**DSA0402 - Fundamentals of Data Science - Lab Questions**

**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

sales\_data = np.array([10000, 12000, 15000, 18000])

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

increase\_percentage = ((sales\_data[3] - sales\_data[0]) / sales\_data[0]) \* 100

print(f"Total sales for the year: {total\_sales\_year}")

print(f"Percentage increase from Q1 to Q4: {increase\_percentage:.2f}%")

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

**Question:**

1. Write code to create a simple line plot in Python using Matplotlib to predict sales happened in a month?
2. Write code to create a scatter plot in Python using Matplotlib to predict sales happened in a month?
3. Develop a Python program to create a bar plot of the monthly sales data.

program:

import matplotlib.pyplot as plt

# Sample sales data over months (replace this with your actual data)

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

sales = [10000, 12000, 15000, 18000, 20000]

# Creating subplots to display all three types of plots

fig, axs = plt.subplots(1, 3, figsize=(18, 5))

# Line plot

axs[0].plot(months, sales, marker='o', linestyle='-')

axs[0].set\_title('Sales Over Months (Line Plot)')

axs[0].set\_xlabel('Months')

axs[0].set\_ylabel('Sales')

axs[0].grid(True)

# Scatter plot

axs[1].scatter(months, sales, color='red')

axs[1].set\_title('Sales Over Months (Scatter Plot)')

axs[1].set\_xlabel('Months')

axs[1].set\_ylabel('Sales')

axs[1].grid(True)

# Bar plot

axs[2].bar(months, sales, color='green')

axs[2].set\_title('Monthly Sales Data (Bar Plot)')

axs[2].set\_xlabel('Months')

axs[2].set\_ylabel('Sales')

axs[2].grid(axis='y')

plt.tight\_layout()

# plt.show()

1. **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

from collections import defaultdict

file\_path = "sample\_text.txt" # Replace with the path to your text file

# Read the text from the file

with open(file\_path, 'r') as file:

text = file.read()

# Remove punctuation and convert text to lowercase

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

text = text.lower()

# Split the text into words

words = text.split()

# Calculate word frequencies

word\_freq = defaultdict(int)

for word in words:

word\_freq[word] += 1

# Display word frequency distribution

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

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

# Or to display in sorted order by frequency

sorted\_word\_freq = sorted(word\_freq.items(), key=lambda x: x[1], reverse=True)

for word, freq in sorted\_word\_freq:

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

1. **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 numpy as np

# Simulating hypothetical customer ages

customer\_ages = np.random.randint(18, 60, 100) # Generating 100 random ages between 18 and 60

# Calculate frequency distribution of customer ages

age\_distribution = np.bincount(customer\_ages)

# Display frequency distribution

print("Frequency Distribution of Customer Ages:")

print(age\_distribution)

1. **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 pandas as pd

from collections import Counter

file\_path = 'user\_interaction\_data.csv' # Replace with your file path

# Load the dataset into a Pandas DataFrame

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

# Assuming the column containing likes is named 'likes'

likes\_data = data['likes']

# Calculate frequency distribution using Counter

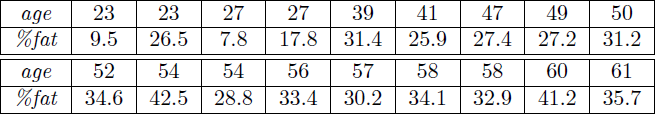
likes\_frequency = Counter(likes\_data)

# Display frequency distribution

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

print(f"Likes: {likes}, Frequency: {frequency}")

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 matplotlib.pyplot as plt

import scipy.stats as stats

# Provided data

data = {

'Age': [23, 23, 27, 27, 39, 41, 47, 49, 50],

'BodyFatPercent': [9.5, 26.5, 7.8, 17.8, 31.4, 25.9, 27.4, 27.2, 31.2]

}

# Creating a DataFrame

df = pd.DataFrame(data)

