PSTAT 131 Final Project

Alek Lichucki(8938326), Chun Jen Chung(5248305), Eric Luong(4987632), Rohan Mani(7975915)

Introduction

What factors determine the quality of white wine originating from Portugal, also known as Vinho Verde? During this project, we attempted to find the best machine learning model to predict whether white wine is above the median quality or not. We used a data set that has many different statistics about each wine, as well as the quality of the wine. From this, we attempted to use the data to predict whether or not the wine is above the median quality or not. To do this we use 4 different machine learning methods: K nearest neighbors, logistic regression, classification tree, and random forest. Once we build the model we will evaluate each model using a test set, as well as 10-fold cross-validation to then find a final model that best predicts if white wine is above the median quality or not.

Data

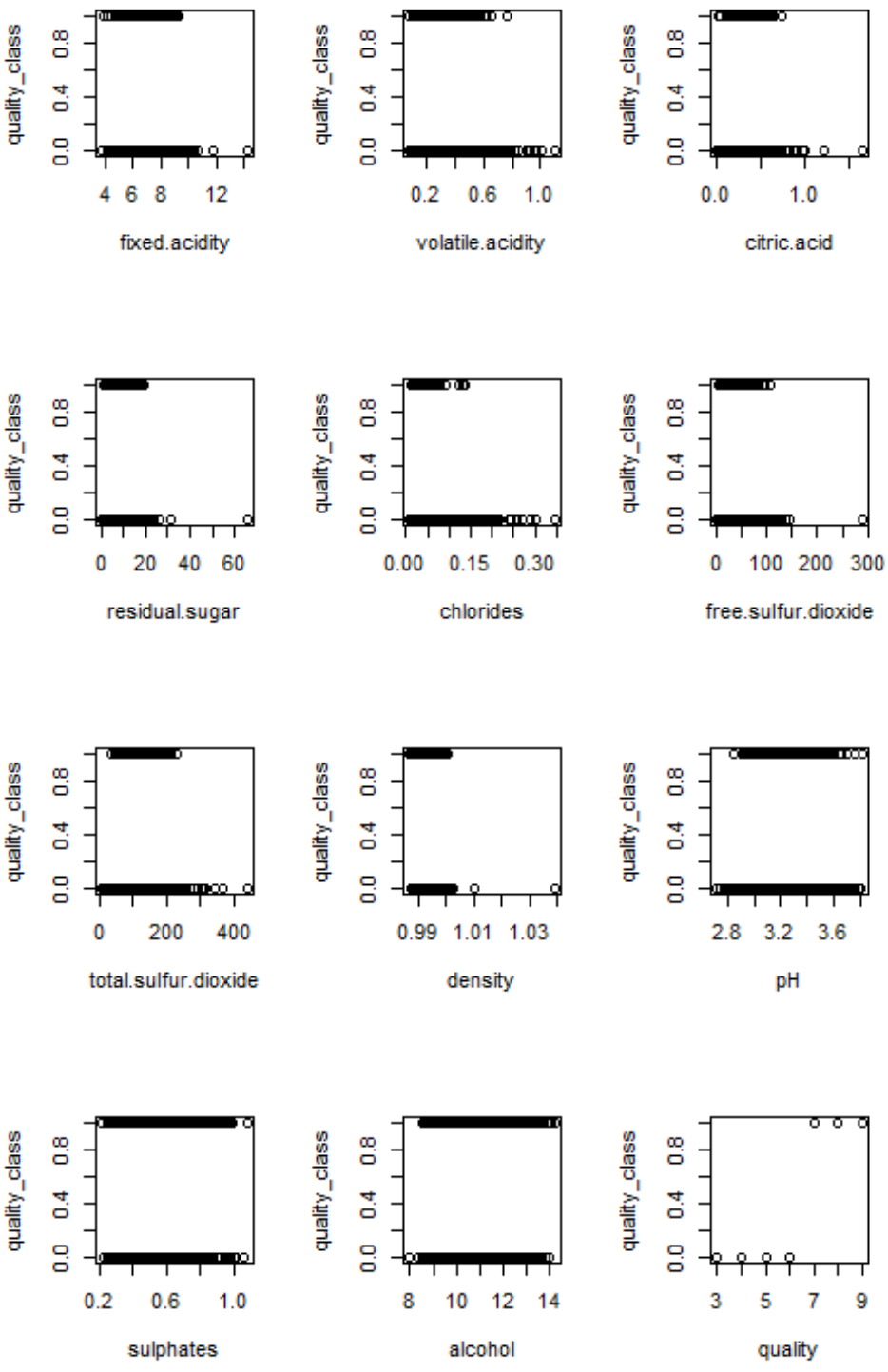
Data Set URL: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

