DATA 621 - Business Analytics and Data Mining

Fall 2020 - Group 2 - Homework #5

Avraham Adler, Samantha Deokinanan, Amber Ferger, John Kellogg, Bryan Persaud, Jeff Shamp

11/28/2020

Contents

ASSIGNMENT

The assignment for HW5 is to analyze and model a dataset containing approximately 12,000 records representing commercially available wines. The `TARGET` response variable represents the number of sample cases of wine purchased by wine distribution companies after sampling that 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 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.

The objective of this assignment 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. *HINT:* Sometimes, the fact that a variable is missing is actually predictive of the target. You can only use the variables given to you (or variables that you derive from the variables provided).

DATA EXPLORATION

Variables

The data is composed of the following variables:

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
INDEX	Identification Variable (do not use)	None
TARGET	Number of Cases Purchased	None
AcidIndex	Proprietary method of testing total acidity of wine by using a weighted average	
Alcohol	Alcohol Content	
Chlorides	Chloride content of wine	
CitricAcid	Citric Acid Content	
Density	Density of Wine	
FixedAcidity	Fixed Acidity of Wine	

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
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	

There are 12795 observations. All of these predictors are numeric, although `LabelAppeal`, `AcidIndex` and `STARS` all appear to be ordinal factors and not true numerics. For the purpose of this assignment, we can treat them as integers.

Missing Data

There are a lot of missing values for some of the predictors.

Table 2: Predictors with Missing Observations

Predictors	Missing	Percentage
STARS	3359	26.25
Sulphates	1210	9.46
TotalSulfurDioxide	682	5.33
Alcohol	653	5.10
FreeSulfurDioxide	647	5.06
Chlorides	638	4.99
ResidualSugar	616	4.81
pH	395	3.09
FixedAcidity	0	0.00
VolatileAcidity	0	0.00
CitricAcid	0	0.00
Density	0	0.00
LabelAppeal	0	0.00
AcidIndex	0	0.00

`STARS` is especially sparse, but as noted in the assignment, this may be an indication in and of its own. It is reasonable to assume that the vintners of a good wine want it to be rated. If they do not submit it for rating, that may be an indication of their lack of faith in the wine. How to factor this into the analysis will be decided individually by the modelers in this assignment. The other missing variables are all less than 10%, and their handling via imputation or otherwise will be done on a model-by-model basis as well.

Summary Statistics and Graphs

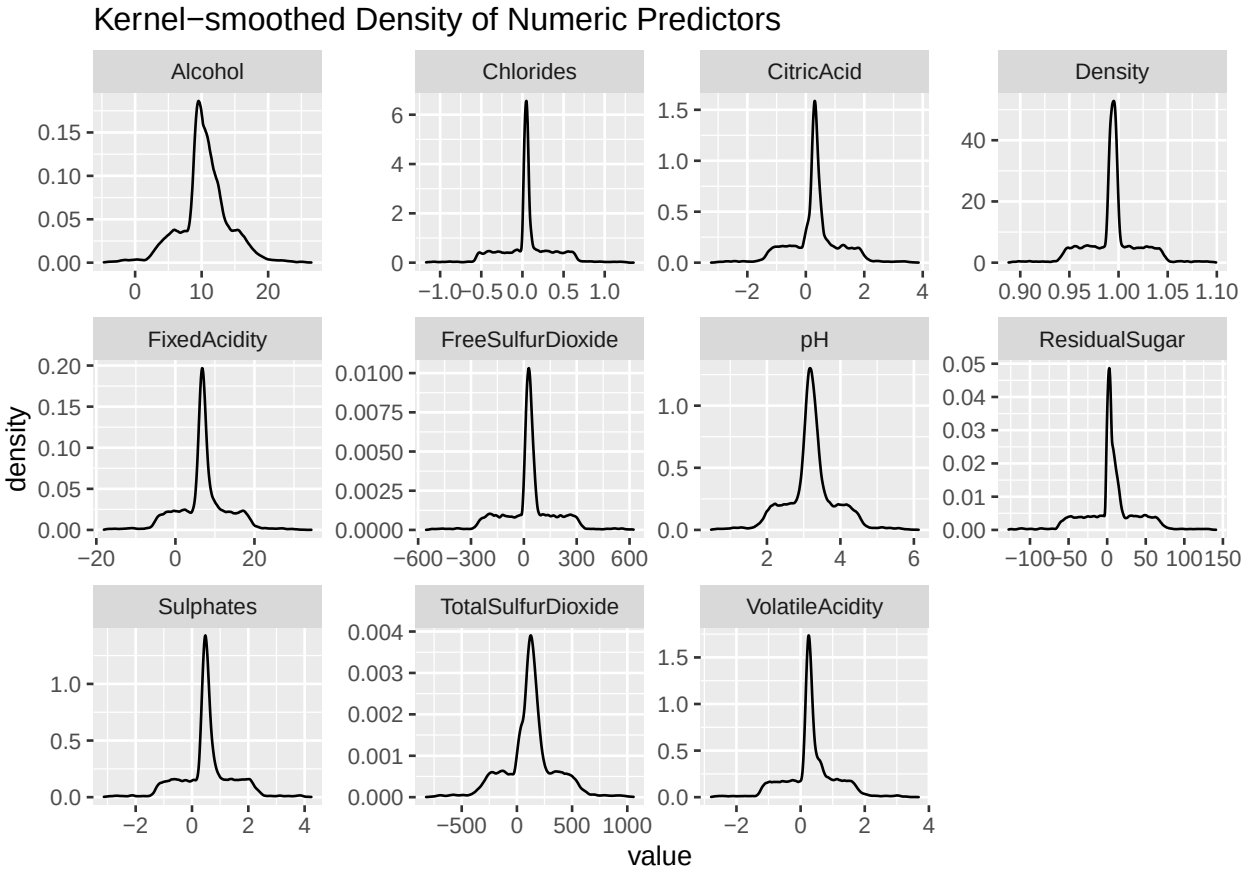
As all of the data are numeric, we can investigate the distributions of the predictors both tabularly and graphically.

The numeric predictor variables have the following summary statistics, ignoring missing values:

Table 3: Summary Statistitics for Numeric Variables

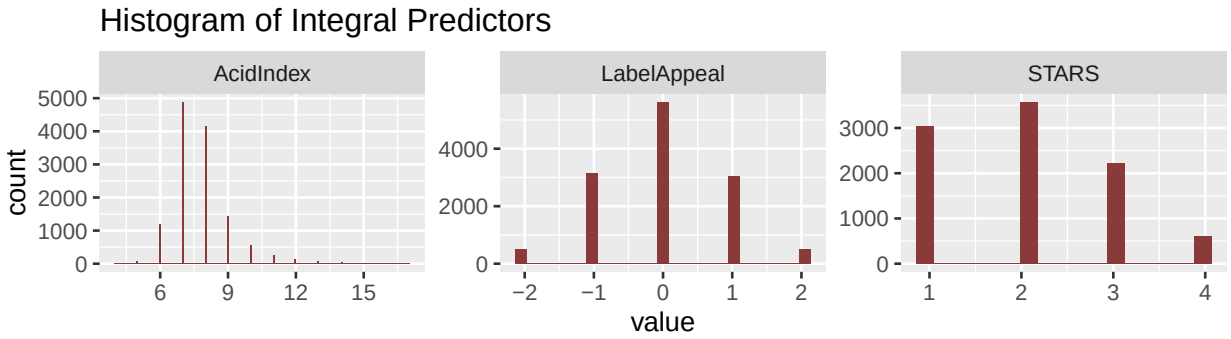
metric	Mean	SD	Min	Q1	Median	Q3	Max	IQR
AcidIndex	7.773	1.324	4.000	7.000	8.000	8.000	17.000	1.000
Alcohol	10.489	3.728	-4.700	9.000	10.400	12.400	26.500	3.400
Chlorides	0.055	0.318	-1.171	-0.031	0.046	0.153	1.351	0.184
CitricAcid	0.308	0.862	-3.240	0.030	0.310	0.580	3.860	0.550
Density	0.994	0.027	0.888	0.988	0.994	1.001	1.099	0.013
FixedAcidity	7.076	6.318	-18.100	5.200	6.900	9.500	34.400	4.300
FreeSulfurDioxide	30.846	148.715	-555.000	0.000	30.000	70.000	623.000	70.000
LabelAppeal	-0.009	0.891	-2.000	-1.000	0.000	1.000	2.000	2.000
ResidualSugar	5.419	33.749	-127.800	-2.000	3.900	15.900	141.150	17.900
STARS	2.042	0.903	1.000	1.000	2.000	3.000	4.000	2.000
Sulphates	0.527	0.932	-3.130	0.280	0.500	0.860	4.240	0.580
TotalSulfurDioxide	120.714	231.913	-823.000	27.000	123.000	208.000	1057.000	181.000
VolatileAcidity	0.324	0.784	-2.790	0.130	0.280	0.640	3.680	0.510
pH	3.208	0.680	0.480	2.960	3.200	3.470	6.130	0.510

A kernel-smoothed density plot of the distributions of the non-integer numeric predictors is below, followed by a histogram of the integral-valued predictors.



