**8.Write a java program to prepare a simulated dataset with unique instances**.

import java.io.\*;

class Creatingdataset

{

public static void main(String []arg)throws IOException

{

int i,j,m=0;//initalize variables

int t1=0;

BufferedReader b=new BufferedReader(new InputStreamReader(System.in));

//Java BufferedReader class is used to read the text from a character-based input stream.

//It can be used to read data line by line by readLine() method.

System.out.println("Enter the number of transaction :");

int n=Integer.parseInt(b.readLine());//here n is the number of transactions

System.out.println("items :1--Milk 2--Bread 3--Coffee 4--Juice 5--Cookies 6--Jam 7--Tea 8--Butter 9--Sugar 10--Water");

int item[][]=new int[n][10];//cresting array of 10 items.

for(i=0;i<n;i++)//loop generating for number of transactions

for(j=0;j<10;j++)//loop generating for items array

item[i][j]=0; //initailizing unique items with their frequency as 0.

String[] itemlist={"MILK","BREAD","COFFEE","JUICE","COOKIES","JAM","TEA","BUTTER","SUGAR","WATER"};

for(i=0;i<n;i++)

{ System.out.println("Transaction "+(i+1)+" :");//incrementing for each items in 'n' transactions.

for(j=0;j<10;j++)

{

//System.out.println(itemlist[j]);

System.out.println("Is Item "+itemlist[j]+" present in this transaction(1/0)? :");

//checking whether items from itemlistis present in transaction or not where 0- not present,1-present.

item[i][j]=Integer.parseInt(b.readLine());

//reading for each items from itemlist in n transaction.

}

}

for(i=0;i<n;i++){

System.out.print(“transaction ”+i+”:”);

for(j=0;j<10;j++){

System.out.print(item[i][j]+" ");

}

System.out.println();

}

}

}

**Output:**

Enter the number of transaction :

2

items :1--Milk 2--Bread 3--Coffee 4--Juice 5--Cookies 6--Jam 7--Tea 8--Butter 9--Sugar 10--Water

Transaction 1 :

Is Item MILK present in this transaction(1/0)? :

1

Is Item BREAD present in this transaction(1/0)? :

0

Is Item COFFEE present in this transaction(1/0)? :

1

Is Item JUICE present in this transaction(1/0)? :

1

Is Item COOKIES present in this transaction(1/0)? :

1

Is Item JAM present in this transaction(1/0)? :

1

Is Item TEA present in this transaction(1/0)? :

0

Is Item BUTTER present in this transaction(1/0)? :

1

Is Item SUGAR present in this transaction(1/0)? :

0

Is Item WATER present in this transaction(1/0)? :

1

Transaction 2 :

Is Item MILK present in this transaction(1/0)? :

1

Is Item BREAD present in this transaction(1/0)? :

0

Is Item COFFEE present in this transaction(1/0)? :

1

Is Item JUICE present in this transaction(1/0)? :

0

Is Item COOKIES present in this transaction(1/0)? :

1

Is Item JAM present in this transaction(1/0)? :

1

Is Item TEA present in this transaction(1/0)? :

1

Is Item BUTTER present in this transaction(1/0)? :

1

Is Item SUGAR present in this transaction(1/0)? :

0

Is Item WATER present in this transaction(1/0)? :

1

Transaction 1: 1 0 1 1 1 1 0 1 0 1

Transaction 2: 1 0 1 0 1 1 1 1 0 1

**9.write a python program to generate frequent item sets/association rules using apriori algorithm.**

source code:

#importating required libraries

#pip install apyori

import pandas as pd

from apyori import apriori

#Loading the dataset

#Sample data set can be downloaded from the below link

# https://drive.google.com/file/d/1y5DYn0dGoSbC22xowBq2d4po6h1JxcTQ/view?usp=sharing

# Datasets consist of different products given 7500 transactions over the course of a week at a French retail store

store\_data = pd.read\_csv('store\_data.csv', header=None)

#Looking at the rows and columns

print("rows and columes of dataset:",end="")

print(store\_data.shape)

#Convert Pandas Data Frame into a list of lists

records = []

for i in range(0, 7501):

records.append([str(store\_data.values[i,j]) for j in range(0, 20)])

#Build the Apriori model

#min\_support parameter-used to select the items with support values greater than the value specified by the parameter

#min\_confidence parameter filters those rules that have confidence greater than the confidence threshold specified by the parameter

#min\_lift parameter specifies the minimum lift value for the short listed rules

#min\_length parameter specifies the minimum number of items that you want in your rules

association\_rules = apriori(records, min\_support=0.0045, min\_confidence=0.2, min\_lift=3, min\_length=2)

association\_results = list(association\_rules)

#Printing a single rule

print("printing a single rule")

print(association\_results[0])

print("=====================================")

#Printing the rule, the support, the confidence, and lift for each rule

for item in association\_results:

# first index of the inner list

# Contains base item and add item

pair = item[0]

items = [x for x in pair]

print("Rule: " + items[0] + " -> " + items[1])

#second index of the inner list

print("Support: " + str(item[1]))

#third index of the list located at 0th

#of the third index of the inner list

print("Confidence: " + str(item[2][0][2]))

pfrint("Lift: " + str(item[2][0][3]))

print("=====================================")

**output:**

rows and columes of dataset:(7501, 20)

printing a single rule

RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'light cream'}), items\_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.84395061728395)])

=====================================

Rule: chicken -> light cream

Support: 0.004532728969470737

Confidence: 0.29059829059829057

Lift: 4.84395061728395

=====================================

Rule: escalope -> mushroom cream sauce

Support: 0.005732568990801226

Confidence: 0.3006993006993007

Lift: 3.790832696715049

=====================================

Rule: pasta -> escalope

Support: 0.005865884548726837

Confidence: 0.3728813559322034

Lift: 4.700811850163794

=====================================

Rule: ground beef -> herb & pepper

Support: 0.015997866951073192

Confidence: 0.3234501347708895

Lift: 3.2919938411349285

=====================================

Rule: ground beef -> tomato sauce

Support: 0.005332622317024397

Confidence: 0.3773584905660377

Lift: 3.840659481324083

=====================================

Rule: whole wheat pasta -> olive oil

Support: 0.007998933475536596

Confidence: 0.2714932126696833

Lift: 4.122410097642296

=====================================

Rule: pasta -> shrimp

Support: 0.005065991201173177

Confidence: 0.3220338983050847

Lift: 4.506672147735896

=====================================

Rule: chicken -> nan

Support: 0.004532728969470737

Confidence: 0.29059829059829057

Lift: 4.84395061728395

=====================================

Rule: chocolate -> frozen vegetables

Support: 0.005332622317024397

Confidence: 0.23255813953488375

Lift: 3.2545123221103784

=====================================

Rule: ground beef -> spaghetti

Support: 0.004799360085321957

Confidence: 0.5714285714285714

Lift: 3.2819951870487856

**10.write a program to calculate chi-square value using python. Report your observations.**

'''The chi2\_contingency() function of scipy.stats module takes as input, the contingency table in 2d array format. It returns a tuple containing test statistics, the p-value, degrees of freedom and expected table(the one we created from the calculated values) in that order.

Hence, we need to compare the obtained p-value with alpha value of 0.05.'''

from scipy.stats import chi2\_contingency

# defining the table

data = [[207, 282, 241], [234, 242, 232]]

stat, p, dof, expected = chi2\_contingency(data)

# interpret p-value

alpha = 0.05

print("p value is " + str(p))

if p <= alpha:

print('Dependent (reject H0)')

else:

print('Independent (H0 holds true)')

**Output :**

p value is 0.1031971404730939

Independent (H0 holds true)

**11.write a program of navie Bayesian classification using python programming language.**

source code:

#Importing the libraries

import numpy as np

import pandas as pd

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import LabelEncoder

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

from sklearn.metrics import accuracy\_score

#Loading the dataset

#https://www.kaggle.com/pranavpandey2511/tennis-weather

play\_tennis = pd.read\_csv("tennis.csv")

#LabelEncoder converts a categorical data into a number ranging from 0 to n-1, where n is the number of classes in the variable.

#Number is given in the alphabetical order.

#For example, in case of Outlook, there are 3 clasess – Overcast, Rain, Sunny. These are represented as 0,1,2 in alphabetical order

number = LabelEncoder()

play\_tennis['outlook'] = number.fit\_transform(play\_tennis['outlook'])

play\_tennis['temp'] = number.fit\_transform(play\_tennis['temp'])

play\_tennis['humidity'] = number.fit\_transform(play\_tennis['humidity'])

play\_tennis['windy'] = number.fit\_transform(play\_tennis['windy'])

play\_tennis['play'] = number.fit\_transform(play\_tennis['play'])

#Displaying the transformed data

print(play\_tennis)

features = ["outlook", "temp", "humidity", "windy"]

target = "play"

#Splliting the dataset into test and training data set

features\_train, features\_test, target\_train, target\_test = train\_test\_split(play\_tennis[features],play\_tennis[target],test\_size = 0.33,random\_state = 54)

#Creating the model which uses gaussian naviebayesiean

model = GaussianNB()

model.fit(features\_train, target\_train)

#make predictions on the test feature

pred = model.predict(features\_test)

#Accurecy score measuring

accuracy = accuracy\_score(target\_test, pred)

#predicting the outcome1 when weather is sunny, hot, high, weak

print( model.predict([[2,1,0,1]]))

#predicting the outcome1 when weather is rainy, mild, high, weak

print( model.predict([[1,2,0,1]]))

