Wine Project

Leena Elsharkawy

14/06/2020

Contents

1. Introduction

Data Science is a field of studies which combines scientific methods and statistical analysis to extract information from data. The extracted information can then be cleaned up and presented or can be utilized to form inferences. The utilization of data to construct models which can then be used for prediction is called machine learning. Machine learning is a subset of data science in which a "training set" is used to predict outcomes. This can be quite useful for filtering, recommending, and predicting.

1.1. Overview

In this project, machine learning principals are used to predict the quality for a data set of 6497 Portuguese "Vinho Verde" wines. The wine data is downloaded from the UCI Machine Learning Repository and is then extracted, tidied, and divided into sets. The information initially downloaded is messy, therefore needs to be "cleaned up" and restructured to make it easily readable. Originally the data is collected for red (1599) and white (4898) wines seperately, thus they are inspected seperately but the decision is made to predict the quality together despite the color. Nonetheless, the wine color is added to the already existing 12 variables (including the quality) to account for the role of color on the quality. In further sections, it will be seen that the information provided on the wines is its physiocochemical make up including the different values for varying acids and components involved. The 11 attributes which play a role in the prediction of wine quality are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content which are all values determined via testing. The final attribute is the one added by this study which is wine color, it is important not to neglect this bit of information when combining the data.

Post inspection, the data is divided into **train_set** and **test_set** or the validation set, where the validation set is not used for any training but rather only to validate the models. The train set is then used to train various models in order to predict the wine quality in the test set with the lowest root mean square error (RMSE) possible, RMSE compares predicted quality to the true quality for the test set. The RMSE takes the mean of the error squared for each data point in the set. The RMSE is minimized for methods via the tuning process to optimize the parameters and the final values from each method. The goal is to minimize the RMSE as much as possible to provide a more accurate prediction for the wine quality. In this project, more than 9 different methods are carried out to inspect the effect of various methods on the prediction as well as to combine the best methods to provide an even better prediction. The methods can be divided into 4 categories: linear regression, tree-based models, ensemble models, and support vector machine. The resulting RMSE is calculated and documented for each method then saved into a table for comparison. From this final **results** table, it is evident that the best way to predict the wine quality is to combine the methods with the lowest two RMSE values.

2. Methodology

The first step to start running the project after running the required libraries is to obtain the data and prepare it for utilization. There were 6 steps necessary to prepare the data for the movie lens project and they are as follow:

1. Download
2. Extract
3. Tidy
4. Inspect
5. Visualize
6. Partition

2.1. Data Preparation

First the data that will be utilized was downloaded from the URL link. Once the files are downloaded the next step is to read and extract the necessary data. After having obtained the data it is important to clean it or tidy it up. This step makes the information useful to both the user and the program. In this case, this was done by combing the red and white wine data, detecting any NA's and removing them, and removing outlier data (which comes after inspection). Additionally, the fator **type** is added to enusre that the wine type is taken into account. Once the data is rearranged in a way that makes it easy to read, it is then inspected and visualized.

```
#Download and rearrange red wine data
url1<-"https://archive.ics.uci.edu/ml/machine-learning-databases
    /wine-quality/winequality-red.csv"
dl1<-tempfile()
download.file(url1,dl1)
red<-read.csv(dl1, sep=";")

#Download and rearrange white wine data
url2<-"https://archive.ics.uci.edu/ml/machine-learning-databases
    /wine-quality/winequality-white.csv"
dl2<-tempfile()
download.file(url2,dl2)
white<-read.csv(dl2, sep=";")

#Add a column characterising the type of wine
red<- red%>%mutate(type = "red")
white<- white%>%mutate(type = "white")

#Combine the wines
```

```r
wine<-rbind(red,white)

#change type to a factor
wine$type<-as.factor(wine$type)

#Check for NA's and remove if they exist
for (x in 1:12){print(sum(is.na(wine[,x]))) }
wine<-na.omit(wine)
```

2.2. Data Inspection and Visualization

The first method of inspection was to see the structure of the data, which allows for undertanding of the various variables measured for the wines. The number of observations was examined, as well as all the column names/variables available. It showed that there were nearly 6500 different wines withalmost 5000 being white.

```
[1] This is the structure for the wine data set:

'data.frame':   6497 obs. of  13 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 0.07
 $ 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 ...
 $ type                : Factor w/ 2 levels "red","white": 1 1 1 1 1 1 1 1 1 1 ...
```

Since there are two different kinds of wine included it only makes sense to look at the types individually before looking at the data as a whole, figures 1 and 2 show the quality vs the wine colors.Therefore we shall inspect the distribution of quality for each type then see how the different variables correlate to quality for red and white wines, shown in tables 1 and 2. Through the inspection of the data we begin to realize that there may be outliers.

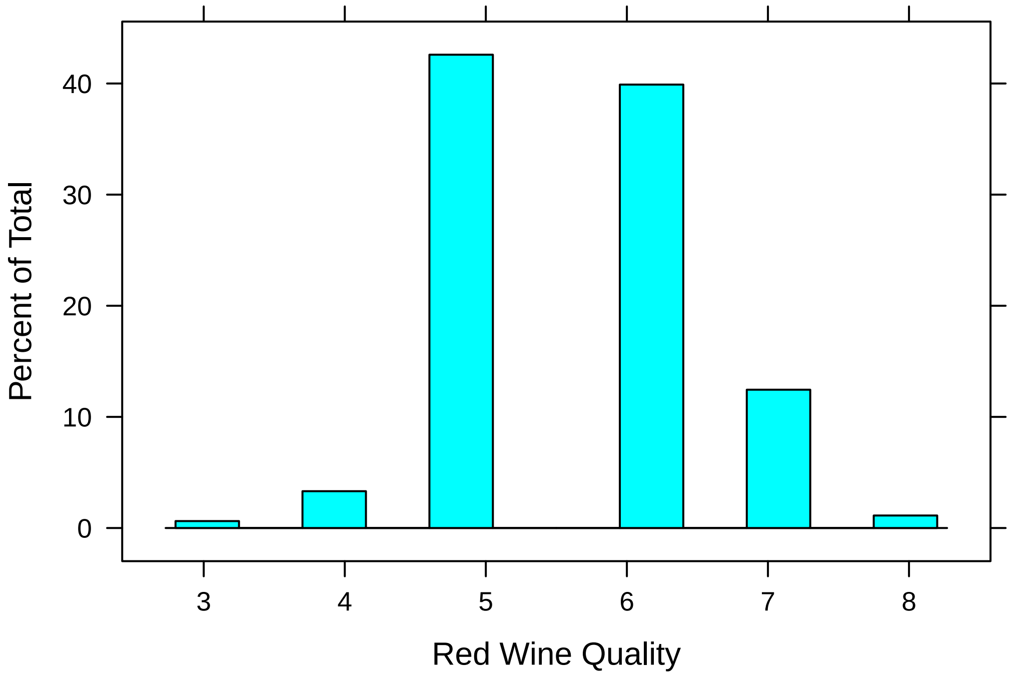

Figure 1: Quality Histogram for Red Wine

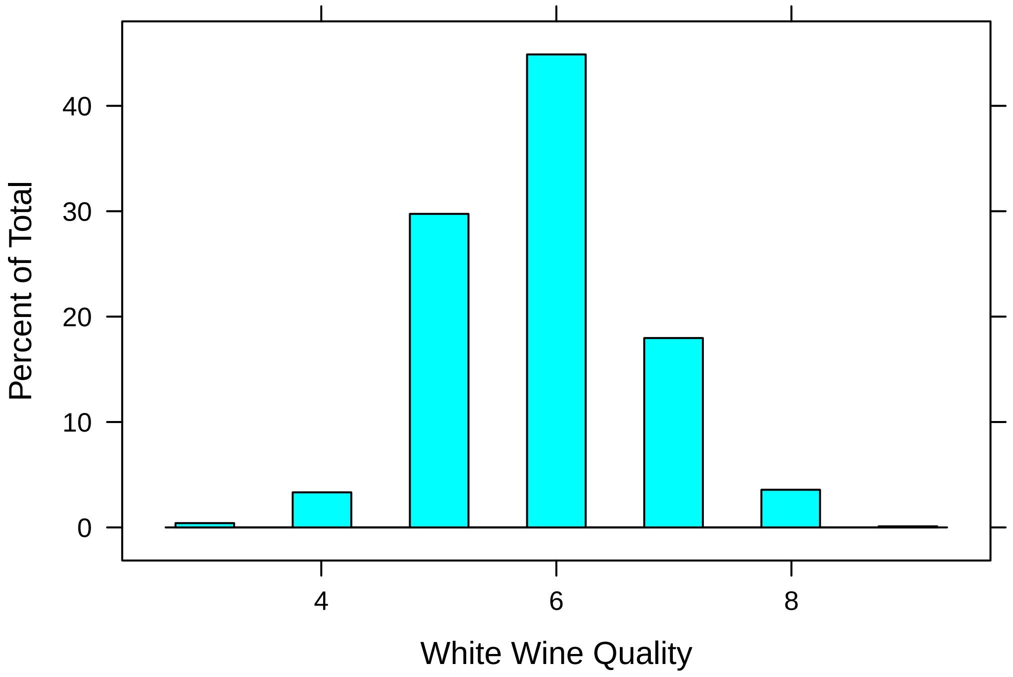

Figure 2: Quality Histogram for White Wine

Table 1: Average Variable Values for Wine Types

Variables	RED	WHITE
fixed.acidity	8.3196373	6.8547877
volatile.acidity	0.5278205	0.2782411
citric.acid	0.2709756	0.3341915
residual.sugar	2.5388055	6.3914149
chlorides	0.0874665	0.0457724
free.sulfur.dioxide	15.8749218	35.3080849
total.sulfur.dioxide	46.4677924	138.3606574
density	0.9967467	0.9940274
pH	3.3111132	3.1882666
sulphates	0.6581488	0.4898469
alcohol	10.4229831	10.5142670
quality	5.6360225	5.8779094

Table 2: Variable Correlation for Wine Types

Variables	cor_coeff
fixed.acidity	-0.0767432
volatile.acidity	-0.2656995
citric.acid	0.0855317
residual.sugar	-0.0369805
chlorides	-0.2006655
free.sulfur.dioxide	0.0554631
total.sulfur.dioxide	-0.0413855
density	-0.3058579
pH	0.0195057
sulphates	0.0384854
alcohol	0.4443185
quality	1.0000000

Now that we have seen some distinctions between the two wines, we undertsand the importance of adding the wine type as a varriable to our dataset. Next, the data is visualized. Since there are many different variables attributing to the quality, they will be plotted on by one against the quality in figures 3-13.

Since there are outliers that are seen through the inspection and visualization step, it is best to determine if our hunch is true via the boxplot. We see that, it is true then we remove the data to make for more accurate predicitons.

