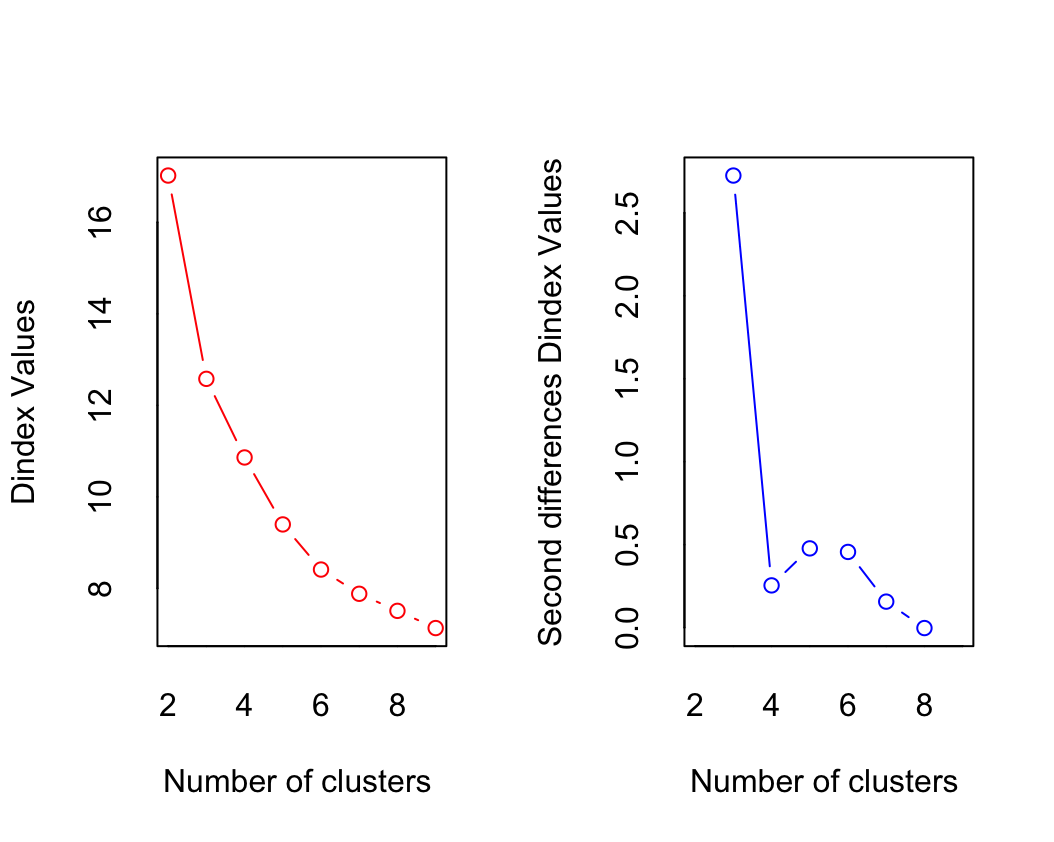
Homework II

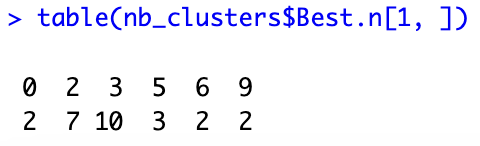
Data Mining

113550802 - Elisabeth Le Leslé & 113550819 - Guillaume Drui

**Question 1**:

After loading the necessary libraries (NbClust, cluster, factoextra) and removing the last column (“red\_wine\_data$quality”), we determine the optimal number of clusters. We first use the NbClust package, which yields the following graph.

**Dindex Values Plot (Left)**: This plot shows a significant drop in the Dindex values as the number of clusters increases from 2 to around 4 or 5. After that, the decrease becomes less pronounced, indicating that adding more clusters contributes less to improving the clustering quality.

**Second Differences Dindex Values Plot (Right)**: This plot helps identify the "elbow" point where the improvement in clustering starts to diminish. A noticeable drop occurs from 2 to 4 clusters, and then the second differences become more stable, suggesting a potential optimal cluster count around 4 or 5.

Next, we run the following command, which yields:

It provides a summary of the optimal number of clusters suggested by the various methods used in the NbClust package.

We interpret the table as : the 1st line being the potential number of clusters and the 2nd as the frequency of recommendations for a cluster number by the methods of the NbClust package.

The most suggested optimal number of clusters is **3** since it has the highest count (10 methods) recommending it.

Furthermore, we apply the kmeans() function, which yields the **Within Cluster Sum of Squares (WCSS)**, that measures the compactness of the clusters and that we aim to minimize, as well as the **Between Cluster Sum of Squares (BCSS)** that measures the variance between clusters and that we aim to maximize.

