Name :: Ilyas Shaikh

Roll no :: 201

CS – 368 Section II Data Analytics

*# Assignment1 Set A Q1. (SLR)*

# Importing Libraries

import pandas as pd

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

# Loading datasets

df = pd.read\_csv('Advertising.csv')

df.head(5)

# Identifying Independent and Target variables

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("\n Training Set of X=\n",x\_train)

print("\n Testing Set of X=\n",x\_test)

print("\n Training Set of Y=\n",y\_train)

print("\n Testing Set of Y=\n",y\_test)

# Creating object of Linear Regression

from sklearn.linear\_model import LinearRegression

clf = LinearRegression()

# fitting the x\_train and y\_train variables.

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

# Predicting output by passing x\_test

pred\_x=clf.predict(x\_test)

print("\n Predicted 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

25 262.9 3.5 19.5

15 195.4 47.7 52.9

152 197.6 23.3 14.2

170 50.0 11.6 18.4

49 66.9 11.7 36.8

.. ... ... ...

80 76.4 26.7 22.3

50 199.8 3.1 34.6

195 38.2 3.7 13.8

157 149.8 1.3 24.3

132 8.4 27.2 2.1

[140 rows x 3 columns]

Testing Set of X=

TV Radio Newspaper

188 286.0 13.9 3.7

145 140.3 1.9 9.0

115 75.1 35.0 52.7

35 290.7 4.1 8.5

36 266.9 43.8 5.0

105 137.9 46.4 59.0

46 89.7 9.9 35.7

130 0.7 39.6 8.7

30 292.9 28.3 43.2

18 69.2 20.5 18.3

64 131.1 42.8 28.9

144 96.2 14.8 38.9

48 227.2 15.8 49.9

17 281.4 39.6 55.8

5 8.7 48.9 75.0

158 11.7 36.9 45.2

146 240.1 7.3 8.7

187 191.1 28.7 18.2

74 213.4 24.6 13.1

108 13.1 0.4 25.6

10 66.1 5.8 24.2

78 5.4 29.9 9.4

0 230.1 37.8 69.2

70 199.1 30.6 38.7

37 74.7 49.4 45.7

68 237.4 27.5 11.0

168 215.4 23.6 57.6

183 287.6 43.0 71.8

66 31.5 24.6 2.2

95 163.3 31.6 52.9

39 228.0 37.7 32.0

Training Set of Y=

25 12.0

15 22.4

152 16.6

170 8.4

49 9.7

...

80 11.8

50 11.4

195 7.6

157 10.1

132 5.7

Name: Sales, Length: 140, dtype: float64

Testing Set of Y=

188 15.9

145 10.3

115 12.6

35 12.8

36 25.4

105 19.2

46 10.6

130 1.6

30 21.4

18 11.3

64 18.0

144 11.4

48 14.8

17 24.4

5 7.2

158 7.3

72 8.8

98 25.4

41 17.1…..

147 25.4

99 17.2

60 8.1

3 18.5

85 15.2

139 20.7

185 22.6

65 9.3

189 6.7

42 20.7

68 18.9

Name: Sales, dtype: float64

Predicted Values of x = [18.59381546 9.80839467 12.92305949 16.88544969 23.46940384 17.8354517

8.89262889 10.8624517 21.3463337 10.15525124 17.08289248 10.09389861

15.99201391 22.90379359 12.47413048 10.54267682 10.66792059 23.82473166

17.23370498 23.10696395 16.91889394 5.89071954 17.46341307 14.86166824

19.8882216 20.86551712 8.13749954 6.2716088 21.58983775 12.29124596

10.05634066 15.25938383 17.11867455 17.35944551 3.76414486 7.15410316

9.20568938 20.18404702 17.67443696 12.67367038 11.26849888 13.29955267

18.1969787 14.21633609 13.86189923 12.01767649 15.7206252 18.99326983

16.90417043 23.70304009 9.40114052 16.17049204 20.366321 4.60191128

7.90538969 5.67012944 20.55981351 15.48768469 12.0049726 13.55364595]

Accuracy of model = 0.8781488502664777

*# Assignment1 Set A Q2. (SLR)*

*# Assignment1 Set A Q3*

# 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

import matplotlib.pyplot as plt

# Reading dataset

dataset = pd.read\_csv("User\_Data.csv")

# Splitting dataset into dependent(Purchase) and independent(Age and estimated salary) variables

x = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

print(x[:10])

print(y[:10])

# Spitting into training and test set

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)

y\_pred=logistic\_regression.predict(x\_test)

