Data Mining Assignment – 2

**We will import all the libraries required in only one cell so that it is much better understandable to the viewer about the libraries we have imported.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.svm import OneClassSVM

from urllib.request import urlopen

from bs4 import BeautifulSoup

from statsmodels.tsa.ar\_model import AutoReg

from statsmodels.tsa.arima.model import ARIMA

## Question 1:

**Questions 1(a) to 1(f) are pen-and-paper exercises (brief answers and justifications are expected). In all responses, please show your workings (equations, justifications). Please try to use an editor instead of taking pictures or scanning actual paper. Pen and paper here refer to not using any programming.**

* 1. **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?**
  2. **Let denote the set of frequent -itemsets. For why must every frequent -itemset be a superset of an itemset in ?**
  3. **Let . Compute the set of candidates that is obtained by joining every pair of joinable itemsets from**
  4. **Let denote the support of the association rule:  
       
       
     Let denote the support of the association rule:  
       
        
       
     What is the relationship between and ?**
  5. **What is the support of the rule:  
     in the transaction dataset below shown in Figure 1?**

**A group of black text

Description automatically generated**

**Figure 1. Dataset 1**

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

### Part a:

The advantage of using Apriori algorithm over computing the support of every subset of an itemset can be given by the efficiency of the Apriori algorithm and also some optimisation techniques. Here are some advantages noted below:

1. As we know, the Apriori property states that if an itemset is frequent, then all of its subsets must also be frequent. Due to this property, the Apriori algorithm has the ability to reduce the search space so that we can avoid the computation of non-frequent subsets. As we will not be computing support for the itemsets that are non-frequent, we will be saving a lot of computational resources. This will again lead to an easy implementation as compared to computing the support of every subset.
2. Using the frequent itemsets, Apriori algorithm generates candidate itemsets, instead of calculating the support for each and every itemset that could be possible. By doing this, we can decrease the number of candidate sets, and the focus will be particularly on the frequent itemset, which will in turn increase the efficiency.
3. Apriori algorithm is faster than that of computing the support of every subset because of its structure.
4. If we are working on a large dataset, then computing the support of every subset of an itemset will not be efficient. We can then make use of Apriori algorithm as it is more efficient.

### Part b:

Let us now answer to the question that why every frequent k-itemset must be a superset of an itemset in L1:

1. As we know, the Apriori property states that if an itemset is frequent, then all of its subsets must also be frequent, and as stated above that it helps to reduce the search space while finding frequent itemsets.
2. For K = 1, L1 represents the set of frequent 1-itemsets. Now, if we apply the Apriori property to this then we would say that, every subset of a frequent 1-itemset must also be frequent. Note that this will also include the subsets of size 0, which is known as the empty set and subsets of size 1.
3. Now, let us look at a case where we have k ≥ 2. Now, If we have a frequent k-itemset, it must include at least one item from L1 because, every item in L1 will be the frequent 1-itemset. Now, if we take any subset of the frequent k-itemset with k - 1 elements then, it is a frequent k - 1 itemset. Now, as L1 consists of frequent 1-itemsets, every subset of the frequent k-itemset with k - 1, the elements are also frequent.
4. Hence, we can say that the frequent k-itemset is the superset of the itemsets in L1. Also, we can say that the Apriori here is true.

### Part c:

Now, we will be calculating the set of candidate C3 by joining every pair of joinable itemsets from L2. Now, we know that, two itemsets can be joinable if their first k - 1 elements are same and also the last element of the first itemset is lesser than that of the second itemset.

We have L2 = {{1,2},{1,4},{2,3},{2,4},{3,5}}, now, let us compute its C3.

First we need to check whether the 2 itemsets are joinable or not: l1 = {1,2} and l2 = {1,4} --- Now, these two itemsets have the first element as same, and the last element of the first itemset is lesser than that of the second itemset. So, we can join these two itemsets.

l1 = {1,2} and l3 = {2,3} --- Now, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l1 = {1,2} and l4 = {2,4} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l1 = {1,2} and l5 = {3,5} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l2 = {1,4} and l3 = {2,3} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l2 = {1,4} and l4 = {2,4} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l2 = {1,4} and l5 = {3,5} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l3 = {2,3} and l4 = {2,4} --- Now, these two itemsets have the first element as same, and the last element of the first itemset is lesser than that of the second itemset. So, we can join these two itemsets.

l3 = {2,3} and l5 = {3,5} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

l4 = {2,4} and l5 = {3,5} --- Again, these two itemsets do not have same first element. So, no need to check for the next condition, and we will simply say that we cannot join these two itemsets.

We can see that we have only 2 pairs of itemsets that can be joined in order to create C3. Let us now join them: Joining l1 and l2 --- {1,2,4} Joining l3 and l4 --- {2,3,4}

So, our computed C3 will be C3 = {{1,2,4}, {2,3,4}}

### Part d:

If we want to know about the support of association rule then, it is nothing but the frequency of the itemsets in a dataset. We are provided with two supports of association rule, S1 and S2. S1 --- {boarding pass, passport} -> {flight} S2 --- {boarding pass} -> {flight}

In the support of association rule S1, we can interpret that it is associated with a rule where both boarding pass and the passport is mandatory to board a flight. While, in the support of association rule S2, we can interpret that it is associated with rule where only passport is mandatory to board a flight.

We can also easily interpret that S2 is a subset of S1. Since this is the case, we can say that support S1, which is getting more specific, will be having a less value than that of S2, which is also a general one.

The relationship between the two supports of association rule, S1 and S2, gives us the insights about how the dataset follows association rule. In our case, it provides us the insights on how does our dataset behaves regarding the presence of boarding pass in the dataset with or without a passport.

### Part e:

We need to calculate the support of the given rule {} -> {Eggs} in a transaction dataset given as:

[['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 support count Sa is given by Sa = Na/N, so first we will need to calculate Na:

Na is nothing but the total number of itemsets that is containing {Eggs}. We have 5 transactions, and in those 5 transactions, we have transaction 1, transaction 2, transaction 3 and transaction 5 that have {Eggs} in them. So our Na becomes 4.

N is nothing but the total number of transactions in the dataset. We have 5 transactions, so N becomes 5.

Sa gives us the support, so Sa = Na/N = 4/5 = 0.8

So, we can say that there are 80% of the transactions in the dataset that contains {Eggs} in them.

### Part f:

In order to determine the maximum length of the dataset [['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']] we will be using Apriori algorithm to do it. We will be computing frequent itemsets for the transaction dataset and continue this process until we do not have any further frequent itemsets.

Let us first compute the L1 for the dataset:

C1:

Itemset = {Milk, Onion, Nutmeg, Kidney Beans, Eggs, Yogurt, Dill, Apple, Unicorn, Corn, Ice cream}

Their support count is:

Support count = {3, 3, 2, 5, 4, 3, 1, 1, 1, 2, 1}

Support Threshold = {0.6, 0.6, 0.4, 1, 0.8, 0.6, 0.2, 0.2, 0.2, 0.4, 0.2}

As none of the itemset's support threshold is less than the minimum support threshold which is 0.2, then all the elements will be included in the L1 from the itemset.

L1 = {Milk, Onion, Nutmeg, Kidney Beans, Eggs, Yogurt, Dill, Apple, Unicorn, Corn, Ice cream}

Now, we will compute L2:

C2:

L1 = {Milk, Onion, Nutmeg, Kidney Beans, Eggs, Yogurt, Dill, Apple, Unicorn, Corn, Ice cream}

C2 is generated from L1 by joining one element with the other.

C2:

Itemset = {{Milk, Onion}, {Milk, Nutmeg}, {Milk, Kidney Beans}, {Milk, Eggs}, {Milk, Yogurt}, {Milk, Dill}, {Milk, Apple}, {Milk, Unicorn}, {Milk, Corn}, {Milk, Ice cream}, {Onion, Nutmeg}, {Onion, Kidney Beans}, {Onion, Eggs}, {Onion, Yogurt}, {Onion, Dill}, {Onion, Apple}, {Onion, Unicorn}, {Onion, Corn}, {Onion, Ice cream}, {Nutmeg, Kidney Beans}, {Nutmeg, Eggs}, {Nutmeg, Yogurt}, {Nutmeg, Dill}, {Nutmeg, Apple}, {Nutmeg, Unicorn}, {Nutmeg, Corn}, {Nutmeg, Ice cream}, {Kidney Beans, Eggs}, {Kidney Beans, Yogurt}, {Kidney Beans, Dill}, {Kidney Beans, Apple}, {Kidney Beans, Unicorn}, {Kidney Beans, Corn}, {Kidney Beans, Ice cream}, {Eggs, Yogurt}, {Eggs, Dill}, {Eggs, Apple}, {Eggs, Unicorn}, {Eggs, Corn}, {Eggs, Ice cream}, {Yogurt, Dill}, {Yogurt, Apple}, {Yogurt, Unicorn}, {Yogurt, Corn}, {Yogurt, Ice cream}, {Dill, Apple}, {Dill, Unicorn}, {Dill, Corn}, {Dill, Ice cream}, {Apple, Unicorn}, {Apple, Corn}, {Apple, Ice cream}, {Unicorn, Corn}, {Unicorn, Ice cream}, {Corn, Ice cream}}

Their support count is:

Support count = {1, 1, 3, 2, 2, 0, 1, 1, 1, 0, 2, 3, 3, 2, 1, 0, 0, 1, 1, 2, 2, 2, 1, 0, 0, 0, 0, 4, 3, 1, 1, 1, 2, 1, 2, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1}

Support Threshold = {0.2, 0.2, 0.6, 0.4, 0.4, 0, 0.2, 0.2, 0.2, 0, 0.4, 0.6, 0.6, 0.4, 0.2, 0, 0, 0.2, 0.2, 0.4, 0.4, 0.4, 0.2, 0, 0, 0, 0, 0.8, 0.6, 0.2, 0.2, 0.2, 0.4, 0.2, 0.4, 0.2, 0.2, 0, 0.2, 0.2, 0.2, 0, 0.2, 0.2, 0, 0, 0, 0, 0, 0, 0, 0, 0.2, 0, 0.2}

Let us calculate the L2 by removing the itemsets that has lower support threshold than 0.2.