The original response variable was quality, this is a categorical variable with 9 different levels indicating the quality of the wine. This ranges from 1-9 with 9 being the best quality wine. We then used this to make the response variable we used for the project which is quality_class which is a binary variable with 0 being below the median quality and 1 being above the median quality. Our dataset includes 11 explanatory features worth investigating: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. Each of these are continuous variables with no missing data. The table of summary statistics is included below. In this we also made the column for quality_class, to determine this we found whether or not each observation was above the median or not, if the observation's quality was above the median we assigned it to 1, otherwise, it was assigned to 0. We also used stepwise regession to find which variables are significant predictors.

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
```

```
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol         quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000

##
## Call:
## glm(formula = quality_class ~ (fixed.acidity + volatile.acidity +
##     citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + density + pH + sulphates + alcohol +
##     quality) - quality, family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1436  -0.6725  -0.4114  -0.1798   2.8331
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           6.362e+02  9.412e+01   6.759 1.39e-11 ***
## fixed.acidity         5.521e-01  9.053e-02   6.099 1.07e-09 ***
## volatile.acidity     -3.785e+00  4.885e-01  -7.749 9.28e-15 ***
## citric.acid          -7.378e-01  4.010e-01  -1.840 0.065776 .
## residual.sugar        2.952e-01  3.564e-02   8.283  < 2e-16 ***
## chlorides            -1.264e+01  3.816e+00  -3.312 0.000926 ***
## free.sulfur.dioxide   8.645e-03  3.130e-03   2.762 0.005749 **
## total.sulfur.dioxide -2.696e-04  1.506e-03  -0.179 0.857936
## density              -6.591e+02  9.540e+01  -6.909 4.89e-12 ***
## pH                    3.343e+00  4.268e-01   7.832 4.81e-15 ***
## sulphates             2.168e+00  3.475e-01   6.238 4.42e-10 ***
## alcohol               1.423e-01  1.139e-01   1.250 0.211334
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5116.8  on 4897  degrees of freedom
## Residual deviance: 4143.2  on 4886  degrees of freedom
## AIC: 4167.2
##
## Number of Fisher Scoring iterations: 6
```


quality_class
0.0
0.4
0.8
4 6 8 12
fixed.acidity
0.2 0.6 1.0
volatile.acidity
0.0 1.0
citric.acid
0 20 40 60
residual.sugar
0.00 0.15 0.30
chlorides
0 100 200 300
free.sulfur.dioxide
0 200 400
total.sulfur.dioxide
0.99 1.01 1.03
density
2.8 3.2 3.6
pH
0.2 0.6 1.0
sulphates
8 10 12 14
alcohol
