**COURSE: FUNDAMENTALS OF DATA SCIENCE**

**CODE:DSA0406**

**NAME: JUHAINA AFREEN A**

**REG: 192224158**

**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.**

**CODE:**

import pandas as pd import numpy as np

import scipy.stats as stats data = pd.read\_csv("rare-elements.csv") concentrations = data.iloc[:, 0].dropna().values sample\_size = int(input("Enter the sample size: ")) confidence\_level = float(input("Enter the confidence level (e.g., 0.95 for 95%): ")) precision = float(input("Enter the desired level of precision: "))

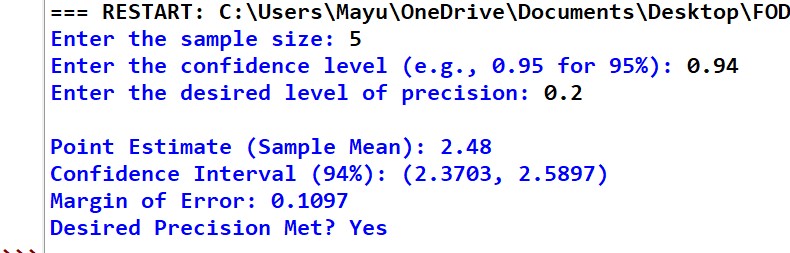
np.random.seed(0) sample = np.random.choice(concentrations, size=sample\_size, replace=False) sample\_mean = np.mean(sample)

std\_error = np.std(sample, ddof=1) / np.sqrt(sample\_size) z\_score = stats.norm.ppf((1 + confidence\_level) / 2) margin\_of\_error = z\_score \* std\_error lower\_bound = sample\_mean - margin\_of\_error upper\_bound = sample\_mean + margin\_of\_error print("\nPoint Estimate (Sample Mean):", round(sample\_mean, 4))

print(f"Confidence Interval ({int(confidence\_level\*100)}%): ({round(lower\_bound, 4)},

{round(upper\_bound, 4)})") print("Margin of Error:", round(margin\_of\_error, 4)) print("Desired Precision Met?" , "Yes" if margin\_of\_error <= precision else "No")

**OUTPUT:**



**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.**

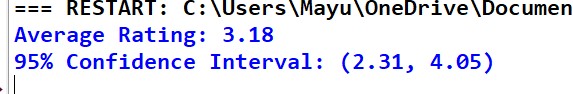
**Question: You will use the pandas library to calculate confidence intervals to estimate the true population mean rating.**

**CODE:**

import pandas as pd import numpy as np from scipy import stats df = pd.read\_csv("customer\_review.csv")

ratings = df['rating'].dropna() mean\_rating = ratings.mean() std\_dev = ratings.std(ddof=1) n = len(ratings) confidence = 0.95 alpha = 1 - confidence z\_score = stats.norm.ppf(1 - alpha / 2) margin\_of\_error = z\_score \* (std\_dev / np.sqrt(n)) ci\_lower = mean\_rating - margin\_of\_error ci\_upper = mean\_rating + margin\_of\_error print(f"Average Rating: {mean\_rating:.2f}") print(f"{int(confidence\*100)}% Confidence Interval: ({ci\_lower:.2f}, {ci\_upper:.2f})")

**OUTPUT:**

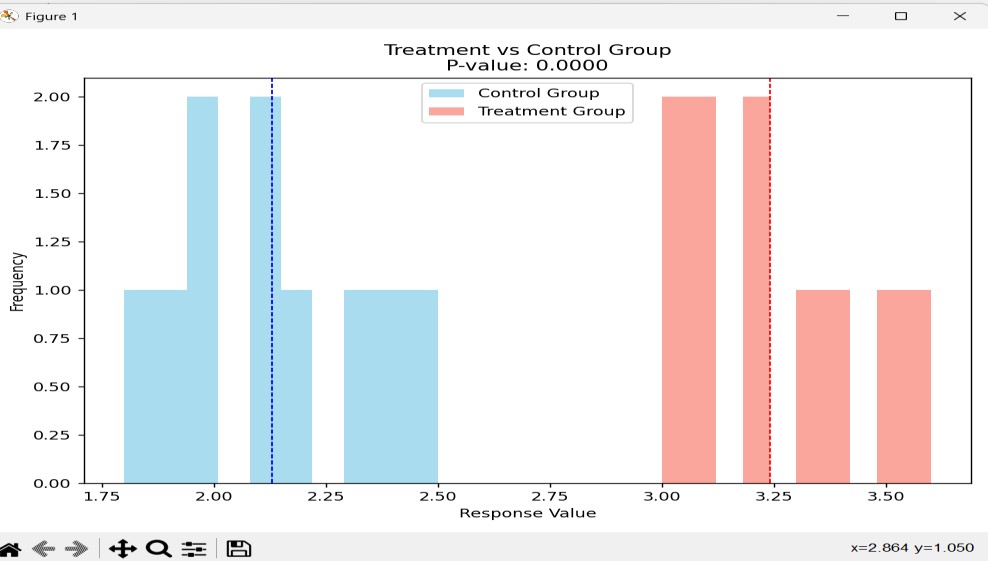


1. **You are a researcher working in a medical lab, investigating the effectiveness of a new treatment for a specific disease. You have collected data from a clinical trial with two groups: a control group receiving a placebo, and a treatment group receiving the new drug.Your goal is to analyze the data 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**.