L2 = {{Milk, Onion}, {Milk, Nutmeg}, {Milk, Kidney Beans}, {Milk, Eggs}, {Milk, Yogurt}, {Milk, Apple}, {Milk, Unicorn}, {Milk, Corn}, {Onion, Nutmeg}, {Onion, Kidney Beans}, {Onion, Eggs}, {Onion, Yogurt}, {Onion, Dill}, {Onion, Corn}, {Onion, Ice cream}, {Nutmeg, Kidney Beans}, {Nutmeg, Eggs}, {Nutmeg, Yogurt}, {Nutmeg, Dill}, {Kidney Beans, Eggs}, {Kidney Beans, Yogurt}, {Kidney Beans, Dill}, {Kidney Beans, Apple}, {Kidney Beans, Unicorn}, {Kidney Beans, Corn}, {Kidney Beans, Ice cream}, {Eggs, Yogurt}, {Eggs, Dill}, {Eggs, Apple}, {Eggs, Corn}, {Eggs, Ice cream}, {Yogurt, Dill}, {Yogurt, Unicorn}, {Yogurt, Corn}, {Unicorn, Corn}, {Corn, Ice cream}}

Now, we will compute L3:

C3:

L2 = {{Milk, Onion}, {Milk, Nutmeg}, {Milk, Kidney Beans}, {Milk, Eggs}, {Milk, Yogurt}, {Milk, Apple}, {Milk, Unicorn}, {Milk, Corn}, {Onion, Nutmeg}, {Onion, Kidney Beans}, {Onion, Eggs}, {Onion, Yogurt}, {Onion, Dill}, {Onion, Corn}, {Onion, Ice cream}, {Nutmeg, Kidney Beans}, {Nutmeg, Eggs}, {Nutmeg, Yogurt}, {Nutmeg, Dill}, {Kidney Beans, Eggs}, {Kidney Beans, Yogurt}, {Kidney Beans, Dill}, {Kidney Beans, Apple}, {Kidney Beans, Unicorn}, {Kidney Beans, Corn}, {Kidney Beans, Ice cream}, {Eggs, Yogurt}, {Eggs, Dill}, {Eggs, Apple}, {Eggs, Corn}, {Eggs, Ice cream}, {Yogurt, Dill}, {Yogurt, Unicorn}, {Yogurt, Corn}, {Unicorn, Corn}, {Corn, Ice cream}}

The itemset will contain the values that are joined from L2. The two itemsets will be considered as joinable if their first element is same. Below are the joinable itemsets:

{{Milk, Onion}, {Milk, Nutmeg}}

{{Milk, Onion}, {Milk, Kidney Beans}}

{{Milk, Onion}, {Milk, Eggs}}

{{Milk, Onion}, {Milk, Yogurt}}

{{Milk, Onion}, {Milk, Apple}}

{{Milk, Onion}, {Milk, Unicorn}}

{{Milk, Onion}, {Milk, Corn}}

{{Milk, Nutmeg}, {Milk, Kidney Beans}}

{{Milk, Nutmeg}, {Milk, Eggs}}

{{Milk, Nutmeg}, {Milk, Yogurt}}

{{Milk, Nutmeg}, {Milk, Apple}}

{{Milk, Nutmeg}, {Milk, Unicorn}}

{{Milk, Nutmeg}, {Milk, Corn}}

{{Milk, Kidney Beans}, {Milk, Eggs}}

{{Milk, Kidney Beans}, {Milk, Yogurt}}

{{Milk, Kidney Beans}, {Milk, Apple}}

{{Milk, Kidney Beans}, {Milk, Unicorn}}

{{Milk, Kidney Beans}, {Milk, Corn}}

{{Milk, Eggs}, {Milk, Yogurt}}

{{Milk, Eggs}, {Milk, Apple}}

{{Milk, Eggs}, {Milk, Unicorn}}

{{Milk, Eggs}, {Milk, Corn}}

{{Milk, Yogurt}, {Milk, Apple}}

{{Milk, Yogurt}, {Milk, Unicorn}}

{{Milk, Yogurt}, {Milk, Corn}}

{{Milk, Apple}, {Milk, Unicorn}}

{{Milk, Apple}, {Milk, Corn}}

{{Milk, Unicorn}, {Milk, Corn}}

{{Onion, Nutmeg}, {Onion, Kidney Beans}}

{{Onion, Nutmeg}, {Onion, Eggs}}

{{Onion, Nutmeg}, {Onion, Yogurt}}

{{Onion, Nutmeg}, {Onion, Dill}}

{{Onion, Nutmeg}, {Onion, Corn}}

{{Onion, Nutmeg}, {Onion, Ice cream}}

{{Onion, Kidney Beans}, {Onion, Eggs}}

{{Onion, Kidney Beans}, {Onion, Yogurt}}

{{Onion, Kidney Beans}, {Onion, Dill}}

{{Onion, Kidney Beans}, {Onion, Corn}}

{{Onion, Kidney Beans}, {Onion, Ice cream}}

{{Onion, Eggs}, {Onion, Yogurt}}

{{Onion, Eggs}, {Onion, Dill}}

{{Onion, Eggs}, {Onion, Corn}}

{{Onion, Eggs}, {Onion, Ice cream}}

{{Onion, Yogurt}, {Onion, Dill}}

{{Onion, Yogurt}, {Onion, Corn}}

{{Onion, Yogurt}, {Onion, Ice cream}}

{{Onion, Dill}, {Onion, Corn}}

{{Onion, Dill}, {Onion, Ice cream}}

{{Onion, Corn}, {Onion, Ice cream}}

{{Nutmeg, Kidney Beans}, {Nutmeg, Eggs}}

{{Nutmeg, Kidney Beans}, {Nutmeg, Yogurt}}

{{Nutmeg, Kidney Beans}, {Nutmeg, Dill}}

{{Nutmeg, Eggs}, {Nutmeg, Yogurt}}

{{Nutmeg, Eggs}, {Nutmeg, Dill}}

{{Nutmeg, Yogurt}, {Nutmeg, Dill}}

{{Kidney Beans, Eggs}, {Kidney Beans, Yogurt}}

{{Kidney Beans, Eggs}, {Kidney Beans, Dill}}

{{Kidney Beans, Eggs}, {Kidney Beans, Apple}}

{{Kidney Beans, Eggs}, {Kidney Beans, Unicorn}}

{{Kidney Beans, Eggs}, {Kidney Beans, Corn}}

{{Kidney Beans, Eggs}, {Kidney Beans, Ice cream}}

{{Kidney Beans, Yogurt}, {Kidney Beans, Dill}}

{{Kidney Beans, Yogurt}, {Kidney Beans, Apple}}

{{Kidney Beans, Yogurt}, {Kidney Beans, Unicorn}}

{{Kidney Beans, Yogurt}, {Kidney Beans, Corn}}

{{Kidney Beans, Yogurt}, {Kidney Beans, Ice cream}}

{{Kidney Beans, Dill}, {Kidney Beans, Apple}}

{{Kidney Beans, Dill}, {Kidney Beans, Unicorn}}

{{Kidney Beans, Dill}, {Kidney Beans, Corn}}

{{Kidney Beans, Dill}, {Kidney Beans, Ice cream}}

{{Kidney Beans, Apple}, {Kidney Beans, Unicorn}}

{{Kidney Beans, Apple}, {Kidney Beans, Corn}}

{{Kidney Beans, Apple}, {Kidney Beans, Ice cream}}

{{Kidney Beans, Unicorn}, {Kidney Beans, Corn}}

{{Kidney Beans, Unicorn}, {Kidney Beans, Ice cream}}

{{Kidney Beans, Corn}, {Kidney Beans, Ice cream}}

{{Eggs, Yogurt}, {Eggs, Dill}}

{{Eggs, Yogurt}, {Eggs, Apple}}

{{Eggs, Yogurt}, {Eggs, Corn}}

{{Eggs, Yogurt}, {Eggs, Ice cream}}

{{Eggs, Dill}, {Eggs, Apple}}

{{Eggs, Dill}, {Eggs, Corn}}

{{Eggs, Dill}, {Eggs, Ice cream}}

{{Eggs, Apple}, {Eggs, Corn}}

{{Eggs, Apple}, {Eggs, Ice cream}}

{{Eggs, Corn}, {Eggs, Ice cream}}

{{Yogurt, Dill}, {Yogurt, Unicorn}}

{{Yogurt, Dill}, {Yogurt, Corn}}

{{Yogurt, Unicorn}, {Yogurt, Corn}}

C3 = {{Milk, Onion, Nutmeg}, {Milk, Onion, Kidney Beans}, {Milk, Onion, Eggs}, {Milk, Onion, Yogurt}, {Milk, Onion, Apple}, {Milk, Onion, Unicorn}, {Milk, Onion, Corn}, {Milk, Nutmeg, Kidney Beans}, {Milk, Nutmeg, Eggs}, {Milk, Nutmeg, Yogurt}, {Milk, Nutmeg, Apple}, {Milk, Nutmeg, Unicorn}, {Milk, Nutmeg, Corn}, {Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Yogurt}, {Milk, Kidney Beans, Apple}, {Milk, Kidney Beans, Unicorn}, {Milk, Kidney Beans, Corn}, {Milk, Eggs, Yogurt}, {Milk, Eggs, Apple}, {Milk, Eggs, Unicorn}, {Milk, Eggs, Corn}, {Milk, Yogurt, Apple}, {Milk, Yogurt, Unicorn}, {Milk, Yogurt, Corn}, {Milk, Apple, Unicorn}, {Milk, Apple, Corn}, {Milk, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans}, {Onion, Nutmeg, Eggs}, {Onion, Nutmeg, Yogurt}, {Onion, Nutmeg, Dill}, {Onion, Nutmeg, Corn}, {Onion, Nutmeg, Ice cream}, {Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Yogurt}, {Onion, Kidney Beans, Dill}, {Onion, Kidney Beans, Corn}, {Onion, Kidney Beans, Ice cream}, {Onion, Eggs, Yogurt}, {Onion, Eggs, Dill}, {Onion, Eggs, Corn}, {Onion, Eggs, Ice cream}, {Onion, Yogurt, Dill}, {Onion, Yogurt, Corn}, {Onion, Yogurt, Ice cream}, {Onion, Dill, Corn}, {Onion, Dill, Ice cream}, {Onion, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs}, {Nutmeg, Kidney Beans, Yogurt}, {Nutmeg, Kidney Beans, Dill}, {Nutmeg, Eggs, Yogurt}, {Nutmeg, Eggs, Dill}, {Nutmeg, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Dill}, {Kidney Beans, Eggs, Apple}, {Kidney Beans, Eggs, Unicorn}, {Kidney Beans, Eggs, Corn}, {Kidney Beans, Eggs, Ice cream}, {Kidney Beans, Yogurt, Dill}, {Kidney Beans, Yogurt, Apple}, {Kidney Beans, Yogurt, Unicorn}, {Kidney Beans, Yogurt, Corn}, {Kidney Beans, Yogurt, Ice cream}, {Kidney Beans, Dill, Apple}, {Kidney Beans, Dill, Unicorn}, {Kidney Beans, Dill, Corn}, {Kidney Beans, Dill, Ice cream}, {Kidney Beans, Apple, Unicorn}, {Kidney Beans, Apple, Corn}, {Kidney Beans, Apple, Ice cream}, {Kidney Beans, Unicorn, Corn}, {Kidney Beans, Unicorn, Ice cream}, {Kidney Beans, Corn, Ice cream}, {Eggs, Yogurt, Dill}, {Eggs, Yogurt, Apple}, {Eggs, Yogurt, Corn}, {Eggs, Yogurt, Ice cream}, {Eggs, Dill, Apple}, {Eggs, Dill, Corn}, {Eggs, Dill, Ice cream}, {Eggs, Apple, Corn}, {Eggs, Apple, Ice cream}, {Eggs, Corn, Ice cream}, {Yogurt, Dill, Unicorn}, {Yogurt, Dill, Corn}, {Yogurt, Unicorn, Corn}}

Their support count is:

Support count = {1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 2, 2, 2, 1, 0, 0, 3, 2, 1, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1}

Support Threshold = {0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0.2, 0.2, 0.2, 0, 0, 0, 0.4, 0.4, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0.2, 0.2, 0, 0, 0.2, 0.4, 0.4, 0.4, 0.2, 0, 0, 0.6, 0.4, 0.2, 0.2, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0.2, 0.4, 0.4, 0.2, 0.4, 0.2, 0.2, 0.4, 0.2, 0.2, 0, 0.2, 0.2, 0, 0, 0, 0, 0, 0, 0, 0, 0.2, 0, 0.2, 0.2, 0, 0, 0, 0, 0, 0, 0, 0, 0.2, 0, 0, 0.2}

Let us calculate the L3 by removing the itemsets that has lower support threshold than 0.2.

L3 = {{Milk, Onion, Nutmeg}, {Milk, Onion, Kidney Beans}, {Milk, Onion, Eggs}, {Milk, Onion, Yogurt}, {Milk, Nutmeg, Kidney Beans}, {Milk, Nutmeg, Eggs}, {Milk, Nutmeg, Yogurt}, {Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Yogurt}, {Milk, Kidney Beans, Apple}, {Milk, Kidney Beans, Unicorn}, {Milk, Kidney Beans, Corn}, {Milk, Eggs, Yogurt}, {Milk, Eggs, Apple}, {Milk, Yogurt, Unicorn}, {Milk, Yogurt, Corn}, {Milk, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans}, {Onion, Nutmeg, Eggs}, {Onion, Nutmeg, Yogurt}, {Onion, Nutmeg, Dill}, {Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Yogurt}, {Onion, Kidney Beans, Dill}, {Onion, Kidney Beans, Corn}, {Onion, Kidney Beans, Ice cream}, {Onion, Eggs, Yogurt}, {Onion, Eggs, Dill}, {Onion, Eggs, Corn}, {Onion, Eggs, Ice cream}, {Onion, Yogurt, Dill}, {Onion, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs}, {Nutmeg, Kidney Beans, Yogurt}, {Nutmeg, Kidney Beans, Dill}, {Nutmeg, Eggs, Yogurt}, {Nutmeg, Eggs, Dill}, {Nutmeg, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Dill}, {Kidney Beans, Eggs, Apple}, {Kidney Beans, Eggs, Corn}, {Kidney Beans, Eggs, Ice cream}, {Kidney Beans, Yogurt, Dill}, {Kidney Beans, Yogurt, Unicorn}, {Kidney Beans, Yogurt, Corn}, {Kidney Beans, Unicorn, Corn}, {Kidney Beans, Corn, Ice cream}, {Eggs, Yogurt, Dill}, {Eggs, Corn, Ice cream}, {Yogurt, Unicorn, Corn}}

Now, we will compute L4:

C4:

L3 = {{Milk, Onion, Nutmeg}, {Milk, Onion, Kidney Beans}, {Milk, Onion, Eggs}, {Milk, Onion, Yogurt}, {Milk, Nutmeg, Kidney Beans}, {Milk, Nutmeg, Eggs}, {Milk, Nutmeg, Yogurt}, {Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Yogurt}, {Milk, Kidney Beans, Apple}, {Milk, Kidney Beans, Unicorn}, {Milk, Kidney Beans, Corn}, {Milk, Eggs, Yogurt}, {Milk, Eggs, Apple}, {Milk, Yogurt, Unicorn}, {Milk, Yogurt, Corn}, {Milk, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans}, {Onion, Nutmeg, Eggs}, {Onion, Nutmeg, Yogurt}, {Onion, Nutmeg, Dill}, {Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Yogurt}, {Onion, Kidney Beans, Dill}, {Onion, Kidney Beans, Corn}, {Onion, Kidney Beans, Ice cream}, {Onion, Eggs, Yogurt}, {Onion, Eggs, Dill}, {Onion, Eggs, Corn}, {Onion, Eggs, Ice cream}, {Onion, Yogurt, Dill}, {Onion, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs}, {Nutmeg, Kidney Beans, Yogurt}, {Nutmeg, Kidney Beans, Dill}, {Nutmeg, Eggs, Yogurt}, {Nutmeg, Eggs, Dill}, {Nutmeg, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Dill}, {Kidney Beans, Eggs, Apple}, {Kidney Beans, Eggs, Corn}, {Kidney Beans, Eggs, Ice cream}, {Kidney Beans, Yogurt, Dill}, {Kidney Beans, Yogurt, Unicorn}, {Kidney Beans, Yogurt, Corn}, {Kidney Beans, Unicorn, Corn}, {Kidney Beans, Corn, Ice cream}, {Eggs, Yogurt, Dill}, {Eggs, Corn, Ice cream}, {Yogurt, Unicorn, Corn}}

