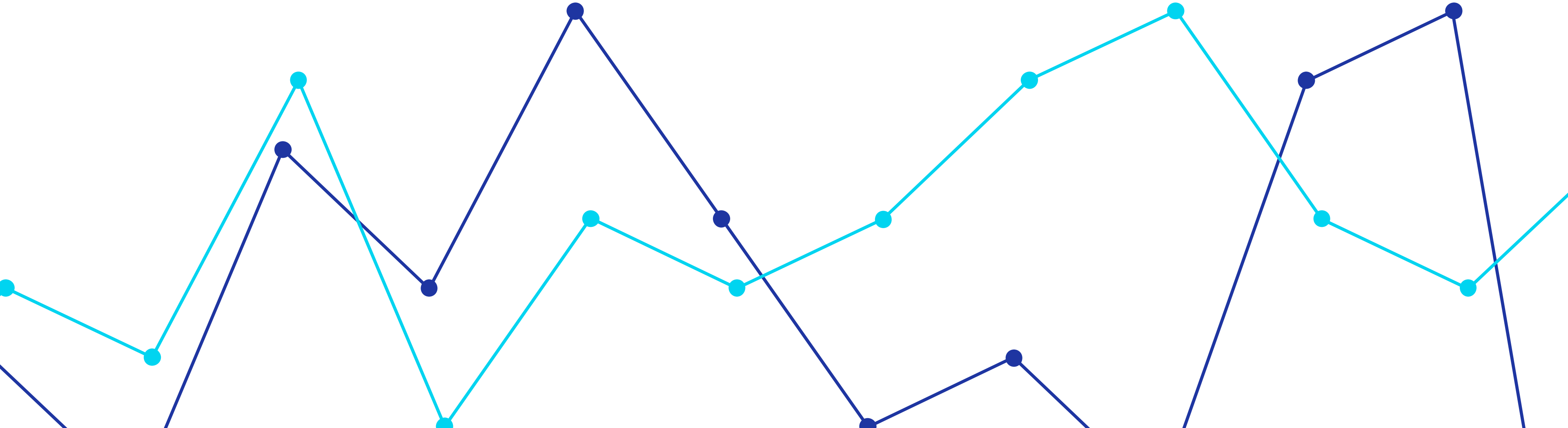
Wine Alcohol Level Prediction

Yagiz Sezersan



Objective

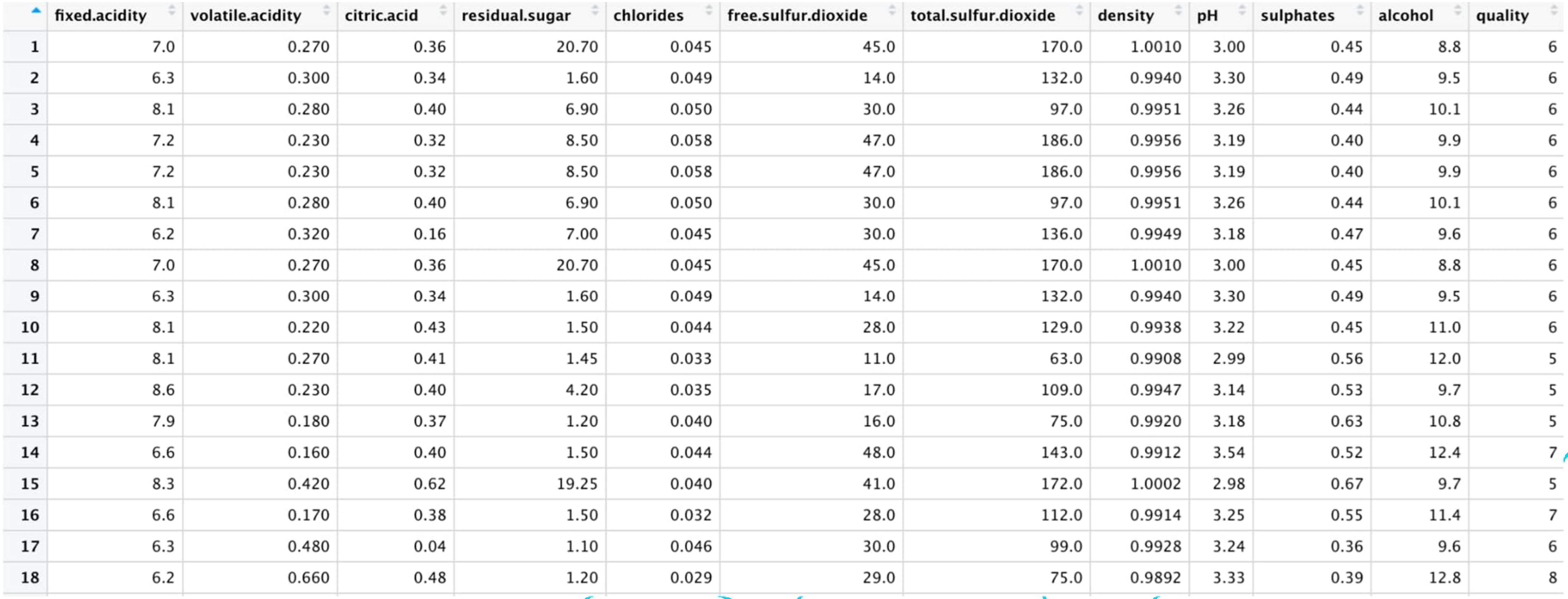
Predict alcohol level content from white wines given relevant data regarding the wine qualities and compositions.

