Assessment of Wine Quality
STAT4702 Final Project

Jordan Chazin, Nitish Appanasamy, Patrick Rogan, Rohit Gernapudi

Summary

Two datasets containing assessments of Portuguese *Vinho Verde* red and white wines were examined [1]. Wine quality ranging from 1 to 10 (from lowest quality to highest quality) was determined by an average of at least three expert evaluators. Additionally, eleven physicochemical tests were performed on individual wines and the values are reported in the dataset. For each type of wine, we predict its quality by fitting several models (linear, polynomial, spline-based GAM, Random Forest, and KNN) as a function of these physicochemical characteristics. We developed models using the quality as a qualitative variable (classification) as well as quantitative variable (regression). Through our models, we identified important predictors and suggested top performing model for each wine class.

Data Characteristics

Basic summary statistics for all predictors and the response variable were generated, see Table 1. Note the response variable is categorical. However, several regression models were fit assuming a continuous response. This was done to identify relevant predictors and evaluate the structure of the data. These models are discussed in greater detail in later sections.

Attribute	Min	Max	Range	Mean	Median
Fixed Acidity	3.8	15.9	12.1	7.22	7
Volatile Acidity	0.08	1.58	1.50	0.340	0.29
Citric Acid	0	1.66	1.66	0.319	0.31
Residual Sugar	0.6	65.8	65.2	5.44	3.0
Chlorides	0.009	0.611	0.602	0.0560	0.047
Free Sulfur Dioxide	1	289	288	30.5	29
Total Sulfur Dioxide	6	440	434	115.7	118
Density	0.99	1.04	0.05	0.995	0.995
pH	2.72	4.01	1.29	3.218	3.21
Sulphates	0.22	2	1.78	0.531	0.51
Alcohol	8	14.9	6.9	10.49	10.3
Quality	3	9	6	5.8	6

Table 1. Characteristics for combined red and white wines.

Initial Analysis

As we have two different classes of data (Red and White), we first test if the data is linearly separable or not. As an initial indication, we calculated mean quality difference between the two classes by performing a one-way test of ANOVA.

Equation for One way ANOVA : $y_{i,j} = \mu_i + \varepsilon_{i,j}$

where i is $1 \leq i \leq 2$ representing the two classes and j is $1 \leq j \leq N_j$ where Nj represents the number of observations in each class (Red or White). With the help of one way ANOVA, we determined that the mean quality of the two classes were distinct, rejecting the null Hypothesis of both the class can be used as a single dataset. The results of the test are:

One Way ANOVA

Residuals:

Min	1Q	Median	3Q	Max
-2.8779	-0.8779	0.1221	0.3640	3.1221

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.87791	0.01239	474.429	<2e-16 ***
d	-0.24189	0.02497	-9.686	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8671 on 6495 degrees of freedom
Multiple R-squared: 0.01424, Adjusted R-squared: 0.01409
F-statistic: 93.81 on 1 and 6495 DF, p-value: < 2.2e-16

LDA/QDA

In an effort to further investigate the separability of the classes, we performed LDA and QDA on the entire dataset (the combined set of red and white wine) to predict the class of the wine - Red or White.

Prediction of the Wine type using LDA with *all* the predictor variables (i.e. no covariate subselection was performed), *including* the quality variable, resulted in 99.23% being correctly classified. Prediction of wine type using LDA with just quality as the lone predictor variable gives us a correctly classified percentage of 75.3%. This is reflected in summary statistics for red and white wines, see Table 2 and is graphically depicted in Figure 1.

A parallel set of analyses using QDA (instead of LDA) using the predictor variables gave us similar results, with the LDA outperforming the QDA in terms of percentage of correct wine-type classification. This analysis also led us to conclude that using the combined data set as a predictor for classification of the wine quality would be less successful than using just the data set of white wine for predicting white wine quality and red wine for predicting red wine quality respectively.

Attribute	Mean (Red)	σ (Red)		Mean (White)	σ (White)
Fixed Acidity	8.32	1.741		6.85	0.844
Volatile Acidity	0.53	0.179		0.28	0.101
Citric Acid	0.27	0.195		0.33	0.121
Residual Sugar	2.54	1.410		6.39	5.072
Chlorides	0.09	0.047		0.05	0.022
Free Sulfur Dioxide	15.87	10.460		35.31	17.007
Total Sulfur Dioxide	46.48	32.895		138.36	42.498
Density	0.997	0.0019		0.994	0.0030
pH	3.31	0.154		3.19	0.151
Sulphates	0.66	0.170		0.49	0.114
Alcohol	10.42	1.066		10.51	1.231
Quality	5.64	0.808		5.88	0.886

Table 2. Characteristics for red and white wines.

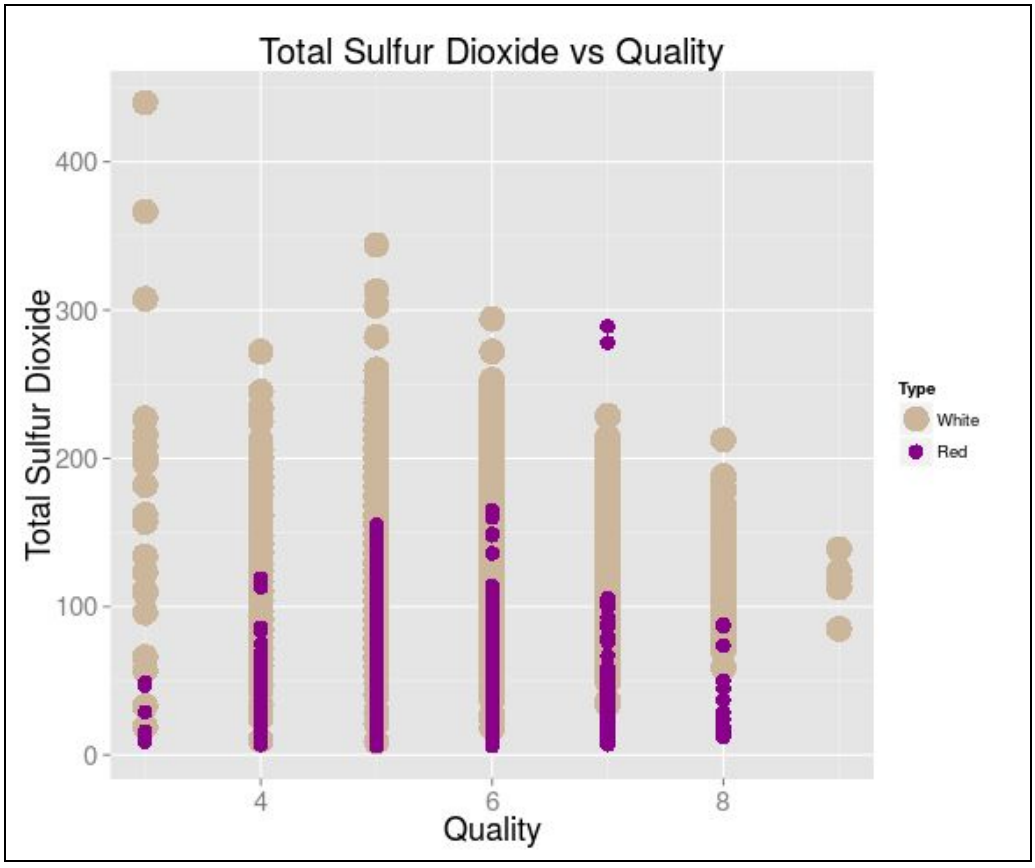

Figure 1. Illustration of partial separability in two dimensions for red and white wine classes.

Conditioning Number Test

We performed a conditioning number test on the data to identify possible presence of multicollinearity in the dataset. Using the *rcond* function in R, the reciprocal of the conditioning number was calculated for the matrix $X^T X$ where X is the matrix of the predictor variables. The reciprocal conditioning number obtained was *8.3412e-08*. The reciprocal conditioning number after centering and scaling the data was *0.0097*. The values of the two reciprocal conditioning numbers indicate the presence of a degree of multicollinearity in the data. As a corrective measure, the best subset selection method was used to choose the best covariates potentially reducing the chances of multicollinearity in the predictor variables data.

Red Wine Analysis

The red wine dataset was split into a training and test sets in an 80:20 ratio to perform the analysis. First, all subset selection methods (Best, Forward, Backward, and Sequential Replacement) were used to identify the subset of predictors that form an effective model. The efficacy of the model is evaluated using the adjusted R^2 value. From all the methods, a subset of eight predictors was found to generate the highest adjusted R^2 value (0.3567). These predictors (*volatile.acidity, citric.acid, chlorides, free.sulfur.dioxide, total.sulfur.dioxide, pH, sulphates,* and *alcohol*) were used in the generation of our subsequent models. Below, we first report the values of best subset selection here, following by our regression and classification analysis of the data. The appendix contains the supporting R code for all the results reported here.

Best Subset Selection

Best subset selection was implemented to determine relevant predictors, see Figure 2.

```
Selection Algorithm: exhaustive
          fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates alcohol
1  ( 1 )  " "           " "              " "         " "            " "       " "                " "                " "     " " " "       "*"
2  ( 1 )  " "           "*"              " "         " "            " "       " "                " "                " "     " " " "       "*"
3  ( 1 )  " "           "*"              " "         " "            " "       " "                " "                " "     " " "*"       "*"
4  ( 1 )  " "           "*"              " "         " "            " "       " "                "*"                " "     " " "*"       "*"
5  ( 1 )  " "           "*"              " "         " "            "*"       " "                "*"                " "     " " "*"       "*"
6  ( 1 )  " "           "*"              " "         " "            "*"       " "                "*"                " "     "*" "*"       "*"
7  ( 1 )  " "           "*"              " "         " "            "*"       "*"                "*"                " "     "*" "*"       "*"
8  ( 1 )  " "           "*"              "*"         " "            "*"       "*"                "*"                " "     "*" "*"       "*"
9  ( 1 )  " "           "*"              "*"         "*"            "*"       "*"                "*"                " "     "*" "*"       "*"
10  ( 1 ) "*"           "*"              "*"         "*"            "*"       "*"                "*"                " "     "*" "*"       "*"
11  ( 1 ) "*"           "*"              "*"         "*"            "*"       "*"                "*"                "*"     "*" "*"       "*"
```

Figure 2. Best Subset Selection. Note that predictors were only added to the model, suggesting these predictors may not be collinear.

The adjusted R^2 values of the models, by number of variables in the subset are shown in Table 3. Pair plots were generated, see Figure 3.

Number of Variables	1	2	3	4	5	6	7	8	9	10	11
Adjusted R^2	0.226	0.316	0.335	0.342	0.349	0.354	0.3566	**0.3567**	0.3565	0.356	0.356

Table 3. Adjusted R^2 values for Best Subset Selection.

