Predicting Wine Quality Through Machine Learning

By: Arthur Setiawan & Jeffrey Ochavillo

Abstract:

Wine in the United States is a way of life, and many Americans almost always pair their dinners with a glass of wine or two. Red wine specifically when consumed in moderation has multiple health benefits ranging from extra antioxidants, increased heart health, blood pressure regulation, and diabetes risk reduction. Data scientists are interested in how they can maximize the quality of their wine consumption to enjoy the occassional alcoholic beverage, while also taking advantage of the potential health benefits that come with drinking wine. This dataset specifically focuses on a red wine variant, and unfortunately excludes some essential information such as grape type, brand, and selling price which hides why some wines may be higher quality than others. Unsupervised learning methods such as Principal Component Analysis, K-Means Clustering, and Hierarchical Clustering were performed on the dataset to understand the underlying structure of the dataset, prior to fitting a neural network model onto the wine dataset. The convolutional neural network that was specifically designed to predict wine quality based on physicochemical test predictor variables when fit with hyperparameters with 40 epochs, batch size of 64, and dropout rate of 0.4 yields a test accuracy of 53.6% and validation loss of 0.63 when the model is fit with an adam optimizer.

Introduction:

Wine plays a very important role in the day-to-day lives of Americans and Europeans specifically, and is a daily diet for most. Based on a 2021 study, the average American adult consumes 3.18 gallons of wine per year, whereas the average Portuguese adult consumes 13.7 gallons of wine per year! European countries also account for 48% of the world's wine consumption, which seems like an extremely high number considering that the most densely populated countries are in fact located in Asia. The goal of this report is to maximize the quality of red wine that an individual can drink, especially focused on what specific physicochemical variables better predict or better correlate to a higher quality red wine.

This dataset was extracted from a Kaggle practice dataset referencing the UC Irvine open source dataset, and strictly focuses on red wine of the Portuguese 'Vinho Verde' variant. This dataset has omitted any brand name labels, grape types, and most importantly price which are all highly essential factors to understanding why some wines are categorized to having much higher quality than others. This wine dataset is also more heavily skewed towards wines of generic or medium quality, since extremely high quality wines are few and far between. In an attempt to cluster wine quality groups, it has been split into three groups, bad quality groups have a quality rating of less than 5, medium quality groups have a quality rating of between 5 and 6.5, and good quality groups have a quality rating higher than 6.5.

Theoretical Background:

Principal Component Analysis (PCA)

One of the first approaches made to the wine dataset was to look at its quality was Principal Component Analysis. This method is a machine learning technique that aims to transform a high-dimensional dataset into a lower-dimensional representation while keeping as much of the original integrity of the data as possible.

The important idea for PCA is principal components. Principal components are new sets of variables found which are linear combinations of the original variables. The order of these components goes from the first principal component captures the maximum amount of variance in the data, followed by the second component gets the second highest variance, and continues from there.

The processes for PCA are: standardizing the data, computing the covariance of the matrix, compute the eigenvectors and eigenvalues, selecting the principal components, then transforming the data. In PCA the data needs to be standardized, in other words it needs to be rescaled to have a mean of zero and unit variance. This ensures that variables with different measures do not overwhelm the whole dataset for analysis. Computing the variance allows the model to identify the relationship between all the variables. It shows the strength of the relationship of all the variables onto the variable to predict. In this case further elaborating upon the relationship strength of all the variables against the quality of the wine. The next steps include the computation for the covariance matrix is the calculation for the eigenvectors and eigenvalues which is done inside the algorithm of PCA. So within the R software, the function prcomp() is useful because it calculates all the eigenvectors and eigenvalues that is needed for the covariance matrix to capture the first and second principal components.

But following all the steps from above, after running the PCA algorithm against the selected wine dataset, an important step is to calculate the Proportion Variance Explained (PVE) by each principal component in a PCA model. By calculating the PVE, what is determined is how much of the original variance is captured by each principal component. This is useful in analysis to determine the number of principal components to retain. Overall, it helps in deciding the appropriate dimensionality reduction and understanding the information content retained in the reduced wine dataset.

K-Means Clustering

The second technique applied to this dataset was K-Means Clustering. This is another unsupervised learning technique that takes advantage of partitioning the dataset into groups or clusters based on similarity. The goal is to get a number of centroids and assign a data point to the nearest centroid based on a distance metric, usually Euclidean distance.

The process for K-Means Clustering is: specifying the number of clusters, initializing the centroids, assigning the data points to those clusters, updating the centroids, iterating through the previous steps, then finally finalizing the clusters. For the dataset, the number of clusters set was 3 based on the distinction of quality of wine: bad, medium, and good. This is all accomplished by selecting all the columns in the wine data set except for “quality” and “quality group” since they are not used for the clustering process. Then 20 was selected for the amount of times for the algorithm to run different initializations for the centroid, and the best clustering result will be chosen based on the lowest-within-cluster variance.

Another step is taken which is calculating the Singular Value Decomposition (SVD) on the winescale dataset. SVD is a matrix factorization technique that decomposes the matrix into three matrices. But in the case of the experiment, it was used to reinforce the findings initially found, which came about the same.

Convolutional Neural Networks (CNN)

CNNs are a class of deep learning models signed for processing grid-like data, examples are images or spectrograms in this case. The foundations lie in the utility of capturing spatial hierarchies of features and learning meaningful representations directly from raw input data.

A core characteristic or feature of CNNs are convolutional layers. The purpose of these layers is to perform local receptive field operations by convolving learned filters or kernels with input data. By doing this, the network can extract patterns and features while preserving spatial relationships. The filters are then applied throughout the input so the pattern detection is applied in different regions or parts of the input. By stacking the layer, CNNs can learn more complex and abstract representations.

After the going through convolving, pooling layers are applied, either max or average pooling to downsize the feature maps, reducing spatial dimensions while retaining the more valuable and interpretable features. By downsampling, it helps the neural network be invariant to spatial translations and reduces the computational burden by minimizing the number of parameters in subsequent layers.

A big contributor to the neural networks are the functions applied. Common activation functions applied were Rectified Linear Unit (“relu”). ReLU introduced nonlinearity by setting negative values to 0. By applying activation functions allows nonlinear variables to be inputted and increases the models’ capabilities to learn more complex patterns and representations.

Afterwards, connected layers of the CNNs are working together after convolving and pooling of the layers. What these connected layers do is they connect every node, input, or neuron to every or node, input, or neuron in the next layer. This allows the network to learn high-level abstractions and formations based on previously given or extracted features. The output layer of the CNN utilizes an activation function, example would be a softman, to produce probability distribution over the classes for classification tasks.

When training the model, backpropagation is applied. This is when gradients are computed and used to update the network’s weights and biases between the predicted outputs and labels. This is used mostly on categorical cross-entropy to minimize the defined loss function. In the model used, optimization algorithms like the Adam optimizer were applied to update the parameters and find the optimal configuration.

The reason for going through deep learning methods is mostly because of the versatility of learning from complex patterns and different relationships in the data. It would be thought to do really well in making predictions for the quality of wine in the dataset.

Methodology:

Data Preparation and Exploration

The first step taken to clean the data was renaming variables with underscores instead of spaces such that it is easily understood and referable throughout the code without any issues. Due to the nature of the wine industry and the dataset, rating a wine to have a high quality greater than 6.5 is rare, meaning that most wines fall into the medium quality rating category. Having the dataset being more representative of average or regular wines rather than great or poor quality wines is more representative of a real world scenario rather than balancing out observation quantities for each quality group. The dataset was fairly clean, and had no missing data which attributed to the lack of data preparation needed to make this dataset work well for the purposes of the research.

Exploratory data analysis was performed to understand relationships between important variables that may be important and correlate to the response variable. From the figure 1, it is apparent that higher alcohol content is synonymous to better wine quality. Outliers also exist in regular quality wines having high alcohol contents especially considering many wines are of the medium or regular quality rating.

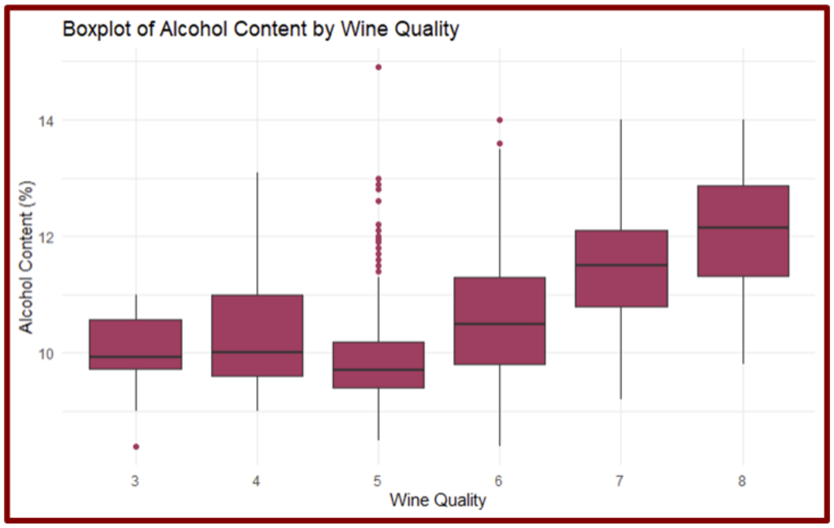


Figure 1 - Boxplot of Alcohol Content by Wine Quality

Best Subset Selection

The second step for data processing is through best subset selection, where results yielded that the best performing model was one with 5 predictors instead of the full 12, and having a test MSE of 0.34. The best subset model includes the following predictors: volatile acidity, chlorides, total sulfur dioxides, sulfates, and alcohol. As a result, the worst mean squared error was a model with only 1 predictor variable, having an error of 0.43. This means that some of the predictor variables, most likely volatile acidity, total sulfur dioxides, and alcohol, are most likely to be strong predictors of wine quality since they exist in principal component 1 and principal component 2 as well. Based on this validation, of what variables will be best representative of the ideal model, the 5 predictors from best subset selection will be used to compute model accuracies and validation losses.

Computational Results:

PCA

When implementing PCA on our dataset, our first principal component represents the direction in data which captures maximum amount of variance or explains the most significant portion of overall variance in the dataset. For principal component 1, the most important variables are fixed acidity, citric acid, pH, and density. The second principal component, on the other hand, captures additional variance in the data which is uncorrelated to the first principal component. Principal component 2 helps uncover underlying patterns or factors which contribute to data variation, and in the dataset's case the most important variables are total sulfur dioxide, free sulfur dioxide, and alcohol.

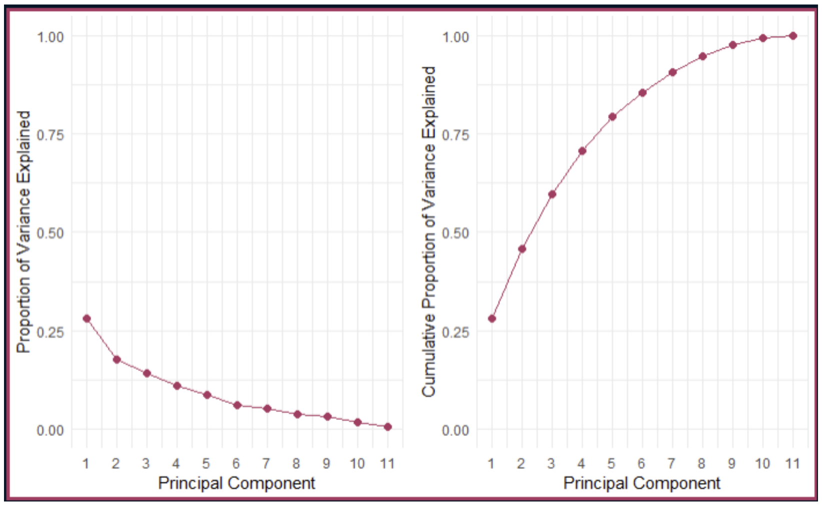


Figure 2 - Proportion of Variance Explained Plot

The smooth curve representing the proportion of variance explained (PVE) plot indicates an increase in cumulative PVE plateaus as more principal components are added. This shows that the extra principal components added to the model have less contributions compared to principal components that have been added prior. This steeper curve at the beginning shows the diminishing return aspect, where at the beginning only few principal components explain a large portion of variance, whereas the last few principal components are less contributing to the variance covered. It's important to note that the shape of the PVE plot is really just one interpretation to represent our dataset, it is advised that multiple iterations of exploratory data analysis is used to come to a more concrete conclusion.

K-Means Clustering

Clustering is initiated by choosing 3 clusters that correspond to the classification subgroups for wine quality rating: bad (3.5 to 5), medium (5 to 6.5), and good (> 6.5). When clustered by R IDE's k-means function with 20 iterations, as seen in figure 3 it will yield a result of a three split cluster, the middle cluster which represents the medium quality wines, the upper cluster which represents the good quality wines, and the lowest cluster which represents the bad quality wines. It can be concluded from this that best wines according to the automated algorithm have an imbalance level of medium acidity but high total sulfur dioxide, whereas for medium quality wines, it will have medium high total sulfur dioxide but also a high range of acidity, and similarly for the bad wines, it will have too low of sulfur dioxide but has an extremely high range of acidity. High acidity means that the wine is imbalanced in flavor, which is in fact true after reading some research papers on the science of wine. Surprisingly the model was able to correctly detect the underlying structure of the dataset, which is extremely exciting to see.

