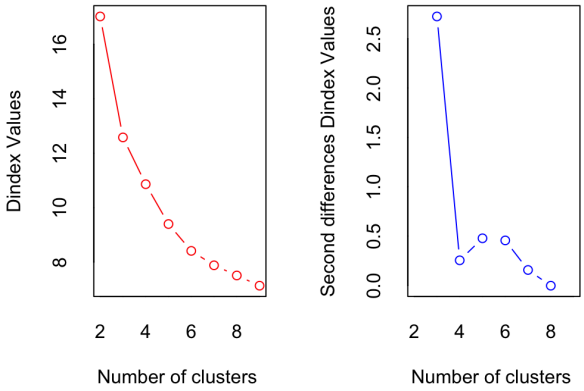
Homework II
Data Mining

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:

```
> table(nb_clusters$Best.n[1, ])

 0  2  3  5  6  9
 2  7 10  3  2  2
```

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:

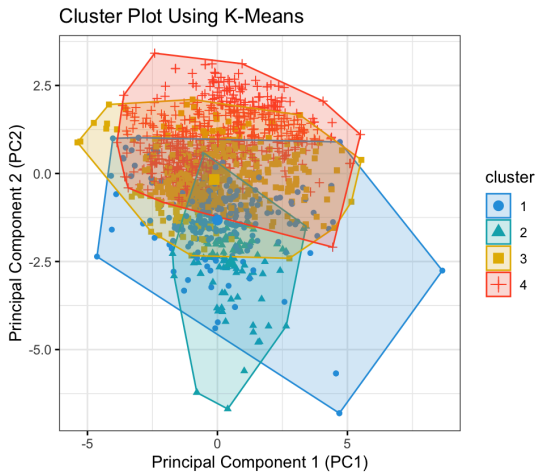
- optimal_clusters==3: between_SS / total_SS = 79.3 %
- optimal_clusters==4: between_SS / total_SS = 85.2 %

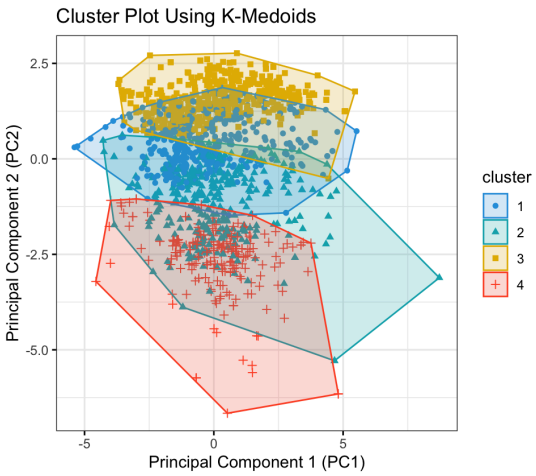
- optimal_clusters==5: between_SS / total_SS = 88.4 %

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:

```
> print(centroid_coordinates)
  fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
1      8.201566        0.5263601   0.2539530       2.398630 0.09010763
2      8.026471        0.5494608   0.3193137       3.345588 0.08958824
3      8.075379        0.5487121   0.2787121       2.908523 0.09114015
4      8.533934        0.5181579   0.2733657       2.388850 0.08395429
  free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates   alcohol
1           19.132094             47.44814 0.9967407 3.329706 0.6733072 10.406132
2           29.897059            130.07843 0.9970477 3.228824 0.6890196  9.856863
3           24.700758             82.84848 0.9970042 3.321856 0.6440152 10.153977
4            8.361496             20.65928 0.9966142 3.305651 0.6482271 10.613250
```

```
> print(medoid_coordinates)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           7.1             0.46        0.14            2.8     0.076
2           8.3             0.42        0.38            2.5     0.094
3           9.3             0.50        0.36            1.8     0.084
4           8.0             0.60        0.22            2.1     0.080
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  15                   37 0.99624 3.36      0.49    10.7
2                  24                   60 0.99790 3.31      0.70    10.8
3                   6                   17 0.99704 3.27      0.77    10.8
4                  25                  105 0.99613 3.30      0.49     9.9
```