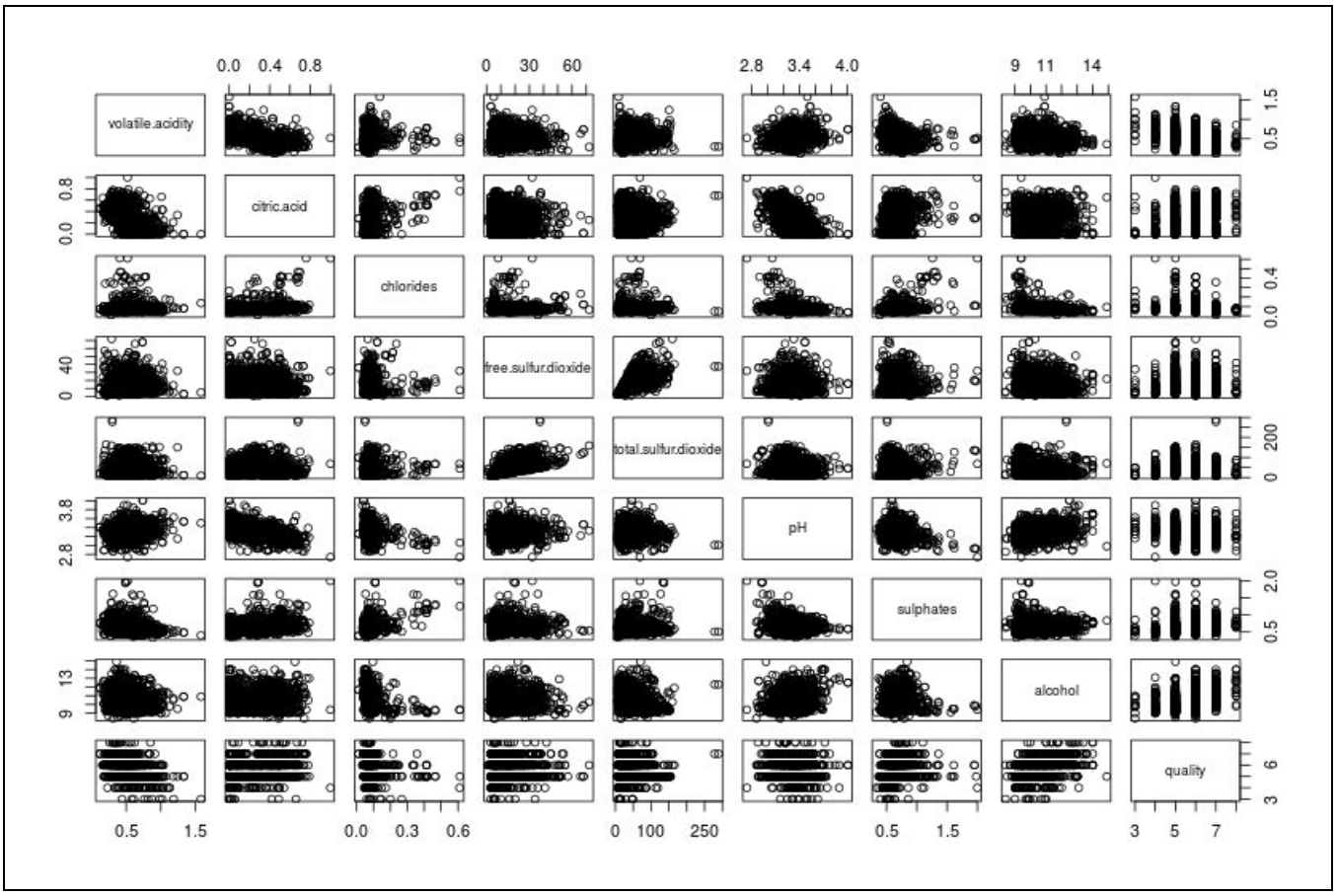

Figure 3. Pair plots of predictors identified by best subset selection and quality.

Linear Regression

The response variable was treated as continuous and a linear model was fit using the best subset of predictors. The diagnostic plots for linear regression with 8 variables is shown in Figure 4.

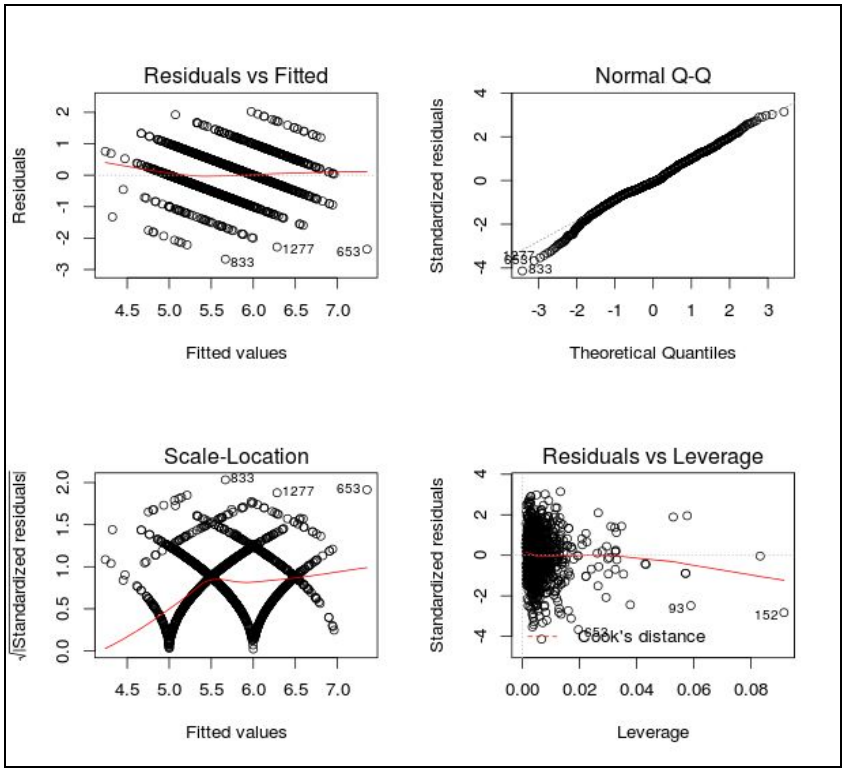

Figure 4. Diagnostic plots of linear regression fit of best subset of predictors. Note the lack of trend in the residuals plot and the nearly normally-distributed residuals.

The normal Q-Q plot of the standardized residuals show that the residuals are close to normal in distribution with slightly heavier tail and the standardized residuals appear close to linear, suggesting the data distribution can be approximated as normal. The model is summarized below.

Call:
lm(formula = quality ~ wine.red$volatile.acidity + wine.red$citric.acid +
wine.red$chlorides + wine.red$free.sulfur.dioxide + wine.red$total.sulfur.dioxide +
wine.red$pH + wine.red$sulphates + wine.red$alcohol, data = wine.red)

Residuals:

Min	1Q	Median	3Q	Max
-2.66890	-0.37044	-0.04474	0.45697	2.02363

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.6680876	0.4608410	10.129	< 2e-16 ***
wine.red$volatile.acidity	-1.0736123	0.1159362	-9.260	< 2e-16 ***
wine.red$citric.acid	-0.1295444	0.1217717	-1.064	0.2876
wine.red$chlorides	-1.9494185	0.4026906	-4.841	1.42e-06 ***
wine.red$free.sulfur.dioxide	0.0047601	0.0021463	2.218	0.0267 *
wine.red$total.sulfur.dioxide	-0.0033658	0.0006954	-4.840	1.42e-06 ***
wine.red$pH	-0.5491501	0.1331350	-4.125	3.90e-05 ***

wine.red$sulphates	0.8914283	0.1102122	8.088	1.19e-15 ***
wine.red$alcohol	0.2928780	0.0171280	17.099	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1590 degrees of freedom
Multiple R-squared: 0.3599, Adjusted R-squared: 0.3567
F-statistic: 111.8 on 8 and 1590 DF, p-value: < 2.2e-16

Note that while Citric Acid was selected by best subset selection, it does not appear significant in this model. For simplicity, this term was left in the model and subsequent models derived from the best subset selection process.

ANOVA was performed on the variables to further analyze our linear regression analysis of Red Wine. Results of this test are shown below.

> anova(lm.red.8var)

Analysis of Variance Table

Response variable: ***quality***

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
volatile.acidity	1	158.97	158.967	378.9116	< 2.2e-16 ***
citric.acid	1	0.17	0.168	0.4011	0.52663
chlorides	1	13.43	13.432	32.0155	1.811e-08 ***
free.sulfur.dioxide	1	2.66	2.656	6.3302	0.01197 *
total.sulfur.dioxide	1	28.69	28.692	68.3909	2.799e-16 ***
pH	1	0.21	0.213	0.5077	0.47625
sulphates	1	48.31	48.308	115.1458	< 2.2e-16 ***
alcohol	1	122.67	122.667	292.3871	< 2.2e-16 ***
Residuals		1590	667.06	0.420	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

As with linear regression, Citric Acid does not appear to be statistically significant. pH has a similarly high p-value, suggesting this term may not significantly contribute to the model. However, both terms were used in subsequent model fits.

The test MSE was estimated for both the linear model fit using all predictors and for the model using the best subset. Similarly, outputs from these models were rounded to the nearest integer and the classification error rate was calculated, see Table 4.

	Linear Regression (all predictors)	Linear Regression (best subset)	Polynomial Regression (best subset, select HO terms)*
Test MSE	0.4150	0.4148	0.3999
Classification Error Rate**	0.4119	0.4119	0.3962

* For each predictor, relevant higher order (HO) terms were identified using F-test ANOVA, see Regression section.
** Linear regression model output values were rounded to integers and approximate classification error rates were calculated.

Table 4. Test MSE and classification error values for regression models.

Following the selection of the best subset and basic linear regression, the following models were fit: multi-class logistic regression with linear predictor terms, multi-class logistic regression with polynomial predictor terms, GAM models composed of thin plate regression splines and linear terms, and two additional classification models.

Polynomial Regression and Multiclass Logistic Regression

Linear, quadratic, and cubic models were fit for each of the predictors as well. F-test ANOVA was used to compare the individual models and statistically significant ($p < 0.05$) quadratic and cubic terms were identified (see Table 5). The polynomial terms identified in this process were used in polynomial regression and multi-class logistic regression models. As with linear regression, test MSE and classification error rate values were calculated for the polynomial regression model, see Table 4.

Polynomial terms were also used in the generation of multiclass logistic regression models; classification error rates were calculated, see Table 7.

