Data 621 - Homework 5

Group 2: William Aiken, Donald Butler, Michael Ippolito, Bharani Nittala, and Leticia Salazar

December 11, 2022

Contents

Overview

We will explore, analyze and model a data set containing information on approximately 12,000 commercially available wines. The variables are mostly related to the chemical properties of the wine being sold. The response variable is the number of sample cases of wine that were purchased by wine distribution companies after sampling a wine. These cases would be used to provide tasting samples to restaurants and wine stores around the United States. The more sample cases purchased, the more likely is a wine to be sold at a high end restaurant. A larger wine manufacturer is studying the data in order to predict the number of wine cases ordered based upon the wine characteristics. If the wine manufacturer can predict the number of cases, then that manufacturer will be able to adjust their wine offering to maximize sales.

Objective

Build a count regression model to predict the number of cases of wine that will be sold given certain properties of the wine. HINT: Sometimes, the face that a variable is missing is actually predictive of the target.

Description

Below is a short description of the variables of interest in the data set:

VARIABLE NAME:	DEFINITION:	THEORETICAL EFFECT:
INDEX	Identification Variable (do not use)	None
TARGET	Number of Cases Purchased	None
AcidIndex	Proprietary method of testing totalacidity of wine by using a weighted average	
Alcohol	Alcohol Content	
Chorides	Cholride content of wine	
CitricAcid	Citric Acid Content	
Density	Density of Wine	
FixedAcidity	Fixed Acidity of Wiine	
FreeSulfurDioxide	Sulfur Dioxide content of wine	
LabelAppeal	Marketing Score indicating the appeal of label design for consumers. High numbers suggest customers like the label design. Negative numbers suggest customers don't like the design.	Many consumers purchase based on the visual appeal of the wine label design. Higher numbers suggest better sales.
ResidualSugar	Residual Sugar of wine	
STARS	Wine rating by a team of experts: 4 Stars = Excellent, 1 Star = Poor	A high number of stars suggests high sales
Sulphates	Sulfate content of Wine	
TotalSulfurDioxide	Total Sulfur Dioxide of Wine	
VolatileAcidity	Volatile Acid content of wine	
pH	pH of wine	

Load Libraries

These are the libraries used to explore, prepare, analyze and build our models

```
library(tidyverse)
library(dplyr)
library(corrplot)
library(skimr)
library(DataExplorer)
library(ggplot2)
library(hrbrthemes)
library(mice)
library(MASS)
library(dvmisc)
library(gridExtra)
library(lattice)
library(faraway)
library(pscl)
```

Load Data set

We have included the original data sets in our GitHub account and read from this location. Below we are showing the training data set:

```
##   INDEX TARGET FixedAcidity VolatileAcidity CitricAcid ResidualSugar Chlorides
## 1     1      3          3.2           1.160      -0.98          54.2    -0.567
## 2     2      3          4.5           0.160      -0.81          26.1    -0.425
## 3     4      5          7.1           2.640      -0.88          14.8     0.037
## 4     5      3          5.7           0.385       0.04          18.8    -0.425
## 5     6      4          8.0           0.330      -1.26           9.4        NA
## 6     7      0         11.3           0.320       0.59           2.2     0.556
##   FreeSulfurDioxide TotalSulfurDioxide Density   pH Sulphates Alcohol
## 1                NA                268 0.99280 3.33     -0.59     9.9
## 2                15               -327 1.02792 3.38      0.70      NA
## 3               214                142 0.99518 3.12      0.48    22.0
## 4                22                115 0.99640 2.24      1.83     6.2
## 5              -167                108 0.99457 3.12      1.77    13.7
## 6               -37                 15 0.99940 3.20      1.29    15.4
##   LabelAppeal AcidIndex STARS
## 1           0         8     2
## 2          -1         7     3
## 3          -1         8     3
## 4          -1         6     1
## 5           0         9     2
## 6           0        11    NA
```

Data Exploration

Using the `summary()` function lets start exploring the training and evaluation data.

Training:

```
##      INDEX           TARGET       FixedAcidity     VolatileAcidity
##  Min.   :    1   Min.   :0.000   Min.   :-18.100   Min.   :-2.7900
##  1st Qu.: 4038   1st Qu.:2.000   1st Qu.:  5.200   1st Qu.: 0.1300
##  Median : 8110   Median :3.000   Median :  6.900   Median : 0.2800
##  Mean   : 8070   Mean   :3.029   Mean   :  7.076   Mean   : 0.3241
##  3rd Qu.:12106   3rd Qu.:4.000   3rd Qu.:  9.500   3rd Qu.: 0.6400
##  Max.   :16129   Max.   :8.000   Max.   : 34.400   Max.   : 3.6800
##
##    CitricAcid      ResidualSugar        Chlorides       FreeSulfurDioxide
##  Min.   :-3.2400   Min.   :-127.800   Min.   :-1.1710   Min.   :-555.00
##  1st Qu.: 0.0300   1st Qu.:  -2.000   1st Qu.:-0.0310   1st Qu.:   0.00
##  Median : 0.3100   Median :   3.900   Median : 0.0460   Median :  30.00
##  Mean   : 0.3084   Mean   :   5.419   Mean   : 0.0548   Mean   :  30.85
##  3rd Qu.: 0.5800   3rd Qu.:  15.900   3rd Qu.: 0.1530   3rd Qu.:  70.00
##  Max.   : 3.8600   Max.   : 141.150   Max.   : 1.3510   Max.   : 623.00
```

```
##                   NA's   :616        NA's   :638       NA's   :647
##  TotalSulfurDioxide    Density             pH          Sulphates
##  Min.   :-823.0     Min.   :0.8881   Min.   :0.480   Min.   :-3.1300
##  1st Qu.:  27.0     1st Qu.:0.9877   1st Qu.:2.960   1st Qu.: 0.2800
##  Median : 123.0     Median :0.9945   Median :3.200   Median : 0.5000
##  Mean   : 120.7     Mean   :0.9942   Mean   :3.208   Mean   : 0.5271
##  3rd Qu.: 208.0     3rd Qu.:1.0005   3rd Qu.:3.470   3rd Qu.: 0.8600
##  Max.   :1057.0     Max.   :1.0992   Max.   :6.130   Max.   : 4.2400
##  NA's   :682                         NA's   :395     NA's   :1210
##     Alcohol       LabelAppeal          AcidIndex          STARS
##  Min.   :-4.70   Min.   :-2.000000   Min.   : 4.000   Min.   :1.000
##  1st Qu.: 9.00   1st Qu.:-1.000000   1st Qu.: 7.000   1st Qu.:1.000
##  Median :10.40   Median : 0.000000   Median : 8.000   Median :2.000
##  Mean   :10.49   Mean   :-0.009066   Mean   : 7.773   Mean   :2.042
##  3rd Qu.:12.40   3rd Qu.: 1.000000   3rd Qu.: 8.000   3rd Qu.:3.000
##  Max.   :26.50   Max.   : 2.000000   Max.   :17.000   Max.   :4.000
##  NA's   :653                                          NA's   :3359
```

Evaluation:

```
##        IN          TARGET         FixedAcidity     VolatileAcidity
##  Min.   :    3   Mode:logical   Min.   :-18.200   Min.   :-2.8300
##  1st Qu.: 4018   NA's:3335      1st Qu.:  5.200   1st Qu.: 0.0800
##  Median : 7906                  Median :  6.900   Median : 0.2800
##  Mean   : 8048                  Mean   :  6.864   Mean   : 0.3103
##  3rd Qu.:12061                  3rd Qu.:  9.000   3rd Qu.: 0.6300
##  Max.   :16130                  Max.   : 33.500   Max.   : 3.6100
##
##    CitricAcid      ResidualSugar        Chlorides        FreeSulfurDioxide
##  Min.   :-3.1200   Min.   :-128.300   Min.   :-1.15000   Min.   :-563.00
##  1st Qu.: 0.0000   1st Qu.:  -2.600   1st Qu.: 0.01600   1st Qu.:   3.00
##  Median : 0.3100   Median :   3.600   Median : 0.04700   Median :  30.00
##  Mean   : 0.3124   Mean   :   5.319   Mean   : 0.06143   Mean   :  34.95
##  3rd Qu.: 0.6050   3rd Qu.:  17.200   3rd Qu.: 0.17100   3rd Qu.:  79.25
##  Max.   : 3.7600   Max.   : 145.400   Max.   : 1.26300   Max.   : 617.00
##                    NA's   :168        NA's   :138        NA's   :152
##  TotalSulfurDioxide    Density             pH          Sulphates
##  Min.   :-769.00    Min.   :0.8898   Min.   :0.600   Min.   :-3.0700
##  1st Qu.:  27.25    1st Qu.:0.9883   1st Qu.:2.980   1st Qu.: 0.3300
##  Median : 124.00    Median :0.9946   Median :3.210   Median : 0.5000
##  Mean   : 123.41    Mean   :0.9947   Mean   :3.237   Mean   : 0.5346
##  3rd Qu.: 210.00    3rd Qu.:1.0005   3rd Qu.:3.490   3rd Qu.: 0.8200
##  Max.   :1004.00    Max.   :1.0998   Max.   :6.210   Max.   : 4.1800
##  NA's   :157                         NA's   :104     NA's   :310
##     Alcohol       LabelAppeal          AcidIndex          STARS
##  Min.   :-4.20   Min.   :-2.00000   Min.   : 5.000   Min.   :1.00
##  1st Qu.: 9.00   1st Qu.:-1.00000   1st Qu.: 7.000   1st Qu.:1.00
##  Median :10.40   Median : 0.00000   Median : 8.000   Median :2.00
##  Mean   :10.58   Mean   : 0.01349   Mean   : 7.748   Mean   :2.04
##  3rd Qu.:12.50   3rd Qu.: 1.00000   3rd Qu.: 8.000   3rd Qu.:3.00
##  Max.   :25.60   Max.   : 2.00000   Max.   :17.000   Max.   :4.00
##  NA's   :185                                         NA's   :841
```

Using the `DataExplorer` package we use the `create_report` function which pulls a full data profile from our training data set and create an html file with basic statistics, structure, missing data, distribution visualizations, correlation matrix and principal component analysis for our data. You can find these output in our github.

```
# Do not render since it will produce a separate html file
# Remove TARGET from eval report since it will contain all
# NAs and will make the correlation plot fail to render
DataExplorer::create_report(dftrain, output_file = "training_report.html")
DataExplorer::create_report(dfeval %>%
    dplyr::select(-TARGET), output_file = "eval_report.html")
```

Based on this our training data includes 12795 records and 16 variables whereas the evaluation data includes 3335 records and 16 variables.

Training:

```
## 'data.frame':    12795 obs. of  16 variables:
##  $ INDEX              : int  1 2 4 5 6 7 8 11 12 13 ...
##  $ TARGET             : int  3 3 5 3 4 0 0 4 3 6 ...
##  $ FixedAcidity       : num  3.2 4.5 7.1 5.7 8 11.3 7.7 6.5 14.8 5.5 ...
##  $ VolatileAcidity    : num  1.16 0.16 2.64 0.385 0.33 0.32 0.29 -1.22 0.27 -0.22 ...
##  $ CitricAcid         : num  -0.98 -0.81 -0.88 0.04 -1.26 0.59 -0.4 0.34 1.05 0.39 ...
##  $ ResidualSugar      : num  54.2 26.1 14.8 18.8 9.4 ...
##  $ Chlorides          : num  -0.567 -0.425 0.037 -0.425 NA 0.556 0.06 0.04 -0.007 -0.277 ...
##  $ FreeSulfurDioxide  : num  NA 15 214 22 -167 -37 287 523 -213 62 ...
##  $ TotalSulfurDioxide: num  268 -327 142 115 108 15 156 551 NA 180 ...
##  $ Density            : num  0.993 1.028 0.995 0.996 0.995 ...
##  $ pH                 : num  3.33 3.38 3.12 2.24 3.12 3.2 3.49 3.2 4.93 3.09 ...
##  $ Sulphates          : num  -0.59 0.7 0.48 1.83 1.77 1.29 1.21 NA 0.26 0.75 ...
##  $ Alcohol            : num  9.9 NA 22 6.2 13.7 15.4 10.3 11.6 15 12.6 ...
##  $ LabelAppeal        : int  0 -1 -1 -1 0 0 0 1 0 0 ...
##  $ AcidIndex          : int  8 7 8 6 9 11 8 7 6 8 ...
##  $ STARS              : int  2 3 3 1 2 NA NA 3 NA 4 ...
```