# Print the Accuracy and plot the Confusion Matrix

confusion\_matrix = pd.crosstab(y\_test, y\_pred, rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix, annot=True)

plt.show()

print('Accuracy: ',metrics.accuracy\_score(y\_test, y\_pred))

# Print testdata and predicted data

print ("Test Data Values\n",x\_test)

print ("\nPredicted Values\n",y\_pred)

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]]

[0 0 0 0 0 0 0 1 0 0]

Accuracy: 0.68

Test Data Values

[[ 30 87000]

[ 38 50000]

[ 35 75000]

[ 30 79000]

[ 35 50000]

[ 27 20000]

[ 31 15000]

[ 36 144000]

[ 18 68000]

[ 47 43000]

[ 30 49000]

[ 28 55000]

[ 37 55000]

[ 27 84000]

[ 35 20000] …..

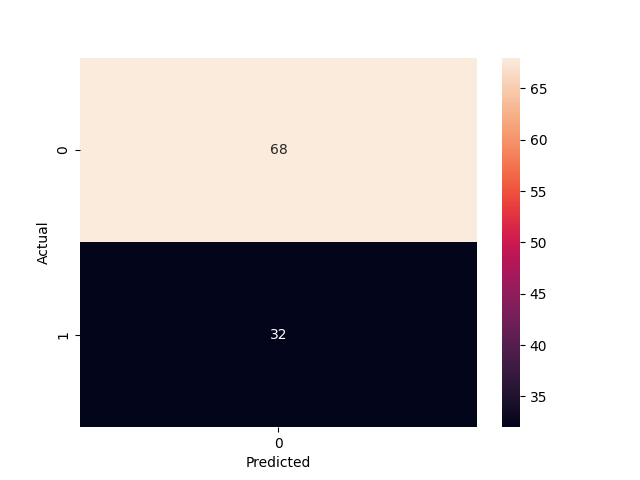
Predicted Values

[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?: [0]



*# Assignment1 SetB Q1*

# Collecting Data

import pandas as pd

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

from matplotlib import pyplot as plt

df = pd.read\_csv("Fish.csv")

# Displaying data

print(df.head(5))

# Identifying Independent and Target variables

X = df[['Length1','Length2','Length3','Height','Width']]

Y = df['Weight']

# X.head()

# 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("\n Training Set of X=\n",x\_train)

print("\n Testing Set of X=\n",x\_test)

print("\n Training Set of Y=\n",y\_train)

print("\n Testing Set of Y=\n",y\_test)

# Creating object of Linear Regression

from sklearn.linear\_model import LinearRegression

clf = LinearRegression()

# fitting the x\_train and y\_train variables.

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

# Predicting output by passing x\_test

pred\_x=clf.predict(x\_test)

print("\n Predicted 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 Width

0 23.2 25.4 30.0 11.5200 4.0200

23 31.8 35.0 40.6 15.4686 6.1306

13 29.5 32.0 37.3 13.9129 5.0728

18 30.9 33.5 38.6 15.6330 5.1338

42 19.4 21.0 23.7 6.1146 3.2943

.. ... ... ... ... ...

127 41.1 44.0 46.6 12.4888 7.5958

97 22.0 24.0 25.5 6.3750 3.8250

45 20.5 22.5 25.3 7.0334 3.8203

41 19.1 20.8 23.1 6.1677 3.3957

25 31.8 35.0 40.9 16.3600 6.0532

[111 rows x 5 columns]

Testing Set of X=

Length1 Length2 Length3 Height Width

28 32.8 36.0 41.6 16.8896 6.1984

114 34.5 37.0 39.4 10.8350 6.2646

34 38.0 41.0 46.5 17.6235 6.3705

142 56.0 60.0 64.0 9.6000 6.1440

148 10.4 11.0 12.0 2.1960 1.3800

70 23.0 25.0 28.0 11.0880 4.1440

155 11.7 12.4 13.5 2.4300 1.2690

4 26.5 29.0 34.0 12.4440 5.1340

125 40.1 43.0 45.5 12.5125 7.4165

130 32.7 35.0 38.8 5.9364 4.3844

15 29.4 32.0 37.2 15.4380 5.5800

5 26.8 29.7 34.7 13.6024 4.9274

90 20.0 22.0 23.5 5.5225 3.9950

118 36.6 39.0 41.3 12.4313 7.3514

51 23.6 25.2 27.9 7.0866 3.9060

149 10.7 11.2 12.4 2.0832 1.2772

19 31.0 33.5 38.7 14.4738 5.7276 …

Training Set of Y=

0 242.0

23 680.0

13 340.0

18 610.0

42 120.0

...