Predictor	Linear Term	Quadratic Term	Cubic term
Alcohol	*		**
Volatile Acidity	*	*	
Sulphates	*	*	*
Total Sulfur Dioxide	*		*
Chlorides	*	*	**
pH	*		
Free Sulfur Dioxide	*		
Citric Acid	*		**

* Predictor used in polynomial models.
** Cubic term significant, but quadratic term was not. These terms were not included in polynomial models.

Table 5. Best subset higher-order predictors identified as relevant using F-test ANOVA.

Wilcoxon Test

The one-sample Wilcoxon signed test was performed on the residuals of the linear regression models fitted, in order to ascertain whether we were capturing the true model and/or whether the error terms were distributed symmetrically about the predicated values. The results are shown in the table below.

Wilcoxon Test on Linear Regression Model: **Red Wine**	Wilcoxon Test on Polynomial Regression Model: **Red Wine**
V = 640310, p-value = 0.9695 Alternative hypothesis: true. Location != 0	V = 409190, p-value = 0.9178 Alternative hypothesis: true. Location is != 0

Table 6. Summary of one-sample Wilcoxon test on Regression Models.

GAM Analysis

The predictors identified by best subset were used in the creation of general additive models (GAMs), where a thin plate regression spline was fit for each predictor. The continuous outputs from this model were rounded and approximate error rates were established, see Table 7.

	Multiclass Logistic Regression (All Linear Predictors)	**Multiclass Logistic Regression (Best Subset Linear Predictors)**	**Multiclass Logistic Regression (Polynomial Predictors)**	**Multiclass Logistic Regression (Limited Polynomial Predictors)***	**Best Classification GAM****
Classification Error Rate	0.4025	0.4025	0.4025	0.3742	0.3899
*The three predictors found to contribute the least to the adjusted R^2 were removed to reduce model complexity. **Of the fitted GAM models, value reported for the model with the lowest training error rate.					

Table 7. Test classification error rates for classification models.

Individual predictors were then plotted as a crude check for highly linear terms, see Figure 5. For terms appearing linear, F-test ANOVA was then used to determine if a linear term would produce a superior model compared to regression splines. While several predictors did appear to be modelled better by linear terms, GAMs incorporating linear predictors produced inferior test error rates to the GAM model employing regression splines alone.

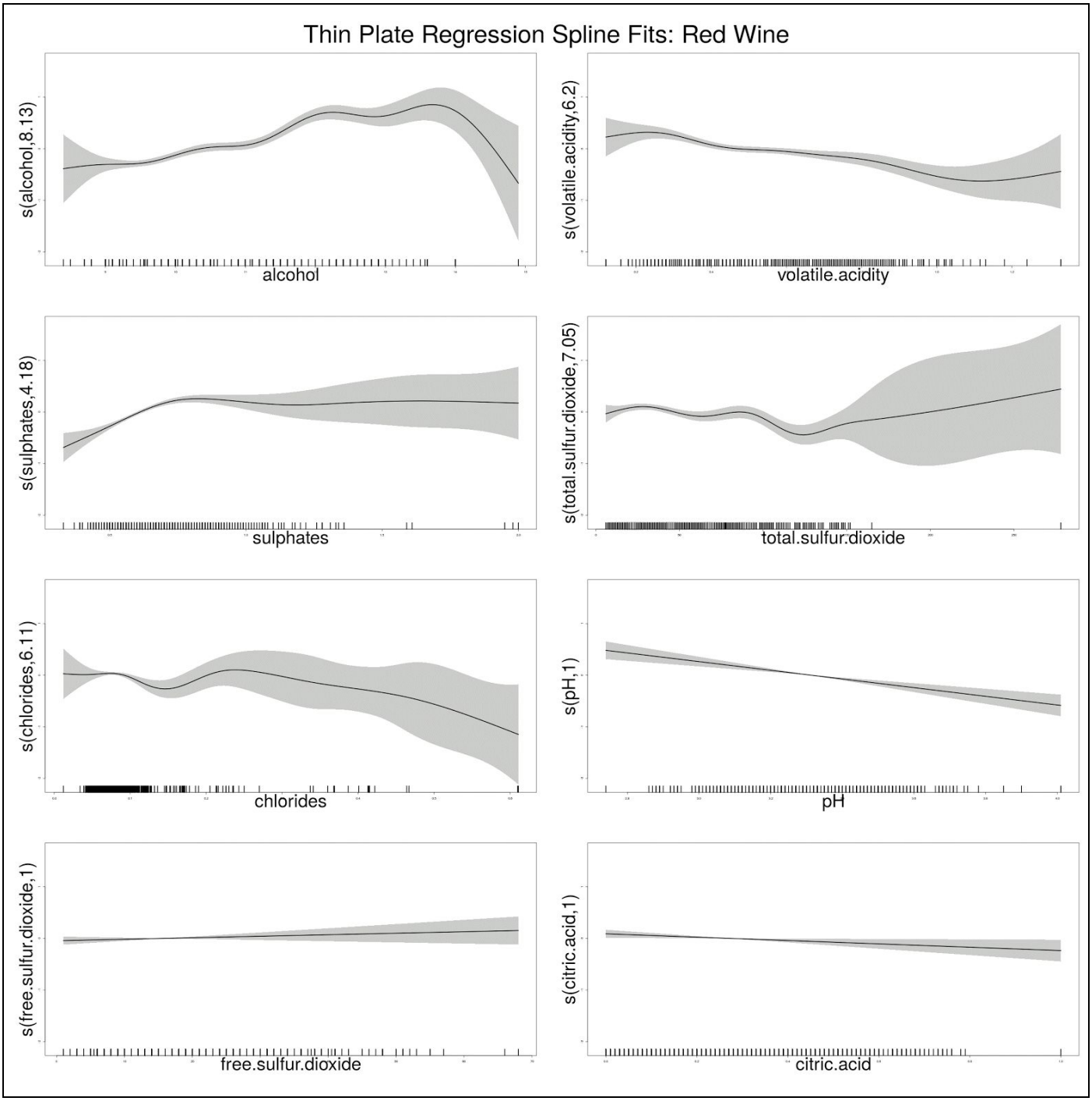

Figure 5. Regression spline fits with 95% confidence intervals. pH, Free Sulfur Dioxide and Citric Acid appear linear while the other predictors have higher degrees of nonlinearity.

Classification

Three additional models were fit by different classification models using the 8 variables found using best subset selection. The classification accuracies are reported below.

Classification Model Type	Random Forest	KNN(K=5)	Multinomial Logistic Regression(Linear)
Classification Accuracy	0.696	0.61	0.60

Table 8. Accuracy rates for classification methods.

As we can see that random forest model performs better, we now want to perform per quality level analysis to verify if any specific quality class is affecting the overall accuracy of the model. Per quality level classification error values are given in the table below(Quality Vs Number of trees(n)).

Quality Level (no. of instances)	Number of trees (n=1000)	Number of trees(n=2000)	Number of trees (n=3000)
3 (4)	0	0	0
4 (9)	0	0	0
5 (118)	0.796	**0.805**	0.805
6 (129)	0.744	**0.744**	0.728
7 (34)	**0.529**	0.470	0.5
8 (6)	0.166	**0.166**	0.166

Table 9. Quality level for random forests with differing numbers of trees.

Due to high variance in the sample size, the accuracy of the model greatly varies per quality level. We can also see that the performance per quality level changes with the number of trees. Hence, we suggest to build a random forest model with n=2000 for optimum classification performance for the red wine dataset.

White Wine Analysis

The white wine dataset was analyzed in a manner similar to the red wine dataset. The data was split into training and test sets (80:20), best subset selection was performed, and regression and classification models were fit to the data.

Best Subset Selection

Best subset selection was performed on all predictors. The model selection process is summarized in Figure 6.

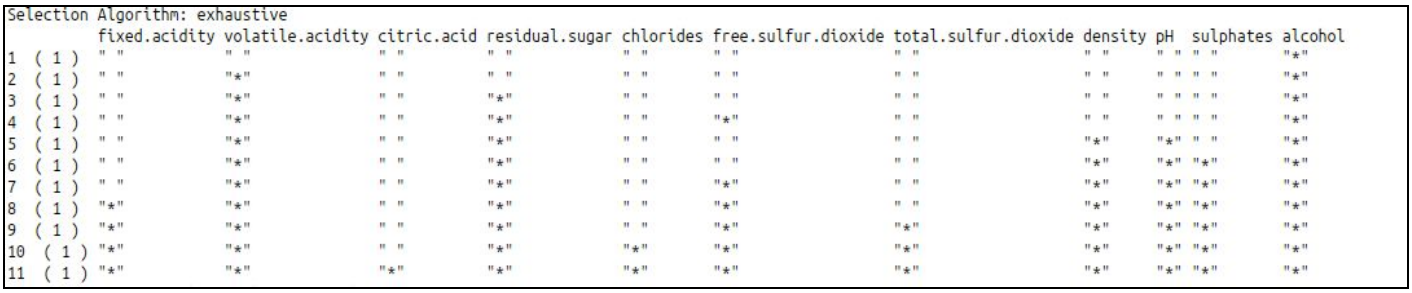

Figure 6. Best Subset Selection. Note that the behavior of Free Sulfur Dioxide, suggesting there may be some collinearity with this term.

The adjusted R^2 values of the models, by number of variables in the subset are shown in Table 10.

Number of Variables	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
adjusted R^2	0.189	0.240	0.258	0.263	0.270	0.275	0.279	**0.28057**	0.28051	0.2803	0.2802

Table 10. Adjusted R^2 values for Best Subset Selection.

As with red wine, an eight predictor model produces the lowest adjusted R^2. The predictors that were chosen for white wine to build the models were *fixed.acidity, volatile.acidity, residual.sugar, free.sulfur.dioxide, density, pH, sulphates, alcohol*. Pair plots for the subset are shown in Figure 7.

