## Task 1: Supervised and Unsupervised Learning

### 1.1. Classification

In this task, our objective is to develop 3 different classification models that predict if the patient deceased during the follow-up period and then compare and discuss the results. The 3 classification algorithms I use are Naive Bayes, k-nearest neighbor (kNN) and support vector machine (SVM).

#### 1.1.1. Dataset Analysis

I have a labeled multivariate dataset that contains the medical records of 299 patients who had heart failure, collected during their follow-up period. There are 13 numeric features including anaemia, diabetes, high blood pressure, smoking, sex, age, platelets, serum creatinine, creatinine phosphokinase (CPK), ejection fraction, serum sodium, time, and our target, death event.

Through the Statistic node I know that there are a total of 299 observations, with no missing values found in any cells. The pie chart in figure 1.1a tells us that death event, our target class, has an imbalanced class distribution, in which 67.89% are False (0) and 32.11% are True (1).

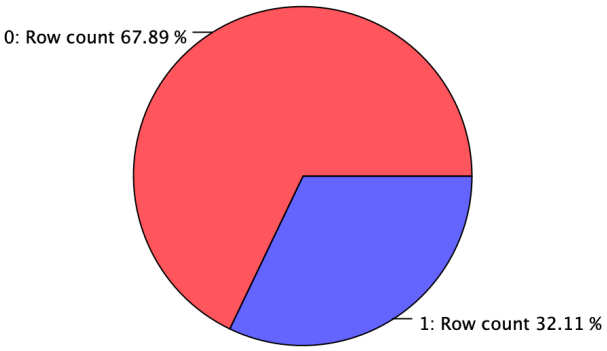


Figure 1.1a. Pie Chart of death event distribution

Next, in figure 1.1b I have a boxplot of age distribution of those 299 patients. The median age is 60, with a maximum age of 95 and a minimum age of 40. The interquartile range is 19 years (Q3 - Q1 = 70 - 51). I can see that the distribution of age is nearly symmetric by where the median stands.

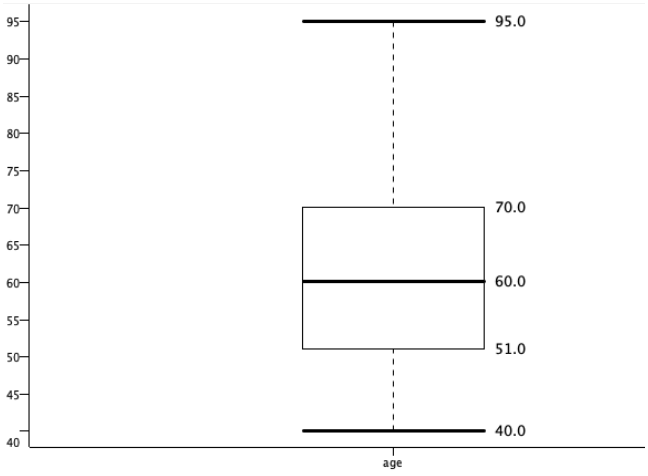


Figure 1.1b. A boxplot showing the age of the patients

#### 1.1.2. Preprocessing

First, death event appears to be a numeric type data and thus I will transform it into string type using the Number To String node to make it a possible target class.

It is important to check if our dataset contains any outliers since they (if any) may make our models, particularly those that are sensitive to outliers, misleading. Figure 1.1c shows box plots of all attributes. I use Gaussian normalization on our dataset to make them more readable. As I can see, heavy amount of extreme outliers (beyond the outer fence) appear to exist in CPK, platelets, and serum

creatinine. Since I cannot guarantee that our dataset is errorless and was collected without any sampling problems. I decide to remove these 3 variables from the dataset.

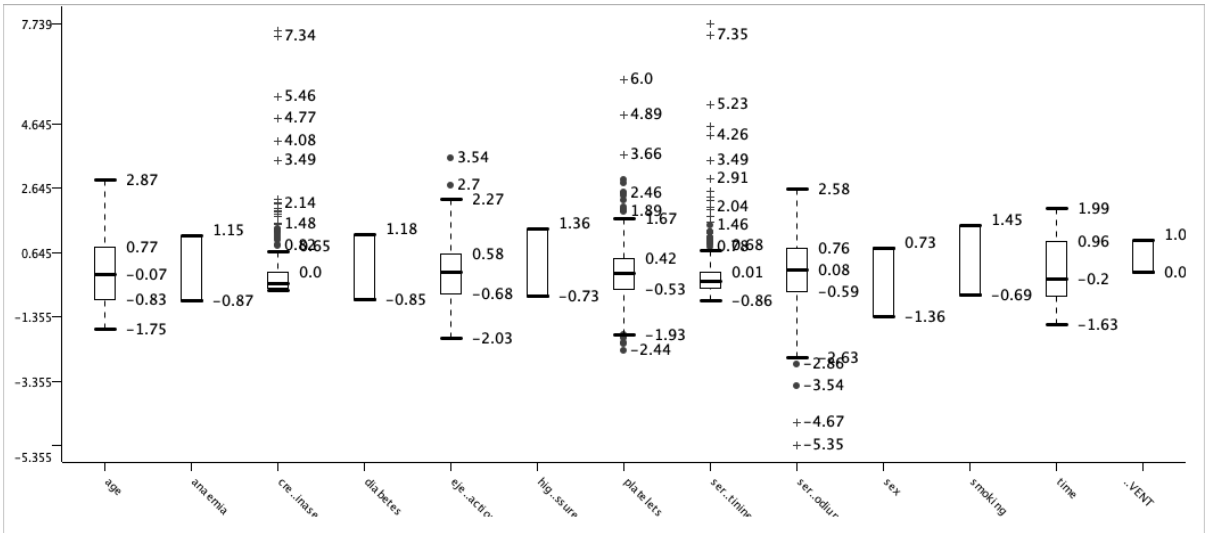


Figure 1.1c. Box plots of all attributes after Gaussian normalization

I want to avoid multicollinearity in our Naive Bayes classification model (it assumes that all attributes are independent from others). I use VIF node (threshold=5) to help us diagnose the extent of it. In table 1.1d, I can see that the extent of multicollinearity for each attribute is very low, thus I can safely assume that it does not affect our model at all.

**VIF Values**

The table shows Variance Inflation Factors (VIF) across all numeric variables.

| | |
|---|---|
| age | 1.11 |
| anaemia | 1.09 |
| creatinine_phosphokinase | 1.07 |
| diabetes | 1.06 |
| ejection_fraction | 1.07 |
| high_blood_pressure | 1.07 |
| platelets | 1.05 |
| serum_creatinine | 1.08 |
| serum_sodium | 1.10 |
| sex | 1.34 |
| smoking | 1.29 |
| time | 1.14 |

Table 1.1d. VIF values of each attributes

At the end of the day I want to find out those predictors that have a cause-effect relationship with our target class death event. Based on this objective, I can safely exclude those attributes with weak correlations (0.1> corr >-0.1) to death event shown in figure 1.1e (anaemia, CPK, diabetes, high

blood pressure, platelets, sex, and smoking) by the Column Filter node.

I also need to run a feature scaling for both k-NN and SVM models (algorithms that take consideration of the distance between data points). This is to avoid dominance of scaling factors and ensure every feature is on the same scale in their contributions to the results. After trying different normalizations through the Normalizer node, I decide to run a Gaussian normalization (higher accuracy) to help us with this.

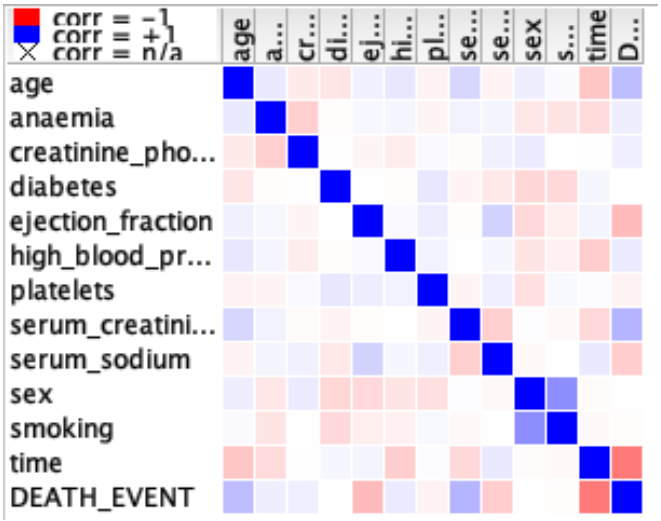


Figure 1.1e. A correlation matrix showing the strength and direction of linear relationships between attributes

### 1.1.3. Parameter Tuning

Because of the small size of dataset, I will apply 10-fold cross-validation by random sampling on all models separately so that I can use all our samples both for training and testing to get more representative results.

