**Slip 1**

**Q1.Write a R program to add, multiply and divide two vectors of integertype. (Vector length should be minimum 4) .**

**Ans**

**> v1 <- c(2, 4, 6, 8)**

**> v2 <- c(1, 2, 3, 4)**

**>**

**> add\_result <- v1 + v2**

**> multiply\_result <- v1 \* v2**

**> divide\_result <- v1 / v2**

**>**

**> add\_result**

**[1] 3 6 9 12**

**> multiply\_result**

**[1] 2 8 18 32**

**> divide\_result**

**[1] 2 2 2 2**

**Q2.Consider the student data set. It can be downloaded from: https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaqO 5\_6dIOw . Write a programme in python to apply simple linear regression and find out mean absolute error, mean squared error and root mean squared erro**r.

**Ans**

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import numpy as np

data = pd.read\_csv('C:/Users/HP/Downloads/student\_scores.csv')

print(data.columns)

X = data[['Hours']]

y = data['Scores']

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

model = LinearRegression()

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

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

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

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

rmse = np.sqrt(mse)

mae, mse, rmse

**Output:**

**Index(['Hours', 'Scores'], dtype='object')**

**(3.9207511902099244, 18.943211722315272, 4.352380006653288)**

**Slip 2**

Q**1. Write an R program to calculate the multiplication table using afunction.**

**Ans**

**> multiplication\_table <- function(n) {**

**+ for (i in 1:10) {**

**+ print(paste(n, "x", i, "=", n \* i))**

**+ }**

**+ }**

**>**

**> multiplication\_table(5)**

**[1] "5 x 1 = 5"**

**[1] "5 x 2 = 10"**

**[1] "5 x 3 = 15"**

**[1] "5 x 4 = 20"**

**[1] "5 x 5 = 25"**

**[1] "5 x 6 = 30"**

**[1] "5 x 7 = 35"**

**[1] "5 x 8 = 40"**

**[1] "5 x 9 = 45"**

**[1] "5 x 10 = 50"**

**Q2. Write a python program to implement k-means algorithms on asynthetic dataset..**

**Ans**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import make\_blobs

X, \_ = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60, random\_state=0)

kmeans = KMeans(n\_clusters=4)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

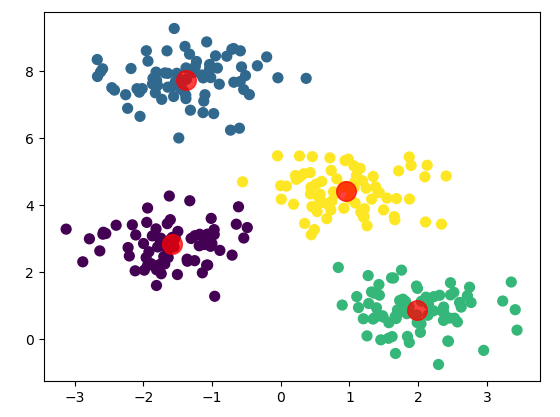
plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75)

plt.show()

**Output:**

****

**Slip 3**

**Q1. Write a R program to reverse a number and also calculate the sum ofdigits of that number. [10 Marks]**

**Ans**

**> reverse\_and\_sum <- function(num) {**

**+ reverse\_num <- as.numeric(paste(rev(strsplit(as.character(num), NULL)[[1]]), collapse = ""))**

**+ sum\_digits <- sum(as.numeric(unlist(strsplit(as.character(num), NULL))))**

**+ list(reversed = reverse\_num, sum\_of\_digits = sum\_digits)**

**+ }**

**>**

**> reverse\_and\_sum(12345)**

**$reversed**

**[1] 54321**

**$sum\_of\_digits**

**[1] 15**

**Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b0 and b1.( use numpypackage) x=[0,1,2,3,4,5,6,7,8,9,11,13] y = ([1, 3, 2, 5, 7, 8, 8, 9, 10, 12,16, 18]**

**Ans**

import numpy as np

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12, 16, 18])

n = len(x)

mean\_x = np.mean(x)

mean\_y = np.mean(y)

b1 = np.sum((x - mean\_x) \* (y - mean\_y)) / np.sum((x - mean\_x)\*\*2)

b0 = mean\_y - b1 \* mean\_x

b0, b1

**Output:**

**(0.838709677419355, 1.2889200561009817)**

**Slip 4**

**Q1. Write a R program to calculate the sum of two matrices of given size. [10 Marks]**

**Ans**

**> matrix1 <- matrix(1:6, nrow = 2)**

**> matrix2 <- matrix(7:12, nrow = 2)**

**>**

**> sum\_result <- matrix1 + matrix2**

**> sum\_result**

**[,1] [,2] [,3]**

**[1,] 8 12 16**

**[2,] 10 14 18**

**Q2. Consider following dataset weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy','Sunn y','Overcast','Overcast','Rainy'] temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild'] play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']. Use Naïve Bayes algorithm to predict [0: Overcast, 2: Mild]tuple belongs to which class whether to play the sports or not.**

**Ans**

import numpy as np

import pandas as pd

from sklearn.naive\_bayes import GaussianNB

weather = ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny',

'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy']

temp = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool',

'Mild', 'Mild', 'Mild', 'Hot', 'Mild']

play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes',

'Yes', 'Yes', 'Yes', 'Yes', 'No']

data = pd.DataFrame({'Weather': weather, 'Temperature': temp, 'Play': play})

