```
#s1q1
#Q1.Write a R program to add,multiply and divide two vectors of
integertype. (Vector #length should be minimum 4)
vector1 <- c(10, 20, 30, 40, 50)
vector2 <- c(2, 4, 6, 8, 10)

addition_result <- vector1 + vector2 subtraction_result <- vector1 vector2
multiplication_result <- vector1 * vector2 division_result <- vector1 /
vector2

cat("Vector 1:", vector1, "\n")
cat("Vector 2:", vector2, "\n")
cat("\nResults:\n")
cat("Addition (Vector 1 +Vector 2):", addition_result, "\n")
cat("Subtraction (Vector 1 Vector 2):", subtraction_result, "\n")

cat("Multiplication (Vector 1 * Vector 2):", multiplication_result,
"\n") cat("Division (Vector 1 / Vector 2):", division_result, "\n")

#s1q2
"""Q2. Consider the student data set. It can be downloaded from:
https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaq0 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 error"""
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/student_scores.csv")
print(data.head())
```

```
x=data[ [ 'Hours']]#[ ['hours']] make sure it 2d array
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)
print(y_test)
y_pred=model.predict(x_test)
print(y_pred)
mae=mean_absolute_error(y_pred,y_test)
mse=mean_squared_error(y_pred,y_test)
rmse=np.sqrt(mse)

print("Mean Absolute error",mae)
print("Mear Sqaured error",mse)
print("root mean squared error", rmse)
```

```
#s2q1
#Q1. Write an R program to calculate the multiplication table using a function mul_table<- function(n)
{
for(i in 1:10)
{
res<-n*i
cat(res, "\n")
}
}
mul_table(4)
```

#s2q2

```
'''Q2.Write a python program to implement k-means algorithms on
asynthetic dataset.
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=300, centers=4, n_features=2,
random_state=42)

model = KMeans(n_clusters=4, random_state=42)
model.fit(X)

labels =
model.labels_
centroids = model.cluster_centers_

print("Cluster Labels:\n", labels) print("Centroids:\n", centroids)
```