Data Overview

We get the data set from the UCI Machine Learning Repository and the link to the data set is following:

https://archive.ics.uci.edu/ml/datasets/wine+quality

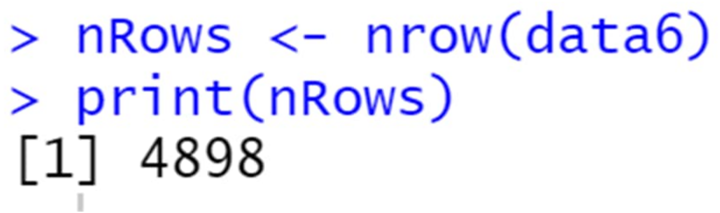
Wine Quality - UC Irvine Machine Learning Repository



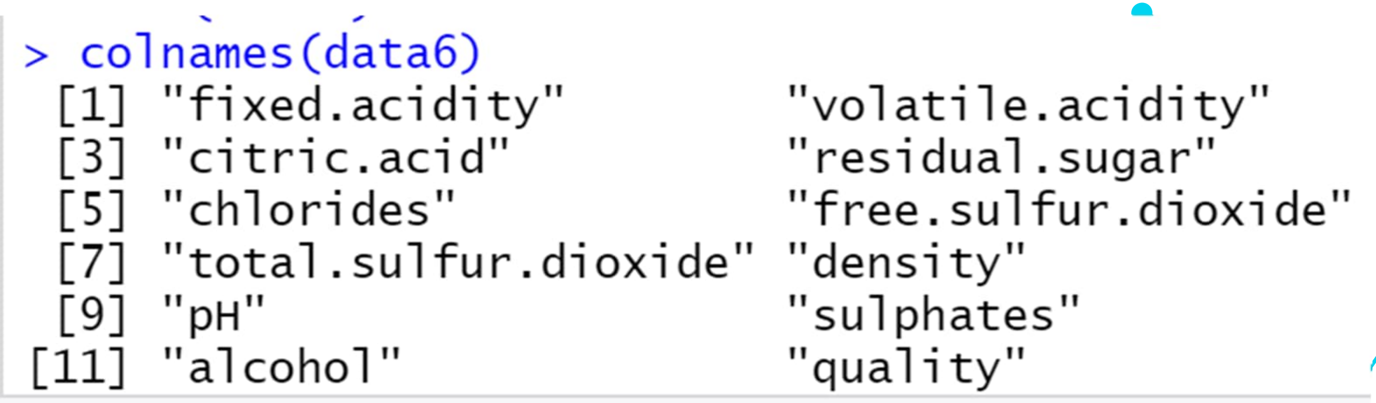
	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
1	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
2	6.3	0.300	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
3	8.1	0.280	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
4	7.2	0.230	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
5	7.2	0.230	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
6	8.1	0.280	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
7	6.2	0.320	0.16	7.00	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6
8	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
9	6.3	0.300	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
10	8.1	0.220	0.43	1.50	0.044	28.0	129.0	0.9938	3.22	0.45	11.0	6
11	8.1	0.270	0.41	1.45	0.033	11.0	63.0	0.9908	2.99	0.56	12.0	5
12	8.6	0.230	0.40	4.20	0.035	17.0	109.0	0.9947	3.14	0.53	9.7	5
13	7.9	0.180	0.37	1.20	0.040	16.0	75.0	0.9920	3.18	0.63	10.8	5
14	6.6	0.160	0.40	1.50	0.044	48.0	143.0	0.9912	3.54	0.52	12.4	7
15	8.3	0.420	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67	9.7	5
16	6.6	0.170	0.38	1.50	0.032	28.0	112.0	0.9914	3.25	0.55	11.4	7
17	6.3	0.480	0.04	1.10	0.046	30.0	99.0	0.9928	3.24	0.36	9.6	6
18	6.2	0.660	0.48	1.20	0.029	29.0	75.0	0.9892	3.33	0.39	12.8	8

