HW #5

Critical Thinking Group One

2021-05-14

Contents

Authorship

Critical Thinking Group 1

- Angel Claudio
- Bonnie Cooper
- Manolis Manoli
- Magnus Skonberg
- Christian Thieme
- Leo Yi

Abstract

We will explore, analyze and model a data set containing approximately 12,000 records representing various commercially available wines. The variables are primarily 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 the wine is to be sold at a high end restaurant. A large 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.

Our objective is to build a **count regression model** to predict the number of cases of wine that will be sold given certain properties of the wine.

Data Exploration

The goal of exploratory data analysis is to enhance the precision of the questions we're asking while building a firm understanding of the data at hand. The aim is to familiarize ourselves with the status of missing values, outliers, predictive strength, and correlation to then take the actions necessary to optimize our data set when we prepare our data prior to model building.

First, we get to know the structure and value ranges and proportion of missing values and correlation with the target variable. We then more thoroughly explore and address the high proportion of missing `STARS` values, visualize independent variable distributions, visualize independent variables vs. target via boxplot to spot outliers and such, and then explore whether multicollinearity exists within our set.

After this point, we should have enough insight to prepare our data and build our model(s).

Data Structure

To start, we utilize the built-in `glimpse` method to gain insight into the dimensions, variable characteristics, and value range for our training dataset:

```
## Rows: 12,795
## Columns: 17
## $ INDEX             <dbl> 1, 2, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 19~
## $ TARGET            <dbl> 3, 3, 5, 3, 4, 0, 0, 4, 3, 6, 0, 4, 3, 7, 4, 0, 0, ~
## $ FixedAcidity      <dbl> 3.2, 4.5, 7.1, 5.7, 8.0, 11.3, 7.7, 6.5, 14.8, 5.5,~
```

```
## $ VolatileAcidity    <dbl> 1.160, 0.160, 2.640, 0.385, 0.330, 0.320, 0.290, -1~
## $ CitricAcid         <dbl> -0.98, -0.81, -0.88, 0.04, -1.26, 0.59, -0.40, 0.34~
## $ ResidualSugar      <dbl> 54.20, 26.10, 14.80, 18.80, 9.40, 2.20, 21.50, 1.40~
## $ Chlorides          <dbl> -0.567, -0.425, 0.037, -0.425, NA, 0.556, 0.060, 0.~
## $ FreeSulfurDioxide  <dbl> NA, 15, 214, 22, -167, -37, 287, 523, -213, 62, 551~
## $ TotalSulfurDioxide <dbl> 268, -327, 142, 115, 108, 15, 156, 551, NA, 180, 65~
## $ Density            <dbl> 0.99280, 1.02792, 0.99518, 0.99640, 0.99457, 0.9994~
## $ pH                 <dbl> 3.33, 3.38, 3.12, 2.24, 3.12, 3.20, 3.49, 3.20, 4.9~
## $ Sulphates          <dbl> -0.59, 0.70, 0.48, 1.83, 1.77, 1.29, 1.21, NA, 0.26~
## $ Alcohol            <dbl> 9.9, NA, 22.0, 6.2, 13.7, 15.4, 10.3, 11.6, 15.0, 1~
## $ LabelAppeal        <dbl> 0, -1, -1, -1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 2, 0, 0, ~
## $ AcidIndex          <dbl> 8, 7, 8, 6, 9, 11, 8, 7, 6, 8, 5, 10, 7, 8, 9, 8, 9~
## $ STARS              <dbl> 2, 3, 3, 1, 2, NA, NA, 3, NA, 4, 1, 2, 2, 3, NA, NA~
## $ dataset            <chr> "train", "train", "train", "train", "train", "train~
```

From above, we see that our training dataset has **16** features (of type double) and **12,795** observations, with varying ranges (shown across each row). In looking at the data it appears that `LabelAppeal`, `AcidIndex`, and `STARS` appear to be categorical features. We'll investigate this more in the EDA process.

We also note that:

- The `INDEX` variable appears to be impertinent
- STARS appears to have quite a few NAs
- There appears to be a significant difference in the scale of many of the features (ie. `STARS` vs. `TotaSulfurDioxide`).We may need to normalize this dataset before modeling

Now let's get a high level look at our distributions:

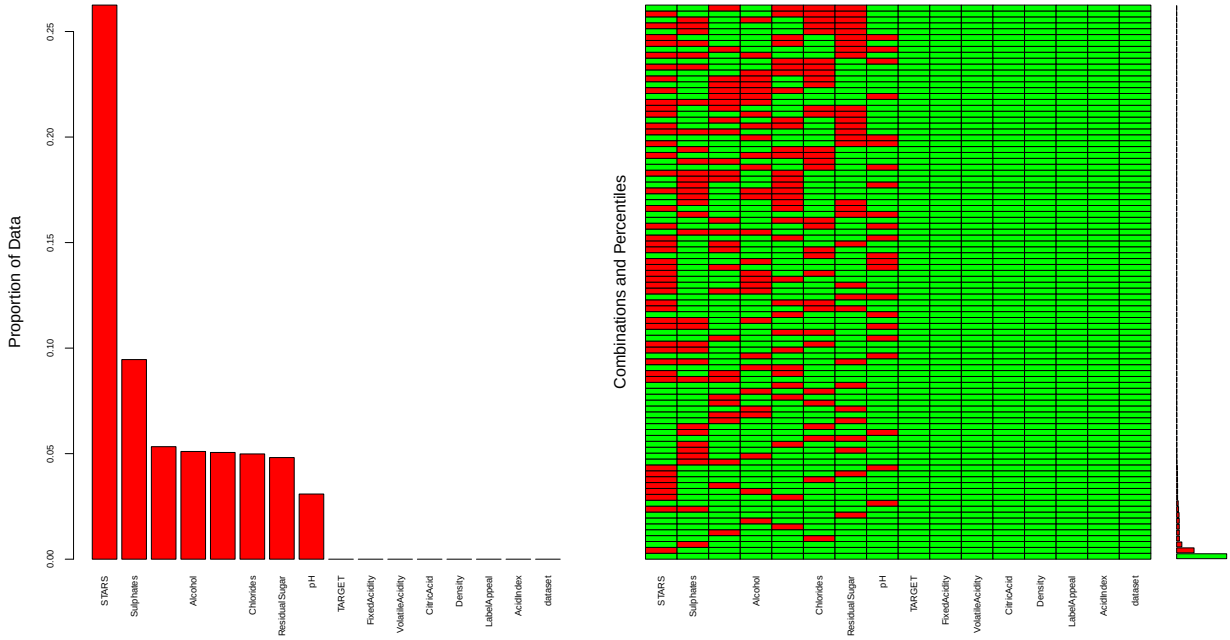
```
##      INDEX           TARGET       FixedAcidity     VolatileAcidity
##  Min.   :    1   Min.   :0.000   Min.   :-18.100   Min.   :-2.7900
##  1st Qu.: 4038   1st Qu.:2.000   1st Qu.:  5.200   1st Qu.: 0.1300
##  Median : 8110   Median :3.000   Median :  6.900   Median : 0.2800
##  Mean   : 8070   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
##    dataset
##  Length:12795
##  Class :character
##  Mode  :character
##
##
##
##
```

We note that many of these features minimums are below 0.

Missing Values

First, we'll drop `INDEX` from consideration and then investigate our missing values:



From the proportion chart above on the left we can see that:

- `STARS` has ~26% missing values. These missing values most likely indicate that the wine has not been rated by a team of experts.
- `Sulphates` has ~10% of it's values missing. This feature does have a 0 value, so these values appear to be missing as opposed to signifying no sulphates.
- `TotalSulfurDioxide` has ~5% missing values. This feature does have a 0 value, so these values appear to be missing as opposed to signifying no sulfur dioxides.
- `Alcohol` has ~5% missing values. Alcohol does not appear to have a 0 value, so we'll need to investigate if blanks indicate no alcohol in the wine

- `FreeSulfurDioxide` has ~5% missing values. FreeSulfurDioxide does not appear to have a 0 value, so we'll need to investigate if blanks indicate no FreeSulfurDioxide in the wine
- `Chlorides` has ~5% missing values. This feature does have a 0 value, so these values appear to be missing as opposed to signifying no chlorides
- `ResidualSugar` has ~5% missing values.This feature does have a 0 value, so these values appear to be missing as opposed to signifying no residual sugars
- `pH` has ~3% missing values. pH does not appear to have a 0 value, so we'll need to investigate if blanks indicate a 0 pH value (seems unlikely)

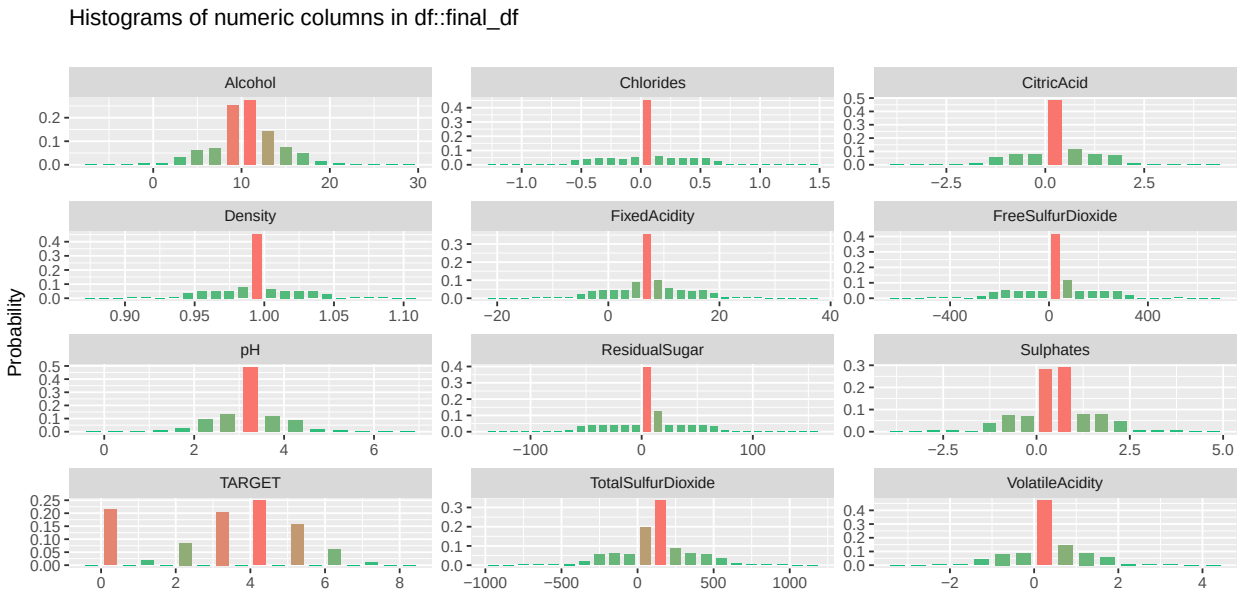
On the combinations and percentiles chart we note that combinations look fairly random with less than 2.5% of missing data having the same pattern.

