Table of Contents

[1. Purpose 3](#_Toc53834647)

[2. Methodology 3](#_Toc53834648)

[2.1 3](#_Toc53834649)

[3. Analysis 3](#_Toc53834650)

[4. Analysis 3](#_Toc53834651)

# 1. Purpose

The purpose of this report is to identify if the predictive performance of supervised (Regression tree & Neural network) and unsupervised (K-means clustering & K-NN regression) machine learning models is more accurate than that of the simpler regression models such as linear & logistics regressions. Within that context, the data of interest is the relationships between wine quality and physiochemical properties.

On top of the standard predictive accuracy measures such as RMSE, MAE, MAPE & MASE, a specific loss function is created to target the premium wine market. This will help winemaker to more accurately predict the wine quality score during the production process in order to optimise their marketing mix and maximise profit.

# 2. Methodology

Various Regression Tree with CP values ranging between 0.01 to 0.1 were modelled. As expected, when pruning the regression tree, the greater the CP value, the fewer physiochemical property segments featured.

Besides, multiple Neural Network (NN) were trained with one hidden layer and different number of nodes (4 & 5) using several seeds to determine the stability of the models. It’s well-known NN that may be unstable and using less layers / nodes (4 nodes in this case) or the same seed every time will improve the stability of the model. It’s also worth to note that using too many layers or nodes may reduce the model’s predictive accuracy, possibly due to overfitting . Sensitivity analysis using Lek’s profile was also run for all explanatory variables with default number of grouping (6) and 3 groups.

Moving on to K-means Clustering, the Elbow method indicated that the most appropriate nu. Moving from 6 to 7 clusters still demonstrated a sharp improvement in performance and from there on out, little improvement could be gained.

The Gap method demonstrates that the most appropriate k=2 as there was little to be gained from going to 3. If more clusters needed to be chosen, the next most appropriate point was a k=9.

The Silhouette method suggests that a k=2 would be most appropriate. However, if more groups were wanting to be selected, the next highest point was a k=7.

## 2.4. K-NN

K-NN model was trained and predictions were obtained. The model tended to perform in the middle of the pack and it was not possible to easily explain the impact certain wine characteristics had on wine quality.

# 3. Analysis

## 3.1. Regression Tree

When conducting the VIP regression tree, ‘Alc, ‘Sulphates’ and ‘VA’ are considered some of the most important variables, which aligns with what is represented in the regression tree. However, despite featuring as the second most important variable in the VIP regression tree, ‘Density’ does not feature as a segment on any regression tree, regardless of their CP value.

As outlined in Appendix X, the predicted quality score ranges from 5.1 to 6.6, subject to the pathway followed. For example, wines with lower alcohol and sulphate levels are predicted to have a quality score of 5.1.

## 3.2. Neural Network

Garson’s method depicts the most important variables in explaining quality are ‘CA’,’Alc’ and ‘TSD’. The Olden’s method demonstrates a slightly different picture, with ‘Alc’,’FA’ being the most important variables. Using Olden’s method, ‘CA’ is considered one of the least important variables and likely there are positive and negative weights associated with this variable.

Lek’s profile with 2 clusters was performed and demonstrates the impact wine properties has on the final quality score. Higher levels of alcohol has an increasing impact on quality score, across the middle and upper ranges. There’s an inverse relationship between the level of ‘TSD’, ‘VA’ and ‘pH’ on quality score. As these properties increase, the quality score decreases. The sharpest quality score reductions occurring with ‘pH’ and ‘VA’ variables.

## 3.3. K-Means Clustering

The k-means clustering k=7 (appendix) indicates that those clusters with high quality scores (1, 5 and 6) tend to be high in alcohol. These three clusters also seem to indicate that lower ‘TSD’ and ‘FSD’ levels tend to produce better quality scores. Interestingly, cluster 7 has one of the lowest ‘FSD’ and ‘TSD’ levels but is also very low on the quality score. It may indicate that this property in conjunction with a variable like alcohol might help determine quality score.

K-means with k=2 (appendix) provides a clearer determination of drivers of quality of wine. It appears wines higher in alcohol, sulphates, ‘Ch’, ‘RS’, ‘CA’ and ‘FA’ have a positive impact, whilst ‘VA’, ‘FSD’, ‘TSD’, and ‘pH’ have a negative impact.

# 4. Comparison to non-machine learning methods

The non-linear methods determined that ‘FSD’, ’Sulphates’ and ‘Alcohol’ had positive effects on red wine with ‘VA’, ’Ch’, ’TSD’, ’pH’ all having negative effects on the quality of the wine.

To an extent, analysis using machine learning methods indicated similar findings. Alcohol was the clearest driver on wine quality, with regression tree, neural networks and clustering models all indicating its importance. The other important variables then varied, depending on which method was performed. ‘TSD’ and ‘VA’ appeared to be other common variables in models with a negative impact on wine quality.

Overall, none of the machine learning methods did not predict particularly well when compared to the non-machine learning methods when comparing across loss functions including the winemaker’s loss function (APPENDIX).

Some machine learning methods were unlikely to ever predict better due to the nature of their models. For example, the regression tree has an inability to predict wine scores at a more granular level, which is evident by the actual quality scores featured in the dataset spanning from 3 to 8. This demonstrates that although predictive insights can be easily understood using this method, the segmentation of quality scores is too holistic.

The k-means clustering also suffered a similar problem in that predictive scores are limited to the k selected. Utilizing the clusters as a pre-filter, the regression improved the loss function metrics but not good enough to trump any of the non-machine learning methods.

The neural network with 1 layer and 4 nodes proved to provide the most consistent predictions and would be considered to be the best machine learning method in attempting to minimize all the loss functions.

The reasonings for the machine learning methods performing worse than the non-machine learning methods may be due to a lack of data, overfitting and, for certain models, an oversimplification. It may also be due to the fact that these methods may not be well suited to this particular problem and would fare better when trying to classify.

