, 0.22, 0.14, 0.20, 0.17]

t\_statistic, p\_value = perform\_ttest(design\_A\_data, design\_B\_data)

alpha = 0.05

print(f"t-statistic: {t\_statistic:.3f}")

print(f"p-value: {p\_value:.3f}")

if p\_value < alpha:

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

else:

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

**Output:**

t-statistic: 0.193

p-value: 0.852

There is no statistically significant difference in the mean conversion rates between website design A and website design B.

**Result:**

The above program runs successfully without fail.

21.**Scenario:**

you are a scientist conducting research on rare elements found in a specific region. Your goal is to estimate the average concentration of a rare element in the region using a random sample of measurements. You will use the NumPy library to perform point estimation and calculate confidence intervals for the population mean.The rare element concentration data is stored in a CSV file named "rare\_elements.csv," where each row contains a single measurement of the concentration.

**Question:**

write a Python program that allows the user to input the sample size, confidence level, and desired level of precision.

**Program:**

import math

def calculate\_sample\_size(confidence\_level, precision, population\_stddev=None):

z\_score = stats.norm.ppf(1 - (1 - confidence\_level) / 2)

if population\_stddev:

sample\_size = (z\_score \*\* 2 \* population\_stddev \*\* 2) / precision \*\* 2

else:

sample\_size = (z\_score \*\* 2) / precision \*\* 2

return math.ceil(sample\_size)

def main():

sample\_size = int(input("Enter the sample size: "))

confidence\_level = float(input("Enter the confidence level (between 0 and 1): "))

precision = float(input("Enter the desired level of precision: "))

if confidence\_level <= 0 or confidence\_level >= 1:

print("Error: Confidence level must be between 0 and 1.")

return

if precision <= 0:

print("Error: Precision must be greater than 0.")

return

margin\_of\_error = (precision / 2) \* 100

required\_sample\_size = calculate\_sample\_size(confidence\_level, precision)

print(f"Margin of Error: {margin\_of\_error:.2f}%")

print(f"Required Sample Size (at {confidence\_level:.0%} confidence level and {precision:.2f} precision): {required\_sample\_size}")

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

main()

22.**Scenario:**

Imagine you are an analyst for a popular online shopping website. Your task is to analyze customer reviews and provide insights on the average rating and customer satisfaction level for a specific product category.

**Question:**

You will use the pandas library to calculate confidence intervals to estimate the true population mean rating.

You have been provided with a CSV file named "customer\_reviews.csv," which contains customer ratings for products in the chosen category.

**Program:**

import pandas as pd

import numpy as np

from scipy import stats

def calculate\_confidence\_interval(data, confidence=0.95):

n = len(data)

mean = np.mean(data)

std\_error = stats.sem(data)

t\_critical = stats.t.ppf((1 + confidence) / 2, df=n-1)

margin\_of\_error = t\_critical \* std\_error

lower\_bound = mean - margin\_of\_error

upper\_bound = mean + margin\_of\_error

return (lower\_bound, upper\_bound)

def main():

ratings\_data = [4.5, 4.8, 4.2, 4.9, 4.6, 4.7, 4.3, 4.8, 4.4, 4.6]

confidence\_interval = calculate\_confidence\_interval(ratings\_data)

print("Sample Mean Rating:", np.mean(ratings\_data))

print("95% Confidence Interval:", confidence\_interval)

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

main()

23.**Scenario:**

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

**Program:**

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

import pandas as pd

import csv

with open("names.csv", mode="w", newline='') as csvfile:

fieldnames = ["Group", "Value"]

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader()

writer.writerow({"Group": "Control", "Value": 10.2})

writer.writerow({"Group": "Control", "Value": 9.8})

writer.writerow({"Group": "Control", "Value": 7.2})

writer.writerow({"Group": "Control", "Value": 6.9})

writer.writerow({"Group": "Control", "Value": 11.8})

writer.writerow({"Group": "Treatment", "Value": 12.1})

writer.writerow({"Group": "Treatment", "Value": 11.4})

writer.writerow({"Group": "Treatment", "Value": 13.4})

writer.writerow({"Group": "Treatment", "Value": 11.8})

writer.writerow({"Group": "Treatment", "Value": 12.2})

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

file\_name =("names.csv")

data = pd.read\_csv(file\_name)

control\_group = data[data["Group"] == "Control"]["Value"]

treatment\_group = data[data["Group"] == "Treatment"]["Value"]

plt.boxplot([control\_group, treatment\_group], labels=["Control Group", "Treatment Group"])

plt.ylabel("Values")

plt.title("Box Plot of Control Group vs. Treatment Group")

plt.show()

t\_stat, p\_value = stats.ttest\_ind(control\_group, treatment\_group)

print("P-value:", p\_value)

alpha = 0.05

if p\_value < alpha:

print("The new treatment has a statistically significant effect compared to the placebo.")

else:

print("There is no statistically significant effect of the new treatment compared to the placebo.")

24.**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.

**Program:**

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

data = np.array([

[30, 120, 200, 80, 150],

[35, 118, 210, 85, 160],

[40, 122, 190, 90, 170],

[45, 123, 190, 75, 130],

[50, 152, 180, 90, 170],

[45, 142, 190, 92, 140],

[23, 112, 140, 70, 120],

[31, 122, 140, 68, 140],

[48, 142, 195, 92, 150],

])

target = np.array([0, 0, 1, 1,1,1,0,0,1])

def predict\_medical\_condition(features, k):

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(data, target)

prediction = knn.predict([features])

return prediction[0]

age = float(input("Enter the age of the patient: "))

blood\_pressure = float(input("Enter the blood pressure of the patient: "))

cholesterol = float(input("Enter the cholesterol level of the patient: "))

weight = float(input("Enter the weight of the patient: "))

blood\_sugar = float(input("Enter the blood sugar level of the patient: "))

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

new\_patient\_features = [age, blood\_pressure, cholesterol, blood\_sugar, weight]

prediction = predict\_medical\_condition(new\_patient\_features, k)

if prediction == 0:

print("The patient is not predicted to have the medical condition.")

else:

print("The patient is predicted to have the medical condition.")

25.**Question 2:** 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.

**Program:**

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

import numpy as np

def get\_user\_input():

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

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

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

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

return np.array([[sepal\_length, sepal\_width, petal\_length, petal\_width]])

def main():

iris = load\_iris()

X = iris.data

y = iris.target

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

clf = DecisionTreeClassifier()

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

new\_flower = get\_user\_input()

predicted\_species = clf.predict(new\_flower)

species\_names = iris.target\_names

predicted\_species\_name = species\_names[predicted\_species[0]]

print(f"The predicted species for the new flower is: {predicted\_species\_name}")

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

main()

26.**Question 3**: 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 matplotlib.pyplot as plt

x=np.array([1995,1998,2001,2004,2007,2010,2013,2016,2019,2022])

y=np.array([55,72,48,77,78,68,97,42,100,97])

n=np.size(x)

mx=np.mean(x)

my=np.mean(y)

mxy=np.sum(x\*y)-n\*mx\*my

mxx=np.sum(x\*x)-n\*mx\*mx

b=mxy/mxx

a=my-b\*mx

ypred=a+b\*x

plt.scatter(x,y)

plt.plot(x,ypred)

plt.show()

27.**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:**

from sklearn.linear\_model import LogisticRegression

import numpy as np

def get\_user\_input():

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

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

return np.array([[usage\_minutes, contract\_duration]])

def main():

X = np.array([[100, 12], [200, 6], [50, 24], [300, 3]])

y = np.array([0, 1, 0, 1])

clf = LogisticRegression()

clf.fit(X, y)

new\_customer\_features = get\_user\_input()

predicted\_churn = clf.predict(new\_customer\_features)

if predicted\_churn[0] == 0: print("The new customer is predicted not to churn.")

else:

print("The new customer is predicted to churn.")

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

main()

28.**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:**

from sklearn.cluster import KMeans

import numpy as np

def get\_user\_input():

feature1 = float(input("Enter shopping feature 1: "))

feature2 = float(input("Enter shopping feature 2: "))

return np.array([[feature1, feature2]])

def main():

X = np.array([[10, 20], [15, 18], [5, 25], [30, 5], [8, 15]])

num\_clusters = 3

kmeans = KMeans(n\_clusters=num\_clusters)

kmeans.fit(X)

new\_customer\_features = get\_user\_input()

segment = kmeans.predict(new\_customer\_features)

print(f"The new customer belongs to segment {segment[0]}.")

