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
```

```r
library(mccr)
```

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

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

# Red and White wine from the same dataset; however, separated by type
# Red Total: 8666, White Total: 3764
Red <- read.csv("Red.csv")
White <- read.csv("White.csv")

# Combine the datasets together, Total: 12430
totalWine <- rbind(data = Red, data = White)
totWine <- rbind(data = Red, data = White)

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

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

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



# 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
```

A. Divide into 80/20 train/test

```r
set.seed(1)

i <- sample(1:nrow(totalWine), nrow(totalWine) * 0.8, replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]
```

B. Data Exploration

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

```
##    Country              Rating       NumberOfRatings       Price
##  Length:9918        Min.   :0.0000   Min.   :   25.0   Min.   :   3.70
##  Class :character   1st Qu.:1.0000   1st Qu.:   56.0   1st Qu.:   9.95
##  Mode  :character   Median :1.0000   Median :  124.0   Median :  16.39
##                     Mean   :0.9982   Mean   :  343.5   Mean   :  33.87
```

```
##                      3rd Qu.:1.0000   3rd Qu.:  319.0   3rd Qu.:  32.87
##                      Max.   :1.0000   Max.   :20293.0   Max.   :3410.79
##      Year
##  Length:9918
##  Class :character
##  Mode  :character
##
##
##
```

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

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

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

```
## 'data.frame':    9918 obs. of  5 variables:
##  $ Country        : chr  "France" "Spain" "Italy" "France" ...
##  $ Rating         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ NumberOfRatings: num  135 84 38 70 47 79 95 32 196 680 ...
##  $ Price          : num  23.3 11.6 17.4 59.9 47 ...
##  $ Year           : chr  "2017" "2015" "2006" "2017" ...
```

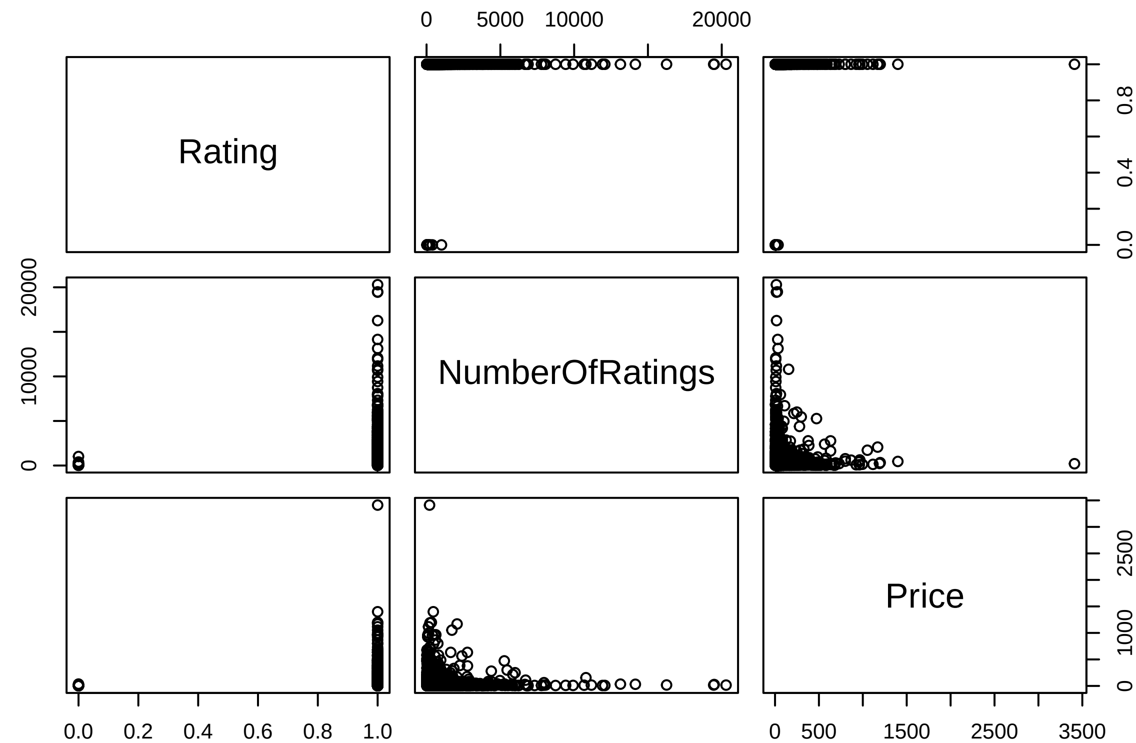
```r
# 4) head() functions
head(train)
```

```
##            Country Rating NumberOfRatings Price Year
## data.1018   France      1             135 23.29 2017
## data.8031    Spain      1              84 11.63 2015
## data.4787    Italy      1              38 17.45 2006
## data.17351  France      1              70 59.90 2017
## data.10911  France      1              47 46.95 2006
## data.19051 Austria      1              79 42.50 2018
```

```r
# 5) cor() and pairs()
cor(train[,c(2:4)])
```