**CODE:**

import numpy as np import matplotlib.pyplot as plt from scipy import stats control = np.array([2.3, 2.1, 1.8, 2.5, 2.0, 2.2, 1.9, 2.4, 2.1, 2.0]) treatment = np.array([3.1, 3.5, 3.0, 3.4, 3.2, 3.3, 3.1, 3.6, 3.0, 3.2]) t\_stat, p\_value = stats.ttest\_ind(treatment, control) print(f"P-value: {p\_value:.4f}") plt.figure(figsize=(8, 6)) plt.hist(control, alpha=0.7, label='Control Group', color='skyblue', bins=10) plt.hist(treatment, alpha=0.7, label='Treatment Group', color='salmon', bins=10) plt.axvline(np.mean(control), color='blue', linestyle='dashed', linewidth=1) plt.axvline(np.mean(treatment), color='red', linestyle='dashed', linewidth=1) plt.title(f'Treatment vs Control Group\nP-value: {p\_value:.4f}') plt.xlabel('Response Value') plt.ylabel('Frequency') plt.legend() plt.tight\_layout() plt.show() **OUTPUT:**





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.**

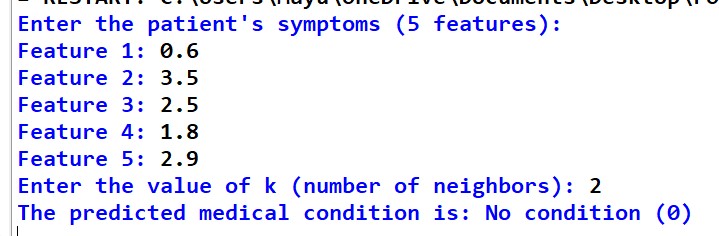
**CODE:**

import numpy as np from sklearn.neighbors import KNeighborsClassifier from sklearn.model\_selection import train\_test\_split from sklearn.datasets import make\_classification

X, y = make\_classification(n\_samples=100, n\_features=5, n\_classes=2, random\_state=42) X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) knn = KNeighborsClassifier() knn.fit(X\_train, y\_train) print("Enter the patient's symptoms (5 features): ") symptoms = [float(input(f"Feature {i+1}: ")) for i in range(5)] k = int(input("Enter the value of k (number of neighbors): ")) knn.n\_neighbors = k prediction = knn.predict([symptoms])

print(f"The predicted medical condition is: {'Condition present (1)' if prediction[0] == 1 else 'No condition (0)'}")

**OUTPUT:**



**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.**

**CODE:**

from sklearn.datasets import load\_iris from sklearn.tree import DecisionTreeClassifier import numpy as np

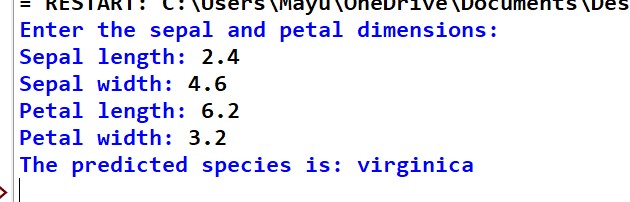
iris = load\_iris()

X, y = iris.data, iris.target clf = DecisionTreeClassifier()

clf.fit(X, y)

print("Enter the sepal and petal dimensions:") sepal\_length = float(input("Sepal length: ")) sepal\_width = float(input("Sepal width: ")) petal\_length = float(input("Petal length: ")) petal\_width = float(input("Petal width: ")) new\_flower = np.array([[sepal\_length, sepal\_width, petal\_length, petal\_width]]) prediction = clf.predict(new\_flower) species = iris.target\_names[prediction[0]] print(f"The predicted species is: {species}")

**OUTPUT:**



**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.**

**CODE:**

import numpy as np import pandas as pd from sklearn.linear\_model import LinearRegression from sklearn.model\_selection import train\_test\_split data = {

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

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

"age": [10, 15, 20, 18, 5, 7, 3, 6, 4, 8],

"price": [245000, 312000, 279000, 308000, 199000, 219000, 405000, 410000, 240000, 300000]

