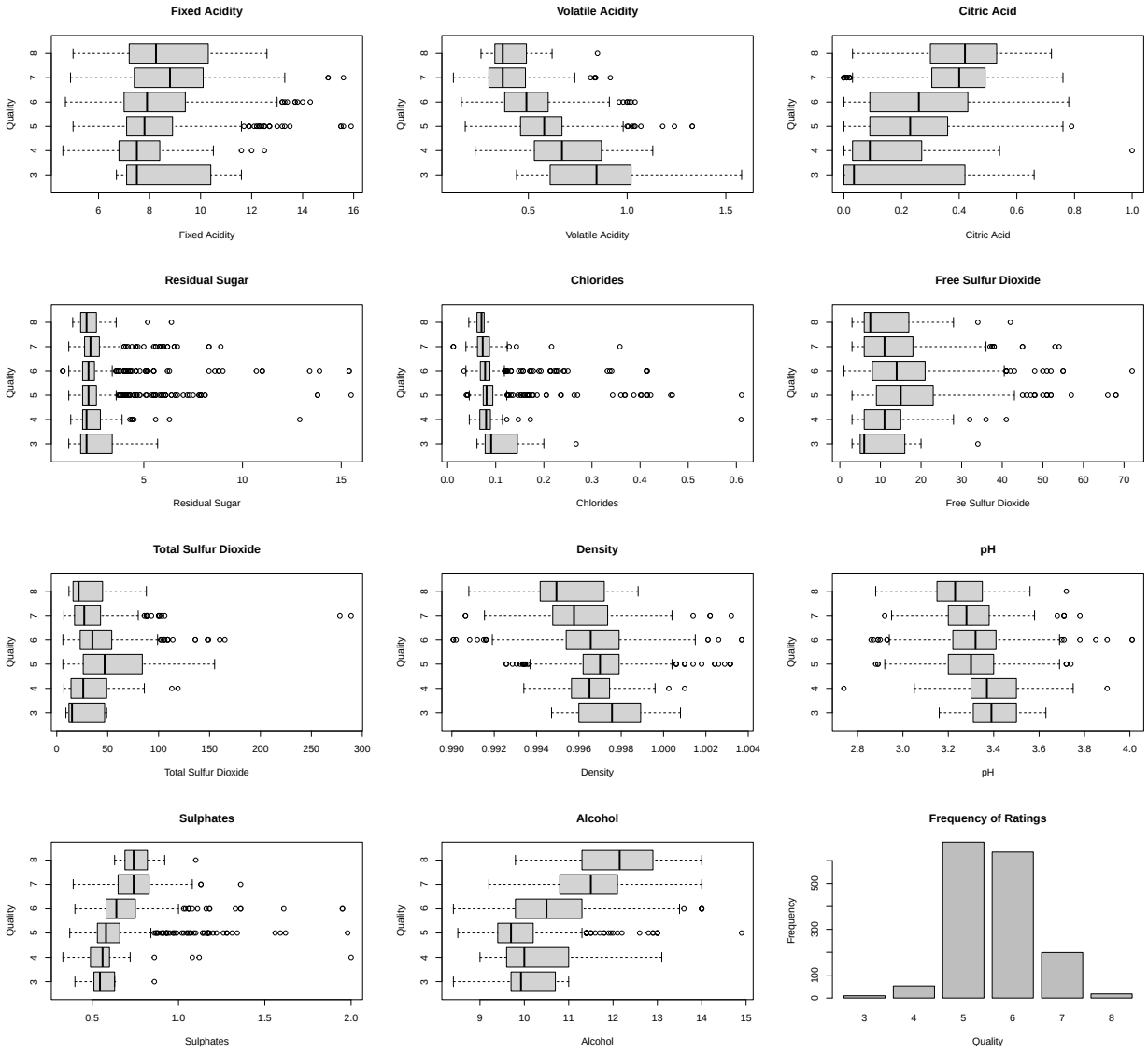
Applying Machine Learning to Predict Wine Quality Ratings

Will Firmin

5/22/2022

For this project I used data on samples of wine from Northern Portugal. The data includes 12 variables: 11 are results from physico-chemical tests on the wine, and the final variable is a quality rating from 1 to 10. In the data, only ratings from 3 to 8 are present. The aim of the project is to use these physico-chemical properties of wine to predict its quality. The plots below show the relationship between quality and each predictor variable. Since there are only six discrete levels on the y axis, a scatter plot is hard to read. Therefore I have used box plots to show the distribution for each level of quality. The most significant relationships seem to be with volatile acidity, citric acid, sulphates, and alcohol. The final plot shows the frequency of each rating level. It's important to note that the data is dominated by middle values 5 and 6, with few observations of very good or bad ratings. For this reason, classification will be applied first to two classes: "Good" signifying a score greater than or equal to 6, and "Bad" signifying a score less than or equal to 5. Later I will apply classification with six classes, one for each rating level. Finally, I will use regression methods.

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



Binary Classification

Logistic Regression

The first model applied to the "Good" vs "Bad" classification problem is logistic regression. The summary of the model shows that the most significant predictors are volatile acidity, citric acid, total sulfur dioxide, sulphates, and alcohol. This is largely consistent with the visual observations made above. The table of predictions indicates that the majority are correctly classified. The accuracy of the model on the validation set is about 74%, which is much greater than simply guessing. Guessing the most common rating (Good) gives an accuracy of 51.25%, far below 74%.

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = wineC,
##     subset = train)
##
```

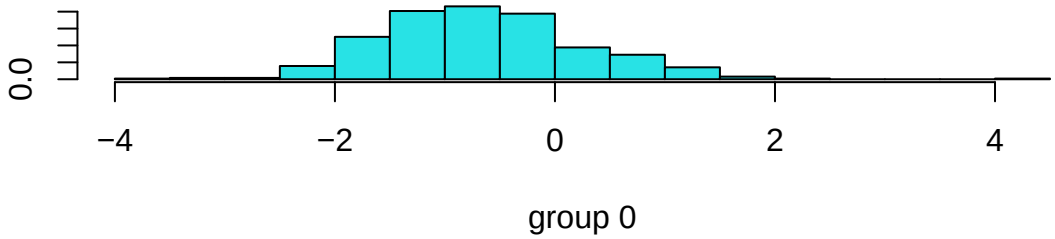
```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4343  -0.8301   0.3128   0.8060   2.2911
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           57.779068  96.092487   0.601   0.5476
## fixed.acidity          0.141564   0.119470   1.185   0.2360
## volatile.acidity      -3.175509   0.588609  -5.395 6.85e-08 ***
## citric.acid           -1.416295   0.683255  -2.073   0.0382 *
## residual.sugar         0.010385   0.067937   0.153   0.8785
## chlorides             -3.528084   1.813208  -1.946   0.0517 .
## free.sulfur.dioxide    0.015112   0.009999   1.511   0.1307
## total.sulfur.dioxide  -0.015412   0.003556  -4.334 1.47e-05 ***
## density              -64.890175  98.105536  -0.661   0.5083
## pH                    -0.791903   0.859508  -0.921   0.3569
## sulphates              2.505958   0.539572   4.644 3.41e-06 ***
## alcohol                0.935579   0.127084   7.362 1.81e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1542.5  on 1118  degrees of freedom
## Residual deviance: 1146.0  on 1107  degrees of freedom
## AIC: 1170
##
## Number of Fisher Scoring iterations: 4


##
## log.pred   0   1
##        0 180  71
##        1  54 175

## [1] Logistic Binary Classification Accuracy: 0.739583333333333
```

Linear Discriminant Analysis

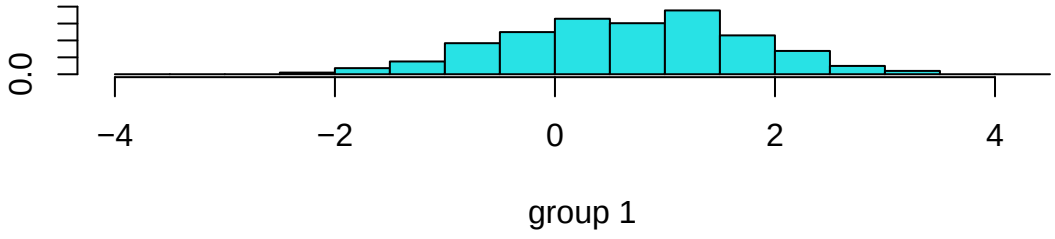
The next model, LDA, is trained with cross validation. The group means reported by the model show that there are sizable differences between good and bad wine in some predictors. The plot shows that the two classes are somewhat separated, allowing for some accurate classification. The LDA model resulted in a cross validation accuracy of about 74% and a validation set accuracy of 73.8%. This is slightly less accurate than the logistic regression model.

```
## Call:
## lda(quality ~ ., data = wineC, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4557641 0.5442359
##
## Group means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
```

```
## 0      8.117255        0.5857941   0.2369608       2.564314 0.09368627
## 1      8.444499        0.4738916   0.2993596       2.486946 0.08284236
##   free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates
## 0            16.66078             53.32843 0.9970740 3.313118 0.6220000
## 1            14.98358             38.11987 0.9963625 3.311806 0.6893924
##     alcohol
## 0  9.919706
## 1 10.912972
##
## Coefficients of linear discriminants:
##                               LD1
## fixed.acidity          0.10565560
## volatile.acidity      -2.39814916
## citric.acid           -0.96155440
## residual.sugar        -0.00372854
## chlorides             -3.11417484
## free.sulfur.dioxide    0.01418617
## total.sulfur.dioxide  -0.01293252
## density              -41.45880096
## pH                    -0.55841507
## sulphates              1.99796298
## alcohol                0.70769506

## Linear Discriminant Analysis
##
## 1599 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1007, 1007, 1008, 1007, 1007, 1007, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7399131  0.4773125

##
## lda.pred   0   1
##        0 180  72
##        1  54 174

## [1] LDA Accuracy: 0.7375
```



group 0



group 1

Quadratic Discriminant Analysis

The QDA model is applied next, and it finds a cross validation accuracy of only 71.8% and a validation set accuracy of 69.8%. This decrease in accuracy from the logistic and LDA models indicate a more linear relationship in the data.

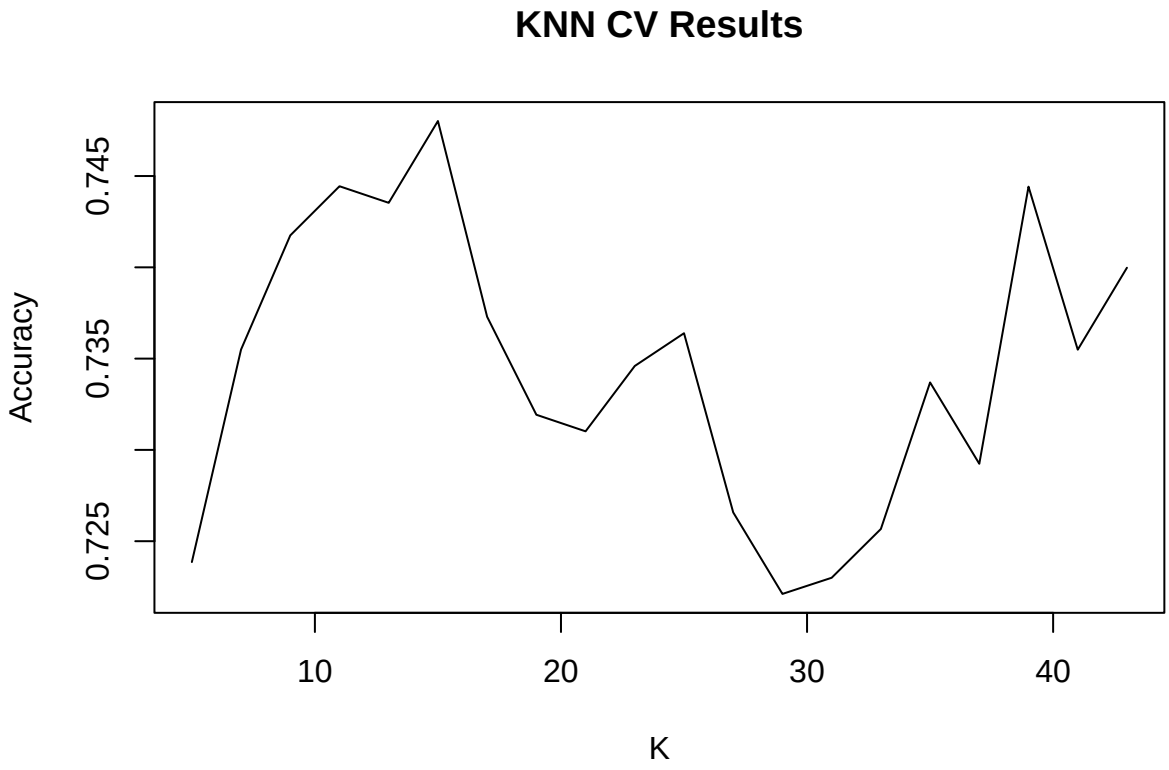
```
## Quadratic Discriminant Analysis
##
## 1599 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1008, 1007, 1007, 1007, 1007, 1007, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.717648  0.4188196

##
## qda.pred   0   1
##        0 139  50
##        1  95 196

## [1] QDA Accuracy: 0.697916666666667
```

K Nearest Neighbors

The KNN model is applied with the value of K being selected through cross validation. The most accurate model during cross validation uses the 15 nearest data points and has an accuracy of 74.8%. However, its accuracy on the validation set is only 69.4%. This would confirm the earlier suspicion that a linear model is best.