The numeric predictors all look to be basically symmetrical, but non-Gaussian in that there is a strong spike near the median, but the tails on either side are thicker than would be for a normal distribution with that

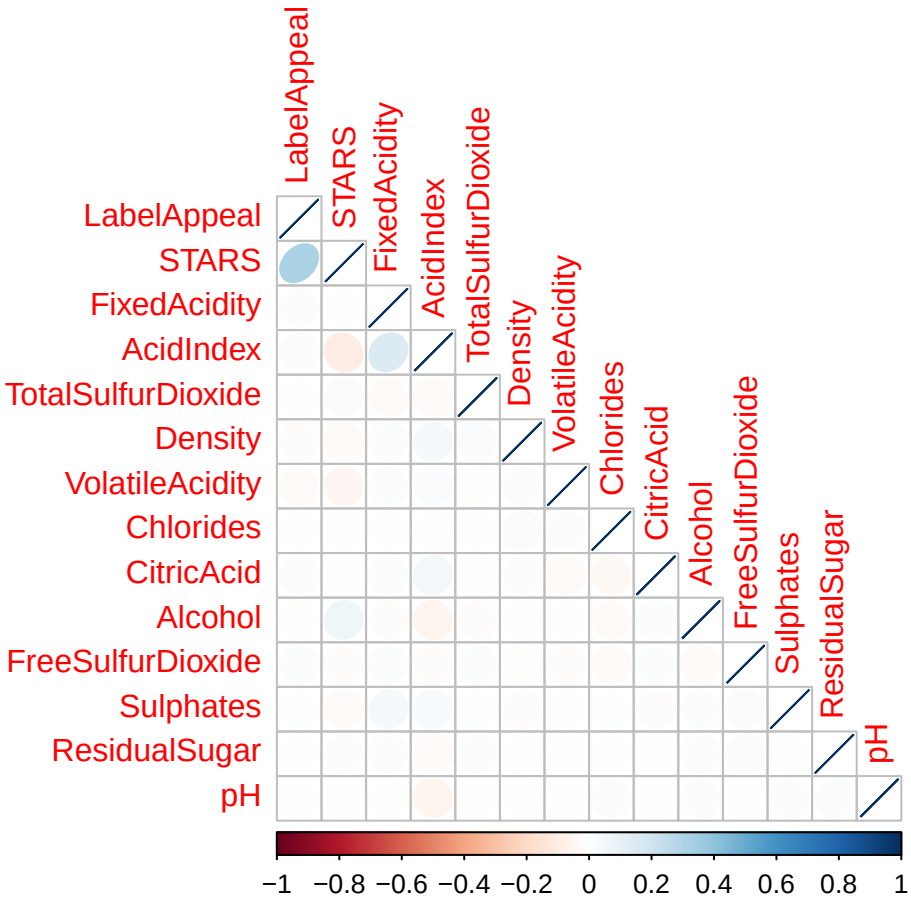
low of a standard deviation. This is also why a boxplot would be a poor graphical indicator, as the spike at the medians would lead to a compressed inter-quartile range and a plethora of outliers.



Of these three, `LabelAppeal` seems to be the most "normal" of the lot; `STARS` and `AcidIndex` look to be more Poisson in shape.

Correlations

The corrgram below graphically represents the correlations between the numeric predictor variables, when ignoring the missing variables.

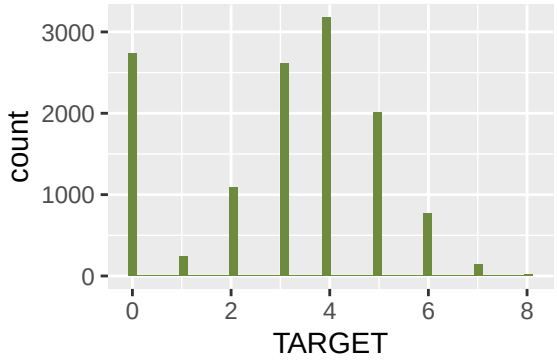


There is very little correlation between the variables. The only pairs with some level correlation are:

- `STARS` being positively correlated with `LabelAppeal`
  - This is interesting. Could it be that wine connoisseurs are impacted by the visual appearance of the label and not just the flavor?
- `AcidIndex` having some positive correlation with `FixedAcidity`
  - One may have suspected a higher correlation, to be frank.
- `AcidIndex` having some *negative* correlation with `STARS`
  - Wine reviewers may not like too much acidity, it seems.

TARGET variable

Lastly, the complete dataset exhibits an average of 3.03 cases being bought, with the distribution below:



DATA PREPARATION

First, preparation necessary for all the models equally will be performed. The subsequent imputation and feature generation, if necessary will be discussed in a separate subsection for each modeler.

Training & Testing Split

All the models will be trained on the same approximately 70% of the training set, reserving 30% for validation of which model to select for the count estimation on the supplied evaluation set.

Poisson Model #1 & Negative Binomial Model #1

Missing Data

The missing data can be split into two groups: `STARS` and the rest. For all the others, I do not think there is signal encoded in the "missingness" and thus will use some form of imputation. `STARS` is different as discussed in the DATA EXPLORATION section. Therefore, I will create a new feature called `Rated` which will be true if `STARS` is missing, and then only use this variable and its interaction with `STARS` in the model, and not `STARS` by itself.

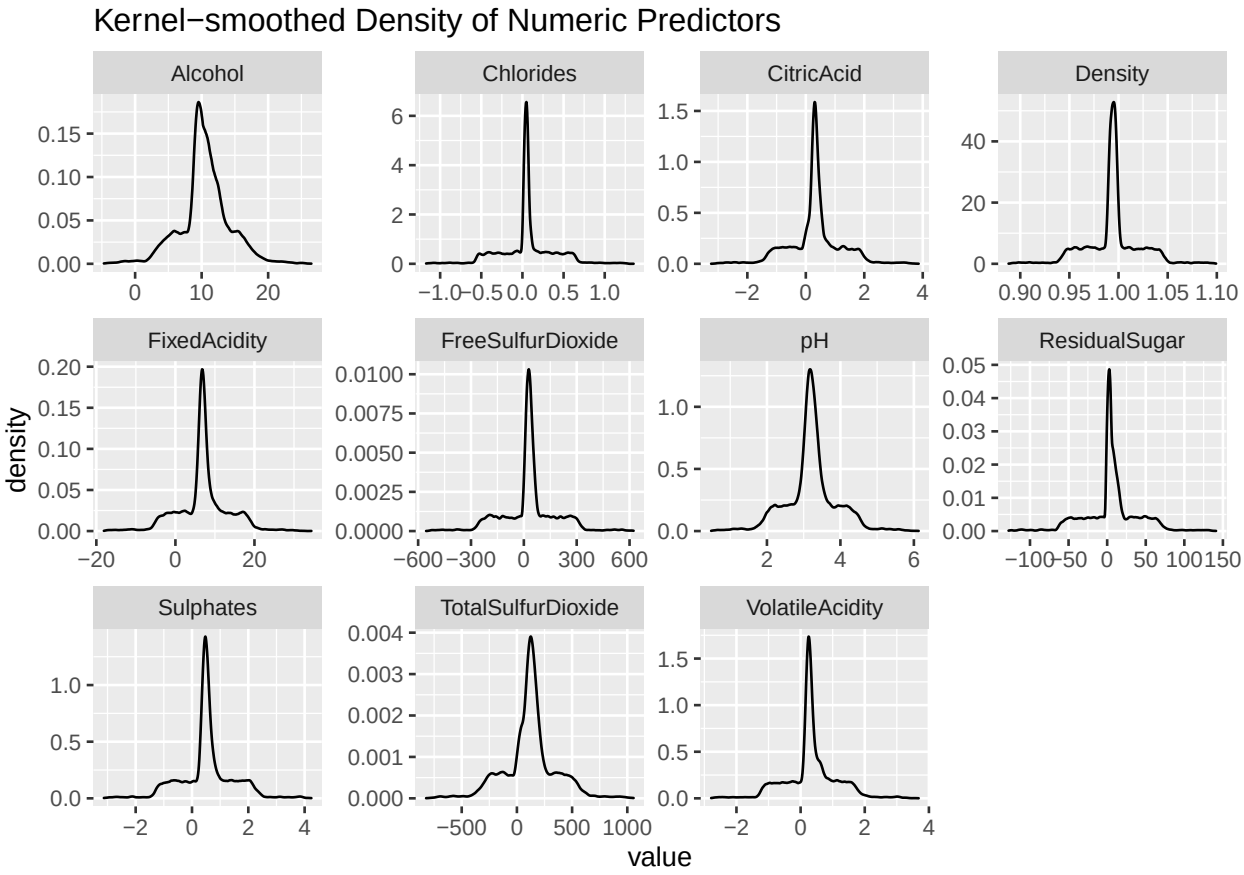
Imputation

Imputation will be handled through bagging. Instead of looking at a nearest neighbors approach, which also requires centering and scaling, once can create a set of bagged trees. As per Kuhn (2019):

