# COMP3340 Assignment 1 part 3

**Exercise 1:**

The Concrete Compressive Strength dataset of the UCI Machine Learning repository in <https://archive.ics.uci.edu/ml/datasets.php> is used for this exercise;

First 22 rows of dataset:



Since this dataset is large and contains over 1000 samples I randomized the data and then took a subset of 200 samples using the function ‘rowShufflerSubsetter()’ to use for the construction of the relative neighbourhood graph of the samples and features. Once that was complete 2 matrices were constructed using the same algorithm and dissimilarity metric previously developed for Assignment 1 part 2 which is the Euclidean distance since the concrete dataset is numerical data and the function used to construct the distance matrices is ‘concregteFeatSampDistMats()’ which takes the subsetted concrete data as an argument and returns the distance matrices between samples and features using the Euclidean distance.

Then the rng() function from the cccd package was used to construct the relative neighbourhood graphs of these distance matrices;

Graphical user interface, text, application, email

Description automatically generated

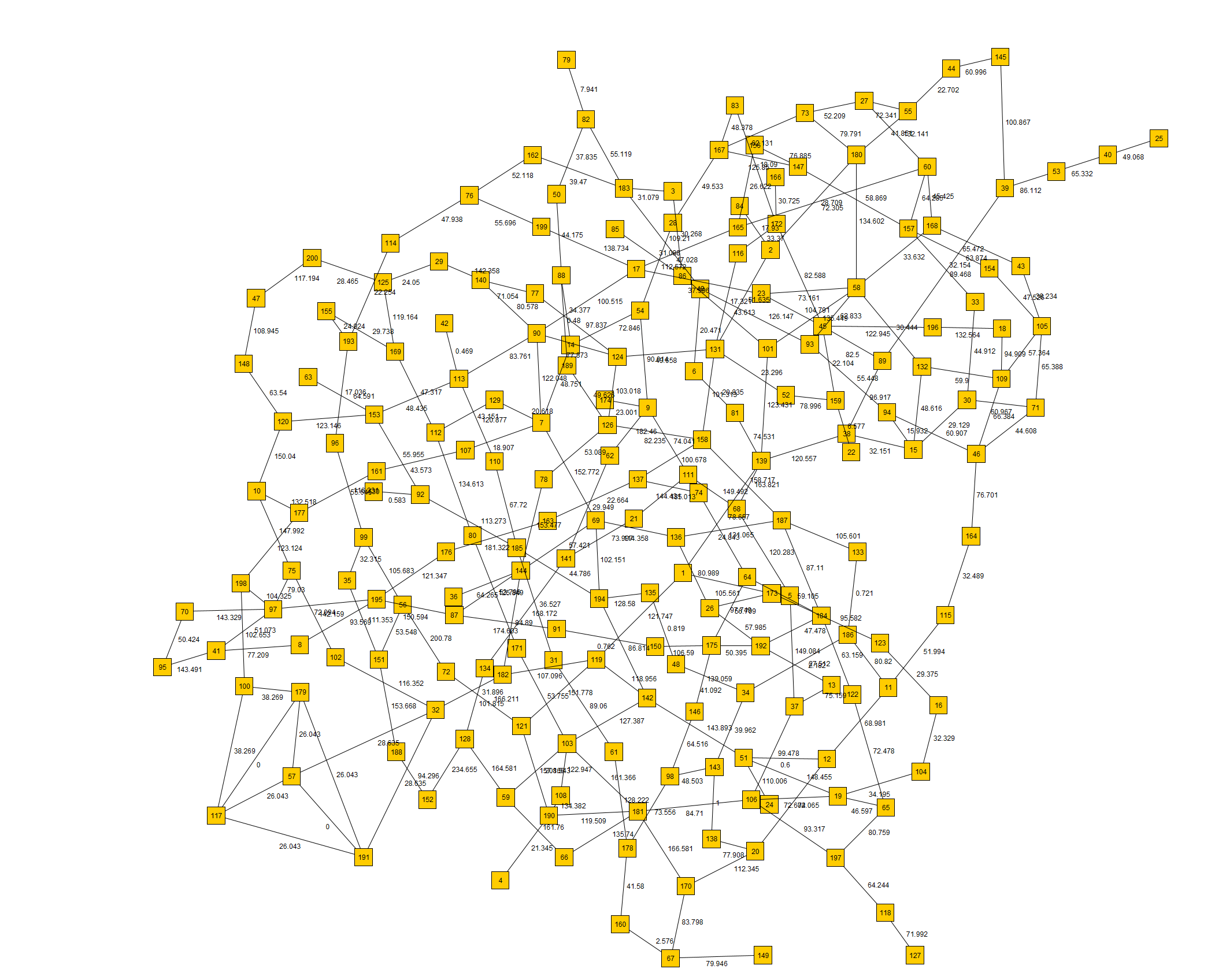
RNG of distance matrix between features of concrete dataset calculated using Euclidean distance:

**A picture containing chart

Description automatically generated**

RNG of distance matrix between samples of concrete dataset calculated using Euclidean distance:

(Subset of rows; 200 samples)



The function that returns these graphs is ‘RNGConcFeatSamps()’.

**Exercise 2:**

The US Presidency dataset was used for this exercise and the dataset was randomized and then split into 15 rows for the training set and 16 rows for the testing set and the each row number corresponding to the year which is the row identifier in the dataset is listed below for the training and test set; the function which completes this task is ‘rowShuffler()’:

Training set:

Graphical user interface, application, table, Excel

Description automatically generated

Testing Set:

Graphical user interface, application, table, Excel

Description automatically generated

Samples used for Training set

|  |
| --- |
| 1964:17 |
| 1888:5 |
| 1864:1 |
| 1920:25 |
| 1924:10 |
| 1880:4 |
| 1972:18 |
| 1980:31 |
| 1948:15 |
| 1904:7 |
| 1932:26 |
| 1976:30 |
| 1944:14 |
| 1960:28 |
| 1896:23 |

Samples used for Testing set:

|  |
| --- |
| 1872:3 |
| 1916:9 |
| 1912:24 |
| 1928:11 |
| 1884:21 |
| 1876:20 |
| 1940:13 |
| 1936:12 |
| 1868:2 |
| 1860:19 |
| 1956:16 |
| 1908:8 |
| 1968:29 |
| 1952:27 |
| 1900:6 |
| 1892:22 |

The Boruta feature selection method from the Boruta package in r was used as it was used in the previous assignment: This is the steps of the algorithm:

Boruta works as a wrapper algorithm around Random forest.

Step wise working of the Boruta Algorithm;

1) It adds random variation to the dataset by creating shuffled copies of all our features (Questions)

2) Secondly, a random forest classifier is trained on the extended dataset and a feature importance measure is applied, Default: Mean Decrease Accuracy to help identify the importance of our features (Questions) where higher means are important.

3) At each iteration, the algorithm checks whether a real feature has a greater importance than the best of its shuffled copies from step 1 and constantly removes features which are regarded as highly unimportant

4) Lastly, the Boruta algorithm stops either when all features are confirmed or rejected, or the specified limit of random forest runs has been reached.

Text, letter

Description automatically generated Text

Description automatically generated

The output of the Boruta feature selection on the US Presidency dataset is displayed below:

A picture containing text, electronics, scoreboard

Description automatically generated

As you can see displayed above the features deemed important by the Boruta feature selection method were Q4, Q5, Q10, Q12. So these were the features used in the classification process in weka. (these features with the subset of samples we’ve previously seen). The function which implements this method is ‘borutaFeatSel()’

The J48 pruned decision tree classifier was used in order to classify the training and test datasets as you can see below:

Training Set Results:

Table

Description automatically generated

Test Set Results:

Table

Description automatically generated

Table

Description automatically generated

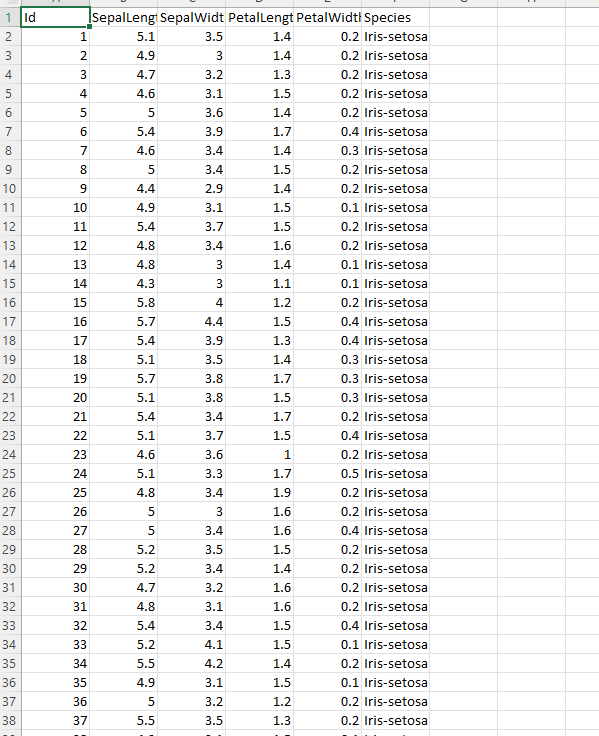
As we can see above the classifier performed very well on both training and testing sets by correctly classifying over 90% of instance in both training and test. We can see all our performance measures in the table above were all above 0.8 and hence performing very well. Although performance is strong there was not a significant increase in performance from the training to testing set this could be due to there being a very similar number of samples in each dataset. To potentially improve the testing set results significantly this could be repeated with a larger dataset where the split favours the training set in size so the classifier can learn from it more effectively and then apply what it learnt to the testing set.

**Exercise 3:**

**Exercise 4:**

**Exercise 5:**

The iris dataset was used for this task as it is an appropriate dataset which can be classified and produces strong clustering results:



The Euclidean distance was used as a distance measure as this was an appropriate measure to use for the iris dataset as it has numerical data. The Euclidean distance was used to calculate a distance matrix between the rows of the iris dataset after the id and species column were removed and this distance matrix was used as input into the function which creates the knn graph and the MST graph; ‘commonEdgesMstKnn()’ throughput the process the functions mst() from the igraph package in r and the nng() function from the cccd package in r were used to calculate the minimum spanning tree and the knn graph;

Graphical user interface, text, application, email

Description automatically generated

**Graphical user interface, text, application, email

Description automatically generatednng():Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

**a)**

The MST Graph of the iris dataset using the Euclidean distance between samples:

INSERT: IrisSamplesMST.pdf

**b)**

The Knn Graph of the iris dataset using the Euclidean distance between samples and k=2:

Chart, scatter chart

Description automatically generated

**c)**

The process that was undertook in order to identify the edges of the MST that are also a part of the KNN graph was first constructing the mst and knn graphs and then taking both of these graphs as variables and using these variables as arguments in the intersection() function from the igraph package in r:



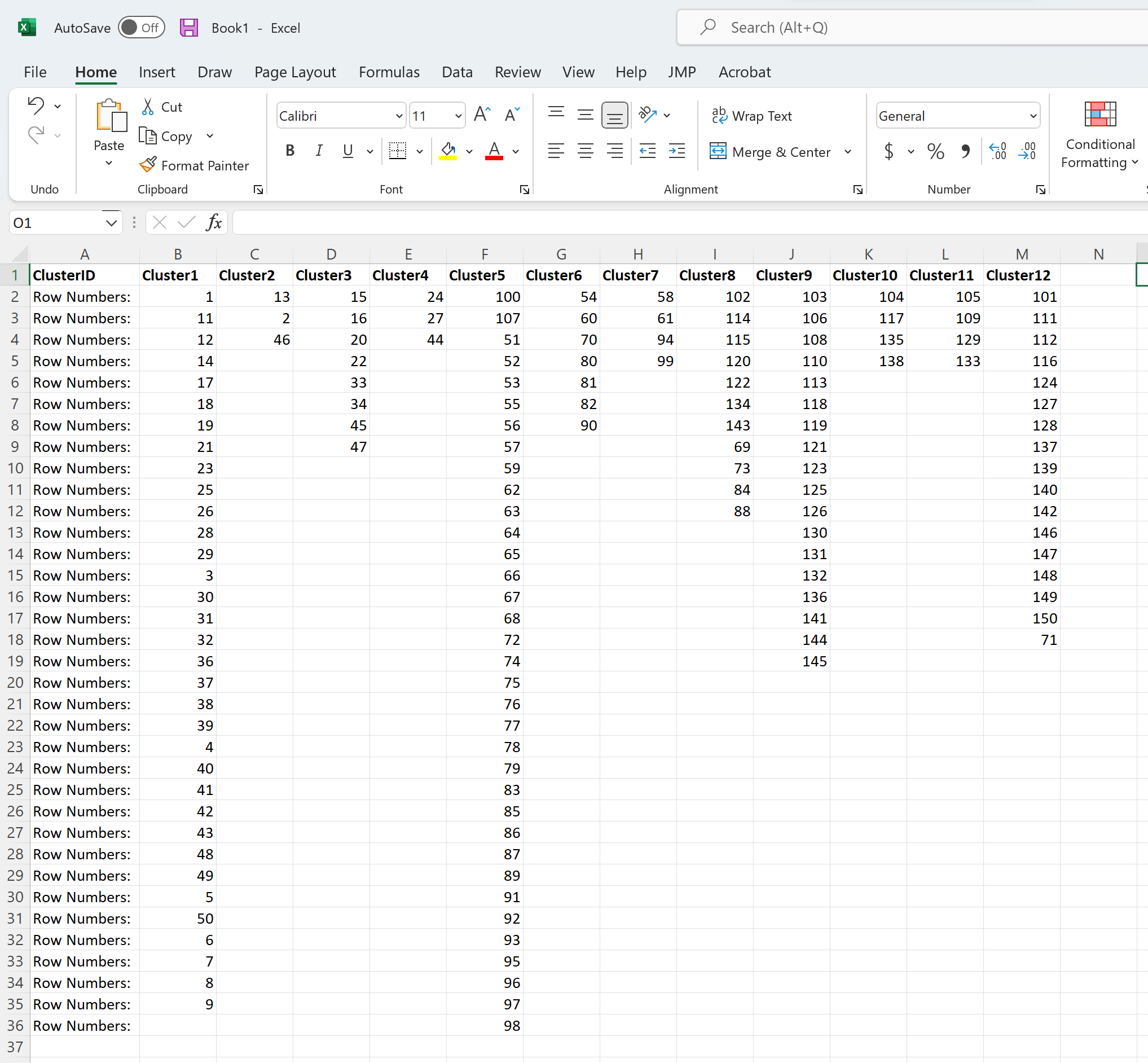
**Graphical user interface, text, application, email

Description automatically generatedText

Description automatically generated**

This function returns the edges that are present in both graphs and once we have those edges and label the vertices and edge weights appropriately the graph is then constructed which is the MST-KNN graph.