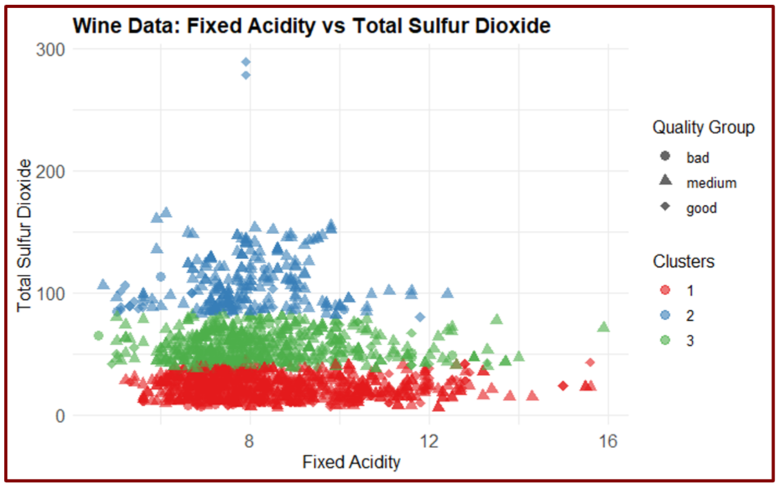


Figure 3 - kMeans Automated Cluster

Hierarchical Clustering

When applying hierarchical clustering to our dataset taking a sample of 20 different wines, it was concluded that scaled complete and average linkages performed the best due to its balanced distribution based on plots. From this, it can be interpreted that the clusters formed by these methods exhibit similar characteristics and capture the important patterns in the data. From plots it can be seen that the complete linkage at height 7 branches out into four clusters, and the average linkage at height 3 branches out into three clusters. Both are representative of the three different underlying wine quality groups that exist in the dataset that is provided. Although hierarchical clustering may not tell everything about the nature of the data, it does bolster the confidence of researchers to see that results are consistent throughout the analyses.

CNN

From the result of best subset selection, the most significant variables were volatile acidity, chlorides, total sulfur dioxide, sulfates, and alcohol to fit the CNN model. After splitting the wine data set into its training and testing portions, the model of the CNN was built. After experimenting with different model structures, the final model that was ultimately decided on had an epoch of 40, batch 64, and a dropout 0.4. More about this model is that is contained 128 units, an activation function of ReLU, two additional dense layers containing 64 units and 32, a dropout rate of 0.4 to prevent overfitting by reducing the reliance on specific neurons, and one more dense layer containing 1 unit and and linear activation function that will have the output be weighted on the sum of the input. Overall, this overall accuracy that was received was around 0.536 or roughly 54%. The performance of the neural network can be shown below in Figure 3.

In previous model structures, comparing epochs as high as 200 to 60, it was noticeable that increasing the number of epochs would lead the model to higher chances of overfitting. That led to the direction of straying away from really high epoch values, but also noticed that an epoch of 60, batch 32, dropout of 0.6 would fluctuate in accuracy from 30% to 50%. Due to that inconsistency, keeping the epoch to 40, batch of 64, dropout of 0.4 was the most stable result received for determining the quality of wine based on volatile acidity, chlorides, total sulfur dioxide, sulfates, and alcohol.

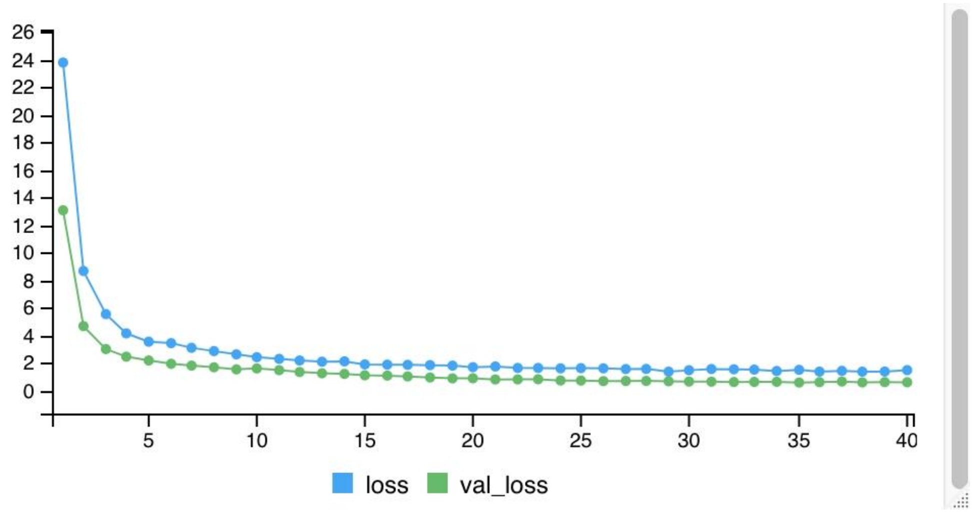


Figure 3 - Model Performance with Epochs 40

Discussion:

The “x” from PCA can be interpreted as: the X matrix represents data that has been transformed, where each column corresponds to a singular principal component. The “rotation” matrix from PCA is the linear combination of predictor variables that makes up a singular principal component.

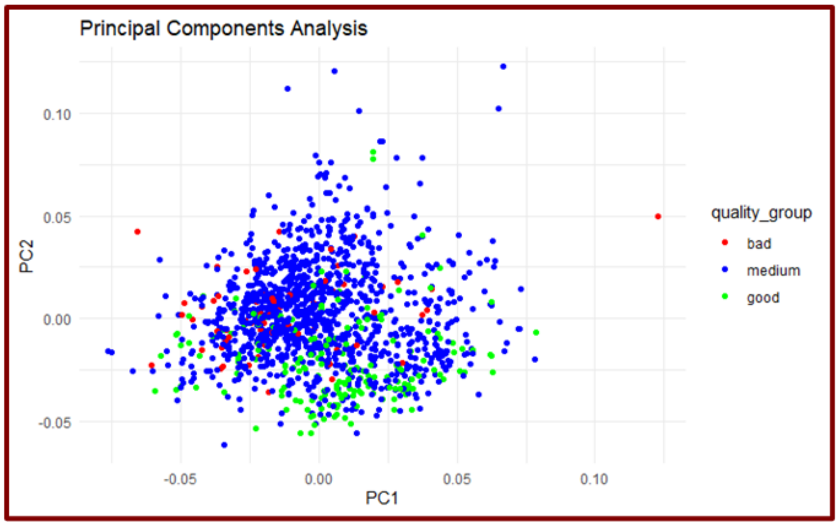


Figure 4 - PCA PC1 & PC2 Plot

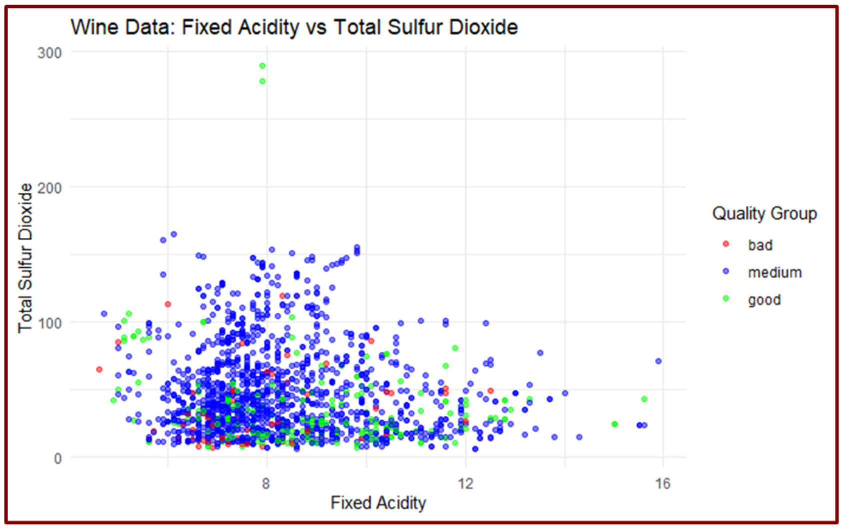
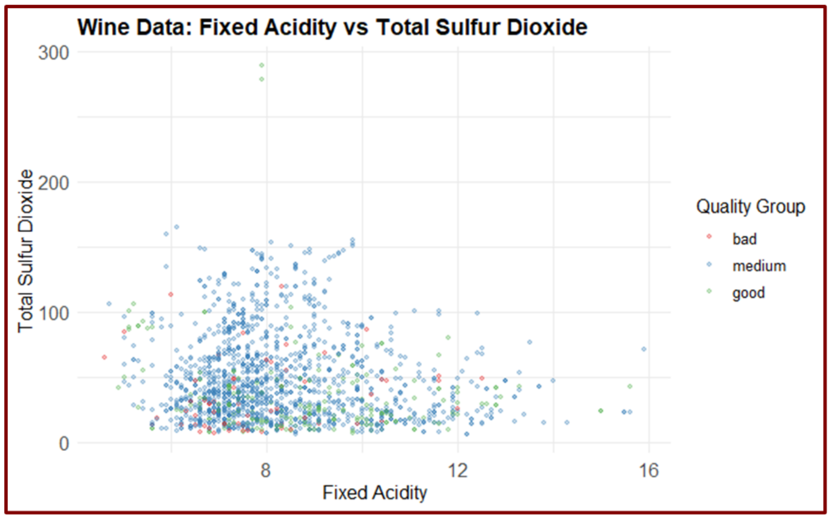


Figure 5 - TRUE Dataset, Fixed Acidity based on Total Sulfur Dioxide

In both figures 4 and 5, the inherent figure that is observed is the overarching theme of centered dense scattering in a focused location, having the most dense combination of all three quality groups. It does not differ much from plotting on two original features, which means that the principal component 1 and principal component 2 are great representations of the underlying data! After clustering, the observations belonging to the same cluster can be seen as the most highlighted points, whereas the opaque points stay the same. This results in a 34%

difference from true data. Clustering attempts to somewhat equally distribute quality groups more so than the true nature of the dataset. Based on these ranging plots, it can be interpreted that the groups truly live in each clustered space, where the medium dominates the center, good stretches to the medium left and bottom right quadrants of the graphs. And the bad quality group is centralized on the lower left quadrant, being densely covered by the sheer number of mediocre quality wines.

Figure 6 - Solid (Different) vs. Opaque (Same)



Unsupervised Learning vs. Supervised Learning

From comparing the results from the unsupervised methods of PCA and K-Means Clustering to the supervised method of CNN, it is very clear that they all have different purposes and means for solving the question of why some wines may be higher quality than others.

The purpose of PCA is to help identify the underlying structure, patterns, variables in the data. In this case of the dataset, the most important variables are total sulfur dioxide, free sulfur dioxide, and alcohol. For K-Means Clustering, it helps to identify the distinct groups of clusters within the data. It provides cluster labels for each data point, allowing for proper grouping and classification. After grouping the data to their distinct ratings of bad, medium, and good, which qualities of wine contained certain characteristics from acidity and sulfur dioxide. The biggest takeaway from this method was that it predicted or identified the structure of the dataset really well.

Now unlike the unsupervised learning methods, Neural Networks are designed to automatically learn hierarchical representations. The model learns the relationship between input features and output targets during the training process and produces predicted outputs baked on the learned weights and biases.

But overall, PCA focused more on reducing the dimensionality and capturing variance in the wine dataset, K-Means clustering succeeded in identifying cluster or groups within the data for the quality of wine and significant , and Neural Networks was useful in predicting wine quality based on physicochemical test predictor variables but also will vary in accuracy based on the structure of the model. All methods are useful and versatile but depending on what questions need to be answered, the methods will vary.

Conclusions:

Overall, it was interesting to see and analyze volatile acidity, chlorides, total sulfur dioxides, sulfates, and alcohol. Taking advantage of these powerful machine learning methods to make predictions on what factors contribute and differentiate between good to bad quality wine is amazing. Either clustering the variables, looking at the different principal components, or experimenting with various model structures in neural networks, the performance of the model surely is impressive. If time permitted, looking into alcohol percentage was another variable of interest. In the future, this analysis could further benefit from exploring more essential data points that have been omitted from the original dataset to maintain confidentiality in the wine market, namely wine brand, grape type, and bottle price. But from this report, it is clear that measuring the quality of various wines by their chemical factors is quite possible and this can truly be impactful for the broader community because these same methods and processes can be applied to other objects such as various beverages, foods, clothes, and more.

Bibliography/References:

US wine consumption. Wine Institute. (n.d.). https://wineinstitute.org/our-industry/statistics/us-wine-consumption/

Conway, J., & 15, M. (2023, May 15). *Countries with highest wine consumption per capita*. Statista. https://www.statista.com/statistics/232754/leading-20-countries-of-wine-consumption/#:~:text=People%20in%20Portugal%20consume%20more,consumption%20of%20nearly%2052%20liters.

Appendix:

Written_Homework4

Jeffrey Ochavillo and Arthur Setiawan

2023-05-28

Install Libraries

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.1     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(ISLR2)
library(RSpectra)
```

Read Dataset

```
library(readr)
wine_data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The question we are trying to answer is how we can maximize predicting the quality of red wine based on the different variables that exist, specifically different acidity, sugar levels, chlorides, density, pH, sulphates, alcohol content, and sulfur dioxide. These variables are highly attributed from the type of grapes that are being used, as well as the soil that the vineyards grow on.