```
##
## predK   0   1
##     0 151  64
##     1  83 182
```

```
## [1] KNN Accuracy: 0.69375
```

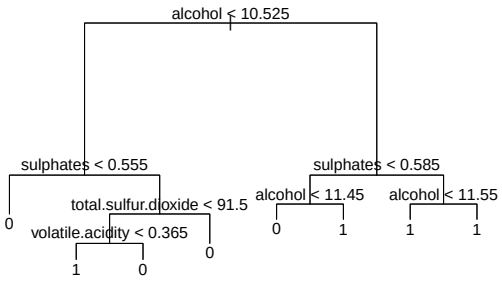


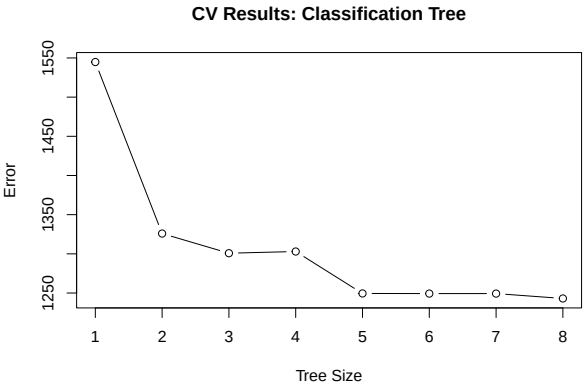
Classification Tree

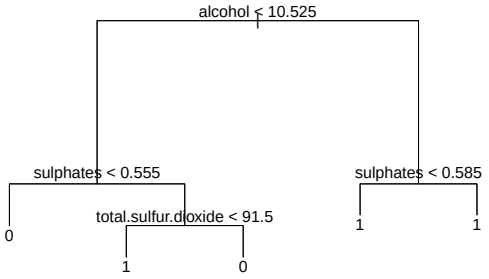
A simple classification tree model finds that alcohol, sulphates, total sulfur dioxide, and volatile acidity are the strongest predictors. This is consistent with earlier findings. The plot of the tree shows that alcohol is used to make the first split, followed by sulphates. It seems like higher values for alcohol and sulphates lead to higher ratings. Lower values for total sulfur dioxide and volatile acidity appear to increase ratings, but this might only be in the subsections of the data split by alcohol and sulphates. I use cross validation to find the optimal length of the tree, which is the full complexity. The plot also shows that most improvement has been made by the time the tree reaches 5 terminal nodes, so I will test both models. On the validation set, the full tree scores an accuracy of 70.6%, while the pruned tree scores 65.6%. This is expected given the

cross validation results, but the pruned tree has the worst accuracy of all the models so far, so the tree is not worth pruning.

```
##
## Classification tree:
## tree(formula = factor(quality) ~ ., data = wineC, subset = train)
## Variables actually used in tree construction:
## [1] "alcohol"              "sulphates"            "total.sulfur.dioxide"
## [4] "volatile.acidity"
## Number of terminal nodes:  8
## Residual mean deviance:  1.008 = 1119 / 1111
## Misclassification error rate: 0.2574 = 288 / 1119

## [1] Full Tree Accuracy: 0.70625

## [1] Pruned Tree Accuracy: 0.65625
```







Support Vector Machines

Support Vector Classifier The SVC model is fitted using cross validation on a range of cost values. The best value comes at cost = 0.2976. Using this parameter, the SVC model score an accuracy of 72.9% on the validation set. This is not the best so far, but it is better than the trees.

```
##       cost
## 6 0.2976351
```

```
##
## svm.pred   0   1
##        0 182  78
##        1  52 168
```

```
## [1] SVC Accuracy: 0.729166666666667
```

Radial SVM The radial SVM is fit similarly to the SVC model using cross validation on a range of both cost and gamma values. The best performing model has cost = 0.8859 and gamma = 0.4642. The cost parameter for this model is higher than the SVC model. The validation set accuracy reaches 75.6%, the highest of any model so far.

```
##          cost     gamma
## 101 0.8858668 0.4641589
```

```
##
## svmR.pred   0   1
##         0 172  55
##         1  62 191
```

```
## [1] SVM (Radial) Accuracy: 0.75625
```

Polynomial SVM The polynomial SVM is fit like the rest, using cross validation to select both cost and degree. The best model here uses cost=1000 and degree=2.371. The cost parameter is much higher than the other models, and the resulting accuracy of this SVM is 61.0%, doing much worse than the other two support vector methods. The prediction table shows that when the true value is 0, the model has a hard time figuring this out, guessing 1 and 0 almost equally. The other two support vector models did not have this problem.

```
##    cost   degree
## 20 1000 2.371374
```

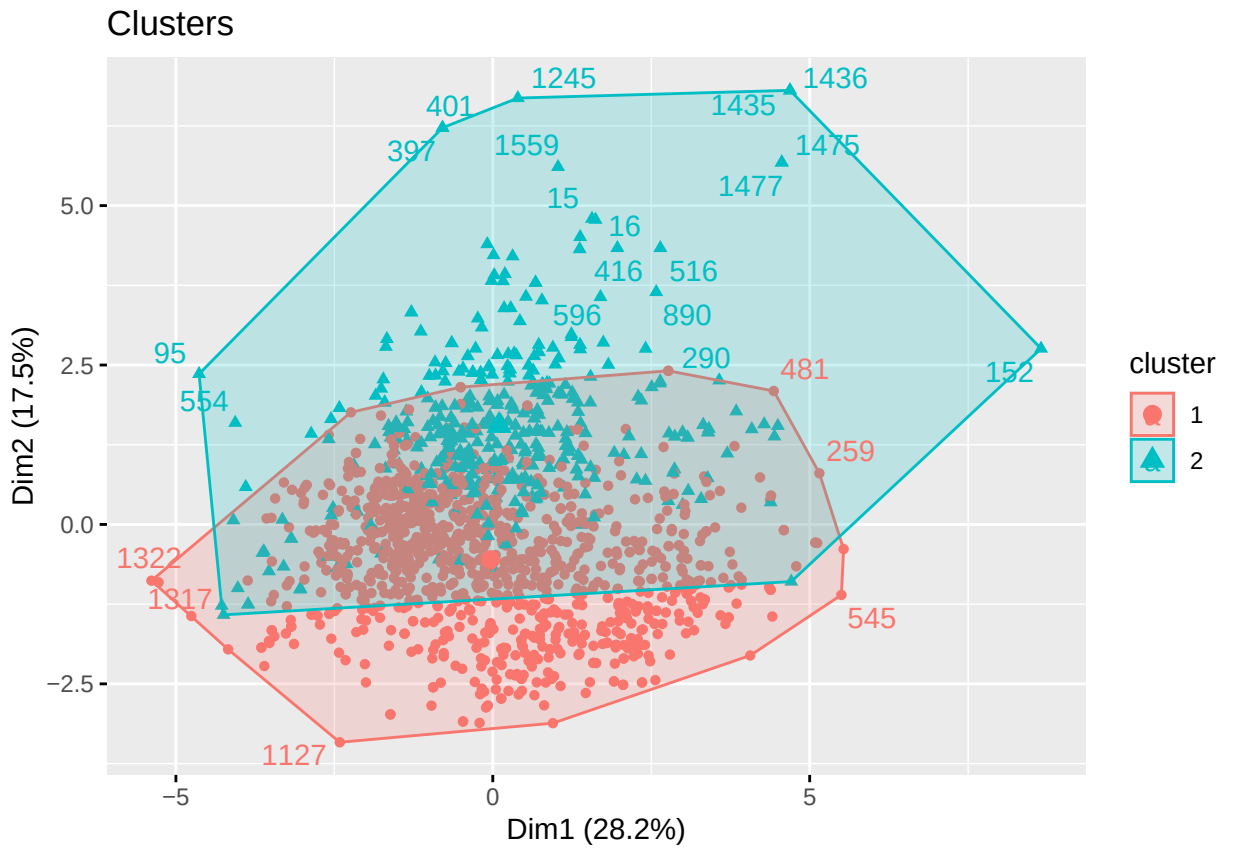
```
##
## svmP.pred   0   1
##         0 120  73
##         1 114 173
```

```
## [1] "SVM (Polynomial) Accuracy: 0.610416666666667"
```

K Means Clustering

Finally, I use K Means clustering to see if there are any interesting clusters within the data. I specify two clusters with the aim of observing some kind of separation between good and bad wine. When comparing the quality compositions of each cluster, there appears to be significant separation. Cluster 1 is made up mostly of good wine, while cluster 2 is dominated by bad wine in an almost 2:1 ratio.

```
## [1] Cluster 1
## [1] Good : 0.599660729431722
## [1] Bad : 0.400339270568278
## [1] Cluster 2
## [1] Good : 0.352380952380952
## [1] Bad : 0.647619047619048
```



Multiclass Classification

Going forward, I use the same or similar models to classify the wine into distinct ratings. Since the data is dominated by ratings of 5 and 6, this will likely be reflected in the results. For future comparison with regression methods, I will also evaluate models based on their RMSE. First, I use the mean rating of the training set and calculate the RMSE from using that as a prediction. This results in an RMSE of 0.7629, which I will use as a benchmark for other models.

```
## [1] RMSE from Training Mean: 0.762926955171138
```

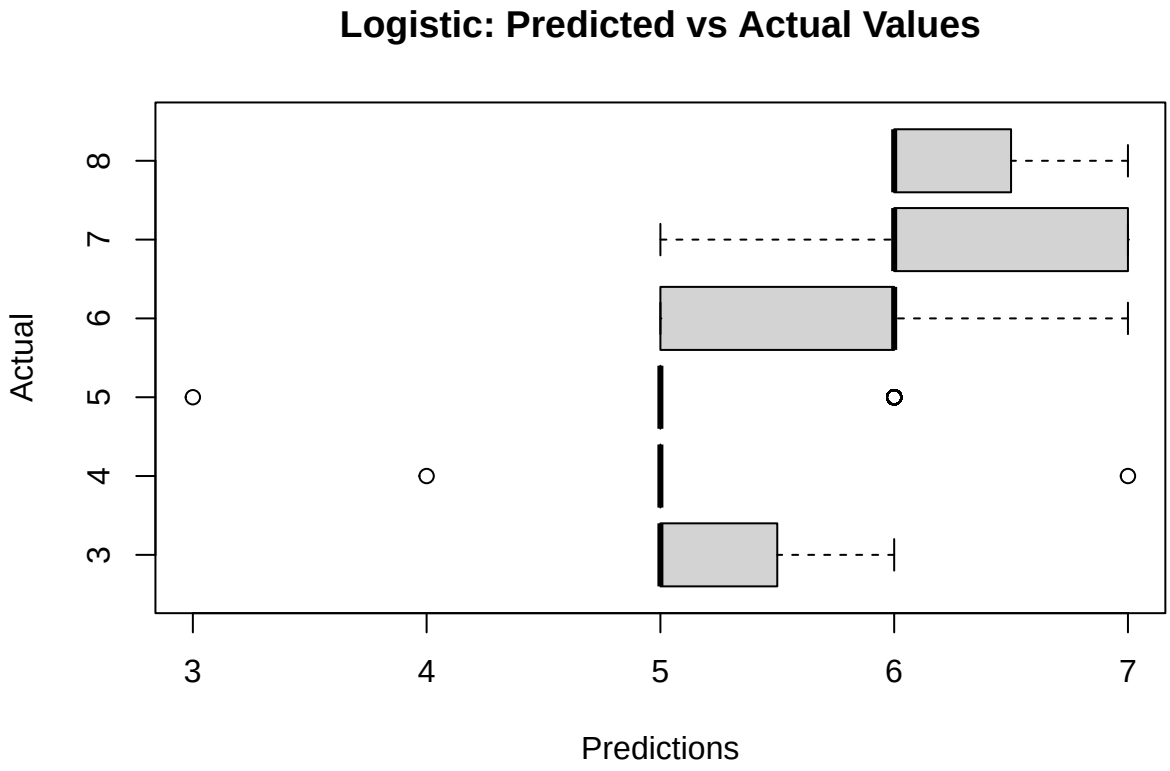
Logistic Regression

Using a multinomial logistic regression, it achieves an accuracy of 63.1%. The prediction tables show how the model is dominated by ratings of 5 and 6, with no guesses for 8 and only one guess for 3. The model is decently accurate for ratings of 7, correctly identifying 20 out of 48 ratings of 7 and being correct in 20 out of 31 predictions for 7. This is despite the fact that these ratings make up a small portion of the data. The RMSE is 0.6630, which is about 0.1 lower than the base RMSE from the mean. The box plot shows the true values against the predicted values. Since the true values are very discrete, a scatter plot is difficult to read. Therefore I use box plots to better represent the distribution of predictions for each true rating. The box plots are by true rating so they can be compared to those for regression, when the predictions are not discrete. We can see from this plot that the model consistently guesses 5s correctly. The plot would ideally show a strong positive correlation between the predictions and actual values, but that isn't exactly the case here.