The itemset will contain the values that are joined from L3. The two itemsets will be considered as joinable if their k - 1 elements are same. Below are the joinable itemsets:

{{Milk, Onion, Nutmeg}, {Milk, Onion, Kidney Beans}}

{{Milk, Onion, Nutmeg}, {Milk, Onion, Eggs}}

{{Milk, Onion, Nutmeg}, {Milk, Onion, Yogurt}}

{{Milk, Onion, Kidney Beans}, {Milk, Onion, Eggs}}

{{Milk, Onion, Kidney Beans}, {Milk, Onion, Yogurt}}

{{Milk, Onion, Eggs}, {Milk, Onion, Yogurt}}

{{Milk, Nutmeg, Kidney Beans}, {Milk, Nutmeg, Eggs}}

{{Milk, Nutmeg, Kidney Beans}, {Milk, Nutmeg, Yogurt}}

{{Milk, Nutmeg, Eggs}, {Milk, Nutmeg, Yogurt}}

{{Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Yogurt}}

{{Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Apple}}

{{Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Unicorn}}

{{Milk, Kidney Beans, Eggs}, {Milk, Kidney Beans, Corn}}

{{Milk, Kidney Beans, Yogurt}, {Milk, Kidney Beans, Apple}}

{{Milk, Kidney Beans, Yogurt}, {Milk, Kidney Beans, Unicorn}}

{{Milk, Kidney Beans, Yogurt}, {Milk, Kidney Beans, Corn}}

{{Milk, Kidney Beans, Apple}, {Milk, Kidney Beans, Unicorn}}

{{Milk, Kidney Beans, Apple}, {Milk, Kidney Beans, Corn}}

{{Milk, Kidney Beans, Unicorn}, {Milk, Kidney Beans, Corn}}

{{Milk, Eggs, Yogurt}, {Milk, Eggs, Apple}}

{{Milk, Yogurt, Unicorn}, {Milk, Yogurt, Corn}}

{{Onion, Nutmeg, Kidney Beans}, {Onion, Nutmeg, Eggs}}

{{Onion, Nutmeg, Kidney Beans}, {Onion, Nutmeg, Yogurt}}

{{Onion, Nutmeg, Kidney Beans}, {Onion, Nutmeg, Dill}}

{{Onion, Nutmeg, Eggs}, {Onion, Nutmeg, Yogurt}}

{{Onion, Nutmeg, Eggs}, {Onion, Nutmeg, Dill}}

{{Onion, Nutmeg, Yogurt}, {Onion, Nutmeg, Dill}}

{{Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Yogurt}}

{{Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Dill}}

{{Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Corn}}

{{Onion, Kidney Beans, Eggs}, {Onion, Kidney Beans, Ice cream}}

{{Onion, Kidney Beans, Yogurt}, {Onion, Kidney Beans, Dill}}

{{Onion, Kidney Beans, Yogurt}, {Onion, Kidney Beans, Corn}}

{{Onion, Kidney Beans, Yogurt}, {Onion, Kidney Beans, Ice cream}}

{{Onion, Kidney Beans, Dill}, {Onion, Kidney Beans, Corn}}

{{Onion, Kidney Beans, Dill}, {Onion, Kidney Beans, Ice cream}}

{{Onion, Kidney Beans, Corn}, {Onion, Kidney Beans, Ice cream}}

{{Onion, Eggs, Yogurt}, {Onion, Eggs, Dill}}

{{Onion, Eggs, Yogurt}, {Onion, Eggs, Corn}}

{{Onion, Eggs, Yogurt}, {Onion, Eggs, Ice cream}}

{{Onion, Eggs, Dill}, {Onion, Eggs, Corn}}

{{Onion, Eggs, Dill}, {Onion, Eggs, Ice cream}}

{{Onion, Eggs, Corn}, {Onion, Eggs, Ice cream}}

{{Nutmeg, Kidney Beans, Eggs}, {Nutmeg, Kidney Beans, Yogurt}}

{{Nutmeg, Kidney Beans, Eggs}, {Nutmeg, Kidney Beans, Dill}}

{{Nutmeg, Kidney Beans, Yogurt}, {Nutmeg, Kidney Beans, Dill}}

{{Nutmeg, Eggs, Yogurt}, {Nutmeg, Eggs, Dill}}

{{Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Dill}}

{{Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Apple}}

{{Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Corn}}

{{Kidney Beans, Eggs, Yogurt}, {Kidney Beans, Eggs, Ice cream}}

{{Kidney Beans, Eggs, Dill}, {Kidney Beans, Eggs, Apple}}

{{Kidney Beans, Eggs, Dill}, {Kidney Beans, Eggs, Corn}}

{{Kidney Beans, Eggs, Dill}, {Kidney Beans, Eggs, Ice cream}}

{{Kidney Beans, Eggs, Apple}, {Kidney Beans, Eggs, Corn}}

{{Kidney Beans, Eggs, Apple}, {Kidney Beans, Eggs, Ice cream}}

{{Kidney Beans, Eggs, Corn}, {Kidney Beans, Eggs, Ice cream}}

{{Kidney Beans, Yogurt, Dill}, {Kidney Beans, Yogurt, Unicorn}}

{{Kidney Beans, Yogurt, Dill}, {Kidney Beans, Yogurt, Corn}}

{{Kidney Beans, Yogurt, Unicorn}, {Kidney Beans, Yogurt, Corn}}

C4 = {{Milk, Onion, Nutmeg, Kidney Beans}, {Milk, Onion, Nutmeg, Eggs}, {Milk, Onion, Nutmeg, Yogurt}, {Milk, Onion, Kidney Beans, Eggs}, {Milk, Onion, Kidney Beans, Yogurt}, {Milk, Onion, Eggs, Yogurt}, {Milk, Nutmeg, Kidney Beans, Eggs}, {Milk, Nutmeg, Kidney Beans, Yogurt}, {Milk, Nutmeg, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Apple}, {Milk, Kidney Beans, Eggs, Unicorn}, {Milk, Kidney Beans, Eggs, Corn}, {Milk, Kidney Beans, Yogurt, Apple}, {Milk, Kidney Beans, Yogurt, Unicorn}, {Milk, Kidney Beans, Yogurt, Corn}, {Milk, Kidney Beans, Apple, Unicorn}, {Milk, Kidney Beans, Apple, Corn}, {Milk, Kidney Beans, Unicorn, Corn}, {Milk, Eggs, Yogurt, Apple}, {Milk, Yogurt, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans, Eggs}, {Onion, Nutmeg, Kidney Beans, Yogurt}, {Onion, Nutmeg, Kidney Beans, Dill}, {Onion, Nutmeg, Eggs, Yogurt}, {Onion, Nutmeg, Eggs, Dill}, {Onion, Nutmeg, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt}, {Onion, Kidney Beans, Eggs, Dill}, {Onion, Kidney Beans, Eggs, Corn}, {Onion, Kidney Beans, Eggs, Ice cream}, {Onion, Kidney Beans, Yogurt, Dill}, {Onion, Kidney Beans, Yogurt, Corn}, {Onion, Kidney Beans, Yogurt, Ice cream}, {Onion, Kidney Beans, Dill, Corn}, {Onion, Kidney Beans, Dill, Ice cream}, {Onion, Kidney Beans, Corn, Ice cream}, {Onion, Eggs, Yogurt, Dill}, {Onion, Eggs, Yogurt, Corn}, {Onion, Eggs, Yogurt, Ice cream}, {Onion, Eggs, Dill, Corn}, {Onion, Eggs, Dill, Ice cream}, {Onion, Eggs, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs, Yogurt}, {Nutmeg, Kidney Beans, Eggs, Dill}, {Nutmeg, Kidney Beans, Yogurt, Dill}, {Nutmeg, Eggs, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt, Apple}, {Kidney Beans, Eggs, Yogurt, Corn}, {Kidney Beans, Eggs, Yogurt, Ice cream}, {Kidney Beans, Eggs, Dill, Apple}, {Kidney Beans, Eggs, Dill, Corn}, {Kidney Beans, Eggs, Dill, Ice cream}, {Kidney Beans, Eggs, Apple, Corn}, {Kidney Beans, Eggs, Apple, Ice cream}, {Kidney Beans, Eggs, Corn, Ice cream}, {Kidney Beans, Yogurt, Dill, Unicorn}, {Kidney Beans, Yogurt, Dill, Corn}, {Kidney Beans, Yogurt, Unicorn, Corn}}

Their support count is:

Support count = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1}

Support Threshold = {0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0.2, 0.2, 0, 0, 0.2, 0, 0.2, 0.4, 0.4, 0.2, 0.4, 0.2, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0.2, 0.2, 0, 0, 0, 0, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0, 0, 0, 0.2, 0, 0, 0.2}

Let us calculate the L4 by removing the itemsets that has lower support threshold than 0.2.