Data Type Conversions

We noted earlier that several data type conversions were necessary. We'll convert `LabelAppeal`, `AcidIndex`, and `STARS` to factors. Before changing `STARS`, we'll need to convert the nulls to 'Not Rated'.

Numeric Variable Distributions

Earlier we'd noted the vast difference in ranges between many of our variables. To explore this point further and gain greater insight as to the distribution for each of our variables, we visit the plots produced via utilization of inspectdf's `inspect_num` function:



From the distribution plots above we note that:

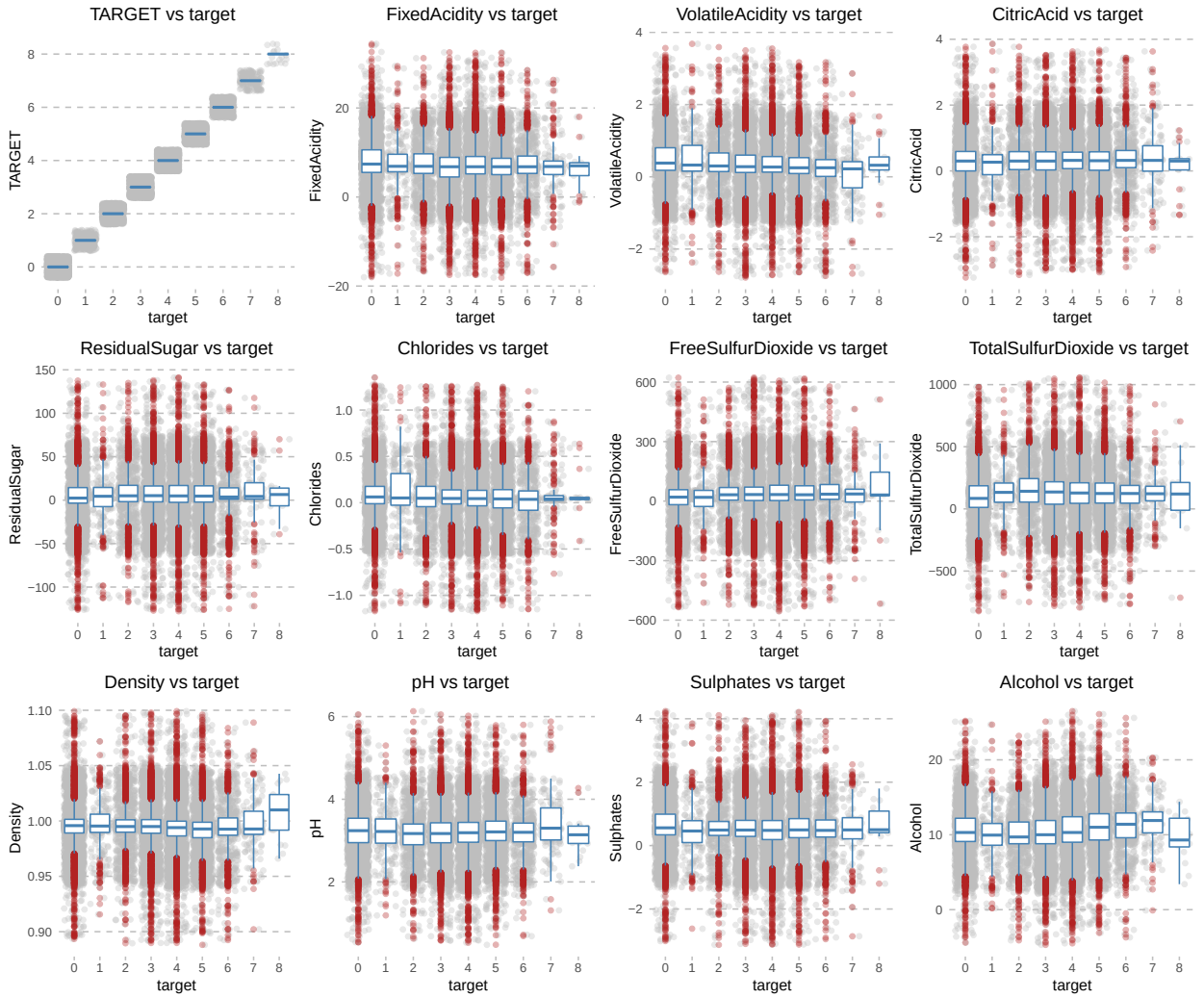
- `Alcohol` has a relatively normal distribution with a mean value of ~10 which makes sense when we consider this variable is representative of alcohol content.
- `Chlorides` has a significant frequency (40%) spike at ~0. *We may consider using this variable value as a flag.*
- `CitricAcid` has a significant frequency (50%) spike at ~0. *We may consider using this variable value as a flag.*

- `Density` has a significant frequency (50%) spike at ~0.99. *We may consider using this variable value as a flag.*
- `FixedAcidity`has a significant frequency (35%) spike at ~3. *We may consider using this variable value as a flag.*
- `FreeSulfurDioxide` has a significant frequency (40%) spike at ~0. *We may consider using this variable value as a flag.*
- `pH` follows a relatively normal distribution with a significant (50%) frequency spike centered about 3. Better wines generally sit in the 3-4 pH range which it appears the majority of our wines do.
- `ResidualSugar` has a significant frequency (40%) spike at 0. *We may consider using this variable value as a flag.*
- `Sulphates` follows a normal distribution centered between 0 and 1. 50-60% of values fall between 0 and 1.5.
- `TARGET` follows a bimodal distribution. Aside from the spike at 0 it appears to be a normal distribution centered about 4. This means that case sales of 4 is most common while those above and below reduce in frequency until we reach 0 (where it spikes again).
- `TotalSulfurDioxide` has two significant frequency spikes. One (30%) at ~100 and another (~20%) at 0. *We may consider using these variable values as flags.*
- `VolatileAcidity` has a significant (45%) frequency spike at 0. *We may consider using this variable value as a flag.*

From the distributions above, we confirm that our model building and analysis may be improved by incorporating flag / dummy variables to account for the numerous, significant frequency spikes observed above.

Boxplots

Now that we've got a basic understanding of the distribution of each of our features, let's turn our attention to their relationship with `TARGET` which is our target variable. In beginning this analysis, we found that understanding the relationship with our target variable was difficult without also seeing the quantity of observations in each group. With this in mind, we've layered our boxplots on top of a dotplot in order to see both quartiles and outliers, as well as the quantity of observations.



From above we gather that:

- There are outliers in all of our numeric feature
- The purchase of 1, 7, and 8 boxes is much rarer than 2-6. We would expect 7 and 8 to be rare, however, we weren't necessarily expecting to see 1 box be rare
- The range for 7-8 boxes sold is much lower than the others (except 1 box)
- It does not appear that there are very many significant relationships between the features and our target variable, with the exception of: `Density` - range and median value for 8 boxes is significantly higher than others and `Alcohol` - range and median value for 7 sold boxes is quite a bit higher than others, however, 8 boxes sold is significantly lower than all others

Contingency Tables

Now, let's create contingency tables for each of our categorical variables so we can identify any relationships:

LabelAppeal

```
##
##          0    1    2    3    4    5    6    7    8
##   -2   102  136  177   74   14    1    0    0    0
```

```
##   -1  671   89  755 1118  413   88    2    0    0
##   0  1193   19  152 1347 1972  775  155    4    0
##   1   660    0    7   70  765 1040  425   79    2
##   2   108    0    0    2   13  110  183   59   15
```

In looking at the contingency table above, we can see that there appears to be a relationship between `LabelAppeal` and the number of cases purchased, `TARGET`. The higher the label appeal, the more cases purchased.

AcidIndex

Acid index is a proprietary method of testing total acidity using a weighted average.

```
##
##        0    1    2    3    4    5    6    7    8
##   4    1    0    0    0    0    2    0    0    0
##   5   11    0    3   16   21   15    8    1    0
##   6  170   18   91  259  318  234   86   21    0
##   7  727   88  461  996 1324  867  354   57    4
##   8  782   89  351  904 1090  632  238   44   12
##   9  437   28  126  285  305  174   57   14    1
##   10 257   13   40  102   73   54    9    3    0
##   11 169    6   12   29   23   16    2    1    0
##   12  92    2    3    9    8    8    5    1    0
##   13  42    0    2    4   11    7    3    0    0
##   14  32    0    2    6    2    2    3    0    0
##   15   4    0    0    1    2    1    0    0    0
##   16   4    0    0    0    0    1    0    0    0
##   17   6    0    0    0    0    1    0    0    0
```

In looking at the table above, we don't see a linear relationship between these variables, however, we do note that it looks like wines that sell more boxes, typically have an acid index between 6 and 9.

