SLIP 1

QUE1=>

vector1 = seq(10,40 , length.out=4)

vector2 = c(20, 10, 40, 40)

print("Original Vectors:")

print(vector1)

print(vector2)

add vector1+vector2

cat("Sum of vector is ",add, "\n")

sub\_vector= vector1-vector2

cat("Substraction of vector is ",sub\_vector, "\n")

mul\_vector= vector1 \* vector2

cat("Multiplication of vector is ",mul\_vector, "\n")

div\_vector = vector1 / vector2

cat("Division of two Vectors:",div\_vector,"\n")

QUE2=>

import numpy as np

from sklearn.linear\_model import LinearRegression

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

#values from student\_dataset.csv file

X = np.array([2.5,5.1,3.2,8.5,3.5]).reshape(-1, 1)

y = np.array([21,47,27,75,30])

# Create a linear regression model

model = LinearRegression()

model.fit(X, y)

# Predict the values using the trained model

y\_pred = model.predict(X)

# Calculate mean absolute error (MAE)

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

# Calculate mean squared error (MSE)

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

# Calculate root mean squared error (RMSE)

rmse = np.sqrt(mse)

# Print the results

print("Mean Absolute Error (MAE):", mae)

print("Mean Squared Error (MSE):", mse)

print("Root Mean Squared Error (RMSE):", rmse)

SLIP2

QUE1=>

table<-function(number)

{

for(t in 1:10)

{

print(paste(number,'\*',t,'=',number\*t))

}

}

table(2)

QUE2=>

#Importing libraries

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

data = make\_blobs(n\_samples=300, n\_features=2, centers=5,

cluster\_std=1.8,random\_state=101)

data[0].shape

data[1]

plt.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='brg')

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=5)

kmeans.fit(data[0])

kmeans.cluster\_centers\_

kmeans.labels\_f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(10,6))

ax1.set\_title('K Means')

ax1.scatter(data[0][:,0],data[0][:,1],c=kmeans.labels\_,cmap='brg')

ax2.set\_title("Original")

ax2.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='brg')

SLIP3

QUE1=>

n=567

Reverse=function(n)

{

sum=0

rev=0

while(n>0)

{

r=n%%10

sum=sum+r

rev=rev\*10+r

n=n%/%10

}

cat("reverse=",rev)

cat("sum of number=",sum)

}

Reverse(n)

QUE2=>

import numpy as np# Data

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])

# Calculate the mean of x and y

mean\_x = np.mean(x)

mean\_y = np.mean(y)

# Calculate the differences between each data point and the mean

x\_diff = x - mean\_x

y\_diff = y - mean\_y

# Calculate b1 (slope) by taking the dot product of x\_diff and y\_diff divided by the dot product of x\_diff with itself

b1 = np.sum(x\_diff \* y\_diff) / np.sum(x\_diff \*\* 2)

# Calculate b0 (intercept) using the formula b0 = mean(y) - b1 \* mean(x)

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

# Print the estimated coefficients

print("Estimated Coefficient b0 (Intercept):", b0)

print("Estimated Coefficient b1 (Slope):", b1)

SLIP4

QUE1=>

matrix1<-matrix(c(1,2,3,4,5,6),nrow=2)

print(matrix1)

matrix2<-matrix(c(7,8,9,10,11,12),nrow=2)

print(matrix2)

result<-matrix1+matrix2

cat("Addition : ","\n")

print(result)

QUE2=>

# Define the dataset

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']

# Define the new tuple [0:Overcast, 2:Mild]

new\_weather = 'Overcast'

new\_temp = 'Mild'

# Calculate the prior probabilities

total\_samples = len(play)

count\_yes = play.count('Yes')

count\_no = play.count('No')

prior\_yes = count\_yes / total\_samples

prior\_no = count\_no / total\_samples

# Calculate the likelihood for 'Yes' class

likelihood\_yes = (weather.count(new\_weather) / count\_yes) \* (temp.count(new\_temp) / count\_yes)

# Calculate the likelihood for 'No' class

likelihood\_no = (weather.count(new\_weather) / count\_no) \* (temp.count(new\_temp) / count\_no)

# Calculate the posterior probabilities

posterior\_yes = prior\_yes \* likelihood\_yes

posterior\_no = prior\_no \* likelihood\_no

# Predict the class with the highest posterior probability

if posterior\_yes > posterior\_no:

prediction = 'Yes'

else:

prediction = 'No'

print("Predicted class for [{}: {}, {}: {}] is: {}".format(0, new\_weather, 2, new\_temp, prediction))

SLIP5

QUE1=>

# Create two factors

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

factor2 <- factor(c("X", "Y", "Z"))

# Concatenate the factors

concatenated\_factors <- c(factor1, factor2)