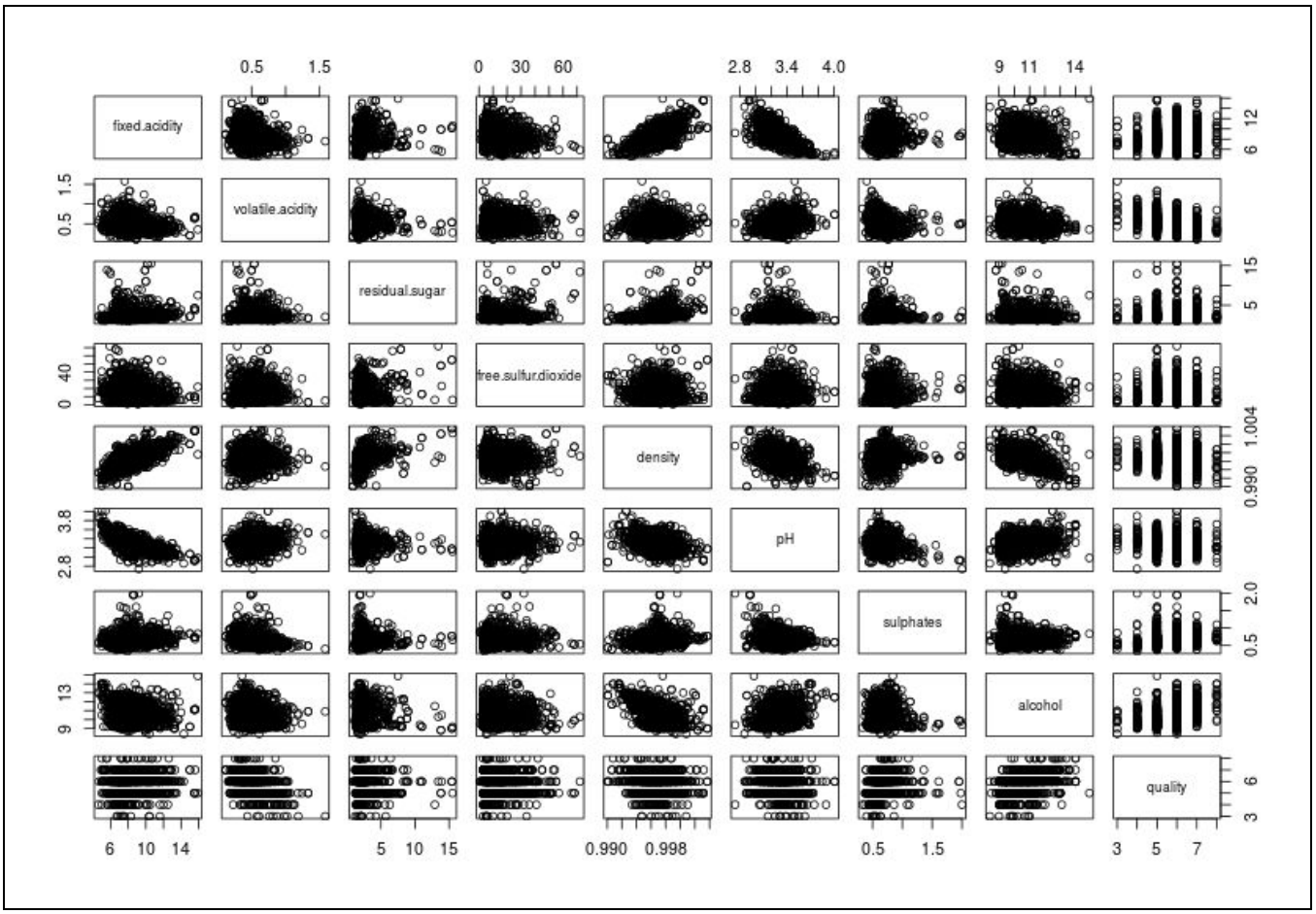

Figure 7. Pair plots of predictors identified by best subset selection and quality.

Linear Regression

We can see from the set of diagnostic plots below, that linear regression on the 8 variables without generalization does not fit the data exactly. The normal Q-Q plot of the standardized residuals show that the residuals are almost normal in distribution with slightly heavier tails. Test MSE and approximated classification error rates are given in Table 10.

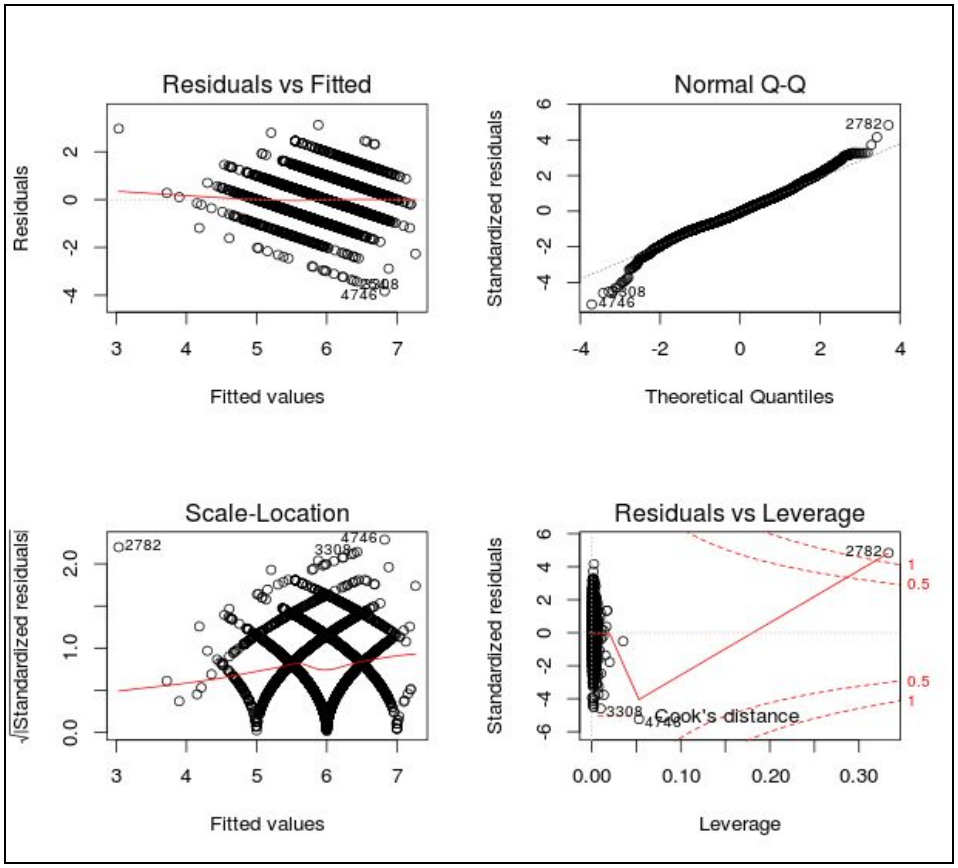

Figure 8. Diagnostic plots of linear regression fit of best subset of predictors. Note while there is a lack of trend in the residuals plot, the standardized residuals appear to not be as normally distributed as those of the red wine dataset.

The summary of the model built using the 8 variables, plus the ANOVA test results, is shown here:

```
> summary(lm.white.8var)

Call:
lm(formula = model, data = wine.white)

Residuals:
     Min       1Q   Median       3Q      Max
-3.8246 -0.4938 -0.0396  0.4660  3.1208

Coefficients:
                      Estimate   Std. Error  t value  Pr(>|t|)
(Intercept)           1.541e+02  1.810e+01    8.514   < 2e-16 ***
fixed.acidity         6.810e-02  2.043e-02    3.333   0.000864 ***
volatile.acidity     -1.888e+00  1.095e-01  -17.242   < 2e-16 ***
residual.sugar        8.285e-02  7.287e-03   11.370   < 2e-16 ***
free.sulfur.dioxide   3.349e-03  6.766e-04    4.950   7.67e-07 ***
density              -1.543e+02  1.834e+01   -8.411   < 2e-16 ***
pH                    6.942e-01  1.034e-01    6.717   2.07e-11 ***
```

sulphates	6.285e-01	9.997e-02	6.287	3.52e-10 ***
alcohol	1.932e-01	2.408e-02	8.021	1.31e-15 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7512 on 4889 degrees of freedom
Multiple R-squared: 0.2818, Adjusted R-squared: 0.2806
F-statistic: 239.7 on 8 and 4889 DF, p-value: < 2.2e-16

Unlike predictors selected by best subset selection for red wine, when fit to a linear model the predictors determined by best subset selection for white wine are all statistically significant in a linear model. These predictors were kept in subsequent models.

Analysis of Variance Table

Response: **quality**

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
fixed.acidity	1	49.62	49.62	87.9392	< 2.2e-16 ***
volatile.acidity	1	149.60	149.60	265.1160	< 2.2e-16 ***
residual.sugar	1	21.51	21.51	38.1169	7.202e-10 ***
free.sulfur.dioxide	1	0.18	0.18	0.3243	0.569
density	1	660.18	660.18	1169.9520	< 2.2e-16 ***
pH	1	122.80	122.80	217.6202	< 2.2e-16 ***
sulphates	1	42.01	42.01	74.4426	< 2.2e-16 ***
alcohol	1	36.30	36.30	64.3296	1.307e-15 ***
Residuals		4889	2758.78	0.56	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Unlike the ANOVA results for the linear regression analysis for red wine, only one variable (free.sulfur.dioxide) has high p-value. The test MSE was estimated for both the linear model fit using all predictors and for the model using the best subset. Similarly, outputs from these models were rounded to the nearest integer and the classification error rate was calculated, see Table 13.

Polynomial Regression and Multiclass Logistic Regression

As with red wine, relevant polynomial terms were identified by fitting linear, quadratic and cubic models for each predictor and then performing F-test ANOVA. Higher order predictor terms deemed statistically significant are summarized in Table 11.

Predictor	Linear Term	Quadratic Term	Cubic term
Alcohol	*	*	*
Volatile Acidity	*		**
Residual Sugar	*		
Density	*	*	*
pH	*		**
Sulphates	*	*	*
Fixed Acidity	*	*	*
Free Sulfur Dioxide	*	*	*

* Predictor used in polynomial models.
** Cubic term significant, but quadratic term was not. These terms were not included in polynomial models.

Table 11. Best subset higher order predictors identified as relevant using F-test ANOVA.

A polynomial model was fit using using these predictors, test MSE, and approximated classification error rates were calculated, see Table 12.

	Linear Regression (all predictors)	Linear Regression (best subset)	Polynomial Regression (best subset, select HO terms)*
Test MSE	0.598	0.596	0.559
Classification Error Rate**	0.506	0.493	0.523

* For each predictor, relevant higher order (HO) terms were identified using F-test ANOVA, see Regression section.
** Linear regression model output values were rounded to integers and approximate classification error rates were calculated.

Table 12. Test MSE and classification error values for regression models.

Similarly, multiclass logistic regression was performed. Classification error rates are summarized in Table 14.

Wilcoxon Test

Similar to our analysis of the red wine data set, for the white wine data set we conducted a one-sample Wilcoxon signed test on the residuals of the linear regression models fitted, in order to ascertain whether we were capturing the true model and/or whether the error terms were distributed symmetrically about the predicated values. The results are displayed in the table below:

Wilcoxon Test on Linear Regression Model - **White Wine**:	Wilcoxon Test on Polynomial Regression Model - **White Wine:**
V = 5869500, p-value = 0.1914 Alternative hypothesis: true. Location != 0	V = 3767900, p-value = 0.1594 Alternative hypothesis: true. Location != 0

Table 13. Summary of one-sample Wilcoxon test on Regression Models.

