T.Y.B.C.A.(Science)

DSE II BCA 357- Laboratory (Data Mining) Workbook

Savitribai Phule Pune University

Answers

R Programming

Set A

1. Write a R program to add, multiply and divide two vectors of integer type. (vector length should be minimum 4)
   Solution :

```
> vector1 = seq(10,40 , length.out=4)
> vector2 = c(20, 10, 40, 40)

> print("Original Vectors:")
[1] "Original Vectors:"

> print(vector1)
[1] 10 20 30 40

> print(vector2)
[1] 20 10 40 40

> add= vector1+vector2
> cat("Sum of vector is ",add, "\n")
Sum of vector is  30 30 70 80

> sub_vector= vector1-vector2
> cat("Substraction of vector is ",sub_vector, "\n")
Substraction of vector is  -10 10 -10 0

> mul_vector= vector1 * vector2
> cat("Multiplication of vector is ",mul_vector, "\n")
Multiplication of vector is  200 200 1200 1600

> print("Division of two Vectors:")
[1] "Division of two Vectors:"
> div_vector = vector1 / vector2
> print(div_vector)
[1] 0.50 2.00 0.75 1.00
```

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

```
number <- as.integer(readline(prompt = "Please Enter a Number for Table: "))

Disp_table(number)
Disp_table=function(number)
{
```

```
 for( t in 1:10)
 {
    print ( paste ( number, '*', t, '=', number * t))
 }
}
```

Output

```
[1] "3 * 1 = 3"
[1] "3 * 2 = 6"
[1] "3 * 3 = 9"
[1] "3 * 4 = 12"
[1] "3 * 5 = 15"
[1] "3 * 6 = 16"
[1] "3 * 7 = 21"
[1] "3 * 8 = 24"
[1] "3 * 9 = 27"
[1] "3 * 10 = 30"
```

3. Write a R program to sort a list of strings in ascending and descending order.

```
> stud_name = c("Ram","Sham","Arjun","Raj")
> print(stud_name)
[1] "Ram"   "Sham"  "Arjun" "Raj"
> cat("Names in ascending order ",print(sort(stud_name)),"\n")
[1] "Arjun" "Raj"   "Ram"   "Sham"
Names in ascending order  Arjun Raj Ram Sham
> cat("Names in ascending order ",print(sort(stud_name,decreasing = TRUE)),"\n")
[1] "Sham"  "Ram"   "Raj"   "Arjun"
Names in ascending order  Sham Ram Raj Arjun
```

4. 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.

```
#create list using vector

> list_data <- list("Ram Sharma","Sham Varma","Raj Jadhav", "Ved Sharma")

#display list
> print(list_data)
[[1]]
[1] "Ram Sharma"

[[2]]
[1] "Sham Varma"
```

```
[[3]]
[1] "Raj Jadhav"

[[4]]
[1] "Ved Sharma"

#create new employee
new_Emp <-"Kavya Anjali"
#Add new employee at the end
list_data <-append(list_data,new_Emp)

print(list_data)
[[1]]
[1] "Ram Sharma"

[[2]]
[1] "Sham Varma"

[[3]]
[1] "Raj Jadhav"

[[4]]
[1] "Ved Sharma"

[[5]]
[1] "Kavya Anjali"



#remove 3 employee
list_data[3] <- NULL
print(list_data)

[[1]]
[1] "Ram Sharma"

[[2]]
[1] "Sham Varma"

[[3]]
[1] "Ved Sharma"

[[4]]
[1] "Kavya Anjali"
```

Set B

T.Y.B.C.A.(Science)

DSE II BCA 357- Laboratory (Data Mining) Workbook

Savitribai Phule Pune University

Answers

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

```
Reverse_Sum = function(n)
{

 sum=0
 rev=0

 while(n>0)
 {
  r = n%%10
  sum= sum+r;
  rev=rev*10+r
  n=n%/%10  # %/% is used for integer division
 }
 print(paste("Sum of digit : ",sum))
 print(paste("Reverse of number : ",rev))
}
n = as.integer(readline(prompt = "Enter a number :"))
Reverse_Sum(n)
```

Output

```
Enter a number :123
[1] "Sum of digit :  6"
[1] "Reverse of number :  321"
```

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

```
> # R program to add two matrices
>
> # Creating 1st Matrix
> A = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
>
> # Creating 2nd Matrix
> B = matrix(c(7, 8, 9, 10, 11, 12), nrow = 2, ncol = 3)
>
> # Getting number of rows and columns
> num_of_rows = nrow(A)
> num_of_cols = ncol(A)
>
> # Creating matrix to store results
> add = matrix(, nrow = num_of_rows, ncol = num_of_cols)
>
> # Printing Original matrices
> print(A)
```

```
   [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> print(B)
   [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
>
> # Calculating diff of matrices
> for(row in 1:num_of_rows)
+ {
+   for(col in 1:num_of_cols)
+   {
+     add[row, col] <- A[row, col] + B[row, col]
+   }
+ }
>
> # Printing resultant matrix
> print(add)
   [,1] [,2] [,3]
[1,]    8   12   16
[2,]   10   14   18
```

3. Write a R program to concatenate two given factors.

```
> fac1 <- factor(letters[1:3])
> print ("Factor1 : ")
[1] "Factor1 : "
> print (fac1)
[1] a b c
Levels: a b c
> sapply(fac1,class)
[1] "factor" "factor" "factor"
>
> fac2 <- factor(c(1:4))
> print ("Factor2 : ")
[1] "Factor2 : "
> print (fac2)
[1] 1 2 3 4
Levels: 1 2 3 4
> sapply(fac2,class)
[1] "factor" "factor" "factor" "factor"
>
> # extracting levels of factor1
```

```
> level1 <- levels(fac1)[fac1]
>
> # extracting levels of factor2
> level2 <- levels(fac2)[fac2]
>
> # combine into one factor
> combined <- factor(c( level1,level2 ))
> print ("Combined Factor : ")
[1] "Combined Factor : "
> print (combined)
[1] a b c 1 2 3 4
Levels: 1 2 3 4 a b c
>
> sapply(combined,class)
[1] "factor" "factor" "factor" "factor" "factor" "factor" "factor"
>
```