X = pd.get\_dummies(data[['Weather', 'Temperature']])

y = data['Play'].map({'No': 0, 'Yes': 1})

model = GaussianNB()

model.fit(X, y)

new\_data = pd.DataFrame({'Weather\_Overcast': [1], 'Weather\_Rainy': [0], 'Weather\_Sunny': [0],

'Temperature\_Cool': [0], 'Temperature\_Hot': [0], 'Temperature\_Mild': [1]})

prediction = model.predict(new\_data)

result = 'Yes' if prediction[0] == 1 else 'No'

result

**Output:**

**“Yes”**

**Slip 5**

**Q1. Write a R program to concatenate two given factors. [10 Marks]**

**Ans**

> factor1 <- factor(c("A", "B", "C"))

> factor2 <- factor(c("D", "E", "F"))

>

> concatenated\_factor <- factor(c(levels(factor1), levels(factor2)))

> concatenated\_factor

[1] A B C D E F

Levels: A B C D E F

**Q2. Write a Python program build Decision Tree Classifier using Scikit- learn package for diabetes data set (download database from** [**https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset?resource=download**](https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset?resource=download) **)**

**Ans**

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

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

data = pd.read\_csv('C:/Users/HP/Downloads/diabetes.csv')

X = data.drop('Outcome', axis=1)

y = data['Outcome']

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

model = DecisionTreeClassifier()

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

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

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

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

accuracy, report

**Output:**

**(0.7532467532467533,**

**' precision recall f1-score support\n\n 0 0.84 0.77 0.80 99\n 1 0.63 0.73 0.68 55\n\n accuracy 0.75 154\n macro avg 0.74 0.75 0.74 154\nweighted avg 0.76 0.75 0.76 154\n')**

**Slip 6**

**Q1. Write a R programto create a data frame using two given vectors and displaythe duplicate elements.**

**Ans**

> vector1 <- c(1, 2, 3, 4, 5, 2, 6)

> vector2 <- c("A", "B", "C", "D", "B", "E", "F")

>

> data\_frame <- data.frame(Vector1 = vector1, Vector2 = vector2)

>

> duplicates <- data\_frame[duplicated(data\_frame) | duplicated(data\_frame, fromLast = TRUE), ]

> duplicates

[1] Vector1 Vector2

<0 rows> (or 0-length row.names)

>

**Q2. Write a python program to implement hierarchical Agglomerative clusteringalgorithm. (Download Customer.csv dataset from github.com** [**https://www.kaggle.com/datasets/mithunkuamr/customer-csv**](https://www.kaggle.com/datasets/mithunkuamr/customer-csv) **).**

**Ans**

pip install pandas numpy matplotlib seaborn scikit-learn scipy

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

data = pd.read\_csv("C:/Users/HP/Downloads/customers.csv")

data = data.select\_dtypes(include=[np.number]) # Select only numerical columns

data\_scaled = StandardScaler().fit\_transform(data)

linked = linkage(data\_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Sample Index')

plt.ylabel('Distance')

plt.show()

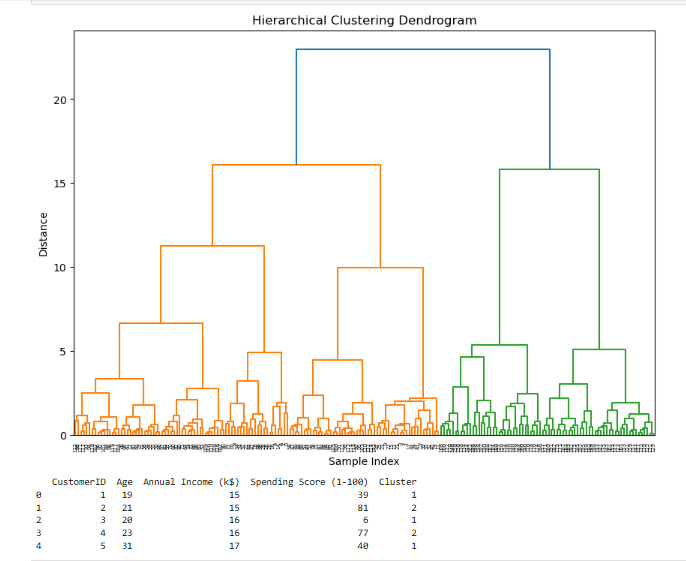
model = AgglomerativeClustering(n\_clusters=3) # Set the number of clusters

clusters = model.fit\_predict(data\_scaled)

data['Cluster'] = clusters

print(data.head())

**Output:**

****

**Slip 7**

**Q1. Write a R program to create a sequence of numbers from 20 to 50 and findthe mean of numbers from 20 to 60 and sum of numbers from 51 to 91.**

**Ans**

**> sequence <- 20:50**

**> mean\_20\_to\_60 <- mean(20:60)**

**> sum\_51\_to\_91 <- sum(51:91)**

**>**

**> mean\_20\_to\_60**

**[1] 40**

**> sum\_51\_to\_91**

**[1] 2911**

**Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse theperformance of the model (Use sklearn package) x = np.array([1,2,3,4,5,6,7,8]) y = np.array([7,14,15,18,19,21,26,23])**