**output:**

outlook temp humidity windy play

0 sunny hot high False no

1 sunny hot high True no

2 overcast hot high False yes

3 rainy mild high False yes

4 rainy cool normal False yes

5 rainy cool normal True no

6 overcast cool normal True yes

7 sunny mild high False no

8 sunny cool normal False yes

9 rainy mild normal False yes

10 sunny mild normal True yes

11 overcast mild high True yes

12 overcast hot normal False yes

13 rainy mild high True no

Dataset after transformation

outlook temp humidity windy play

0 2 1 0 0 0

1 2 1 0 1 0

2 0 1 0 0 1

3 1 2 0 0 1

4 1 0 1 0 1

5 1 0 1 1 0

6 0 0 1 1 1

7 2 2 0 0 0

8 2 0 1 0 1

9 1 2 1 0 1

10 2 2 1 1 1

11 0 2 0 1 1

12 0 1 1 0 1

13 1 2 0 1 0

predicting for sunny, hot, high, weak

[0]

predicting for rainy, mild, high, weak

[1]

**12.Implement a java program to perform apriori algorithm.**

import java.io.\*;

class Apriorialgorithm

{

public static void main(String []arg)throws IOException

{

int i,j,m=0;//initalize variables

int t1=0;

BufferedReader b=new BufferedReader(new InputStreamReader(System.in));

//Java BufferedReader class is used to read the text from a character-based input stream.

//It can be used to read data line by line by readLine() method.

System.out.println("Enter the number of transaction :");

int n=Integer.parseInt(b.readLine());//here n is the number of transactions

System.out.println("items :1--Milk 2--Bread 3--Coffee 4--Juice 5--Cookies 6--Jam 7--Tea 8--Butter 9--Sugar 10--Water");

int item[][]=new int[n][10];//cresting array of 10 items.

for(i=0;i<n;i++)//loop generating for number of transactions

for(j=0;j<10;j++)//loop generating for items array

item[i][j]=0; //initailizing unique items with their frequency as 0.

String[] itemlist={"MILK","BREAD","COFFEE","JUICE","COOKIES","JAM","TEA","BUTTER","SUGAR","WATER"};

//getting 10 items into array called itemlist.

int nt[]=new int[10];

int q[]=new int[10];

for(i=0;i<n;i++)

{ System.out.println("Transaction "+(i+1)+" :");//incrementing for each items in 'n' transactions.

for(j=0;j<10;j++)

{

//System.out.println(itemlist[j]);

System.out.println("Is Item "+itemlist[j]+" present in this transaction(1/0)? :");

//checking whether items from itemlistis present in transaction or not where 0- not present,1-present.

item[i][j]=Integer.parseInt(b.readLine());

//reading for each items from itemlist in n transaction.

}

}

for(j=0;j<10;j++)

{ for(i=0;i<n;i++)

{if(item[i][j]==1)//checking whether atleast there would be multiple items repeated at each n transaction.

nt[j]=nt[j]+1;//if condition is satisfied then we increment for all n transaction of items.

}

System.out.println("Number of Item "+itemlist[j]+" :"+nt[j]);

//generating number of multiple items repeated at their transaction with frequency number.

}

for(j=0;j<10;j++)

{ if(((nt[j]/(float)n)\*100)>=50)//calculating items with their threshold values.

q[j]=1;//segregating present items left after removal of items which is below threshold into array

else

q[j]=0;//segregating not present items removed as items are below the threshold values

if(q[j]==1)

{t1++;//getting the count of repetitions of same items

System.out.println("Item "+itemlist[j]+" is selected ");

//generating particular item which is selected after threshold calculating.

}

}

for(j=0;j<10;j++)

{ for(i=0;i<n;i++)

{

if(q[j]==0)

{

item[i][j]=0;

}

}

}

int nt1[][]=new int[10][10];//creating array for 2-frequency itemset

for(j=0;j<10;j++)

{ for(m=j+1;m<10;m++)//generating unique items for 2-frequency itemlist

{ for(i=0;i<n;i++)

{ if(item[i][j]==1 &&item[i][m]==1)

//checking there would atleast 1 itemset in 1-frequency itemset and 2-frequency itemlist.

{ nt1[j][m]=nt1[j][m]+1;

//incrementing for each items with all other items in 2-frequency itemset

}

}

if(nt1[j][m]!=0)//if 2-frequency itemlist is present

System.out.println("Number of Items of "+itemlist[j]+"& "+itemlist[m]+" :"+nt1[j][m]);

//printing number of items of each items with other items with their frequency.