```
## Call:
## nnet::multinom(formula = .outcome ~ ., data = dat, decay = param$decay)
##
## Coefficients:
##   (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar
## 4   12.268849    -0.3811757        -4.721874   -1.921828     -0.1547333
## 5   -2.818620    -0.2705864        -7.284544   -1.592367     -0.3050784
## 6    0.430362    -0.1574736        -9.915406   -3.143702     -0.3385994
## 7   12.841774    -0.2619752       -11.679394   -2.544609     -0.1568228
## 8   10.752679    -0.6367343        -7.344396    0.349823     -0.3870262
##    chlorides free.sulfur.dioxide total.sulfur.dioxide   density         pH
## 4  -8.620334        -0.046405129          0.034284551 -5.151393  -3.646719
## 5 -11.680105        -0.007591673          0.044732170 21.836353  -5.366425
## 6 -13.582024         0.005062219          0.030420956 11.283595  -5.891750
## 7 -23.495671         0.012391775          0.014603423 -4.315503  -7.123874
## 8 -49.188135         0.011868527          0.007273219  7.980098 -13.088652
##   sulphates   alcohol
## 4  3.518983 1.2671826
## 5  2.977010 0.9748738
## 6  4.951917 1.8788797
## 7  7.685613 2.4666715
## 8  9.192534 3.2262319
##
## Std. Errors:
##   (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar
## 4    2.913908     0.4038921         2.036955    1.779854      0.2670391
## 5    1.725044     0.3662661         1.949473    1.322537      0.2479156
## 6    1.610177     0.3655136         1.981409    1.290608      0.2516612
## 7    1.967323     0.3701594         2.115271    1.385663      0.2564177
## 8    3.566414     0.4128088         2.884842    2.436042      0.3833236
##   chlorides free.sulfur.dioxide total.sulfur.dioxide  density       pH
## 4 3.2444453          0.09536527           0.03797518 2.846711 2.438190
## 5 2.0556699          0.09166381           0.03707850 1.683836 2.100408
## 6 2.0382407          0.09195866           0.03715524 1.576517 2.089259
## 7 3.3797873          0.09285134           0.03757010 1.927947 2.179425
## 8 0.8930737          0.10029939           0.04021570 3.487311 2.890667
##   sulphates   alcohol
## 4  1.650815 0.8442123
## 5  1.268196 0.8257521
## 6  1.251449 0.8271061
## 7  1.311371 0.8317149
## 8  1.865814 0.8827421
##
## Residual Deviance: 2105.208
## AIC: 2225.208


##
## log.pred2   3   4   5   6   7   8
##         3   0   0   1   0   0   0
##         4   0   1   0   0   0   0
##         5   2  12 170  74   1   0
##         6   1   0  46 112  27   2
##         7   0   1   0   9  20   1
##         8   0   0   0   0   0   0
```

```
## [1] Logistic Multiclass Accuracy: 0.63125
```

```
## [1] Logistic RMSE: 0.663010809363869
```



Linear Discriminant Analysis

Reapplying the LDA model, cross validation reports an accuracy of 57.1%. The validation set gives 61.7% and an RMSE of 0.6997. This is worse than the logistic model in both aspects. The plot of the predicted vs actual values is very similar to that for the logistic model. There are a few more predictions on the extreme ends, which show up on the plot as outliers. QDA could not be performed for the multiclass data since one of the classes did not have enough data for the model to be fitted.

```
## Call:
## lda(quality ~ ., data = wine, subset = train)
##
## Prior probabilities of groups:
##           3           4           5           6           7           8
## 0.006255585 0.034852547 0.414655943 0.395889187 0.134941912 0.013404826
##
## Group means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 3      7.785714        0.9678571   0.1114286       2.764286 0.11371429
## 4      7.538462        0.6855128   0.1538462       2.579487 0.09412821
## 5      8.170905        0.5716487   0.2458405       2.560022 0.09334698
```

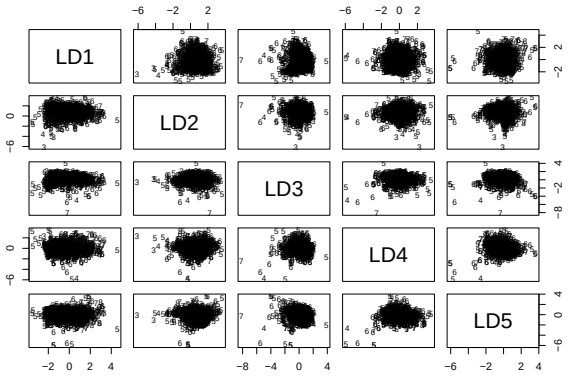
```
## 6      8.392551        0.4939052   0.2788262       2.416027 0.08578555
## 7      8.586755        0.4188079   0.3512583       2.687748 0.07574834
## 8      8.546667        0.4373333   0.3833333       2.560000 0.06733333
##   free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates
## 3            9.285714             24.85714 0.9968571 3.422857 0.5314286
## 4           12.153846             36.25641 0.9962192 3.389231 0.6107692
## 5           17.150862             55.19289 0.9971491 3.305065 0.6243103
## 6           15.393905             39.67720 0.9965642 3.316117 0.6724831
## 7           13.930464             33.86093 0.9959053 3.303179 0.7305960
## 8           13.466667             35.00000 0.9950087 3.271333 0.7740000
##     alcohol
## 3 10.035714
## 4 10.237179
## 5  9.891272
## 6 10.685290
## 7 11.456402
## 8 12.166667
##
## Coefficients of linear discriminants:
##                                  LD1          LD2          LD3           LD4
## fixed.acidity         1.069752e-01  -0.51377054   0.27136542  -0.561896940
## volatile.acidity     -1.990416e+00  -5.32451297  -2.76799377   3.913513272
## citric.acid          -3.273154e-01  -1.82128597  -4.10981077   2.558568678
## residual.sugar        7.414266e-02  -0.35523491  -0.27300905  -0.567223689
## chlorides            -4.855604e+00  -1.85949930   9.74386631  -0.934576993
## free.sulfur.dioxide   7.428088e-03   0.01598614   0.03445718   0.050412048
## total.sulfur.dioxide -1.099192e-02   0.01038235  -0.01874689  -0.008251097
## density              -1.347266e+02 639.27702872 261.89955456 670.579733457
## pH                   -7.450726e-01  -4.32981852   2.98076347  -5.484862004
## sulphates             2.485824e+00  -1.26976448  -1.45779723  -1.665145547
## alcohol               6.888398e-01   0.40075244   0.15124778   0.979901736
##                                 LD5
## fixed.acidity        -0.339598976
## volatile.acidity     -0.061739078
## citric.acid          -3.110575827
## residual.sugar       -0.410469199
## chlorides            -3.207673376
## free.sulfur.dioxide  -0.017593559
## total.sulfur.dioxide  0.008232468
## density              45.121899072
## pH                   -4.004127394
## sulphates             1.170764273
## alcohol               0.280126741
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5
## 0.8275 0.1245 0.0346 0.0098 0.0036

## Linear Discriminant Analysis
##
## 1599 samples
##   11 predictor
##    6 classes: '3', '4', '5', '6', '7', '8'
##
```

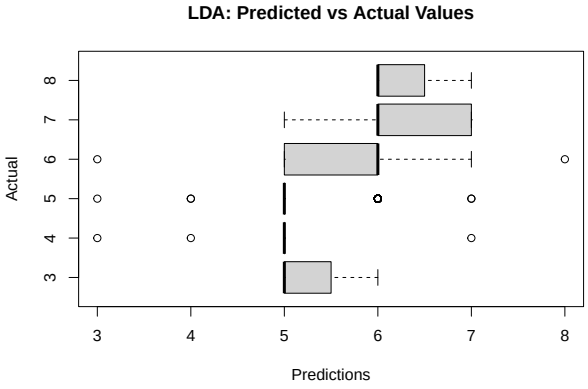
```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1009, 1005, 1008, 1008, 1008, 1006, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5709423  0.3171422
```

```
##
## lda.pred2   3   4   5   6   7   8
##         3   0   1   1   1   0   0
##         4   0   1   2   0   0   0
##         5   2  11 167  74   1   0
##         6   1   0  45 107  26   2
##         7   0   1   2  12  21   1
##         8   0   0   0   1   0   0
```

```
## [1] LDA Accuracy: 0.616666666666667
```

```
## [1] LDA RMSE: 0.699702317656111
```





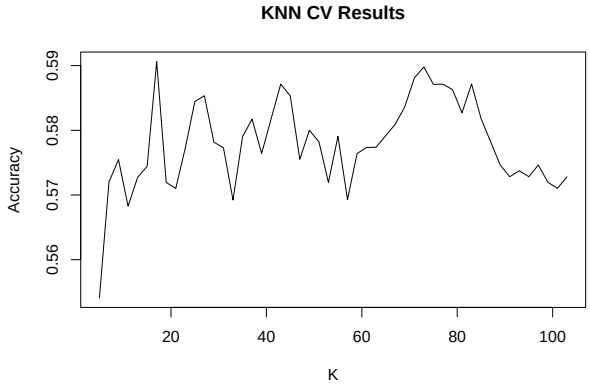
K Nearest Neighbors

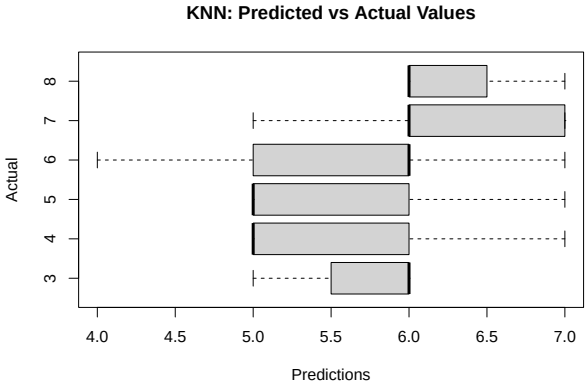
The KNN model is again applied with cross validation over a range of K values. This time, the best parameter was selected as K=17 with a training accuracy of 59.1%. The validation set accuracy is 58.3%, and the RMSE is 0.7486. This is much worse than the previous methods and is close to the mean guess. This is consistent with the previous finding that the relationship between quality and the variables is more linear. The plot shows decent performance for ratings of 5 and 6, but the other ratings are all over the place.

```
##
## predK2   3   4   5   6   7   8
##      3   0   0   0   0   0   0
##      4   0   0   0   2   0   0
##      5   1   8 148  63   3   0
##      6   2   5  67 114  27   2
##      7   0   1   2  16  18   1
##      8   0   0   0   0   0   0
```

```
## [1] KNN Accuracy: 0.583333333333333
```

```
## [1] "KNN RMSE: 0.748609822715857"
```





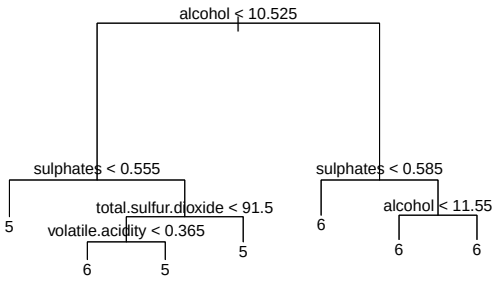
Classification Tree

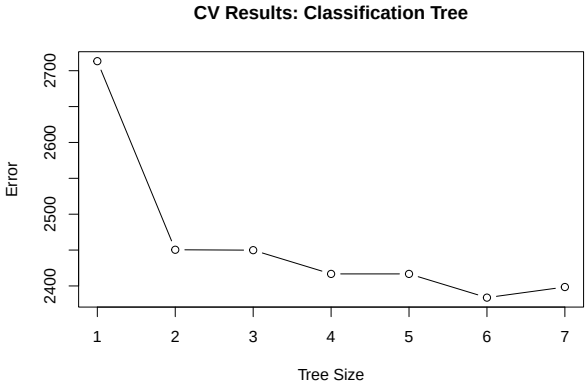
The classification tree uses the same variables as before. The tree mainly indicates that greater alcohol content leads to higher ratings. Cross validation shows that pruning the tree to a size of 6 is best. This makes alcohol the only variable in the tree that changes the outcome. There are several sibling terminal nodes that have the same outcome. This pruned tree has an accuracy of 56.0% and an RMSE of 0.7624. This is about 0.005 lower than the mean guess RMSE, so this is not a good model. It is worth noting that the mean guess RMSE might have a small benefit in not being an integer, whereas the classification techniques only guess integers. This would allow the mean to be closer on average to actual values. This model only makes predictions of 5 or 6 based on the tree diagram, but interestingly the plot of predictions and actual values indicates that there are some predictions for 7. According to the plot, almost all of the true 5 ratings are predicted as such. Most of the 6 ratings, however, are also predicted as 5.

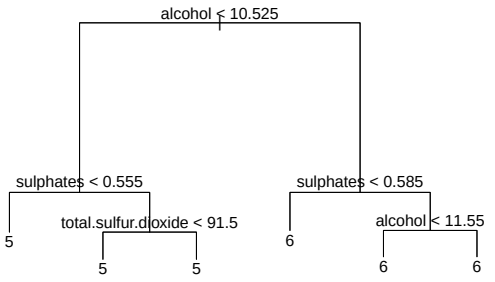
```
##
## Classification tree:
## tree(formula = factor(quality) ~ ., data = wine, subset = train)
## Variables actually used in tree construction:
## [1] "alcohol"              "sulphates"            "total.sulfur.dioxide"
## [4] "volatile.acidity"
## Number of terminal nodes:  7
## Residual mean deviance:  1.962 = 2182 / 1112
## Misclassification error rate: 0.4245 = 475 / 1119
```

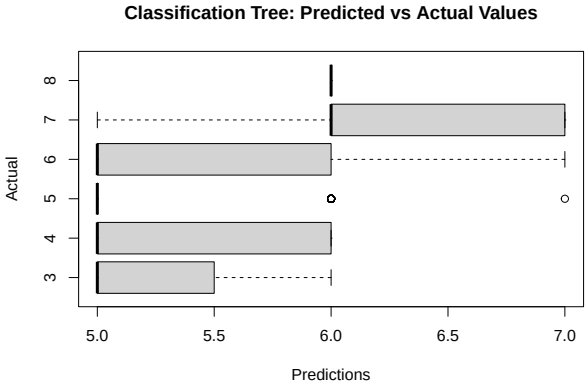
```
## [1] Pruned Accuracy: 0.560416666666667
```