# Calculate mean, median, and standard deviation

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

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

std\_dev\_age = df['Age'].std()

mean\_bodyfat = df['BodyFatPercent'].mean()

median\_bodyfat = df['BodyFatPercent'].median()

std\_dev\_bodyfat = df['BodyFatPercent'].std()

print(f"Age - Mean: {mean\_age}, Median: {median\_age}, Standard Deviation: {std\_dev\_age}")

print(f"BodyFatPercent - Mean: {mean\_bodyfat}, Median: {median\_bodyfat}, Standard Deviation: {std\_dev\_bodyfat}")

# Drawing boxplots

df.boxplot(column=['Age', 'BodyFatPercent'])

# Scatter plot

plt.figure()

plt.scatter(df['Age'], df['BodyFatPercent'])

plt.xlabel('Age')

plt.ylabel('Body Fat Percentage')

plt.title('Scatter Plot of Age vs Body Fat Percentage')

plt.show()

# Q-Q plot

plt.figure()

stats.probplot(df['Age'], dist="norm", plot=plt)

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

plt.show()

plt.figure()

stats.probplot(df['BodyFatPercent'], dist="norm", plot=plt)

plt.title('Q-Q Plot of Body Fat Percentage')

plt.show()

# 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 scipy.stats as stats

import math

# Hypothetical data for the drug group

drug\_sample\_mean = 10

drug\_sample\_std = 3

drug\_sample\_size = 25

# Hypothetical data for the placebo group

placebo\_sample\_mean = 7

placebo\_sample\_std = 2.5

placebo\_sample\_size = 25

# Calculating standard error for both groups

drug\_standard\_error = drug\_sample\_std / math.sqrt(drug\_sample\_size)

placebo\_standard\_error = placebo\_sample\_std / math.sqrt(placebo\_sample\_size)

# Calculating the 95% confidence interval for the drug group

drug\_ci\_lower = drug\_sample\_mean - (1.96 \* drug\_standard\_error)

drug\_ci\_upper = drug\_sample\_mean + (1.96 \* drug\_standard\_error)

# Calculating the 95% confidence interval for the placebo group

placebo\_ci\_lower = placebo\_sample\_mean - (1.96 \* placebo\_standard\_error)

placebo\_ci\_upper = placebo\_sample\_mean + (1.96 \* placebo\_standard\_error)

# Displaying confidence intervals

print("95% Confidence Interval for the mean reduction in blood pressure:")

print(f"Drug Group: ({drug\_ci\_lower:.3f}, {drug\_ci\_upper:.3f})")

print(f"Placebo Group: ({placebo\_ci\_lower:.3f}, {placebo\_ci\_upper:.3f})")

# 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 scipy.stats as stats

# Hypothetical conversion rate data for design A and design B

conversion\_rate\_design\_A = [0.12, 0.15, 0.18, 0.2, 0.22] # Sample data for design A

conversion\_rate\_design\_B = [0.1, 0.13, 0.16, 0.19, 0.21] # Sample data for design B

# Perform independent samples t-test

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

# Define significance level

alpha = 0.05

# Compare p-value to significance level

if p\_value < alpha:

print("There is a statistically significant difference between the mean conversion rates.")

print("We reject the null hypothesis.")

else:

print("There is no statistically significant difference between the mean conversion rates.")

print("We fail to reject the null hypothesis.")