}

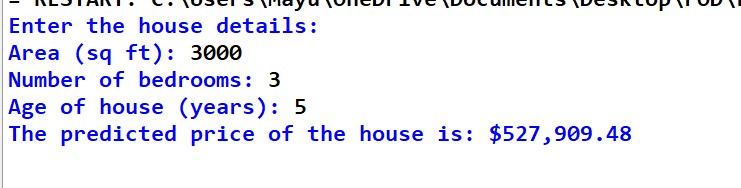
df = pd.DataFrame(data)

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) print("Enter the house details:") area = float(input("Area (sq ft): "))

bedrooms = int(input("Number of bedrooms: ")) age = int(input("Age of house (years): ")) new\_house = pd.DataFrame([[area, bedrooms, age]], columns=["area", "bedrooms", "age"]) predicted\_price = model.predict(new\_house) print(f"The predicted price of the house is: ${predicted\_price[0]:,.2f}") **OUTPUT:**



**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.**

**CODE:**

import numpy as np import pandas as pd from sklearn.linear\_model import LogisticRegression from sklearn.model\_selection import train\_test\_split data = {

"usage\_minutes": [200, 450, 150, 600, 120, 350, 500, 300, 700, 400],

"contract\_duration": [12, 24, 6, 36, 3, 18, 30, 9, 48, 15],

"monthly\_bill": [50, 80, 30, 100, 25, 60, 90, 40, 110, 55],

"customer\_service\_calls": [1, 3, 0, 4, 0, 2, 5, 1, 6, 2],

"churn": [0, 1, 0, 1, 0, 0, 1, 0, 1, 0]

}

df = pd.DataFrame(data)

X = df[["usage\_minutes", "contract\_duration", "monthly\_bill", "customer\_service\_calls"]] y = df["churn"]

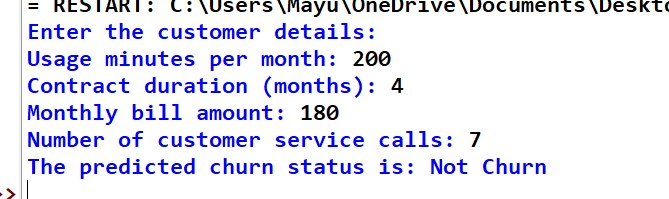
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) model = LogisticRegression() model.fit(X\_train, y\_train) print("Enter the customer details:") usage\_minutes = float(input("Usage minutes per month: ")) contract\_duration = int(input("Contract duration (months): ")) monthly\_bill = float(input("Monthly bill amount: ")) customer\_service\_calls = int(input("Number of customer service calls: "))

new\_customer = pd.DataFrame([[usage\_minutes, contract\_duration, monthly\_bill, customer\_service\_calls]],

columns=["usage\_minutes", "contract\_duration", "monthly\_bill",

"customer\_service\_calls"]) prediction = model.predict(new\_customer) churn\_status = "Churn" if prediction[0] == 1 else "Not Churn"

print(f"The predicted churn status is: {churn\_status}") **OUTPUT:**



**28. Question: K-Means Clustering for Customer Segmentation You are working for an ecommerce 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.**

**CODE:**

import numpy as np import pandas as pd from sklearn.cluster import KMeans from sklearn.preprocessing import StandardScaler

data = {

'Annual Spending': [500, 1500, 2500, 800, 2200, 3000],

'Purchase Frequency': [20, 50, 90, 25, 85, 120],

'Product Diversity': [5, 15, 25, 6, 20, 30]

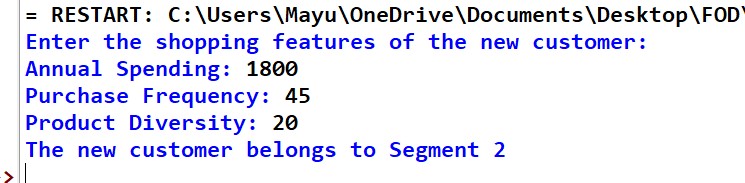
}

df = pd.DataFrame(data) scaler = StandardScaler() scaled\_data = scaler.fit\_transform(df) num\_clusters = 3 kmeans = KMeans(n\_clusters=num\_clusters, random\_state=42) df['Segment'] = kmeans.fit\_predict(scaled\_data) def classify\_new\_customer():

print("Enter the shopping features of the new customer:") annual\_spending = float(input("Annual Spending: ")) purchase\_frequency = float(input("Purchase Frequency: ")) product\_diversity = float(input("Product Diversity: ")) new\_data = pd.DataFrame([[annual\_spending, purchase\_frequency, product\_diversity]],

columns=df.columns[:-1])

new\_data\_scaled = scaler.transform(new\_data) segment = kmeans.predict(new\_data\_scaled)[0] print(f"The new customer belongs to Segment {segment}") classify\_new\_customer() **OUTPUT:**