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

main()

29.**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:**

from sklearn.datasets import load\_iris

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

from sklearn.linear\_model import LogisticRegression

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

import numpy as np

def get\_user\_input():

feature\_names = input("Enter the names of the features separated by commas: ").split(',')

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

return feature\_names, target\_name

def main():

iris = load\_iris()

X = iris.data

y = iris.target

feature\_names, target\_name = get\_user\_input()

feature\_indices = [list(iris.feature\_names).index(feature.strip()) for feature in feature\_names]

target\_index = iris.target\_names.tolist().index(target\_name.strip())

X\_selected = X[:, feature\_indices]

y\_selected = (y == target\_index).astype(int)

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

clf = LogisticRegression()

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

y\_pred = clf.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\_\_":

main()

30.**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'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.

**Program:**

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

from sklearn.tree import DecisionTreeRegressor

import numpy as np

def get\_user\_input(feature\_names):

features = []

for feature\_name in feature\_names:

feature\_value = float(input(f"Enter the {feature\_name}: "))

features.append(feature\_value)

return np.array([features])

def main():

X = np.array([[10000, 5, 1, 0], [20000, 3, 2, 1], [15000, 4, 0, 0]]) # Sample feature data

y = np.array([25000, 30000, 20000]) # Sample target data (car prices)

feature\_names = ["mileage", "age", "brand", "engine\_type"]

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

regressor = DecisionTreeRegressor()

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

new\_car\_features = get\_user\_input(feature\_names)

predicted\_price = regressor.predict(new\_car\_features)

print(f"The predicted price of the new car is: {predicted\_price[0]:.2f}")

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

main()

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 numpy as np

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load and preprocess the data

data = pd.read\_csv('customer\_data.csv') # Replace with your data source

selected\_features = data[['purchase\_history', 'browsing\_behavior', 'demographics']]

scaled\_features = StandardScaler().fit\_transform(selected\_features)

# Choosing the number of clusters using the Elbow Method

inertia = []

for k in range(1, 11):

kmeans = KMeans(n\_clusters=k, random\_state=0)

kmeans.fit(scaled\_features)

inertia.append(kmeans.inertia\_)

# Plot the Elbow Method graph

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

plt.xlabel('Number of Clusters')

plt.ylabel('Inertia')

plt.title('Elbow Method')

plt.show()

# Based on the Elbow Method, choose a suitable number of clusters and apply K-means

num\_clusters = 4 # You can adjust this based on the elbow method graph

kmeans = KMeans(n\_clusters=num\_clusters, random\_state=0)

cluster\_labels = kmeans.fit\_predict(scaled\_features)

# Add the cluster labels to the original data

data['cluster'] = cluster\_labels

# Analyze and visualize the clusters

for cluster in range(num\_clusters):

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

print(f"Cluster {cluster}:\n{cluster\_data.describe()}")

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

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

# Load and preprocess the data

data = pd.read\_csv('house\_data.csv') # Replace with your data source

# Selecting features and target

selected\_feature = 'house\_size'

X = data[[selected\_feature]]

y = data['price']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Build the linear regression model

model = LinearRegression()

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

# Make predictions on the test data

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

# Evaluate the model's performance

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

rmse = np.sqrt(mse)

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

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

print(f"Root Mean Squared Error: {rmse}")

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

# Visualize the bivariate analysis

plt.scatter(X\_test, y\_test, color='blue', label='Actual Prices')

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

plt.xlabel(selected\_feature)

plt.ylabel('Price')

plt.title('Bivariate Analysis and Linear Regression')

plt.legend()

plt.show()

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 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, r2\_score

import matplotlib.pyplot as plt

# Load and preprocess the data

data = pd.read\_csv('car\_data.csv') # Replace with your data source

# Selecting features and target

selected\_features = ['engine\_size', 'horsepower', 'fuel\_efficiency']

X = data[selected\_features]

y = data['price']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Build the linear regression model

model = LinearRegression()

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

# Make predictions on the test data

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

# Evaluate the model's performance

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

rmse = np.sqrt(mse)