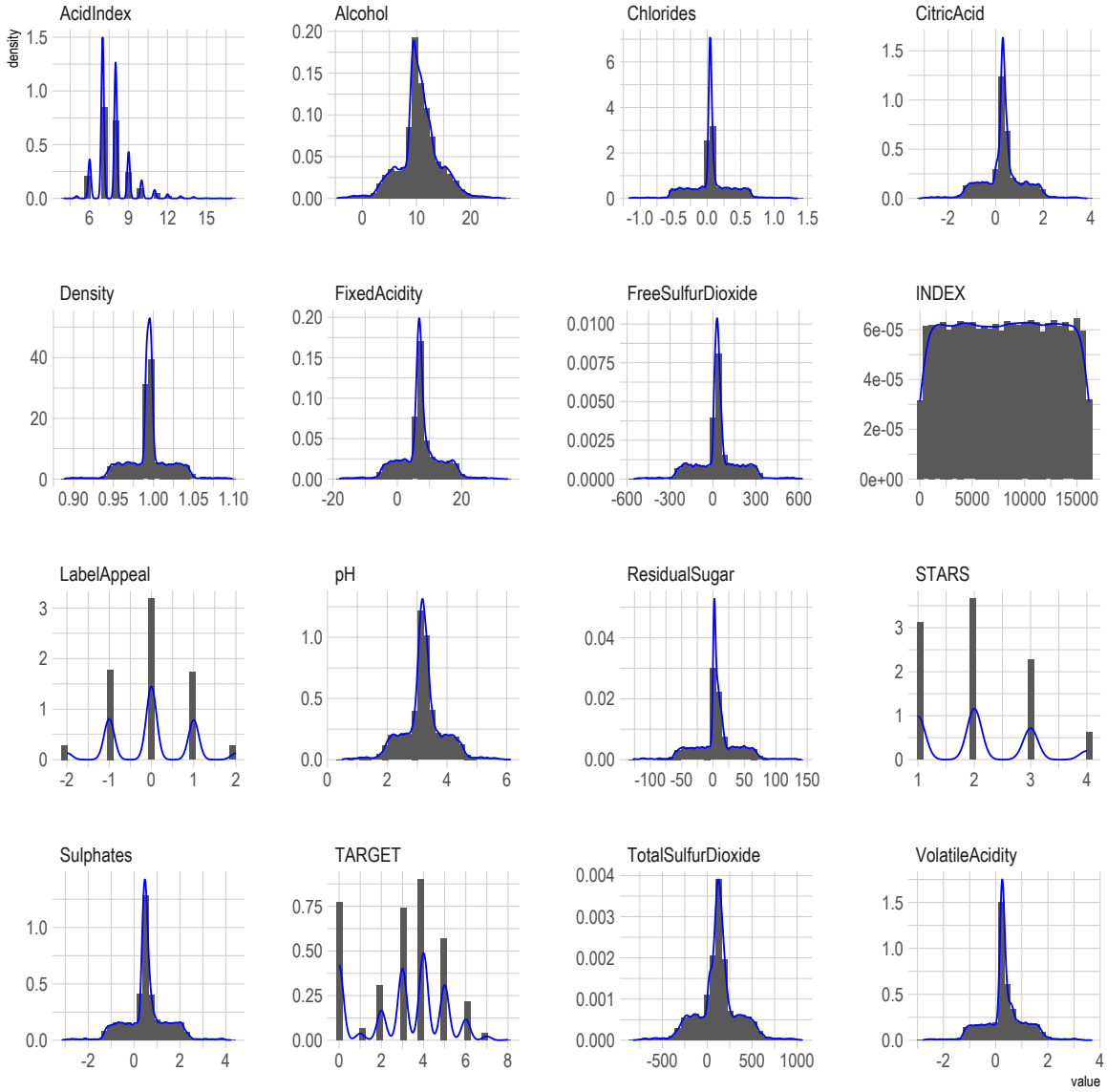
Evaluation:

```
## 'data.frame':    3335 obs. of  16 variables:
##  $ IN                 : int  3 9 10 18 21 30 31 37 39 47 ...
##  $ TARGET             : logi  NA NA NA NA NA NA ...
##  $ FixedAcidity       : num  5.4 12.4 7.2 6.2 11.4 17.6 15.5 15.9 11.6 3.8 ...
##  $ VolatileAcidity    : num  -0.86 0.385 1.75 0.1 0.21 0.04 0.53 1.19 0.32 0.22 ...
##  $ CitricAcid         : num  0.27 -0.76 0.17 1.8 0.28 -1.15 -0.53 1.14 0.55 0.31 ...
##  $ ResidualSugar      : num  -10.7 -19.7 -33 1 1.2 1.4 4.6 31.9 -50.9 -7.7 ...
##  $ Chlorides          : num  0.092 1.169 0.065 -0.179 0.038 ...
##  $ FreeSulfurDioxide  : num  23 -37 9 104 70 -250 10 115 35 40 ...
##  $ TotalSulfurDioxide: num  398 68 76 89 53 140 17 381 83 129 ...
##  $ Density            : num  0.985 0.99 1.046 0.989 1.029 ...
##  $ pH                 : num  5.02 3.37 4.61 3.2 2.54 3.06 3.07 2.99 3.32 4.72 ...
```

```
##  $ Sulphates          : num  0.64 1.09 0.68 2.11 -0.07 -0.02 0.75 0.31 2.18 -0.64 ...
##  $ Alcohol            : num  12.3 16 8.55 12.3 4.8 11.4 8.5 11.4 -0.5 10.9 ...
##  $ LabelAppeal        : int  -1 0 0 -1 0 1 0 1 0 0 ...
##  $ AcidIndex          : int  6 6 8 8 10 8 12 7 12 7 ...
##  $ STARS              : int  NA 2 1 1 NA 4 3 NA NA NA ...
```

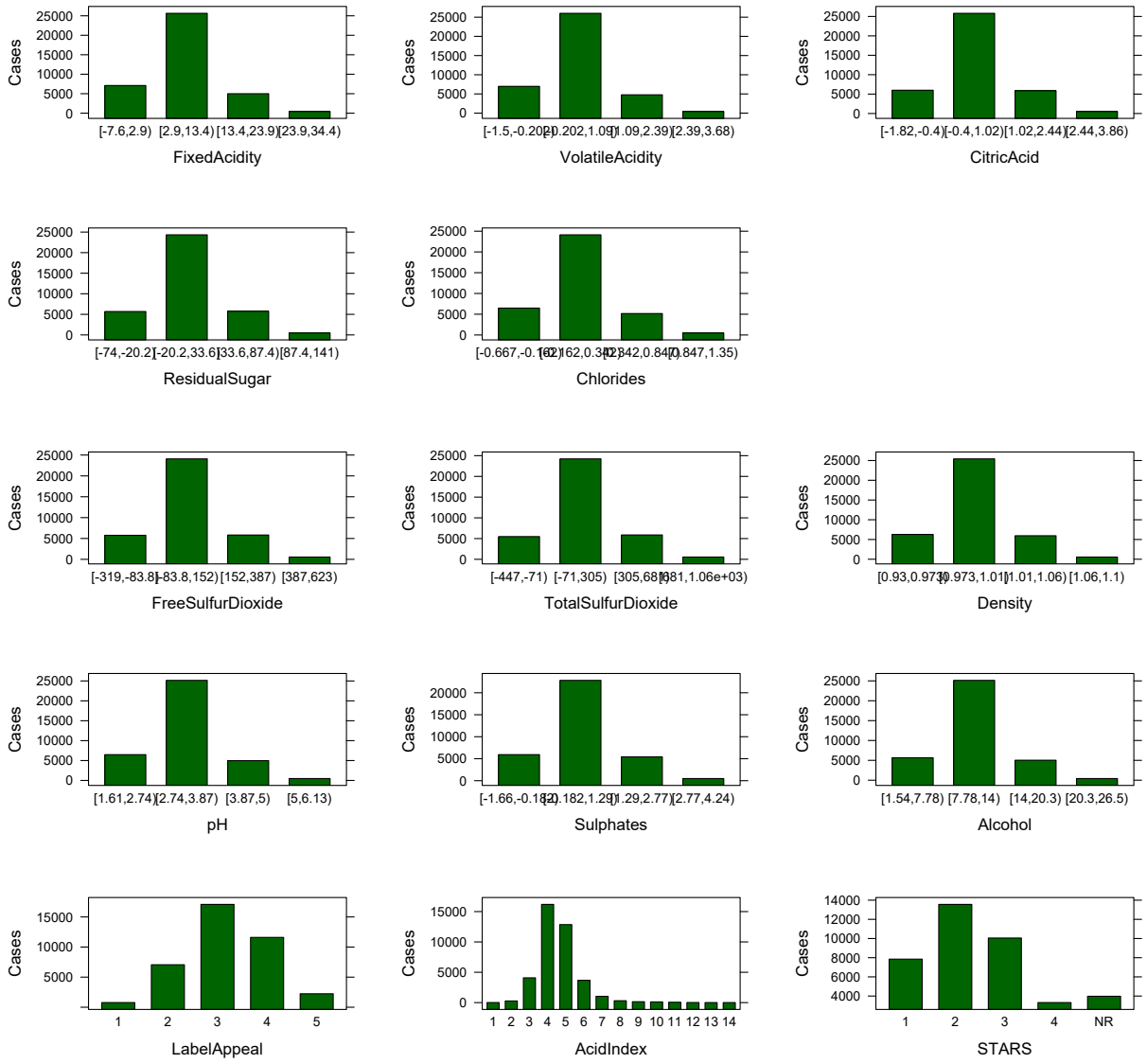
Lets take a look at the distribution of each variables in the training data set.

Based on the plots below, we can tell that most of the variables seem to be normally distributed with the exception of `AcidIndex` and `STARS` being right skewed. `INDEX` shows a uniform distribution but has no effect on our data so during the data preparation stage we will be removing it.



The fact that some wines are not rated could be a potential predictor. We'll treat NAs as its own star rating. It is noted that there are a significant number of zeros in the TARGET variable, meaning that no cases were sold of that particular variety of wine. In addition, the fact that some wines are not rated could be a potential predictor. We'll treat NAs as its own star rating.

We'll also look at the number of cases of wine sold against the predictors.



As shown, more cases of wine are sold for mid-range values of all categories of acidity, sugar, chlorides, the dioxides, density, pH, sulphates, and alcohol. Surprisingly, more cases were sold for labels that had mid-range label appeal. A lower acid index seemed to indicate more cases sold. And more cases were sold for wines rated only two stars, indicating that consumers may consider higher-starred wines as too pricey.

Data Preparation

Data preparation was performed on both the training and evaluation data sets but will only be displayed for the training data. We'll also need to removing the `INDEX` variable.

Now we'll impute missing values using R's Multiple Imputation by Chained Equations (MICE) package. We'll avoid imputing the STARS variable as the absence of a star rating may be a significant predictor.

Lets look at another summary to make sure there aren't any NAs where we're not expecting them.

Training data:

```
##      TARGET       FixedAcidity     VolatileAcidity     CitricAcid
##  Min.   :0.000   Min.   :-18.100   Min.   :-2.7900   Min.   :-3.2400
##  1st Qu.:2.000   1st Qu.:  5.200   1st Qu.: 0.1300   1st Qu.: 0.0300
##  Median :3.000   Median :  6.900   Median : 0.2800   Median : 0.3100
##  Mean   :3.029   Mean   :  7.076   Mean   : 0.3241   Mean   : 0.3084
##  3rd Qu.:4.000   3rd Qu.:  9.500   3rd Qu.: 0.6400   3rd Qu.: 0.5800
##  Max.   :8.000   Max.   : 34.400   Max.   : 3.6800   Max.   : 3.8600
##  ResidualSugar        Chlorides        FreeSulfurDioxide TotalSulfurDioxide
##  Min.   :-127.800   Min.   :-1.17100   Min.   :-555.00   Min.   :-823.0
##  1st Qu.:  -2.000   1st Qu.:-0.02900   1st Qu.:  -1.00   1st Qu.:  27.0
##  Median :   3.900   Median : 0.04600   Median :  30.00   Median : 124.0
##  Mean   :   5.481   Mean   : 0.05501   Mean   :  30.13   Mean   : 120.9
##  3rd Qu.:  16.000   3rd Qu.: 0.15400   3rd Qu.:  69.00   3rd Qu.: 208.0
##  Max.   : 141.150   Max.   : 1.35100   Max.   : 623.00   Max.   :1057.0
##     Density             pH          Sulphates          Alcohol
##  Min.   :0.8881   Min.   :0.480   Min.   :-3.1300   Min.   :-4.70
##  1st Qu.:0.9877   1st Qu.:2.950   1st Qu.: 0.2800   1st Qu.: 9.00
##  Median :0.9945   Median :3.200   Median : 0.5000   Median :10.40
##  Mean   :0.9942   Mean   :3.207   Mean   : 0.5253   Mean   :10.48
##  3rd Qu.:1.0005   3rd Qu.:3.470   3rd Qu.: 0.8600   3rd Qu.:12.40
##  Max.   :1.0992   Max.   :6.130   Max.   : 4.2400   Max.   :26.50
##   LabelAppeal          AcidIndex         STARS
##  Min.   :-2.000000   Min.   : 4.000   Length:12795
##  1st Qu.:-1.000000   1st Qu.: 7.000   Class :character
##  Median : 0.000000   Median : 8.000   Mode  :character
##  Mean   :-0.009066   Mean   : 7.773
##  3rd Qu.: 1.000000   3rd Qu.: 8.000
##  Max.   : 2.000000   Max.   :17.000
```

Evaluation data:

```
##   FixedAcidity     VolatileAcidity     CitricAcid      ResidualSugar
```

```
##  Min.   :-18.200   Min.   :-2.8300   Min.   :-3.1200   Min.   :-128.300
##  1st Qu.:  5.200   1st Qu.: 0.0800   1st Qu.: 0.0000   1st Qu.:  -2.450
##  Median :  6.900   Median : 0.2800   Median : 0.3100   Median :   3.600
##  Mean   :  6.864   Mean   : 0.3103   Mean   : 0.3124   Mean   :   5.339
##  3rd Qu.:  9.000   3rd Qu.: 0.6300   3rd Qu.: 0.6050   3rd Qu.:  17.300
##  Max.   : 33.500   Max.   : 3.6100   Max.   : 3.7600   Max.   : 145.400
##    Chlorides        FreeSulfurDioxide TotalSulfurDioxide    Density
##  Min.   :-1.15000   Min.   :-563.00   Min.   :-769.0     Min.   :0.8898
##  1st Qu.: 0.01800   1st Qu.:   3.00   1st Qu.:  28.0     1st Qu.:0.9883
##  Median : 0.04700   Median :  30.00   Median : 124.0     Median :0.9946
##  Mean   : 0.06261   Mean   :  35.71   Mean   : 122.9     Mean   :0.9947
##  3rd Qu.: 0.17200   3rd Qu.:  81.00   3rd Qu.: 209.5     3rd Qu.:1.0005
##  Max.   : 1.26300   Max.   : 617.00   Max.   :1004.0     Max.   :1.0998
##        pH          Sulphates          Alcohol       LabelAppeal
##  Min.   :0.600   Min.   :-3.0700   Min.   :-4.20   Min.   :-2.00000
##  1st Qu.:2.980   1st Qu.: 0.3300   1st Qu.: 9.00   1st Qu.:-1.00000
##  Median :3.210   Median : 0.5000   Median :10.40   Median : 0.00000
##  Mean   :3.238   Mean   : 0.5376   Mean   :10.58   Mean   : 0.01349
##  3rd Qu.:3.485   3rd Qu.: 0.8300   3rd Qu.:12.50   3rd Qu.: 1.00000
##  Max.   :6.210   Max.   : 4.1800   Max.   :25.60   Max.   : 2.00000
##    AcidIndex         STARS            TARGET
##  Min.   : 5.000   Length:3335        Mode:logical
##  1st Qu.: 7.000   Class :character   NA's:3335
##  Median : 8.000   Mode  :character
##  Mean   : 7.748
##  3rd Qu.: 8.000
##  Max.   :17.000
```

Build Models

Based on the data, we'll try several model types: a poisson general linear model (GLM), a Gaussian multiple linear model, a negative binomial model, and a zero-inflated poisson model. If none of these models produce acceptable results, we will consider a hierarchical model, first splitting the data using a binomial model by zero cases sold vs more than zero cases sold, then using a poisson model against the non-zero cases.