Data Overview

The data set have total 4898 rows and 12 columns as shows in the figure



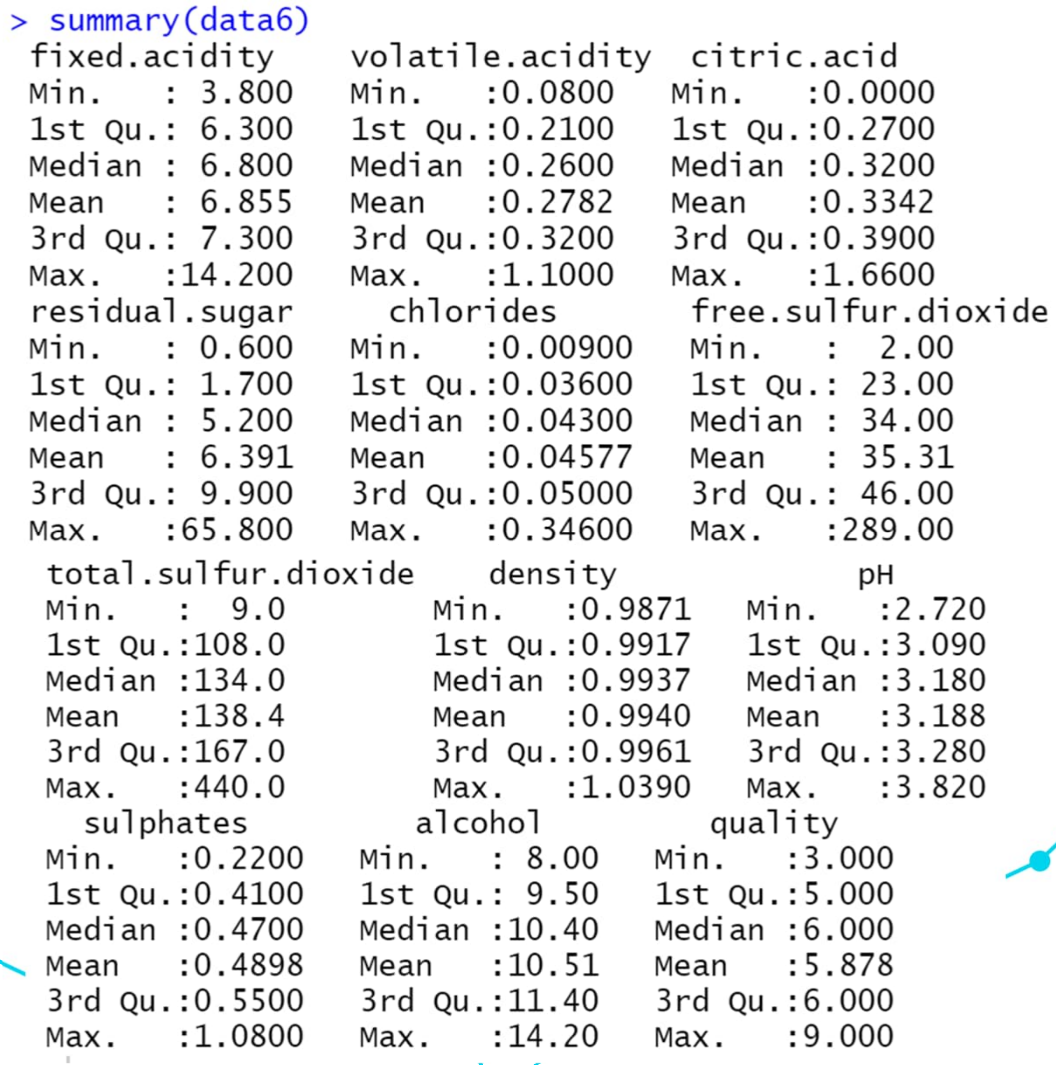
```
> nRows <- nrow(data6)
> print(nRows)
[1] 4898
```



```
> colnames(data6)
 [1] "fixed.acidity"        "volatile.acidity"
 [3] "citric.acid"          "residual.sugar"
 [5] "chlorides"            "free.sulfur.dioxide"
 [7] "total.sulfur.dioxide" "density"
 [9] "pH"                   "sulphates"
[11] "alcohol"              "quality"
```

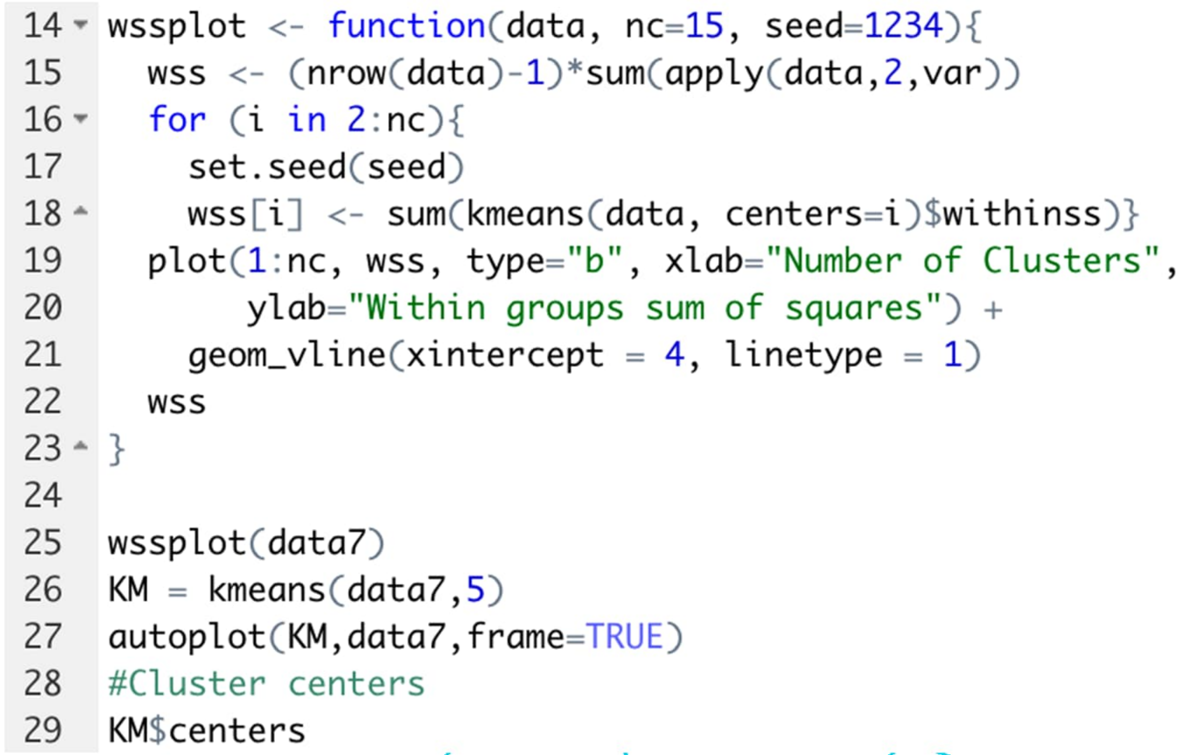
Data Overview

The summary function is a built-in R function used to produce result summaries of various model fitting functions.