More specifically, it gives us the ratio of between SS to total SS (in %), which indicates that a significant portion of the total variance is explained by the separation between clusters, and that we aim to maximize.

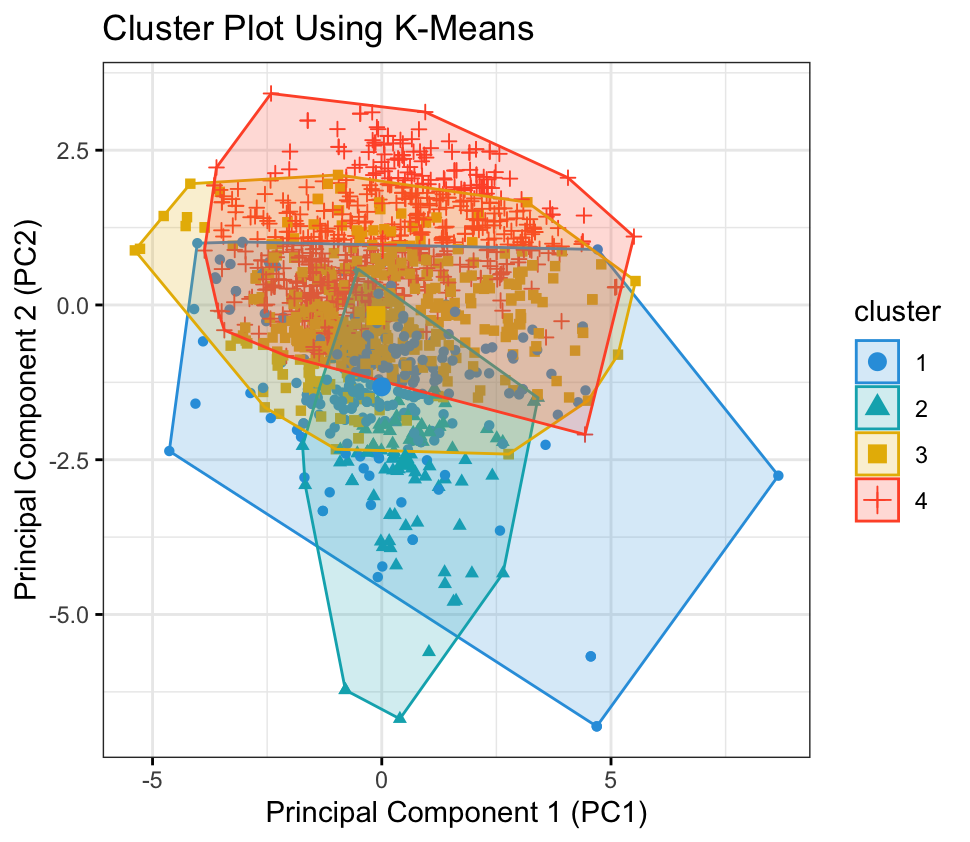
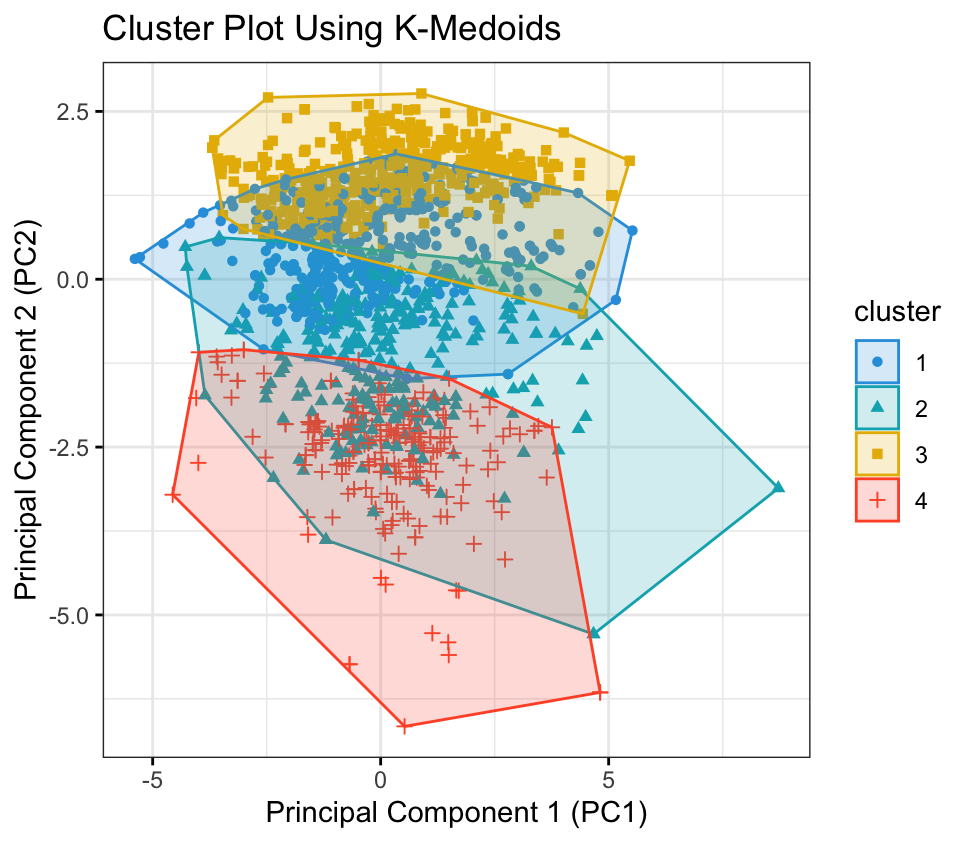
These are the respective % when taking optimal\_clusters as:

* optimal\_clusters==3: between\_SS / total\_SS = 79.3 %
* optimal\_clusters==4: between\_SS / total\_SS = 85.2 %
* optimal\_clusters==5: between\_SS / total\_SS = 88.4 %

We notice a positive correlation between the ratio and the number of clusters, yet this could cause overfitting.

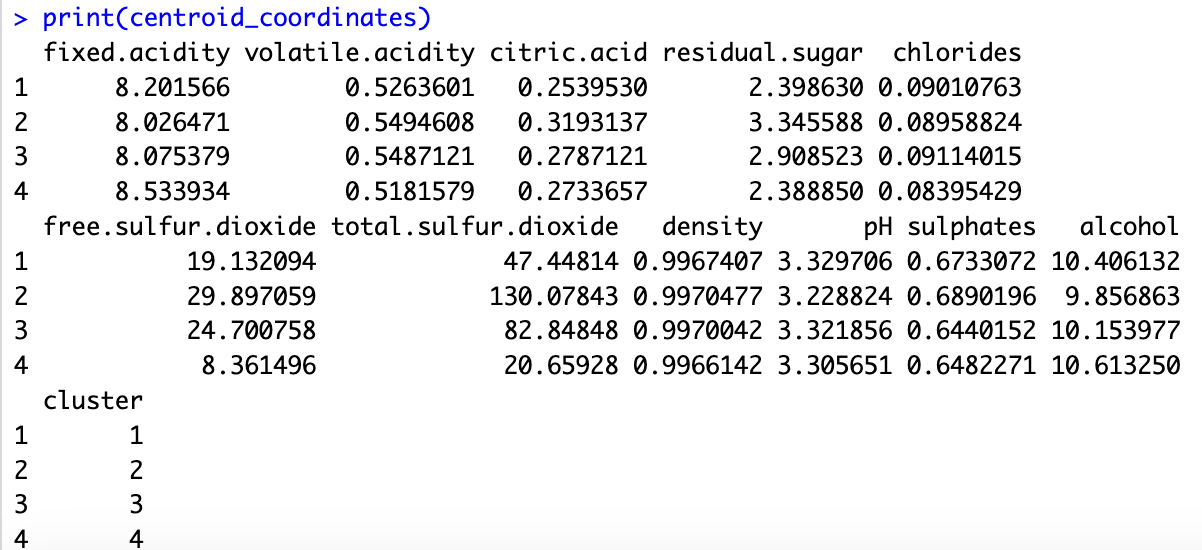
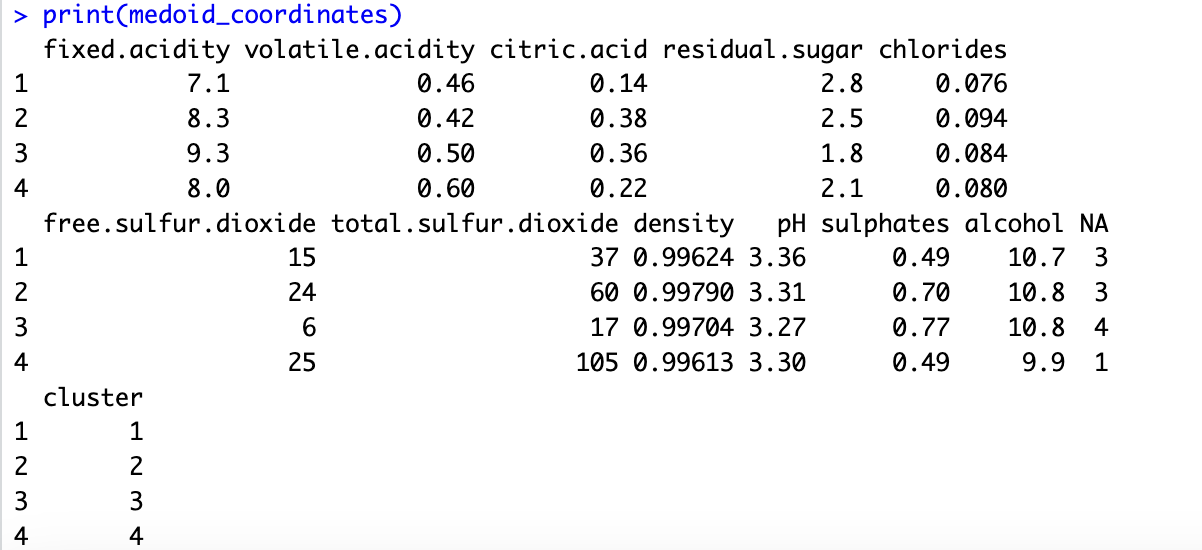
We then choose to set 4 as the optimal number of clusters, as a compromise between the graphs and the ratio value.