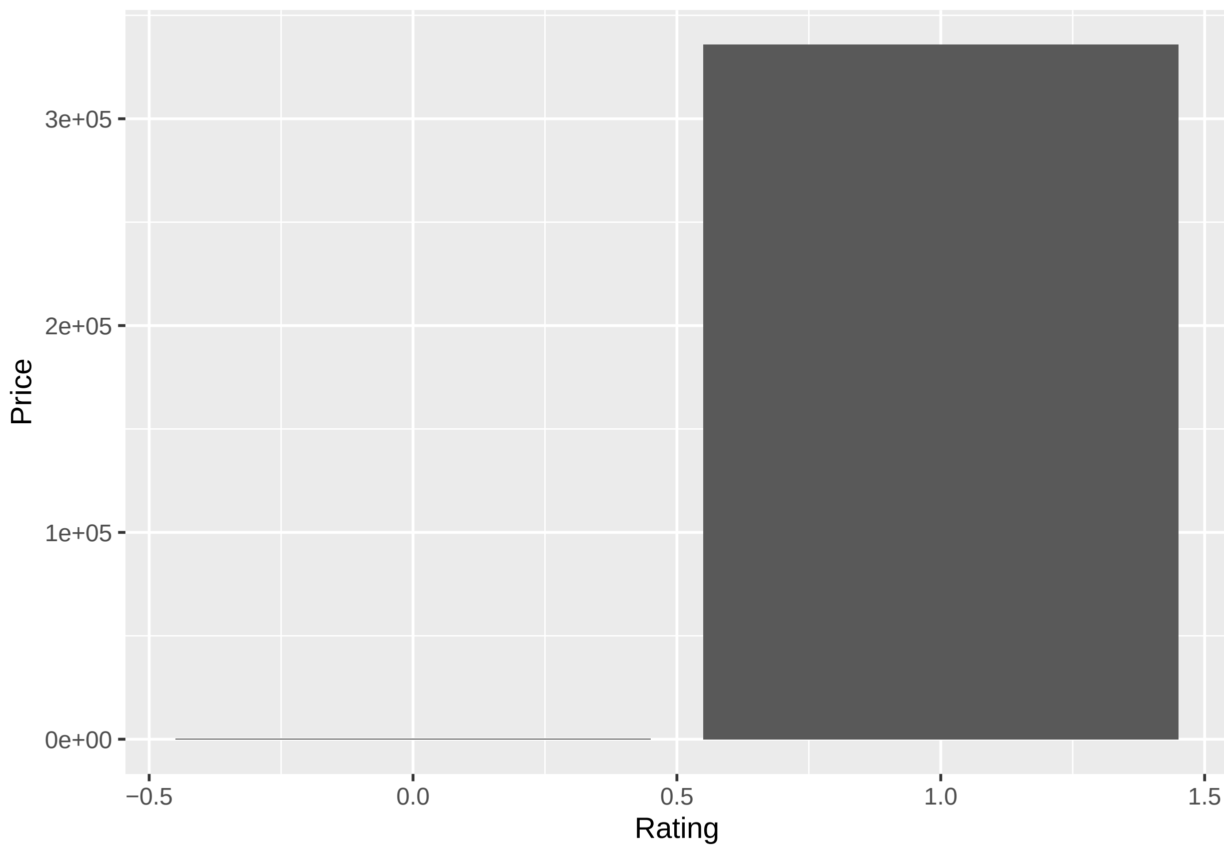
```
##                     Rating NumberOfRatings      Price
## Rating          1.00000000      0.01061826 0.01445800
## NumberOfRatings 0.01061826      1.00000000 0.02235485
## Price           0.01445800      0.02235485 1.00000000
```

```r
pairs(train[,c(2:4)])
```

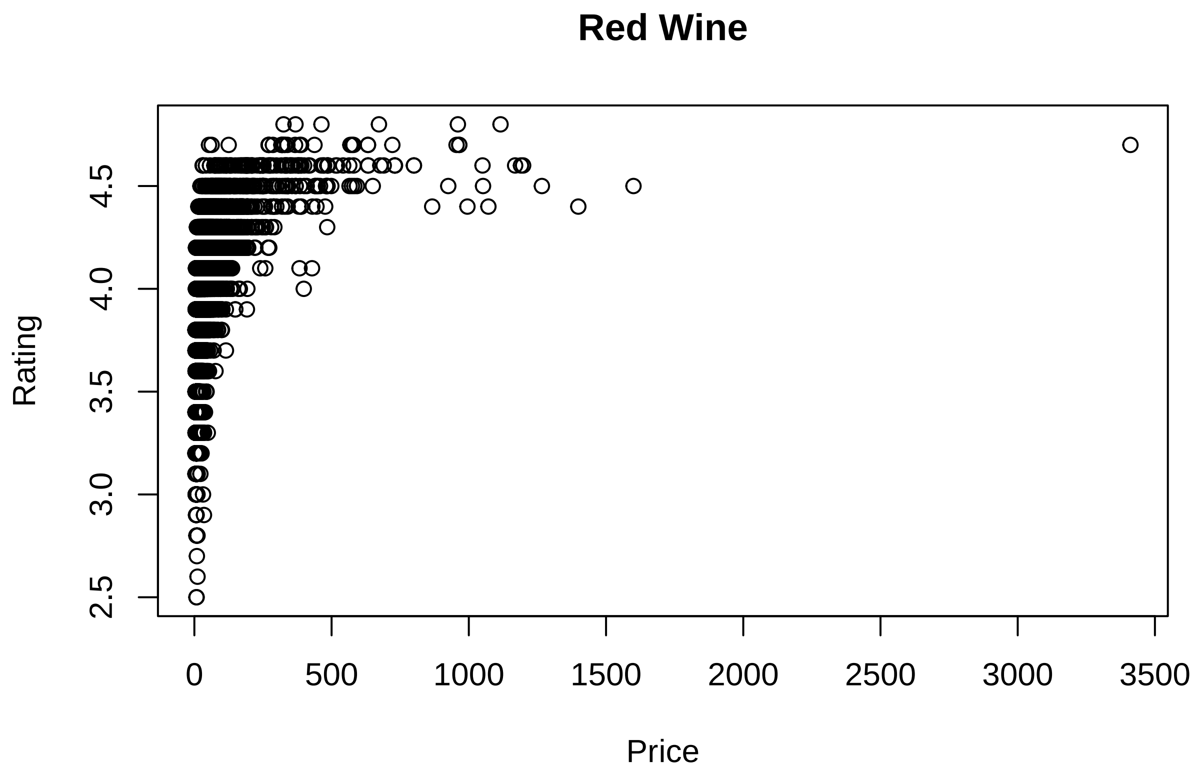


C. Informative Graphs

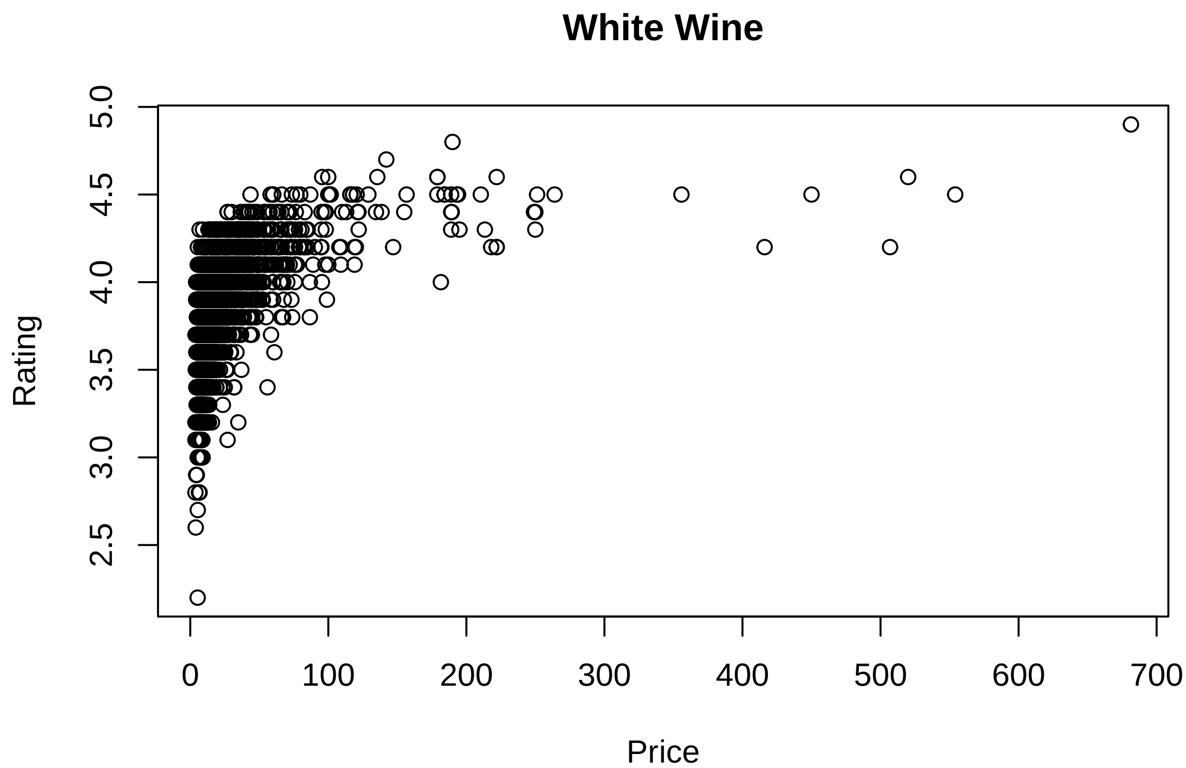
```
par(mfrow = c(1,2))

# Comparing types with Price
ggplot(data = train, aes(x = Rating, y = Price)) +
  geom_bar(stat = "identity")
```



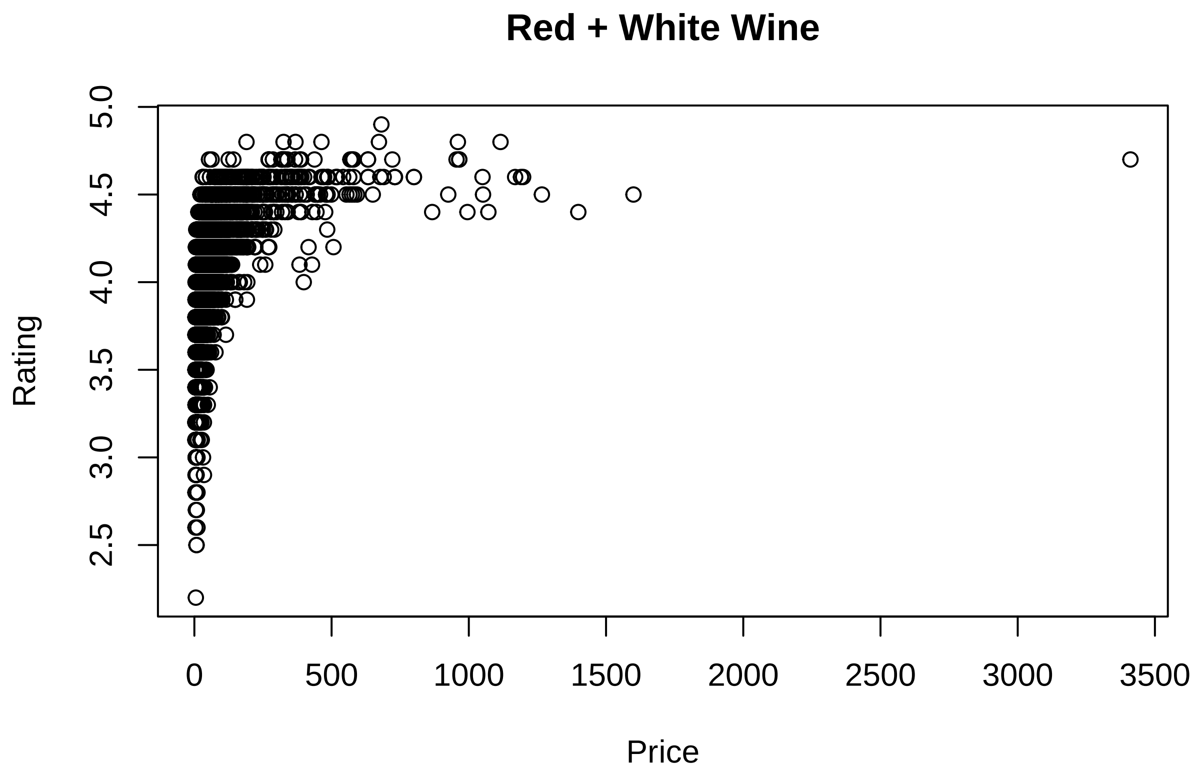

```r
# Red
plot(Rating ~ Price, data = Red, main = "Red Wine", xlab = "Price", ylab = "Rating")
```



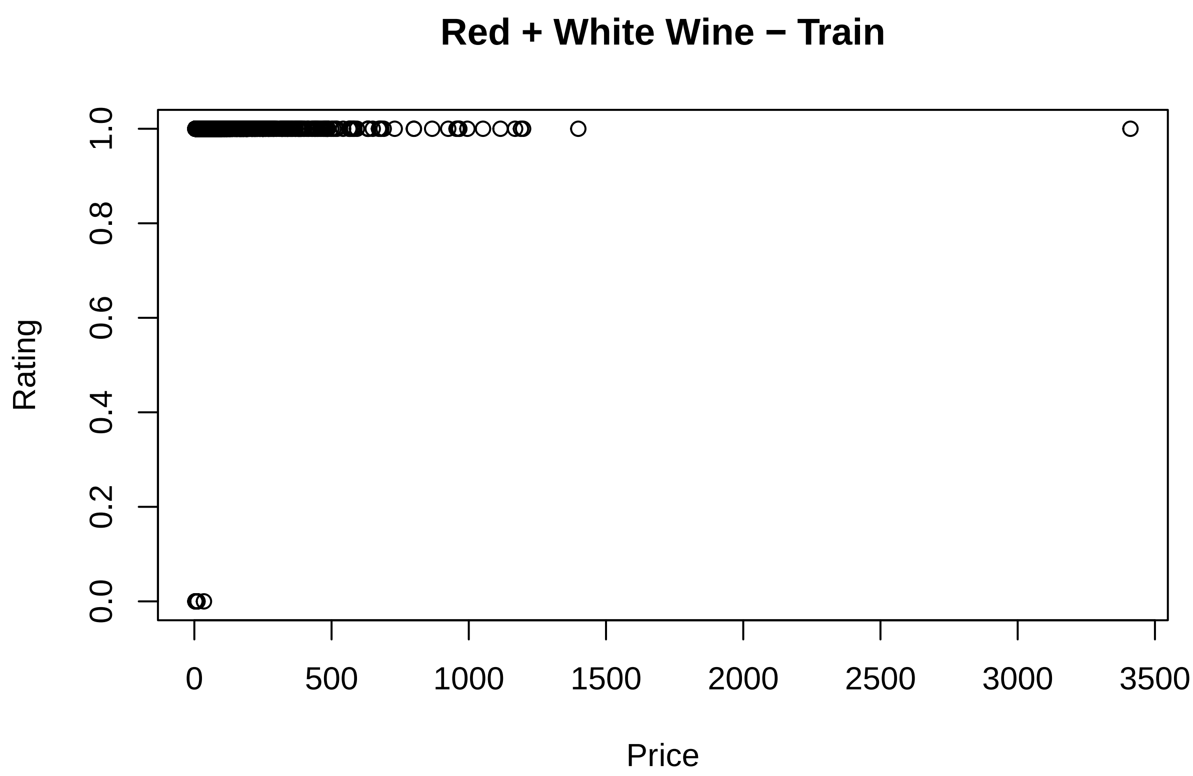

```r
# White
plot(Rating ~ Price, data = White, main = "White Wine", xlab = "Price", ylab = "Rating")
```




```r
# Total
plot(Rating ~ Price, data = totWine, main = "Red + White Wine", xlab = "Price", ylab = "Rating")
```




```
# After setting ratings to 1 and 0
plot(Rating ~ Price, data = train, main = "Red + White Wine - Train", xlab = "Price", ylab = "Rating")
```



D. Logistic Regression Model + Summary

```
glm1 <- glm(Rating ~ Price, data = train, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm1)
```

```
##
## Call:
## glm(formula = Rating ~ Price, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.8341   0.0052   0.0363   0.0774   0.1617
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.4619     0.6708   5.161 2.46e-07 ***
## Price         0.2349     0.0715   3.286  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 263.19  on 9917  degrees of freedom
## Residual deviance: 234.90  on 9916  degrees of freedom
## AIC: 238.9
##
## Number of Fisher Scoring iterations: 13
```

```
probs <- predict(glm1, nevdata = test, type = "response")
pred <- ifelse(probs > 0.5, 1, 0)
acc1 <- mean(pred == as.integer (test$Rating))
```

```
## Warning in pred == as.integer(test$Rating): longer object length is not a
## multiple of shorter object length
```

```
print(paste ("glm1 accuracy = ", acc1))
```

```
## [1] "glm1 accuracy =  0.997983464408147"
```

E. Naïve Bayes Model + Output + Evaluation

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.3
```

```
nb1 <- naiveBayes(Rating ~ ., data = train)

# Output
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##           0           1
## 0.001814882 0.998185118
##
## Conditional probabilities:
##    Country
## Y      Argentina    Australia      Austria       Brazil     Bulgaria
##   0 0.0555555556 0.0000000000 0.0555555556 0.0555555556 0.0000000000
##   1 0.0235353535 0.0248484848 0.0380808081 0.0030303030 0.0002020202
##    Country
## Y         Canada        Chile        China      Croatia Czech Republic
##   0 0.0000000000 0.0000000000 0.0555555556 0.0000000000   0.0000000000
```

```
##   1 0.0001010101 0.0336363636 0.0002020202 0.0005050505   0.0002020202
##    Country
## Y         France      Georgia      Germany       Greece      Hungary
##   0 0.1111111111 0.0000000000 0.0000000000 0.1111111111 0.0000000000
##   1 0.2378787879 0.0009090909 0.0918181818 0.0016161616 0.0012121212
##    Country
## Y         Israel        Italy      Lebanon   Luxembourg       Mexico
##   0 0.0000000000 0.2777777778 0.0000000000 0.0000000000 0.0000000000
##   1 0.0015151515 0.2763636364 0.0009090909 0.0003030303 0.0001010101
##    Country
## Y        Moldova  New Zealand     Portugal      Romania     Slovakia
##   0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##   1 0.0010101010 0.0119191919 0.0252525253 0.0025252525 0.0001010101
##    Country
## Y       Slovenia South Africa        Spain  Switzerland       Turkey
##   0 0.0000000000 0.1666666667 0.0555555556 0.0000000000 0.0000000000
##   1 0.0011111111 0.0646464646 0.1133333333 0.0019191919 0.0007070707
##    Country
## Y   United States      Uruguay
##   0  0.0555555556 0.0000000000
##   1  0.0400000000 0.0005050505
##
##    NumberOfRatings
## Y       [,1]     [,2]
##   0 143.3889 235.3193
##   1 343.8533 804.2182
##
##    Price
## Y        [,1]      [,2]
##   0  8.772222  6.960148
##   1 33.913559 74.076015
##
##    Year
## Y           1988         1989         1990         1991         1992
##   0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##   1 0.0001010101 0.0002020202 0.0002020202 0.0001010101 0.0001010101
##    Year
## Y           1993         1995         1996         1997         1998
##   0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##   1 0.0002020202 0.0002020202 0.0004040404 0.0005050505 0.0004040404
##    Year
## Y           1999         2000         2001         2002         2003
##   0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##   1 0.0013131313 0.0014141414 0.0010101010 0.0007070707 0.0011111111
##    Year
## Y           2004         2005         2006         2007         2008
##   0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##   1 0.0025252525 0.0137373737 0.0040404040 0.0043434343 0.0070707071
##    Year
## Y           2009         2010         2011         2012         2013
##   0 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.1111111111
##   1 0.0067676768 0.0141414141 0.0255555556 0.0312121212 0.0483838384
##    Year
## Y           2014         2015         2016         2017         2018
```

```
##   0 0.0000000000 0.1111111111 0.0555555556 0.1111111111 0.4444444444
##   1 0.0709090909 0.1351515152 0.1814141414 0.1853535354 0.2049494949
##    Year
## Y          2019         N.V.
##   0 0.0555555556 0.1111111111
##   1 0.0555555556 0.0009090909
```

```r
# Evaluate
# Predict
p1 <- predict(nb1, newdata = test, type = "class")
table(p1, test$Rating)
```

```
##
## p1     0    1
##   0    0   25
##   1    5 2450
```

```r
# Mean
mean(p1 == test$Rating)
```

```
## [1] 0.9879032
```

F. Classification Models + Compare

-> Logistic Regression Accuracy = 0.9979839 -> Naive Bayes Accuracy = 0.9879032

-> Area under Logistic Regression ROC = 0.9210101 -> Area under Naive Bayes ROC = 0.4949495

-> We can conclude LR is doing good while NB is performing a little randomly

```r
# Logistic
pred1 <- predict(glm1, newdata = test, type = "response")
probs1 <- ifelse(pred1 > 0.5, 1, 0)
acc1 <- mean(probs1 == test$Rating)

acc1
```

```
## [1] 0.9979839
```

```r
head(table(pred1, test$Rating))
```

```
##
## pred1               0 1
##   0.986557665294291 0 1
##   0.98857269421588  0 1
##   0.988678361167715 0 1
##   0.988835053038242 0 1
##   0.989091454847214 0 1
##   0.989142032053988 0 1
```

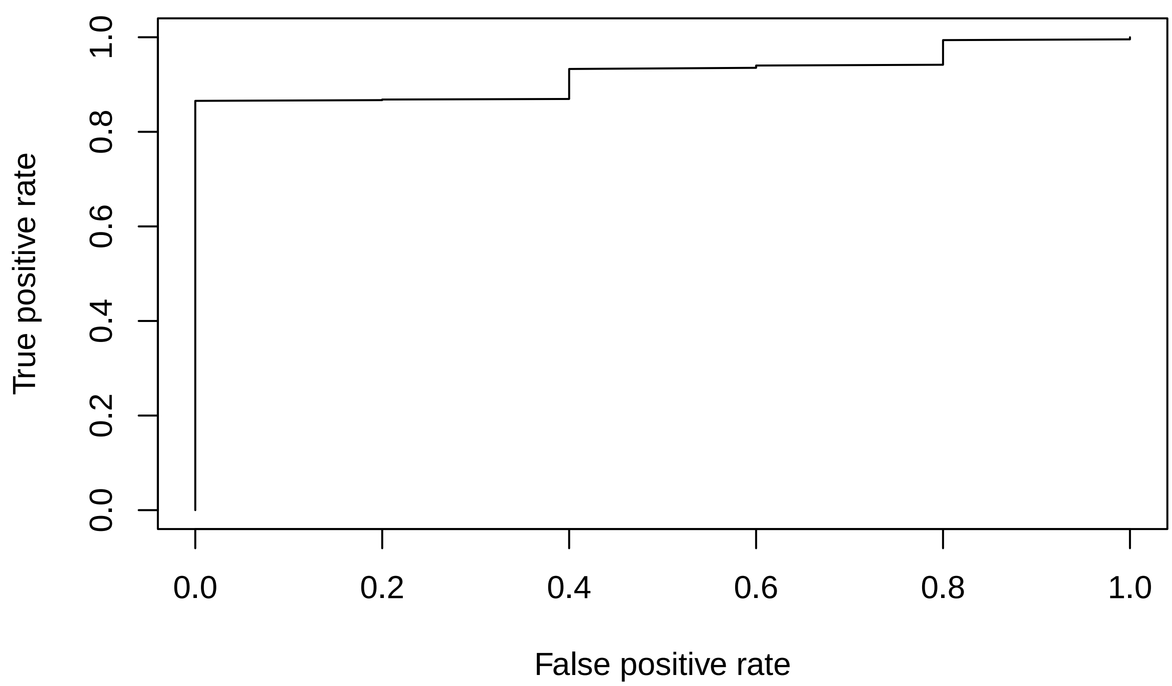
```r
# Naive Bayes
pred2 <- predict(nb1, newdata = test, type = "class")
acc2 <- mean(pred2 == test$Rating)

acc2
```

```
## [1] 0.9879032
```

```r
head(table(pred2, test$Rating))
```

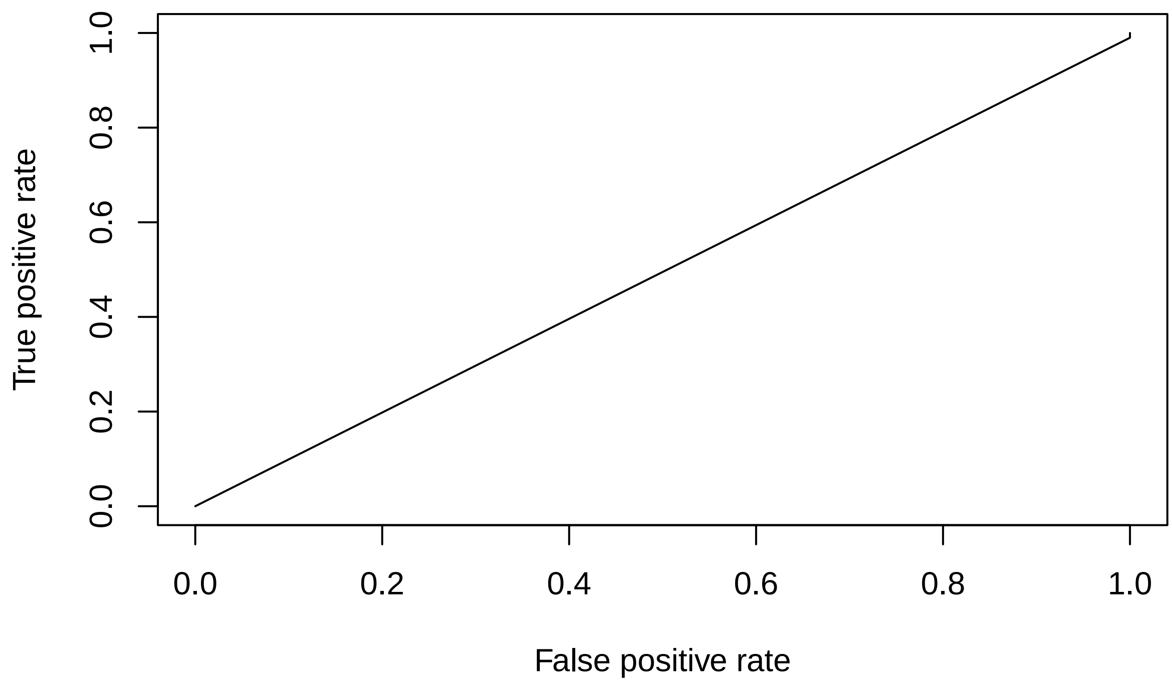
```
##
## pred2    0    1
##     0    0   25
##     1    5 2450
```

```r
p1 <- predict(glm1, newdata = test, type = "response")
pr1 <- prediction(p1, test$Rating)
prf1 <- performance(pr1, measure = "tpr", x.measure = "fpr")

plot(prf1)
```




```r
auc1 <- performance(pr1, measure = "auc")
auc1 <- auc1 @ y.values[[1]]
```

```r
p2 <- predict(nb1, newdata = test, type = "class")
pr2 <- prediction(as.numeric(p2), as.numeric(test$Rating))
prf2 <- performance(pr2, measure = "tpr", x.measure = "fpr")

plot(prf2)
```




```r
auc2 <- performance(pr2, measure = "auc")
auc2 <- auc2 @ y.values[[1]]
```

G. Strengths and Weaknesses of Naïve Bayes and Logistic Regression

- The strengths of Naive Bayes includes the easy implementation as well as working well with rather small sets of data; the interpretation of the data output is also on the easier side.
- While Naive Bayes is good with smaller data, it starts to struggle with larger sets of data. The method is naive due to assuming that each input variable is independent.
- The strengths of Logistic Regression include easy to implement, interpret, and efficient to train. It also does not make assumptions about distributions of classes in feature space.
- However, the number of observations is lesser than the number of features, which can lead to overfitting.

H. Benefits, Drawbacks, Experience

- Accuracy: The function tells us the rate of correct predictions over the number of observations. It's one of the more simple and common matrics to use. It might not be the best in calculating accuracy in very imbalanced sets of data.

- ROC and AUC: ROC compares the prediction between True Value Rates and False Value Rates. Although I think most ROC graphs should be more cuvred like a Square Root Function; however, my graph turned out kind of jagged like a staricase. AUC tells us the area under the ROC curve and gives an indication on how good the model is on a scale of 0.5 to 1, where 1 is the best. In my case, AUC1 is 0.9210101 and AUC2 is 0.4949495, which means AUC1 is doing better than AUC2.

- MCC: This is another method of accuracy; however it considers all other differences in class distribution.