#s3q1

```
#Q1. Write a R program to reverse a number and also calculate the sum
ofdigits of that number.
rev_num<- function(n)
{
```

```
rev<-0
while(n!=0)
{




}
}
rem<-n%%10
rev<-rev * 10 + rem
n<-n %/% 10

cat (rev, "\n")#/n_is imp

rev_num(876)



#s3q2
'''Q2. Consider the following observations/data. And apply simple
linear regression and find out estimated coefficients b0 and
bl.( 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]
import pandas as pd
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])

lenn=len(x)
sum_x=np.sum(x)
sum_y=np.sum(y)
sum_xy=np.sum(x*y)
sum_x_sq=np.sum(x**2)
```

```
b1=(lenn * sum_xy - sum_x * sum_y)/(lenn * sum_x_sq

b0=(sum_y
b1 * sum_x)/lenn

print("bo:",b0)
print("b1",b1)
sum_x**2)
```

```
#s4q1
#Q1. Write a R program to calculate the sum of two matrices of given
size add_matrices <- function(matrix1, matrix2) {

}
return(matrix1 + matrix2)
```

```
rows <- as.integer(readline(prompt "Enter the number of rows:
")) cols <- as.integer(readline(prompt = "Enter the number of
columns: "))

cat("Enter elements of the first matrix:\n")
matrix1 <- matrix(scan(n = rows * cols), nrow = rows, ncol = cols,
byrow

cat("Enter elements of the second matrix:\n")
= TRUE)


matrix2 <- matrix(scan(n = rows * cols), nrow = rows,ncol =cols, byrow
TRUE)
```

```
result <- add_matrices(matrix1, matrix2)

cat("The sum of the matrices is:\n") print(result)

#s4q2
'''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.

from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

weather=
['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny','Sunny', 'Rainy', 'Sunny',
'Overca st', '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'
]

encoder=LabelEncoder()
weather_encoded-encoder.fit_transform(weather)
print("Whearer encoded", weather_encoded)

temp_encoded-encoder.fit_transform(temp)print("Temprature encoded", temp_encoded)

play_encoded-encoder.fit_transform(play) print("Play Encoded", play_encoded)

model=GaussianNB()

feature=list(zip (weather_encoded, temp_encoded))#combine wheather and temp

model.fit(feature, play_encoded)

predicted=model.predict([[0,2]])#use [[]]

predicted_lable-encoder.inverse_transform(predicted)

print("Predition for Overcast and mild", predicted_lable)
```

```
#s5q1 Q1. Write a R program to concatenate two given factors
f1<-factor(c("a","b","c"))
f2<-factor(c("1","2","3"))
comb<-c(f1, f2)
comb_f<-factor (comb)#convert into factor

cat("\nFactors", levels(comb_f))#use levels
```

```
#s5q2
'''2. Write a Python program build Decision Tree Classifier using
Scikit- learn package for
diabetes data set (download database from
https://www.kaggle.com/uciml/pimaindians-diabetes-database)

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,
confusion_matrix

data =
pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/diabetes.csv")

print(data.head())
```

```
x = data.drop(['Outcome'], axis=1)
x
```

```
# Get everything except 'Outcome'
y data['Outcome'] # Target variable

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

model
DecisionTreeClassifier(random_state=42)

model.fit(x_train, y_train)

y_pred
model.predict(x_test)

print("Accuracy score:" , accuracy_score(y_test, y_pred))
print("Classification report:\n", classification_report(y_test,
y_pred))
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
```

```
#s6q1 Q1. Write a R programto create a data frame using two given
vectors and displaythe duplicate elements.
vector1 <- c(1, 2, 3, 1, 4, 2)
vector2 <- c(5, 2, 7, 8, 2, 1)

df <- data.frame(Column1 = vector1, Column2 = vector2)

cat("Original Data Frame:\n")
print(df)
```

```
common_elements <- intersect(vector1, vector2)


cat("\nCommon Elements in Both Vectors:\n")
print(common_elements)
```

```
#s6q2
'''Q2. Write a python program to implement hierarchical
Agglomerative clusteringalgorithm. (Download Customer.csv dataset
from github.com)'' '
import pandas as pd
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import LabelEncoder
data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/customers.csv")
print(data)
le=LabelEncoder()
data['Gender']=le.fit_transform(data['Gender'])
x=data[['Gender', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

model=AgglomerativeClustering(n_clusters=5,metric='euclidean',
linkage='ward')

res=model.fit_predict(x) data['cluster']=res

print(data[['CustomerID', 'cluster']])#it shows which customer
belongs to which cluster
```

```
#s7q1 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.
sequence_20_50 <- seq(20,50)
cat("Sequence from 20 to 50:", sequence_20_50,"\n")

mean_20_60 <- mean (20:60)
cat("Mean of numbers from 20 to 60:", mean_20_60, "\n")

sum_51_91 <- sum(51:91)
cat("Sum of numbers from 51 to 91:", sum_51_91,"\n")
```

