ST 590 - Statistical Learning and Data Mining

Final Project - Analysis of Wine Quality Data

2 May 2016

Maria Adonay, Ying Cai, & Kaleigh Ragan

Table of Contents

Data Description

Observations from 6497 varieties of red and white wine were collected from the Vinho Verde region of Portugal. Of those varieties, 1599 were red and 4898 were white. Data on the following 12 properties were collected - fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. The first 11 observations were quantitative chemical properties while the 12^{th} observation was an ordinal score based on sensory data. This quality score was determined by taking the median of 3 independent testers' ratings on a scale from 1 (worst) to 10 (best).

Purpose

The purpose of this analysis was to identify relationships between chemical properties of wine to the human quality of tasting in order to make the quality appraisal more objective and less reliant on practices dependent on human assessment.

Regression Methods and Discussion

Ordinary Least Squares

First we fit an ordinary least squares (OLS) regression model to the data using all 13 predictors (Eq 1, p10). We see from the output (Figure 1, p11) that volatile acidity, chlorides, free sulfur dioxide, total sulfur dioxide, pH, sulphates and alcohol are all significant predictors in the model. The OLS model generally has good interpretability but it can be difficult to determine any interaction effect with so many variables in the model. We see the adjusted R^2 is 0.3561, which means we are only explaining about 35.6% of the variability in the data. This is a pretty low value and we may be able to improve the model performance, as well as interpretability, by implementing model selection techniques and regularization.

Subset Selection

Subset selection methods aim to reduce the size of the linear model by isolating a subset of the predictors that are believed to relate to the response. Three subset selection procedures were utilized in this analysis: best subset selection along with both forward and backward stepwise model selection. Best subset selection starts with the intercept only model then loops through all possible variable combinations for a model of size 1 through p, the number of predictors. Once all possible models of each size are generated, one best model from each size is selected based on lowest residual sum of squares (RSS) value, which is equivalent to selecting based on largest adjusted R^2. Forward stepwise selection starts with the intercept only model then considers all one-variable models, choosing the model with the smallest RSS, or largest adjusted R^2. It continues to add variables to the model one at a time, keeping on the predictor that gives the most improvement to the model, i.e. decreases RSS. Backwards stepwise selection is similar to the forward selection in that it proceeds through each model size one variable at a time. The key difference is that we start with the full model and at each step, remove the variable that gives the least improvement to the RSS.

In each of these methods we then need to decide which of the best size models given by the procedure would be the best fit overall. We can either indirectly or directly estimate the test error then choose the size model that gives the lowest test mean squared error (MSE). To indirectly estimate test error we can

use calculations that adjust the training error to compensate for bias caused by overfitting to the training data. The calculations we will use in this analysis are Mallow's Cp, BIC and the adjusted R^2. We will utilize the validation set approach to directly estimate the test error. This approach involves running the subset selection procedure on a set of training data and using each model generated to predict on a set of test data. We will compare the performance of each model in predicting on the test data and choose the model with lowest MSE. The last step of this process is to perform the subset selection method on the full data, choosing the model size indicated by the lowest test MSE, to achieve more accurate coefficient estimates. Other cross-validation methods could also be used to directly estimate the test error but we will focus on the validation set approach for this analysis.

We first ran the best subset procedure on the wine data and used indirect estimation of the test error to decide the best model size. We compared the values of Mallow's Cp, BIC and Adjusted R^2 for each model and selected the model with the minimum Mallow's Cp and BIC values and the model with the largest adjusted R^2. Figure 2 (p11) shows a plot of these values for each model size. Mallow's Cp indicated the seven variable model (Eq 2, p10) would give the best prediction, BIC choose the 6 variable model (Eq 3, p10) as the best predictor of wine quality and adjusted R^2 chose the eight variable model (Eq 4, p10) to be best. Using the validation set approach we see the six variable model gives the best fit to the data. When we apply the best subset selection method to the entire dataset and choose the 6 variable model, we get the same coefficients on the predictors as the model selected by BIC.

Forward subset selection gives the exact same results as best subset selection when using the indirect estimators for test MSE. The validation set approach chooses the seven variable model which is the same model chosen by the Mallow's Cp criteria on best subset selection. Backward selection yields the exact results given by forward selection, using both direct and indirect estimation. Based on these results we will choose the seven variable model (Eq 2, p10) to be the best subset of our predictors, since is it selected most often..

Shrinkage Methods: Ridge Regression & Lasso

Shrinkage methods are an alternative to subset selection. Instead of completely removing predictors from the model, we can shrink the coefficient estimates down toward zero. This is a nice alternative because shrinking the coefficients reduces the variance in the estimates while keeping all variables in the model. For our analysis, in the attempt to fit the best least squares model possible, we will use two shrinkage methods: ridge regression and the lasso method.

Ridge regression estimates the coefficients much in the same way that least squares does. However, ridge regression adds an additional 'shrinkage penalty', a tuning parameter, lambda, times the sum of the squared coefficient estimates. The shrinkage penalty is small when all the estimates are close to zero, thus rewarding the model when the values are shrunk toward zero. If the tuning parameter is equal to zero then the shrinkage penalty is zero and has no effect, giving us just the OLS model. As the tuning parameter goes to infinity, the shrinkage penalty has more effect and shrinks the coefficient estimates down to zero. This method gives a different set of coefficients for the full model for every value of the tuning parameter. The lasso method is similar to ridge regression except the tuning parameter, lambda, is multiplied by the sum of the absolute value of each coefficient. This slight change in the shrinkage penalty actually allows for some coefficients to be zero when the tuning parameter is large enough. Thereby shrinking the coefficient estimates and also essentially performing subset selection. We will use cross-validation to select the tuning parameter in both methods.

To fit a model predicting wine quality using ridge regression, we first use 10-fold cross-validation to choose the tuning parameter, lambda. The value of 0.0482 is selected as the lambda with the smallest cross-validation error yielding a test MSE of 0.4202. Some of the coefficients come very close to zero (free sulfur dioxide and total sulfur dioxide) but no coefficients are pushed to exactly zero (Eq 5, p10). We also use 10-fold cross-validation to choose the tuning parameter of the lasso method. For our data the lambda with the smallest cross-validation error is 0.0097, which gives a test MSE of 0.4365. In this model the predictors fixed acidity, citric acid and density are all shrunk to zero (Eq 6, p10). The ridge regression method does give a model with slightly lower test MSE, however, the difference in the error is not huge so we would likely use the model given by the lasso method because it is easier to interpret.

Dimension Reduction Methods: Principal Components Regression & Partial Least Squares

The difference with dimension reduction methods and previously mentioned methods is that dimension reduction methods involve transforming the predictors rather than using a combination of the predictors in their original state. Once the predictors have been transformed into something that is smaller in dimension, but still thought to retain all of the information from the original predictors, a least squares model is fit using those transformed variables.

Although the wine quality dataset does not involve a massive number of predictors, it is still possible to lessen the number of predictors necessary to accurately determine the quality of a wine. It is important to note that with an approach like principal components regression (PCR), the interpretability of a model is compromised.

Principal components (PCs) are meant to reduce the variance in observations through identifying an initial direction within the data that accounts for the most variation. The subsequent PC is the one that accounts for the most of the remaining variation, while remaining perpendicular to the direction of the first PC. The remaining PCs follow this same process until there are no more remaining dimensions within the data. This sequential method for determining PCs ensures that each direction is orthogonal to all of the others, and is a combination of information from a combination of all the original predictors, i.e. unsupervised and *not* a feature selection method.