127 1000.0

97 145.0

45 160.0

41 110.0

25 725.0

Name: Weight, Length: 111, dtype: float64

Testing Set of Y=

28 850.0

114 700.0

34 950.0

142 1600.0

148 9.7

70 273.0

144 1650.0

7 390.0

31 955.0

2 340.0

135 510.0

105 250.0

54 390.0

137 500.0

51 180.0

106 250.0

Name: Weight, dtype: float64

Predicted Values of x = [ 718.15845189 707.88703496 876.43379894 1052.25429245 -170.91041783

364.75872929 1180.82741174 471.33365784 792.08344837 378.7220852

631.46307632 370.53648159 510.00625097 676.54643684 -88.44732471

597.30801717 929.85026923 675.01260387 -173.90201759 -147.2093725

455.65357439 911.35285695 428.77883599 612.03156531 464.25122685

176.87389771 854.9795413 288.26248985 -171.48143404 634.79847223

389.49591612 547.3790247 256.95293373 -195.86936319 434.19688632

37.89464618 299.24913193 1034.76674338 234.43809375 580.54028895…

229.3078826 230.42740327 379.61223022]

*# Assignment1 Set B Q2*

# 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

import matplotlib.pyplot as plt

# Reading dataset

data = pd.read\_csv("Iris.csv")

print('Iris-setosa')

setosa = data['Species'] == 'Iris-setosa'

print(data[setosa].describe())

print('\nIris-versicolor')

setosa = data['Species'] == 'Iris-versicolor'

print(data[setosa].describe())

print('\nIris-virginica')

setosa = data['Species'] == 'Iris-virginica'

print(data[setosa].describe())

# Splitting dataset into dependent(Purchase) and independent(Age and estimated salary) variables

x = data[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]

y = data['Species']

print(x[:10])

print(y[:10])

# Spitting into training and test set

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)

y\_pred=logistic\_regression.predict(x\_test)

# Print the Accuracy and plot the Confusion Matrix

confusion\_matrix = pd.crosstab(y\_test, y\_pred, rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix, annot=True)

print('Accuracy: ',metrics.accuracy\_score(y\_test, y\_pred))

plt.show()

# Print testdata and predicted data

print ("Test Data Values:\n",x\_test)

print ("\n Predicted values:\n",y\_pred)

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-versicolor

Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm

count 50.00000 50.000000 50.000000 50.000000 50.000000

mean 75.50000 5.936000 2.770000 4.260000 1.326000

std 14.57738 0.516171 0.313798 0.469911 0.197753

min 51.00000 4.900000 2.000000 3.000000 1.000000

25% 63.25000 5.600000 2.525000 4.000000 1.200000

50% 75.50000 5.900000 2.800000 4.350000 1.300000

75% 87.75000 6.300000 3.000000 4.600000 1.500000

max 100.00000 7.000000 3.400000 5.100000 1.800000

Iris-virginica

Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm

count 50.00000 50.00000 50.000000 50.000000 50.00000

mean 125.50000 6.58800 2.974000 5.552000 2.02600

std 14.57738 0.63588 0.322497 0.551895 0.27465

min 101.00000 4.90000 2.200000 4.500000 1.40000

25% 113.25000 6.22500 2.800000 5.100000 1.80000

50% 125.50000 6.50000 3.000000 5.550000 2.00000

75% 137.75000 6.90000 3.175000 5.875000 2.30000

max 150.00000 7.90000 3.800000 6.900000 2.50000

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

0 Iris-setosa

1 Iris-setosa

2 Iris-setosa

3 Iris-setosa

4 Iris-setosa

5 Iris-setosa

6 Iris-setosa

7 Iris-setosa

8 Iris-setosa

9 Iris-setosa

Name: Species, dtype: object

Accuracy: 0.9736842105263158

Test Data Values:

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

0.1

78 6.0 2.9 4.5 1.5

Predicted values:

['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-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'

'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'