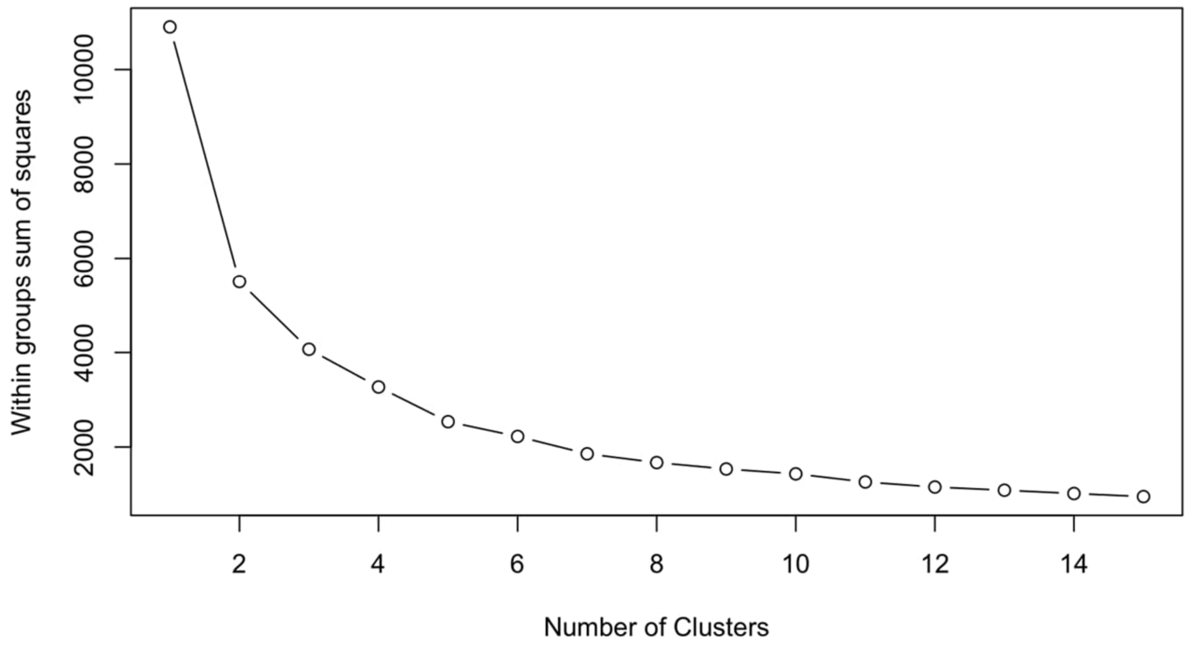
```
> summary(data6)
 fixed.acidity    volatile.acidity  citric.acid
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700
 Median : 6.800   Median :0.2600   Median :0.3200
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600
 residual.sugar     chlorides        free.sulfur.dioxide
 Min.   : 0.600   Min.   :0.00900   Min.   :  2.00
 1st Qu.: 1.700   1st Qu.:0.03600   1st Qu.: 23.00
 Median : 5.200   Median :0.04300   Median : 34.00
 Mean   : 6.391   Mean   :0.04577   Mean   : 35.31
 3rd Qu.: 9.900   3rd Qu.:0.05000   3rd Qu.: 46.00
 Max.   :65.800   Max.   :0.34600   Max.   :289.00
 total.sulfur.dioxide    density             pH
 Min.   :  9.0        Min.   :0.9871   Min.   :2.720
 1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090
 Median :134.0        Median :0.9937   Median :3.180
 Mean   :138.4        Mean   :0.9940   Mean   :3.188
 3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280
 Max.   :440.0        Max.   :1.0390   Max.   :3.820
   sulphates          alcohol          quality
 Min.   :0.2200   Min.   : 8.00   Min.   :3.000
 1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
 Median :0.4700   Median :10.40   Median :6.000
 Mean   :0.4898   Mean   :10.51   Mean   :5.878
 3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
 Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Clustering



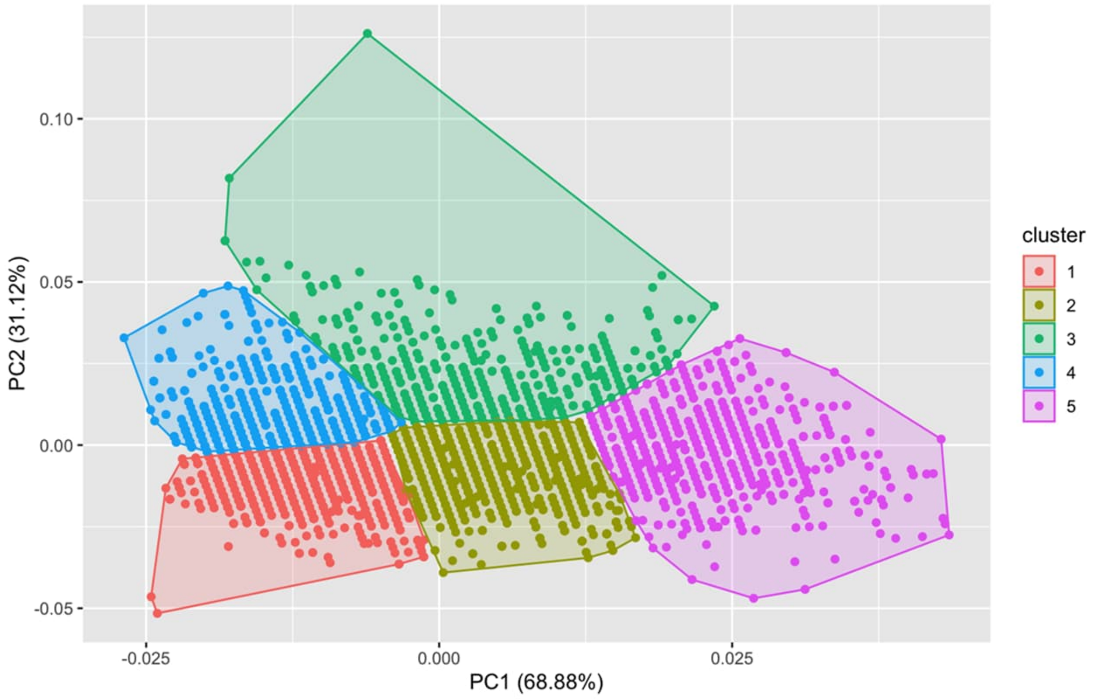
```r
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares") +
    geom_vline(xintercept = 4, linetype = 1)
  wss
}

