WINE QUALITY ASSESMENT

BY

VIKRANTH REDDY MOOLA

Table of Contents

ABSTRACT:

In this project, we compared the performance of four different classification algorithms in predicting the quality of red wine. In first part of the report, we discussed about dataset and explored the relationship between the output variable and all predictor variables. We also discussed about the Principal Component Analysis and investigated if we can perform any feature extraction to increase the percentage of accuracy. After this, we implemented four classification algorithms: K-Nearest Neighbour, Support Vector Machine and Random Forest and Neural Networks. We then found the best tune for each model and compared the accuracies of each model with the benchmark value set by an academic study to conclude which model is the best for our dataset.

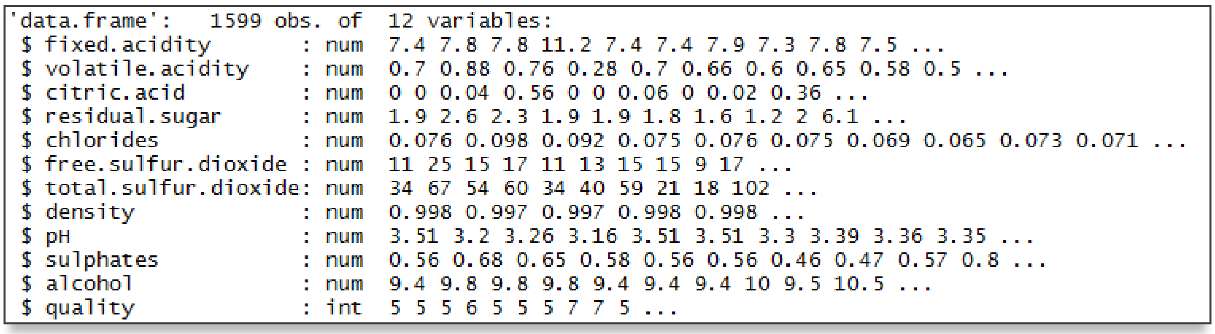
INTRODUCTION:

Now a day's wine is being enjoyed by the wide range of consumers. To support such rapid growth, the wine industry is investing heavily in new technologies which help improve the quality and sales of wines. Quality of the wine and customer satisfaction are directly proportional, and it is the customer satisfaction that brings revenue to the wine industries. Portugal is one of the top 10 exporters of wine and Portugal's *Vinho verde* recorded an increase of 36% in export from 1997-2007. The *Vinho verde* red wine dataset that we chose for our project represents the quality of wines based on a lot of physiochemical attributes such as fixed acidity, sulphates, alcohol, pH to name a few. The values for our output variable range from 0-10, with 0 representing the poorest quality and 10 representing the highest quality wine. With the help of the many data mining techniques which we learnt through this course, we would like to analyze the data and also see which model is the most efficient in predicting the right results and also if possible uncover some results which would help in improving the quality of wine. In this report we will discuss in detail about the various data mining techniques we implement and also substantiate our conclusion with graphs and accuracy metrics.

DATA SOURCES & DATASET STRUCTURE:

The dataset that we used for our project was obtained from the UCI machine learning repository.

Our dataset contains a total of 1599 records and 12 variables, including the output variable.

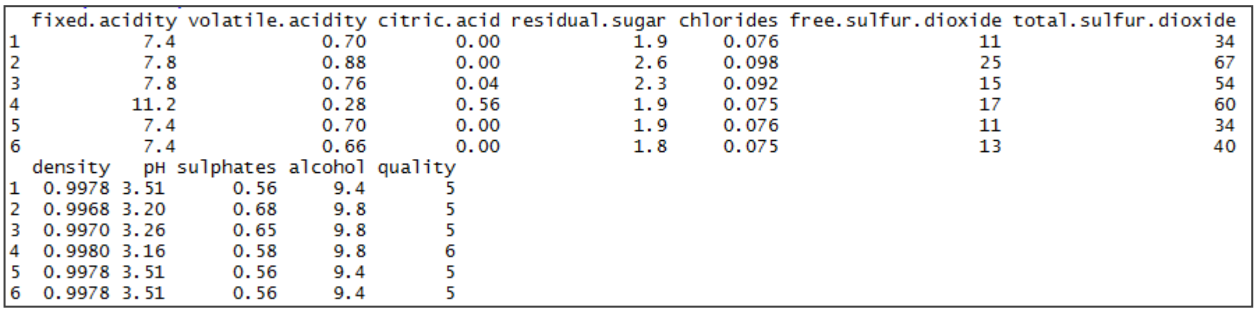


```
'data.frame':	1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

Fig 1: Structure of the dataset

From the above image we observe that all of our 11 predictor values are of type numeric and the output feature quality is of type integer.

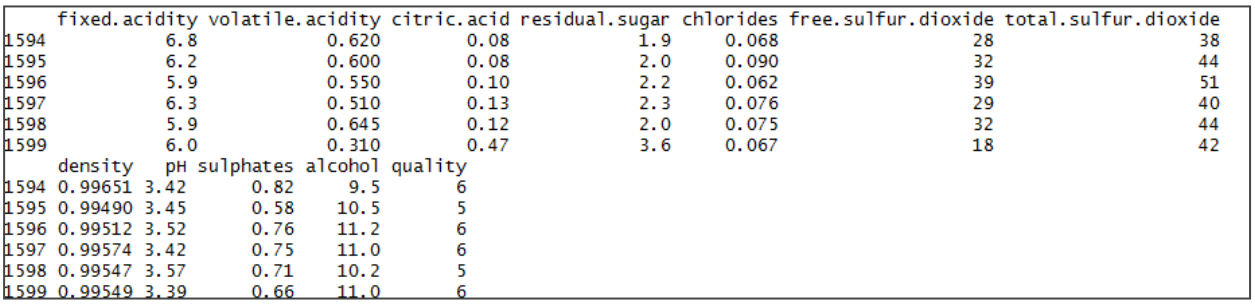
To take a peek into the dataset, we use the function head, which will return the first 6 record of our dataset:



```
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide
1           7.4             0.70        0.00            1.9     0.076                  11                   34
2           7.8             0.88        0.00            2.6     0.098                  25                   67
3           7.8             0.76        0.04            2.3     0.092                  15                   54
4          11.2             0.28        0.56            1.9     0.075                  17                   60
5           7.4             0.70        0.00            1.9     0.076                  11                   34
6           7.4             0.66        0.00            1.8     0.075                  13                   40
  density   pH sulphates alcohol quality
1  0.9978 3.51      0.56     9.4       5
2  0.9968 3.20      0.68     9.8       5
3  0.9970 3.26      0.65     9.8       5
4  0.9980 3.16      0.58     9.8       6
5  0.9978 3.51      0.56     9.4       5
6  0.9978 3.51      0.56     9.4       5
```

Fig 2: First 6 records of the dataset

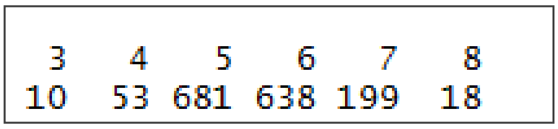
Similarly using the function tail will return the last 6 records of our dataset:



```
     fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide
1594           6.8            0.620        0.08            1.9     0.068                  28                   38
1595           6.2            0.600        0.08            2.0     0.090                  32                   44
1596           5.9            0.550        0.10            2.2     0.062                  39                   51
1597           6.3            0.510        0.13            2.3     0.076                  29                   40
1598           5.9            0.645        0.12            2.0     0.075                  32                   44
1599           6.0            0.310        0.47            3.6     0.067                  18                   42
     density   pH sulphates alcohol quality
1594 0.99651 3.42      0.82     9.5       6
1595 0.99490 3.45      0.58    10.5       5
1596 0.99512 3.52      0.76    11.2       6
1597 0.99574 3.42      0.75    11.0       6
1598 0.99547 3.57      0.71    10.2       5
1599 0.99549 3.39      0.66    11.0       6
```

Fig 3: Last 6 records of the dataset

The image below gives us the class distribution of our response variable quality:



```
  3   4   5   6   7   8
 10  53 681 638 199  18
```

Fig4: Class distribution

We can observe that for the quality values of 3 and 8, there are not many samples which leads to a class imbalance. One thing that we can do to overcome the imbalances is merging the classes. An academic study that we have referred to didn't merge the classes. Since we are comparing our model accuracies with the benchmark values of the academic study, we proceeded with out merging the classes for our results to be comparable.

Features of the Dataset:

Fixed acidity	Total Sulphur Dioxide
Volatile acidity	Density
Citric Acid	pH
Residual Sugar	Sulphates
Chlorides	Alcohol
Free Sulphur Dioxide	Quality

PROPOSED METHODS:

We treat our problem statement as a classification problem and plan to implement the following 4 classification algorithms:

1) K-Nearest Neighbours
2) Support Vector Machine
3) Random Forest
4) Neural Networks

K-Nearest Neighbours:

K nearest neighbours is a wisely used non parametric lazy learning algorithm. The KNN algorithm finds a group of K nearest neighbours in the training set which are the closest to the test object, and the assignment of the class to the test object is based on a majority vote from the K nearest neighbours. There are three important factors which need to be considered while using the KNN algorithm:

a) The set of stored records
b) A distance metric to compute the distance between two objects
c) K, the number of neighbours being taken into consideration.

Many people in the machine learning do not prefer KNN, citing the fact that it is very simple, however despite being one of the simplest machine learning algorithms, KNN is highly effective in producing the results.

KNN being a lazy algorithm does not use any training points or generalization, which means there is no explicit training phase, which results in a very fast-moving training phase. Lack of generalization means that all the training data is retained even during the testing phase.

One of the main reasons why we chose KNN for our project is because KNN works very efficiently for multi modal classes, i.e., when the output feature has many classes

Support Vector Machine:

Support vector machine is a classification algorithm that has quickly become popular. In a two class learning task the aim of a support vector machine is to identify a classification function that separates the members of the two classes. The classification function can be obtained geometrically. In a linearly separable case, the function is a separating hyper plane.

Many such hyper planes may exist and the best hyper plane that is considered is the hyper plane that maximizes the distance between the two classes, this distance is known as margin,

The reason behind choosing the largest margin is to make sure the classification works perfectly even for the testing data.

One of the main reason why we chose Support vector machine is its highly robust nature and high efficiency, also SVM uses the kernel, which implicitly contains a non-linear transformation. So, prior human knowledge of whether the data is linearly separable or not is not necessary.

Random Forest:

Decision trees are very simple to implement. However, they are outperformed by number of other models. Random Forest algorithm improves the performance of Decision tree. It is a non-parametric supervised learning method used for classification and regression. It creates multiple trees just as the normal decision tree algorithm works. But however, at every split, only a random subset of features is considered instead of all the features. This reduces the correlation between the trees. It makes the trees more independent thereby reducing the variance.

Some of the reasons why we chose Random forest are:

a) Random forest performs an implicit feature selection and gives us a variable importance plot, through which we can understand how important is each feature. In our case, with the help of variable importance, we can identify which features are most important in estimating the quality of the wine.
b) Random forest requires no input preparation. It accepts inputs with all types, such as numeric, float, categorical and requires no scaling what so ever.

Neural Networks:

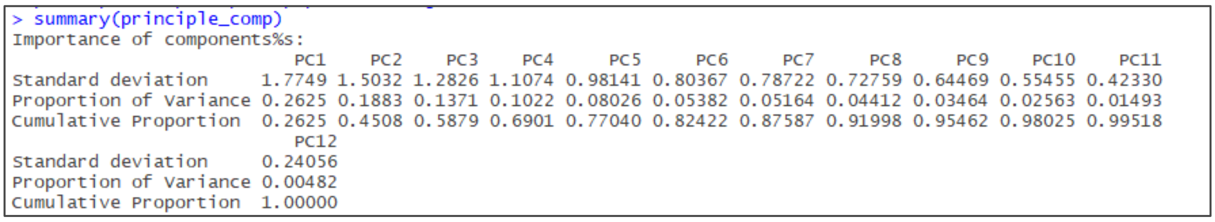
The idea of the neural networks has come from the working of our central nervous system. We can draw an analogy between the central nervous system and the neural networks. Like neurons in the nervous system, the neural networks have connected nodes. A neural network can have many hidden layers. The main function of the hidden layer is to transform the data in to some format that the output layer can use. A hidden layer contains many nodes. The nodes are analogous to neurons. Every node is multiplied by some weight. When we train the model, these weights are adjusted by the neural network to produce the correct result. If we consider the input layer, every node represents a word in the dataset. After multiplying the inputs with weights, the data passes through an activation function. It is the activation function that defines the output of every node. Every layer functions in the same way as discussed. Coming to the output layer, the number of nodes in the output layer will be equal to the number of features in the input dataset. The output layer nodes are also multiplied with weights, but they are passed through different activation function.

The main reason to choose Neural networks for our project is that neural networks give better classification results by considering the non-linear boundaries and we can also get rid of the problem of overfitting using the regularizer settings.

<u>EXPLORATORY ANALYSIS:</u>

Before training the machine learning models on a dataset we should do some exploratory analysis. I have presented some of the exploratory analysis plots in the Appendix. There might be some features in the dataset that will be strongly correlated with each other. Including all those redundant features does not add any information. If we include all the features, it means that we are dealing with large feature space which has a negative impact on the model's accuracy due to the curse of dimensionality. Therefore, we have done Principal Component Analysis for the feature extraction.

<u>Principal Component Analysis:</u>