Thus, since each PC has the possibility of incorporating information from all of the predictors, an interpretation is not always clear. While it is possible to look at the loadings, i.e. the weights of the original predictors in the new PC, to back-calculate the contributions of each original predictor, it is not always intuitive. However, when we recall the purpose of this analysis - to devise a more objective method for determining wine quality - we acknowledge that we may not necessarily need an interpretive model. Acting under the assumption that the physicochemical data is easy to collect, we may argue that we are willing to sacrifice some interpretability for a more predictive model.

The number of PCs, M, for PCR is generally determined through cross-validation. As seen in Figure 3a (p11), the lowest cross-validation error (0.6673) occurs when there are $M = 9$ components. Thus, these nine PCs were selected out of the possible 11 to be incorporated into the PCR model for prediction.

In contrast to the unsupervised nature of PCR, partial least squares (PLS) is a dimension reduction method that takes the response into account while creating a new set of features to be later incorporated in a least squares model. The iterative process of determining the new features of PLS is quite similar to that of PCR, except that each each new, orthogonal direction does not necessarily emphasize the

minimization of variance in the observations. Rather, it attempts to control for variance while also considering the relationship between the original predictors and the response.

Similar to PCR, the number of PCs, *M*, for PLS is also generally determined through cross-validation. As seen in Figure 3b (p11), the lowest cross-validation error (0.6678) occurs when there are $M = 4$ components. Thus, these four PCs were selected out of the possible 11 to be incorporated into the PLS model for prediction.

Comparison of Regression Methods

Best subset selection is computationally intensive because there are 2^p different models, in our case 2048. Since there are over 2000 models to choose from, best subset selection is also prone to overfitting, which increases variation in the coefficient estimates. Forward and backward selection are less computationally intensive but are not guaranteed to find the best possible model overall. A constraint of backward selection is that it cannot be used then the number of predictors is greater than the number of observations in the sample; however, we do not have to worry about that with our data.

We looked at ridge regression in comparison to least squares because ridge regression offers some advantages over OLS. When the tuning parameter in ridge regression is increased we see a decrease in model variance. There is always a trade-off between bias and variance in a model but the increase in bias for ridge regression is rather small. If we can substantially decrease variance while only increasing bias slightly, this will lead to a better model for our prediction. Ridge regression also has computational advantages over the subset selection methods since we can compute all models for different lambda values at once. The biggest drawback to ridge regression is that all variables remain in the model no matter how small their coefficients are. This is generally fine for model accuracy but will cause our model to be less interpretable. To combat these disadvantages we applied the lasso shrinkage method. It has the advantage of reducing variance with a slight increase in bias while also letting some coefficients go to zero therefore increasing interpretability of our model. It did not perform as well as the ridge regression, however, but the methods yielded very similar results.

In considering PCR and PLS, we note that both methods yield predictors of reduced dimension that are thought to preserve that information contained by the original predictors. However, while PCR focuses on explaining the variation in observations, PLS focuses on explaining the variation in observations while also considering the relationship to the response. This means that PLS, while often not as closely fitted to the predictors as PCR, tends to better explain the response. Yet, we note that the advantages that PLS presents over PCR to reduce bias frequently increase variance. Thus, one dimension reduction method does not typically perform much better than the other.

Based on test MSE, we find that forward subset selection and ridge regression give the best fit to the data (Figure 4a, p12). We see this reflected in the plot of adjusted R-squared values for each test (Figure 4b, p12). R-squared, We find that the methods rank in the following order, from least to greatest accuracy: forward subset selection, ridge regression, backward subset selection, lasso, best subset selection, principal components regression, ordinary least squares, partial least squares.

Classification Methods and Discussion

For the classification methods, we first need to collapse the ratings into categorical levels. The range of quality is (3,8), and the number of observations at each rate is given in Table 1a (p15). We decided to collapse the ratings into two levels: Low (3,4,5) and High (6,7,8). These levels were chosen so each will have an approximately equal number of observations. If we chose to collapse the ratings into three levels (Low/Med/High), the Med level would be very large while the High and Low levels would have too few observations.The number of observations in each class in both the training dataset and the test dataset are given in Table 1b (p15).

Logistic Regression

Logistic regression is a classification method which gives an output that is the probability our response belongs to a particular class. This is a simple, but useful model, that performs well on data that is linearly related. To predict the probability of belonging to a particular class, the model we use is *e* to the power of our linear function over one plus *e* to the power of the linear function, i.e. $\frac{e^{power\ of\ linear\ function}}{1+e^{power\ of\ linear\ function}}$. This ensures that we get a value in the range (0,1), which is in line with the interpretation of probabilities.

In our analysis, we fit the multiple logistic regression as our first method in predicting the quality of a wine. Using all predictors in our regression initially, we find that the *p*-values for some predictors and the intercept (β_0) is larger than 0.05, and even near 1 in some cases. This indicates that some of the predictors are not related to the response in this model. Therefore, we use the best subset selection and the lasso shrinkage method again on our newly classified data to select a subset of predictors to use in our model. Both methods select a 5-variable model containing the following variables: volatile acidity, chlorides, total sulfur dioxide, sulphates, and alcohol. This gives a result of 0.77, meaning we correctly predicted the class of the observation 77% of the time. Our misclassification error rate on the training data is then 23%.

K-Nearest Neighbors

K-nearest neighbors (KNN) is a classification method that does not require knowledge of the distribution of the response given the predictors. For this method we compute the distance, usually Euclidean, between all pairwise points, choose the *K* closest observations to a point, and predict to the class with the highest proportion of points in that group. The choice of *K* is very important in this method. A smaller *K* looks only at observations close to the training point and therefore results in what is known as a 'local fitting'. This very flexible method results in low bias but can lead to very high variance. On the other hand, a large value of *K* will give a more global, inflexible fit with high bias and lower variance.

To perform K-nearest neighbors classification, we first need to choose the value of *K*. Based on our cross-validation procedure, and by looking at the ROC, accuracy, sensitivity and specificity plots (Figure 5, p12-13), we determine the best value of *K* to use in predicting wine quality based on a subset of its chemical properties is one. We then run the KNN procedure with *K* = 1 and obtain a misclassification error rate of 31.1%. This method does not seem suitable for our dataset since we achieve a lower misclassification rate just by using regular logistic regression.

Tree-Based Methods: Random Forest and Boosting

Tree based methods aim to stratify the predictor space into simpler regions containing only a handful of the observations in a dataset. Since we are using this method for classification, i.e. to predict the response of an observation that falls in a particular region, we would find the mode, i.e. the class with the highest proportion in that region. Trees have the advantage of being easily interpreted and are very simple and easy to visualize. However, one tree created for a dataset may not predict as well as some of the other classification methods because of its simplistic nature. In this case, we can use random forest, bagging, and boosting as ways to improve the misclassification rate of a decision tree. Bagging uses bootstrap sampling to repeatedly take subsets of the data, create a tree for each sample, and average the predictions from all created trees. This reduces variance and keeps the bias low. However, since we are sampling over and over from the same data, this tends to produce decision trees that are very similar. This means the trees are highly correlated and variance would not be reduced as much. Instead, we can use random forests, which is similar to the bagging method but instead, each tree is built with only a subset of the predictors. This reduces the correlation among the trees and reduces the variance as well. Boosting works in much the same way as bagging, but instead of creating a new tree with every sample, the tree is grown, information from previous trees is added progressively. This slow approach reduces the risk of overfitting to the training data and usually performs well in predicting the response. As such, we will explore random forests and boosting in our analysis of the wine data.