#s7q2

```
'''Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients bl and b1 Also analyse theperformance of the model
(Use sklearn package)
x = np.array([1,2,3,4,5,6,7,8])
x
y = np.array([7,14,15,18,19,21,26,23])'''
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score

x=np.array([1,2,3,4,5,6,7,8]).reshape(-1,1)#reshape
y=np.array([7,14,15,18,19,21,26,23])

model=LinearRegression()
```

```
model.fit(x,y)

b0=model.intercept_
b1=model.coef_[0]

print("b0:",b0)
print("bl:",b1)

y_pred=model.predict(x)

mae=mean_absolute_error(y,y_pred)
mse=mean_squared_error(y,y_pred)
r2=r2_score(y,y_pred)

print("Mean absolute error",mae) print("Mean Squared error", mse)
print("r2 score", r2)
```

```
#s8q1 Q1. Write a R program to get the first 10 Fibonacci numbers.
fib 10 <- numeric(10)
fib_10[1] <- 0
fib_10[2] <- 1

for (i in 3:10) {

}
fib_10[i] <- fib_10[i-1] + fib_10[i-2]


cat("First 10 Fibonacci numbers are:", fib_10, "\n")
```

```
#s8q2
'''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)'''
import pandas as pd
from sklearn.cluster import KMeans

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/CC GENERAL.csv")


x=data.drop(columns=['CUST_ID'])#everything except custid

x.fillna(x.mean(), inplace=True)#fill missing value only after emoving
the cust_id(i.e only on x)

print(x)
model=KMeans(n_clusters=5, random_state=42)
res=model.fit_predict(x)

data['cluster']=res

print(data[['CUST_ID','cluster']])
```

```
#s9q1 Q1. Write an R program to create a Data frames which contain details of 5 employees and display summary of the data
employees <- data.frame(
Name = c("Raj", "TT", "Rohit", "Swa", "Jnvi"),
Age = c( 28, 25, 30, 27, 26),
Department
Salary
)
c("HR", "IT", "Finance", "Marketing", "Sales"),
c(50000, 55000, 60000, 52000, 58000)

print(employees)
summary(employees)


#s9q2
''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.'''
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,recall_score,precision_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 (kernel='linear', random_state=42)
model.fit(x_train,y_train)

y_pred=model.predict(x_test)
print("Y_ACTUAL:",y_test)
print("Y_PREDICTED:",y_pred)

asl=accuracy_score (y_test,y_pred)
rs=recall_score(y_test,y_pred)
ps=precision_score(y_test,y_pred)

print("Accurecy score:",as1) print("Precison score:",ps)print("Recall
Score:",rs)
```

```
#s10q1
#Q1. Write a R program to find the maximum and the minimum value of a
givenvector

numbers <- c(12, 45,7, 89, 23, 56, 34)

max_value <- max(numbers)

min_value <- min(numbers)

cat("Maximum value:", max_value,"\n") cat("Minimum value:",
min_value, "\n")
```

```
#s10q2 INCOMPLETE
'''Q2. Write a Python Programme to read the dataset ("Iris.csv"). dataset download from (https://archive.ics.uci.edu/ml/datasets/iris) and apply Apriori algorithm.'''
import pandas as pd
from apyori import apriori

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/Iris.csv")
transactions=[]
for i in range(0,len(data)):
transactions.append([str(data.values[i, j]) for j in range(0, len(data.columns))])#convert into
transactions


rules = apriori(transactions, min_support=0.005,min_lift=3, min_length=2)
assc_rules=list(rules)

print(f"Total number of rules: {len(assc_rules)}\n")
print("hiiiii")
print(assc_rules[0])
```

```
#s11q1
#Q1. Write a R program to find all elements of a given list that are not inanother given list. list1 <- list("x", "y", "z")
list2 <- list("X","Y","Z", "x","y", "z")

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

print(result)
```

```
#s11q2
'''Q2. Write a python program to implement hierarchical clustering algorithm. (Download Wholesale customers data dataset from github.com).'
import pandas as pd
from sklearn.cluster import AgglomerativeClustering

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/customers.csv")
print(data)

x=data.iloc[:,[3,4]].values
```

```
model=AgglomerativeClustering(n_clusters=5, metric='euclidean',
linkage='ward')

res=model.fit_predict(x) data['cluster']=res

print(data[['CustomerID', 'cluster']])#it shows which customer
belongs to which cluster
```

```
#s12q1 Q1.Write a R program to create a Dataframes which contain
details of 5employees and #display the details.
employees <- data.frame(
empno = c(101,102, 103, 104, 105),
empname = c("Raj", "TT", "Rohit",
gender
Swa", "Jnvi"),
c("Male", "Male", "Male", "Female", "Female"),
age = c(28, 25,30, 27,26),
designation = c("Manager", "Analyst", "Developer", "Designer",
"Tester"), stringsAsFactors
FALSE #imp
)

print(employees)
```

```
#s12q2
'''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.a
sp'''
import pandas as pd
from sklearn.linear_model import LinearRegression

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/cars.csv")
x=data[ [ 'Volume', 'Weight']] x=data[['Volume'
y=data[ [ 'C02']]

print(x)
print(y)
model=LinearRegression()

model.fit(x,y)

co2=model.predict([[2300,1200]])
print("Predicteed emmision fro sample input 2300 and 1200:")
print(co2)
```

```
#s13q1
#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. dice_numbers <- c(1, 2,3, 4, 5, 6)
frequency <- c(7, 2, 6,3, 4, 8)

pie(frequency,
labels = dice_numbers,
main = "Distribution of Digits on a Dice",
col = rainbow(length(dice_numbers)))
legend("topright",
legend
dice_numbers,
fill = rainbow(length(dice_numbers)),title = "Dice Numbers")
```