## 3.3 Winemakers loss function

Winemaker loss function was developed based on the below assumed pricing model, it is assumed price of wine increases significantly when the quality score is 7 or above. It is also assumed that the extra cost on marketing a high-quality wine is approximately $2 per bottle. It is evident from these assumptions that the loss of revenue due to underestimation significantly outweighs any expenditure loss due to marketing, therefore overestimation is preferred rather than underestimation for high quality wines.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Wine Quality Score** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Wine Price** | $ 5-7 | $ 5-7 | $ 7-8 | $ 7-8 | $ 8-10 | $ 8-10 | $ 20 | $ 30 | $ 40 | $ 50 |

Business use cases were used to develop below matrix to allocate over and under estimation factors to different wine quality scores based on the difference between predicted and actual quality score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0 - 22%** | **22% - 29%** | **29% - 36%** | **> 36%** |
| **Underestimate factor if actual > 7** | 8 | 18 | 28 | 38 |
| **Overestimate factor if actual < 7** | 2 | 4 | 4 | 4 |
| **Underestimate factor if actual < 7** | 1 | 1 | 1 | 1 |
| **Overestimate factor if actual > 7** | 1 | 1 | 1 | 1 |



# 4. Appendix

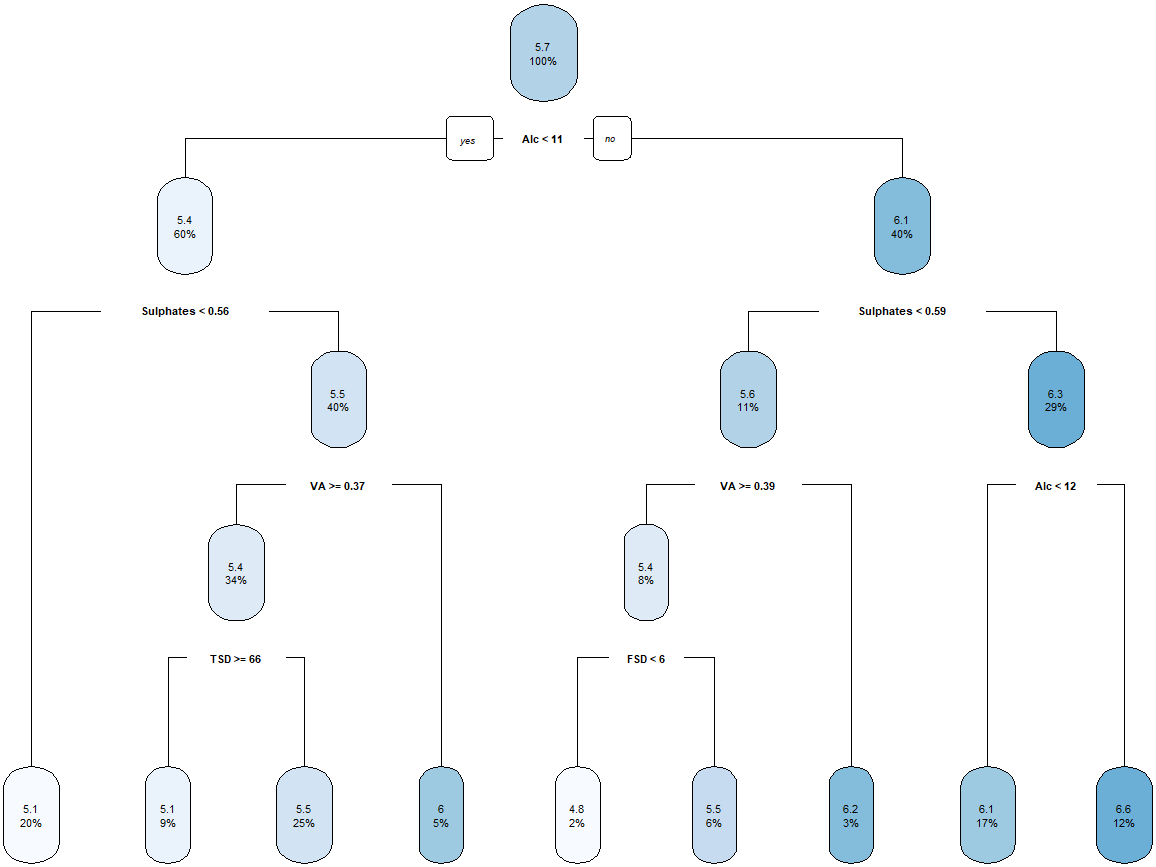
K-means=7



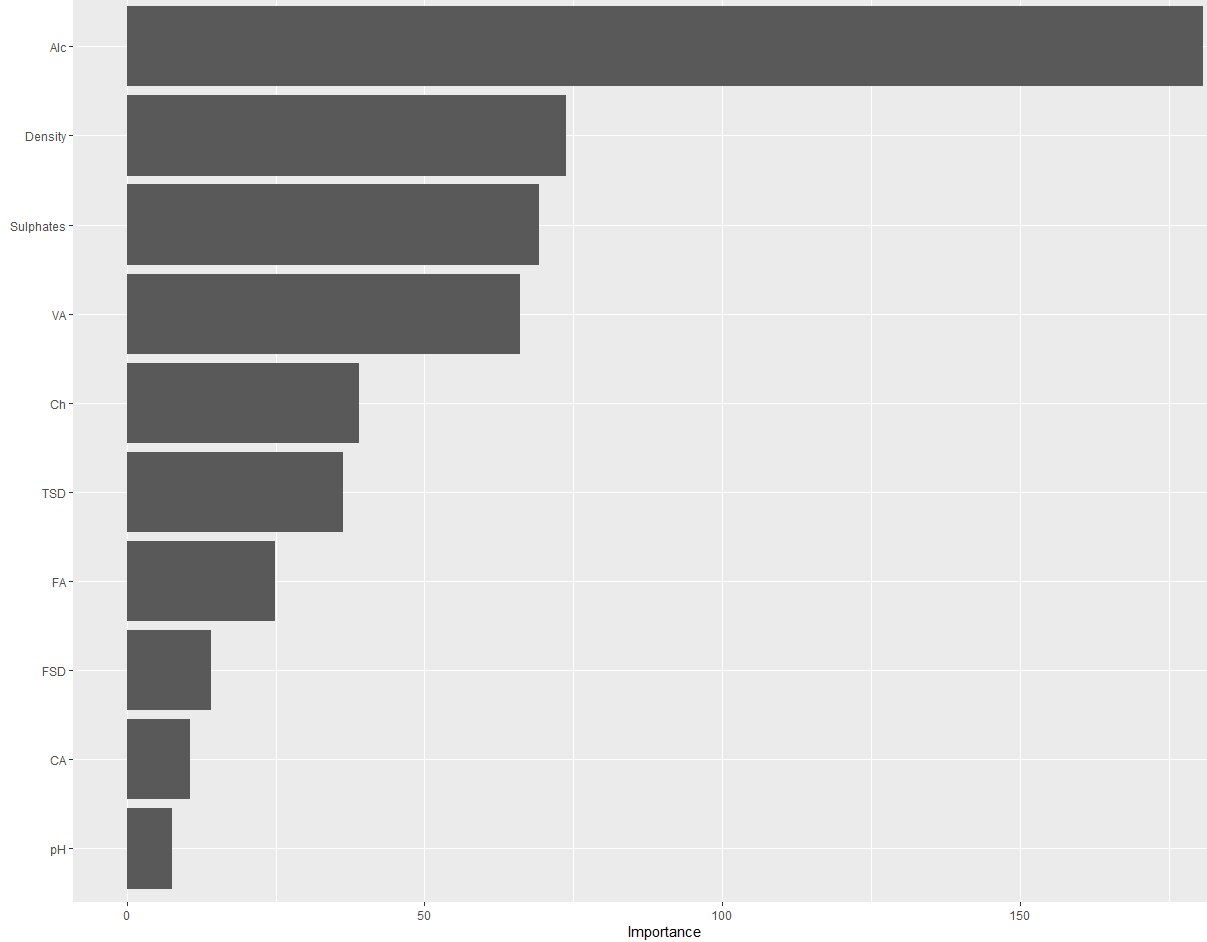
k-means=2