4. Write a R program to create a data frame using two given vectors and display the duplicate elements

```
> companies <- data.frame(Shares = c("TCS", "Reliance", "HDFC Bank", "Infosys",
 "Reliance"),
+                 Price = c(3200, 1900, 1500, 2200, 1900))
> companies
    Shares Price
1      TCS  3200
2  Reliance  1900
3 HDFC Bank  1500
4    Infosys 2200
5  Reliance  1900
> cat("After removing Duplicates ", "\n")
After removing Duplicates
> companies[duplicated(companies),]
   Shares Price
5 Reliance  1900
```

Set C

1. Write a R program to perform the following:
   a. Display all rows of the data set having weight greater than 120.

```
> women
  height weight
```

```
1    58   115
2    59   117
3    60   120
4    61   123
5    62   126
6    63   129
7    64   132
8    65   135
9    66   139
10   67   142
11   68   146
12   69   150
13   70   154
14   71   159
15   72   164
> result <- women[women$weight >120,]
> result
  height weight
4    61   123
5    62   126
6    63   129
7    64   132
8    65   135
9    66   139
10   67   142
11   68   146
12   69   150
13   70   154
14   71   159
15   72   164
```

b. Display all rows of data set in ascending order of weight.
(Use inbuilt data set woman)

```
> data=women
> sorted_data=data[order(data$weight),]
> sorted_data
  height weight
1    58   115
2    59   117
3    60   120
4    61   123
5    62   126
6    63   129
7    64   132
8    65   135
9    66   139
```

```
10   67   142
11   68   146
12   69   150
13   70   154
14   71   159
15   72   164
>
```

2. Write a R program to perform the following:
   a. Display all the cars having mpg more than 20.

```
> data=mtcars
> result <- data[data$mpg>20,]
> result
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

   b. Subset the data set by mpg column for values greater than 15.0

```
subset(data,data$mpg>15.0)
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant        18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE     16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL     17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC    15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
```

```
Fiat 128         32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic      30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla   33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona    21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin      15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9        27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2    26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L   15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino     19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Volvo 142E       21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

c. Display all cars having four gears.
(Use inbuilt data set mtcar)

```
> data[data$gear==4,]
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
>
```

3. Write a R Program to perform the following:
   a. Create a Scattered plot to compare wind speed and temperature.

```
> data=airquality
> data <- na.omit(data)

plot(data$Wind,data$Temp,main="Wind Speed vs Temperature",xlab="Wind Speed",ylab="Te
mperature",xlim=c(2,20),ylim=c(50,100),axes =TRUE)
```

   b. Create a Scattered plot to show the relationship between ozone and wind values by giving appropriate values to colour argument.

```
mlev <- levels(with(airquality, as.factor(Month)))
# This command extracts the level values but the mlev is of class character

> data=airquality
> data <- na.omit(data)


with(data,plot(Ozone ~ Wind,pch=mlev,col=mlev))

#We have seen that for the plot() function the color option is called col. For
the #shape option it is called pch which stands for print character.
```

c. Create a Bar plot to show the ozone level for all the days having temperature > 70.
(Use inbuilt dataset airquality)

```
> data<-data[data$Temp >70,]
> data <-na.omit(data)
> barplot(height=data$Ozone,main="ozone level for all the days having temper
ature > 70", xlab="Temperature", ylab="Ozone", names.arg = data$Temp,borde
r = "dark blue", col="pink")
```

Data Pre-Processing

1) Write a python program to convert Categorical values in numeric format for a given Dataset (https://codefires.com/how-convert-categorical-data-numerical-data-python/)

```
import pandas as pd
info = {
   'Gender' : ['Male', 'Female', 'Female', 'Male', 'Female', 'Female'],
   'Position' : ['Head', 'Asst.Prof.', 'Associate Prof.', 'Asst.Prof.', 'Head', 'Asst.Prof.']
}
df = pd.DataFrame(info)
print(df)
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

gender_encoded = le.fit_transform(df['Gender'])
encoded_position = le.fit_transform(df['Position'])


df['Encoded_Gender'] = gender_encoded
df['Encoded_Position'] = encoded_position
print(df)
```

```
In [12]: import pandas as pd
info = {
    'Gender' : ['Male', 'Female', 'Female', 'Male', 'Female', 'Female'],
    'Position' : ['Head', 'Asst.Prof.', 'Associate Prof.', 'Asst.Prof.', 'Head', 'Asst.Prof.']
}
df = pd.DataFrame(info)
print(df)
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

gender_encoded = le.fit_transform(df['Gender'])
encoded_position = le.fit_transform(df['Position'])


df['Encoded_Gender'] = gender_encoded
df['Encoded_Position'] = encoded_position
print(df)
```

```
   Gender         Position
0    Male             Head
1  Female       Asst.Prof.
2  Female  Associate Prof.
3    Male       Asst.Prof.
4  Female             Head
5  Female       Asst.Prof.
   Gender         Position  Encoded_Gender  Encoded_Position
0    Male             Head               1                 2
1  Female       Asst.Prof.               0                 1
2  Female  Associate Prof.               0                 0
3    Male       Asst.Prof.               1                 1
4  Female             Head               0                 2
5  Female       Asst.Prof.               0                 1
```

Using hot encoder
from sklearn.preprocessing import OneHotEncoder

gender_encoded = le.fit_transform(df['Gender'])
gender_encoded = gender_encoded.reshape(len(gender_encoded), -1)
one = OneHotEncoder(sparse=False)

print(one.fit_transform(gender_encoded))

```
: from sklearn.preprocessing import OneHotEncoder

gender_encoded = le.fit_transform(df['Gender'])
gender_encoded = gender_encoded.reshape(len(gender_encoded), -1)
one = OneHotEncoder(sparse=False)

print(one.fit_transform(gender_encoded))
```

```
[[0. 1.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [1. 0.]]
```