```
## [1] Pruned RMSE: 0.762397534098845
```







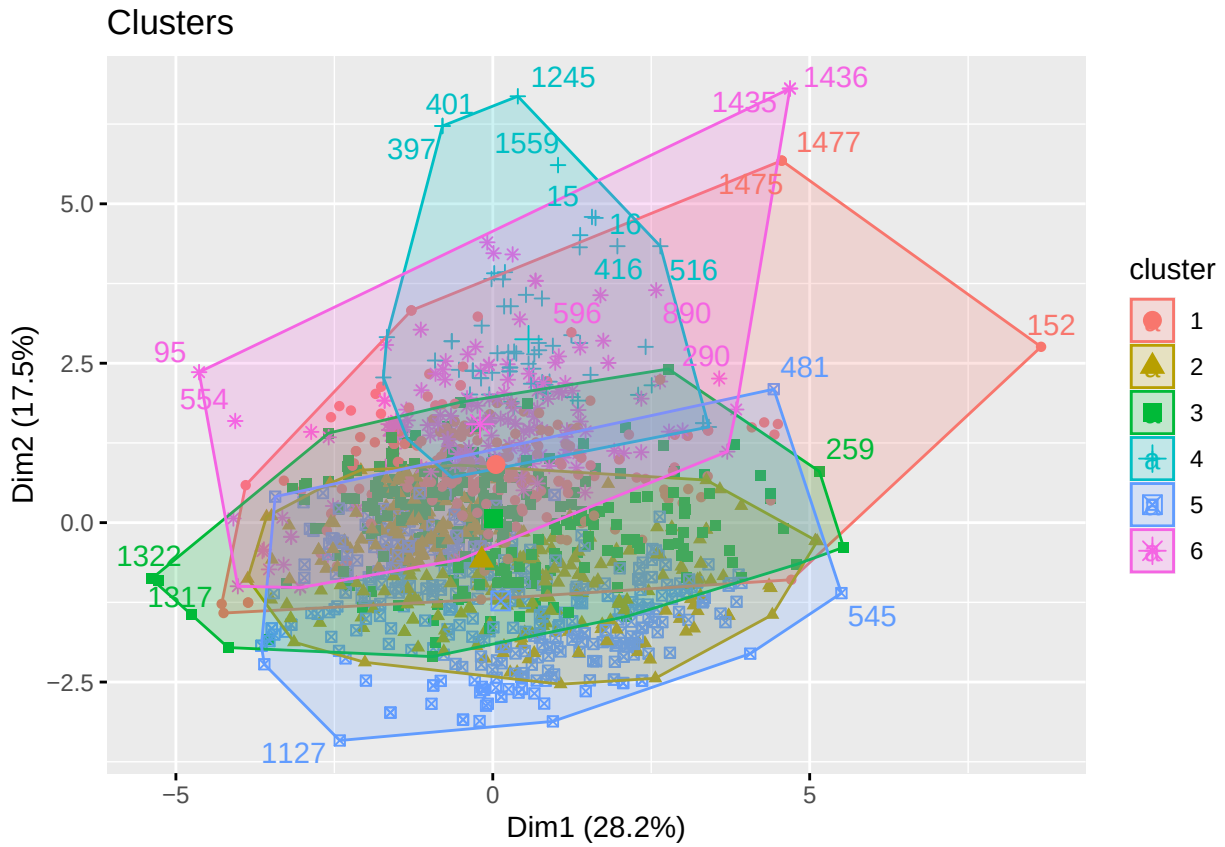


K Means Clustering

I use K Means clustering again but with 6 centers, one for each rating. Looking at the rating composition of each cluster, cluster 1 is very representative of the overall distribution. The other 5 clusters are different. Cluster 2 is heavier on the high ratings, having more 6s and 7s. Cluster 3 is very similar, and cluster 4 is dominated with 87% 5s. Cluster 5 has more 7s than normal (about 20%), and cluster 6 has more (65%) 5s.

```
## [1] Cluster 1
## [1] 3 : 0
## [1] 4 : 0.0289256198347107
## [1] 5 : 0.471074380165289
## [1] 6 : 0.433884297520661
## [1] 7 : 0.0619834710743802
## [1] 8 : 0.00413223140495868
## [1] Cluster 2
## [1] 3 : 0.0025974025974026
## [1] 4 : 0.0311688311688312
## [1] 5 : 0.358441558441558
## [1] 6 : 0.464935064935065
## [1] 7 : 0.135064935064935
## [1] 8 : 0.00779220779220779
## [1] Cluster 3
## [1] 3 : 0.00925925925925926
## [1] 4 : 0.0246913580246914
## [1] 5 : 0.37037037037037
```

```
## [1] 6 : 0.45679012345679
## [1] 7 : 0.12962962962963
## [1] 8 : 0.00925925925925926
## [1] Cluster 4
## [1] 3 : 0
## [1] 4 : 0.0144927536231884
## [1] 5 : 0.869565217391304
## [1] 6 : 0.0869565217391304
## [1] 7 : 0.0289855072463768
## [1] 8 : 0
## [1] Cluster 5
## [1] 3 : 0.015
## [1] 4 : 0.0525
## [1] 5 : 0.3325
## [1] 6 : 0.385
## [1] 7 : 0.1925
## [1] 8 : 0.0225
## [1] Cluster 6
## [1] 3 : 0
## [1] 4 : 0.0223463687150838
## [1] 5 : 0.64804469273743
## [1] 6 : 0.256983240223464
## [1] 7 : 0.0614525139664804
## [1] 8 : 0.0111731843575419
```

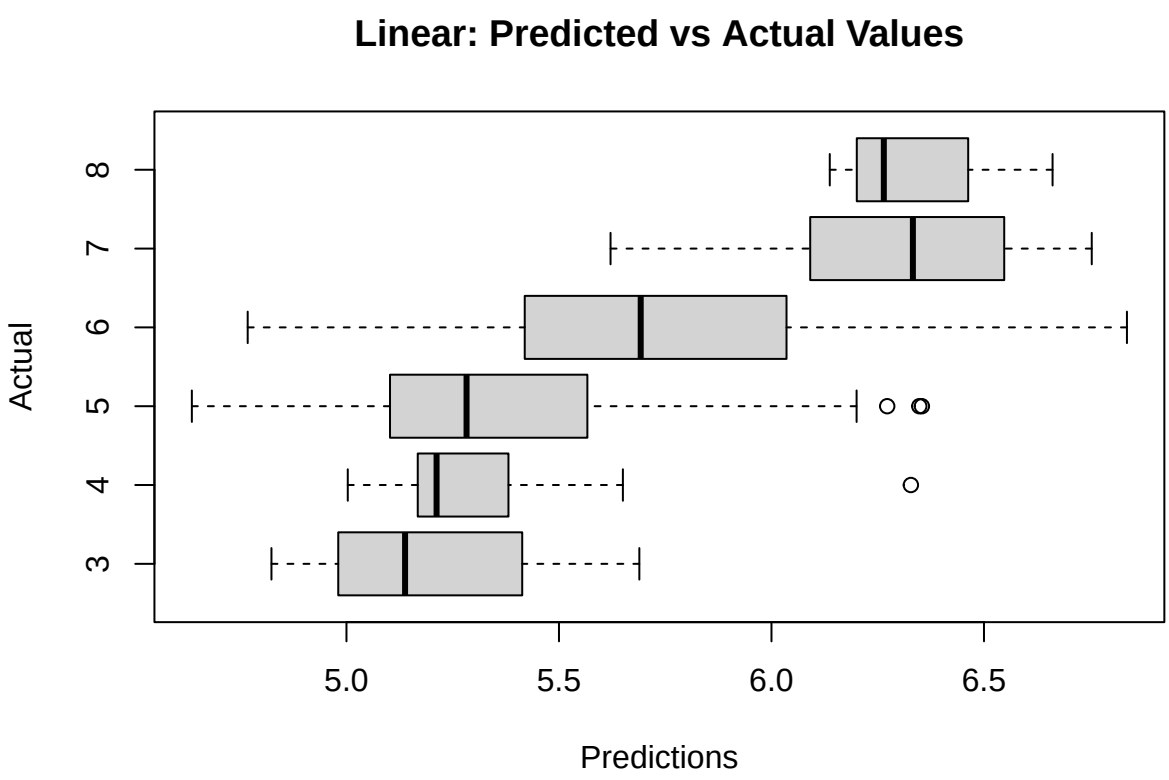


Regression

Linear Regression

Moving from classification to regression, a simple linear regression finds that the most important variables are volatile acidity, chlorides, total sulfur dioxide, sulphates, and alcohol. This is mostly consistent with previous findings. The RMSE from this linear model is 0.6063, which is much better than all previous models. It also strengthens the claim of a linear relationship between ratings and the other variables. The plot shows a much stronger trend than previous models, although the predictions mostly range between 5 and 6.5.

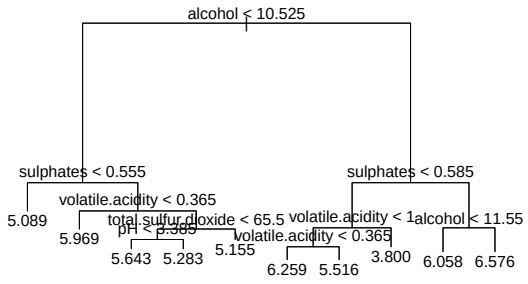
```
##
## Call:
## lm(formula = quality ~ ., data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68911 -0.36652 -0.04699  0.45202  2.02498
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.197e+01  2.119e+01   1.036   0.3002
## fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357
## volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
## citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150
## residual.sugar        1.633e-02  1.500e-02   1.089   0.2765
## chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
## free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *
## total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
## density              -1.788e+01  2.163e+01  -0.827   0.4086
## pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *
## sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
## alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.648 on 1587 degrees of freedom
## Multiple R-squared:  0.3606, Adjusted R-squared:  0.3561
## F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16

## [1] Linear RMSE: 0.606306274202604
```

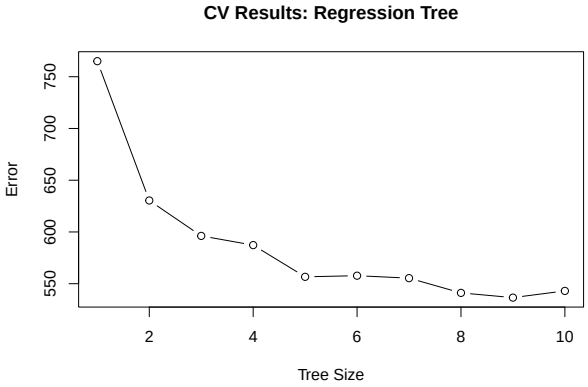


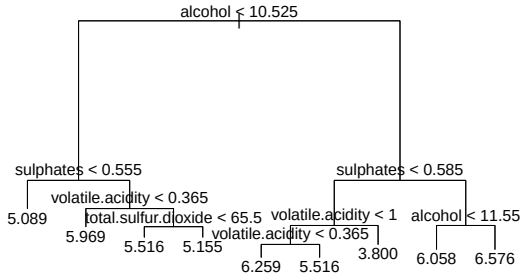
Regression Tree

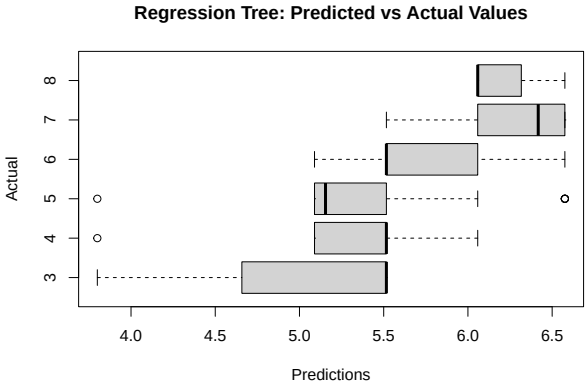
The regression tree finds the same important variables as previous models, with the inclusion of pH. The tree again begins with alcohol content, followed by sulphates. Cross validation finds that the best size is 9, so I prune the tree to this length. The tree shows that greater values in alcohol and sulphates lead to higher ratings, while lower values in volatile acidity and total sulfur dioxide do the same. The RMSE for this tree is 0.6343, which is greater than all previous models but the linear regression. The plot shows a decent upward trend, but not as strong as the linear model's.

```
##
## Regression tree:
## tree(formula = quality ~ ., data = wine, subset = train)
## Variables actually used in tree construction:
## [1] "alcohol"              "sulphates"            "volatile.acidity"
## [4] "total.sulfur.dioxide" "pH"
## Number of terminal nodes:  10
## Residual mean deviance:  0.424 = 470.2 / 1109
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.64300 -0.28280 -0.08929  0.00000  0.42450  2.03100

## [1] Regression Tree RMSE: 0.634284748417547
```





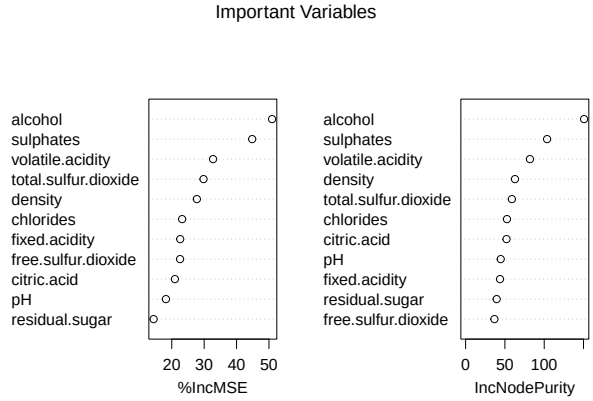


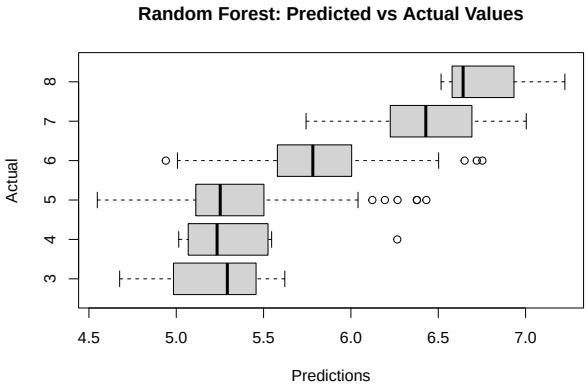


Random Forest

The random forest model indicates that the four most important variables are alcohol, sulphates, volatile acidity, and total sulfur dioxide, which is the same four that have been showing up in previous models. The RMSE on the validation set is 0.5510, which is much lower than all previous models, including the linear model. The plot shows a strong upward trend except at lower levels, where the predictions are very similar for each true rating value.