For Naive Bayes, after trying different values (20, 100, 150, 200, 250, 400, 500) of the maximum number of unique nominal values per attribute, setting it at 200 yields the highest accuracy. I will use it while keep other settings default.

For k-NN, among 70/30, 80/20 and 90/10 partitioning, 90/10 yields the highest accuracy. After trying different values (3 to 10) of number of neighbors to consider (k), I found that setting it at 4 yields the highest accuracy. Therefore I decide to go for 90/10 partitioning and k=4 while keep other settings default.

For SVM, among 70/30, 80/20 and 90/10 partitioning, 90/10 yields the highest accuracy. After trying different kernels, using the hyper-tangent kernel yields the highest accuracy several times, but the polynomial kernel (considers both the given features of input samples and combinations of these attributes to determine their similarity) gives us a close and more consistent accuracy. Therefore I decide to go for 90/10 partitioning and polynomial kernel while keep other settings default.

### 1.1.4. Results

Table 1.1f — For 90% training and 10% testing:

| Algorithms | TP | FP | TN | FN | Recall | Precision | Sensitivity | Specificity | AUC | F-measure | Accuracy (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Naive Bayes | 42 | 7 | 173 | 48 | 0.467 | **0.857** | 0.467 | **0.961** | **0.88** | 0.604 | 79.63 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| k-NN | 28 | 8 | 176 | 58 | 0.326 | 0.778 | 0.326 | 0.957 | 0.761 | 0.459 | 75.556 |
| SVM | 51 | 18 | 166 | 35 | **0.593** | 0.739 | **0.593** | 0.902 | 0.831 | **0.658** | **80.37** |

Table 1.1g — For 10-fold cross-validation:

| Algorithms | TP | FP | TN | FN | Recall | Precision | Sensitivity | Specificity | AUC | F-measure | Accuracy (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Naive Bayes | 64 | 19 | 184 | 32 | 0.667 | 0.771 | 0.667 | 0.906 | **0.857** | 0.715 | 82.943 |
| k-NN | 51 | 14 | 189 | 45 | 0.531 | 0.785 | 0.531 | **0.931** | 0.809 | 0.634 | 80.268 |
| SVM | 66 | 17 | 186 | 30 | **0.688** | **0.795** | **0.688** | 0.916 | 0.851 | **0.737** | **84.281** |

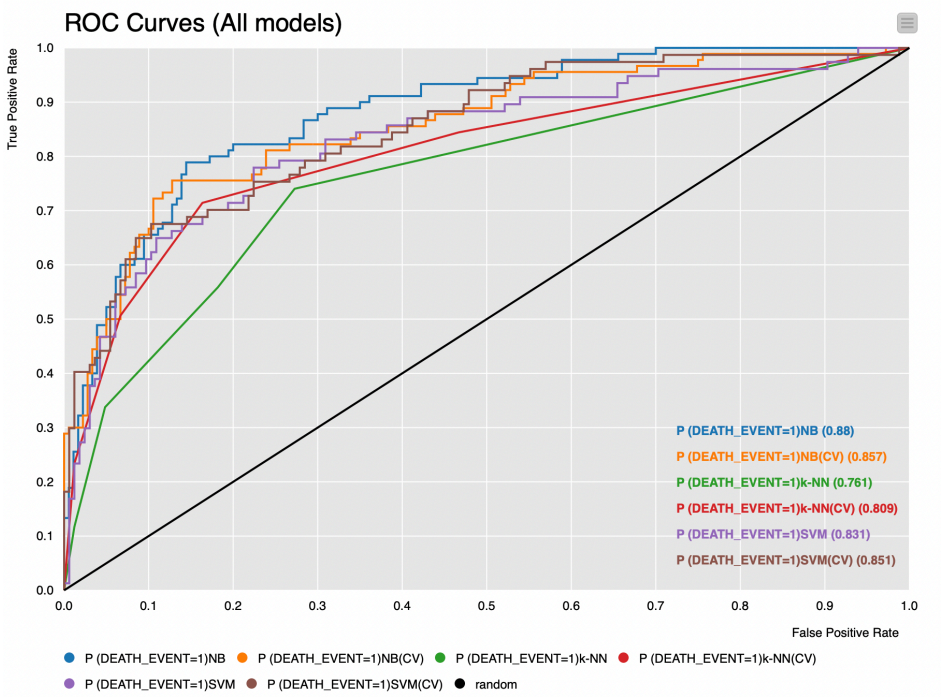


Figure 1.1h. ROC Curves of all models with AUC (bottom right)

From the tables above I can see that without 10-fold cross-validation k-NN performs the worst in most metrics. SVM has better recall/sensitivity (correctly identifying 59.3% of patients who actually deceased) than both Naive Bayes and k-NN do. Naive Bayes has the highest precision (when it predicts that a patient deceased during the follow-up period, it is correct 85.7% of the time), the highest specificity (correctly identifying 96.1% of patients who actually did not decease), and the highest AUC. SVM has the highest overall accuracy.

Unsurprisingly the models with 10-fold cross-validation give higher and closer numbers in most metrics. SVM is the leader in all metrics except for specificity and AUC. This small dataset slightly

favors SVM more than Bayes. All but two of the metrics has leader changes: SVM replaces Naive Bayes as the leader in precision (although all the models have very close numbers). K-NN takes over Naive Bayes in specificity. I will have our conclusion based on the results of 10-fold cross-validation.

### 1.1.5. Conclusion

Accuracy alone is not enough for us to make a decision on which model to use. Based on our objective, I aim to correctly identify as many patients who deceased during the follow-up period as possible, or in other words, to avoid incorrectly classifying a patient who deceased as “not deceased”. In this case, I prefer the one with the highest recall (SVM). However, if I also want to predict death events for new patients, then in addition to the last aim, I may also aim to avoid incorrectly classifying a patient who will not decease as "will decease” and give the patient unnecessary treatments. In that case, I will go for the one with the highest F-measure (SVM, again).

Although SVM seems to have better numbers, in such imbalanced data AUC is non-negligible. Fortunately SVM and Naive Bayes have nearly identical AUC (0.851 vs 0.857) so it does not create any selection problem for us at all (still prefer SVM). If I also consider the computational cost and time, Naive Bayes can be a strong contender because it is computational-wise less expensive and faster with a comparable performance (just 0.021 in Recall and 0.022 in F-measure behind SVM).

## 1.2. Regression

In this task, our objective is to use socio-demographic data, environmental factors, health status, and life habits to develop 2 different regression models that predict whether a semen sample is normal or altered and then compare and discuss the results. The 2 regression algorithms I use are linear regression and regression tree.

### 1.2.1. Dataset Analysis

I have a labeled multivariate dataset that contains the socio-demographic data, environmental factors, health status, and life habits of 100 volunteers who donated a semen sample in 2013. There are 9 numeric attributes including childish diseases, accident or serious trauma (accident), surgical intervention, season, age, frequency of alcohol consumption (alcohol), number of hours spent sitting per day (sitting hours), high fevers in the last year (fevers), smoking habit, and 1 string type data which is also our target, output: diagnosis. A range of normalisation has been applied to the input data.

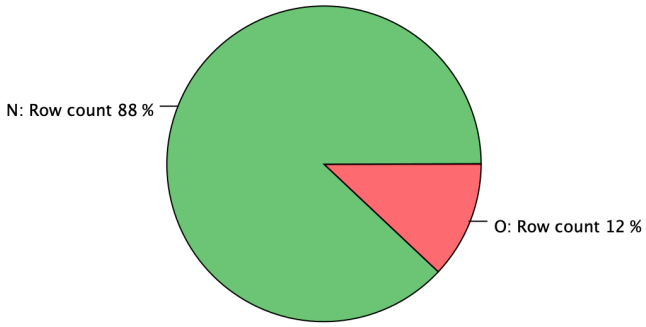


Figure 1.2a. Pie chart of output: diagnosis distribution

Through the Statistic node I know that there are a total of 100 observations, with no missing values found in any cells. The pie chart in figure 1.2a tells us that output: diagnosis, our target class, has a highly imbalanced class distribution, in which 88% are normal (N) and only 12% are altered (O).