#s13q2

```
'''Q2. Write a Python program to read "Students Performance.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 :'''
import pandas as pd
data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/Students
Performance.csv")

print("Shape of dataset:", data.shape)
```

```
print("Dataset")
print(data.head())
```

```
#s14q1
#Q1. Write a script in R to create a list of employees (name) and perform the following:
#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
employees <- list("Raj", "TT", "Swa", "Rohit", "Jnvi")

cat("Employees in the list:\n")
print(employees)

employees <- append(employees, "sjl")
cat("\nEmployees after adding a new one:\n")
print(employees)


employees <- employees [-3]
cat("\nEmployees after removing the third element:\n")
print(employees)
```

```
#s14q2
'''Q2. Write a Python Programme to apply Apriori algorithm on Groceries dataset. Dataset can be downloaded from
(https://github.com/amankharwal/Websitedata/blob/master/Groceries
```

_dataset.csv).

Also display support and confidence for each rule.

```
import pandas as pd
from apyori import apriori

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/Groceries_data set.csv")
transactions=[]
for i in range(0,len(data)):
transactions.append([str(data.values[i,j]) for j in range(0,len(data.columns))])
rules=apriori(transactions,min_support=0.005,min_cinfidence=0.3,min_lift=3,min_length=2)

rules=list(rules)

print("Rules Generated are \n")
print(rules[0],"\n","\n")


print("***
for i in range(0,len(rules)):
print(rules[i][0])
***\n\nRules in formatted manner \n\n")
```

```
#s15q1 Q1.Write a R program to add, multiply and divide two vectors of integer type. (vector length #should be minimum 4)
vector1 <- c(10, 20, 30, 40, 50)
vector2 <-c(2, 4, 6,8,10)

addition_result <- vector1 + vector2
```

```
subtraction_result <- vector1 - vector2 multiplication_result <- vector1
vector2 division_result <- vector1 / vector2

cat("Vector 1:", vector1, "\n")
cat("Vector 2:", vector2, "\n")
cat("\nResults:\n")
cat("Addition (Vector 1 + Vector 2):", addition_result, "\n")
cat("Subtraction (Vector 1 Vector 2):", subtraction_result, "\n")

cat("Multiplication (Vector 1 * Vector 2):", multiplication_result,
"\n") cat("Division (Vector 1 / Vector 2):", division_result, "\n")
```

#s15q2

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://www.w3schools.com/python/python_ml_decision_tree.asp

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

data = pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/shows.csv")

print(data)
le=LabelEncoder()
data['Nationality']=le.fit_transform(data['Nationality'])
data['Go']=le.fit_transform(data['Go'])
print(data)
features =
['Exeperince', 'Age', 'Rank', 'Nationality']
x= data[features]
y
```

```
clf clf
data['Go']


DecisionTreeClassifier()
clf.fit(x, y)

prediction
clf.predict([[10, 40, 7, 0]])

print("Prediction (1 for Yes, 0 for No):", prediction[0])
```

```
#s16q1
#Q1. Write a R program to create a simple bar plot of given data
data <- data.frame(


)
Year = c(2001, 2002, 2003),
Export = c(26, 32, 35),
Import
c(35, 40, 50)


barplot_height <- rbind(data$Export, data$Import)

barplot(barplot_height,
beside = TRUE,
names.arg = data$Year,
```

```
col = c("blue","red"),
main = "Export and Import Data",
xlab="Year"
ylab legend
"Value",
rownames (barplot_height))


#s16q2 check once
'''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)

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,
confusion_matrix

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/diabetes.csv")

model=DecisionTreeClassifier()

