Name:
Class: T.Y.B.SC (Comp.Sci)
Roll No:

Assignment 1: Linear and Logistic Regression

SET:-A

1. Create ‘sales’ Data set having 5 columns namely: ID, TV, Radio, Newspaper and Sales.(random 500 entries) Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets. then divide the training and testing sets into a 7:3 ratio, respectively and print them. Build a simple linear regression model

```
#importing Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

#Loading Datasets
df=pd.read_csv("Advertising.csv")
df.head(5)

#Identifying Independent and target varibles
X=df[['TV','Radio','Newspaper']]
Y=df['Sales']
X.head(5)

#Splitting datasets Into Training and Testing Sets
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)

#print the data
print("\nTraining Set of X",x_train)
print("\nTesting Set of X",x_test)
print("\nTraining Set of Y",y_train)
print("\nTesting Set of Y",y_test)

#Creating object of Linear Regression
from sklearn.linear_model import LinearRegression
clf=LinearRegression()
```

```
#Fitting The x_train and y_train varibles
clf.fit(x_train,y_train)

#Predicting Output By passing x_test
pred_x=clf.predict(x_test)
print("\nPredicted values of X",pred_x)

#Test accuracy
accuracy=clf.score(x_test,y_test)
print("\n\n Accuracy of Model",accuracy)
```

Output:

```
Training Set of X        TV  Radio  Newspaper
112  175.7   15.4       2.4
54   262.7   28.8      15.9
143  104.6    5.7      34.4
124  229.5   32.3      74.2
77   120.5   28.5      14.2
28   248.8   27.1      22.9
42   293.6   27.7       1.8
50   199.8    3.1      34.6
51   100.4    9.6       3.6
91    28.6    1.5      33.0
115   75.1   35.0      52.7
101  296.4   36.3     100.9
71   109.8   14.3      31.7
104  238.2   34.3       5.3

Testing Set of X         TV  Radio  Newspaper
118  125.7   36.9      79.2
37    74.7   49.4      45.7
183  287.6   43.0      71.8
0    230.1   37.8      69.2
99   135.2   41.7      45.9
47   239.9   41.5      18.5
176  248.4   30.2      20.3
30   292.9   28.3      43.2
171  164.5   20.9      47.4
145  140.3    1.9       9.0
158   11.7   36.9      45.2
153  171.3   39.7      37.7
```

164 117.2 14.7 5.4
85 193.2 18.4 65.7
7 120.2 19.6 11.6
119 19.4 16.0 22.3

Training Set of Y 112 14.1
54 20.2
143 10.4
124 19.7
77 14.2
28 18.9
42 20.7
50 11.4
51 10.7
91 7.3
115 12.6
101 23.8
71 12.4
104 20.7
9 10.6
141 19.2
128 24.7
40 16.6
52 22.6
187 17.3
102 14.8
39 21.5
14 19.0

Name: Sales, Length: 140, dtype: float64

Testing Set of Y 118 15.9
37 14.7
183 26.2
0 22.1
99 17.2
47 23.2
176 20.2
30 21.4
171 14.5
145 10.3
158 7.3

```
153    19.0
164    11.9
85     15.2
7      13.2

Name: Sales, dtype: float64

Predicted values of X [15.48636218 15.83877424 23.95829971 20.39574226 17.05393606
21.87532538
 20.05680154 21.53329694 14.32909434  9.81466676 10.60364413 18.33454731
 11.28183922 15.01200293 12.32480652  7.04938384  9.19623276 16.48738441
 14.67315117 13.49798672  7.7326357   9.71926893 10.86802172  4.63469735
  4.58791452  7.59179748 12.53638752 16.13122113  7.94045701 12.05928765
 17.61839651 21.05487729 15.90072195 13.8730696  20.68464495 15.35292364
 23.30937904 10.02552662 21.00361146 11.86211747 12.70796669 13.36214139
 17.95822975 16.52409225 13.61407458  8.92813164 12.05411349 10.78167546
  5.87093476 13.75207464 17.45333969  9.91961673 12.93121223 16.77764695
 14.51356082 19.25701022 12.61470086 16.54524362 17.19236167 17.06299047]


 Accuracy of Model 0.8527485576171627
```

2. Create ‘realestate’ Data set having 4 columns namely: ID,flat, houses and purchases (random 500 entries). Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets and print them. Build a simple linear regression model for predicting purchases.

```
#importing Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
from matplotlib import pyplot as plt

#Loading Datasets
dataset=pd.read_csv("User_Data.csv")

#Splitting Dataset into dependent (Purchase) and Independent(Age and Estimated
salary)varibles
```

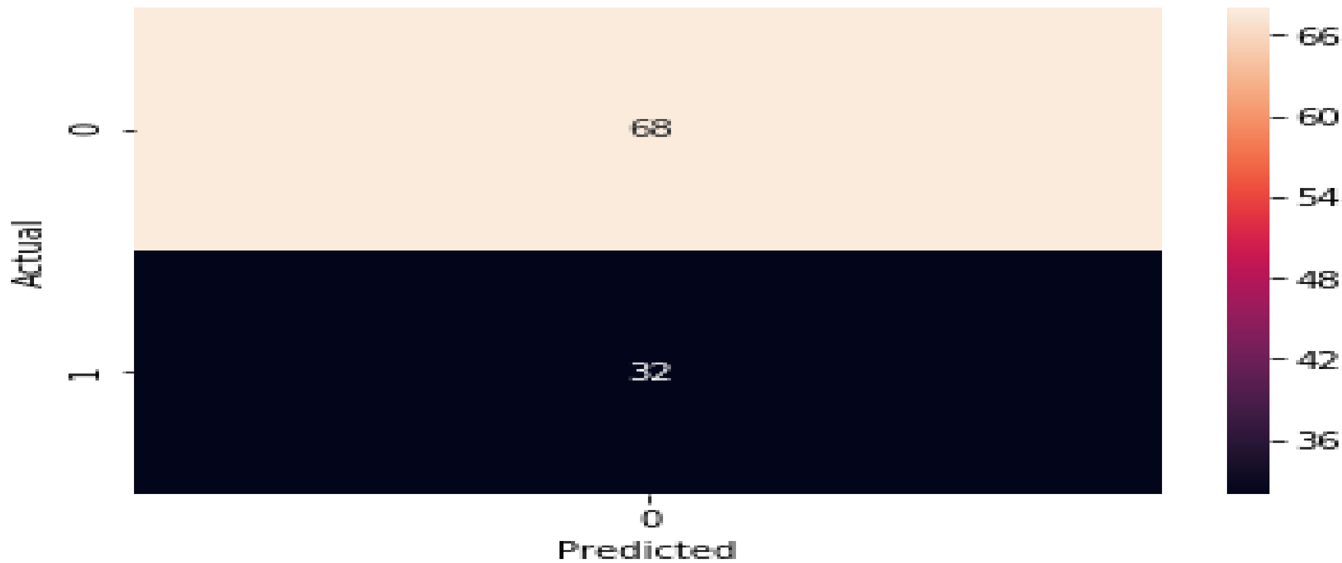
```
x=dataset.iloc[:,[2,3]].values
y=dataset.iloc[:,4].values
print(x[:10])
print(x[:10])

#Splitting datasets Into Training and Testing Sets
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)

#Performing Logistic Regression
logistic_regression=LogisticRegression()
logistic_regression.fit(x_train,y_train)
pred_y=logistic_regression.predict(x_test)

#Print the accuract and Plot the Confusion Matrix
confusion_matrix=pd.crosstab(y_test,pred_y,rownames=['Actual'],colnames=['Predicted'])
sn.heatmap(confusion_matrix,annot=True)
print('Accuracy:',metrics.accuracy_score(y_test,pred_y))
plt.show()

#Print testdata and PredictedData
print(x_test)
print(pred_y)

new_pred=logistic_regression.predict([[32,150000]])
print("Person with given age and Salary will buy a car?:",new_pred)
```

OutPut:-

```
[[    19  19000]
 [    35  20000]
 [    26  43000]
 [    27  57000]
 [    19  76000]
 [    27  58000]
 [    27  84000]
 [    32 150000]
 [    25  33000]
 [    35  65000]]
[[    19  19000]
 [    35  20000]
 [    26  43000]
 [    27  57000]
 [    19  76000]
```

[27 58000]

Accuracy: 0.68



[[30 87000]
 [38 50000]
 [35 75000]
 [30 79000]
 [35 50000]
 [27 20000]
 [31 15000]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Person with given age and Salary will buy a car?:

SET:-B

1. Build a simple linear regression model for Fish Species Weight Prediction. (download dataset https://www.kaggle.com/aungpyaeap/fish-market?select=Fish.csv)

```
#collecting data
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

#Loading Datasets
df=pd.read_csv("Fish.csv")
#Displaying data
print(df.head(5))
#Identifying Independent and target varibles
X=df[['Length1','Length2','Length3','Height','Weight']]
Y=df['Weight']

#Splitting datasets Into Training and Testing Sets
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)

#print the data
print("\nTraining Set of X",x_train)
print("\nTesting Set of X",x_test)
print("\nTraining Set of Y",y_train)
print("\nTesting Set of Y",y_test)

#Creating object of Linear Regression
from sklearn.linear_model import LinearRegression
clf=LinearRegression()

#Fitting The x_train and y_train varibles
clf.fit(x_train,y_train)
```

```
#Predicting Output By passing x_test
pred_x=clf.predict(x_test)
print("\nPredicted values of X",pred_x)

#Test accuracy
accuracy=clf.score(x_test,y_test)
print("\n\n Accuracy of Model",accuracy)
```

OutPut:-

```
  Species  Weight  Length1  Length2  Length3   Height   Width
0   Bream   242.0    23.2     25.4    30.0  11.5200  4.0200
1   Bream   290.0    24.0     26.3    31.2  12.4800  4.3056
2   Bream   340.0    23.9     26.5    31.1  12.3778  4.6961
3   Bream   363.0    26.3     29.0    33.5  12.7300  4.4555
4   Bream   430.0    26.5     29.0    34.0  12.4440  5.1340

Training Set of X      Length1  Length2  Length3   Height  Weight
148    10.4    11.0    12.0   2.1960    9.7
23     31.8    35.0    40.6  15.4686   680.0
22     31.5    34.5    39.7  15.5227   620.0
5      26.8    29.7    34.7  13.6024   450.0
128    30.0    32.3    34.8   5.5680   200.0
17     30.4    33.0    38.5  14.9380   700.0
113    34.0    36.0    38.3  10.6091   700.0
1      24.0    26.3    31.2  12.4800   290.0
127    41.1    44.0    46.6  12.4888  1000.0



[111 rows x 5 columns]

Testing Set of X      Length1  Length2  Length3   Height  Weight
124    39.8    43.0    45.2  11.9328  1000.0
88     20.0    22.0    23.5   6.1100   130.0
147    10.1    10.6    11.6   1.7284     7.0
24     31.9    35.0    40.5  16.2405   700.0
35     12.9    14.1    16.2   4.1472    40.0
97     22.0    24.0    25.5   6.3750   145.0
67     19.0    20.7    23.2   9.3960   170.0
131    34.8    37.3    39.8   6.2884   300.0
85     19.3    21.3    22.8   6.3840   130.0
```

```
Training Set of Y 148       9.7
23      680.0
22      620.0
5       450.0
128     200.0
17      700.0
113     700.0
1       290.0
199     180.0
137     500.0
14      600.0
2       340.0

102     300.0

Name: Weight, Length: 111, dtype: float64

Testing Set of Y 124    1000.0
88      130.0
147       7.0
24      700.0
35       40.0
97      145.0
67      170.0
131     300.0
85      130.0
0       242.0
152       9.9
10      475.0
112     685.0



Name: Weight, dtype: float64

Predicted values of X [1000.   130.     7.   700.    40.   145.   170.   300.   130.   242.
    9.9  475.   685.    32.   169.   150.    40.   273.    69.   925.
  200.   720.   300.   110.   150.   170.   110.   955.   800.  1100.
  770.    85.   345.   265.   600.   290.   685.   135.   950.    85.
```

110.   7.5   12.2 450.   430.   300.   160.   840.]

Accuracy of Model 1.0

2. Use the iris dataset. Write a Python program to view some basic statistical details like percentile, mean, std etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-virginica'. Apply logistic regression on the dataset to identify different species (setosa, versicolor, verginica) of Iris flowers given just 4 features: sepal and petal lengths and widths.. Find the accuracy of the model. Signature of the instructor Date Assignment Evaluation.

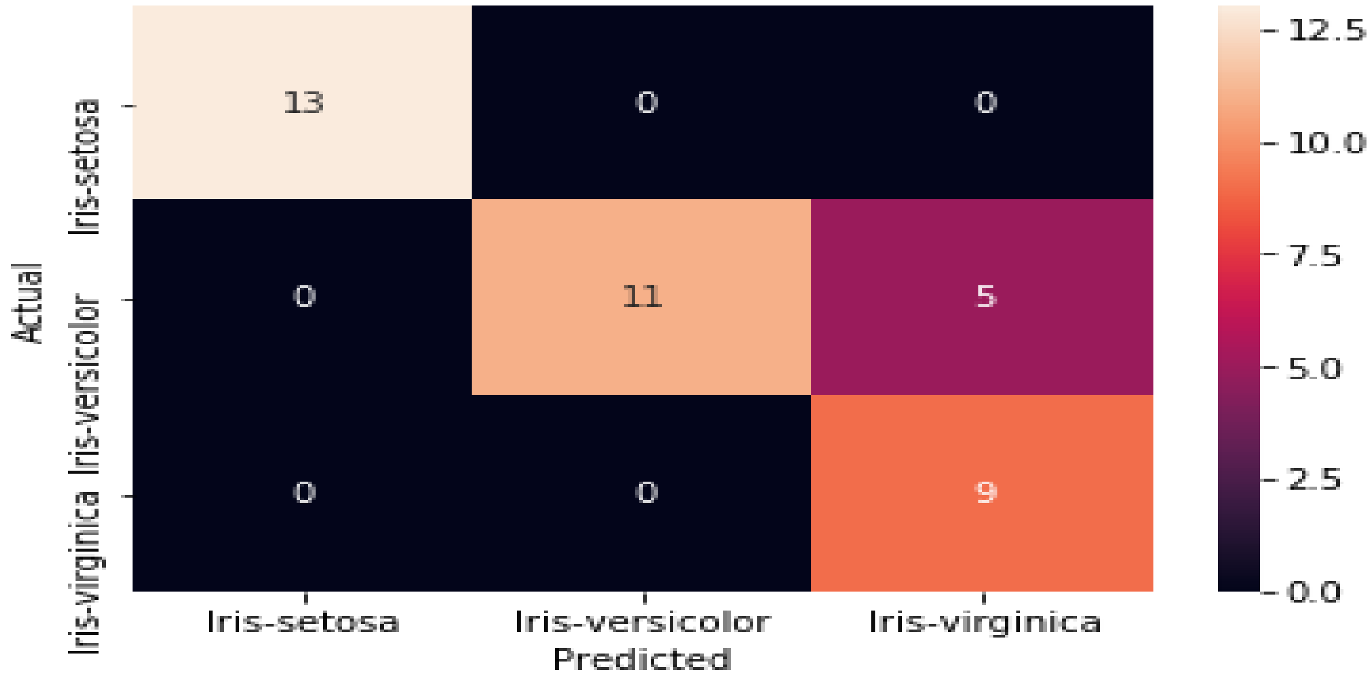
```
#importing Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
from matplotlib import pyplot as plt

#Reading DataSet
data=pd.read_csv("Iris.csv")
print('Iris-setosa')
setosa=data['Species']=='Iris-setosa'
print(data[setosa].describe())
print('Iris-vesicolor')
setosa=data['Species']=='Iris-vesicolor'
print(data[setosa].describe())
print('Iris-virgincia')
setosa=data['Species']=='Iris-virgincia'
print(data[setosa].describe())

#Splitting Dataset into dependent (Purchase) and Independent(Age and Estimated
salary)varibles
x=data[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]
y=data['Species']
print(x[:10])
print(x[:10])
```

```
#Splitting datasets Into Training and Testing Sets
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)

#Performing Logistic Regression
logistic_regression=LogisticRegression()
logistic_regression.fit(x_train,y_train)
pred_y=logistic_regression.predict(x_test)


#Print the accuract and Plot the Confusion Matrix
confusion_matrix=pd.crosstab(y_test,pred_y,rownames=['Actual'],colnames=['Predicted'])
sn.heatmap(confusion_matrix,annot=True)
print('Accuracy:',metrics.accuracy_score(y_test,pred_y))
plt.show()

#Print testdata and PredictedData
print(x_test)
print(pred_y)

new_pred=logistic_regression.predict([[5.8,2.4,3.2,5.6]])
print("Predicted Species?:",new_pred)
```

OutPut:-

```
Iris-setosa
              Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count   50.00000       50.00000     50.000000      50.000000      50.00000
mean    25.50000        5.00600      3.418000       1.464000       0.24400
std     14.57738        0.35249      0.381024       0.173511       0.10721
min      1.00000        4.30000      2.300000       1.000000       0.10000
25%     13.25000        4.80000      3.125000       1.400000       0.20000
50%     25.50000        5.00000      3.400000       1.500000       0.20000
75%     37.75000        5.20000      3.675000       1.575000       0.30000
max     50.00000        5.80000      4.400000       1.900000       0.60000
Iris-vesicolor
        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  0.0            0.0           0.0            0.0           0.0
mean   NaN            NaN           NaN            NaN           NaN
std    NaN            NaN           NaN            NaN           NaN
min    NaN            NaN           NaN            NaN           NaN
25%    NaN            NaN           NaN            NaN           NaN
50%    NaN            NaN           NaN            NaN           NaN
```

```
75%    NaN           NaN          NaN           NaN          NaN
max    NaN           NaN          NaN           NaN          NaN
Iris-virgincia
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  0.0            0.0           0.0            0.0           0.0
mean   NaN            NaN           NaN            NaN           NaN
std    NaN            NaN           NaN            NaN           NaN
min    NaN            NaN           NaN            NaN           NaN
25%    NaN            NaN           NaN            NaN           NaN
50%    NaN            NaN           NaN            NaN           NaN
75%    NaN            NaN           NaN            NaN           NaN
max    NaN            NaN           NaN            NaN           NaN
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
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
5            5.4           3.9            1.7           0.4
6            4.6           3.4            1.4           0.3
7            5.0           3.4            1.5           0.2
8            4.4           2.9            1.4           0.2
9            4.9           3.1            1.5           0.1
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
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
5            5.4           3.9            1.7           0.4
6            4.6           3.4            1.4           0.3
7            5.0           3.4            1.5           0.2
8            4.4           2.9            1.4           0.2
9            4.9           3.1            1.5           0.1
Accuracy: 0.868421052631579
```



	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
114	5.8	2.8	5.1	2.4
62	6.0	2.2	4.0	1.0
33	5.5	4.2	1.4	0.2
107	7.3	2.9	6.3	1.8
7	5.0	3.4	1.5	0.2
100	6.3	3.3	6.0	2.5
40	5.0	3.5	1.3	0.3
86	6.7	3.1	4.7	1.5
76	6.8	2.8	4.8	1.4
71	6.1	2.8	4.0	1.3

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica']
Predicted Species?: ['Iris-virginica']
```

Assignment 2: Frequent itemset and Association rule mining

Set:-A

1. Create the following dataset in python Convert the categorical values into numeric format. DATA ANALYTICS ASSIGNMENT 2 | Prepared by : Dr. Poonam Ponde Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min_sup values.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori,association_rules
#Create the sample dataset
transactions=[['Bread','Milk'],['Bread','Diaper','Beer','Eggs'],['Milk','Diaper','Beer','Coke'],['Bread'
,'Milk','Diaper','Beer'],['Bread','Milk','Diaper','Coke']]

#Transform it into the right format via Transactioon Encoder as Follows:
from mlxtend.preprocessing import TransactionEncoder
te=TransactionEncoder()
te_arrary=te.fit(transactions).transform(transactions)
df=pd.DataFrame(te_arrary,columns=te.columns_)
print(df)

#Fint the Frequent Itemsets
freq_items=apriori(df,min_support=0.5,use_colnames=True)
print(freq_items)

#Generate The association Rules
rules=association_rules(freq_items,metric='support',min_threshold=0.05)
rules=rules.sort_values(['support','confidence'],ascending=[False,False])
print(rules)
```

OutPut:-

```
==================== RESTART: C:/Users/Admin/Desktop/11.py ====================
    Beer  Bread   Coke  Diaper   Eggs   Milk
0  False   True  False   False  False   True
1   True   True  False    True   True  False
2   True  False   True    True  False   True
3   True   True  False    True  False   True
4  False   True   True    True  False   True
   support       itemsets
```

```
0     0.6          (Beer)
1     0.8         (Bread)
2     0.8        (Diaper)
3     0.8          (Milk)
4     0.6   (Beer, Diaper)
5     0.6  (Bread, Diaper)
6     0.6    (Milk, Bread)
7     0.6   (Milk, Diaper)
  antecedents consequents  antecedent support  ...    lift  leverage  conviction
0      (Beer)    (Diaper)                 0.6  ...  1.2500      0.12         inf
1    (Diaper)      (Beer)                 0.8  ...  1.2500      0.12         1.6
2     (Bread)    (Diaper)                 0.8  ...  0.9375     -0.04         0.8
3    (Diaper)     (Bread)                 0.8  ...  0.9375     -0.04         0.8
4      (Milk)     (Bread)                 0.8  ...  0.9375     -0.04         0.8
5     (Bread)      (Milk)                 0.8  ...  0.9375     -0.04         0.8
6      (Milk)    (Diaper)                 0.8  ...  0.9375     -0.04         0.8
7    (Diaper)      (Milk)                 0.8  ...  0.9375     -0.04         0.8

[8 rows x 9 columns]
```

2. Create your own transactions dataset and apply the above process on your dataset.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori,association_rules
#Create the sample dataset
transactions=[['Bread','Milk'],['Bread','Apple','Beer','Eggs'],['Milk','Apple','Beer','Coke'],['Bread','
Milk','Apple','Beer'],['Bread','Milk','Apple','Coke']]

#Transform it into the right format via Transactioon Encoder as Follows:
from mlxtend.preprocessing import TransactionEncoder
te=TransactionEncoder()
te_array=te.fit(transactions).transform(transactions)
df=pd.DataFrame(te_array,columns=te.columns_)
print(df)

#Fint the Frequent Itemsets
freq_items=apriori(df,min_support=0.5,use_colnames=True)
print(freq_items)

#Generate The association Rules
rules=association_rules(freq_items,metric='support',min_threshold=0.05)
rules=rules.sort_values(['support','confidence'],ascending=[False,False])
```

print(rules)

OutPut:-

```
===== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/22.py ====
   Apple   Beer  Bread   Coke   Eggs   Milk
0  False  False   True  False  False   True
1   True   True   True  False   True  False
2   True   True  False   True  False   True
3   True   True   True  False  False   True
4   True  False   True   True  False   True
   support       itemsets
0      0.8        (Apple)
1      0.6         (Beer)
2      0.8        (Bread)
3      0.8         (Milk)
4      0.6   (Beer, Apple)
5      0.6  (Bread, Apple)
6      0.6   (Milk, Apple)
7      0.6   (Bread, Milk)
  antecedents consequents  antecedent support  ...    lift  leverage  conviction
0      (Beer)     (Apple)                 0.6  ...  1.2500      0.12         inf
1     (Apple)      (Beer)                 0.8  ...  1.2500      0.12         1.6
2     (Bread)     (Apple)                 0.8  ...  0.9375     -0.04         0.8
3     (Apple)     (Bread)                 0.8  ...  0.9375     -0.04         0.8
4      (Milk)     (Apple)                 0.8  ...  0.9375     -0.04         0.8
5     (Apple)      (Milk)                 0.8  ...  0.9375     -0.04         0.8
6     (Bread)      (Milk)                 0.8  ...  0.9375     -0.04         0.8
7      (Milk)     (Bread)                 0.8  ...  0.9375     -0.04         0.8