2) Write a python program to rescale the data between 0 and 1. (use inbuilt dataset) (IRIS DATA SET) https://machinelearningmastery.com/rescaling-data-for-machine-learning-in-python-with-scikit-learn/

1) Normalisation

# Normalize the data attributes for the Iris dataset.
from sklearn.datasets import load_iris
from sklearn import preprocessing
# load the iris dataset
iris = load_iris()

print(iris.data.shape)
# separate the data from the target attributes
X = iris.data
y = iris.target
# normalize the data attributes
normalized_X = preprocessing.normalize(X)

```
# Normalize the data attributes for the Iris dataset.
from sklearn.datasets import load_iris
from sklearn import preprocessing
# load the iris dataset
iris = load_iris()
print(iris.data.shape)
# separate the data from the target attributes
X = iris.data
y = iris.target
# normalize the data attributes
normalized_X = preprocessing.normalize(X)
```

```
(150, 4)
```

2) Data Standardization

# Standardize the data attributes for the Iris dataset.
from sklearn.datasets import load_iris
from sklearn import preprocessing
# load the Iris dataset
iris = load_iris()
print(iris.data.shape)
# separate the data and target attributes
X = iris.data
y = iris.target
# standardize the data attributes
standardized_X = preprocessing.scale(X)

```
# Standardize the data attributes for the Iris dataset.
from sklearn.datasets import load_iris
from sklearn import preprocessing
# load the Iris dataset
iris = load_iris()
print(iris.data.shape)
# separate the data and target attributes
X = iris.data
y = iris.target
# standardize the data attributes
standardized_X = preprocessing.scale(X)
```

```
(150, 4)
```

3) Write a python program to splitting the dataset into training and testing set

1) Using pandas

```
import pandas as pd
from sklearn.datasets import load_iris

iris_data = load_iris()
df = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
print(df)
training_data = df.sample(frac=0.8, random_state=25)
testing_data = df.drop(training_data.index)

print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

```
import pandas as pd
from sklearn.datasets import load_iris

iris_data = load_iris()
df = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
print(df)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
```

```
training_data = df.sample(frac=0.8, random_state=25)
testing_data = df.drop(training_data.index)

print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

```
No. of training examples: 120
No. of testing examples: 30
```

2) Using scikit-learn

```
from sklearn.model_selection import train_test_split

training_data, testing_data = train_test_split(df, test_size=0.2,
random_state=25)

print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

```
from sklearn.model_selection import train_test_split

training_data, testing_data = train_test_split(df, test_size=0.2, random_state=25)

print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

```
No. of training examples: 120
No. of testing examples: 30
```

1. Write a python program to find all null values in a given data set and remove them.

```
import pandas as pd
dataset = pd.read_csv('city_day.csv')
dataset
dataset.isnull()
dataset.isnull().head(10)
dataset.isnull().sum()
dataset.isnull().head().sum()
modifieddataset=dataset.fillna(" ")
modifieddataset.isnull().sum()
```

dataset=dataset.dropna()

2. Write a python program the Categorical values in numeric format for a given dataset.

```
import numpy as np
import pandas as pd

dataset = pd.read_csv('Data2.csv')
dataset
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
dataset['outlook'] = le.fit_transform(dataset.outlook)
dataset['temp'] = le.fit_transform(dataset.temp)
dataset['humidity'] = le.fit_transform(dataset.humidity)
dataset['playgolf'] = le.fit_transform(dataset.playgolf)
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,5].values
from sklearn.preprocessing import StandardScaler
st_x = StandardScaler()
x1 = st_x.fit_transform(x)
print(x1)
```

3. Write a python program to splitting the dataset into training and testing set.

```
import numpy as np
import pandas as pd
dataset = pd.read_csv("play_tennis.csv")
dataset
from sklearn import preprocessing
```

```
le = preprocessing.LabelEncoder()
outlook_encoded = le.fit_transform(dataset.outlook)
print(outlook_encoded)
temp_encoded = le.fit_transform(dataset.temp)
print(temp_encoded)
humidity_encoded = le.fit_transform(dataset.humidity)
wind_encoded = le.fit_transform(dataset.wind)
play_encoded = le.fit_transform(dataset.play)
dataset['outlook'] = le.fit_transform(dataset.outlook)
dataset['temp'] = le.fit_transform(dataset.temp)
dataset['humidity'] = le.fit_transform(dataset.humidity)
dataset['wind'] = le.fit_transform(dataset.wind)
dataset['play'] = le.fit_transform(dataset.play)
x=dataset.iloc[:,:-1].values
print(x)
y=dataset.iloc[:,4].values
print(y)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
print(x_train)
print(x_test)
```

4. Write a python program to implement complete data pre-processing in a given data set.(missing value, encoding categorical value, Splitting the dataset into the training and test sets and feature scaling.

5. Write a python program to Perform Classification using Decision Tree algorithm.

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
tree.plot_tree(clf)
X_pred = clf.predict(X)
X_pred == y
```

Numpy :

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Operations using NumPy

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Example 1

```
import numpy as np
a = np.array([1,2,3])
print a
```

The output is as follows –

```
[1, 2, 3]
```

Example 2

```
# more than one dimensions
import numpy as np
a = np.array([[1, 2], [3, 4]])
print a
```

The output is as follows −

```
[[1, 2]
 [3, 4]]
```

NumPy – Data Types

bool_
Boolean (True or False) stored as a byte
int_
Default integer type (same as C long; normally either int64 or int32)
intc
Identical to C int (normally int32 or int64)
intp
An integer used for indexing (same as C ssize_t; normally either int32 or int64)
int8
Byte (-128 to 127)
int16
Integer (-32768 to 32767)
float_
Shorthand for float64
float64
Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
float64
Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_
Shorthand for complex128
complex64
Complex number, represented by two 32-bit floats (real and imaginary components)

Example 1

```
# using array-scalar type
import numpy as np
dt = np.dtype(np.int32)
print dt
```

The output is as follows −

```
int32
```

pandas:

Pandas is built on top of two core Python libraries—matplotlib for data visualization and NumPy for mathematical operations. Pandas acts as a wrapper over these libraries, allowing you to access many of matplotlib's and NumPy's methods with less code.

Pandas features

Time series analysis

- Time Series / Date functionality
- Times series analysis with pandas
- Timeseries with pandas

T.Y.B.C.A.(Science)
DSE II BCA 357- Laboratory (Data Mining) Workbook
Savitribai Phule Pune University
Answers

split-apply-combine

Split-apply-combine is a common strategy used during analysis to summarize data—you split data into logical subgroups, apply some function to each subgroup, and stick the results back together again. In pandas, this is accomplished using the `groupby()` function and whatever functions you want to apply to the subgroups.

- Group By: split-apply-combine
- Summarizing Data in Python with Pandas
- Using Pandas: Split-Apply-Combine

Data visualization

- Visualization
- Simple Graphing with IPython and Pandas
- Beautiful Plots With Pandas and Matplotlib

Pivot tables

- Reshaping and Pivot Tables
- Pandas Pivot Table Explained
- Pivot Tables in Python

Working with missing data

- Working with missing data
- Handling missing data

Common features

Creating Objects
Viewing Data
Selection
Manipulating Data
Grouping Data
Merging, Joining and Concatenating
Working with Date and Time
Working With Text Data
Working with CSV and Excel files
Operations
Visualization
Applications and Projects
Miscellaneous

sklearn.preprocessing

The `sklearn.preprocessing` package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

sklearn.preprocessing.LabelEncoder

Encode target labels with value between 0 and n_classes-1.
This transformer should be used to encode target values, *i.e.* `y`, and not the input `X`.

Attributes

lasses_*ndarray of shape (n_classes,)*
Holds the label for each class.

Methods

`fit`(y)	Fit label encoder.
`fit_transform`(y)	Fit label encoder and return encoded labels.

Answers Prepared By: Lab Book Team

get_params([deep])	Get parameters for this estimator.
inverse_transform(y)	Transform labels back to original encoding.
set_params(**params)	Set the parameters of this estimator.
transform(y)	Transform labels to normalized encoding.

fit(*y*)

Fit label encoder.

Parameters
y*array-like of shape (n_samples,)*
Target values.
Returns
self*returns an instance of self.*
Fitted label encoder.

fit_transform(*y*)

Fit label encoder and return encoded labels.

Parameters
y*array-like of shape (n_samples,)*
Target values.
Returns
y*array-like of shape (n_samples,)*
Encoded labels.

transform(*y*)

Transform labels to normalized encoding.

Parameters
y*array-like of shape (n_samples,)*
Target values.
Returns
y*array-like of shape (n_samples,)*
Labels as normalized encodings.

pandas.DataFrame.iloc

Differences between `loc` and `iloc`

The main distinction between `loc` and `iloc` is:

- `loc` is label-based, which means that you have to specify rows and columns based on their row and column **labels**.
- `iloc` is integer position-based, so you have to specify rows and columns by their **integer position values** (0-based integer position).

Syntax

```
loc[row_label, column_label]
iloc[row_position, column_position]
```

With `loc`, we can pass the row label `'Fri'` and the column label `'Temperature'`.

```
# To get Friday's temperature
>>> df.loc['Fri', 'Temperature']10.51
```

The equivalent `iloc` statement should take the row number `4` and the column number `1` .

```
# The equivalent `iloc` statement
>>> df.iloc[4, 1]10.51
```

We can also use `:` to return all data. For example, to get all rows:

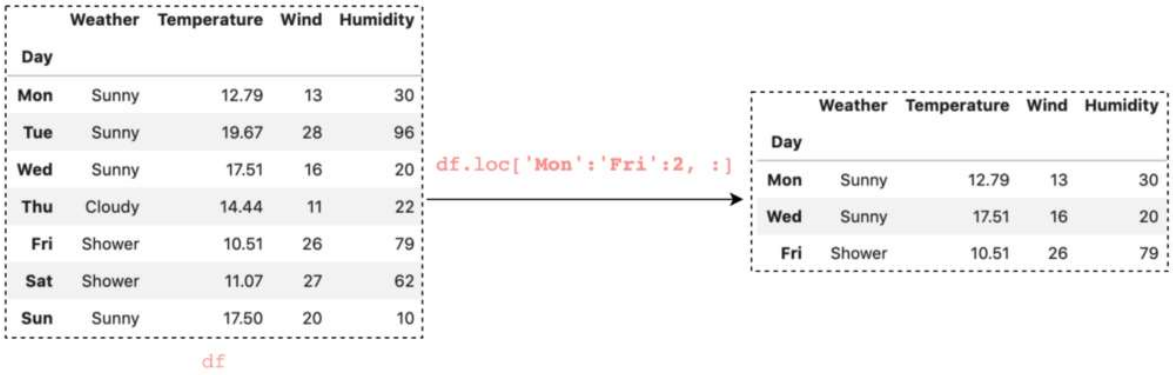
```
# To get all rows
>>> df.loc[:, 'Temperature']Day
Mon    12.79
Tue    19.67
Wed    17.51
Thu    14.44
Fri    10.51
Sat    11.07
Sun    17.50
Name: Temperature, dtype: float64# The equivalent `iloc` statement
>>> df.iloc[:, 1]
```

And to get all columns:

```
# To get all columns
>>> df.loc['Fri', :]Weather        Shower
Temperature     10.51
Wind               26
Humidity           79
Name: Fri, dtype: object# The equivalent `iloc` statement
>>> df.iloc[4, :]
```

We can use the syntax `A:B:S` to select data from label **A** to label **B** with step size **S** (Both **A** and **B** are included):