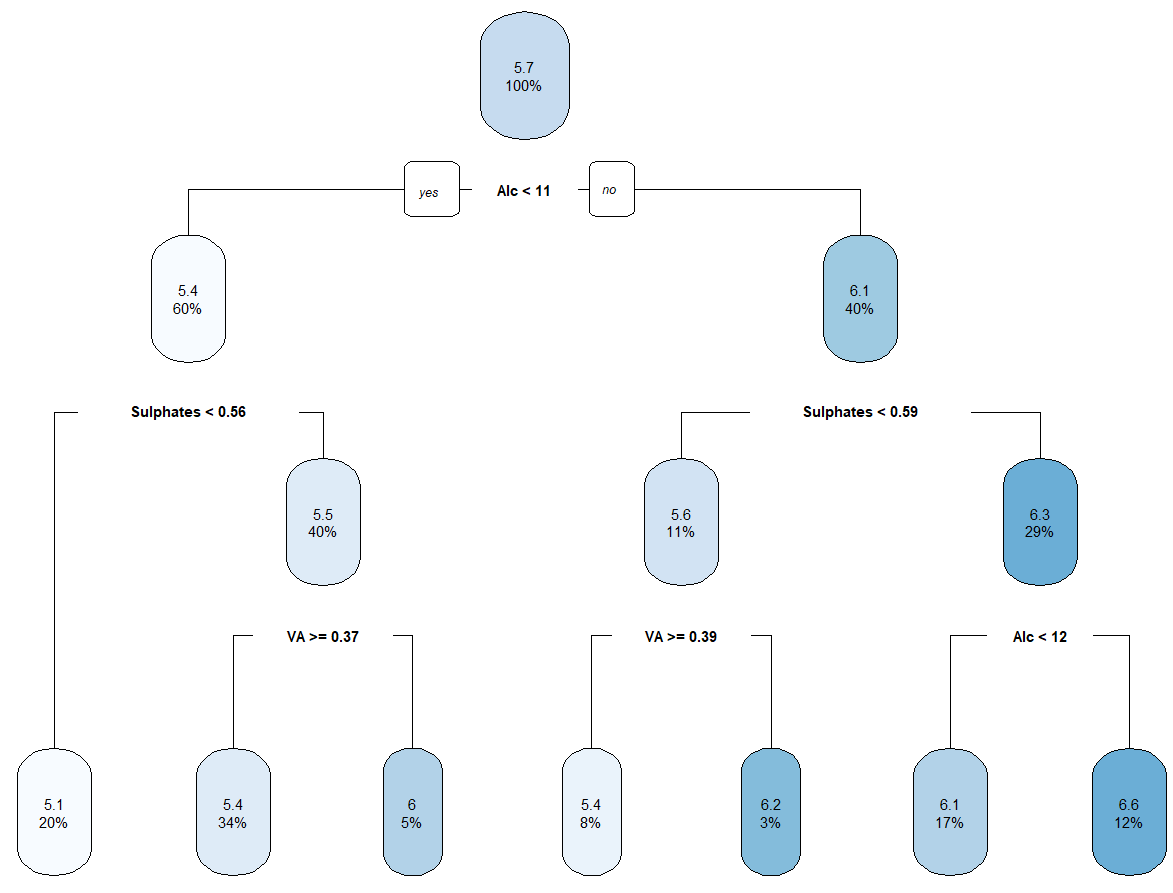
Regression Tree



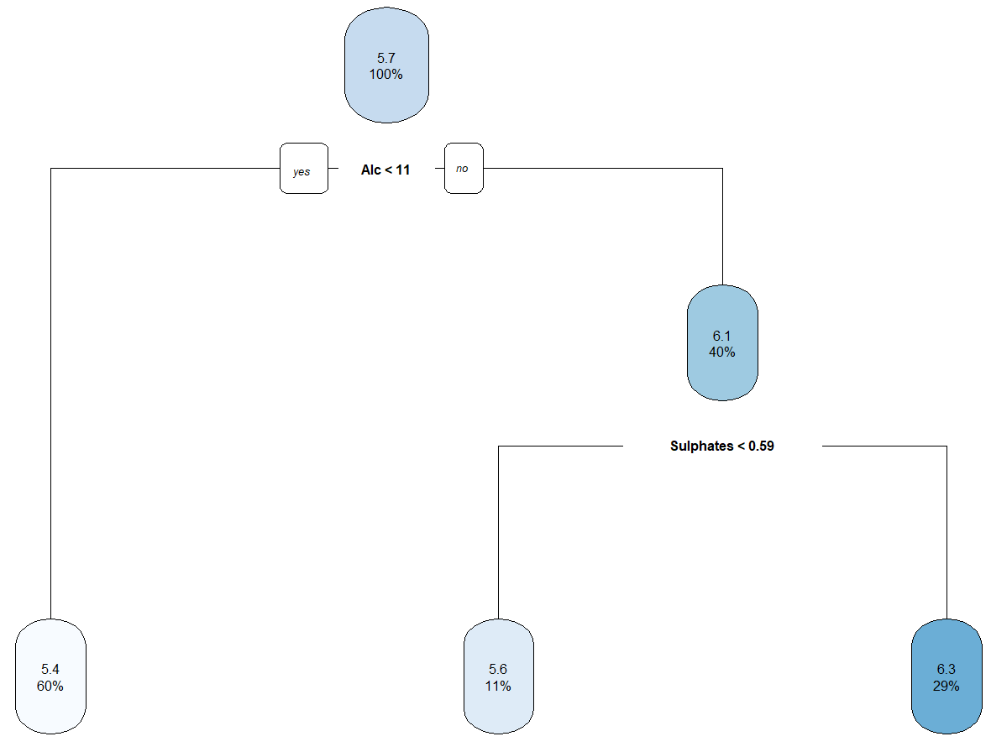
VIP Regression Tree



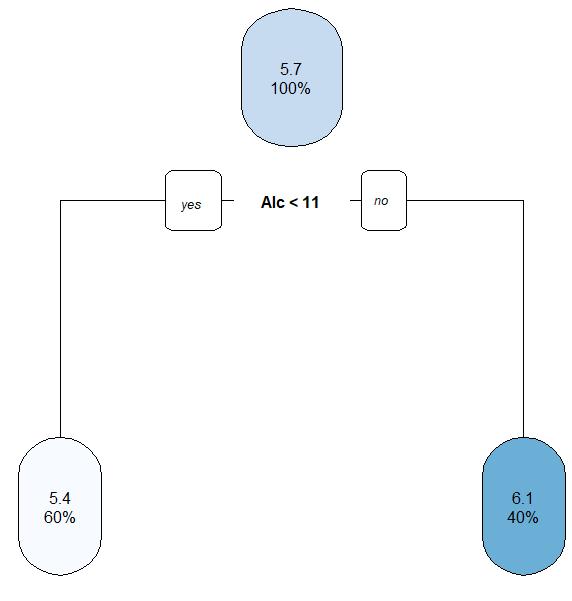
Pruned Tree CP – 0.02



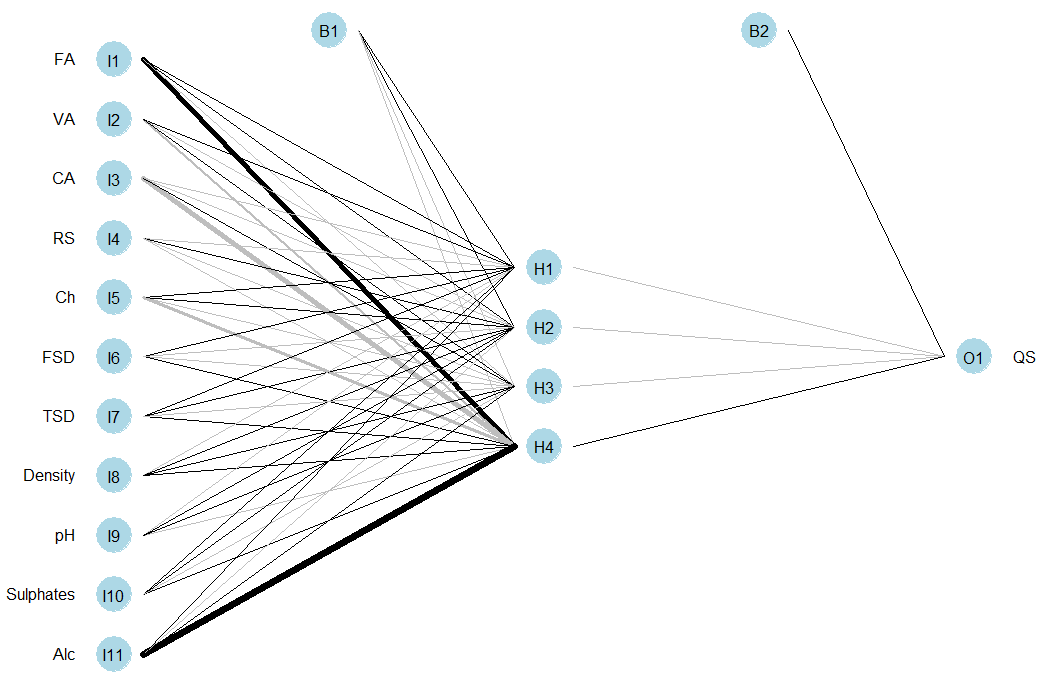
Pruned Tree CP-0.05



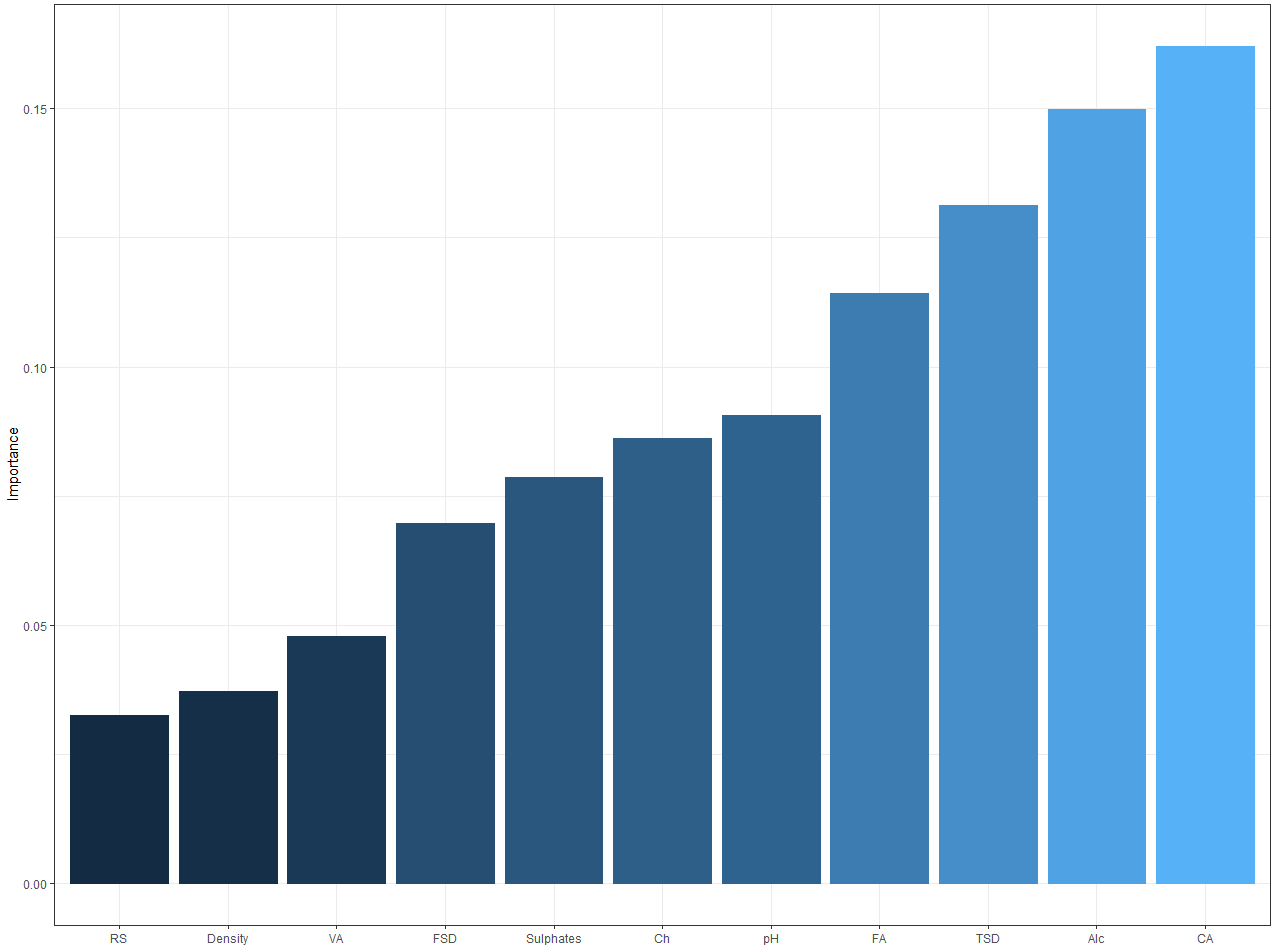
Pruned Tree CP – 0.1



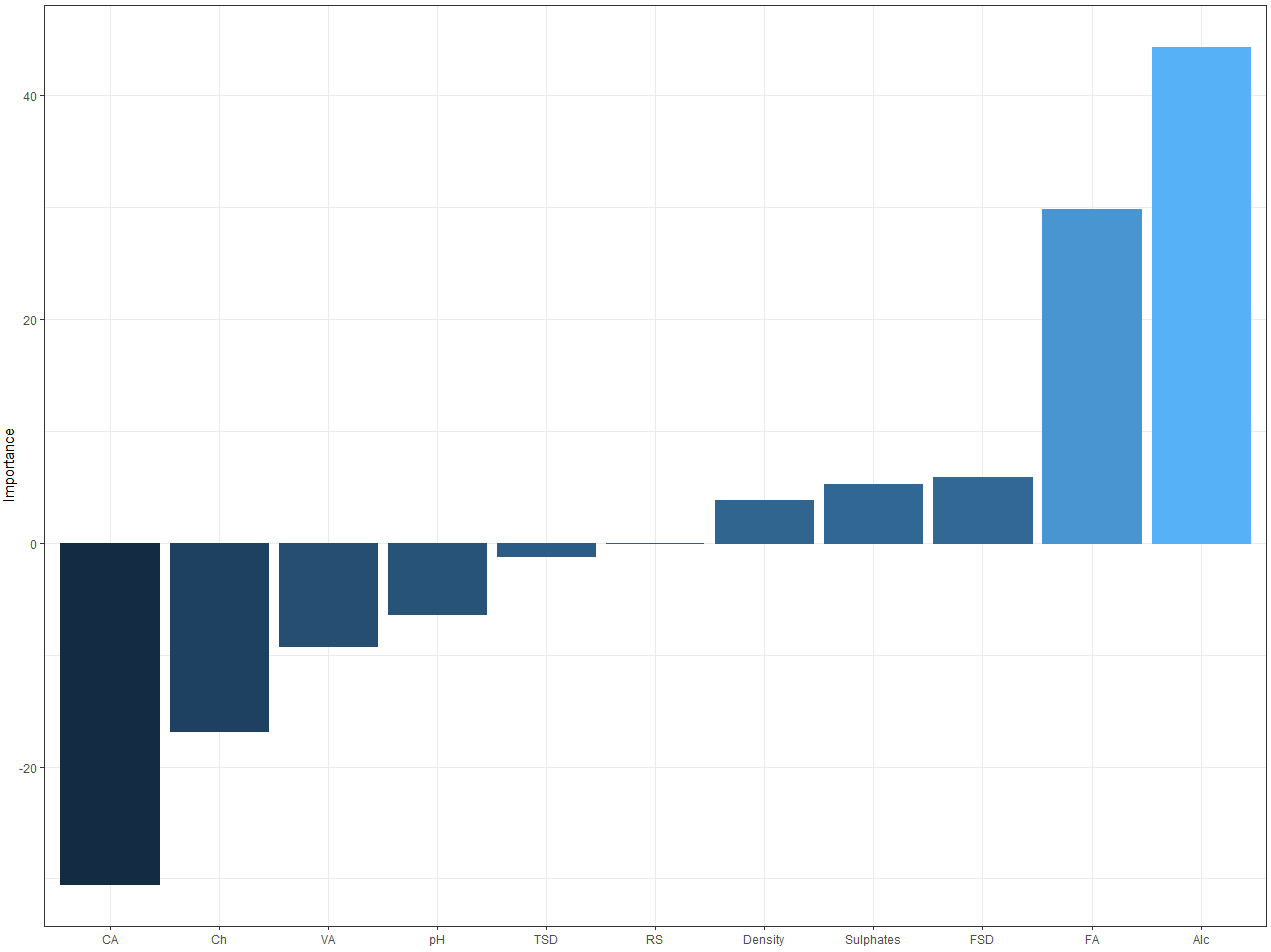
Neural Net (seed123)



Garson’s Importance (NN)



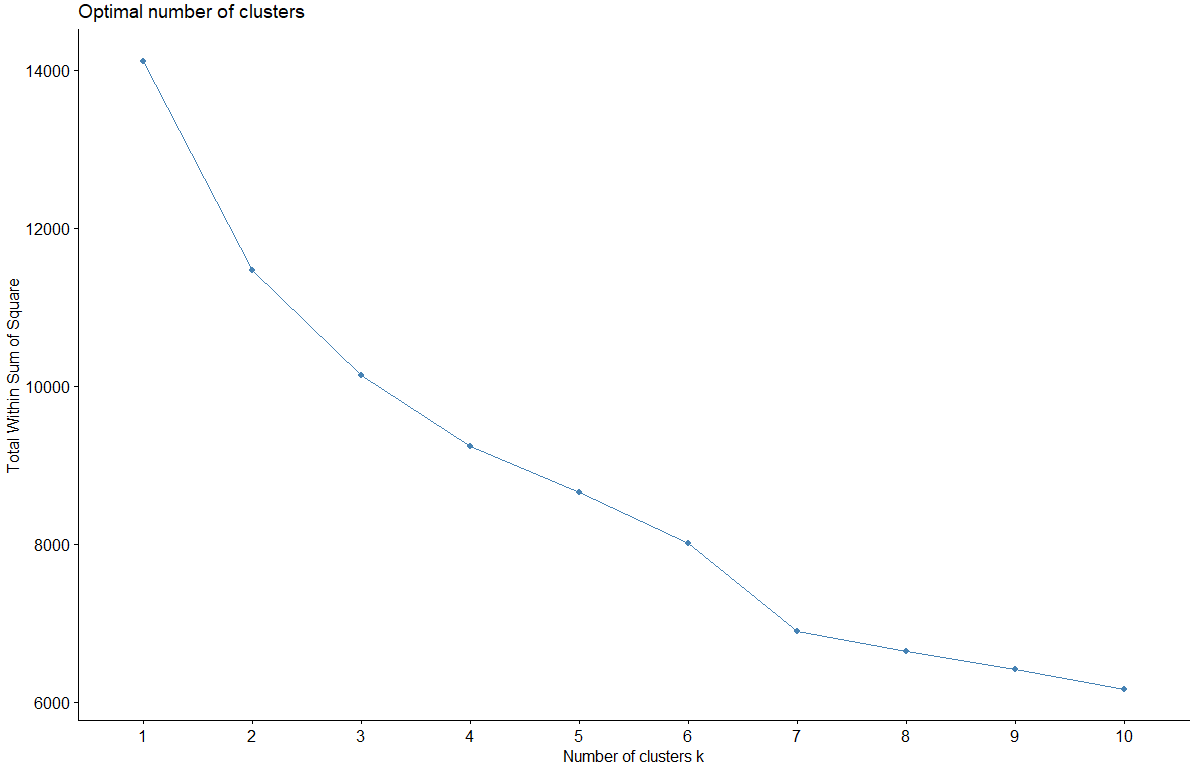
Olden’s Importance (NN)