3 5 7 9
quality

Methods

Our analysis plan is to use 4 different methods. We will first use KNN, then delve deeper into logistic regression, proceed with examples of a tree diagram, and end with random forest. Random forest is our ensemble method, while the others are non-ensemble methods. We will carry out the model selection by comparing the average test error for each of the different methods. We will find this average test error by looking at each of the test errors for each fold. We are using 10-fold cross-validation, so each fold will have about 450 observations in it, resulting in the test set is about 450 observations while the training set has a little over 4000 observations. For the tree model, we will use pruning to tune the model more. For logistic regression, we are seeking to minimize the FPR and FNR to reduce test error, aligning our threshold where we find the minimum. We will find the best threshold value for each of the folds and average them to get the average threshold value. For KNN we will also look to minimize error and through this, we will find the optimal number of neighbors to use for the model. Finally, with random forests, we will also look to minimize the error by finding the optimal number of entries by using 10-fold cross-validation and use the built-in function randomForest to find the optimum trees. We will also use stepwise regression to find which predictors to use and which ones to not use.

Model Building

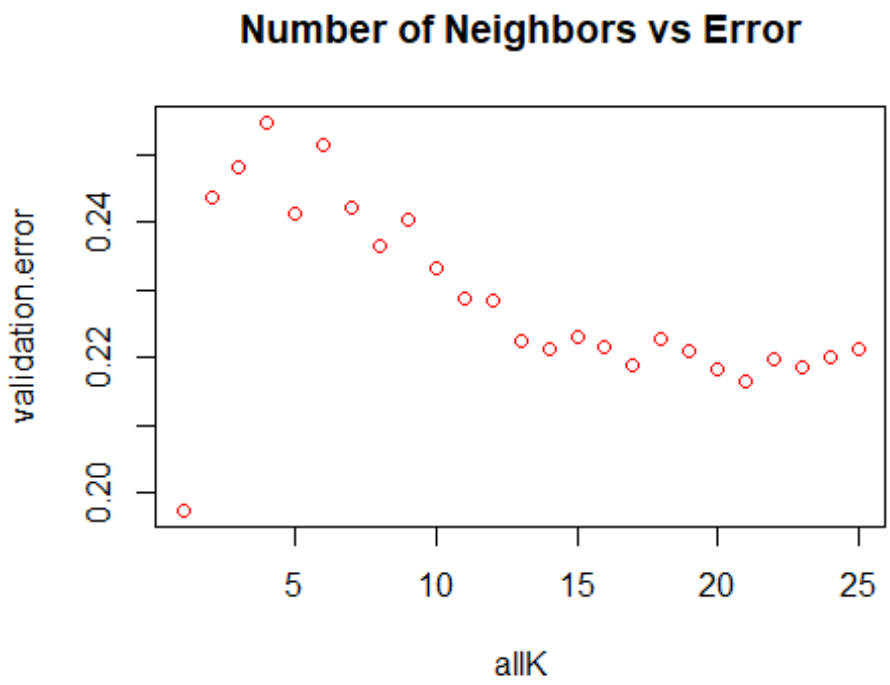
K Nearest Neighbors

The first method that we considered was KNN based on the initial training and test data. We first looked to see which variables had a relationship with our data. We found that quality_class shouldn't be used since this is what we are trying to classify. And quality shouldn't since quality_class is derived from quality. We then also found that the variables alcohol, total.sulfur.dioxide, and citric.acid shouldn't be used. This is due to these variables having little relationship to predicting quality_class. Once we found which variables that we wanted to include, we began to work on building the KNN model. Once we had the initial KNN model, we then had to decide which value of k was best. We classified the best value of k as the one that minimized the test error. From this, we found that the value that best minimized the test error was k = 1.

```
## [1] 1
```

Once we found the best value of k, we then found the test error of the KNN model. After finding the test error from the model, we performed 10-fold cross-validation. From this, we confirmed that for each of the folds that k = 1 was the best number of neighbors. Then we found the average test error for each of the folds and found that the average test error is about .19.