Figure 1.2b tells us about the average of each attribute based on output: diagnosis. I can observe several conspicuous differences between N and O:

1. O is more likely recorded in hotter seasons (winter=-1, summer=0.33)
2. O has higher number in accident (yes=0, no=1)
3. O has closer fever records in time (>3m=-1, <3m and >12m=0, no=1)
4. O records higher frequency of alcohol consumption (several times a day=0, never=1)
5. O records higher level of smoking habit (never=-1, daily=1)

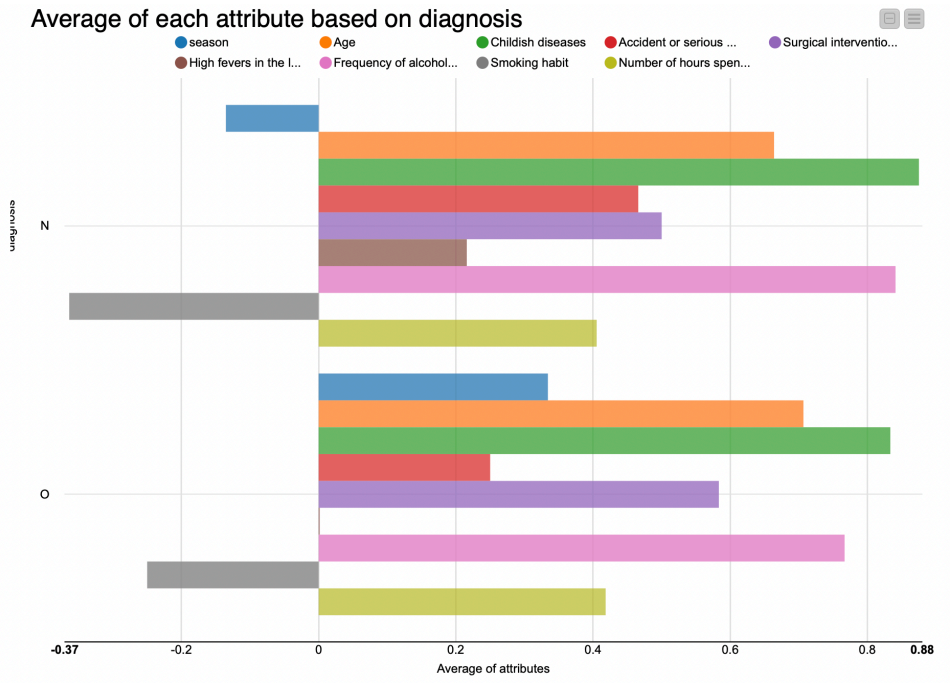


Figure 1.2b. Average of each attribute based on diagnosis

### 1.2.2. Preprocessing

Regression models are used to predict continuous values. Since our target output: diagnosis appears to be a string type data, I transform it into numeric type using the Category To Number node.

It is important to check if our dataset contains any outliers since they (if any) may make our models, particularly those that are sensitive to outliers, misleading. Figure 1.2c shows box plots of all attributes. I use Gaussian normalization on our dataset to make them more readable. As I can see, extreme outliers (beyond the outer fence) appear to exist in childish diseases and alcohol but I do not know the amount.
I use the Numeric Outliers node to help us with this (table 1.2d) and find that 13% of observations of Childish diseases are extreme outliers (k=3), which causes some serious concerns. Since I cannot guarantee that our dataset is errorless and was collected without any sampling problems, I decided to eliminate this attribute from our dataset.

| Row ID | Outlier column | Member count | Outlier count |
|---|---|---|---|
| Row0 | season | 100 | 0 |
| Row1 | Age | 100 | 0 |
| Row2 | Childish diseases | 100 | 13 |
| Row3 | Accident or serious trauma | 100 | 0 |
| Row4 | Surgical intervention | 100 | 0 |
| Row5 | High fevers in the last yea | 100 | 0 |
| Row6 | Frequency of alcohol consumption | 100 | 1 |
| Row7 | Smoking habit | 100 | 0 |
| Row8 | Number of hours spent sitting per day | 100 | 0 |

Table 1.2d. Outlier count in each attribute

ross all numeric variable

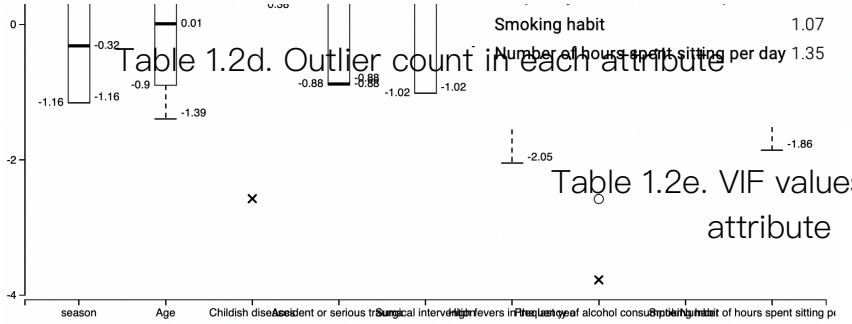


Table 1.2e. VIF values of each attribute

Figure 1.2c. Box plots of all attributes after Gaussian normalization

Since linear regression assumes that all features are independent from the others (Regression tree makes no such assumption), I do not want multicollinearity exist in our model. I use VIF node (threshold=5) to help us diagnose the extent of it. In table 1.2e, I can see that the VIF values for each attribute is low, thus I can safely assume that it does not affect our model at all.

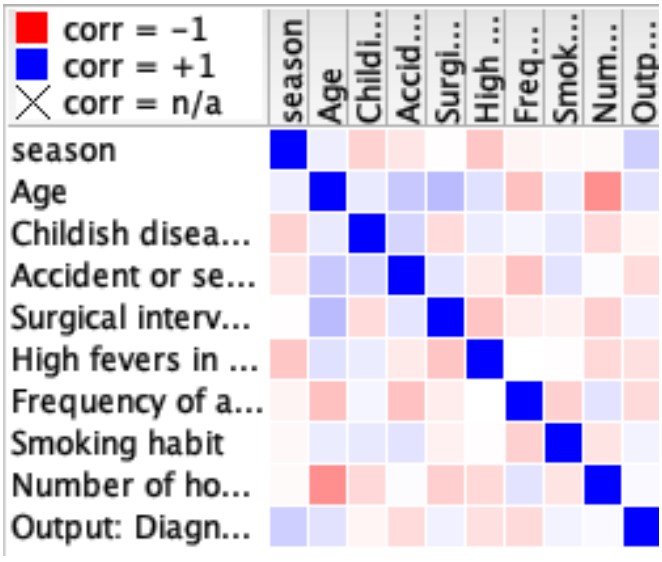


Figure 1.2f. A correlation matrix showing the strength and direction of linear relationships between attributes

In the end I want to find out those predictors that have a cause-effect relationship with our target class output: diagnosis. Based on this objective, I can safely exclude those attributes with weak correlations (0.1> corr >-0.1) to output: diagnosis shown in figure 1.2f (childish diseases, surgical intervention, smoking habit, and sitting hours) by the Column Filter node.

I also want to run a feature scaling for both algorithms and see how the results differ from those without normalisation. After trying different normalizations through the Normalizer node, I decide to run a Gaussian normalization (higher R^2) to help us with this.

### 1.2.3. Parameter Tuning

Because the dataset size is small, I want to use a 90/10 partitioning by random sampling, but then I find that the small testing set very often contains purely actual values of 0 that causes the

model return an R^2 of "-∞". Thus I change our mind into using a 70/30 partitioning instead.

For the hyper-parameters in the Linear Regression Learner node, those 6 filtered features become the input variables while output: diagnosis is our output. All other settings are kept unchanged.

For the hyper-parameters in the Simple Regression Tree Learner node, I try different combinations of minimum split node size and minimum node size (e.g. 4:2, 10:5, 20:10) and all of them give volatile scores in multiple trials. Thus I decide to keep everything as default.