Based on the Kaggle dataset, a quality of higher than 6.5 means it is a ‘good’ quality red wine

Note to self: Maybe we should split clusters to >= 6.5, 5 <= x < 6.5, 3 <= x < 5

```
selected_vars <- c("fixed acidity", "volatile acidity", "citric acid", "residual sugar",
                   "chlorides", "free sulfur dioxide", "total sulfur dioxide", "density",
                   "pH", "sulphates", "alcohol", "quality", "quality_group")

#Substitute space with underscore.
new_vars <- gsub(" ", "_", selected_vars)

#Redefine colnames for ease of use
colnames(wine_data) <- new_vars

#Keep base for debugging
wine_data_base <- wine_data

#Define quality group
wine_data$quality_group <- ifelse(wine_data$quality > 6.5, "good",
                                           ifelse(wine_data$quality >= 5 & wine_data$quality <
6.5, "medium", "bad"))

#Factor quality_group
wine_data$quality_group <- factor(wine_data$quality_group, levels = c("bad", "medium", "good"))

#Define Wine Data Selected, removing our response variable 'quality' and quality_group
wine_data_selected <- wine_data[, new_vars] %>% select(-c('quality','quality_group'))
```

Initial Look

```
summary(wine_data_selected)
```

```
##  fixed_acidity   volatile_acidity  citric_acid    residual_sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free_sulfur_dioxide total_sulfur_dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :3.310   Median :0.6200   Median :10.20
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90
```

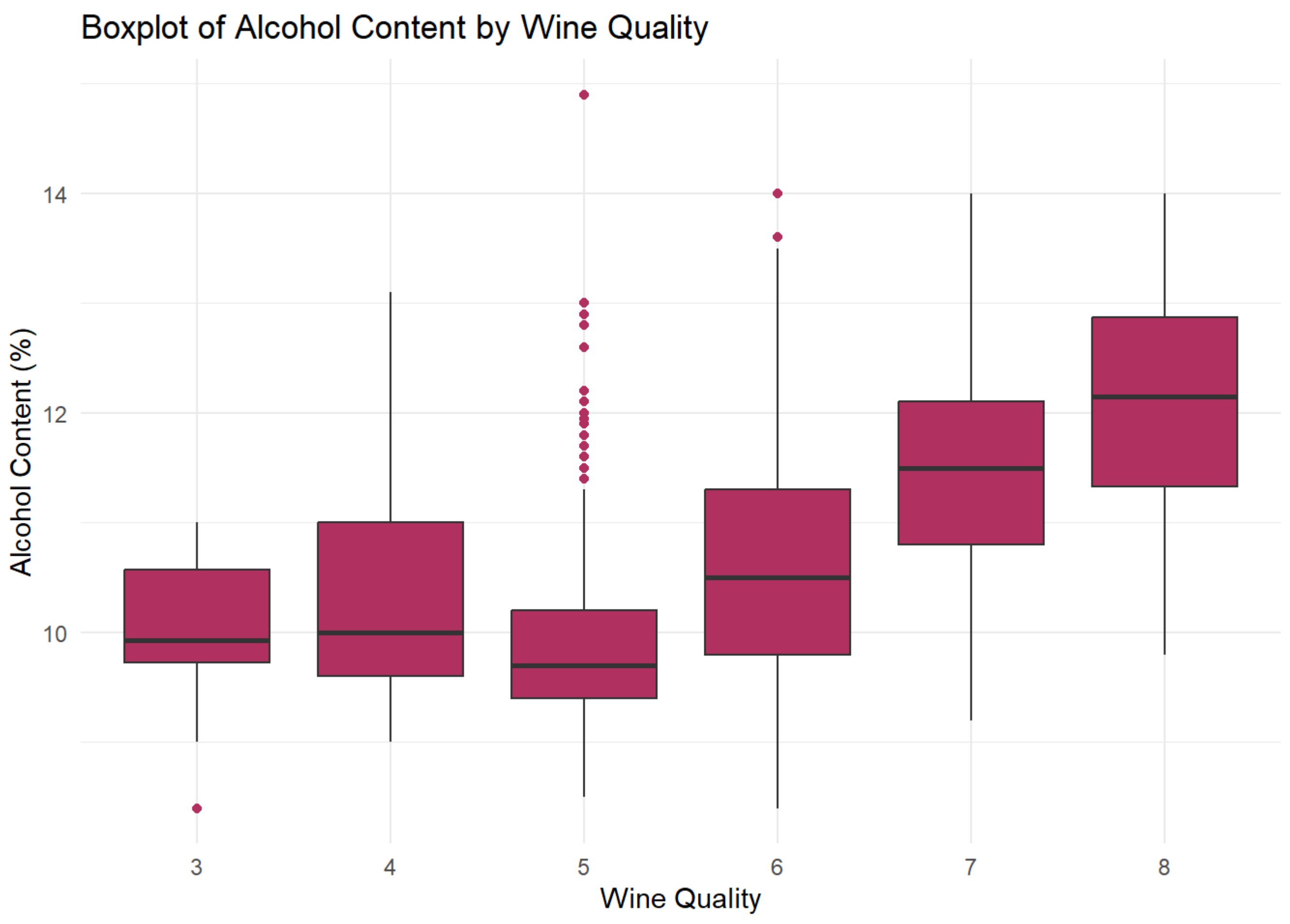
```
head(wine_data_selected)
```

```
## # A tibble: 6 × 11
##   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7.4             0.7         0               1.9     0.076
## 2           7.8             0.88        0               2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.7         0               1.9     0.076
## 6           7.4             0.66        0               1.8     0.075
## # ℹ 6 more variables: free_sulfur_dioxide <dbl>, total_sulfur_dioxide <dbl>,
## #   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>
```

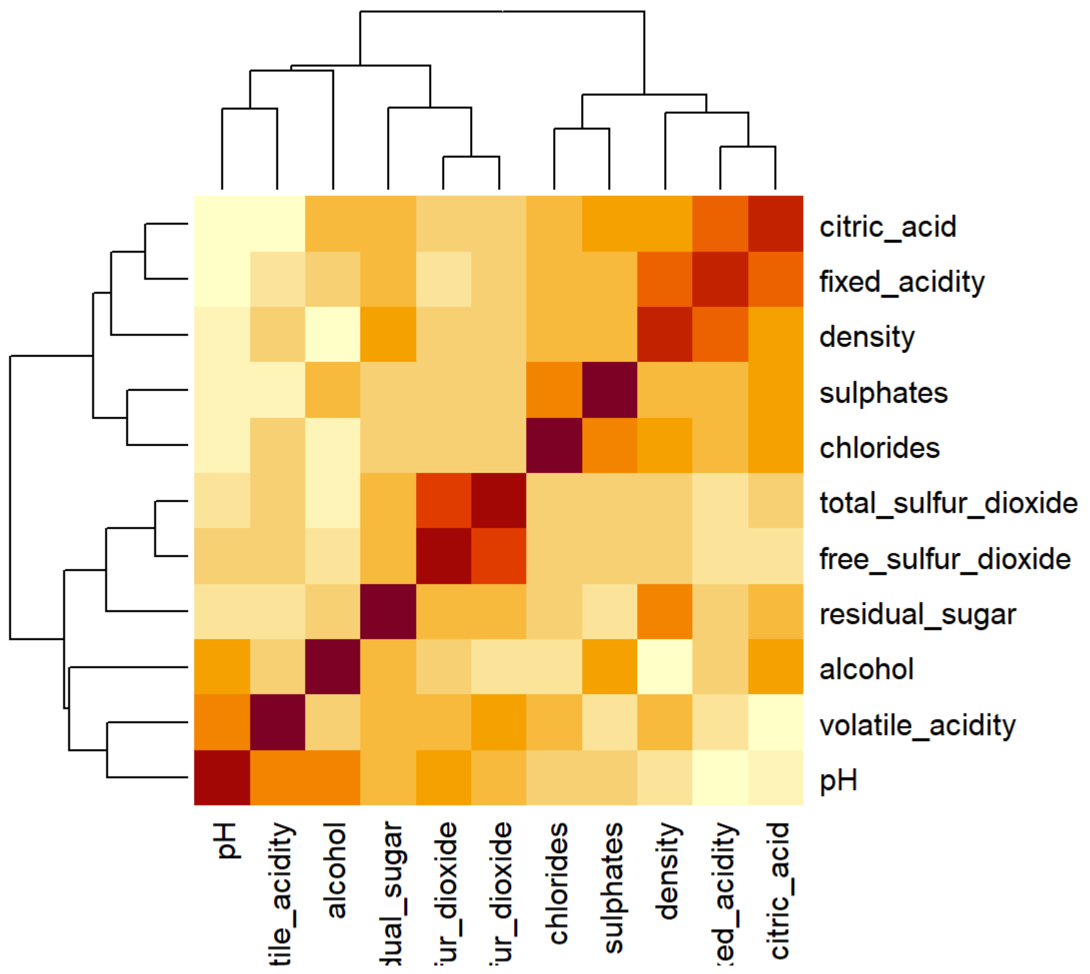
EDAs

Boxplot of Alcohol Content by Wine Quality

```
ggplot(wine_data, aes(x = factor(quality), y = alcohol)) +
  geom_boxplot(fill = "maroon", color = "#333333", outlier.color = "maroon", lwd = 0.5) +
  labs(x = "Wine Quality", y = "Alcohol Content (%)") +
  ggtitle("Boxplot of Alcohol Content by Wine Quality") +
  theme_minimal()
```



```
correlation_matrix <- cor(wine_data_selected)
heatmap(correlation_matrix)
```



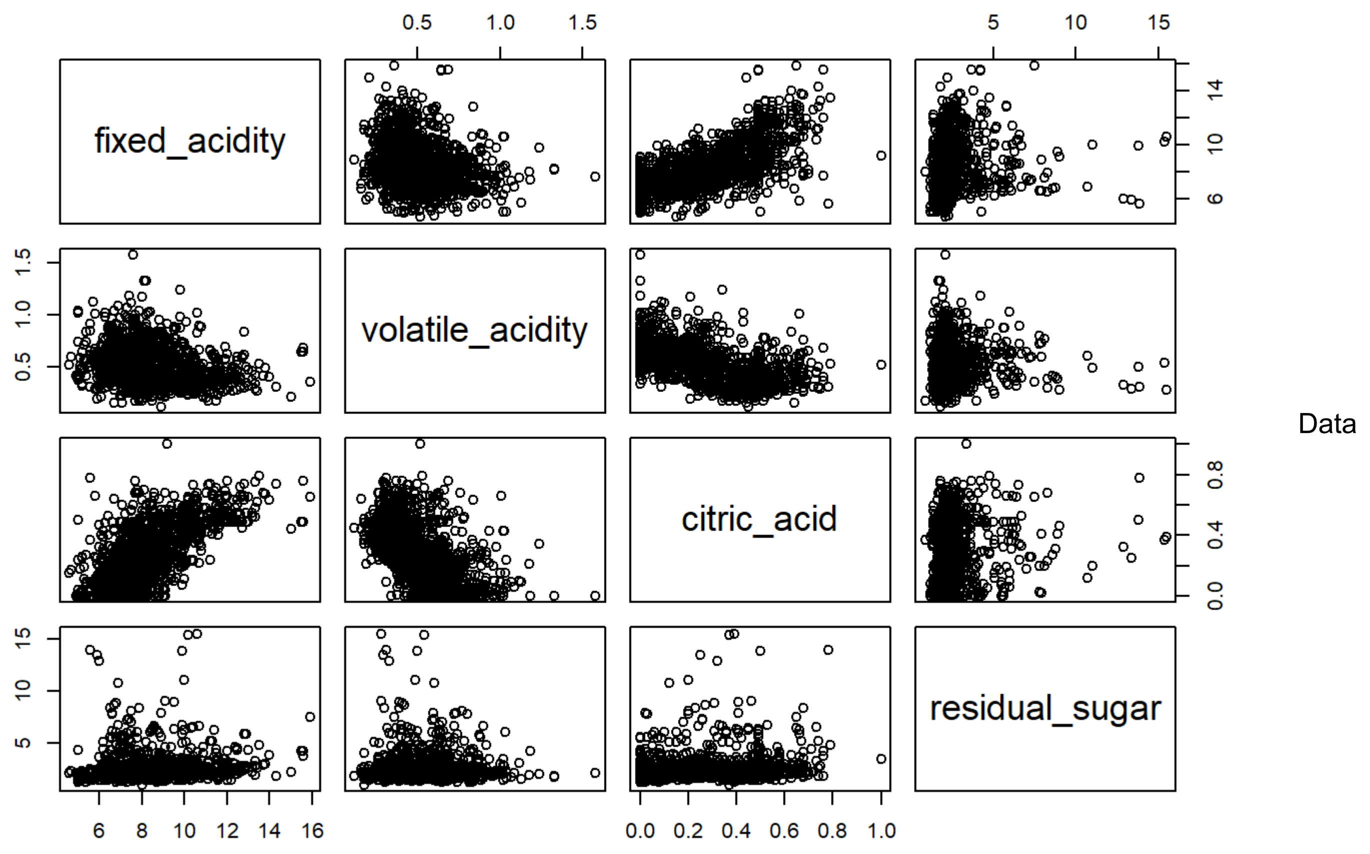
Check for N/As

```
sum(is.na(wine_data_selected))
```

```
## [1] 0
```