**Ans**

**pip install numpy scikit-learn**

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)

y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

model = LinearRegression()

model.fit(x, y)

b1 = model.coef\_[0]

b0 = model.intercept\_

y\_pred = model.predict(x)

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

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

b0, b1, mse, r2

**Output:**

**(7.642857142857142, 2.2738095238095237, 3.4657738095238084, 0.8867741072947811)**

**Slip 8**

**Q1. Write a R program to get the first 10 Fibonacci numbers. [10 Marks]**

**Ans**

> fibonacci <- numeric(10)

> fibonacci[1] <- 0

> fibonacci[2] <- 1

>

> for (i in 3:10) {

+ fibonacci[i] <- fibonacci[i - 1] + fibonacci[i - 2]

+ }

>

> fibonacci

[1] 0 1 1 2 3 5 8 13 21 34

**Q2. Write a python program to implement k-means algorithm to build prediction model (Use Credit Card Dataset CC GENERAL.csv Download from kaggle.com**

[**https://www.kaggle.com/datasets/arjunbhasin2013/ccdata**](https://www.kaggle.com/datasets/arjunbhasin2013/ccdata) **)**

**Ans**

**pip install pandas numpy matplotlib scikit-learn**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

data = pd.read\_csv('C:/Users/HP/Downloads/CC GENERAL.csv')

data\_cleaned = data.drop(['CUST\_ID'], axis=1).dropna()

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data\_cleaned)

inertia = []

silhouette\_scores = []

k\_values = range(2, 11)

for k in k\_values:

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

kmeans.fit(data\_scaled)

inertia.append(kmeans.inertia\_)

silhouette\_scores.append(silhouette\_score(data\_scaled, kmeans.labels\_))

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.plot(k\_values, inertia, marker='o')

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

plt.ylabel('Inertia')

plt.subplot(1, 2, 2)

plt.plot(k\_values, silhouette\_scores, marker='o')

plt.title('Silhouette Scores')

plt.xlabel('Number of Clusters')

plt.ylabel('Silhouette Score')

plt.show()

optimal\_k = 5

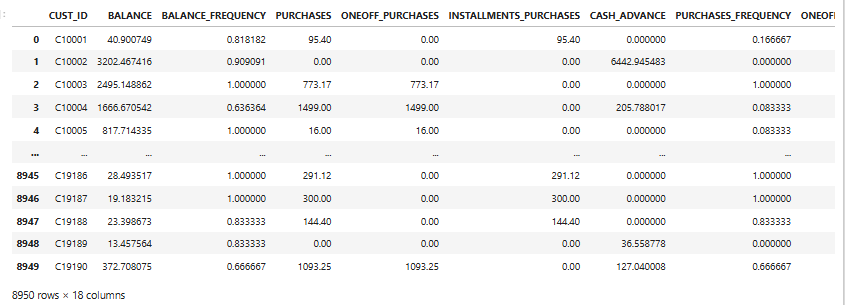
kmeans\_final = KMeans(n\_clusters=optimal\_k, random\_state=42)

kmeans\_final.fit(data\_scaled)

data['Cluster'] = kmeans\_final.labels\_

print(data.head())

**Output:**

****

**Slip 9**

**Q1. Write an R program to create a Data frames which contain details of 5 employees and display summary of the data. [10 Marks]**

**Ans**

**> employees <- data.frame(**

**+ EmployeeID = 1:5,**

**+ Name = c("Alice", "Bob", "Charlie", "David", "Eva"),**

**+ Age = c(25, 30, 35, 40, 28),**

**+ Department = c("HR", "Finance", "IT", "Marketing", "Sales"),**

**+ Salary = c(50000, 60000, 55000, 65000, 48000)**

**+ )**

**>**

**> summary(employees)**

**EmployeeID Name Age Department Salary**

**Min. :1 Length:5 Min. :25.0 Length:5 Min. :48000**

**1st Qu.:2 Class :character 1st Qu.:28.0 Class :character 1st Qu.:50000**

**Median :3 Mode :character Median :30.0 Mode :character Median :55000**

**Mean :3 Mean :31.6 Mean :55600**

**3rd Qu.:4 3rd Qu.:35.0 3rd Qu.:60000**

**Max. :5 Max. :40.0 Max. :65000**

**>**

**Q2. Write a Python program to build an SVM model to Cancer dataset. The dataset is available in the scikit-learn library. Check the accuracyof model with precision and recall.**

**Ans**

from sklearn import datasets

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

from sklearn.svm import SVC

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

cancer = datasets.load\_breast\_cancer()

X = cancer.data

y = cancer.target

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

model = SVC()

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

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

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

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

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

accuracy, precision, recall

**Output:**

**(0.9473684210526315, 0.922077922077922, 1.0)**

**Slip 10**

**Q1. Write a R program to find the maximum and the minimum value of a givenvector**

**Ans**

**> vector <- c(12, 5, 23, 8, 19)**

**>**

**> max\_value <- max(vector)**

**> min\_value <- min(vector)**

**>**

**> max\_value**

**[1] 23**

**> min\_value**

**[1] 5**

**>**

**Q2. Write a Python Programme to read the dataset (“Iris.csv”). dataset download from (**[**https://www.kaggle.com/datasets/saurabh00007/iriscsv**](https://www.kaggle.com/datasets/saurabh00007/iriscsv) **) and apply Apriori algorithm.**