}

}

for(j=0;j<10;j++)

{ for(m=j+1;m<10;m++)

{

if(((nt1[j][m]/(float)n)\*100)>=50)

q[j]=1;

else

q[j]=0;

if(q[j]==1)

{

System.out.println("Item "+itemlist[j]+"& "+itemlist[m]+" is selected ");

}

}

}

}

}

**Output:**

Enter the number of transaction :

2

items :1--Milk 2--Bread 3--Coffee 4--Juice 5--Cookies 6--Jam 7--Tea 8--Butter 9--Sugar 10--Water

Transaction 1 :

Is Item MILK present in this transaction(1/0)? :

1

Is Item BREAD present in this transaction(1/0)? :

0

Is Item COFFEE present in this transaction(1/0)? :

1

Is Item JUICE present in this transaction(1/0)? :

0

Is Item COOKIES present in this transaction(1/0)? :

0

Is Item JAM present in this transaction(1/0)? :

0

Is Item TEA present in this transaction(1/0)? :

1

Is Item BUTTER present in this transaction(1/0)? :

1

Is Item SUGAR present in this transaction(1/0)? :

0

Is Item WATER present in this transaction(1/0)? :

0

Transaction 2 :

Is Item MILK present in this transaction(1/0)? :

1

Is Item BREAD present in this transaction(1/0)? :

1

Is Item COFFEE present in this transaction(1/0)? :

0

Is Item JUICE present in this transaction(1/0)? :

0

Is Item COOKIES present in this transaction(1/0)? :

1

Is Item JAM present in this transaction(1/0)? :

0

Is Item TEA present in this transaction(1/0)? :

0

Is Item BUTTER present in this transaction(1/0)? :

0

Is Item SUGAR present in this transaction(1/0)? :

1

Is Item WATER present in this transaction(1/0)? :

1

Number of Item MILK :2

Number of Item BREAD :1

Number of Item COFFEE :1

Number of Item JUICE :0

Number of Item COOKIES :1

Number of Item JAM :0

Number of Item TEA :1

Number of Item BUTTER :1

Number of Item SUGAR :1

Number of Item WATER :1

Item MILK is selected

Item BREAD is selected

Item COFFEE is selected

Item COOKIES is selected

Item TEA is selected

Item BUTTER is selected

Item SUGAR is selected

Item WATER is selected

Number of Items of MILK& BREAD :1

Number of Items of MILK& COFFEE :1

Number of Items of MILK& COOKIES :1

Number of Items of MILK& TEA :1

Number of Items of MILK& BUTTER :1

Number of Items of MILK& SUGAR :1

Number of Items of MILK& WATER :1

Number of Items of BREAD& COOKIES :1

Number of Items of BREAD& SUGAR :1

Number of Items of BREAD& WATER :1

Number of Items of COFFEE& TEA :1

Number of Items of COFFEE& BUTTER :1

Number of Items of COOKIES& SUGAR :1

Number of Items of COOKIES& WATER :1

Number of Items of TEA& BUTTER :1

Number of Items of SUGAR& WATER :1

Item MILK& BREAD is selected

Item MILK& COFFEE is selected

Item MILK& COOKIES is selected

Item MILK& TEA is selected

Item MILK& BUTTER is selected

Item MILK& SUGAR is selected

Item MILK& WATER is selected

Item BREAD& COOKIES is selected

Item BREAD& SUGAR is selected

Item BREAD& WATER is selected

Item COFFEE& TEA is selected

Item COFFEE& BUTTER is selected

Item COOKIES& SUGAR is selected

Item COOKIES& WATER is selected

Item TEA& BUTTER is selected

Item SUGAR& WATER is selected

**13.Write program to cluster your choice of data using simple k-means algorithm using JDK.**

class Kmeans

{

public static void main(String[] args) {

double data[]={2,4.111,-10,12.049,3.45,20.1,30,11};//,25,17,23}; // initial data

double noofclusters=3;

double centroid[][]=new double[][]{

{0,0,0},

{2,4.111,30}

};

System.out.println(getCentroid(data,noofclusters,centroid));

}

public static double[][] getCentroid(double data[],double noofclusters,double centroid[][]){

double distance[][]=new double[(int) noofclusters][data.length];

double cluster[]=new double[data.length];

double clusternodecount[]=new double[(int) noofclusters];

centroid[0]=centroid[1];

centroid[1]=new double[]{0,0,0};

System.out.println("========== Starting to get new centroid =========");

for(double i=0;i<noofclusters;i++){

for(double j=0;j<data.length;j++){

//System.out.println(distance[i][j]+"("+i+","+j+")="+data[j]+"("+j+")-"+centroid[0][i]+"="+(data[j]-centroid[0][i]));

distance[(int) i][(int) j]=Math.abs(data[(int) j]-centroid[0][(int) i]);

System.out.print(distance[(int) i][(int) j]+" ,");

//System.out.println("Centroid: "+centroid[0][i]);

}

System.out.println();

}

for(double j=0;j<data.length;j++){

double smallerDistance=0;

if(distance[0][(int) j]<distance[1][(int) j] && distance[0][(int) j]<distance[2][(int) j])

smallerDistance=0;

if(distance[1][(int) j]<distance[0][(int) j] && distance[1][(int) j]<distance[2][(int) j])

smallerDistance=1;

if(distance[2][(int) j]<distance[0][(int) j] && distance[2][(int) j]<distance[1][(int) j])

smallerDistance=2;//

centroid[1][(int) smallerDistance]=centroid[1][(int) smallerDistance]+data[(int) j];

clusternodecount[(int) smallerDistance]=clusternodecount[(int) smallerDistance]+1;

cluster[(int) j]=smallerDistance;