**d)**

Below is a table displaying the 12 clusters and the rows numbers (samples) of the iris dataset distance matrix calculated earlier which correspond to each cluster from the mst-knn common edges graph produced for this exercise; 

The function used to generate these results is within the source code is ‘commonEdgesMstKnn()’

**Exercise 6:**

The kmeans() function from the stats package in r was used on the iris datasets distance matrix produced in question 5 in order to produce clustering results. There were 3 clusters produced using this method as that appeared optimal based on the Within sum of squares plot produced from the wssPlot() in r :

**Text

Description automatically generated with medium confidence**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

The kmneans plot produced far less clusters than the mst-knn but appeared to show strong results as you can see below; the output below is produced using the function ‘kmeansIris()’:

Chart, line chart

Description automatically generated

**a)**

An lnter – rater reliability method is a method that provides a way to measure the level of agreement between multiple raters or judges. It can be used to assess the reliability of answers or results produced by different items on any test which provides a series of outcomes. When a test has a low inter-rater reliability, this indicates that the items on this test are unclear and potentially unnecessary. Some examples of methods include ‘percent agreement’, Cohens kappa, scotts pi etc. These methods ultimately provide a percentage of items that the two judgers or raters agreed on.

**b)**

Percent agreement can be used to calculate the inter-rater reliability between the 2 clustering results as we can see which clusters in the mst-knn were also in the k-means results.

**c)**

**Exercise 7:**

**a)**

In Data Mining Lazy Classification is a classification method where the process of training datasets is delayed until the test instance is received for the class prediction. When the testing data has appeared, the classification is implemented based on the data which is most related in the stored training data. Class imbalance can introduce many problems in machine learning, particularly classification problems. Imbalanced data can decrease a classification models accuracy. Useful information about the data could be ignored which could be crucial in building rule-based classifiers like Random Forests or decision trees. Another problem of class imbalance may occur when implementing random under sampling the sample chosen could be biased and hence produce an inaccurate representation of the population.

**b)**

Confusion matrix for US Presidency subset (training set used in question 2)

Table

Description automatically generated

Graphical user interface, application

Description automatically generated

The Matthews correlation coefficient in this case was 0.8729

Confusion matrix for Alzheimers disease training set which produced good results for sensitivity, specificity and accuracy: Table

Description automatically generated

Chart

Description automatically generated with medium confidence

The Matthews correlation coefficient in this case is 0.803

Confusion matrix for Alzheimers disease test set which produced good results for sensitivity, specificity and accuracy:

Table

Description automatically generated

Text

Description automatically generated

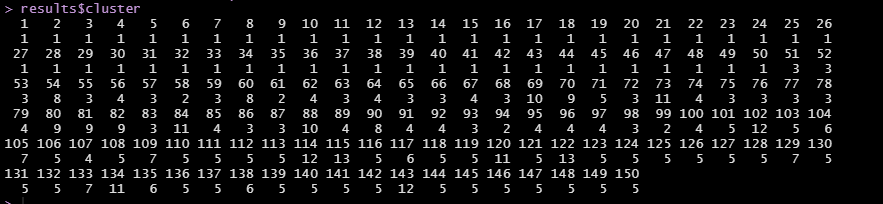
The Matthews correlation coefficient in this case is 0.785.

**Exercise 8:**

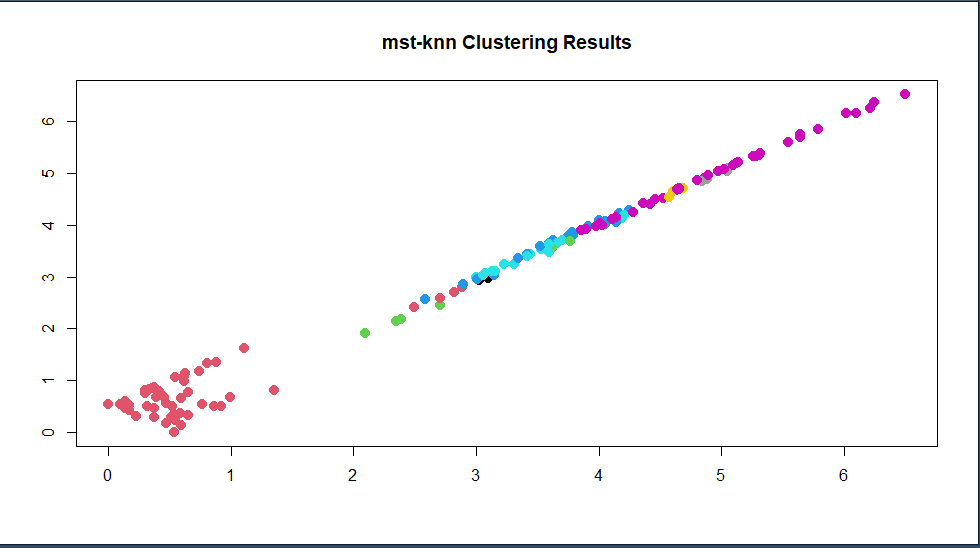
The Iris dataset was used for this exercise and it was firstly used to construct a distance matrix of dimensions 150 by 150 based on the Euclidean distance between the samples of the dataset.

