1. **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 numpy as np

import pandas as pd

import scipy.stats as stats

data = pd.read\_csv("C:/Users/krish/OneDrive/Documents/ds/rare.csv")

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: "))

sample = np.random.choice(data['concentration'], size=sample\_size)

sample\_mean = np.mean(sample)

sample\_std = np.std(sample, ddof=1)

if sample\_size > 30:

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

else:

t\_critical = stats.t.ppf(1 - (1 - confidence\_level) / 2, df=sample\_size - 1)

margin\_of\_error = precision \* (sample\_std / np.sqrt(sample\_size))

if sample\_size > 30:

confidence\_interval = (

sample\_mean - z\_critical \* margin\_of\_error,

sample\_mean + z\_critical \* margin\_of\_error

)

else:

confidence\_interval = (

sample\_mean - t\_critical \* margin\_of\_error,

sample\_mean + t\_critical \* margin\_of\_error

)

print(f"Sample Mean: {sample\_mean:.4f}")

print(f"Standard Error: {sample\_std:.4f}")

print(f"Margin of Error: {margin\_of\_error:.4f}")

print(f"Confidence Interval ({confidence\_level\*100}%): {confidence\_interval[0]:.4f} - {confidence\_interval[1]:.4f}")

**Output:**

Enter the sample size: 10

Enter the confidence level (between 0 and 1): 0

Enter the desired level of precision: 8

Sample Mean: 0.0308

Standard Error: 0.0095

Margin of Error: 0.0239

Confidence Interval (0.0%): 0.0308 - 0.0308

1. **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 scipy.stats as stats

data = pd.DataFrame({

'Product': ['Laptop', 'Headphones', 'Phone', 'Sneakers', 'Dress', 'Book', 'Tablet', 'Shoes',

'Smartwatch', 'Camera', 'Backpack', 'Watch', 'Speaker', 'Headset', 'Guitar', 'Ebook', 'Handbag'],

'Category': ['Electronics', 'Electronics', 'Electronics', 'Fashion', 'Fashion', 'Books', 'Electronics', 'Fashion',

'Electronics', 'Electronics', 'Fashion', 'Fashion', 'Electronics', 'Electronics', 'Music', 'Books', 'Fashion'],

'Rating': [4.5, 4.2, 4.8, 4.6, 4.0, 4.3, 4.7, 4.5, 4.4, 4.9, 4.2, 4.7, 4.3, 4.6, 4.8, 4.5, 4.1]

})

product\_category = 'Electronics'

category\_reviews = data[data['Category'] == product\_category]

sample\_mean = category\_reviews['Rating'].mean()

sample\_std = category\_reviews['Rating'].std()

sample\_size = len(category\_reviews)

confidence\_level = 0.95

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

margin\_of\_error = t\_critical \* (sample\_std / (sample\_size \*\* 0.5))

confidence\_interval = (sample\_mean - margin\_of\_error, sample\_mean + margin\_of\_error)

print(f"Sample Mean Rating for {product\_category}: {sample\_mean:.2f}")

print(f"Confidence Interval ({confidence\_level \* 100}%): ({confidence\_interval[0]:.2f}, {confidence\_interval[1]:.2f})")

**Output:**

Sample Mean Rating for Electronics: 4.55

Confidence Interval (95.0%): (4.35, 4.75)

1. **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 scipy.stats as stats

import matplotlib.pyplot as plt

control\_group = [85, 88, 84, 79, 91, 86, 87, 83, 80, 82]

treatment\_group = [78, 92, 89, 95, 90, 93, 96, 85, 88, 91]

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

alpha = 0.05

plt.figure(figsize=(8, 5))

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

plt.title('Box Plot of Control vs. Treatment Groups')

plt.ylabel('Values')

plt.axhline(np.mean(control\_group), color='red', linestyle='--', label='Control Mean')

plt.axhline(np.mean(treatment\_group), color='green', linestyle='--', label='Treatment Mean')

plt.legend()

plt.text(1.2, 95, f'p-value: {p\_value:.4f}', fontsize=12, color='blue')

plt.show()

if p\_value < alpha:

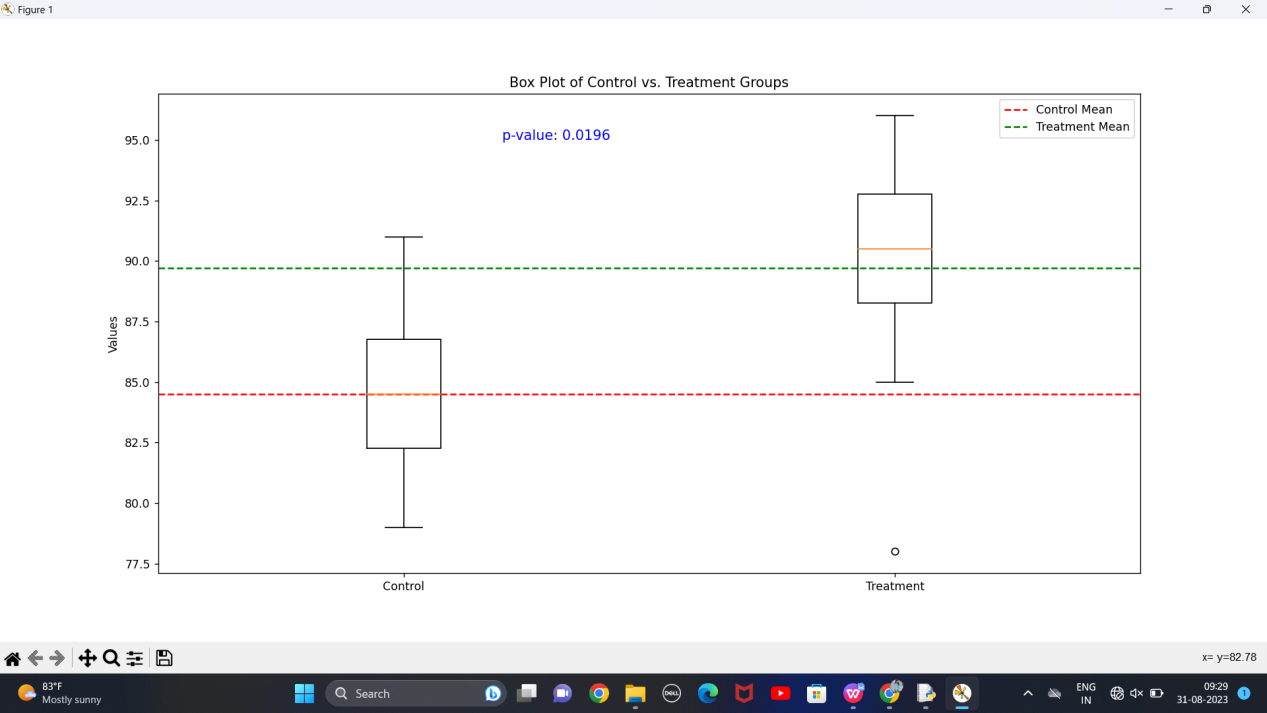
print("Reject the null hypothesis. The new treatment has a statistically significant effect.")

else:

print("Fail to reject the null hypothesis. The new treatment does not have a statistically significant effect.")

**Output:**

Reject the null hypothesis. The new treatment has a statistically significant effect.

65

1. **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.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

data = np.genfromtxt('C:/Users/krish/OneDrive/Documents/ds/medical.csv',delimiter=',', skip\_header=True)

features = data[:, :-1]

labels = data[:, -1]

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

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

knn\_classifier = KNeighborsClassifier(n\_neighbors=k)

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

new\_patient\_features = []

for i in range(features.shape[1]):

feature\_value = float(input(f"Enter the value for symptom {i + 1}: "))

new\_patient\_features.append(feature\_value)

prediction = knn\_classifier.predict([new\_patient\_features])

if prediction[0] == 0:

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

else:

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

**Output:**

Enter the number of neighbors (k): 3

Enter the value for symptom 1: 1

Enter the value for symptom 2: 1

Enter the value for symptom 3: 1

The new patient is predicted to have the medical condition.