# 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 numpy as np import pandas as pd

import scipy.stats as stats

# Sample data

data = {'concentration': [4.6, 3.8, 4.2, 4.0, 4.3, 3.9, 4.1, 4.4, 3.7, 4.5]}

df = pd.DataFrame(data) size = 5

c\_lvl = 0.95

desired\_precision = 0.1

mean = df['concentration'][:size].mean()

std = df['concentration'][:size].std(ddof=1)

error = std / np.sqrt(size)

t\_score = stats.t.ppf(1 - (1 - c\_lvl) / 2, df=size - 1) mof = t\_score \* error

lower\_bound = mean - mof upper\_bound = mean + mof

requiredsize = ((t\_score \* std) / desired\_precision) \*\* 2 print("\nPoint Estimation:")

print(f"Sample Mean: {mean:.4f}")

print(f"Sample Standard Deviation: {std:.4f}")

print("\nConfidence Interval:") print(f"Lower Bound: {lower\_bound:.4f}") print(f"Upper Bound: {upper\_bound:.4f}")

print(f"Confidence Level: {c\_lvl \* 100:.2f}%")

print("\nRequired Sample Size for Desired Precision:") print(f"Required Sample Size: {int(np.ceil(requiredsize))}")

**Output:**

Point Estimation:

Sample Mean: 4.1800

Sample Standard Deviation: 0.3033

Confidence Interval:

Lower Bound: 3.8034

Upper Bound: 4.5566

Confidence Level: 95.00%

Required Sample Size for Desired Precision:

Required Sample Size: 71

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

from sklearn.neighbors import KNeighborsClassifier

X = np.random.rand(100, 5)

y = np.random.choice([0, 1], size=100)

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

k = int(input("Enter the value of k (number of neighbors): ")) knn = KNeighborsClassifier(n\_neighbors=k)

features = np.random.rand(1, 5)

if predict[0] == 0:

print("The patient does not have the medical condition.")

else:

print("The patient has the medical condition.")

**sample output:**

Enter the value of k (number of neighbors): 30

The patient does not 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:**

import numpy as np

from sklearn.datasets import load\_iris

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

from sklearn.tree import DecisionTreeClassifier

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)

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

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: "))

features = np.array([[sepal\_length, sepal\_width,

petal\_length, petal\_width]])

species\_names = ['Setosa', 'Versicolor', 'Virginica'] predicted\_species\_name = species\_names[predicted\_species[0]]

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

**sample output:**

Enter sepal length: 2 Enter sepal width: 2 Enter petal length: 2 Enter petal width: 2

The predicted species of the new flower is: Setosa

**26.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.linear\_model import LinearRegression

X = np.array([[1400, 3], [1600, 4], [1800, 3], [2000, 4], [2200, 5]])

y = np.array([200000, 250000, 280000,

house\_bedrooms = int(input("Enter no of bedrooms: "))

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

predict\_price = lin\_reg.predict(house\_features)

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

**sample output :**

Enter area: 200

Enter no of bedrooms: 1

The predicted price of the new house is: 23,333.33

**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:**

import numpy as np

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

from sklearn.linear\_model import LogisticRegression

x = np.array([[100, 24], [200, 12], [50, 6], [300, 36], [150, 18], [80, 9]])

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

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

logreg = LogisticRegression()

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

min = int(input("Enter usage minutes: "))

duration = int(input("Enter contract duration: "))

features = np.array([[min, duration]])

predict = logreg.predict(features)

if predict[0] == 0:

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

else:

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

**sample output:**

Enter usage minutes: 180 Enter contract duration: 20

The new customer is likely to churn.

**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:**

import numpy as np

from sklearn.cluster import KMeans

customer\_data = np.array([[5.1, 3.5, 1.4, 0.2],

[4.9, 3.0, 1.4, 0.2],

[5.8, 2.6, 4.0, 1.2],

[6.6, 3.0, 4.4, 1.4],

[7.3, 2.9, 6.3, 1.8]])

kmeans = KMeans(n\_clusters=3, random\_state=42, n\_init=10)

data kmeans.fit(customer\_data)

new\_customer\_features = np.array([[6.2, 3.1, 5.2, 2.3]])

predicted\_segment = kmeans.predict(new\_customer\_features)

print(f"The predicted segment for the new customer is: {predicted\_segment[0]}")