**Ans**

**pip install mlxtend**

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

data = pd.read\_csv('C:/Users/HP/Downloads/Iris.csv')

data.drop(['Id', 'Species'], axis=1, inplace=True)

# Convert the data into a format suitable for the Apriori algorithm

data\_binarized = data.apply(lambda x: pd.Series(1, index=x[x > 0].index), axis=1).fillna(0)

frequent\_itemsets = apriori(data\_binarized, min\_support=0.1, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

print(frequent\_itemsets)

print(rules)

**Output:**

**support itemsets**

**0 1.0 (SepalLengthCm)**

**1 1.0 (SepalWidthCm)**

**2 1.0 (PetalLengthCm)**

**3 1.0 (PetalWidthCm)**

**4 1.0 (SepalLengthCm, SepalWidthCm)**

**5 1.0 (SepalLengthCm, PetalLengthCm)**

**6 1.0 (SepalLengthCm, PetalWidthCm)**

**7 1.0 (PetalLengthCm, SepalWidthCm)**

**8 1.0 (SepalWidthCm, PetalWidthCm)**

**9 1.0 (PetalLengthCm, PetalWidthCm)**

**.**

**..**

**.**

**.**

**Slip 11**

**Q1. Write a R program to find all elements of a given list that are not inanother given list. = list("x", "y", "z") = list("X", "Y", "Z", "x", "y", "z")**

**Ans**

**> list1 <- list("x", "y", "z")**

**> list2 <- list("X", "Y", "Z", "x", "y", "z")**

**>**

**> result <- setdiff(unlist(list1), unlist(list2))**

**> result**

**character(0)**

**>**

**Q2. Write a python program to implement hierarchical clustering algorithm.(Download Wholesale customers data dataset from github.com).**

**Ans**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

data = pd.read\_csv("C:/Users/HP/Downloads/customers.csv")

data = data.select\_dtypes(include=[np.number]) # Select only numerical columns

data\_scaled = StandardScaler().fit\_transform(data)

linked = linkage(data\_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Sample Index')

plt.ylabel('Distance')

plt.show()

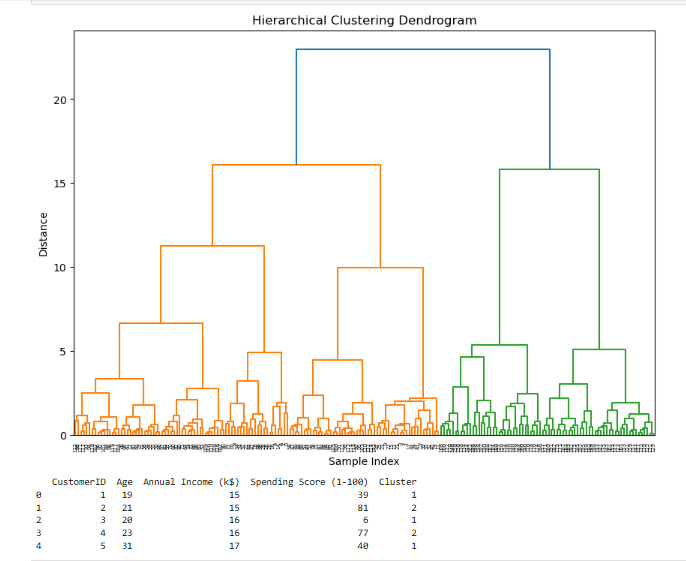
model = AgglomerativeClustering(n\_clusters=3) # Set the number of clusters

clusters = model.fit\_predict(data\_scaled)

data['Cluster'] = clusters

print(data.head())

**Output:**

****

**Slip 12**

**Q1. Write a R program to create a Dataframes which contain details of 5employees and display the details. Employee contain (empno,empname,gender,age,designation) [10 Marks]**

**Ans**

**> employees <- data.frame(**

**+ empno = 1:5,**

**+ empname = c("Alice", "Bob", "Charlie", "David", "Eva"),**

**+ gender = c("F", "M", "M", "M", "F"),**

**+ age = c(25, 30, 35, 40, 28),**

**+ designation = c("HR", "Finance", "IT", "Marketing", "Sales")**

**+ )**

**>**

**> print(employees)**

**empno empname gender age designation**

**1 1 Alice F 25 HR**

**2 2 Bob M 30 Finance**

**3 3 Charlie M 35 IT**

**4 4 David M 40 Marketing**

**5 5 Eva F 28 Sales**

**>**

**Q2. Write a python program to implement multiple Linear Regression modelfor a car dataset. Dataset can be downloaded from:** [**https://github.com/krishnaik06/Multiple-Linear-Regression/blob/master/50\_Startups.csv**](https://github.com/krishnaik06/Multiple-Linear-Regression/blob/master/50_Startups.csv)

**Ans**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import LabelEncoder

url = 'https://raw.githubusercontent.com/krishnaik06/Multiple-Linear-Regression/master/50\_Startups.csv'

data = pd.read\_csv(url)

le = LabelEncoder()

data['State'] = le.fit\_transform(data['State'])

X = data.iloc[:, :-1].values

y = data.iloc[:, -1].values

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

regressor = LinearRegression()

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

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

print(np.column\_stack((y\_pred, y\_test)))

**Output:**

**[[103959.40508021 103282.38 ]**

**[132398.73236694 144259.4 ]**

**[133529.03790148 146121.95 ]**

**[ 72958.28368314 77798.83 ]**

**[179534.78737166 191050.39 ]**

**[115533.62584099 105008.31 ]**

**[ 67476.95847883 81229.06 ]**

**[ 98504.36199396 97483.56 ]**

**[114789.16083781 110352.25 ]**

**[168972.21909945 166187.94 ]]**

**Slip 13**

**Q1. Draw a pie chart using R programming for the following datadistribution:**

**Digits on Dice 1 2 3 4 5 6**

**Frequency of getting each number 7 2 6 3 4 8**

**Ans**

**>**

**> frequencies <- c(7, 2, 6, 3, 4, 8)**

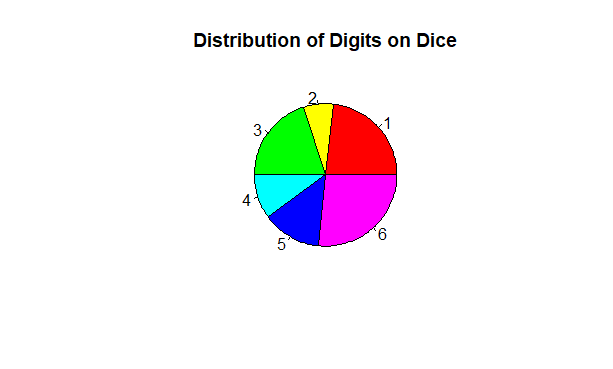
**> labels <- c("1", "2", "3", "4", "5", "6")**

**>**

**> pie(frequencies, labels = labels, main = "Distribution of Digits on Dice", col = rainbow(length(frequencies)))**

**>**

**Output:**

****

**Q2. Write a Python program to read “StudentsPerformance.csv” file. Solvefollowing: - To display the shape of dataset. - To display the top rows of the dataset with their columns.Note: Download dataset from following link : (**[**https://github.com/rashida048/Datasets/blob/master/StudentsPerformance.csv**](https://github.com/rashida048/Datasets/blob/master/StudentsPerformance.csv) **)**

**Ans**

import pandas as pd

url = 'https://raw.githubusercontent.com/rashida048/Datasets/master/StudentsPerformance.csv'

data = pd.read\_csv(url)

print(data.shape)

print(data.head())

**Output:**

**(1000, 8)**

**gender race/ethnicity parental level of education lunch \**

**0 female group B bachelor's degree standard**

**1 female group C some college standard**

**2 female group B master's degree standard**

**3 male group A associate's degree free/reduced**

**4 male group C some college standard**

**test preparation course math score reading score writing score**

**0 none 72 72 74**

**1 completed 69 90 88**

**2 none 90 95 93**

**3 none 47 57 44**

**4 none 76 78 75**

**Slip 14**

**Q1. Write a script in R to create a list of employees (name) and perform thefollowing: a. Display names of employees in the list. b. Add an employee at the end of the list c. Remove the third element of the list. [10 Marks]**

**Ans**

> employees <- list("Alice", "Bob", "Charlie", "David", "Eva")

>

> print(employees)

[[1]]

[1] "Alice"

[[2]]

[1] "Bob"

[[3]]

[1] "Charlie"

[[4]]

[1] "David"

[[5]]

[1] "Eva"

>

> employees <- c(employees, "Frank")

>

> employees <- employees[-3]

>

> print(employees)

[[1]]

[1] "Alice"

[[2]]

[1] "Bob"

[[3]]

[1] "David"

[[4]]

[1] "Eva"

[[5]]

[1] "Frank"

>

**Q2. Write a Python Programme to apply Apriori algorithm on Groceries dataset. Dataset can be downloaded from (**[**https://github.com/amankharwal/Website-data/blob/master/Groceries\_dataset.csv**](https://github.com/amankharwal/Website-data/blob/master/Groceries_dataset.csv) **). Also display support and confidence for each rule.**

**Ans**

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

url = 'https://raw.githubusercontent.com/amankharwal/Website-data/master/Groceries\_dataset.csv'

data = pd.read\_csv(url)

basket = data.groupby(['Member\_number', 'itemDescription'])['itemDescription'].count().unstack().reset\_index().fillna(0).set\_index('Member\_number')

basket = basket.applymap(lambda x: 1 if x > 0 else 0)

frequent\_itemsets = apriori(basket, min\_support=0.01, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.2)

print(rules[['antecedents', 'consequents', 'support', 'confidence']])

**Output:**

**antecedents consequents \**

**0 (UHT-milk) (bottled water)**

**1 (UHT-milk) (other vegetables)**

**2 (UHT-milk) (pip fruit)**

**3 (UHT-milk) (rolls/buns)**

**4 (UHT-milk) (root vegetables)**

**... ... ...**

**5839 (rolls/buns, yogurt, other vegetables) (soda, whole milk)**

**5840 (soda, whole milk, yogurt) (rolls/buns, other vegetables)**

**5841 (soda, rolls/buns, whole milk) (yogurt, other vegetables)**

**5842 (soda, rolls/buns, yogurt) (whole milk, other vegetables)**

**5843 (rolls/buns, whole milk, yogurt) (soda, other vegetables)**

**support confidence**

**0 0.021293 0.271242**

**1 0.038994 0.496732**

**2 0.017188 0.218954**

**3 0.031042 0.395425**

**4 0.021036 0.267974**

**... ... ...**

**5839 0.013597 0.259804**

**5840 0.013597 0.250000**

**5841 0.013597 0.208661**

**5842 0.013597 0.321212**

**5843 0.013597 0.206226**

**[5844 rows x 4 columns]**

**Slip 15**

**Q1.Write a R program to add, multiply and divide two vectors of integer type.(vector length should be minimum 4) [10 Marks]**