STARS

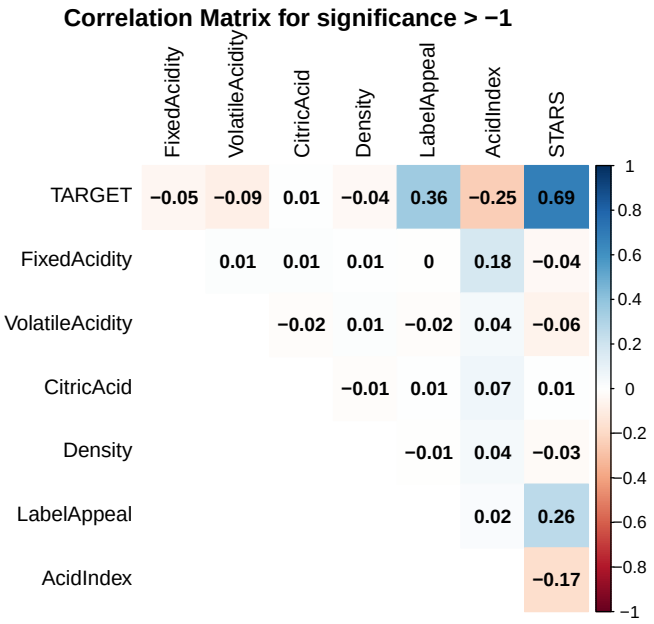
STARS is a wine rating by a team of experts. The higher the number, the better the rating.

```
##
##                0    1    2    3    4    5    6    7    8
##   Not Rated 2038  126  335  457  260  101   32    8    2
##   1          607   98  469  916  716  214   22    0    0
##   2           89   20  253  948 1333  716  199   12    0
##   3            0    0   34  290  764  750  313   57    4
##   4            0    0    0    0  104  233  199   65   11
```

In general, we do see a fairly strong relationship here. The higher the rating, the more boxes sold. Additionally, as noted before, there are many wines that have not been rated.

Correlation Matrix

Having reviewed the relationship each of our numeric and categorical features has with `TARGET`, we turn our attention to exploring the relationship these variables have with one another via **correlation matrix**. We consider only variables with a correlation significant > 0.1 in our plot:



Although `FixedAcidity` is correlated with `AcidityIndex`, while `LabelAppeal` and `AcidIndex` are correlated with `STARS`, it appears that **multicollinearity is not a concern**. Additionally, the above output confirms our (3) strongest predictors while highlighting the weak predictive potential of our independent, numeric variables. It is interesting that our numeric predictors have such a weak relationship with `TARGET`, where our 3 categorical features have a strong relationship.

EDA Summary

The training dataset has 15 variables (after dropping `INDEX`) and 12795 observations. The remediation actions became apparent over the course of our exploratory data analysis:

- There are numerous features with a weak relationship to `TARGET` that we'll consider at dropping.
- There are outliers in every feature that will need to be addressed.
- Imputing missing values will be an important factor in our modeling. From our analysis, KNN seems to be a logical choice since wines of similar compositions should have similar values.
- Many of our predictors did not have a relationship with our target variable. As such, we may need to craft new features to extract additional signal.
- Pay attention to multi-collinearity between `STARS` and `LabelAppeal` (what appear to be our 2 strongest predictors).

The recommendations above provide a "starting line" for our data preparation.

. . . . . .

Data Preparation & Model Building

With insights gained via EDA, we set out to explore different means of model optimization to (later) select the model with the strongest predictive capability when we cast our predictions.

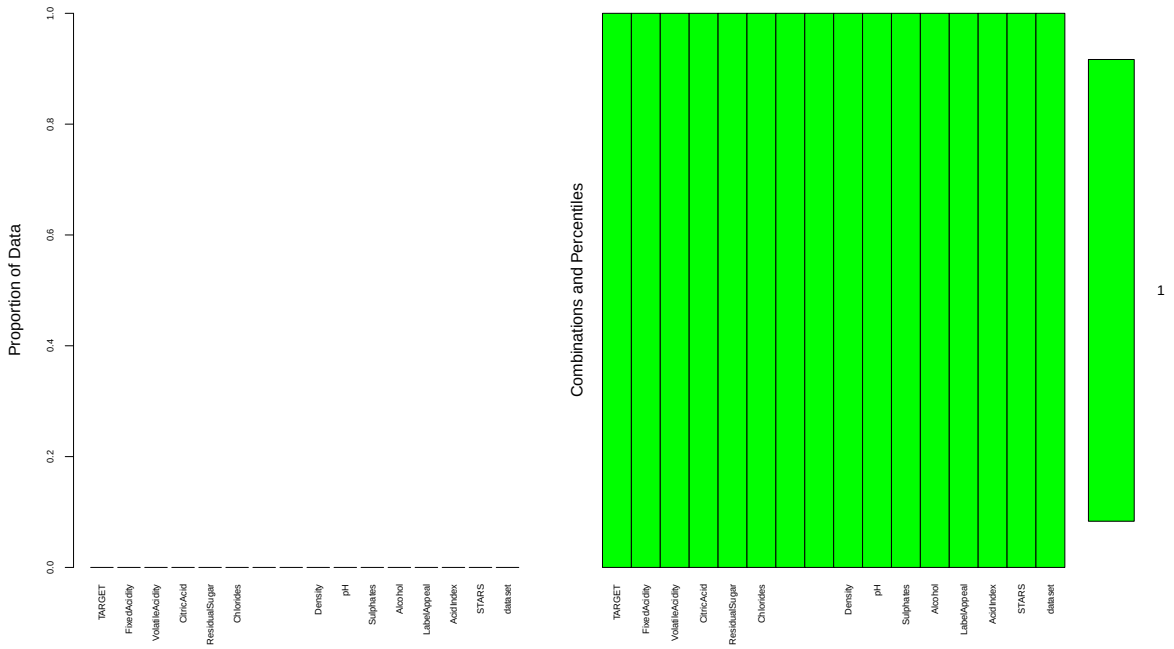
We impute missing values, explore the performance of our "baseline model" (`model_1`), deal with multi-collinearity / impertinent features, and hone our baseline model via outlier-handling, normalization, the exploration of over/underdispersion, AIC optimization and the numerous modeling methods (ie. Poisson vs. quasi-Poisson).

Missing Value Imputation

We compare the performance of KNN and predictive mean matching (PMM) imputation. We used both methods in our initial model run and observed that predictive mean matching performed slightly better than KNN. As such, we proceed with PMM.

```
##
##  iter imp variable
##    1   1  TARGET  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
##    2   1  TARGET  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
##    3   1  TARGET  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
##    4   1  TARGET  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
##    5   1  TARGET  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
```

```
##             TARGET       FixedAcidity    VolatileAcidity         CitricAcid
##               3335                  0                  0                  0
##      ResidualSugar          Chlorides  FreeSulfurDioxide TotalSulfurDioxide
##                  0                  0                  0                  0
##            Density                 pH          Sulphates            Alcohol
##                  0                  0                  0                  0
##        LabelAppeal          AcidIndex              STARS            dataset
##                  0                  0                  0                  0
```



Baseline model

With our dataset properly shaped, and prior to any further dataset transformations or model extensions (ie. use of the quasi-Poisson model), we explore the performance of our "baseline model" (`model_1`):

```
##
## Call:
```

```
## glm(formula = TARGET ~ ., family = poisson(link = "log"), data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2226  -0.6497  -0.0051   0.4460   3.6809
##
## Coefficients:
##                       Estimate   Std. Error z value    Pr(>|z|)
## (Intercept)         1.051339182  0.201262141   5.224 0.000000175 ***
## FixedAcidity        0.000158890  0.000820334   0.194    0.846419
## VolatileAcidity    -0.029658307  0.006532943  -4.540 0.000005631 ***
## CitricAcid          0.004514248  0.005897209   0.765    0.443980
## ResidualSugar       0.000004695  0.000151331   0.031    0.975250
## Chlorides          -0.039321401  0.016120399  -2.439    0.014718 *
## FreeSulfurDioxide   0.000080942  0.000034202   2.367    0.017953 *
## TotalSulfurDioxide  0.000077999  0.000022154   3.521    0.000430 ***
## Density            -0.281342266  0.191811704  -1.467    0.142441
## pH                 -0.010161579  0.007519218  -1.351    0.176563
## Sulphates          -0.010744820  0.005488742  -1.958    0.050276 .
## Alcohol             0.004379406  0.001371540   3.193    0.001408 **
## LabelAppeal.L       0.544779437  0.027545777  19.777     < 2e-16 ***
## LabelAppeal.Q      -0.070833964  0.023113890  -3.065    0.002180 **
## LabelAppeal.C       0.016622854  0.016299060   1.020    0.307792
## LabelAppeal^4       0.008013394  0.010362938   0.773    0.439360
## AcidIndex.L        -1.164544869  0.300133549  -3.880    0.000104 ***
## AcidIndex.Q        -0.013411143  0.291695677  -0.046    0.963329
## AcidIndex.C        -0.055515532  0.263082817  -0.211    0.832872
## AcidIndex^4        -0.368044272  0.252514389  -1.458    0.144973
## AcidIndex^5        -0.361244979  0.250314029  -1.443    0.148973
## AcidIndex^6         0.159862427  0.238420028   0.671    0.502534
## AcidIndex^7         0.220901576  0.216708707   1.019    0.308038
## AcidIndex^8         0.160527317  0.186732717   0.860    0.389975
## AcidIndex^9         0.085149651  0.152495453   0.558    0.576588
## AcidIndex^10        0.203999527  0.117874290   1.731    0.083514 .
## AcidIndex^11        0.199188765  0.087954266   2.265    0.023532 *
## AcidIndex^12        0.094929662  0.069445755   1.367    0.171637
## AcidIndex^13       -0.034039119  0.054418875  -0.626    0.531642
## STARS.L             0.967293059  0.016468577  58.736     < 2e-16 ***
## STARS.Q            -0.392953646  0.014155477 -27.760     < 2e-16 ***
## STARS.C             0.138960897  0.012131549  11.455     < 2e-16 ***
## STARS^4            -0.004127305  0.009904748  -0.417    0.676898
## ---
## 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: 13527  on 12762  degrees of freedom
## AIC: 45535
##
## Number of Fisher Scoring iterations: 6
```