Then, we visualize clusters using k-means and k-medoids using fviz\_cluster():

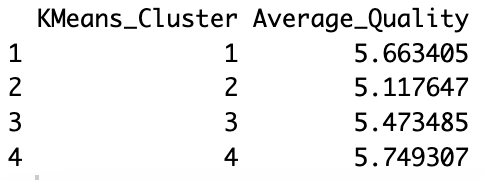
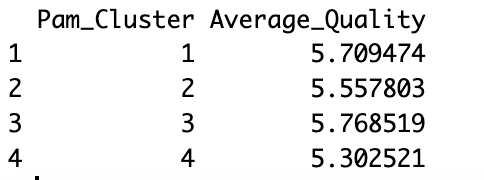


We notice from these graphs that, due to the high number of input features, the quality of a wine depends on many factors. Thus, the clusters overlap and are hard to discern. However, the clusters are better defined using the K-medoids method than K-means clustering.

Furthermore, we extract the coordinates of each cluster for K-means and K-medoids:



Finally, we use the aggregate() function to calculate the average quality within each cluster for k-means and k-medoids:



**Question 2:**

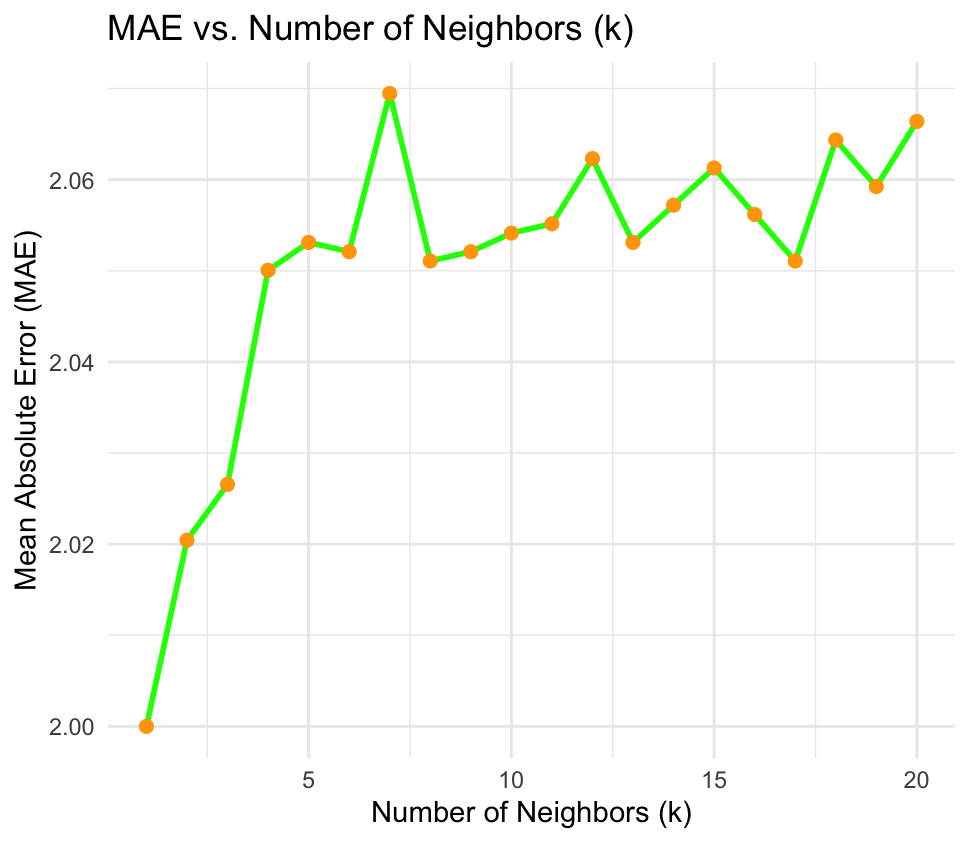
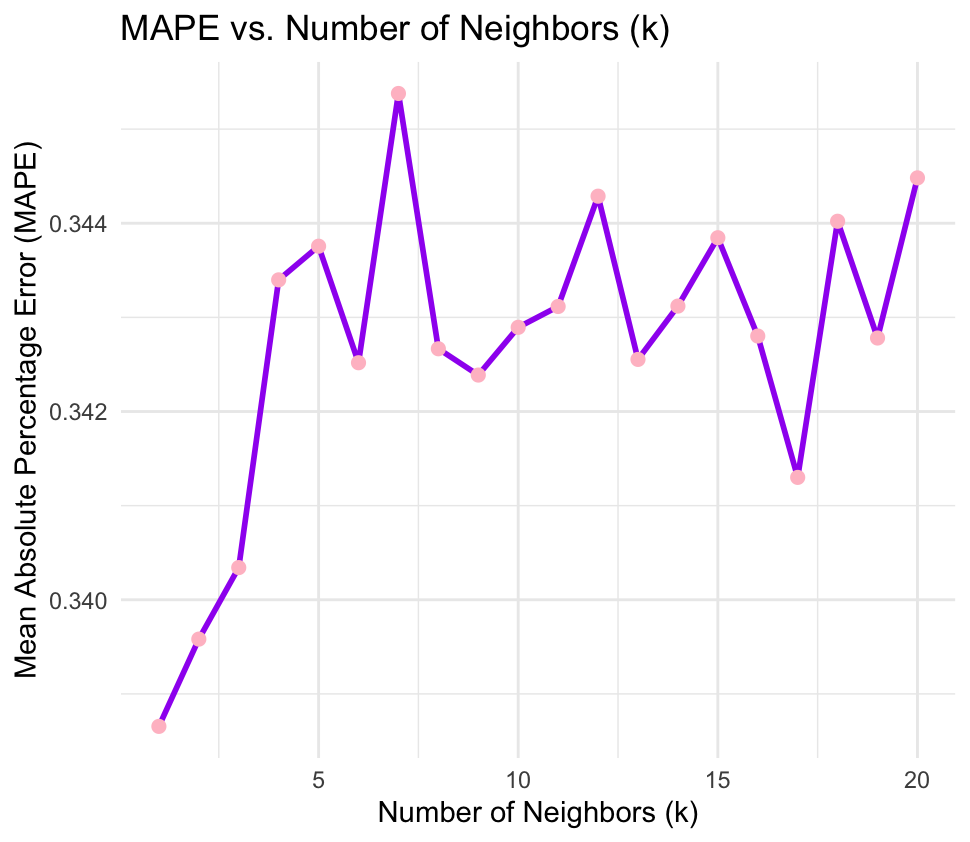
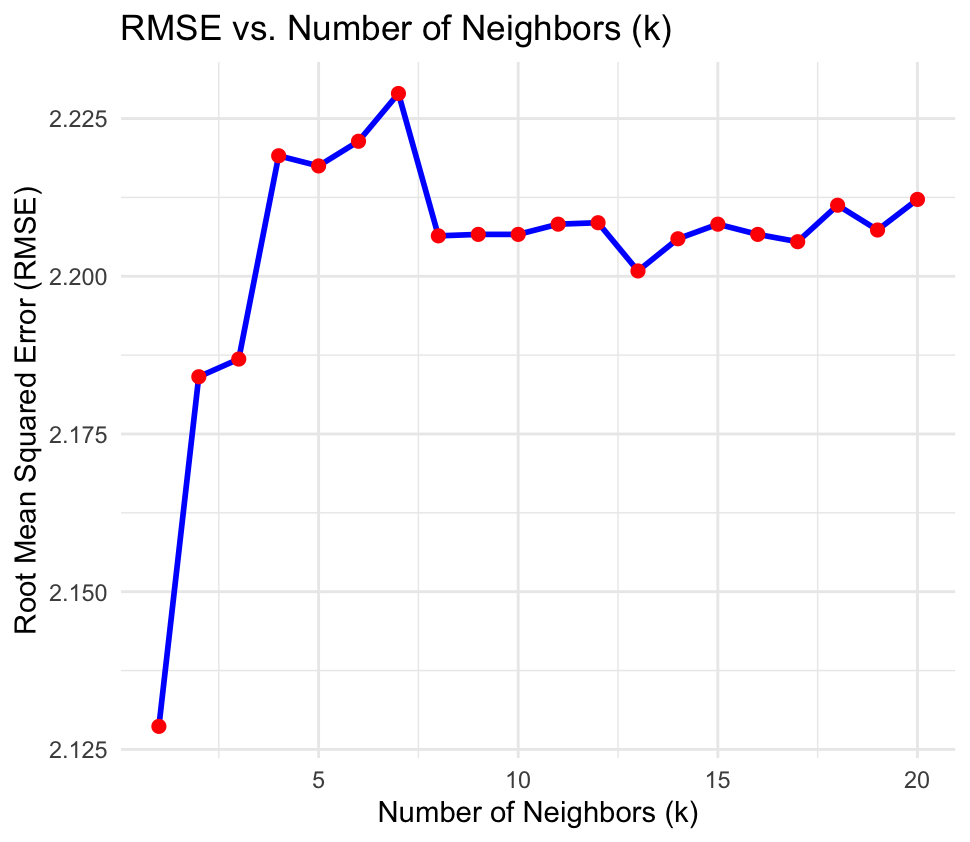
After importing the necessary libraries (caret, class, Metrics) and loading the dataset, we fit the Multiple Linear Regression (MLR) Model using the lm() function where “quality” is the response variable and all other variables are predictors.