### 1.2.4. Results

Table 1.2g:

| Algorithms | Normalized | R^2 | MAE | RMSE |
|---|---|---|---|---|
| Linear Regression | No | 0.014 | 0.247 | **0.397** |
| | Yes | **0.046** | 0.635 | 1.017 |
| Regression Tree | No | -0.44 | **0.2** | 0.447 |
| | Yes | -0.226 | 0.459 | 1.152 |

In the table above, I compare both algorithms in terms of R-squared, mean absolute error (MAE) and root mean squared error (RMSE). As I can see, They both have low numbers in R-squared regardless of applying normalisation. Regression Tree even has negative numbers (meaning the fit of the model is worse than that of a horizontal line). Regression Tree has slightly better MAE (0.2 vs 0.247) than Linear Regression does, while Linear Regression does slightly better than Regression Tree in RMSE (0.397 vs 0.447). Note that normalising the models gives poorer results in both error measures. Since they have similar numbers in both error measures, I would prefer to

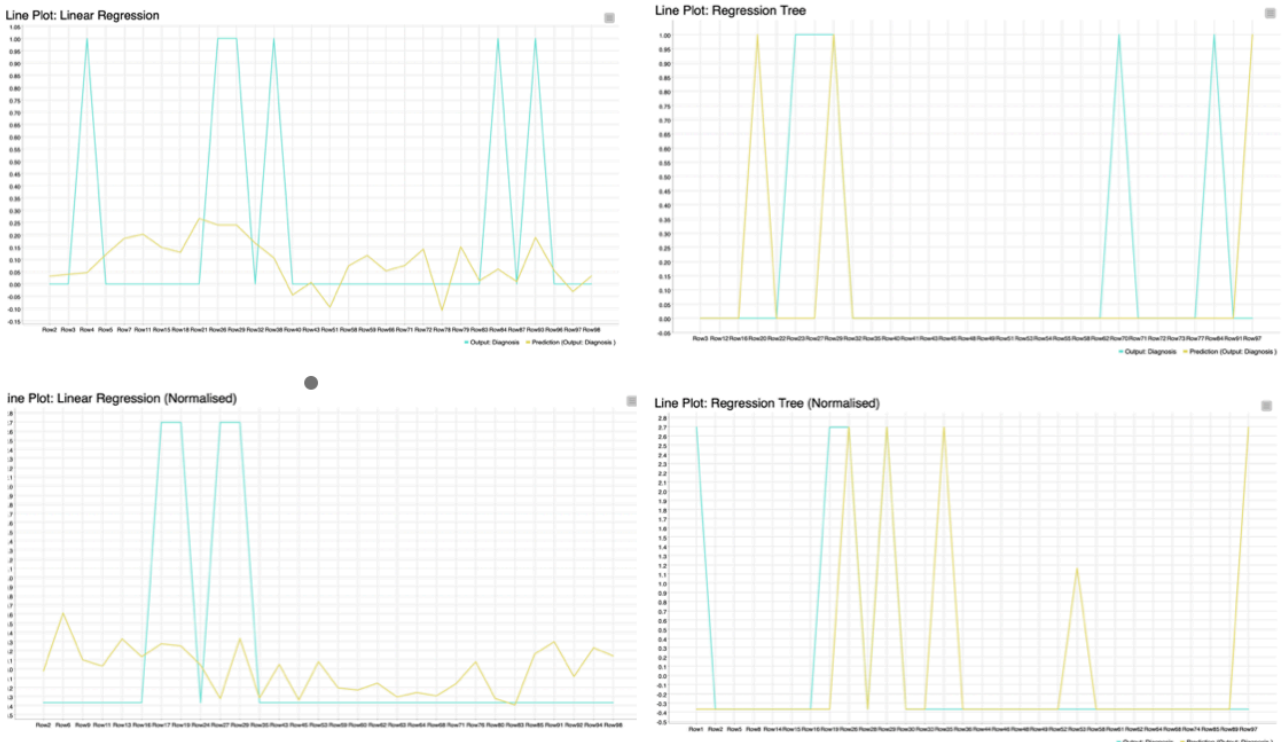


Figure 1.2h. Line graphs of all models

use Linear Regression in this case because of it's better R-squared. Line graphs of all models are shown in figure 1.2h. None of the fitted lines show reliable prediction of the actual values.
In table 1.2i and 1.2j I can see that none of the predictors are statistically significant in our Linear Regression models.

### 1.2.5. Conclusion

In this task, I use 2 regression methods and find that Linear Regression is the better performer between them. However, due to the limitation brought by the sample size and the imbalanced distribution of target class, I need to be cautious when I are analysing the results. Our findings imply that those predictors used do not have significant impact on our target, output: diagnosis. The scores suggest that both algorithms used may not be the ideal methods for this case.

## 1.3 Clustering

In this task, our objective is to develop 2 different clustering models that segment a dataset of absenteeism at work into groups and then compare and discuss the results. The 2 clustering algorithms I use are k-means and hierarchical.

### 1.3.1. Dataset Analysis

I have a multivariate dataset that contains the records of absenteeism at work at a courier company in Brazil from Jul 07 to Jul 10. There are 21 numeric attributes from absence information, environmental factors, demographic data, health status, and life habits including ID, reason for absence, months absence, day of the week, seasons, transportation expense, distance from residence to work (distance), service time, age, work load average/day (workload), hit target, disciplinary failure, education, son, social drinker, social smoker, pet, weight, height, body mass index, and absenteeism time in hours (absenteeism).

**Statistics on Linear Regression**

| Variable | Coeff. | Std. Err. | t-value | P>\|t\| |
|---|---|---|---|---|
| season | 0.0669 | 0.0441 | 1.5158 | 0.1345 |
| Age | 0.3269 | 0.3068 | 1.0655 | 0.2906 |
| Accident or serious trauma | -0.0735 | 0.0754 | -0.9751 | 0.3332 |
| High fevers in the last yea | -0.0628 | 0.059 | -1.0638 | 0.2914 |
| Frequency of alcohol consumption | -0.1605 | 0.2311 | -0.6945 | 0.4899 |
| Intercept | 0.0506 | 0.3164 | 0.1599 | 0.8735 |

Multiple R-Squared: 0.1009
Adjusted R-Squared: 0.0307

Table 1.2i. Statistics on Linear Regression

**Statistics on Linear Regression**

| Variable | Coeff. | Std. Err. | t-value | P>\|t\| |
|---|---|---|---|---|
| season | -0.003 | 0.1269 | -0.0233 | 0.9815 |
| Age | 0.1393 | 0.1216 | 1.1459 | 0.2561 |
| Accident or serious trauma | -0.2142 | 0.1185 | -1.8069 | 0.0755 |
| High fevers in the last yea | -0.2064 | 0.1282 | -1.6106 | 0.1122 |
| Frequency of alcohol consumption | -0.0696 | 0.1222 | -0.5695 | 0.571 |
| Intercept | -0.0147 | 0.1182 | -0.1246 | 0.9013 |

Multiple R-Squared: 0.094
Adjusted R-Squared: 0.0232

Table 1.2j. Statistics on Linear Regression (Normalised)

Through the Statistic node I know that there are a total of 740 observations from 36 distinct IDs, with no missing values found in any cells.

### 1.3.2. Preprocessing

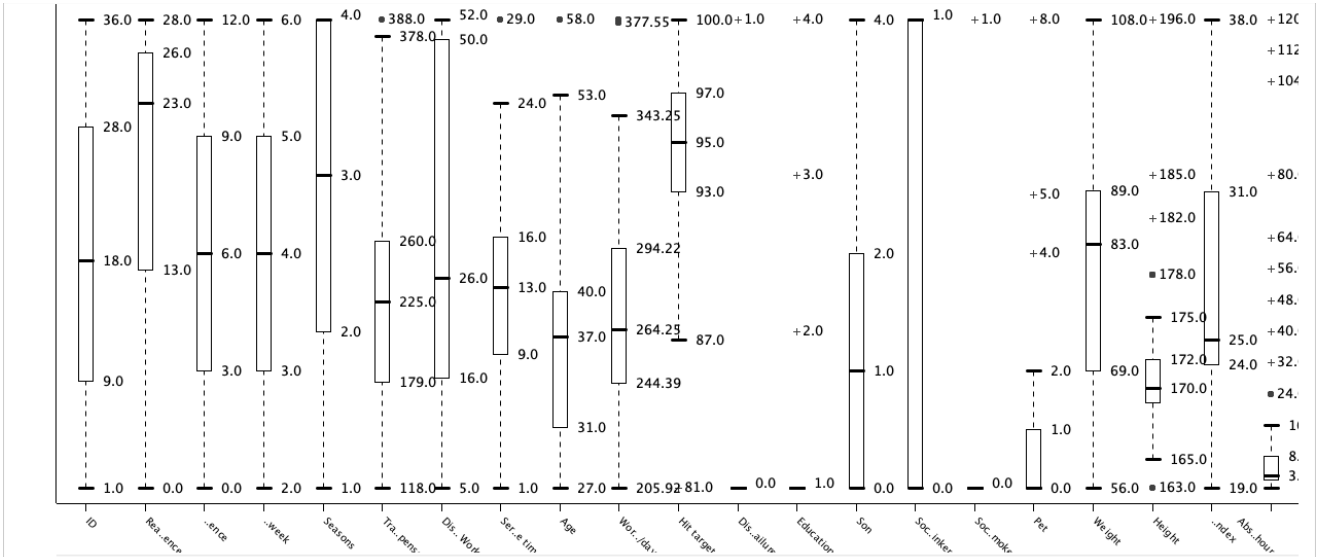

Figure 1.3a. Box plots of all attributes after normalization

It is important to check if our dataset contains any outliers since they (if any) may make our models, particularly those that are sensitive to outliers, misleading. Figure 1.3a shows box plots of all attributes. I use min-max normalization on our dataset to make them more readable. As I can see, extreme outliers (beyond the outer fence) appear to exist in several attributes like disciplinary failure, education, social smoker, pet, height, and absenteeism.