Our "baseline model" has:

- A fair proportion of **high p-values** (ie.`FixedAcidity`) which may indicate a need for further variable exclusion.
- A **null deviance of 22861** and **residual deviance of 13527** on > 12000 degrees of freedom. Higher numbers indicate worse fit. We want lower deviance values.
- An **AIC value of 45535**. The AIC value is a useful metric for model selection because it takes into account goodness of fit *and* model simplicity. We want a lower AIC value.

Overall, our first model indicates that our model does not fit the data well. We are explaining ~41% (1 - 13,527/22,861) of the deviance of the data.

Multicollinearity

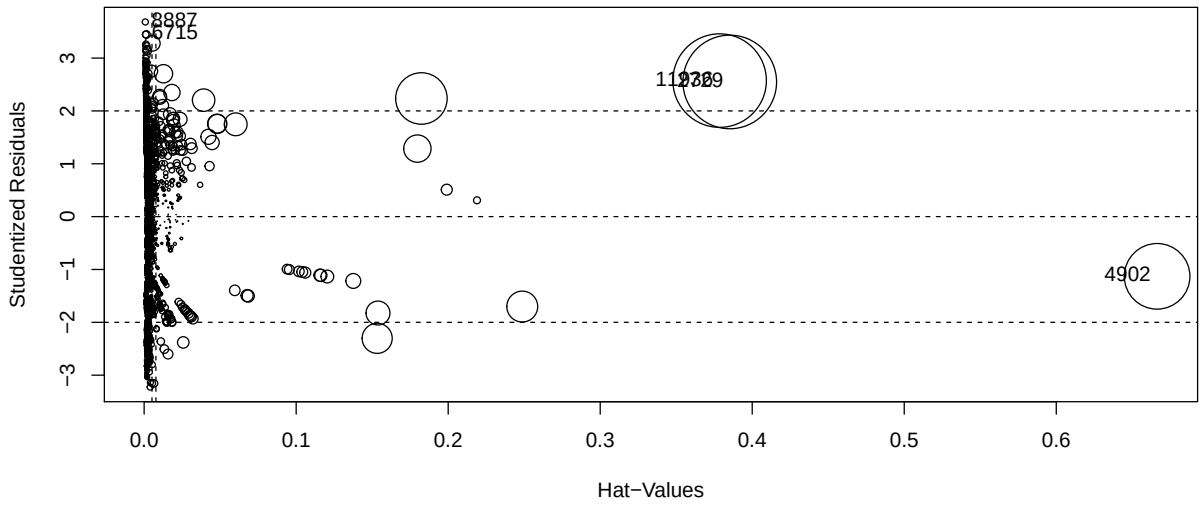
Having run our first model, we can check our multicollinearity:

```
##                        GVIF Df GVIF^(1/(2*Df))
## FixedAcidity       1.023010  1        1.011440
## VolatileAcidity    1.005972  1        1.002982
## CitricAcid         1.006582  1        1.003286
## ResidualSugar      1.005070  1        1.002532
## Chlorides          1.004181  1        1.002088
## FreeSulfurDioxide  1.004866  1        1.002430
## TotalSulfurDioxide 1.005763  1        1.002877
## Density            1.004751  1        1.002373
## pH                 1.008176  1        1.004080
## Sulphates          1.003408  1        1.001702
## Alcohol            1.012949  1        1.006454
## LabelAppeal        1.141980  4        1.016734
## AcidIndex          1.085817 13        1.003172
## STARS              1.179397  4        1.020840
```

Looking at the above output, our GVIF can be be squared and taken against the normal VIF rule of no greater than 5 or 10. In squaring any of these numbers, we see we would fall well below the 5 threshold, therefore, we conclude that multicollinearity is not an issue in this dataset.

Outliers

We can adjust the standard Poisson model to allow for more variation in the response. However, before doing that, we need to check whether the large size of deviance is related to outliers:




```
##         StudRes          Hat        CookD
## 2729   2.548215 0.3852321072 0.1522503777
## 4902  -1.129522 0.6663433162 0.0748338587
## 6715   3.446658 0.0010293164 0.0007500940
## 8887   3.683819 0.0007804109 0.0006496918
## 11936  2.573896 0.3785527360 0.1519813475
```

In looking at the above plot, we do see several abnormal outliers. We took a look at those rows with a cook's distance greater than 4x the mean and noted that >250 observations met our outlier criteria. **While we had originally removed these observations from the dataset, it later caused an issue when casting predictions so we excluded outlier handling (code available in Appendix).**

The write up that follows is how we had proceed prior to reaching that decision:

In removing several outliers, we improved the model slightly, checked our diagnostic plots, and noted that our residuals are not normal and don't have equal variance. Additionally, it looks like they have some curvature. This indicates that our data would likely benefit from some type of transformation. Additionally, the Poisson model is probably not the best model for this data. Before moving on, we explore whether over dispersion is an issue.

Over/Underdispersion

We can get an approximation of the over/under dispersion using the Pearson's chi-squared statistic and degrees of freedom.

Based on the dispersion value, 0.8790033, it doesn't look like this dataset has a problem with over dispersion. We also see that our p-value is 1, which does not allow us to reject the null hypothesis. While over dispersion does not look to be an issue, we can run a quasi-Poisson model for completeness. The quasi-Poisson model integrates the dispersion parameter into the Poisson model.

quasi-Poisson Model

Being that our model is not fitting the data well, we elect to further explore the role of over-dispersion. We run a quasi-Poisson model, which estimates the dispersion parameter, in an attempt to mitigate its effects:

```
##
## Call:
## glm(formula = TARGET ~ ., family = quasipoisson, data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2226  -0.6497  -0.0051   0.4460   3.6809
##
## Coefficients:
##                        Estimate   Std. Error t value     Pr(>|t|)
## (Intercept)         1.051339182  0.188449198   5.579 0.0000000247 ***
## FixedAcidity        0.000158890  0.000768109   0.207     0.836124
## VolatileAcidity    -0.029658307  0.006117037  -4.848 0.0000012588 ***
## CitricAcid          0.004514248  0.005521775   0.818     0.413638
## ResidualSugar       0.000004695  0.000141697   0.033     0.973568
## Chlorides          -0.039321401  0.015094126  -2.605     0.009196 **
## FreeSulfurDioxide   0.000080942  0.000032025   2.527     0.011500 *
## TotalSulfurDioxide  0.000077999  0.000020744   3.760     0.000171 ***
## Density            -0.281342266  0.179600405  -1.566     0.117259
## pH                 -0.010161579  0.007040523  -1.443     0.148961
## Sulphates          -0.010744820  0.005139312  -2.091     0.036574 *
## Alcohol             0.004379406  0.001284224   3.410     0.000651 ***
## LabelAppeal.L       0.544779437  0.025792131  21.122      < 2e-16 ***
## LabelAppeal.Q      -0.070833964  0.021642391  -3.273     0.001067 **
## LabelAppeal.C       0.016622854  0.015261413   1.089     0.276083
## LabelAppeal^4       0.008013394  0.009703203   0.826     0.408904
## AcidIndex.L        -1.164544869  0.281026160  -4.144 0.0000343648 ***
## AcidIndex.Q        -0.013411143  0.273125469  -0.049     0.960838
## AcidIndex.C        -0.055515532  0.246334188  -0.225     0.821698
## AcidIndex^4        -0.368044272  0.236438577  -1.557     0.119586
## AcidIndex^5        -0.361244979  0.234378299  -1.541     0.123271
## AcidIndex^6         0.159862427  0.223241504   0.716     0.473945
## AcidIndex^7         0.220901576  0.202912390   1.089     0.276327
## AcidIndex^8         0.160527317  0.174844761   0.918     0.358577
## AcidIndex^9         0.085149651  0.142787142   0.596     0.550959
## AcidIndex^10        0.203999527  0.110370064   1.848     0.064579 .
## AcidIndex^11        0.199188765  0.082354838   2.419     0.015591 *
## AcidIndex^12        0.094929662  0.065024633   1.460     0.144341
## AcidIndex^13       -0.034039119  0.050954409  -0.668     0.504126
## STARS.L             0.967293059  0.015420138  62.729      < 2e-16 ***
## STARS.Q            -0.392953646  0.013254298 -29.647      < 2e-16 ***
## STARS.C             0.138960897  0.011359219  12.233      < 2e-16 ***
## STARS^4            -0.004127305  0.009274182  -0.445     0.656304
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 0.8767271)
##
##     Null deviance: 22861  on 12794  degrees of freedom
## Residual deviance: 13527  on 12762  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

In looking at the output above, we see that our standard error values have decreased slightly. However our null and residual deviance values haven't changed. Thus, it appears that the quasi-Poisson model offers a slightly better fit.

From this point, we move on to the exploration of a negative binomial model to compare its results to those of our Poisson and quasi-Poisson models.

Negative Binomial Model

Negative binomial regression can be used for over-dispersed count data. We see this when the conditional variance exceeds the conditional mean. While we are not convinced the data is over-dispersed, we run the model for sake of completeness (to be able to rule it out).

Here we use `glm.nb` as it uses maximum likelihood to estimate the link parameter, k. k corresponds to an assumption about the type of distribution of the response.