wssplot(data7)
KM = kmeans(data7,5)
autoplot(KM,data7,frame=TRUE)
#Cluster centers
KM$centers
```

Visualizations Clustering



Visualizations Clustering

K means clusters for Alcohol vs. Fixed Acidity



Key insights from Clustering model

Elbow method

This method shows a clear view of how many clusters there should be.

Clear clusters

There are 5 clear clusters in the plot and those refer to various different wines

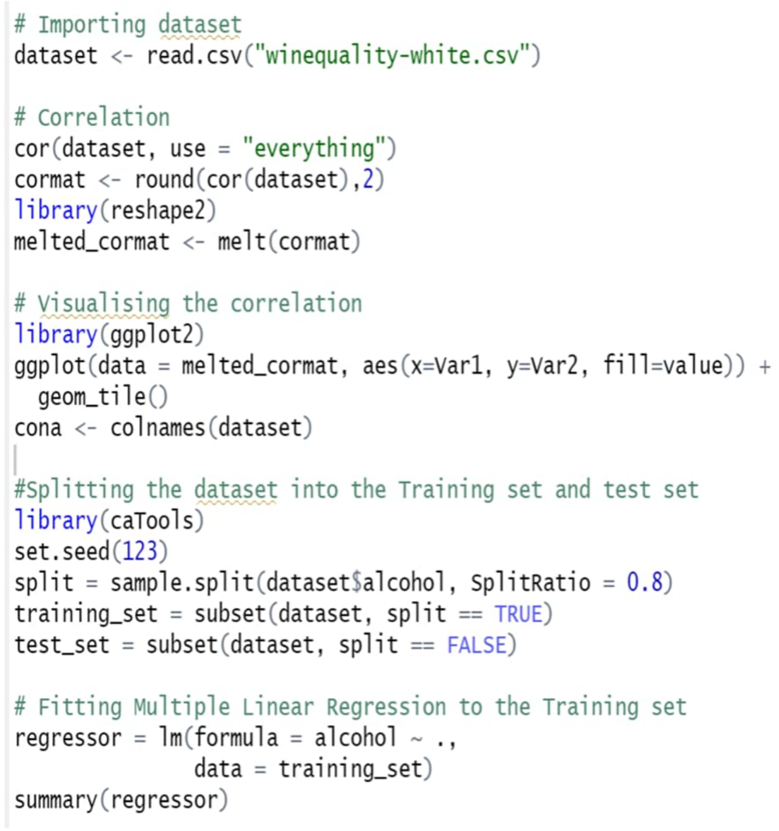
Balance for alcohol and acidity

Alcohol plays a big part in taste of wine, therefore a balance between alcohol and acidity plays a big part

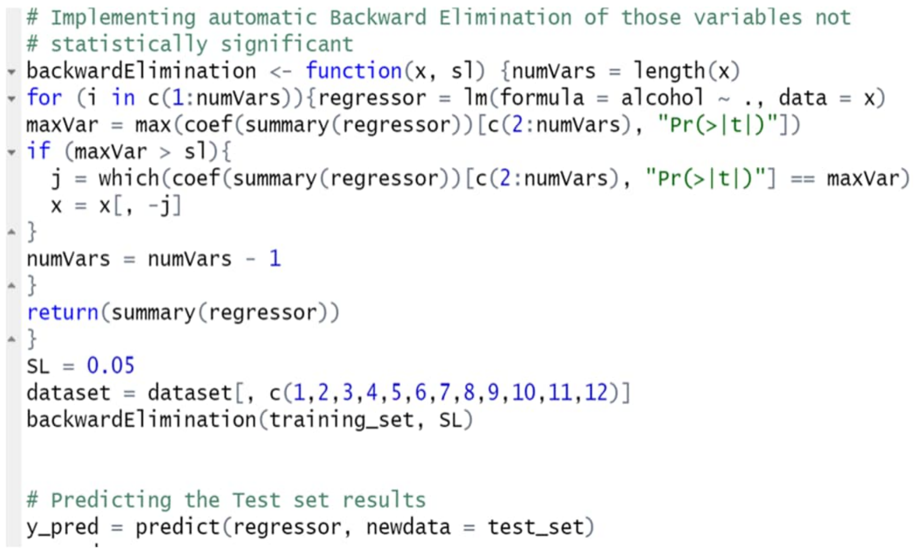
Easy of code

This method can be applied to other features as well for wine segmentation

Linear Regression



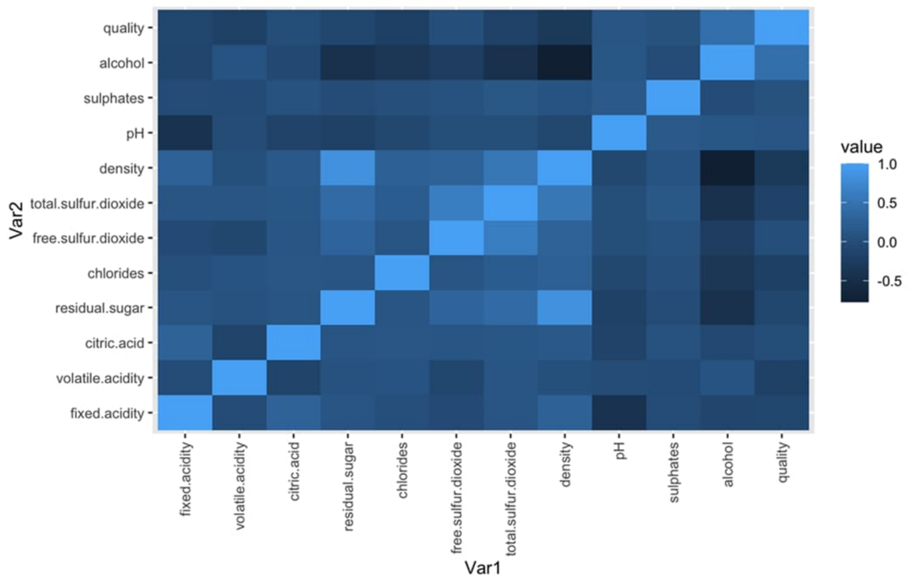
```r
# Importing dataset
dataset <- read.csv("winequality-white.csv")