GAM Analysis

We also performed analysis using a GAM, the results of which are seen, on a predictor-by-predictor basis in Figure 9. Note that unlike GAM fit for red wine, almost all of the terms here have some degree of linearity.

	Multiclass Logistic Regression (All Linear Predictors)	Multiclass Logistic Regression (Best Subset Linear Predictors)	Multiclass Logistic Regression (Polynomial Predictors)	Best Classification GAM*
Classification Error Rate	0.4591	0.4591	0.4839	0.4891
*Of the fitted GAM models, value reported for the model with the lowest training error rate.				

Table 14. Test classification error rates for classification models.

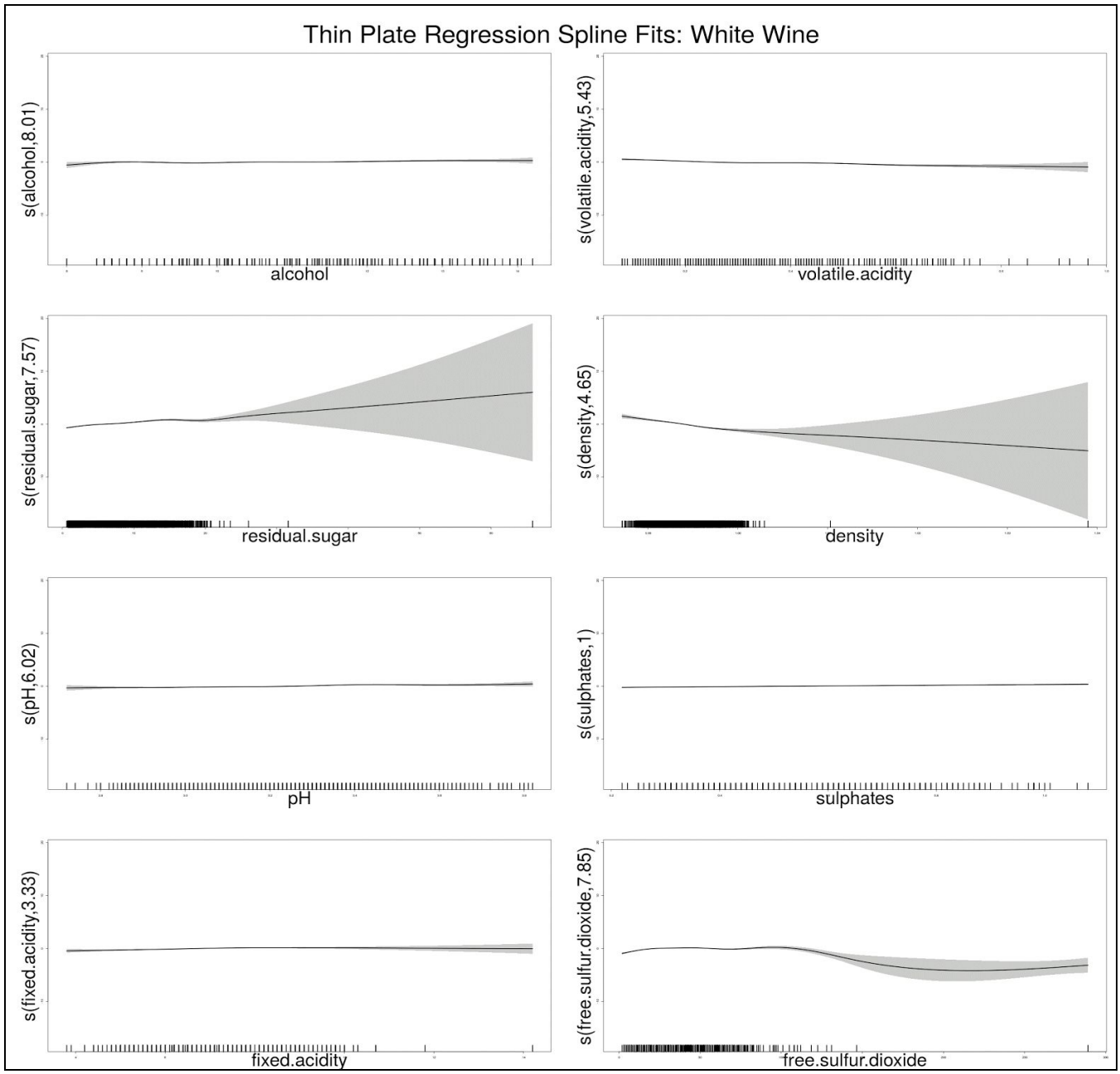

Figure 9. Regression spline fits with 95% confidence intervals. Note the high degree of linearity in all terms.

Classification

As discussed earlier, we fit different classification models using the 8 variables. The classification accuracies are reported in Table 15.

Classification Model Type	Random Forest	KNN(K=5)	Multinomial Logistic Regression(Linear)
Classification Accuracy	0.681	0.522	0.541

Table 15. Classification accuracy rates.

As we can see that random forest model performs better, we now want to perform per quality level analysis to verify if any specific quality class is affecting the overall quality. Per quality level classification accuracy rates are given in the table below (Quality Vs Number of trees(n)).

Quality Level (no. of instances)	Number of trees (n=1000)	Number of trees(n=4000)	Number of trees (n=8000)
3 (5)	0	0	0
4 (30)	0.267	0.267	**0.267**
5 (291)	0.659	**0.663**	0.663
6 (424)	**0.811**	0.801	0.808
7 (190)	0.552	0.557	**0.557**
8 (38)	**0.368**	0.368	0.368
9(1)	0	0	0

Table 16. Quality level for random forests with differing numbers of trees.

Due to different sample sizes for each quality class, the classification accuracies are varying. Also from the table, we can see that the classification accuracy vary with the number of trees and there is no one set configuration that will evaluate best to all the quality levels. However, we suggested building a random forest model with n=1000, for optimum classification performance for white wine dataset.

Conclusions

In this project, we have analysed wine quality dataset which has two classes. We performed regression and classification using different models to predict quality of wine using available predictors. Based on our initial analysis of both the datasets using one way ANOVA and LDA/QDA, we observed that red and white wine datasets were separable. Following which, the two types of wine were evaluated individually. Our investigation revealed that the predictors identified by best subset selection for red wine were much more nonlinear than the corresponding predictors identified by best subset selection for white wine. Also, different sets of predictors identified by best subset selection for the two wine classes also validate our initial direction of performing statistical methods on the two wine classes separately. As such, nonlinear models were slightly more effective in predicting wine quality for red wine than for white. However, for both classes of wine, the random forest technique resulted in models with the lowest test error rate.

References

[1] 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.