Poisson Models

- Possion Model 1

```
##
## Call:
## glm(formula = TARGET ~ ., family = "poisson", data = cleandf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2694  -0.6569  -0.0037   0.4499   3.7620
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)         1.886e+00  1.951e-01   9.664  < 2e-16 ***
## FixedAcidity        7.930e-05  8.199e-04   0.097 0.922955
## VolatileAcidity    -3.042e-02  6.527e-03  -4.660 3.17e-06 ***
## CitricAcid          4.961e-03  5.897e-03   0.841 0.400149
## ResidualSugar       4.860e-05  1.514e-04   0.321 0.748161
## Chlorides          -3.049e-02  1.611e-02  -1.893 0.058385 .
## FreeSulfurDioxide   8.167e-05  3.421e-05   2.387 0.016976 *
## TotalSulfurDioxide  8.197e-05  2.216e-05   3.699 0.000216 ***
## Density            -2.739e-01  1.918e-01  -1.428 0.153356
## pH                 -1.312e-02  7.517e-03  -1.745 0.081046 .
## Sulphates          -1.354e-02  5.481e-03  -2.470 0.013505 *
## Alcohol             3.216e-03  1.374e-03   2.341 0.019226 *
## LabelAppeal         1.594e-01  6.127e-03  26.023  < 2e-16 ***
## AcidIndex          -7.991e-02  4.576e-03 -17.463  < 2e-16 ***
## STARS2              3.223e-01  1.434e-02  22.478  < 2e-16 ***
## STARS3              4.409e-01  1.562e-02  28.230  < 2e-16 ***
## STARS4              5.553e-01  2.168e-02  25.618  < 2e-16 ***
## STARSNR            -7.665e-01  1.954e-02 -39.220  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 22861  on 12794  degrees of freedom
## Residual deviance: 13653  on 12777  degrees of freedom
## AIC: 45631
##
## Number of Fisher Scoring iterations: 6
```

- Possion Model with stepwise AIC approach

```
##
## Call:
## glm(formula = TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide +
##     TotalSulfurDioxide + Density + pH + Sulphates + Alcohol +
##     LabelAppeal + AcidIndex + STARS, family = "poisson", data = cleandf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2711  -0.6592  -0.0036   0.4499   3.7602
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.887e+00  1.951e-01   9.673  < 2e-16 ***
## VolatileAcidity    -3.054e-02  6.526e-03  -4.679 2.88e-06 ***
## Chlorides          -3.067e-02  1.611e-02  -1.904 0.056907 .
## FreeSulfurDioxide   8.211e-05  3.420e-05   2.401 0.016358 *
## TotalSulfurDioxide  8.209e-05  2.215e-05   3.705 0.000211 ***
## Density            -2.758e-01  1.918e-01  -1.438 0.150445
## pH                 -1.306e-02  7.516e-03  -1.737 0.082319 .
## Sulphates          -1.360e-02  5.479e-03  -2.481 0.013088 *
## Alcohol             3.235e-03  1.373e-03   2.356 0.018469 *
## LabelAppeal         1.595e-01  6.126e-03  26.032  < 2e-16 ***
## AcidIndex          -7.960e-02  4.520e-03 -17.609  < 2e-16 ***
```

```
## STARS2              3.225e-01  1.434e-02  22.495  < 2e-16 ***
## STARS3              4.409e-01  1.561e-02  28.237  < 2e-16 ***
## STARS4              5.555e-01  2.167e-02  25.633  < 2e-16 ***
## STARSNR            -7.666e-01  1.954e-02 -39.229  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 22861  on 12794  degrees of freedom
## Residual deviance: 13654  on 12780  degrees of freedom
## AIC: 45626
##
## Number of Fisher Scoring iterations: 6
```

Multiple Linear Regression Models

- MLR Model 1

```
##
## Call:
## lm(formula = TARGET ~ ., data = cleandf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8344 -0.8564  0.0228  0.8437  6.1635
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         5.064e+00  4.421e-01  11.456  < 2e-16 ***
## FixedAcidity        6.293e-04  1.859e-03   0.338 0.735042
## VolatileAcidity    -9.435e-02  1.478e-02  -6.383 1.79e-10 ***
## CitricAcid          1.710e-02  1.344e-02   1.273 0.203186
## ResidualSugar       1.534e-04  3.439e-04   0.446 0.655594
## Chlorides          -9.418e-02  3.639e-02  -2.588 0.009671 **
## FreeSulfurDioxide   2.463e-04  7.794e-05   3.160 0.001580 **
## TotalSulfurDioxide  2.366e-04  4.999e-05   4.732 2.25e-06 ***
## Density            -7.995e-01  4.359e-01  -1.834 0.066691 .
## pH                 -3.329e-02  1.702e-02  -1.956 0.050468 .
## Sulphates          -3.489e-02  1.244e-02  -2.804 0.005048 **
## Alcohol             1.112e-02  3.109e-03   3.577 0.000349 ***
## LabelAppeal         4.671e-01  1.363e-02  34.268  < 2e-16 ***
## AcidIndex          -2.001e-01  9.101e-03 -21.990  < 2e-16 ***
## STARS2              1.032e+00  3.258e-02  31.677  < 2e-16 ***
## STARS3              1.601e+00  3.766e-02  42.522  < 2e-16 ***
## STARS4              2.291e+00  5.969e-02  38.383  < 2e-16 ***
## STARSNR            -1.361e+00  3.293e-02 -41.335  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.306 on 12777 degrees of freedom
## Multiple R-squared:  0.5409, Adjusted R-squared:  0.5402
## F-statistic: 885.3 on 17 and 12777 DF,  p-value: < 2.2e-16
```

- MLR Model 2

```
##
## Call:
## lm(formula = TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide +
##     TotalSulfurDioxide + Density + pH + Sulphates + Alcohol +
##     LabelAppeal + AcidIndex + STARS, data = cleandf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8336 -0.8582  0.0234  0.8443  6.1570
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          5.073e+00  4.420e-01  11.476  < 2e-16 ***
## VolatileAcidity     -9.469e-02  1.478e-02  -6.408 1.53e-10 ***
## Chlorides           -9.479e-02  3.639e-02  -2.605 0.009204 **
## FreeSulfurDioxide    2.481e-04  7.791e-05   3.184 0.001454 **
## TotalSulfurDioxide   2.374e-04  4.997e-05   4.750 2.06e-06 ***
## Density             -8.077e-01  4.359e-01  -1.853 0.063883 .
## pH                  -3.334e-02  1.702e-02  -1.959 0.050115 .
## Sulphates           -3.508e-02  1.243e-02  -2.821 0.004798 **
## Alcohol              1.116e-02  3.107e-03   3.590 0.000332 ***
## LabelAppeal          4.671e-01  1.363e-02  34.272  < 2e-16 ***
## AcidIndex           -1.988e-01  8.943e-03 -22.231  < 2e-16 ***
## STARS2               1.033e+00  3.257e-02  31.703  < 2e-16 ***
## STARS3               1.602e+00  3.765e-02  42.535  < 2e-16 ***
## STARS4               2.292e+00  5.967e-02  38.407  < 2e-16 ***
## STARSNR             -1.362e+00  3.293e-02 -41.350  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.306 on 12780 degrees of freedom
## Multiple R-squared:  0.5408, Adjusted R-squared:  0.5403
## F-statistic:  1075 on 14 and 12780 DF,  p-value: < 2.2e-16
```

Negative Binomial Model

Since this is count data, we'll also try out a negative binomial model.

- NB model 1

```
##
## Call:
## glm.nb(formula = TARGET ~ ., data = cleandf, init.theta = 40718.61261,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2693  -0.6569  -0.0037   0.4499   3.7619
```

```
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.886e+00  1.951e-01   9.663  < 2e-16 ***
## FixedAcidity         7.929e-05  8.200e-04   0.097 0.922966
## VolatileAcidity     -3.042e-02  6.528e-03  -4.660 3.17e-06 ***
## CitricAcid           4.961e-03  5.897e-03   0.841 0.400160
## ResidualSugar        4.860e-05  1.514e-04   0.321 0.748136
## Chlorides           -3.049e-02  1.611e-02  -1.893 0.058389 .
## FreeSulfurDioxide    8.167e-05  3.421e-05   2.387 0.016977 *
## TotalSulfurDioxide   8.197e-05  2.216e-05   3.699 0.000216 ***
## Density             -2.739e-01  1.918e-01  -1.428 0.153367
## pH                  -1.312e-02  7.518e-03  -1.745 0.081040 .
## Sulphates           -1.354e-02  5.481e-03  -2.470 0.013506 *
## Alcohol              3.216e-03  1.374e-03   2.341 0.019236 *
## LabelAppeal          1.594e-01  6.127e-03  26.022  < 2e-16 ***
## AcidIndex           -7.991e-02  4.576e-03 -17.462  < 2e-16 ***
## STARS2               3.223e-01  1.434e-02  22.477  < 2e-16 ***
## STARS3               4.409e-01  1.562e-02  28.229  < 2e-16 ***
## STARS4               5.553e-01  2.168e-02  25.617  < 2e-16 ***
## STARSNR             -7.665e-01  1.954e-02 -39.219  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(40718.61) family taken to be 1)
##
##     Null deviance: 22860  on 12794  degrees of freedom
## Residual deviance: 13653  on 12777  degrees of freedom
## AIC: 45634
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  40719
##           Std. Err.:  34234
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -45595.54
```

- NB model 2

```
##
## Call:
## glm.nb(formula = TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide +
##     TotalSulfurDioxide + Density + pH + Sulphates + Alcohol +
##     LabelAppeal + AcidIndex + STARS, data = cleandf, init.theta = 40715.76818,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2710  -0.6592  -0.0036   0.4498   3.7600
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)         1.887e+00  1.951e-01   9.672  < 2e-16 ***
## VolatileAcidity    -3.054e-02  6.526e-03  -4.679 2.88e-06 ***
## Chlorides          -3.067e-02  1.611e-02  -1.904 0.056911 .
## FreeSulfurDioxide   8.211e-05  3.420e-05   2.401 0.016359 *
## TotalSulfurDioxide  8.209e-05  2.216e-05   3.705 0.000211 ***
## Density            -2.758e-01  1.918e-01  -1.438 0.150456
## pH                 -1.306e-02  7.517e-03  -1.737 0.082312 .
## Sulphates          -1.360e-02  5.479e-03  -2.481 0.013088 *
## Alcohol             3.235e-03  1.373e-03   2.356 0.018478 *
## LabelAppeal         1.595e-01  6.127e-03  26.030  < 2e-16 ***
## AcidIndex          -7.960e-02  4.521e-03 -17.609  < 2e-16 ***
## STARS2              3.225e-01  1.434e-02  22.494  < 2e-16 ***
## STARS3              4.409e-01  1.562e-02  28.235  < 2e-16 ***
## STARS4              5.555e-01  2.167e-02  25.631  < 2e-16 ***
## STARSNR            -7.666e-01  1.954e-02 -39.228  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(40715.77) family taken to be 1)
##
##     Null deviance: 22860  on 12794  degrees of freedom
## Residual deviance: 13653  on 12780  degrees of freedom
## AIC: 45628
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  40716
##           Std. Err.:  34232
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -45596.35
```