MST-KNN Method:

The MST-knn clustering method produced 13 clusters with the first 50 rows taking up the first cluster and the rest of the clusters are more generously spread as you can see from the output below:



This is better demonstrated from the plot below where you can see the different clusters represent different clusters which consist of the rows or samples of the iris dataset;



K-Means Method:

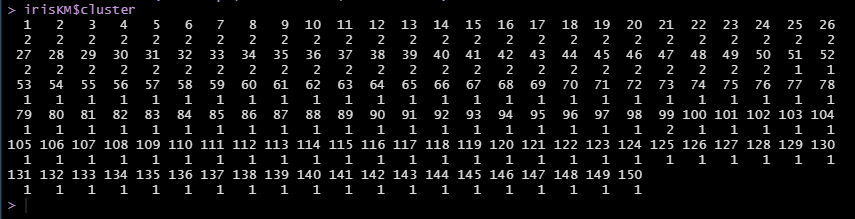
The k-means method firstly utilised the within sum of squares plot which you can see below;

Chart

Description automatically generated

From the wss plot above we can see that the elbow or ‘kick’ occurs when the number of clusters is equal to 2 so that will be the centres argument into our k-means algorithm so 2 clusters will be produced.

The cluster distribution can be displayed below:



We can see that cluster 1 and cluster 2 are both large clusters but 1 has a larger set of rows within it. We can see further analyse of this result in the below plot:

Chart, scatter chart

Description automatically generated

Hierarchical Method:

Hierarchical clustering different to k-means clustering in the sense that we do not give our algorithm a number of clusters as an argument so we don’t know how many clusters we will end up with much like the mst-knn method. We end up with a tree like visual representation of the observations (samples of iris dataset) called a dendrogram. The dendrogram helps us visualize the clustering’s obtained for each possible number of clusters from 1 to n.

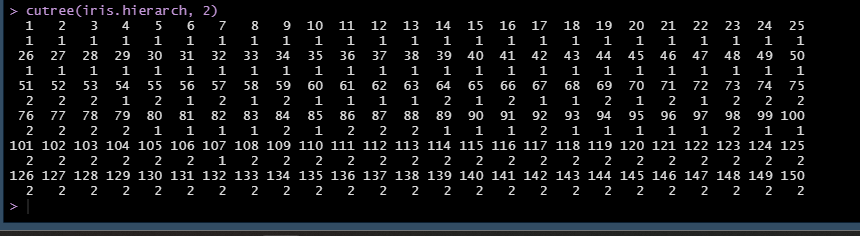
Diagram

Description automatically generated

Observations that fuse at the vary bottom of the tree are considered quite similar to each other whereas the observations that fuse at the top of the dendrogram will be quite different to each other.

Complete linkage was used for this method and this ensures maximal intercluster dissimilarity and it computes all pairwise dissimilarities between the observations in cluster 1 and the observations in cluster 2, and records the largest of the dissimilarities.