**Ans**

**> vector1 <- c(4, 8, 15, 16)**

**> vector2 <- c(23, 42, 8, 4)**

**>**

**> addition <- vector1 + vector2**

**> multiplication <- vector1 \* vector2**

**> division <- vector1 / vector2**

**>**

**> addition**

**[1] 27 50 23 20**

**> multiplication**

**[1] 92 336 120 64**

**> division**

**[1] 0.1739130 0.1904762 1.8750000 4.0000000**

**Q2. Write a Python program build Decision Tree Classifier forshows.csvfrom pandas and predict class label for show starring a 40 years old American comedian, with 10 years of experience, and a comedy ranking of 7? Create a csv file as shown in** [**https://github.com/mahesh147/Decision-Tree-Classifier/blob/master/Social\_Network\_Ads.csv**](https://github.com/mahesh147/Decision-Tree-Classifier/blob/master/Social_Network_Ads.csv)

**Ans**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

data = pd.read\_csv("C:/Users/HP/Downloads/Social\_Network\_Ads.csv")

X = data[['Age', 'EstimatedSalary', 'Gender']]

y = data['Purchased']

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

clf = DecisionTreeClassifier()

clf.fit(X, y)

prediction = clf.predict([[40, 0, 1]]) # 40 years, EstimatedSalary set to 0, Male

print(f"Prediction: {prediction[0]}")

**Output:**

**Prediction: 0**

**Slip 16**

**Q1. Write a R program to create a simple bar plot of given data**

**Year Export Import**

**2001 26 35**

**2002 32 40**

**2003 35 50**

**Ans**

**> years <- c(2001, 2002, 2003)**

**> export <- c(26, 32, 35)**

**> import <- c(35, 40, 50)**

**>**

**> barplot(rbind(export, import),**

**+ beside = TRUE,**

**+ names.arg = years,**

**+ col = c("blue", "red"),**

**+ legend = c("Export", "Import"),**

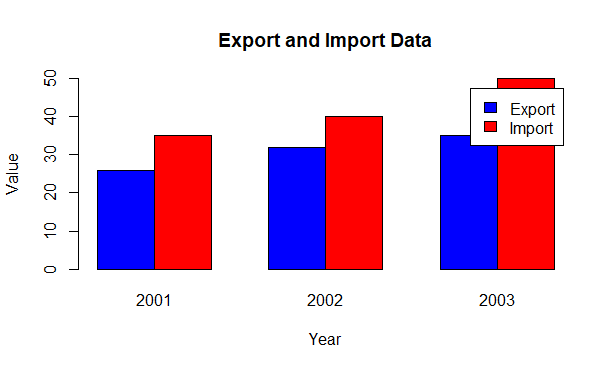
**+ main = "Export and Import Data",**

**+ xlab = "Year",**

**+ ylab = "Value")**

**>**

**Output:**

****

**Q2. Write a Python program build Decision Tree Classifier using Scikit-learnpackage for diabetes data set (download database from** [**https://www.kaggle.com/uciml/pima-indiansdiabetes-database**](https://www.kaggle.com/uciml/pima-indiansdiabetes-database)**)**

[**https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset/blob/master/diabetes.csv**](https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset/blob/master/diabetes.csv)

**Ans**

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

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

url = 'https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset/raw/master/diabetes.csv'

data = pd.read\_csv(url)

X = data.drop('Outcome', axis=1)

y = data['Outcome']

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)

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

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

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

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

**Output:**

**Accuracy: 0.77**

**precision recall f1-score support**

**0 0.84 0.79 0.81 99**

**1 0.66 0.73 0.69 55**

**accuracy 0.77 154**

**macro avg 0.75 0.76 0.75 154**

**weighted avg 0.77 0.77 0.77 154**

**Slip 17**

**Q1. Write a R program to get the first 20 Fibonacci numbers.**

**Ans**

**> fibonacci <- numeric(20)**

**> fibonacci[1] <- 0**

**> fibonacci[2] <- 1**

**>**

**> for (i in 3:20) {**

**+ fibonacci[i] <- fibonacci[i - 1] + fibonacci[i - 2]**

**+ }**

**>**

**> fibonacci**

**[1] 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610**

**[17] 987 1597 2584 4181**

**>**

**Q2. Write a python programme to implement multiple linear regression modelfor stock market data frame as follows: Stock\_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2 016,20,16,2016,2016,2016,2016,2016,2016,2016,2016,2016], 'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],**

**'Interest\_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1 .75,1.75,1.75,1.75,1.75,1.75],**

**'Unemployment\_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5 .9,6.2,6.2,6.1], 'Stock\_Index\_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047, 965,943,958,971,949,884,866,876,822,704,719] } And draw a graph of stock market price verses interest rate.**