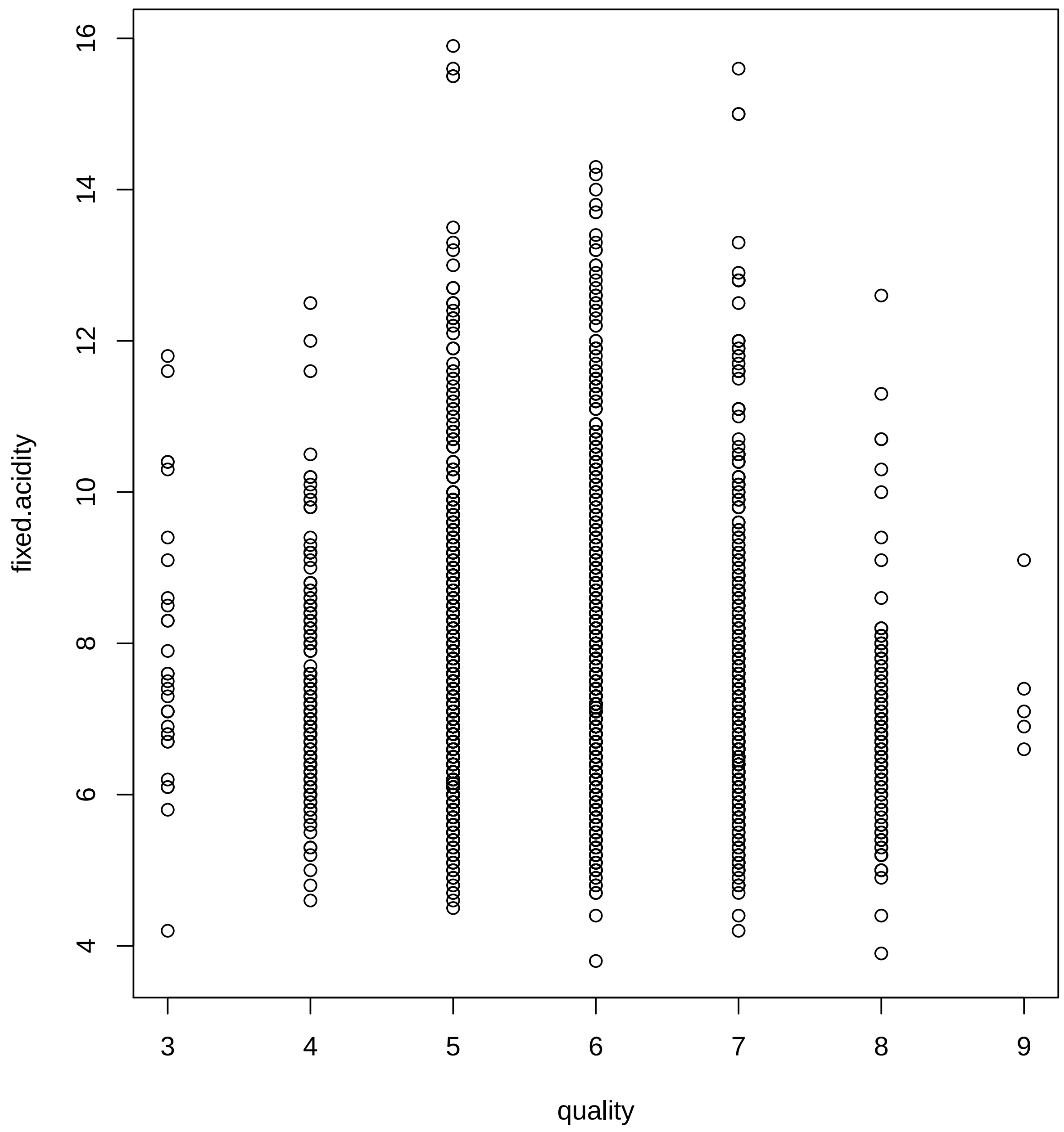

Figure 3: Variable Attributes vs. Quality

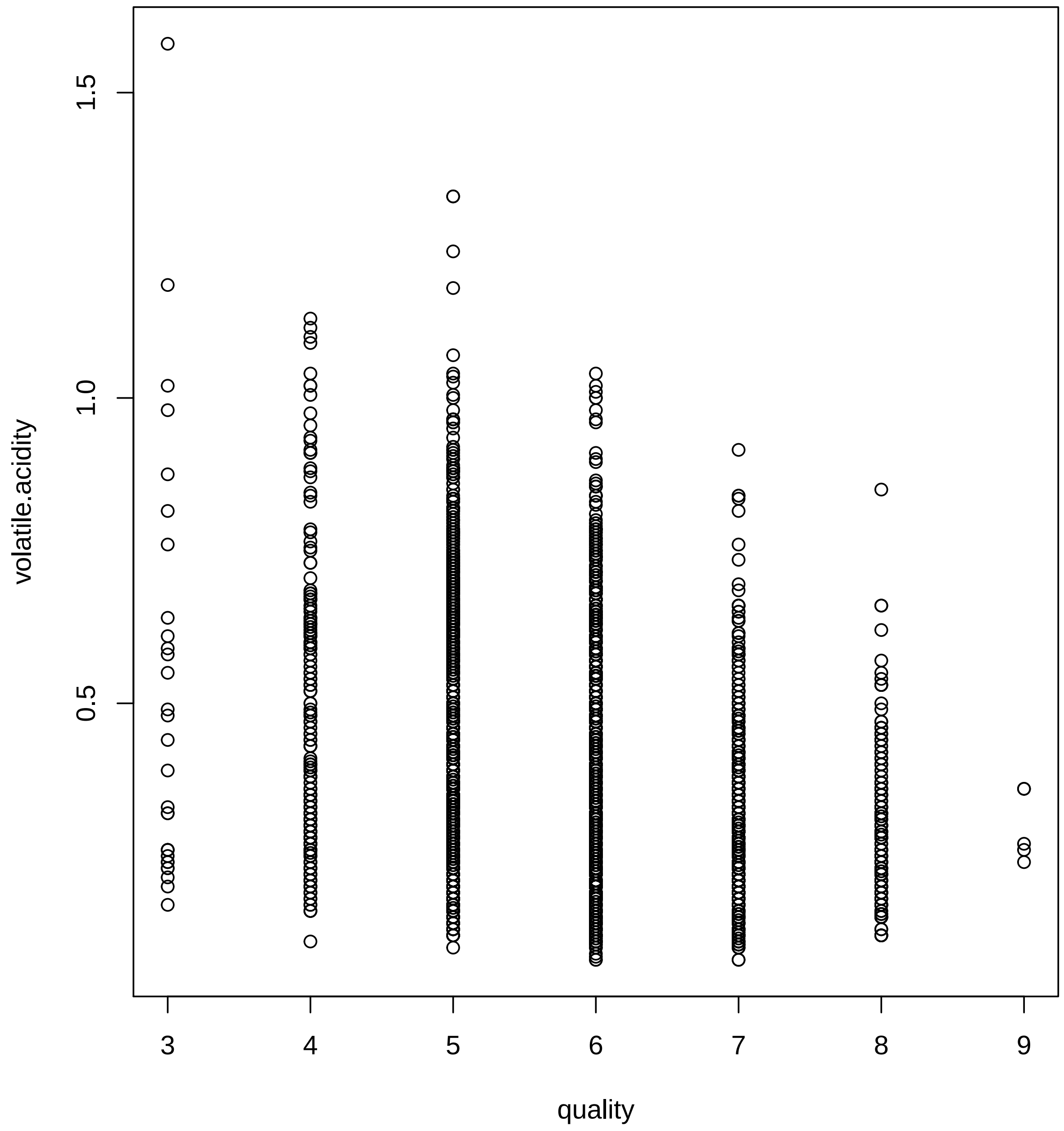

Figure 4: Variable Attributes vs. Quality

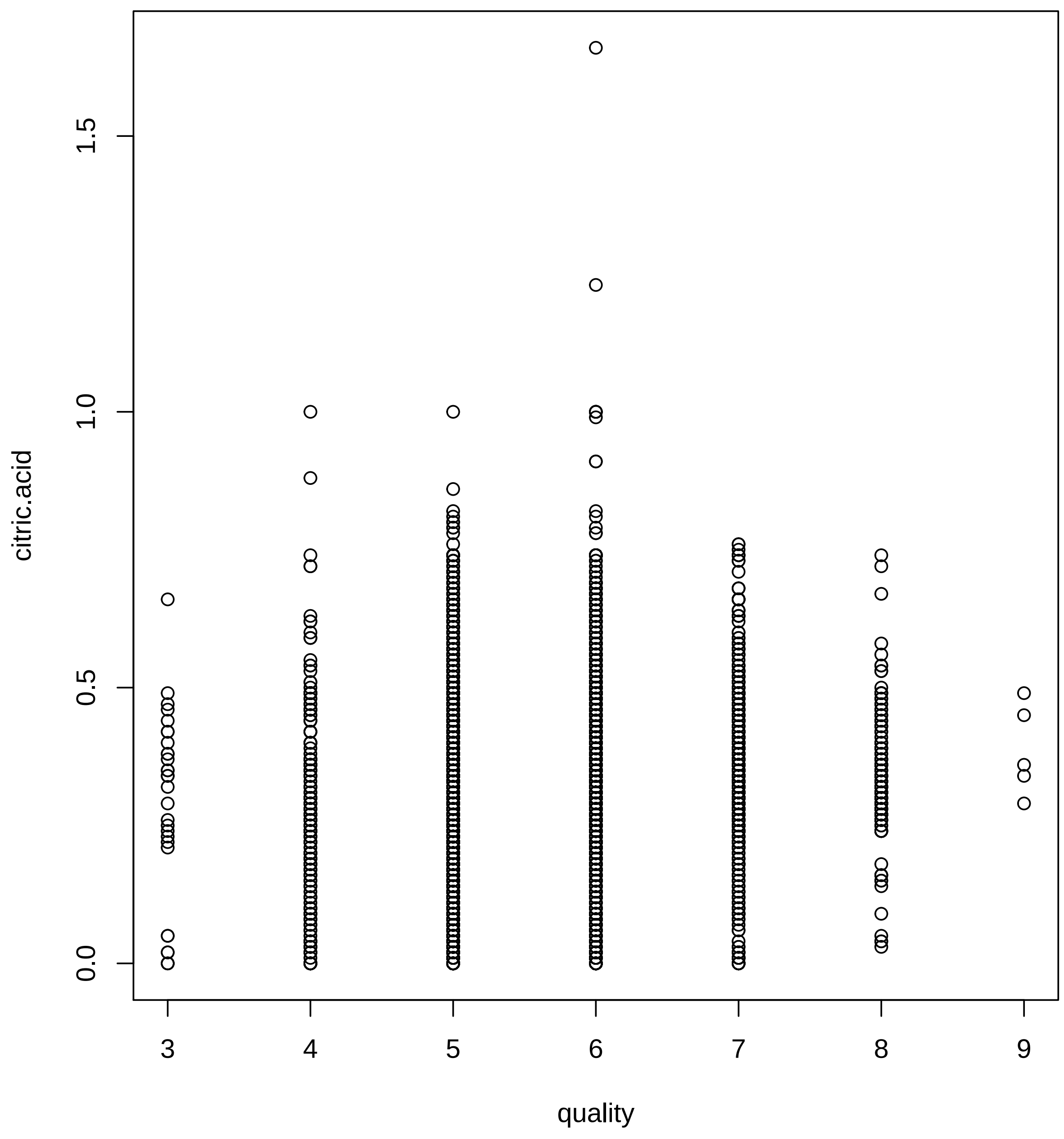

Figure 5: Variable Attributes vs. Quality

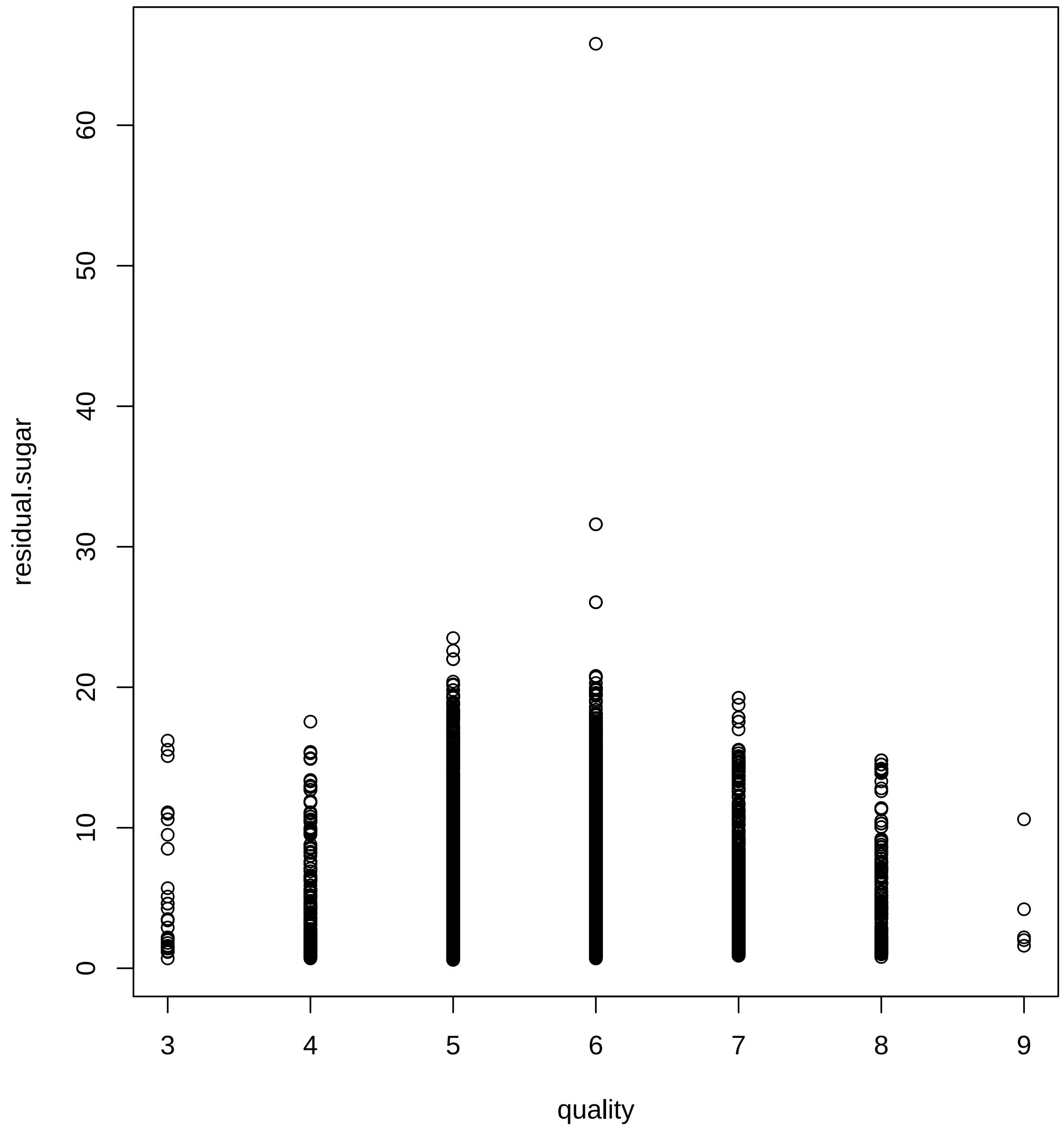

Figure 6: Variable Attributes vs. Quality

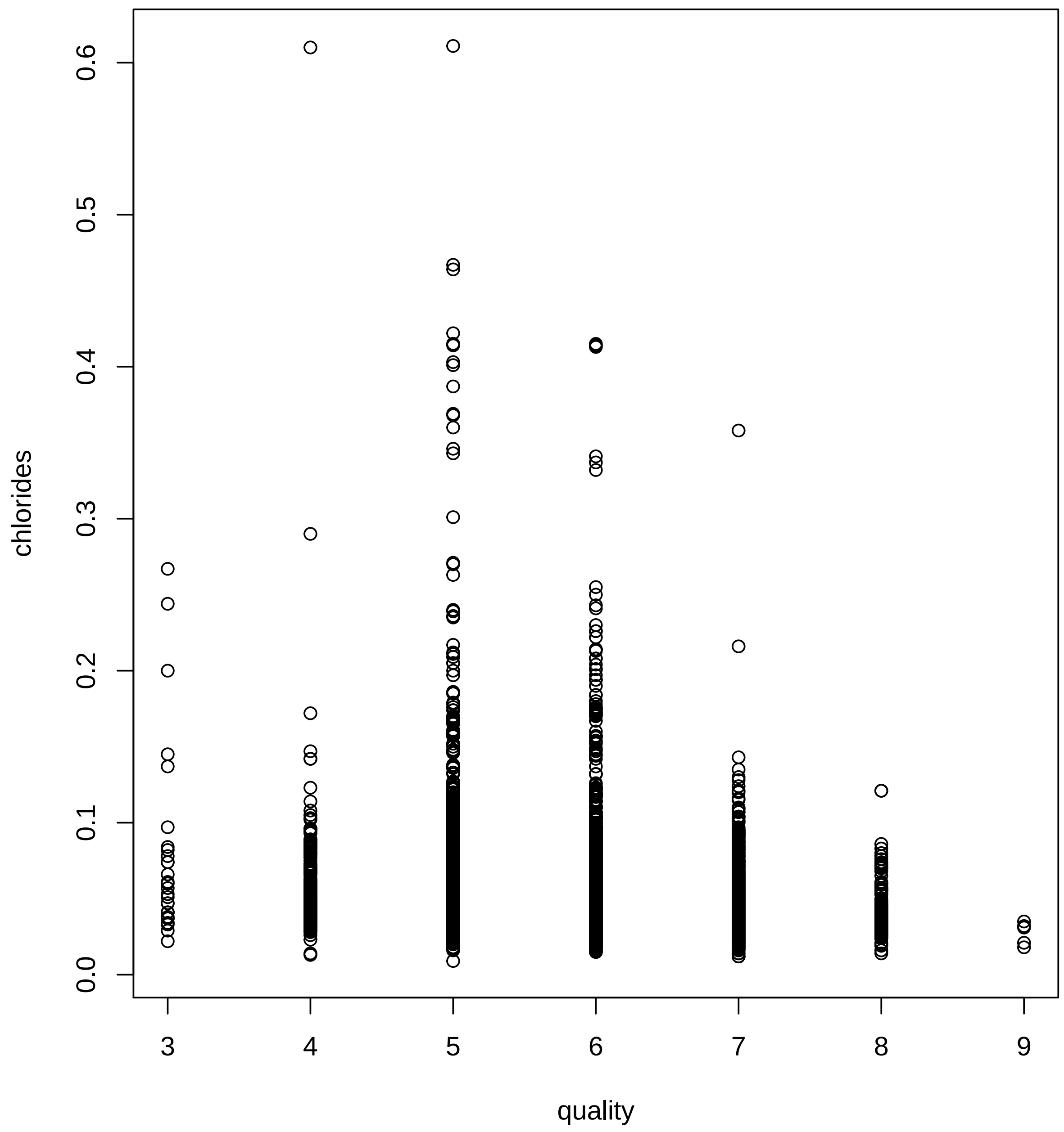

Figure 7: Variable Attributes vs. Quality

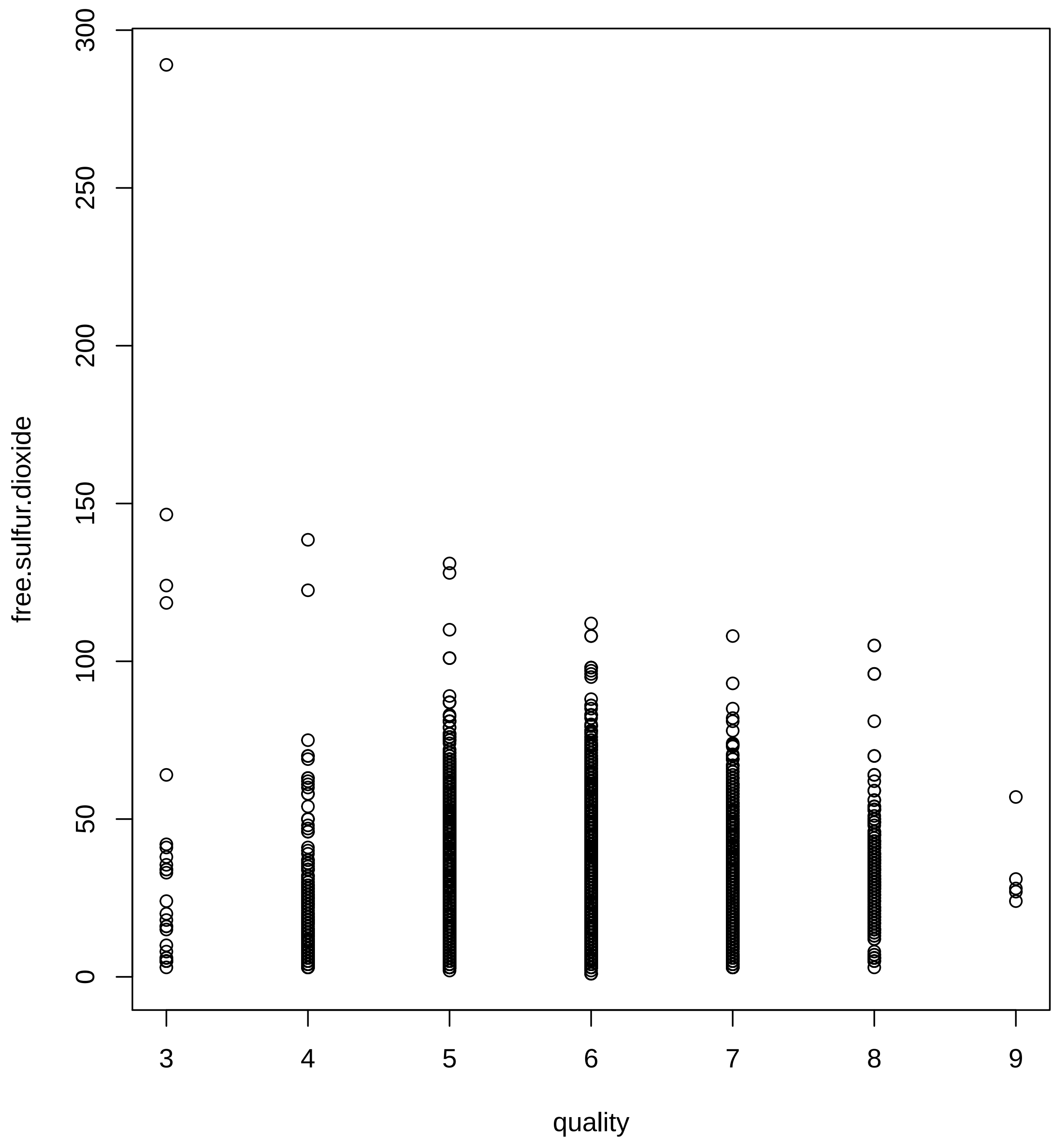

Figure 8: Variable Attributes vs. Quality

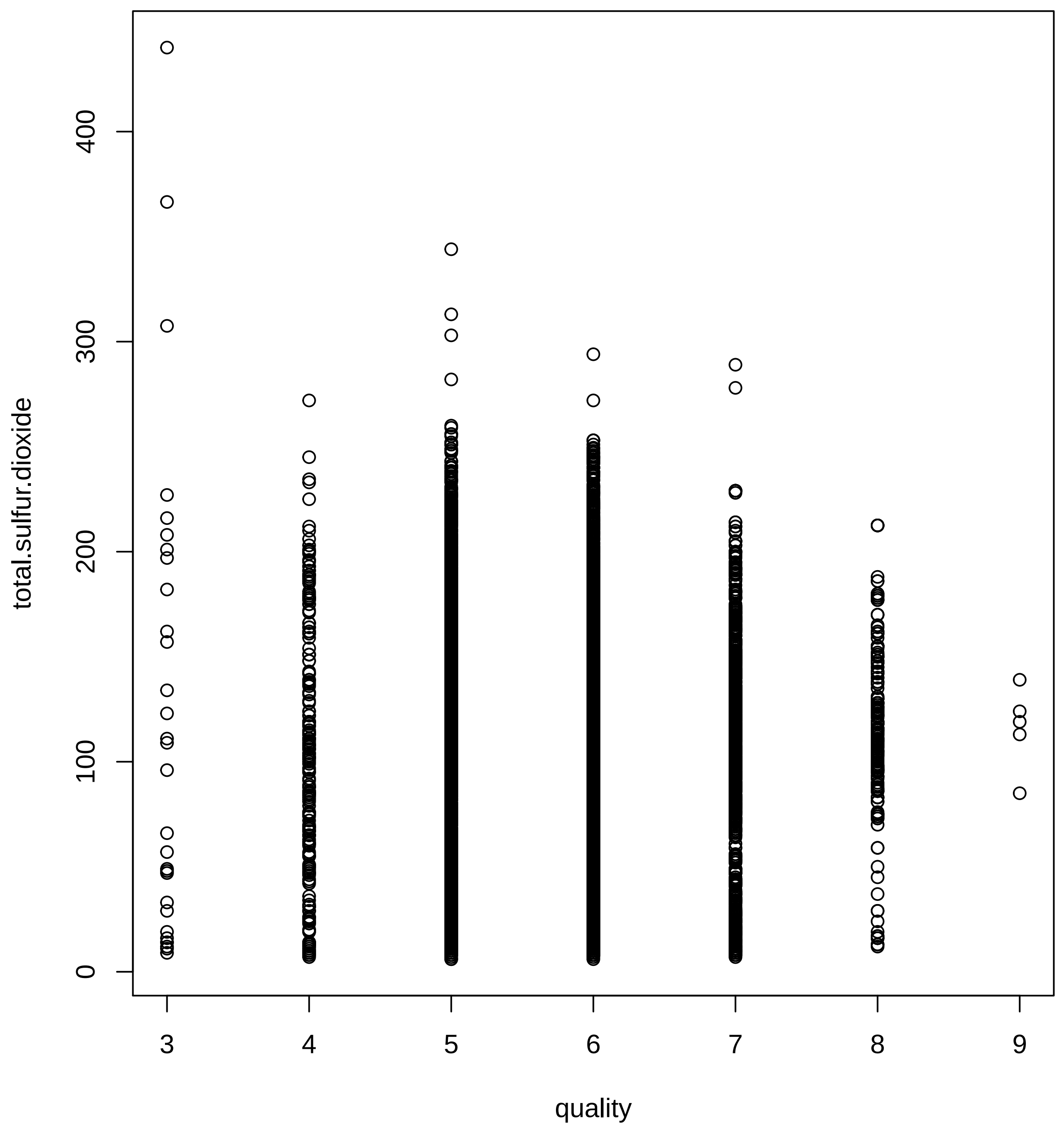

Figure 9: Variable Attributes vs. Quality

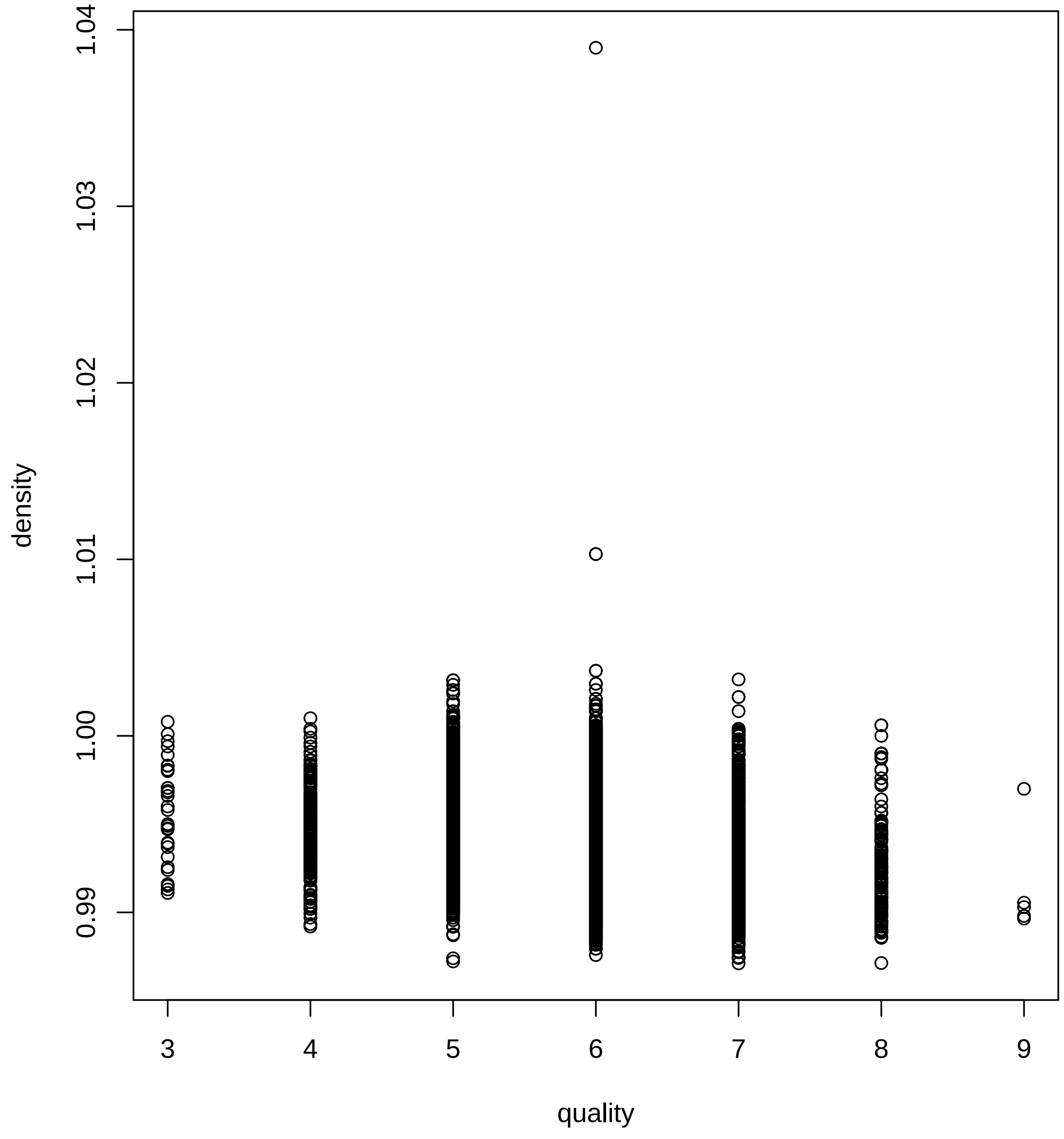

Figure 10: Variable Attributes vs. Quality

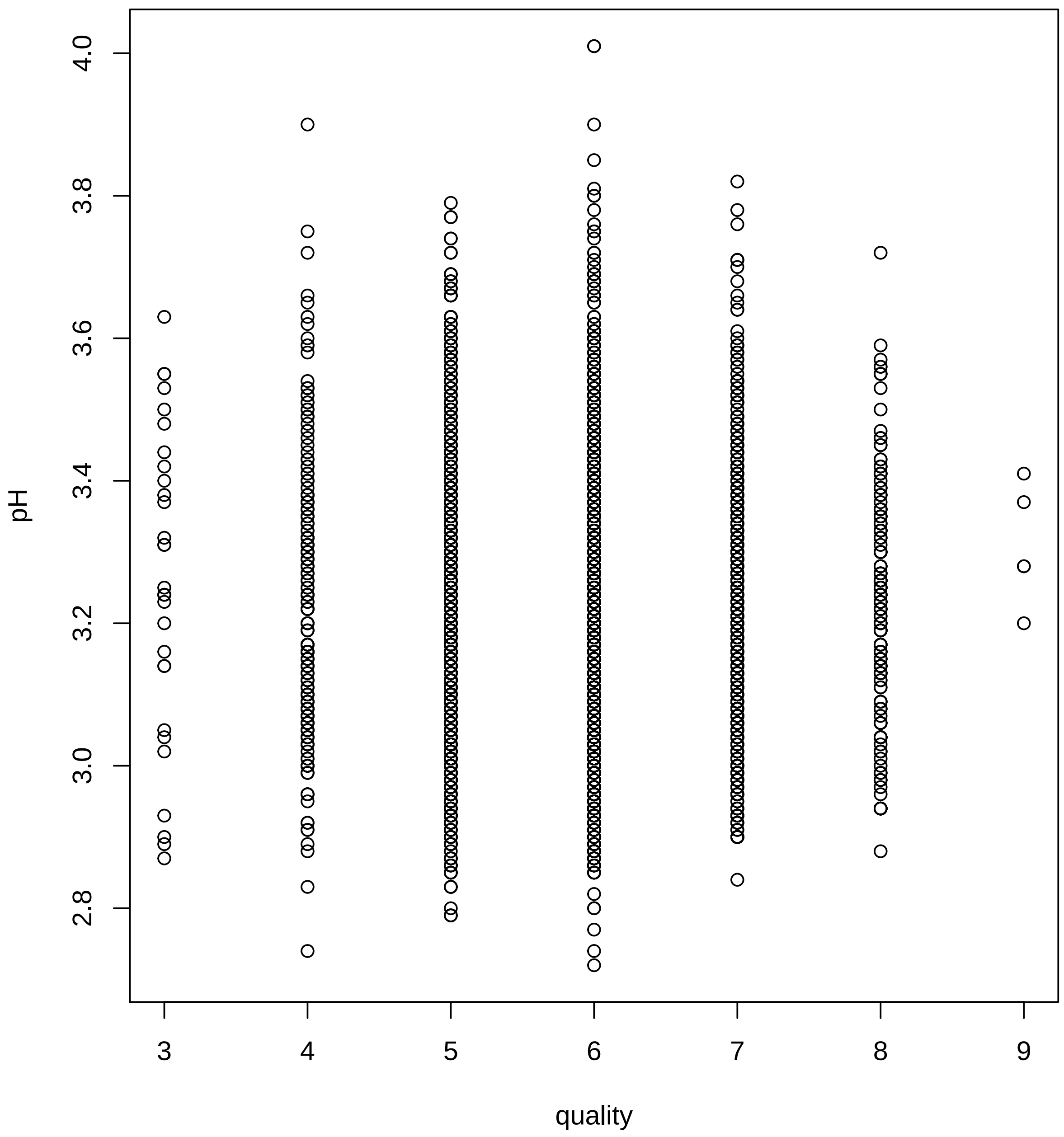

Figure 11: Variable Attributes vs. Quality

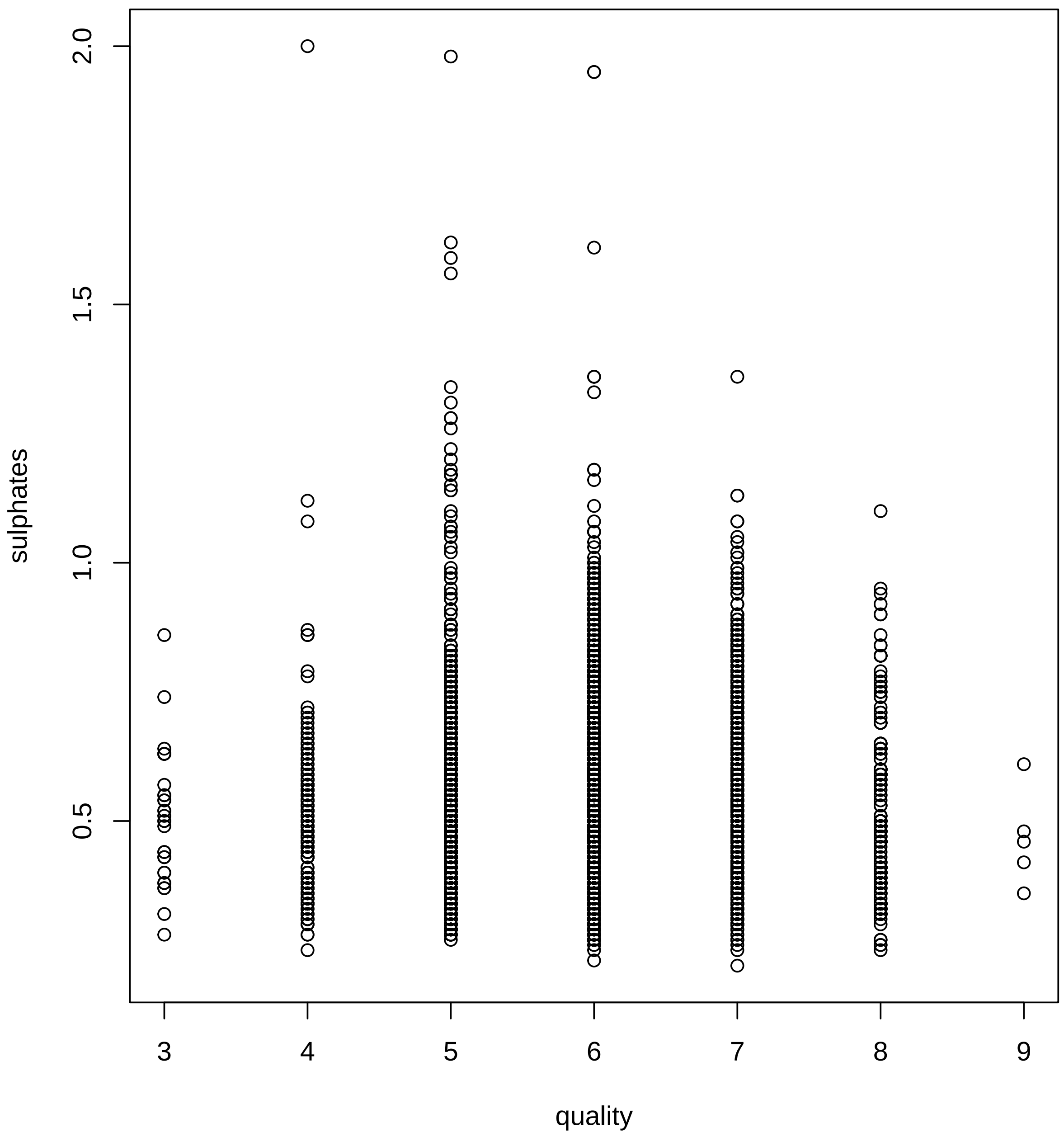