In our analysis we constructed a decision tree, pruned to size eight using cross-validation (Figure 6, p13), and applied random forests and boosting to improve the method. The misclassification error rate for the decision tree with 8 terminal nodes is 25.5%. Using boosting we reduced the error rate to 20.5%. However, random forests had the best misclassification error rate of 15.9%, where the subset of predictors, *m*, was approximately equal to 3.

Support Vector Machines

The idea behind support vector classifiers and machines is simple. If we can draw lines separating the data into binary classes, whether the relationship is linear or nonlinear, we can predict the class of an observation by where it lies in relation to the hyperplane. The use of support vector machines is what allows the decision boundary to be non-linear. We can compute a linear boundary by using a linear kernel, or different types of non-linear boundaries, e.g. a radial kernel or a polynomial kernel.

We applied the support vector machine to the wine data using the linear, radial, and polynomial kernels in turn. The polynomial kernel gave a misclassification error rate of 23.4%. The linear and radial kernels give a slightly better fit with misclassification error rates of 22.2% and 19.9%, respectively. While these methods do well to reduce the misclassification error rate, they do not perform as well as the random forest, which also gives a much simpler interpretation of the data.

Comparison of Classification Methods

Logistic regression is a good overall starting point for classification problems. It is simple and can be easily interpreted. However, this method has drawbacks when it comes to highly non-linear data. In these cases, another method, such as KNN or decision trees may prove to yield better classification.

KNN works well when the data are highly non-linear because it is a non-parametric approach. This means that we make no assumptions about the data before we execute the method. The performance of K-

nearest neighbors depends heavily on the choice of *K*. Small values will overfit the training data while large values will give higher bias because they have a more global fit to the data. One area in which KNN is lacking is that it does not give any kind of importance to the predictors.

Decision trees usually also work best if there is a non-linear relationship between the response and the predictors since it does not follow the same kind of linear structure as that of other linear models. However, trees are extremely easy to interpret and therefore offer an advantage over other methods.

Support vector machines and support vector classifiers offer a flexible advantage over the other methods by allowing for some violations of the method and some misclassification. They can often produce similar results to the logistic regression, but logistic regression generally performs better when there is more overlap to the data.

Based on misclassification error rates, we find the random forests method applied to a classification tree gives the best prediction of wine quality (Figure 7, p13). The methods are ranked in the following order, from least to greatest accuracy: random forests, SVM with a radial kernel, boosting, SVM with a linear kernel, logistic regression, SVM with a polynomial kernel, pruned classification decision tree, K-nearest neighbors.

Issues

Due to time constraints, we could not include every regression, classification, or unsupervised learning method in our analysis. We started our analysis by selecting a handful of methods for both regression and classification.

The regression methods we employed did not include: polynomial regression, regression or smoothing splines, local regression, or generalized additive models. Looking back on our analysis, we feel next time we would incorporate some of these methods to improve the overall poor fit our regression methods gave to the data. All our regression methods resulted in test errors in the 40-50% range. These are terrible error rates and could have been improved by adding polynomial terms to the model, fitting with splines and using generalized additive models to predict wine quality using all, or a substantial subset, of our predictors. One regression methods in particular we did not use in our analysis was the regression tree method. We decided to use trees as a classification method instead of regression because we felt like we would be able to make class predictions more easily than predicting the exact numerical rating an individual would give.

For our classification methods we made the decision to leave out the linear discriminant analysis as well as quadratic discriminant analysis. The linear discriminant model, along the quadratic model, assume the distribution of the predictors in each class is approximately normal. We created histograms of each of the predictor variables (Figure 8a & 8b, p14) and saw only pH and density look approximately normal. Linear and quadratic discriminant analysis are good methods to use when the response variable has more than two classes. Discriminant analysis also has a similar procedure for fitting to logistic regression. Therefore, we would expect the two methods to give very similar results. For these reasons we decided it would not be advantageous to apply either discriminant analysis method to our dataset.

Upon initial consideration, we felt this dataset did not exactly fit in with the ideas of unsupervised learning, therefore we chose to focus on the supervised learning methods we knew we could apply. While

unsupervised learning methods can still be utilized on datasets with a response variable, they are intended to provide additional understanding of the relationship between predictors. We do have interest in the relationship between predictors but the goal of our analysis is to find the best method for predicting wine quality given a set of chemical properties of the wine. In future analyses, we could make use of unsupervised learning methods to determine the relationships among predictors and if we can exclude any predictors from our model based on these relationships.

Conclusion

There are many different statistical learning methods that can be applied to a dataset, both supervised and unsupervised. Our analysis did not touch on all possible methods, however, we feel we have thoroughly analyzed the data, using supervised learning techniques for both regression and classification approaches. The regression methods used were: ordinary least squares, subset selection (best, forward and backward), shrinkage methods (ridge regression and lasso) and dimension reduction methods (principal components regression and partial least squares). Of these methods, we determined that the forward subset selection method gave the best fit for predicting wine quality (Eq 2, p10). We also created a categorical rating variable from the numeric quality rating to test the following classification methods: logistic regression, K-nearest neighbors, classification trees (along with random forests and boosting) and support vector machines. Of these methods we determined Random Forest ($m = 3$) to be the best method for predicting high or low wine quality.

Our best regression method, forward subset selection, resulted in a test MSE of 0.419. The random forests method gave a much smaller classification error rate of 0.159 (Table 2, p15). While the forward subset selection performs the best out of all the regression methods we used in predicting wine quality, its values of test MSE is rather high. Meaning we are only predicting correctly about 40% of the time. This is not a very good error rate and would suggest that using random forests in classification decision trees would give a much better prediction of the wine quality rating by an individual, given a subset of the wine's chemical properties.

Appendix: Equations

(Eq 1)

$$\textbf{\textit{quality}} = 21.965 + 0.025 * fixed.acidity - 1.084 * volatile.acidity - 0.183 * citric.acid + 0.016 * residual.sugar - 1.874 * chlorides + 0.004 * free.sulfur.dioxide - 0.003 * total.sulfur.dioxide - 17.881 * density - 0.414 * pH + 0.916 * sulphates + 0.276 * alcohol$$

(Eq 2)

$$\textbf{\textit{quality}} = 4.43 - 1.013 * volatile.acidity - 2.019 * chlorides + 0.005 * free.sulfur.dioxide - 0.003 * total.sulfur.dioxide - 0.483 * pH + 0.883 * sulphates + 0.289 * alcohol$$

(Eq 3)

$$\textbf{\textit{quality}} = 4.296 - 1.038 * volatile.acidity - 2.002 * chlorides - 0.002 * total.sulfur.dioxide - 0.435 * pH + 0.889 * sulphates + 0.291 * alcohol$$

(Eq 4)

$$\textbf{\textit{quality}} = 4.668 - 1.074 * volatile.acidity - 0.13 * citric.acid - 1.949 * chlorides + 0.005 * free.sulfur.dioxide - 0.549 * pH + 0.891 * sulphates + 0.293 * alcohol$$