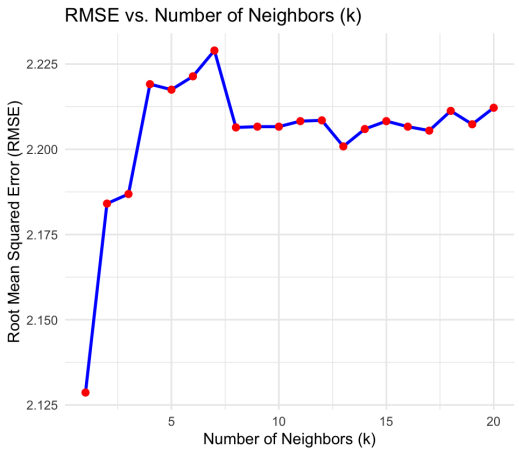
Finally, we use the aggregate() function to calculate the average quality within each cluster for k-means and k-medoids:

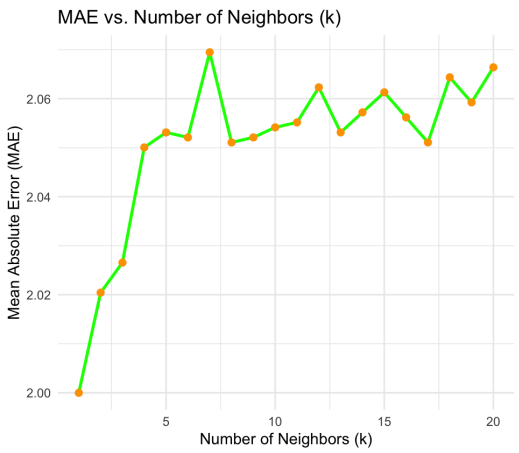
	KMeans_Cluster	Average_Quality
1	1	5.663405
2	2	5.117647
3	3	5.473485
4	4	5.749307

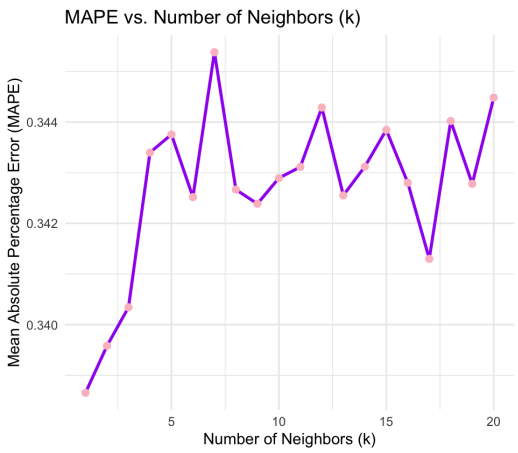
	Pam_Cluster	Average_Quality
1	1	5.709474
2	2	5.557803
3	3	5.768519
4	4	5.302521

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.

```
KNN Performance Metrics:
> cat("Lowest RMSE for k=3:", knn_rmse, "\n")
Lowest RMSE for k=3: 2.184307
> cat("Lowest MAE for k=3:", knn_mae, "\n")
Lowest MAE for k=3: 2.019408
> cat("Lowest MAPE for k=3:", knn_mape, "\n")
Lowest MAPE for k=3: 0.3388865
```

```
MLR Performance Metrics:
> cat("RMSE:", mlr_rmse, "\n")
RMSE: 0.7656343
> cat("MAE:", mlr_mae, "\n")
MAE: 0.5855124
> cat("MAPE:", mlr_mape, "\n")
MAPE: 0.1035572
```

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.