Zero-inflated Poisson Model

Since there are a number of zeros in the target variable, we'll try a zero-inflated poisson model.

```
##
## Call:
## zeroinfl(formula = TARGET ~ ., data = cleandf)
##
## Pearson residuals:
##       Min        1Q    Median        3Q       Max
## -2.325792 -0.419611 -0.003391  0.382557  5.438918
##
## Count model coefficients (poisson with log link):
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.525e+00  2.014e-01   7.570 3.74e-14 ***
## FixedAcidity        3.486e-04  8.406e-04   0.415  0.67837
## VolatileAcidity    -1.236e-02  6.712e-03  -1.842  0.06548 .
## CitricAcid          7.340e-04  6.020e-03   0.122  0.90295
## ResidualSugar      -8.070e-05  1.549e-04  -0.521  0.60234
## Chlorides          -2.032e-02  1.653e-02  -1.229  0.21894
```

```
## FreeSulfurDioxide   2.366e-05  3.457e-05   0.684  0.49378
## TotalSulfurDioxide -1.092e-05  2.202e-05  -0.496  0.61998
## Density            -2.718e-01  1.979e-01  -1.373  0.16967
## pH                  4.490e-03  7.715e-03   0.582  0.56059
## Sulphates          -1.567e-03  5.633e-03  -0.278  0.78085
## Alcohol             6.616e-03  1.403e-03   4.715 2.42e-06 ***
## LabelAppeal         2.320e-01  6.317e-03  36.721  < 2e-16 ***
## AcidIndex          -1.934e-02  4.898e-03  -3.948 7.88e-05 ***
## STARS2              1.295e-01  1.501e-02   8.626  < 2e-16 ***
## STARS3              2.241e-01  1.622e-02  13.815  < 2e-16 ***
## STARS4              3.118e-01  2.217e-02  14.064  < 2e-16 ***
## STARSNR            -6.370e-02  2.116e-02  -3.010  0.00261 **
##
## Zero-inflation model coefficients (binomial with logit link):
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -6.719e+00  1.328e+00  -5.061 4.17e-07 ***
## FixedAcidity        8.791e-04  5.476e-03   0.161 0.872469
## VolatileAcidity     1.847e-01  4.321e-02   4.275 1.91e-05 ***
## CitricAcid         -2.580e-02  3.950e-02  -0.653 0.513654
## ResidualSugar      -1.062e-03  1.014e-03  -1.048 0.294575
## Chlorides           3.621e-02  1.058e-01   0.342 0.732099
## FreeSulfurDioxide  -6.494e-04  2.347e-04  -2.766 0.005670 **
## TotalSulfurDioxide -9.435e-04  1.469e-04  -6.421 1.35e-10 ***
## Density             7.103e-01  1.302e+00   0.546 0.585367
## pH                  2.030e-01  4.993e-02   4.066 4.79e-05 ***
## Sulphates           1.226e-01  3.670e-02   3.342 0.000832 ***
## Alcohol             2.924e-02  9.204e-03   3.177 0.001490 **
## LabelAppeal         7.278e-01  4.249e-02  17.129  < 2e-16 ***
## AcidIndex           4.288e-01  2.618e-02  16.380  < 2e-16 ***
## STARS2             -3.651e+00  3.184e-01 -11.468  < 2e-16 ***
## STARS3             -1.841e+01  3.765e+02  -0.049 0.960991
## STARS4             -1.854e+01  7.028e+02  -0.026 0.978951
## STARSNR             2.075e+00  7.650e-02  27.127  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 41
## Log-likelihood: -2.038e+04 on 36 Df


##
## Call:
## zeroinfl(formula = TARGET ~ VolatileAcidity + FreeSulfurDioxide + TotalSulfurDioxide +
##     pH + Sulphates + Alcohol + LabelAppeal + AcidIndex + STARS, data = cleandf)
##
## Pearson residuals:
##       Min        1Q    Median        3Q       Max
## -2.320830 -0.419217 -0.004949  0.383814  5.352351
##
## Count model coefficients (poisson with log link):
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.255e+00  5.040e-02  24.908  < 2e-16 ***
## VolatileAcidity    -1.249e-02  6.709e-03  -1.861  0.06268 .
## FreeSulfurDioxide   2.329e-05  3.453e-05   0.674  0.50005
## TotalSulfurDioxide -1.170e-05  2.201e-05  -0.532  0.59488
```

```
## pH                   4.604e-03  7.712e-03   0.597  0.55052
## Sulphates           -1.414e-03  5.630e-03  -0.251  0.80176
## Alcohol              6.677e-03  1.402e-03   4.761 1.93e-06 ***
## LabelAppeal          2.319e-01  6.316e-03  36.717  < 2e-16 ***
## AcidIndex           -1.942e-02  4.841e-03  -4.011 6.04e-05 ***
## STARS2               1.293e-01  1.501e-02   8.619  < 2e-16 ***
## STARS3               2.244e-01  1.622e-02  13.835  < 2e-16 ***
## STARS4               3.119e-01  2.217e-02  14.069  < 2e-16 ***
## STARSNR             -6.402e-02  2.116e-02  -3.026  0.00248 **
##
## Zero-inflation model coefficients (binomial with logit link):
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -6.019e+00  3.065e-01 -19.638  < 2e-16 ***
## VolatileAcidity      1.855e-01  4.322e-02   4.291 1.78e-05 ***
## FreeSulfurDioxide   -6.644e-04  2.344e-04  -2.835 0.004586 **
## TotalSulfurDioxide -9.494e-04  1.468e-04  -6.467 9.97e-11 ***
## pH                   2.038e-01  4.991e-02   4.082 4.46e-05 ***
## Sulphates            1.227e-01  3.669e-02   3.343 0.000827 ***
## Alcohol              2.919e-02  9.196e-03   3.174 0.001501 **
## LabelAppeal          7.278e-01  4.246e-02  17.140  < 2e-16 ***
## AcidIndex            4.288e-01  2.558e-02  16.760  < 2e-16 ***
## STARS2              -3.663e+00  3.206e-01 -11.425  < 2e-16 ***
## STARS3              -1.841e+01  3.777e+02  -0.049 0.961124
## STARS4              -1.855e+01  7.034e+02  -0.026 0.978964
## STARSNR              2.077e+00  7.646e-02  27.160  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 31
## Log-likelihood: -2.038e+04 on 26 Df
```

Select Models

In this section, an optimal model will be selected based on its performance when trained on the data. To select the models, we'll use AIC and MSE to measure accuracy of the predicted values.

Below, the Poisson, multiple linear regression, negative binomial, and zero-inflated models have been compared to select the model with the lowest AIC.

Comparison of Poisson Models

We'll need to compare the AIC's of each Poisson Model.

Poisson Model 1:

```
## [1] 45631.12
```

Poisson Model 2:

```
## [1] 45625.93
```

Poisson Model 2 proves to have the lower AIC of the two, with a 33947.74 AIC. Below is the formula for Poisson Model 2.

```
## [[1]]
## TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide + TotalSulfurDioxide +
##     Density + pH + Sulphates + Alcohol + LabelAppeal + AcidIndex +
##     STARS
```

Comparsion of Multiple Linear Models

We'll need to compare the Adjusted R Squares of each Linear Model.

Linear Model 1:

```
## [1] 0.5402418
```

Linear Model 2:

```
## [1] 0.5402805
```

Linear Model 2 proves to have the higher Adjusted R Squares, with a value of 0.540473. Below is the formula for Linear Model 2.

```
## [[1]]
## TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide + TotalSulfurDioxide +
##     Density + pH + Sulphates + Alcohol + LabelAppeal + AcidIndex +
##     STARS
```

Comparsion of Negative Binomial Models

Now we'll compare the AICs of the two negative binomial models.

NB Model 1:

```
## [1] 45633.54
```

NB Model 2:

```
## [1] 45628.35
```

The AIC of model 2 is lower than that of model 1.

Comparsion of Zero-inflated Poisson Models

Now we'll compare the AICs of the two zero-inflated binomial models.

ZI Model 1:

```
## [1] 40831.52
```

ZI Model 2:

```
## [1] 40817.64
```

The AIC of model 2 is lower than that of model 1.

Mean Square Error

The Mean Square Error measures the averaged square different between the estimated values and the actual value. The lower the value of the MSE, the more accurately the model is able to predict the values.

$$\text{MSE} = \frac{1}{n}\sum(y - \hat{y})^2$$

Comparison of Models

By evaluating the AIC's and MSE's of each model, we can choose the best one be looking at the lowest AIC and lowest MSE.

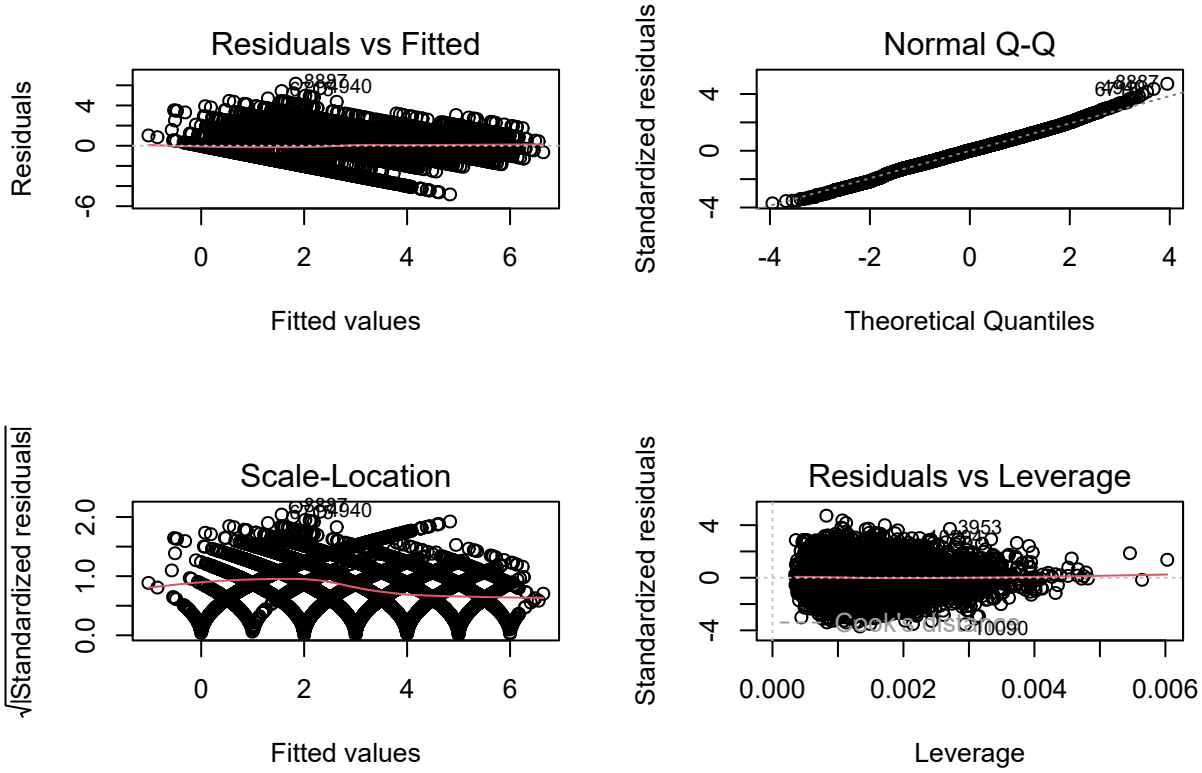
Possion Model 1	Possion Model 2	Linear Model 1	Linear Model 2
6.7	6.7	1.7	1.7
45631.1	45625.9	43165.9	43161.9

Neg Binom Model 1	Neg Binom Model 2	Zero-Infl Model 1	Zero-Infl Model 2
6.7	6.7	1.6	1.6
45633.5	45628.4	40831.5	40817.6

Based on the above, the linear model has better model statistics than the poisson and negative binomial models, but the zero-inflated poison is better overall. We'll check the accuracy of each model in a later section, but for now we'll evaluate the modesl for validity in case we want to use them for our predictions.

While the linear model may have better statistics than the poisson and binomial, we'll need to validate whether the assumptions of the linear model hold: 1) Homoschedastic residuals 2) Normally distributed residuals 3) Independence of predictors 4) Linearity of response

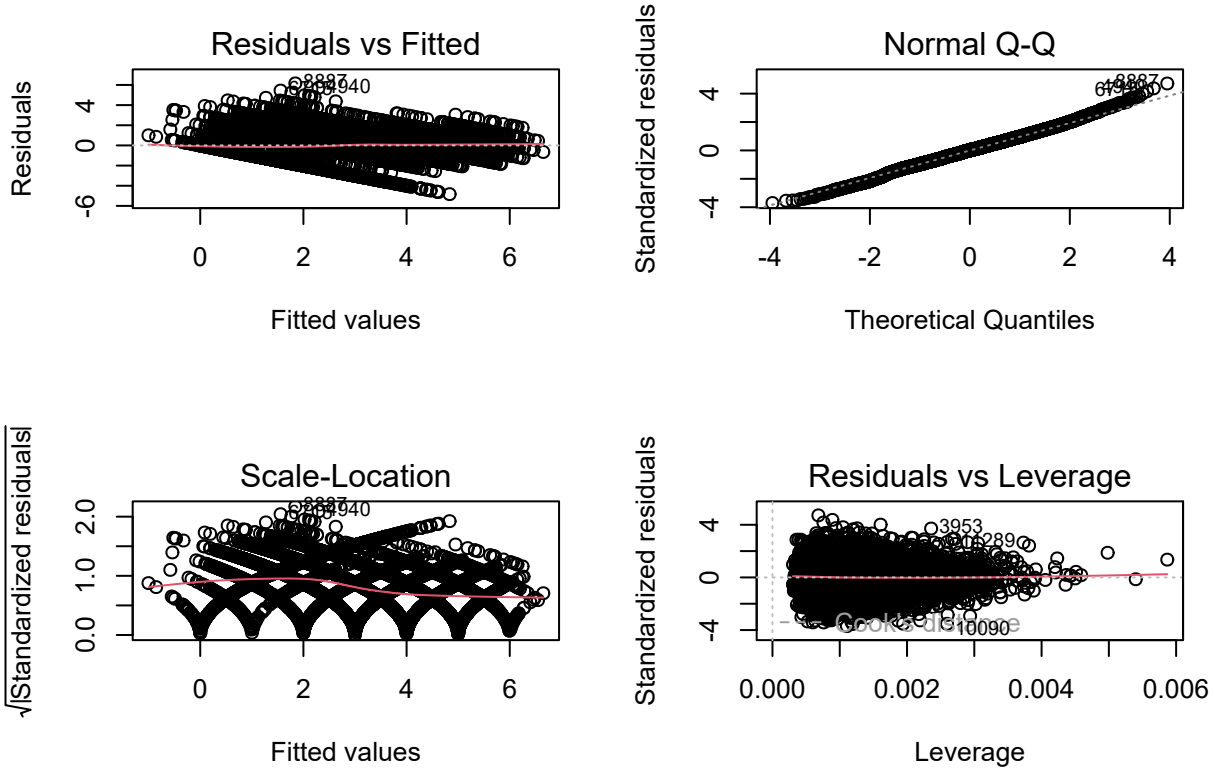
```
## [1] "------------------------------------------------"
## lm(formula = TARGET ~ ., data = cleandf)
```




```
## [1] ""
## [1] "Shapiro test for normality of residuals cannot be performed; sample length must be between 3 an
## [1] "Breusch-Pagan test for homoschedasticity: The p-value of 0 is <= 0.05 and the test statistic is
## [1] ""
## [1] "Variance inflation factor (VIF)"
## [1] "<=1: not correlated, 1-5: moderately correlated, >5: strongly correlated"
##  [1] 4.000000 1.158627 1.106445 1.088710 1.051877 1.043412 1.034711 1.018575
##  [9] 1.017208 1.007600 1.006926 1.006610 1.005576 1.005375 1.004908 1.003808
## [17] 1.003793 1.003698 1.003595 1.003457 1.003300 1.002784 1.002684 1.002451
## [25] 1.002201 1.001902 1.001847 1.001796 1.001100 1.000000 1.000000 1.000000
## [33] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [41] 1.000000 1.000000
## [1] ""
## [1] "Model scores:"
## [1] "    adjusted R-squared: 0.54"
## [1] "    AIC: 43165.949"
## [1] "    BIC: 43307.628"
## [1] "    Mallow's Cp: 12"
## [1] "    mean squared error: 1.704"
## [1] ""
## [1] "Leverage point cutoff: 0.00296991012114107"
## [1] ""
## [1] "First 10 points of influence:"
## [1] "    case #11: 0.003"
## [1] "    case #28: 0.003"
## [1] "    case #51: 0.003"
```

```
## [1] "    case #138: 0.004"
## [1] "    case #177: 0.003"
## [1] "    case #274: 0.003"
## [1] "    case #412: 0.003"
## [1] "    case #516: 0.003"
## [1] "    case #554: 0.004"
## [1] "    case #560: 0.004"
## [1] "    case #600: 0.004"
## [1] ""

## [1] "-------------------------------------------------"
## lm(formula = TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide +
##     TotalSulfurDioxide + Density + pH + Sulphates + Alcohol +
##     LabelAppeal + AcidIndex + STARS, data = cleandf)
```




```
## [1] ""
## [1] "Shapiro test for normality of residuals cannot be performed; sample length must be between 3 an
## [1] "Breusch-Pagan test for homoschedasticity: The p-value of 0 is <= 0.05 and the test statistic is
## [1] ""
## [1] "Variance inflation factor (VIF)"
## [1] "<=1: not correlated, 1-5: moderately correlated, >5: strongly correlated"
##  [1] 4.000000 1.157074 1.106337 1.051826 1.051338 1.025348 1.018404 1.006956
##  [9] 1.006570 1.005474 1.004716 1.004211 1.003543 1.003472 1.003323 1.003280
## [17] 1.002902 1.002733 1.002355 1.002103 1.001770 1.001660 1.001450 1.000000
## [25] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
```

```
## [33] 1.000000
## [1] ""
## [1] "Model scores:"
## [1] "    adjusted R-squared: 0.54"
## [1] "    AIC: 43161.876"
## [1] "    BIC: 43281.184"
## [1] "    Mallow's Cp: 9"
## [1] "    mean squared error: 1.704"
## [1] ""
## [1] "Leverage point cutoff: 0.0025009769441188"
## [1] ""
## [1] "First 10 points of influence:"
## [1] "    case #27: 0.003"
## [1] "    case #28: 0.003"
## [1] "    case #32: 0.003"
## [1] "    case #51: 0.003"
## [1] "    case #138: 0.003"
## [1] "    case #156: 0.003"
## [1] "    case #274: 0.003"
## [1] "    case #442: 0.003"
## [1] "    case #450: 0.003"
## [1] "    case #465: 0.003"
## [1] "    case #495: 0.003"
## [1] ""
```