```
> summary(principle_comp)
Importance of components%s:
                          PC1    PC2    PC3    PC4     PC5     PC6     PC7     PC8     PC9    PC10    PC11
Standard deviation     1.7749 1.5032 1.2826 1.1074 0.98141 0.80367 0.78722 0.72759 0.64469 0.55455 0.42330
Proportion of Variance 0.2625 0.1883 0.1371 0.1022 0.08026 0.05382 0.05164 0.04412 0.03464 0.02563 0.01493
Cumulative Proportion  0.2625 0.4508 0.5879 0.6901 0.77040 0.82422 0.87587 0.91998 0.95462 0.98025 0.99518
                          PC12
Standard deviation     0.24056
Proportion of Variance 0.00482
Cumulative Proportion  1.00000
```

Fig 5: Summary of principle component analysis

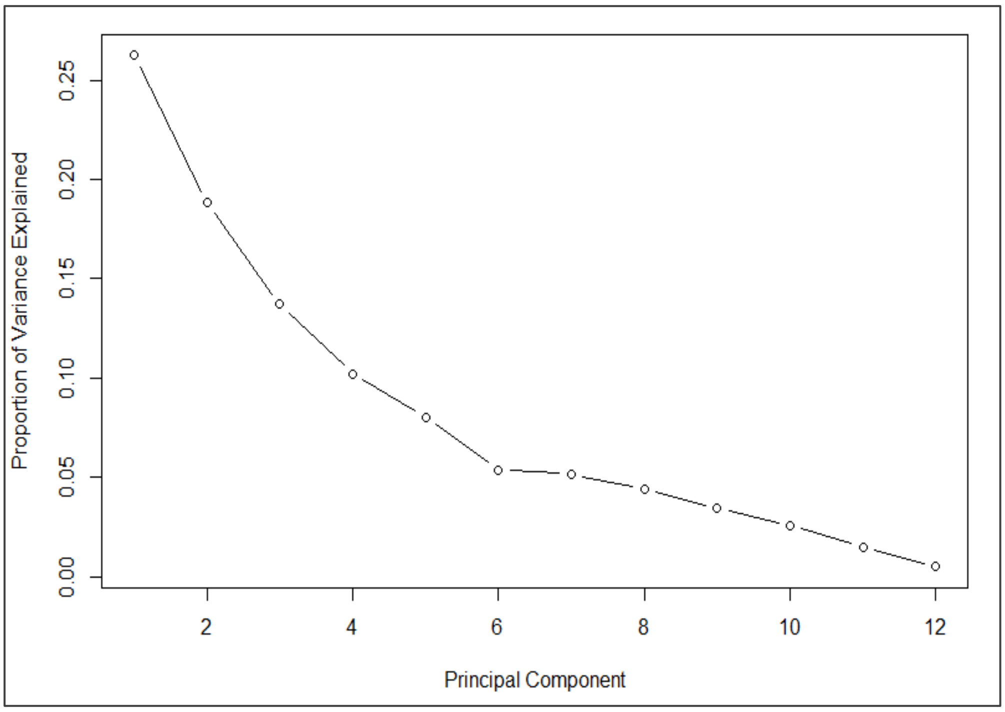


Fig 6: Screen plot for principle component

The main objective is to include all the principal components which explain the maximum variance. Higher the explained variance, higher is the information contained in the components. From the above graph and screenshot, we can see that first principal component explains 26.25% variance, second principal component explains 18.83% and so on. The graph shows that all the input features explains considerable amount of variance. Hence, we have retained all the features while training and tuning the models.

ANALYSIS & RESULTS OF PREDICTIVE MODELING:

For predictive modeling, we made use of the caret package in R. Caret package offers a lot of inbuilt functions which make training and testing easy.

a) K Nearest Neighbours:

For K nearest neighbours, we took into consideration 5 different K values, which are 3, 5, 7, 9, 11 and made use of Cos, Gaussian and rectangular kernel and two types of distances, Manhattan and Euclidean. The below graph summarizes the different K values, kernel functions and distances.

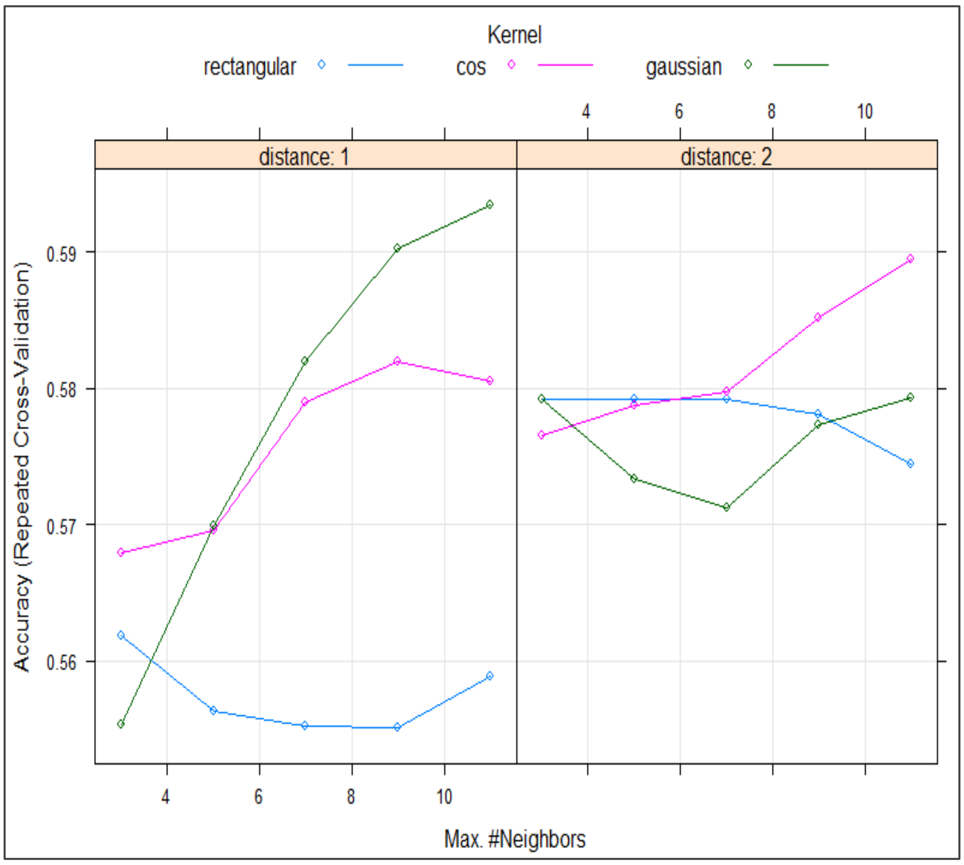


Fig7: Summary graph for K nearest neighbours