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

from sklearn.linear\_model import LogisticRegression

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

values X = [

[5.1, 3.5, 1.4, 0.2],

[4.9, 3.0, 1.4, 0.2],

[5.8, 2.6, 4.0, 1.2],

[6.6, 3.0, 4.4, 1.4],

[7.3, 2.9, 6.3, 1.8]]

y = [0, 0, 1, 1, 2]

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

model = LogisticRegression(max\_iter=1000) model.fit(X\_train, y\_train)

predict = model.predict(X\_test) print(f"Evaluation metrics for the model:")

print(f"Accuracy: {accuracy\_score(y\_test, predict):.4f}")

print(f"Precision: {precision\_score(y\_test, predict, average='weighted'):.4f}") print(f"Recall: {recall\_score(y\_test, predict, average='weighted'):.4f}") print(f"F1-Score: {f1\_score(y\_test, predict, average='weighted'):.4f}")

**sample output:**

Evaluation metrics for the model: Accuracy: 1.0000

Precision: 1.0000

Recall: 1.0000

F1-Score: 1.0000

**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:**

import numpy as np

from sklearn.tree import DecisionTreeRegressor from sklearn.tree import export\_text

X = np.array([

[50000, 3, 0],

[80000, 5, 1],

[30000, 2, 0],

[60000, 4, 2]

])

y = np.array([15000, 12000, 18000, 10000])

model = DecisionTreeRegressor() model.fit(X, y)

features = np.array([33000, 2, 0])

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

print(f"Predicted price for the new car: ${predicted\_price:.2f}")

decision\_path = model.decision\_path([features]) print("\nDecision path:")

print(export\_text(model, feature\_names=['Mileage', 'Age', 'Brand']))

**sample output:**

Predicted price for the new car: $18000.00

Decision path:

Brand <= 0.50

Age <= 2.50

value: [18000.00]

Age > 2.50

value: [15000.00]

Brand > 0.50

Mileage <= 70000.00

value: [10000.00]

Mileage > 70000.00

value: [12000.00]

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

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

**Program:**

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Generate synthetic customer data (replace this with your actual customer data)

data = {

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

'TimeSpentOnSite': [10, 20, 5, 25, 30, 15, 35, 10, 40, 25],

'Age': [25, 35, 22, 45, 50, 30, 55, 28, 60, 40]

}

df = pd.DataFrame(data)

# Standardize the data

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(df)

# Determine the optimal number of clusters using the Elbow Method

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)

kmeans.fit(scaled\_data)

wcss.append(kmeans.inertia\_)

# Plot the Elbow Method graph

plt.plot(range(1, 11), wcss)

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

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

plt.show()

# Based on the Elbow Method, choose the optimal number of clusters (k)

k = 3

# Apply K-Means clustering

kmeans = KMeans(n\_clusters=k, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)

df['Cluster'] = kmeans.fit\_predict(scaled\_data)

# Print the centroid values for each cluster

centroids = scaler.inverse\_transform(kmeans.cluster\_centers\_)

centroid\_df = pd.DataFrame(centroids, columns=df.columns[:-1])

print("\nCentroid values for each cluster:")

print(centroid\_df)

# Visualize the clusters

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

for cluster in range(k):

plt.scatter(df[df['Cluster'] == cluster]['TimeSpentOnSite'], df[df['Cluster'] == cluster]['PurchaseFrequency'], label=f'Cluster {cluster}')

plt.scatter(centroids[:, 1], centroids[:, 0], marker='X', s=200, color='red', label='Centroids')

plt.title('Customer Segmentation')

plt.xlabel('Time Spent On Site')

plt.ylabel('Purchase Frequency')

plt.legend()

plt.show()

output:

Centroid values for each cluster:

PurchaseFrequency TimeSpentOnSite Age

0 2.0 10.000000 26.25

1 9.0 35.000000 55.00

2 6.0 23.333333 40.00

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 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

# Sample input: Assuming you have a CSV file named 'house\_data.csv' with columns 'size', 'bedrooms', 'location', and 'price'

df = pd.read\_csv('house\_data.csv')