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.fit(x_train,y_train) y_pred-model.predict(x_test)

as1=accuracy_score (y_test,y_pred)
cr=classification_report(y_test,y_pred) cm=confusion_matrix(y_test,y_pred)

print("Accuracy score:",asl)
print("Classification Report: ",cr)
```

```
print("Confusion matrix",cm)
```

```
#s17q1 Q1.Write a R program to get the first 20 Fibonacci numbers
fibonacci_numbers <- numeric(20)
fibonacci_numbers [1]

fibonacci_numbers [2]<- 1

for (i in 3:20) {

}
fibonacci_numbers[i] <- fibonacci_numbers[i - 1] +
fibonacci_numbers[i - 2]


print(fibonacci_numbers)
```

```
#s17q2
    ''Q2.Write a python programme to implement multiple linear regression model for stock
market data frame as follows:
Stock_Market
              {'Year':
[2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2016, 2
016,20,16,2016,2016,2016,2016,2016,2016,2016,2016],
'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': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1
.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.''
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as
plt

stock_market= {
'Year':
    [2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017,2016,2016,2016,2016,2016,2016,2016,2016,2
016,2016,2016,2016],
'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':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1
.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] }

data=pd.DataFrame(stock_market)#convert into dataframe

x=data[ [ 'Interest_Rate', 'Unemployment_Rate']]
y=data['Stock_Index_Price']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=42)

model=Linear Regression()

model.fit(x_train,y_train)

print("b0",model. intercept_)
print("b1",model.coef_)

plt.figure(figsize=(10,6))
plt.scatter (data['Interest_Rate'], data['Stock_Index_Price'], color='red')
plt.title("INTREST VS STOCK PRICCE")
plt.xlabel("INTREST")
plt.ylabel("STOCK_INDEX")
plt.grid(True)
plt.show()
```

```
#s18q1 Q1. Write a R program to find the maximum and the minimum
value of a givenvector numbers <- c(12, 45, 7, 89,23, 56,34)

max_value <- max(numbers)

min_value <- min(numbers)

    cat("Maximum value:", max_value, "\n") cat("Minimum value:",
                          min_value, "\n")

#s18q2
'''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])'''
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error,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)
```

```
y_pred=model.predict(x)

mae=mean_absolute_error(y_pred,y)
mse=mean_squared_error(y_pred,y)
r2=r2_score(y_pred,y)

print("b0",model.intercept_)
print("b1",model.coef_)


print("Mean absolute error", mae) print("Mean Squared error",mse) print("r2
score", r2)
```

```
#s19q1 Q1. Write aR program to create a Dataframes which contain
details of 5 Studentsand display the #details
students <- data.frame(

Rollno Studname
c(1, 2, 3, 4, 5),

c("Raj", "TT", "Rohit", "Swa", "Jnvi"),
Address = c("Address1", "Address2", "Address3","Address4", "Address5"),
Marks = c(85, 90, 78, 92, 88)


print(students)
```

```
#s19q2
'''Q2. Write a python program to implement multiple Linear Regression model for a car dataset. Dataset can be downloaded from:
https://www.w3schools.com/python/python_ml_multiple_regression.as'''
import pandas as pd
from sklearn.linear_model import LinearRegression

data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/cars.csv")
x=data[ [ 'Volume','Weight']]
y=data[ [ 'C02']]

print(x)
print(y)
model=LinearRegression()

model.fit(x,y)

co2=model.predict([[2300,1200]])
print("Predicteed emmision fro sample input 2300 and 1200:")
print(co2)
```

```
#s20q1 Q1. Write a R program to create a data frame from four given vectors. vector1 <- c(1,2,3, 4, 5)
```

```
vector2 <- c("A", "B" "C", "D", "E")
vector3 <- c("Yes", "No". "Yes" "No", "Yes")
vector4 <- c(10.5, 20.2, 30.3, 40.1, 50.0)

data_frame <- data.frame(
Column1 = vector1,
Column2 = vector2,
Column3 = vector3,
Column4 = vector4


print(data_frame)
```

```
#s20q2
'''Q2. Write a python program to implement hierarchical
Agglomerative clusteringalgorithm. (Download Customer.csv dataset
from github.com)'' '
import pandas as pd
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import LabelEncoder
data=pd.read_csv("C:/Users/pruth_q9y7zoz/Downloads/customers.csv")
print(data)
le=LabelEncoder()
data['Gender']=le.fit_transform(data['Gender'])
x=data[['Gender', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

model=AgglomerativeClustering(n_clusters=5,metric='euclidean',
linkage='ward')

res=model.fit_predict(x) data['cluster']=res
```

```
print(data[['CustomerID', 'cluster']])#it shows which customer
belongs to which cluster
```