**CHANDIGARH UNIVERSITY**

**UNIVERSITY INSTITUTE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Kanishk Soni Submitted To: Er. Sudhanshu Sharma** | |
| **Subject Name** | **Machine Learning Lab** |
| **Subject Code** | **20CSP-317** |
| **Branch** | **BE-CSE** |
| **Semester** | **5th** |

**EXPERIMENT-10**

**Student’s Name: Kanishk Soni UID: 20BCS9398**

**Section/Group: 707\_WM\_B Subject Code: 20CSP-317**

**Subject Name: ML Lab Date of performance:9/11/2022**

**Branch: BE CSE Semester:5th**

**Aim:** Implement Apriori Algorithm.

**Objective:** To do Apriori Algorithm on data set.

**Software/Hardware Requirements:** Windows 7 & above version.

**Tools to be used:**

1. Anaconda Jupyter Notebook,
2. numpy, matplotlib.

**Introduction to Apriori Algorithm:**

Apriori Algorithm is a Machine Learning algorithm which is used to gain insight into the structured relationships between different items involved. The most prominent practical application of the algorithm is to recommend products based on the products already present in the user’s cart. Walmart especially has made great use of the algorithm in suggesting products to it’s users.

**Code:**

--

import numpy as np

import pandas as pd

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

# Loading the Data

data = pd.read\_excel('Online Retail.xlsx')

data.head()

# Exploring the columns of the data

data.columns

# Exploring the different regions of transactions

data.Country.unique()

# Stripping extra spaces in the description

data['Description'] = data['Description'].str.strip()

# Dropping the rows without any invoice number

data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

data['InvoiceNo'] = data['InvoiceNo'].astype('str')

# Dropping all transactions which were done on credit

data = data[~data['InvoiceNo'].str.contains('C')]

# Transactions done in France

basket\_France = (data[data['Country'] =="France"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

# Transactions done in the United Kingdom

basket\_UK = (data[data['Country'] =="United Kingdom"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

# Transactions done in Portugal

basket\_Por = (data[data['Country'] =="Portugal"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

basket\_Sweden = (data[data['Country'] =="Sweden"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

# Defining the hot encoding function to make the data suitable

# for the concerned libraries

def hot\_encode(x):

if(x<= 0):

return 0

if(x>= 1):

return 1

# Encoding the datasets

basket\_encoded = basket\_France.applymap(hot\_encode)

basket\_France = basket\_encoded

basket\_encoded = basket\_UK.applymap(hot\_encode)

basket\_UK = basket\_encoded

basket\_encoded = basket\_Por.applymap(hot\_encode)

basket\_Por = basket\_encoded

basket\_encoded = basket\_Sweden.applymap(hot\_encode)

basket\_Sweden = basket\_encoded

# Building the model

frq\_items = apriori(basket\_France, min\_support = 0.05, use\_colnames = True)

# Collecting the inferred rules in a dataframe

rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

print(rules.head())

frq\_items = apriori(basket\_Por, min\_support = 0.05, use\_colnames = True)

rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

print(rules.head())

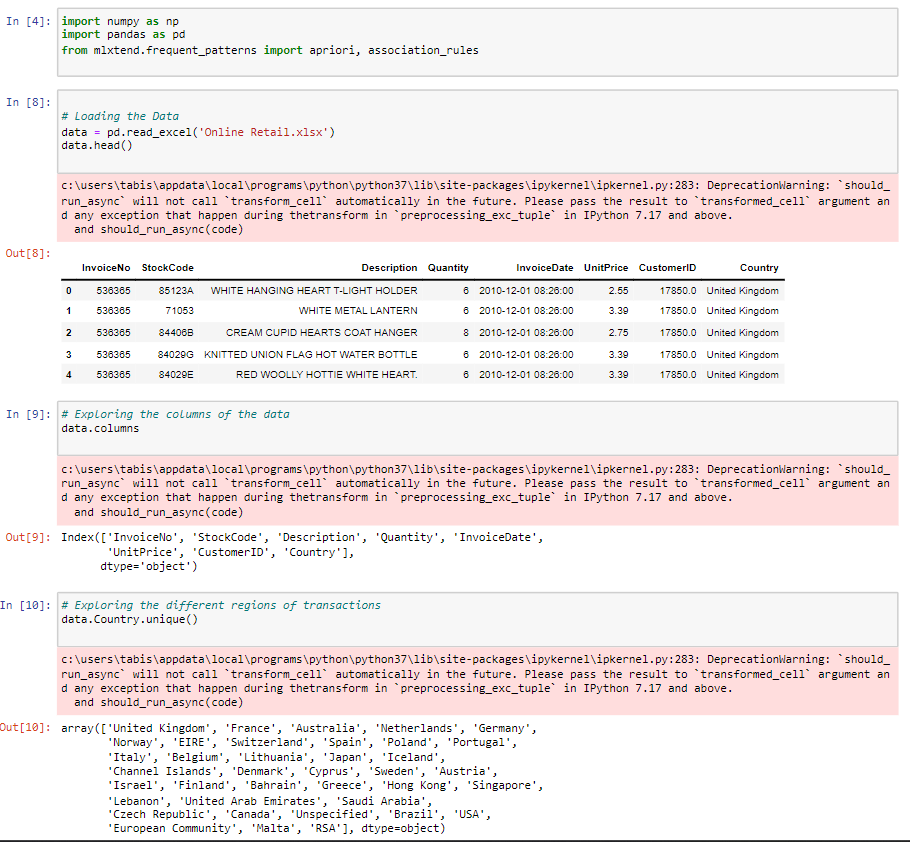
frq\_items = apriori(basket\_Sweden, min\_support = 0.05, use\_colnames = True)

rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

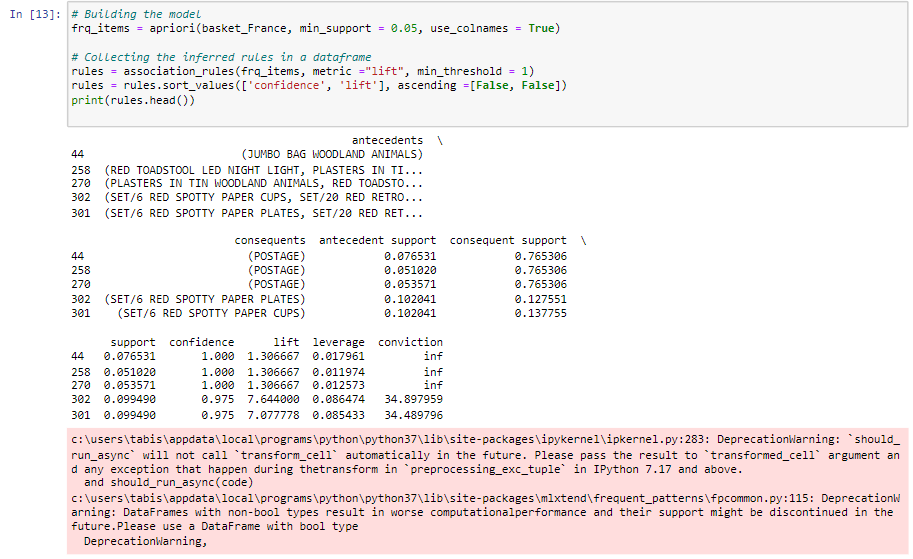
print(rules.head())

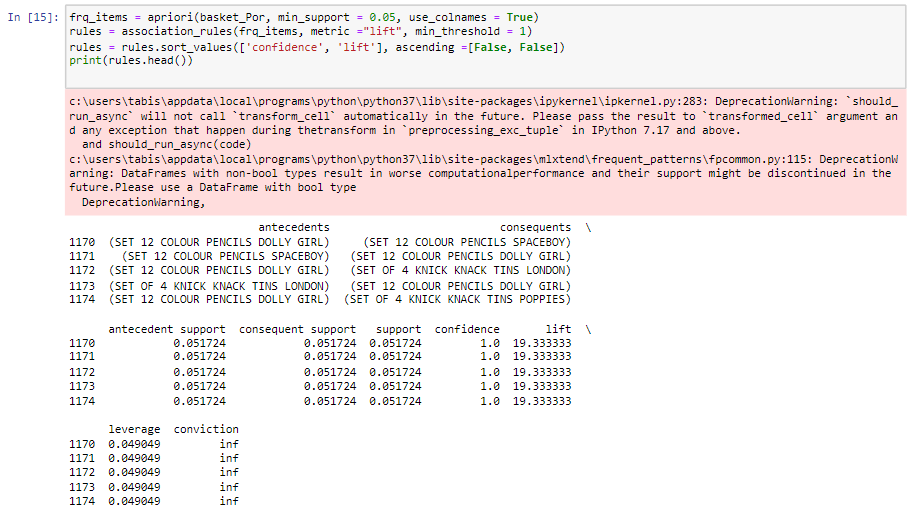
**Output:**

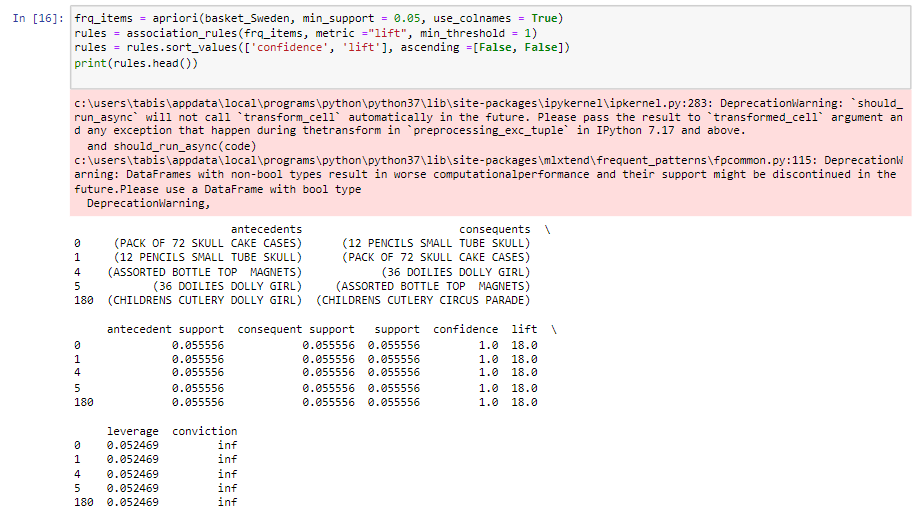
****

****

****

****

****

****