L4 = {{Milk, Onion, Nutmeg, Kidney Beans}, {Milk, Onion, Nutmeg, Eggs}, {Milk, Onion, Nutmeg, Yogurt}, {Milk, Onion, Kidney Beans, Eggs}, {Milk, Onion, Kidney Beans, Yogurt}, {Milk, Onion, Eggs, Yogurt}, {Milk, Nutmeg, Kidney Beans, Eggs}, {Milk, Nutmeg, Kidney Beans, Yogurt}, {Milk, Nutmeg, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Apple}, {Milk, Kidney Beans, Yogurt, Unicorn}, {Milk, Kidney Beans, Yogurt, Corn}, {Milk, Kidney Beans, Unicorn, Corn}, {Milk, Yogurt, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans, Eggs}, {Onion, Nutmeg, Kidney Beans, Yogurt}, {Onion, Nutmeg, Kidney Beans, Dill}, {Onion, Nutmeg, Eggs, Yogurt}, {Onion, Nutmeg, Eggs, Dill}, {Onion, Nutmeg, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt}, {Onion, Kidney Beans, Eggs, Dill}, {Onion, Kidney Beans, Eggs, Corn}, {Onion, Kidney Beans, Eggs, Ice cream}, {Onion, Kidney Beans, Yogurt, Dill}, {Onion, Kidney Beans, Corn, Ice cream}, {Onion, Eggs, Yogurt, Dill}, {Onion, Eggs, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs, Yogurt}, {Nutmeg, Kidney Beans, Eggs, Dill}, {Nutmeg, Kidney Beans, Yogurt, Dill}, {Nutmeg, Eggs, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt, Dill}, {Kidney Beans, Eggs, Corn, Ice cream}, {Kidney Beans, Yogurt, Unicorn, Corn}}

Now, we will compute L5:

C5:

L4 = {{Milk, Onion, Nutmeg, Kidney Beans}, {Milk, Onion, Nutmeg, Eggs}, {Milk, Onion, Nutmeg, Yogurt}, {Milk, Onion, Kidney Beans, Eggs}, {Milk, Onion, Kidney Beans, Yogurt}, {Milk, Onion, Eggs, Yogurt}, {Milk, Nutmeg, Kidney Beans, Eggs}, {Milk, Nutmeg, Kidney Beans, Yogurt}, {Milk, Nutmeg, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Apple}, {Milk, Kidney Beans, Yogurt, Unicorn}, {Milk, Kidney Beans, Yogurt, Corn}, {Milk, Kidney Beans, Unicorn, Corn}, {Milk, Yogurt, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans, Eggs}, {Onion, Nutmeg, Kidney Beans, Yogurt}, {Onion, Nutmeg, Kidney Beans, Dill}, {Onion, Nutmeg, Eggs, Yogurt}, {Onion, Nutmeg, Eggs, Dill}, {Onion, Nutmeg, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt}, {Onion, Kidney Beans, Eggs, Dill}, {Onion, Kidney Beans, Eggs, Corn}, {Onion, Kidney Beans, Eggs, Ice cream}, {Onion, Kidney Beans, Yogurt, Dill}, {Onion, Kidney Beans, Corn, Ice cream}, {Onion, Eggs, Yogurt, Dill}, {Onion, Eggs, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs, Yogurt}, {Nutmeg, Kidney Beans, Eggs, Dill}, {Nutmeg, Kidney Beans, Yogurt, Dill}, {Nutmeg, Eggs, Yogurt, Dill}, {Kidney Beans, Eggs, Yogurt, Dill}, {Kidney Beans, Eggs, Corn, Ice cream}, {Kidney Beans, Yogurt, Unicorn, Corn}}

The itemset will contain the values that are joined from L4. The two itemsets will be considered as joinable if their k - 1 elements are same. Below are the joinable itemsets:

{{Milk, Onion, Nutmeg, Kidney Beans}, {Milk, Onion, Nutmeg, Eggs}}

{{Milk, Onion, Nutmeg, Kidney Beans}, {Milk, Onion, Nutmeg, Yogurt}}

{{Milk, Onion, Nutmeg, Eggs}, {Milk, Onion, Nutmeg, Yogurt}}

{{Milk, Onion, Kidney Beans, Eggs}, {Milk, Onion, Kidney Beans, Yogurt}

{{Milk, Nutmeg, Kidney Beans, Eggs}, {Milk, Nutmeg, Kidney Beans, Yogurt}}

{{Milk, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Apple}}

{{Milk, Kidney Beans, Yogurt, Unicorn}, {Milk, Kidney Beans, Yogurt, Corn}}

{{Onion, Nutmeg, Kidney Beans, Eggs}, {Onion, Nutmeg, Kidney Beans, Yogurt}}

{{Onion, Nutmeg, Kidney Beans, Eggs}, {Onion, Nutmeg, Kidney Beans, Dill}}

{{Onion, Nutmeg, Kidney Beans, Yogurt}, {Onion, Nutmeg, Kidney Beans, Dill}}

{{Onion, Nutmeg, Eggs, Yogurt}, {Onion, Nutmeg, Eggs, Dill}}

{{Onion, Kidney Beans, Eggs, Yogurt}, {Onion, Kidney Beans, Eggs, Dill}}

{{Onion, Kidney Beans, Eggs, Yogurt}, {Onion, Kidney Beans, Eggs, Corn}}

{{Onion, Kidney Beans, Eggs, Yogurt}, {Onion, Kidney Beans, Eggs, Ice cream}}

{{Onion, Kidney Beans, Eggs, Dill}, {Onion, Kidney Beans, Eggs, Corn}}

{{Onion, Kidney Beans, Eggs, Dill}, {Onion, Kidney Beans, Eggs, Ice cream}}

{{Onion, Kidney Beans, Eggs, Corn}, {Onion, Kidney Beans, Eggs, Ice cream}}

{{Nutmeg, Kidney Beans, Eggs, Yogurt}, {Nutmeg, Kidney Beans, Eggs, Dill}}

C5 = {{Milk, Onion, Nutmeg, Kidney Beans, Eggs}, {Milk, Onion, Nutmeg, Kidney Beans, Yogurt}, {Milk, Onion, Nutmeg, Eggs, Yogurt}, {Milk, Onion, Kidney Beans, Eggs, Yogurt}, {Milk, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Eggs, Yogurt, Apple}, {Milk, Kidney Beans, Yogurt, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Dill}, {Onion, Nutmeg, Kidney Beans, Yogurt, Dill}, {Onion, Nutmeg, Eggs, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt, Corn}, {Onion, Kidney Beans, Eggs, Yogurt, Ice cream}, {Onion, Kidney Beans, Eggs, Dill, Corn}, {Onion, Kidney Beans, Eggs, Dill, Ice cream}, {Onion, Kidney Beans, Eggs, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs, Yogurt, Dill}}

Their support count is:

Support count = {1, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1}

Support Threshold = {0.2, 0.2, 0.2, 0.2, 0.2, 0, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0.2, 0.2}

Let us calculate the L5 by removing the itemsets that has lower support threshold than 0.2.

L5 = {{Milk, Onion, Nutmeg, Kidney Beans, Eggs}, {Milk, Onion, Nutmeg, Kidney Beans, Yogurt}, {Milk, Onion, Nutmeg, Eggs, Yogurt}, {Milk, Onion, Kidney Beans, Eggs, Yogurt}, {Milk, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Yogurt, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Dill}, {Onion, Nutmeg, Kidney Beans, Yogurt, Dill}, {Onion, Nutmeg, Eggs, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs, Yogurt, Dill}}

Now, we will compute L6:

C6:

L5 = {{Milk, Onion, Nutmeg, Kidney Beans, Eggs}, {Milk, Onion, Nutmeg, Kidney Beans, Yogurt}, {Milk, Onion, Nutmeg, Eggs, Yogurt}, {Milk, Onion, Kidney Beans, Eggs, Yogurt}, {Milk, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Milk, Kidney Beans, Yogurt, Unicorn, Corn}, {Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Dill}, {Onion, Nutmeg, Kidney Beans, Yogurt, Dill}, {Onion, Nutmeg, Eggs, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Yogurt, Dill}, {Onion, Kidney Beans, Eggs, Corn, Ice cream}, {Nutmeg, Kidney Beans, Eggs, Yogurt, Dill}}

The itemset will contain the values that are joined from L5. The two itemsets will be considered as joinable if their k - 1 elements are same. Below are the joinable itemsets:

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

{{Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Dill}}

C6 = {{Milk, Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Yogurt, Dill}}

Their support count is:

Support count = {1, 1}

Support Threshold = {0.2, 0.2}

Let us calculate the L6 by removing the itemsets that has lower support count than 0.2.