Below the cluster labels for each of the samples associated with a given cut of the dendrogram from the cutree() function

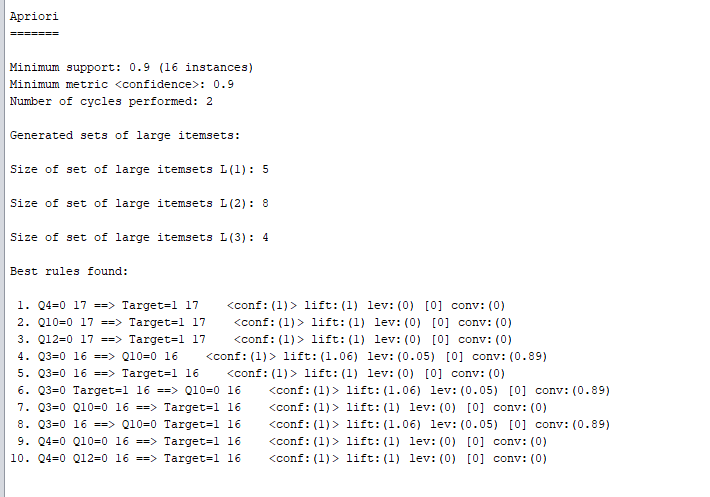


From the 3 different unsupervised clustering algorithms we can see that all the results vary for each even when applied on the same dataset. The Hierarchical clusteroing results were more similar to the k-means algorithm results with a smaller amount of cluster produced by the algorithm but the MST-Knn was the outlier in comparison to these two as it contained 13 clusters which was far more than our other 2 methods.

**Exercise 9:**

**a)**

Apriori results for US incumbent (target = 1):



For the Incumbent data the maximum size of the itemset found by weka was 8 for L(2) as you can see above and L(1) had 5 and L(3) had 4.

Confidence & lift for top 3 rules of US Incumbent set:

1. Confidence: 1 Lift: 1
2. Confidence: 1 Lift: 1
3. Confidence: 1 Lift: 1

Apriori results for US Challenger (target=0):

Text

Description automatically generated

For the Challenger data the maximum size of the itemset found by weka was 15 for L(2) as you can see above and L(1) had 10 and L(3) had 6.

Confidence & lift for top 3 rules of US Challenger set:

1. Confidence: 1 Lift: 1
2. Confidence: 1 Lift: 1
3. Confidence: 1 Lift: 1

**b)**

The steps of Apriori Algorithm:

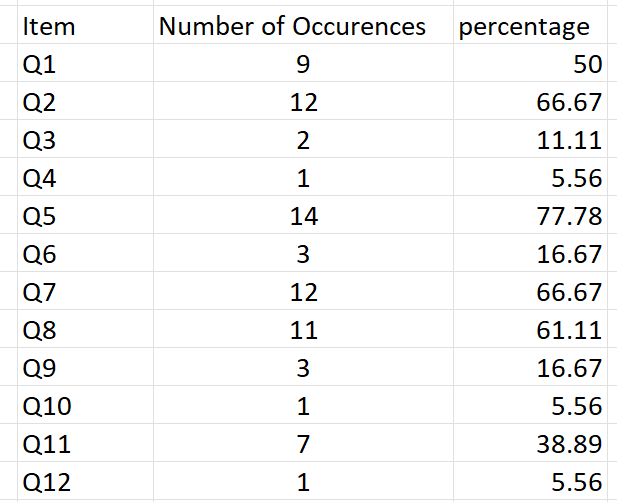
1. Computing the support for each individual item
2. Deciding on the support threshold
3. Selecting the frequent items
4. Finding the support of the frequent items
5. Repeat for larger sets
6. Generate association rules and compute confidence
7. Compute lift

(Our question has only asked for the step by step process of finding the association rules so they are displayed below)

**Apriori Algorithm and steps for US Incumbent:**

Step 1:

Computing the support for each individual item: This can be implemented by counting how many times each question (item) was present in a transaction (year): (Support percentages are calculated by taking the number of occurrences an item has in transactions and dividing by the total number of transactions (18) and multiplying by 100)



