**Market Basket Analysis with Association Rule Learning and Data Analysis**

**BATCH MEMBERS**

**Abinaya.K(abinayak2020@gmail.com)**

**Dhanalakshmi.S(dhanalakshmisekar26@gmail.com)**

**Kokila.M(kokila071004@gmail.com)**

**Mahalakshmi.R(**[**maha606105@gmail.com**](mailto:maha606105@gmail.com)**)**

**DATASETLINK:** **https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis**

**1) Import Libraries**

import pandas as pd

import numpy as np

from mlxtend.frequent\_patterns import apriori, association\_rules

import plotly.express as px

**2) Data Pre-processing**

df\_ = pd.read\_csv("../input/market-basket-analysis/Assignment-1\_Data.csv", sep = ";")

df = df\_.copy()

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3444: DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or set low\_memory=False.

exec(code\_obj, self.user\_global\_ns, self.user\_ns)

df.head(10)

def check\_df(dataframe, head=5):

print("##################### Shape #####################")

print(dataframe.shape)

print("##################### Types #####################")

print(dataframe.dtypes)

print("##################### Head #####################")

print(dataframe.head(head))

print("##################### Tail #####################")

print(dataframe.tail(head))

print("##################### NA #####################")

print(dataframe.isnull().sum())

check\_df(df)

##################### Shape #####################

(522064, 7)

##################### Types #####################

BillNo object

Itemname object

Quantity int64

Date object

Price object

CustomerID float64

Country object

dtype: object

##################### Head #####################

BillNo Itemname Quantity Date \

0 536365 WHITE HANGING HEART T-LIGHT HOLDER 6 01.12.2010 08:26

1 536365 WHITE METAL LANTERN 6 01.12.2010 08:26

2 536365 CREAM CUPID HEARTS COAT HANGER 8 01.12.2010 08:26

3 536365 KNITTED UNION FLAG HOT WATER BOTTLE 6 01.12.2010 08:26

4 536365 RED WOOLLY HOTTIE WHITE HEART. 6 01.12.2010 08:26

Price CustomerID Country

0 2,55 17850.0 United Kingdom

1 3,39 17850.0 United Kingdom

2 2,75 17850.0 United Kingdom

3 3,39 17850.0 United Kingdom

4 3,39 17850.0 United Kingdom

##################### Tail #####################

BillNo Itemname Quantity Date \

522059 581587 PACK OF 20 SPACEBOY NAPKINS 12 09.12.2011 12:50

522060 581587 CHILDREN'S APRON DOLLY GIRL 6 09.12.2011 12:50

522061 581587 CHILDRENS CUTLERY DOLLY GIRL 4 09.12.2011 12:50

522062 581587 CHILDRENS CUTLERY CIRCUS PARADE 4 09.12.2011 12:50

522063 581587 BAKING SET 9 PIECE RETROSPOT 3 09.12.2011 12:50

Price CustomerID Country

522059 0,85 12680.0 France

522060 2,1 12680.0 France

522061 4,15 12680.0 France

522062 4,15 12680.0 France

522063 4,95 12680.0 France

##################### NA #####################

BillNo 0

Itemname 1455

Quantity 0

Date 0

Price 0

CustomerID 134041

Country 0

dtype: int64

# Drop na values

df.dropna(inplace=True)

# Quantity and Price should be greater than 0

df = df[df["Quantity"] > 0]

# We have to change the price column datatype as a numeric

df ['Price'] = pd.to\_numeric(df['Price'], errors='coerce')

df = df[df["Price"] > 0]

check\_df(df)

##################### Shape #####################

(1537, 7)

##################### Types #####################

BillNo object

Itemname object

Quantity int64

Date object

Price float64

CustomerID float64

Country object

dtype: object

##################### Head #####################

BillNo Itemname Quantity Date \

45 536370 POSTAGE 3 01.12.2010 08:45

237 536392 RUSTIC SEVENTEEN DRAWER SIDEBOARD 1 01.12.2010 10:29

377 536403 POSTAGE 1 01.12.2010 11:27

1113 536527 POSTAGE 1 01.12.2010 13:04

4348 536779 Bank Charges 1 02.12.2010 15:08

Price CustomerID Country

45 18.0 12583.0 France

237 165.0 13705.0 United Kingdom

377 15.0 12791.0 Netherlands

1113 18.0 12662.0 Germany

4348 15.0 15823.0 United Kingdom

##################### Tail #####################

BillNo Itemname Quantity Date Price CustomerID \

521357 581493 POSTAGE 1 09.12.2011 10:10 15.0 12423.0

521375 581494 POSTAGE 2 09.12.2011 10:13 18.0 12518.0

521885 581570 POSTAGE 1 09.12.2011 11:59 18.0 12662.0

521922 581574 POSTAGE 2 09.12.2011 12:09 18.0 12526.0

521923 581578 POSTAGE 3 09.12.2011 12:16 18.0 12713.0

Country

521357 Belgium

521375 Germany

521885 Germany

521922 Germany

521923 Germany

##################### NA #####################

BillNo 0

Itemname 0

Quantity 0

Date 0

Price 0

CustomerID 0

Country 0

dtype: int64

**3) Exploratory Data Analysis and Some Visualizations**

total\_sales = df

total\_sales["Total\_Price"] = total\_sales["Price"] \* total\_sales["Quantity"]