```
##
## Call:
## glm.nb(formula = TARGET ~ ., data = train2, init.theta = 40983.19218,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2225  -0.6497  -0.0051   0.4460   3.6808
##
## Coefficients:
##                        Estimate   Std. Error z value    Pr(>|z|)
## (Intercept)         1.051342169  0.201270944   5.224 0.000000176 ***
## FixedAcidity        0.000158887  0.000820371   0.194    0.846428
## VolatileAcidity    -0.029659130  0.006533238  -4.540 0.000005633 ***
## CitricAcid          0.004514361  0.005897478   0.765    0.443990
## ResidualSugar       0.000004699  0.000151338   0.031    0.975230
## Chlorides          -0.039322577  0.016121126  -2.439    0.014720 *
## FreeSulfurDioxide   0.000080945  0.000034204   2.367    0.017955 *
## TotalSulfurDioxide  0.000078003  0.000022155   3.521    0.000430 ***
## Density            -0.281346097  0.191820388  -1.467    0.142453
## pH                 -0.010162410  0.007519557  -1.351    0.176547
## Sulphates          -0.010745280  0.005488990  -1.958    0.050276 .
## Alcohol             0.004379338  0.001371602   3.193    0.001409 **
## LabelAppeal.L       0.544775975  0.027546693  19.776     < 2e-16 ***
## LabelAppeal.Q      -0.070834193  0.023114667  -3.064    0.002181 **
## LabelAppeal.C       0.016623570  0.016299641   1.020    0.307788
## LabelAppeal^4       0.008013658  0.010363349   0.773    0.439363
## AcidIndex.L        -1.164574150  0.300141543  -3.880    0.000104 ***
## AcidIndex.Q        -0.013404226  0.291703905  -0.046    0.963349
## AcidIndex.C        -0.055524752  0.263090664  -0.211    0.832850
## AcidIndex^4        -0.368044110  0.252521564  -1.457    0.144985
## AcidIndex^5        -0.361254512  0.250320142  -1.443    0.148973
## AcidIndex^6         0.159870436  0.238425079   0.671    0.502522
## AcidIndex^7         0.220903292  0.216713238   1.019    0.308044
## AcidIndex^8         0.160531547  0.186737058   0.860    0.389973
## AcidIndex^9         0.085149543  0.152499289   0.558    0.576598
## AcidIndex^10        0.204000196  0.117877236   1.731    0.083520 .
## AcidIndex^11        0.199188851  0.087956380   2.265    0.023535 *
```

```
## AcidIndex^12        0.094930597  0.069447301   1.367    0.171643
## AcidIndex^13       -0.034039713  0.054420025  -0.625    0.531643
## STARS.L             0.967294311  0.016469282  58.733     < 2e-16 ***
## STARS.Q            -0.392952487  0.014156101 -27.759     < 2e-16 ***
## STARS.C             0.138960593  0.012132074  11.454     < 2e-16 ***
## STARS^4            -0.004127144  0.009905193  -0.417    0.676924
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(40983.19) family taken to be 1)
##
##     Null deviance: 22860  on 12794  degrees of freedom
## Residual deviance: 13527  on 12762  degrees of freedom
## AIC: 45538
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  40983
##           Std. Err.:  34370
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -45469.63
```

In looking at the above model, we see a worsened performance relative to the quasi-Poisson and thus we can rule out the negative binomial model as an effective solution.

For the last modeling method to be explored, we'll look into the zero inflated count model.

Zero Inflated Count Models

Zero-inflated poisson regression is used to model count data that has an excess of zero counts. In looking at our histogram above of `TARGET`, we have about 20%+ of our target variable containing 0s. Thus, the model appears to have promise and we'll explore its performance relative to that of the quasi Poisson model (our strongest yet):

```
##
## Call:
## zeroinfl(formula = TARGET ~ ., data = train2)
##
## Pearson residuals:
##       Min        1Q    Median        3Q       Max
## -2.279049 -0.425715 -0.001251  0.381829  5.031806
##
## Count model coefficients (poisson with log link):
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.416953804         NA      NA       NA
## FixedAcidity        0.000376410         NA      NA       NA
## VolatileAcidity    -0.012268915         NA      NA       NA
## CitricAcid          0.000585853         NA      NA       NA
## ResidualSugar      -0.000055165         NA      NA       NA
## Chlorides          -0.025847701         NA      NA       NA
## FreeSulfurDioxide   0.000023596         NA      NA       NA
```

```
## TotalSulfurDioxide -0.000007671         NA      NA       NA
## Density            -0.248229094         NA      NA       NA
## pH                  0.003624977         NA      NA       NA
## Sulphates          -0.000389208         NA      NA       NA
## Alcohol             0.006786766         NA      NA       NA
## LabelAppeal.L       0.832723260         NA      NA       NA
## LabelAppeal.Q      -0.176099911         NA      NA       NA
## LabelAppeal.C       0.038402760         NA      NA       NA
## LabelAppeal^4       0.001718201         NA      NA       NA
## AcidIndex.L         0.043672719         NA      NA       NA
## AcidIndex.Q         0.044871219         NA      NA       NA
## AcidIndex.C        -0.103898401         NA      NA       NA
## AcidIndex^4        -0.242661238         NA      NA       NA
## AcidIndex^5        -0.110467299         NA      NA       NA
## AcidIndex^6        -0.029854108         NA      NA       NA
## AcidIndex^7        -0.069292475         NA      NA       NA
## AcidIndex^8        -0.066404144         NA      NA       NA
## AcidIndex^9        -0.028263868         NA      NA       NA
## AcidIndex^10        0.032236801         NA      NA       NA
## AcidIndex^11        0.003856555         NA      NA       NA
## AcidIndex^12       -0.004007973         NA      NA       NA
## AcidIndex^13        0.004730182         NA      NA       NA
## STARS.L             0.306977959         NA      NA       NA
## STARS.Q             0.013818248         NA      NA       NA
## STARS.C            -0.018949636         NA      NA       NA
## STARS^4             0.013350757         NA      NA       NA
##
## Zero-inflation model coefficients (binomial with logit link):
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -10.4540600         NA      NA       NA
## FixedAcidity         0.0009602         NA      NA       NA
## VolatileAcidity      0.1779257         NA      NA       NA
## CitricAcid          -0.0215562         NA      NA       NA
## ResidualSugar       -0.0006404         NA      NA       NA
## Chlorides            0.0878636         NA      NA       NA
## FreeSulfurDioxide   -0.0006443         NA      NA       NA
## TotalSulfurDioxide  -0.0009194         NA      NA       NA
## Density              0.8259237         NA      NA       NA
## pH                   0.1888600         NA      NA       NA
## Sulphates            0.1175738         NA      NA       NA
## Alcohol              0.0235019         NA      NA       NA
## LabelAppeal.L        2.6317664         NA      NA       NA
## LabelAppeal.Q       -0.5572984         NA      NA       NA
## LabelAppeal.C        0.1548684         NA      NA       NA
## LabelAppeal^4       -0.1100028         NA      NA       NA
## AcidIndex.L         13.9234419         NA      NA       NA
## AcidIndex.Q          7.9132778         NA      NA       NA
## AcidIndex.C          5.1430495         NA      NA       NA
## AcidIndex^4          2.4878436         NA      NA       NA
## AcidIndex^5         -0.2517390         NA      NA       NA
## AcidIndex^6         -3.9418030         NA      NA       NA
## AcidIndex^7         -4.3728180         NA      NA       NA
## AcidIndex^8         -3.9574156         NA      NA       NA
## AcidIndex^9         -2.3889156         NA      NA       NA
```

```
## AcidIndex^10        -1.4168196         NA      NA       NA
## AcidIndex^11        -0.6417687         NA      NA       NA
## AcidIndex^12        -0.0933077         NA      NA       NA
## AcidIndex^13         0.5059652         NA      NA       NA
## STARS.L            -22.2561267         NA      NA       NA
## STARS.Q             -0.1671574         NA      NA       NA
## STARS.C             10.1683165         NA      NA       NA
## STARS^4              8.0903380         NA      NA       NA
##
## Number of iterations in BFGS optimization: 68
## Log-likelihood: -2.031e+04 on 66 Df
```

We use the Vuong test to determine which model is better. Unfortunately, the `vuong` function does not accept the quasi-Poisson model as an input, but we can test the Poisson model which had similar, but slightly worse results.