```
# Slicing with step
df.loc['Mon':'Fri':2 , :]
```



Day	Weather	Temperature	Wind	Humidity
Mon	Sunny	12.79	13	30
Tue	Sunny	19.67	28	96
Wed	Sunny	17.51	16	20
Thu	Cloudy	14.44	11	22
Fri	Shower	10.51	26	79
Sat	Shower	11.07	27	62
Sun	Sunny	17.50	20	10

df

Day	Weather	Temperature	Wind	Humidity
Mon	Sunny	12.79	13	30
Wed	Sunny	17.51	16	20
Fri	Shower	10.51	26	79

`iloc` with slice

With `iloc`, we can also use the syntax `n:m` to select data from position **n** (included) to position **m** (excluded). However, the main difference here is that the endpoint (**m**) is excluded from the `iloc` result.
For example, selecting columns from position 0 up to 3 (excluded):

```
df.iloc[[1, 2], 0 : 3]
```

T.Y.B.C.A.(Science)

DSE II BCA 357- Laboratory (Data Mining) Workbook

Savitribai Phule Pune University

Answers

Day	Weather	Temperature	Wind	Humidity
Mon	Sunny	12.79	13	30
Tue	Sunny	19.67	28	96
Wed	Sunny	17.51	16	20
Thu	Cloudy	14.44	11	22
Fri	Shower	10.51	26	79
Sat	Shower	11.07	27	62
Sun	Sunny	17.50	20	10

df

df.iloc[[1, 2], 0 : 3]

Day	Weather	Temperature	Wind
Tue	Sunny	19.67	28
Wed	Sunny	17.51	16

In all Estimators:

`model.fit()` : fit training data. For supervised learning applications, this accepts two arguments: the data `X` and the labels `y` (e.g. `model.fit(X, y)`). For unsupervised learning applications, this accepts only a single argument, the data `X` (e.g. `model.fit(X)`).

In supervised estimators:

`model.predict()` : given a trained model, predict the label of a new set of data. This method accepts one argument, the new data `X_new` (e.g. `model.predict(X_new)`), and returns the learned label for each object in the array.

`model.predict_proba()` : For classification problems, some estimators also provide this method, which returns the probability that a new observation has each categorical label. In this case, the label with the highest probability is returned by `model.predict()`.

`model.score()` : for classification or regression problems, most (all?) estimators implement a score method. Scores are between 0 and 1, with a larger score indicating a better fit.

In unsupervised estimators:

`model.transform()` : given an unsupervised model, transform new data into the new basis. This also accepts one argument `X_new`, and returns the new representation of the data based on the unsupervised model.

`model.fit_transform()` : some estimators implement this method, which more efficiently performs a fit and a transform on the same input data.

Classification:

SET A)

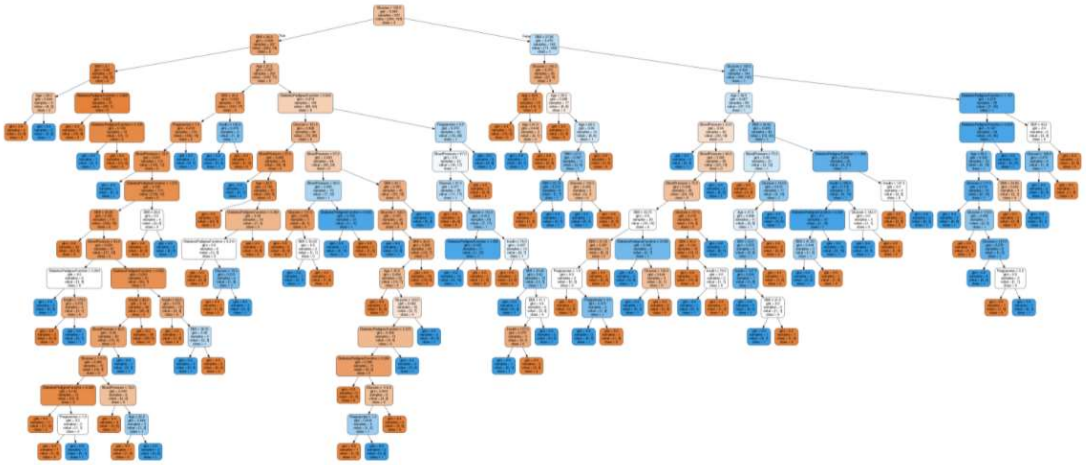
1) write a Python program build Decision Tree Classifier using Scikit-learn package for diabetes data set (download database from https://www.kaggle.com/uciml/pima-indians-diabetes-database)

```
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("C:\\Users\\Asus\\Desktop\\DMDWLAb Book Material\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))
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(classifier, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names =
feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```