1. **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:**

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

iris = load\_iris()

X = iris.data

y = iris.target

clf = DecisionTreeClassifier(random\_state=42)

clf.fit(X, y)

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): "))

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

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}")

**Output:**

Enter sepal length (cm): 50

Enter sepal width (cm): 110

Enter petal length (cm): 30

Enter petal width (cm): 67

The predicted species for the new flower is: virginica

1. **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

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

from sklearn.linear\_model import LinearRegression

data = np.array([[1200, 2, 250000],

[1500, 3, 320000],

[1000, 2, 200000],

[1800, 4, 400000],

[2000, 3, 380000]])

X = data[:, :-1]

y = data[:, -1]

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)

new\_area = float(input("Enter the area of the new house: "))

new\_bedrooms = int(input("Enter the number of bedrooms in the new house: "))

new\_house\_features = np.array([[new\_area, new\_bedrooms]])

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

print(f"Predicted price for the new house: ${predicted\_price[0]:,.2f}")

**Output:**

Enter the area of the new house: 2000

Enter the number of bedrooms in the new house: 6

Predicted price for the new house: $505,151.52

1. **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

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

from sklearn.linear\_model import LogisticRegression

data = np.array([[300, 12, 1],

[150, 6, 0],

[450, 24, 1],

[200, 18, 0],

[350, 9, 1]])

X = data[:, :-1]

y = data[:, -1]

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)

new\_usage\_minutes = float(input("Enter usage minutes for the new customer: "))

new\_contract\_duration = int(input("Enter contract duration for the new customer: "))

new\_customer\_features = np.array([[new\_usage\_minutes, new\_contract\_duration]])

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

if predicted\_churn[0] == 0:

print("The new customer is predicted to not churn.")

else:

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

**Output:**

Enter usage minutes for the new customer: 40

Enter contract duration for the new customer: 7

The new customer is predicted to not churn.

1. **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

from sklearn.cluster import KMeans

customer\_data = np.array([

[35, 75000, 2, 75, 4.5, 60],

[28, 60000, 0, 120, 4.2, 45],

[45, 85000, 3, 95, 4.8, 90],

[32, 55000, 1, 60, 4.0, 30],

[40, 90000, 2, 80, 4.6, 75]

])

num\_clusters = 3

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

kmeans.fit(customer\_data)

cluster\_labels = kmeans.labels\_

new\_customer\_features = np.array([

float(input("Enter age: ")),

float(input("Enter income: ")),

float(input("Enter number of children: ")),

float(input("Enter average transaction amount: ")),

float(input("Enter product reviews: ")),

float(input("Enter time spent shopping: "))

])

predicted\_cluster = kmeans.predict([new\_customer\_features])

print("Predicted cluster for the new customer:", predicted\_cluster[0])

**Output:**

Enter age: 40

Enter income: 200000

Enter number of children: 3

Enter average transaction amount: 4

Enter product reviews: 2

Enter time spent shopping: 50

Predicted cluster for the new customer: 0

1. **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 numpy as np

import pandas as pd

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

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

from sklearn import datasets

from sklearn.linear\_model import LogisticRegression

data = datasets.load\_iris()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = pd.Series(data.target)

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)

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

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

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

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}")

**Output:**

Enter the names of the features separated by commas: 1,40,50,90

Enter the name of the target variable: 6

Accuracy: 1.00

Precision: 1.00

Recall: 1.00

F1-Score: 1.00

1. **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]])

y = np.array([25000, 30000, 20000])

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()

**Output:**

Enter the mileage: 40

Enter the age: 3

Enter the brand: 234

Enter the engine\_type: 2

The predicted price of the new car is: 30000.00