From the above graph we can conclude that the **optimum number of neighbours** that need to be considered for the highest accuracy **is 11** and similarly **the most efficient kernel is Gaussian,** and the most **effective distance is Manhattan.** The model, when implemented on the **testing** set produced an **accuracy of 60.6%** and a **kappa value** of **0.3722**

A Confusion matrix is generated for the test set and is presented in the Appendix

b) <u>Support Vector Machine:</u>

For support vector machine, we tried to find the optimal values of C and sigma, which are the margin distance and inverse kernel width respectively. The following graph gives the accuracy for various C and sigma values. The range for C that we used was 1-10 and the range for sigma was 0.1 to 1, with increments of 0.1

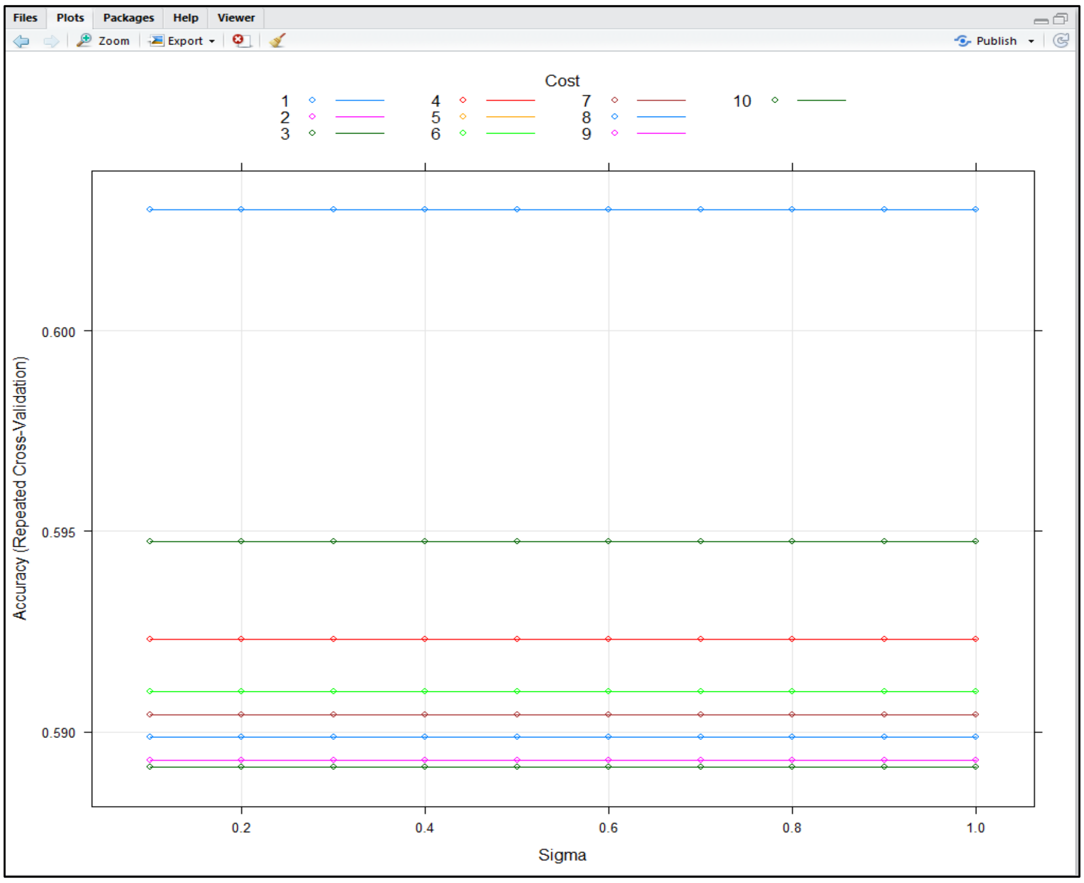


Fig 8: Support vector machine summary graph

Even though from the graph we exactly cannot predict the most optimum values, with the help of the **best tune** part of our training model we can identify the optimum values of C and sigma. The **optimum value for C** was **1** and for **sigma** it was **1.** When the model was tested, it gave an **accuracy of 64.92%,** with a **kappa value** of **0.3946.**

A Confusion matrix is generated for the test set and is presented in the Appendix.

c) <u>Random Forest:</u>

For random forest, we try to find out the optimum value of mtry, which is the number of variables that are randomly selected as candidates at each split. Here we have passed a mtry of 1 to 11 into the train function's tuneGrid argument.

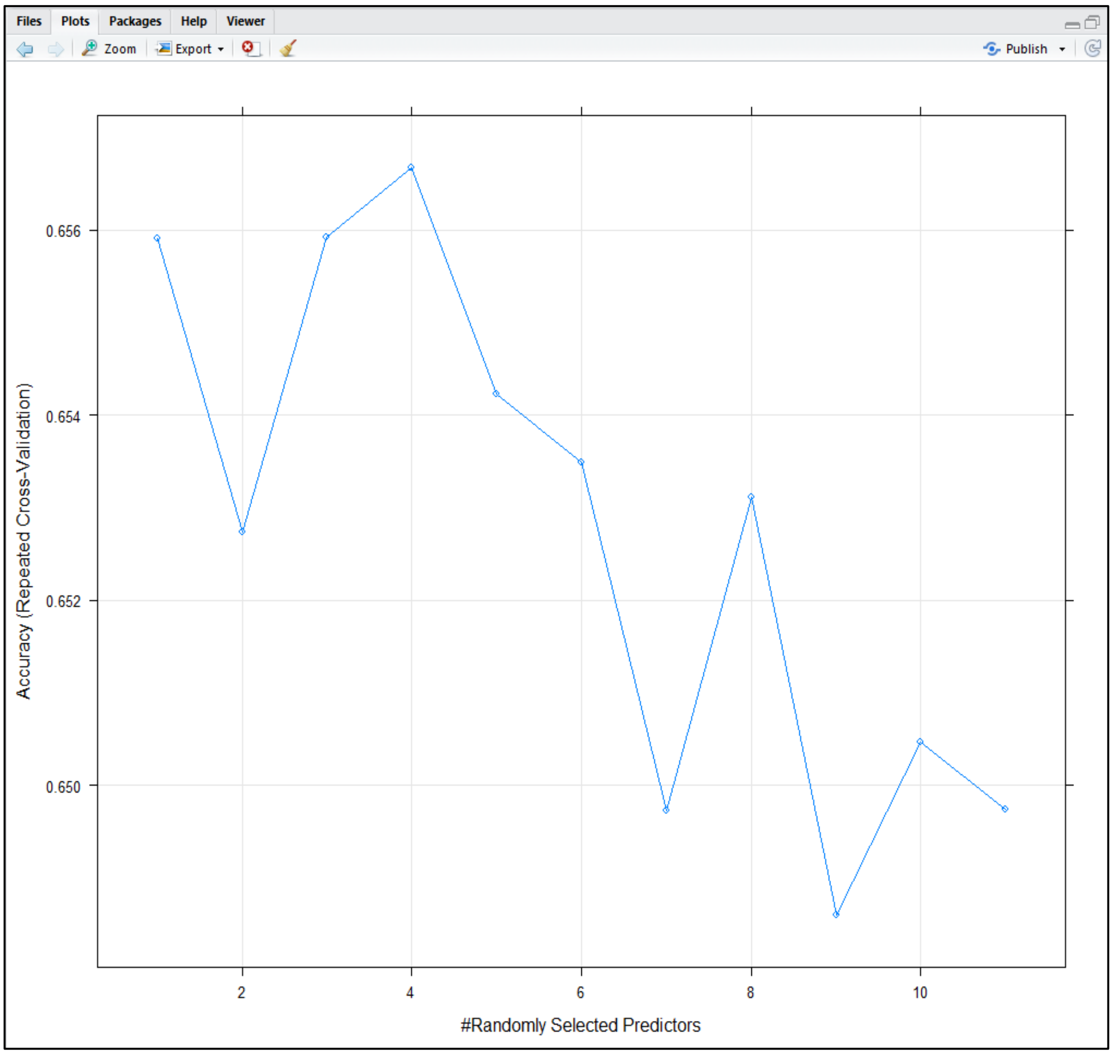


Fig 9: Random Forest summary

From the above graph we can conclude that an **mtry** value of **4** gives the highest accuracy**.** After implementing the trained random forest function **on the testing set**, we observed an **accuracy value of 67.92%** and a **kappa value** of **0.4633**
A Confusion matrix is generated for the test set and is presented in the Appendix.

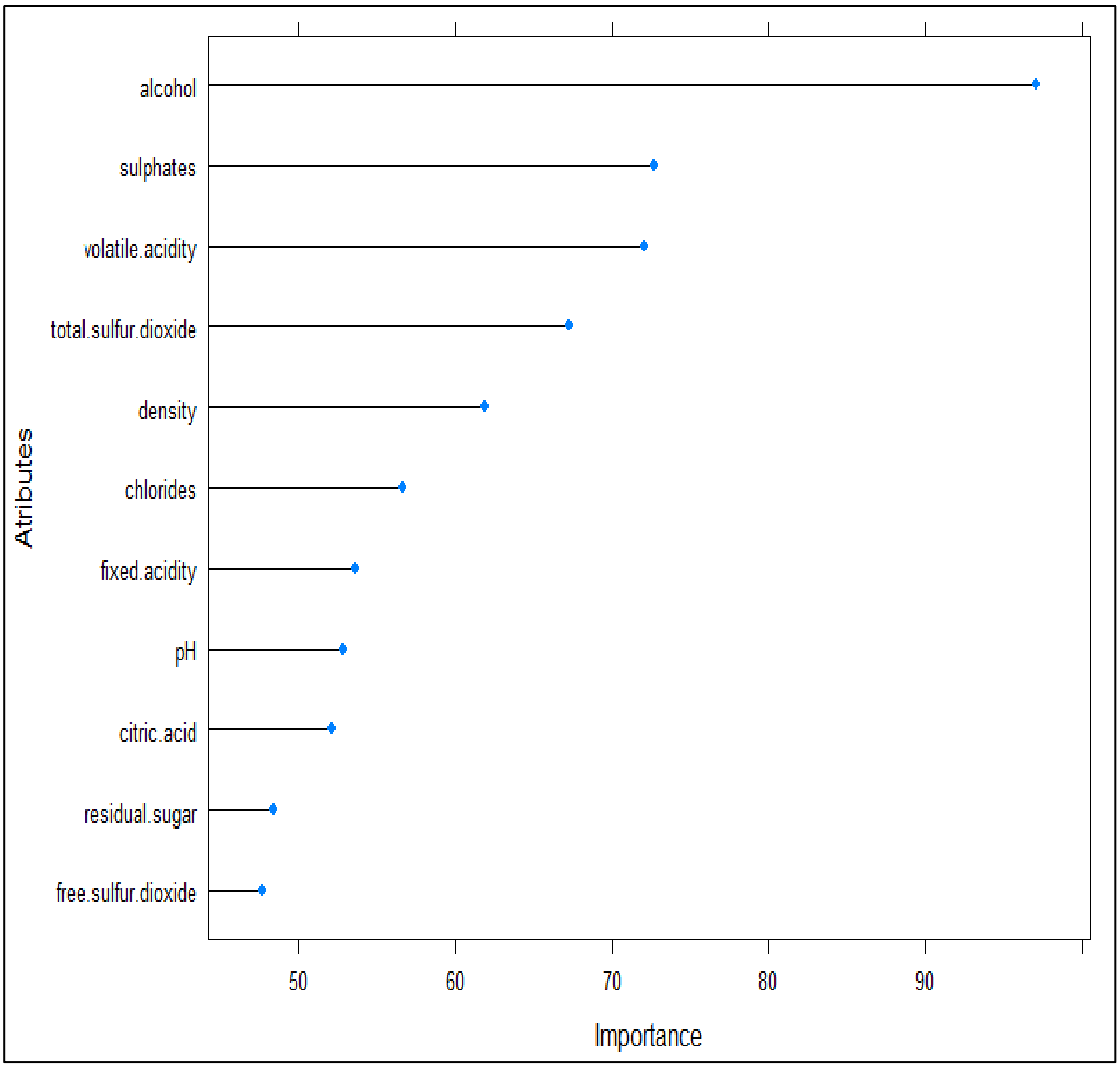


Fig 10: Variable importance graph