#total\_sales.columns

total\_sales\_per\_customer = total\_sales.groupby(["CustomerID", "Country"]).agg({"Total\_Price": "sum"})

total\_sales\_per\_customer.head(10)

**Top 10 Shoppers and Their Coutries**

total\_sales\_per\_customer.reset\_index(inplace=True)

total\_sales\_per\_customer.sort\_values(by = "Total\_Price", ascending = False).head(10)

# consider that for all time period

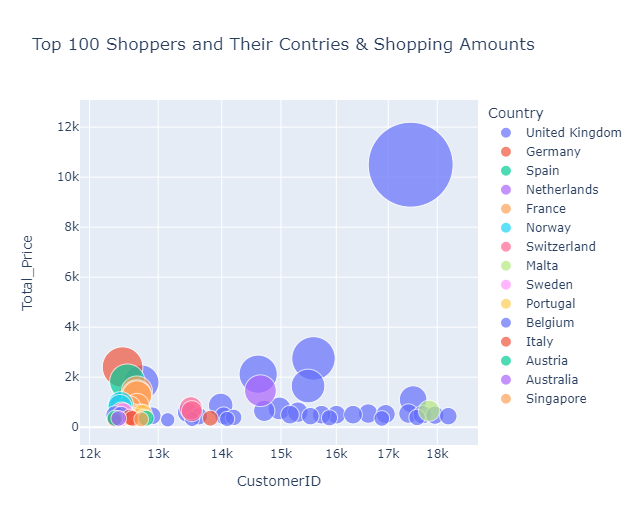
data\_fig = total\_sales\_per\_customer.sort\_values(by = "Total\_Price", ascending = False).head(100)

fig = px.scatter(data\_fig, x="CustomerID", y="Total\_Price",

size="Total\_Price", color="Country",

hover\_name="Country", log\_x=True, size\_max=60, title="Top 100 Shoppers and Their Contries & Shopping Amounts")

fig.show()



# consider that for all time period

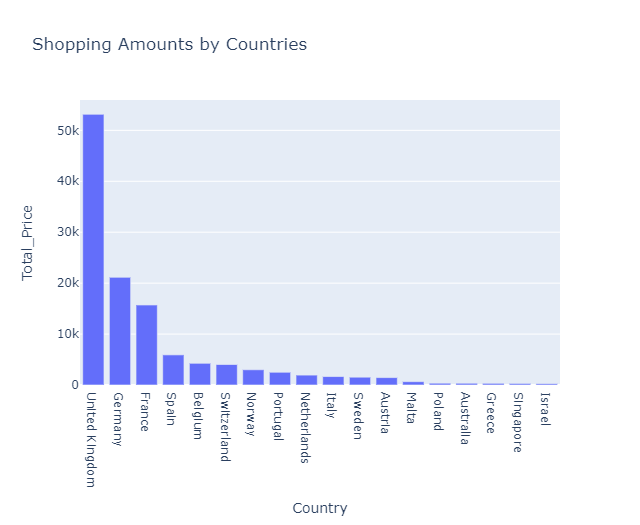
#total\_sales\_per\_customer.head(20)

total\_sales\_per\_customer.groupby(["Country"]).agg({"Total\_Price":"sum"}).reset\_index().sort\_values(by="Total\_Price", ascending=False )

data = total\_sales\_per\_customer.groupby(["Country"]).agg({"Total\_Price":"sum"}).reset\_index().sort\_values(by="Total\_Price", ascending=False )

fig = px.bar(data, x='Country', y='Total\_Price' , title = "Shopping Amounts by Countries")

fig.show()



**4) Invoice Product Matrix**

#df\_united\_kingdom = df.loc[df["Country"]=="United Kingdom"]

df\_invoice\_product\_matrix = df.groupby(['BillNo', 'Itemname']). \

agg({"Quantity": "sum"}).unstack().fillna(0). \

applymap(lambda x: 1 if x > 0 else 0)

df\_invoice\_product\_matrix.head(10)

5) Assosiciation Rules Learning

frequent\_itemsets = apriori(df\_invoice\_product\_matrix, min\_support=0.001, use\_colnames=True)

frequent\_itemsets.sort\_values("support", ascending=False)

rules = association\_rules(frequent\_itemsets, metric="support", min\_threshold=0.001)

rules.sort\_values("support", ascending=False).head(10)

sorted\_rules = rules.sort\_values("lift", ascending=False)

# We can try below products in the loop

# ('Quantity', 'VANILLA SCENT CANDLE JEWELLED BOX')

# ('Quantity', 'DOORMAT MERRY CHRISTMAS RED')

# ('Quantity', 'DOORMAT RESPECTABLE HOUSE')

# ('Quantity', 'DOORMAT SPOTTY HOME SWEET HOME')

# ('Quantity', 'DOORMAT UNION FLAG')

# ('Quantity', 'DOORMAT UNION FLAG')

recommendation\_list = []

for i, product in sorted\_rules["antecedents"].items():

for j in list(product):

if j == ('Quantity', 'MARIE ANTOINETTE TRINKET BOX GOLD'):

recommendation\_list.append(list(sorted\_rules.iloc[i]["consequents"]))