```
## [1] 0.2000839
```



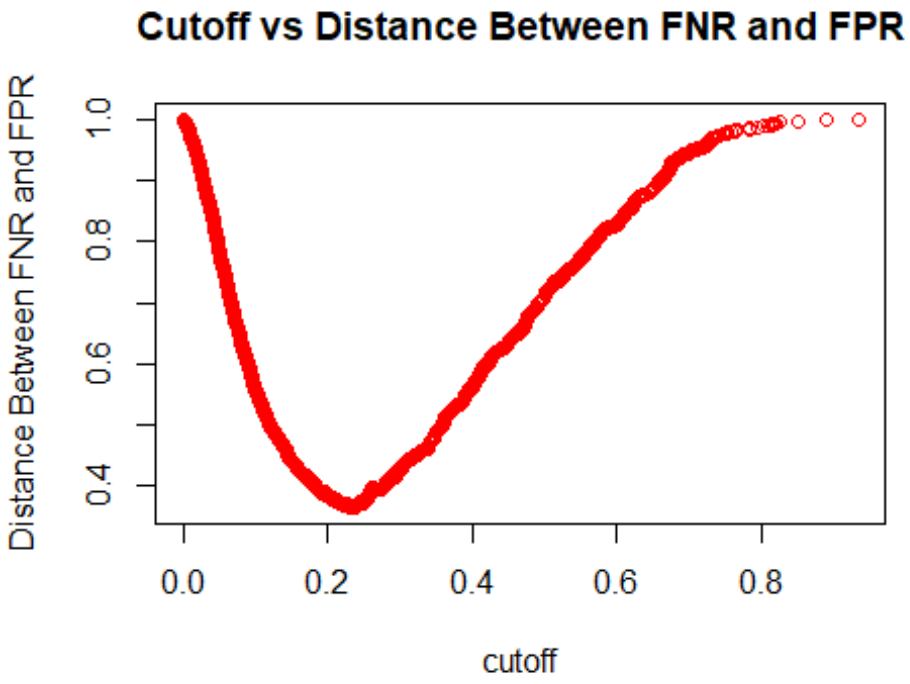
Logistic Regression

The second type of model we looked at was logistic regression. To build the logistic regression model, we found that all of the predictors were significant in predicting quality_class. Next, we then proceeded to find the best threshold value. To find the best threshold value for determining quality_class, we minimized the distance between the false-negative rate and the false positive rate for the training data. We found this value to be about .24.

```
## [1] 0.2369717
```

This was repeated through 10-fold cross-validation to find the average threshold value. Then once we found the optimal threshold value to use, we ran the model on the test set. Once we ran the model on the test set, we found the test error associated with the model. Once this was done, we performed 10-fold cross-validation to find the average error rate from the logistic regression model, which ended up being .26.

```
## [1] 0.2623572
```



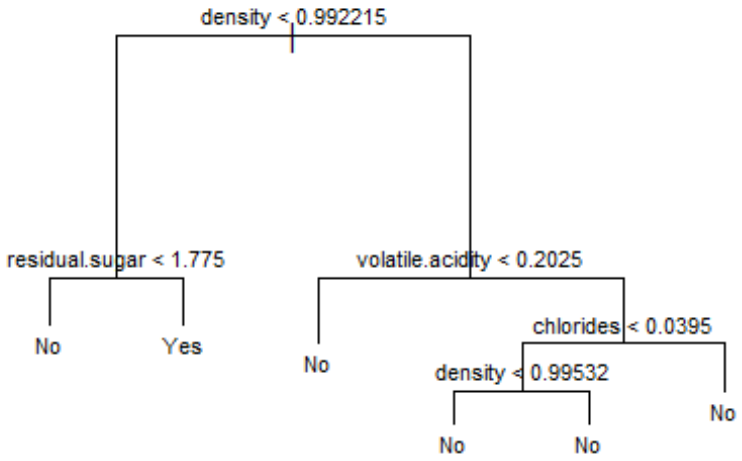
Tree

The third model that we considered was a tree model. To build the tree model, we first had to consider which variables to use for predicting quality_class. To do this, we considered the same variables as what we used for KNN. From here, we then built the initial tree model, then we pruned the model and found that the best model was one with 3 leaves. From here, we then used the model we built off of the training data to predict the test data. Then from here, we found the error rate from the tree model. Once this was done, we then performed 10-fold cross-validation to find the best number of leaves for each model and the average error rate for the models. We found that the best number of leaves for each model was 3, with one exception where one of the folds had the best number of leaves being 1.