```
## Vuong Non-Nested Hypothesis Test-Statistic:
## (test-statistic is asymptotically distributed N(0,1) under the
##  null that the models are indistinguishible)
## -------------------------------------------------------------
##               Vuong z-statistic             H_A    p-value
## Raw                   -44.54962 model2 > model1 < 2.22e-16
## AIC-corrected         -43.94246 model2 > model1 < 2.22e-16
## BIC-corrected         -41.67869 model2 > model1 < 2.22e-16
```

We can see that our p-value is significant which tells us that the zero-inflated regression model is fitting better than the Poisson model. From this point, we move forward with the assumption that this model is our best fitting model.

Now that we've identified which modeling approach to take, we proceed to engineer features, normalize and optimize based on AIC value.

Feature Engineering

We engineer features with the intention of expanding representative capabilities:

```
##
## Call:
## zeroinfl(formula = TARGET ~ ., data = train3)
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -2.30893 -0.41804 -0.01036  0.38156  5.04024
##
## Count model coefficients (poisson with log link):
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.463243228         NA      NA       NA
## FixedAcidity        0.000317823         NA      NA       NA
## VolatileAcidity    -0.012190045         NA      NA       NA
## CitricAcid          0.000438523         NA      NA       NA
## ResidualSugar      -0.000059857         NA      NA       NA
## Chlorides          -0.028723122         NA      NA       NA
## FreeSulfurDioxide   0.000021883         NA      NA       NA
```

```
## TotalSulfurDioxide -0.000006392         NA      NA       NA
## Density            -0.236719429         NA      NA       NA
## pH                  0.001451730         NA      NA       NA
## Sulphates           0.001466126         NA      NA       NA
## Alcohol             0.003646848         NA      NA       NA
## LabelAppeal.L       0.831911251         NA      NA       NA
## LabelAppeal.Q      -0.175326700         NA      NA       NA
## LabelAppeal.C       0.037202051         NA      NA       NA
## LabelAppeal^4       0.001659063         NA      NA       NA
## AcidIndex.L         0.048415698         NA      NA       NA
## AcidIndex.Q         0.051948112         NA      NA       NA
## AcidIndex.C        -0.090258847         NA      NA       NA
## AcidIndex^4        -0.224898666         NA      NA       NA
## AcidIndex^5        -0.095313438         NA      NA       NA
## AcidIndex^6        -0.024712093         NA      NA       NA
## AcidIndex^7        -0.072110329         NA      NA       NA
## AcidIndex^8        -0.078708593         NA      NA       NA
## AcidIndex^9        -0.041024441         NA      NA       NA
## AcidIndex^10        0.021070572         NA      NA       NA
## AcidIndex^11       -0.002109190         NA      NA       NA
## AcidIndex^12       -0.008785938         NA      NA       NA
## AcidIndex^13        0.000420258         NA      NA       NA
## STARS.L             0.336793884         NA      NA       NA
## STARS.Q             0.002999529         NA      NA       NA
## STARS.C            -0.029054908         NA      NA       NA
## STARS^4             0.026452460         NA      NA       NA
## hiD_S1             -0.000566303         NA      NA       NA
## loDSA1              0.023509308         NA      NA       NA
## HiSTARS_Alc1       -0.051811736         NA      NA       NA
##
## Zero-inflation model coefficients (binomial with logit link):
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -12.4477540         NA      NA       NA
## FixedAcidity         0.0012154         NA      NA       NA
## VolatileAcidity      0.1840525         NA      NA       NA
## CitricAcid          -0.0236669         NA      NA       NA
## ResidualSugar       -0.0006721         NA      NA       NA
## Chlorides            0.1219975         NA      NA       NA
## FreeSulfurDioxide   -0.0006364         NA      NA       NA
## TotalSulfurDioxide  -0.0009133         NA      NA       NA
## Density              2.7310468         NA      NA       NA
## pH                   0.1838282         NA      NA       NA
## Sulphates            0.1196387         NA      NA       NA
## Alcohol              0.0180206         NA      NA       NA
## LabelAppeal.L        2.6266835         NA      NA       NA
## LabelAppeal.Q       -0.5360188         NA      NA       NA
## LabelAppeal.C        0.1516024         NA      NA       NA
## LabelAppeal^4       -0.1065023         NA      NA       NA
## AcidIndex.L         14.0070638         NA      NA       NA
## AcidIndex.Q          8.0443656         NA      NA       NA
## AcidIndex.C          5.0890677         NA      NA       NA
## AcidIndex^4          2.4420963         NA      NA       NA
## AcidIndex^5         -0.3630338         NA      NA       NA
## AcidIndex^6         -3.9950661         NA      NA       NA
```

```
## AcidIndex^7         -4.4198596         NA      NA       NA
## AcidIndex^8         -3.9788406         NA      NA       NA
## AcidIndex^9         -2.4006096         NA      NA       NA
## AcidIndex^10        -1.4089643         NA      NA       NA
## AcidIndex^11        -0.6346996         NA      NA       NA
## AcidIndex^12        -0.0966578         NA      NA       NA
## AcidIndex^13         0.4971226         NA      NA       NA
## STARS.L            -22.5623855         NA      NA       NA
## STARS.Q              0.1769976         NA      NA       NA
## STARS.C             10.3956105         NA      NA       NA
## STARS^4              7.8250472         NA      NA       NA
## hiD_S1              -0.1912933         NA      NA       NA
## loDSA1              -0.2845549         NA      NA       NA
## HiSTARS_Alc1         0.9679218         NA      NA       NA
##
## Number of iterations in BFGS optimization: 75
## Log-likelihood: -2.03e+04 on 72 Df
```

It appears that with the addition of new features and some rounding, our model provides a significantly better fit than it did before.

We now continue dataset transformations by exploring the effect of normalization.

Normalization

Normalization of numeric variables reduced deviance, increased our standard error and led to an infinite AIC value (code available in Appendix). As such, we elected not to normalize our final model.

AIC Optimization

Before finalizing our model, we attempted to optimize our model's AIC value via application of the `stepAIC` function. The final steps of this function are shown below:

```
Step:  AIC=39545.35
TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide + TotalSulfurDioxide +
    pH + Sulphates + Alcohol + LabelAppeal + AcidIndex + STARS +
    loDSA + HiSTARS_Alc

                     Df   AIC
<none>                   39545
- Chlorides           2 39546
- loDSA               2 39546
- Alcohol             2 39548
- FreeSulfurDioxide   2 39548
- Sulphates           2 39550
- HiSTARS_Alc         2 39555
- pH                  2 39557
- VolatileAcidity     2 39563
- TotalSulfurDioxide  2 39579
- AcidIndex          18 39925
- LabelAppeal         8 41130
- STARS               8 42530
```

Our last attempt at feature selection greatly reduced dimensions (dropping 5 variables) while having a significant positive impact on AIC value (improved to 39,545.35). As such, we proceed with this model.

. . . . . .

Model Selection

The model we selected from all of those we've explored upto this point, our final model, is a Zero Inflated Count Model with missing values imputed (via PMM imputation), outliers handled, features engineered, and features selected (through AIC optimization).

The AIC value for our best model was significantly lower than all models we had seen prior, and while this number is nothing to get excited about (it's still quite high), we made significant inroads from when we set out with our "baseline" model.

We run our final model and inspect its coefficients:

```
##
## Call:
## zeroinfl(formula = TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide +
##     TotalSulfurDioxide + pH + Sulphates + Alcohol + LabelAppeal + AcidIndex +
##     STARS + loDSA + HiSTARS_Alc, data = train3)
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -2.30810 -0.41882 -0.01125  0.38191  4.97387
##
## Count model coefficients (poisson with log link):
##                       Estimate  Std. Error z value          Pr(>|z|)
## (Intercept)         1.22867842  0.06610617  18.586           < 2e-16 ***
## VolatileAcidity    -0.01231669  0.00671397  -1.834          0.066582 .
## Chlorides          -0.02911814  0.01645259  -1.770          0.076757 .
## FreeSulfurDioxide   0.00002105  0.00003452   0.610          0.541929
## TotalSulfurDioxide -0.00000722  0.00002203  -0.328          0.743122
## pH                  0.00129129  0.00735153   0.176          0.860570
## Sulphates           0.00166059  0.00578873   0.287          0.774215
## Alcohol             0.00369974  0.00182607   2.026          0.042757 *
## LabelAppeal.L       0.83213100  0.02964194  28.073           < 2e-16 ***
## LabelAppeal.Q      -0.17641273  0.02472292  -7.136 0.000000000000964 ***
## LabelAppeal.C       0.03734009  0.01726586   2.163          0.030568 *
## LabelAppeal^4       0.00148726  0.01077667   0.138          0.890235
## AcidIndex.L         0.03969741  0.30328703   0.131          0.895862
## AcidIndex.Q         0.04403579  0.29524594   0.149          0.881436
## AcidIndex.C        -0.10048184  0.26729394  -0.376          0.706974
## AcidIndex^4        -0.23191176  0.25638981  -0.905          0.365715
## AcidIndex^5        -0.10276645  0.25277073  -0.407          0.684331
## AcidIndex^6        -0.02988082  0.23978506  -0.125          0.900828
## AcidIndex^7        -0.07571027  0.21837179  -0.347          0.728814
## AcidIndex^8        -0.07835339  0.18935900  -0.414          0.679034
## AcidIndex^9        -0.03920741  0.15603603  -0.251          0.801604
## AcidIndex^10        0.02399683  0.12168910   0.197          0.843673
## AcidIndex^11       -0.00006590  0.09137541  -0.001          0.999425
## AcidIndex^12       -0.00808141  0.07256029  -0.111          0.911319
## AcidIndex^13        0.00007360  0.05697921   0.001          0.998969
```