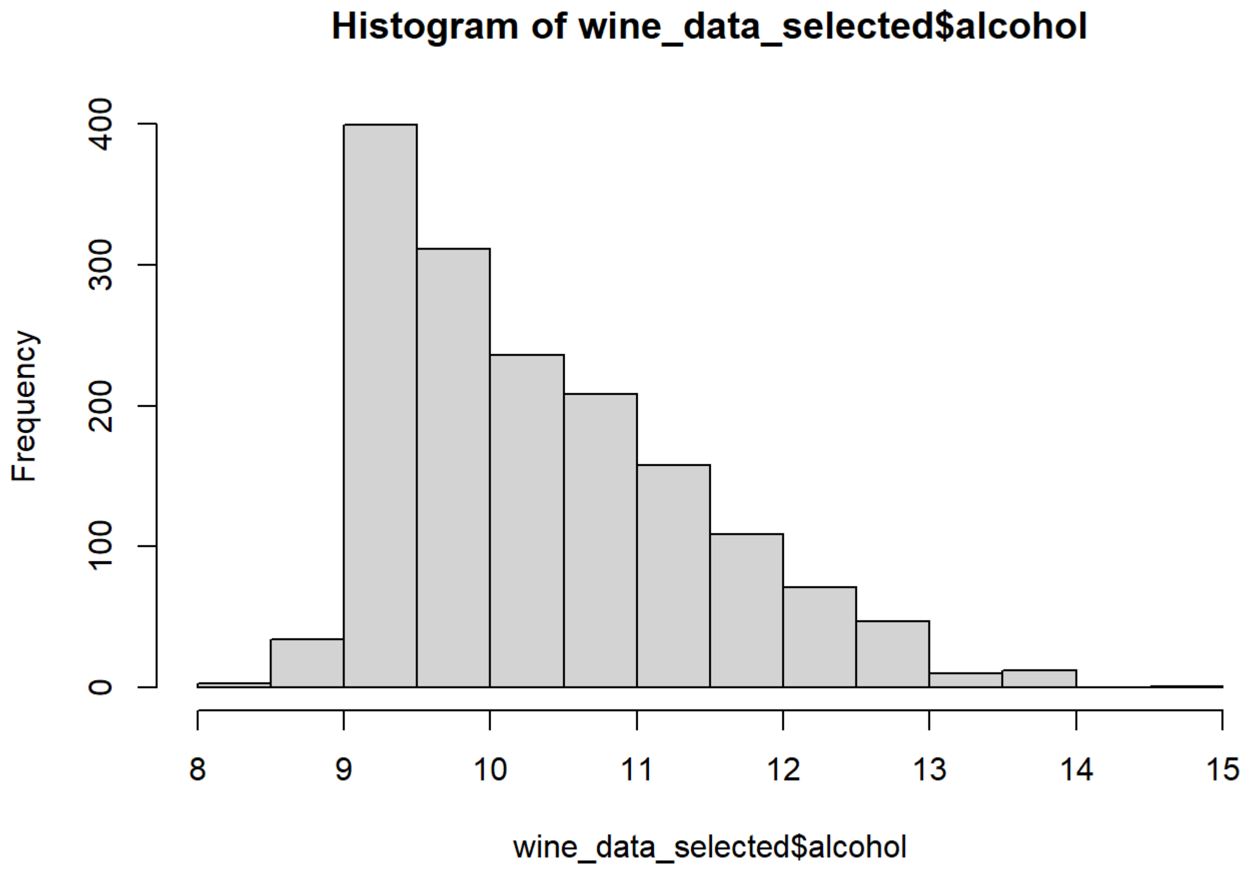
Feature Relationships: Explore relationships between variables to uncover patterns or dependencies. You can use scatter plots, correlation analysis, or interactive visualizations for this purpose.

```
pairs(wine_data_selected[, 1:4])  # Scatter plot matrix of the first four variables
```



Distributions: Examine the distribution of variables to understand their shape and skewness. Histograms, density plots, or quantile-quantile (Q-Q) plots can help in assessing the distributional characteristics.

```
hist(wine_data_selected$alcohol)
```



Best Subset Selection

```
library(leaps)
# Split the data into training and testing sets
set.seed(123)  # For reproducibility
train_indices <- sample(1:nrow(wine_data_base), nrow(wine_data_base) * 0.8)  # 80% for training
train_data <- wine_data_base[train_indices, ]
test_data <- wine_data_base[-train_indices, ]

regfit_best <- regsubsets(quality ~ ., data = train_data, nvmax = 11)
test.mat <- model.matrix(object = quality ~., data = test_data)
errors <- data.frame(c(1:11), rep(0, 11))

colnames(errors) <- c("Num_Predictors", "MSE")

# Show Lowest MSEs in Order with Loop
for (i in 1:11) {
  coefi <- coef(regfit_best, id = i)
  pred <- test.mat[, names(coefi)] %*% coefi
  errors$MSE[i] <- mean((test_data$quality - pred)^2)
}

# Best MSE
best_mse <- min(errors$MSE)
best_num_predictors <- errors$Num_Predictors[which.min(errors$MSE)]
best_model <- summary(regfit_best)$which[which.min(errors$MSE), -1]
best_predictors <- names(best_model)[best_model != 0]
paste("The test MSE for the Best Subset model with", best_num_predictors, "predictors was:", bes
t_mse)
```

```
## [1] "The test MSE for the Best Subset model with 5 predictors was: 0.341511547693687"
```

```
cat("The best subset model includes the following predictors:", paste(best_predictors, collapse
= ", "))
```

```
## The best subset model includes the following predictors: volatile_acidity, chlorides, total_s
ulfur_dioxide, sulphates, alcohol
```

PCA Implementation and Analysis

Check our PCA results for interpretation purposes

```
pca_model <- prcomp(wine_data_selected, scale = TRUE)
names(pca_model)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```
cat('\n=================================================================================
==============\n')
```

```
##
## ===============================================================================================
==========
```

```
pca_model
```

```
## Standard deviations (1, .., p=11):
##  [1] 1.7604353 1.3877715 1.2452082 1.1014684 0.9794346 0.8121627 0.7640623
##  [8] 0.6503512 0.5870623 0.4258323 0.2440457
##
## Rotation (n x k) = (11 x 11):
##                              PC1          PC2         PC3          PC4
## fixed_acidity         0.48931422 -0.110502738  0.12330157 -0.229617370
## volatile_acidity     -0.23858436  0.274930480  0.44996253  0.078959783
## citric_acid           0.46363166 -0.151791356 -0.23824707 -0.079418256
## residual_sugar        0.14610715  0.272080238 -0.10128338 -0.372792562
## chlorides             0.21224658  0.148051555  0.09261383  0.666194756
## free_sulfur_dioxide  -0.03615752  0.513566812 -0.42879287 -0.043537818
## total_sulfur_dioxide  0.02357485  0.569486959 -0.32241450 -0.034577115
## density               0.39535301  0.233575490  0.33887135 -0.174499758
## pH                   -0.43851962  0.006710793 -0.05769735 -0.003787746
## sulphates             0.24292133 -0.037553916 -0.27978615  0.550872362
## alcohol              -0.11323206 -0.386180959 -0.47167322 -0.122181088
##                              PC5         PC6         PC7         PC8
## fixed_acidity         0.08261366 -0.10147858  0.35022736 -0.17759545
## volatile_acidity     -0.21873452 -0.41144893  0.53373510 -0.07877531
## citric_acid           0.05857268 -0.06959338 -0.10549701 -0.37751558
## residual_sugar       -0.73214429 -0.04915555 -0.29066341  0.29984469
## chlorides            -0.24650090 -0.30433857 -0.37041337 -0.35700936
## free_sulfur_dioxide   0.15915198  0.01400021  0.11659611 -0.20478050
## total_sulfur_dioxide  0.22246456 -0.13630755  0.09366237  0.01903597
## density              -0.15707671  0.39115230  0.17048116 -0.23922267
## pH                   -0.26752977  0.52211645  0.02513762 -0.56139075
## sulphates            -0.22596222  0.38126343  0.44746911  0.37460432
## alcohol              -0.35068141 -0.36164504  0.32765090 -0.21762556
##                               PC9        PC10         PC11
## fixed_acidity         0.194020908  0.24952314  0.639691452
## volatile_acidity     -0.129110301 -0.36592473  0.002388597
## citric_acid          -0.381449669 -0.62167708 -0.070910304
## residual_sugar        0.007522949 -0.09287208  0.184029964
## chlorides             0.111338666  0.21767112  0.053065322
## free_sulfur_dioxide   0.635405218 -0.24848326 -0.051420865
## total_sulfur_dioxide -0.592115893  0.37075027  0.068701598
## density               0.020718675  0.23999012 -0.567331898
## pH                   -0.167745886  0.01096960  0.340710903
## sulphates            -0.058367062 -0.11232046  0.069555381
## alcohol               0.037603106  0.30301450 -0.314525906
```

```
cat('\n================================================================================================\n')
```

```
##
## ============================================================================================
==========
```

```
summary(pca_model)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7604 1.3878 1.2452 1.1015 0.97943 0.81216 0.76406
## Proportion of Variance 0.2817 0.1751 0.1410 0.1103 0.08721 0.05996 0.05307
## Cumulative Proportion  0.2817 0.4568 0.5978 0.7081 0.79528 0.85525 0.90832
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.65035 0.58706 0.42583 0.24405
## Proportion of Variance 0.03845 0.03133 0.01648 0.00541
## Cumulative Proportion  0.94677 0.97810 0.99459 1.00000
```

To see how descriptive the principal components are, we plot proportion of variance in data that explained by each principal component.

```
# Calculate variance explained
pve <- pca_model$sdev^2 / sum(pca_model$sdev^2)

# Create a data frame for plotting
df <- data.frame(component = 1:length(pve), pve = pve,
                 cumulative_pve = cumsum(pve))

# Plot Proportion of Variance Explained
pve_plot <- ggplot(data = df) +
  geom_line(aes(x = component, y = pve), color = "maroon") +
  geom_point(aes(x = component, y = pve), color = "maroon", size = 2) +
  labs(x = "Principal Component", y = "Proportion of Variance Explained") +
  scale_x_continuous(breaks = 1:length(pve)) +
  ylim(0, 1) +
  theme_minimal()

# Plot Cumulative Proportion of Variance Explained
cumulative_pve_plot <- ggplot(data = df) +
  geom_line(aes(x = component, y = cumulative_pve), color = "maroon") +
  geom_point(aes(x = component, y = cumulative_pve), color = "maroon", size = 2) +
  labs(x = "Principal Component", y = "Cumulative Proportion of Variance Explained") +
  scale_x_continuous(breaks = 1:length(pve)) +
  ylim(0, 1) +
  theme_minimal()
```

Looking at the principal components also tells us something about the data. We can take the absolute value of the principal components to get the importance each variable has, knowing that magnitude, rather than sign, is important.

For the first principal component, fixed acidity, citric acid, pH, and density are the most important variables

```
pca_model$rotation[,1] %>% abs() %>% sort(decreasing=TRUE)
```

```
##       fixed_acidity         citric_acid                  pH
##          0.48931422          0.46363166          0.43851962
##             density           sulphates    volatile_acidity
##          0.39535301          0.24292133          0.23858436
##           chlorides      residual_sugar             alcohol
##          0.21224658          0.14610715          0.11323206
##  free_sulfur_dioxide total_sulfur_dioxide
##          0.03615752          0.02357485
```

For the second principal component, total sulfur dioxide, free sulfur dioxide, alcohol, and volatile acidity are the most important variables

```
pca_model$rotation[,2] %>% abs() %>% sort(decreasing=TRUE)
```

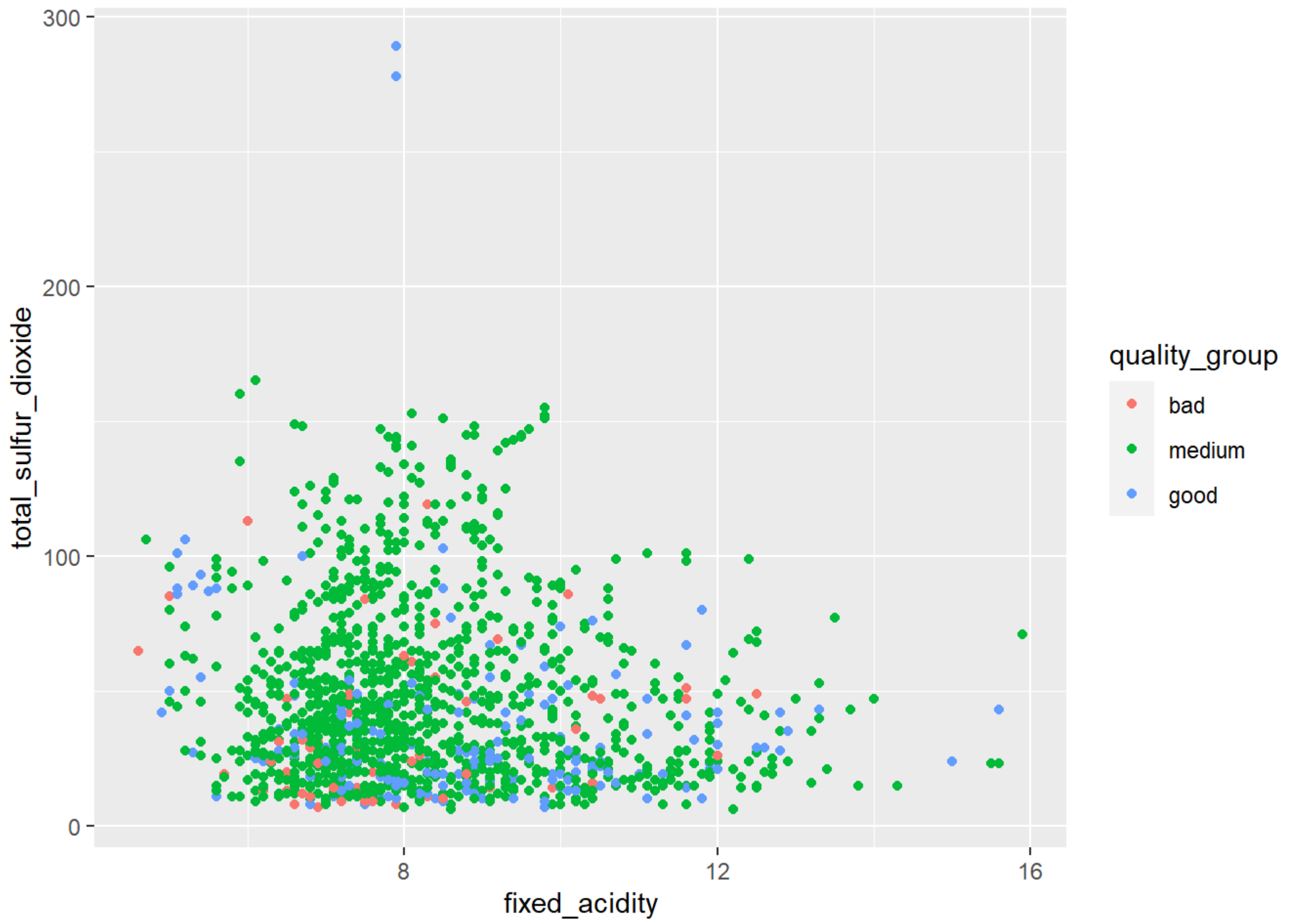
```
## total_sulfur_dioxide  free_sulfur_dioxide             alcohol
##          0.569486959          0.513566812         0.386180959
##     volatile_acidity       residual_sugar             density
##          0.274930480          0.272080238         0.233575490
##          citric_acid            chlorides       fixed_acidity
##          0.151791356          0.148051555         0.110502738
##            sulphates                   pH
##          0.037553916          0.006710793
```