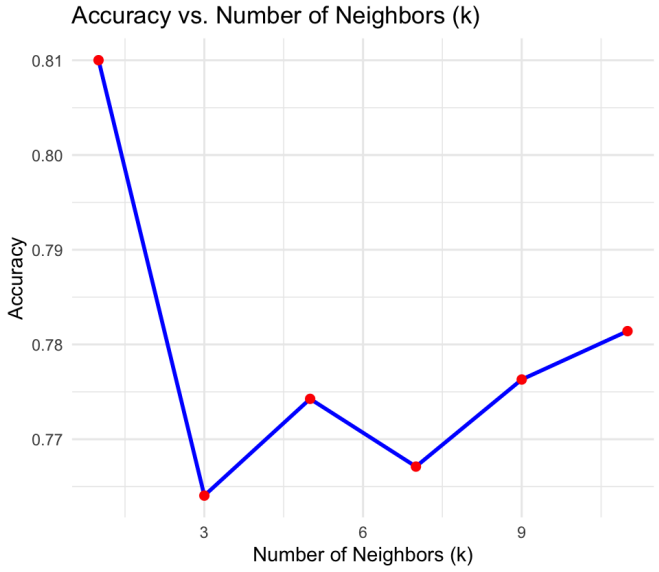
Question 3:

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 the all the odd k from 1 to 11 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 that the peak at k=5 is the optimal value, with the following accuracy results:

```
> class_accuracy
     Good      Poor
0.8725038 0.5914634
> overall_accuracy
[1] 0.7783453
```

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:

```
> # Print the best cluster numbers
> cat("Best number of clusters (Xie-Beni):", best_xie_beni, "\n")
Best number of clusters (Xie-Beni): 2
> cat("Best number of clusters (Fukuyama-Sugeno):", best_fukuyama, "\n")
Best number of clusters (Fukuyama-Sugeno): 3
```

1. **For Xie-Beni:**

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

```
Cluster Centroids:
> print(result$centers)
  Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI DiabetesPedigreeFunction      Age
1    3.170496 114.8575      69.88424      28.35180 109.7956 32.35977                0.5036941 29.93616
2    3.866297 153.6923      72.73501      31.73031 361.9844 35.44282                0.5701097 33.84151
```

The outcome table is given by:

```
outcome_labels   1   2
           No  234  28
           Yes  95  35
```

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.

2. **For Fukuyama:**

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

```
Cluster Centroids:
> print(result2$centers)
  Pregnancies  Glucose BloodPressure SkinThickness   Insulin      BMI DiabetesPedigreeFunction      Age
1    3.495964 161.2932      71.90684      33.81221 489.06279 35.83253                0.5940245 33.30340
2    2.829651 104.7838      68.75288      26.99916  81.35823 31.33945                0.4763402 28.15821
3    3.613538 136.7702      72.93519      31.21091 197.50938 34.81848                0.5850556 33.32327
```

The outcome table is given by:

```
outcome_labels   1   2   3
           No   10 178  74
           Yes  15  39  76
```

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:

	[,1]	[,2]	[,3]	[,4]
Pregnancies	3.4107739	1.3323955	1.3577033	7.1707302
Glucose	132.2062135	132.6159592	101.9393189	137.3059848
BloodPressure	71.1335771	71.9458445	65.8258147	76.1547957
SkinThickness	26.2176921	35.5431718	25.3662925	32.5261078
Insulin	137.7550611	244.3736998	94.3251444	198.6898689
BMI	31.9908454	38.3554520	30.7239327	33.8072928
DiabetesPedigreeFunction	0.3613397	0.8074633	0.4354509	0.5868227
Age	30.2020296	26.6052215	23.4558303	44.3683097

- The covariance matrix for each cluster is given by:

Cluster 1:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	3.32109504	-6.0608890	-0.5455346	-0.99060843	-0.6224874	-0.90781044	-0.025298983	1.581413615
Glucose	-6.06088897	768.7449253	31.7010627	22.28540417	529.3109484	18.46797464	0.429213584	0.262785619
BloodPressure	-0.54553458	31.7010627	89.3487137	-1.58702764	-28.4169377	8.40773208	-0.216307196	0.786087861
SkinThickness	-0.99060843	22.2854042	-1.5870276	74.31546679	-11.0635924	29.72781605	-0.035382016	2.436806460
Insulin	-0.62248739	529.3109484	-28.4169377	-11.06359239	4758.5001880	15.63721681	-0.776666201	-21.238881273
BMI	-0.90781044	18.4679746	8.4077321	29.72781605	15.6372168	26.33019288	0.045562292	1.304640553
DiabetesPedigreeFunction	-0.02529898	0.4292136	-0.2163072	-0.03538202	-0.7766662	0.04556229	0.022564972	0.002089339
Age	1.58141362	0.2627856	0.7860879	2.43680646	-21.2388813	1.30464055	0.002089339	29.439774401