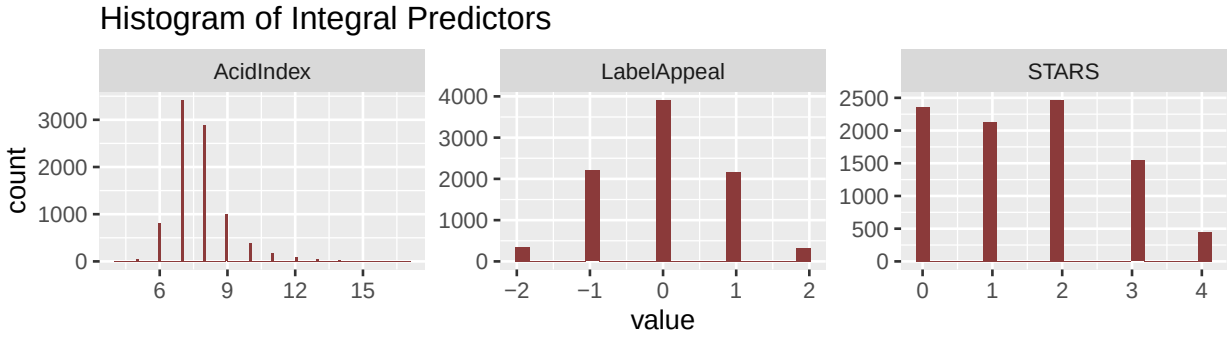
> *For each predictor in the data, a bagged tree is created using all of the other predictors in the training set. When a new sample has a missing predictor value, the bagged model is used to predict the value. While, in theory, this is a more powerful method of imputing, the computational costs are much higher than the nearest neighbor technique.*

The bagged trees will use `STARS` as a predictor, as that contains valuable information, but prior to the imputation, the missing `STARS` values will be replaced by 0 so as not to be imputed, as per the explanation above. The preprocessing will also check for near-zero value predictors.

It will be instructive to compare the shapes of the distributions after imputation to those before imputation.



Thankfully, none of the shapes appears to have changed significantly, which implies the imputation was in line with the original distributions. Now for the histograms.



Here too, `AcidIndex` and `LabelAppeal` look very similar to their pre-imputation distribution. What stands out is `STARS`. Both the Poisson and the negative binomial models are either unimodal or have to consecutive values sharing the mode, due to their inherent nature as discrete distributions. What is seen here is that 1 star is empirically less common than 0 (unrated) or 2 stars. On the one hand, this could simply be noise (process variance) or parameter error (finite sample size). On the other, it could indicate the need for a zero-inflated version of the counting distributions.

Poisson Model #2 & Multiple Linear Regression Model #1

The right balance of the chemical properties of the wine is essential to make it taste, look, and smell delightful, in addition to having a long shelf-life. Typically, citric acid and alcohol are the main positive attributes a

wine should have, while volatile acidity and sulfates are the main negative ones. However, the dataset has negative values instead of near-zero in some of these variables describing the chemical properties of the wine. Because little information is given whether or not these variables were normalized, absolute values will be taken assuming there was an error made during data collection.

Lastly, to handle missing data for `STARS`, they will be replaced by 0, while the chemical properties will be imputed. Because the predictive mean matching imputation method combines the standard linear regression and the nearest-neighbor imputation approach, Model Set #2 elected to fill in the missing values of a continuous variable by using mean-matching imputation. Afterward, the data is pre-processed to fulfill the assumption of normality using the Yeo-Johnson transformation (2000). This technique attempts to find the value of lambda that minimizes the Kullback-Leibler distance between the normal distribution and the transformed distribution. This method has the advantage of working without having to worry about the domain of x.

Multiple Linear Regression #2 & Negative Binomial Model #2

Missing Data

These models will deal with missing numerical data by imputing the NA's using the Multivariate imputation by chained equations (MICE) method. Multiple imputation involves creating multiple predictions for each missing value, helping to account for the uncertainty in the individual imputations.

BUILD MODELS

Poisson Model #1 & Negative Binomial Model #1

The same person is building these two models, and the model setups will be similar. Each model will follow a similar format, with only the family changing. Each will start with the saturated model including all the individual predictors ***except*** for `STARS`. There will also be the following interactions:

- `Rated` and `STARS`
  - As discussed above, by having `Rating` on its own and this interaction the effect of both having a rating and the effect of the rating level can be measured, without penalizing unrated wines a second time for a 0.
- The full four-way interaction between `AcidIndex`, `FixedAcidity`, `VolatileAcidity`, and `pH`
  - There are **four** separate predictors relating to acidity. Despite their Pearson product-moment correlations being low, I am suspicious of too much weight being given to acidity. By including their interactions, effects can be tempered, or magnified, as necessary. `CitricAcid` is not included here as is not merely an acid but a flavor provider as well. Also, five-way interactions lead to migraines.
- `FreeSulfurDioxide` and `TotalSulfurDioxide`
  - Again, a case of multiple theoretically related predictors.
- `LabelAppeal` with `Rated` and `LabelAppeal` with `Rated:STARS`
  - This is a later addition. The top three predictors in magnitude were `LabelAppeal`, `Rated`, and `Rated:STARS`. With all three being so strong, there may be interactions between them. However, as `STARS` alone has been removed from contention, only the interactions between the two others will be considered.

Poisson Model # 1

For the Poisson model, a forward and backwards stepwise procedure based on AIC will be used, and the model with the lowest AIC on the training set will be the selected to be tested against the testing set. Using stepAIC precludes the use of k-fold cross validation.

Table 4: Model 1 Poisson Regression Output

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.6445126	0.0711463	9.058977	0.0000000
FixedAcidity	0.0124723	0.0063319	1.969764	0.0488654
Chlorides	-0.0414426	0.0195207	-2.123013	0.0337527
FreeSulfurDioxide	0.0001406	0.0000501	2.808675	0.0049746
TotalSulfurDioxide	0.0001207	0.0000285	4.242120	0.0000221
Sulphates	-0.0114631	0.0068371	-1.676611	0.0936185
Alcohol	0.0035384	0.0016977	2.084244	0.0371380
AcidIndex	-0.0542179	0.0092789	-5.843129	0.0000000
RatedRated	0.6677926	0.0258530	25.830393	0.0000000
`FreeSulfurDioxide:TotalSulfurDioxide`	-0.0000003	0.0000002	-1.734851	0.0827673
`FixedAcidity:AcidIndex`	-0.0016864	0.0008254	-2.043253	0.0410274
`VolatileAcidity:AcidIndex`	-0.0044320	0.0010190	-4.349341	0.0000137
`pH:AcidIndex`	-0.0025848	0.0011945	-2.163933	0.0304695
`STARS:RatedRated`	0.1912813	0.0077136	24.797789	0.0000000
`LabelAppeal:RatedRated`	0.2468805	0.0191838	12.869198	0.0000000
`LabelAppeal:STARS:RatedRated`	-0.0301714	0.0079675	-3.786833	0.0001526

Model Checking A Poisson model should have a dispersion parameter close to 1, as the Poisson by definition has its mean and variance equal. This model has a dispersion parameter of 0.891 which is deemed acceptable.

Coefficient Discussion Similar to the master data set, the training data set has an empirical mean purchased cases of 3.02. Having no other information, the Poisson model expects about 1.91 cases, due to the intercept value of 0.6445126. This means that the model identifies factors which lean more to *increasing* purchases than to decreasing purchases.

The absolute magnitude of the coefficients makes sense. Factors related to acidity tend to lower the number of cases bought. The factors with the greatest magnitude effect are all positive, and they are the fact of being rated, and the interactions between that and the rating level and label appeal. Keeping everything else equal, merely being rated increases the number of cases by about $e^{0.6677926} \approx 1.95$! Each additional star is predicted to contribute another 1.2 cases to the total purchase. Interestingly, `LabelAppeal` has fallen out of the model; its contributions are through its interactions. As surmised, looking at all three individually may have proven too strong, as the three-way interaction coefficient tempers the others.

All the coefficients are significant at at least the 5% level except for Sulphates, but in terms of AIC, leaving it in provided a better model than did taking it out.

Variable Importance

Table 5: Poisson Model Variable Importance

Coefficients	Overall
RatedRated	25.83
`STARS:RatedRated`	24.80
`LabelAppeal:RatedRated`	12.87
AcidIndex	5.84
`VolatileAcidity:AcidIndex`	4.35
TotalSulfurDioxide	4.24
`LabelAppeal:STARS:RatedRated`	3.79
FreeSulfurDioxide	2.81
`pH:AcidIndex`	2.16
Chlorides	2.12

Coefficients	Overall
Alcohol	2.08
FixedAcidity:AcidIndex	2.04
FixedAcidity	1.97
FreeSulfurDioxide:TotalSulfurDioxide	1.73
Sulphates	1.68

The variable importance can also be seen from the trace of the step AIC procedure. The further down the list below a variable preceded by a - is, the more important it is, as its removal would cause a greater disruption to the deviance and the AIC. The three rating variables, being rated, the rating level, and the label appeal, are the most important variables when it comes to future case purchases.