Figure 12: Variable Attributes vs. Quality

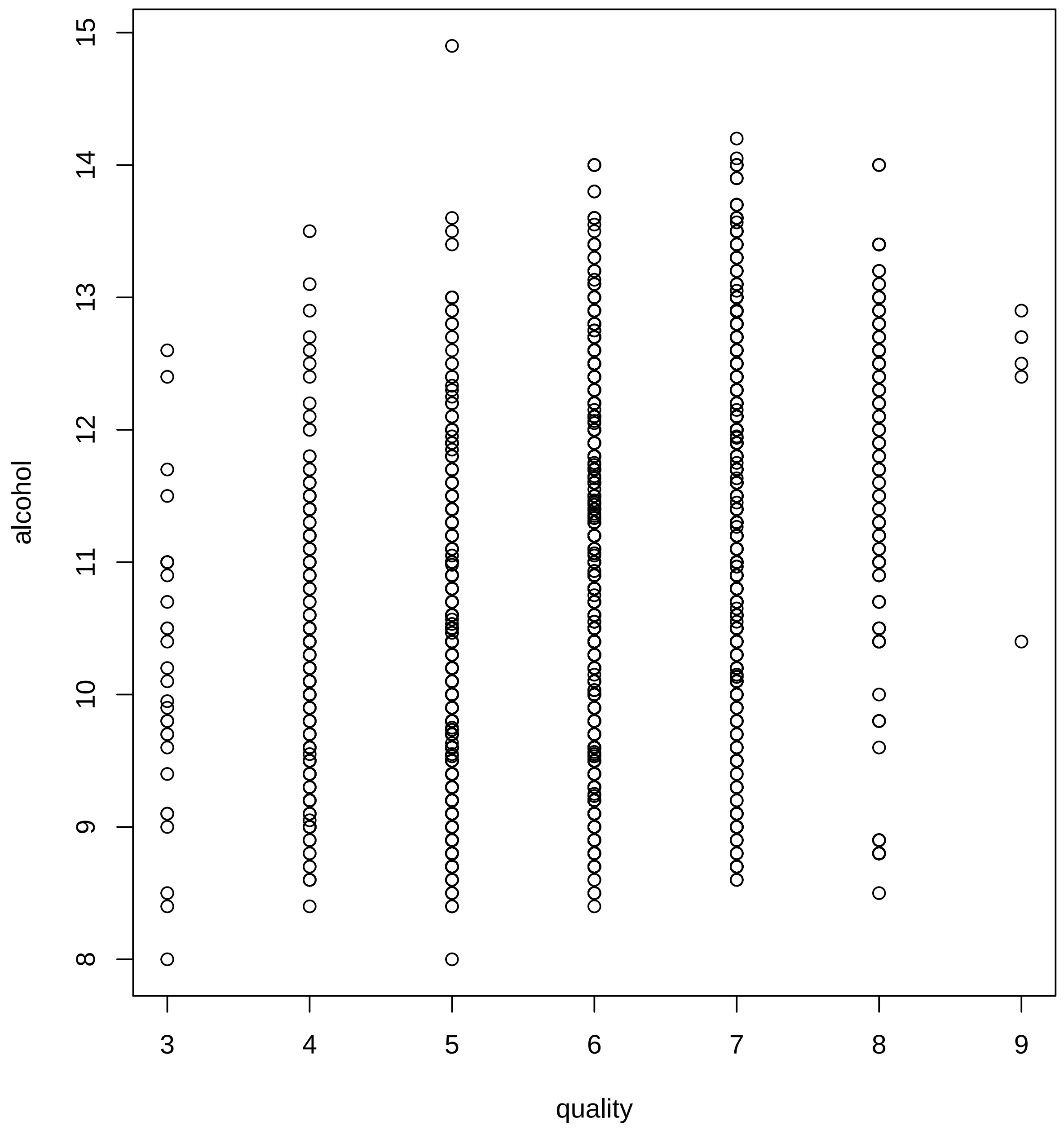

Figure 13: Variable Attributes vs. Quality

```
outlier<-boxplot(wine$quality,plot=FALSE)$out
wine_updated<-wine%>%filter(!(quality %in% outlier))
```

Prior to partitioning the data, the **seed** was set to 1 to ensure that the results will remain the same regardless of the computer used. Then the data was split into the **train_set** and **test_set** set with a 80-20 split. In order to allow for a good training model, the data used should be large, however we cannot use the entire datset for training because then there will be nothing left for validation. Thus the training set needed to have more data than the test, therefore 50-50 was not an option. Typically, the perfered splits are 80-20, 85-15 or 90-10. Since our dataset is not extremely large 80-20 was chosen for the test set to have a healthy amount of observations.

```
set.seed(1, sample.kind = "Rounding")
test_index<-createDataPartition(wine_updated$quality,
                                times=1,p=0.2,list=FALSE)
train_set<-wine_updated[-test_index,]
test_set<-wine_updated[test_index,]
```

2.3. Modeling

The first and most important step that was used for every method was the **RMSE** function. Thus, the function was introduced and defined in the program as:

$$RMSE = \sqrt{\sum(truequality - predictedquality)^2}$$

. Next a data frame called **results** was made to store all RMSE information for for the predictions on the test set. Although it is important to see check the RMSE for the trainset to ensure there is no overtraining, this was not added to the data frame because some models have multiple RMSE avlues and would make the table difficult to read.

In the caret package which is used for modeling, there are about 180 different modeling methods.These models can be divided in several ways, the simpliest way to determine which methods are applicable for any given case is by looking at the "type" in the caret package manual. There are three types: regression, classification, and cluster. Since the information to be predicted is the quality of the wine, a quantitve measure, we require regression. There are many different models that can be used for regression, however, some methods/ groups of methods were disregarded for this work due to the complexity and space requirements. Neural Networks methods are just some of the examples of methods which were not considered due to space and time restrictions.