As shown, the residuals are heteroschedastic and exhibit a clearly defined pattern. While the Shapiro test for normality couldn't be performed due to the sample size, the QQ plot shows visually that the residuals are not normally distributed.

Now we'll see if the Poisson models exhibit any overdispersion, which could lead us to invalidate the models and go with a different model instead. This can be done by dividing the residual deviance by the degrees of freedom.

```
## [1] "Poisson model 1 overdispersion: 1.06856809632371"
```

```
## [1] "Poisson model 2 overdispersion: 1.06838108649939"
```

Since the overdispersion parameter isn't much greater than 1 (generally, less than 1.10), this suggests that the poisson model is a good fit, and we can be relatively confident that we don't need to go with the negative binomial.

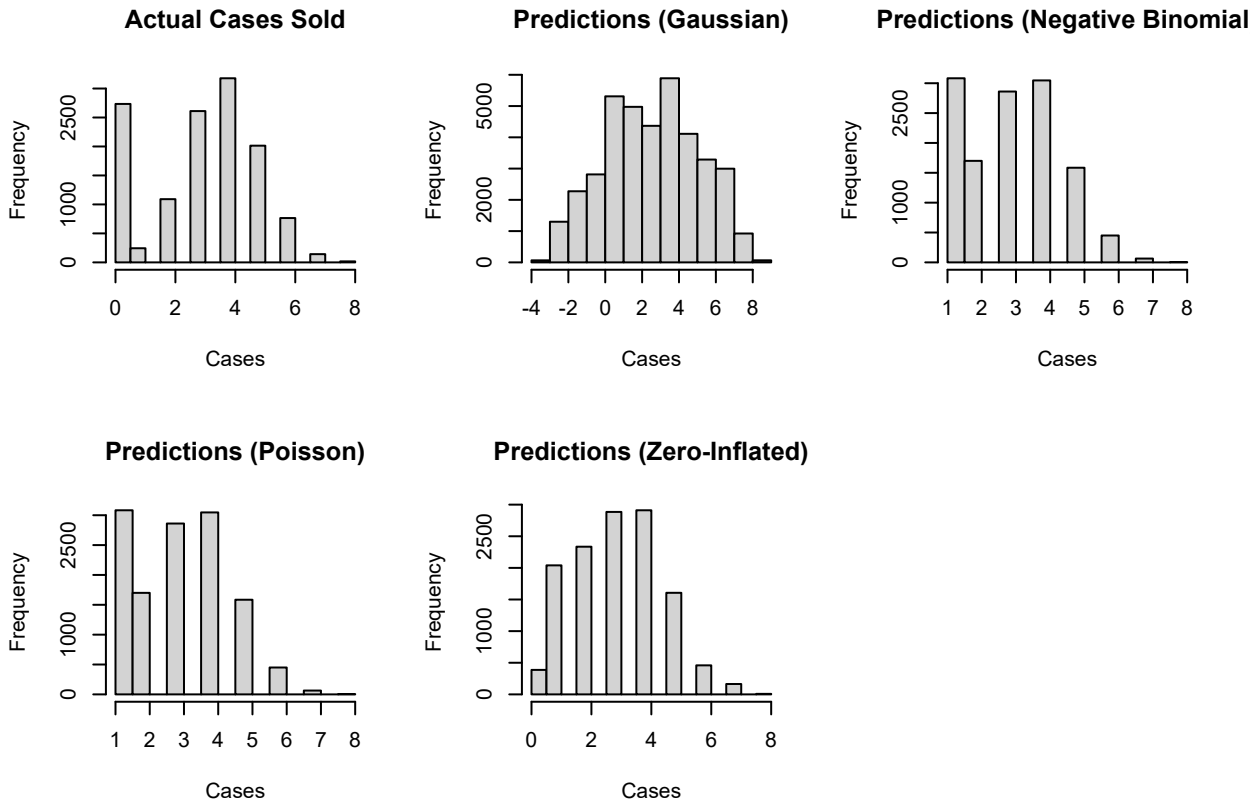
Now we'll check the model accuracy by predicting the number of cases sold based on our model parameters. Since we're dealing with whole cases of wine, we'll round the prediction to the nearest integer.

```
## [1] "Gaussian accuracy: 5004 of 12795 (39.1%)"
```

```
## [1] "Negative binomial model accuracy: 3771 of 12795 (29.5%)"
```

```
## [1] "Poisson accuracy: 3771 of 12795 (29.5%)"
```

```
## [1] "Zero-inflated poisson accuracy: 4430 of 12795 (34.6%)"
```



None of the models performs very well. Because of the zero counts, we'll take a different tack rather than a simple single-model approach. Instead, we'll try doing a hierarchical model, in which we'll first model zero counts using a binomial model, then we'll model positive counts with a different count-based model.

Summary of binomial model 1:

```
##
## Call:
## glm(formula = pos_cases ~ . - TARGET, family = binomial(), data = cleandf_binom)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1703   0.0001   0.1591   0.4743   2.5444
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          5.884e+00  1.101e+00   5.345 9.03e-08 ***
## FixedAcidity         2.408e-04  4.578e-03   0.053 0.958042
## VolatileAcidity     -1.822e-01  3.646e-02  -4.999 5.77e-07 ***
## CitricAcid           3.023e-02  3.322e-02   0.910 0.362775
## ResidualSugar        9.570e-04  8.548e-04   1.120 0.262897
## Chlorides           -8.527e-02  8.934e-02  -0.954 0.339845
## FreeSulfurDioxide    5.224e-04  1.952e-04   2.676 0.007441 **
## TotalSulfurDioxide   8.286e-04  1.232e-04   6.723 1.78e-11 ***
## Density             -6.692e-01  1.082e+00  -0.619 0.536088
## pH                  -1.799e-01  4.197e-02  -4.286 1.82e-05 ***
## Sulphates           -1.019e-01  3.078e-02  -3.310 0.000932 ***
```

```
## Alcohol            -2.177e-02  7.691e-03  -2.831 0.004637 **
## LabelAppeal        -4.686e-01  3.330e-02 -14.074  < 2e-16 ***
## AcidIndex          -3.894e-01  2.183e-02 -17.837  < 2e-16 ***
## STARS2              2.432e+00  1.195e-01  20.348  < 2e-16 ***
## STARS3              1.841e+01  2.203e+02   0.084 0.933378
## STARS4              1.854e+01  4.206e+02   0.044 0.964834
## STARSNR            -1.825e+00  6.142e-02 -29.711  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13275.8  on 12794  degrees of freedom
## Residual deviance:  7616.3  on 12777  degrees of freedom
## AIC: 7652.3
##
## Number of Fisher Scoring iterations: 18
```

Summary of binomial model 2:

```
##
## Call:
## glm(formula = pos_cases ~ VolatileAcidity + FreeSulfurDioxide +
##     TotalSulfurDioxide + pH + Sulphates + Alcohol + LabelAppeal +
##     AcidIndex + STARS, family = binomial(), data = cleandf_binom)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1778   0.0001   0.1597   0.4749   2.5852
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         5.226e+00  2.507e-01  20.841  < 2e-16 ***
## VolatileAcidity    -1.825e-01  3.645e-02  -5.008 5.51e-07 ***
## FreeSulfurDioxide   5.357e-04  1.949e-04   2.748 0.005999 **
## TotalSulfurDioxide  8.366e-04  1.231e-04   6.796 1.08e-11 ***
## pH                 -1.806e-01  4.195e-02  -4.305 1.67e-05 ***
## Sulphates          -1.021e-01  3.077e-02  -3.319 0.000903 ***
## Alcohol            -2.180e-02  7.686e-03  -2.836 0.004568 **
## LabelAppeal        -4.689e-01  3.328e-02 -14.091  < 2e-16 ***
## AcidIndex          -3.887e-01  2.140e-02 -18.164  < 2e-16 ***
## STARS2              2.434e+00  1.195e-01  20.369  < 2e-16 ***
## STARS3              1.841e+01  2.203e+02   0.084 0.933396
## STARS4              1.855e+01  4.208e+02   0.044 0.964840
## STARSNR            -1.826e+00  6.136e-02 -29.764  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13275.8  on 12794  degrees of freedom
## Residual deviance:  7619.7  on 12782  degrees of freedom
## AIC: 7645.7
##
```

```
## Number of Fisher Scoring iterations: 18
```

AIC of binomial model 1:

```
## [1] 7652.3
```

AIC of binomial model 2:

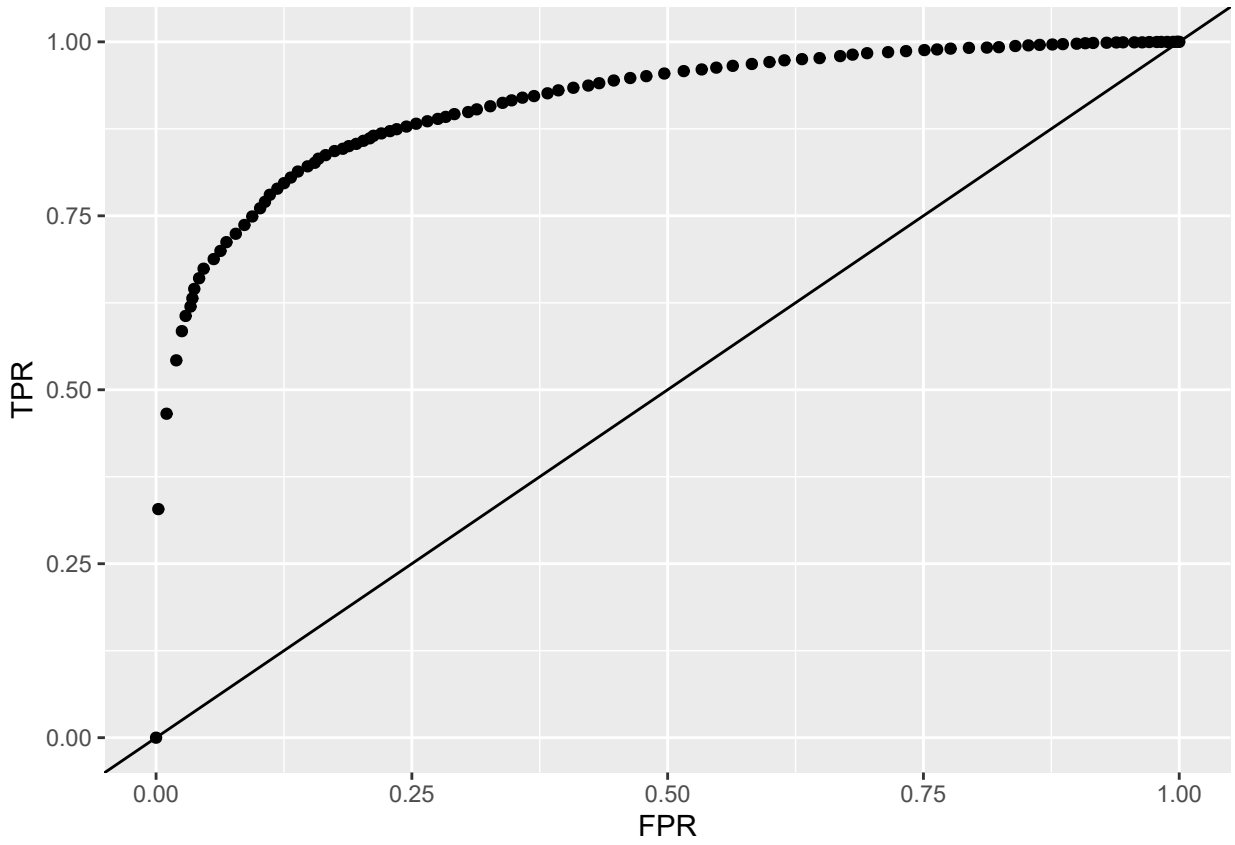
```
## [1] 7645.727
```

Since the AIC of the step-reduced binomial model is lower, we'll use it to predict whether any cases of each wine variety were sold. Then we'll check the accuracy of the predictions against the training data.