//System.out.println("Centerid at 1: "+centroid[1][smallerDistance]);

//System.out.print(cluster[j]+", ");

}

//for(double j=0;j<data.length;j++)

//System.out.println("c at out: "+cluster[j]);

System.out.println("======================================== ");

System.out.println("New clusters are ");

// cluster[]= { 0 1 0 1 0 2 2 1}

// data[]={2,4,-10,12,3,20,30,11};

for(double i=0;i<noofclusters;i++){

System.out.print("C"+(i+1)+": ");

for(double l=0;l<data.length;l++){

if(cluster[(int) l]==i)

System.out.print(data[(int) l]+" ,");

}

System.out.println();

}

System.out.println("======================================== ");

System.out.println("New centroid is ");

for(double j=0;j<noofclusters;j++){

centroid[1][(int) j]=centroid[1][(int) j]/clusternodecount[(int) j];

System.out.print(centroid[1][(int) j]+",");

}

System.out.println();

booleanisAchived=true;

for(double j=0;j<noofclusters;j++){

if(isAchived&& centroid[0][(int) j] == centroid[1][(int) j]){

isAchived=true;

continue;

}

isAchived=false;

}

if(!isAchived){

getCentroid(data,noofclusters,centroid);

}

if(isAchived){

System.out.println("======================================== ");

System.out.println(" Final Cluster is ");

for(double i=0;i<noofclusters;i++){

System.out.print("C"+(i+1)+":");

for(double j=0;j<data.length;j++){

if(cluster[(int) j]==i)

System.out.print(data[(int) j]+" ,");

}

System.out.println();

}

}

return centroid;

}

}

**Output:**

========== Starting to get new centroid =========

0.0 ,2.1109999999999998 ,12.0 ,10.049 ,1.4500000000000002 ,18.1 ,28.0 ,9.0 ,

2.1109999999999998 ,0.0 ,14.111 ,7.938 ,0.6609999999999996 ,15.989 ,25.889 ,6.889 ,

28.0 ,25.889 ,40.0 ,17.951 ,26.55 ,9.899999999999999 ,0.0 ,19.0 ,

========================================

New clusters are

C1.0: 2.0 ,-10.0 ,

C2.0: 4.111 ,12.049 ,3.45 ,11.0 ,

C3.0: 20.1 ,30.0 ,

========================================

New centroid is

-4.0,7.6525,25.05,

========== Starting to get new centroid =========

6.0 ,8.111 ,6.0 ,16.049 ,7.45 ,24.1 ,34.0 ,15.0 ,

5.6525 ,3.5415 ,17.6525 ,4.3965 ,4.2025 ,12.447500000000002 ,22.3475 ,3.3475 ,

23.05 ,20.939 ,35.05 ,13.001000000000001 ,21.6 ,4.949999999999999 ,4.949999999999999 ,14.05 ,

========================================

New clusters are

C1.0: -10.0 ,

C2.0: 2.0 ,4.111 ,12.049 ,3.45 ,11.0 ,

C3.0: 20.1 ,30.0 ,

========================================

New centroid is

-10.0,6.522,25.05,

========== Starting to get new centroid =========

12.0 ,14.111 ,0.0 ,22.049 ,13.45 ,30.1 ,40.0 ,21.0 ,

4.522 ,2.4110000000000005 ,16.522 ,5.526999999999999 ,3.072 ,13.578000000000001 ,23.478 ,4.478 ,

23.05 ,20.939 ,35.05 ,13.001000000000001 ,21.6 ,4.949999999999999 ,4.949999999999999 ,14.05 ,

========================================

New clusters are

C1.0: -10.0 ,

C2.0: 2.0 ,4.111 ,12.049 ,3.45 ,11.0 ,

C3.0: 20.1 ,30.0 ,

========================================

New centroid is

-10.0,6.522,25.05,

========================================

Final Cluster is

C1.0:-10.0 ,

C2.0:2.0 ,4.111 ,12.049 ,3.45 ,11.0 ,

C3.0:20.1 ,30.0 ,

[[D@63961c42

**14.Write program to cluster analysis using simple k-means algorithm python programming language.**

'''Exercise for k means tutorial

1)Use iris flower dataset from sklearn library and try to form clusters of flowers using petal width and length features. Drop other two features for simplicity.