Cluster 2:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	1.23493945	-5.45548121	-1.6142069	-1.3059930	-6.945144	-2.45885688	-0.0128475095	5.923165e-01
Glucose	-5.45548121	974.88153506	-4.0407219	5.5084183	3268.253004	20.01030563	-0.0471176563	-1.290361e+01
BloodPressure	-1.61420695	-4.04072186	258.9899810	59.7011252	-131.529980	53.98906090	-0.7246441131	6.196061e+00
SkinThickness	-1.30599304	5.50841826	59.7011252	133.2348801	-44.460129	53.00761337	-0.3040074961	6.705485e+00
Insulin	-6.94514424	3268.25300358	-131.5299803	-44.4601286	25617.432024	94.86363889	-3.8791028223	-1.133474e+02
BMI	-2.45885688	20.01030563	53.9890609	53.0076134	94.863639	67.76453576	0.0436616393	3.248521e+00
DiabetesPedigreeFunction	-0.01284751	-0.04711766	-0.7246441	-0.3040075	-3.879103	0.04366164	0.2972690104	-8.689779e-04
Age	0.59231646	-12.90360713	6.1960607	6.7054847	-113.347390	3.24852101	-0.0008689779	1.267516e+01

Cluster 3:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	1.01082540	-1.7471383	-0.5214381	-0.9180868	-0.4727451	-1.29497803	-0.010199623	0.142395786
Glucose	-1.74713835	243.0352953	2.1034481	2.3016064	279.5995552	3.16686820	0.127822026	-0.737865344
BloodPressure	-0.52143810	2.1034481	131.0601050	7.4492148	-11.8671620	16.81580680	-0.369538149	1.781031678
SkinThickness	-0.91808682	2.3016064	7.4492148	95.7469549	-4.2640645	37.66471530	-0.108541922	4.545895320
Insulin	-0.47274512	279.5995552	-11.8671620	-4.2640645	2351.1127749	8.52905462	-0.374734759	-10.428912759
BMI	-1.29497803	3.1668682	16.8158068	37.6647153	8.5290546	37.94118851	0.051660321	2.256596154
DiabetesPedigreeFunction	-0.01019962	0.1278220	-0.3695381	-0.1085419	-0.3747348	0.05166032	0.045809365	-0.004871433
Age	0.14239579	-0.7378653	1.7810317	4.5458953	-10.4289128	2.25659615	-0.004871433	4.503469343

Cluster 4:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	11.78898849	-7.2092225	-0.8259846	-1.1059797	-4.358776	-1.06849055	-0.076616010	4.900865955
Glucose	-7.20922245	1074.7477544	25.6109898	21.7428472	2219.474921	26.34444924	0.243730276	-6.997262894
BloodPressure	-0.82598464	25.6109898	115.6354441	13.5955233	-95.418739	17.19036970	-0.287965166	1.550931241
SkinThickness	-1.10597973	21.7428472	13.5955233	78.0153013	-33.386898	28.44285752	-0.101360788	0.236799738
Insulin	-4.35877572	2219.4749211	-95.4187395	-33.3868983	17808.459359	64.21165989	-2.730528333	-78.676328433
BMI	-1.06849055	26.3444492	17.1903697	28.4428575	64.211660	36.93250036	0.074762048	0.446648358
DiabetesPedigreeFunction	-0.07661601	0.2437303	-0.2879652	-0.1013608	-2.730528	0.07476205	0.097439925	0.009597146
Age	4.90086596	-6.9972629	1.5509312	0.2367997	-78.676328	0.44664836	0.009597146	90.230451033

- The outcome table is:

outcome_labels	1	2	3	4
No	50	35	138	39
Yes	31	29	11	59

- The Mahalanbis distance is:
    - Average Mahalanobis distance for cluster 1: 2.7003
    - Average Mahalanobis distance for cluster 2: 2.6825
    - Average Mahalanobis distance for cluster 3: 2.7488
    - Average Mahalanobis distance for cluster 4: 2.7257

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.