```
## [1] Random Forest RMSE: 0.55095682751229
```

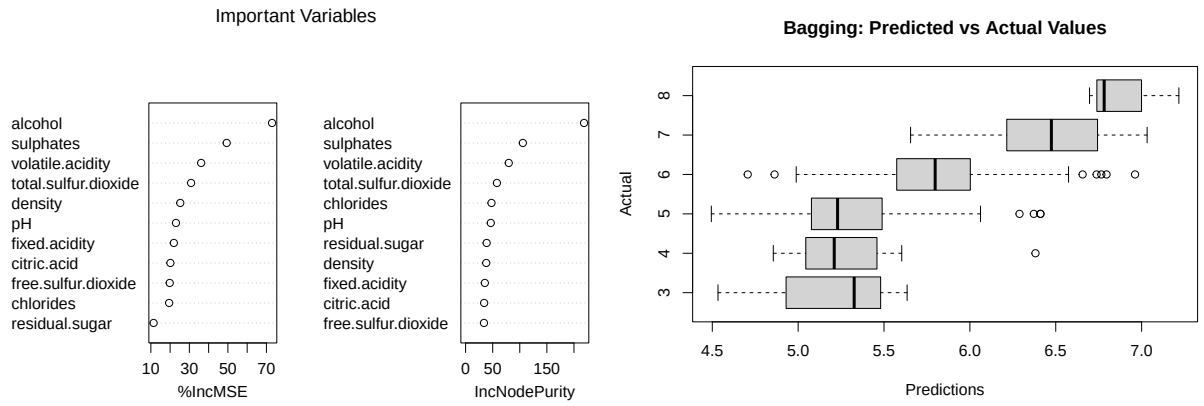




Bagging

Similar to the random forest, bagging gives an RMSE of 0.5521. The top four important variables are identical to that of the random forest model. Lower down, pH is given a higher importance level in this model. The plot of predictions is also very similar to that of the random forest model.

```
## [1] Bagging RMSE: 0.552096112724211
```

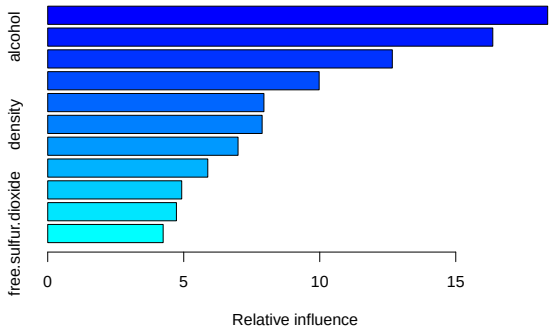


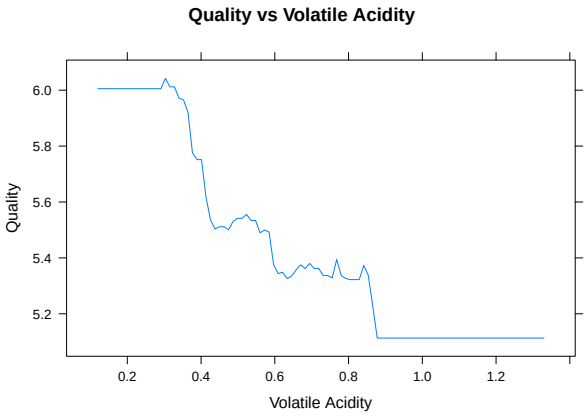
Boosting

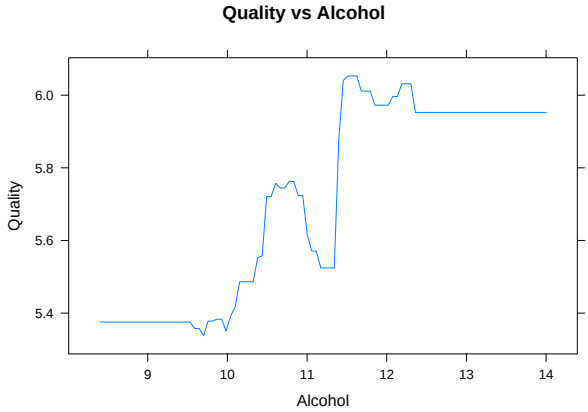
Boosting provides a different list of top four important variables. This time, volatile acidity is at the top of the list, followed by alcohol, sulphates, and chlorides. The plots show that quality increases whith alcohol and sulphates but decreases with volatile acidity. This is consistent with the findings from the regression tree. The boosting RMSE is 0.4942, which is another significant drop from all previous models. The plot of predictions vs actual values has a strong positive trend, and is more distinct at the lower values than the random forest and bagging models.

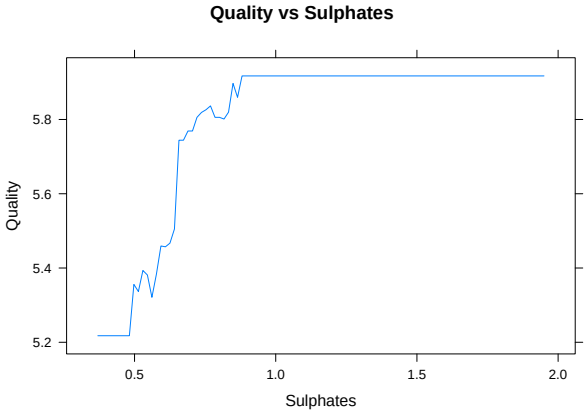
```
##                                       var   rel.inf
## volatile.acidity         volatile.acidity 18.379366
## alcohol                           alcohol 16.361812
## sulphates                       sulphates 12.668108
## chlorides                       chlorides  9.978477
## total.sulfur.dioxide total.sulfur.dioxide  7.948213
## density                           density  7.882166
## citric.acid                   citric.acid  6.996979
## fixed.acidity               fixed.acidity  5.884200
## pH                                     pH  4.925827
## residual.sugar             residual.sugar  4.731237
## free.sulfur.dioxide   free.sulfur.dioxide  4.243616
```

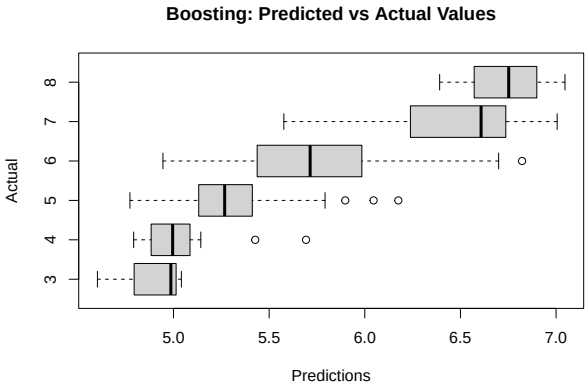
```
## [1] Boost RMSE: 0.494193095778106
```









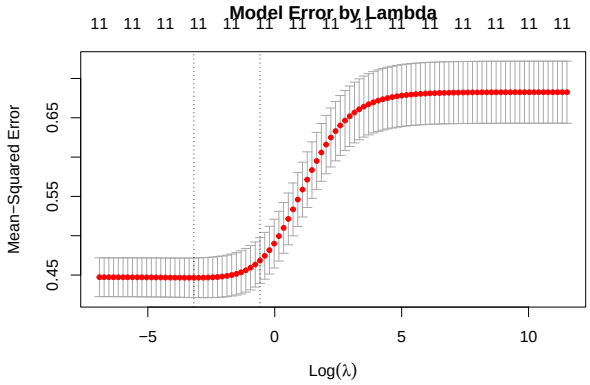


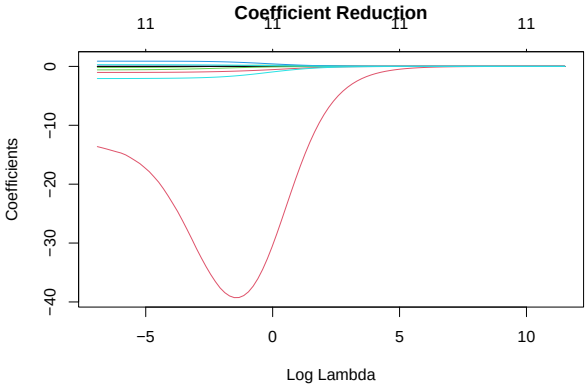
Ridge Regression

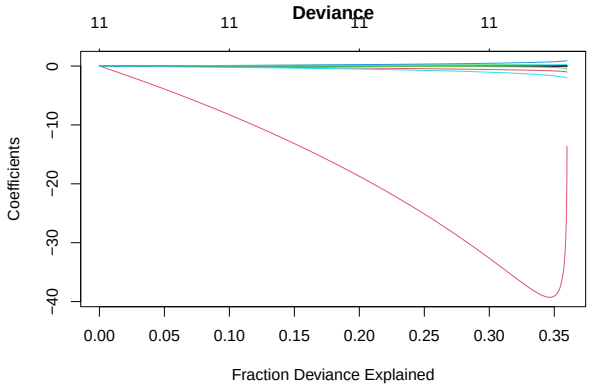
In an attempt to improve the linear regression model, which had a strong RMSE on its own, I will use several regularization techniques. Ridge regression finds the best lambda value to be 0.04132. This results in an RMSE of 0.6099, which is slightly higher than the standard linear regression. The plot of predictions vs actual values is very similar to the linear regression as well. This is likely due to a low lambda value, so decreases in variance don't appear to be worth the corresponding increase in bias.

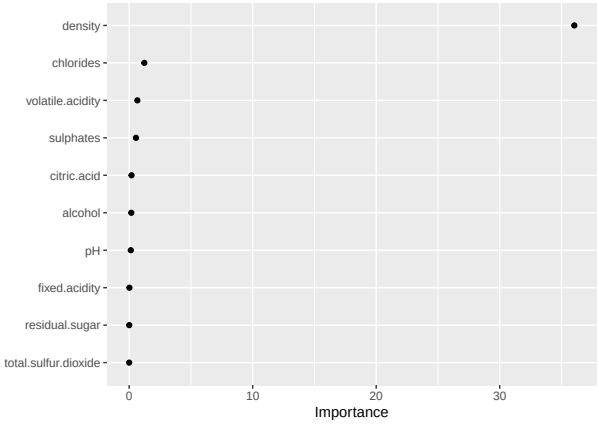
```
## [1] Best Lambda: 0.0413201240011534
```

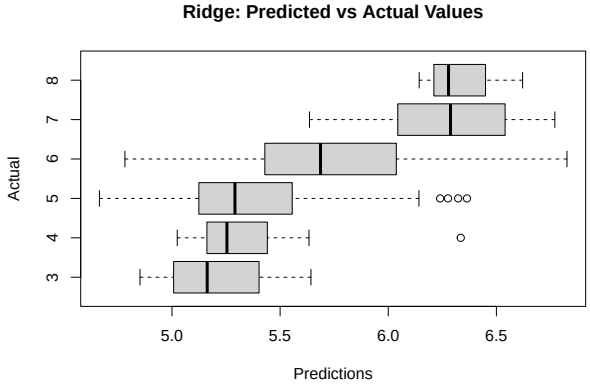
```
## [1] Ridge RMSE: 0.60993564233406
```











Lasso Regression

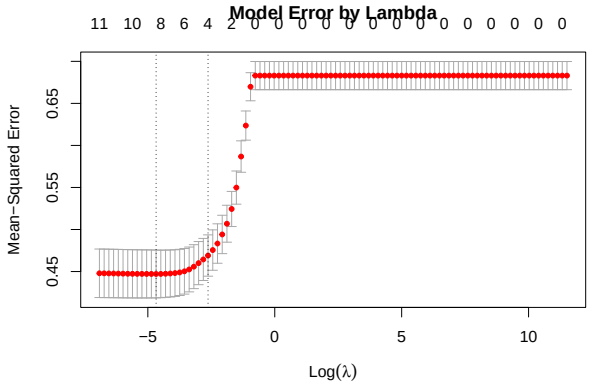
Lasso finds very similar results. It has a low lambda value and similar RMSE at 0.6109. The model reduces fixed acidity, citric acid, and residual sugar coefficients to zero. The plot of predictions is very similar to other models as well. Estimating a separate linear model with the remaining variables gives a similar RMSE of 0.6115.

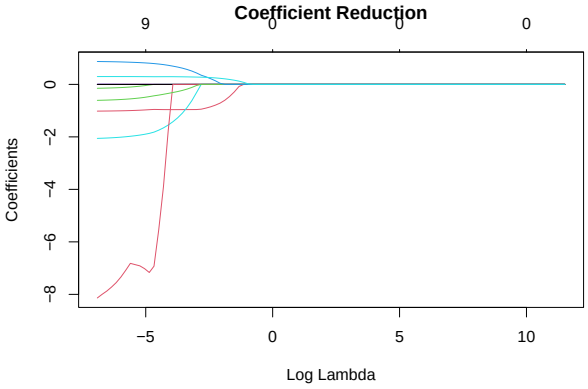
```
## [1] Best Lambda: 0.0093260334688322
```

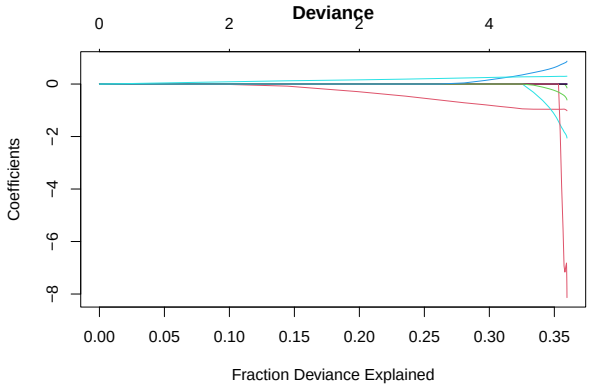
```
## [1] Lasso RMSE: 0.610947415531676
```

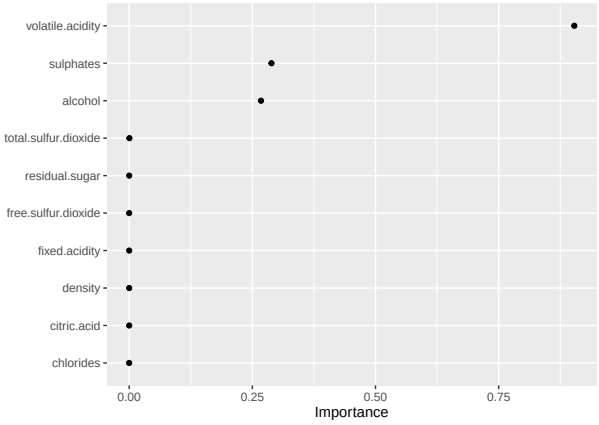
```
## [1] Coefficients Reduced to Zero: fixed.acidity, citric.acid, residual.sugar
```

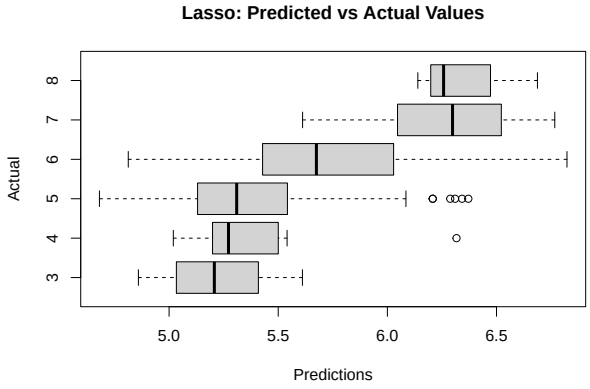
```
## [1] Linear with Lasso Selection RMSE: 0.611494207351364
```











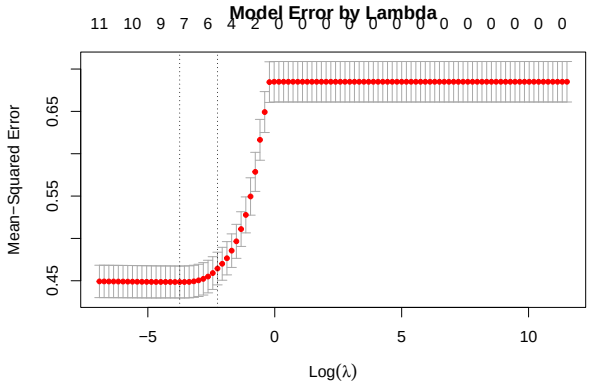
Elastic Net Regression

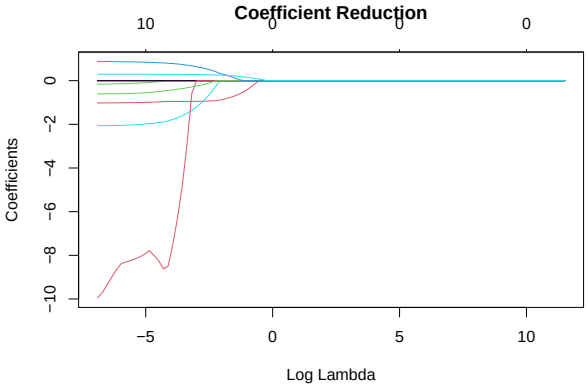
Using elastic net provides similar results as well. It gives a low lambda value (0.0236), reduce the same coefficients to zero, and give an RMSE of 0.6113. The plot of predictions is still very similar.

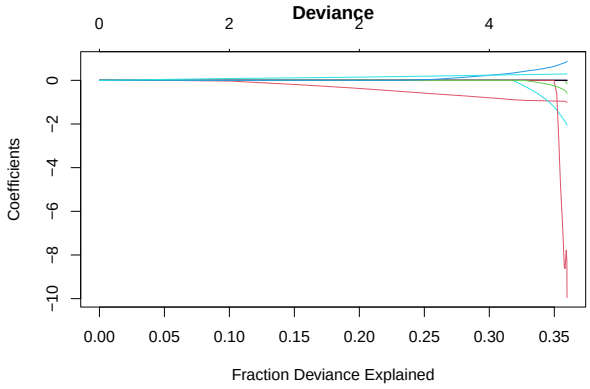
```
## [1] Best Lambda: 0.0236448941264541
```

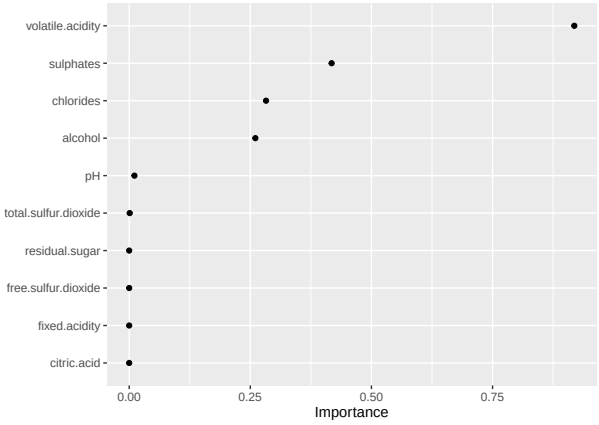
```
## [1] Coefficients Reduced to Zero: fixed.acidity, citric.acid, residual.sugar
```

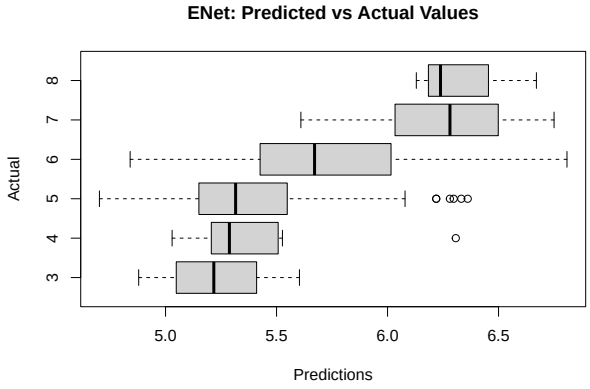
```
## [1] Elastic Net RMSE: 0.611282954713385
```









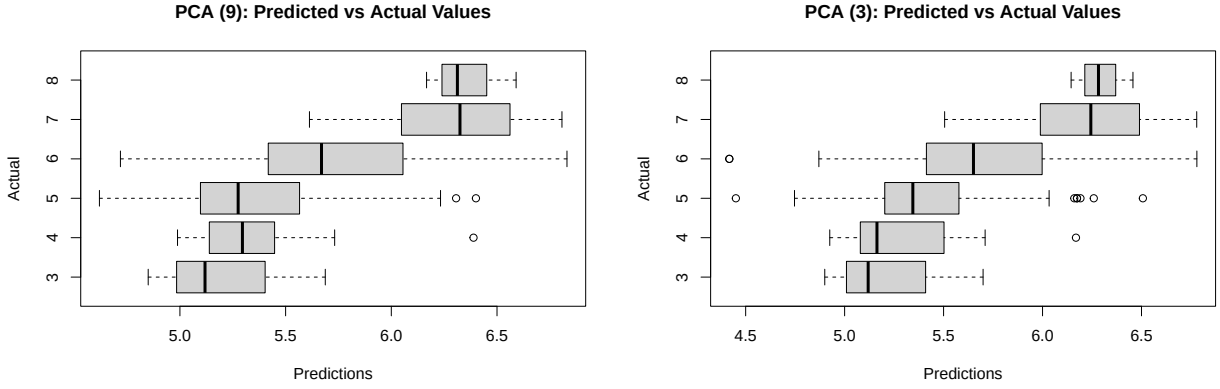


Partial Components Analysis

PCA is used to reduce the number of dimensions without completely annihilating a variable. The model finds that the best cross validated error ocurrs at 9 out of 11 components. It is worth noting, however, that the error is greatly reduced by the third component, accounting for only 60% of the variance in the predictor variables. I will test models with both 9 and 3 components. The PCA model with three components gives an RMSE on the validation set of 0.6327, and the model with nine gives an RMSE of 0.6117. The plots of

their predictions against the actual values are very similar and demonstrate some upward trend with less distinction at the extreme values.