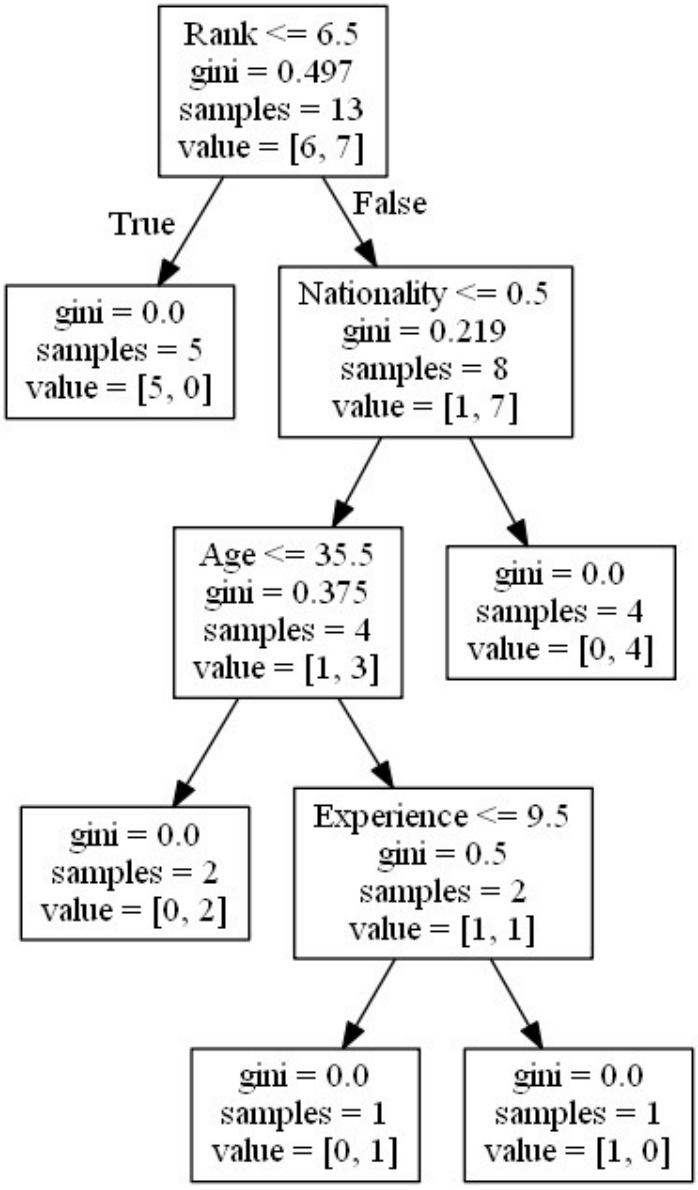
1. open command prompt and type command "pip install graphviz"
2. go to my computer(this pc) and search with the keyword "graphviz"
3. open the graphviz folder and copy its path and save it in notepad
4. In graphviz look for the bin folder and copy the folder by right click of your mouse
5. now again head back to my computer and search for "pydotplus"
6. a folder named *pydotplus* is displayed. Open it and paste the copy of bin folder (of Graphviz) that you copied earlier
7. head to control panel>system and security> system settings> advanced settings> environmental variables> add new path
8. add the path that you copied in notepad and click a series of "ok"

that's it now you can enjoy using graphviz

2)Write a Python program build Decision Tree Classifier for shows.csv from 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
from sklearn import tree
import pydotplus
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import matplotlib.image as pltimg
df = pandas.read_csv("c:\shows.csv")
d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)
features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']
dtree = DecisionTreeClassifier()
```

```
dtree = dtree.fit(X, y)
data = tree.export_graphviz(dtree, out_file=None, feature_names=features)
graph = pydotplus.graph_from_dot_data(data)
graph.write_png('mydecisiontree.png')
img=pltimg.imread('mydecisiontree.png')
imgplot = plt.imshow(img)
plt.show()
```



SET B

1) Consider following dataset

```
weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','S
unny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy']
temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mi
```

ld','Mild','Hot','Mild']
play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Y
es','No']. Use Naïve Bayes algorithm to predict[0:Overcast, 2:Mild]
tuple belongs to which class whether to play the sports or not.
Ans→

```
# Assigning features and label variables
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']
# Import LabelEncoder
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
wheather_encoded=le.fit_transform(weather)
print (wheather_encoded)
# Converting string labels into numbers
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)
print ("Temp:",temp_encoded)
print ("Play:",label)
#Combinig weather and temp into single listof tuples
features=zip(wheather_encoded,temp_encoded)
print (features)
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
model = GaussianNB()

# Train the model using the training sets
model.fit(features,label)

#Predict Output
predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print "Predicted Value:", predicted
```

output: Predicted Value: [1]
Here, 1 indicates that players can 'play'.

SET C

1) Write a Python program to build SVM model to Cancer dataset. The dataset is available in the scikit-learn library. Check the accuracy of model with precision and recall.

```
#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))
```

Association Rules

SET A)

1)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.

Ans→{

```
 "cells": [
  {
   "cell_type": "markdown",
   "id": "b58228cb",
   "metadata": {},
```

```
  "source": [
   "# SET - A\n",
   "\n",
   "### 1) Write a code to read the dataset (“Iris.csv”). dataset download from
(https://archive.ics.uci.edu/ml/datasets/iris) and apply Apriori algorithm."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 1,
  "id": "31f28134",
  "metadata": {},
  "outputs": [],
  "source": [
   "import numpy as np\n",
   "import matplotlib.pyplot as plt\n",
   "import pandas as pd\n",
   "from apyori import apriori"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "91ef7af6",
  "metadata": {},
  "outputs": [],
  "source": [
   "store_data=pd.read_csv('iris.csv',header=None)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "cd4c9ed9",
  "metadata": {},
  "outputs": [],
  "source": [
   "store_data.head()\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "88d01808",
  "metadata": {},
  "outputs": [],
  "source": [
   "records = []\n",
   "for i in range(0,300):\n",
```

```
  "    records.append([str(store_data.values[i,j]) for j in range(0,20)])\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "ba30cca3",
 "metadata": {},
 "outputs": [],
 "source": [

"association_rules=apriori(records,min_support=0.0045,min_confidence=0.2,min_lift=3,min
_length=2)\n",
  "association_results=list(association_rules)\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "8ab0102a",
 "metadata": {},
 "outputs": [],
 "source": [
  "print(len(association_results))\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "daa923d5",
 "metadata": {},
 "outputs": [],
 "source": [
  "print(association_results[0])\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "4f9ceaad",
 "metadata": {},
 "outputs": [],
 "source": [
  "for item in association_results:\n",
  "    pair = item[0]\n",
  "    items = [x for x in pair]\n",
  "    print(\"Rule:\"+items[0]+\"->\"+items[1])\n",
  "    \n",
  "    print(\"Support:\"+str(item[1]))\n",
```

```
   "\n",
   "    print(\"Confidence:\"+str(item[2][0][2]))\n",
   "    print(\"Lift:\"+str(item[2][0][3]))\n",
   "    print(\"=====================================\")"
  ]
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3 (ipykernel)",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.7.9"
 }
},
"nbformat": 4,
"nbformat_minor": 5
}
```


(does the dataset wrong)

2) 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.

Ans→ {


```
 "cells": [
  {
   "cell_type": "markdown",
   "id": "6df9f70d",
   "metadata": {
    "id": "6df9f70d"
   },
   "source": [
    "# SET - A\n",
    "\n",
    "### 2) Write a code to read the dataset (“Groceries.csv”) dataset download from
(https://github.com/amankharwal/Website-
```

data/blob/master/Groceries_dataset.csv) and apply Apriori algorithm also display support and confidence for each rule."