It is also interesting to note that there are a number of variables whose inclusion as predictors would result in a lower deviance model than the selected, but said drop in deviance was not enough to offset the possible parameter error. These include VolatileAcidity, LabelAppeal, pH, Density, ResidualSugar, VolatileAcidity:pH, and CitricAcid.

```
                                          Df Deviance   AIC
<none>                                         9550.7 31902
+ VolatileAcidity                          1   9549.4 31903
- Sulphates                                1   9553.5 31903
- `FreeSulfurDioxide:TotalSulfurDioxide`   1   9553.7 31903
+ LabelAppeal                              1   9549.9 31903
+ pH                                       1   9550.3 31904
+ Density                                  1   9550.4 31904
+ ResidualSugar                            1   9550.4 31904
+ `VolatileAcidity:pH`                     1   9550.5 31904
+ CitricAcid                               1   9550.6 31904
- FixedAcidity                             1   9554.6 31904
+ `FixedAcidity:pH:AcidIndex`              1   9550.7 31904
+ `FixedAcidity:VolatileAcidity`           1   9550.7 31904
+ `FixedAcidity:pH`                        1   9550.7 31904
+ `FixedAcidity:VolatileAcidity:pH`        1   9550.7 31904
+ `VolatileAcidity:pH:AcidIndex`           1   9550.7 31904
+ `FixedAcidity:VolatileAcidity:AcidIndex` 1   9550.7 31904
+ `FixedAcidity:VolatileAcidity:pH:AcidIndex` 1 9550.7 31904
- `FixedAcidity:AcidIndex`                 1   9554.9 31904
- Alcohol                                  1   9555.1 31904
- Chlorides                                1   9555.2 31904
- `pH:AcidIndex`                           1   9555.4 31905
- FreeSulfurDioxide                        1   9558.6 31908
- `LabelAppeal:STARS:RatedRated`           1   9565.2 31914
- TotalSulfurDioxide                       1   9568.7 31918
- `VolatileAcidity:AcidIndex`              1   9569.6 31919
- AcidIndex                                1   9585.3 31935
- `LabelAppeal:RatedRated`                 1   9719.7 32069
- `STARS:RatedRated`                       1  10152.1 32501
- RatedRated                               1  10247.3 32597
```

Negative Binomial Model # 1

As the dispersion from the Poisson model was less than 1, at first blush, it is not expected that the negative binomial model will outperform the Poisson.

Table 6: Model 1 Negative Binomial Regression Output

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.6997832	0.0784097	8.9246971	0.0000000
FixedAcidity	0.0123393	0.0063341	1.9480863	0.0514046
Chlorides	-0.0411280	0.0195218	-2.1067747	0.0351371
FreeSulfurDioxide	0.0001407	0.0000501	2.8096359	0.0049598
TotalSulfurDioxide	0.0001206	0.0000285	4.2395532	0.0000224
pH	-0.0186003	0.0091214	-2.0392068	0.0414294
Sulphates	-0.0115323	0.0068379	-1.6865233	0.0916951
Alcohol	0.0035455	0.0016977	2.0883469	0.0367666
LabelAppeal	-0.0189542	0.0209811	-0.9033938	0.3663169
AcidIndex	-0.0624112	0.0086310	-7.2310798	0.0000000
RatedRated	0.6715318	0.0261944	25.6364947	0.0000000
FreeSulfurDioxide:TotalSulfurDioxide	-0.0000003	0.0000002	-1.7394668	0.0819527
FixedAcidity:AcidIndex	-0.0016684	0.0008256	-2.0207354	0.0433072
AcidIndex:VolatileAcidity	-0.0044354	0.0010191	-4.3524334	0.0000135
RatedRated:STARS	0.1912883	0.0077141	24.7971608	0.0000000
LabelAppeal:RatedRated	0.2659146	0.0284217	9.3560429	0.0000000
LabelAppeal:RatedRated:STARS	-0.0302032	0.0079678	-3.7906395	0.0001503

Coefficient Discussion Similar to the Poisson model, having no other information, the negative binomial model expects about 2.01 cases, due to the intercept value of 0.6997832. This means that this model as well identifies factors which lean more to *increasing* purchases than to decreasing purchases.

The absolute magnitude of the coefficients makes less sense in this model than in the Poisson. The factors related to appeal do increase the number of cases bought. However, the factors related to acidity tend to show more back-and-forth interactions than in the Poisson model.

Also, there are more "insignificant" coefficients in this model, but these are all variables for which the AIC would increase should they have been removed.

Variable Importance

Table 7: Negative Binomial Model Variable Importance

Coefficients	Overall
RatedRated	25.64
RatedRated:STARS	24.80
LabelAppeal:RatedRated	9.36
AcidIndex	7.23
AcidIndex:VolatileAcidity	4.35
TotalSulfurDioxide	4.24
LabelAppeal:RatedRated:STARS	3.79
FreeSulfurDioxide	2.81
Chlorides	2.11
Alcohol	2.09
pH	2.04
FixedAcidity:AcidIndex	2.02
FixedAcidity	1.95
FreeSulfurDioxide:TotalSulfurDioxide	1.74
Sulphates	1.69
LabelAppeal	0.90

The trace of the step procedure is less complete when using the `glm.nb` function in `R` and is not listed here.

Lastly, the deviance for the negative binomial model is actually lower than that for the Poisson model, but its AIC is higher. Based on AIC only, the Poisson model's parsimony beats out the negative binomial's explanatory powers.

Table 8: Poisson 1 and NegBinom 1 Goodness-of-Fit Comparison

Model	Deviance	AIC
Poisson 1	9550.721	31901.98
Negative Binomial 1	9550.027	31905.97

Poisson Model #2 & Multiple Linear Regression Model #1

The `TARGET` variable has 2734 observations with zero values. This could be a result of the manufacturer not releasing the wine into the market, or that there were no orders placed. Therefore, backward stepwise variable elimination for a zero-inflated Poisson (ZIP) analysis will be investigated to understand the conditions. This model is further compared to the same Poisson model to determine whether there is an improvement.

In the ZIP model, the independent variable, $TARGET_i$, take zero values $TARGET_i \sim 0$ with the probability ω_i or values from Poisson distribution $TARGET_i \sim \text{Pois}(\lambda_i)$ with probability $(1-\omega_i)$. For $i = 1, \ldots, n$, it can be written as:

$$P(Y_i = y_i) = \begin{cases} \omega_i + (1-\omega_i)e^{-\lambda_i}) & \text{if } y_i = 0 \\ (1-\omega_i)\frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!} & \text{if } y_i > 0 \end{cases}$$

Therefore, the zero-inflated Poisson model have two parameters, λ_i and ω_i, and linked with the predictor variables with the following:

$$\log\left(\frac{\omega_i}{1-\omega_i}\right) = \sum_{j=1}^{t} \gamma_{ji} Z_{ji}$$

$$\log\left(\lambda_i\right) = \sum_{j=1}^{k} \beta_{ji} X_{ji}$$

where $Z_1, \ldots, Z_l$ are the dependent variables for the first equation and $X_1, \ldots, X_k$ for the second one. The expected value and variance of the number of cases purchased for the i^{th} wine in the ZIP model are respectively:

$$\text{E}(Y_i) = \lambda_i(1-\omega_i)$$
$$\text{Var}(Y_i) = (1-\omega_i)(\lambda_i - \omega_i\lambda_i^2)$$

Similar to Poisson regression, the ZIP model assumes that the average number of cases purchased equals the variance. For more information, refer to Zero-Inflated Poisson Regression.

Poisson Model #2

Zero-inflated count models are two-component mixture models combining a point mass at zero with a proper count distribution. Thus, there are two sources of zeros: zeros may come from both the point mass and from the count component.

Table 9: Zero Inflated Poisson Model #2 Count Output

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.1551566	0.0506392	22.811490	0.0000000
Alcohol	0.0084491	0.0020513	4.118965	0.0000381
LabelAppeal	0.2322692	0.0075435	30.790457	0.0000000
AcidIndex	-0.0213374	0.0056891	-3.750552	0.0001764
STARS	0.1739679	0.0106999	16.258817	0.0000000

Table 10: Zero Inflated Poisson Model #2 Zero Output

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-3.2105180	0.4220949	-7.606152	0.0000000
VolatileAcidity	1.0186079	0.3253492	3.130814	0.0017432
CitricAcid	-0.6568394	0.2869301	-2.289196	0.0220679
FreeSulfurDioxide	-0.0478439	0.0236301	-2.024706	0.0428975
TotalSulfurDioxide	-0.0751285	0.0113563	-6.615602	0.0000000
pH	0.2047448	0.0555731	3.684244	0.0002294
Sulphates	1.0211364	0.2848177	3.585227	0.0003368
Alcohol	0.0304149	0.0133842	2.272447	0.0230595
LabelAppeal	0.7145872	0.0515254	13.868650	0.0000000
AcidIndex	0.4202252	0.0303711	13.836369	0.0000000
STARS	-3.1491864	0.0893872	-35.230837	0.0000000

Model Checking

```
## 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                   -49.09068 model2 > model1 < 2.22e-16
## AIC-corrected         -48.95022 model2 > model1 < 2.22e-16
## BIC-corrected         -48.45156 model2 > model1 < 2.22e-16
```

This model has a dispersion parameter of 0.476 which suggests under-dispersion (i.e., variance $<$ mean). This parameter indicates how many times larger the variance is than the mean. Since the dispersion was less than one, the conditional variance is smaller than the conditional mean. Models that ignore under-dispersion will be overly conservative and may fail to detect significant patterns. Generalized Poisson (GP) and Conway-Maxwell-Poisson (CMP) distributions are better choices for such modeling because they can handle both over-dispersion and under-dispersion. But for the scope of this assignment, it's apparent, from the Vuong test with the same Poisson model and the ZIP model, that the zero-inflated Poisson model is a slight improvement over the Poisson model. Therefore, the ZIP model will still be considered among others.