The above figure shows the variable importance graph. It shows the importance of each feature in determining the quality of the wine. As we can see from the graph, alcohol plays a major role in predicting the quality of wine and free.sulphur.dioxide has the least importance.

d) <u>Neural Networks:</u>

For neural networks we try to identify the optimal values of the number of hidden layers and the weight decay which produce the highest accuracy:

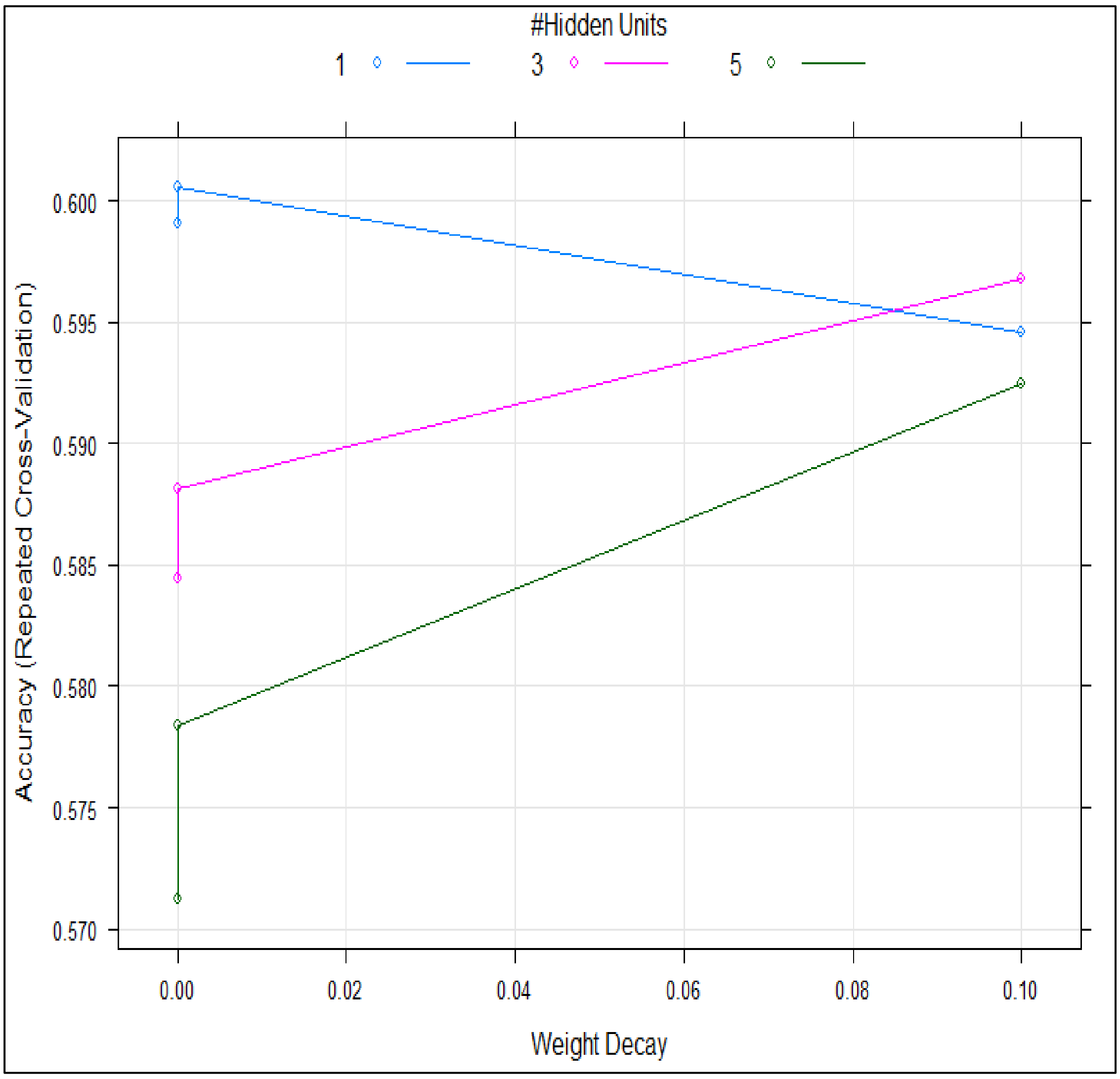
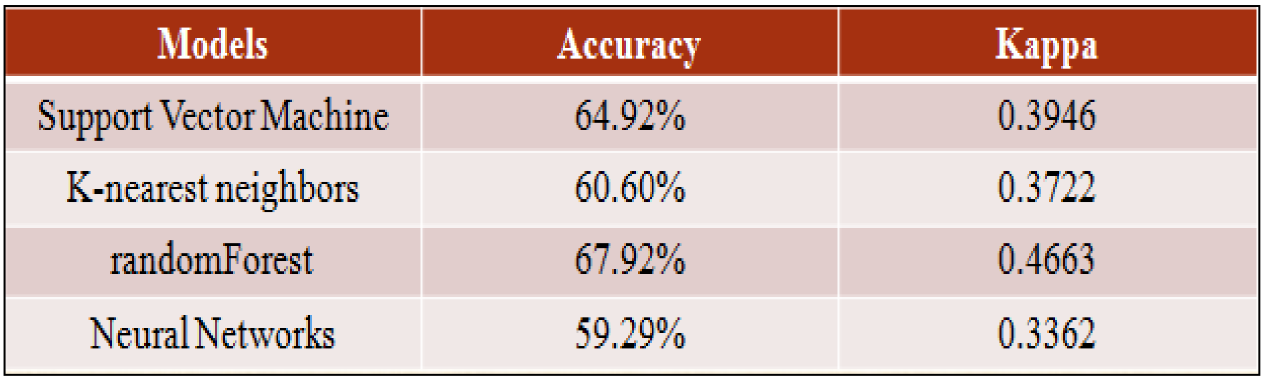


Fig 11: Neural network summary

From the graph we can see that an **optimal value** of **1 hidden layer** and **0.0001 weight decay** produced the highest accuracy. We observed an **accuracy** of **59.29%** and a **kappa value of 0.3362**, when the model was tested **on the testing set.**
A Confusion matrix is generated for the test set and is presented in the Appendix.

CONCULSION:



Models	Accuracy	Kappa
Support Vector Machine	64.92%	0.3946
K-nearest neighbors	60.60%	0.3722
randomForest	67.92%	0.4663
Neural Networks	59.29%	0.3362

Fig 12: Summary of all the models

The benchmark accuracy (%) and the benchmark kappa values for the red wine according to an academic study are 62.4±0.4 and 38.7±0.7 respectively. By comparing the accuracy values of our models with the benchmark values we came to a conclusion that the **randomForest** model performed well when compared to all the other models. However, all the models did not perform well in predicting the quality of wines of highest and lowest wines. This may be due to the class imbalances present in the dataset.

BIBILIOGRAPHY:

1) An introduction to statistical learning with applications in R, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani.
2) Top 10 Algorithms for data mining- Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg.
3) P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.
4) Random Forest: A Review Eesha Goel, Er. Abhilasha- International Journal of Advanced Research in Computer Science and Software Engineering.