I use the Numeric Outliers node to exemplify our findings of extreme outliers (k=3) and I can see that in table 1.3b. Since I cannot guarantee that our dataset is errorless and was collected without any sampling problems, I decide to take these 6 attributes away.

| Row ID | Outlier column | Member count | Outlier count |
|---|---|---|---|
| Row0 | ID | 740 | 0 |
| Row1 | Reason for absence | 740 | 0 |
| Row2 | Month of absence | 740 | 0 |
| Row3 | Day of the week | 740 | 0 |
| Row4 | Seasons | 740 | 0 |
| Row5 | Transportation expense | 740 | 0 |
| Row6 | Distance from Residence to Work | 740 | 0 |
| Row7 | Service time | 740 | 0 |
| Row8 | Age | 740 | 0 |
| Row9 | Work load Average/day | 740 | 0 |
| Row10 | Hit target | 740 | 0 |
| Row11 | Disciplinary failure | 740 | 40 |
| Row12 | Education | 740 | 129 |
| Row13 | Son | 740 | 0 |
| Row14 | Social drinker | 740 | 0 |
| Row15 | Social smoker | 740 | 54 |
| Row16 | Pet | 740 | 14 |
| Row17 | Weight | 740 | 0 |
| Row18 | Height | 740 | 56 |
| Row19 | Body mass index | 740 | 0 |
| Row20 | Absenteeism time in hours | 740 | 28 |

Table 1.3b. Outlier count in each attribute

Multicollinearity can lead to distorted results in distance-based algorithms like k-means and hierarchical clustering by doubling weight. I use VIF node (threshold=5) to help us diagnose the extent of it in our dataset. In table 1.3c, I can see that the VIF values for weight, height, and body mass index are sky-high, thus I will eliminate them along with the attributes with extreme outliers.

**VIF Values**

The table shows Variance Inflation Factors (VIF) across all numeric variables.

| | |
|---|---|
| ID | 2.35 |
| Reason for absence | 1.68 |
| Month of absence | 1.83 |
| Day of the week | 1.11 |
| Seasons | 1.43 |
| Transportation expense | 1.88 |
| Distance from Residence to Work | 2.64 |
| Service time | 3.89 |
| Age | 2.91 |
| Work load Average/day | 1.26 |
| Hit target | 1.46 |
| Disciplinary failure | 1.65 |
| Education | 1.85 |
| Son | 1.40 |
| Social drinker | 2.78 |
| Social smoker | 1.33 |
| Pet | 1.87 |
| Weight | *185.55 (consider eliminating)* |
| Height | *33.38 (consider eliminating)* |
| Body mass index | *172.19 (consider eliminating)* |
| Absenteeism time in hours | 1.19 |

Table 1.3c. VIF values of each attribute

I also want to run a feature scaling for both algorithms and see how the results differ from those without it. After trying different normalizations through the Normalizer node, I decide to run a min-max normalization (higher mean silhouette coefficient (MSC)) to help us with this.

### 1.3.3. Parameter Tuning

For k-Means clustering, I have evaluated 70/30, 80/20, and 90/10 partitioning by random sampling and find that 90/10 leads to higher MSC for this small dataset.

For the hyper-parameters in the k-Means node, I evaluate the content of clusters in each k-value from 2 to 10 and find that the optimal k=2, so I settle with k=2. I then evaluate both centroid initializations and find that using first k rows consistently yields better MSC. Also, 99 and 999 iterations are the same here so will keep this setting unchanged.

For the hyper-parameters in the Distance Matrix Calculate node, I will go for Euclidian distance because I find that using it gives slightly better MSC than using Manhattan distance.

For the hyper-parameters in the Hierarchical Clustering (DisMatrix) node, I find that single linkage type yield the best MSC at the number of clusters = 2 (Hierarchical Cluster Assigner (local)), so I will use such settings.

### 1.3.4. Results

Table 1.3d:

| Algorithms | Normalized | MSC |
|---|---|---|
| k-Means | No | 0.408 |
| | Yes | 0.267 |
| Hierarchical | No | **0.452** |
| | Yes | 0.174 |

I compare both algorithms by their respective MSC. Hierarchical clustering has a larger MSC than k-Means does (0.452 vs 0.408). However, if I normalise them in advance, it will be k-Means clustering that stands out from the group (0.174 vs 0.267). It is obvious that normalisation does not do them any favour in MSC performance because the numbers of MSC become lower after it is applied.

The numbers of observations each cluster covers using both clustering methods are shown in table1.3e. I can see that the k-Means models divide clusters into similar sizes (~56% vs ~44% in relative frequency) while that of hierarchical models are far apart in size (~99% vs ~1% in relative frequency). In comparison, the cluster segmentation in the k-Means models looks more meaningful.

| Row ID | Cluster | Count (Cluster) | Relative Frequency (Cluster) |
|---|---|---|---|
| Row0 | cluster_0 | 374 | 0.562 |
| Row1 | cluster_1 | 292 | 0.438 |

k–Means clustering

| Row ID | Cluster number | Count (Cluster number) | Relative Frequency (Cluster number) |
|---|---|---|---|
| Row0 | 0 | 735 | 0.993 |
| Row1 | 1 | 5 | 0.007 |

Hierarchical clustering

| Row ID | Cluster | Count (Cluster) | Relative Frequency (Cluster) |
|---|---|---|---|
| Row0 | cluster_1 | 379 | 0.569 |
| Row1 | cluster_0 | 287 | 0.431 |

k–Means clustering (normalised)

| Row ID | Cluster number | Count (Cluster number) | Relative Frequency (Cluster number) |
|---|---|---|---|
| Row0 | 0 | 739 | 0.999 |
| Row1 | 1 | 1 | 0.001 |

Hierarchical clustering (normalised)

Table 1.3e. Cluster Coverage of all clustering models

From table 1.3f, I can observe some significant differences between the clusters in the k-Means model:

1. Mean transportation expense ((cluster 0) 268.82 vs (cluster 1) 160.11) — figure 1.3g
2. Mean service time ((cluster 0) 11.24 vs (cluster 1) 14.41)
3. Mean son ((cluster 0) 1.45 vs (cluster 1) 0.49)

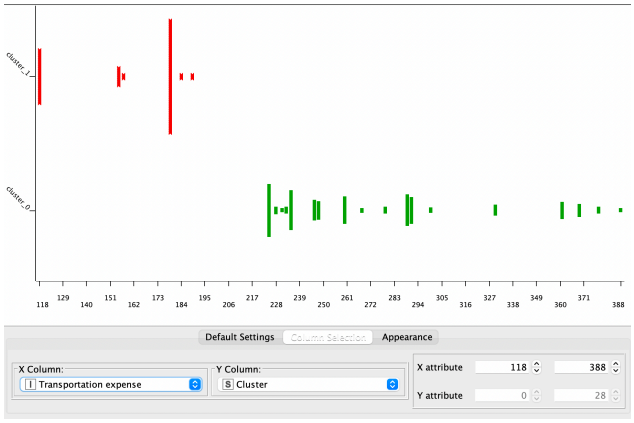


Figure 1.3g. Scatter plot of clusters based on transportation expense by k–Means clustering

