**data mining (10 marks)**

**1. Write a R program to add, multiply and divide two vectors of integertype. (Vector length**

**should be minimum 4**

**=>**

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

vector2<-c(4,5,6,7,8)

sum<-vector1+vector2

cat("sum of vector is:",sum,"\n")

mul<-vector1\*vector2

cat("multiplication",mul,"\n")

div<-vector1/vector2

cat("division",div,"\n")

**2. Write an R program to calculate the multiplication table using a function**

vector<-function(n,a)

for(i in 1:a)

{

result=n\*i

cat(paste(n,"X",i,"=",result,"\n"))

}

number<-5

a<-10

vector(number,a)

**3. Write a R program to reverse a number and also calculate the sum of digits of that number.**

num=123

sum=0

rev=0

while(num>0)

{

r=num%%10

sum=sum+r

rev=rev\*10+r

num=num%/%10

}

cat(paste("reverse num is:",rev))

cat(paste("sum of digit is:",sum))

4. **Write a R program to calculate the sum of two matrices of given size.**

addMatrices <- function(mat1, mat2) {

# Check if matrices have the same dimensions

if (dim(mat1) != dim(mat2)) {

stop("Matrices must have the same dimensions.")

}

result <- mat1 + mat2

return(result)

}

# Example matrices

matrix1 <- matrix(1:4, nrow = 2, ncol = 2)

matrix2 <- matrix(5:8, nrow = 2, ncol = 2)

resultMatrix <- addMatrices(matrix1, matrix2)

print(resultMatrix)

**5. Write a R program to concatenate two given factors**.

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

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

# Concatenate the factors

concatenated\_factors <- c(factor1, factor2)

print(concatenated\_factors)

**6. Write a R programto create a data frame using two given vectors and display the duplicate**

elements.

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

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

# Create a data frame

my\_data\_frame <- data.frame(Column1 = vector1, Column2 = vector2)

# Display the data frame

print("Data Frame:")

print(my\_data\_frame)

# Find and display duplicate elements

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

print("\nDuplicate Elements:")

print(duplicates)

7. **Write a R program to create a sequence of numbers from 20 to 50 and find the mean of**

**numbers from 20 to 60 and sum of numbers from 51 to** 91.

sequence\_20\_to\_50 <- seq(20, 50)

mean\_20\_to\_60 <- mean(seq(20, 60))

sum\_51\_to\_91 <- sum(seq(51, 91))

print(paste("Sequence from 20 to 50:", toString(sequence\_20\_to\_50)))

print(paste("Mean of numbers from 20 to 60:", mean\_20\_to\_60))

print(paste("Sum of numbers from 51 to 91:", sum\_51\_to\_91))

**8. Write a R program to get the first 10 Fibonacci number**

fibonacci <- function(n) {

if (n == 1) {

return(0)

} else if (n == 2) {

return(1)

} else {

return(fibonacci(n - 1) + fibonacci(n - 2))

}

}

fibonacci\_numbers <- sapply(1:10, fibonacci)

print(paste("First 10 Fibonacci numbers:", toString(fibonacci\_numbers)))

**9. Write an R program to create a Data frames which contain details of 5 employees and**

**display summary of the data.**

employee\_data <- data.frame(

Employee\_ID = c(1, 2, 3, 4, 5),

Name = c("John", "Alice", "Bob", "Eva", "Mike"),

Age = c(30, 25, 35, 28, 32),

Department = c("IT", "HR", "Finance", "Marketing", "Operations"),

Salary = c(60000, 50000, 70000, 55000, 75000)

)

print("Employee Data:")

print(employee\_data)

summary(employee\_data)

**10. Write a R program to find the maximum and the minimum value of a given vector**

my\_vector <- c(3, 8, 1, 5, 7)

max\_value <- max(my\_vector)

min\_value <- min(my\_vector)

cat("Maximum Value:", max\_value, "\n")

cat("Minimum Value:", min\_value,"\n")

**11. Write a R program to find all elements of a given list that are not in another given list**

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

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

A <- list("x", "y", "z")

B <- list("X", "Y", "Z", "x", "y", "z")

result <- setdiff(A, B)

print(result)

14. **Write a R program to create a Dataframes which contain details of 5employees and display**