5) A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges- Janmenjoy Nayak, Bighnaraj Naik* and H. S. Behera.

APPENDIX:

Before we implemented our models, we performed exploratory analysis on our data, below are the figures that were obtained for exploratory analysis:

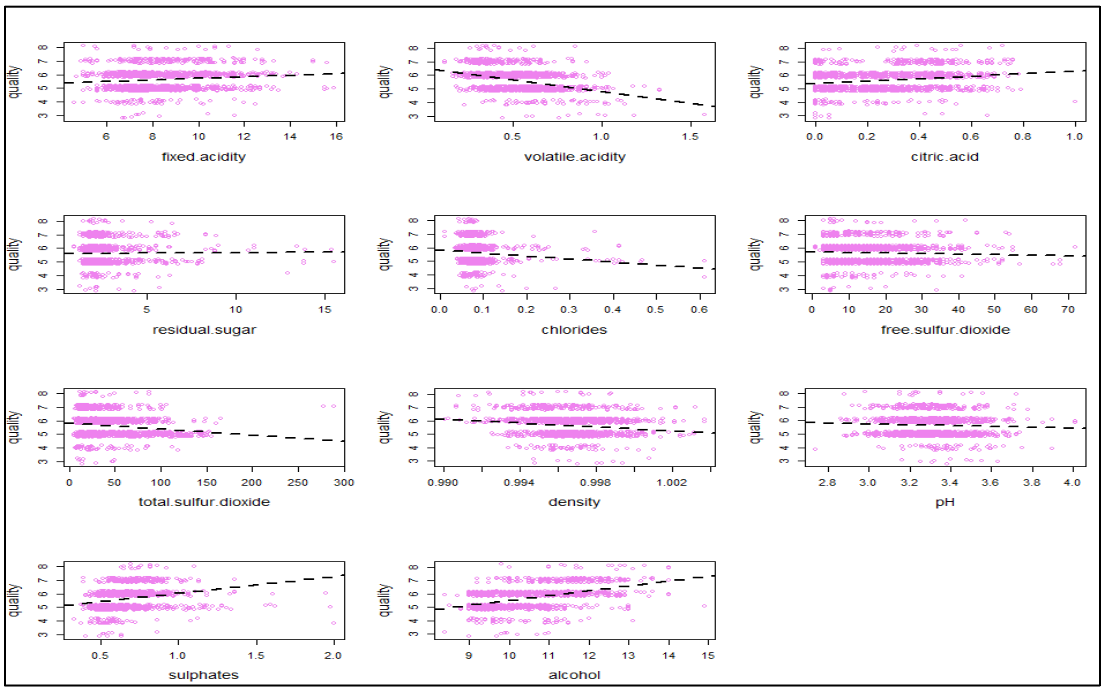


Fig 13: Plots between output variable and input features

The black line on each graph is the linear regression of quality variable as a function of the plot's predictor variable. From the above graphs it is evident that volatile **acidity, sulphates and alcohol** have the **strongest relationship with quality** and **Free sulphur dioxide and residual sugar** have the **weakest relationship with quality**.

To see if there exists a correlation between the various input features, we generated

Pearson correlation matrix:

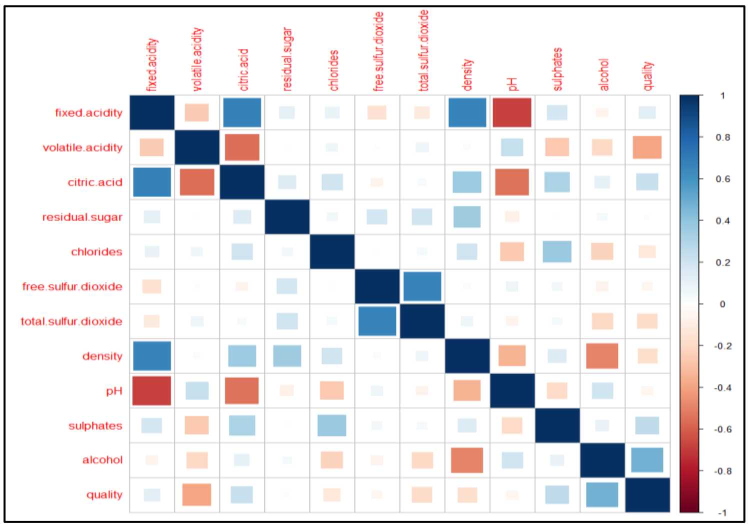
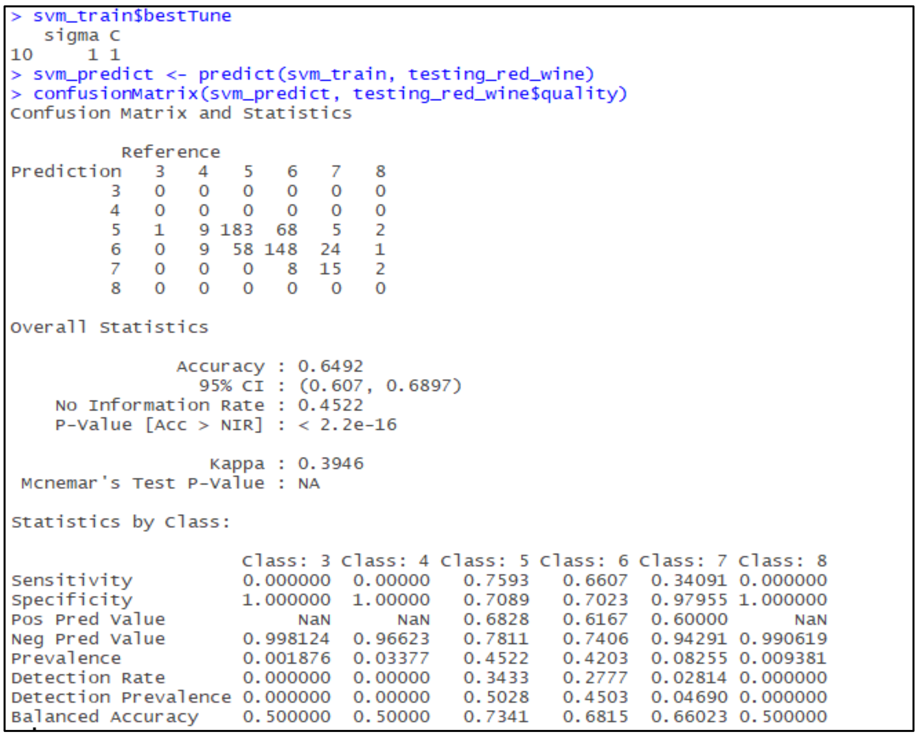


Fig 14: Correlation matrix

From the above correlation plot, we can see that **pH, residual.sugar and free.sulpur.dioxide** are weakly correlated with quality.

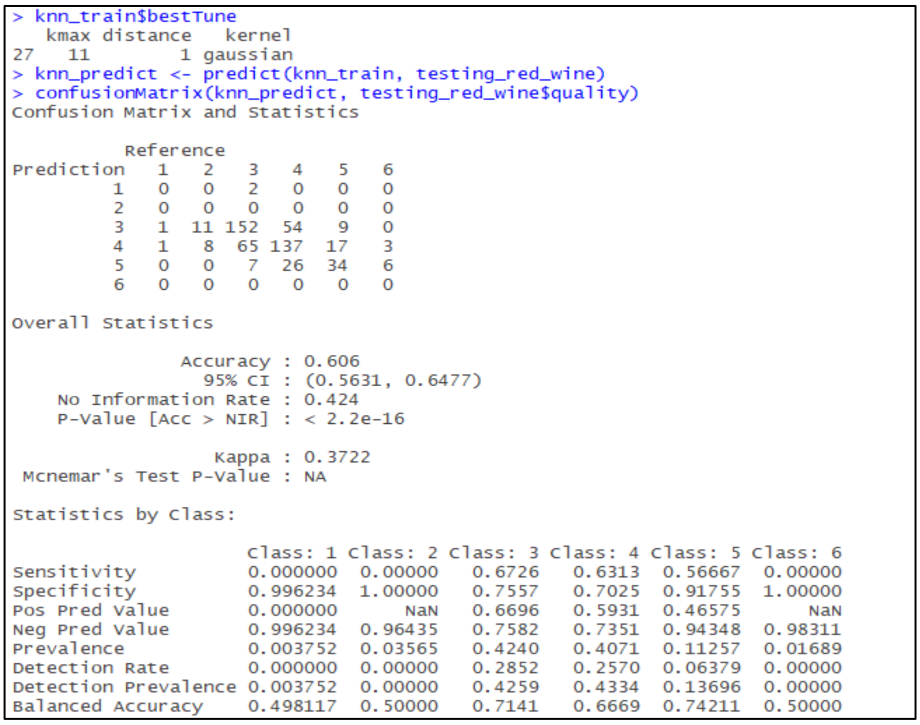
After training our models, we generated confusion matrices of the test dataset for each model:

<u>Confusion Matrices of the Predictive Models:</u>



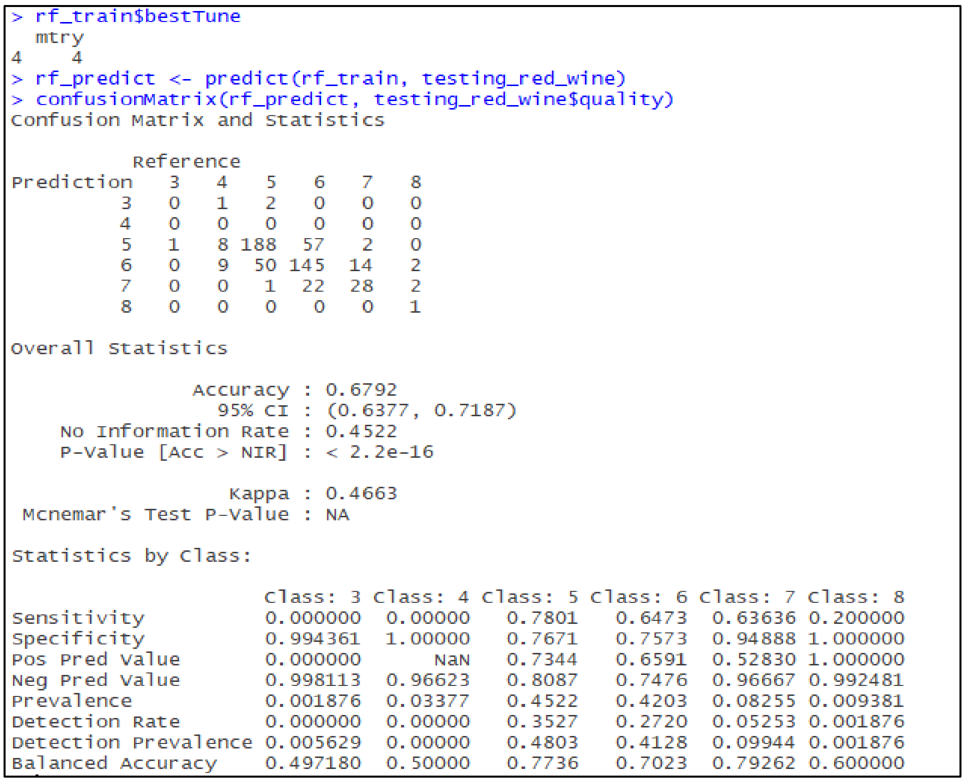
```
> svm_train$bestTune
   sigma C
10     1 1
> svm_predict <- predict(svm_train, testing_red_wine)
> confusionMatrix(svm_predict, testing_red_wine$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8
         3   0   0   0   0   0   0
         4   0   0   0   0   0   0
         5   1   9 183  68   5   2
         6   0   9  58 148  24   1
         7   0   0   0   8  15   2
         8   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.6492
                 95% CI : (0.607, 0.6897)
    No Information Rate : 0.4522
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3946