Appendix: Supporting R Code

```
library(leaps)
library(caret)
library(splines)
library(nnet) # multinom
library(MNP)
library(ggplot2)

white = as.data.frame(read.csv("winequality-white.csv", sep = ";",
                                  stringsAsFactors=FALSE))
red = as.data.frame(read.csv("winequality-red.csv", sep = ";", stringsAsFactors=FALSE))
red$quality = as.numeric(red$quality)
attach(red)
white$quality = as.numeric(white$quality)

cobmined = rbind(white, red)
summaryStats = data.frame(min = unname(apply(cobmined,2,min)), max =
unname(apply(cobmined,2,max)),
                          range = t(diff(unname(apply(cobmined,2,range)))), mean =
unname(apply(cobmined,2,mean)),
                          med = unname(apply(cobmined,2,median)))

summaryStats2 = data.frame(meanR = unname(apply(red,2,mean)), sdR =
sqrt(unname(apply(red,2,var))),
                          meanR = unname(apply(white,2,mean)), sdR =
sqrt(unname(apply(white,2,var))))
red2 = red
white2 = white

Type = rep("White",dim(white)[1])
whiteLab = cbind(white2, Type)

Type = rep("Red",dim(red)[1])
redLab = cbind(red2, Type)

cobmined2 = rbind(whiteLab, redLab)

bestRed = red[,c(2,3,5,6,7,9,10,11,12)]

pairs(bestRed)

bestWhite = red[,c(1,2,4,6,8,9,10,11,12)]

pairs(bestWhite)

png("Wines.png",width=583,height=486, units = "px")
ggplot(data = cobmined2, aes(color=Type)) + geom_point(aes(x=quality,y=total.sulfur.dioxide,
size = Type, colour=Type)) +
  scale_size_discrete(range = c(7, 4)) +
```

```
  scale_colour_manual(values=c("bisque3", "darkmagenta")) +
  ggtitle("Total Sulfur Dioxide vs Quality") +
  theme(axis.title.x = element_text(size = rel(1.5)), axis.text=element_text(size=15)) +
  theme(axis.title.y = element_text(size = rel(1.5)), axis.text=element_text(size=15)) +
  theme(plot.title=element_text(size=20))+
  labs(x = "Quality",y = "Total Sulfur Dioxide")
graphics.off()

set.seed(1)
train.ix = as.integer(createDataPartition(red$quality, p = .80, list = FALSE, times = 1))
train = rep(FALSE,dim(red)[1])
train[train.ix] = TRUE

#Investigate Red wine:

regfit.full = regsubsets(red$quality ~ ., data = red, subset = train)
reg.summary = summary(regfit.full)
which.max(reg.summary$rsq)

par(mfrow = c(2,2))
plot(regfit.full, scale = "r2")
plot(regfit.full, scale = "adjr2")
plot(regfit.full, scale = "Cp")
plot(regfit.full, scale = "bic")

#Using best subset, it looks like the following eight predictors have give the highest
ajusted R^2 (in order according to R^2):

#* alcohol
#* volatile.acidity
#* sulphates
#* total.sulfur.dioxide
#* chlorides
#* pH
#* free.sulfur.dioxide
#* citric.acid

l.fit = lm(quality ~., data = red, subset = train)
par(mfrow = c(2,2))
plot(l.fit)

#Construct a polynomial lm and then a glm using regression splines. Compare the two.

# alcohol
fit.1 = lm(quality ~ alcohol, data = red, subset = train)
fit.2 = lm(quality ~ poly(alcohol, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(alcohol, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# Linear and cubic terms are significant, since quadratic is not only use linear
```

```
# volatile.acidity
fit.1 = lm(quality ~ volatile.acidity , data = red, subset = train)
fit.2 = lm(quality ~ poly(volatile.acidity, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(volatile.acidity, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# Linear and quadratic terms are significant

# sulphates
fit.1 = lm(quality ~ sulphates , data = red, subset = train)
fit.2 = lm(quality ~ poly(sulphates, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(sulphates, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms are significant

# total.sulfur.dioxide
fit.1 = lm(quality ~ total.sulfur.dioxide , data = red, subset = train)
fit.2 = lm(quality ~ poly(total.sulfur.dioxide, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(total.sulfur.dioxide, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# Linear and cubic terms are significant, since quadratic is not only use linear

# chlorides
fit.1 = lm(quality ~ chlorides , data = red, subset = train)
fit.2 = lm(quality ~ poly(chlorides, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(chlorides, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms are significant

# pH
fit.1 = lm(quality ~ pH , data = red, subset = train)
fit.2 = lm(quality ~ poly(pH, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(pH, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# Only linear is significant

# free.sulfur.dioxide
fit.1 = lm(quality ~ free.sulfur.dioxide , data = red, subset = train)
fit.2 = lm(quality ~ poly(free.sulfur.dioxide, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(free.sulfur.dioxide, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# Only linear is significant

# citric.acid
fit.1 = lm(quality ~ citric.acid , data = red, subset = train)
fit.2 = lm(quality ~ poly(citric.acid, 2), data = red, subset = train)
fit.3 = lm(quality ~ poly(citric.acid, 3), data = red, subset = train)
anova(fit.1,fit.2,fit.3)
# Linear and cubic are significant, only use linear

# Fit and test lm
```

```
best.lm = lm(quality ~ alcohol + volatile.acidity + sulphates +
             total.sulfur.dioxide + chlorides + pH + free.sulfur.dioxide + citric.acid,
             data = red, subset = train)
poly.fit = lm(quality ~ alcohol + poly(volatile.acidity ,2) + poly(sulphates ,3) +
             total.sulfur.dioxide + poly(chlorides ,3) + pH + free.sulfur.dioxide +
citric.acid,
             data = red, subset = train)

wilcox.test(unname(poly.fit$residuals), y = NULL,alternative = "two.sided",mu = 0, paired =
FALSE, exact = NULL,
             correct = TRUE,conf.int = FALSE, conf.level = 0.95)

# Convert Linear, linear with best subset and polynomial to factors and then score
classificaton error rate
predictVals = round(unname(predict(l.fit, red))[!train], digits = 0)

par(mfrow = c(2,2))
plot(poly.fit )

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #
1- count/length(predictVals) #0.4119497

predictVals = round(unname(predict(best.lm, red))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #
1- count/length(predictVals) #0.4119497

predictVals = round(unname(predict(poly.fit, red))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
```

```
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #
1- count/length(predictVals) #0.3962264

par(mfrow = c(2,2))
plot(poly.fit)

estTestMSE_linear_best = mean(((quality - unname(predict(best.lm, red)))[!train])^2)
estTestMSE_linear = mean(((quality - unname(predict(l.fit, red)))[!train])^2)
estTestMSE_poly = mean(((quality - unname(predict(poly.fit, red)))[!train])^2)

# The fit with logistic
red$quality <- as.factor(red$quality)
multnom.model = multinom(quality ~ ., data = red,subset = train)

#Testing the prediction accuracy
predictVals <- predict(multnom.model, newdata=red)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.5974843
1-count/length(predictVals) #test error rate 0.4025157

# Fit with best subset
multnom.model = multinom(quality ~ alcohol + volatile.acidity + sulphates +
                           total.sulfur.dioxide +chlorides + pH + free.sulfur.dioxide +
citric.acid
                           , data = red,subset = train)

#Testing the prediction accuracy
predictVals <- predict(multnom.model, newdata=red)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
```

```
       count=count+1
  }
}
count/length(predictVals) #0.5974843, slightly worse
1-count/length(predictVals) #test error rate 0.4025157

# Fit with polynomial logistic
multnom.model <- multinom(quality ~ alcohol + poly(volatile.acidity ,2) + poly(sulphates ,3)
+
                          total.sulfur.dioxide + poly(chlorides ,3) + pH +
free.sulfur.dioxide + citric.acid,
                          data = red, subset = train)
predictVals <- predict(multnom.model, newdata=red)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.5974843
1-count/length(predictVals) #0.4025157

# Fit with polynomial logistic, removing final three linear terms of best subset:
# pH + free.sulfur.dioxide + citric.acid
multnom.model <- multinom(quality ~ alcohol + poly(volatile.acidity ,2) + poly(sulphates ,3)
+
                          total.sulfur.dioxide + poly(chlorides ,3),
                          data = red, subset = train)
predictVals <- predict(multnom.model, newdata=red)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.6257862
1-count/length(predictVals)

# GAM
detach("package:nnet",unload = TRUE) #Unload nnet
library(mgcv) #both multinom #search
```

```
red$quality = as.numeric(red$quality) #gam with thin plane splines
gam.ft = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp")
             + s(sulphates,bs="tp") + s(total.sulfur.dioxide,bs="tp") +
             s(chlorides,bs="tp")+ s(pH,bs="tp")+ s(free.sulfur.dioxide,bs="tp")+
             s(citric.acid,bs="tp"), data = red[train,])

estTestMSE_gam = mean(((red$quality - unname(predict(gam.ft, red)))[!train])^2)

#gam.check(gam.ft, k.rep = 1000)
png("GAM_w_splines_red",width=583*4.5,height=486*5.5, units = "px")  #preserve w*h 6*5 ratio
par(mfrow = c(4,2), mar=c(10,10,2,2)+0.5, oma = c(0, 0, 10, 0))
plot.gam(gam.ft, shade =TRUE, lwd=2.5, cex.axis=1, cex.lab = 5.5)
mtext("Thin Plate Regression Spline Fits: Red Wine", outer = TRUE, cex = 5)
dev.off()


#try doubling k' to 18 for 1, 2 and 4
gam.ft2 = gam(quality ~ s(alcohol,bs="tp",k=18) + s(volatile.acidity,bs="tp",k=18) +
             s(sulphates,bs="tp") + s(total.sulfur.dioxide,bs="tp", k= 18) +
             s(chlorides,bs="tp"), data = red[train,]) #slightly worse

estTestMSE_gam = mean(((quality - unname(predict(gam.ft, red)))[!train])^2)
#slightly worse


#Calculate
predictVals = round(unname(predict(gam.ft, red))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.6006289
1- count/length(predictVals) #0.3993711

par(mfrow = c(2 ,3))
plot.gam(gam.ft,se=TRUE)

# volatile.acidity looks linear, try gam with linear fit/cubic spline
gam.ft2 = gam(quality ~ s(alcohol,bs="cr") + volatile.acidity +
             s(sulphates,bs="cr") +
             s(total.sulfur.dioxide,bs="cr") + s(chlorides,bs="cr"), data = red[train,])
estTestMSE_gam2 = mean(((red$quality - unname(predict(gam.ft2, red)))[!train])^2)
anova(gam.ft, gam.ft2, test = 'F') # just barely below 5%
```

```
predictVals = round(unname(predict(gam.ft2, red))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == red[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.5974843 but a worse model

detach(red)
detach("package:mgcv,unload" = TRUE) #Unload mgcv
library(nnet) #load nnet

attach(white)

par(mfrow = c(2,2))
plot(regfit.full, scale = "r2")
plot(regfit.full, scale = "adjr2")
plot(regfit.full, scale = "Cp")
plot(regfit.full, scale = "bic")

#Using best subset:
#* alcohol
#* volatile.acidity
#* residual.sugar
#* density
#* pH
#* sulphates
#* fixed.acidity
#* free.sulfur.dioxide

l.fit = lm(quality ~., data = white, subset = train)
par(mfrow = c(2,2))
plot(l.fit)

#Construct a polynomial lm and then a glm using regression splines. Compare the two.

# alcohol
fit.1 = lm(quality ~ alcohol, data = white, subset = train)
fit.2 = lm(quality ~ poly(alcohol, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(alcohol, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms are significant

# volatile.acidity
fit.1 = lm(quality ~ volatile.acidity, data = white, subset = train)
```

```
fit.2 = lm(quality ~ poly(volatile.acidity, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(volatile.acidity, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# Linear and cubic significant, only use linear

# residual.sugar
fit.1 = lm(quality ~ residual.sugar, data = white, subset = train)
fit.2 = lm(quality ~ poly(residual.sugar, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(residual.sugar, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# Only Linear

# density
fit.1 = lm(quality ~ density, data = white, subset = train)
fit.2 = lm(quality ~ poly(density, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(density, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms

# pH
fit.1 = lm(quality ~ pH, data = white, subset = train)
fit.2 = lm(quality ~ poly(pH, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(pH, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# Linear and cubic significant, only use linear

# sulphates
fit.1 = lm(quality ~ sulphates, data = white, subset = train)
fit.2 = lm(quality ~ poly(sulphates, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(sulphates, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms

# fixed.acidity
fit.1 = lm(quality ~ fixed.acidity, data = white, subset = train)
fit.2 = lm(quality ~ poly(fixed.acidity, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(fixed.acidity, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms

# free.sulfur.dioxide
fit.1 = lm(quality ~ free.sulfur.dioxide, data = white, subset = train)
fit.2 = lm(quality ~ poly(free.sulfur.dioxide, 2), data = white, subset = train)
fit.3 = lm(quality ~ poly(free.sulfur.dioxide, 3), data = white, subset = train)
anova(fit.1,fit.2,fit.3)
# All terms

# Fit and test lm
best.linear = lm(quality ~ alcohol + volatile.acidity + residual.sugar+
             density+pH+sulphates + fixed.acidity +
```

```
              free.sulfur.dioxide, data=white, subset=train)

poly.fit = lm(quality ~ poly(alcohol,3) + volatile.acidity + residual.sugar+
              poly(density,3)+pH+poly(sulphates,3) + poly(fixed.acidity, 3) +
              poly(free.sulfur.dioxide, 3), data=white, subset=train)

wilcox.test(unname(poly.fit$residuals), y = NULL,alternative = "two.sided",mu = 0, paired =
FALSE, exact = NULL,
              correct = TRUE,conf.int = FALSE, conf.level = 0.95)

estTestMSE_linear = mean(((quality - unname(predict(l.fit, white)))[!train])^2)
estTestMSE_linear_best =  mean(((quality - unname(predict(best.linear, white)))[!train])^2)
estTestMSE_poly = mean(((quality - unname(predict(poly.fit, white)))[!train])^2)


# Convert Linear, linear with best subset and polynomial to factors and then score
classificaton error rate
predictVals = round(unname(predict(l.fit, white))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #
1- count/length(predictVals) #0.5056995

predictVals = round(unname(predict(best.lm, white))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #
1- count/length(predictVals) #0.4932642

predictVals = round(unname(predict(poly.fit, white))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
```

```
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #
1- count/length(predictVals) #0.5233161

par(mfrow = c(2,2))
plot(poly.fit)

# The fit with logistic
white$quality <- as.factor(white$quality)
multnom.model = multinom(quality ~ ., data = white,subset = train)

#Testing the prediction accuracy
predictVals <- predict(multnom.model, newdata=white)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.5450777
1-count/length(predictVals)

# Fit with best subset
multnom.model = multinom(quality ~ alcohol + volatile.acidity + residual.sugar+
                           density + pH + sulphates + fixed.acidity +
                           free.sulfur.dioxide, data=white, subset=train)

#Testing the prediction accuracy
predictVals <- predict(multnom.model, newdata=white)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) # 0.5409326, slightly better
1-count/length(predictVals)
```

```
multnom.model = multinom(quality ~ poly(alcohol,3) + volatile.acidity + residual.sugar+
                           poly(density,3)+pH+poly(sulphates,3) + poly(fixed.acidity, 3) +
                           poly(free.sulfur.dioxide, 3), data=white, subset=train)

#Testing the prediction accuracy
predictVals <- predict(multnom.model, newdata=white)[!train]

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) # 0.5160622 slightly worse
1-count/length(predictVals)

detach("package:nnet",unload = TRUE) #Unload nnet
library(mgcv) #both multinom
white$quality = as.numeric(white$quality) #gam with thin plane splines
gam.ft = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              s(residual.sugar,bs="tp") + s(density,bs="tp") + s(pH,bs="tp") +
              s(sulphates,bs="tp") + s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])

estTestMSE_gam = mean(((quality - unname(predict(gam.ft, white)))[!train])^2)
par(mfrow = c(3 ,3))
plot.gam(gam.ft,se=TRUE)

png("GAM_w_splines_white",width=583*4.5,height=486*5.5, units = "px")  #preserve w*h 6*5
ratio
par(mfrow = c(4,2), mar=c(10,10,2,2)+0.5, oma = c(0, 0, 10, 0))
plot.gam(gam.ft, shade =TRUE, lwd=2.5, cex.axis=1, cex.lab = 5.5)
mtext("Thin Plate Regression Spline Fits: White Wine", outer = TRUE, cex = 5)
dev.off()



#Calculate
predictVals = round(unname(predict(gam.ft, white))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
```

```
      count=count+1
  }
}
count/length(predictVals) #0.5139896 again, slightly worse
1-count/length(predictVals)
# Looks like all terms are at least somewhat linear,
# try a modified gam for each
gam.ft2 = gam(quality ~ alcohol + s(volatile.acidity,bs="tp") + s(residual.sugar,bs="tp")+
              s(density,bs="tp") + s(pH,bs="tp") + s(sulphates,bs="tp")
+s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])
anova(gam.ft, gam.ft2, test = "F") # Model not improved

gam.ft3 = gam(quality ~ s(alcohol,bs="tp") + volatile.acidity + s(residual.sugar,bs="tp")+
              s(density,bs="tp") + s(pH,bs="tp") + s(sulphates,bs="tp") +
s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])
anova(gam.ft, gam.ft3, test = "F") # linear is better

gam.ft4 = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              residual.sugar +s(density,bs="tp") + s(pH, bs = "tp")+ s(sulphates,bs="tp") +
s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"),
              data = white[train,])
anova(gam.ft, gam.ft4, test = "F")   # linear is better

gam.ft5 = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              s(residual.sugar,bs="tp") + density + s(pH,bs="tp") +
              s(sulphates,bs = "tp") + s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])
anova(gam.ft, gam.ft5, test = "F")  # linear is better

gam.ft6 = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              s(residual.sugar,bs="tp") + s(density,bs = "tp") + pH +
              s(sulphates,bs = "tp") + s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])
anova(gam.ft, gam.ft6, test = "F") # Model not improved

gam.ft7 = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              s(residual.sugar,bs="tp") + s(density, bs = "tp") + s(pH,bs="tp") +
              sulphates + s(fixed.acidity,bs="tp")+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])
anova(gam.ft, gam.ft7, test = "F")  # linear is better

gam.ft8 = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              s(residual.sugar,bs="tp") + s(density, bs = "tp") + s(pH,bs="tp") +
              s(sulphates,bs = "tp") + fixed.acidity+
              s(free.sulfur.dioxide,bs="tp"), data = white[train,])
anova(gam.ft, gam.ft8, test = "F")   # linear is better
```

```
gam.ft9 = gam(quality ~ s(alcohol,bs="tp") + s(volatile.acidity,bs="tp") +
              s(residual.sugar,bs="tp") + s(density, bs = "tp") + s(pH,bs="tp") +
              s(sulphates,bs = "tp") + s(fixed.acidity,bs="tp")+
              free.sulfur.dioxide, data = white[train,])
anova(gam.ft, gam.ft9, test = "F")  # linear is better

gam.ft10 = gam(quality ~ s(alcohol,bs="tp") + volatile.acidity +
              residual.sugar + density + s(pH,bs="tp") +
              sulphates + fixed.acidity+
              free.sulfur.dioxide, data = white[train,])

#Calculate
predictVals = round(unname(predict(gam.ft10, white))[!train], digits = 0)

i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == white[!train,]$quality[i])
  {
       count=count+1
  }
}
count/length(predictVals) #0.5108808
1-count/length(predictVals)
```