```
## STARS.L              0.33723422  0.01956604  17.236           < 2e-16 ***
## STARS.Q              0.00307680  0.01501957   0.205          0.837687
## STARS.C             -0.02919367  0.01296621  -2.252          0.024353 *
## STARS^4              0.02636982  0.01087747   2.424          0.015340 *
## loDSA1               0.02635579  0.02380216   1.107          0.268170
## HiSTARS_Alc1        -0.05226222  0.01583633  -3.300          0.000966 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                        Estimate  Std. Error z value       Pr(>|z|)
## (Intercept)         -9.8323274 225.5914055  -0.044       0.965235
## VolatileAcidity      0.1846139   0.0437355   4.221 0.000024306692 ***
## Chlorides            0.1255092   0.1072458   1.170       0.241882
## FreeSulfurDioxide   -0.0006514   0.0002361  -2.759       0.005790 **
## TotalSulfurDioxide  -0.0009186   0.0001486  -6.180 0.000000000639 ***
## pH                   0.1841863   0.0477278   3.859       0.000114 ***
## Sulphates            0.1026980   0.0379650   2.705       0.006829 **
## Alcohol              0.0174400   0.0097644   1.786       0.074087 .
## LabelAppeal.L        2.6286285   0.2495834  10.532        < 2e-16 ***
## LabelAppeal.Q       -0.5496481   0.2107973  -2.607       0.009121 **
## LabelAppeal.C        0.1550803   0.1380995   1.123       0.261454
## LabelAppeal^4       -0.1098077   0.0755631  -1.453       0.146170
## AcidIndex.L         13.9838380 428.2291658   0.033       0.973950
## AcidIndex.Q          7.9826264 411.5255633   0.019       0.984524
## AcidIndex.C          5.1561154 312.4716610   0.017       0.986835
## AcidIndex^4          2.4745055 225.2295847   0.011       0.991234
## AcidIndex^5         -0.2926696 217.5853897  -0.001       0.998927
## AcidIndex^6         -3.9677768 233.6218686  -0.017       0.986450
## AcidIndex^7         -4.3978330 217.0440027  -0.020       0.983834
## AcidIndex^8         -3.9671513 168.9438363  -0.023       0.981266
## AcidIndex^9         -2.3990040 110.5882490  -0.022       0.982693
## AcidIndex^10        -1.4017614  60.6112987  -0.023       0.981549
## AcidIndex^11        -0.6267831  27.2430855  -0.023       0.981645
## AcidIndex^12        -0.0930161   9.5479640  -0.010       0.992227
## AcidIndex^13         0.4954594   2.2818768   0.217       0.828109
## STARS.L            -22.6770127 693.1334899  -0.033       0.973901
## STARS.Q              0.1540346 585.8045129   0.000       0.999790
## STARS.C             10.4230641 535.7316605   0.019       0.984478
## STARS^4              7.8394569 342.4885091   0.023       0.981738
## loDSA1              -0.2938185   0.1651079  -1.780       0.075149 .
## HiSTARS_Alc1         0.9743707   0.8552041   1.139       0.254560
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 62
## Log-likelihood: -2.03e+04 on 62 Df
```

Below the model call, we observe first an an output block containing Poisson regression coefficients and then an output block corresponding to the zero inflation model. Each model contains standard errors, z-scores, and p-values for the coefficients where the second (zero inflation) block includes logit coefficients for predicting excess zeros along with their standard errors, z-scores, and p-values.

To better interpret our zero inflation coefficients, we take the exponent of our log coefficients:

```
##        count_(Intercept)    count_VolatileAcidity          count_Chlorides
```

```
##                 3.41671                 0.98776                 0.97130
##  count_FreeSulfurDioxide count_TotalSulfurDioxide               count_pH
##                 1.00002                 0.99999                 1.00129
##         count_Sulphates           count_Alcohol     count_LabelAppeal.L
##                 1.00166                 1.00371                 2.29821
##     count_LabelAppeal.Q     count_LabelAppeal.C     count_LabelAppeal^4
##                 0.83827                 1.03805                 1.00149
##       count_AcidIndex.L       count_AcidIndex.Q       count_AcidIndex.C
##                 1.04050                 1.04502                 0.90440
##       count_AcidIndex^4       count_AcidIndex^5       count_AcidIndex^6
##                 0.79302                 0.90234                 0.97056
##       count_AcidIndex^7       count_AcidIndex^8       count_AcidIndex^9
##                 0.92708                 0.92464                 0.96155
##      count_AcidIndex^10      count_AcidIndex^11      count_AcidIndex^12
##                 1.02429                 0.99993                 0.99195
##      count_AcidIndex^13           count_STARS.L           count_STARS.Q
##                 1.00007                 1.40107                 1.00308
##           count_STARS.C           count_STARS^4            count_loDSA1
##                 0.97123                 1.02672                 1.02671
##      count_HiSTARS_Alc1        zero_(Intercept)    zero_VolatileAcidity
##                 0.94908                 0.00005                 1.20275
##          zero_Chlorides  zero_FreeSulfurDioxide zero_TotalSulfurDioxide
##                 1.13373                 0.99935                 0.99908
##                 zero_pH          zero_Sulphates            zero_Alcohol
##                 1.20224                 1.10816                 1.01759
##      zero_LabelAppeal.L      zero_LabelAppeal.Q      zero_LabelAppeal.C
##                13.85476                 0.57715                 1.16775
##      zero_LabelAppeal^4        zero_AcidIndex.L        zero_AcidIndex.Q
##                 0.89601           1183323.98802              2929.61528
##        zero_AcidIndex.C        zero_AcidIndex^4        zero_AcidIndex^5
##                173.48921                11.87583                 0.74627
##        zero_AcidIndex^6        zero_AcidIndex^7        zero_AcidIndex^8
##                 0.01892                 0.01230                 0.01893
##        zero_AcidIndex^9       zero_AcidIndex^10       zero_AcidIndex^11
##                 0.09081                 0.24616                 0.53431
##       zero_AcidIndex^12       zero_AcidIndex^13           zero_STARS.L
##                 0.91118                 1.64125                 0.00000
##           zero_STARS.Q           zero_STARS.C           zero_STARS^4
##                 1.16653             33626.31034              2538.82572
##            zero_loDSA1       zero_HiSTARS_Alc1
##                 0.74541                 2.64950
```

Being that the exponent is the inverse of log, our resulting coefficients can be interpreted for predicting excess zeros. A higher variable value means a higher probability of excess zeros for that variable. From above, we observe that our highest coefficients and thus the variables most likely to have excess zeros are: `zero_HiSTARS_Alc1` (1797.67), `zero_STARS.C` (304.32), and `zero_AcidIndex.L` (63.86). In addition to the interpretive power of predicting the location and magnitude of zeros per variable, we may also interpret this finding as a sortof re-affirmation of our earlier finding that `STARS` and `AcidIndex` are highly correlated to our `TARGET` variable.

Moving forward, we cast our predictions.

Predictions

We select our test dataset, feed our model, and output the first 6 entries of our predictions and the corresponding summary statistics:

```
## 1 2 3 4 5 6
## 2 4 2 2 1 6
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000001 1.777882 3.042171 3.061625 4.149138 7.330538
```

.....

References

- GVIF
- Adjusting for Overdispersion in Poisson Regression
- Overdispersion and Underdispersion in Negative Binomial/Poisson Regression
- Diagnostic Plots for Count Regression

Appendix R Statistical Code

Below you'll find all the code we'd used in working our way through this assignment:

```
# Libraries and Options

knitr::opts_chunk$set(echo = F, warning = F, message = F, eval = T,
                      fig.height = 5, fig.width = 10)

library(ggplot2)
library(knitr)
library(inspectdf)
library(corrplot)
library(tidyverse)
library(tidyr)
library(car)
library(AER)
library(faraway)
library(mice)
library(vcd)
library(caret)
library(kableExtra)
library(boot)
library(pscl) #predict.zeroinfl
library(MASS)
library(VIM) #KNN imputation


options(scipen = 9)
set.seed(123)
```

```r
boxplot_depend_vs_independ <- function(df_train, target_name) {

  train_int_names <- df_train %>% select_if(is.numeric)
  int_names <- names(train_int_names)
  myGlist <- vector('list', length(int_names))
  names(myGlist) <- int_names

  for (i in int_names) {

   myGlist[[i]] <-
       ggplot(df_train, aes_string(x = target_name, y = i)) +
        geom_boxplot(color = 'steelblue', outlier.color = 'firebrick',
                     outlier.alpha = 0.35) +
        labs(title = paste0(i,' vs target'), y = i, x= 'target') +
        theme_minimal() +
        theme(
          plot.title = element_text(hjust = 0.45),
          panel.grid.major.y =  element_line(color = "grey",
                                             linetype = "dashed"),
          panel.grid.major.x = element_blank(),
          panel.grid.minor.y = element_blank(),
          panel.grid.minor.x = element_blank(),
          axis.ticks.x = element_line(color = "grey")
        )

      }

    myGlist <- within(myGlist, rm(target_name))
    gridExtra::grid.arrange(grobs = myGlist, ncol = 3)
}

plot_corr_matrix <- function(dataframe, significance_threshold){
  title <- paste0('Correlation Matrix for significance > ',
                  significance_threshold)

  df_cor <- dataframe %>% mutate_if(is.character, as.factor)

  df_cor <- df_cor %>% mutate_if(is.factor, as.numeric)
  #run a correlation and drop the insignificant ones
  corr <- cor(df_cor)
  #prepare to drop duplicates and correlations of 1
  corr[lower.tri(corr,diag=TRUE)] <- NA
  #drop perfect correlations
  corr[corr == 1] <- NA
  #turn into a 3-column table
  corr <- as.data.frame(as.table(corr))
  #remove the NA values from above
  corr <- na.omit(corr)
  #select significant values
  corr <- subset(corr, abs(Freq) > significance_threshold)
  #sort by highest correlation
  corr <- corr[order(-abs(corr$Freq)),]
  #print table
```