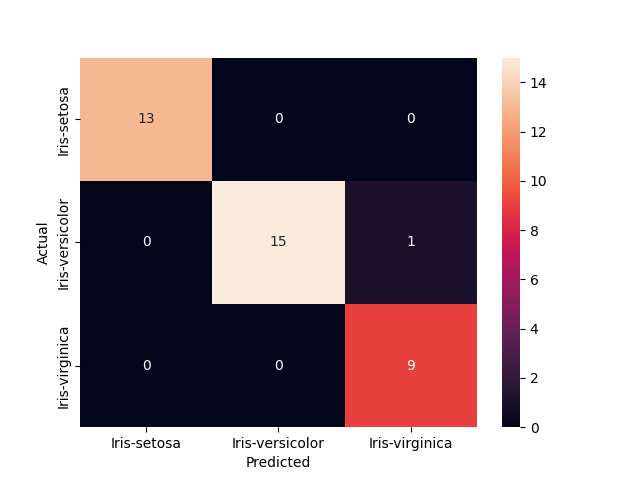
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'

'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'

'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'

'Iris-setosa' 'Iris-virginica']

Predicted Species: ['Iris-virginica']



*# Assignment 2 SET A Q1*

# Import the libraries

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 the TransactionEncoder 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("Encoded Data:\n",df)

# Find the frequent itemsets

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print("\n Frequent Itemset:\n",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("\n Association Rules:\n",rules)

Output ::

Encoded Data:

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

Frequent Itemset:

support itemsets

0 0.6 (Beer)

1 0.8 ( Bread)

2 0.8 (Diaper)

3 0.8 (Milk)

4 0.6 (Diaper, Beer)

5 0.6 (Diaper, Bread)

6 0.6 (Bread, Milk)

7 0.6 (Diaper, Milk)

Association Rules:

antecedents consequents antecedent support consequent support support \

1 (Beer) (Diaper) 0.6 0.8 0.6

0 (Diaper) (Beer) 0.8 0.6 0.6

2 (Diaper) (Bread) 0.8 0.8 0.6

3 (Bread) (Diaper) 0.8 0.8 0.6

4 (Bread) (Milk) 0.8 0.8 0.6

5 (Milk) (Bread) 0.8 0.8 0.6

6 (Diaper) (Milk) 0.8 0.8 0.6

7 (Milk) (Diaper) 0.8 0.8 0.6

confidence lift leverage conviction

1 1.00 1.2500 0.12 inf

0 0.75 1.2500 0.12 1.6

2 0.75 0.9375 -0.04 0.8

3 0.75 0.9375 -0.04 0.8

4 0.75 0.9375 -0.04 0.8

5 0.75 0.9375 -0.04 0.8

6 0.75 0.9375 -0.04 0.8

7 0.75 0.9375 -0.04 0.8

*# Assignment 2 SET A Q2*

# Import the libraries

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'],['Bread','Milk','Beer'],['Bread','Milk','Apple','Coke']]

# transform it into the right format via the TransactionEncoder 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)

# Find the frequent itemsets

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print("Frequent Itemsets are:\n",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("\n Association Rules are:\n",rules)

Output ::

Apple Beer Bread Coke Eggs Milk

0 False False True False False True

1 True True True False True False

2 True True False False False True

3 False True True False False True

4 True False True True False True

Frequent Itemsets are:

support itemsets

0 0.6 (Apple)

1 0.6 (Beer)

2 0.8 (Bread)

3 0.8 (Milk)

4 0.6 (Bread, Milk)

Association Rules are:

antecedents consequents antecedent support consequent support support \

0 (Bread) (Milk) 0.8 0.8 0.6

1 (Milk) (Bread) 0.8 0.8 0.6

confidence lift leverage conviction

0 0.75 0.9375 -0.04 0.8

1 0.75 0.9375 -0.04 0.8

*# Assignment 2 SET B Q1*

# Import the libraries

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))

# Preprocessing data dropping NULL values

data=data.dropna()

data.info()

# Using positive Quantity 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)

# Encode data

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

basket\_encode\_plus=basket\_plus.applymap(encode\_units)

print("\n -----------Encoded Basket------------\n",basket\_encode\_plus)

# Filter data

basket\_filter\_plus=basket\_encode\_plus[(basket\_encode\_plus>0).sum(axis=1)>=2]

print("\n -----------Filtered Basket------------\n",basket\_filter\_plus)

# Find the frequent itemsets

freq\_items = apriori(basket\_filter\_plus, min\_support = 0.03, use\_colnames = True)

print("--------Frequent Items---------\n",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--------",rules)

Output::