2)Figure out if any preprocessing such as scaling would help here

3)Draw elbow plot and from that figure out optimal value of k'''

#importing the Libraries

from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

from sklearn.datasets import load\_iris

%matplotlib inline

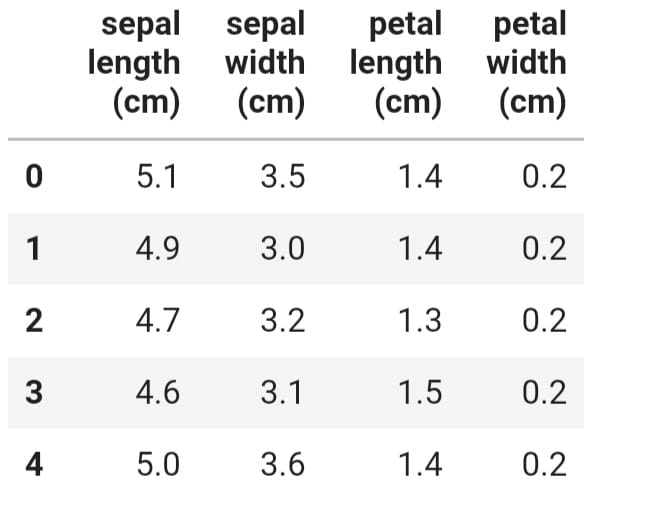
#Load the iris dataset into a pandas as an example

iris = load\_iris()

df = pd.DataFrame(iris.data,columns=iris.feature\_names)

#Printing top 5 records for viewing the dataset sample

print(df.head())

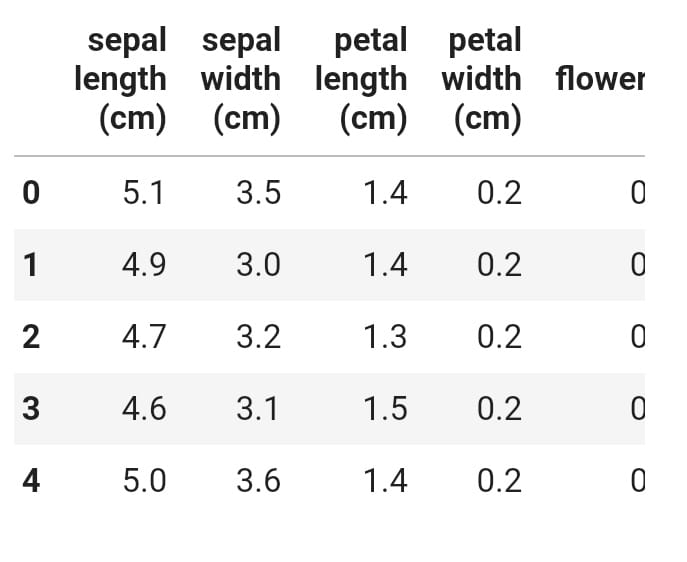


#Add the target data of the species/flower of iris to the dataframe

df['flower'] = iris.target

#Now Again see viewing the top5 records

print(df.head())

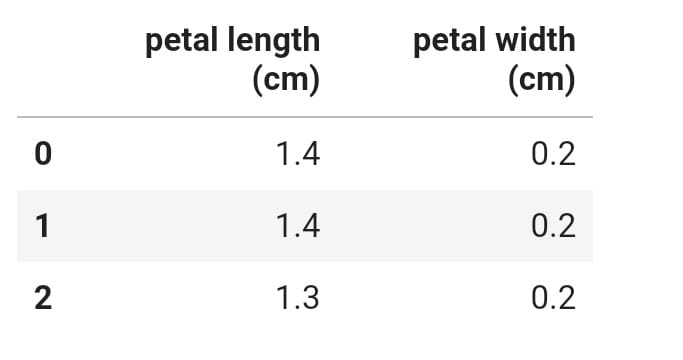


#As a sample we are only taking petal length and petal width from dataframe for better understanding of #k-means.so we drop sepal length sepal width and target

df.drop(['sepal length (cm)', 'sepal width (cm)', 'flower'],axis='columns',inplace=True)

#Now lets see the top 3 rows in dataframe

print(df.head())



#Lets apply KMeansalgortithm with 3 clusters

km = KMeans(n\_clusters=3)

yp = km.fit\_predict(df)

#Lets add one more feature cluster which represents the cluster where an instance belong

df['cluster']=yp

#Divide The DataFrame into 3 seperatedataframes where each represents a cluster

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

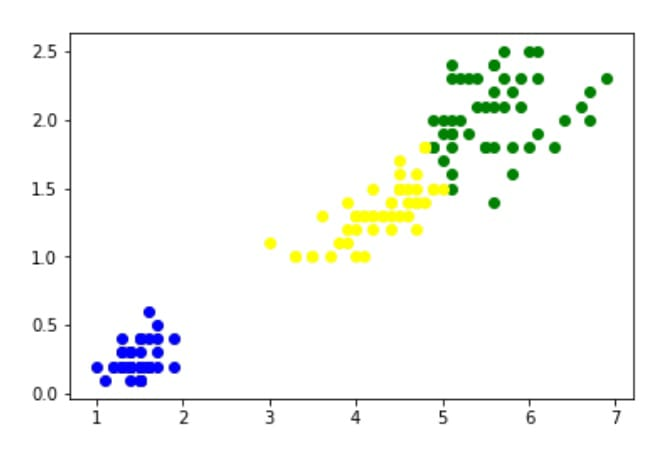
df3 = df[df.cluster==2]