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000  0.00000   0.7593   0.6607  0.34091 0.000000
Specificity          1.000000  1.00000   0.7089   0.7023  0.97955 1.000000
Pos Pred Value            NaN      NaN   0.6828   0.6167  0.60000      NaN
Neg Pred Value       0.998124  0.96623   0.7811   0.7406  0.94291 0.990619
Prevalence           0.001876  0.03377   0.4522   0.4203  0.08255 0.009381
Detection Rate       0.000000  0.00000   0.3433   0.2777  0.02814 0.000000
Detection Prevalence 0.000000  0.00000   0.5028   0.4503  0.04690 0.000000
Balanced Accuracy    0.500000  0.50000   0.7341   0.6815  0.66023 0.500000
```

Fig 15: SVM confusion matrix



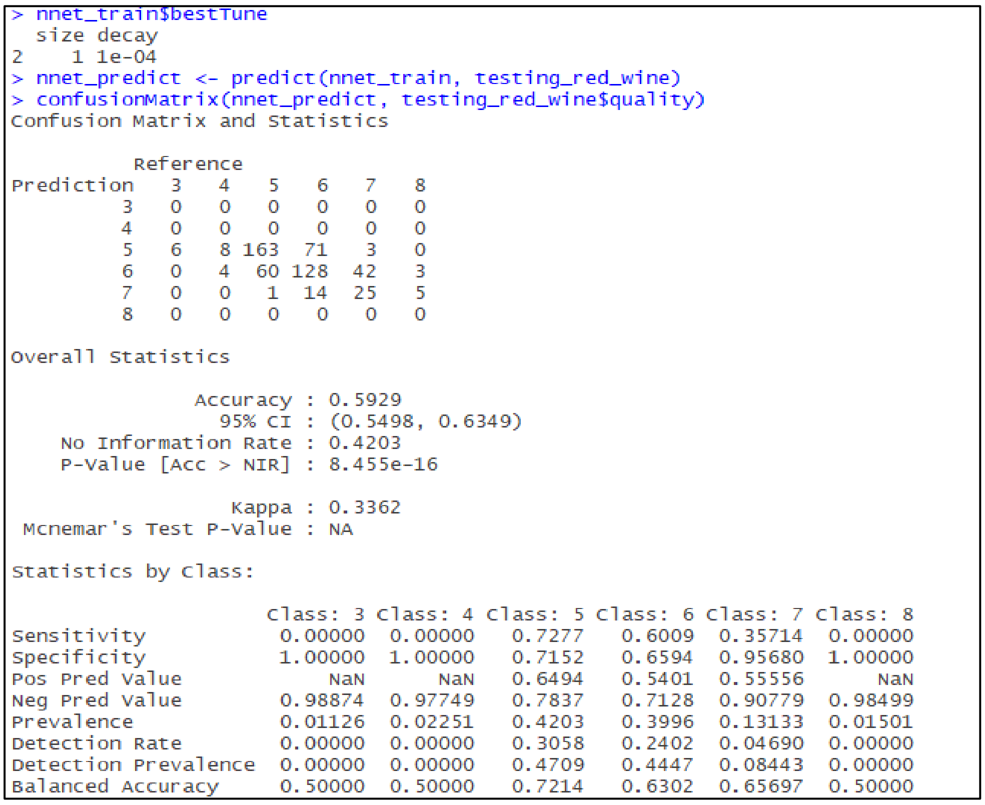
```
> knn_train$bestTune
   kmax distance   kernel
27   11        1 gaussian
> knn_predict <- predict(knn_train, testing_red_wine)
> confusionMatrix(knn_predict, testing_red_wine$quality)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4   5   6
         1   0   0   2   0   0   0
         2   0   0   0   0   0   0
         3   1  11 152  54   9   0
         4   1   8  65 137  17   3
         5   0   0   7  26  34   6
         6   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.606
                 95% CI : (0.5631, 0.6477)
    No Information Rate : 0.424
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3722
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
Sensitivity          0.000000  0.00000   0.6726   0.6313  0.56667  0.00000
Specificity          0.996234  1.00000   0.7557   0.7025  0.91755  1.00000
Pos Pred Value       0.000000      NaN   0.6696   0.5931  0.46575      NaN
Neg Pred Value       0.996234  0.96435   0.7582   0.7351  0.94348  0.98311
Prevalence           0.003752  0.03565   0.4240   0.4071  0.11257  0.01689
Detection Rate       0.000000  0.00000   0.2852   0.2570  0.06379  0.00000
Detection Prevalence 0.003752  0.00000   0.4259   0.4334  0.13696  0.00000
Balanced Accuracy    0.498117  0.50000   0.7141   0.6669  0.74211  0.50000
```

Fig 16: KNN confusion matrix