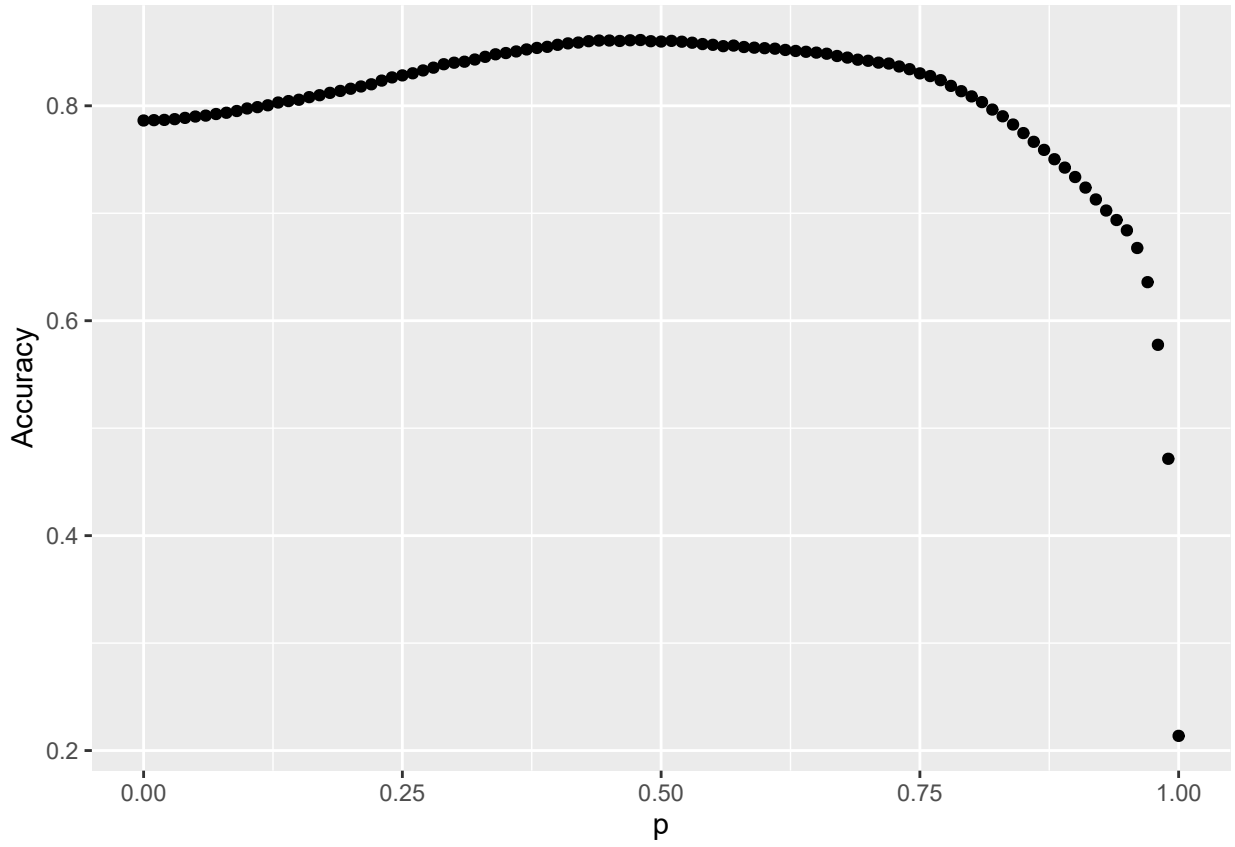
```
## [1] "Binomial model accuracy with p of 0.5: 11001 of 12795 (86%)"
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
```

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



```
##
## [[2]]
```




```
##
## [[3]]
## [1] -0.9112545
##
## [[4]]
## [1] 0.8611958
##
## [[5]]
## [1] 0.48

## [1] 0.48

## [1] "Binomial model accuracy with p of 0.48: 11019 of 12795 (86.1%)"
```

Since the accuracy is very good, we'll use that to predict zero counts, then we'll use a separate count-based model to predict positive case counts.

Start with poisson modeling:

```
## [1] "AIC of poisson full model: 45631.1162685305"

## [1] "AIC of poisson step-reduced model: 45625.9319872646"
```

Check for overdispersion of the poisson models:

```
## [1] "Poisson full model overdispersion: 1.06856809632371"
```

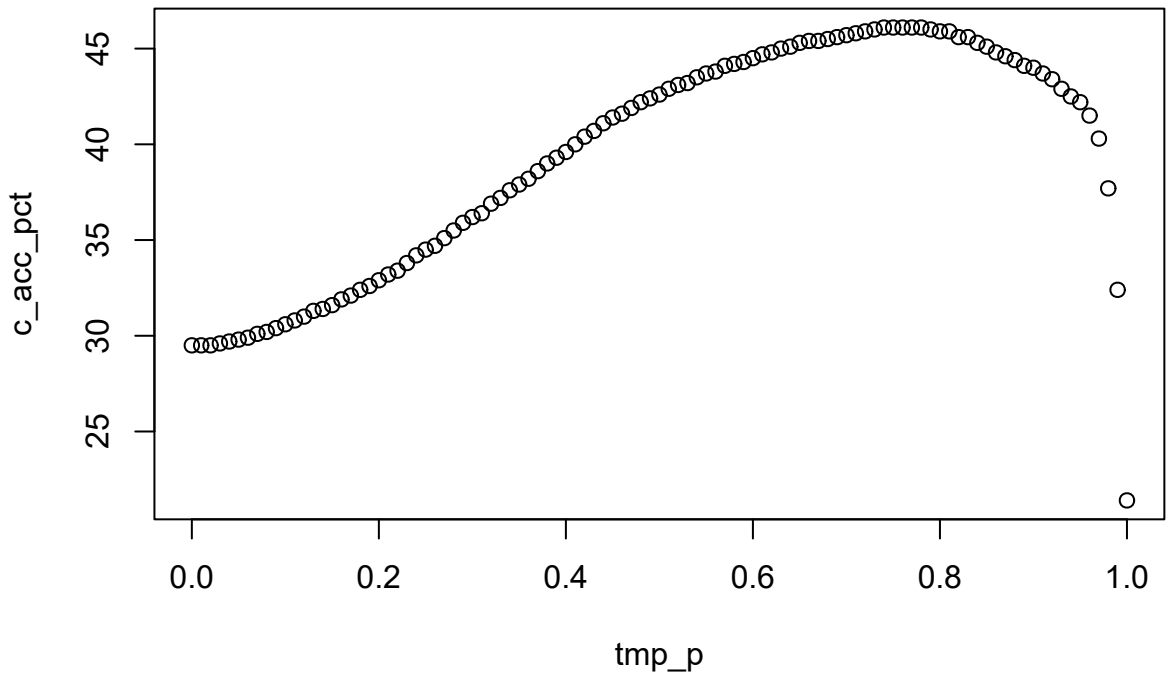
```
## [1] "Poisson step-reduced model overdispersion: 1.06838108649939"
```

Neither of these is much greater than one, but we'll try negative binomial modeling to see if it yields a better result anyway.

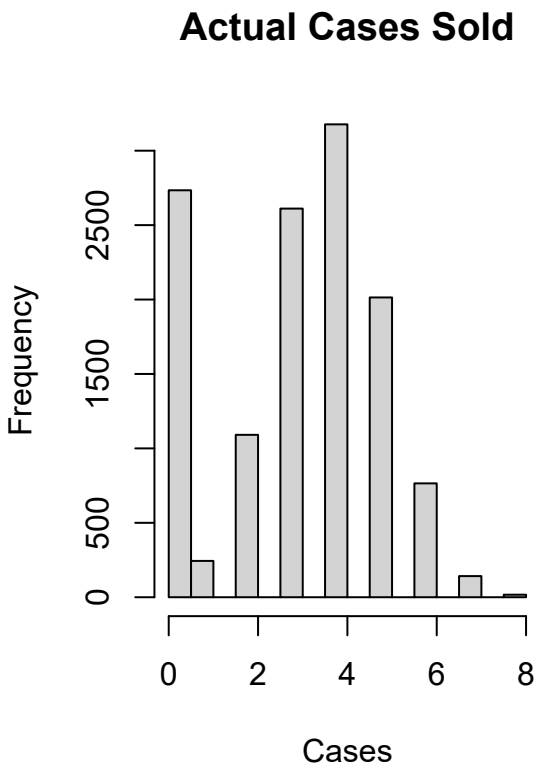
```
## [1] "AIC of negative binomial full model: 45633.535456137"
```

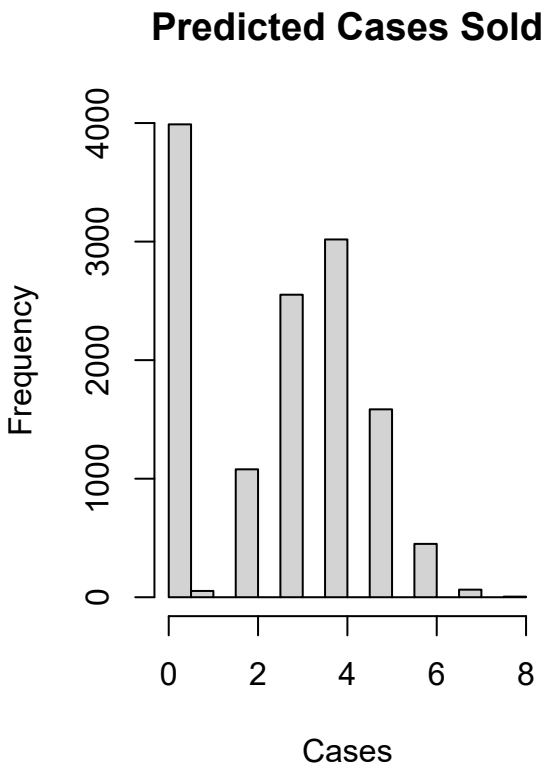
```
## [1] "AIC of negative binomial step-reduced model: 45628.3511872782"
```

Since the AIC of the poisson step-reduced model is lowest, we'll use it to predict cases sold against the training data. First, we'll vary the cutoff value of p. Then we'll use a poisson model against the non-zero cases and generate an overall accuracy for that cutoff value. Then we'll choose the cutoff value that yields the maximum accuracy, and we'll use this to generate our predictions against the evaluation data.




```
## [1] "Max accuracy is 46.1% when p=0.74"
```

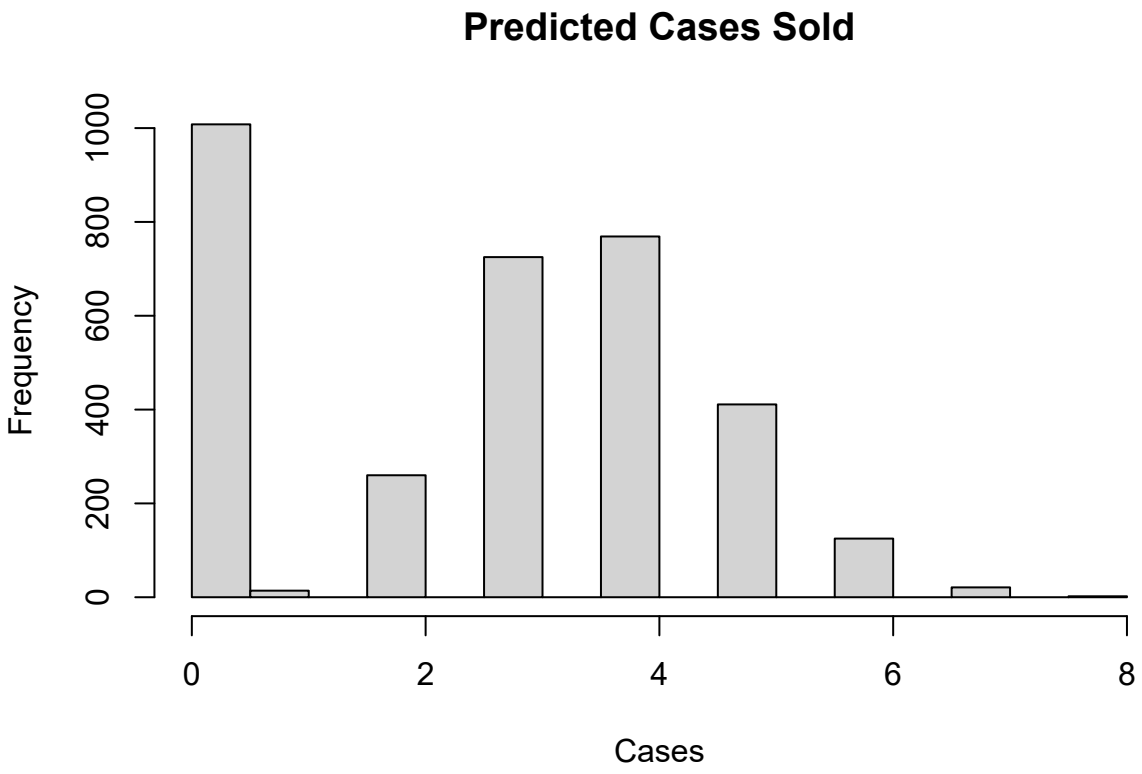





```
## [1] "Close prediction accuracy (within one case of actual): 78.3%"
```

Now we'll use this cutoff value against the evaluation data.

```
## # A tibble: 10 x 17
##    Fixed~1 Volat~2 Citri~3 Resid~4 Chlor~5 FreeS~6 Total~7 Density    pH Sulph~8
##      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
##  1     5.4   -0.86    0.27   -10.7   0.092      23     398   0.985  5.02    0.64
##  2     7.2    1.75    0.17   -33     0.065       9      76   1.05   4.61    0.68
##  3    11.4    0.21    0.28     1.2   0.038      70      53   1.03   2.54   -0.07
##  4    15.9    1.19    1.14    31.9  -0.299     115     381   1.03   2.99    0.31
##  5    11.6    0.32    0.55   -50.9   0.076      35      83   1.00   3.32    2.18
##  6     3.8    0.22    0.31    -7.7   0.039      40     129   0.906  4.72   -0.64
##  7     9     -0.21    0.04    51.4   0.237    -213    -527   0.995  3.16    0.7
##  8    13      0.21    0.32    -3.2  -0.263     111     141   0.959  3.2     1.78
##  9    17.9   -0.42   -0.91     7.1   0.045    -177     169   0.953  3.17   -1.12
## 10    11.7    1.18   -0.94   -62     0.675       7    -393   1.00   3.96    0.69
## # ... with 7 more variables: Alcohol <dbl>, LabelAppeal <int>, AcidIndex <int>,
## #   STARS <chr>, TARGET <lgl>, pred_p <dbl>, pred_target <dbl>, and abbreviated
## #   variable names 1: FixedAcidity, 2: VolatileAcidity, 3: CitricAcid,
## #   4: ResidualSugar, 5: Chlorides, 6: FreeSulfurDioxide,
## #   7: TotalSulfurDioxide, 8: Sulphates
```



Conclusion

We attempted several models: Gaussian linear, poisson, negative binomial, and zero-inflated poisson. Based on the low performance of these models, we decided to use a hierarchical approach. We first modeled zero counts vs non-zero counts using a binomial model, then modeled the counts of the non-zeroes using a Poisson model. While the binomial model performed well (86.1% accuracy) predicting zero vs non-zero counts, we were only able to obtain an accuracy of 46.1% overall. However, the histograms of the actual vs predicted cases sold are very similar, indicating that, while the exact numbers may not be completely accurate, the numbers are generally within the same range. This is confirmed by the fact that the accuracy is 78% of predicting the number of cases within one case of the actual value.