#Lets plot those three clusters to visualize graphically

plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue')

plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='green')

plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='yellow')



#Now use Elbow plot to find optimal k value

sse = []#sse means mean squared error

k\_rng = range(1,10)

for k in k\_rng:

km = KMeans(n\_clusters=k)

km.fit(df)

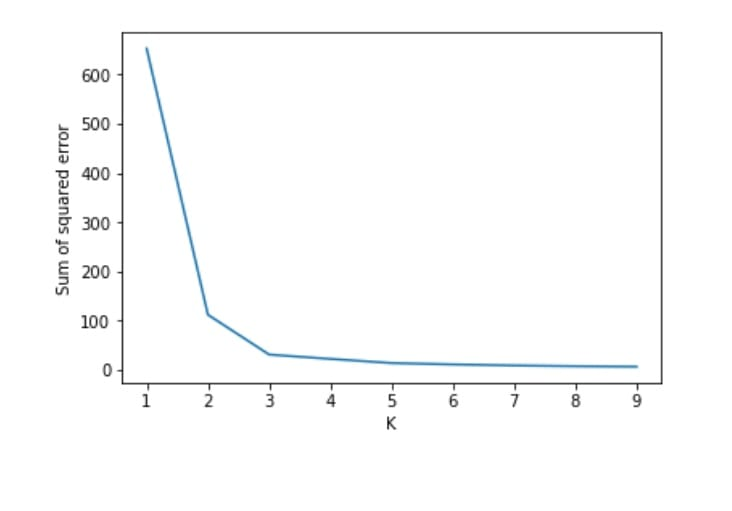
sse.append(km.inertia\_)

plt.xlabel('K')

plt.ylabel('Sum of squared error')

plt.plot(k\_rng,sse)

output:



**15.write a program to compute/display dissimilarity matrix(For your data set containing at least four instances with two attributes) using python.**

'''

The dissimilarity matrix (also called distance matrix) describes pairwise distinction between M objects. It is a square symmetrical MxM matrix with the (ij)th element equal to the value of a chosen measure of distinction between the (i)th and the (j)th object. The diagonal elements are either not considered or are usually equal to zero - i.e. the distinction between an object and itself is postulated as zero.

'''

# import important libraries

import numpy as np

from scipy.spatial import distance\_matrix

# Create the matrices

x = np.array([[1,2],[2,1],[2,2],[9,8]])

y = np.array([[5,0],[1,2],[2,0],[7,8]])

# Display the matrices

print("matrix x:\n", x)

print("matrix y:\n", y)

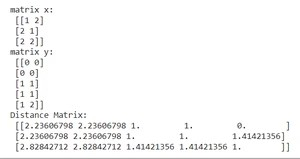
# compute the distance matrix

dist\_mat = distance\_matrix(x, y, p=2)

# display distance matrix

print("Distance Matrix:\n", dist\_mat)

output:



**16.Visualize the data set using the matplotlib in python (histogram, box plot, bar plot, pie chart, scatter plot)s**

Source code:

Scatter:

from sklearn import datasets

import matplotlib.pyplot as plt

import numpy as np

iris = datasets.load\_iris() # load dataset

X\_iris = iris.data[:, :2] # only take the first two features

Y\_iris = iris.target

n\_classes = 3

for i in range(n\_classes):

index = np.where(Y\_iris == i)

plt.scatter(X\_iris[index, 0], X\_iris[index, 1],

label=iris.target\_names[i])

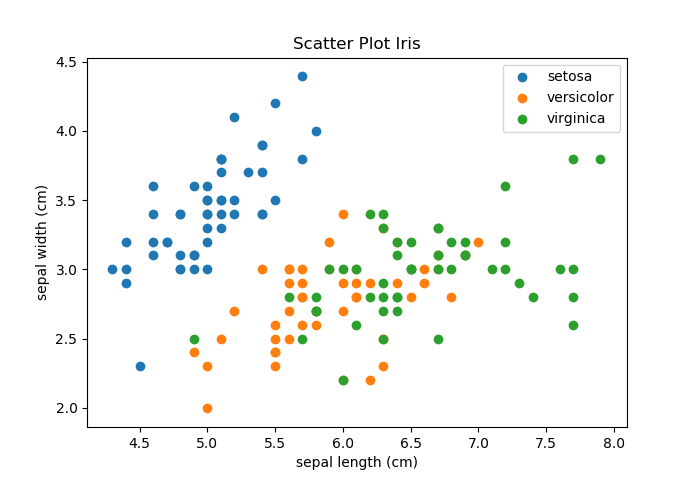
plt.legend()

plt.xlabel(iris.feature\_names[0])

plt.ylabel(iris.feature\_names[1])

plt.show()