```
## Data:    X dimension: 1119 11
##  Y dimension: 1119 1
## Fit method: svdpc
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8268   0.8240   0.7565   0.6850   0.6827   0.6795   0.6801
## adjCV       0.8268   0.8239   0.7559   0.6847   0.6825   0.6793   0.6800
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV      0.6755   0.6729   0.6684    0.6686    0.6690
## adjCV   0.6753   0.6727   0.6681    0.6683    0.6686
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X        28.3425    45.63    59.74    71.23    80.12    85.89    91.10    94.86
## quality   0.8885    16.59    31.74    32.33    33.02    33.08    34.01    34.67
##          9 comps  10 comps  11 comps
## X          97.90     99.48    100.00
## quality    35.63     35.86     35.98

## [1] PCA (9) RMSE: 0.611712848914302

## [1] PCA (3) RMSE: 0.632661396392125
```

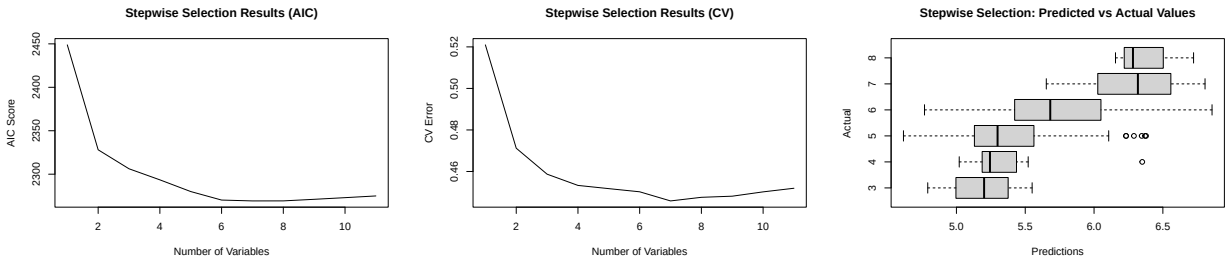


Forward Stepwise Selection (Linear Model)

In order to find the best subset of variables, I use forward stepwise selection, using both AIC and cross validation error. Both methods find that seven variables should be used, and both methods agree on which variables to include. Using this model, the validation set gives an RMSE of 0.6123, which is slightly more than the original linear model but uses four fewer variables. The plot of predictions is very similar to previous models.

```
## [1] Used Variables (AIC): Alcohol, Volatile Acidity, Sulphates, Total Sulfur Dioxide, Chlorides, pH,
```

```
## [1] Unused Variables (AIC): Citric Acid, Density, Residual Sugar, Fixed Acidity

## [1] Used Variables: (CV) Alcohol, Volatile Acidity, Sulphates, Total Sulfur Dioxide, Chlorides, pH, F

##
## Call:
## lm(formula = formula.best, data = wine, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.40615 -0.38512 -0.04607  0.45098  1.94447
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.7834613  0.5546792   8.624  < 2e-16 ***
## alcohol               0.3093595  0.0207331  14.921  < 2e-16 ***
## volatile.acidity     -1.0816790  0.1414013  -7.650 4.35e-14 ***
## sulphates             0.8777227  0.1369913   6.407 2.19e-10 ***
## total.sulfur.dioxide -0.0026454  0.0006513  -4.062 5.21e-05 ***
## chlorides            -2.0273731  0.4760480  -4.259 2.23e-05 ***
## pH                   -0.6080616  0.1602919  -3.793 0.000157 ***
## residual.sugar        0.0024361  0.0150240   0.162 0.871217
## citric.acid          -0.2437766  0.1494258  -1.631 0.103085
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6644 on 1110 degrees of freedom
## Multiple R-squared:  0.3583, Adjusted R-squared:  0.3537
## F-statistic: 77.47 on 8 and 1110 DF,  p-value: < 2.2e-16

## [1] Stepwise Selection RMSE: 0.612328587800291
```

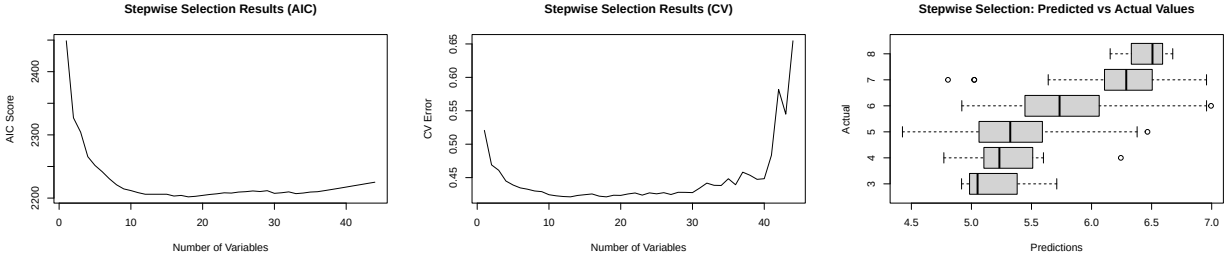


Forward Stepwise Selection (Nonlinear Model)

In an attempt to capture any nonlinear relationships, I use forward stepwise selection with various degrees of each variable, ranging from 1 to 4. I again use both AIC and cross validation error to find the best subset. They again both agree on the same 18 variables, some of which are higher powers of other variables. With all of these, the validation set RMSE is 0.6427, which is worse than a number of other models. The prediction plot shows a strong upward trend across all values. I do not focus on nonlinear models for this data set, since the discrete response variable might cause this to lead to overfitting, and previous models have shown strong linearity in the data as opposed to nonlinear models.