(Eq 5)

$$\textbf{\textit{quality}} = 32.891 + 0.031 * fixed.acidity - 1.006 * volatile.acidity - 0.07 * citric.acid + 0.019 * residual.sugar - 1.799 * chlorides + 0.004 * free.sulfur.dioxide - 0.003 * total.sulfur.dioxide - 29.069 * density - 0.298 * pH + 0.881 * sulphates + 0.251 * alcohol$$

(Eq 6)

$$\textbf{\textit{quality}} = 4.12 - 1.026 * volatile.acidity + 0.0002 * residual.sugar - 1.652 * chlorides + 0.002 * free.sulfur.dioxide - 0.003 * total.sulfur.dioxide - 0.366 * pH + 0.808 * sulphates + 0.285 * alcohol$$

Appendix: Figures

***Figure 1**: OLS Output*

```
> lm.fit=lm(quality~., data=wine)

> summary(lm.fit)

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
```

***Figure 2**: Mallow's Cp, BIC and Adjusted R^2 for each model*

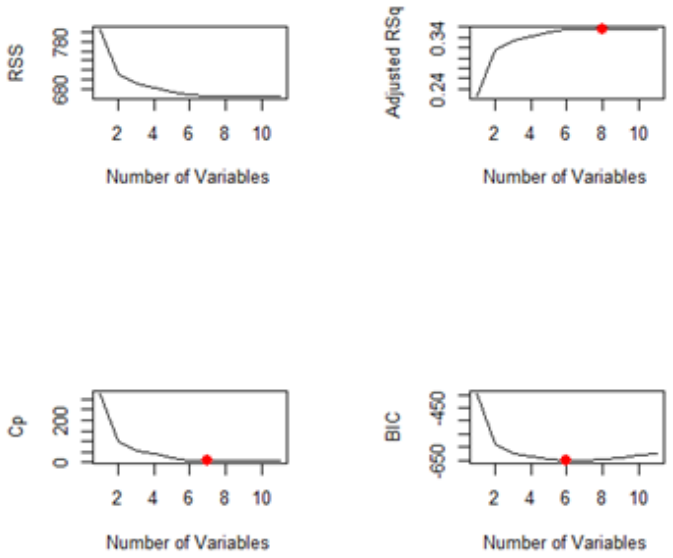
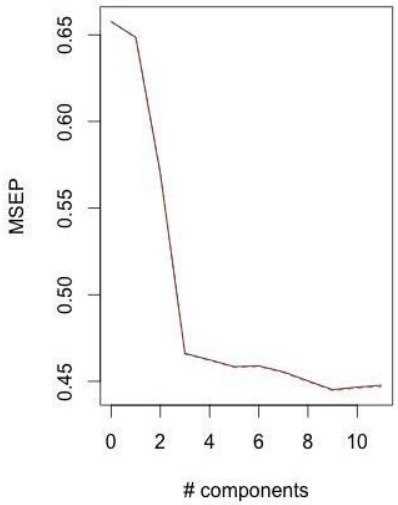


***Figure 3**: Component Number Selection*

a. PCR



b. PLS

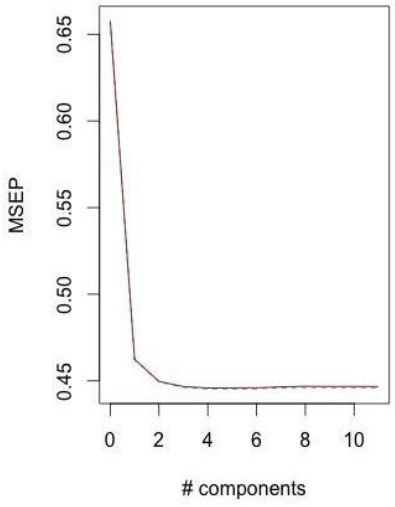
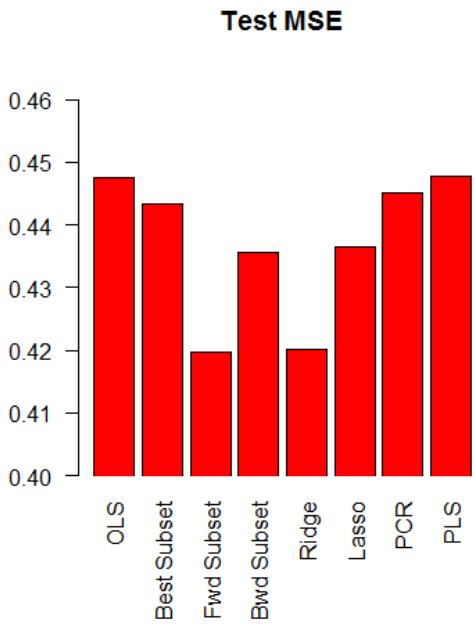


***Figure 4**: Regression Methods Comparison*

a. Test MSE



b. Test Adjusted R^2

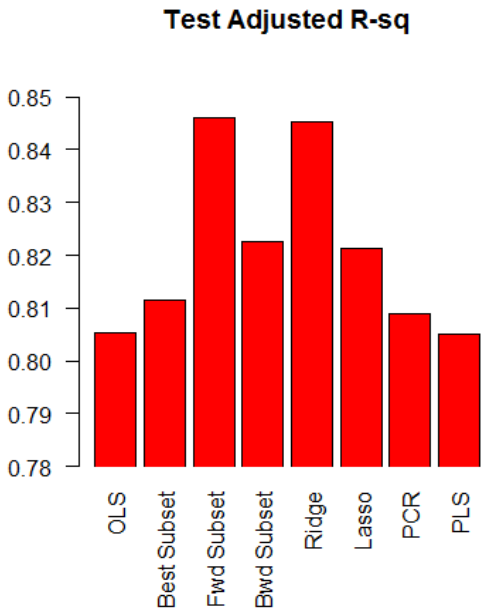
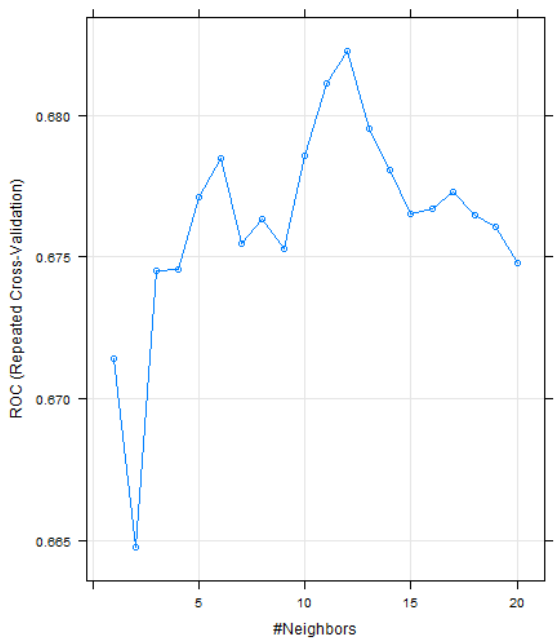
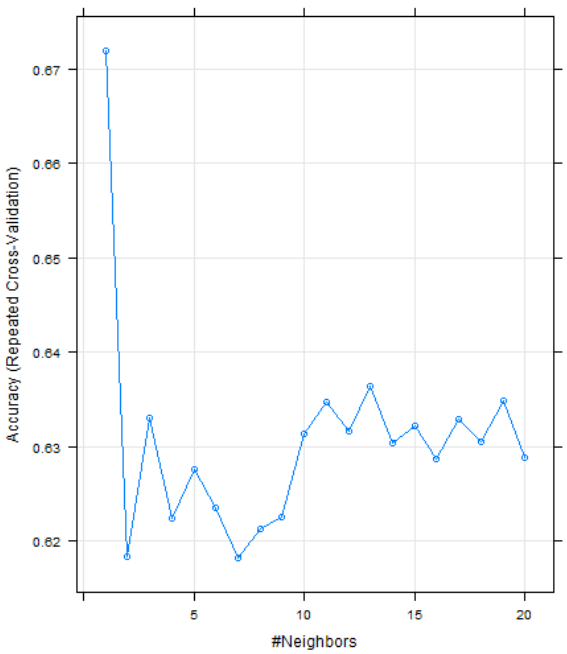