**the details. Employee contain (empno,empname,gender,age,designation)**

employee\_data <- data.frame(

empno = c(101, 102, 103, 104, 105),

empname = c("John", "Alice", "Bob", "Eva", "Mike"),

gender = c("Male", "Female", "Male", "Female", "Male"),

age = c(28, 25, 32, 29, 35),

designation = c("Manager", "Engineer", "Analyst", "Manager", "Developer")

)

print(employee\_data)

**15.Draw a pie chart using R programming for the following data distribution:**

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

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

pie(digits\_distribution,labels =labels, main = "dice data Distribution", col = rainbow(length(labels)))

legend("topright", legend = labels, fill = rainbow(length(labels)))

**16. 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.**

employee\_list <- c("sahil", "aditi", "priya", "darshana")

cat(" Names of employees:\n")

print(employee\_list)

new\_employee <- "pooja"

employee\_list <- c(employee\_list, new\_employee)

cat("Names of employees after adding", new\_employee, ":\n")

print(employee\_list)

employee\_list <- employee\_list[-3]

cat(" Names of employees after removing the third element:\n")

print(employee\_list)

**17. Write a R program to add, multiply and divide two vectors of integer type.(vector length should**

**be minimum 4)**

same as question 1

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

years <- c(2001, 2002, 2003)

export\_data <- c(26, 32, 35)

import\_data <- c(35, 40, 50)

barplot\_heights <- rbind(export\_data, import\_data)

barplot(barplot\_heights, beside = TRUE, col = c("blue", "green"),

names.arg = years, legend.text = c("Export", "Import"),

main = "Export and Import Data Over Years",

xlab = "Year", ylab = "Value")

legend("topright", legend = c("Export", "Import"), fill = c("blue", "green"))

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

fibonacci <- function(n) {

if (n == 1) {

return(0)

} else if (n == 2) {

return(1)

} else {

return(fibonacci(n - 1) + fibonacci(n - 2))

}

}

fibonacci\_numbers <- sapply(1:20, fibonacci)

print(paste("First 10 Fibonacci numbers:", toString(fibonacci\_numbers)))

**20. Write a R program to find the maximum and the minimum value of a given vector**

**same as above question**

**21. Write aR program to create a Dataframes which contain details of 5 Studentsand display the**

**details. Students contain (Rollno,Studname,Address,Marks)**

student\_data <- data.frame(

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

Studname = c("Ankita", "sahil", "pradnya", "Darshana", "sita"),

Address = c("123 pune St", "456 nigdi St", "789 dehu St", "101 nigdi St", "202 akurdi St"),

Marks = c(85, 92, 78, 95, 88)

)

cat("Student Details:\n")

print(student\_data)

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

names <- c("Ankita", "rahul", "aditi", "sahil")

ages <- c(25, 30, 22, 28)

scores <- c(90, 85, 88, 92)

grades <- c("A", "B", "B+", "A-")

student\_data <- data.frame(

Name = names,

Age = ages,

Score = scores,

Grade = grades

)

cat("Student Data Frame:\n")

print(student\_data)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DM(20 marks)**

**1. Write a python program to implement hierarchical Agglomerativeclustering algorithm.**

**(Download Customer.csv dataset from github.com).**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.preprocessing import OneHotEncoder

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

features = data[['Milk', 'Fresh']]

encoder = OneHotEncoder(sparse=False)

encoded\_features = encoder.fit\_transform(features)

n\_clusters = 5

agglomerative\_cluster = AgglomerativeClustering(n\_clusters=n\_clusters, linkage='ward')

clusters = agglomerative\_cluster.fit\_predict(encoded\_features)

linkage\_matrix = linkage(encoded\_features, method='ward')

dendrogram(linkage\_matrix)

plt.show()

data['Cluster'] = clusters

print(data[['Frozen', 'Cluster']])

2. 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

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

url = "C:\\Users\\Shweta\\Downloads\\multiple\_Regression.csv"

df = pd.read\_csv("C:\\Users\\Shweta\\Downloads\\multiple\_Regression.csv")

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

y = df['CO2']

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)

predictions = model.predict(X\_test)