Coefficient Discussion All of the predictors in both the count and inflation portions of the model are statistically significant. This model fits the data significantly better than the null model, i.e., the intercept-only model. The count model contains information about variables that the parent Poisson model used to estimate `TARGET` on the condition that `TARGET` > 0. The coefficients for all 4 regression variables are statistically significant at a 99% confidence level. The coefficient for `AcidIndex` is negative, meaning that as the total acidity of wine goes up by a unit, the number of purchased cases goes down by 2%. One the other hand, for the other variables, i.e. `Alcohol`, `LabelAppeal` and `STARS`, as these increase, the number of purchased cases will also increase. In particular, `LabelAppeal` can account for a 26% increase in the number of wine cases sold if the visual appeal of the wine scores increase by one unit and all other variables held constant. Whereas, there is a 19% increase in cases sold for each increase in the wine rating by a team of experts, `STARS`.

The zero model contains information about variables that the nested Logistic Regression model has been used to estimate the probability of whether or not cases are purchased. Examining the variables with a negative regression coefficient, this suggests that as the factor for the specific chemical properties increases, the probability of no case being purchase decreases. This is in line with our intuition, particularly for the sulfur dioxide levels. When a wine has an unusually small percentage of the Free and Total SO_2, this suggests the wine is chemically and/or microbially unstable. Thus, if a wine manufacturer were to increase the sulfur dioxide levels by one unit, the odds that the wine would certainly not be purchased would decrease by a factor of 0.92 to 0.95. However, `LabelAppeal` is among the positive coefficients for the zero model. This is interpreted as the higher a rating is for a bottle of wine, the more likely there will be no case purchase, and it doesn't seem reasonable.

Multiple Linear Regrssion Model #1

Multiple Linear Regression Model #1 will be using the step function has an option to use a process that both adds and removes variables in the model. It uses AIC (Akaike information criterion) as a selection criterion. A full model and a null are defined, and then the function will follow a procedure to find the model with the lowest AIC.

Table 11: Multiple Linear Regression Model #1 Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.6514356	0.1487622	17.823311	0.0000000
STARS	1.5772775	0.0194755	80.987819	0.0000000
LabelAppeal	0.4419578	0.0160496	27.537042	0.0000000
AcidIndex	-0.1937162	0.0106928	-18.116475	0.0000000
TotalSulfurDioxide	0.0235825	0.0039231	6.011211	0.0000000
VolatileAcidity	-0.5136805	0.1094072	-4.695127	0.0000027
FreeSulfurDioxide	0.0252754	0.0081548	3.099454	0.0019448
Alcohol	0.0140245	0.0044893	3.123950	0.0017901
CitricAcid	0.2616322	0.0965745	2.709123	0.0067590
Chlorides	-0.1603614	0.0585925	-2.736894	0.0062145
pH	-0.0367182	0.0185925	-1.974887	0.0483117
Sulphates	-0.1527785	0.0952889	-1.603320	0.1088993

Coefficient Discussion Contributing coefficients are those which lie within the 95% level of significance. A positive coefficient indicates that as the value of the predictor increases, the response variable also tends to increase. A negative coefficient suggests that as the predictor increases, the response variable tends to decrease.

The intercept itself suggests that with no other information about the wine, there will be cases sold. The other positive predictors suggest that increasing `STARS` and `LabelAppeal` will result in an increase in the number of case purchased. These make sense intuitively. Moreover, as discussed in the Poisson Model #2, the right balance of SO_2 lead to better wine, thus the coefficients are rather small and have a smaller effect when it is increased. This reasoning is similar for `Alcohol` levels. Next, `CitricAcid`, commonly used as an acid supplement during fermentation, helps winemakers boost the acidity of their wine and stabilize ferric haze. So, it makes sense that the higher `CitricAcid` is, the more cases would be sold. But once again, these need to be balance right.

On the contrary, an increase in a unit of volatile acidity results in the decrease of case purchases. Volatile acidity is caused by bacteria or certain yeasts. This also can increase the acid index and pH if it is high. This is why sulfur dioxide is used to carefully control it. In all, this model should not be taken as fixed, when it comes to chemical processes, the wine manufacturer needs to take care of what to increase and decrease in order to reach equilibrium.

Negative Binomial Model #2 & Multiple Linear Regrssion Model #2

Negative Binomial Model #2

Baseline Model We will start with a simple negative binomial regression model to serve as a baseline. This includes all variables in the dataset.

Table 12: Negative Binomial Regression Model #2 - Base Output

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.342	0.232	5.788	0.000
FixedAcidity	-0.001	0.001	-0.718	0.473
VolatileAcidity	-0.038	0.008	-4.901	0.000
CitricAcid	0.005	0.007	0.727	0.467
ResidualSugar	0.000	0.000	0.663	0.507
Chlorides	-0.065	0.019	-3.427	0.001
FreeSulfurDioxide	0.000	0.000	2.515	0.012
TotalSulfurDioxide	0.000	0.000	4.268	0.000
Density	-0.140	0.227	-0.615	0.539
pH	-0.027	0.009	-2.968	0.003
Sulphates	-0.013	0.006	-1.963	0.050
Alcohol	0.003	0.002	1.889	0.059
LabelAppeal	0.142	0.007	19.492	0.000
AcidIndex	-0.096	0.005	-17.832	0.000
STARS	0.334	0.007	50.202	0.000

Immediately, we can see a few variables that do not meet the 0.05 p-value threshold.

Enhanced Model Backwards stepwise regression is performed and the result is a model with the following variables *removed*: `FixedAcidity`, `CitricAcid`, `ResidualSugar`, and `Density`. Because the `Sulphates` variable is very close to the 0.05 p-value threshold of significance, we will remove it as well.

Table 13: Negative Binomial Regression Model #2 - Final Output

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.199	0.058	20.539	0.000
VolatileAcidity	-0.039	0.008	-4.964	0.000
Chlorides	-0.066	0.019	-3.467	0.001
FreeSulfurDioxide	0.000	0.000	2.479	0.013
TotalSulfurDioxide	0.000	0.000	4.269	0.000
pH	-0.027	0.009	-2.991	0.003
Alcohol	0.003	0.002	1.892	0.058
LabelAppeal	0.142	0.007	19.477	0.000
AcidIndex	-0.097	0.005	-18.225	0.000
STARS	0.335	0.007	50.308	0.000

Coefficient Discussion The intercept of the final model tells us that there is typically ~1 case of wine purchased. Some things to note:

- Stars and label appeal contribute the most positively to the number of cases of wine purchased. Intuitively, this makes sense - the higher the rating and the prettier the bottle, the more cases we would expect to be sold.

- Free Sulfur Dioxide and Total Sulfur Dioxide have a very tiny effect on the number of cases sold (rounded to 3 decimal places, the impact is 0!)
- All of the remaining variables except alcohol have a negative effect on the number of cases purchased.

Multiple Linear Regrssion Model # 2

One drawback to traditional multiple linear regression models is that the forecast counts are not integers. We will deal with this by rounding the predictions to the nearest whole number.

Baseline Model We will start with a simple linear model to serve as a baseline. This includes all variables in the dataset.

Table 14: Multiple Regression Model #2 - Base Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.260	0.571	5.711	0.000
FixedAcidity	-0.002	0.002	-0.674	0.500
VolatileAcidity	-0.113	0.019	-5.888	0.000
CitricAcid	0.017	0.018	0.994	0.320
ResidualSugar	0.000	0.000	0.925	0.355
Chlorides	-0.200	0.047	-4.266	0.000
FreeSulfurDioxide	0.000	0.000	3.036	0.002
TotalSulfurDioxide	0.000	0.000	4.830	0.000
Density	-0.436	0.561	-0.777	0.437
pH	-0.068	0.022	-3.066	0.002
Sulphates	-0.031	0.016	-1.924	0.054
Alcohol	0.012	0.004	3.038	0.002
LabelAppeal	0.432	0.018	24.369	0.000
AcidIndex	-0.235	0.012	-20.093	0.000
STARS	1.145	0.018	64.470	0.000

Most of the variables in the dataset are significant, but we can immediately see a few that are above the 0.05 p-value threshold.

Enhanced Model Backwards stepwise regression is performed and the result is a model with the following variables *removed*: `FixedAcidity`, `CitricAcid`, `ResidualSugar`, and `Density`.