Most of the methods used in this project fall under 4 groups: linear regression , tree based models, ensemble models, and support vector machines. Linear regression is the most basic technique used for modeling therefore it was used first with linear modeling (**lm**) and least angle regression (**lars**) being commonly used. Furthermore linear regression with feature extraction was used for principle component analysis (**pcr**) and independent component regression (**icr**), as a way to obtain summarative information from the variables available. Tree based models are very good at interperating non-linear data and provide a good visualization of relationships, hence CART (**rpart**), bagged CART (**treebag**), and stochastic gradient boosting (**gbm**) were used. Ensemble models generally use multiple learning algorithms to enhance the performance, thus generalized additive model using LOESS (**gamLoess**). Bagged CART and stochastic gradient boosting are also considered ensemble models. Finally, the support vector machine (svm) models are carried out using linear kernerl (**svmLinear**), polynomial kernel (**svmPoly**), and radial basis function (**svmRadial**).

2.3.1. Linear Regression

These are simple methods, where tuning parameters are only used for icr and pcr in the training function thus making them quick to run.There are three steps which were used for lm and lars methods, first it was trained using the training set, then predicted for the test set, then the RMSE was saved to the **results** data frame. The following example provides a look into how this was carried out:

```
lars<-train(quality~.,data=train_set, method= "lars")
y_lars<-predict(lars, test_set)
results<-rbind(results,data.frame(Method="Least Angle Regression"
                                  , RMSE=RMSE(test_set$quality,y_lars)))
```

However, for icr and pcr, tuning parameters exist, the parameter for both is number of components. Since there are only 13 attributes as part of the data set it makes sense to try 1-10 or 1-13. Never will the answer be more than 12 since the 13th attribute is the quality its self. This is done similarly for both icr and pcr, and looks as follows.

```
icr<-train(quality~.,data=train_set, method= "icr",
           tuneGrid=data.frame(n.comp=seq(1,10,1)))
plot(icr)
y_icr<-predict(icr, test_set)
results<-rbind(results,data.frame(Method="Independent Component Regression",
                                  RMSE=RMSE(test_set$quality,y_icr)))
```

After training the model the plots created from the rmse vs the tuning parameters are used to find the best value for ncomp on th training set, this can be seen in figures 14 and 15 for icr and pcr respectively. Icr uses 9 components while pcr is best with all 12 components.

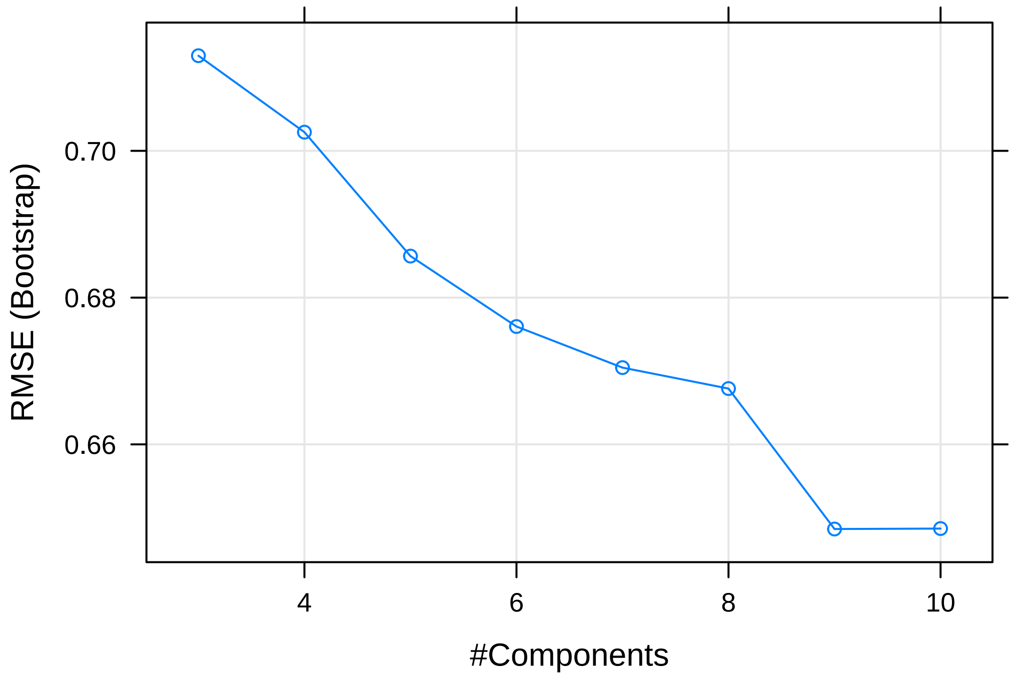

Figure 14: Tuning Plot for Independent Component Regression

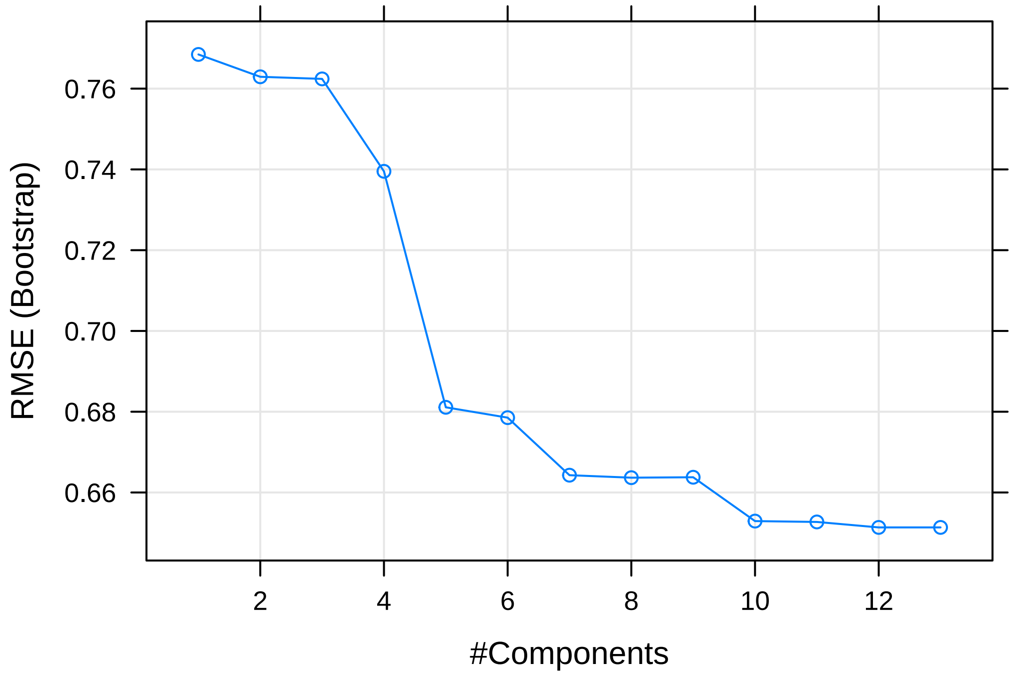

Figure 15: Tuning Plot for Principal Component Analysis

2.3.2. Tree-Based Models

Tree based models use a decision tree pronicpal to train and predict information. To train the models different procedure is required depending on the tunning parameters. For each model the number of parameters differs. For the Bagged Tree, there are no tuning parameters and thus the procedure above can be used. However CART and gbm each have tuning parameters, not that when we have tuning parameters we plot the trained model to see how the RMSE changes with the parameters. CART has one parameter the complexity paramter or cp this is used to control the size of the decision tree. Several values for cp are tested ranging from 0-0.05 as shown by the code below. the plot in figure 16 shows the variation for RMSE with cp which is 0.004167.

```
#Evaluate the model
rpart<-train(quality~.,data=train_set, method= "rpart",
             tuneGrid= data.frame(cp=seq(0,0.05,len=25)))
plot(rpart)
y_rpart<-predict(rpart, test_set)
results<-rbind(results,data.frame(Method="CART",
                                  RMSE=RMSE(test_set$quality,y_rpart)))
```

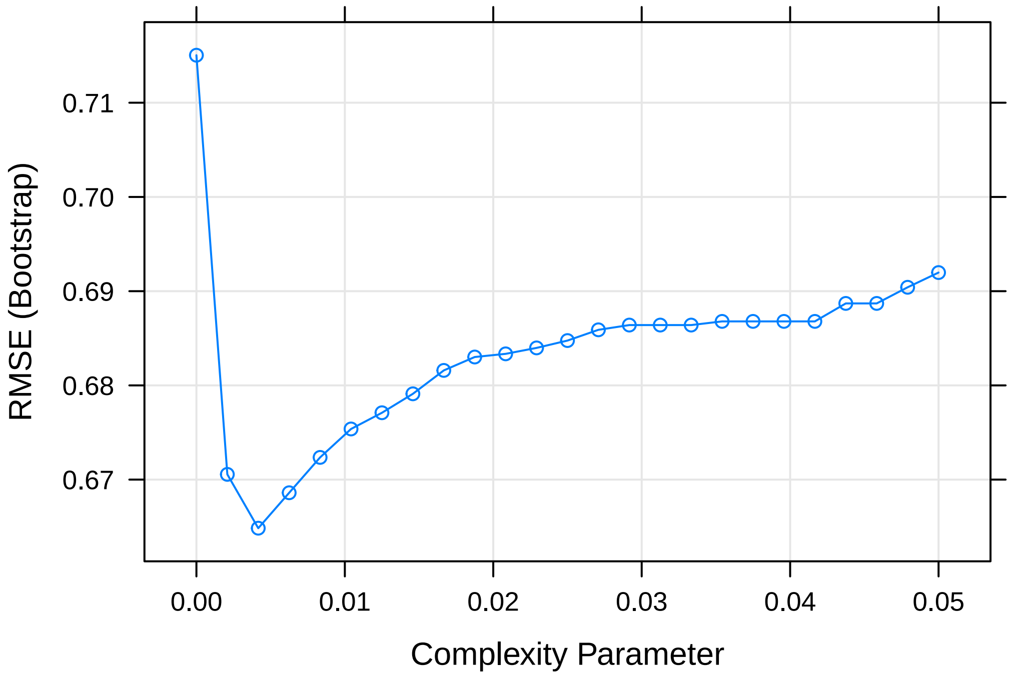

Figure 16: Tuning Plot for CART

As for the gbm model, there are 4 tuning parameters two of which were tested and 2 were kept constant. The two parameters kept constant were the shrinkage kept at 0.1 and the minimum observations in one node which was held at 10. The trainset had over 5000 thus placing the min at 10 allows for large room for the model to move. The paramters that were tuned were the interaction depts 5-10, and the number of trees 100-700. The code is show below. The plot in figure 17 shows the RMSE with tree depth and number of trees which are 9 and 700. It is important to note that at the 500-700 trees the RMSE gets very close, with less than a 0.001 drop for an extra 100 trees.

```
grid<-expand.grid(interaction.depth=seq(1,10), n.trees=seq(50,500,50),
                  shrinkage=0.1, n.minobsinnode=10)
gbm<-train(quality~.,data=train_set, method= "gbm", tuneGrid=grid)
plot(gbm)
y_gbm<-predict(gbm, test_set)
RMSE(test_set$quality,y_gbm)
results<-rbind(results,data.frame(Method="Stochastic Gradient Boosting",
                                  RMSE=RMSE(test_set$quality,y_gbm)))
```