```
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "id": "56e6340c",
    "metadata": {
     "id": "56e6340c"
    },
    "outputs": [],
    "source": [
     "import numpy as np\n",
     "import matplotlib.pyplot as plt\n",
     "import pandas as pd\n",
     "from apyori import apriori"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "id": "7091c7cf",
    "metadata": {
     "id": "7091c7cf"
    },
    "outputs": [],
    "source": [
     "store_data=pd.read_csv('Groceries_dataset.csv',header=None)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "id": "3ca62642",
    "metadata": {
     "id": "3ca62642"
    },
    "outputs": [],
    "source": [
     "store_data.head()\n"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": null,
    "id": "29964735",
    "metadata": {
     "id": "29964735"
```

```
   },
   "outputs": [],
   "source": [
    "records = []\n",
    "for i in range(0,300):\n",
    "    records.append([str(store_data.values[i,j]) for j in range(0,20)])\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "0beac933",
   "metadata": {
    "id": "0beac933"
   },
   "outputs": [],
   "source": [

"association_rules=apriori(records,min_support=0.0045,min_confidence=0.2,min_lift
=3,min_length=2)\n",
    "association_results=list(association_rules)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "e4d1fd53",
   "metadata": {
    "id": "e4d1fd53"
   },
   "outputs": [],
   "source": [
    "print(len(association_results))\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "55580074",
   "metadata": {
    "id": "55580074"
   },
   "outputs": [],
   "source": [
    "print(association_results[0])\n"
   ]
  },
  {
   "cell_type": "code",
```

```
   "execution_count": null,
   "id": "80937225",
   "metadata": {
    "id": "80937225"
   },
   "outputs": [],
   "source": [
    "for item in association_results:\n",
    "    pair = item[0]\n",
    "    items = [x for x in pair]\n",
    "    print(\"Rule:\"+items[0]+\"->\"+items[1])\n",
    "    \n",
    "    print(\"Support:\"+str(item[1]))\n",
    "\n",
    "    print(\"Confidence:\"+str(item[2][0][2]))\n",
    "    print(\"Lift:\"+str(item[2][0][3]))\n",
    "    print(\"=====================================\")"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.7.9"
  },
  "colab": {
   "name": "Data Mining Assignment-3.ipynb",
   "provenance": []
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```


SET B

1)Write a Python program to read “StudentsPerformance.csv” file. Solve following:

- To display the shape of dataset.
- To display the top rows of the dataset with their columns.
- To display the number of rows randomly.
- To display the number of columns and names of the columns.

Note: Download dataset from following link :
(https://www.kaggle.com/spscientist/students-performance-inexams?select=StudentsPerformance.csv)

Ans→{

```
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "name": "Data Mining Assignment-3 SET-B-1.ipynb",
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "markdown",
   "source": [
    "### SET-B\n",
    "\n",
    "1) Write a Python program to read \"StudentsPerformance.csv\" file. solve the following :\n",
    "- To display the shape of dataset.\n",
    "- To display the top rows of the dataset with their columns.\n",
    "- To display the number of rows randomly.\n",
    "- To display the number of columns and names of the columns.\n",
    "- Note : Download dataset from following link:\n",
    "(https://www.kaggle.com/spscientist/students-performance-in-exams?select=StudentsPerformance.csv)"
   ],
   "metadata": {
    "id": "0hhW5uEs_wK2"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "# Import required libraries\n",
    "import numpy as np\n",
    "import matplotlib.pyplot as plt\n",
```

```
    "import pandas as pd\n"
   ],
   "metadata": {
    "id": "W61H7Yo7E_sP"
   },
   "execution_count": 2,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "# Read the downloaded dataset\n",
    "store_data=pd.read_csv('StudentsPerformance.csv',header=None)"
   ],
   "metadata": {
    "id": "uC2jGgIFFVa3"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "# To display the shape of dataset. (By Using shape method)\n",
    "store_data.shape"
   ],
   "metadata": {
    "id": "wU6-JdtCF3ar"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "# To display the top rows of the dataset with their columns.(By using head method\n",
    "store_data.head()"
   ],
   "metadata": {
    "id": "xHtDSrSsGT2v"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "# To display the number of rows randomly.(By using sample method)\n",
    "store_data.sample(10)"
```

```
    ],
    "metadata": {
     "id": "2Gwsi4oTG9QN"
    },
    "execution_count": null,
    "outputs": []
   },
   {
    "cell_type": "code",
    "source": [
     "# To display the number of columns and names of the columns. (By using columns
method)\n",
     "store_data.columns()"
    ],
    "metadata": {
     "id": "ZdXc3aoUHO80"
    },
    "execution_count": null,
    "outputs": []
   }
  ]
}
```