Step 2:

Deciding on the support threshold: Now that we have support values and percentages for all items we decide to use the support value of 60%

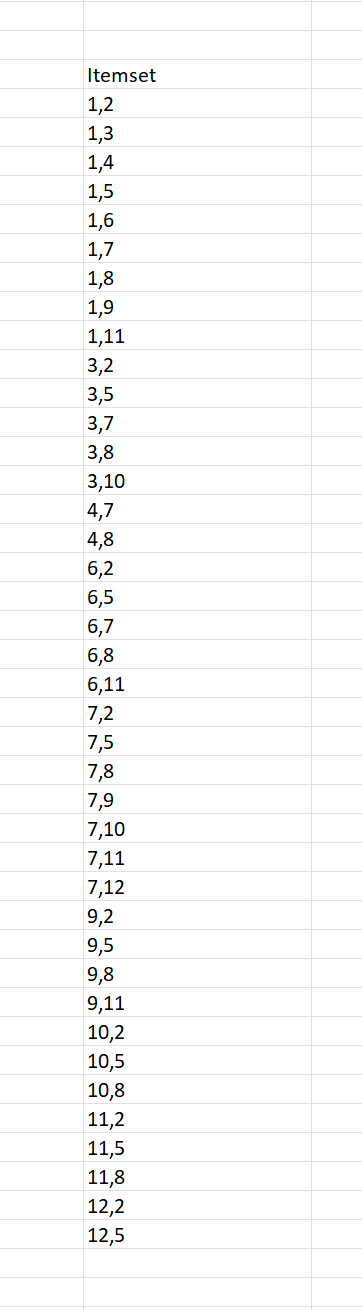
Step 3: we will now filter out any items that are less than 60% so we are left with selecting the items: {Q2, Q5, Q7, Q8} and all other items are less than 60% and removed

Step 4:

Finding the support of frequent itemsets,

Now we can do the same analysis but using pairs of items instead of individual items whilst ignoring all pairs that contain any of the non-frequent items (Questions).:

List of pairs which contains {Q1, Q3, Q4, Q6, Q9, Q10, Q11, Q12}:



From above 1 refers to Q1 etc.

And the combinations that do need to be included are:

Table

Description automatically generated

Step 5:

As you can see above none of the combinations are above the minimum support level of 60% or 10.8

So we will just take the individual items from step 1 as our frequent sets and they are the individual items: {Q2, Q5, Q7, Q8}

Step6:

Generate association rules and compute confidence

Confidence = the probability of the items occurring together/ the probability of the occurrence of the antecedant

Q2 = 1 => target = 1

Q5=1 => target = 1

Q7 = 1 => target =1

Q8 = 1 => target = 1

The same process is applied for the challenger file except the rules will be identifying the target as 0 for absence of a certain item or question.

**Exercise 12:**

**a)**

Cross Validation is a technique that uses multiple statistical data validation techniques to validate prediction results of a data model. Cross Validation can be utilised when assessing the ways in which results produced from statistical analysis generalize to a set of independent data. It is used for evaluating machine learning models by training several models on reduced sets of the available input data and evaluating these based on the reduced data.

**b)**

Bootstrapping is a data resampling method that helps to create several samples of datasets by performing sampling several times from existing data. It can help estimate the standard error and the construction of the confidence interval for datasets and implementing the hypothesis testing for the different samples a large dataset.

The method ultimately takes samples from an existing dataset repeatedly with replacements to create new datasets for sampling and these sample datasets are used to produce confident statistics.

**c)**

Imputation is a method which takes missing data or values in a dataset and replaces tghenm with substituted values. When a substitution for a single data point is made this is called a unit imputation and if the substitution is for a component of the data then it is called the item imputation. Imputation will consider all cases for replacement of missing data in a dataset by the substituted values that are collected based on available information about the given dataset