# Print the result

print(concatenated\_factors)

QUE2=>

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import metrics

pima = pd.read\_csv("diabetes.csv")

pima.head()

import seaborn as sns

corr = pima.corr()

ax = sns.heatmap(corr, vmin=-1, vmax=1, center=0,

cmap=sns.diverging\_palette(20, 220, n=200),

square=True

)

ax.set\_xticklabels(

ax.get\_xticklabels(),

rotation=45,

horizontalalignment='right'

);

# feature selection

feature\_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age', 'Glucose',

'BloodPressure',

'DiabetesPedigreeFunction']

x = pima[feature\_cols]

y = pima.Outcome

# split data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state=1)

# build model

classifier = DecisionTreeClassifier()

classifier = classifier.fit(X\_train, Y\_train)

# predict

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

print(y\_pred)

from sklearn.metrics import confusion\_matrix

confusion\_matrix(Y\_test, y\_pred)

print(confusion\_matrix(Y\_test, y\_pred))

# accuracy

print("Accuracy:", metrics.accuracy\_score(Y\_test,y\_pred))

SLIP6

QUE1=>

# Create two vectors

vector1 <- c("apple", "banana", "orange", "apple", "grape")

vector2 <- c(10, 20, 15, 10, 25)

# Create a data frame

my\_data\_frame <- data.frame(Fruit = vector1, Quantity = vector2)

# Display the data frame

print("Original Data Frame:")

print(my\_data\_frame)

# Identify and display duplicate rows

duplicate\_rows <- my\_data\_frame[duplicated(my\_data\_frame) | duplicated(my\_data\_frame, fromLast = TRUE), ]

print("\nDuplicate Rows:")

print(duplicate\_rows)

QUE2=>

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Customer.csv')

X = dataset.iloc[:, [3, 4]].values

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

SLIP7

QUE1=>

print("Sequence of numbers from 20 to 50:")

print(seq(20,50))

print("Mean of numbers from 20 to 60:")

print(mean(20:60))

print("Sum of numbers from 51 to 91:")

print(sum(51:91))

QUE2=>

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

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

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

slope, intercept, r, p, std\_err = stats.linregress(x, y)

def myfunc(x):

return slope \* x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

SLIP8

QUE1=>

Fibonacci <- numeric(10)

Fibonacci[1] <- Fibonacci[2] <- 1

for (i in 3:10) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]

print("First 10 Fibonacci numbers:")

print(Fibonacci)

QUE2=> (output is not displaying though it is the program given by maam)

# Importing Preprocessing Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Importing Datasets

dataset = pd.read\_csv('CC GENERAL.csv')

X = dataset.iloc[:, 1:].values

# Dataset Contains Multiple Missing values

# Replacing Missing Value by Most Repeated/Frequent Number in that column

# Use Imputer with strategy 'most\_frequent'

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan, strategy="most\_frequent")

imputer = imputer.fit(X)

X = imputer.transform(X)

# Applying Feature Scalling with StandardScaler

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X = sc\_X.fit\_transform(X)

# For Finding Optimal Number of Cluster use Elbow Method

from sklearn.cluster import KMeans

# Apply K-Means Again With Optimal Number of Cluster that we got from Elbow method i.e. 8

kmeans = KMeans(n\_clusters=8, init='k-means++', max\_iter=300, n\_init=10,

random\_state=0)

y\_kmeans = kmeans.fit\_predict(X)

# Finally Append new Column i.e Cluster to Actual Dataset

dataset['Cluster'] = y\_kmeans

dataset.head()

SLIP9

QUE1=>

Employees = data.frame(Name=c("Amit S","Dikisha R","Shweta J", "Jikita A","Riya M"),

Gender=c("M","M","F","F","F"),

Age=c(23,22,25,26,32),

Designation=c("Clerk","Manager","Exective","CEO","ASSISTANT"),

SSN=c("123-34-2346","123-44-779","556-24-433","123-98-987","679-77-576"))

print("Details of the employees:")

print(Employees)

QUE2=>

#Import scikit-learn dataset library

from sklearn import datasets

#Load dataset

cancer = datasets.load\_breast\_cancer()

# print the names of the 13 features

print("Features: ", cancer.feature\_names)

# print the label type of cancer('malignant' 'benign')

print("Labels: ", cancer.target\_names)

# print data(feature)shape

cancer.data.shape

# print the cancer data features (top 5 records)

print(cancer.data[0:5])

# print the cancer labels (0:malignant, 1:benign)

print(cancer.target)

# Import train\_test\_split function

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

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target,

test\_size=0.3,random\_state=109) # 70% training and 30% test

#Import svm model

from sklearn import svm

#Create a svm Classifier

clf = svm.SVC(kernel='linear') #Linear Kernel

#Train the model using the training sets

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

#Predict the response for test dataset

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

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Model Accuracy: how often is the classifier correct?

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

SLIP10

QUE1=>

nums = c(10, 20, 30, 40, 50, 60)

print('Original vector:')

print(nums)

print(paste("Maximum value of the said vector:",max(nums)))

print(paste("Minimum value of the said vector:",min(nums)))

QUE2=>

\*\*doubt\*\*

(for apriori algorithm program to run we’ve to install apyori library first)

SLIP11

QUE1=>

list1 = list("x", "y", "z")

list2 = list("X", "Y", "Z", "x", "y", "z")

print("Original lists:")

print(list1)

print(list2)

print("All elements of list1 that are not in list2:")

setdiff(list2, list1)

QUE2=>

import matplotlib.pyplot as mtp

import pandas as pd

dataset = pd.read\_csv('wholesaleCustomer.csv')

dataset

x = dataset.iloc[:, [3, 4]].values

print(x)

import scipy.cluster.hierarchy as shc

dendro = shc.dendrogram(shc.linkage(x, method="ward"))

mtp.title("Dendrogrma Plot")

mtp.ylabel("Euclidean Distances")

mtp.xlabel("Customers")

mtp.show()

from sklearn.cluster import AgglomerativeClustering

hc= AgglomerativeClustering(n\_clusters=5, affinity='euclidean', linkage='ward')

y\_pred= hc.fit\_predict(x)

mtp.scatter(x[y\_pred == 0, 0], x[y\_pred == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')

mtp.scatter(x[y\_pred == 1, 0], x[y\_pred == 1, 1], s = 100, c = 'green', label = 'Cluster 2')

mtp.scatter(x[y\_pred== 2, 0], x[y\_pred == 2, 1], s = 100, c = 'red', label = 'Cluster 3')

mtp.scatter(x[y\_pred == 3, 0], x[y\_pred == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

mtp.scatter(x[y\_pred == 4, 0], x[y\_pred == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

mtp.title('Clusters of customers')

mtp.xlabel('Milk')

mtp.ylabel('Grocery')

mtp.legend()

mtp.show()

SLIP12

QUE1=>

Employees = data.frame(empno=c(1,2,3,4,5),

empname=c("Amit S","Dikish R","Shweta J", "Jikita A","Riya M"),

Gender=c("M","M","F","F","F"),

Age=c(23,22,25,26,32),

Designation=c("Clerk","Manager","Exective","CEO","ASSISTANT"))

print("Details of the employees:")

print(Employees)

QUE2=>

import pandas

from sklearn import linear\_model

df = pandas.read\_csv("car.csv")

X = df[['Weight', 'Volume']]

y = df['CO2']

regr = linear\_model.LinearRegression()

regr.fit(X, y)

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)

SLIP13

QUE1=>

digits <- c(7,2,6,3,4,8)

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

# Plot the chart.

pie(digits, Frequency)

QUE2=>

import pandas as pd

students\_data = pd.read\_csv("StudentsPerformance.csv")

# Display the shape of the dataset

print("Shape of the dataset:", students\_data.shape)

# Display the top rows of the dataset

print("\nTop rows of the dataset:")

print(students\_data.head()

SLIP14

QUE1=>

list\_data <- list("Ram Sharma","Sham Varma","Raj Jadhav", "Ved Sharma")

print(list\_data)

new\_Emp <-"Kavya Anjali"

list\_data <-append(list\_data,new\_Emp)

print(list\_data)

list\_data[3] <- NULL

print(list\_data)

QUE2=>

\*\*doubt\*\*

(for apriori algorithm program to run we’ve to install apyori library first)

SLIP15

QUE1=>

vector1 = seq(10,40 , length.out=4)

vector2 = c(20, 10, 40, 40)

print("Original Vectors:")

print(vector1)

print(vector2)

add= vector1+vector2

cat("Sum of vector is ",add, "\n")

sub\_vector= vector1-vector2

cat("Substraction of vector is ",sub\_vector, "\n")

mul\_vector= vector1 \* vector2

cat("Multiplication of vector is ",mul\_vector, "\n")

div\_vector = vector1 / vector2

cat("Division of two Vectors:",div\_vector)

QUE2=>

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder

# Load the dataset

data = pd.DataFrame({

'Age': [36, 42, 23, 52, 43, 44, 66, 35, 52, 35, 24, 18, 45],

'Experience': [10, 12, 4, 4, 21, 14, 3, 14, 13, 5, 3, 3, 9],

'Rank': [9, 4, 6, 4, 8, 5, 7, 9, 7, 9, 5, 7, 9],

'Nationality': ['UK', 'USA', 'N', 'USA', 'USA', 'UK', 'N', 'UK', 'N', 'N', 'USA', 'UK', 'UK'],

'Go': ['NO', 'NO', 'NO', 'NO', 'YES', 'NO', 'YES', 'YES', 'YES', 'YES', 'NO', 'YES', 'YES']

})

# Encode 'Nationality' feature

label\_encoder = LabelEncoder()

data['Nationality'] = label\_encoder.fit\_transform(data['Nationality'])

# Define features (X) and target (y)

X = data.drop(columns=['Go'])

y = data['Go']

# Create a Decision Tree Classifier

clf = DecisionTreeClassifier(random\_state=42)

# Train the classifier on the entire dataset

clf.fit(X, y)

# Create a new data point for prediction

new\_data = pd.DataFrame({

'Age': [40],

'Experience': [10],

'Rank': [7],

'Nationality': ['USA'})

# Encode 'Nationality' feature

new\_data['Nationality'] = label\_encoder.transform(new\_data['Nationality'])

# Make a prediction for the new data point

prediction = clf.predict(new\_data)

print("Can the comedian go to the show:", prediction[0])

SLIP16

QUE1=>

# Import lattice

library(lattice)

# Create data

gfg <- data.frame(x = c(26,35,32,40,35,50),

grp = rep(c("group 1", "group 2",

"group 3"),

each = 2),

subgroup = LETTERS[1:2])

# Create grouped barplot using lattice

barchart(x ~ grp, data = gfg, groups = subgroup)

QUE2=>

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import metrics

pima = pd.read\_csv("diabetes.csv")

pima.head()

import seaborn as sns

corr = pima.corr()

ax = sns.heatmap(

corr,

vmin=-1, vmax=1, center=0,

cmap=sns.diverging\_palette(20, 220, n=200),

square=True

)

ax.set\_xticklabels(

ax.get\_xticklabels(),

rotation=45,

horizontalalignment='right'

);

# feature selection

feature\_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age', 'Glucose',

'BloodPressure',

'DiabetesPedigreeFunction']

x = pima[feature\_cols]

y = pima.Outcome

# split data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state=1)

# build model

classifier = DecisionTreeClassifier()

classifier = classifier.fit(X\_train, Y\_train)

# predict

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

print(y\_pred)

from sklearn.metrics import confusion\_matrix

confusion\_matrix(Y\_test, y\_pred)

print(confusion\_matrix(Y\_test, y\_pred))

# accuracy

print("Accuracy:", metrics.accuracy\_score(Y\_test,y\_pred))

SLIP17

QUE1=>

Fibonacci <- numeric(20)

Fibonacci[1] <- Fibonacci[2] <- 1

for (i in 3:20) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]

print("First 20 Fibonacci numbers:")

print(Fibonacci)

QUE2=>

import pandas as pd

import matplotlib.pyplot as plt

Stock\_Market = {'Year':

[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,

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]

}

df = pd.DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

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

plt.title('Stock Index Price Vs Interest Rate', fontsize=14)

plt.xlabel('Interest Rate', fontsize=14)

plt.ylabel('Stock Index Price', fontsize=14)

plt.grid(True)

plt.show()

SLIP18

QUE1=>

nums = c(10, 20, 30, 40, 50, 60)

print('Original vector:')

print(nums)

print(paste("Maximum value of the said vector:",max(nums)))

print(paste("Minimum value of the said vector:",min(nums)))

QUE2=>

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

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

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

slope, intercept, r, p, std\_err = stats.linregress(x, y)

def myfunc(x):

return slope \* x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

SLIP19

QUE1=>

Students = data.frame(Rollno=c(21,22,23,24,25),

Name=c("Riya M","Shweta J","Aarya D", "JAMES A","LAURA M"),

Addresss=c("Bhekrai nagar","Hadapsar","Uruli kanchan","Hadapsar","Bhekrai nagar"),

Marks=c(80,67,90,92,70))

print("Details of the Students:")

print(Students)

QUE2=>

import pandas

from sklearn import linear\_model

from sklearn.linear\_model import LinearRegression

df = pandas.read\_csv("car.csv")

X = df[['Weight', 'Volume']]

y = df['CO2']

regr = linear\_model.LinearRegression()

regr.fit(X, y)

test\_y = regr.predict(X)

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)

SLIP20

QUE1=>

name = c('Aarya', 'Riya', 'Shweta', 'Anjali', 'Geeta', 'Mayuri', 'Kirti', 'Akansha', 'Kavita', 'Jagruti')

score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)

attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)

qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

print("Original data frame:")

print(name)

print(score)

print(attempts)

print(qualify)

df = data.frame(name, score, attempts, qualify)

print(df)

QUE2=>

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Customer.csv')

X = dataset.iloc[:, [3, 4]].values

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()