# Correlation
cor(dataset, use = "everything")
cormat <- round(cor(dataset),2)
library(reshape2)
melted_cormat <- melt(cormat)

# Visualising the correlation
library(ggplot2)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
cona <- colnames(dataset)

#Splitting the dataset into the Training set and test set
library(caTools)
set.seed(123)
split = sample.split(dataset$alcohol, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Fitting Multiple Linear Regression to the Training set
regressor = lm(formula = alcohol ~ .,
               data = training_set)
summary(regressor)
```

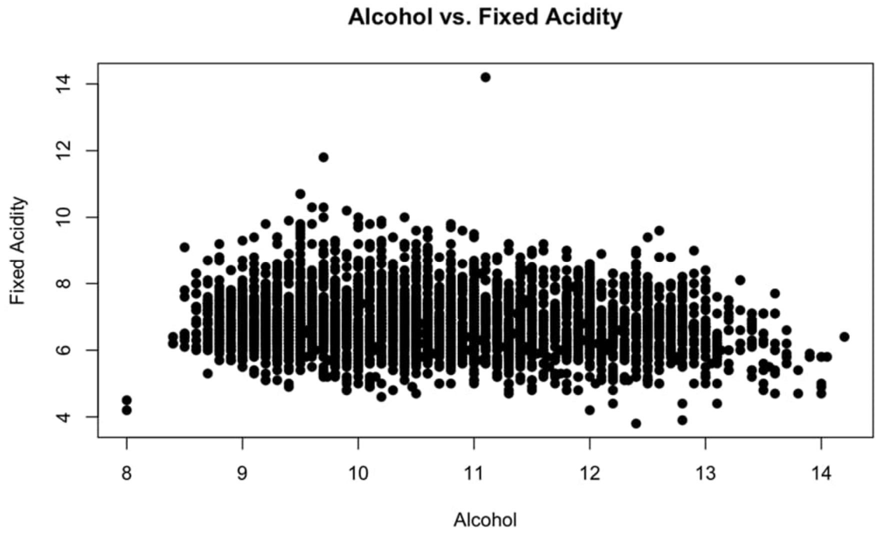


```r
# Implementing automatic Backward Elimination of those variables not
# statistically significant
backwardElimination <- function(x, sl) {numVars = length(x)
for (i in c(1:numVars)){regressor = lm(formula = alcohol ~ ., data = x)
maxVar = max(coef(summary(regressor))[c(2:numVars), "Pr(>|t|)"])
if (maxVar > sl){
  j = which(coef(summary(regressor))[c(2:numVars), "Pr(>|t|)"] == maxVar)
  x = x[, -j]
}
numVars = numVars - 1
}
return(summary(regressor))
}
SL = 0.05
dataset = dataset[, c(1,2,3,4,5,6,7,8,9,10,11,12)]
backwardElimination(training_set, SL)


# Predicting the Test set results
y_pred = predict(regressor, newdata = test_set)
```

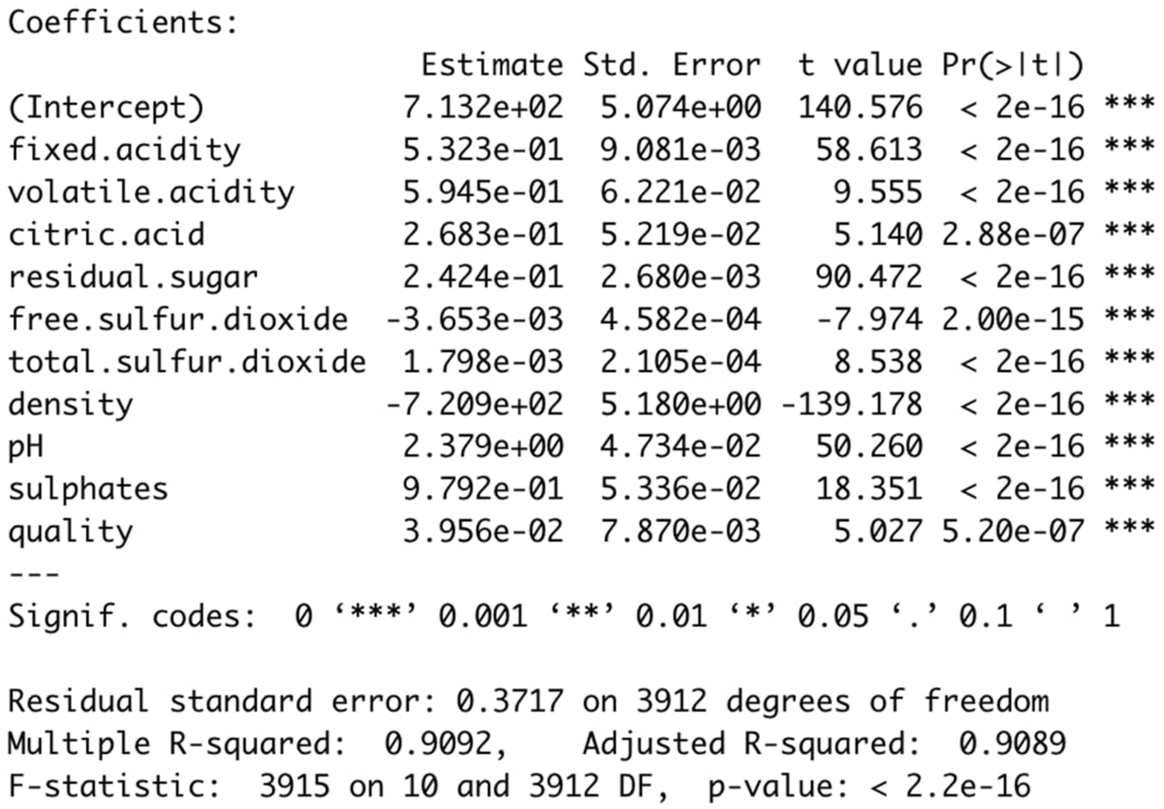
Visualizations Linear Regression

Correlation Heatmap





Linear Regression Results