```
## [1] 3
```

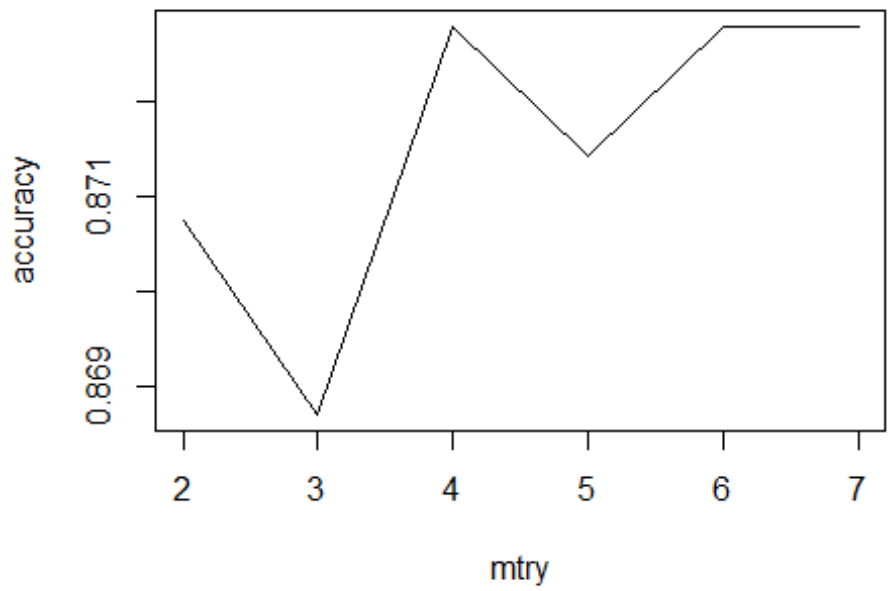
Then we found the average error rate for each of the folds which was .22.

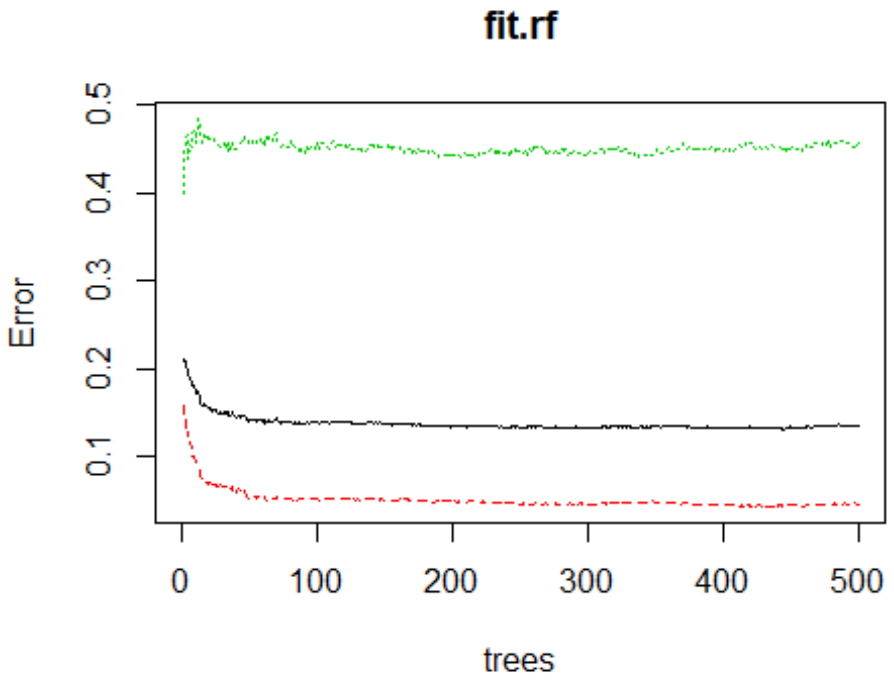
```
## [1] 0.2178557
```



Random Forest model

The fourth model that we decided to use is the Random Forest model. To build the random forest model, we first need to consider which predictor we are going to use for predicting the quality_class. We use the same variable as the previous method. Once the variable has been chosen, we start by using the 10-folds cross-validation to find the optimum entry number. We pick the one with the highest accuracy, which is 4 (might be different from a computer). Then we split the data of training set as 70% and testing set as 30% then by using the build-in random forest function we had obtained the accuracy of 87.68%, which compared to the previous method this method gives the error rate around 12%~13% (depends on the computer).





Model Evaluation

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    0    1
##          0 1109  137
##          1   44  179
##
##                Accuracy : 0.8768
##                  95% CI : (0.8589, 0.8932)
##     No Information Rate : 0.7849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5915
##
##  Mcnemar's Test P-Value : 8.014e-12
##
##             Sensitivity : 0.9618
##             Specificity : 0.5665
##          Pos Pred Value : 0.8900
##          Neg Pred Value : 0.8027
##              Prevalence : 0.7849
##          Detection Rate : 0.7549
##    Detection Prevalence : 0.8482
##       Balanced Accuracy : 0.7641
##
##        'Positive' Class : 0
##
```