***Figure 5**: Best K for K-Nearest Neighbors*

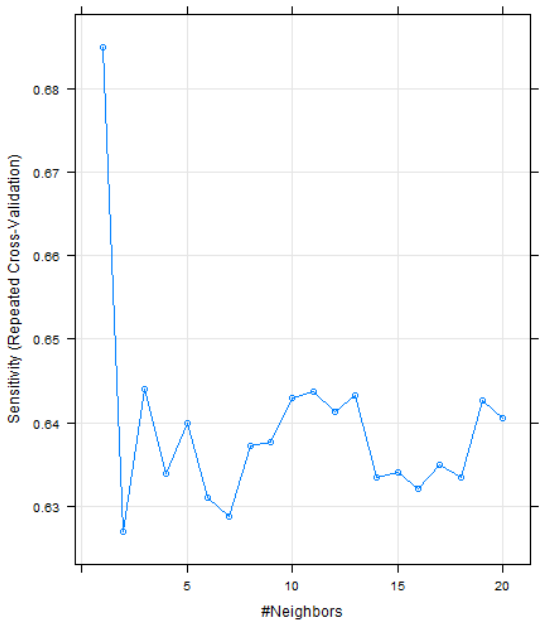
a. ROC



b. Accuracy



c. *Sensitivity*



d. *Specificity*

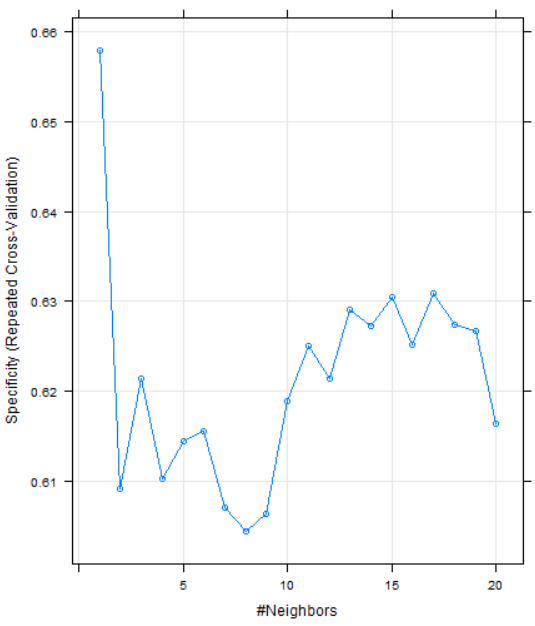


Figure 6: *Classification Tree (Pruned, Nodes=8)*

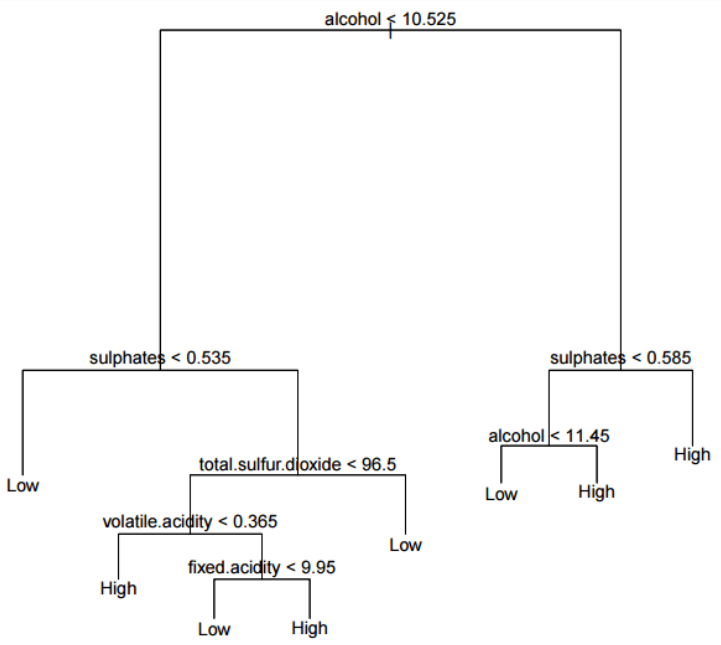


Figure 7: *Test Misclassification Error Rates*

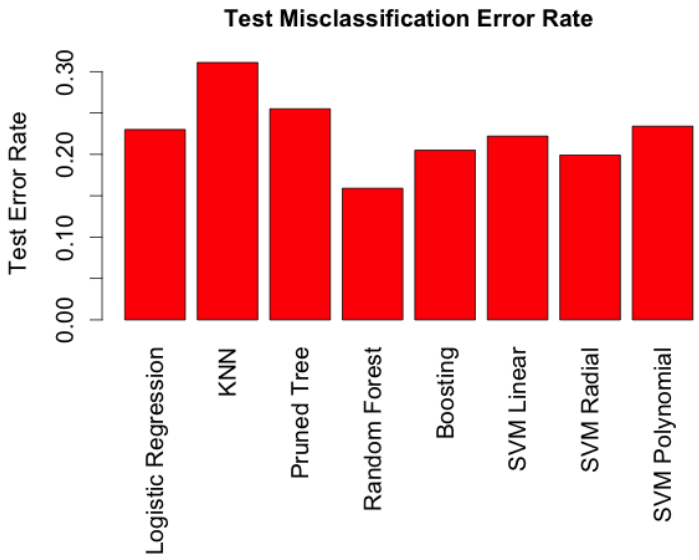
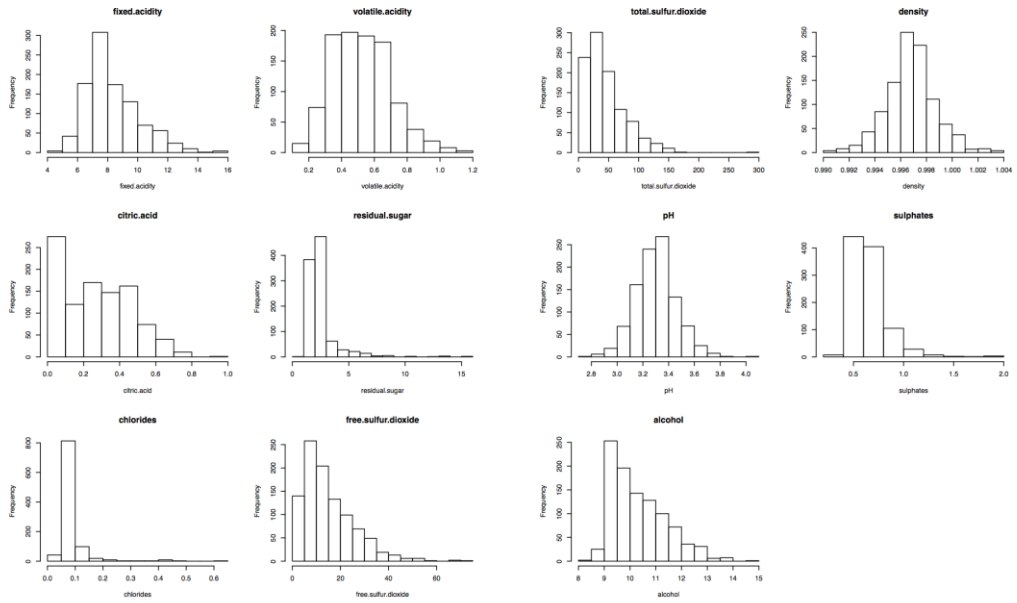
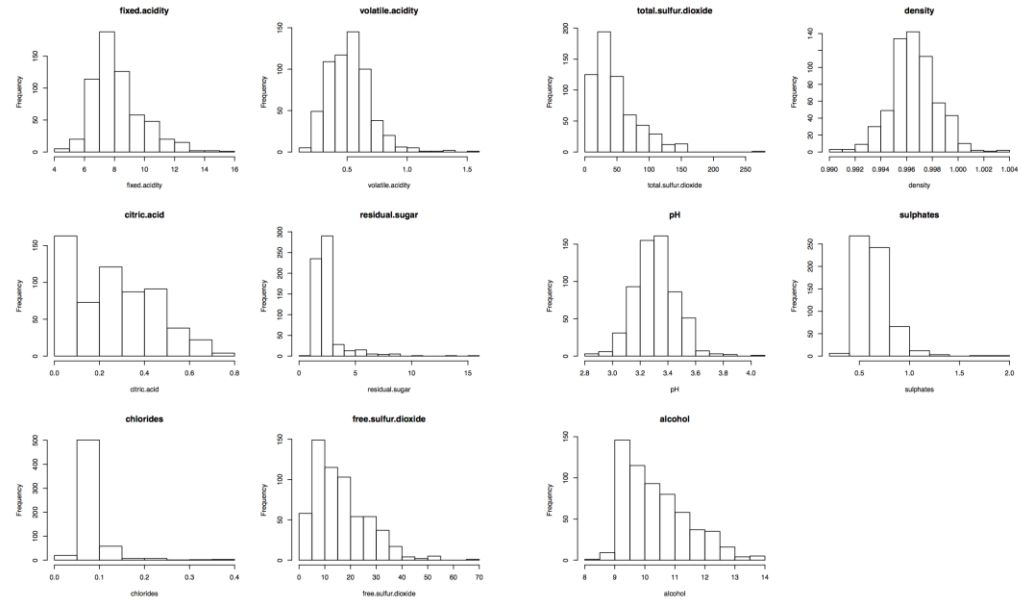


Figure 8: *LDA/QDA, Distributions of Predictors*

a. Training Data Set



b. Test Data Set



Appendix: Tables

***Table 1**: Dividing the Data into Test and Train.*

a. *Frequency of Each Rate*

```
> table(wine$quality)

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

b. *Training/Test Dataset Counts*

```
> table(wine.train$levels)

High  Low
 516  484
> table(wine.test$levels)

High  Low
 339  260
```

***Table 2:** Misclassification Error Rate Table*

Method	Logistic Regression	KNN (K = 1)	Pruned Tree (Nodes = 8)	Random Forest (m = 3)	Boosting (depth = 3)	SVM Linear	SVM Radial	SVM Polynomial
Test Error Rate	0.23	0.311	0.255	0.159	0.205	0.222	0.199	0.234

Appendix: Code

```
##############################################################################################
#### Title: ST 590 Final Project
#### Authors: Maria Adonay, Ying Cai, Kaleigh Ragan
#### Created: 4.14.16
#### Due: 5.2.16
#### Purpose: Analyze Wine Quality Data - Red Wine
##############################################################################################

# CLEAN SESSION
rm(list=ls(all=TRUE))
clc <- function() cat(rep("\n",50))
clc()

# READ IN DATA
require(RCurl)
u <- "http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
tc <- getURL(u, ssl.verifypeer=FALSE)
wine <- read.csv(textConnection(tc), sep=";")

  ##############################################################################################
                #### REGRESSION METHODS ####
  ##############################################################################################

# LIBRARY CALLS
library(ISLR)
library(leaps)
library(glmnet)
library(class)
library(caret)
library(pROC)
library(pls)