```
Coefficients:
                       Estimate Std. Error  t value Pr(>|t|)    
(Intercept)           7.132e+02  5.074e+00  140.576  < 2e-16 ***
fixed.acidity         5.323e-01  9.081e-03   58.613  < 2e-16 ***
volatile.acidity      5.945e-01  6.221e-02    9.555  < 2e-16 ***
citric.acid           2.683e-01  5.219e-02    5.140 2.88e-07 ***
residual.sugar        2.424e-01  2.680e-03   90.472  < 2e-16 ***
free.sulfur.dioxide  -3.653e-03  4.582e-04   -7.974 2.00e-15 ***
total.sulfur.dioxide  1.798e-03  2.105e-04    8.538  < 2e-16 ***
density              -7.209e+02  5.180e+00 -139.178  < 2e-16 ***
pH                    2.379e+00  4.734e-02   50.260  < 2e-16 ***
sulphates             9.792e-01  5.336e-02   18.351  < 2e-16 ***
quality               3.956e-02  7.870e-03    5.027 5.20e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3717 on 3912 degrees of freedom
Multiple R-squared:  0.9092,	Adjusted R-squared:  0.9089 
F-statistic:  3915 on 10 and 3912 DF,  p-value: < 2.2e-16
```

Key insights from linear regression model

High accuracy

Model obtained an adjusted R-squared of 0.909

Few Features

Implemented backward elimination to remove features not statistically significant

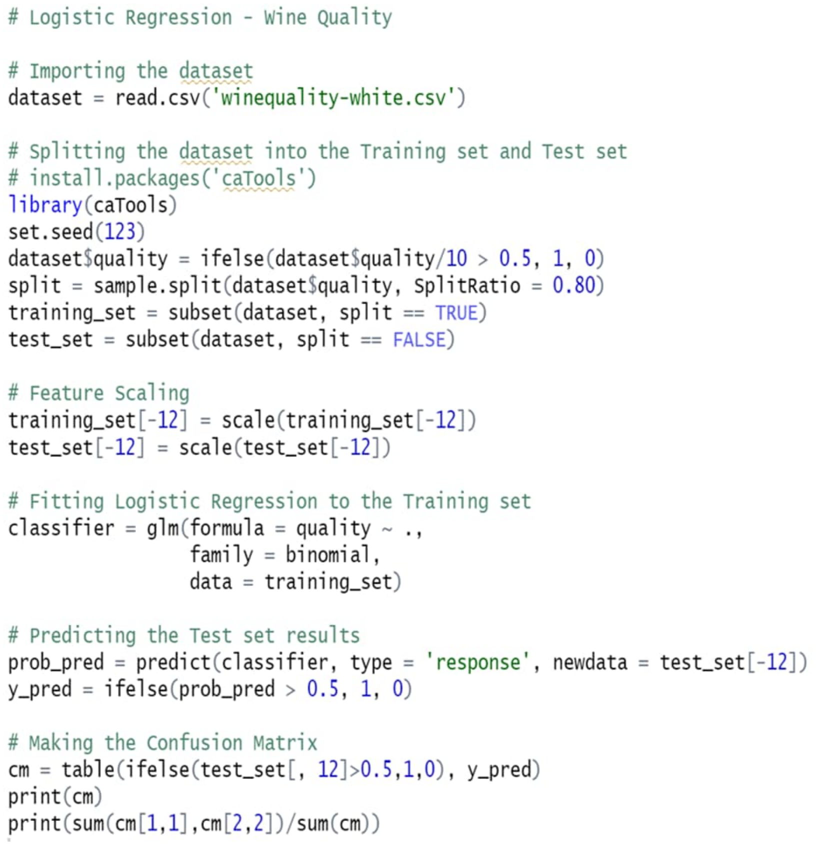
Minimum code

Code can be implemented in quick succession and with all features. (Scores will vary)

Feature correlation

Obtained a high R-squared even with many attributes with low correlation

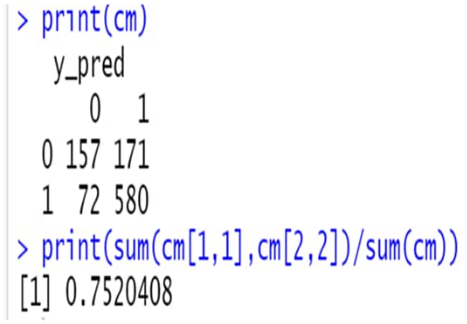
Logistic Regression



```
# Logistic Regression - Wine Quality

# Importing the dataset
dataset = read.csv('winequality-white.csv')

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
dataset$quality = ifelse(dataset$quality/10 > 0.5, 1, 0)
split = sample.split(dataset$quality, SplitRatio = 0.80)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[-12] = scale(training_set[-12])
test_set[-12] = scale(test_set[-12])