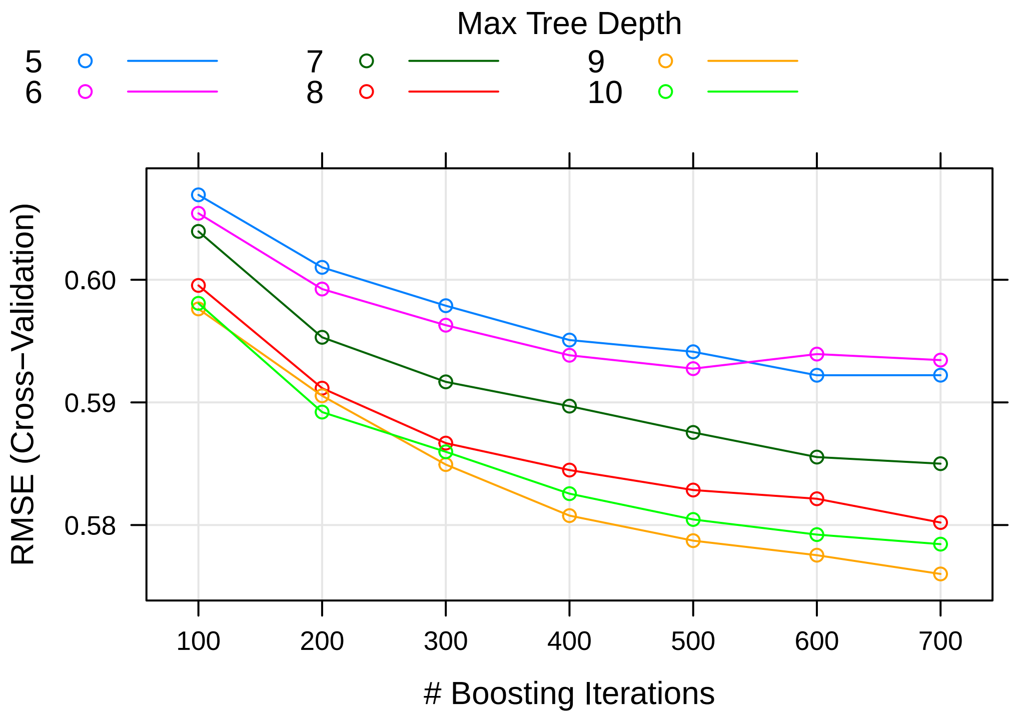

Figure 17: Tuning Plot for stochastic gradient boosting

2.3.3. Ensemble Models

Technically, gbm and treebag are ensemble models however they were already evaluated. Therefore, gamLoess will be the only one to be evaluated in this section.The tuning parameter used was the span which ranged from 0.15-0.65 and the degree which can be 1 or 2 but only 1 was chosen. The code showing the model is shown below.

```
grid<- expand.grid(span=seq(0.15,0.65,len=10), degree=1)
gamLeoss<-train(quality~.,data=train_set, method= "gamLoess", tuneGrid= grid)
y_gamLeoss<-predict(gamLeoss, test_set)
results<-rbind(results,data.frame(Method="Generalized Additive Model using LOESS",
                                  RMSE=RMSE(test_set$quality,y_gamLeoss)))
```

2.3.4. Support Vector Machine

There are many models under support vector machine however, only a few were used. With varying kernels linear, polynomial and radial basis function. First they were evaluated without tuning parameters then the one with the best RMSE was tweaked for varying parameter. The initial models were run in the same manner as the linear regression. Then the tunning was done in stages to determine which range to look at without having a large run. first the c or cost values were inspected from 1-3 then from 2-3 with sigma from 0-1 then the ideal cost was found and used for the final run with sigma ranging from 0.6-1.8. Thus, the only differences were in the tuneGrid. then they were trained and predicted as usual and ploted in figures 18-20 to see the optimal points.

```
grid<-expand.grid(C=seq(1,3,0.5),sigma=0.086)
grid<-expand.grid(C=seq(2,3,0.25),sigma=seq(0,0.1,0.02))
grid<-expand.grid(C=2.75,sigma=seq(0.08,0.16,0.02))
```

Each of the tuning runs provided more information, as shown in the figures (18-20). The first one showed that the C was about 2.50, thus for the run after the range was narrower but with more break ups. The second run showed that 2.5 and 2.75 were very close in RMSE but that 2.75 was better. Thus that was the c chosen for the final run. Meanwhile, in the first iteration the sigma is set to 0.086 because that was the default value used before the tuning started. Then in the second run a range was creating including that value, from 0-0.1. From that it was shown that 0.1 had the lowest RMSE which meant that further inspection was necessary. In the final run the sigma is done for values between 0.08-0.16. the lowest value is found at 0.12. Thus the final model will have C=2.75 and sigma=0.12.

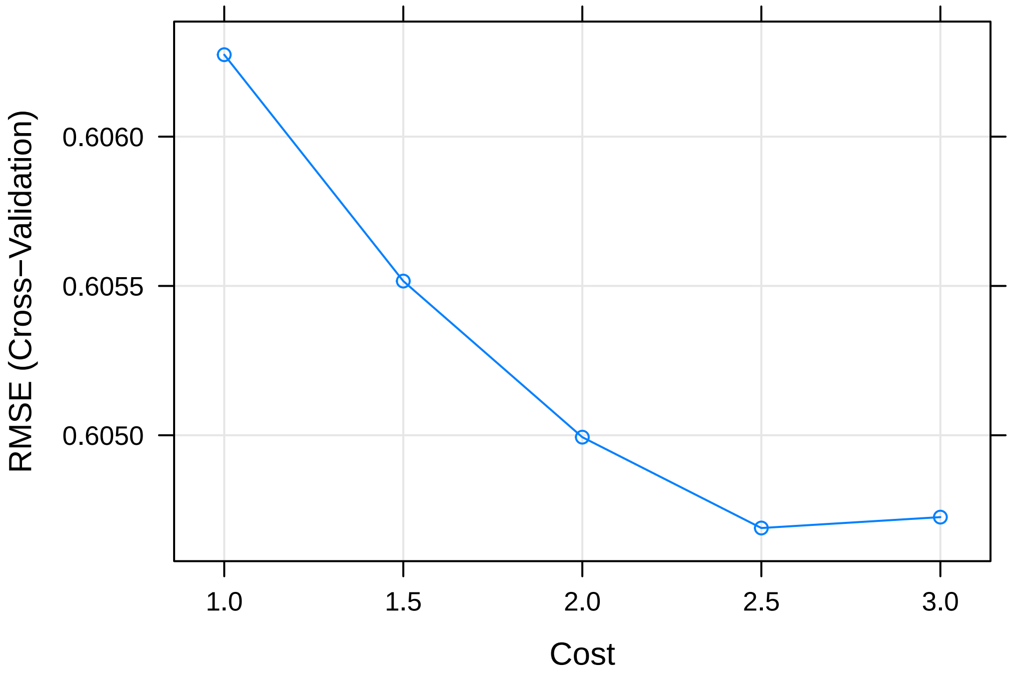

Figure 18: Tuning Plot for SVM Radial

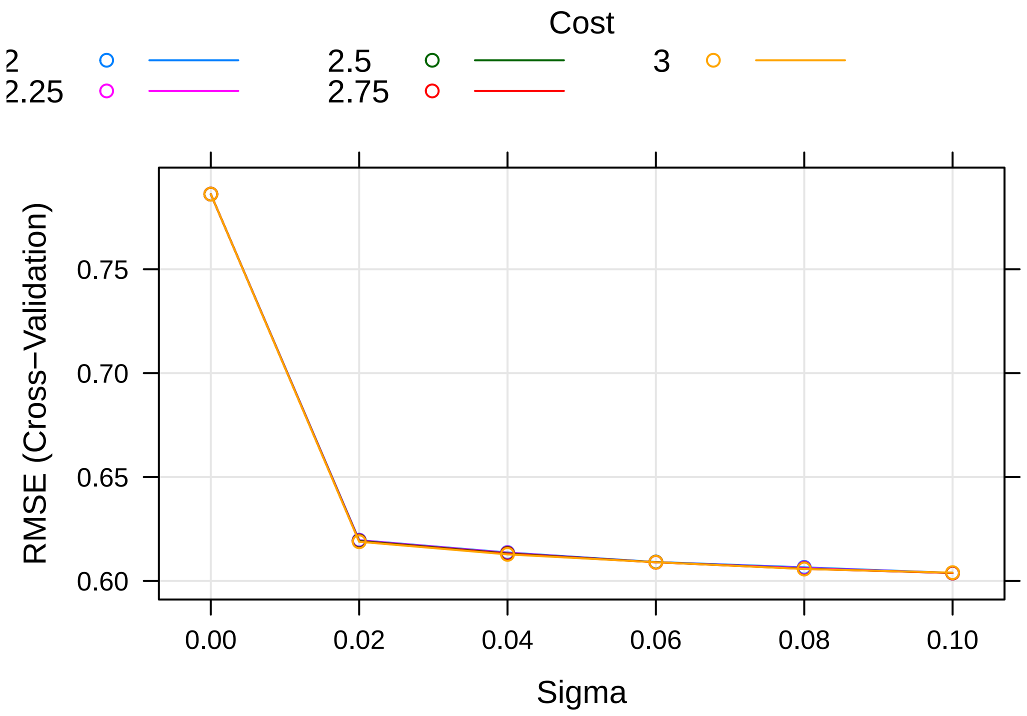

Figure 19: Tuning Plot for SVM Radial

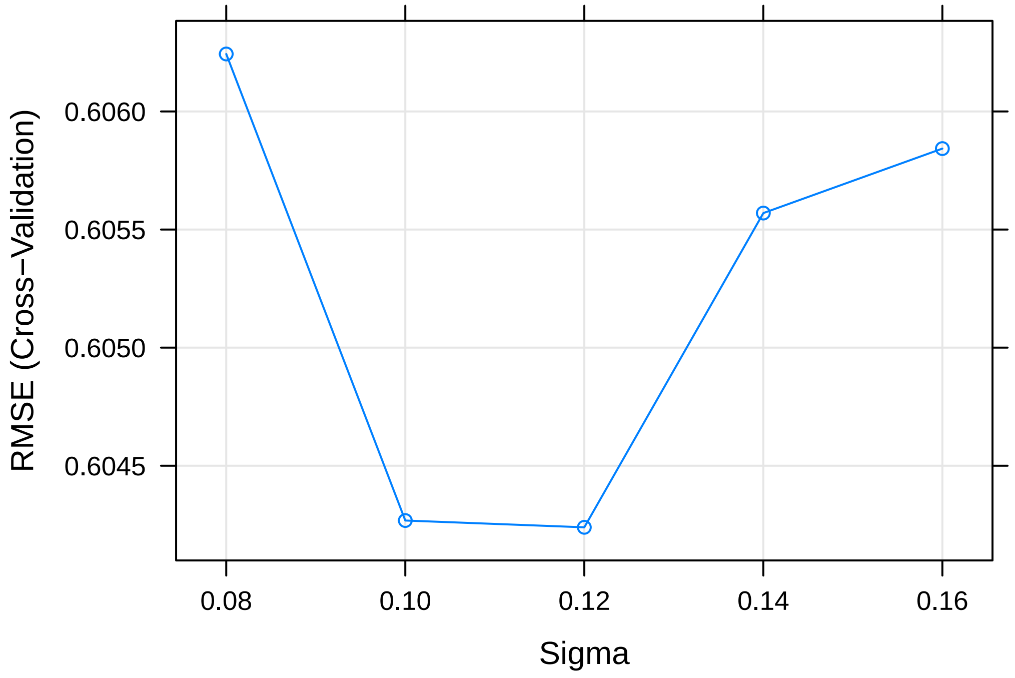

Figure 20: Tuning Plot for SVM Radial

3. Results and Discussion

When carrying out the models for this project it is important to pay attention to the results from the tuning parameters as well as the RMSE values. The tuning parameter optimization is a big step to find the results of each method as best as possible. Note that it is not necessary to create a new model to carry out the optimized parameters. This is because in R, the tuning parameter or the combination of parameters which provide the highest accurary or lowest RMSE (depending on the information predicted) are automatically used in what is called the final model. There are two ways to obtain the parameters to be used bestTune and finalModel.Before creating the final table with all the RMSE values, first a table was made for the SVM methods. Note that in table 3, the base case for varying SVM methods show that using the radial basis function provides the lowest RMSE, thus this method was further analyized for optimal tuning parameters. The tuning was not done all at once since the range was unknown, and the grid size would be large therefore occupying more time for each run. The optimization for these techniques is carried out on the training set, the test set was just used to further validate the arguments made.