```
> rf_train$bestTune
  mtry
4    4
> rf_predict <- predict(rf_train, testing_red_wine)
> confusionMatrix(rf_predict, testing_red_wine$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8
         3   0   1   2   0   0   0
         4   0   0   0   0   0   0
         5   1   8 188  57   2   0
         6   0   9  50 145  14   2
         7   0   0   1  22  28   2
         8   0   0   0   0   0   1

Overall Statistics
                                          
               Accuracy : 0.6792          
                 95% CI : (0.6377, 0.7187)
    No Information Rate : 0.4522          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.4663          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000  0.00000   0.7801   0.6473  0.63636 0.200000
Specificity          0.994361  1.00000   0.7671   0.7573  0.94888 1.000000
Pos Pred Value       0.000000      NaN   0.7344   0.6591  0.52830 1.000000
Neg Pred Value       0.998113  0.96623   0.8087   0.7476  0.96667 0.992481
Prevalence           0.001876  0.03377   0.4522   0.4203  0.08255 0.009381
Detection Rate       0.000000  0.00000   0.3527   0.2720  0.05253 0.001876
Detection Prevalence 0.005629  0.00000   0.4803   0.4128  0.09944 0.001876
Balanced Accuracy    0.497180  0.50000   0.7736   0.7023  0.79262 0.600000
```

Fig 17: Random Forest confusion matrix



```
> nnet_train$bestTune
  size decay
2    1 1e-04
> nnet_predict <- predict(nnet_train, testing_red_wine)
> confusionMatrix(nnet_predict, testing_red_wine$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8
         3   0   0   0   0   0   0
         4   0   0   0   0   0   0
         5   6   8 163  71   3   0
         6   0   4  60 128  42   3
         7   0   0   1  14  25   5
         8   0   0   0   0   0   0

Overall Statistics
                                          
               Accuracy : 0.5929          
                 95% CI : (0.5498, 0.6349)
    No Information Rate : 0.4203          
    P-Value [Acc > NIR] : 8.455e-16       
                                          
                  Kappa : 0.3362          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity           0.00000  0.00000   0.7277   0.6009  0.35714  0.00000
Specificity           1.00000  1.00000   0.7152   0.6594  0.95680  1.00000
Pos Pred Value            NaN      NaN   0.6494   0.5401  0.55556      NaN
Neg Pred Value        0.98874  0.97749   0.7837   0.7128  0.90779  0.98499
Prevalence            0.01126  0.02251   0.4203   0.3996  0.13133  0.01501
Detection Rate        0.00000  0.00000   0.3058   0.2402  0.04690  0.00000
Detection Prevalence  0.00000  0.00000   0.4709   0.4447  0.08443  0.00000
Balanced Accuracy     0.50000  0.50000   0.7214   0.6302  0.65697  0.50000
```

Fig 18: Neural Network confusion matrix