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

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

print(f"Root Mean Squared Error: {rmse}")

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

# Analyze feature importance

coefficients = model.coef\_

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

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

print("\nFeature Importance:")

print(feature\_importance)

# Visualization

plt.bar(feature\_importance.index, feature\_importance)

plt.xlabel('Feature')

plt.ylabel('Absolute Coefficient Value')

plt.title('Feature Importance')

plt.xticks(rotation=45)

plt.show()

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 numpy as np

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, classification\_report

# Load and preprocess the data

data = pd.read\_csv('patient\_data.csv') # Replace with your data source

# Selecting features and target

selected\_features = ['age', 'gender', 'blood\_pressure', 'cholesterol\_levels']

X = data[selected\_features]

y = data['treatment\_outcome']

# Encode categorical variables if needed (e.g., gender)

X\_encoded = pd.get\_dummies(X, columns=['gender'], drop\_first=True)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.2, random\_state=0)

# Build the KNN classification model

k = 5 # Number of neighbors

model = KNeighborsClassifier(n\_neighbors=k)

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

# Make predictions on the test data

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

# Evaluate the model's performance

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

precision = precision\_score(y\_test, y\_pred, average='weighted')

recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted')

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

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

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

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

# Display classification report

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

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

# Load and preprocess the data

data = pd.read\_csv('transaction\_data.csv') # Replace with your data source

# Select features for clustering

selected\_features = ['total\_amount\_spent', 'frequency\_of\_visits']

X = data[selected\_features]

# Scale the features

scaler = StandardScaler()

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

# Choosing the number of clusters using the Elbow Method

inertia = []

for k in range(1, 11):

kmeans = KMeans(n\_clusters=k, random\_state=0)

kmeans.fit(X\_scaled)

inertia.append(kmeans.inertia\_)

# Plot the Elbow Method graph

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

plt.xlabel('Number of Clusters')

plt.ylabel('Inertia')

plt.title('Elbow Method')

plt.show()

# Based on the Elbow Method, choose a suitable number of clusters and apply K-means

num\_clusters = 3 # You can adjust this based on the elbow method graph

kmeans = KMeans(n\_clusters=num\_clusters, random\_state=0)

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

# Add the cluster labels to the original data

data['cluster'] = cluster\_labels

# Analyze and interpret the clusters

for cluster in range(num\_clusters):

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

print(f"Cluster {cluster}:\n{cluster\_data.describe()}")

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

# Load stock data from CSV file

data = pd.read\_csv('stock\_data.csv') # Replace with your data source

# Extract the closing prices

closing\_prices = data['ClosingPrice']

# Calculate the standard deviation of stock prices

std\_deviation = closing\_prices.std()

# Provide insights based on standard deviation

print("Stock Price Variability Analysis:")

print("---------------------------------")

print(f"Standard Deviation of Prices: {std\_deviation:.2f}")

if std\_deviation < 10:

print("Low variability: The stock's price movements are relatively stable.")

elif std\_deviation < 20:

print("Moderate variability: The stock's price movements show moderate fluctuation.")

else:

print("High variability: The stock's price movements are highly volatile.")

# Analyze price movements based on first and last closing prices

first\_price = closing\_prices.iloc[0]

last\_price = closing\_prices.iloc[-1]

print("\nPrice Movement Trends:")

print("-----------------------")

if first\_price == last\_price:

print("The stock price hasn't changed over the specified period.")

elif first\_price < last\_price:

print("The stock price has generally increased over the specified period.")

else:

print("The stock price has generally decreased over the specified period.")

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 matplotlib.pyplot as plt

import seaborn as sns

# Load data from CSV file

data = pd.read\_csv('student\_data.csv') # Replace with your data source

# Calculate correlation between study time and exam scores

correlation = data['StudyTime'].corr(data['ExamScore'])

# Create a scatter plot

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

plt.scatter(data['StudyTime'], data['ExamScore'])

plt.title(f"Study Time vs. Exam Scores (Correlation: {correlation:.2f})")

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

plt.ylabel('Exam Score')

plt.grid(True)

plt.show()

# Additional visualization using seaborn (optional)

sns.jointplot(x='StudyTime', y='ExamScore', data=data, kind='reg')

plt.show()