Table 15: Multiple Regression Model #2 - Final Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.829	0.136	20.813	0.000
VolatileAcidity	-0.113	0.019	-5.926	0.000
Chlorides	-0.201	0.047	-4.290	0.000
FreeSulfurDioxide	0.000	0.000	3.045	0.002
TotalSulfurDioxide	0.000	0.000	4.846	0.000
pH	-0.068	0.022	-3.074	0.002
Sulphates	-0.031	0.016	-1.943	0.052
Alcohol	0.012	0.004	3.058	0.002
LabelAppeal	0.432	0.018	24.362	0.000
AcidIndex	-0.236	0.012	-20.551	0.000
STARS	1.145	0.018	64.530	0.000

Coefficient Discussion The intercept of the final model tells us that there are typically ~3 cases of wine purchased. Some things to note:

- As expected, the number of stars has the largest, positive impact on the number of cases sold. This makes sense – the higher the rating of a wine, the more of it we would expect to be purchased.
- Label appeal has the second largest impact on the number of cases sold. They say one shouldn't judge a book by its cover (or in this case, a wine by its label), but it does appear to have quite a significant impact on sales.
- Alcohol also has a small positive impact on the number of cases purchased. This means that the more alcohol a wine has, the more cases of it will be purchased.
- Acid Index, Sulphates, pH, Chlorides, and VolatileAcidity all have a negative impact on the number of cases sold.

SELECT MODELS

Model Selection Criteria

We will look at RMSE and MAE as our metrics and select the model which performs best on both. If no model performs best on both, the AIC of the model results on the training set will be used to break the tie, as the AIC is an estimate of the average out-of-sample performance of the model.

Model Test Results

The test set data needs to be processed identically to the training set. Since the number of cases needs to be integral the predicted values will be rounded to the nearest integer.

Poisson Model 1

Poisson Model 1 has an RMSE of 1.4, an MAE of 0.99, and a training AIC of 31902.

Poisson Model 2

Poisson Model 2 has an RMSE of 1.3, an MAE of 0.91, and a training AIC of 28632.

Negative Binomial Model 1

Negative Binomial Model 1 has an RMSE of 2.7, an MAE of 2.3, and a training AIC of 31906.

Negative Binomial Model 2

Negative Binomial Model 2 has an RMSE of 2.6, an MAE of 2.3, and a training AIC of 33634.

Multiple Linear Regression Model 1

Multiple Linear Regression Model 2 has an RMSE of 1.3, an MAE of 1, and a training AIC of 30202.

Multiple Linear Regression Model 2

Multiple Linear Regression Model 1 has an RMSE of 1.5, an MAE of 1.1, and a training AIC of 31726.

Comparsion

Table 16: Model Results on Test Set

Models	RMSE	MAE	TrainAIC
Poisson 1	1.352	0.986	31901.98
Poisson 2	1.315	0.911	28631.91
Neg. Binom. 1	2.667	2.252	31905.97
Neg. Binom. 2	2.643	2.316	33634.08
MultiLinear 1	1.347	1.015	30201.88
MultiLinear 2	1.463	1.145	31725.71

Selection

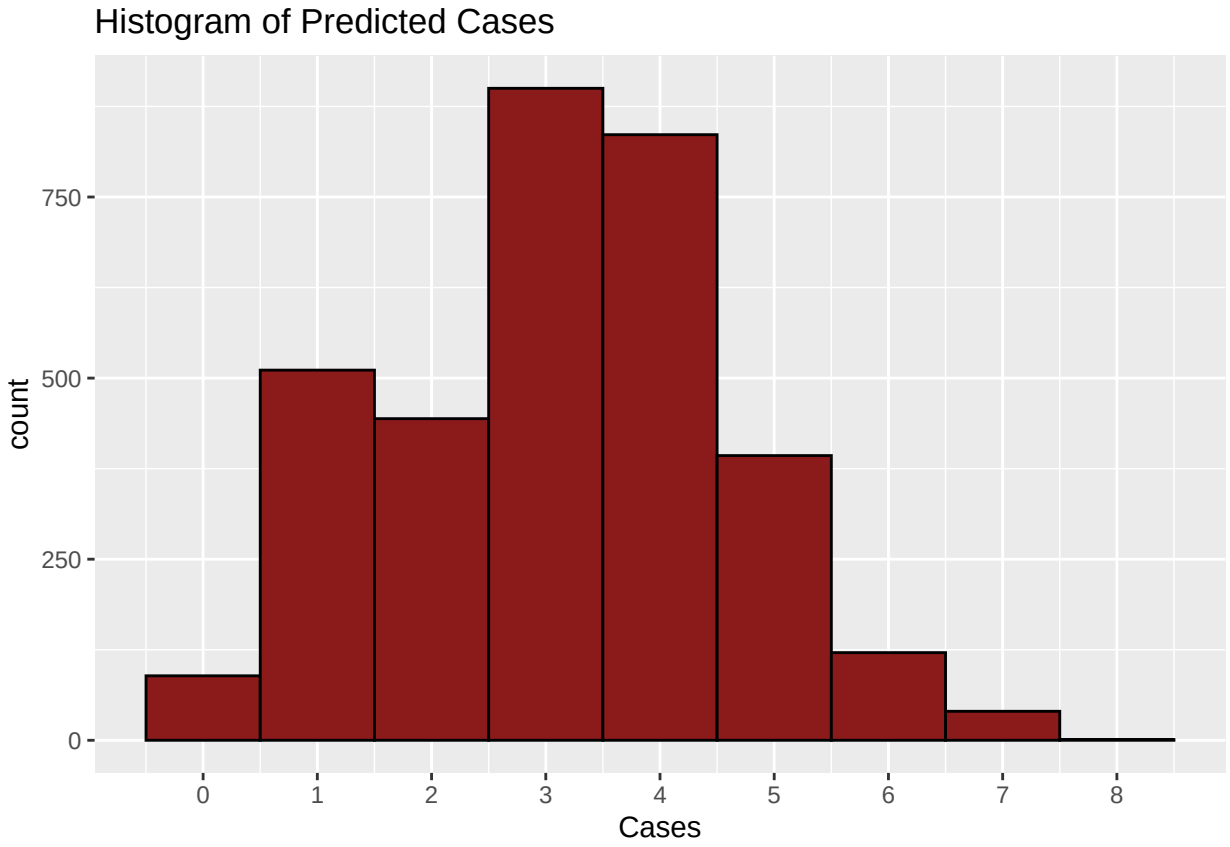
The zero-inflated **Poisson Model 2** performed best on all the metrics and will be used to create the prediction on the evaluation set.

PREDICTION

With 12795 observations in the evaluation set, the predicted cases for each row will not be printed, rather some summary statistics and a histogram of the evaluations will be shown. The actual predictions are available in the `m2evalP` data object, as seen in the CODE APPENDIX.

Table 17: Predicted Cases

m2evalP	Freq
0	89
1	511
2	444
3	900
4	836
5	393
6	121
7	40
8	1



REFERENCES

- Kuhn, M. (2019, March 27). *The* `caret` *Package*. https://topepo.github.io/caret/pre-processing.html#imputation
- Yeo, I., & Johnson, R. (2000). A New Family of Power Transformations to Improve Normality or Symmetry. **Biometrika**, 87(4), 954–959. Retrieved November 26, 2020, from http://www.jstor.org/stable/2673623