Table 3: The RMSE Results from Support Vector Machine Methods

Method	RMSE
SVM with Linear Kernel	0.6626488
SVM with Polynomial Kernel	0.6437384
SVM with Radial Basis Function Kernel	0.6198124
SVM with Radial Basis Function Kernel (tuning C (2.5))	0.6144164
SVM with Radial Basis Function Kernel (tuning C (2.75) and sigma (0.1))	0.6115897
SVM with Radial Basis Function Kernel (C (2.75) tuning sigma (0.12))	0.6094837

From this table it is evident that the radial basis function is the most accurate as the RMSE is almost 0.02 and 0.04 lower than that of polynomial and linear kernels. Using the optimized tuning data, as explained in the section above, the RMSE is at the lowest at 0.6042 for the training set and 0.6094837 for the test set as shown above. Once this value is found it can be added to the **results** table for the RMSE. As has been learnt in the machine learning course, the ensemble or combination of methods can be used to permit for more accurate predictions. Thus we analyse doing so. However, joing many different methods with varying RMSE can end up being more detrimental than helpful. Hence, only 2 models will be combined for our ensemble. The two models to be used are chosen on the basis of having the lowest RMSE in the **TRAINING SET**, shown in figure 4. The reason for that is, as we said before the test set is actually our validation set despite the name. After inspecting all the RMSE values in table 4 on the traing set it is found that stochastic gradient boosting (**gbm**) and support vector machine with radial basis function kernel (**svmRadial**) have the lowest RMSE at 0.5760 and 0.6042 respectively. Hence, a combination of the two for the prediction is added to the **results** table.

Table 4: The RMSE Results from All Methods on the Train Set

Method	RMSE
Linear Regression	0.6506137
Least Angle Regression	0.6512044
Independent Component Regression	0.6484663
Principal Component Analysis	0.6513508
CART	0.6648430
Bagged CART	0.6593461
Stochastic Gradient Boosting	0.5760108
Generalized Additive Model using LOESS	0.6806351
Support Vector Machines with Radial Basis Function Kernel	0.6042393

Table 5: The RMSE Results from All Methods on the Test Set

Method	RMSE
Linear Regression	0.6604274
Least Angle Regression	0.6604274
Independent Component Regression	0.6628577
Principal Component Analysis	0.6604274
CART	0.6807619
Bagged CART	0.6762862
Stochastic Gradient Boosting	0.5891340
Generalized Additive Model using LOESS	0.6507150
Support Vector Machines with Radial Basis Function Kernel	0.6094837
Ensemble:GBM and SvmRadial	0.5817475

Table 5 shows the RMSE values for all utilized methods, it is evident from this table that all methods utilized here provide a somewhat good prediction. Some might say this is luck, however it is not. Initially model averaged neural networks was a method utilized without tuning, not only did the run take an extremely long time but the RMSE was over 4.0. Thus it was decided that this avenue would not be persued any further and would be removed from the data. Generally speaking, the linear regression models had the same RMSE which hovers towards the top to middle of the pack. The two methods which seemed to have the 'worst', relatively, RMSE belong to the tree-based models, which is ironic because the best method is also tree-based. However, **gbm** is not only a tree-based method it is also an ensemble method. Based on the **gamLeoss** and **gbm** values in the RMSE table it is clear that ensemble methods are amongst the best in predicting wine quality. The SVM models vary depending on the kernel used as seen in the previous table. The best perdiction comes from the ensemble created which combines GBM and SVM Radial which provides an RMSE value of 0.5817475. This value is 0.0073865 lower than the second lowest value for gbm and 0.0990144 lower than the highest value for CART method.

Therefore, the combination of methods was a good decision as it provided the lowest RMSE value. To better see how the predictions are lets visualize the true quality against that of the predicted. This can be seen in the plot in figure 21. It can be noticed that for wines that have a quality of 4, the prediction varies heavily never predicting a 4.For 5's most of the predictions hover around the 5-5.5 range which is acceptable with very few going above 6. For the 6's the largest density is the 5.5-6.25 range. As for the 7's they are predicted to be 6.25-6.75. It seems as thou the lower values tend to be over predicted, and the upper values tend to be under predicted. It can also be seen that there is a larger degree of error in predicting white wine, as the red wines in the 4 range dont ever make it to the 6, the majority of red 5's are predicted within 0.5 of the 5. For the 6's and 7's they are less accurate but they do not span the same wide range as the white.

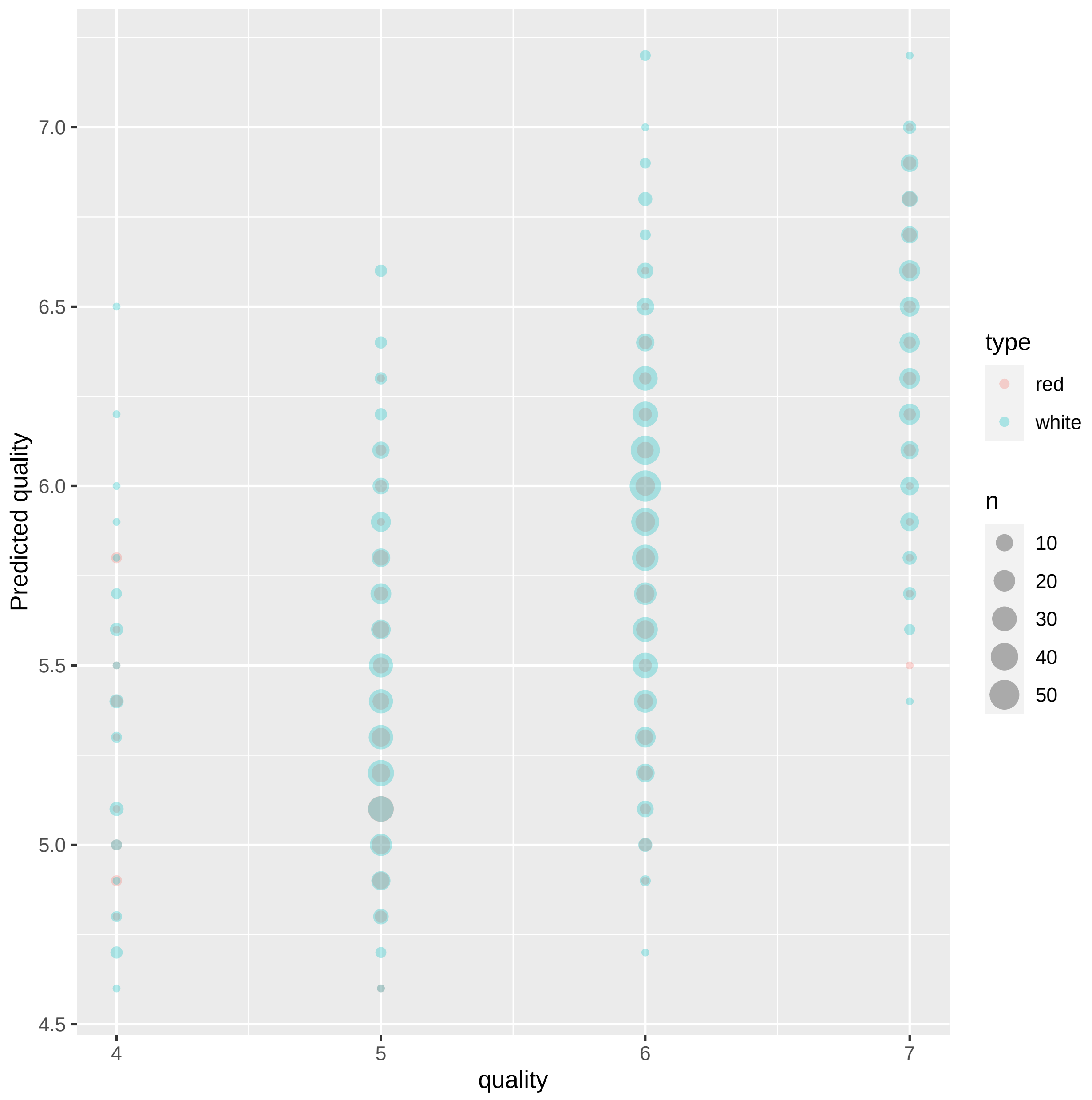

Figure 21: True Quality vs. Predicted Quality

4. Conclusion

The wine quality Project was carried out with the objective of predicting wine quality despite color for Portuguese "Vinho Verde" wines by reducing the RMSE. Throughout the project various methods were used to predict the quality for a validation set (called **test_set**). First the data was obtained, cleaned, and then inspected. Afterwhich it was split to prepare for machine learning methods. The methods included linear regression, tree-based models, ensemble models, and SVM. The various methods are trained and tuned to find optimal RMSE values. The methods with the two lowest RMSE values for the training set were combined to provide an enhanced prediction and thus a lower RMSE. This method is called the ensemble for stochastic gradient boosting and support vector machine with radial basis function kernel which provides an RMSE value for the validation set of 0.5817475. This project provides visualization of wines Portuguese wines and their quality. It can be seen from this dataset that most of their wines have a quality randing from 4-7 and thus anything above or below is considered an outlier. The techniques explored in this project have the potential to predict wine quality based on physicochemical components. However due to limitations with space, time, and computing abilities only so much could be done without crashing RStudios. As some neural network methods such as MLP continuously caused the Rsession to crash, and other RVM methods would not finish training after 18 hours. If taken further and enhanced via more consuming modeling techniques such as neural networks, random forst, and more radial basis function models, this work has the potential to provide what could only be described as the golden ratio of wine flavor. Thus producers can work to alter the make up of the wine to reach the golden ratio and thus the highest quality possible.

Citation: 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, 2009.