Project Report

NAME: HARI KRISHNAN

COURSE: AI and ML

Association Rule Mining: Market Basket Analysis

Question:

Apriori is a statistical algorithm for implementing associate rule mining, that primarily relies on three components: Life, Support and Confidence. Using this algorithm try to find the rules that describe the relation between each of the products that were brought by the customers as described in

Prerequisites

What things you need to install the software and how to install them:

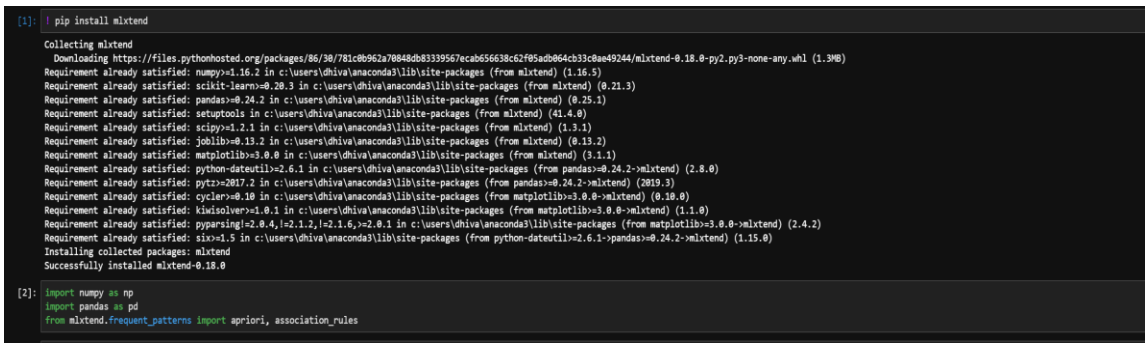
Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-asan-internal-or-externalcommand/ . Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6

Dataset Link: Store Data

https://drive.google.com/file/d/1y5DYn0dGoSbC22xowBq2d4po6h1JxcTQ/view?usp=sharing

Importing library and dataset:



Read the dataset & Using the model

```
[8]: df = pd.read_csv("BreadBasket_DMS.csv")
     df.shape

[8]: (21293, 4)

[9]: df.columns

[9]: Index(['Date', 'Time', 'Transaction', 'Item'], dtype='object')

[10]: df1 = df.loc[df["Item"] == "None", :]

[11]: hot_encoded_df = df.groupby(["Transaction", "Item"])["Item"].count().unstack().reset_index().fillna(0).set_index("Transaction")

[12]: def encode_units(x):
          if x <= 0:
              return 0
          else:
              return 1

[13]: hot_encoded_df = hot_encoded_df.applymap(encode_units)
      hot_encoded_df.shape

[13]: (9531, 95)

[14]: freq_itemsets = apriori(hot_encoded_df, min_support = 0.005, use_colnames = True)
      freq_itemsets
```

[14]:

	support	itemsets
0	0.036093	(Alfajores)
1	0.015948	(Baguette)
2	0.005036	(Bakewell)
3	0.324940	(Bread)
4	0.039765	(Brownie)
...	...	...
120	0.006820	(Hot chocolate, Cake, Coffee)
121	0.009967	(Tea, Cake, Coffee)
122	0.005141	(NONE, Sandwich, Coffee)
123	0.005876	(NONE, Tea, Coffee)
124	0.005351	(Tea, Sandwich, Coffee)

125 rows × 2 columns

Output

```
[15]: rules = association_rules(freq_itemsets, metric = "confidence", min_threshold = 0.4)
      rules.shape

[15]: (27, 9)

[16]: rules[(rules["lift"] > 1.1) & (rules["confidence"] > 0.5) & (rules["support"] > 0.02)]
```

[16]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2	(Cake)	(Coffee)	0.103137	0.475081	0.054349	0.526958	1.109196	0.005350	1.109667
7	(Juice)	(Coffee)	0.038296	0.475081	0.020460	0.534247	1.124537	0.002266	1.127031
9	(Medialuna)	(Coffee)	0.061379	0.475081	0.034939	0.569231	1.198175	0.005779	1.218561
11	(NONE)	(Coffee)	0.079005	0.475081	0.042073	0.532537	1.120938	0.004539	1.122908
12	(Pastry)	(Coffee)	0.085510	0.475081	0.047214	0.552147	1.162216	0.006590	1.172079
14	(Sandwich)	(Coffee)	0.071346	0.475081	0.037981	0.532353	1.120551	0.004086	1.122468
19	(Toast)	(Coffee)	0.033365	0.475081	0.023502	0.704403	1.482699	0.007651	1.775789