We extract significant predictors meaning those with p-values less than 0.05, in our case "fixed.acidity", "volatile.acidity", "residual.sugar", "free.sulfur.dioxide", "density", "pH", "sulphates", "alcohol" (8/11 total).

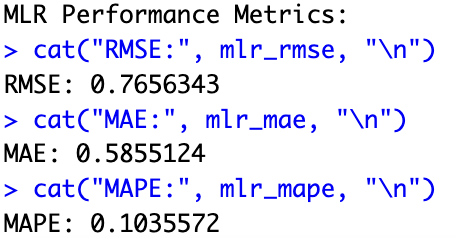
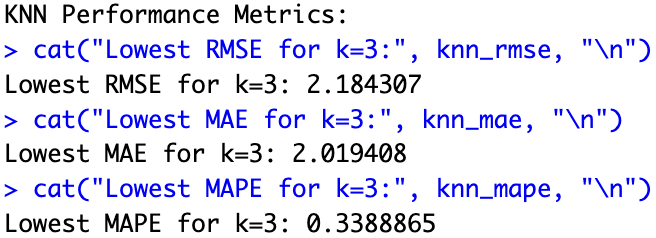
We then prepare the dataset for KNN by splitting the data into training (80%) and testing (20%) sets, only keeping the significant columns, and normalizing it while adding back the “quality” column.

Next, we wish to determine the optimal k for the KNN. To do so, we iterate k from 1 to 20 and perform KNN as well as calculate performance metrics Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).

After computing each of their values w.r.t. k, we plot their individual graphs in an attempt to find the k that minimizes their graph the most. Here are the graphs:

We notice that they all exponentially increase from k=4, thus we choose k=3 for our KNN.

We obtain the following values for the performance metrics, which we compare with MLR.



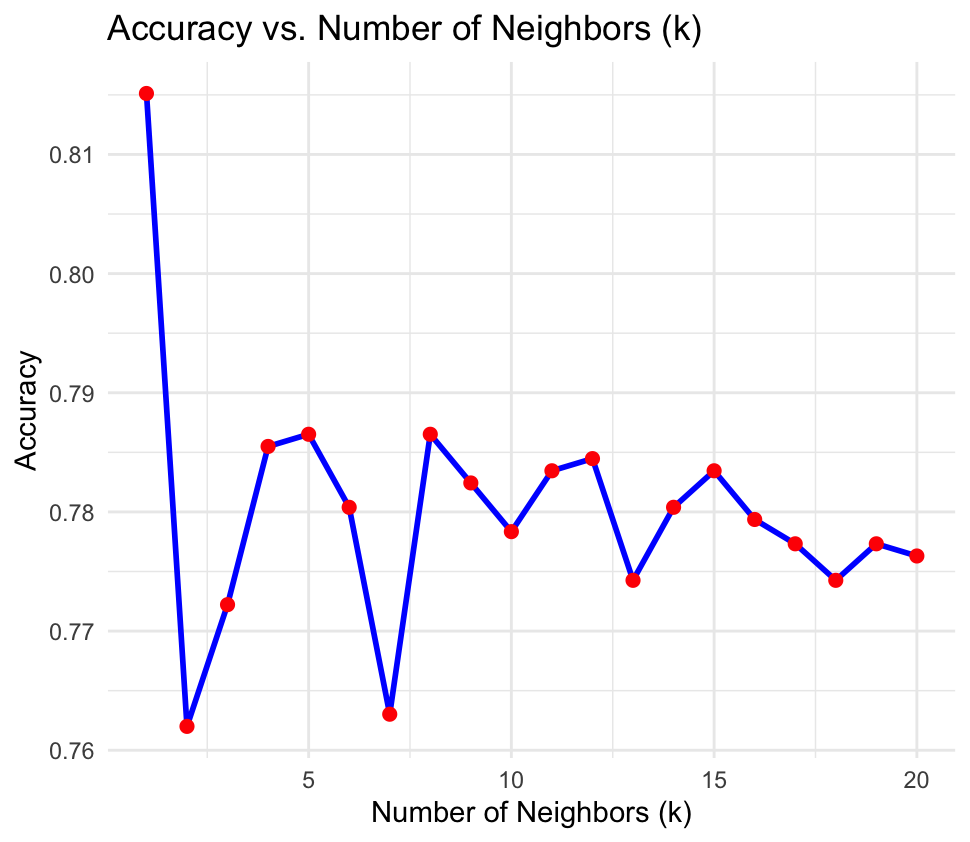
From these results, we deduce that MLR is more adapted to our model. This can be explained by:

* A **linear relationship between the predictors and the target variable**. If the underlying relationship is indeed linear, MLR can model this effectively and provide accurate predictions.
* A **high dimensional dataset** (in our case there are 8 features thus 8 dimensions). MLR performs well as it can give a clear interpretation of how each predictor influences the outcome.
* A **high training data size**. MLR generally scales better with larger datasets than KNN, as it requires fitting a model once, whereas KNN requires calculating distances for all training points for each prediction, which can be computationally expensive as the dataset grows.
* The **presence of noise and outliers**. MLR can be more robust to noise if regularization techniques are applied (like Ridge or Lasso regression), which can help prevent overfitting, whereas KNN is sensitive to outliers and noise because it relies on the nearest neighbors. Outliers can disproportionately influence predictions.

**Question 3:**

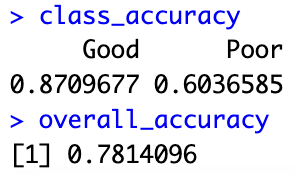
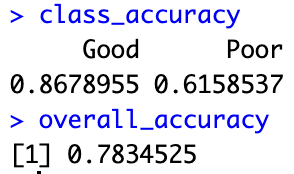
To construct a KNN capable of conducting the forecast of a binary rating (“Good” or “Poor”), we start by preprocessing the data. We modify the “quality” column by using the as.factor() function, creating a binary response variable:

* Poor: Quality 3, 4, 5
* Good: Quality 6, 7, 8

Similar to the previous question, we split the data into training and testing sets then normalize it. We then iterate k from 1 to 20 to determine the optimal k (that best maximizes the overall accuracy of the rating forecast).

To do so, we plot the accuracy w.r.t k:

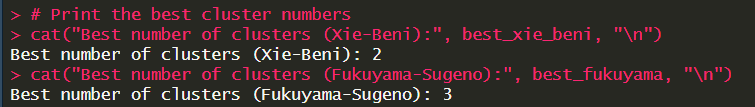
From the plot, we determine two potential optimal values of k: 5 and 8, with respective accuracy results:

For k=5: For k=8:

Thus, we pick k=5 as the optimal number of neighbors to maximize model accuracy.

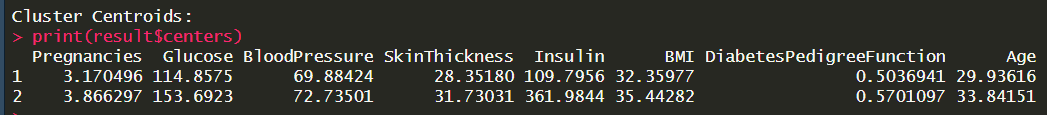
**Question 4:**

The first step is to remove the zeros in this task which is done by setting the zeros to NA and then using na.omit to remove those rows. We also remove the last column and perform fuzzy C-Means. We find that the best number of clusters are:

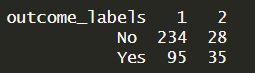


1. **For Xie-Beni:**

These are the two clusters with the centroids being the following values



The outcome table is given by:

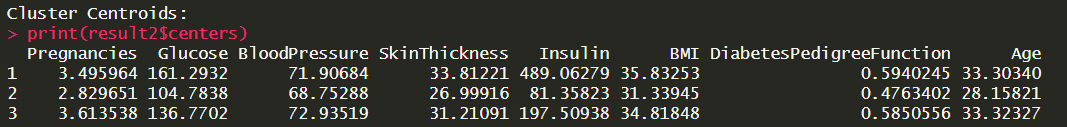


Finally the Euclidian distance between the two clusters is 255.2491

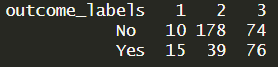
For Xie-Beni we can clearly see that there are two groups: those with high Insulin and those with low. This directly correlates to the diabetes prediction shown in the table.