Output:



Bar plot

from sklearn import datasets

import matplotlib.pyplot as plt

iris = datasets.load\_iris()

X\_iris = iris.data

Y\_iris = iris.target

average = X\_iris[Y\_iris == 0].mean(axis=0)

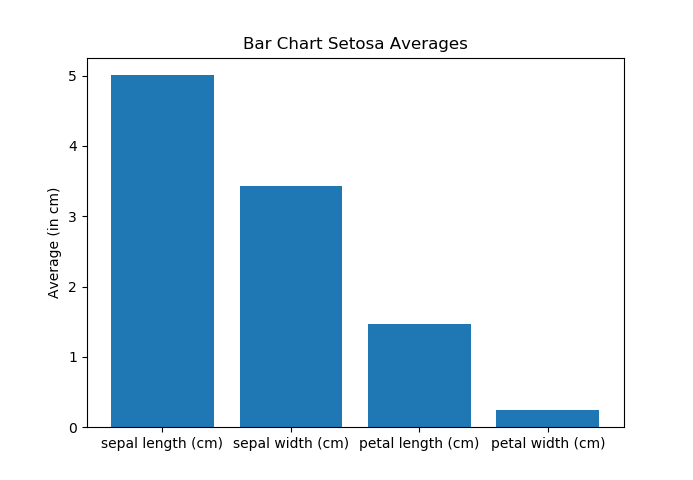
plt.bar(iris.feature\_names, average)

plt.title("Bar Chart Setosa Averages")

plt.ylabel("Average (in cm)")

plt.show()

output:



# Histograms

from sklearn import datasets

import matplotlib.pyplot as plt

bins = 20

iris = datasets.load\_iris()

X\_iris = iris.data

X\_sepal = X\_iris[:, 0]

plt.hist(X\_sepal, bins)

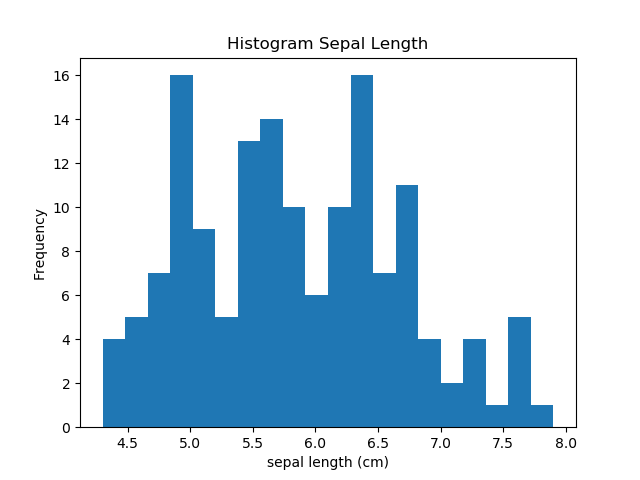
plt.title("Histogram Sepal Length")

plt.xlabel(iris.feature\_names[0])

plt.ylabel("Frequency")

plt.show()

output:



# Boxplots

from sklearn import datasets

import matplotlib.pyplot as plt

iris = datasets.load\_iris()

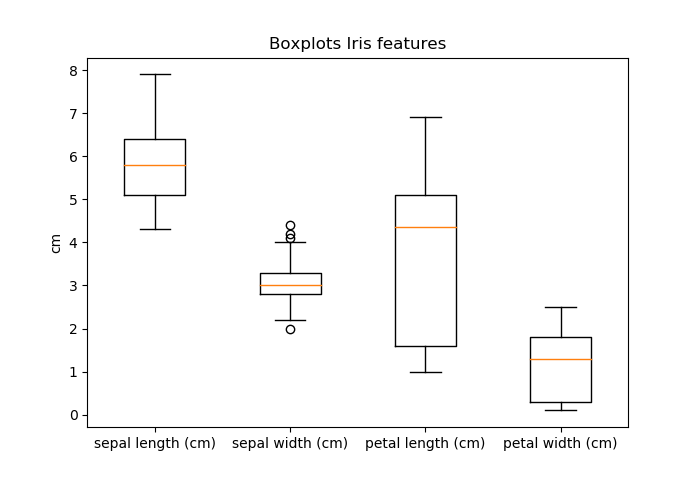
X\_iris = iris.data

plt.boxplot(X\_iris, labels=[iris.feature\_names[0], iris.feature\_names[1], iris.feature\_names[2], iris.feature\_names[3]])

plt.title("Boxplots Iris features")

plt.ylabel("cm")

plt.show()



Pie plot

import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read\_csv("iris.csv")

ax=plt.subplots(1,1,figsize=(10,8))

iris['Species'].value\_counts().plot.pie(explode=[0.1,0.1,0.1],autopct='%1.1f%%',shadow=True,figsize=(10,8))

plt.title("Iris Species %")

plt.show()

output:

