Classification - Neo Zhao - CS4375

Linear Models

- Logistic Regression uses a qualitative target variable to predict. In this project, I have found the ratings of Red and White wine. I will be setting all ratings > 3 to 1 and all ratings < 3 to 0. While there were about 32 observations that were exactly 3, we will omit them as it will not mess with the data too much out of 12,000+ observations. The Linear Model for classification will create a sort of barrier to seperate into different classes. In this project, Ratings > 3 and Ratings < 3 will be predicted into 2 different classes.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## Warning: package 'tibble' was built under R version 4.1.3
```

```
## Warning: package 'tidyr' was built under R version 4.1.3
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
## Warning: package 'purrr' was built under R version 4.1.3
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
## Warning: package 'forcats' was built under R version 4.1.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.1.3
```

```
library(mccr)
```

```
## Warning: package 'mccr' was built under R version 4.1.3
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.3
```

```
library(rpart)

# Source: https://www.kaggle.com/datasets/budnyak/wine-rating-and-price?select=Red.csv

# Red, White, Rose, and Sparkling wine are all from the same dataset; however, separated by type

# Red Total: 8666
Red <- read.csv("Red.csv")

# White Total: 3764
White <- read.csv("White.csv")

# Rose Total: 397
Rose <- read.csv("Rose.csv")

# Sparkling Total: 1007
Sparkling <- read.csv("Sparkling.csv")

# Combine the datasets together, Total: 13058
totalWine <- rbind(data = Red, data = White, data = Rose, data = Sparkling)

# Rename ï..Name to just Name
names(totalWine)[1] <- "Name"

# Omit Names, Winery, & Region Column
totalWine <- subset(totalWine, select = -c(Name, Winery, Region))
```

```r
# Omit all records where Year = N.V.
totalWine <- subset(totalWine, totalWine$Year != "N.V.")

# Omit all records where Rating = 3, Total: 12398
totalWine <- subset(totalWine, totalWine$Rating != 3)

# Omit all records before 2000s
totalWine <- subset(totalWine, totalWine$Year >= 2000)

# Replace ratings with 1 if Rating > 3 and replace with 0 if Rating < 3
totalWine$Rating[totalWine$Rating <= 3] <- 0
totalWine$Rating[totalWine$Rating > 3] <- 1

# Reorder Columns
totalWine <- totalWine[,c(1,2,3,5,4)]
```

```r
# Split to 80/20 Train/Test
set.seed(512)
i <- sample(1 : nrow(totalWine), round(nrow(totalWine) * 0.8), replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]
```

Data Exploration

```r
# 1) summary()
summary(train)
```

```
##    Country              Rating       NumberOfRatings       Year
##  Length:10401       Min.   :0.0000   Min.   :   25.0   Length:10401
##  Class :character   1st Qu.:1.0000   1st Qu.:   55.0   Class :character
##  Mode  :character   Median :1.0000   Median :  123.0   Mode  :character
##                     Mean   :0.9982   Mean   :  333.3
##                     3rd Qu.:1.0000   3rd Qu.:  315.0
##                     Max.   :1.0000   Max.   :20293.0
##      Price
##  Min.   :   3.55
##  1st Qu.:   9.90
##  Median :  15.90
##  Mean   :  32.37
##  3rd Qu.:  31.99
##  Max.   :1599.95
```

```r
# 2) is.na()
colSums(is.na(train))
```

```
##         Country          Rating NumberOfRatings            Year           Price
##               0               0               0               0               0
```

```r
# 3) str()
str(train)
```

```
## 'data.frame':    10401 obs. of  5 variables:
##  $ Country        : chr  "Austria" "Italy" "Italy" "France" ...
##  $ Rating         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ NumberOfRatings: num  30 28 375 40 290 42 791 110 193 282 ...
##  $ Year           : chr  "2017" "2018" "2018" "2017" ...
##  $ Price          : num  13.2 12.2 14.4 9.5 69.4 ...
```

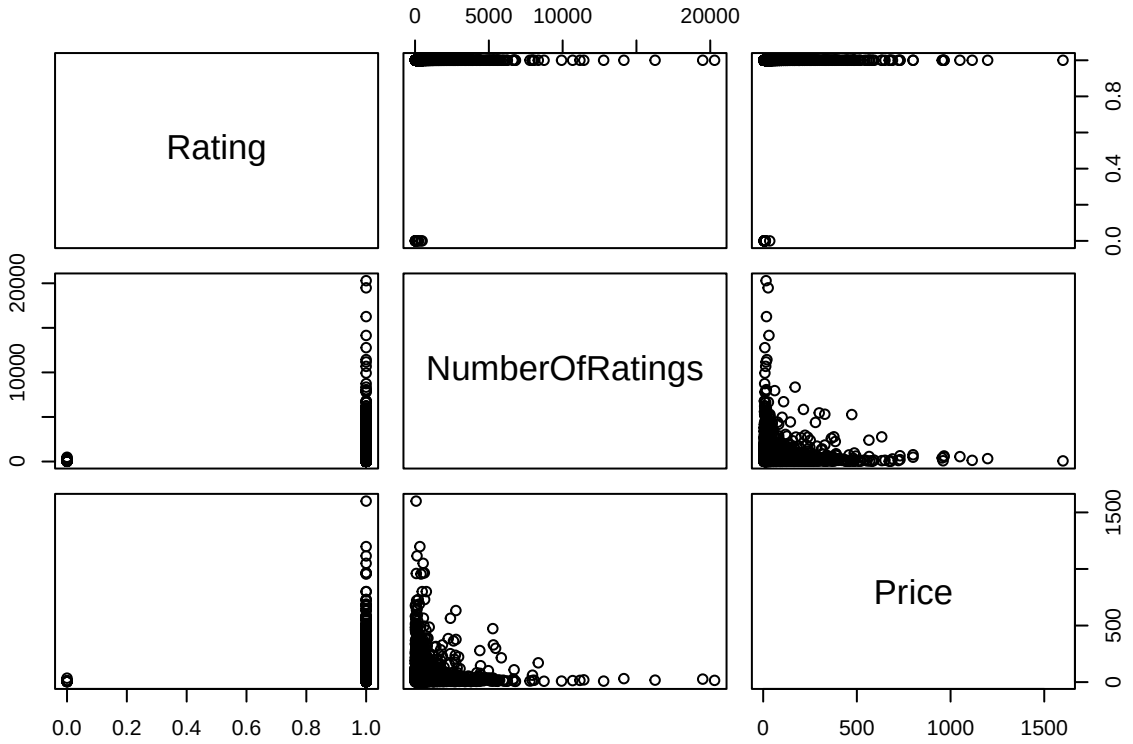
```r
# 4) head() functions
head(train)
```

```
##            Country Rating NumberOfRatings Year Price
## data.28810 Austria      1              30 2017 13.24
## data.19110   Italy      1              28 2018 12.18
## data.33841   Italy      1             375 2018 14.45
## data.84110  France      1              40 2017  9.50
## data.53111  France      1             290 2008 69.36
## data.5227  Germany      1              42 2017 22.00
```

```r
# 5) cor() and pairs()
cor(train[,c(-1, -4)])
```

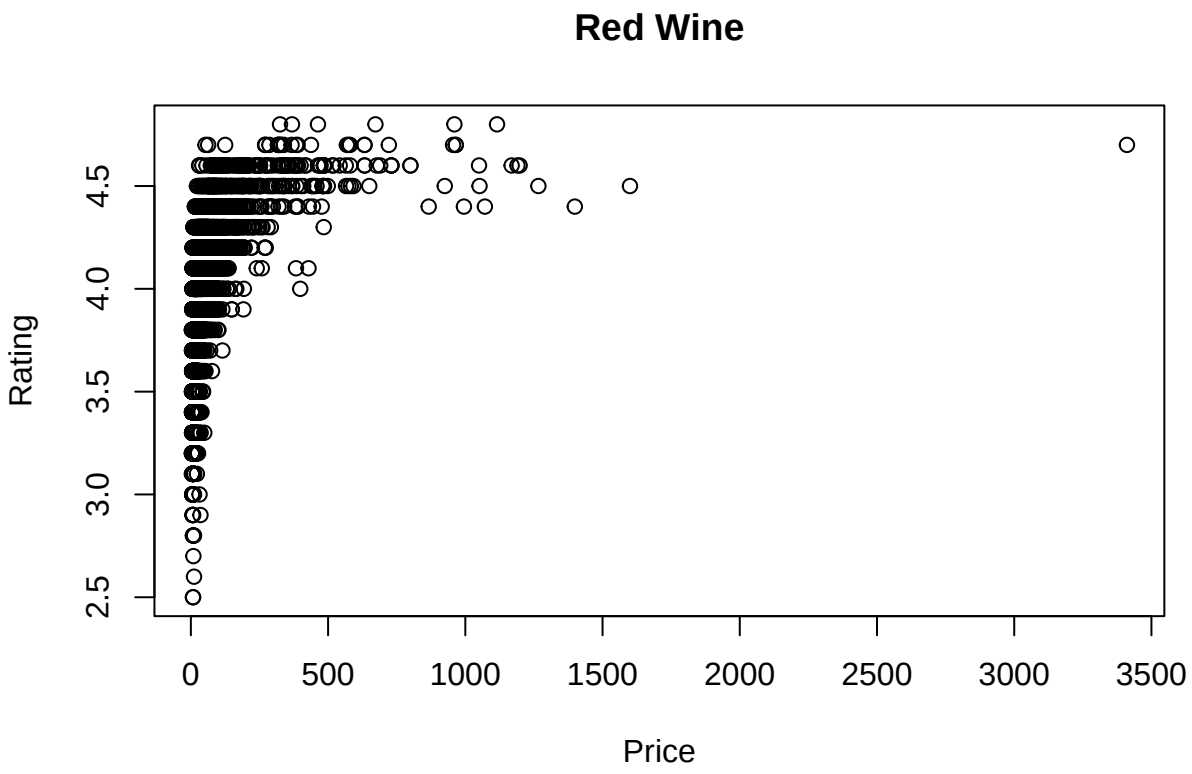
```
##                      Rating NumberOfRatings      Price
## Rating          1.00000000      0.01301710 0.01641215
## NumberOfRatings 0.01301710      1.00000000 0.03158961
## Price           0.01641215      0.03158961 1.00000000
```

```r
pairs(train[,c(-1, -4)])
```

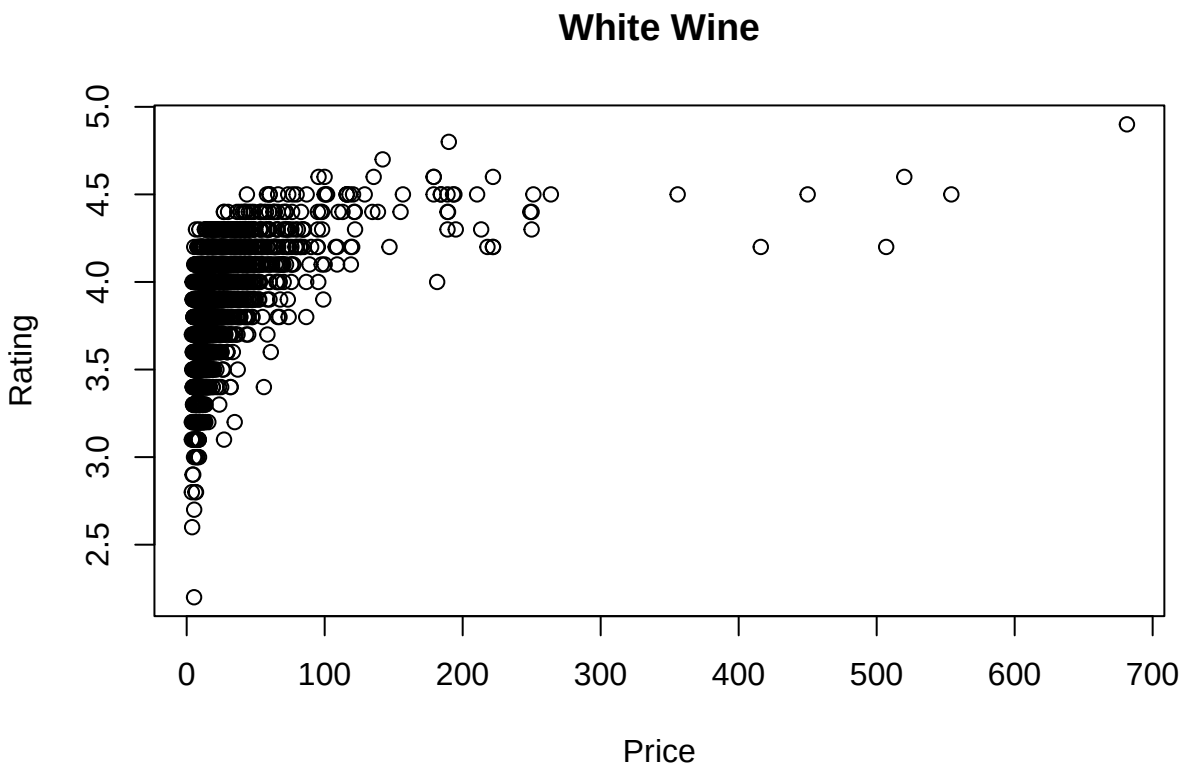


Informative Graphs

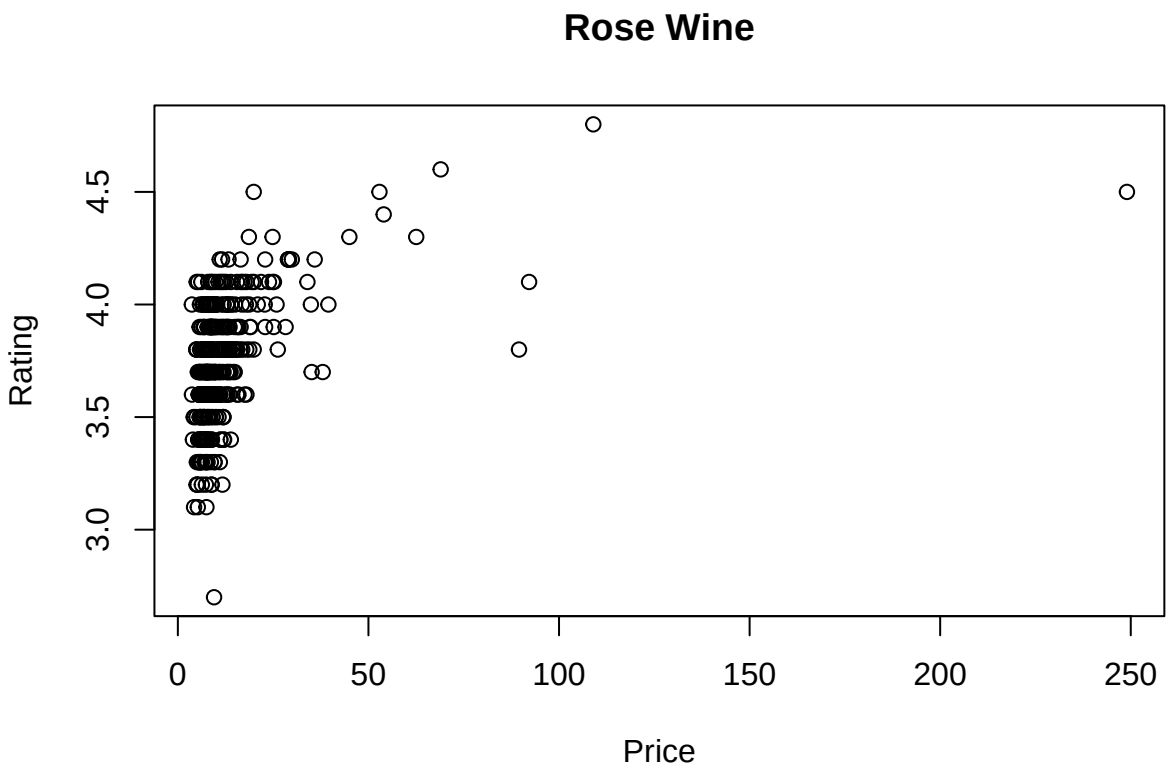
```
# Red
plot(Rating ~ Price, data = Red, main = "Red Wine", xlab = "Price", ylab = "Rating")
```



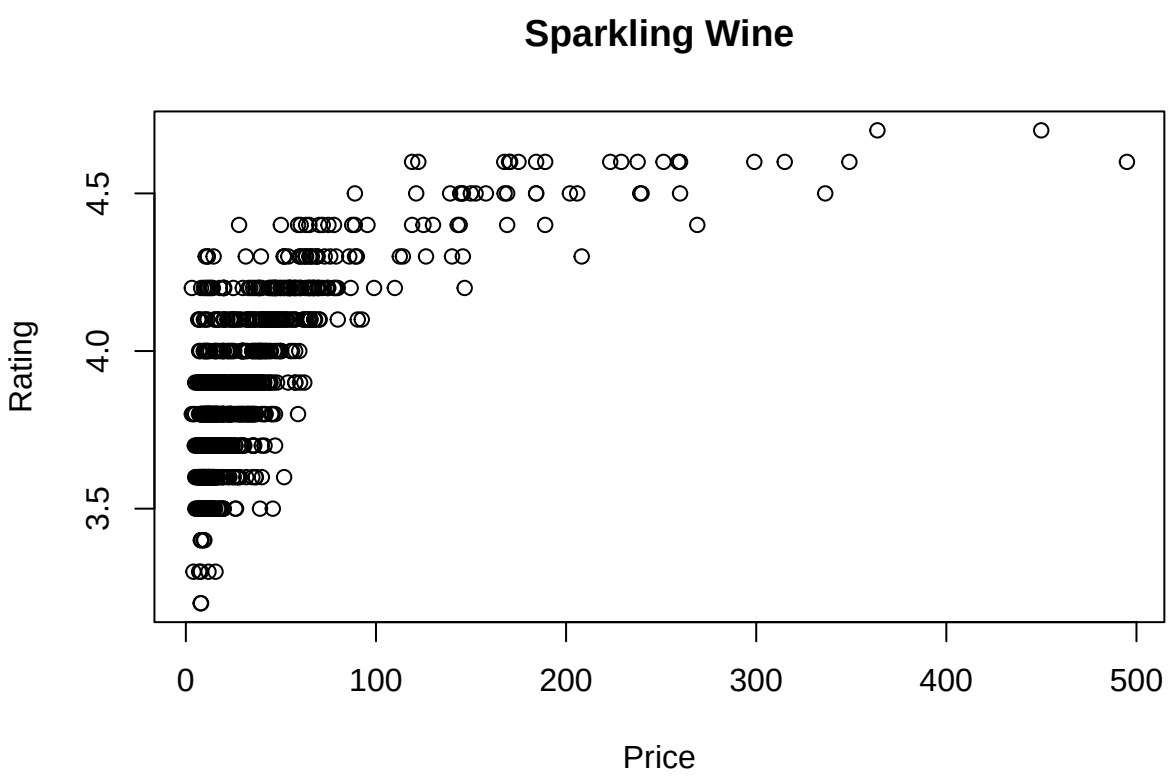

```
# White
plot(Rating ~ Price, data = White, main = "White Wine", xlab = "Price", ylab = "Rating")
```




```
# Rose
plot(Rating ~ Price, data = Rose, main = "Rose Wine", xlab = "Price", ylab = "Rating")
```




```
# Sparkling
plot(Rating ~ Price, data = Sparkling, main = "Sparkling Wine", xlab = "Price", ylab = "Rating")
```



Linear Regression

- The correlation for the Linear Model was 28%, which is not the best. Can kNN do better?

```
# Split to 80/20 Train/Test
set.seed(512)
i <- sample(1 : nrow(totalWine), round(nrow(totalWine) * 0.8), replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]
```

```
# Linear Regression
lm1 <- lm(Price ~ ., data = train)
summary(lm1)
```

```
##
## Call:
## lm(formula = Price ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -140.66  -16.83   -4.85    3.82 1485.31
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)            1.537e+02  1.929e+01   7.968 1.78e-15 ***
## CountryAustralia       1.431e+01  5.051e+00   2.834 0.004612 **
## CountryAustria         4.796e-01  4.611e+00   0.104 0.917163
## CountryBrazil         -2.973e+01  1.010e+01  -2.944 0.003242 **
## CountryBulgaria       -1.873e+01  3.938e+01  -0.476 0.634246
## CountryCanada         -1.113e+01  5.557e+01  -0.200 0.841312
## CountryChile          -2.561e+00  4.649e+00  -0.551 0.581798
## CountryChina           1.378e+01  3.249e+01   0.424 0.671380
## CountryCroatia        -3.666e+00  2.796e+01  -0.131 0.895701
## CountryCzech Republic  8.670e+00  3.938e+01   0.220 0.825741
## CountryFrance          1.592e+01  3.791e+00   4.199 2.70e-05 ***
## CountryGeorgia        -6.545e+00  1.792e+01  -0.365 0.714856
## CountryGermany         1.974e+00  4.037e+00   0.489 0.624886
## CountryGreece         -1.353e+01  1.323e+01  -1.023 0.306345
## CountryHungary        -1.249e+01  1.477e+01  -0.846 0.397714
## CountryIsrael          7.564e-01  1.322e+01   0.057 0.954378
## CountryItaly           3.099e+00  3.738e+00   0.829 0.407097
## CountryLebanon        -5.349e+01  1.744e+01  -3.068 0.002160 **
## CountryLuxembourg     -5.893e+00  2.796e+01  -0.211 0.833066
## CountryMexico         -4.476e+01  5.561e+01  -0.805 0.420822
## CountryMoldova         1.077e+01  1.790e+01   0.602 0.547292
## CountryNew Zealand     3.462e+00  6.086e+00   0.569 0.569550
## CountryPortugal       -9.279e+00  5.013e+00  -1.851 0.064210 .
## CountryRomania        -1.422e+01  1.076e+01  -1.322 0.186269
## CountrySlovakia       -4.192e+00  3.938e+01  -0.106 0.915231
## CountrySlovenia       -1.277e+01  1.641e+01  -0.778 0.436482
## CountrySouth Africa   -1.904e+00  4.182e+00  -0.455 0.648910
## CountrySpain          -6.159e+00  3.928e+00  -1.568 0.116915
## CountrySwitzerland     1.316e+01  1.323e+01   0.995 0.319887
## CountryTurkey         -5.833e+00  1.884e+01  -0.310 0.756865
## CountryUnited Kingdom  2.399e+01  3.224e+01   0.744 0.456732
## CountryUnited States   1.558e+01  4.487e+00   3.472 0.000518 ***
## CountryUruguay         9.324e-02  3.222e+01   0.003 0.997691
## Rating                 1.132e+01  1.289e+01   0.878 0.380085
## NumberOfRatings        1.779e-04  7.561e-04   0.235 0.814017
## Year2001              -1.033e+02  2.313e+01  -4.468 7.98e-06 ***
## Year2002              -1.322e+01  2.516e+01  -0.525 0.599292
## Year2003              -4.910e+01  2.118e+01  -2.319 0.020434 *
## Year2004              -4.823e+01  1.738e+01  -2.775 0.005532 **
## Year2005              -6.626e+01  1.480e+01  -4.476 7.69e-06 ***
## Year2006              -3.309e+01  1.608e+01  -2.058 0.039656 *
## Year2007              -8.354e+01  1.608e+01  -5.194 2.10e-07 ***
## Year2008              -7.517e+01  1.522e+01  -4.940 7.94e-07 ***
## Year2009              -6.452e+01  1.531e+01  -4.214 2.53e-05 ***
## Year2010              -6.635e+01  1.459e+01  -4.546 5.53e-06 ***
## Year2011              -1.106e+02  1.432e+01  -7.721 1.26e-14 ***
## Year2012              -1.184e+02  1.422e+01  -8.322  < 2e-16 ***
## Year2013              -1.118e+02  1.414e+01  -7.911 2.81e-15 ***
## Year2014              -1.302e+02  1.406e+01  -9.259  < 2e-16 ***
## Year2015              -1.335e+02  1.399e+01  -9.540  < 2e-16 ***
## Year2016              -1.390e+02  1.397e+01  -9.947  < 2e-16 ***
## Year2017              -1.484e+02  1.398e+01 -10.620  < 2e-16 ***
## Year2018              -1.560e+02  1.397e+01 -11.170  < 2e-16 ***
## Year2019              -1.577e+02  1.408e+01 -11.203  < 2e-16 ***
```

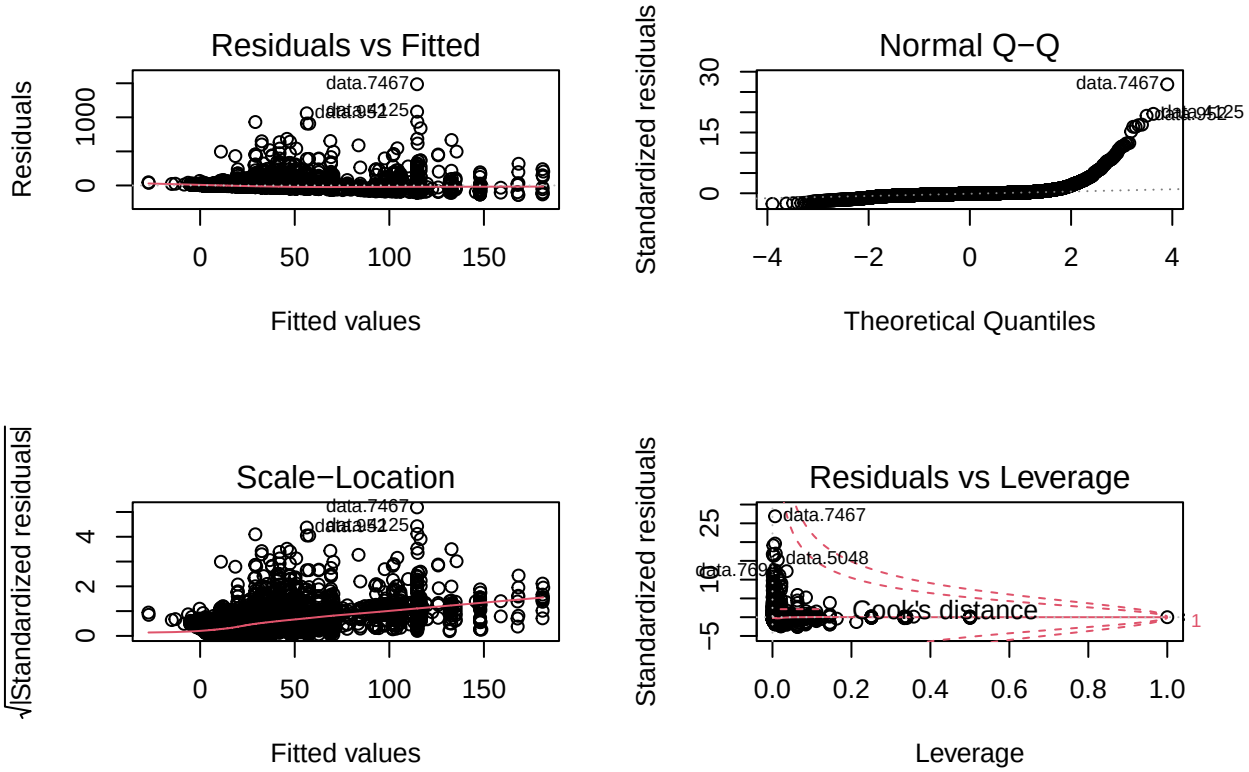
```
## Year2020              -1.557e+02  4.166e+01  -3.738 0.000186 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.44 on 10346 degrees of freedom
## Multiple R-squared:  0.1758, Adjusted R-squared:  0.1715
## F-statistic: 40.86 on 54 and 10346 DF,  p-value: < 2.2e-16
```

```
# Plotting Residuals
par(mfrow = c(2,2))
plot(lm1)
```

```
## Warning: not plotting observations with leverage one:
##   2844

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```




```
pred1 <- predict(lm1, newdata = test)
cor1 <- cor(pred1, test$Price)
mse1 <- mean((pred1 - test$Price) ^ 2)
rmse1 <- sqrt(mse1)

print(paste ("Cor = ", cor1))
```

```
## [1] "Cor =  0.286075085867337"
```

```
print(paste ("MSE = ", mse1))
```

```
## [1] "MSE =  7411.42374513563"
```

kNN Clustering - Regression

- The correlation for the knn Regression was 18%, which is lower than Linear Regression. The MSE is slightly higher than the Linear Regression; however, they are both very very high. Let's scale the data to see if it does any better.
- The scaled correlation is 16%, which is only slightly higher and does not compete with the Linear Regression correlation. The MSE also is higher by 200.

```
# Linear Model with all Predictors
lm1 <- lm(Price ~. , data = totalWine)
summary(lm1)
```

```
##
## Call:
## lm(formula = Price ~ ., data = totalWine)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -126.3  -17.4   -4.7    4.1 3339.8
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1.343e+02  2.012e+01   6.678 2.52e-11 ***
## CountryAustralia        1.220e+01  5.068e+00   2.407 0.016091 *
## CountryAustria         -1.263e-01  4.625e+00  -0.027 0.978224
## CountryBrazil          -3.170e+01  1.018e+01  -3.113 0.001859 **
## CountryBulgaria        -1.867e+01  4.454e+01  -0.419 0.675105
## CountryCanada          -8.088e+00  3.643e+01  -0.222 0.824297
## CountryChile           -2.781e+00  4.706e+00  -0.591 0.554529
## CountryChina            1.363e+01  3.669e+01   0.372 0.710215
## CountryCroatia         -7.545e-01  2.589e+01  -0.029 0.976748
## CountryCzech Republic   8.460e+00  4.454e+01   0.190 0.849366
## CountryFrance           1.664e+01  3.809e+00   4.370 1.25e-05 ***
## CountryGeorgia         -9.457e+00  1.780e+01  -0.531 0.595225
## CountryGermany          8.223e-01  4.065e+00   0.202 0.839686
## CountryGreece          -1.467e+01  1.359e+01  -1.080 0.280170
## CountryHungary         -1.133e+01  1.485e+01  -0.763 0.445731
## CountryIsrael           4.673e-01  1.358e+01   0.034 0.972551
## CountryItaly            2.271e+00  3.753e+00   0.605 0.545145
## CountryLebanon         -4.613e+01  1.806e+01  -2.554 0.010663 *
## CountryLuxembourg      -6.139e+00  2.831e+01  -0.217 0.828314
## CountryMexico          -4.446e+01  6.292e+01  -0.707 0.479799
## CountryMoldova          8.904e+00  1.778e+01   0.501 0.616533
## CountryNew Zealand      2.225e+00  6.077e+00   0.366 0.714200
## CountryPortugal        -9.243e+00  5.013e+00  -1.844 0.065254 .
## CountryRomania         -1.291e+01  1.094e+01  -1.180 0.238146
```

```
## CountrySlovakia       -4.423e+00  4.454e+01  -0.099 0.920899
## CountrySlovenia       -1.115e+01  1.611e+01  -0.692 0.489010
## CountrySouth Africa   -2.057e+00  4.213e+00  -0.488 0.625335
## CountrySpain          -7.616e+00  3.957e+00  -1.925 0.054273 .
## CountrySwitzerland     1.173e+01  1.359e+01   0.863 0.388047
## CountryTurkey         -6.051e+00  2.019e+01  -0.300 0.764373
## CountryUnited Kingdom  2.291e+01  3.645e+01   0.629 0.529662
## CountryUnited States   1.724e+01  4.525e+00   3.810 0.000140 ***
## CountryUruguay        -4.929e+00  2.831e+01  -0.174 0.861773
## Rating                 1.169e+01  1.353e+01   0.864 0.387773
## NumberOfRatings       -4.241e-05  7.729e-04  -0.055 0.956245
## Year2001              -7.797e+01  2.315e+01  -3.368 0.000760 ***
## Year2002               8.136e+00  2.379e+01   0.342 0.732404
## Year2003              -4.472e+01  2.168e+01  -2.063 0.039171 *
## Year2004              -2.270e+01  1.798e+01  -1.262 0.206858
## Year2005              -5.360e+01  1.526e+01  -3.513 0.000445 ***
## Year2006              -2.156e+01  1.643e+01  -1.313 0.189300
## Year2007              -6.625e+01  1.655e+01  -4.002 6.31e-05 ***
## Year2008              -6.175e+01  1.571e+01  -3.930 8.55e-05 ***
## Year2009              -4.727e+01  1.575e+01  -3.001 0.002698 **
## Year2010              -4.261e+01  1.513e+01  -2.816 0.004871 **
## Year2011              -9.091e+01  1.486e+01  -6.116 9.86e-10 ***
## Year2012              -9.166e+01  1.476e+01  -6.211 5.42e-10 ***
## Year2013              -9.287e+01  1.466e+01  -6.334 2.46e-10 ***
## Year2014              -1.095e+02  1.459e+01  -7.508 6.42e-14 ***
## Year2015              -1.144e+02  1.452e+01  -7.876 3.64e-15 ***
## Year2016              -1.200e+02  1.450e+01  -8.275  < 2e-16 ***
## Year2017              -1.290e+02  1.450e+01  -8.895  < 2e-16 ***
## Year2018              -1.364e+02  1.450e+01  -9.410  < 2e-16 ***
## Year2019              -1.382e+02  1.461e+01  -9.460  < 2e-16 ***
## Year2020              -1.365e+02  4.673e+01  -2.922 0.003484 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 62.77 on 12946 degrees of freedom
## Multiple R-squared:  0.1437, Adjusted R-squared:  0.1402
## F-statistic: 40.25 on 54 and 12946 DF,  p-value: < 2.2e-16
```

```
# kNN Regression
train$Year <- as.numeric(train$Year)
test$Year <- as.numeric(test$Year)

fit <- knnreg(train[,2:4], train[,5], k = 3)
predictions <- predict(fit, test[,2:4])
cor_knn1 <- cor(predictions, test$Price)
mse_knn1 <- mean((predictions - test$Price) ^ 2)

print(paste ("Cor = ", cor_knn1))
```

```
## [1] "Cor =  0.144326965892682"
```

```
print(paste ("MSE = ", mse_knn1))
```

```
## [1] "MSE =  8339.00445431074"
```

```r
# Scaled Data
trainScaled <- train[,2:4]
means <- sapply(trainScaled, mean)
stdvs <- sapply(trainScaled, sd)
trainScaled <- scale(trainScaled, center = means, scale = stdvs)
testScaled <- scale(test[,2:4], center = means, scale = stdvs)

fit <- knnreg(trainScaled, train$Price, k = 3)
predictions <- predict(fit, testScaled)
cor_knn2 <- cor(predictions, test$Price)
mse_knn2 <- mean((predictions - test$Price) ^ 2)

print(paste ("Cor = ", cor_knn2))
```

```
## [1] "Cor =  0.156600727004581"
```

```r
print(paste ("MSE = ", mse_knn2))
```

```
## [1] "MSE =  8540.41943844092"
```

```r
# Find the best k
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for(k in seq(1, 39, 2)) {
  fit_k <- knnreg(trainScaled, train$Price, k = k)
  pred_k <- predict(fit_k, testScaled)
  cor_k[i] <- cor(pred_k, test$Price)
  mse_k[i] <- mean((pred_k - test$Price) ^ 2)
  print(paste("k =", k, cor_k[i], mse_k[i]))
  i <- i + 1
  }
```

```
## [1] "k = 1 0.14971547828315 8733.85481517744"
## [1] "k = 3 0.156600727004581 8540.41943844092"
## [1] "k = 5 0.19254514508324 8137.8273836492"
## [1] "k = 7 0.206961108293334 7964.02968262835"
## [1] "k = 9 0.216169216962526 7837.23398280285"
## [1] "k = 11 0.219298917207792 7795.90876325253"
## [1] "k = 13 0.224040872744051 7754.28636155107"
## [1] "k = 15 0.237991093562273 7672.41063668214"
## [1] "k = 17 0.239263217866369 7655.56569456753"
## [1] "k = 19 0.251763394961229 7585.52035848211"
## [1] "k = 21 0.260803404112644 7536.12620291362"
## [1] "k = 23 0.262369864981181 7525.35858373136"
## [1] "k = 25 0.257582541852901 7546.46687939518"
## [1] "k = 27 0.258330064134136 7541.17142022897"
## [1] "k = 29 0.257741981953208 7543.12787555399"
## [1] "k = 31 0.261665637884775 7524.07476665214"
## [1] "k = 33 0.261581645953466 7524.08871709695"
## [1] "k = 35 0.26154737945751 7524.11340620002"
## [1] "k = 37 0.262004107354673 7521.51313316421"
## [1] "k = 39 0.265673421632523 7504.17715569421"
```

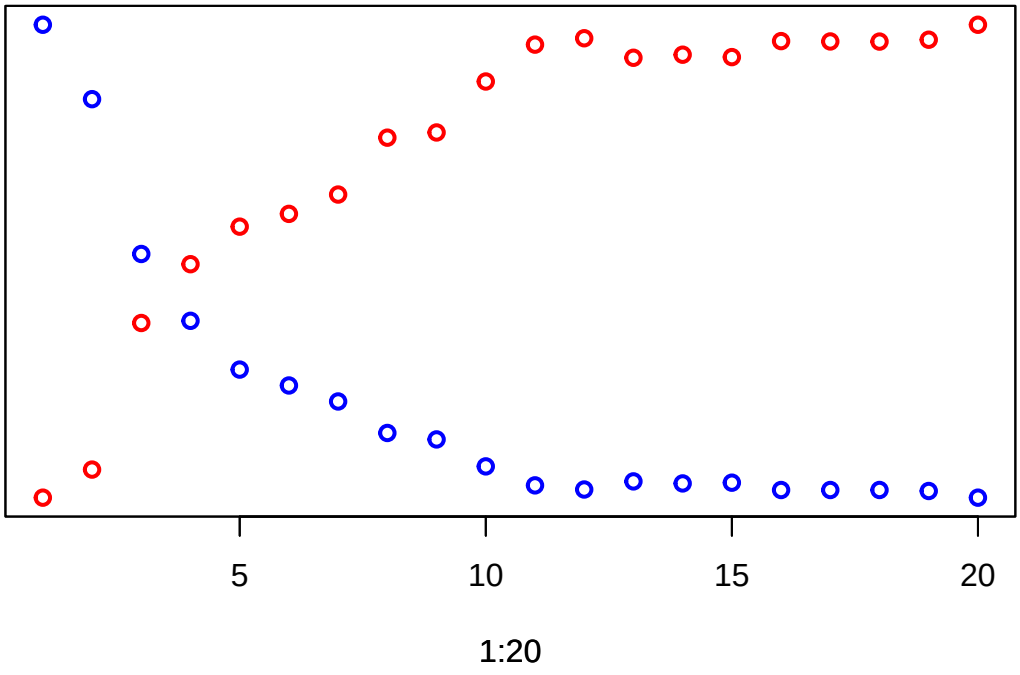
```r
plot(1:20, cor_k, lwd = 2, col = 'red', ylab = "", yaxt = 'n')
par(new = TRUE)
plot(1:20, mse_k, lwd = 2, col = 'blue', labels = FALSE, ylab = "", yaxt = 'n')
```

```
## Warning in plot.window(...): "labels" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "labels" is not a graphical parameter

## Warning in box(...): "labels" is not a graphical parameter

## Warning in title(...): "labels" is not a graphical parameter
```




```r
# Let's Pick a different k = 39

fit_39 <- knnreg(trainScaled, train$Price, k = 39)
predictions_39 <- predict(fit_39, testScaled)
cor_knn39 <- cor(predictions_39, test$Price)
mse_knn39 <- mean((predictions_39 - test$Price) ^ 2)

print(paste ("Cor = ", cor_knn39))
```

```
## [1] "Cor =  0.265673421632523"
```

```
print(paste ("MSE = ", mse_knn39))
```

```
## [1] "MSE =  7504.17715569421"
```

Decision Trees

- My last code block makes another tree. The accuracy is an insane 99.8%!

```
# Split to 80/20 Train/Test again just for fun I guess
set.seed(1002)
i <- sample(1 : nrow(totalWine), round(nrow(totalWine) * 0.8), replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]
```

```
# Using Tree
tree1 <- tree(Price ~ ., data = train, method = "class")
```

```
## Warning in tree(Price ~ ., data = train, method = "class"): NAs introduced by
## coercion
```

```
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = Price ~ ., data = train, method = "class")
## Variables actually used in tree construction:
## [1] "Year"            "NumberOfRatings"
## Number of terminal nodes:  5
## Residual mean deviance:  4054 = 42140000 / 10400
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -137.900  -17.430   -6.694    0.000    1.108 3349.000
```

```
# Plotting Tree pt. 1
plot(tree1, uniform = TRUE, margin = 0.2)
```

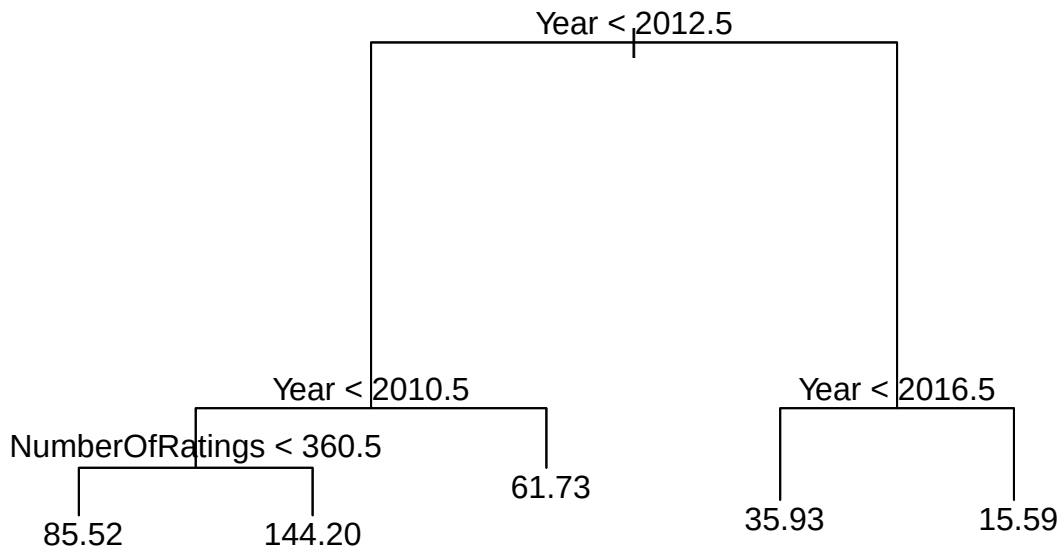
```
## Warning in text.default(x[1L], y[1L], "|", ...): "uniform" is not a graphical
## parameter
```

```
## Warning in text.default(x[1L], y[1L], "|", ...): "margin" is not a graphical
## parameter
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "uniform" is not a
## graphical parameter
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "margin" is not a
## graphical parameter
```

```
text(tree1)
```




```
# Hehe Tree Pt. 2
tree2 <- tree(Rating ~ ., data = train)
```

```
## Warning in tree(Rating ~ ., data = train): NAs introduced by coercion
```

```
tree2
```

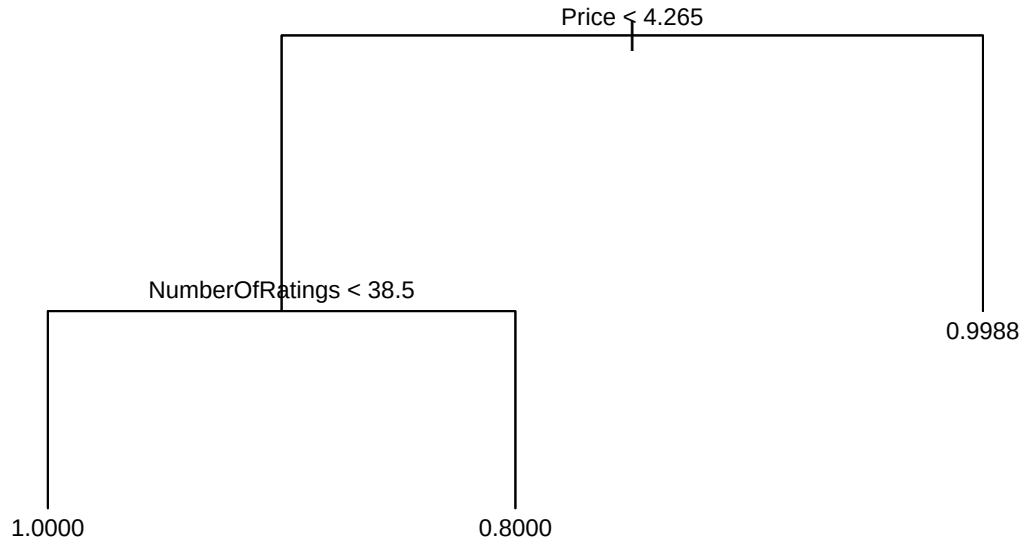
```
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 10401 13.980 0.9987
##   2) Price < 4.265 17  1.765 0.8824
##     4) NumberOfRatings < 38.5 7  0.000 1.0000 *
##     5) NumberOfRatings > 38.5 10  1.600 0.8000 *
##   3) Price > 4.265 10384 11.990 0.9988 *
```

```
summary(tree2)
```

```
##
## Regression tree:
## tree(formula = Rating ~ ., data = train)
```

```
## Variables actually used in tree construction:
## [1] "Price"           "NumberOfRatings"
## Number of terminal nodes:  3
## Residual mean deviance:  0.001307 = 13.59 / 10400
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.998800  0.001156  0.001156  0.000000  0.001156  0.200000
```

```
# Plotting Tree pt. 2
plot(tree2)
text(tree2, cex = 0.75, pretty = 0)
```




```
set.seed(4375)
i <- sample(150, 100, replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]
tree3 <- tree(Rating ~ ., data = train)
```

```
## Warning in tree(Rating ~ ., data = train): NAs introduced by coercion
```

```
pred <- predict(tree3, newdata = test)
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
table(pred, test$Rating)
```

```
##
## pred     0     1
##    1    22 12879
```

```
mean(pred == test$Rating)
```

```
## [1] 0.9982947
```