1. **For Fukuyama:**

These are the three clusters with the centroids being the following values



The outcome table is given by:



Finally the Euclidian distance between the clusters is:

Distance between centroid 1 and 2: 411.7278

Distance between centroid 1 and 3: 292.5981

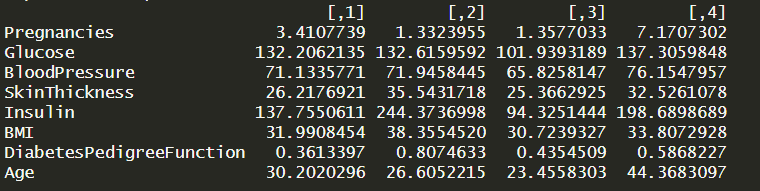
Distance between centroid 2 and 3: 120.7844

For Fukuyama-Sugeno the data seems again to be separated by insulin level but we could also argue that it is separated by glucose level. One key observation is that the prediction is more ambiguous for people with diabetes because clusters 1 and 3 have a +/- 50/50 chance of outputting yes or no.

**Question 5:**

Like in the previous exercise we remove the zeros and the last column. However instead of fuzzy C-Means we use GMC. Here are the results:

* The best number of clusters is 4
* The associated centroids are:

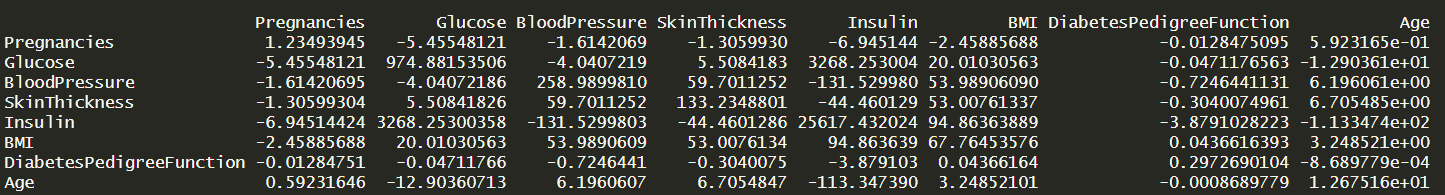


* The covariance matrix for each cluster is given by:

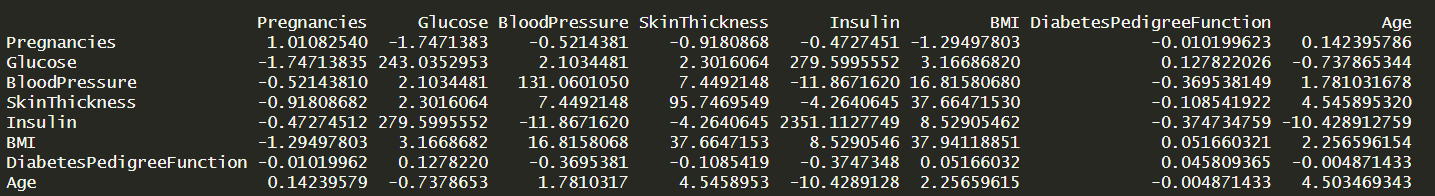
Cluster 1:



Cluster 2:



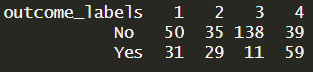
Cluster 3:



Cluster 4:



* The outcome table is:



* The Mahalanbis distance between each cluster is:
* Mahalanobis distance between centroid 1 and 2: 4.0048
* Mahalanobis distance between centroid 1 and 3: 2.1077
* Mahalanobis distance between centroid 1 and 4: 3.9565
* Mahalanobis distance between centroid 2 and 3: 1.8757
* Mahalanobis distance between centroid 2 and 4: 6.9578
* Mahalanobis distance between centroid 3 and 4: 11.8400

Observations: We see that the 4th clusters has a higher age group with more pregnancies and insulin levels. As shown in the table, those persons have the highest diabetes risk out of all groups. Group 3 is characterized by a low age glucose, insulin and pregnancies. All those facts contribute to this group being the least at risk of having diabetes. Groups 1 and 2 differ in pregnancies and insulin levels but the prediction of whether they have diabetes or not is ambiguous.

**Question 6:**

In this exercise we first separate the features from the labels in the data. We then do hierarchical clustering using the 4 different linkage methods: single, complete, average, Ward’s.

Using DB index:

The best number of clusters for Single Link is 2

The best number of clusters for Complete Link is 6

The best number of clusters for Group Average is 2

The best number of clusters for Ward's Method is 2

Majority class for:

* Single link: 1 Male, 2 Female
* Complete link: 1 Male, 2 Male, 3 Male, 4 Female, 5 Male, 6 Male
* Group average: 1 Male, 2 Female
* Ward’s method: 1 Male, 2 Female

All methods except complete link separate the data into two groups: Male and female. Complete link is able to find more subgroups but they stay male dominated.