# SPLIT WINE DATA INTO TRAINING & TEST SETS
train <- sample(dim(wine)[1], 1000)
wine.train <- wine[train, ]
wine.test <- wine[-train, ]

##########################################
## ORDINARY LEAST SQUARES ##

lm.fit=lm(quality~., data=wine)
summary(lm.fit)

# Fit a linear model using least squares on the training set.
```

```
lm.fit <- lm(quality ~ ., data=wine.train)
lm.pred <- predict(lm.fit, wine.test)
# Report the test error obtained
lm.mse <- mean((wine.test[, "quality"] - lm.pred)^2)
lm.mse

#############################################
## SUBSET SELECTION: BEST SUBSET SELECTION ##

# Mallow's Cp, BIC and Adj R-Sq Criteria for single best model selection.
regfit.full=regsubsets(quality~., wine, nvmax=11)
reg.summary=summary(regfit.full)
which.max(reg.summary$adjr2)
which.min(reg.summary$cp)
which.min(reg.summary$bic)
# Plot each criteria and display min or max value where appropriate
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
points(which.max(reg.summary$adjr2),reg.summary$adjr2[8], col="red",cex=2,pch=20)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
points(which.min(reg.summary$cp),reg.summary$cp[7],col="red",cex=2,pch=20)
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(which.min(reg.summary$bic), reg.summary$bic[6],col="red",cex=2,pch=20)
coef(regfit.full, which.max(reg.summary$adjr2))
coef(regfit.full, which.min(reg.summary$cp))
coef(regfit.full, which.min(reg.summary$bic))

# Validation Set Approach for best subset model size selection.
set.seed(2)
train=sample(c(TRUE,FALSE), nrow(wine), rep=TRUE)
test=(!train)
regfit.best=regsubsets(quality~., data=wine[train,], nvmax=11)
test.mat=model.matrix(quality~., data=wine[test,])
val.errors=rep(NA,11)
for(i in 1:11){
  coefi=coef(regfit.best,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((wine$quality[test]-pred)^2)
}
# Print model size with lowest test MSE.
which.min(val.errors)
best.mse=val.errors[which.min(val.errors)]
best.mse
# Run best subset selection again on full data to obtain coefficients of the
# 6 variable model.
regfit.best=regsubsets(quality~., data=wine, nvmax=11)
coef(regfit.best, which.min(val.errors))
```