Appendix

```
# Load Libraries:
library(tidyverse)
library(dplyr)
library(corrplot)
library(skimr)
library(DataExplorer)
library(ggplot2)
```

```
library(hrbrthemes)
library(mice)
library(MASS)
library(dvmisc)
library(gridExtra)
library(lattice)
library(faraway)
library(pscl)

# Load Data set:
dftrain <- read.csv("https://raw.githubusercontent.com/letisalba/Data_621/master/Homework_5/csv/wine-tra
dfeval <- read.csv("https://raw.githubusercontent.com/letisalba/Data_621/master/Homework_5/csv/wine-eval
head(dftrain)
summary(dftrain)
summary(dfeval)
DataExplorer::create_report(dftrain, output_file = "training_report.html")
DataExplorer::create_report(dfeval %>%
    dplyr::select(-TARGET), output_file = "eval_report.html")
str(dftrain)
str(dfeval)
plot_train <- dftrain %>%
    gather(key = "variable", value = "value")
ggplot(plot_train) + geom_histogram(aes(x = value, y = ..density..),
    bins = 30) + geom_density(aes(x = value), color = "blue") +
    theme_ipsum() + facet_wrap(. ~ variable, scales = "free",
    ncol = 4)

# Create logical variable to indicate whether there is a
# star rating for this wine
dftrain <- dftrain %>%
    mutate(STARS = ifelse(is.na(STARS), "NR", STARS))
dfeval <- dfeval %>%
    mutate(STARS = ifelse(is.na(STARS), "NR", STARS))

# Look at cases sold vs predictors
plt <- vector("list", ncol(dftrain) - 1)
for (i in seq(3, 16)) {
    # skip INDEX and TARGET variables
    if (class(dftrain[, i]) == "numeric") {
        tmpmin <- min(dftrain[, i], na.rm = T)
        tmpinterval <- (max(dftrain[, i], na.rm = T) - tmpmin)/5
        tmpcuts <- c()
        for (j in seq(1, 5)) {
            tmpcuts <- c(tmpcuts, tmpmin + (j * tmpinterval))
        }
        # dftrain$x <- dftrain[, i] %>% cut(breaks=5,
        # ordered_result=T, right=F)
        dftrain$x <- dftrain[, i] %>%
            cut(breaks = tmpcuts, ordered_result = T, right = F)
    } else {
        dftrain$x <- dftrain[, i]
    }
    dftmp <- dftrain %>%
```

```
        group_by(x) %>%
        summarize(ct = sum(TARGET))
    plt[[i]] <- barchart(dftmp$ct ~ dftmp$x, horiz = F, col = "darkgreen",
        xlab = colnames(dftrain)[i], ylab = "Cases")
}
dftrain <- subset(dftrain, select = -x)  # remove temporary variable
grid.arrange(grobs = plt[3:7], ncol = 3, nrow = 2)
grid.arrange(grobs = plt[8:13], ncol = 3, nrow = 2)
grid.arrange(grobs = plt[14:16], ncol = 3, nrow = 2)

# Remove index columns
dftrain <- dftrain %>%
    dplyr::select(-INDEX)
dfeval <- dfeval %>%
    dplyr::select(-IN)

# Impute missing values in training data (except for STARS)
dftrain_imputed <- mice(dftrain %>%
    dplyr::select(-STARS), m = 5, maxit = 5, method = "pmm")
cleandf <- complete(dftrain_imputed) %>%
    mutate(STARS = dftrain$STARS)

# Impute missing values in eval data (except for STARS and
# TARGET)
dfeval_imputed <- mice(dfeval %>%
    dplyr::select(-STARS, -TARGET), m = 5, maxit = 5, method = "pmm")
cleandf_eval <- complete(dfeval_imputed) %>%
    mutate(STARS = dfeval$STARS, TARGET = dfeval$TARGET)

# Look again at summary
summary(cleandf)
summary(cleandf_eval)

# Possion Model 1
p_mod1 <- glm(TARGET ~ ., family = "poisson", data = cleandf)
summary(p_mod1)

# Possion Model with stepwise AIC approach
p_mod2 <- stepAIC(p_mod1, trace = F)
summary(p_mod2)

# MLR Model 1
lm_mod1 <- lm(TARGET ~ ., data = cleandf)
aic_lm_mod1 = AIC(lm_mod1)
summary(lm_mod1)

# MLR Model 2
lm_mod2 <- stepAIC(lm_mod1, trace = F)
aic_lm_mod2 = AIC(lm_mod2)
summary(lm_mod2)

# NB model 1
nb_mod1 <- glm.nb(TARGET ~ ., data = cleandf)
```

```r
aic_nb_mod1 = AIC(nb_mod1)
summary(nb_mod1)

# NB model 2
nb_mod2 <- stepAIC(nb_mod1, trace = F)
aic_nb_mod2 = AIC(nb_mod2)
summary(nb_mod2)

# Zero-inflated Poisson Model
zi_mod1 <- zeroinfl(TARGET ~ ., data = cleandf)
summary(zi_mod1)
aic_zi_mod1 = AIC(zi_mod1)
zi_mod2 <- stepAIC(zi_mod1, trace = F)  # this takes a long time to run
summary(zi_mod2)
aic_zi_mod2 = AIC(zi_mod2)

# Poisson Model 1:
aic_p_mod1 <- p_mod1$aic
aic_p_mod1

# Poisson Model 2:
aic_p_mod2 <- p_mod2$aic
aic_p_mod2

# Poisson - Minimum AIC
c(p_mod1$formula, p_mod2$formula)[which.min(c(p_mod1$aic, p_mod2$aic))]

# Linear Model 1:
r2_lm_mod1 <- summary(lm_mod1)$adj.r.squared
r2_lm_mod1

# Linear Model 2:
r2_lm_mod2 <- summary(lm_mod2)$adj.r.squared
r2_lm_mod2

# Multiple Linear Regression Model - Highest Adjusted R
# Squared
c(formula(lm_mod1), formula(lm_mod2))[which.max(c(summary(lm_mod1)$adj.r.squared,
    summary(lm_mod2)$adj.r.squared))]

# NB Model 1:
aic_nb_mod1

# NB Model 2:
aic_nb_mod2

# ZI Model 1:
aic_zi_mod1

# ZI Model 2:
aic_zi_mod2

# Mean Square Error:
```

```r
mse <- function(df, model) {
    mean((df$TARGET - predict(model))^2)
}
mse_p_mod1 <- mse(cleandf, p_mod1)
mse_p_mod2 <- mse(cleandf, p_mod2)
mse_lm_mod1 <- get_mse(lm_mod1)
mse_lm_mod2 <- get_mse(lm_mod2)
mse_nb_mod1 <- mse(cleandf, nb_mod1)
mse_nb_mod2 <- mse(cleandf, nb_mod2)
mse_zi_mod1 <- mse(cleandf, zi_mod1)
mse_zi_mod2 <- mse(cleandf, zi_mod2)

# Comparison of Models:
models <- c("Possion Model 1", "Possion Model 2", "Linear Model 1",
    "Linear Model 2", "Neg Binom Model 1", "Neg Binom Model 2",
    "Zero-Infl Model 1", "Zero-Infl Model 2")
MSE <- round(c(mse_p_mod1, mse_p_mod2, mse_lm_mod1, mse_lm_mod2,
    mse_nb_mod1, mse_nb_mod2, mse_zi_mod1, mse_zi_mod2), 1)
AIC <- round(c(aic_p_mod1, aic_p_mod2, aic_lm_mod1, aic_lm_mod2,
    aic_nb_mod1, aic_nb_mod2, aic_zi_mod1, aic_zi_mod2), 1)
knitr::kable(rbind(MSE[1:4], AIC[1:4]), col.names = models[1:4])
knitr::kable(rbind(MSE[5:8], AIC[5:8]), col.names = models[5:8])

# LM model evaluation

# Load libraries
library(car)
library(lmtest)
library(olsrr)

# Define function to calculate mean squared error
calc_mse <- function(lmod) {
    return(mean((summary(lmod))$residuals^2))
}

# Define function to aid in model analysis
ModelAnalysis <- function(lmod) {

    # Plot residuals
    print("----------------------------------------------------")
    print(lmod$call)
    par(mfrow = c(2, 2))
    plot(lmod)
    print("")

    # Shapiro test to determine normality of residuals Null
    # hypothesis: the residuals are normal.  If the p-value
    # is small, reject the null, i.e., consider the
    # residuals *not* normally distributed.
    if (length(lmod$fitted.values) > 3 & length(lmod$fitted.values) <
        5000) {
        st <- shapiro.test(lmod$residuals)
        if (st$p.value <= 0.05) {
```

```
        print(paste0("Shapiro test for normality: The p-value of ",
            st$p.value, " is <= 0.05, so reject the null; i.e., the residuals are NOT NORMAL"))
    } else {
        print(paste0("Shapiro test for normality: The p-value of ",
            st$p.value, " is > 0.05, so do not reject the null; i.e., the residuals are NORMAL"))
    }
    print("")
} else {
    print("Shapiro test for normality of residuals cannot be performed; sample length must be betwe
}

# Breusch-Pagan test to determine homoschedasticity of
# residuals Null hypothesis: the residuals are
# homoschedastic.  If the p-value is small, reject the
# null, i.e., consider the residuals heteroschedastic.
bp <- bptest(lmod)
if (bp$p.value > 0.05 & bp$statistic < 10) {
    print(paste0("Breusch-Pagan test for homoschedasticity: The p-value of ",
        bp$p.value, " is > 0.05 and the test statistic of ",
        bp$statistic, " is < 10, so don't reject the null; i.e., the residuals are HOMOSCHEDASTIC"))
} else if (bp$p.value <= 0.05) {
    print(paste0("Breusch-Pagan test for homoschedasticity: The p-value of ",
        bp$p.value, " is <= 0.05 and the test statistic is ",
        bp$statistic, ", so reject the null; i.e., the residuals are HETEROSCHEDASTIC"))
} else {
    print(paste0("Breusch-Pagan test for homoschedasticity: The p-value of ",
        bp$p.value, " and test statistic of ", bp$statistic,
        " are inconclusive, so homoschedasticity can't be determined using this test."))
}
print("")

# Visually look for colinearity - dont do this for
# large models pairs(model.matrix(lmod))

# Variance inflation factor (VIF)
print("Variance inflation factor (VIF)")
print("<=1: not correlated, 1-5: moderately correlated, >5: strongly correlated")
print(sort(vif(lmod), decreasing = T))
print("")

# Standardized residual plots (look for points outside
# of 2 or 3 stdev)
p <- length(summary(lmod)$coeff[, 1] - 1)  # number of model parameters
stanres <- rstandard(lmod)
for (i in seq(1, ceiling(p/4))) {
    par(mfrow = c(2, 2))
    starti <- ((i - 1) * 4) + 1
    for (j in seq(starti, starti + 3)) {
        if (j + 1 <= ncol(model.matrix(lmod))) {
            # Skip these plots since we're pretty sure
            # that a linear model isn't valid here
            # plot(model.matrix(lmod)[, j + 1],
            # stanres,
```

```
                # xlab=colnames(model.matrix(lmod))[j + 1],
                # ylab='Standardized residuals')
                # abline(h=c(-2, 2), lt=3, col='blue')
                # abline(h=c(-3, 3), lt=2, col='red')
            }
        }
    }

    # Model scores
    print("Model scores:")
    print(paste0("    adjusted R-squared: ", round(summary(lmod)$adj.r.squared,
        3)))
    print(paste0("    AIC: ", round(AIC(lmod, k = 2), 3)))
    print(paste0("    BIC: ", round(BIC(lmod), 3)))
    print(paste0("    Mallow's Cp: ", round(ols_mallows_cp(lmod,
        fullmodel = lmod), 3)))
    print(paste0("    mean squared error: ", round(calc_mse(lmod),
        3)))
    print("")

    # Find leverage point cutoff
    n <- length(lmod$residuals)
    cutoff <- 2 * (p + 1)/n
    print(paste0("Leverage point cutoff: ", cutoff))
    print("")

    # Show points of influence
    print("First 10 points of influence:")
    poi <- lm.influence(lmod)$hat
    len_poi <- length(poi)
    ct <- 0
    for (i in seq(1, length(poi))) {
        if (poi[i] > cutoff) {
            ct <- ct + 1
            print(paste0("    case #", i, ": ", round(poi[i],
                3)))
        }
        if (ct > 10) {
            break
        }
    }
    print("")

}
ModelAnalysis(lm_mod1)
ModelAnalysis(lm_mod2)

# Calc overdispersion
op1 <- p_mod1$deviance/p_mod1$df.residual
op2 <- p_mod2$deviance/p_mod2$df.residual
print(paste0("Poisson model 1 overdispersion: ", op1))
print(paste0("Poisson model 2 overdispersion: ", op2))
```

```r
# Calc accuracies

# Show histogram of actual cases sold to compare with
# histograms generated by predictions
cleandfnew <- cleandf
par(mfrow = c(2, 3))
hist(cleandfnew$TARGET, xlab = "Cases", main = "Actual Cases Sold")

# Gaussian
cleandfnew$pred_target <- predict(lm_mod2, cleandfnew, interval = "prediction")
cleandfnew$correct <- ifelse(round(cleandfnew$pred_target, 0) ==
    cleandfnew$TARGET, 1, 0)
acc_num <- sum(cleandfnew$correct)
acc_pct <- round(100 * acc_num/nrow(cleandfnew), 1)
print(paste0("Gaussian accuracy: ", acc_num, " of ", nrow(cleandfnew),
    " (", acc_pct, "%)"))
hist(round(cleandfnew$pred_target, 0), xlab = "Cases", main = "Predictions (Gaussian)")

# Negative binomial
cleandfnew$pred_target <- exp(predict(nb_mod2, cleandfnew, interval = "prediction"))
cleandfnew$correct <- ifelse(round(cleandfnew$pred_target, 0) ==
    cleandfnew$TARGET, 1, 0)
acc_num <- sum(cleandfnew$correct)
acc_pct <- round(100 * acc_num/nrow(cleandfnew), 1)
print(paste0("Negative binomial model accuracy: ", acc_num, " of ",
    nrow(cleandfnew), " (", acc_pct, "%)"))
hist(round(cleandfnew$pred_target, 0), xlab = "Cases", main = "Predictions (Negative Binomial)")

# Poisson
cleandfnew$pred_target <- exp(predict(p_mod2, cleandfnew, interval = "prediction"))
cleandfnew$correct <- ifelse(round(cleandfnew$pred_target, 0) ==
    cleandfnew$TARGET, 1, 0)
acc_num <- sum(cleandfnew$correct)
acc_pct <- round(100 * acc_num/nrow(cleandfnew), 1)
print(paste0("Poisson accuracy: ", acc_num, " of ", nrow(cleandfnew),
    " (", acc_pct, "%)"))
hist(round(cleandfnew$pred_target, 0), xlab = "Cases", main = "Predictions (Poisson)")

# Zero-inflated
cleandfnew$pred_target <- predict(zi_mod2, cleandfnew, interval = "prediction")
cleandfnew$correct <- ifelse(round(cleandfnew$pred_target, 0) ==
    cleandfnew$TARGET, 1, 0)
acc_num <- sum(cleandfnew$correct)
acc_pct <- round(100 * acc_num/nrow(cleandfnew), 1)
print(paste0("Zero-inflated poisson accuracy: ", acc_num, " of ",
    nrow(cleandfnew), " (", acc_pct, "%)"))
hist(round(cleandfnew$pred_target, 0), xlab = "Cases", main = "Predictions (Zero-Inflated)")

# Hierarchical modeling

# Binomial on whether cases were sold or not
cleandf_binom <- cleandf %>%
    mutate(pos_cases = ifelse(TARGET > 0, T, F))
```

```
b_mod1 <- glm(pos_cases ~ . - TARGET, family = binomial(), data = cleandf_binom)
b_mod2 <- stepAIC(b_mod1, trace = F)
aic_b_mod1 <- AIC(b_mod1)
aic_b_mod2 <- AIC(b_mod2)

# Summary of binomial model 1:
summary(b_mod1)

# Summary of binomial model 2:
summary(b_mod2)

# AIC of binomial model 1:
aic_b_mod1

# AIC of binomial model 2:
aic_b_mod2

# Check overdispersion
cleandf_binom$pred_p <- ilogit(predict(b_mod2, cleandf_binom,
    interval = "prediction"))
cleandf_binom$pred_pos_cases <- ifelse(cleandf_binom$pred_p >
    0.5, T, F)
cleandf_binom$correct_pos_cases <- ifelse(cleandf_binom$pred_pos_cases ==
    cleandf_binom$pos_cases, 1, 0)
acc_num <- sum(cleandf_binom$correct_pos_cases)
acc_pct <- round(100 * acc_num/nrow(cleandf_binom), 1)
print(paste0("Binomial model accuracy with p of 0.5: ", acc_num,
    " of ", nrow(cleandf_binom), " (", acc_pct, "%)"))

# ROC
roc_func <- function(data) {
    temp_x <- rep(0, 101)
    temp_y <- rep(0, 101)
    temp_z <- rep(0, 101)
    temp_seq <- seq(from = 0, to = 1, by = 0.01)
    max_acc <- 0
    p_max_acc <- 0
    for (i in 1:length(temp_seq)) {
        df <- data %>%
            mutate(scored.class = as.logical(pred_p > temp_seq[i])) %>%
            mutate(true.class = as.logical(TARGET > 0)) %>%
            mutate(TP = ifelse(true.class == T & scored.class ==
                T, 1, 0), FP = ifelse(true.class == F & scored.class ==
                T, 1, 0), FN = ifelse(true.class == T & scored.class ==
                F, 1, 0), TN = ifelse(true.class == F & scored.class ==
                F, 1, 0))
        TPR <- sum(df$TP)/(sum(df$TP) + sum(df$FN))
        FPR <- sum(df$FP)/(sum(df$FP) + sum(df$TN))
        acc <- (sum(df$TP) + sum(df$TN))/nrow(df)
        if (acc > max_acc) {
            max_acc <- acc
            p_max_acc <- temp_seq[i]
        }
```

```
        temp_x[i] <- FPR
        temp_y[i] <- TPR
        temp_z[i] <- acc
    }
    temp_df <- bind_cols(temp_x, temp_y, temp_seq, temp_z) %>%
        as.data.frame()
    names(temp_df) <- c("FPR", "TPR", "p", "Accuracy")
    plt <- ggplot2::ggplot(data = temp_df, aes(x = FPR, y = TPR)) +
        geom_point() + geom_abline()
    plt2 <- ggplot2::ggplot(data = temp_df, aes(x = p, y = Accuracy)) +
        geom_point()
    AUC <- pracma::trapz(temp_x, temp_y)
    output <- list(plt, plt2, AUC, max_acc, p_max_acc)
    return(output)
}

# Generate ROC curve
rf <- roc_func(cleandf_binom)
print(rf)
rf[[5]]

# Find p at the maximum value for accuracy
cleandf_binom$pred_p <- ilogit(predict(b_mod2, cleandf_binom,
    interval = "prediction"))
cleandf_binom$pred_pos_cases <- ifelse(cleandf_binom$pred_p >
    rf[[5]], T, F)
cleandf_binom$correct_pos_cases <- ifelse(cleandf_binom$pred_pos_cases ==
    cleandf_binom$pos_cases, 1, 0)
acc_num <- sum(cleandf_binom$correct_pos_cases)
acc_pct <- round(100 * acc_num/nrow(cleandf_binom), 1)
print(paste0("Binomial model accuracy with p of ", rf[[5]], ": ",
    acc_num, " of ", nrow(cleandf_binom), " (", acc_pct, "%)"))

# Poisson against non-zero counts
p_mod3 <- glm(TARGET ~ ., family = poisson(), data = cleandf["TARGET" >
    0])
aic_p_mod3 <- AIC(p_mod3)
p_mod4 <- stepAIC(p_mod3, trace = F)
aic_p_mod4 <- AIC(p_mod4)
print(paste0("AIC of poisson full model: ", aic_p_mod3))
print(paste0("AIC of poisson step-reduced model: ", aic_p_mod4))

# Check for overdispersion of the poisson models:
op3 <- p_mod3$deviance/p_mod3$df.residual
op4 <- p_mod4$deviance/p_mod4$df.residual
print(paste0("Poisson full model overdispersion: ", op3))
print(paste0("Poisson step-reduced model overdispersion: ", op4))

# Try negative binomial
nb_mod3 <- glm.nb(TARGET ~ ., data = cleandf["TARGET" > 0])
aic_nb_mod3 <- AIC(nb_mod3)
nb_mod4 <- stepAIC(nb_mod3, trace = F)
aic_nb_mod4 <- AIC(nb_mod4)
```

```
print(paste0("AIC of negative binomial full model: ", aic_nb_mod3))
print(paste0("AIC of negative binomial step-reduced model: ",
    aic_nb_mod4))

# Vary p to find the greatest accuracy
tmp_p <- seq(from = 0, to = 1, by = 0.01)
c_acc_pct <- c()
max_acc <- 0
p_max_acc <- 0
for (i in seq(1, length(tmp_p))) {

    # Make initial prediction on zero cases vs non-zero
    # cases
    cleandf$pred_p <- ilogit(predict(b_mod2, cleandf, interval = "prediction"))
    cleandf$pred_target <- ifelse(cleandf$pred_p > tmp_p[i],
        NA, 0)

    # Split df into zero cases and non-zero cases
    cleandf1 <- cleandf %>%
        filter(pred_target == 0)
    cleandf2 <- cleandf %>%
        filter(is.na(pred_target))

    # Make predications against non-zero cases
    cleandf2$pred_target <- round(exp(predict(p_mod4, cleandf2,
        interval = "prediction")), 0)
    cleandfnew <- rbind(cleandf1, cleandf2)

    # Calculate accuracy
    cleandfnew$correct <- ifelse(cleandfnew$pred_target == cleandfnew$TARGET,
        1, 0)
    acc_num <- sum(cleandfnew$correct)
    acc_pct <- round(100 * acc_num/nrow(cleandfnew), 1)
    c_acc_pct <- c(c_acc_pct, acc_pct)
    if (acc_pct > max_acc) {
        max_acc <- acc_pct
        p_max_acc <- tmp_p[i]
    }
    # print(paste0('Hierarchical model accuracy: ',
    # acc_num, ' of ', nrow(cleandfnew), ' (', acc_pct,
    # '%)'))
}
plot(c_acc_pct ~ tmp_p)
print(paste0("Max accuracy is ", max_acc, "% when p=", p_max_acc))

# Show histogram at max accuracy
cleandf$pred_p <- ilogit(predict(b_mod2, cleandf, interval = "prediction"))
cleandf$pred_target <- ifelse(cleandf$pred_p > p_max_acc, NA,
    0)
cleandf1 <- cleandf %>%
    filter(pred_target == 0)
cleandf2 <- cleandf %>%
    filter(is.na(pred_target))
```

```
cleandf2$pred_target <- round(exp(predict(p_mod4, cleandf2, interval = "prediction")),
    0)
cleandfnew <- rbind(cleandf1, cleandf2)
par(mfrow = c(1, 2))
hist(cleandfnew$TARGET, xlab = "Cases", main = "Actual Cases Sold")
hist(cleandfnew$pred_target, xlab = "Cases", main = "Predicted Cases Sold")

# Calculate 'close' predictions (within one case of the
# actual number)
cleandfnew$close <- ifelse((cleandfnew$pred_target >= cleandfnew$TARGET -
    1) & (cleandfnew$pred_target <= cleandfnew$TARGET + 1), 1,
    0)
acc_pct <- round(100 * sum(cleandfnew$close)/nrow(cleandfnew),
    1)
print(paste0("Close prediction accuracy (within one case of actual): ",
    acc_pct, "%"))

# Now we'll use this cutoff value against the evaluation
# data.

# Predictions zero vs non-zero
cleandf_eval$pred_p <- ilogit(predict(b_mod2, cleandf_eval, interval = "prediction"))
cleandf_eval$pred_target <- ifelse(cleandf_eval$pred_p > p_max_acc,
    NA, 0)

# Split df into zero cases and non-zero cases
cleandf_eval1 <- cleandf_eval %>%
    filter(pred_target == 0)
cleandf_eval2 <- cleandf_eval %>%
    filter(is.na(pred_target))

# Make predications against non-zero cases
cleandf_eval2$pred_target <- round(exp(predict(p_mod4, cleandf_eval2,
    interval = "prediction")), 0)
cleandf_eval_new <- rbind(cleandf_eval1, cleandf_eval2)

# Show results
cleandf_eval_new %>%
    head(10) %>%
    as_tibble()
hist(cleandf_eval_new$pred_target, xlab = "Cases", main = "Predicted Cases Sold")
write.csv(cleandf_eval, "wine_predictions2.csv", row.names = FALSE)
```

References

https://englianhu.files.wordpress.com/2016/01/faraway-extending-the-linear-model-with-r-e28093-2006.pdf