CODE APPENDIX

```
# Load necessary libraries
library(ggplot2)
library(knitr)
library(MASS)
library(caret)
library(corrplot)
library(mice)
library(pscl)
library(mpath)
library(data.table)

# Set master seed
set.seed(65408)
```

```
# Set filepaths for data ingestion
urlRemote  = "https://raw.githubusercontent.com/"
pathGithub = "aadler/DT621_Fall2020_Group2/master/HW5/data/"
fileTrain = "wine-training-data.csv"
fileEval = "wine-evaluation-data.csv"

# Read training file
DT <- fread(paste0(urlRemote, pathGithub, fileTrain))

# Number of training observations
ntrnobs <- dim(DT)[[1]]

# Get the names of the predictor variables
nmtrn <- names(DT)[-(1:2)]
nmtrnINT <- c('AcidIndex', 'LabelAppeal', 'STARS')
nmtrnDUB <- setdiff(nmtrn, nmtrnINT)
```

```
missingPreds <- transpose(DT[, lapply(.SD, function(x) {sum(is.na(x))}),
                             .SDcols = nmtrn],
                          keep.names = 'Predictors')
setnames(missingPreds, 'V1', 'Missing')
missingPreds[, Percentage := Missing / ntrnobs * 100]
setorder(missingPreds, -Missing)
kable(missingPreds, digits = 2L, caption = 'Predictors with Missing Observations')
```

```
# Isolate numeric only predictors
predictorDT <- DT[, .SD, .SDcols = nmtrn]

# Melt them from wide to long format
predictorDTM <- melt(predictorDT, variable.name = 'metric',
                     value.name = 'value',  variable.factor = FALSE,
                     measure.vars = nmtrn)

# Calculate summary statistics
statsN <- predictorDTM[, .(Mean = mean(value, na.rm = TRUE),
                           SD = sd(value, na.rm = TRUE),
                           Min = min(value, na.rm = TRUE),
                           Q1 = quantile(value, prob = 0.25, na.rm = TRUE),
                           Median = median(value, na.rm = TRUE),
                           Q3 = quantile(value, prob = 0.75, na.rm = TRUE),
                           Max = max(value, na.rm = TRUE),
                           IQR = IQR(value, na.rm = TRUE)), keyby = 'metric']

# Print the table
kable(statsN, digits = 3L, align = 'r',
      caption = "Summary Statistitics for Numeric Variables")
```

```
# Using Epanechnikov kernel to generate kernel-smoothed densities
ggplot(predictorDTM[!is.na(metric) & metric %chin% nmtrnDUB], aes(x = value)) +
  geom_density(kernel = 'epanechnikov') +
  facet_wrap( ~ metric, scales = 'free') +
  ggtitle('Kernel-smoothed Density of Numeric Predictors')
```

```
# Freedman-Diaconis rule for bin widths
FDbin <- function(x) {
```

```
  result <- 2 * IQR(x, na.rm = TRUE) / (length(x) ^ (1 / 3))
  return(ifelse(result == 0, 0.5, result))
}
# Generate histogram
ggplot(predictorDTM[!is.na(metric) & metric %chin% nmtrnINT], aes(x = value)) +
  geom_histogram(binwidth = FDbin, fill = 'indianred4') +
  facet_wrap( ~ metric, scales = 'free') +
  ggtitle('Histogram of Integral Predictors')
```

```
# Create corrgram
corrplot::corrplot(cor(DT[, ..nmtrn], use = 'complete.obs'),
         method = 'ellipse', type = 'lower', order = 'hclust',
         hclust.method = 'ward.D2')
```

```
# Plot distribution of TARGET
ggplot(DT[, .(TARGET)], aes(x = TARGET)) +
  geom_histogram(binwidth = FDbin, fill = 'darkolivegreen4')
```

```
# Create training and testing split
set.seed(1004)
trnIDX <- createDataPartition(DT$TARGET, p = 0.7)
trnSet <- DT[trnIDX$Resample1, ]
tstSet <- DT[!trnIDX$Resample1, ]
```

```
# P1 & NB1: Copy the training set to leave a pristine version for others
m1trn <- copy(trnSet)

# P1 & NB1: Add the "rated" factor variable
m1trn[, Rated := factor(ifelse(is.na(STARS), 'Unrated', 'Rated'),
                        levels = c('Unrated', 'Rated'),
                        labels = c('Unrated', 'Rated'))]
```

```
# P1 & NB1: Create bagged-tree imputation model
set.seed(89)
m1trnI <- preProcess(m1trn, method =c('nzv', 'bagImpute'))

# P1 & NB1: Replace missing STARS with 0s
m1trn[is.na(STARS), STARS := 0L]

# P1 & NB1: Impute the remaining NAs
m1trnImp <- predict(m1trnI, newdata = m1trn)
```

```
# P1 & NB1: Isolate numeric only predictors
m1predictorDT <- m1trnImp[, .SD, .SDcols = nmtrn]

# P1 & NB1: Melt them from wide to long format
m1predictorDTM <- melt(m1predictorDT, variable.name = 'metric',
                       value.name = 'value',  variable.factor = FALSE,
                       measure.vars = nmtrn)

# P1 & NB1: Using Epanechnikov kernel to generate kernel-smoothed densities
ggplot(predictorDTM[metric %chin% nmtrnDUB], aes(x = value)) +
  geom_density(kernel = 'epanechnikov') +
  facet_wrap( ~ metric, scales = 'free') +
  ggtitle('Kernel-smoothed Density of Numeric Predictors')
```

```r
# P1 & NB1: Generate histogram
ggplot(m1predictorDTM[metric %chin% nmtrnINT], aes(x = value)) +
  geom_histogram(binwidth = FDbin, fill = 'indianred4') +
  facet_wrap( ~ metric, scales = 'free') +
  ggtitle('Histogram of Integral Predictors')
```

```r
# P2 & ML 1: Clean and impute
set.seed(525)
m2train <- copy(trnSet[,-1])
m2train[,c(2:8,11:12)] <- abs(m2train[,c(2:8,11:12)])
m2train$STARS[is.na(m2train$STARS)] <- 0
m2train.impute <- mice(m2train, method = 'pmm', print = FALSE)
m2train <- complete(m2train.impute)
m2train.r <- m2train$TARGET
m2train.p <- m2train[,-1]
pre.process <- preProcess(m2train.p, method = c("YeoJohnson"))
m2train.p <- predict(pre.process, m2train.p)
m2train[,c(2:13,15)] <- m2train.p[,c(1:12,14)]
```

```r
# ML2 & NB2 data transformation
# MICE imputation on train and test set
set.seed(123)
m3train <- complete(mice(data = trnSet[, -c('INDEX')],
                         method = "pmm", print = FALSE), 3)
m3test <- complete(mice(data = tstSet[, -c('INDEX')],
                         method = "pmm", print = FALSE), 3)
```

```r
# P1: Using stepAIC so turn off cross-validation
trc <- trainControl(method = 'none')
set.seed(350047)
m1PFit <- train(TARGET ~ . + FreeSulfurDioxide:TotalSulfurDioxide +
                  AcidIndex * FixedAcidity * VolatileAcidity * pH +
                  Rated:STARS + LabelAppeal:Rated + LabelAppeal:Rated:STARS -
                  INDEX - STARS, data = m1trnImp,
                trControl = trc, method = 'glmStepAIC',
                family = poisson(link = 'log'), direction = 'both', trace = 0)
# P1: Save AIC
m1PFitAIC <- AIC(m1PFit$finalModel)
```

```r
# P1:Print results. This needed to be in its own section so that the LaTeX
# commands to change the text size can be wrapped around it.
kable(summary(m1PFit$finalModel)$coefficients,
      caption = "Model 1 Poisson Regression Output")
```

```r
# P1: Check the dispersion of the model
m1PFitDisp <- sum(residuals(m1PFit$finalModel, type = 'pearson') ^ 2) /
  m1PFit$finalModel$df.residual
```

```r
# P1: Extract the Poisson variable importance and order it for display
vIP <- varImp(m1PFit$finalModel)
vIPn <- row.names(vIP)
vIPDT <- data.table(Coefficients = vIPn, vIP)
setorder(vIPDT, -Overall)
kable(vIPDT, digits = 2L, caption = "Poisson Model Variable Importance")
```

```r
# NB1: There is no built-in AIC stepping in caret, so using basic glm.nb and
# then calling stepAIC on saturated model. Once done, investigated some one-off
# possible enhancements due to issues with glm.nb
# See https://stackoverflow.com/questions/11749977/why-does-glm-nb-throw-a-missing-value-error-only-on-
# For speed, only final called model is evaluated.
set.seed(872)
m1NBFitStart <- glm.nb(TARGET ~ . + FreeSulfurDioxide:TotalSulfurDioxide +
                         AcidIndex * FixedAcidity * VolatileAcidity * pH +
                         Rated:STARS + LabelAppeal:Rated +
                         LabelAppeal:Rated:STARS - INDEX - STARS,
                       data = m1trnImp, link = log)
m1NBFitStep <- stepAIC(m1NBFitStart, direction = 'both', trace = 0)

# Last call from automated procedure
# TARGET ~ FixedAcidity + VolatileAcidity + Chlorides + FreeSulfurDioxide +
#     TotalSulfurDioxide + pH + Sulphates + Alcohol + LabelAppeal +
#     AcidIndex + Rated + FreeSulfurDioxide:TotalSulfurDioxide +
#     FixedAcidity:AcidIndex + VolatileAcidity:AcidIndex + Rated:STARS +
#     LabelAppeal:Rated + LabelAppeal:Rated:STARS

# Removing VolatileAcidity improved the AIC

m1NBFit <- glm.nb(formula = TARGET ~ FixedAcidity + Chlorides +
                    FreeSulfurDioxide + TotalSulfurDioxide + pH + Sulphates +