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

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

3. **Consider the following observations/data. And apply simple linear regression and find outestimated coefficients b1 and b1 Also analyse the performance of the model (Use sklearnpackage)**

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

**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\_squared\_error

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]

predictions = model.predict(x)

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

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

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

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

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

plt.plot(x, predictions, color='red', linewidth=2, label='Regression Line')

plt.xlabel('x')

plt.ylabel('y')

plt.legend()

plt.show()

**4. 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**

import pandas as pd

from sklearn.linear\_model import LinearRegression

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)

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

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

model = LinearRegression()

model.fit(X, y)

b0 = model.intercept\_

b1 = model.coef\_[0]

predictions = model.predict(X)

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

plt.plot(df['Interest\_Rate'], predictions, color='red', linewidth=2, label='Regression Line')

plt.xlabel('Interest Rate')

plt.ylabel('Stock Index Price')

plt.legend()

plt.show()

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

from sklearn.tree import DecisionTreeClassifier

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

# Load the dataset

data = pd.read\_csv('path\_to\_your\_diabetes\_dataset.csv') # Replace 'path\_to\_your\_diabetes\_dataset.csv' with your file path

# Display the first few rows to understand the data

print(data.head())

# Separate features and target variable

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

y = data['Outcome'] # Target variable

# Split the data into training and testing sets

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

clf = DecisionTreeClassifier()

# Train the classifier

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

# Predict on the test set

predictions = clf.predict(X\_test)

# Evaluate the model

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

print(f"Accuracy: {accuracy}")

# Other evaluation metrics

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, predictions))

print("\nClassification Report:")

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

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

​

# Read the data

data = pd.read\_csv("C:\\Users\\Shweta\\Downloads\\shows.csv")

​

# Prepare features and target variable

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

y = data['Label']

​

# Create and train the Decision Tree Classifier

model = DecisionTreeClassifier()

model.fit(X, y)

​

# Define the input for prediction

input\_data = {'Age': 40, 'Experience': 10, 'Ranking': 7}

​

# Convert input to DataFrame

input\_df = pd.DataFrame([input\_data])

​

# Predict the label for the input

predicted\_label = model.predict(input\_df)

​

print("Predicted label:", predicted\_label)

​

​

**7. 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**

import pandas as pd

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

# Load the Groceries dataset

df = pd.read\_csv("C:\\Users\\Shweta\\Downloads\\groceries - groceries.csv")

# Display the first few rows of the dataset

print("Sample of the Groceries dataset:")

print(df.head())

# Preprocess the data: Convert the dataset into a one-hot encoded DataFrame

oht = pd.get\_dummies(df, columns=['Petal Width Cm'])

# Apply Apriori algorithm to find frequent itemsets

frequent\_itemsets = apriori(oht, min\_support=0.02, use\_colnames=True)

# Generate association rules

rules = association\_rules(frequent\_itemsets, metric='lift', min\_threshold=1.0)

# Display frequent itemsets and association rules

print("\nFrequent Itemsets:")

print(frequent\_itemsets)

print("\nAssociation Rules:")

print(rules)

**8. 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://www.kaggle.com/spscientist/students-performance-inexams?**

**select=StudentsPerformance.csv)**

import pandas as pd

# Load the dataset

df = pd.read\_csv("C:\\Users\\Shweta\\DM\_Assignments\\Assignment\_4\\StudentsPerformance.csv")

# Display the shape of the dataset

print("Shape of the dataset:", df.shape)

# Display the top rows of the dataset with their columns

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

print(df.head())

# Display the number of rows randomly (e.g., first 5 rows)

num\_random\_rows = 5

print(f"\nRandom {num\_random\_rows} rows:")

print(df.sample(num\_random\_rows))

# Display the number of columns and names of the columns

num\_columns = len(df.columns)

column\_names = df.columns.tolist()

print(f"\nNumber of columns: {num\_columns}")

print("Column names:", column\_names)

**9)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**

**----->**

similar as Q2

**10)Write a python program to implement hierarchical clustering algorithm.**

**(Download Wholesale customers data dataset from github.com).**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

from sklearn.preprocessing import StandardScaler

from scipy.cluster.hierarchy import dendrogram, linkage

# Load the Wholesale Customers dataset

data = pd.read\_csv("C:\\Users\\Shweta\\DM\_Assignments\\Assignment\_6\\customer.csv")