# Fitting Logistic Regression to the Training set
classifier = glm(formula = quality ~ .,
                 family = binomial,
                 data = training_set)

# Predicting the Test set results
prob_pred = predict(classifier, type = 'response', newdata = test_set[-12])
y_pred = ifelse(prob_pred > 0.5, 1, 0)

# Making the Confusion Matrix
cm = table(ifelse(test_set[, 12]>0.5,1,0), y_pred)
print(cm)
print(sum(cm[1,1],cm[2,2])/sum(cm))
```

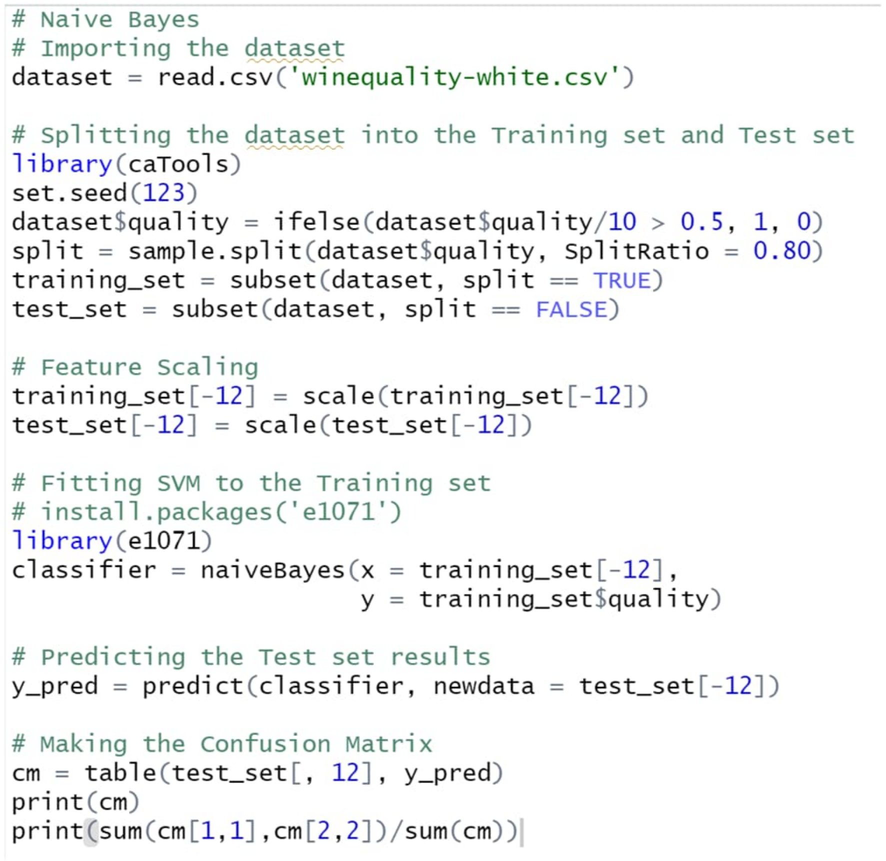
We used Logistic regression to predict the quality of white wine based on the chemical attributes of the wine. 0 is not good, 1 is good. we converted the data by using our quality score divided by 10. For those score above 0.5, we identified as good (1), below 0.5 as bad (0) .

Accuracy is 0.752.



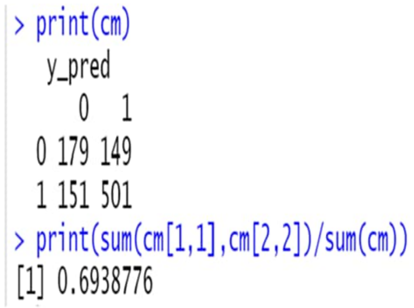
```
> print(cm)
   y_pred
      0   1
  0 157 171
  1  72 580
> print(sum(cm[1,1],cm[2,2])/sum(cm))
[1] 0.7520408
```

Naive Bayes



```
# Naive Bayes
# Importing the dataset
dataset = read.csv('winequality-white.csv')

# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
dataset$quality = ifelse(dataset$quality/10 > 0.5, 1, 0)
split = sample.split(dataset$quality, SplitRatio = 0.80)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[-12] = scale(training_set[-12])
test_set[-12] = scale(test_set[-12])

# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = naiveBayes(x = training_set[-12],
                        y = training_set$quality)

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-12])

# Making the Confusion Matrix
cm = table(test_set[, 12], y_pred)
print(cm)
print(sum(cm[1,1],cm[2,2])/sum(cm))
```

We also tried Naive Bayes to predict the quality of white wine based on the chemical attributes of the wine. 0 is not good, 1 is good. we converted the data by using our quality score divided by 10. For those score above 0.5, we identified as good (1), below 0.5 as bad (0) .
Accuracy: 0.6939.



```
> print(cm)
   y_pred
     0   1
  0 179 149
  1 151 501
> print(sum(cm[1,1],cm[2,2])/sum(cm))
[1] 0.6938776
```

Summary

In this project, we used a white wine dataset to predict alcohol levels based on physical and chemical properties.

At the beginning of the project, we did a review of the data, deduced the main indicators of our data set.
We then applied the clustering method to look for patterns.
Then we tried 3 different models Linear, Logistic, Naive Bayes, the accuracy of the linear model is good, so we chose it as the best model.
For logistic regression, we created training and testing sets that we used to build the model. After we applied the Naive Bayes classifier, these algorithms resulted in a prediction accuracy of about 70 percent, but we were not satisfied with the results.

References

[1] - Cortez,Paulo, Cerdeira,A., Almeida,F., Matos,T. & Reis,J.. (2009). Wine Quality. UCI Machine Learning Repository.