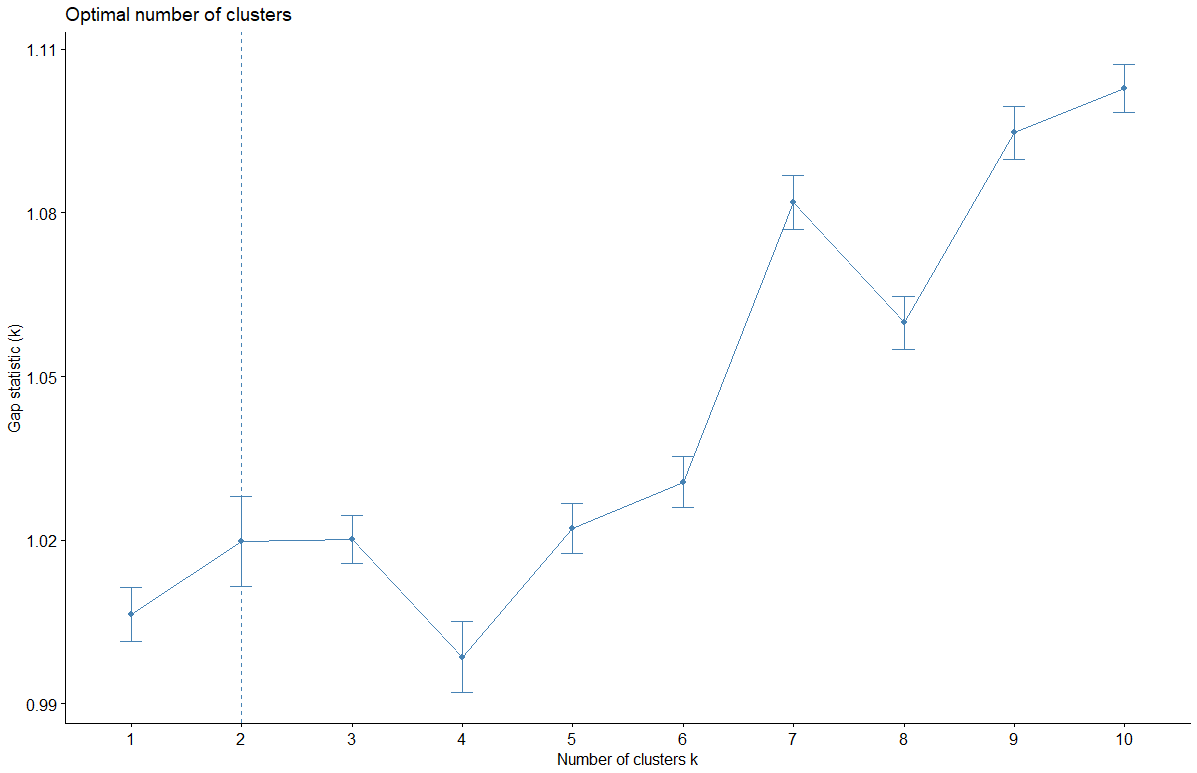
Lek’s Profile



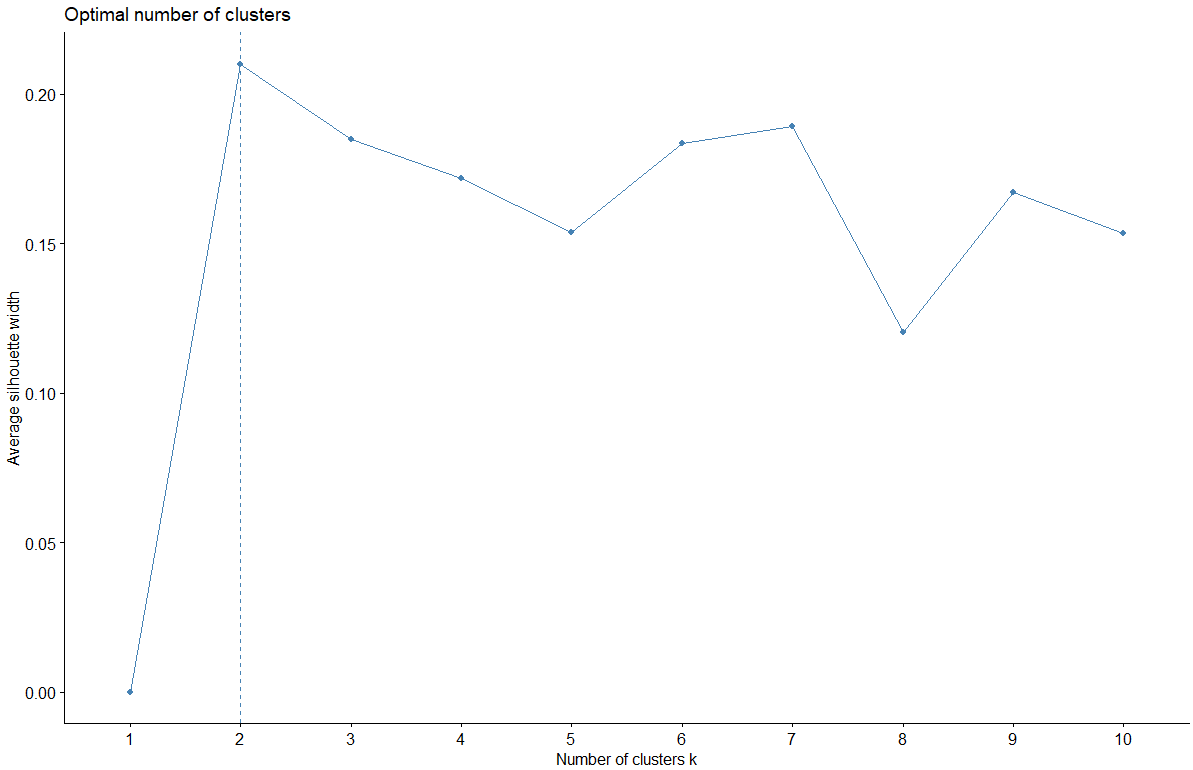
ELBOW Method K-means



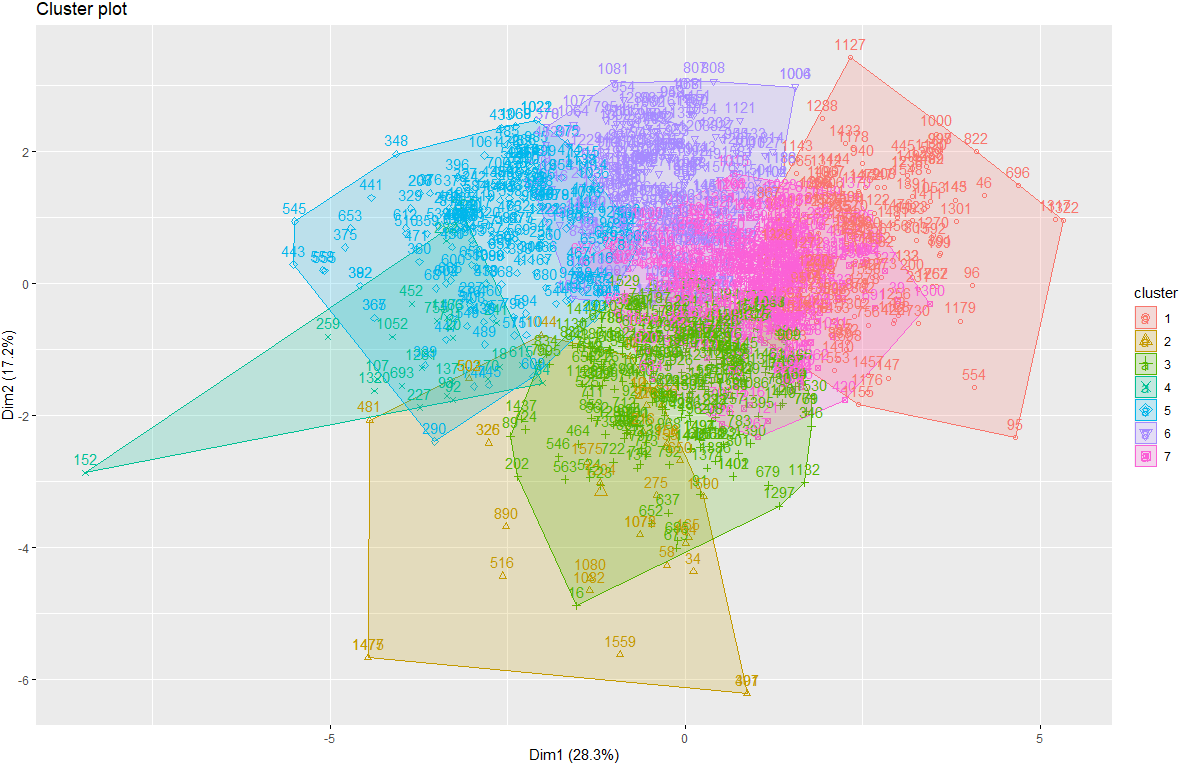
Gap Method K-means



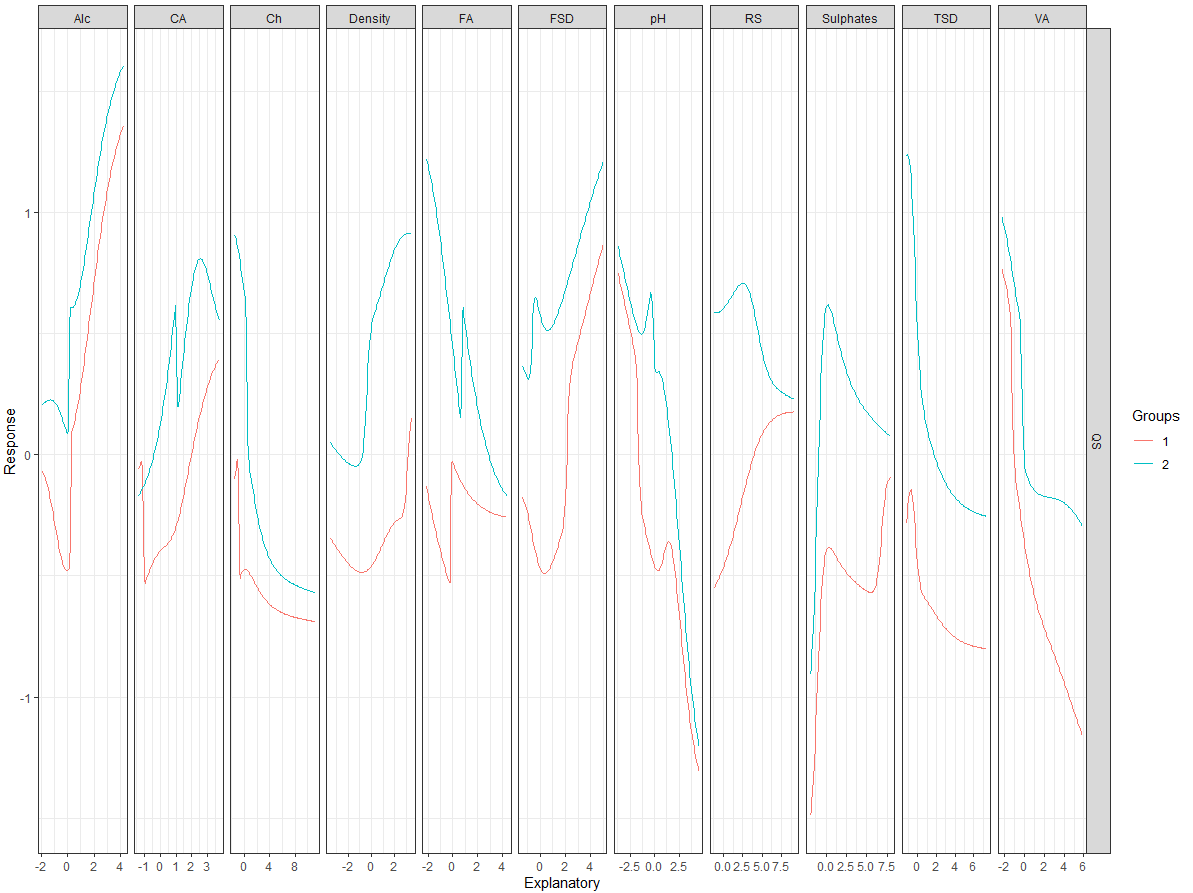
Silhoutte Method K-means



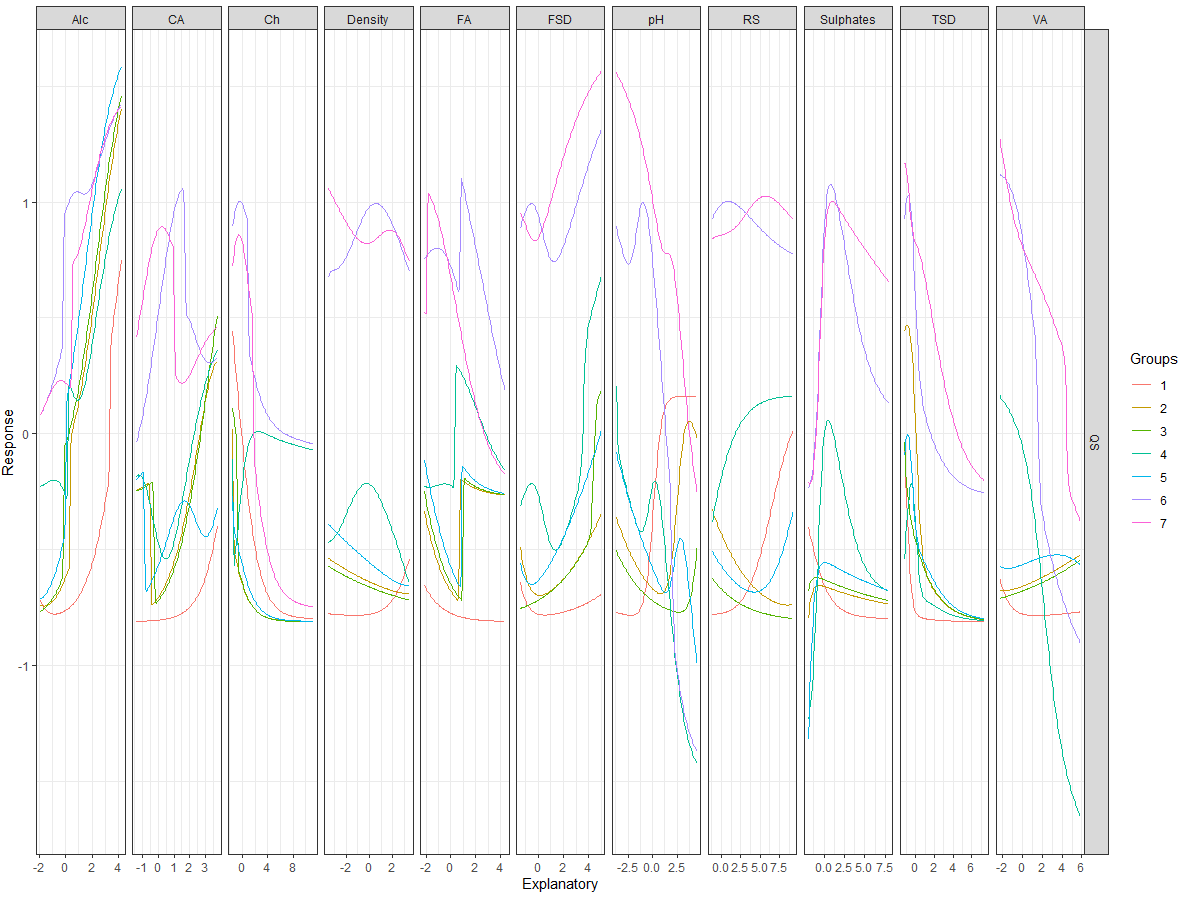
Cluster Plot (can we do more than 2 dimensons? 3d version?)



LEK’s cluster=2



Lek’s cluster=7



Loss function

|  |
| --- |
| wine\_maker\_loss\_function = function(error,actual){  relative=100\*(error/actual)  lossDF=as.data.frame(t(rbind(relative,actual)))    highQualityOverestimate =lossDF[lossDF$actual>=7 & lossDF$relative <= 0, 1]  highQualityUnderestimateWithin22 =lossDF[lossDF$actual>=7 & lossDF$relative > 0 & lossDF$relative <= 22, 1]  highQualityUnderestimateWithin22to29 =lossDF[lossDF$actual>=7 & lossDF$relative > 22 & lossDF$relative <= 29, 1]  highQualityUnderestimateWithin29to36 =lossDF[lossDF$actual>=7 & lossDF$relative > 29 & lossDF$relative <= 36, 1]  highQualityUnderestimateMoreThan36 =lossDF[lossDF$actual>=7 & lossDF$relative > 36, 1]      lowQualityUnderestimate =lossDF[lossDF$actual<7 & lossDF$relative>=0, 