```
## [1] Used Variables (AIC): alcohol, volatile.acidity2, sulphates, sulphates2, pH4, sulphates4, density
```

```
## [1] Used Variables (CV): alcohol, volatile.acidity, sulphates, sulphates2, pH4, fixed.acidity4, sulph

## [1] "Best at 18"

##
## Call:
## lm(formula = formula.best2, data = P, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.34546 -0.39905 -0.01379  0.41564  1.91286
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            6.876e+03  2.726e+03   2.523 0.011789 *
## alcohol                2.598e-01  2.991e-02   8.687  < 2e-16 ***
## volatile.acidity       1.996e-01  4.750e-01   0.420 0.674401
## sulphates              1.020e+01  1.863e+00   5.475 5.41e-08 ***
## sulphates2            -8.345e+00  1.933e+00  -4.318 1.72e-05 ***
## pH4                    6.581e-03  8.655e-03   0.760 0.447221
## fixed.acidity4        -1.521e-03  4.986e-04  -3.050 0.002345 **
## sulphates3             2.027e+00  5.926e-01   3.421 0.000647 ***
## density               -1.034e+04  4.102e+03  -2.520 0.011883 *
## chlorides             -1.510e+00  5.171e-01  -2.921 0.003564 **
## total.sulfur.dioxide  -3.272e-03  1.968e-03  -1.662 0.096771 .
## density3               3.450e+03  1.375e+03   2.509 0.012264 *
## fixed.acidity2        -8.786e-01  2.914e-01  -3.015 0.002632 **
## citric.acid3           1.163e-01  3.609e-01   0.322 0.747280
## fixed.acidity          5.658e+00  1.848e+00   3.062 0.002251 **
## volatile.acidity2     -7.455e-01  3.720e-01  -2.004 0.045301 *
## fixed.acidity3         5.996e-02  1.994e-02   3.006 0.002705 **
## pH                    -1.652e+00  1.317e+00  -1.254 0.210154
## free.sulfur.dioxide    2.503e-03  2.690e-03   0.930 0.352388
## total.sulfur.dioxide2  3.027e-06  1.108e-05   0.273 0.784682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6426 on 1099 degrees of freedom
## Multiple R-squared:  0.4057, Adjusted R-squared:  0.3954
## F-statistic: 39.48 on 19 and 1099 DF,  p-value: < 2.2e-16

## [1] Stepwise Selection (Nonlinear) RMSE: 0.624673536380458
```



Conclusion

At the end, boosting provides by far the most accurate model. It is clear that there is predictive power within the variables, so physico-chemical properties of wine can be used to predict human preference. In addition, several models indicated that the data follows more linear relationships. The results from these models could be used to make more enjoyable wine even before tasting. Improvements could be made if the data included more observations for extreme values, such as amazing or terrible wines. This could allow the models to identify outstanding wines that should be pursued or failures that should not be produced. One of the weaknesses of the data is the domination of the middle values, 5 and 6. Even accounting for this in the beginning, wines can be categorized as above or below average and appropriately categorized with significant accuracy.

Source:

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

Source Code:

```
#load data
wine = read.csv("winequality-red.csv")
head(wine)

#Create labels for each variable
labels = c(fixed.acidity="Fixed Acidity",
           volatile.acidity="Volatile Acidity",
           citric.acid="Citric Acid",
           residual.sugar="Residual Sugar",
           chlorides="Chlorides",
           free.sulfur.dioxide="Free Sulfur Dioxide",
           total.sulfur.dioxide="Total Sulfur Dioxide",
           density="Density",
           pH="pH",
           sulphates="Sulphates",
           alcohol="Alcohol",
           quality="Quality")

#Plot the data for each variable vs quality
for(var in colnames(wine)[colnames(wine)!="quality"]){
  formula = as.formula(paste(var,"~ quality"))
  boxplot(formula, wine, horizontal=T, ylab="Quality",xlab=labels[var], main=labels[var])
}

#plot distribution of ratings
barplot(table(wine$quality), xlab="Quality",ylab="Frequency",main="Frequency of Ratings")

#Create dataframe of predictors, Y, training set
X = wine
X$quality = NULL
Y = wine$quality

set.seed(123)
sampleSize = round(0.7*nrow(wine))
```

```
train = sample((1:nrow(wine)),sampleSize)

#Create a separate data set for binary classification
wineC = wine
wineC$quality = as.numeric(wineC$quality > 5)

#Fit and evaluate a logistic model
model.log = glm(quality ~ ., data = wineC, subset=train, family="binomial")
summary(model.log)
log.pred = round(predict(model.log, wineC[-train,], type="response"))
table(log.pred, wineC[-train,"quality"])
log.acc = mean(log.pred == wineC[-train, "quality"])
print(paste("Logistic Binary Classification Accuracy:",log.acc),quote=F)

#Fit and evaluate an LDA model
train.control = trainControl(method = "cv", number = 10)
lda.fit = lda(quality ~ ., data = wineC, subset=train)
print(lda.fit)
plot(lda.fit)

lda.model = train(factor(quality) ~ ., data=wineC, method="lda", subset=train, trControl=train.control)
print(lda.model)
lda.pred = predict(lda.model, wineC[-train,])
table(lda.pred, wineC[-train,"quality"])
lda.acc = mean(lda.pred == wineC[-train, "quality"])
print(paste("LDA Accuracy:",lda.acc),quote=FALSE)

#Fit and evaluate a QDA model
qda.model = train(factor(quality) ~ ., data=wineC, method="qda", subset=train, trControl=train.control)
print(qda.model)
qda.pred = predict(qda.model, wineC[-train,])
table(qda.pred, wineC[-train,"quality"])
qda.acc = mean(qda.pred == wineC[-train, "quality"])
print(paste("QDA Accuracy:",qda.acc),quote=FALSE)

#Fit and evaluate a KNN model
set.seed(7564)
knn.model = train(factor(quality) ~ ., data=wineC, method="knn", trControl = train.control, subset=trai
print(knn.model)
plot(knn.model$results$k, knn.model$results$Accuracy, type='l', main="KNN CV Results",xlab="K",ylab="Ac
predK = predict(knn.model, wineC[-train,])
table(predK, wineC[-train, "quality"])
accuracyK = mean(predK == wineC[-train, "quality"])
print(paste("KNN Accuracy:",accuracyK),quote=FALSE)

#Fit and evaluate a Binary Classification Tree
set.seed(959)
tree.class = tree(factor(quality) ~ ., data=wineC, subset=train)
summary(tree.class)

plot(tree.class, main="Classification Tree")
text(tree.class, pretty=0)
```

```
treeCV.class = cv.tree(tree.class)
plot(treeCV.class$size, treeCV.class$dev, type="b", main="CV Results: Classification Tree", xlab="Tree S
# Best at 5 or most complicated

prune.class = prune.tree(tree.class, best=5)
plot(prune.class)
text(prune.class, pretty=0)
# Both terminal nodes on the right are 1?

class.pred = predict(tree.class, wineC[-train,], type="class")
class.acc = mean(class.pred == wineC[-train, "quality"])
print(paste("Full Tree Accuracy:",class.acc),quote=F)

prune.pred = predict(prune.class, wineC[-train,], type="class")
prune.acc = mean(prune.pred == wineC[-train, "quality"])
print(paste("Pruned Tree Accuracy:",prune.acc),quote=F)

#Fit and evaluate an SVC model
set.seed(656)
cost = 10^c(-3,-2, seq(-1,2, length=20), 3)
#tune.out = tune(svm, as.factor(quality)~., data=wineC[train,], kernel="linear", ranges=list(cost=cost)
tune.out = readRDS("tuneOutSVC.RData")
tune.out$best.parameters

svm1 = tune.out$best.model
svm.pred = predict(svm1, wineC[-train,])
table(svm.pred, wineC[-train, "quality"])
svm.acc = mean(svm.pred == wineC[-train, "quality"])
print(paste("SVC Accuracy:",svm.acc),quote=F)

#Fit and evaluate a SVM (radial) model
set.seed(650)
cost = 10^c(-3,-2, seq(-1,2, length=20), 3)
gamma = 10^c(-3,-2, seq(-1,2, length=10), 3)

#tune.outR = tune(svm, as.factor(quality)~., data=wineC[train,], kernel="radial", ranges=list(cost=cost
tune.outR = readRDS("tuneOutR.RData")

tune.outR$best.parameters

svmR = tune.outR$best.model
svmR.pred = predict(svmR, wineC[-train,])
table(svmR.pred, wineC[-train, "quality"])
svmR.acc = mean(svmR.pred == wineC[-train, "quality"])
print(paste("SVM (Radial) Accuracy:",svmR.acc),quote=F)

#Fit and evaluate an SVM (polynomial) model
set.seed(659)
degree = 10^c(-3,-2, seq(-1,2, length=20), 3)
degree = 10^seq(-3,1.5, length=5)
cost2 = 10^seq(-3,3,length=5)
#tune.outP = tune(svm, as.factor(quality)~., data=wineC[train,], kernel="polynomial", ranges=list(cost=
tune.outP = readRDS("tuneOutP.RData")
```

```
tune.outP$best.parameters

svmP = tune.outP$best.model
svmP.pred = predict(svmP, wineC[-train,])
table(svmP.pred, wineC[-train, "quality"])
svmP.acc = mean(svmP.pred == wineC[-train, "quality"])
print(paste("SVM (Polynomial) Accuracy:",svmP.acc))

#Fit and analyze a K-Means clustering model
set.seed(4)
totalAccK = c()
C=2
kmean_model = kmeans(wineC[,-12], centers=C)

fviz_cluster(kmean_model, data=wineC[,-12], main="Clusters", repel=TRUE)

results = c(Good=1,Bad=0)
accuracyKM = 0
for(k in (1:C)){
  print(paste("Cluster",k), quote=FALSE)
  result = wineC$quality[kmean_model$cluster == k]
  choice = c()
  for(res in names(results)){
    choice[res] = sum(result == results[res])/length(result)
    print(paste(names(results[which(results==results[res])]),":",choice[res]),quote=FALSE)
  }
  choice1 = names(choice[which(choice==max(choice))])
  accuracyKM = accuracyKM + sum(result == results[choice1])
}

#Use the mean quality to find a benchmark RMSE
set.seed(555)
#Mean benchmark:
mean.pred = rep(mean(wine[train,"quality"]),nrow(wine)-sampleSize)
mean.RMSE = sqrt(mean((mean.pred-wine[-train,"quality"])^2))
print(paste("RMSE from Training Mean:",mean.RMSE),quote=F)

#Fit a multinomial logistic model
model.log2 = train(factor(quality) ~ ., data=wine, method="multinom", trControl = train.control, subset=

#Create a function to convert factors to numbers
factorToNum = function(v){
  return(as.numeric(as.character(v)))
}

#Evaluate model
summary(model.log2)
log.pred2 = predict(model.log2, wine[-train,])
table(log.pred2, wine[-train,"quality"])
log.acc2 = mean(log.pred2 == wine[-train, "quality"])
print(paste("Logistic Multiclass Accuracy:",log.acc2),quote=F)

log.RMSE = sqrt(mean((factorToNum(log.pred2) - wine[-train, "quality"])^2))
```

```
print(paste("Logistic RMSE:",log.RMSE),quote=F)

boxplot(factorToNum(log.pred2) ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual"

#Fit and evaluate a multi class LDA model
lda.fit2 = lda(quality ~ ., data = wine, subset=train)
print(lda.fit2)
plot(lda.fit2)

lda.model2 = train(factor(quality) ~ ., data=wine, method="lda", subset=train, trControl=train.control)
print(lda.model2)
lda.pred2 = predict(lda.model2, wine[-train,])
table(lda.pred2, wine[-train,"quality"])
lda.acc2 = mean(lda.pred2 == wine[-train, "quality"])
print(paste("LDA Accuracy:",lda.acc2),quote=FALSE)

lda.RMSE = sqrt(mean((factorToNum(lda.pred2) - wine[-train,"quality"])^2))
print(paste("LDA RMSE:",lda.RMSE),quote=F)
boxplot(factorToNum(lda.pred2) ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual"

#Fit and evaluate a multiclass KNN model
set.seed(340)
knn.model2 = train(factor(quality) ~ ., data=wine, method="knn", trControl = train.control, subset=trai
print(knn.model2)
plot(knn.model2$results$k, knn.model2$results$Accuracy, type='l', ylab="Accuracy",xlab="K",main="KNN CV
predK2 = predict(knn.model2, wine[-train,])
table(predK2, wine[-train, "quality"])
accuracyK2 = mean(predK2 == wine[-train, "quality"])
print(paste("KNN Accuracy:",accuracyK2),quote=FALSE)

knn.RMSE = sqrt(mean((factorToNum(predK2) - wine[-train, "quality"])^2))
print(paste("KNN RMSE:",knn.RMSE))
plot(predK2, wine[-train,"quality"], xlab="Predictions",ylab="Actual",main="KNN: Predicted vs Actual Va
boxplot(factorToNum(predK2) ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",ma

#Fit and evaluate a multiclass classification tree
set.seed(959)
tree.class2 = tree(factor(quality) ~ ., data=wine, subset=train)
summary(tree.class2)

plot(tree.class2, main="Classification Tree")
text(tree.class2, pretty=0)

treeCV.class2 = cv.tree(tree.class2)
plot(treeCV.class2$size, treeCV.class2$dev, type="b", main="CV Results: Classification Tree", xlab="Tre
# Best at 6

prune.class2 = prune.tree(tree.class2, best=6)
plot(prune.class2)
text(prune.class2, pretty=0)
# Both terminal nodes on the right are 1?

prune.pred2 = predict(prune.class2, wine[-train,], type="class")
```

```
prune.acc2 = mean(prune.pred2 == wine[-train, "quality"])
print(paste("Pruned Accuracy:",prune.acc2),quote=F)

prune.RMSE = sqrt(mean((factorToNum(prune.pred2) - wine[-train,"quality"])^2))
print(paste("Pruned RMSE:",prune.RMSE),quote=F)

boxplot(factorToNum(prune.pred2) ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual

#Fit and analyze K-means clusters with 6 centers
set.seed(5556)
kmean_model2 = kmeans(wine[,-12], centers=6)

fviz_cluster(kmean_model2, data=wine[,-12], main="Clusters", repel=TRUE)

results = (3:8)
for(k in (1:6)){
  print(paste("Cluster",k), quote=FALSE)
  result = wine$quality[kmean_model2$cluster == k]
  for(res in results){
    frequ = sum(result == res)/length(result)
    print(paste(res,":",frequ),quote=FALSE)
  }
}

#Fit and evaluate a linear model for a benchmark in regression
lin_model = lm(quality ~., data=wine)
summary(lin_model)
lin.pred = predict(lin_model, newdata=wine[-train,])
lin.RMSE = sqrt(mean((lin.pred-wine[-train,"quality"])^2))
print(paste("Linear RMSE:",lin.RMSE),quote=F)
boxplot(lin.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Linear:

#Fit and evaluate a regression tree
set.seed(9595)
tree.reg = tree(quality ~ ., data=wine, subset=train)
summary(tree.reg)

plot(tree.reg, main="Classification Tree")
text(tree.reg, pretty=0)

treeCV.reg = cv.tree(tree.reg)
plot(treeCV.reg$size, treeCV.reg$dev, type="b", main="CV Results: Classification Tree", xlab="Tree Size
# Best at 9

prune.reg = prune.tree(tree.reg, best=9)
plot(prune.reg)
text(prune.reg, pretty=0)
# Both terminal nodes on the right are 1?

prune.pred3 = predict(prune.reg, wine[-train,])
prune.RMSE3 = sqrt(mean((prune.pred3 - wine[-train,"quality"])^2))
print(paste("Regression Tree RMSE:",prune.RMSE3),quote=F)
```

```
boxplot(prune.pred3 ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Regr

#Fit and evaluate random forest model
set.seed(321)
forest = randomForest(quality ~ ., data=wine, subset=train, importance=T)
forest.pred = predict(forest, newdata=wine[-train,])
forest.RMSE = sqrt(mean((forest.pred-wine[-train, "quality"])^2))
print(paste("Random Forest RMSE:",forest.RMSE),quote=F)
varImpPlot(forest, main="Important Variables")

boxplot(forest.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Rando

#Fit and evaluate bagging model
set.seed(3231)
bag = randomForest(quality ~ ., data=wine, subset=train, importance=T, mtry=ncol(X))
bag.pred = predict(bag, newdata=wine[-train,])
bag.RMSE = sqrt(mean((bag.pred-wine[-train, "quality"])^2))
print(paste("Bagging RMSE:",bag.RMSE),quote=F)
varImpPlot(bag, main="Important Variables")

boxplot(bag.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Bagging

#Fit and evaluate boosting model
set.seed(3431)
boost = gbm(quality ~.,data=wine[-train,], distribution="gaussian",n.trees=200, interaction.depth=4, cv
summary(boost)

plot(boost, i="volatile.acidity", main="Quality vs Volatile Acidity", xlab="Volatile Acidity",ylab="Qual
plot(boost, i="alcohol", main="Quality vs Alcohol", xlab="Alcohol",ylab="Quality")
plot(boost, i="sulphates", main="Quality vs Sulphates",xlab="Sulphates",ylab="Quality")

boost.pred = predict(boost, wine[-train,])
boost.RMSE = sqrt(mean((boost.pred-wine[-train, "quality"])^2))
print(paste("Boost RMSE:",boost.RMSE),quote=F)
boxplot(boost.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Boosti

#Fit ridge regression model
set.seed(2000)

grid = 10^seq(-3,5,length=100)
model_ridge = cv.glmnet(x = as.matrix(X[train,]), y=Y[train], alpha=0, lambda=grid, standardize=TRUE)
plot(model_ridge, main="Model Error by Lambda")
plot(model_ridge$glmnet.fit, "lambda", label=FALSE, main="Coefficient Reduction")
plot(model_ridge$glmnet.fit,xvar="dev",label=FALSE, main="Deviance")

vip(model_ridge, num_features = 10, geom = "point", main="Most Important Variables")
lambda_ridge = model_ridge$lambda.min
print(paste("Best Lambda:",model_ridge$lambda.min),quote=FALSE)

#Evaluation of best lambda
model_cv_ridge = glmnet(x=as.matrix(X[train,]), y=Y[train], alpha=0, lambda = lambda_ridge, standardize=
ridge.pred = predict(model_cv_ridge, as.matrix(X[-train,]))
ridge.RMSE = sqrt(mean((ridge.pred - Y[-train])^2))
```

```
print(paste("Ridge RMSE:",ridge.RMSE),quote=F)

boxplot(ridge.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Ridge

#Fit lasso regression model
model.lasso = cv.glmnet(x = as.matrix(X[train,]), y=Y[train], alpha=1, lambda=grid, standardize=TRUE)
plot(model.lasso, main="Model Error by Lambda")
plot(model.lasso$glmnet.fit, "lambda", label=FALSE, main="Coefficient Reduction")
plot(model.lasso$glmnet.fit,xvar="dev",label=FALSE, main="Deviance")

vip(model.lasso, num_features = 10, geom = "point", main="Most Important Variables")
lambda.lasso = model.lasso$lambda.min
print(paste("Best Lambda:",model.lasso$lambda.min),quote=FALSE)

#Evaluation of best lambda
model_cv_lasso = glmnet(x=as.matrix(X[train,]), y=Y[train], alpha=1, lambda = lambda.lasso, standardize=
lasso.pred = predict(model_cv_lasso, as.matrix(X[-train,]))
lasso.RMSE = sqrt(mean((lasso.pred - Y[-train])^2))
print(paste("Lasso RMSE:",lasso.RMSE),quote=F)

variable_importanceL = coef(model.lasso, s='lambda.min')
ordered_var_imp_lasso = variable_importanceL[,1][order(-abs(variable_importanceL[,1]))]
lasso.big = rownames(as.data.frame(ordered_var_imp_lasso[ordered_var_imp_lasso != 0]))
lasso.zero = rownames(as.data.frame(ordered_var_imp_lasso[ordered_var_imp_lasso == 0]))
print(paste("Coefficients Reduced to Zero:",paste(lasso.zero,collapse=", ")), quote=F)

boxplot(lasso.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Lasso

#linear model without removed:
lassoData = wine[,c(lasso.big[-1], "quality")]
model.lasso2 = glm(quality ~ ., data=lassoData, subset=train)
lasso.pred2 = predict(model.lasso2, wine[-train,])
lasso.RMSE2 = sqrt(mean((lasso.pred2 - Y[-train])^2))
print(paste("Linear with Lasso Selection RMSE:",lasso.RMSE2),quote=F)

#Fit ENet regression model
set.seed(787)
model.enet = cv.glmnet(x = as.matrix(X[train,]), y=Y[train], alpha=0.5, lambda=grid, standardize=TRUE)
plot(model.enet, main="Model Error by Lambda")
plot(model.enet$glmnet.fit, "lambda", label=FALSE, main="Coefficient Reduction")
plot(model.enet$glmnet.fit,xvar="dev",label=FALSE, main="Deviance")

vip(model.enet, num_features = 10, geom = "point", main="Most Important Variables")
lambda.enet = model.enet$lambda.min
print(paste("Best Lambda:",model.enet$lambda.min),quote=FALSE)

variable_importanceE = coef(model.enet, s='lambda.min')
ordered_var_imp_enet = variable_importanceE[,1][order(-abs(variable_importanceE[,1]))]
enet.big = rownames(as.data.frame(ordered_var_imp_enet[ordered_var_imp_enet != 0]))
enet.zero = rownames(as.data.frame(ordered_var_imp_enet[ordered_var_imp_enet == 0]))
print(paste("Coefficients Reduced to Zero:",paste(enet.zero, collapse=", ")),quote=F)

#Evaluation of best lambda
```

```
model_cv_enet = glmnet(x=as.matrix(X[train,]), y=Y[train], alpha=0.5, lambda = lambda.enet, standardize=
enet.pred = predict(model_cv_enet, as.matrix(X[-train,]))
enet.RMSE = sqrt(mean((enet.pred - Y[-train])^2))
print(paste("Elastic Net RMSE:",enet.RMSE),quote=F)

boxplot(enet.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="ENet: R

#Fit and evaluate PCA model
set.seed(2002)
pcr.fit = pcr(quality ~ ., data=wine, scale=T, validation="CV", subset=train)
summary(pcr.fit)

pcr.pred1 = predict(pcr.fit, wine[-train,], ncomp=9)
pcr.RMSE1 = sqrt(mean((pcr.pred1-Y[-train])^2))
print(paste("PCA (9) RMSE:",pcr.RMSE1),quote=F)

pcr.pred2 = predict(pcr.fit, wine[-train,], ncomp=3)
pcr.RMSE2 = sqrt(mean((pcr.pred2-Y[-train])^2))
print(paste("PCA (3) RMSE:",pcr.RMSE2),quote=F)

boxplot(pcr.pred1 ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="PCA (9

boxplot(pcr.pred2 ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="PCA (3

#Use stepwise selection (AIC)
set.seed(655)
used = c()
variables = colnames(X)
aicTotal = c()
for(i in (1:ncol(X))){
  aicStep = c()
  for(variable in variables[!(variables %in% used)]){
    formula = as.formula(paste("quality~",paste(used, collapse="+"),"+",variable, sep=""))
    model = glm(formula, data=wine, subset=train)
    aicStep[variable] = AIC(model)
  }
  pick = names(aicStep[which(aicStep == min(aicStep))])
  used[i] = pick
  aicTotal[i] = aicStep[pick]
}
plot(aicTotal, type='l', main="Stepwise Selection Results (AIC)",xlab="Number of Variables",ylab="AIC Sc
print(paste("Used Variables (AIC):", paste(labels[used[(1:7)]], collapse=", ")), quote=F)
print(paste("Unused Variables (AIC):",paste(labels[used[-(1:7)]], collapse=", ")), quote=F)
#This corresponds to insignificant variables in the initial linear model

#Use stepwise selection (CV)
used = c()
variables = colnames(X)
aicTotal = c()
for(i in (1:ncol(X))){
  aicStep = c()
  for(variable in variables[!(variables %in% used)]){
    formula = as.formula(paste("quality~",paste(used, collapse="+"),"+",variable, sep=""))
```

```
    model = glm(formula, data=wine, subset=train)
    aicStep[variable] = cv.glm(wine[train,], model, K=10)$delta[1] #AIC(model)
  }
  pick = names(aicStep[which(aicStep == min(aicStep))])
  used[i] = pick
  aicTotal[i] = aicStep[pick]
}
plot(aicTotal, type='l', main="Stepwise Selection Results (CV)",xlab="Number of Variables",ylab="CV Erro
print(paste("Used Variables: (CV)", paste(labels[used[(1:7)]], collapse=", ")), quote=F)
#Best at 7

formula.best = as.formula(paste("quality~",paste(used[(1:7)],collapse="+"),"+",variable, sep=""))
step.best = lm(formula.best, data=wine, subset=train)
summary(step.best)
step.pred = predict(step.best, wine[-train,])
step.RMSE = sqrt(mean((step.pred - wine[-train, "quality"])^2))
print(paste("Stepwise Selection RMSE:",step.RMSE),quote=F)


boxplot(step.pred ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Stepwis

#use stepwise selection for nonlinear variables (AIC)
P = X
predictors = colnames(X)
powers = (2:4)
for(pred in predictors){
  for(d in powers){
    cname = paste(pred, d, sep="")
    P[,cname] = P[,pred]^d
  }
}

set.seed(5676)
used = c()
variables = colnames(P)
aicTotal = c()
for(i in (1:ncol(P))){
  aicStep = c()
  for(variable in variables[!(variables %in% used)]){
    formula = as.formula(paste("Y~",paste(used, collapse="+"),"+",variable, sep=""))
    model = lm(formula, data=P, subset=train)
    aicStep[variable] = AIC(model)
  }
  pick = names(aicStep[which(aicStep == min(aicStep))])
  used[i] = pick
  aicTotal[i] = aicStep[pick]
}
plot(aicTotal, type='l',main="Stepwise Selection Results (AIC)",xlab="Number of Variables",ylab="AIC Sc
print(paste("Used Variables (AIC):", paste(used[(1:18)], collapse=", ")), quote=F)

#Use stepwise selection for nonlinear variables (CV)
cvData2 = data.frame(Y,P)
used = c()
```

```
variables = colnames(P)
aicTotal = c()
for(i in (1:ncol(P))){
  aicStep = c()
  for(variable in variables[!(variables %in% used)]){
    formula = as.formula(paste("Y~",paste(used, collapse="+"),"+",variable, sep=""))
    model = glm(formula, data=P, subset=train)
    aicStep[variable] = cv.glm(cvData2[train,], model, K=10)$delta[1]#AIC(model)
  }
  pick = names(aicStep[which(aicStep == min(aicStep))])
  used[i] = pick
  aicTotal[i] = aicStep[pick]
}
plot(aicTotal, type='l', main="Stepwise Selection Results (CV)",xlab="Number of Variables",ylab="CV Erro
print(paste("Used Variables (CV):", paste(used[(1:18)], collapse=", ")), quote=F)

print(paste("Best at",which(aicTotal == min(aicTotal))))
#Best at 18

formula.best2 = as.formula(paste("Y~",paste(used[(1:18)],collapse="+"),"+",variable, sep=""))
step.best2 = lm(formula.best2, data=P, subset=train)
summary(step.best2)
step.pred2 = predict(step.best2, P[-train,])
step.RMSE2 = sqrt(mean((step.pred2 - wine[-train, "quality"])^2))
print(paste("Stepwise Selection (Nonlinear) RMSE:",step.RMSE2),quote=F)

boxplot(step.pred2 ~ wine[-train,"quality"], horizontal=T, xlab="Predictions",ylab="Actual",main="Stepwi
```