Code for linear and classification analysis of red wine data:

```
library(leaps)
library(nnet)
library(randomForest)
library(class)
library(MASS)
library(stats)
library(gbm)

wine.red <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-red.csv", header=TRUE,
sep=";")
wine.white <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-white.csv",
header=TRUE, sep=";")

wine.red.total <- wine.red
wine.red.total$class <- 0
wine.white.total <- wine.white
wine.white.total$class <- 1

wine.total <- rbind(wine.red, wine.white)
wine.total.classificiation <- rbind(wine.red.total, wine.white.total)
```

```
#Linear Regressions
lm.basic.fit.red <- lm(quality ~ . , data=wine.red)
summary(lm.basic.fit.red)
lm.basic.fit.white <- lm(quality ~ . , data=wine.white)
summary(lm.basic.fit.white)
lm.basic.fit <- lm(quality ~ ., data=wine.total)
summary(lm.basic.fit)

#ANOVA
anova(lm.basic.fit.red)
anova(lm.basic.fit.white)
anova(lm.basic.fit)


#Subset Variable Selection
regsubsets.red.out.exhaustive <- regsubsets(quality ~ ., data=wine.red, nbest=1, nvmax=NULL,
method="exhaustive")
summary(regsubsets.red.out.exhaustive)
a <- summary(regsubsets.red.out.exhaustive)
a$adjr2

regsubsets.red.out.forward <- regsubsets(quality ~ ., data=wine.red, nbest=1, nvmax=NULL,
method="forward")
summary(regsubsets.red.out.forward)
a <- summary(regsubsets.red.out.forward)
a$adjr2


regsubsets.red.out.backward <- regsubsets(quality ~ ., data=wine.red, nbest=1, nvmax=NULL,
method="backward")
summary(regsubsets.red.out.backward)
a <- summary(regsubsets.red.out.backward)
a$adjr2


regsubsets.red.out.seqrep <- regsubsets(quality ~ ., data=wine.red, nbest=1, nvmax=NULL,
method="seqrep")
summary(regsubsets.red.out.seqrep)
a <- summary(regsubsets.red.out.seqrep)
a$adjr2

modelStr = quality ~
volatile.acidity+citric.acid+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+pH+sulphates+
alcohol
lm.red.8var <- lm(modelStr, data=wine.red)
summary(lm.red.8var)
par(mfrow = c(1,1))
sresid <- studres(lm.red.8var)
hist(sresid, freq=FALSE, main="Distribution of Studentized Residuals")
xfit<-seq(min(sresid),max(sresid),length=40)
```

```
yfit<-dnorm(xfit)
lines(xfit, yfit)
ncvTest(lm.red.8var)

anova(lm.red.8var)
par(mfrow = c(1,1))
plot(lm.red.8var)
plot(lm.red.8var, scale("r2"))
plot(lm.red.8var, scale("adjr2"))
plot(lm.red.8var, scale = "Cp")
plot(lm.red.8var, scale = "bic")

plot(train.red$total.sulfur.dioxide, train.red$free.sulfur.dioxide)
plot(train.red)


#Calculating the conditioning number of the correlation matrix
wine.red.noReponse <- wine.red[-12]
wine.red.noresp.maxtrix <- as.matrix(wine.red.noReponse)
transposeWine.red <- t(wine.red.noresp.maxtrix)%*%wine.red.noresp.maxtrix
rcond(transposeWine.red)


wine.red.8var.noResp <- wine.red[-c(1,3,8,12)]
wine.red.8var.noReponse <- wine.red[-12]
wine.red.8var.noresp.matrix <- as.matrix(wine.red.8var.noResp)
transposeWine.8var.red <- t(wine.red.8var.noresp.matrix)%*%wine.red.8var.noresp.matrix
rcond(transposeWine.8var.red)


#after center and scaling the data
red.scale <- scale(wine.red, center = TRUE, scale = TRUE)
red.scale.noResp <- red.scale[,-12]
red.scale.noResp.matrix <- as.matrix(red.scale.noResp)
transpose.red.scale <- t(red.scale.noResp.matrix)%*% red.scale.noResp.matrix
rcond(transpose.red.scale)

regsubsets.red.scale.exhaustive <- regsubsets(quality ~ ., data=red.scale.df, nbest=1,
nvmax=NULL, method="exhaustive")
summary(regsubsets.red.scale.exhaustive)
a <- summary(regsubsets.red.scale.exhaustive)
a$adjr2


#multinomial logistic regression
set.seed(5)
wine.red <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-red.csv", header=TRUE,
sep=";")
x <- sample(c(1:1599), 300, replace=FALSE)
train.red <- wine.red[-x,]
```

```
test.red <- wine.red[x,]

#train.red$quality <- as.factor(train.red$quality)
#test.red$quality <- as.factor(test.red$quality)
model = as.factor(quality) ~
volatile.acidity+citric.acid+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+pH+sulphates+
alcohol
multinom.model <- multinom(model, data=train.red)
#z <- summary(multnom.model)$coefficients/summary(multnom.model)$standard.errors
#p <- (1 - pnorm(abs(z), 0, 1)) * 2

#TRAIN MSE
predictVals <- predict(multinom.model, newdata=train.red)
predictVals <- as.character(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == train.red$quality[i])
  {
       count=count+1
  }
}
print(c("count",count))
trainMSE = count/length(train.red$quality)
print(c("trainMSE",trainMSE))
#[1] "trainMSE"            "0.614615384615385"

#TEST MSE

predictVals <- predict(multinom.model, newdata=test.red)
predictVals <- as.character(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == test.red$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"test.length",length(test.red$quality)))
testMSE = count/length(test.red$quality)
print(c("testMSE",testMSE))
#[1] "testMSE"             "0.595317725752508"

#knn
k=5
x <- sample(c(1:1599), 300, replace=FALSE)
train.red <- wine.red[-x,]
```

```
test.red <- wine.red[x,]

predictVals <- knn(train.red, train.red, train.red$quality, k)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == train.red$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"train.length",length(train.red$quality)))
trainMSE = count/length(train.red$quality)
print(c("trainMSE",trainMSE))
#[1] "trainMSE"            "0.727692307692308"


predictVals <- knn(train.red, test.red, train.red$quality, k)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == test.red$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"test.length",length(test.red$quality)))
testMSE = count/length(test.red$quality)
print(c("testMSE",testMSE))
plot(predictVals, test.red$quality, xlab="predicted values", ylab="actual values")
#[1] "testMSE"             "0.588628762541806"

#Random Forest
set.seed(5)
wine.red <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-red.csv", header=TRUE,
sep=";")
x <- sample(c(1:1599), 300, replace=FALSE)
train.red <- wine.red[-x,]
test.red <- wine.red[x,]
fit <- randomForest(as.factor(quality) ~
volatile.acidity+citric.acid+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+pH+sulphates+
alcohol, data=train.red, importance=TRUE, ntree=3000)
predictVals <- predict(fit, train.red)
predictVals <- as.character(predictVals)
i =0
count <- rep(0,10)
total <- rep(0,10)
for (i in 1:length(predictVals))
```

```
{
  total[train.red$quality[i]] = total[train.red$quality[i]]+1
  if (as.integer(predictVals[i]) == as.integer(train.red$quality[i]))
  {
#      print(c(predictVals[i], train.red$quality[i]))
#      print("Yes")
       count[train.red$quality[i]] = count[train.red$quality[i]]+1
  }
}
for (i in 1:10)
{
  print (c(count[i],total[i]))
  print(c("Quality", i," Value", as.double(count[i])/total[i]))
}
#[1] "trainMSE"            "0.727692307692308"

predictVals <- predict(fit, test.red)
predictVals <- as.character(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == test.red$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"test.length",length(test.red$quality)))
testMSE = count/length(test.red$quality)
print(c("testMSE",testMSE))
#[1] "testMSE"             "0.709030100334448"

predictVals <- predict(fit, test.red)
predictVals <- as.character(predictVals)
count <- rep(0,10)
total <- rep(0,10)
for (i in 1:length(predictVals))
{
  total[test.red$quality[i]] = total[test.red$quality[i]]+1
  if (as.integer(predictVals[i]) == as.integer(test.red$quality[i]))
  {
       #      print(c(predictVals[i], train.red$quality[i]))
       #      print("Yes")
       count[test.red$quality[i]] = count[test.red$quality[i]]+1
  }
}
for (i in 1:10)
{
  print (c(count[i],total[i]))
  print(c("Quality", i," Value", as.double(count[i])/total[i]))
```

```
}
#[1] "testMSE"             "0.709030100334448"

a = c(wine.red$quality,wine.white$quality)
b = rep(1,length(wine.red$quality))
temp = rep(0,length(wine.white$quality))
d = c(b,temp)
data2 = data.frame(a,d)

lm.basic.fit.red <- lm(a ~ d , data2)
summary(lm.basic.fit.red)

#Linear Regressions
lm.basic.fit.red <- lm(quality ~ . , data=wine.red)
x = resid(lm.basic.fit.red)

wilcox.test(x, y = NULL,alternative = "two.sided",mu = 0, paired = FALSE, exact = NULL,
correct = TRUE,conf.int = FALSE, conf.level = 0.95)

#Regression And Classification

modelStr = quality ~
volatile.acidity+citric.acid+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+pH+sulphates+
alcohol
lm.red.8var.model <- lm(modelStr, data=train.red)
summary(lm.red.8var.model)

predictVals <- predict(lm.red.8var.model, test.red)
typeof(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (round(predictVals[i]) == test.red$quality[i])
  {
       count= count+1
  }
}
print(c("count",count,"test.length",length(test.red$quality)))
testF = count/length(test.red$quality)
print(c("testF",testF))
```