**ANS:**

**pip install pandas numpy statsmodels matplotlib**

import pandas as pd

import numpy as np

import statsmodels.api as sm

import matplotlib.pyplot as plt

Stock\_Market = {

'Year': [2017]\*12 + [2016]\*12,

'Month': list(range(12, 0, -1)) \* 2,

'Interest\_Rate': [

2.75, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.25, 2.25, 2.25, 2, 2,

2, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75

],

'Unemployment\_Rate': [

5.3, 5.3, 5.3, 5.3, 5.4, 5.6, 5.5, 5.5, 5.5, 5.6, 5.7, 5.9,

6, 5.9, 5.8, 6.1, 6.2, 6.1, 6.1, 6.1, 5.9, 6.2, 6.2, 6.1

],

'Stock\_Index\_Price': [

1464, 1394, 1357, 1293, 1256, 1254, 1234, 1195, 1159, 1167, 1130, 1075,

1047, 965, 943, 958, 971, 949, 884, 866, 876, 822, 704, 719

]

}

df = pd.DataFrame(Stock\_Market)

X = df[['Interest\_Rate', 'Unemployment\_Rate']]

y = df['Stock\_Index\_Price']

X = sm.add\_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())

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

plt.scatter(df['Interest\_Rate'], df['Stock\_Index\_Price'], color='blue')

plt.title('Stock Market Price vs Interest Rate')

plt.xlabel('Interest Rate')

plt.ylabel('Stock Index Price')

plt.grid()

plt.show()

**OUTPUT:**

**OLS Regression Results**

**==============================================================================**

**Dep. Variable: Stock\_Index\_Price R-squared: 0.898**

**Model: OLS Adj. R-squared: 0.888**

**Method: Least Squares F-statistic: 92.07**

**Date: Wed, 25 Sep 2024 Prob (F-statistic): 4.04e-11**

**Time: 00:47:38 Log-Likelihood: -134.61**

**No. Observations: 24 AIC: 275.2**

**Df Residuals: 21 BIC: 278.8**

**Df Model: 2**

**Covariance Type: nonrobust**

**=====================================================================================**

**coef std err t P>|t| [0.025 0.975]**

**-------------------------------------------------------------------------------------**

**const 1798.4040 899.248 2.000 0.059 -71.685 3668.493**

**Interest\_Rate 345.5401 111.367 3.103 0.005 113.940 577.140**

**Unemployment\_Rate -250.1466 117.950 -2.121 0.046 -495.437 -4.856**

**==============================================================================**

**Omnibus: 2.691 Durbin-Watson: 0.530**

**Prob(Omnibus): 0.260 Jarque-Bera (JB): 1.551**

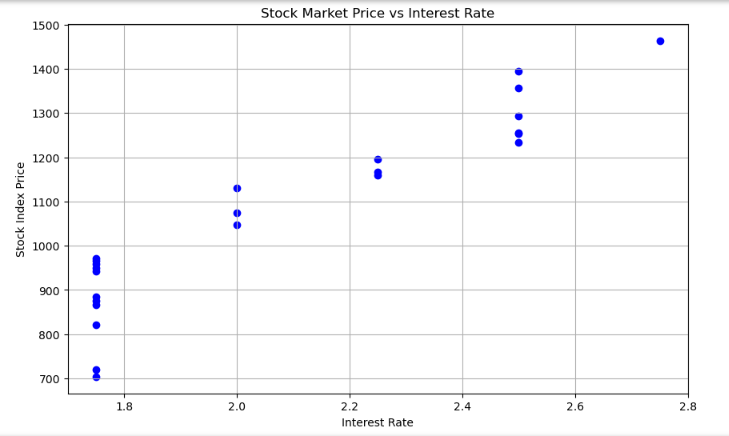
**Skew: -0.612 Prob(JB): 0.461**

**Kurtosis: 3.226 Cond. No. 394.**

**==============================================================================**

**Notes:**

**[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.**

****

**Slip 18**

**Q1. Write a R program to find the maximum and the minimum value of a givenvector [10 Marks]**

**Ans**

**> vector <- c(12, 5, 23, 8, 19)**

**>**

**> max\_value <- max(vector)**

**> min\_value <- min(vector)**

**>**

**> max\_value**

**[1] 23**

**> min\_value**

**[1] 5**

**Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse theperformance of the model (Use sklearn package) x = np.array([1,2,3,4,5,6,7,8]) y = np.array([7,14,15,18,19,21,26,23])**

**Ans**

import numpy as np

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)

y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

model = LinearRegression()

model.fit(x, y)

b0 = model.intercept\_

b1 = model.coef\_[0]

y\_pred = model.predict(x)

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

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

print(f'Estimated coefficients:')

print(f'b0 (intercept): {b0}')

print(f'b1 (slope): {b1}')

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

print(f'R^2 Score: {r2}')

plt.scatter(x, y, color='blue', label='Data Points')

plt.plot(x, y\_pred, color='red', label='Regression Line')

plt.title('Simple Linear Regression')

plt.xlabel('X')