```
##########################################
## SUBSET SELECTION: FORWARD SELECTION ##

# Mallow's Cp, BIC and Adj R-Sq Criteria for best forward selection.
regfit.fwd=regsubsets(quality~., data=wine, nvmax=11, method="forward")
reg.fwd.summary=summary(regfit.fwd)
which.max(reg.fwd.summary$adjr2)
which.min(reg.fwd.summary$cp)
which.min(reg.fwd.summary$bic)
coef(regfit.fwd, which.max(reg.fwd.summary$adjr2))
coef(regfit.fwd, which.min(reg.fwd.summary$cp))
coef(regfit.fwd, which.min(reg.fwd.summary$bic))

# Validation Set Approach for best forward selection model size.
set.seed(123)
train=sample(c(TRUE,FALSE), nrow(wine), rep=TRUE)
test=(!train)
regfit.fwd=regsubsets(quality~., data=wine[train,], nvmax=11, method="forward")
test.mat=model.matrix(quality~., data=wine[test,])
val.errors=rep(NA,11)
for(i in 1:11){
  coefi=coef(regfit.fwd,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((wine$quality[test]-pred)^2)
}
# Print model size with lowest test MSE.
which.min(val.errors)
fwd.mse=val.errors[which.min(val.errors)]
fwd.mse
# Run forward subset selection again on full data to obtain coefficients of the
# 7 variable model.
regfit.fwd=regsubsets(quality~., data=wine, nvmax=11, method="forward")
coef(regfit.fwd, which.min(val.errors))

##########################################
## SUBSET SELECTION: BACKWARD SELECTION ##

# Mallow's Cp, BIC and Adj R-Sq Criteria for best backward selection.
regfit.bwd=regsubsets(quality~., data=wine, nvmax=11, method="backward")
reg.bwd.summary=summary(regfit.bwd)
which.max(reg.bwd.summary$adjr2)
which.min(reg.bwd.summary$cp)
which.min(reg.bwd.summary$bic)
coef(regfit.bwd, which.max(reg.bwd.summary$adjr2))
coef(regfit.bwd, which.min(reg.bwd.summary$cp))
coef(regfit.bwd, which.min(reg.bwd.summary$bic))
```

```
# Validation Set Approach for best forward selection model size.
set.seed(456)
train=sample(c(TRUE,FALSE), nrow(wine), rep=TRUE)
test=(!train)
regfit.bwd=regsubsets(quality~., data=wine[train,], nvmax=11, method="backward")
test.mat=model.matrix(quality~., data=wine[test,])
val.errors=rep(NA,11)
for(i in 1:11){
  coefi=coef(regfit.bwd,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((wine$quality[test]-pred)^2)
}
# Print model size with lowest test MSE.
which.min(val.errors)
bwd.mse=val.errors[which.min(val.errors)]
bwd.mse
# Run forward subset selection again on full data to obtain coefficents of the
# 7 variable model.
regfit.bwd=regsubsets(quality~., data=wine, nvmax=11, method="backward")
coef(regfit.bwd, which.min(val.errors))

############################################
## SHRINKAGE METHODS: RIDGE REGRESSION ##

x=model.matrix(quality~., wine)[,-1]
y=wine$quality
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x, y, alpha=0, lambda=grid)
set.seed(789)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
cv.out=cv.glmnet(x[train,], y[train], alpha=0)
bestlam=cv.out$lambda.min
bestlam
ridge.pred=predict(ridge.mod, s=bestlam, newx=x[test,])
ridge.mse=mean((ridge.pred-y.test)^2)
ridge.mse
out=glmnet(x, y, alpha=0)
predict(out, type="coefficients", s=bestlam)[1:12,]

############################################
## SHRINKAGE METHODS: THE LASSO ##

lasso.mod=glmnet(x[train,], y[train], alpha=1, lambda=grid)
set.seed(123)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
bestlam=cv.out$lambda.min
```

```
bestlam
lasso.pred=predict(lasso.mod, s=bestlam, newx=x[test,])
lasso.mse=mean((lasso.pred-y.test)^2)
lasso.mse
out=glmnet(x, y, alpha=1, lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:12,]
lasso.coef[lasso.coef!=0]

##########################################
## DIMENSION REDUCTION METHODS: PRINCIPAL COMPONENTS REGRESSION ##

# Fit a PCR model on the training set, choose M by CV.
par(mfrow=c(1,1))
pcr.fit <- pcr(quality ~ ., data=wine.train, scale=T, validation="CV")
validationplot(pcr.fit, val.type="MSEP", xlab="# components", ylab="MSEP", main="PCR")
# Report the test error obtained.
pcr.pred <- predict(pcr.fit, wine.test, ncomp=9)
pcr.mse <- mean((wine.test[, "quality"] - data.frame(pcr.pred))^2)
pcr.mse

##########################################
## DIMENSION REDUCTION METHODS: PARTIAL LEAST SQUARES ##

# Fit a PLS model on the training set, choose M chosen by CV.
pls.fit <- plsr(quality ~ ., data=wine.train, scale=T, validation="CV")
validationplot(pls.fit, val.type="MSEP", xlab="# components", ylab="MSEP", main="PLS")
# Report the test error obtained.
pls.pred <- predict(pls.fit, wine.test, ncomp=4)
pls.mse <- mean((wine.test[, "quality"] - data.frame(pls.pred))^2)
pls.mse

#############################################################################################

## COMPARISON OF REGRESSION METHODS ##

test.avg <- mean(wine.test[, "quality"])
# Comparison of test MSE
barplot(c(lm.mse, best.mse, fwd.mse, bwd.mse, ridge.mse, lasso.mse, pcr.mse,
          pls.mse), col="red",xpd = FALSE, las=2, ylim=c(0.4,0.46),
        names.arg=c("OLS", "Best Subset", "Fwd Subset", "Bwd Subset", "Ridge",
                    "Lasso", "PCR", "PLS"), main="Test MSE")
# Comparison of Adjusted R-squared
lm.test.r2 <- (1 - lm.mse) /mean((wine.test[, "quality"] - test.avg)^2)
best.test.r2 <- (1 - best.mse) /mean((wine.test[, "quality"] - test.avg)^2)
fwd.test.r2 <- (1 - fwd.mse) /mean((wine.test[, "quality"] - test.avg)^2)
bwd.test.r2 <- (1 - bwd.mse) /mean((wine.test[, "quality"] - test.avg)^2)
ridge.test.r2 <- (1 - ridge.mse) /mean((wine.test[, "quality"] - test.avg)^2)
```

```
lasso.test.r2 <- (1 - lasso.mse) /mean((wine.test[, "quality"] - test.avg)^2)
pcr.test.r2 <- (1 - pcr.mse) /mean((wine.test[, "quality"] - test.avg)^2)
pls.test.r2 <- (1 - pls.mse) /mean((wine.test[, "quality"] - test.avg)^2)
barplot(c(lm.test.r2, best.test.r2, fwd.test.r2, bwd.test.r2, ridge.test.r2,
        lasso.test.r2, pcr.test.r2, pls.test.r2), col="red",xpd = FALSE,
       las=2, ylim=c(0.78,0.85),
       names.arg=c("OLS", "Best Subset", "Fwd Subset", "Bwd Subset", "Ridge",
                "Lasso", "PCR", "PLS"), main="Test Adjusted R-sq")

  ##############################################################################
                ####                CLASSIFICATION METHODS                ####
  ##############################################################################

# LIBRARY CALLS
library(tree)
library(randomForest)
library(gbm)
library(e1071)

table(wine$quality)