Code for linear and classification analysis of white wine data:

```
library(leaps)
library(nnet)
library(randomForest)
library(class)
wine.white <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-white.csv",
header=TRUE, sep=";")
```

```
wine.white.total <- wine.white
wine.white.total$class <- 1


#White wine Analysis
#Subset Variable Selection
regsubsets.white.out.exhaustive <- regsubsets(quality ~ ., data=wine.white, nbest=1,
nvmax=NULL, method="exhaustive")
summary(regsubsets.white.out.exhaustive)
a <- summary(regsubsets.white.out.exhaustive)
a$adjr2

regsubsets.white.out.forward <- regsubsets(quality ~ ., data=wine.white, nbest=1, nvmax=NULL,
method="forward")
summary(regsubsets.white.out.forward)
a <- summary(regsubsets.white.out.forward)
a$adjr2


regsubsets.white.out.backward <- regsubsets(quality ~ ., data=wine.white, nbest=1,
nvmax=NULL, method="backward")
summary(regsubsets.white.out.backward)
a <- summary(regsubsets.white.out.backward)
a$adjr2


regsubsets.white.out.seqrep <- regsubsets(quality ~ ., data=wine.white, nbest=1, nvmax=NULL,
method="seqrep")
summary(regsubsets.white.out.seqrep)
a <- summary(regsubsets.white.out.seqrep)
a$adjr2

model = quality ~
fixed.acidity+volatile.acidity+residual.sugar+free.sulfur.dioxide+density+pH+sulphates+alcoho
l
lm.white.8var <- lm(model, data=wine.white)
summary(lm.white.8var)
anova(lm.white.8var)
par(mfrow = c(2,2))
plot(lm.white.8var)
plot(lm.white.8var, scale("r2"))
plot(lm.white.8var, scale("adjr2"))
plot(lm.white.8var, scale = "Cp")
plot(lm.white.8var, scale = "bic")



#multinomial logistic regression
set.seed(5)
```

```
wine.white <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-white.csv",
header=TRUE, sep=";")
x <- sample(c(1:length(wine.white$quality)), (1/5)*length(wine.white$quality), replace=FALSE)
train.white <- wine.white[-x,]
test.white <- wine.white[x,]

model = as.factor(quality) ~
fixed.acidity+volatile.acidity+residual.sugar+free.sulfur.dioxide+density+pH+sulphates+alcoho
l
multinom.model <- multinom(model, data=train.white)
#z <- summary(multnom.model)$coefficients/summary(multnom.model)$standard.errors
#p <- (1 - pnorm(abs(z), 0, 1)) * 2

#TRAIN MSE
predictVals <- predict(multinom.model, newdata=train.white)
predictVals <- as.character(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == train.white$quality[i])
  {
       count=count+1
  }
}
print(c("count",count))
trainMSE = count/length(train.white$quality)
print(c("train",trainMSE))

predictVals <- predict(multinom.model, newdata=test.white)
predictVals <- as.character(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == test.white$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"test.length",length(test.white$quality)))
testMSE = count/length(test.white$quality)
print(c("test",testMSE))

#knn
k=5
x <- sample(c(1:length(wine.white$quality)), (1/5)*length(wine.white$quality), replace=FALSE)
train.white <- wine.white[-x,]
test.white <- wine.white[x,]
```

```
predictVals <- knn(train.white, train.white, train.white$quality, k)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == train.white$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"train.length",length(train.white$quality)))
trainMSE = count/length(train.white$quality)
print(c("train",trainMSE))

predictVals <- knn(train.white, test.white, train.white$quality, k)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == test.white$quality[i])
  {
       count=count+1
  }
}
print(c("count",count,"test.length",length(test.white$quality)))
testMSE = count/length(test.white$quality)
print(c("test",testMSE))

#Random Forest
set.seed(5)
wine.white <- read.csv("/home/rohitb/Desktop/StatInfProject/winequality-white.csv",
header=TRUE, sep=";")
x <- sample(c(1:length(wine.white$quality)), (1/5)*length(wine.white$quality), replace=FALSE)
train.red <- wine.white[-x,]
test.red <- wine.white[x,]
fit <- randomForest(as.factor(quality) ~
fixed.acidity+volatile.acidity+residual.sugar+free.sulfur.dioxide+density+pH+sulphates+alcoho
l, data=train.white, importance=TRUE, ntree=1000)
predictVals <- predict(fit, train.white)
predictVals <- as.character(predictVals)
i =0
count <- rep(0,10)
total <- rep(0,10)
for (i in 1:length(predictVals))
{
  total[train.white$quality[i]] = total[train.white$quality[i]]+1
  if (as.integer(predictVals[i]) == as.integer(train.white$quality[i]))
  {
       #      print(c(predictVals[i], train.red$quality[i]))
       #      print("Yes")
```

```
      count[train.white$quality[i]] = count[train.white$quality[i]]+1
  }
}
for (i in 1:10)
{
  print (c(count[i],total[i]))
  print(c("Quality", i," Value", as.double(count[i])/total[i]))
}
#[1] "trainMSE"            "0.727692307692308"

predictVals <- predict(fit, test.white)
predictVals <- as.character(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (predictVals[i] == test.white$quality[i])
  {
      count=count+1
  }
}
print(c("count",count,"test.length",length(test.white$quality)))
testMSE = count/length(test.white$quality)
print(c("testMSE",testMSE))
#[1] "testMSE"             "0.709030100334448"

predictVals <- predict(fit, test.white)
predictVals <- as.character(predictVals)
count <- rep(0,10)
total <- rep(0,10)
for (i in 1:length(predictVals))
{
  total[test.white$quality[i]] = total[test.white$quality[i]]+1
  if (as.integer(predictVals[i]) == as.integer(test.white$quality[i]))
  {
      #      print(c(predictVals[i], train.red$quality[i]))
      #      print("Yes")
      count[test.white$quality[i]] = count[test.white$quality[i]]+1
  }
}
for (i in 1:10)
{
  print (c(count[i],total[i]))
  print(c("Quality", i," Value", as.double(count[i])/total[i]))
}
#[1] "testMSE"             "0.709030100334448"

#Classification with Regression
```

```
model = quality ~
fixed.acidity+volatile.acidity+residual.sugar+free.sulfur.dioxide+density+pH+sulphates+alcoho
l
lm.white.8var <- lm(model, data=wine.white)
summary(lm.white.8var.model)

predictVals <- predict(lm.white.8var.model, test.white)
typeof(predictVals)
i =0
count = 0
for (i in 1:length(predictVals))
{
  if (round(predictVals[i]) == test.white$quality[i])
  {
       count= count+1
  }
}
print(c("count",count,"test.length",length(test.white$quality)))
testF = count/length(test.white$quality)
print(c("testF",testF))
```