L6 = {{Milk, Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Yogurt, Dill}}

Now, we will compute L7:

C7:

L6 = {{Milk, Onion, Nutmeg, Kidney Beans, Eggs, Yogurt}, {Onion, Nutmeg, Kidney Beans, Eggs, Yogurt, Dill}}

The itemset will contain the values that are joined from L6. The two itemsets will be considered as joinable if their k - 1 elements are same. As there are no joinable sets, we cannot generate C7 and if there is no C7, we cannot compute L7.

Hence we can say here that the maximum length of the frequent itemset for a support threshold of 0.2 will be 6.

## Question 2:

**For your answers to the assignment, please include your workings (e.g. equations, code) when this is relevant to the question. Questions 2a & 2b are pen and paper exercises. Question 2c can be addressed either on paper or using code. Question 2d is a coding exercise.**

1. **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 behavioural attributes that could be integrated into this system's algorithm. Provide a rationale for classifying each attribute as either contextual or behavioural.**
2. **Assume that you are provided with the [University of Wisconsin breast cancer dataset](https://archive.ics.uci.edu/ml/machine-learning-databases/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/breast-cancer-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 would you choose this particular method for outlier detection.**
3. **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.**
4. **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?**

### Part a:

Rationale for classifying attribute as Contextual:

1. Time: The transactions that might occur at unusual hours of the day, compared to the account holder's activity, can be considered as "Identity Thefts". To detect these types of thefts, we can consider the history of the account holder's transactions and use that history to compare it with the time where the theft could have taken place. By doing this, we can get the unusual transactions through hours or days, by raising an alarm if there is any previous transactions at that specific hour or time of the day.

2. Location: The transactions that might occur from an unusual location, compared to the account holder's activity, can be considered as "Identity Thefts". In order to detect these, we can compare the history of the transactions of that particular account holder, and raise an alarm if there is a transaction performed at an un-common location used by the holder.

Rationale for classifying attribute as Behavioral:

1. Deviations in transactions: The transactions that might occur with unusual amount spent by the account holder, compared to their activity history. These can also be seen as "Identity Thefts". To detect these, we can look at the spending history of the account holder's transactions and the use this information to determine the sudden spike in the transaction pattern.

2. Repetitive transactions: The transactions that might occur with spikes or drop that are unusual as compared to that of the account holder's history, will also be seen as "Identity Thefts". We can detect these by monitoring the frequencies in the transaction history of the account holder. We can also detect it by monitoring the frequencies in the transactions with a particular account.

### Part b:

Looking at the dataset and its attributes, we can use the "Proximity-Based Approach" in order to detect the outliers in the "Breast Cancer Dataset". Let us first look into the definition of outliers in the context of the "Breast Cancer Dataset".

Outlier definition for Breast Cancer Dataset:

In this context, we can define the outlier as the one that do not follow the pattern which other patients are following. In the dataset, we have an attribute called "Sample code number", which is nothing but the "id number", we also have an attribute called "Class", which specifies the class in which the particular patient belongs, the rest of the other 9 attributes are represented in the form of numbers ranging from 1-10. Now, we can say that since the attribute "Sample code number" is a number associated with the patient and will vary from patient to patient, and also the attribute "Class" is specified as either 2 or 4 and nothing other than that, so we cannot determine the outliers based on these two attributes. However, we can use the remaining 9 attributes to determine the outliers. Let us consider a single attribute to determine the outlier say "Clump Thickness", this attribute ranges from 1 to 10, but if in the dataset, we are having the values of this attribute lying in the range of 3 to 6 and if in for a few patients, the value of this attribute is 10. These patients will be considered as outliers. We can also consider all the 9 attributes while determining the outliers. As we said in the definition, that the outlier is the one that will not follow the pattern, so we can look at the patterns created by the attributes, and specify the outliers that will not follow them. While doing this we also need to remember that the outliers are very few in number, and hence if the pattern that you have determined is not followed by a large number of patients, then we need to re-check the pattern.

Let us now delve further into what exactly will we do in order to determine the outliers using "Proximity-Based Approach".

Use of Proximity-Based Approach:

We will first calculate the distance for each data points (in our case patients) from their nearest neighbour. We will also need to determine the threshold beyond which the patient will be considered as an outlier. This would be easy to determine if we are considering a single attribute like "Clump Thickness", which for instance lies in the range of 3 to 6 then the threshold could be set as below 2 and above 7 (for instance). Now, if a few patients have value for this attribute, as 10, then they are having the value beyond threshold and hence will be considered as outliers. We are able to do this because we know that the data points that are close to each other, will have similar characteristics, and if not, then they are the outliers. In case of "Proximity-Based Approach", we calculate the distance between the two points, and this distance can be calculated through Euclidean distance.

Let us now justify, why we have chosen the "Proximity-Based Approach" for outlier detection.

Justification:

We can say that the non-cancerous patients will have similar characteristics and cancerous patients will have similar characteristics, so we can easily determine the outliers that do not share either of these characteristics. As Proximity-Based Approach is a distance-based approach, hence it will never assume anything regarding the underlying distribution of the data points and hence we can use these approaches for different datasets also. Here, we are determining outliers using distance between data points, which is easy to interpret. It is also easy to implement. In healthcare types of datasets, the outliers can be considered as a representation in which they are deviating from a normal data points. If we detect these outliers at an early stage, we can quickly treat the patient.

### Part c:

To identify the outliers using "Maximum Likelihood Method", we need to first understand this method.

Maximum Likelihood Method:

While determining outliers, this method assumes that data is in the form of normal distribution. In this context, the outliers will be determined by computing the likelihood of each of the data points in the dataset. This is done by using the mean and standard deviation of the normal distribution. We also need to set a threshold value for this, and if a data point is below this threshold, then it is considered as an outlier.

We will now look at the implementation of "Maximum Likelihood Method".

Implementation:

First we calculate the mean and standard deviation of the normal distribution.

Here, Xi is the data point and n is the total number of data points.

So, we will have Mean =

= (270.42)/12

= 22.535

Standard Deviation = )

= ((450.9295)/12)

= (37.577458333333)

= 6.1300455408858

Now, we will calculate the likelihood for each data point using the PDF of normal distribution:

Likelihood(Xi)=​

Likelihood(X1) = ​

Likelihood(X2) = ​

Likelihood(X3) = ​

Likelihood(X4) = ​

Likelihood(X5) = ​

Likelihood(X6) = ​

Likelihood(X7) = ​

Likelihood(X8) = ​

Likelihood(X9) = ​

Likelihood(X10) = ​

Likelihood(X11) = ​

Likelihood(X12) = ​

This is difficult to calculate manually, so let us compute it using the below code.

We have also specified the threshold value, in order to determine the outliers.

# Given data

rainfall\_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])

# Fit normal distribution

mu, sigma = norm.fit(rainfall\_data)

# Calculate likelihood for each data point

likelihoods = norm.pdf(rainfall\_data, mu, sigma)

# Set a threshold for likelihood

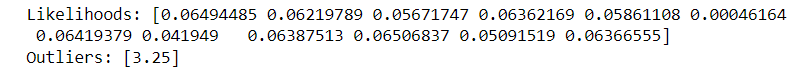
threshold = 0.03

# Identify outliers

outliers = rainfall\_data[likelihoods < threshold]

print("Likelihoods:", likelihoods)

print("Outliers:", outliers)



In the above code, we can see that the likelihood "0.00046164 is below the threshold value, and hence it is the outlier in the dataset.

### Part d:

Here, we are using the stocks.csv dataset to determine the outliers in that dataset by using one-class SVM approach.

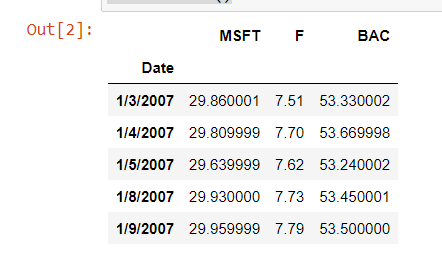
# Load CSV file, set the 'Date' values as the index of each row, and display the first rows of the dataframe

stocks = pd.read\_csv('stocks.csv', header='infer')

stocks.index = stocks['Date']

stocks = stocks.drop(['Date'],axis=1)

stocks.head()



Here, we have extracted the CSV file. The column "Date" is specified as the index of the dataset and then it is dropped because we do not want two columns specifying the same attribute.

Now, as required we need to compute the percentage changes in the daily closing price for each stock.

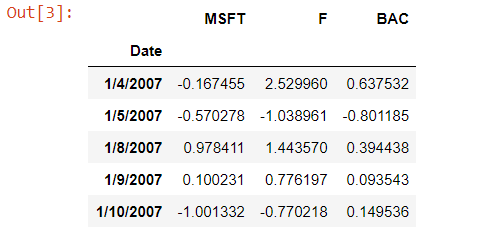
N,d = stocks.shape

# Compute delta, which denotes the percentage of changes in the daily closing price of each stock

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)

delta.head()



We have number of columns and rows from the dataset's shape. The delta is showing the percentage changes in the daily closing price of each stock. The percentage changes computed are stored in cells of delta.

To plot the 3D scatterplot, we will use one-class SVM as required.

# Initialize and train the one-class SVM classifier

ee = OneClassSVM(nu=0.01, gamma='auto')

yhat = ee.fit\_predict(delta)

# 3D scatterplot of the dataset, color-coded for outliers and inliers

fig = plt.figure(figsize=(10, 6))

ax = fig.add\_subplot(111, projection='3d')

p = ax.scatter(delta['MSFT'], delta['F'], delta['BAC'], c=yhat, cmap='jet')

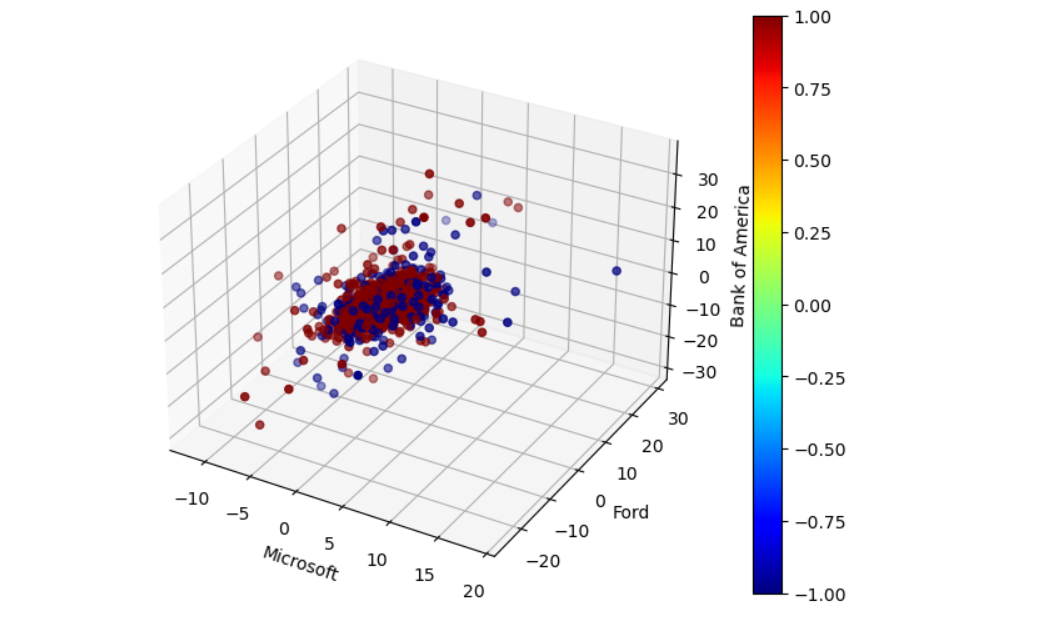
ax.set\_xlabel('Microsoft')

ax.set\_ylabel('Ford')

ax.set\_zlabel('Bank of America')

fig.colorbar(p)

plt.show()



Here, "nu" is the hyperparameter that controls the proportion of outliers. the "gamma" is also a hyperparameter which is set to auto. The x, y and z co-ordinates of the 3D plot is given by the percentage change values for "Microsoft", "Ford" and "Bank of America". The predicted labels given by "yhat" is nothing but the color in the plot which is assigned by the one-class SVM. The outliers and inliers are colored differently.

We will now be computing the frequencies of the outliers and inliers.

outlier\_frequency = np.sum(yhat == -1)

inlier\_frequency = np.sum(yhat == 1)

print(f'Frequency of Outliers: {outlier\_frequency}')

print(f'Frequency of Inliers: {inlier\_frequency}')



In order to calculate the frequencies, we have first computed the value of total number of occurrences where the "yhat" is -1 (for outliers) and 1 (for inliers).

Now, we need to specify, how is one-class SVM approach different from both parametric and proximity-based methods. Let us see:

One- class SVM Vs. Parametric method Vs. Proximity-Based method:

i. In one-class SVM, the approach usually assumes that the inliers are the only class to which the data points belong. The other data points that will somehow deviate from those inliers will be classified as outliers. On the other hand, in parametric method, the method usually assumes that the data points follow a normal distribution or any other specific distribution and then it will identify the outliers as the points that do not follow the same distribution. In the third is the proximity-based method, we have used KNN to identify the outliers that are away from that of the other data points.

ii. In one-class SVM, the data points are bound by a boundary in the feature space. The points that will fall outside that boundary will be classified as outliers. In parametric method, the statistical measures are used to determine the outliers in the dataset. We have computed mean and covariance for that. In proximity-based method, we make use of distances to identify the outliers.

iii. In one-class SVM we have tuned the hyperparameters "nu" and "gamma" and their choices will impact the performance. The parametric methods are sensitive due to assumptions regarding the distribution of data points. The proximity-based methods depends on the choice of k.

iv. In the first case, i.e. the one-class SVM we can see that there are two different colors, one for inliers and other for outliers. However, in the second case, i.e. the parametric method we can see that there is a color bar and the data points that are represented in darker color are classified as outliers. In the third case, i.e. the proximity-based method is similar to that of the second case. This is because both parametric and proximity methods are using distances to determine the outliers.

Now, we need to specify how much percentage of the dataset are classified as outliers.

total\_objects = len(yhat)

print(f'Percentage of Outliers: {(outlier\_frequency / total\_objects) \* 100:.2f}%')

print(f'Percentage of Inliers: {(inlier\_frequency / total\_objects) \* 100:.2f}%')



We have computed the percentage of the dataset that are outliers and inliers by dividing each of their frequencies by the total number of data points and then multiplying it by 100.

We can see that 17.8% of the data points are outliers in our dataset.

## Question 3:

**Questions 3(a)(I) is a pen and paper exercise and 3(a)(II) is a coding exercise. Questions 3(b) is a pen-and-paper exercise.**

1. **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.**

1. **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?**

**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).**

1. **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?**

A diagram of a diagram

Description automatically generated

### Part a (i):

We can see that the HTML code contains 10 unique tags. Let us look at them:

1. <html> : This is the root of the HTML.
2. <body> : This tag has all the contents of the HTML document.
3. <h1> : This is the heading tag, which is the top level heading tag. In our case, the heading is "ECS766P Data Mining - Week 10".
4. <p> : This tag is used for paragraph writing. In our case it has the description of the table.
5. <table> : This tag writes a table. We use this in order to represent data in the form of rows and columns.
6. <thead> : This tag is used to specify the heading of the table.
7. <tr> : We can define a table row using this tag.
8. <th> : The header cell can be defined using this tag.
9. <tbody> : The body of the table will be defined using this tag.
10. <td> : The data in the cell is defined.

### Part a (ii):

# Specifying the correct URL

url = "http://eecs.qmul.ac.uk/~emmanouilb/income\_table.html"

html = urlopen(url)

# Creating a BeautifulSoup object

soup = BeautifulSoup(html, 'html.parser')

# Extracting table rows

rows = soup.find\_all('tr')

# Extracting table values

table\_list = []

for row in rows:

row\_cells = row.find\_all(['td', 'th'])

row\_values = [cell.get\_text(strip=True) for cell in row\_cells]

table\_list.append(row\_values)

# Creating a DataFrame from the table values

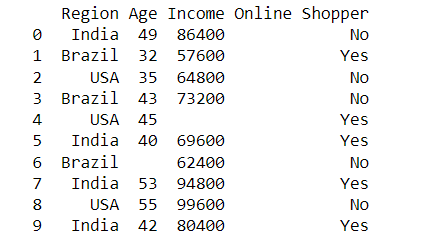
df = pd.DataFrame(table\_list[1:], columns=table\_list[0])

# Cleaning up the DataFrame

df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)

# Print the resulting DataFrame

print(df)



Here, we have used BeautifulSoup library in order to extract the data from the HTML URL. We specify the URL that we need to extract the contents from and "urlopen" function opens the URL which results the response from HTML. Then we create the BeautifulSoup's object by passing the HTML content with the help of "html.parser". We then use the "find\_all" method in order to find all the table rows in the HTML page i.e. "" tags. The for loop will iterate the rows i.e. "" tags and also finding the "" and "" tags. The extracted values are stored in a separate list variable. The extracted list is then converted to a dataframe. The "applymap" function is used to make a lambda function that will help us in removing the leading and trailing whitespaces.

### Part b:

In the figure, we can see that there are 5 nodes, each having a link to some other node. In these nodes, we need to find which of them are hubs and which of them are authorities. Let us first get to know what are the hubs and authorities.

Hubs:

In simple terms, a hub is a type of node or a page, that has many outgoing links. These links generally attach to the authorities. The hubs point to various resources.

Authorities:

In simple terms, an authority is a type of node or a page, that has many incoming links. These links come from different hubs. The authorities are the pages that are known as frequently reviewed pages.

In this scenario, we can say the following things: 1. Nodes 3, 4 and 5 are hubs. This is because, there are a few links that are going out from them. 2. Nodes 1 and 2 are authorities. This is because, there are a few links that are coming to them. 3. The hubs and authorities are interconnected in such a way that all 3 hubs have outgoing nodes that are connected to each of the authorities. So, 2 links are outgoing from each of the hubs and 3 links are incoming to each of the authorities.