# Select features for clustering

X = data[['Fresh', 'Milk']] # Replace 'Feature1', 'Feature2', ... with actual features

# Standardize the data

scaler = StandardScaler()

X\_std = scaler.fit\_transform(X)

# Perform hierarchical clustering

linkage\_matrix = linkage(X\_std, method='ward')

# Plot the dendrogram

dendrogram(linkage\_matrix)

plt.show()

# Fit Agglomerative Clustering

n\_clusters = 3 # You can choose the number of clusters

agg\_clustering = AgglomerativeClustering(n\_clusters=n\_clusters)

clusters = agg\_clustering.fit\_predict(X\_std)

# Print cluster assignments

print("Cluster Assignments:", clusters)

**11)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 mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

# Load the Iris dataset

data = pd.read\_csv("C:\\Users\\Shweta\\DM\_Assignments\\Assignment\_4\\Iris.csv")

# Drop any non-numeric columns

data = data.drop(["Species"], axis=1)

# Define a function to convert numeric values to binary (0 or 1)

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

# Apply the encoding function to the dataset

data = data.applymap(encode\_units)

# Use Apriori to find frequent item sets

frequent\_itemsets = apriori(data, min\_support=0.6, use\_colnames=True)

# Print the frequent item sets

print("Frequent Item Sets:")

print(frequent\_itemsets)

# Find association rules

association\_rules\_df = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1.0)

# Print the association rules

print("\nAssociation Rules:")

print(association\_rules\_df)

**12.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 numpy as np

import pandas as pd

from sklearn.datasets import load\_breast\_cancer

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

from sklearn.svm import SVC

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

# Load the breast cancer dataset

data = load\_breast\_cancer()

X = data.data

y = data.target

# Split the dataset into training and testing sets

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

# Create an SVM classifier

svm\_classifier = SVC(kernel='linear', C=1.0, random\_state=42)

# Fit the SVM model to the training data

svm\_classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = svm\_classifier.predict(X\_test)

# Calculate accuracy, precision, and recall

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

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

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

# Print the results

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

**13)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

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv("C:\\Users\\Shweta\\DM\_Assignments\\Assignment\_6\\CC GENERAL.csv")

# Remove non-essential columns and handle missing values (you may need to adjust this based on your data)

df = df.drop(['CUST\_ID'], axis=1)

df = df.fillna(method='ffill')

# 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 to find the optimal number of clusters

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, select the optimal number of clusters

optimal\_clusters = 3 # Adjust this based on the Elbow Method graph

# Apply K-means clustering with the chosen number of clusters

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

kmeans.fit(scaled\_data)

# Add cluster labels to the dataset

df['Cluster'] = kmeans.labels\_

# Now you can use the 'Cluster' column as a feature for prediction or analysis

# For example, you can print the counts of data points in each cluster

print(df['Cluster'].value\_counts())

# You can explore and analyze your dataset with the cluster labels added

print(df.head())

14.**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])**

------>similar as question 3

**15)Write a python program to implement hierarchical Agglomerative clusteringalgorithm.**

**(Download Customer.csv dataset from github.com).**

---->similar as question 1

**16.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)**

----->similar as question 5

**17.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.**

18**.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]**

--->similar as q3 and q14

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

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import make\_blobs

# Generate a synthetic dataset

n\_samples = 300

n\_features = 2

n\_clusters = 4

X, y = make\_blobs(n\_samples=n\_samples, n\_features=n\_features, centers=n\_clusters, random\_state=42)

# Apply K-means clustering

kmeans = KMeans(n\_clusters=n\_clusters)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

# Visualize the clustering results

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=0.9)

plt.title("K-means Clustering")

plt.show()

**20.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 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

import math

data = pd.read\_csv("C:\\Users\\Shweta\\DM\_Assignments\\Assignment\_5\\Student\_Marks.csv")

x = data['number\_courses'].values.reshape(-1,1)

y = data['time\_study'].values

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

regressor = LinearRegression()

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

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

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

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

rmse = math.sqrt(mse)

print("Mean Absolute Error:", mae)

print("Mean Squared Error:", mse)

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