- 2 Clusters
  - cluster_0 (coverage: 374)
    - ID = 18.20855614973262
    - Reason for absence = 18.093582887700535
    - Month of absence = 6.75668449197861
    - Day of the week = 3.9759358288770055
    - Seasons = 2.6149732620320854
    - Transportation expense = 268.8235294117647
    - Distance from Residence to Work = 29.32620320855615
    - Service time = 11.240641711229946
    - Age = 35.850267379679146
    - Work load Average/day = 271.9169839572193
    - Hit target = 94.34491978609626
    - Son = 1.4491978609625669
    - Social drinker = 0.5614973262032086
  - cluster_1 (coverage: 292)
    - ID = 17.301369863013697
    - Reason for absence = 20.44178082191781
    - Month of absence = 5.684931506849315
    - Day of the week = 3.893835616438356
    - Seasons = 2.469178082191781
    - Transportation expense = 160.1095890410959
    - Distance from Residence to Work = 30.006849315068493
    - Service time = 14.41095890410959
    - Age = 37.25342465753425
    - Work load Average/day = 271.3607054794528
    - Hit target = 94.96917808219177
    - Son = 0.4863013698630137
    - Social drinker = 0.589041095890411

Table 1.3f. cluster view of the k–Means model

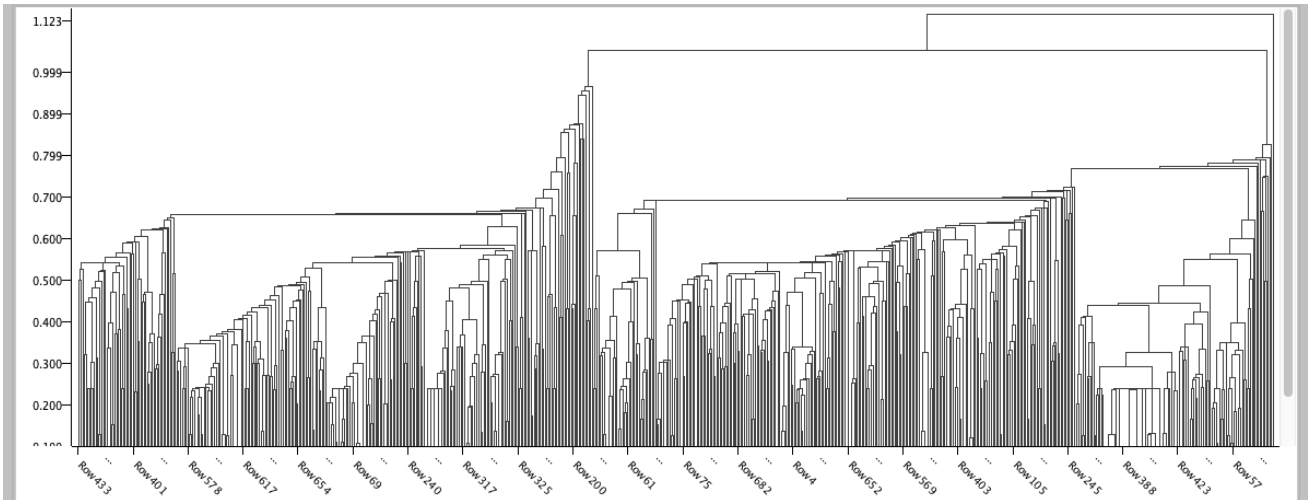


Figure 1.3h. Dendrogram of hierarchical clustering for Absenteeism dataset

### 1.3.5. Conclusion

In this task, I use both k-Means and hierarchical clustering method to treat our dataset. I find that the optimal number of clusters in both methods is 2. The sizes of clusters divided by k-Means are close while those divided by hierarchical method are far apart, which make the hierarchical models meaningless. I therefore focus on the clusters of k-Means models and each has its own characteristics. For example, in the k-Means model without prior normalisation cluster 0 has a larger mean transportation expense and more son and a lower service time, this can be explained by a causation that with more children one may need to commute more frequently to take care of them and that also results in a lower service time. On the other hand, cluster 1 has less transportation expense and son but a longer service time. This can be explained by a causation that with less children one can spend more service time and save more transportation expense.

## 1.4. Association Rules

In this task, our objective is to develop an association rules model that finds sets of items that are frequently bought together in a grocery store, also such model allows us to make reliable inference on the subsequent items to be bought with the items already in cart. The algorithm I use is apriori. I will discuss our findings at the end.

### 1.4.1. Dataset Analysis

I have a dataset that contains the records of 9835 transactions by customers shopping for groceries. There are 169 unique items such as whole milk and beef that appear in the rows that represent the transactions they are in.

### 1.4.2. Preprocessing

Since I are not interested on the number of items bought in each transaction, I use the Column Filter node to filter out the column item(s).

Then I use the Create Collection Column node to aggregate the values in each row and put the lists into a new column named AggregatedValues. As the original table contains many missing values, I also check the “ignore missing values” box.

### 1.4.3. Parameter Tuning

For the Association Rule Learner (Borgelt) node, I want to reduced the table size to only the most interesting association rules left for further analysis, and thus only association rules with a minimum support of 700 (absolute number) and a minimum rule confidence of 25% will be included in our table.

### 1.4.4. Results

Table "default" – Rows: 36 | Spec – Columns: 11 | Properties | Flow Variables

| Row ID | Consequent | Antecedent | ItemSetSupport | RelativeItemSetSupport% | RuleConfidence% | AbsoluteBodySetSup… | RelativeBodySetSupp… | RuleLift | RuleLift% | AbsoluteHeadItemSu… | RelativeHeadItemSup… |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Row8 | whole milk | [whipped/sour cream] | 317 | 3.223 | 45 | 705 | 7.17 | 1.76 | 175.98 | 2,513 | 25.552 |
| Row28 | whole milk | [root vegetables] | 481 | 4.891 | 44.9 | 1,072 | 10.9 | 1.756 | 175.6 | 2,513 | 25.552 |
| Row27 | other vegetables | [root vegetables] | 466 | 4.738 | 43.5 | 1,072 | 10.9 | 2.247 | 224.66 | 1,903 | 19.349 |
| Row7 | other vegetables | [whipped/sour cream] | 284 | 2.888 | 40.3 | 705 | 7.17 | 2.082 | 208.19 | 1,903 | 19.349 |
| Row25 | whole milk | [tropical fruit] | 416 | 4.23 | 40.3 | 1,032 | 10.5 | 1.578 | 157.76 | 2,513 | 25.552 |
| Row31 | whole milk | [yogurt] | 551 | 5.602 | 40.2 | 1,372 | 14 | 1.572 | 157.17 | 2,513 | 25.552 |
| Row11 | whole milk | [pip fruit] | 296 | 3.01 | 39.8 | 744 | 7.56 | 1.557 | 155.7 | 2,513 | 25.552 |
| Row33 | whole milk | [other vegetables] | 736 | 7.484 | 38.7 | 1,903 | 19.3 | 1.514 | 151.36 | 2,513 | 25.552 |
| Row13 | whole milk | [pastry] | 327 | 3.325 | 37.4 | 875 | 8.9 | 1.463 | 146.26 | 2,513 | 25.552 |
| Row16 | whole milk | [citrus fruit] | 300 | 3.05 | 36.9 | 814 | 8.28 | 1.442 | 144.24 | 2,513 | 25.552 |
| Row5 | whole milk | [fruit/vegetable juice] | 262 | 2.664 | 36.8 | 711 | 7.23 | 1.442 | 144.22 | 2,513 | 25.552 |
| Row15 | other vegetables | [citrus fruit] | 284 | 2.888 | 34.9 | 814 | 8.28 | 1.803 | 180.31 | 1,903 | 19.349 |
| Row10 | other vegetables | [pip fruit] | 257 | 2.613 | 34.5 | 744 | 7.56 | 1.785 | 178.52 | 1,903 | 19.349 |
| Row1 | whole milk | [newspapers] | 269 | 2.735 | 34.3 | 785 | 7.98 | 1.341 | 134.11 | 2,513 | 25.552 |
| Row24 | other vegetables | [tropical fruit] | 353 | 3.589 | 34.2 | 1,032 | 10.5 | 1.768 | 176.78 | 1,903 | 19.349 |
| Row18 | rolls/buns | [sausage] | 301 | 3.061 | 32.6 | 924 | 9.4 | 1.771 | 177.1 | 1,809 | 18.393 |
| Row20 | whole milk | [sausage] | 294 | 2.989 | 31.8 | 924 | 9.4 | 1.245 | 124.53 | 2,513 | 25.552 |
| Row22 | whole milk | [bottled water] | 338 | 3.437 | 31.1 | 1,087 | 11.1 | 1.217 | 121.69 | 2,513 | 25.552 |
| Row30 | other vegetables | [yogurt] | 427 | 4.342 | 31.1 | 1,372 | 14 | 1.609 | 160.85 | 1,903 | 19.349 |
| Row26 | root vegetables | [other vegetables,whole milk] | 228 | 2.318 | 31 | 736 | 7.48 | 2.842 | 284.21 | 1,072 | 10.9 |
| Row32 | whole milk | [rolls/buns] | 557 | 5.663 | 30.8 | 1,809 | 18.4 | 1.205 | 120.5 | 2,513 | 25.552 |
| Row29 | yogurt | [other vegetables,whole milk] | 219 | 2.227 | 29.8 | 736 | 7.48 | 2.133 | 213.3 | 1,372 | 13.95 |
| Row34 | other vegetables | [whole milk] | 736 | 7.484 | 29.3 | 2,513 | 25.6 | 1.514 | 151.36 | 1,903 | 19.349 |
| Row4 | other vegetables | [fruit/vegetable juice] | 207 | 2.105 | 29.1 | 711 | 7.23 | 1.505 | 150.47 | 1,903 | 19.349 |
| Row6 | yogurt | [whipped/sour cream] | 204 | 2.074 | 28.9 | 705 | 7.17 | 2.074 | 207.43 | 1,372 | 13.95 |
| Row19 | other vegetables | [sausage] | 265 | 2.695 | 28.7 | 924 | 9.4 | 1.482 | 148.22 | 1,903 | 19.349 |
| Row23 | yogurt | [tropical fruit] | 288 | 2.928 | 27.9 | 1,032 | 10.5 | 2.001 | 200.05 | 1,372 | 13.95 |
| Row9 | tropical fruit | [pip fruit] | 201 | 2.044 | 27 | 744 | 7.56 | 2.575 | 257.46 | 1,032 | 10.493 |
| Row14 | yogurt | [citrus fruit] | 213 | 2.166 | 26.2 | 814 | 8.28 | 1.876 | 187.58 | 1,372 | 13.95 |
| Row21 | soda | [bottled water] | 285 | 2.898 | 26.2 | 1,087 | 11.1 | 1.504 | 150.36 | 1,715 | 17.438 |
| Row3 | yogurt | [fruit/vegetable juice] | 184 | 1.871 | 25.9 | 711 | 7.23 | 1.855 | 185.51 | 1,372 | 13.95 |
| Row17 | soda | [sausage] | 239 | 2.43 | 25.9 | 924 | 9.4 | 1.483 | 148.33 | 1,715 | 17.438 |
| Row35 | whole milk | ? | 2513 | 25.552 | 25.6 | 9,835 | 100 | 1 | 100 | 2,513 | 25.552 |
| Row2 | soda | [fruit/vegetable juice] | 181 | 1.84 | 25.5 | 711 | 7.23 | 1.46 | 145.99 | 1,715 | 17.438 |
| Row0 | whole milk | [bottled beer] | 201 | 2.044 | 25.4 | 792 | 8.05 | 0.993 | 99.324 | 2,513 | 25.552 |
| Row12 | other vegetables | [pastry] | 222 | 2.257 | 25.4 | 875 | 8.9 | 1.311 | 131.12 | 1,903 | 19.349 |

Table 1.4a. association rules with min support of 700 and min confidence of 25%

Table 1.4a shows that are 36 association rules mined. I reorder the table by the Rule Confidence column in descending order. The top row (Row8) indicates that I can be 45% confident that if a whipped/sour cream is bought, then whole milk is also bought together at the same time. The frequency of such item set in our dataset is 317 times. The second row (Row28) indicates that I can be 44.9% confident that if root vegetables are bought, then whole milk is also bought together at the same time. The frequency of such item set in our dataset is 481 times. I compare consequents to confidence percentage (figure 1.4b) and average of all kinds of support of difference products listed in table 1.4a (figure 1.4c).

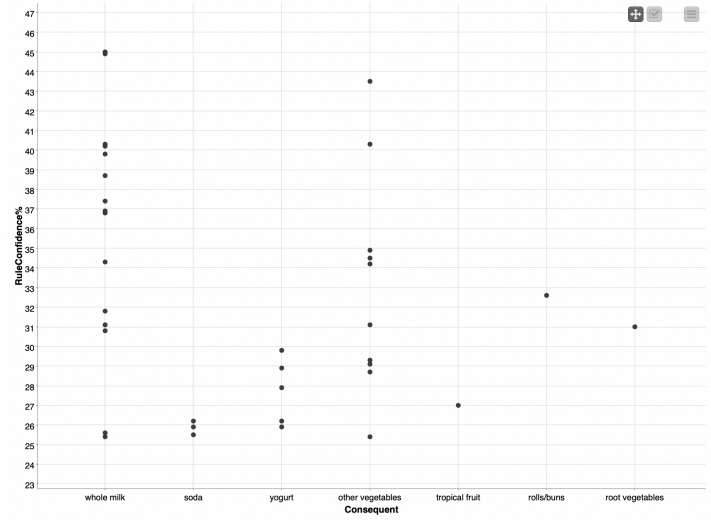


Figure 1.4b. Confidence (%) of consequents

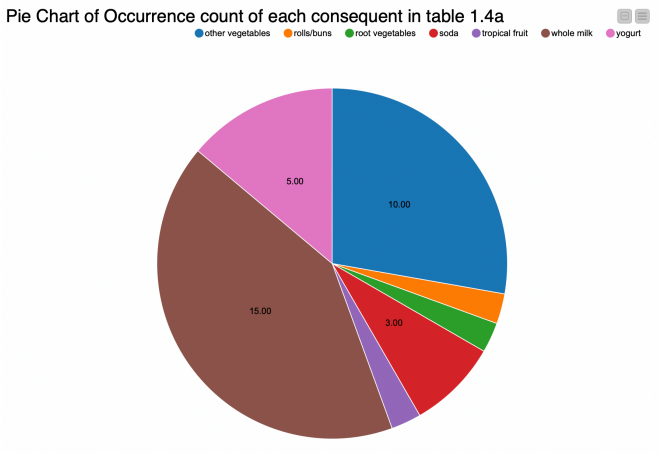


Figure 1.4d. Pie Chart of Occurrence count of each consequent in table 1.4a

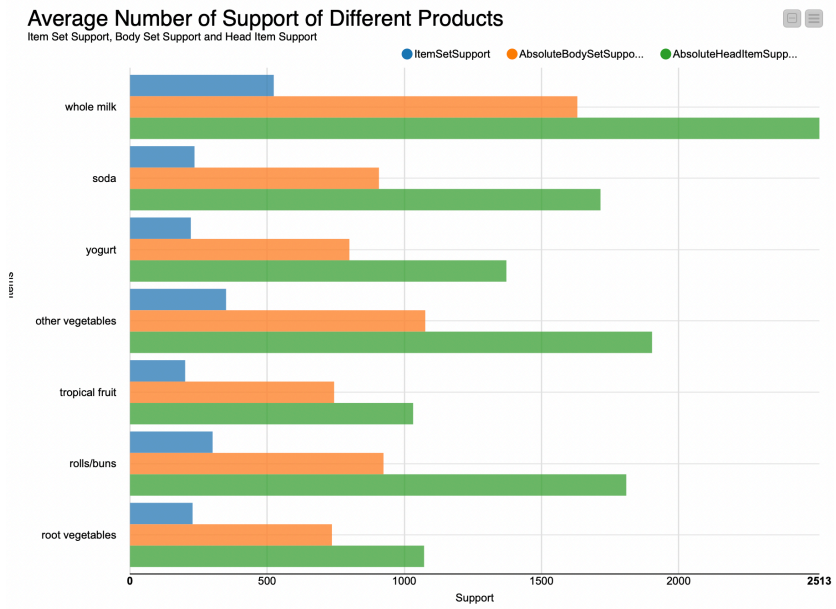


Figure 1.4c Average number of support of different products

I also notice that whole milk is a hot item in our top confidence item sets, occupying 10 consequents in the top 14, followed by other vegetables (4 consequents in the top 14). Figure 1.4d shows the occurrence count of each consequent in table 1.4a.

### 1.4.5. Conclusion

In this task, an association rule algorithm called apriori is used to find combination of items that are bought one after another. I predefine minimum support and minimum confidence in our learner node and mined 36 association rules out of 9835 transactions. The result reveals that whole milk and other vegetables are hot consequent items. It gives the store manager an insight on how they can build up a strategy around these items.

## 2. Text Mining

In this task, our objective is to transform unstructured text into a machine-learning-usable document and then perform 1 clustering using k-Means and 2 classifications using kNN and SVM.