InvoiceNo StockCode Description Quantity \

0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6

1 536365 71053 WHITE METAL LANTERN 6

2 536365 84406B CREAM CUPID HEARTS COAT HANGER 8

3 536365 84029G KNITTED UNION FLAG HOT WATER BOTTLE 6

4 536365 84029E RED WOOLLY HOTTIE WHITE HEART. 6

InvoiceDate UnitPrice CustomerID Country

0 01-12-2010 08:26 2.55 17850.0 United Kingdom

1 01-12-2010 08:26 3.39 17850.0 United Kingdom

2 01-12-2010 08:26 2.75 17850.0 United Kingdom

3 01-12-2010 08:26 3.39 17850.0 United Kingdom

4 01-12-2010 08:26 3.39 17850.0 United Kingdom

<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

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

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 50'S CHRISTMAS GIFT BAG LARGE \

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

Description DOLLY GIRL BEAKER I LOVE LONDON MINI BACKPACK \

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

... ... ...

581586 0.0 0.0

Description NINE DRAWER OFFICE TIDY OVAL WALL MIRROR DIAMANTE \

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

Description RED SPOT GIFT BAG LARGE SET 2 TEA TOWELS I LOVE LONDON \

InvoiceNo

536365 0.0 0.0

536366 0.0 0.0

536367 0.0 0.0

... ... ...

581582 0.0 0.0

581583 0.0 0.0

581584 0.0 0.0

Description SPACEBOY BABY GIFT SET TOADSTOOL BEDSIDE LIGHT ... \

InvoiceNo ...

536365 0.0 0.0 ...

536369 0.0 0.0 ...

... ... ... ...

581582 0.0 0.0 ...

581585 0.0 0.0 ...

581586 0.0 0.0 ...

Description ZINC STAR T-LIGHT HOLDER ZINC SWEETHEART SOAP DISH \

InvoiceNo

536365 0.0 0.0

536368 0.0 0.0

536369 0.0 0.0

... ... ...

581584 0.0 0.0

581585 0.0 0.0

581586 0.0 0.0

Description ZINC SWEETHEART WIRE LETTER RACK ZINC T-LIGHT HOLDER STAR LARGE \

InvoiceNo

536365 0.0 0.0

536366 0.0 0.0

536367 0.0 0.0

581586 0.0 0.0

Description ZINC T-LIGHT HOLDER STARS LARGE ZINC T-LIGHT HOLDER STARS SMALL \

InvoiceNo

536365 0.0 0.0

536366 0.0 0.0

536367 0.0 0.0

536369 0.0 0.0

... ... ...

581582 0.0 0.0

581583 0.0 0.0

Description ZINC TOP 2 DOOR WOODEN SHELF ZINC WILLIE WINKIE CANDLE STICK \

InvoiceNo

536365 0.0 0.0

536366 0.0 0.0

... ... ...

581582 0.0 0.0

581583 0.0 0.0

Description ZINC WIRE KITCHEN ORGANISER 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

[16649 rows x 3844 columns]

*# Assignment 2 SET B Q2*

# 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 ::

----------- Dataset-------------

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 abrasive cleaner artif. sweetener \

0 False False False False

1 False False False False

2 False False False False

3 False False False False

4 False False False False

baby cosmetics bags baking powder bathroom cleaner beef berries \

0 False False False False False False

1 False False False False False False

2 False False False False False False

3 False False False False False False

4 False False False False False False

... turkey vinegar waffles whipped/sour cream whisky white bread \

0 ... False False False False False False

1 ... False False False False False False

2 ... False False False False False False

3 ... False False False False False False

4 ... False False False False False False

white wine whole milk yogurt zwieback

0 False True True False

1 False True False False

2 False False False False

3 False False False False

4 False False False False

[5 rows x 167 columns]

-------------Frequent Itemsets------------------

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 (whole milk, yogurt)

1777 0.000468 (whole milk, zwieback)

[1778 rows x 2 columns]

------------ Assoiation Rules-------------

antecedents consequents antecedent support \

0 (UHT-milk) (butter milk) 0.021386

1 (butter milk) (UHT-milk) 0.017577

2 (UHT-milk) (cream cheese ) 0.021386

3 (cream cheese ) (UHT-milk) 0.023658

4 (soda) (artif. sweetener) 0.097106

consequent support support confidence lift leverage conviction

0 0.017577 0.000601 0.028125 1.600131 0.000226 1.010854

1 0.021386 0.000601 0.034221 1.600131 0.000226 1.013289

2 0.023658 0.000869 0.040625 1.717152 0.000363 1.017685

3 0.021386 0.000869 0.036723 1.717152 0.000363 1.015922

4 0.001938 0.000468 0.004818 2.485725 0.000280 1.002893

Rules identified: 190