K-Means Clustering

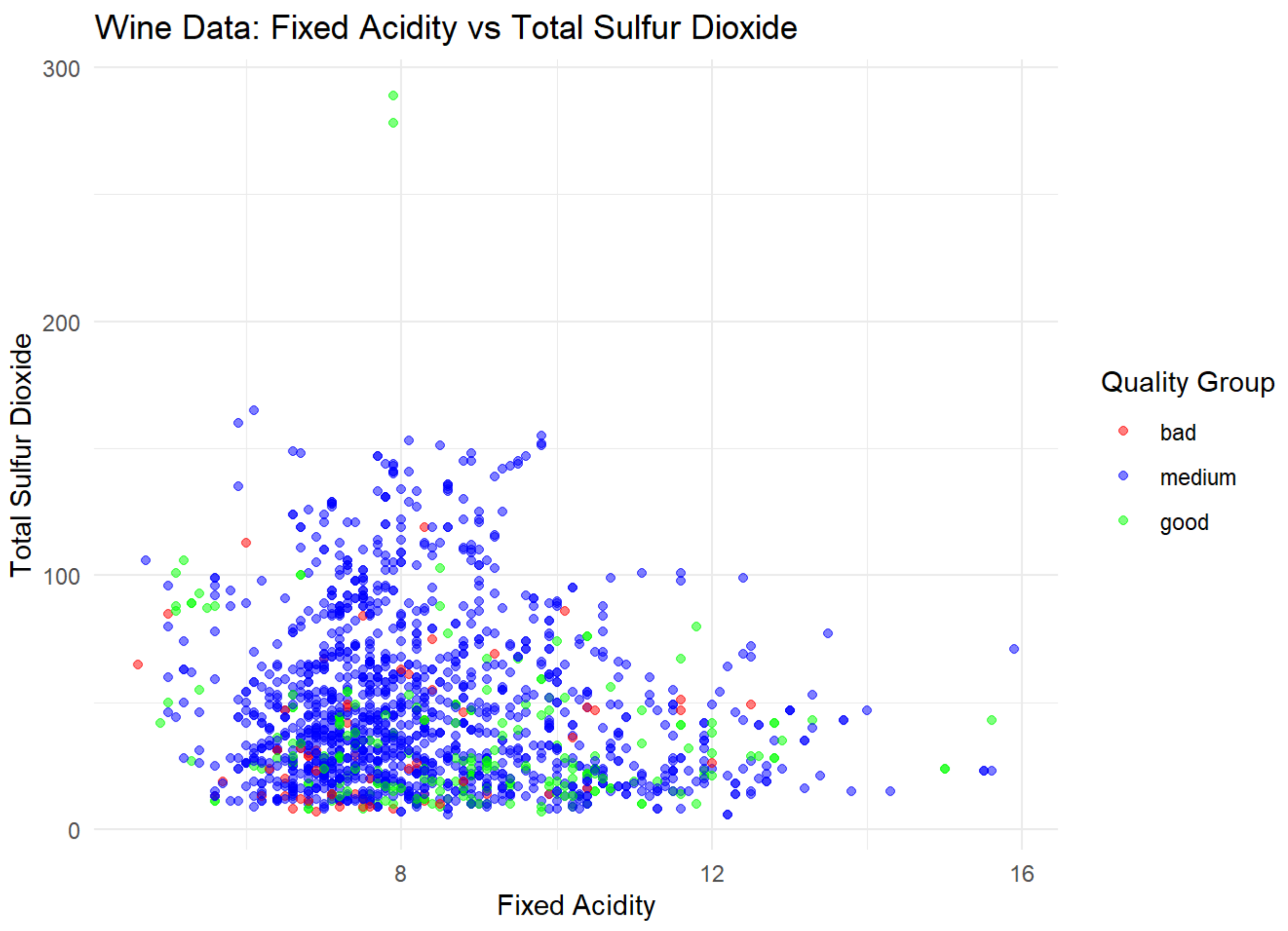
```
#cluster the data into three groups
num_clusters <- 3
kmeans_model <- kmeans(wine_data_selected, centers = num_clusters)
```

Initial plot of original data, blue hue shows quality changes, lighter hue = higher quality. (max 3) darker hue = lower quality (min 3)

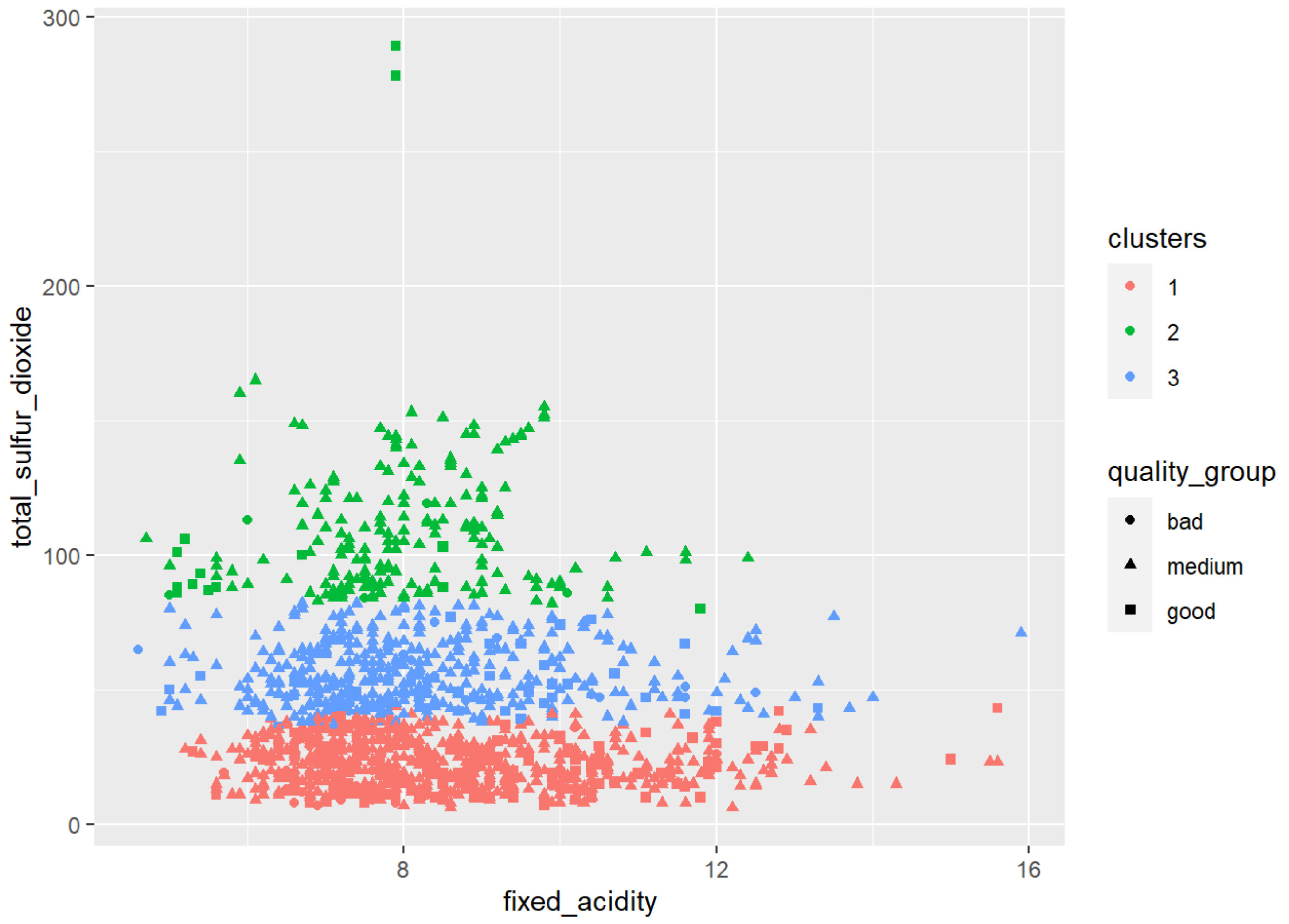
```
ggplot(wine_data, aes(x=fixed_acidity, y=total_sulfur_dioxide, color=quality_group)) + geom_poin
t()
```



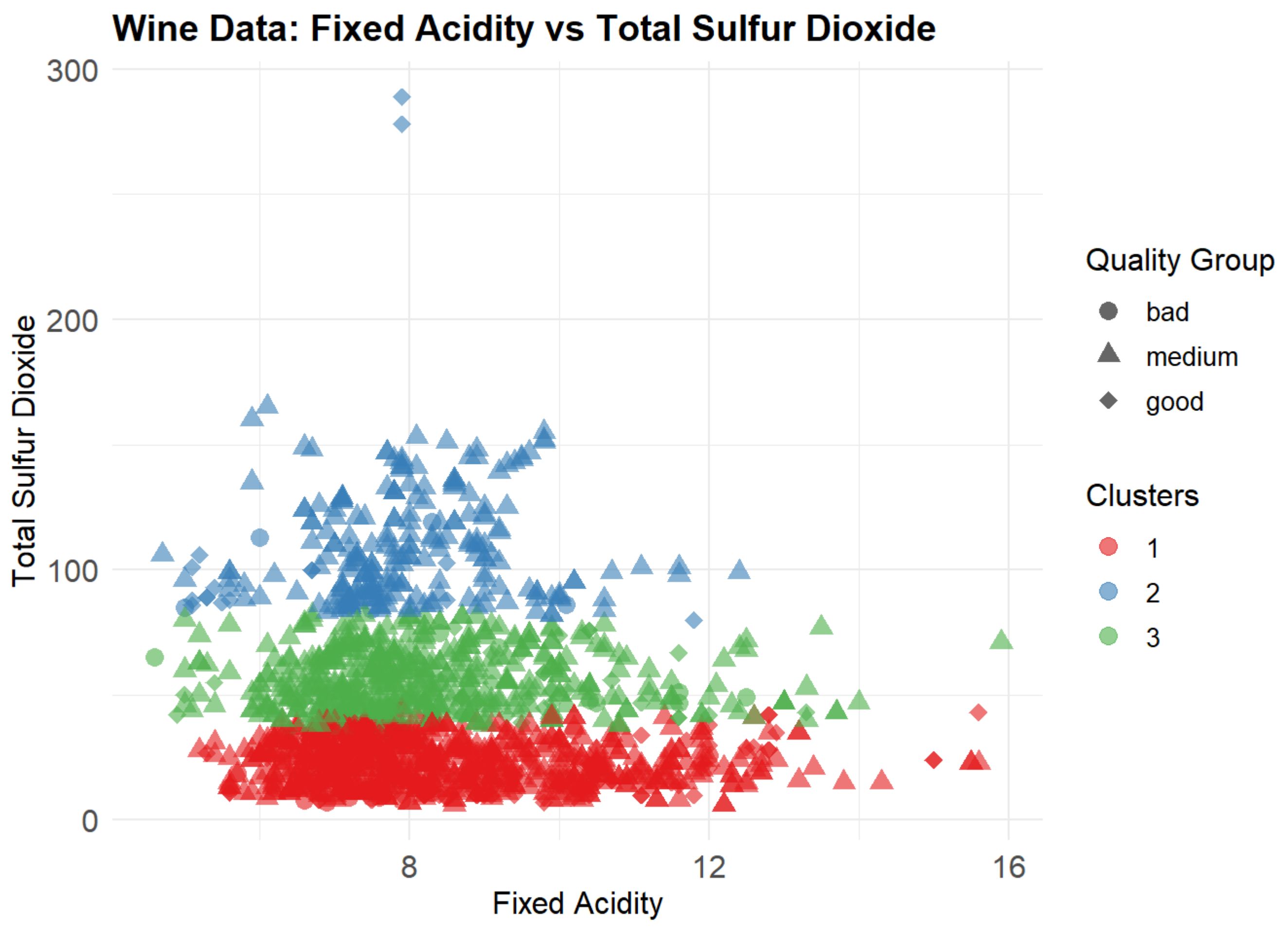
```
ggplot(wine_data, aes(x = fixed_acidity, y = total_sulfur_dioxide, color = quality_group)) +
  geom_point(size = 1.5, alpha = 0.5) +
  scale_color_manual(values = c("red", "blue", "green")) +
  labs(x = "Fixed Acidity", y = "Total Sulfur Dioxide", color = "Quality Group") +
  ggtitle("Wine Data: Fixed Acidity vs Total Sulfur Dioxide") +
  theme_minimal()
```



```
set.seed(2023)
wine.km <- kmeans(select(wine_data, -c('quality','quality_group')), 3, nstart = 20)
wine_data$clusters = as.factor(wine.km$cluster)
ggplot(wine_data, aes(x=fixed_acidity,y=total_sulfur_dioxide, color=clusters, shape=quality_grou
p)) + geom_point()
```



```r
library(ggplot2)

set.seed(2023)
wine.km <- kmeans(select(wine_data, -c('quality', 'quality_group')), 3, nstart = 20)
wine_data$clusters <- as.factor(wine.km$cluster)

ggplot(wine_data, aes(x = fixed_acidity, y = total_sulfur_dioxide, color = clusters, shape = qua
lity_group)) +
  geom_point(size = 3, alpha = 0.6) +
  scale_color_manual(values = c("#E41A1C", "#377EB8", "#4DAF4A")) +
  scale_shape_manual(values = c(16, 17, 18)) +
  labs(x = "Fixed Acidity", y = "Total Sulfur Dioxide", color = "Clusters", shape = "Quality Gro
up") +
  ggtitle("Wine Data: Fixed Acidity vs Total Sulfur Dioxide") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 10))
```

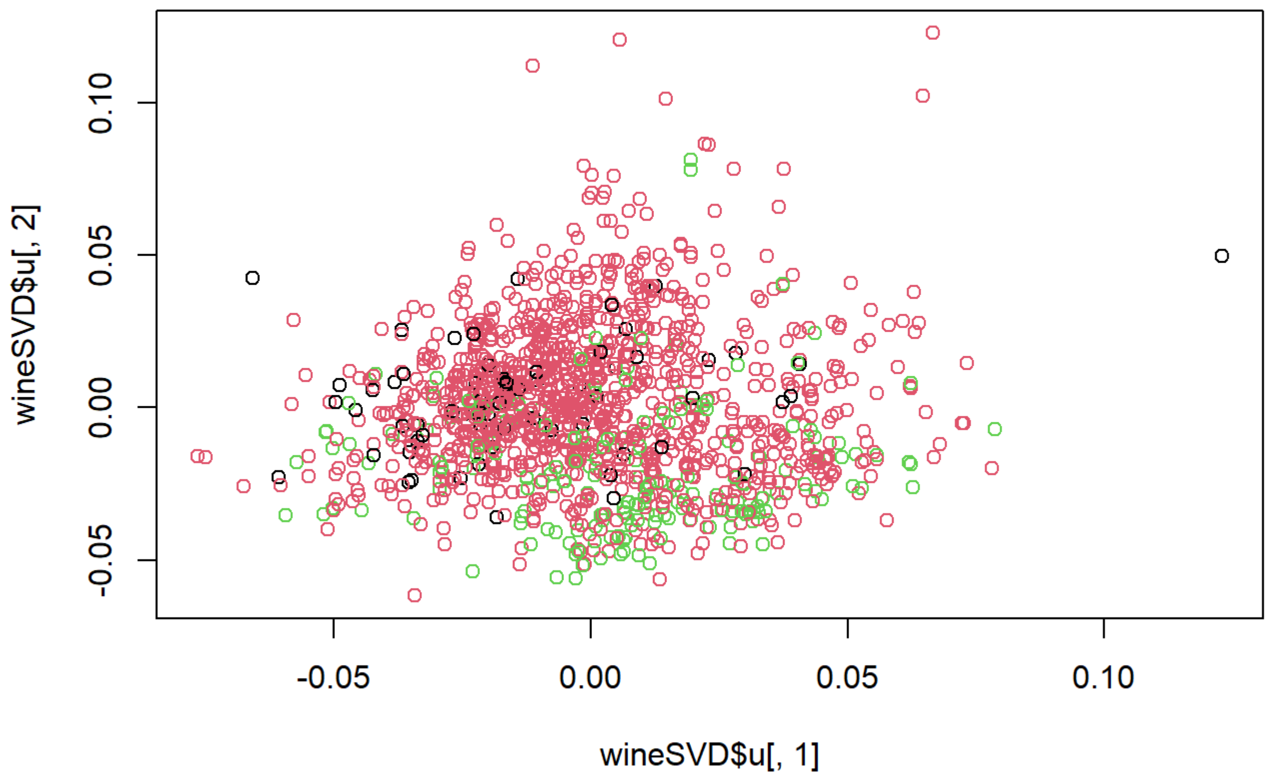


Now take SVD of data

What if we first take the SVD of the data? We first remove the quality label and scale the original data, then take the SVD.

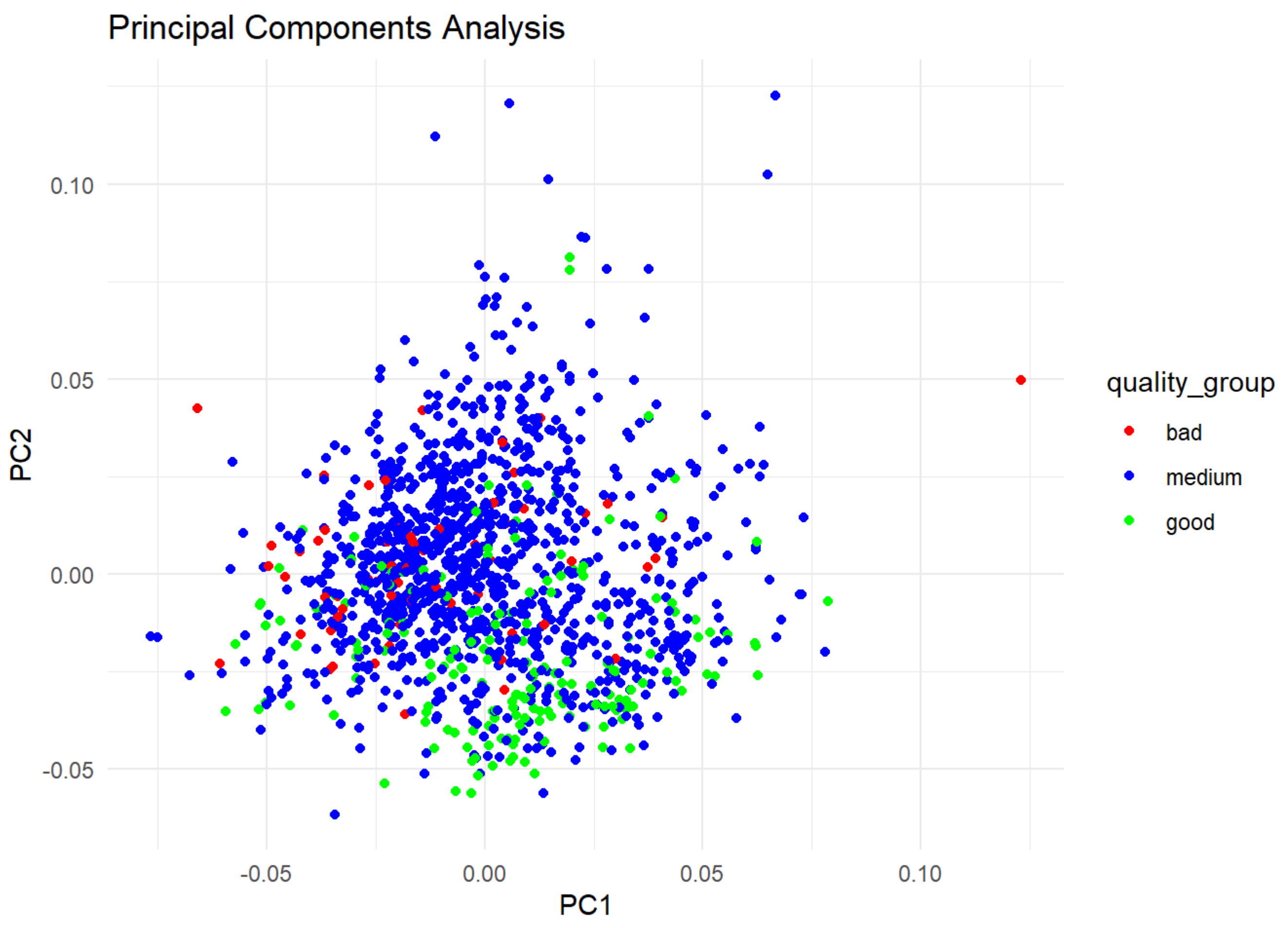
Plotting on the first and third singular vectors, we can see that quality group is decently separated in this case, though not the best.

```
winescale <-  wine_data %>% select(-c(clusters,quality,quality_group)) %>% scale()
wineSVD <- svd(winescale)
plot(wineSVD$u[,1], wineSVD$u[,2], col = wine_data$quality_group)
```



Pretty

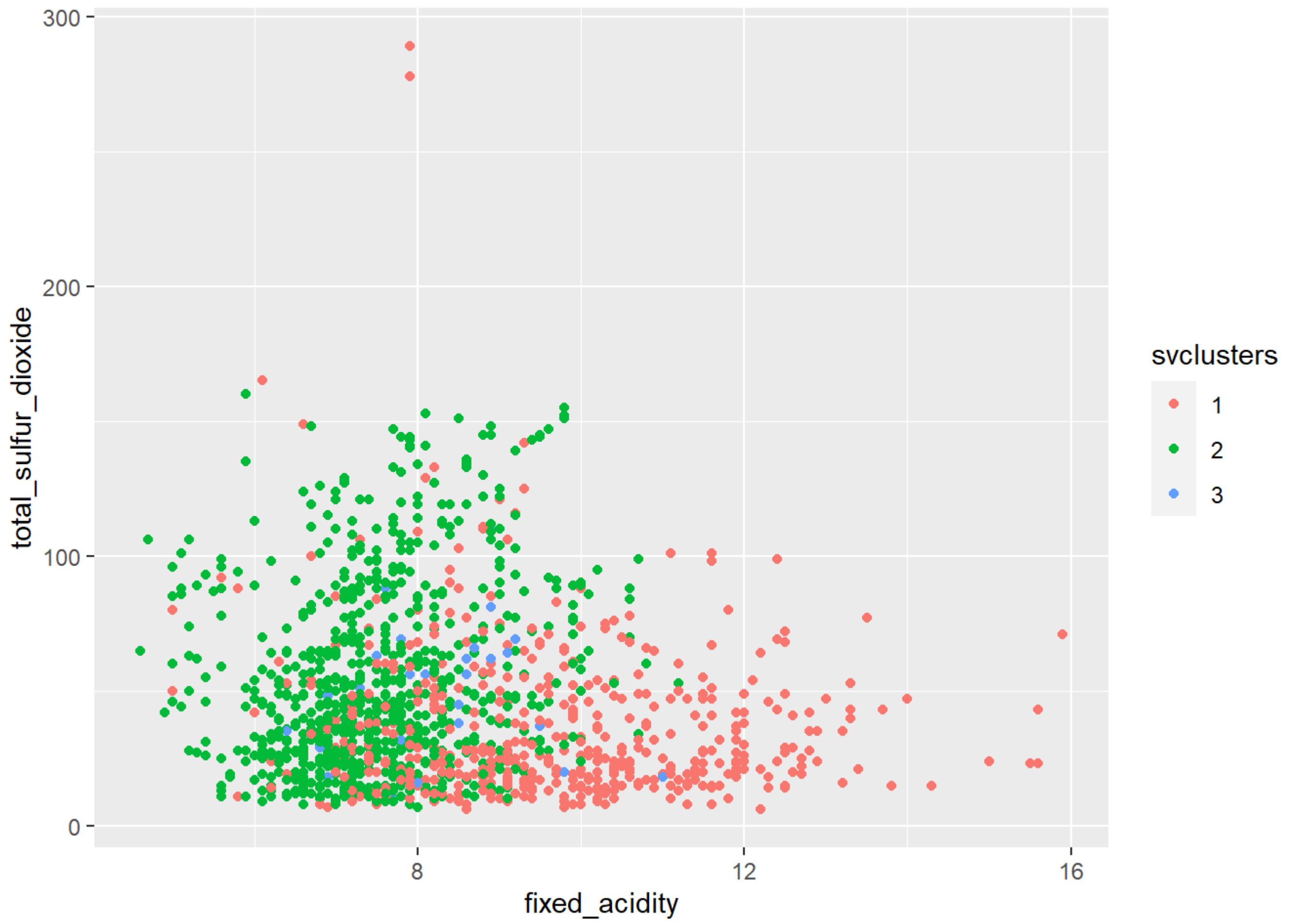
```
library(ggplot2)

# Assuming you have performed SVD on the scaled wine dataset and obtained the U matrix
winescale <- wine_data %>% select(-c(clusters, quality, quality_group)) %>% scale()
wineSVD <- svd(winescale)

# Create a data frame with the first two columns of the U matrix
pca_data <- data.frame(PC1 = wineSVD$u[, 1], PC2 = wineSVD$u[, 2], quality_group = wine_data$qua
lity_group)

# Plot the data using ggplot2
ggplot(pca_data, aes(x = PC1, y = PC2, color = quality_group)) +
  geom_point() +
  scale_color_manual(values = c("red", "blue", "green")) +  # Customize the colors based on qual
ity_group
  labs(x = "PC1", y = "PC2", title = "Principal Components Analysis") +
  theme_minimal()  # Use a minimal theme for a cleaner look
```



Now we apply k-means on the locations of the data in singular vector space. We plot the clusters, agnostic to the quality. This seems somewhat closer to the original, though there is no overbearing theme of one cluster being too dominant like in the actual dataset where 'medium' quality group is dominant

Looks like, green = medium, red = good, blue = bad

```
set.seed(2023)
sv.km <- kmeans(wineSVD$u, 3, nstart = 100)
wine_data$svclusters = as.factor(sv.km$cluster)
ggplot(wine_data, aes(x=fixed_acidity, y=total_sulfur_dioxide, color=svclusters)) + geom_point()
```



ANSWER FOR PART D

We can see that most of the time the cluster is labeled 3, it is bad wine quality. Most of the time the cluster is labeled 2, it is medium quality, and 1, good quality We assign a new label simply based on the output of the clustering and these apparent clusters compared to real quality We also make a variable to tell when these values are different from each other.

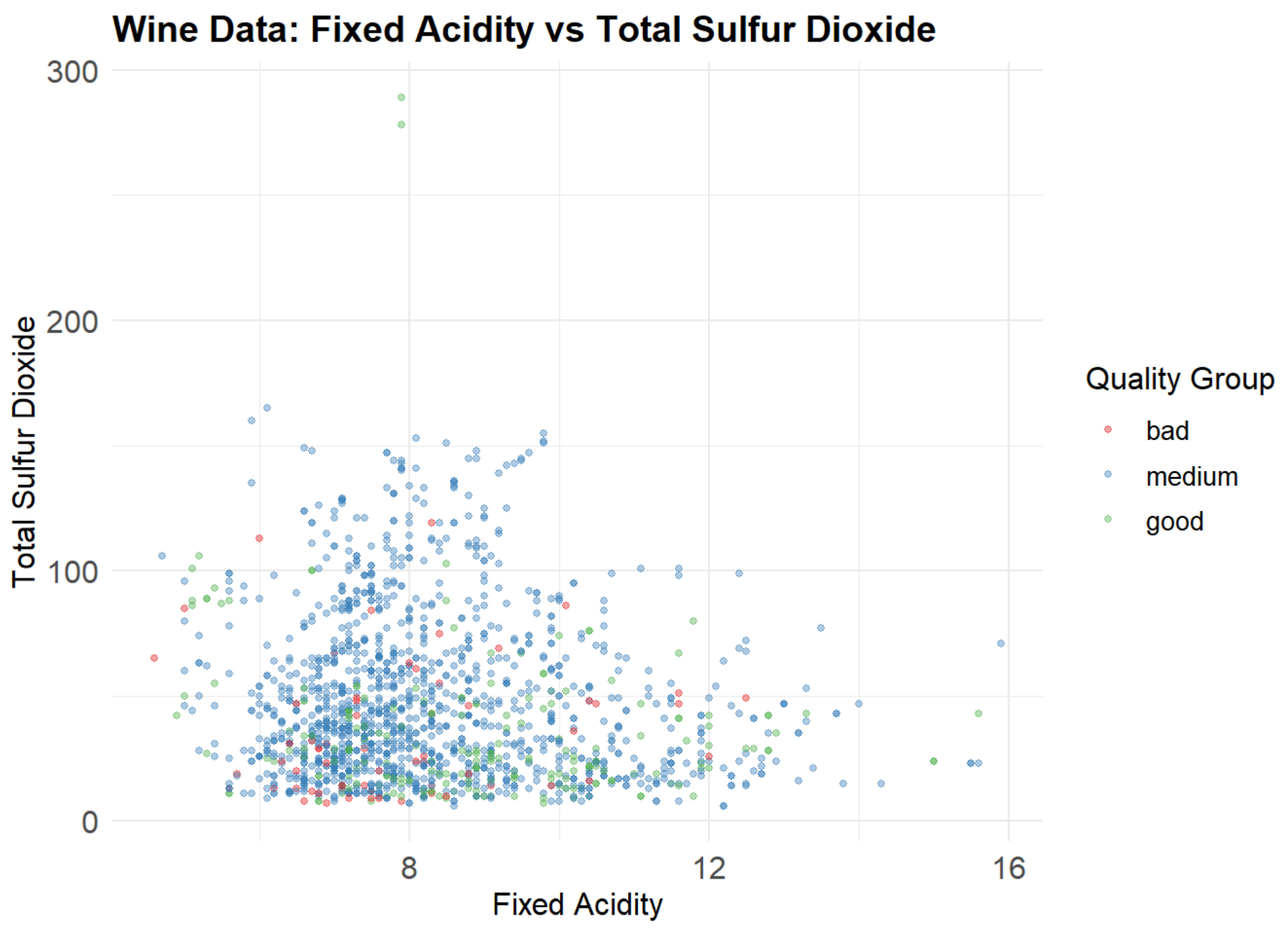
```
wine_data$svlabels = ifelse(wine_data$svclusters == 3, 'bad',
             ifelse(wine_data$svclusters == 2, 'medium',
                              'good' ))
wine_data$different = as.factor(ifelse(wine_data$svlabels == wine_data$quality_group, 0, 1))
```

Plotting the quality groups using color and fill opacity to show when they are different, we can see that the vast majority of the time.

We are seeing that they differ by 34% from the true data.

The clustering attempts to somewhat equally distribute quality groups moreso than the true nature of the dataset.

```
ggplot(wine_data, aes(x = fixed_acidity, y = total_sulfur_dioxide, color = quality_group)) +
  geom_point(size = 1, alpha = 0.4) +
  scale_color_manual(values = c("#E41A1C", "#377EB8", "#4DAF4A")) +
  labs(x = "Fixed Acidity", y = "Total Sulfur Dioxide", color = "Quality Group") +
  ggtitle("Wine Data: Fixed Acidity vs Total Sulfur Dioxide") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 10))
```



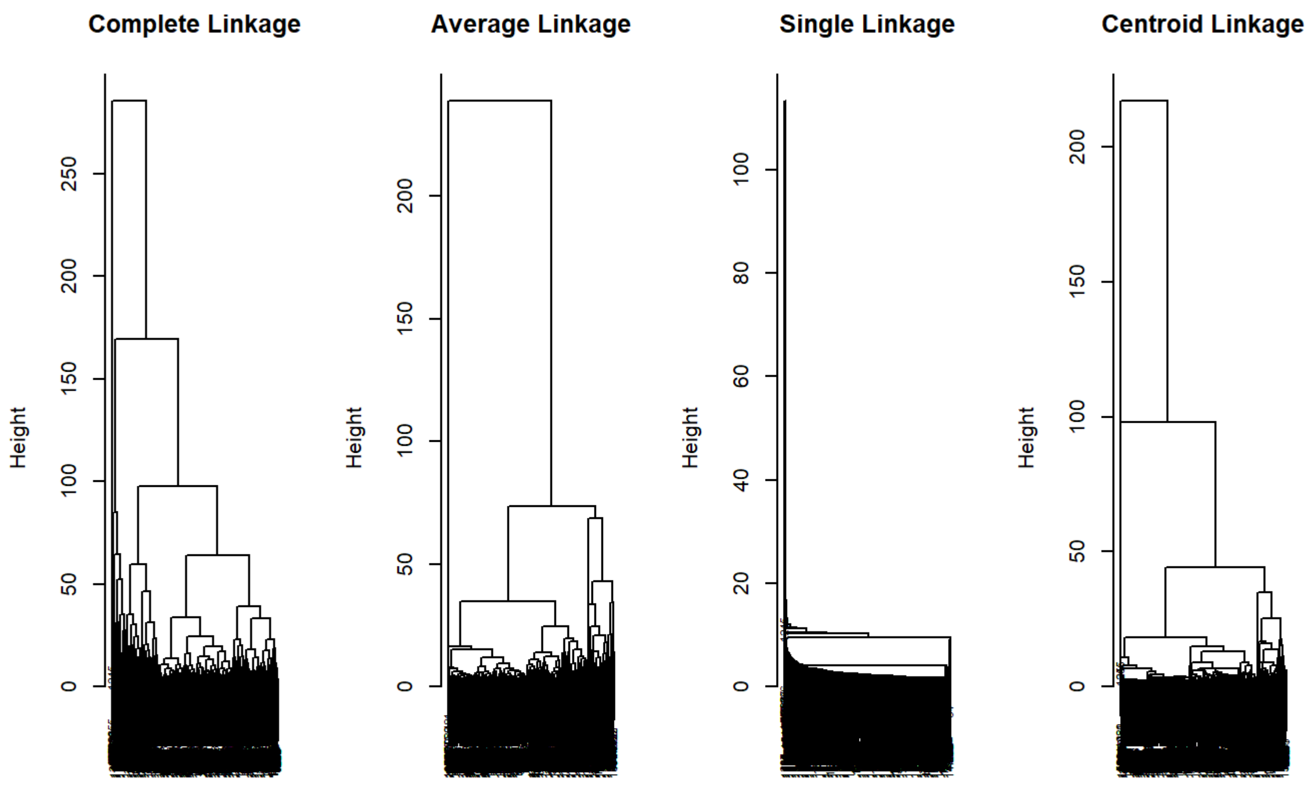
```
sum(wine_data$different==1)/length(wine_data$different)
```

```
## [1] 0.3395872
```

Hierarchical Clustering

Although all linkages seem to not be as interpretable, we can see that the more balanced ones seem to be the complete, average, and centroid linkages. Let's explore further

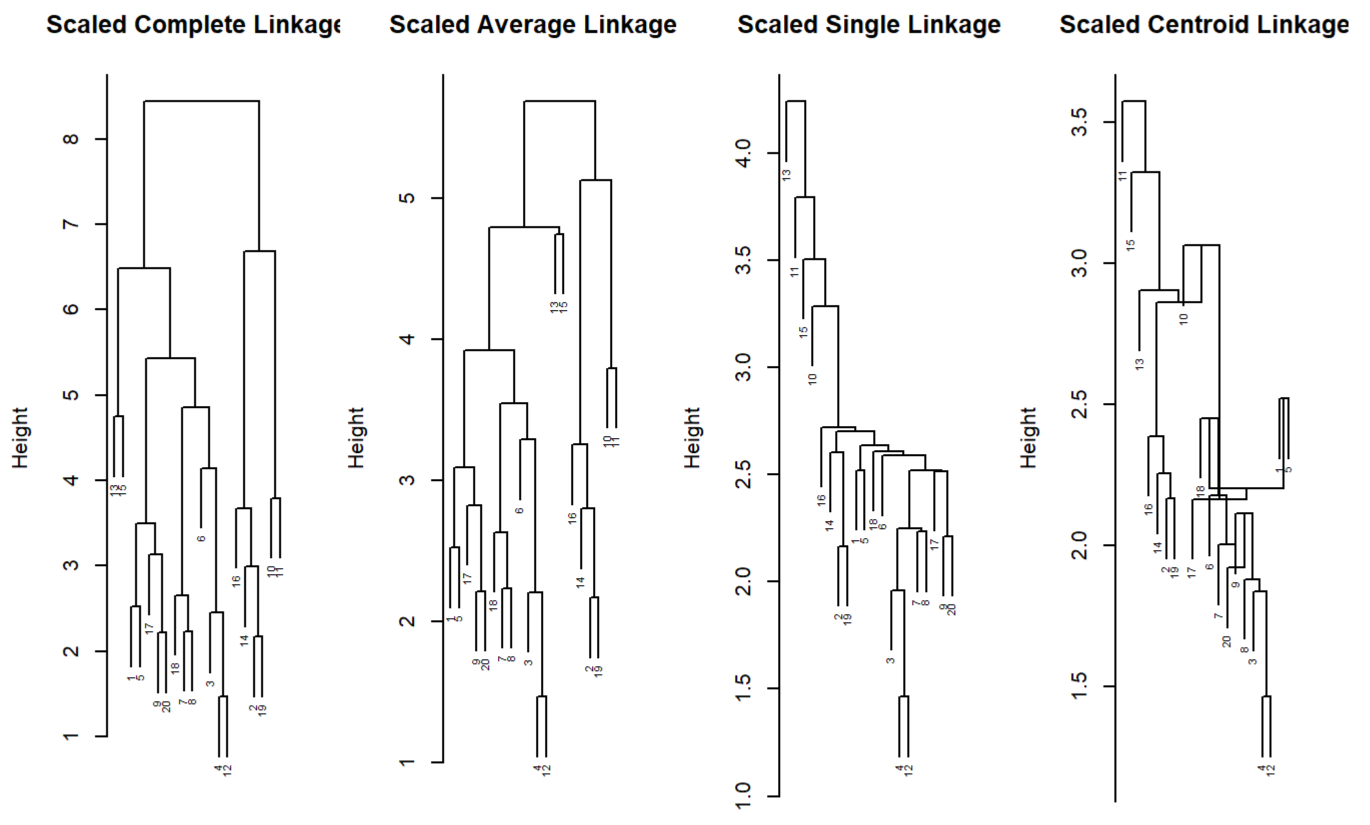
```
set.seed(2023)
wine_data_hc <- wine_data %>% select(-c('clusters','svclusters','quality_group','svlabels','diff
erent'))
small_wine <- wine_data_hc[sample(nrow(wine_data_hc), 20),]

#perform agglomerative hierarchical clustering for all linkage methods
hc.complete <- hclust(dist(wine_data_hc), method = 'complete')
hc.average <- hclust(dist(wine_data_hc), method = 'average')
hc.single <- hclust(dist(wine_data_hc), method = 'single')
hc.centroid <- hclust(dist(wine_data_hc), method = 'centroid')

#Plot
par(mfrow = c(1, 4))
plot(hc.complete, main = "Complete Linkage",
    xlab = "", sub = "", cex = .5)
plot(hc.average, main = "Average Linkage",
    xlab = "", sub = "", cex = .5)
plot(hc.single, main = "Single Linkage",
    xlab = "", sub = "", cex = .5)
plot(hc.centroid, main = "Centroid Linkage",
    xlab = "", sub = "", cex = .5)
```



Now Scale our Data!

On this scaled data, our hierarchical clustering linkges yield good results for complete and average linkages!

```
set.seed(2023)
scaled_small_wine <- wine_data_hc[sample(nrow(wine_data_hc), 20),]
scaled_small_wine <- scaled_small_wine %>% scale()

#perform agglomerative hierarchical clustering for all linkage methods
hc.complete <- hclust(dist(scaled_small_wine), method = 'complete')
hc.average <- hclust(dist(scaled_small_wine), method = 'average')
hc.single <- hclust(dist(scaled_small_wine), method = 'single')
hc.centroid <- hclust(dist(scaled_small_wine), method = 'centroid')

#Plot
par(mfrow = c(1, 4))
plot(hc.complete, main = "Scaled Complete Linkage",
    xlab = "", sub = "", cex = .5)
plot(hc.average, main = "Scaled Average Linkage",
    xlab = "", sub = "", cex = .5)
plot(hc.single, main = "Scaled Single Linkage",
    xlab = "", sub = "", cex = .5)
plot(hc.centroid, main = "Scaled Centroid Linkage",
    xlab = "", sub = "", cex = .5)
```

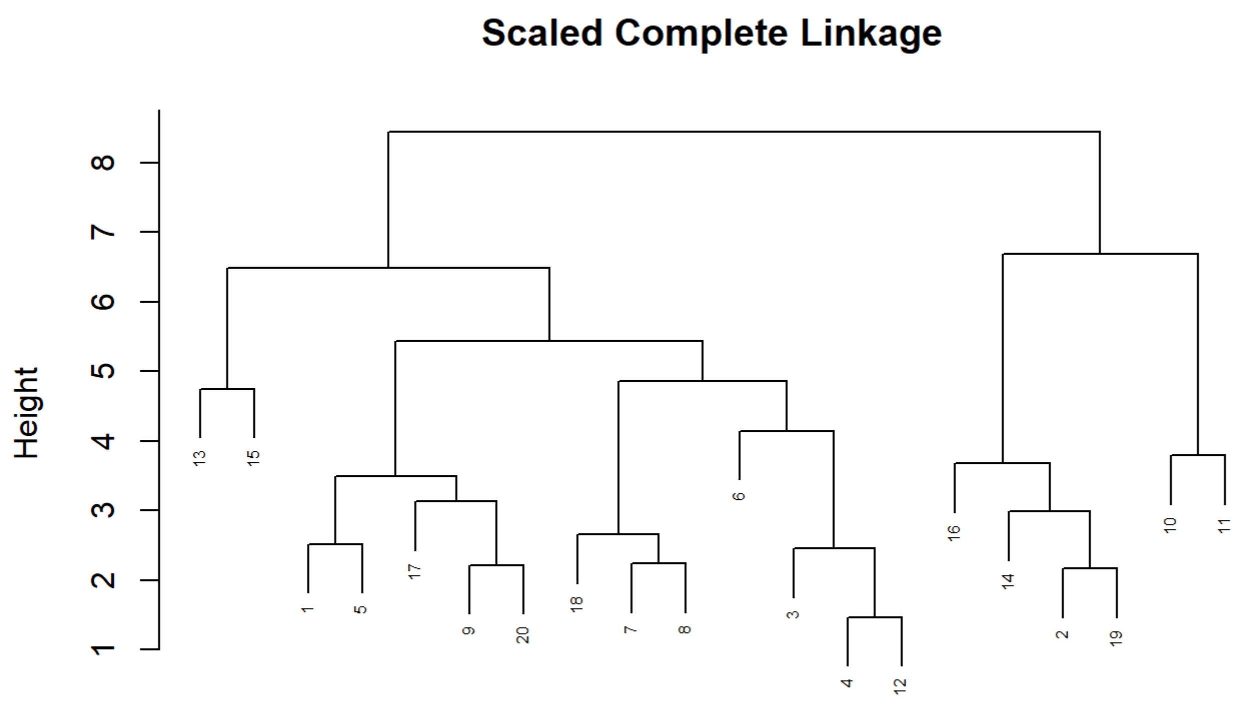


Scaled Complete Linkage

```
#Cut
cutree(hc.complete, 3)
```

```
##  [1] 1 2 1 1 1 1 1 1 1 3 3 1 1 2 1 2 1 1 2 1
```

```
#Plot
plot(hc.complete, main = "Scaled Complete Linkage",
    xlab = "", sub = "", cex = .5)
```

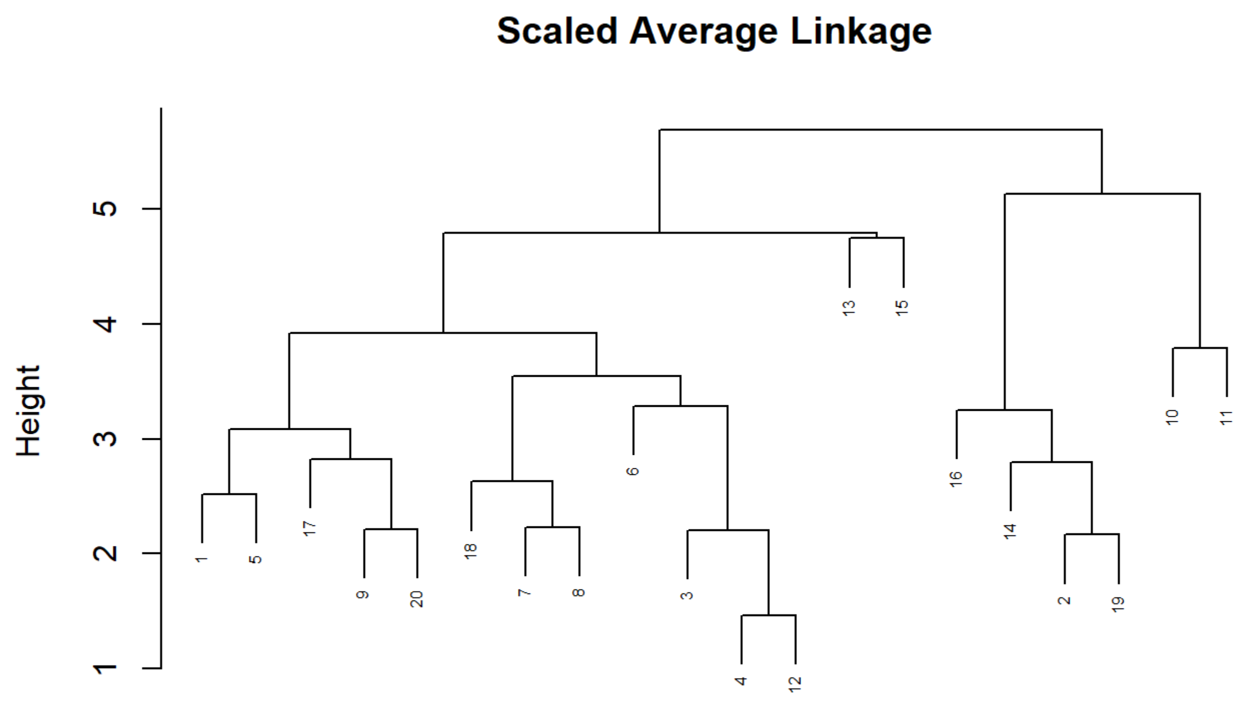


Scaled Average Linkage

```
#Cut
cutree(hc.average, 2)
```

```
##  [1] 1 2 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 1 2 1
```

```
#Plot
plot(hc.average, main = "Scaled Average Linkage",
    xlab = "", sub = "", cex = .5)
```



Supervised Learning Methods

Looking to do classification on this dataset ,.

0.4189827 the loss is calculated using the mean squared error (MSE) loss function.

The MSE loss measures the average squared difference between the predicted values and the true values. A lower MSE loss indicates better performance, as it means that the model's predictions are closer to the actual values.

In the context of your single-layer network on the wine_data dataset, a loss of 0.4189827 suggests that, on average, the model's predictions have a squared difference of 0.4189827 from the true quality values in the training data.

```
#This calculates the accuracy of our model in the end

# Load required libraries
library(readr)
library(keras)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# Define the selected variables
selected_vars <- c("volatile acidity",
                   "chlorides", "total sulfur dioxide",
                    "sulphates", "alcohol")

# Read the CSV file
wine_data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
```

```
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Prepare the data
x <- as.matrix(wine_data[, -ncol(wine_data)])
y <- as.matrix(wine_data$quality)

# Normalize the input features
x <- scale(x)

# Subset the dataset to selected variables
wine_data_subset <- wine_data %>% select(selected_vars)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(selected_vars)
##
##   # Now:
##   data %>% select(all_of(selected_vars))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Split the data into training and testing sets
set.seed(123)
train_indices <- createDataPartition(y, p = 0.8, list = FALSE)
x_train <- x[train_indices, ]
y_train <- y[train_indices, ]
x_test <- x[-train_indices, ]
y_test <- y[-train_indices, ]

# Create a sequential model
model <- keras_model_sequential()

# Add a single layer to the model
model %>%
  layer_dense(units = 1, activation = "linear", input_shape = ncol(x))

# Compile the model
model %>% compile(
  loss = "mean_squared_error",
  optimizer = optimizer_sgd(lr = 0.01)
)

# Train the model
history <- model %>% fit(
  x_train, y_train,
  epochs = 100,
  batch_size = 32,
  validation_split = 0.2
)

# Evaluate the model on test data
test_evaluation <- model %>% evaluate(x_test, y_test)
test_accuracy <- 1 - test_evaluation[[1]]

# Print the test accuracy
cat("Test accuracy:", test_accuracy, "\n")
```

```
## Test accuracy: 0.5828725
```

New Model to calculate the accuracy:

The best subset model includes the following predictors: volatile acidity, chlorides, total sulfur dioxides, sulfates, and alcohol.

```
# Load required libraries
library(caret)

# Read the CSV file
wine_data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Define the selected variables
selected_vars <- c("volatile acidity",
                   "chlorides", "total sulfur dioxide",
                    "sulphates", "alcohol")

# Prepare the data
x <- as.matrix(wine_data[, -ncol(wine_data)])
y <- as.matrix(wine_data$quality)

# Normalize the input features
x <- scale(x)

# Subset the dataset to selected variables
wine_data_subset <- wine_data %>% select(selected_vars)

# Split the data into training and testing sets
set.seed(123)
train_indices <- createDataPartition(y, p = 0.8, list = FALSE)
x_train <- x[train_indices, ]
y_train <- y[train_indices, ]
x_test <- x[-train_indices, ]
y_test <- y[-train_indices, ]

# Create a sequential model
model <- keras_model_sequential()

# Add layers to the model
model %>%
  layer_dense(units = 64, activation = "relu", input_shape = ncol(x)) %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 16, activation = "relu") %>%
  #Add dropout layer
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 1, activation = "linear")

# Compile the model
model %>% compile(
  loss = "mean_squared_error",
  optimizer = optimizer_adam()  # Use Adam optimizer
)

# Train the model
history <- model %>% fit(
  x_train, y_train,
  epochs = 30,
  batch_size = 100,
  validation_split = 0.2
)

# Evaluate the model on test data
test_evaluation <- model %>% evaluate(x_test, y_test)
test_accuracy <- 1 - test_evaluation[[1]]
```

```
# Print the test accuracy
cat("Test accuracy:", test_accuracy, "\n")
```

```
## Test accuracy: 0.1320176
```

```
# Load required libraries
library(caret)

# Read the CSV file
wine_data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Define the selected variables
selected_vars <- c("volatile acidity",
                   "chlorides", "total sulfur dioxide",
                    "sulphates", "alcohol")

# Prepare the data
x <- as.matrix(wine_data[, -ncol(wine_data)])
y <- as.matrix(wine_data$quality)

# Normalize the input features
x <- scale(x)

# Subset the dataset to selected variables
wine_data_subset <- wine_data %>% select(selected_vars)


# Split the data into training and testing sets
set.seed(123)
train_indices <- createDataPartition(y, p = 0.8, list = FALSE)
x_train <- x[train_indices, ]
y_train <- y[train_indices, ]
x_test <- x[-train_indices, ]
y_test <- y[-train_indices, ]

# Create a sequential model
model <- keras_model_sequential()

# Add hidden layers to the model
model %>%
  layer_dense(units = 64, activation = "relu", input_shape = ncol(x)) %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = "linear")

# Compile the model
model %>% compile(
  loss = "mean_squared_error",
  optimizer = optimizer_sgd(lr = 0.001)  # Adjust the learning rate
)

# Train the model
history <- model %>% fit(
  x_train, y_train,
  epochs = 200,  # Increase the number of epochs
  batch_size = 32,
  validation_split = 0.2
)

# Evaluate the model on test data
test_evaluation <- model %>% evaluate(x_test, y_test)
test_accuracy <- 1 - test_evaluation[[1]]
```

```
# Print the test accuracy
cat("Test accuracy:", test_accuracy, "\n")
```

```
## Test accuracy: 0.5732615
```

```
# Load required libraries
library(caret)

# Read the CSV file
wine_data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Define the selected variables
selected_vars <- c("volatile acidity",
                   "chlorides", "total sulfur dioxide",
                    "sulphates", "alcohol")

# Prepare the data
x <- as.matrix(wine_data[, -ncol(wine_data)])
y <- as.matrix(wine_data$quality)

# Normalize the input features
x <- scale(x)

# Subset the dataset to selected variables
wine_data_subset <- wine_data %>% select(selected_vars)


# Split the data into training and testing sets
set.seed(123)
train_indices <- createDataPartition(y, p = 0.8, list = FALSE)
x_train <- x[train_indices, ]
y_train <- y[train_indices, ]
x_test <- x[-train_indices, ]
y_test <- y[-train_indices, ]

# Create a sequential model
model <- keras_model_sequential()

# Add hidden layers to the model
model %>%
  layer_dense(units = 128, activation = "relu", input_shape = ncol(x)) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 1, activation = "linear")

# Compile the model
model %>% compile(
  loss = "mean_squared_error",
  optimizer = optimizer_adam(lr = 0.001)  # Adjust the learning rate and use Adam optimizer
)

# Train the model
history <- model %>% fit(
  x_train, y_train,
  epochs = 40,
  batch_size = 64,
  validation_split = 0.2
)

# Evaluate the model on test data
test_evaluation <- model %>% evaluate(x_test, y_test)
test_accuracy <- 1 - test_evaluation[[1]]
```

```
# Print the test accuracy
cat("Test accuracy:", test_accuracy, "\n")
```

```
## Test accuracy: 0.4735538
```

The best subset model includes the following predictors: volatile acidity, chlorides, total sulfur dioxides, sulfates, and alcohol.

```
# Load required libraries
library(caret)

# Read the CSV file
wine_data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Define the selected variables
selected_vars <- c("volatile acidity",
                   "chlorides", "total sulfur dioxide",
                    "sulphates", "alcohol")

# Prepare the data
x <- as.matrix(wine_data[, -ncol(wine_data)])
y <- as.matrix(wine_data$quality)

# Normalize the input features
x <- scale(x)

# Subset the dataset to selected variables
wine_data_subset <- wine_data %>% select(selected_vars)


# Split the data into training and testing sets
set.seed(123)
train_indices <- createDataPartition(y, p = 0.8, list = FALSE)
x_train <- x[train_indices, ]
y_train <- y[train_indices, ]
x_test <- x[-train_indices, ]
y_test <- y[-train_indices, ]

# Create a sequential model
model <- keras_model_sequential()

# Add hidden layers to the model
model %>%
  layer_dense(units = 256, activation = "relu", input_shape = ncol(x)) %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 1, activation = "linear")

# Compile the model
model %>% compile(
  loss = "mean_squared_error",
  optimizer = optimizer_adam(lr = 0.0005)  # Adjust the learning rate and use Adam optimizer
)

# Train the model
history <- model %>% fit(
  x_train, y_train,
  epochs = 60,  # Increase the number of epochs
  batch_size = 32,
  validation_split = 0.2
)
```

```
# Evaluate the model on test data
test_evaluation <- model %>% evaluate(x_test, y_test)
test_accuracy <- 1 - test_evaluation[[1]]

# Print the test accuracy
cat("Test accuracy:", test_accuracy, "\n")
```

```
## Test accuracy: 0.4548736
```