Report

Chen Chen 21202636

Task 2

Q1 - Outline and discuss market basket analysis.

Market basket analysis is a data mining technique used to analyze the correlation relationships or associations between the items that were purchased by customers. It could help uncover customers' purchasing patterns - which items are frequently bought together, such as when customers buy milk, they are very likely to buy bread as well, so it would be very helpful for retailers to decide on their marketing strategies and decide where to place certain items in their stores, and eventually it will boost sales and increase cross-selling.

The process of analyzing the correlation relationships or the associations of items is called the Association Rule Learning. It usually takes historical transaction data (barcodes scanned at the till) as inputs, and outputs certain rules which satisfy certain metrics. There are many algorithms developed for association rule learning, such as AIS, SETM, Apriori, FP Growth. Of which, Apriori is the most widely used and well-known algorithm in market basket analysis.

Q2 - Compare and contrast the Apriori and FP-Growth algorithms.

Apriori is a well-known algorithm for Association Rule Learning. It takes two main steps: first it generates all the frequent itemsets, and then it generates all the possible rules from the frequent itemsets. However, it is not so efficient in generating frequent itemsets because it needs to generate all the candidates for the frequent itemsets first and discard those which don't exceed the threshold, and it needs to scan the database multiple times (scan the first time to generate all the frequent itemsets for one itemset, and then scan the second time to generate all the frequent itemsets for two itemsets based on the one itemset, and so on). So it's computationally costly if it's a large database. And once it generates all the frequent itemsets, it uses the brute force method to mine them in order to generate all the possible rules.

FP-Growth (Frequent Pattern Growth) is an improved version of the Apriori algorithm which is widely used for Association Rule Learning as well. It doesn't need to generate all the candidates for the frequent itemsets, and it only scans the database twice (scan the first time to get the number of occurrences of each item, scan the second time to sort the items by its occurrence in descending order) and stores the data using a tree structure (FP-tree). Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets.

In summary, the main differences between Apriori and FP-Growth are the following.

Apriori: 1, candidate generation for frequent itemsets
2, brute force method for rule generation - all possible itemsets
3, multiple scanning of database
4, slow for large databases

FP-Growth: 1, no candidate generation
2, depth first search for rule generation - divide and conquer
3, two scanning of database
4, fast for large databases

Q3 - Code Overview
Using snippets of your code, outline the code you wrote for Section 1 of the *submission.ipynb* notebook instructions including how you:
- generated the frequent itemsets
- generated the association rules
- selected for those rules
Your outline should not repeat the code, it should describe what you are doing, and why.

“””
```
encoder = TransactionEncoder().fit(transaction_list)
onehot = encoder.transform(transaction_list).astype("int")
```
“””
These two lines of code are used to fit and transform the transaction_list into one-hot encoding.

“””
```
df_onehot = pd.DataFrame(onehot, columns = encoder.columns_).drop("nan",
axis = 1)
```
“””
This line of code uses the transformed transaction_list and makes it into a dataframe, with column names as all the unique itemset. And it also drops the column which contains “nan” values.

“””

```
freq_itemsets = apriori(df_onehot, min_support = 0.005, use_colnames =
True)
```

“””

This line of code is used to generate the frequent itemsets. It uses the apriori function in mlxtend.frequent_patterns package. The parameters for this function are the dataset, the minimum support value and if it uses the column names, in which I set the dataset to be df_onehot, the minimum support value to be 0.5%, and use_colnames to be True.

The features of this dataset are transformed into one-hot encoding, which can be used directly by Apriori algorithm. The transformed dataset is named as df_onehot.

The minimum support value is set to be 0.5% because I followed the instructions, but it’s also a good threshold for a medium-sized dataset. I also tried setting up with different thresholds, and the result came out in the following:

- 0.1% - 762 frequent itemsets
- 0.5% - 133 frequent itemsets
- 1% - 68 frequent itemsets
- 10% - 4 frequent itemsets

As we can see, if the threshold is set up too high, there would be too few frequent itemsets, and probably we won’t be able to find a useful pattern out of them. Likewise, if the threshold is set up too low, there would be too many frequent itemsets, and it would become too time consuming to find out useful patterns. So the support threshold of 0.5% is adequate for a medium-sized dataset.

“””

```
rules_df = association_rules(freq_itemsets, metric = 'confidence',
min_threshold = 0.1)
```

“””

This line of code is used to generate the association rules. It uses the association_rules function in mlxtend.frequent_patterns package. The parameters for this function are the frequent itemsets, the metric for generating the rules and the threshold for the chosen metric, in which the frequent itemsets are the one generated by the last line of code, the metric is chosen to be confidence, and the threshold of confidence is set to be 10%.

The minimum confidence value is set to be 10% because I want to make sure that the useful rules should at least provide the information that when customers buy the antecedent, there will be at least 10% of the chance that they will also buy the consequent.

“””
print(rules_df.sort_values(by = ['lift'], ascending = False).drop(columns=['antecedent support', 'consequent support', 'conviction']))
“””

This line of code is used to select the rules. rules_df is the dataframe that contains all the association rules generated by the last line of code. And I used the pandas sort_values function to sort the dataframe by the lift value and make it show in descending order. Also, I dropped three useless columns to make the result clearer.

I sorted the result by the lift value because lift is a very important metric along with confidence. When the confidence is high, the strong association between the antecedent and the consequent may happen by chance, but a high lift value can prove that their association doesn’t happen by chance.

Q4 - Result Interpretation
Show some of the rules (up to 10) you have generated including the antecedents, consequents, support, confidence, lift, and leverage for each rule. Interpret the results. Consider which rules may be useful and what could be a good minimum support for this dataset.

Result of the rules (part):

```
          antecedents        consequents   support  confidence     l
ift  \
21            (eggs)       (whole milk)  0.015520    0.312169  1.633
464
25   (root vegetables)  (other vegetables)  0.013087    0.171997  1.
337790
26  (other vegetables)   (root vegetables)  0.013087    0.101790  1.
337790
28      (white bread)  (other vegetables)  0.006116    0.171587  1.3
34602
9           (yogurt)          (berries)  0.012429    0.140000  1.203
414
10          (berries)           (yogurt)  0.012429    0.106840  1.203
414

    leverage
21  0.006019
25  0.003304
```

```
26  0.003304
28  0.001533
9   0.002101
10  0.002101
```

I selected 6 rules and they're shown in descending order of lift. The result shows the associations between the antecedents and the consequents. And their associations are determined by 4 metrics - support, confidence, lift and leverage. Support shows the percentage of transactions (containing both the antecedents and the consequents) covered, confidence shows that when customers buy the antecedents, how likely are they going to buy the consequents, lift can prove that the association level tested by confidence doesn't happen by chance if its value is larger than 1, and leverage shows how much more frequently items appear together than independently, if leverage = 0, then the antecedent and the consequent are independent. We usually focus on support, confidence and either lift or leverage.

Some of the rules are useful, such as (eggs) → (whole milk), with lift = 1.633, indicating that their association doesn't happen by chance, and confidence = 31.22%, which is relatively high, and support = 1.55%, meaning that 1.55% of the transactions are impacted. Thus, this rule can be used for marketing strategies to increase cross-selling.

Besides that, (root vegetables) → (other vegetables) and (other vegetables) → (root vegetables) also show relatively high associations. But since these are all vegetables, and they're usually placed together, they don't appear to be very useful information.

(white bread) → (other vegetables) with relatively high support, confidence and lift can also be used for marketing strategies to increase cross-selling. When customers buy white bread, they tend to buy other vegetables as well.

(yogurt) → (berries) and (berries) → (yogurt) are also useful rules with relatively high support, confidence and lift. It's probably people nowadays like to make their yogurt healthier by putting berries on top.

In summary, I have found 4 useful rules: (eggs) → (whole milk), (white bread) → (other vegetables), yogurt) → (berries) and (berries) → (yogurt), and I recommend them to be used in our marketing strategies. By reviewing the result, I found that a good minimum support could be set to 1% as this will cover all the necessary rules.

Task 3

2. In your report, describe a possible business context for this dataset. Detail how data collection may occur and suggest a data management plan that includes a description of the data, what kind of storage should be used, who should have access to the data, and anything else you deem relevant to the context described.

A possible context for this dataset would be a supermarket recording the transactions, so that they could find out which items have strong connections with each other and thus they can develop their marketing strategies accordingly (i.e. attract customers with product bundles at a lower price, offer discount if customers buy another item along with the existing items in their basket, etc) and decide which items should be placed close to each other. Their goal would be to boost sales and increase cross-selling.

The data could be collected by the sales system at the tills, which input the barcodes of items scanned in each transaction and output the transaction ID and product ID, with each row corresponding to each transaction. The data will be collected automatically by the scans to avoid any mistakes. There are other ways to output the data, such as each row corresponding to a transaction ID and a single item.

The data could be stored in a relational database for a small or medium sized store, because the data collected could be used directly. For large stores, they usually would like to collect more information and analyze more patterns, also the number of transactions would be huge. So it's better to store the data in a data lake.

The store manager should have the ultimate access to the data, including reading, modifying and authorization. The data analysts and marketing personnel should also have access to the data to analyze the data and read the results.

The whole ETL, analysis and modeling can be automated by a data pipeline, which should be updated regularly. For example, there might be new products coming into the store, and the features of the model need to be re-coded to get the new one hot encoding. The final result of the modeling can be shown on a weekly report or a monthly report depending on the size of the store.