Conclusion

Through our analysis of Portuguese white wine data, we found that we are able to classify if the wine being tested is about the median quality or not. We did this by using 4 different methods: KNN, logistic regression, tree, and random forest. Through analysis and tuning of these models using multiple training and test sets, we found that the random forest model was the best one. We found this model ended up having an accuracy of about 88%, with an error rate of about 12%.

The one limitation of this project is that we are not able to classify the quality of the wine directly, but rather we are only able to classify if the wine is above or below the median. From here there are multiple potential research directions. One, if there was more data collected we could more accurately classify the quality of the wine, as well as we could more accurately classify if the wine quality is above or below the median. Another research direction would be to classify and predict the alcohol content of the wine based on its quality, or rather classify and predict another variable that we used as a predictor.

Appendix

All of the code for the project

```
library(ISLR)
library(ggplot2)
library(class)
```

```r
library(plyr)
library(dplyr)
library(ROCR)
library(tree)
library(randomForest)
library(caret)
library(e1071)

#setting up the data and exploratory graphics
set.seed(420)
data = read.csv("winequality-white.csv", sep = ";")
summary(data)
#median(data$quality)
data$quality_class = ifelse(data$quality > 6, 1, 0)
mod_full = lm(quality_class ~ . -quality, data = data)
mod_end = step(mod_full, direction = "backward")
#summary(mod_end)
par(mfrow= c(3, 4))
plot(quality_class ~ ., data = data)
data = data[sample(nrow(data)),]
folds = cut(seq(1, nrow(data)), breaks = 10, labels = FALSE)

#knn
#finding the best number of neighbors using cross validation
allK = 1:25
error.folds = c()
error_vector = c()
for(i in 1:10){
  validation.error = NULL
  test_ind = which(folds == i)
  test_data = data[test_ind,]
  train_data = data[-test_ind,]
  knn_train_data = train_data %>% select(-quality_class, -quality, -
citric.acid, -total.sulfur.dioxide, -alcohol)
  knn_test_data = test_data %>% select(-quality_class, -quality, -
citric.acid, -total.sulfur.dioxide, -alcohol)
  for (j in allK){
    pred.Yval = knn.cv(train = knn_train_data, cl = train_data$quality_class,
k=j)
    validation.error = c(validation.error, mean(pred.Yval !=
train_data$quality_class))
  }
  knn_neighbor = which.min(validation.error)
  error.folds[i] = knn_neighbor
  #error_vector[i] = min(validation.error)
  knn_test = knn(train = knn_train_data, test = knn_test_data, cl =
train_data$quality_class, k = knn_neighbor)
  knn_conf_test = table(Pred = knn_test, Obs = test_data$quality_class)
  #finding test error
  error_vector[i] = 1 - sum(diag(knn_conf_test)/sum(knn_conf_test))
```

```
}
median(error.folds)
mean(error_vector)

#logistic regression
best = c()
error_rate = c()
for (i in 1:10) {
  #building initial log regression model
  test_ind = which(folds == i)
  test_data = data[test_ind,]
  train_data = data[-test_ind,]
  log_fit = glm(quality_class ~ fixed.acidity + volatile.acidity +