### 2.1. Dataset Analysis

I will use yelp labelled, one of the sentiment labelled sentences datasets. It gathers 500 positive and 500 negative sentences that were randomly selected from yelp.com. The features are text sentences extracted from reviews of products, movies, and restaurants. For classification models, the target will be the feature "Sentiment".

### 2.2. Preprocessing (unstructured to structured data)

First, I need some preparation before entering the next step. This includes converting the unstructured text from strings to document format (Strings To Document), renaming "Column1" to "Sentiment" while converting the it into string type (Column Rename) in order to use it as a target in classification, and filtering all columns besides Document and Sentiment (Column Filter). I also assign a part of speech to each term of the document. (POS Tagger).

Entering the clean up step, I first remove punctuation (Punctuation Erasure), numbers (Number Filter), word with less than 3 characters (N Chars Filter) and Stop words (Stop Word Filter). I then convert all text into lowercase (Case Converter).

Then it is the step I apply stemming (Snowball Stemmer). I create a bag of words (Bag of Words Creator) and calculate their inverse document frequency (IDF), relative term frequency (TF — check the "Relative frequency" box), absolute term frequency (TF), and TF-IDF (Java Snippet(simple) — enter java code "return $TF rel$ * $IDF$" in method body) which will then be used as feature vector values.

The last step is to create a document vector (Document Vector) and class labels (Category To Class) and I keep no more than these two types of column (Column Filter).

It is important to check if our dataset contains any outliers since they (if any) may make our distance-based models (kNN and k-Means) misleading. In figure 2.2a, the Numeric Outliers node shows us that in food[NN(POS)] and servic[NN(POS)] more than 5% of the total members are extreme outliers (k=3). Thus I decide to omit both columns (Column Filter).

| Row ID | S Outlier column | I Member count | I Outlier count |
|---|---|---|---|
| Row7 | food[NN(POS)] | 977 | 120 |
| Row84 | servic[NN(POS)] | 977 | 83 |
| Row241 | time[NN(POS)] | 977 | 42 |
| Row82 | friendli[JJ(POS)] | 977 | 27 |
| Row139 | nice[JJ(POS)] | 977 | 25 |
| Row11 | amaz[JJ(POS)] | 977 | 24 |

Figure 2.2a features with the most extreme outliers

Although I have distance-based algorithms, all features are on the same scale (after vectoring) so there is no need to run a feature scaling.

Everything node is in the same setting as week 10 exercise. However, I find a big problem that our class labels are not binary output like the one I have in week 10 exercise (positive or negative) but instead long sentences. I spend a long time trying to fix it but no luck. The consequence is that there are errors when creating SVM and kNN models and even a k-Means model is created it looks like unreliable. Thus I attempt to create an alternative workflow (see Task 2: Text Mining (Alternative)) with some changes:

1. Renaming Column1 to Sentiment and converting it into string type.

2. Connecting Java Snippet (simple) node directly to Partitioning node (removing Document Vector, Category To Class, Column Filter x2).

3. Normalizing (min-max) all features for SVM and k-NN (Normalizer) since they are on different scales (table from Java Snippet (simple)).

4. Checking for extreme outliers in the new table and no concerned amount is found (figure 2.2b)

| Row ID | S Outlier column | I Member count | I Outlier count |
|---|---|---|---|
| Row0 | IDF | 3950 | 0 |
| Row1 | TF rel | 3950 | 111 |
| Row2 | TF abs | 3950 | 70 |
| Row3 | TFIDF | 3950 | 94 |

Figure 2.2b number of extreme outlier of features

I am not sure if this is a right approach but it seems like a solution to allow me to continue into the machine learning stage and evaluation stage with some meaningful models.

**2.3. Parameter Tuning**

For SVM classification, among 70/30, 80/20 and 90/10 partitioning, 70/30 yields the highest accuracy. After trying different kernels, using the RBF kernel yields higher accuracy more frequently than other two kernels. Therefore I decide to go for 70/30 partitioning and RBF kernel while keep other settings default.

For k-NN classification, among 70/30, 80/20 and 90/10 partitioning, 90/10 yields the highest accuracy. after trying different values (3 to 20) of number of neighbors to consider (k), I found that setting it at 18 yields the highest accuracy. I will use it while keeping other settings default.

For k-Means clustering, among 70/30, 80/20 and 90/10 partitioning, 70/30 yields the highest MSC. I evaluate the content of clusters in each k-value from 2 to 10 and find that the optimal k=2. I then evaluate both centroid initializations and find that using random initialisation consistently yields better MSC. Therefore I decide to go for 70/30 partitioning, k=2 and use random initialisation while keep other settings default.

**2.4. Results**

Table 2.2c Classification:

| Algorithms | TP | FP | TN | FN | Recall | Precision | Sensitivity | Specificity | F-measure | Accuracy (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| SVM | 246 | 211 | 395 | 333 | 0.425 | 0.538 | 0.425 | 0.652 | 0.475 | 54.093 |
| k-NN | 85 | 54 | 141 | 115 | 0.425 | **0..612** | 0.425 | **0.723** | **0.501** | **57.215** |

For a text mining case, we care about its true positive and true negative rates. As we can see in table 2.2c, both algorithms have close accuracy (3.1% difference). K-NN has better performance in most metrics. Since k-NN is also less time-consuming than SVM in this case, it is certainly my choice.

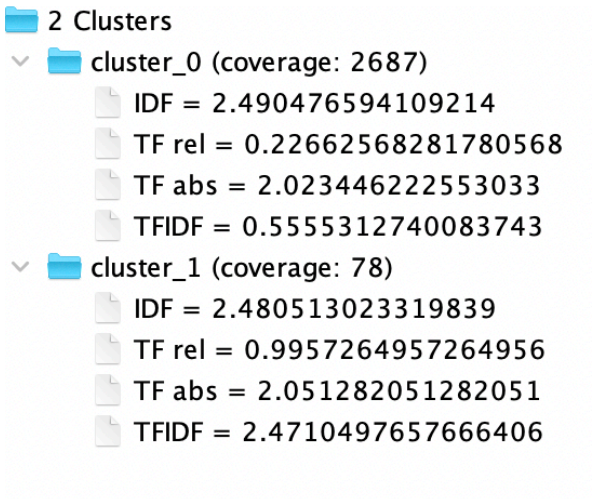


Figure 2.2d Details of clusters

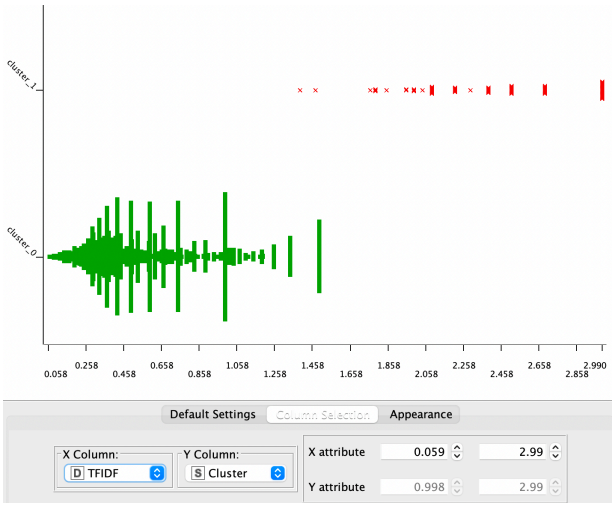


Figure 2.2e scatter plot of TFIDF and clusters

Figure 2.2d shows the details of both clusters. Cluster 0 covers 2687 observations while cluster 1 covers only 78. Their main differences are TF rel (0.227 for cluster 0 vs 0.996 for cluster 1) and TFIDF (0.556 for cluster 0 vs 2.471 for cluster 1). A scatter plot of TFIDF and the clusters are shown in figure 2.2e. Meanwhile, from table 2.2f we can see that both clusters have nearly identical MSC (0.621 for cluster 0 vs 0.617 for cluster 1), with an overall MSC of 0.621.

| Row ID | Mean Silhouette Coefficient |
|---|---|
| cluster_0 | 0.621 |
| cluster_1 | 0.617 |
| Overall | 0.621 |

Table 2.2f Mean silhouette coefficient of clusters

### 2.5. Conclusion

In this task, a text from yelp.com is mined by 3 different algorithms with multiple preprocessing steps. An attempt followed the instructions from our class materials is not fully successful so I adopt an alternative approach that uses possibly incorrect features to continue my work and demonstrate my knowledge gained from class.

K-NN is more favourable than SVM in this case with an accuracy of 3.1% higher. They both perform slight better after normalization. I use k-Means to divide the data into 2 clusters of similar SMC.