# COLLAPSE RATINGS INTO HIGH AND LOW CATEGORIES
wine$levels=ifelse(wine$quality<=5,'Low','High')
wine=wine[,-12]

# # SPLIT WINE DATA INTO TRAINING & TEST SET AGAIN W/ NEW RATING VARIABLE
set.seed(1)
train <- sample(dim(wine)[1], 1000)
wine.train <- wine[train, ]
wine.test <- wine[-train, ]

table(wine.train$levels)
table(wine.test$levels)

###############################################
## LOGISTIC REGRESSION ##

# Use all predictors for logistic regression.
glm.fit=glm(as.factor(levels)~., wine.train, family=binomial)
summary(glm.fit)

# Use Best Subset Selection to choose a reduced model.
set.seed(2)
regfit.train=regsubsets(as.factor(levels)~.,wine.train,nvmax=11)
which.min(summary(regfit.train)$bic)
summary(regfit.train)$which[5,]
```

```
# Use the lasso shrinkage method to choose a reduced model.
set.seed(2)
x=model.matrix(levels~.,wine)[,-1]
y0=wine$levels
y=ifelse(y0=='Low',0,1)
grid=10^seq(10,-2,length=100)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
bestlam=cv.out$lambda.min
lasso.out=glmnet(x[train,],y[train],alpha=1,lambda=grid)
lasso.coef=predict(lasso.out,type='coefficients',s=bestlam)[1:12,]
lasso.coef[lasso.coef!=0]

# Use reduced models given above for logistic regression.
set.seed(3)
glm.fit=glm(as.factor(levels)~volatile.acidity+chlorides+total.sulfur.dioxide+sulphates+alcohol,wine.train,fa
mily=binomial)
summary(glm.fit)
glm.probs=predict(glm.fit,wine.test,type='response')
contrasts(as.factor(wine.train$levels))
glm.pred=rep('High',dim(wine.test)[1])
glm.pred[glm.probs>0.5]='Low'
table(prediction=glm.pred,true=wine.test$levels)
log.mcer=1-mean(glm.pred==wine.test$levels)
log.mcer

###########################################
## LINEAR DISCRIMINANT ANALYSIS ##

varname=names(wine)[1:11]
pdf(file='C:\\Users\\YingCai\\Desktop\\st590\\Train Data
Hist.pdf',paper='a4r',height=10,width=8,title='Histograms of Train Data')
par(mfrow=c(3,4))
for(i in 1:11){
  hist(wine.train[,varname[i]],main=varname[i],xlab=varname[i])
}
dev.off()

pdf(file='C:\\Users\\YingCai\\Desktop\\st590\\Test Data
Hist.pdf',paper='a4r'',height=10,width=8,title='Histograms of Test Data')
par(mfrow=c(3,4))
for(i in 1:11){
  hist(wine.test[,varname[i]],main=varname[i],xlab=varname[i])
}
dev.off()

###########################################
## K NEAREST NEIGHBORS ##
```

```
set.seed(3)
best.k=train(levels~., data=wine.train, method='knn',
          tuneGrid=expand.grid(.k=1:20),
          trControl=trainControl(method='repeatedcv', number=10, repeats=15,
                                 classProbs=T, summaryFunction=multiClassSummary))
## Choose K=1.
plot(best.k,metric='ROC')
plot(best.k,metric='Accuracy')
plot(best.k,metric='Sensitivity')
plot(best.k,metric='Specificity')
set.seed(3)
knn.pred=knn(x[train,],x[-train,],y0[train],k=1,prob=T)
table(prediction=knn.pred,true=wine.test$levels)
knn.mcer=1-mean(knn.pred==wine.test$levels)
knn.mcer

###########################################
## PRUNED TREE ##

tree.wine=tree(as.factor(levels)~.,wine.train)
set.seed(4)
cv.wine=cv.tree(tree.wine,FUN=prune.misclass)
cv.wine
## Best size = 8
prune.wine=prune.misclass(tree.wine,best=8)
tree.pred=predict(prune.wine,wine.test,type='class')
table(prediction=tree.pred,true=wine.test$levels)
plot(prune.wine)
text(prune.wine,pretty=0)
tree.mcer=1-mean(tree.pred==wine.test$levels)
tree.mcer

###########################################
## RANDOM FOREST ##

set.seed(4)
rf.wine=randomForest(as.factor(levels)~.,data=wine.train,mtry=3,ntree=1423,importance=T)
rf.pred=predict(rf.wine,newdata=wine.test,type='response')
table(prediction=rf.pred,true=wine.test$levels)
rf.mcer=1-mean(rf.pred==wine.test$levels)
rf.mcer

###########################################
## BOOSTING ##

set.seed(4)
wine10=data.frame(x,y)
```

```
boost.wine=gbm(y~.,data=wine10[train,],distribution='bernoulli',n.trees=10000,interaction.depth=3,shrinka
ge=0.001)
boost.prob=predict(boost.wine,newdata=wine10[-train,],n.trees=10000,type='response')
boost.pred=rep('Low',dim(wine.test)[1])
boost.pred[boost.prob>0.5]='High'
table(prediction=boost.pred,true=wine.test$levels)
boost.mcer=1-mean(boost.pred==wine.test$levels)
boost.mcer

##########################################
## SVM: SUPPORT VECTOR CLASSIFIER ##

levels.factor=as.factor(wine$levels)
fac.wine=data.frame(wine[,-12],levels.factor)
set.seed(3)
tune.out=tune(svm,levels.factor~.,data=fac.wine[train,],kernel='linear',ranges=list(cost=c(0.1,1,2,3,5)))
summary(tune.out)
table(prediction=predict(tune.out$best.model,newdata=wine.test),truth=wine.test$levels)
SVC.mcer=1-mean(predict(tune.out$best.model,newdata=wine.test)==wine.test$levels)
SVC.mcer

##########################################
## SVM: RADIAL KERNEL ##

set.seed(4)
tune.out=tune(svm,levels.factor~.,data=fac.wine[train,],kernel='radial',ranges=list(cost=c(0.1,1,10,100),ga
mma=c(0.2,0.5,1,2)))
summary(tune.out)
table(prediction=predict(tune.out$best.model,newdata=wine.test),truth=wine.test$levels)

##########################################
## SVM: POLYNOMIAL  KERNEL ##

set.seed(4)
tune.poly=tune(svm,levels.factor~.,data=fac.wine[train,],kernel='polynomial',ranges=list(cost=c(0.1,0.5,1,1
0),degree=c(1,2,3,4)))
summary(tune.poly)
table(prediction=predict(tune.poly$best.model,newdata=wine.test),truth=wine.test$levels)

#############################################################################################

## COMPARISON OF CLASSIFICATION METHODS ##

a=c('Logistic Regression','KNN','Pruned Tree','Random Forest','Boosting','SVM Linear','SVM Radial','SVM
Polynomial')
b=c(0.23,0.311,0.255,0.159,0.205,0.222,0.199,0.234)
names(b)=a
```

```
par(mai=c(3,1.5,1,1))
barplot(b,col='red',width=0.5,cex.names=1.2,cex.lab=1.2,cex.axis=1.2,cex.main=1.2,las=3,ylab='Test
Error Rate',las=3,main='Test Misclassification Error Rate')

################################################################################
```