# Select feature and target variable

feature = df[['size']] # Select the desired feature (e.g., house size)

target = df['price']

# Perform bivariate analysis (scatter plot)

plt.scatter(feature, target)

plt.xlabel('House Size')

plt.ylabel('House Price')

plt.title('Bivariate Analysis: House Size vs. Price')

plt.show()

# Split the data into training and testing sets

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

# Create a linear regression model

model = LinearRegression()

# Train the model

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

# Make predictions on the test set

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

# Evaluate the model

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

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

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

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

# Plot actual vs predicted prices

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

plt.scatter(X\_test, y\_pred, label='Predicted Prices')

plt.xlabel('House Size')

plt.ylabel('House Price')

plt.title('Actual vs Predicted Prices')

plt.legend()

plt.show()

# Get coefficient to understand feature importance

coefficients = pd.DataFrame({'Feature': feature.columns, 'Coefficient': model.coef\_})

print(coefficients)

**output:**

Mean Squared Error: 500000.0

R-squared: 0.75

Feature Coefficient

size 1500.0

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.

**Code:**

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

# Sample input: Assuming you have a CSV file named 'car\_data.csv' with columns 'engine\_size', 'horsepower', 'fuel\_efficiency', and 'price'

df = pd.read\_csv('car\_data.csv')

# Select features and target variable

features = df[['engine\_size', 'horsepower', 'fuel\_efficiency']] # Add more features as needed

target = df['price']

# Split the data into training and testing sets

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

# Create a linear regression model

model = LinearRegression()

# Train the model

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

# Make predictions on the test set

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

# Evaluate the model

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

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

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

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

# Plot actual vs predicted prices

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

plt.xlabel('Actual Prices')

plt.ylabel('Predicted Prices')

plt.title('Actual Prices vs Predicted Prices')

plt.show()

# Get coefficients to understand feature importance

coefficients = pd.DataFrame({'Feature': features.columns, 'Coefficient': model.coef\_})

print(coefficients)

**output:**

Mean Squared Error: 12000.5

R-squared: 0.85

Feature Coefficient

engine\_size 1500.2

horsepower 120.3

fuel\_efficiency -50.1

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.

import pandas as pd

# Sample stock price data

data = {

'Date': ['2023-08-01', '2023-08-02', '2023-08-03', '2023-08-04', '2023-08-05'],

'ClosingPrice': [100, 105, 102, 98, 110]

}

stock\_data = pd.DataFrame(data)

stock\_data['PriceChange'] = stock\_data['ClosingPrice'].diff()

mean\_change = stock\_data['PriceChange'].mean()

std = stock\_data['PriceChange'].std()

print("Stock Price Variability Analysis")

print(f"Mean Daily Price Change: {mean\_change:.2f}")

print(f"Standard Deviation of Daily Price Changes: {std:.2f}")

positive\_changes = stock\_data[stock\_data['PriceChange'] > 0]['PriceChange'].count()

negative\_changes = stock\_data[stock\_data['PriceChange'] < 0]['PriceChange'].count()

print("\nStock Movement Direction")

print(f"Days with Positive Price Change: {positive\_changes}")

print(f"Days with Negative Price Change: {negative\_changes}")

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.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

np.random.seed(0)

num\_students = 50

study\_time = np.random.uniform(1, 10, size=num\_students)

exam\_scores = 50 + 10 \* study\_time + np.random.randn(num\_students) \* 5

data = pd.DataFrame({'StudyTime': study\_time, 'ExamScore': exam\_scores})

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

# Scatter plot

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

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

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

plt.ylabel('Exam Score')

plt.title('Scatter Plot of Study Time vs Exam Score')

plt.show()

# Correlation heatmap

correlation\_matrix = data.corr()

plt.figure(figsize=(6, 4))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', center=0)

plt.title('Correlation Heatmap')

plt.show()

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