**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.**

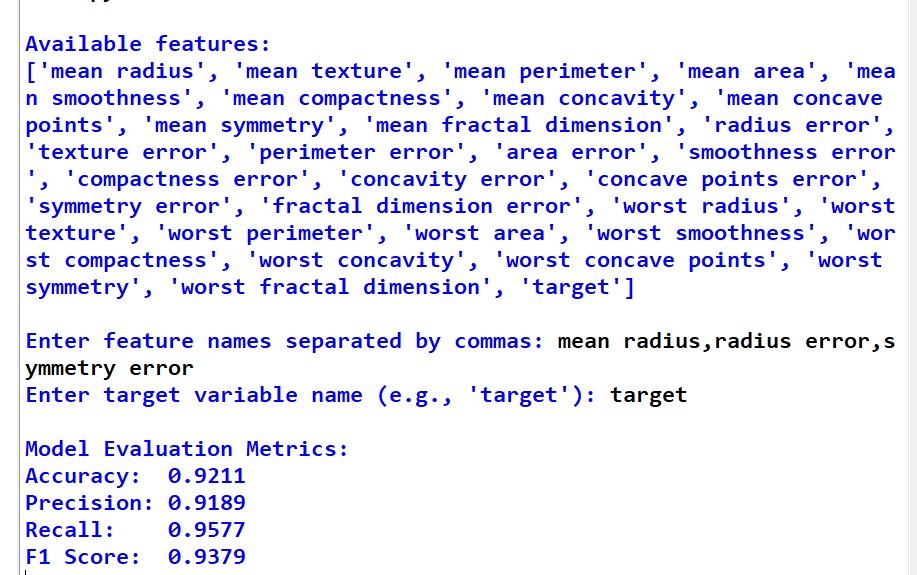
**CODE:**

from sklearn.datasets import load\_breast\_cancer from sklearn.ensemble import RandomForestClassifier from sklearn.model\_selection import train\_test\_split from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score import pandas as pd data = load\_breast\_cancer() df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target print("\nAvailable features:") print(df.columns.tolist()) features\_input = input("\nEnter feature names separated by commas: ") target\_input = input("Enter target variable name (e.g., 'target'): ") features = [f.strip() for f in features\_input.split(",")] target = target\_input.strip() X = df[features] y = df[target]

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

model.fit(X\_train, y\_train) y\_pred = model.predict(X\_test) print("\nModel Evaluation Metrics:") print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}") print(f"Precision: {precision\_score(y\_test, y\_pred):.4f}") print(f"Recall: {recall\_score(y\_test, y\_pred):.4f}") print(f"F1 Score: {f1\_score(y\_test, y\_pred):.4f}") **OUTPUT:**



**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 scikitlearn 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.**

CODE:

import pandas as pd from sklearn.tree import DecisionTreeRegressor, export\_text from sklearn.preprocessing import LabelEncoder from sklearn.model\_selection import train\_test\_split data = {

'mileage': [10000, 50000, 30000, 70000, 25000],

'age': [1, 5, 3, 7, 2],

'brand': ['Toyota', 'BMW', 'Ford', 'BMW', 'Toyota'],

'engine\_type': ['Petrol', 'Diesel', 'Diesel', 'Petrol', 'Electric'],

'price': [20000, 15000, 18000, 12000, 22000]

}

df = pd.DataFrame(data) le\_brand = LabelEncoder() le\_engine = LabelEncoder() df['brand'] = le\_brand.fit\_transform(df['brand']) df['engine\_type'] = le\_engine.fit\_transform(df['engine\_type']) X = df[['mileage', 'age', 'brand', 'engine\_type']] y = df['price']

model = DecisionTreeRegressor()

model.fit(X, y) print("\nEnter the new car details:") mileage = int(input("Mileage (e.g., 30000): ")) age = int(input("Age in years (e.g., 2): ")) print(f"Available brands: {list(le\_brand.classes\_)}") brand\_input = input("Brand: ") brand\_encoded = le\_brand.transform([brand\_input])[0]

print(f"Available engine types: {list(le\_engine.classes\_)}")

engine\_input = input("Engine Type: ") engine\_encoded = le\_engine.transform([engine\_input])[0] new\_car = pd.DataFrame([[mileage, age, brand\_encoded, engine\_encoded]],

columns=['mileage', 'age', 'brand', 'engine\_type'])

predicted\_price = model.predict(new\_car)[0] print(f"\nPredicted Price: ${predicted\_price:.2f}") tree\_rules = export\_text(model, feature\_names=['mileage', 'age', 'brand', 'engine\_type']) print("\nDecision Path:") print(tree\_rules) OUTPUT:

