DATA MINING PROJECT

1.a. What is the advantage of using the Apriori algorithm in comparison with computing the support of every subset of an itemset in order to find the frequent itemsets in a transaction dataset?

<u>Answer:</u> Calculating the support of each subset of an itemset is a time-consuming task that is unnecessary for unlikely itemsets. While the Apriori algorithm employs a level-wise search strategy in which k+1 itemsets are processed using k itemsets, this early elimination of unlikely itemsets improves efficiency by reducing search space.

b. Let $L1$ denote the set of frequent 1-itemsets. For $k \geq 2$ why must every frequent k-itemset be a superset of an itemset in $L1$?

<u>Answer:</u> The Apriori property states that if the superset is a frequent itemset, then its subsets have to be frequent as well. It is a crucial component of the Apriori algorithm since it narrows the search space, boosting efficiency. L1 contains frequent itemsets that meet the minimum support threshold, and from these itemsets, k-itemsets that are also frequent according to the Apriori property are generated. As a result, each frequent k-itemset needs to be a superset of an L1 item.

c. Let $L2$ = {{1,2},{1,4},{2,3},{2,4},{3,5}}. Compute the set of candidates $C3$ that is obtained by joining every pair of joinable itemsets from $L2$.

<u>Answer:</u> The first itemset is created by joining {1,2} and {1,4}, and the second itemset is generated by joining {2,3} and {2,4}. The remaining itemsets cannot be joined.

ITEMSET
{1,2,4}
{2,3,4}

d. Let $S1$ denote the support of the association rule: $\{boarding\ pass, passport\} \Rightarrow \{flight\}$ Let $S2$ denote the support of the association rule: $\{boarding\ pass\} \Rightarrow \{flight\}$ What is the relationship between $S1$ and $S2$?

<u>Answer :</u> As can be seen, the first rule makes both "boarding pass" and "passport" required conditions, which makes the rule more stringent and less likely to occur. If the transaction contains both of these terms, it is most likely to contain “flight”. In contrast to the first rule, the second one is more wide-ranging and states that if the transaction includes a "boarding pass," it is most likely to contain "flight."

As a result, the second rule happens more frequently than or equally often as the first rule. This indicates that S1 <= S2, which is the relationship between S1 and S2.

e. What is the support of the rule: {} ⇒ {$Eggs$} in the transaction dataset below shown in Figure 1?

```
[['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
 ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
 ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
 ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
 ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]
```

The following formula is used to determine the rule {}⇒{Eggs}'s support:
The dataset contains four instances of eggs (Ne = 4).
Total dataset length (L) = 5
<u>As a result, support of the rule Eggs = Ne/L = 4/5 = 0.8</u>

f. In the transaction dataset shown in Figure 1, what is the maximum length of a frequent itemset for a support threshold of 0.2?

C1

Itemset	Support Count
{'Milk'}	0.6
{'Onion'}	0.6
{'Nutmeg'}	0.4
{'Kidney Beans'}	1
{'Eggs'}	0.8
{'Yogurt'}	0.6
{'Dill'}	0.2
{'Apple'}	0.2
{'Unicorn'}	0.2
{'Corn'}	0.4
{'Ice Cream'}	0.2

L1

Itemset	Support Count
{'Milk'}	0.6
{'Onion'}	0.6
{'Nutmeg'}	0.4
{'Kidney Beans'}	1

{‘Eggs’}	0.8
{‘Yogurt’}	0.6
{‘Dill’}	0.2
{‘Apple’}	0.2
{‘Unicorn’}	0.2
{‘Corn’}	0.4
{‘Ice Cream’}	0.2

C2

Itemset	Support Count
{'Eggs', 'Apple'}	0.2
{'Kidney Beans', 'Apple'}	0.2
{'Milk', 'Apple'}	0.2
{'Corn', 'Eggs'}	0.2
{'Corn', 'Ice cream'}	0.2
{'Corn', 'Kidney Beans'}	0.2
{'Corn', 'Milk'}	0.2
{'Corn', 'Onion'}	0.2
{'Corn', 'Unicorn'}	0.2
{'Corn', 'Yogurt'}	0.2
{'Eggs', 'Dill'}	0.2
{'Kidney Beans', 'Dill'}	0.2
{'Nutmeg', 'Dill'}	0.2
{'Onion', 'Dill'}	0.2
{'Yogurt', 'Dill'}	0.2
{'Ice cream', 'Eggs'}	0.2
{'Kidney Beans', 'Eggs'}	0.8
{'Milk', 'Eggs'}	0.4
{'Nutmeg', 'Eggs'}	0.4
{'Onion', 'Eggs'}	0.6
{'Yogurt', 'Eggs'}	0.4
{'Ice cream', 'Kidney Beans'}	0.2
{'Ice cream', 'Onion'}	0.2
{'Milk', 'Kidney Beans'}	0.6
{'Nutmeg', 'Kidney Beans'}	0.4
{'Onion', 'Kidney Beans'}	0.6
{'Unicorn', 'Kidney Beans'}	0.2
{'Yogurt', 'Kidney Beans'}	0.6
{'Nutmeg', 'Milk'}	0.2
{'Milk', 'Onion'}	0.2

{'Milk', 'Unicorn'}	0.2
{'Milk', 'Yogurt'}	0.4
{'Nutmeg', 'Onion'}	0.4
{'Nutmeg', 'Yogurt'}	0.4
{'Yogurt', 'Onion'}	0.4
{'Unicorn', 'Yogurt'}	0.2

L2

Itemset	Support Count
{'Eggs', 'Apple'}	0.2
{'Kidney Beans', 'Apple'}	0.2
{'Milk', 'Apple'}	0.2
{'Corn', 'Eggs'}	0.2
{'Corn', 'Ice cream'}	0.2
{'Corn', 'Kidney Beans'}	0.2
{'Corn', 'Milk'}	0.2
{'Corn', 'Onion'}	0.2
{'Corn', 'Unicorn'}	0.2
{'Corn', 'Yogurt'}	0.2
{'Eggs', 'Dill'}	0.2
{'Kidney Beans', 'Dill'}	0.2
{'Nutmeg', 'Dill'}	0.2
{'Onion', 'Dill'}	0.2
{'Yogurt', 'Dill'}	0.2
{'Ice cream', 'Eggs'}	0.2
{'Kidney Beans', 'Eggs'}	0.8
{'Milk', 'Eggs'}	0.4
{'Nutmeg', 'Eggs'}	0.4
{'Onion', 'Eggs'}	0.6
{'Yogurt', 'Eggs'}	0.4
{'Ice cream', 'Kidney Beans'}	0.2
{'Ice cream', 'Onion'}	0.2
{'Milk', 'Kidney Beans'}	0.6
{'Nutmeg', 'Kidney Beans'}	0.4
{'Onion', 'Kidney Beans'}	0.6
{'Unicorn', 'Kidney Beans'}	0.2
{'Yogurt', 'Kidney Beans'}	0.6
{'Nutmeg', 'Milk'}	0.2
{'Milk', 'Onion'}	0.2
{'Milk', 'Unicorn'}	0.2

{'Milk', 'Yogurt'}	0.4
{'Nutmeg', 'Onion'}	0.4
{'Nutmeg', 'Yogurt'}	0.4
{'Yogurt', 'Onion'}	0.4
{'Unicorn', 'Yogurt'}	0.2

C3 (after pruning)

Itemset	Support Count
{'Eggs', 'Kidney Beans', 'Apple'}	0.2
{'Eggs', 'Milk', 'Apple'}	0.2
{'Milk', 'Kidney Beans', 'Apple'}	0.2
{'Corn', 'Ice cream', 'Eggs'}	0.2
{'Corn', 'Kidney Beans', 'Eggs'}	0.2
{'Corn', 'Onion', 'Eggs'}	0.2
{'Corn', 'Ice cream', 'Kidney Beans'}	0.2
{'Corn', 'Ice cream', 'Onion'}	0.2
{'Corn', 'Milk', 'Kidney Beans'}	0.2
{'Corn', 'Onion', 'Kidney Beans'}	0.2
{'Corn', 'Unicorn', 'Kidney Beans'}	0.2
{'Corn', 'Yogurt', 'Kidney Beans'}	0.2
{'Unicorn', 'Corn', 'Milk'}	0.2
{'Corn', 'Milk', 'Yogurt'}	0.2
{'Corn', 'Unicorn', 'Yogurt'}	0.2
{'Eggs', 'Kidney Beans', 'Dill'}	0.2
{'Eggs', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Onion', 'Dill'}	0.2
{'Eggs', 'Yogurt', 'Dill'}	0.2
{'Nutmeg', 'Kidney Beans', 'Dill'}	0.2
{'Kidney Beans', 'Onion', 'Dill'}	0.2
{'Kidney Beans', 'Yogurt', 'Dill'}	0.2
{'Nutmeg', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Yogurt', 'Dill'}	0.2
{'Yogurt', 'Onion', 'Dill'}	0.2
{'Kidney Beans', 'Ice cream', 'Eggs'}	0.2
{'Ice cream', 'Onion', 'Eggs'}	0.2
{'Milk', 'Kidney Beans', 'Eggs'}	0.4
{'Nutmeg', 'Kidney Beans', 'Eggs'}	0.4
{'Kidney Beans', 'Onion', 'Eggs'}	0.6
{'Kidney Beans', 'Yogurt', 'Eggs'}	0.4
{'Nutmeg', 'Milk', 'Eggs'}	0.2

{'Milk', 'Onion', 'Eggs'}	0.2
{'Milk', 'Yogurt', 'Eggs'}	0.2
{'Nutmeg', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Yogurt', 'Eggs'}	0.4
{'Yogurt', 'Onion', 'Eggs'}	0.4
{'Ice cream', 'Onion', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Kidney Beans'}	0.2
{'Milk', 'Onion', 'Kidney Beans'}	0.2
{'Milk', 'Unicorn', 'Kidney Beans'}	0.2
{'Milk', 'Yogurt', 'Kidney Beans'}	0.4
{'Nutmeg', 'Onion', 'Kidney Beans'}	0.4
{'Nutmeg', 'Yogurt', 'Kidney Beans'}	0.4
{'Yogurt', 'Onion', 'Kidney Beans'}	0.4
{'Unicorn', 'Yogurt', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Onion'}	0.2
{'Nutmeg', 'Milk', 'Yogurt'}	0.2
{'Milk', 'Yogurt', 'Onion'}	0.2
{'Milk', 'Unicorn', 'Yogurt'}	0.2
{'Nutmeg', 'Yogurt', 'Onion'}	0.4

L3

Itemset	Support Count
{'Eggs', 'Kidney Beans', 'Apple'}	0.2
{'Eggs', 'Milk', 'Apple'}	0.2
{'Milk', 'Kidney Beans', 'Apple'}	0.2
{'Corn', 'Ice cream', 'Eggs'}	0.2
{'Corn', 'Kidney Beans', 'Eggs'}	0.2
{'Corn', 'Onion', 'Eggs'}	0.2
{'Corn', 'Ice cream', 'Kidney Beans'}	0.2
{'Corn', 'Ice cream', 'Onion'}	0.2
{'Corn', 'Milk', 'Kidney Beans'}	0.2
{'Corn', 'Onion', 'Kidney Beans'}	0.2
{'Corn', 'Unicorn', 'Kidney Beans'}	0.2
{'Corn', 'Yogurt', 'Kidney Beans'}	0.2
{'Unicorn', 'Corn', 'Milk'}	0.2
{'Corn', 'Milk', 'Yogurt'}	0.2
{'Corn', 'Unicorn', 'Yogurt'}	0.2
{'Eggs', 'Kidney Beans', 'Dill'}	0.2
{'Eggs', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Onion', 'Dill'}	0.2

{'Eggs', 'Yogurt', 'Dill'}	0.2
{'Nutmeg', 'Kidney Beans', 'Dill'}	0.2
{'Kidney Beans', 'Onion', 'Dill'}	0.2
{'Kidney Beans', 'Yogurt', 'Dill'}	0.2
{'Nutmeg', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Yogurt', 'Dill'}	0.2
{'Yogurt', 'Onion', 'Dill'}	0.2
{'Kidney Beans', 'Ice cream', 'Eggs'}	0.2
{'Ice cream', 'Onion', 'Eggs'}	0.2
{'Milk', 'Kidney Beans', 'Eggs'}	0.4
{'Nutmeg', 'Kidney Beans', 'Eggs'}	0.4
{'Kidney Beans', 'Onion', 'Eggs'}	0.6
{'Kidney Beans', 'Yogurt', 'Eggs'}	0.4
{'Nutmeg', 'Milk', 'Eggs'}	0.2
{'Milk', 'Onion', 'Eggs'}	0.2
{'Milk', 'Yogurt', 'Eggs'}	0.2
{'Nutmeg', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Yogurt', 'Eggs'}	0.4
{'Yogurt', 'Onion', 'Eggs'}	0.4
{'Ice cream', 'Onion', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Kidney Beans'}	0.2
{'Milk', 'Onion', 'Kidney Beans'}	0.2
{'Milk', 'Unicorn', 'Kidney Beans'}	0.2
{'Milk', 'Yogurt', 'Kidney Beans'}	0.4
{'Nutmeg', 'Onion', 'Kidney Beans'}	0.4
{'Nutmeg', 'Yogurt', 'Kidney Beans'}	0.4
{'Yogurt', 'Onion', 'Kidney Beans'}	0.4
{'Unicorn', 'Yogurt', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Onion'}	0.2
{'Nutmeg', 'Milk', 'Yogurt'}	0.2
{'Milk', 'Yogurt', 'Onion'}	0.2
{'Milk', 'Unicorn', 'Yogurt'}	0.2
{'Nutmeg', 'Yogurt', 'Onion'}	0.4

C4 (after pruning)

Itemset	Support Count
{'Milk', 'Eggs', 'Kidney Beans', 'Apple'}	0.2

{'Kidney Beans', 'Corn', 'Ice cream', 'Eggs'}	0.2
{'Corn', 'Ice cream', 'Onion', 'Eggs'}	0.2
{'Corn', 'Kidney Beans', 'Onion', 'Eggs'}	0.2
{'Corn', 'Ice cream', 'Onion', 'Kidney Beans'}	0.2
{'Unicorn', 'Corn', 'Milk', 'Kidney Beans'}	0.2
{'Corn', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2
{'Corn', 'Unicorn', 'Yogurt', 'Kidney Beans'}	0.2
{'Unicorn', 'Corn', 'Milk', 'Yogurt'}	0.2
{'Eggs', 'Kidney Beans', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Kidney Beans', 'Onion', 'Dill'}	0.2
{'Eggs', 'Kidney Beans', 'Yogurt', 'Dill'}	0.2
{'Eggs', 'Onion', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Yogurt', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Yogurt', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Kidney Beans', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Kidney Beans', 'Yogurt', 'Dill'}	0.2
{'Kidney Beans', 'Yogurt', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Yogurt', 'Onion', 'Dill'}	0.2
{'Kidney Beans', 'Ice cream', 'Onion', 'Eggs'}	0.2
{'Milk', 'Nutmeg', 'Kidney Beans', 'Eggs'}	0.2
{'Milk', 'Kidney Beans', 'Onion', 'Eggs'}	0.2
{'Milk', 'Kidney Beans', 'Yogurt', 'Eggs'}	0.2
{'Nutmeg', 'Kidney Beans', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Kidney Beans', 'Yogurt', 'Eggs'}	0.4
{'Kidney Beans', 'Yogurt', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Milk', 'Onion', 'Eggs'}	0.2
{'Nutmeg', 'Milk', 'Yogurt', 'Eggs'}	0.2
{'Milk', 'Yogurt', 'Onion', 'Eggs'}	0.2
{'Nutmeg', 'Yogurt', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Milk', 'Onion', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2
{'Milk', 'Yogurt', 'Onion', 'Kidney Beans'}	0.2

{'Milk', 'Unicorn', 'Yogurt', 'Kidney Beans'}	0.2
{'Nutmeg', 'Yogurt', 'Onion', 'Kidney Beans'}	0.4
{'Milk', 'Nutmeg', 'Yogurt', 'Onion'}	0.2

L4

Itemset	Support Count
{'Milk', 'Eggs', 'Kidney Beans', 'Apple'}	0.2
{'Kidney Beans', 'Corn', 'Ice cream', 'Eggs'}	0.2
{'Corn', 'Ice cream', 'Onion', 'Eggs'}	0.2
{'Corn', 'Kidney Beans', 'Onion', 'Eggs'}	0.2
{'Corn', 'Ice cream', 'Onion', 'Kidney Beans'}	0.2
{'Unicorn', 'Corn', 'Milk', 'Kidney Beans'}	0.2
{'Corn', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2
{'Corn', 'Unicorn', 'Yogurt', 'Kidney Beans'}	0.2
{'Unicorn', 'Corn', 'Milk', 'Yogurt'}	0.2
{'Eggs', 'Kidney Beans', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Kidney Beans', 'Onion', 'Dill'}	0.2
{'Eggs', 'Kidney Beans', 'Yogurt', 'Dill'}	0.2
{'Eggs', 'Onion', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Yogurt', 'Dill', 'Nutmeg'}	0.2
{'Eggs', 'Yogurt', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Kidney Beans', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Kidney Beans', 'Yogurt', 'Dill'}	0.2
{'Kidney Beans', 'Yogurt', 'Onion', 'Dill'}	0.2
{'Nutmeg', 'Yogurt', 'Onion', 'Dill'}	0.2
{'Kidney Beans', 'Ice cream', 'Onion', 'Eggs'}	0.2
{'Milk', 'Nutmeg', 'Kidney Beans', 'Eggs'}	0.2
{'Milk', 'Kidney Beans', 'Onion', 'Eggs'}	0.2
{'Milk', 'Kidney Beans', 'Yogurt', 'Eggs'}	0.2
{'Nutmeg', 'Kidney Beans', 'Onion', 'Eggs'}	0.4

{'Nutmeg', 'Kidney Beans', 'Yogurt', 'Eggs'}	0.4
{'Kidney Beans', 'Yogurt', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Milk', 'Onion', 'Eggs'}	0.2
{'Nutmeg', 'Milk', 'Yogurt', 'Eggs'}	0.2
{'Milk', 'Yogurt', 'Onion', 'Eggs'}	0.2
{'Nutmeg', 'Yogurt', 'Onion', 'Eggs'}	0.4
{'Nutmeg', 'Milk', 'Onion', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2
{'Milk', 'Yogurt', 'Onion', 'Kidney Beans'}	0.2
{'Milk', 'Unicorn', 'Yogurt', 'Kidney Beans'}	0.2
{'Nutmeg', 'Yogurt', 'Onion', 'Kidney Beans'}	0.4
{'Milk', 'Nutmeg', 'Yogurt', 'Onion'}	0.2

C5

Itemset	Support Count
{'Onion', 'Corn', 'Ice cream', 'Eggs', 'Kidney Beans'}	0.2
{'Corn', 'Unicorn', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Dill', 'Eggs', 'Kidney Beans'}	0.2
{'Nutmeg', 'Dill', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Dill', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Dill', 'Yogurt', 'Eggs'}	0.2
{'Onion', 'Nutmeg', 'Dill', 'Yogurt', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Milk', 'Eggs', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Milk', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.4

{'Onion', 'Nutmeg', 'Milk', 'Yogurt', 'Eggs'}	0.2
{'Onion', 'Nutmeg', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2

L5

Itemset	Support Count
{'Onion', 'Corn', 'Ice cream', 'Eggs', 'Kidney Beans'}	0.2
{'Corn', 'Unicorn', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Dill', 'Eggs', 'Kidney Beans'}	0.2
{'Nutmeg', 'Dill', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Dill', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Dill', 'Yogurt', 'Eggs'}	0.2
{'Onion', 'Nutmeg', 'Dill', 'Yogurt', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Milk', 'Eggs', 'Kidney Beans'}	0.2
{'Nutmeg', 'Milk', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Milk', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.4
{'Onion', 'Nutmeg', 'Milk', 'Yogurt', 'Eggs'}	0.2
{'Onion', 'Nutmeg', 'Milk', 'Yogurt', 'Kidney Beans'}	0.2

C6

Itemset	Support Count
{'Onion', 'Nutmeg', 'Dill', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Milk', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2

L6

Itemset	Support Count
{'Onion', 'Nutmeg', 'Dill', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2
{'Onion', 'Nutmeg', 'Milk', 'Yogurt', 'Eggs', 'Kidney Beans'}	0.2

Because the itemsets cannot be joined to proceed, we can infer that the maximum length of an itemset with a support of 0.2 is 6.

2.a. For a system designed to prevent identity theft in online transactions, we are focusing on identifying unusual transaction patterns. Propose 2 possible contextual attributes and 2 possible behavioral attributes that could be integrated into this system's algorithm. Provide a rationale for classifying each attribute as either contextual or behavioral.

<u>Contextual attributes that could be present are:</u>
Geographical location: We can obtain contextual information about the overall user's transaction pattern by monitoring the user's transactions in a geographical area. Identity theft may be detected if a user's transaction is suddenly detected in a location far from where the transactions normally take place.

Device type: Transactions made from specific devices that the user is using can be monitored to obtain contextual information. If the user's transaction is unexpectedly made from a new, unidentified device other than the ones shown in the pattern, identity theft may have occurred.

<u>Behavioral attributes that could be present are:</u>
Number of transactions over a predetermined time period: Transactional behavior of the user can be obtained by tracking transactions over a predetermined time period, such as a month or a year. An unusual spike in a user's transactions during a specific period of time may indicate identity theft.

The amount sent in a transaction: The amount sent in a transaction can be tracked to determine the user's behavior when sending or receiving specific amounts. If a user appears to be sending a large amount that deviates from normal behavior, it is possible that identity theft is taking place.

b. Assume that you are provided with the [University of Wisconsin breast cancer dataset](https://archive.ics.uci.edu/ml/machine-learningdatabases/breast-cancer-wisconsin/breast-cancer-wisconsin.data) from the Week 3 lab, and that you are asked to detect outliers from this dataset. Additional information on the dataset attributes can be found [online](https://archive.ics.uci.edu/ml/machine-learning-databases/breastcancer-wisconsin/breast-cancer-wisconsin.names). Explain one possible outlier detection method that you could apply for detecting outliers for this particular dataset, explain what is defined as an outlier for your

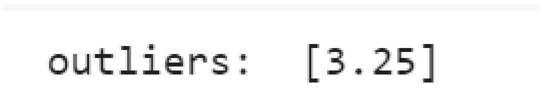
suggested approach given this particular dataset, and justify why you would choose this particular method for outlier detection.

Answer: Since the extreme values in the dataset constitute a group that differs from the normal data present collectively, proximity-based methods can be employed to detect outliers. As a result, the outlier group will be located far from the normal data group, which can be found using this technique. Here, data with extreme values relative to other data with comparable values are deemed to be outliers.

c. The monthly rainfall in the London borough of Tower Hamlets in 2019 had the following amount of precipitation (measured in mm, values from January-December 2018): {22.93, 20.69, 25.75, 23.84, 25.34, 3.25, 23.55, 28.28, 23.72, 22.42, 26.83, 23.82}. Assuming that the data is based on a normal distribution, identify outlier values in the above dataset using the maximum likelihood method.

```
import numpy as np

data = np.array([22.93, 20.69, 25.75, 23.84, 25.34, 3.25, 23.55, 28.28, 23.72, 22.42, 26.83, 23.82])

mean = np.mean(data)
standard_deviation = np.std(data)

sum_value = mean + 3*standard_deviation
difference_value = mean - 3*standard_deviation

outliers = []

for i in range(len(data)):
  if data[i] > sum_value or data[i] < difference_value:
    outliers.append(data[i])

print("outliers: ",outliers)
```

Output:


```
outliers:  [3.25]
```

Answer:

3.25 is the outlier, which can be observed by executing the code above.

d. Using the stock prices (stocks.csv included in the supplementary material) dataset used in sections 1 and 2 of Week 9 lab, estimate the outliers in the dataset using the one-class SVM classifier approach. As input to the classifier, use the percentage of changes in the daily closing price of each stock, as was done in section 1 of the notebook. Use the same SVM settings as in the lab notebook. Plot a 3D scatterplot of the dataset, where each object is color coded according to whether it is an outlier or an inlier. Also compute a histogram and the frequencies of the estimated outlier and inlier labels. In terms of the plotted results, how does the one-class SVM approach for outlier detection differ from the parametric and

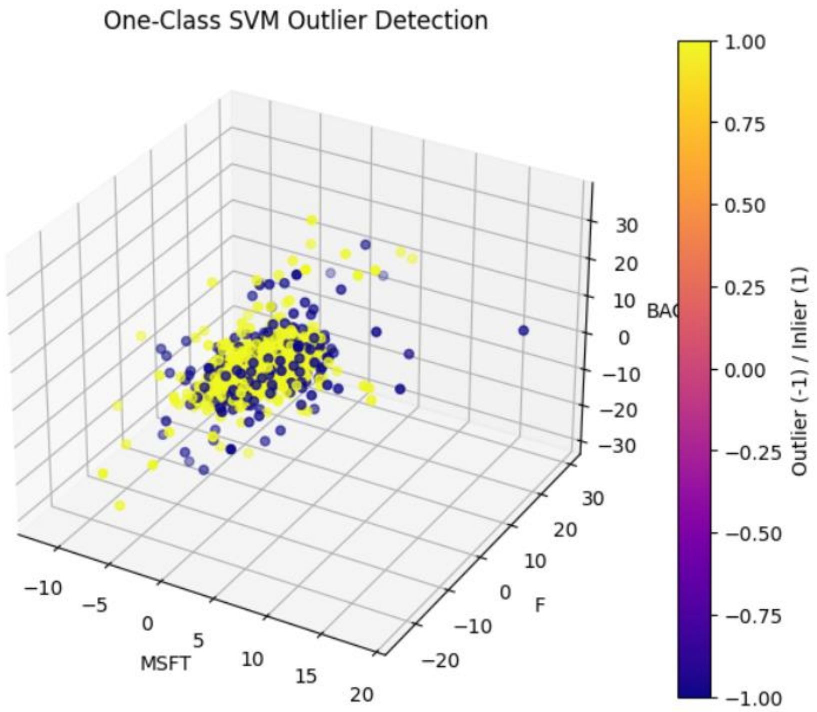
proximity-based methods used in the lab notebook? What percentage of the dataset objects are classified as outliers?

Answer: Since the parametric and proximity-based results use similar concepts, we can infer from the plotted results that they are similar. The distance between a point and the distribution, calculated for data covariance, is determined by the Mahalanobis Distance in a parametric result, whereas the proximity-based approach uses k-NN to determine the distance between a point and its k-th nearest neighbor. However, one-class SVM is a classification-based technique in which the model is trained to draw a boundary between normal data and outliers; when comparing the plots of the two previously mentioned methods, one-class SVM has more outliers. The frequency indicates that 17.799% of dataset objects were identified as outliers by one-class SVM.

Code :

```
import pandas as pd
import numpy as np
from sklearn.svm import OneClassSVM
import matplotlib.pyplot as plt

stocks = pd.read_csv('stocks.csv', header='infer')
stocks.index = stocks['Date']
stocks = stocks.drop(['Date'],axis=1)

N,d = stocks.shape
delta = pd.DataFrame(100*np.divide(stocks.iloc[1:,:].values-stocks.iloc[:N-1,:].values,
stocks.iloc[:N-1,:].values),
                columns=stocks.columns, index=stocks.iloc[1:].index)

X = delta[['MSFT', 'F', 'BAC']]

model = OneClassSVM(nu=0.01,gamma='auto')
model.fit(delta)
outliers_predicted = model.fit_predict(X)

outlier_pd = pd.DataFrame(outliers_predicted, index=delta.index, columns=['Outlier score'])
new_delta = pd.concat((delta,outlier_pd), axis=1)

fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(111, projection='3d')

scatter_plot = ax.scatter(new_delta['MSFT'], new_delta['F'], new_delta['BAC'],
c=new_delta['Outlier score'], cmap='plasma')

ax.set_xlabel('MSFT')
ax.set_ylabel('F')
ax.set_zlabel('BAC')
ax.set_title('One-Class SVM Outlier Detection')

fig.colorbar(scatter_plot, ax=ax, label='Outlier (-1) / Inlier (1)')
plt.show()
```

Output:



Code :

```
x_value = new_delta['Outlier score'].value_counts()/len(new_delta['Outlier score'])
print("frequencies of outlier(-1) and inlier(1) labels: ")
print(x_value)
```

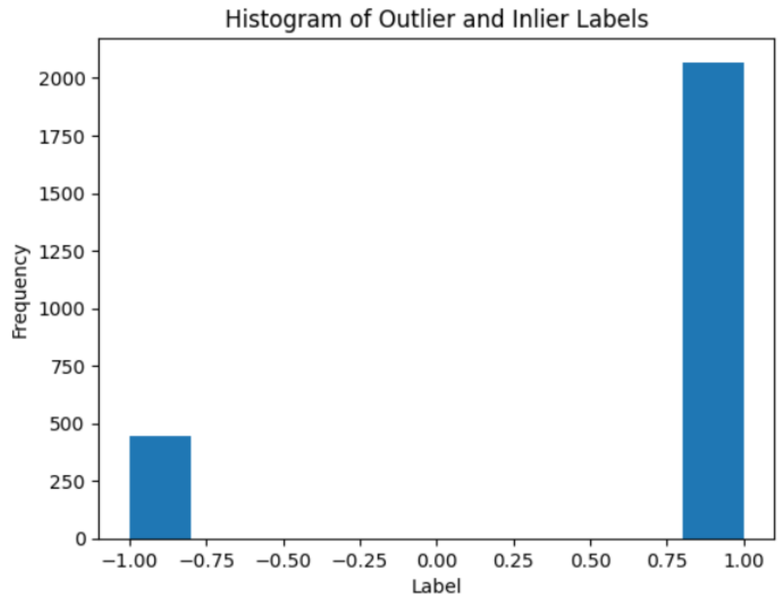
Output:

```
frequencies of outlier(-1) and inlier(1) labels:
 1    0.82201
-1    0.17799
Name: Outlier score, dtype: float64
```

Plotting :

```
plt.hist(new_delta['Outlier score'])
plt.title('Histogram of Outlier and Inlier Labels')
plt.xlabel('Label')
plt.ylabel('Frequency')
plt.show()
```

Output:



3.a. You are provided with the following URL: http://eecs.qmul.ac.uk/~emmanouilb/income_table.html. This webpage includes a table on individuals' income and shopping habits. i. [pen & paper] - Inspect the HTML code of the above URL and provide a short report on the various tags present in the code. What is the function of each unique tag present in the HTML code?

Answer:

The following tags are found in the HTML code:

HTML Tags	**Function**
<html>	The initial tag that appears in an HTML document and holds all other tags.
<head>	The head section of an HTML document includes the meta-data for the document, including "title," "meta," and "link."
<body>	All of the visible elements are present in this section, which serves as the document's main body. It contains numerous other tags, such as

	headings, paragraphs, tables, etc.
<h1>	Heading is one of its many uses; the levels range from <h1> to <h6>, signifying largest to smallest heading. primarily employed to organize the text.
<p>	Used to display a block of text as a paragraph.
<table>	Used to create HTML tables with rows and columns for data organization and display. Other tags like <thead>, <tbody>, <tr>, and <td> are contained within it.
<thead>	Includes <tr> and <th> tags; used to represent header content in the table
<tr>	Represents a table row that has either <th> or <td> in it.
<td>	Appears in the <tr> tag and is represented as a table data cell.
<tbody>	Used to organize the body content in the table and provide table structure. includes tag <tr>

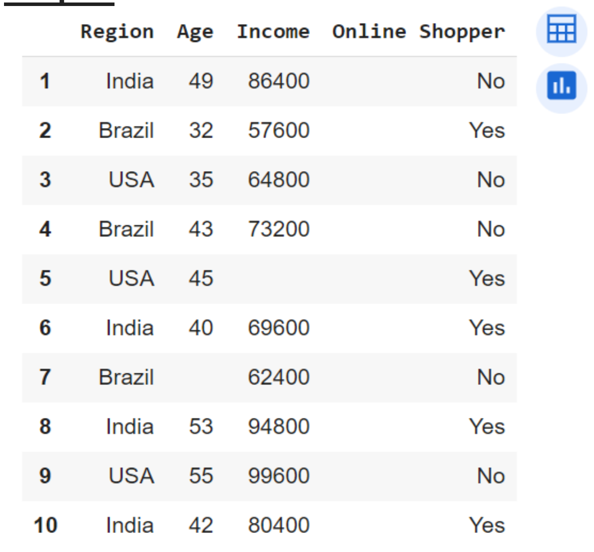
ii. Using Beautiful Soup, scrape the table and convert it into a pandas dataframe. Perform data cleaning when necessary to remove extra characters (no need to handle missing values). In the report include the code that was used to scrape and convert the table and provide evidence that the table has been successfully scraped and converted (e.g. by displaying the contents of the dataframe).

<u>Code:</u>

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from urllib.request import urlopen
from bs4 import BeautifulSoup

url = "http://eecs.qmul.ac.uk/~emmanouilb/income_table.html"
html = urlopen(url)
soup = BeautifulSoup(html, 'lxml')

header_list = []
col_labels = soup.find_all('th')
col_str = str(col_labels)
```

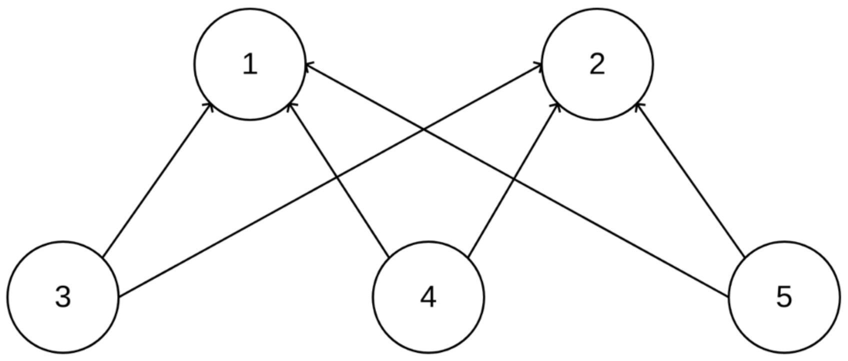
```
cleantext_header = BeautifulSoup(col_str, "lxml").get_text()
header_list.append(cleantext_header)

table_list = []
question1_rows = soup.find_all('tr')
for row in question1_rows:
 row_td = row.find_all('td')
 row_cells = str(row_td)
 row_cleantext = BeautifulSoup(row_cells, "lxml").get_text()
 table_list.append(row_cleantext)

df_header = pd.DataFrame(header_list)
df_header= df_header[0].str.split(',', expand=True)

df_table = pd.DataFrame(table_list)
df_table = df_table[0].str.split(',', expand=True)

frames = [df_header, df_table]
df = pd.concat(frames)

df[0] = df[0].str.strip('[')
df[3] = df[3].str.strip(']')

df = df.rename(columns=df.iloc[0])
df = df.drop(df.index[0])

df.head(10)
```

Output:



```
     Region  Age  Income  Online Shopper
 1    India   49   86400              No
 2   Brazil   32   57600             Yes
 3      USA   35   64800              No
 4   Brazil   43   73200              No
 5      USA   45                     Yes
 6    India   40   69600             Yes
 7   Brazil        62400              No
 8    India   53   94800             Yes
 9      USA   55   99600              No
10    India   42   80400             Yes
```

b. Consider the graph in the figure below as displaying the links for a group of 5 webpages. Which of the 5 nodes would you consider hubs and which would you consider authorities?



<u>Answer :</u>

1 and 2 are considered as authorities because they each have three in-links, while 3, 4, and 5 are considered as hubs because they each have two out-links to the authorities.

4.. is a pen-and-paper exercise; questions 4b is a coding exercise. For all your answers please show your workings (equations or code when applicable).

a. Consider the following sentences related to data mining theory, and assume that each of the below sentences corresponds to a different document:

*** Data refers to characteristics that are collected through observation.**
*** A dataset can be viewed as a collection of objects.**
*** Data objects are described by a number of attributes.**
*** An attribute is a characteristic or feature of an object.**

I. Construct and display the document-term matrix for the above documents. Remove all stop words (here consider as stop words: articles, prepositions, conjunctions, pronouns, and common verbs) and punctuation marks; convert any plural nouns/adjectives to their singular form; and convert verbs to the present tense and first-person singular form, before you construct the matrix.

<u>Answer :</u>

Eliminating every stop word:

Data refer to characteristic collect observation.
Dataset view collection object.
Data object describe number attribute.
Attribute characteristic feature object.

Construction of Document-term Matrix:

	Document-1	Document-2	Document-3	Document-4
Attribute	0	0	1	1
Characteristic	1	0	0	1
Collection	0	1	0	0
Data	1	0	1	0
Describe	0	0	1	0
Feature	0	0	0	1
Number	0	0	1	0
Object	0	1	1	1
Observation	1	0	0	0
Refer	1	0	0	0
View	0	1	0	0

II. Using the above constructed document-term matrix, calculate the inverse document frequency $idf(w)$ for all words w you have identified from the previous question (I).

<u>Answer:</u>

We can determine each word's idf by using the formula below:

$$idf(w) = log10(|D|/|Dw|)$$

where Dw is the number of times the word appears and D is the length of the documents overall.

Inverse Document Frequency:

W	idf(W)
Attribute	0.3010
Characteristic	0.3010
Collection	0.6020
Data	0.3010
Describe	0.6020
Feature	0.6020
Number	0.6020
Object	0.1249
Observation	0.6020

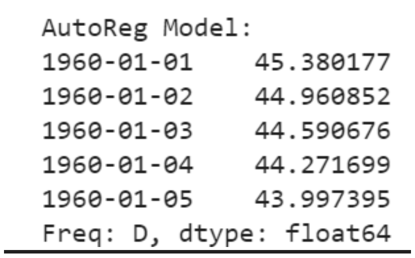
Refer	0.6020
View	0.6020

b. Using the daily births dataset from Week 11 lab notebook, smooth the time series using trailing moving average smoothing and a window size that corresponds to one week; then replace any NaN values with zeros. Perform time series forecasting using the smoothed dataset in order to predict daily births for the first 5 days of 1960, using the models below. Show your forecasting results. AR model with p = 2 ARMA model with p = 2 and q = 2

Code:

```
from pandas import read_csv
import numpy as np
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.arima.model import ARIMA

birth_data = read_csv('births.csv', header=0, index_col=0)

smooth_mean_df = birth_data.rolling(window=7).mean()

smooth_mean_df.fillna(0, inplace=True)

model_ar = AutoReg(smooth_mean_df, lags=2,old_names=False)
model_ar_fit = model_ar.fit()

predict_ar_model = model_ar_fit.predict(len(smooth_mean_df), len(smooth_mean_df)+4)
print("AutoReg Model:")
print(predict_ar_model)
```

Output :



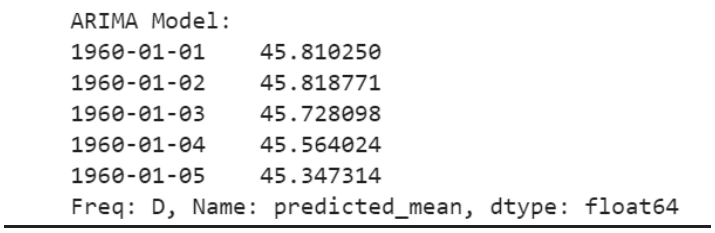
```
AutoReg Model:
1960-01-01    45.380177
1960-01-02    44.960852
1960-01-03    44.590676
1960-01-04    44.271699
1960-01-05    43.997395
Freq: D, dtype: float64
```

Code:

```
model_arima = ARIMA(smooth_mean_df, order=(2, 0, 2))
model_arima_fit = model_arima.fit()
```

```
predict_arima = model_arima_fit.predict(len(smooth_mean_df), len(smooth_mean_df)+4)
print("ARIMA Model:")
print(predict_arima)
```

Output:



```
ARIMA Model:
1960-01-01    45.810250
1960-01-02    45.818771
1960-01-03    45.728098
1960-01-04    45.564024
1960-01-05    45.347314
Freq: D, Name: predicted_mean, dtype: float64
```