2) Write a Python program for Apriori Algorithm using ARM. And Print the Rule, Support, Confidence and Lift.
- (Set Min_Support = 0.0040, Min_Confidence=0.2, Min_Lift=3, Min_Length=2)
Note: Download dataset from following link :
(https://www.kaggle.com/irfanasrullah/groceries)
Ans→{


```
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "name": "Data Mining Assignment-3 SET-B-2.ipynb",
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "markdown",
   "source": [
    "SET - B\n",
```

```
    "\n",
    "2) Write a code to read the dataset (“groceries.csv”). dataset download from (https://www.kaggle.com/irfanasrullah/groceries) and apply Apriori algorithm with min_support=0.0040, min_support=0.2, min_lift=3, min_length=2."
   ],
   "metadata": {
    "id": "pVfZJLMAvGIJ"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "import numpy as np\n",
    "import matplotlib.pyplot as plt\n",
    "import pandas as pd\n",
    "from apyori import apriori"
   ],
   "metadata": {
    "id": "7KuTYHw1vNJf"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "store_data=pd.read_csv('groceries.csv',header=None)"
   ],
   "metadata": {
    "id": "QvlmFxy16yLg"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "store_data.head()"
   ],
   "metadata": {
    "id": "wwm9lexq-pdY"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "records = []\n",
```

```
      "for i in range(0,300):\n",
      "    records.append([str(store_data.values[i,j]) for j in range(0,20)])\n"
     ],
     "metadata": {
      "id": "p4_M413V-tkY"
     },
     "execution_count": null,
     "outputs": []
    },
    {
     "cell_type": "code",
     "source": [

"association_rules=apriori(records,min_support=0.0040,min_confidence=0.2,min_lift=3,min
_length=2)\n",
      "association_results=list(association_rules)\n"
     ],
     "metadata": {
      "id": "7ZTOd9jQ-z5F"
     },
     "execution_count": null,
     "outputs": []
    },
    {
     "cell_type": "code",
     "source": [
      "print(len(association_results))"
     ],
     "metadata": {
      "id": "LYXHcNQs-_Cj"
     },
     "execution_count": null,
     "outputs": []
    },
    {
     "cell_type": "code",
     "source": [
      "print(association_results[0])"
     ],
     "metadata": {
      "id": "1gmQcNZk_Ekl"
     },
     "execution_count": null,
     "outputs": []
    },
    {
     "cell_type": "code",
     "source": [
      "for item in association_results:\n",
```

```
      "    pair = item[0]\n",
      "    items = [x for x in pair]\n",
      "    print(\"Rule:\"+items[0]+\"->\"+items[1])\n", "    \n",
      "    print(\"Support:\"+str(item[1]))\n", "\n",
      "    print(\"Confidence:\"+str(item[2][0][2]))\n", "
print(\"Lift:\"+str(item[2][0][3]))\n",
      "    print(\"========================================\")"
    ],
    "metadata": {
     "id": "mNizziCV_Jp-"
    },
    "execution_count": null,
    "outputs": []
   }
  ]
}
```

Set C

Write a Python Program to implement Apriori algorithm for following data set. Create csv file in excel. Apply Apriori algorithm and print the association results.

Wine	Chips	Bread	Butter	Milk	Apple
Wine	Chips		Butter	Milk	
		Bread	Butter	Milk	Apple
		Bread		Milk	Apple
Wine	Chips	Bread		Milk	
Wine	Chips		Butter	Milk	Apple
Wine	Chips		Butter		Apple
Wine	Chips	Bread	Butter	Milk	Apple
Wine	Chips	Bread	Butter	Milk	Apple
Wine	Chips		Butter	Milk	
			Butter		Apple
Wine		Bread	Butter	Milk	Apple
Wine	Chips	Bread	Butter	Milk	Apple
Wine	Chips	Bread		Milk	Apple

Ans→

Regression Analysis and outlier detection

Set A : Simple Linear Regression

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

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x, y):
```

```
    # number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

    return (b_0, b_1)

def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
            marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')

    # function to show plot
    plt.show()

def main():
    # observations / 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])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {}  \
        \nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)

if __name__ == "__main__":
```

```
main()
```

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

```
import matplotlib.pyplot as plt
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()
```

3) Consider the student data set It can be downloaded from:
https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaqO5_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.

Set B: Multiple Linear Regression

1) 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.asp
Ans→

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("d:dmdataset\carsm.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)
```

2) Write a python programme to implement multiple linear regression model for stock market data frame as follows:

```
Stock_Market = {'Year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,20
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':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.7
5,1.75,1.75,1.75,1.75],
'Unemployment_Rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6
.2,6.1],
'Stock_Index_Price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,
958,971,949,884,866,876,822,704,719] }
```

And draw a graph of stock market price verses interest rate.

Ans→

```
mport 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],
        '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='red')
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()
```

Set C: Outlier Detection

1) Write a programme in python to print the number of outliers. Generate 200 samples, from a normal distribution, cantered around the value 100, with a standard deviation of 5.
Ans→
Z-Score and How It's Used to Determine an Outlier | by Iden W. | Clarusway | Medium

Clustering
Set A
1. Write a python program to implement k-means algorithm to build prediction model (Use Credit Card Dataset CC GENERAL.csv Download from kaggle.com)
Ans→

```
#dataset --> https://www.kaggle.com/mlg-ulb/creditcardfraud/version/3
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

dataset = pd.read_csv('creditcard.csv')

dataset

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

print(x)

from sklearn.cluster import KMeans

wcss_list= []

for i in range(1, 11):

kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)

kmeans.fit(x)

wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list)

mtp.title('The Elobw Method Graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss_list')

mtp.show()

kmeans = KMeans(n_clusters=3, init='k-means++', random_state= 42)

y_predict= kmeans.fit_predict(x)
```

```
        mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label
    = 'Cluster 1') #for first cluster

        mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green',
    label = 'Cluster 2') #for second cluster

mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
    #for third cluster

mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c =
    'yellow', label = 'Centroid')

        mtp.title('Clusters of Credit Card')

        mtp.xlabel('V3')

        mtp.ylabel('V4')

        mtp.legend()

        mtp.show()
```

2. Write a python program to implement hierarchical Agglomerative clustering algorithm. (Download Customer.csv dataset from github.com).

Ans→

```
dataset = pd.read_csv('Mall_Customers.csv')

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

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('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Set B

1. Write a python program to implement k-means algorithms on a synthetic dataset.

Ans→

```
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')
```

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

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset = pd.read_csv('Wholesale customers data.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()
```