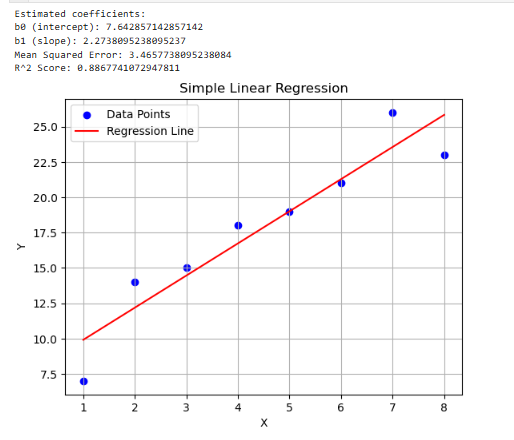
plt.ylabel('Y')

plt.legend()

plt.grid()

plt.show()

**Output:**

****

**Slip 19**

**Q1. Write aR program to create a Dataframes which contain details of 5 Studentsand display the details. Students contain (Rollno,Studname,Address,Marks)**

**Ans**

**> students <- data.frame(**

**+ Rollno = 1:5,**

**+ Studname = c("Alice", "Bob", "Charlie", "David", "Eva"),**

**+ Address = c("123 St", "456 Ave", "789 Blvd", "101 Rd", "202 Ln"),**

**+ Marks = c(85, 90, 78, 88, 92)**

**+ )**

**>**

**> print(students)**

**Rollno Studname Address Marks**

**1 1 Alice 123 St 85**

**2 2 Bob 456 Ave 90**

**3 3 Charlie 789 Blvd 78**

**4 4 David 101 Rd 88**

**5 5 Eva 202 Ln 92**

**Q2. Write a python program to implement multiple Linear Regression modelfor a car dataset. Dataset can be downloaded from: ( https://www.w3schools.com/python/python\_ml\_multiple\_regression.asp )**

**(** [**https://www.kaggle.com/datasets/swapnil1894/multiple-regression**](https://www.kaggle.com/datasets/swapnil1894/multiple-regression) **)**

**Ans**

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

# Load data from CSV

data = pd.read\_csv("C:/Users/HP/Downloads/1.02.Multiplelinearregression.csv")

# Check the columns in the DataFrame

print(data.columns)

# Replace 'SAT', 'Rand' with the correct feature columns if they differ

X = data[['SAT', 'Rand 1,2,3']]

y = data['GPA']

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

model = LinearRegression()

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

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

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

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

print("Mean Squared Error:", mse)

print("R-squared:", r2)

**Output:**

**Index(['SAT', 'GPA', 'Rand 1,2,3'], dtype='object')**

**Mean Squared Error: 0.05766690113933144**

**R-squared: 0.417127123666891**

**Slip 20**

**Q1. Write a R program to create a data frame from four given vectors.**

**Ans**

**> vector1 <- c(1, 2, 3, 4)**

**> vector2 <- c("A", "B", "C", "D")**

**> vector3 <- c(TRUE, FALSE, TRUE, FALSE)**

**> vector4 <- c(10.5, 20.3, 15.6, 30.1)**

**>**

**> data\_frame <- data.frame(**

**+ Numeric = vector1,**

**+ Letters = vector2,**

**+ Logical = vector3,**

**+ Float = vector4**

**+ )**

**>**

**> print(data\_frame)**

**Numeric Letters Logical Float**

**1 1 A TRUE 10.5**

**2 2 B FALSE 20.3**

**3 3 C TRUE 15.6**

**4 4 D FALSE 30.1**

**>**

**Q2. Write a python program to implement hierarchical Agglomerativeclustering algorithm. (Download Customer.csv dataset from github.com).**

**Ans**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

data = pd.read\_csv("C:/Users/HP/Downloads/customers.csv")

data = data.select\_dtypes(include=[np.number]) # Select only numerical columns

data\_scaled = StandardScaler().fit\_transform(data)

linked = linkage(data\_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Sample Index')

plt.ylabel('Distance')

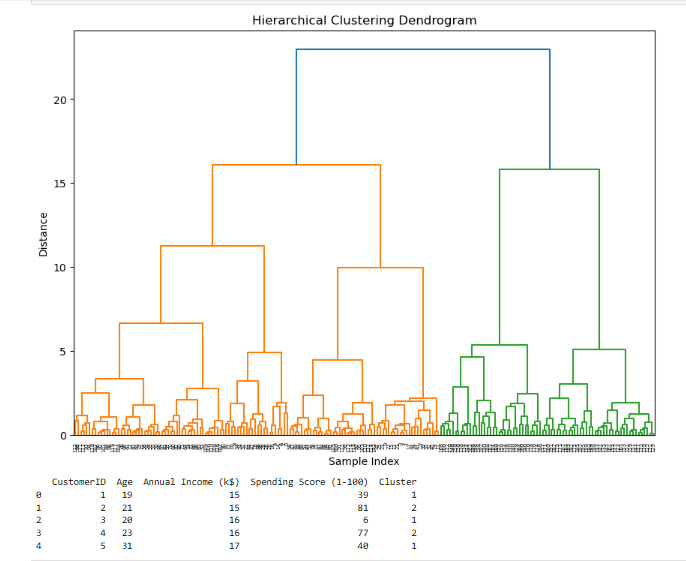
plt.show()

model = AgglomerativeClustering(n\_clusters=3) # Set the number of clusters

clusters = model.fit\_predict(data\_scaled)

data['Cluster'] = clusters

print(data.head())

****