residual.sugar + chlorides + free.sulfur.dioxide +  density + pH + sulphates,
data = train_data, family = "binomial")
  #summary(log_fit)

  #calculating what is the ideal threshold value
  prob_training = predict(log_fit, type="response")
  pred = prediction(prob_training, train_data$quality_class)
  perf = performance(pred, measure="tpr", x.measure="fpr")
  fpr = performance(pred, "fpr")@y.values[[1]]
  cutoff = performance(pred, "fpr")@x.values[[1]]
  fnr = performance(pred,"fnr")@y.values[[1]]
  rate = as.data.frame(cbind(Cutoff=cutoff, FPR=fpr, FNR=fnr))
  rate$distance = sqrt((rate[,2])^2+(rate[,3])^2)
  index = which.min(rate$distance)
  #make into a vector
  best[i] = rate$Cutoff[index]

  #finding the test error
  log_test = predict(log_fit, test_data, type = 'response')
  outcome = ifelse(log_test >= best, 1, 0)
  tbl = table(pred=outcome, out=test_data$quality_class)
  conf_mat = confusionMatrix(tbl)
  #make into vector
  error_rate[i] = 1 - conf_mat$overall[1]
}
mean(best)
mean(error_rate)

#tree
best.prune = c()
error_rate = c()
for (i in 1:10) {
  #building initial tree model
  data$quality_yn = as.factor(ifelse(data$quality_class == 1, "Yes", "No"))
  test_ind = which(folds == i)
  test_data = data[test_ind,]
```

```
  train_data = data[-test_ind,]
  tree_model = tree(quality_yn ~ fixed.acidity + volatile.acidity +
residual.sugar + chlorides + free.sulfur.dioxide +  density + sulphates + pH,
data = train_data)
  #plot(tree_model)
  #text(tree_model, cex = .7)
  tree_pred = predict(tree_model, test_data, type = "class")
  error_tbl = table(tree_pred, test_data$quality_class)
  #error rate, can make this into a vector
  error_rate[i] = 1 - sum(diag(error_tbl))/sum(error_tbl)

  prune = prune.tree(tree_model, k = 0:20, method = "misclass")
  # Best size
  best.prune[i] = prune$size[which.min(prune$dev)]
  #best.prune
}

median(best.prune)
mean(error_rate)

#random forest
set.seed(1234)
data$quality_class = factor(ifelse(data$quality > 6, 1, 0))
x=data[,1:8]
y=data$quality_class
CVgroup <- function(k,datasize,seed){
  cvlist <- list()
  set.seed(seed)
  n <- rep(1:k,ceiling(datasize/k))[1:datasize]
  temp <- sample(n,datasize)
  x <- 1:k
  dataseq <- 1:datasize
  cvlist <- lapply(x,function(x) dataseq[temp==x])
  return(cvlist)
}
k <- 10
datasize <- nrow(data)
cvlist <- CVgroup(k = k,datasize = datasize,seed = 1234)
cvtest <- function(i,j){
  train <- data[-cvlist[[i]],]
  test <- data[cvlist[[i]],]
  model <- randomForest(train$quality_class~.,data = train[,1:8],mtry = j)
  pred <- predict(model,test)
  sum(diag(table(pred,test$quality_class)))/nrow(test)
}
pred <- mdply(cbind(rep(1:3,times=6),rep(2:7,each=3)),cvtest)
plot(aggregate(pred$V1~pred$X2,FUN=mean),type='l',xlab='mtry',ylab='accuracy'
)
```

```r
# by using 10-fold cv we obtain mtry = 4
set.seed(1234)
samp<-sample(1:nrow(data),0.3*nrow(data)) # train and test are redivided
train=data[-samp,]
test=data[samp,]
fit.rf<-randomForest(train$quality_class~.,data = train[,1:8],mtry = 4)
plot(fit.rf)
# Error is not decreasing as number of trees increasing, thus ntree is
suitable
pred<-predict(fit.rf,test)
confusionMatrix(pred,test$quality_class)
```