[8 rows x 9 columns]
```

SET:-B

1. Download the Market basket dataset. Write a python program to read the dataset and display its information. Preprocess the data (drop null values etc.) Convert the categorical values into numeric format. Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori,association_rules
#Create the sample dataset
data =pd.read_csv('OnlineRetail.csv',encoding='ISO-8859-1')
print(data.head(5))

#Preprocesing data droping NULL values
data=data.dropna()
data.info()
#Using Positive Quality Values
data_plus=data[data['Quantity']>=0]
data_plus.info()

#Creating Basket data with Transactions From UK only

basket_plus=data_plus[data_plus['Country']=='United
Kingdom'].groupby(['InvoiceNo','Description'])['Quantity'].sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo')
print("\n***********************Basket with UK
Transaction*******************************\n",basket_plus)


def encode_units(x):
   if x<=0:
      return 0
   if x>=0:
      return 1
basket_encode_plus=basket_plus.applymap(encode_units)
print("\n**********************Encode
Basket***************\n",basket_encode_plus)

#Filter Data
basket_filter_plus=basket_encode_plus[(basket_encode_plus>0).sum(axis=1)>=2]
print("\n**********************Filter Basket***************\n",basket_filter_plus)
```

```
#Fint the Frequent Itemsets
freq_items=apriori(basket_filter_plus,min_support=0.03,use_colnames=True)
print("\n**********************Frequent Items****************",freq_items)
#Generate The association Rules
rules=association_rules(freq_items,metric='lift',min_threshold=1)
rules=rules.sort_values('lift',ascending=False)
print("\n*********************Association_Rules******************\n",rules.head(5))
```

OutPut:-

```
InvoiceNo StockCode  ... CustomerID        Country
0    536365    85123A  ...    17850.0  United Kingdom
1    536365     71053  ...    17850.0  United Kingdom
2    536365    84406B  ...    17850.0  United Kingdom
3    536365    84029G  ...    17850.0  United Kingdom
4    536365    84029E  ...    17850.0  United Kingdom

[5 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  object
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 397924 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    397924 non-null  object
 1   StockCode    397924 non-null  object
 2   Description  397924 non-null  object
```

```
 3   Quantity     397924 non-null  int64
 4   InvoiceDate  397924 non-null  object
 5   UnitPrice    397924 non-null  float64
 6   CustomerID   397924 non-null  float64
 7   Country      397924 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.3+ MB

**********************Basket with UK Transaction******************************
 Description   4 PURPLE FLOCK DINNER CANDLES  ...  ZINC WIRE SWEETHEART LETTER TRAY
InvoiceNo                                     ...
536365                                  0.0  ...                              0.0
536366                                  0.0  ...                              0.0
536367                                  0.0  ...                              0.0
536368                                  0.0  ...                              0.0
536369                                  0.0  ...                              0.0
...                                     ...  ...                              ...
581582                                  0.0  ...                              0.0
581583                                  0.0  ...                              0.0
581584                                  0.0  ...                              0.0
581585                                  0.0  ...                              0.0
581586                                  0.0  ...                              0.0

[16649 rows x 3844 columns]

*********************Encode Basket***************
 Description   4 PURPLE FLOCK DINNER CANDLES  ...  ZINC WIRE SWEETHEART LETTER TRAY
InvoiceNo                                     ...
536365                                    0  ...                                0
536366                                    0  ...                                0
536367                                    0  ...                                0
536368                                    0  ...                                0
536369                                    0  ...                                0
...                                     ...  ...                              ...
581582                                    0  ...                                0
581583                                    0  ...                                0
581584                                    0  ...                                0
581585                                    0  ...                                0
581586                                    0  ...                                0

[16649 rows x 3844 columns]
```

```
**********************Filter Basket****************
 Description   4 PURPLE FLOCK DINNER CANDLES  ...  ZINC WIRE SWEETHEART LETTER TRAY
InvoiceNo                                    ...
536365                                 0  ...                                 0
536366                                 0  ...                                 0
536367                                 0  ...                                 0
536368                                 0  ...                                 0
536372                                 0  ...                                 0
...                                  ...  ...                               ...
581582                                 0  ...                                 0
581583                                 0  ...                                 0
581584                                 0  ...                                 0
581585                                 0  ...                                 0
581586                                 0  ...                                 0

[15376 rows x 3844 columns]

**********************Frequent Items****************       support
itemsets
0    0.040843                        (6 RIBBONS RUSTIC CHARM)
1    0.038176                   (60 TEATIME FAIRY CAKE CASES)
2    0.044550                    (ALARM CLOCK BAKELIKE GREEN)
3    0.031933                     (ALARM CLOCK BAKELIKE PINK)
4    0.049298                     (ALARM CLOCK BAKELIKE RED )
..        ...                                             ...
103  0.055411               (WOODEN PICTURE FRAME WHITE FINISH)
104  0.030957  (ROSES REGENCY TEACUP AND SAUCER , GREEN REGEN...
105  0.032908  (JUMBO BAG PINK POLKADOT, JUMBO BAG RED RETROS...
106  0.031478  (LUNCH BAG RED RETROSPOT, LUNCH BAG  BLACK SKU...
107  0.030632  (LUNCH BAG RED RETROSPOT, LUNCH BAG PINK POLKA...

[108 rows x 2 columns]

*********************Association_Rules******************
                       antecedents  ... conviction
0  (ROSES REGENCY TEACUP AND SAUCER )  ...   3.256952
1   (GREEN REGENCY TEACUP AND SAUCER)  ...   4.302452
6          (LUNCH BAG RED RETROSPOT)  ...   1.630668
7          (LUNCH BAG PINK POLKADOT)  ...   2.088574
2          (JUMBO BAG PINK POLKADOT)  ...   2.416152

[5 rows x 9 columns]
```