## Question 4:

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

1. **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.**

1. **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.**

**ii. Using the above constructed document-term matrix, calculate the inverse document frequency for all words you have identified from the previous question (i).**

1. **Using the daily births dataset from Week 11 lab notebook, smooth the timeseries using trailing moving average smoothing and a window size that corresponds to one week; then replace any NaN values with zeros. Perform timeseries 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   
   ARMA model with and**

### Part a (i):

Here, we need to create a document-term matrix for the below sentences:

\* 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

Let us see that what will be the words that we can use to create a document-term matrix:

1. We need to remove the stop word, which are: "to, that, are, a, can, be, as, of, by, or, and".

2. We need to remove the punctuation marks, in our case the full stops.

3. We need to convert the plural nouns or adjectives to its singular form and convert verbs to their present tense and first-person singular form, which in our case: "refers" = "refer", "collected" = "collect", "viewed" = "view", "described" = "describe".

Now, the document-term matrix will look like below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Word | data | refer | characteristic | collect | observation | dataset |
| Document 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Document 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Document 3 | 1 | 0 | 0 | 0 | 0 | 0 |
| Document 4 | 0 | 0 | 1 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Word | view | collection | object | describe | number | attribute |
| Document 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Document 2 | 1 | 1 | 1 | 0 | 0 | 0 |
| Document 3 | 0 | 0 | 1 | 1 | 1 | 1 |
| Document 4 | 0 | 0 | 1 | 0 | 0 | 1 |

|  |  |
| --- | --- |
| Word | feature |
| Document 1 | 0 |
| Document 2 | 0 |
| Document 3 | 0 |
| Document 4 | 1 |

### Part a (ii):

Here we will be calculating the Inverse Document Frequency (IDF) for each word w. The formula is shows below:

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

idf(data) = log10(|4|/|2|) = log10(2) = 0.3010

idf(refer) = log10(|4|/|1|) = log10(4) = 0.6021

idf(characteristic) = log10(|4|/|2|) = log10(2) = 0.3010

idf(collect) = log10(|4|/|1|) = log10(4) = 0.6021

idf(observation) = log10(|4|/|1|) = log10(4) = 0.6021

idf(dataset) = log10(|4|/|1|) = log10(4) = 0.6021

idf(view) = log10(|4|/|1|) = log10(4) = 0.6021

idf(collection) = log10(|4|/|1|) = log10(4) = 0.6021

idf(object) = log10(|4|/|3|) = log10(1.3333) = 0.1249

idf(describe) = log10(|4|/|1|) = log10(4) = 0.6021

idf(number) = log10(|4|/|1|) = log10(4) = 0.6021

idf(attribute) = log10(|4|/|2|) = log10(2) = 0.3010

idf(feature) = log10(|4|/|1|) = log10(4) = 0.6021

### Part b:

# Load the dataset

series = pd.read\_csv('births.csv', header=0, index\_col=0)

We have loaded the CSV file. It assumes that the dataset has a header, and the first column contains dates (index\_col=0).

# Display the original time series

plt.figure(figsize=(15, 4))

plt.plot(series, label='Original Time Series')

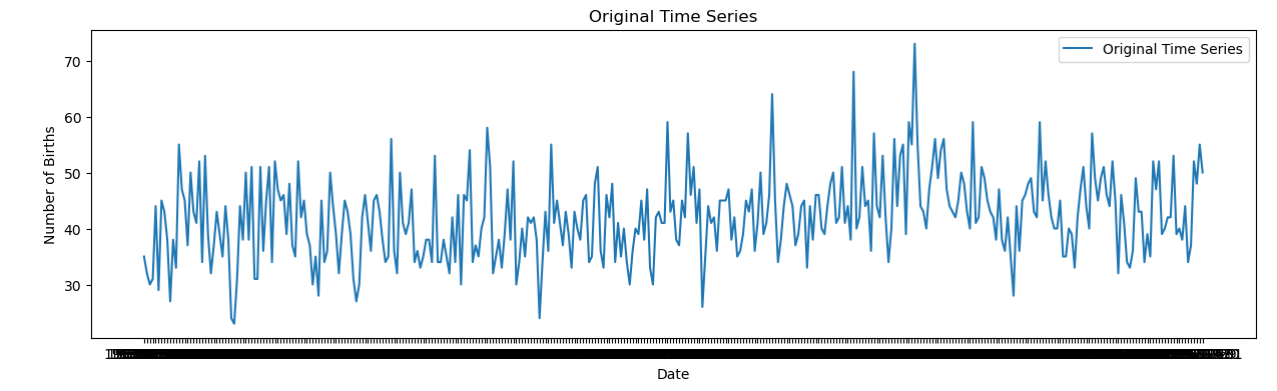
plt.title('Original Time Series')

plt.xlabel('Date')

plt.ylabel('Number of Births')

plt.legend()

plt.show()



We have plotted the original time series.

# Perform trailing moving average smoothing with a window size of 7

window\_size = 7

rolling = series.rolling(window=window\_size)

rolling\_mean = rolling.mean()

# Replace NaN values with zeros

rolling\_mean.fillna(0, inplace=True)

Here, we have applied a trailing moving average smoothing to the original time series data.

The window size is set to 7. We have replaced any NaN values with zeros.

# Display the smoothed time series

plt.figure(figsize=(15, 4))

plt.plot(series, label='Original Time Series')

plt.plot(rolling\_mean, label=f'Smoothed Time Series (Window Size {window\_size})', color='red')

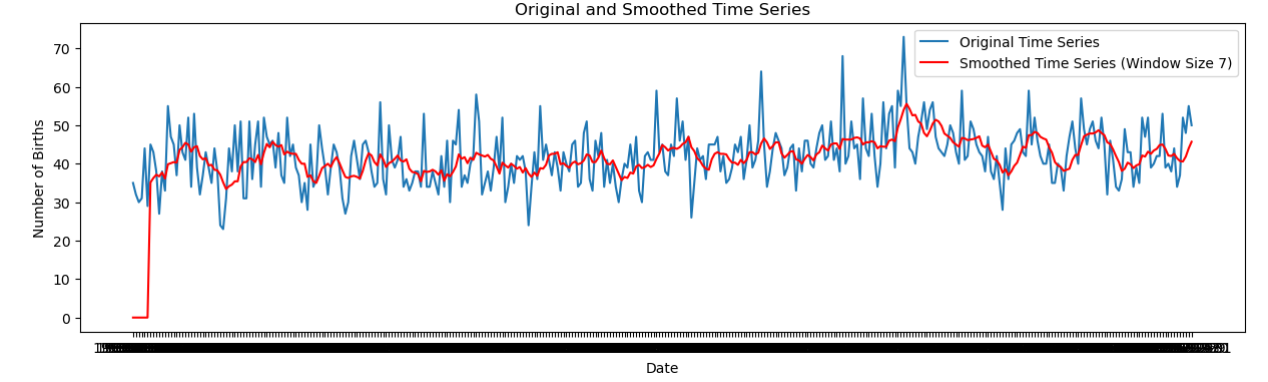
plt.title('Original and Smoothed Time Series')

plt.xlabel('Date')

plt.ylabel('Number of Births')

plt.legend()

plt.show()



Here, we have plotted both the original time series and the smoothed time series on the same plot.

# Forecasting using AR model with p=2

ar\_model = AutoReg(rolling\_mean, lags=2)

ar\_model\_fit = ar\_model.fit()

ar\_forecast = ar\_model\_fit.predict(start=len(rolling\_mean), end=len(rolling\_mean)+4)

# Forecasting using ARMA model with p=2 and q=2

arma\_model = ARIMA(rolling\_mean, order=(2, 0, 2))

arma\_model\_fit = arma\_model.fit()

arma\_forecast = arma\_model\_fit.predict(start=len(rolling\_mean), end=len(rolling\_mean)+4)

We have now performed time series forecasting using an autoregressive (AR) model and an autoregressive moving average (ARMA) model.

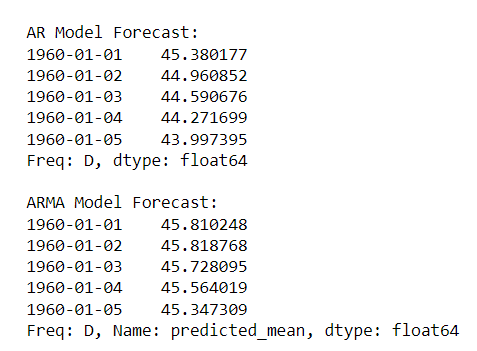
# Display the forecasting results

print("\nAR Model Forecast:")

print(ar\_forecast)

print("\nARMA Model Forecast:")

print(arma\_forecast)



Above are the forecasting results for both the AR model and ARMA model.

End Of Assignment!!