                    Alcohol + LabelAppeal + AcidIndex + Rated +
                    FreeSulfurDioxide:TotalSulfurDioxide +
                    FixedAcidity:AcidIndex + VolatileAcidity:AcidIndex +
                    Rated:STARS + LabelAppeal:Rated + LabelAppeal:Rated:STARS,
                  data = m1trnImp, link = log)

# NB1: Save AIC
m1NBFitAIC <- AIC(m1NBFit)
```

```r
# NB1: Print results. This needed to be in its own section so that the LaTeX
# commands to change the text size can be wrapped around it.
kable(summary(m1NBFit)$coefficients,
      caption = "Model 1 Negative Binomial Regression Output")
```

```r
# NB1: Extract the NegBinom variable importance and order it for display
vINB <- varImp(m1NBFit)
vINBn <- row.names(vINB)
vINBDT <- data.table(Coefficients = vINBn, vINB)
setorder(vINBDT, -Overall)
kable(vINBDT, digits = 2L,
      caption = "Negative Binomial Model Variable Importance")
```

```r
# P1 & NB1: GoF Comparison
m1PNBC <- data.table(Model = c("Poisson 1", "Negative Binomial 1"),
                     Deviance = c(m1PFit$finalModel$deviance,
                                  m1NBFit$deviance),
                     AIC = c(m1PFit$finalModel$aic,
                             m1NBFit$aic))
kable(m1PNBC, digits = 3L,
      caption = "Poisson 1 and NegBinom 1 Goodness-of-Fit Comparison")
```

```r
set.seed(525)
# Zero Inflated Poisson Model
m2.pmodel <- zeroinfl(TARGET~.,
                      dist = 'poisson',
                      model = TRUE,
                      data = m2train,
                      link = "logit")
# Backward Elimination
m2.fpmodel <- be.zeroinfl(m2.pmodel,
                          data = m2train,
                          dist = 'poisson',
                          alpha = 0.05,
                          trace = FALSE)
m2.poisson.aic <- AIC(m2.fpmodel)
```

```r
# P2: Outputs
kable(summary(m2.fpmodel)$coefficients$count,
      caption = "Zero Inflated Poisson Model #2 Count Output")
kable(summary(m2.fpmodel)$coefficients$zero,
      caption = "Zero Inflated Poisson Model #2 Zero Output")
```

```r
# Dispersion statistic
E2 <- resid(m2.fpmodel, type = "pearson")
N  <- nrow(m2train)
p  <- length(coef(m2.fpmodel))
dp <- sum(E2^2) / (N - p)

# Poisson Model #2, same formula from ZIP model
m2.poisson <- glm(TARGET ~ Alcohol + LabelAppeal + AcidIndex + STARS | VolatileAcidity +
                    CitricAcid + FreeSulfurDioxide + TotalSulfurDioxide + pH +
                    Sulphates + Alcohol + LabelAppeal + AcidIndex + STARS,
                  family = poisson(link = 'log'), data = m2train)
vuong(m2.poisson, m2.fpmodel)
```

```r
# Multiple Linear Regression model
set.seed(525)
model.null = lm(TARGET ~ 1, data = m2train)
model.full = lm(TARGET ~ ., data = m2train)

# Step model used to derive final model, m1MLRFit
# step(model.null,
#      scope = list(upper = model.full),
#      direction = "both",
#      data = m2train,
#      trace = FALSE)

m1MLRFit = lm(TARGET ~ STARS + LabelAppeal + AcidIndex + TotalSulfurDioxide +
                VolatileAcidity + FreeSulfurDioxide + Alcohol + CitricAcid +
                Chlorides + pH + Sulphates,
              data = m2train)
```

```r
# ML1: Outputs
kable(summary(m1MLRFit)$coefficient,
      caption = "Multiple Linear Regression Model #1 Output")
```

```
# NB2: Baseline Model
m2NBBase <- glm.nb(TARGET ~ ., data = m3train)
knitr::kable(summary(m2NBBase)$coefficients, digits = 3L,
             caption = 'Negative Binomial Regression Model #2 - Base Output')
```

```
# NB2: Revised Model
m2NBFit <- glm.nb(TARGET ~ VolatileAcidity + Chlorides +
                FreeSulfurDioxide + TotalSulfurDioxide + pH + Alcohol +
                LabelAppeal + AcidIndex + STARS, data = m3train)
knitr::kable(summary(m2NBFit)$coefficients, digits = 3L,
             caption = 'Negative Binomial Regression Model #2 - Final Output')
```

```
# ML2: Baseline model
m2MLBase <- lm(TARGET ~ ., data = m3train)
knitr::kable(summary(m2MLBase)$coefficients, digits = 3L,
             caption = 'Multiple Regression Model #2 - Base Output')
```

```
# ML2: Enhanced model
m2MLFit <- lm(TARGET ~ VolatileAcidity + Chlorides + FreeSulfurDioxide
              + TotalSulfurDioxide + pH + Sulphates + Alcohol + LabelAppeal
              + AcidIndex + STARS, data = m3train)
knitr::kable(summary(m2MLFit)$coefficients, digits = 3L,
             caption = 'Multiple Regression Model #2 - Final Output')
```

```
compTable <- data.table(Models = c("Poisson 1", "Poisson 2",
                                   "Neg. Binom. 1", "Neg. Binom. 2",
                                   "MultiLinear 1", "MultiLinear 2"),
                        RMSE = double(6), MAE = double(6),
                        TrainAIC = double(6))
```

```
# P1 & NB1: process test data
m1tst <- copy(tstSet)
m1tst[, Rated := factor(ifelse(is.na(STARS), 'Unrated', 'Rated'),
                        levels = c('Unrated', 'Rated'),
                        labels = c('Unrated', 'Rated'))]
set.seed(89)
m1tstI <- preProcess(m1tst, method =c('nzv', 'bagImpute'))
m1tst[is.na(STARS), STARS := 0L]
m1tstImp <- predict(m1tstI, newdata = m1tst)
```

```
# P1: Test Model
m1PtstPred <- round(predict(m1PFit, newdata = m1tstImp))

compTable[1, 2] <- m1PRMSE <- RMSE(m1PtstPred, m1tstImp$TARGET)
compTable[1, 3] <- m1PMAE <- MAE(m1PtstPred, m1tstImp$TARGET)
compTable[1, 4] <- m1PFitAIC
```

```
set.seed(525)
m2test <- copy(tstSet[,-1])
m2test[,c(2:8,11:12)] <- abs(m2test[,c(2:8,11:12)])
m2test$STARS[is.na(m2test$STARS)] <- 0
m2test.impute <- mice(m2test, method = 'pmm', print = FALSE)
m2test <- complete(m2test.impute)
m2test.r <- m2test$TARGET
m2test.p <- m2test[,-1]
pre.process <- preProcess(m2test.p, method = c("YeoJohnson"))
```

```
m2test.p <- predict(pre.process, m2test.p)
m2test[,c(2:13,15)] <- m2test.p[,c(1:12,14)]
```

```
# ZIP2: Test Model
m2PtstPred = round(predict(m2.fpmodel, newdata = m2test))
compTable[2, 2] <- m2PRMSE <- RMSE(m2PtstPred, m2test.r)
compTable[2, 3] <- m2PMAE <- MAE(m2PtstPred, m2test.r)
compTable[2, 4] <- m2.poisson.aic
```

```
# NB1: Test Model
m1NBtstPred <- round(predict(m1NBFit, newdata = m1tstImp))
compTable[3, 2] <- m1NBRMSE <- RMSE(m1NBtstPred, m1tstImp$TARGET)
compTable[3, 3] <- m1NBMAE <- MAE(m1NBtstPred, m1tstImp$TARGET)
compTable[3, 4] <- m1NBFitAIC
```

```
# NB2: Test Model
m2NBtstPred <- round(predict(m2NBFit, newdata = m3test))
compTable[4, 2] <- m2NBRMSE <- RMSE(m2NBtstPred, m3test$TARGET)
compTable[4, 3] <- m2NBMAE <- MAE(m2NBtstPred, m3test$TARGET)
compTable[4, 4] <- m2NBFitAIC <- AIC(m2NBFit)
```

```
# MLR1: Test Model
m1MLRtstPred <- round(predict(m1MLRFit, newdata = m2test))
compTable[5, 2] <- m1MLRRMSE <- RMSE(m1MLRtstPred, m2test.r)
compTable[5, 3] <- m1MLRMAE <- MAE(m1MLRtstPred, m2test.r)
compTable[5, 4] <- m1MLRFitAIC <- AIC(m1MLRFit)
```

```
# MLR2: Test Model
m2MLtstPred <- round(predict(m2MLFit, newdata = m3test))
compTable[6, 2] <- m2MLMSE <- RMSE(m2MLtstPred, m3test$TARGET)
compTable[6, 3] <- m2MLMAE <- MAE(m2MLtstPred, m3test$TARGET)
compTable[6, 4] <- m2MLFitAIC <- AIC(m2MLFit)
```

```
# Compare results on test set
kable(compTable, digits = 3L,
      caption = "Model Results on Test Set")
```

```
# Read evaluation but not "IN" since that's dropped right away
m2eval <- fread(paste0(urlRemote, pathGithub, fileEval), drop = 'IN')
nobsE <- dim(DT)[[1]]
```

```
# Process evaluation file as per zero-inflated Poisson model 2
m2eval[,c(2:8,11:12)] <- abs(m2eval[,c(2:8,11:12)])
m2eval$STARS[is.na(m2eval$STARS)] <- 0
set.seed(525)
m2eval.impute <- mice(m2eval, method = 'pmm', print = FALSE)
m2eval <- complete(m2eval.impute)
m2eval.r <- m2eval$TARGET
m2eval.p <- m2eval[, -1]
pre.process <- preProcess(m2eval.p, method = c("YeoJohnson"))
m2eval.p <- predict(pre.process, m2eval.p)
m2eval[, c(2:13, 15)] <- m2eval.p[, c(1:12, 14)]

# Predict Cases
m2evalP = as.data.frame(round(predict(m2.fpmodel, newdata = m2eval)))
colnames(m2evalP) <- 'Cases'
```

```r
kable(table(m2evalP), caption = "Predicted Cases")
ggplot(m2evalP, aes(x = Cases)) +
  geom_histogram(binwidth = 1, center = 0,
                 fill = 'firebrick4', color = 'black') +
  scale_x_continuous(breaks = 0:8) +
  ggtitle("Histogram of Predicted Cases")
```