2. Download the groceries dataset. Write a python program to read the dataset and display its information. Preprocess the data (drop null values etc.) Convert the categorical values into numeric format. Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.

```
# Importing Libraries
import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder



# Read Dataset
basket = pd.read_csv("Groceries_dataset.csv")
print("\n----------- Dataset-------------\n",basket.head(5))

# Preprocessing data dropping NULL values
basket=basket.dropna()
basket.info()

# Grouping into Transactions
basket.itemDescription = basket.itemDescription.transform(lambda x: [x])
basket =
basket.groupby(['Member_number','Date']).sum()['itemDescription'].reset_index(drop=True)

encoder = TransactionEncoder()
transactions = pd.DataFrame(encoder.fit(basket).transform(basket),
columns=encoder.columns_)
print("\n-----------Transaction Data------------------\n",transactions.head(5))


# Apriori and Association Rules
frequent_itemsets = apriori(transactions, min_support= 6/len(basket), use_colnames=True,
max_len = 2)
rules = association_rules(frequent_itemsets, metric="lift",  min_threshold = 1.5)
print("\n-------------Frequent Itemsets------------------\n",frequent_itemsets)
print("\n------------ Assoiation Rules-------------\n",rules.head(5))
print("Rules identified: ", len(rules))
```

OutPut:-

```
   Member_number       Date   itemDescription
0           1808  21-07-2015    tropical fruit
1           2552  05-01-2015        whole milk
2           2300  19-09-2015         pip fruit
3           1187  12-12-2015  other vegetables
4           3037  01-02-2015        whole milk
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.2+ MB

*******************Transaction Data*****************************
   Instant food products  UHT-milk  ...  yogurt  zwieback
0                  False     False  ...    True     False
1                  False     False  ...   False     False
2                  False     False  ...   False     False
3                  False     False  ...   False     False
4                  False     False  ...   False     False

[5 rows x 167 columns]

*********************Frequent ItemsSets***************        support
itemsets
0     0.004010   (Instant food products)
1     0.021386                (UHT-milk)
2     0.001470       (abrasive cleaner)
3     0.001938       (artif. sweetener)
4     0.008087          (baking powder)
...        ...                       ...
1773  0.001069     (yogurt, white bread)
1774  0.001270  (whole milk, white wine)
1775  0.000535      (yogurt, white wine)
1776  0.011161      (yogurt, whole milk)
1777  0.000468    (whole milk, zwieback)

[1778 rows x 2 columns]
```

```
*********************Association_Rules******************
         antecedents      consequents  ...  leverage  conviction
0      (butter milk)       (UHT-milk)  ...  0.000226    1.013289
1         (UHT-milk)    (butter milk)  ...  0.000226    1.010854
2     (cream cheese )       (UHT-milk)  ...  0.000363    1.015922
3         (UHT-milk)  (cream cheese )  ...  0.000363    1.017685
4  (artif. sweetener)           (soda)  ...  0.000280    1.190178

[5 rows x 9 columns]
Rules Identified 190
```