1]  lowQualityOverestimateWithin22 =lossDF[lossDF$actual<7 & lossDF$relative < 0 & lossDF$relative >= -22, 1]  lowQualityOverestimateWithin22to29 =lossDF[lossDF$actual<7 & lossDF$relative < -22 & lossDF$relative >= -29, 1]  lowQualityOverestimateWithin29to36 =lossDF[lossDF$actual<7 & lossDF$relative < -29 & lossDF$relative >= -36, 1]  lowQualityOverestimateMoreThan36 =lossDF[lossDF$actual<7 & lossDF$relative < -36, 1]    loss= sum(1\*abs(highQualityOverestimate)) +  sum(8\*abs(highQualityUnderestimateWithin22)) +  sum(18\*abs(highQualityUnderestimateWithin22to29)) +  sum(28\*abs(highQualityUnderestimateWithin29to36)) +  sum(38\*abs(highQualityUnderestimateMoreThan36)) +  sum(1\*abs(lowQualityUnderestimate)) +  sum(8\*abs(lowQualityOverestimateWithin22)) +  sum(18\*abs(lowQualityOverestimateWithin22to29)) +  sum(28\*abs(lowQualityOverestimateWithin29to36)) +  sum(38\*abs(lowQualityOverestimateMoreThan36))    return(loss/length(error))  } |

An example of business use case: -

Assume a wine has been predicted to be of the quality score of 7, winemaker will price the wine at $20 per bottle and spend $2 on the marketing expenses, expecting an additional profit of $8.

* If actual score is less than 7, then winemaker will lose additional $2 spent on marketing and will have to sell the wine at marked down price between $8 to $10 depending on the degree of error in estimation.
* If actual score is 7 or higher winemaker will either, make $8 or will have revenue loss based on the degree of error estimation. If error is up to 22%, there is no loss, but if error increases beyond 22% loss of revenue up till 29% will be $18, as the actual wine score will be 8 and can be sold at $30. Similarly, if the error is between 29% and 36% the actual wine score in that case is 9 and can be sold at $40 and hence the loss of revenue will be $28. In case the error is over 36% the actual wine score will be 10 with revenue loss of $38 as the wine can be sold for $50.
* In case of underestimating when actual and predicted are less than 7 the loss is will be incurred due to incorrect pricing. Similarly, in case where actual and predicted are both overestimated and are above the score of 7 loss will be incurred due to lack of cost optimization.