```
  # print(corr)
  #turn corr back into matrix in order to plot with corrplot
  mtx_corr <- reshape2::acast(corr, Var1~Var2, value.var="Freq")

  #plot correlations visually
  corrplot(mtx_corr,
           title=title,
           mar=c(0,0,1,0),
           method='color',
           tl.col="black",
           na.label= " ",
           addCoef.col = 'black',
           number.cex = .9)
}
test_URL <- paste0('https://raw.githubusercontent.com/AngelClaudio/',
                   'data-sources/master/csv/wine-evaluation-data.csv')

train_URL <- paste0('https://raw.githubusercontent.com/AngelClaudio/',
                    'data-sources/master/csv/wine-training-data.csv')

train <- readr::read_csv(train_URL)
test <- readr::read_csv(test_URL) %>% dplyr::rename('INDEX' = 'IN')

train$dataset <- 'train'
test$dataset <- 'test'

final_df <- rbind(train, test)

download.file(url = paste0('https://raw.githubusercontent.com/AngelClaudio/',
                           'data-sources/master/Picts/wine_degustation.png'),
          destfile = "image.png",
          mode = 'wb')
knitr::include_graphics(path = "image.png")
glimpse(train)

summary(train)

#drop INDEX
final_df <- final_df %>% dplyr::select(-INDEX)
VIM::aggr(final_df %>% filter(dataset == 'train'), col=c('green','red'), numbers=T, sortVars=T,
          cex.axis = .7,
          ylab=c("Proportion of Data", "Combinations and Percentiles"))
# missing_count <- colSums(is.na(train))
# percent <- (colSums(is.na(train))/dim(train)[1]) * 100
#
#
# missing_vals <- data.frame(missing_count = missing_count, percent = percent) %>% arrange(desc(percent
#
# missing_vals
final_df$STARS <- final_df$STARS %>% tidyr::replace_na('Not Rated')

final_df  <- final_df %>% dplyr::mutate(
  LabelAppeal = factor(LabelAppeal, levels = sort(unique(final_df$LabelAppeal)), ordered = TRUE),
```

```
  AcidIndex = factor(AcidIndex, levels = sort(unique(final_df$AcidIndex)), ordered = TRUE),
  STARS = factor(STARS, levels = c('Not Rated', '1','2','3','4'), ordered = TRUE)
)

#final_df

#Variable distributions
inspectdf::inspect_num(final_df %>% filter(dataset == 'train')) %>%
  show_plot()

#Utilize custom-built box plot generation function

train_int_names <- final_df %>% filter(dataset == 'train') %>% select_if(is.numeric)
int_names <- names(train_int_names)
myGlist <- vector('list', length(int_names))
#myGlist$TARGET <- NULL   how do we get rid of TARGET here?
names(myGlist) <- int_names

for (i in int_names) {

 myGlist[[i]] <-
    ggplot(train_int_names) +
    aes_string(x = as.factor(train_int_names$TARGET), y = i) +

    geom_jitter(color = "gray", alpha = 0.35) +
    geom_boxplot(color = 'steelblue', outlier.color = 'firebrick',
                     outlier.alpha = 0.35) +
      labs(title = paste0(i,' vs target'), y = i, x= 'target') +
      theme_minimal() +
      theme(
        plot.title = element_text(hjust = 0.45),
        panel.grid.major.y =  element_line(color = "grey",
                                           linetype = "dashed"),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(),
        axis.ticks.x = element_line(color = "grey")
      )

    }

  myGlist <- within(myGlist, rm(target_name))
  gridExtra::grid.arrange(grobs = myGlist, ncol = 4)

table(final_df$LabelAppeal, final_df$TARGET)
table(final_df$AcidIndex, final_df$TARGET)
table(final_df$STARS, final_df$TARGET)
#Utilize custom-built correlation matrix generation function
plot_corr_matrix(final_df %>% filter(dataset == 'train'), -1)


#MS Note: we can remove the 1st two chunks, just wanted to document applied methods.
```

```r
#kNN imputation - WORSE performance
##motivating source: https://www.youtube.com/watch?v=u8XvfhBdbMw (~7:00)
#summary(train3) #identify variables with missing values
# final_df <- VIM::kNN(final_df, variable = c('STARS','ResidualSugar', 'Chlorides', 'FreeSulfurDioxide'
#summary(imp_train3) #verify no NA's

#Predictive mean matching - WORSE performance
final_df <- mice(data = final_df, m = 1, method = "pmm", seed = 500)
final_df <- mice::complete(final_df,1)

#NA value removal - same performance
#dim(train3) #12795 x 17
#comp_train3 <- train3[complete.cases(train3), ]
#dim(comp_train3) #9383 x 17

final_df <- final_df %>%
  mutate(TARGET = ifelse(dataset == "test", NA, TARGET)) %>%
  dplyr::select(-contains("_imp"))
colSums(is.na(final_df))
VIM::aggr(final_df %>% filter(dataset == 'train'), col=c('green','red'), numbers=T, sortVars=T,
          cex.axis = .7,
          ylab=c("Proportion of Data", "Combinations and Percentiles"))
#Filter for training dataset and remove dataset variable:
train2 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)

#Baseline model (model 1): prior to data transformations
model_1 <- glm(TARGET ~ ., train2, family = poisson(link = "log"))
summary(model_1)

car::vif(model_1)
influencePlot(model_1)
#We did not end up using this section since predictions were not able to be cast when it was included
cooksD <- cooks.distance(model_1)
influential <- as.numeric(names(cooksD)[(cooksD > (4 * mean(cooksD, na.rm = TRUE)))])
#influential
#comp_train3[influential,] #verify outliers - 121 rows
#final_df <- final_df[-influential,]
#final_df <- final_df[-c(1530,3258,5401,5538,3297,5467,5604,9205,10975),]
train2 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)

#Outlier model (model 3):
#model_2 <- glm(TARGET ~ ., train2, family = poisson(link = "log"))
#summary(model_2)
#glm.diag.plots(model_2)
dispersiontest(model_1)
model_3 <- glm(TARGET ~ ., family=quasipoisson, train2)
summary(model_3)
model4 <- glm.nb(TARGET ~ ., data = train2)
summary(model4)
model5 <- zeroinfl(TARGET ~ ., data = train2)
summary(model5)
pscl::vuong(model_1, model5)
round(final_df$Alcohol,0)
```

```
#Create new features (started with 1st, 3rd quartile values then adjusted):
##high Density, Sulphates
final_df$hiD_S <- as.factor(ifelse(final_df$Density >= 1.00 & final_df$Sulphates >= 0.00, 1, 0))
##low Density, Sulphate, and Alcohol
final_df$loDSA <- as.factor(ifelse(final_df$Density < 1.00 & final_df$Sulphates < 0.40 & final_df$Alcoh
##High LabelAppeal, STARS, AcidIndex
#final_df$LA_STARS_AI <- as.factor(ifelse(as.numeric(final_df$LabelAppeal) > 0 & as.numeric(final_df$ST
##High AcidIndex, Alcohol
#final_df$hiAI_Alc <- as.factor(ifelse(as.numeric(final_df$AcidIndex) > 9 & final_df$Alcohol > 9, 1, 0)
##Sweet Spot: Hi stars, Lower alcohol
final_df$HiSTARS_Alc <- as.factor(ifelse(as.numeric(final_df$STARS) > 2 & final_df$Alcohol < 11.40, 1, 0

#head(final_df) #verify

## Rounding
final_df$FixedAcidity <- round(final_df$FixedAcidity,0)
final_df$VolatileAcidity <- round(final_df$VolatileAcidity,1)
final_df$CitricAcid <- round(final_df$CitricAcid,1)
final_df$ResidualSugar <- round(final_df$ResidualSugar,0)
final_df$Chlorides <- round(final_df$Chlorides,1)
final_df$Density <- round(final_df$Density,2)
final_df$pH <- round(final_df$pH,0)
final_df$Sulphates <- round(final_df$Sulphates,1)
final_df$Alcohol <- round(final_df$Alcohol,0)

#Filter for training dataset and remove dataset variable:
train3 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)
#head(train3)

#Baseline model (model 1): prior to data transformations
model_6 <- zeroinfl(TARGET ~ ., data = train3)
summary(model_6)

#Normalize numeric variables to 0-to-1 scale: reduced deviance BUT caused infinite AIC ...
norm_minmax <- function(x){(x- min(x)) /(max(x)-min(x))}

original_target <- train3$TARGET

norm_df <- train3 %>%
  mutate_if(is.numeric, norm_minmax)

norm_df$TARGET <- original_target

model_7 <- zeroinfl(TARGET ~ ., data = norm_df)
summary(model_7)

download.file(url = paste0('https://github.com/christianthieme/',
                           'Business-Analytics-and-Data-Mining-with-Regression/',
                           'raw/main/AIC.jpg'),
          destfile = "image2.jpg",
          mode = 'wb')
knitr::include_graphics(path = "image2.jpg")
model_8 <- zeroinfl(TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide + TotalSulfurDioxide +
```

```
    pH + Sulphates + Alcohol + LabelAppeal + AcidIndex + STARS +
    loDSA + HiSTARS_Alc, data = train3)

summary(model_8)
round(exp(coef(model_8)),5)
#Prep data set: select test then drop dataset variable
final_test <- final_df %>% filter(dataset == 'test') %>% dplyr::select(-dataset)

#Baseline model predictions
#predict1 <- predict(model_1, newdata=final_test, type="response")
#round(predict1,0)
#summary(predict1)

#Cast predictions using model_8
predict2 <- predict(model_8, newdata=final_test, type="response")
head(round(predict2,0))
summary(predict2)
```