Poisson Regression. Negative Binomial Regression. Multinomial Logistic Regression. Zero Inflated Poisson and Negative Binomial Regression

Data 621: Homework 5 - Group 4

Sachid Deshmukh

Michael Yampol

Vishal Arora

Ann Liu-Ferrara

Dec. 05, 2019

1. Data Exploration

Let’s load training dataset and preview.

	ï..INDEX <int>	TAR… <int>	FixedAcidity <dbl>	VolatileAcidity <dbl>	CitricAcid <dbl>	ResidualSugar <dbl>	Chlorides <dbl>
1	1	3	3.2	1.160	-0.98	54.2	-0.567
2	2	3	4.5	0.160	-0.81	26.1	-0.425
3	4	5	7.1	2.640	-0.88	14.8	0.037
4	5	3	5.7	0.385	0.04	18.8	-0.425
5	6	4	8.0	0.330	-1.26	9.4	NA
6	7	0	11.3	0.320	0.59	2.2	0.556

6 rows | 1-8 of 17 columns

```
## [1] "Number of columns =  16"
```

```
## [1] "Number of rows =  12795"
```

As we can see in the preview, training data has 16 columns and 12795 rows

Let’s analyze datatypes of each column.

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

All the columns are either Integer or numeric types. Please note that Integer columns in the above dataset are excellent candidate for creating categorical variables which can further enhance quality of our models

Let’s check if any variables have missing values. Values which are NULL or NA.

Check missing values

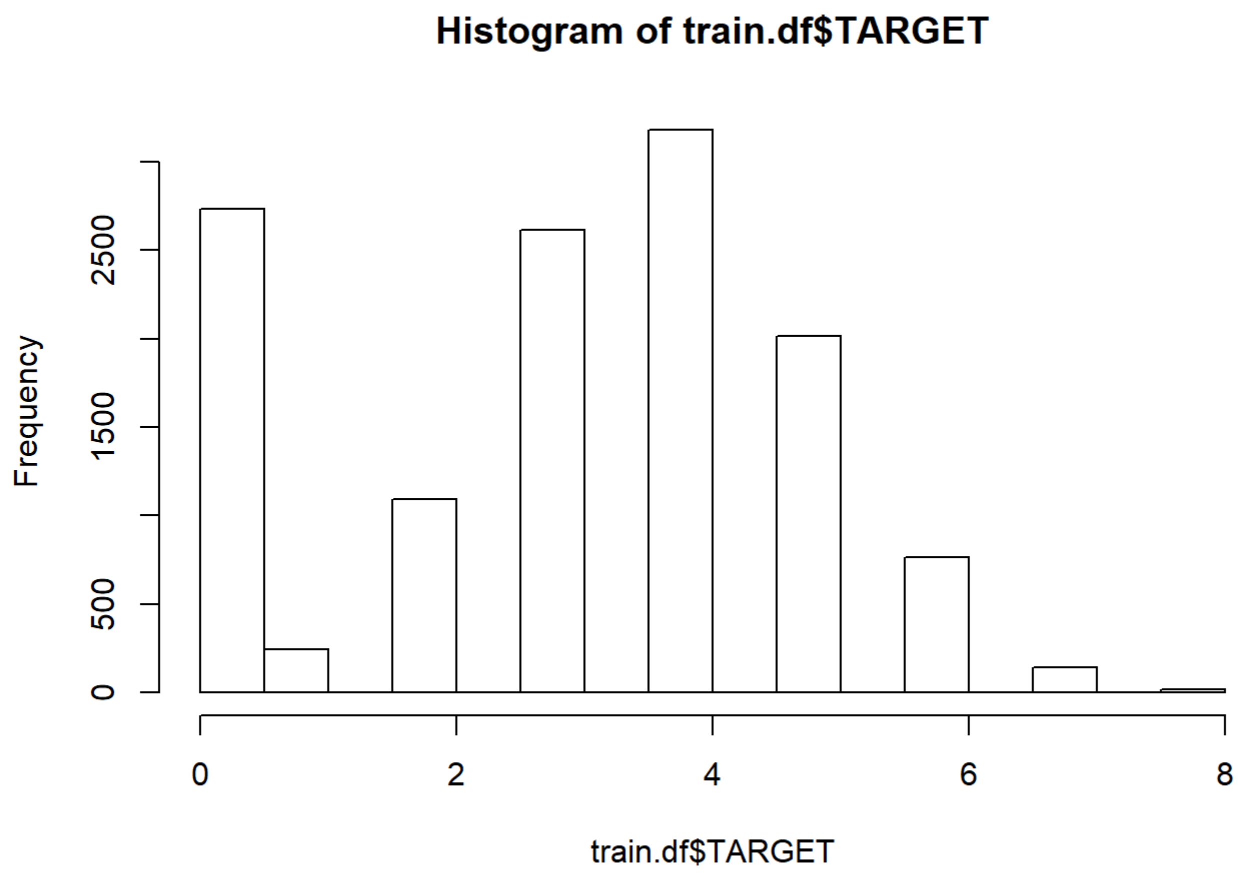
```
## [1] "Number of columns with missing values =  8"
```

```
## [1] "Names of columns with missing values =  ResidualSugar, Chlorides, FreeSulfurDioxide, Tot
alSulfurDioxide, pH, Sulphates, Alcohol, STARS"
```

We can see 8 variables namely ResidualSugar, Chlorides, FreeSulfurDioxide, TotalSulfurDioxide, pH, Sulphates, Alcohol and STARS have missing values. We will impute values for these variables for better model accuracy.

Let’s see distribution of Target variable

```
hist(train.df$TARGET)
```



From the above histogram we can see that distribution of target variable is not exactly Poisson distribution. We can see high number of observations with zeros. Then there is another peak around 4 and then it decreases towards right. Ditribution also shows skew in the right hand side where there are very few observations as sample count increases. This dataset is a very good candidate for zero inflated Poisson/Negative bionomial regression since we can see the number of observations with zero count is more.

2. Data Preparation

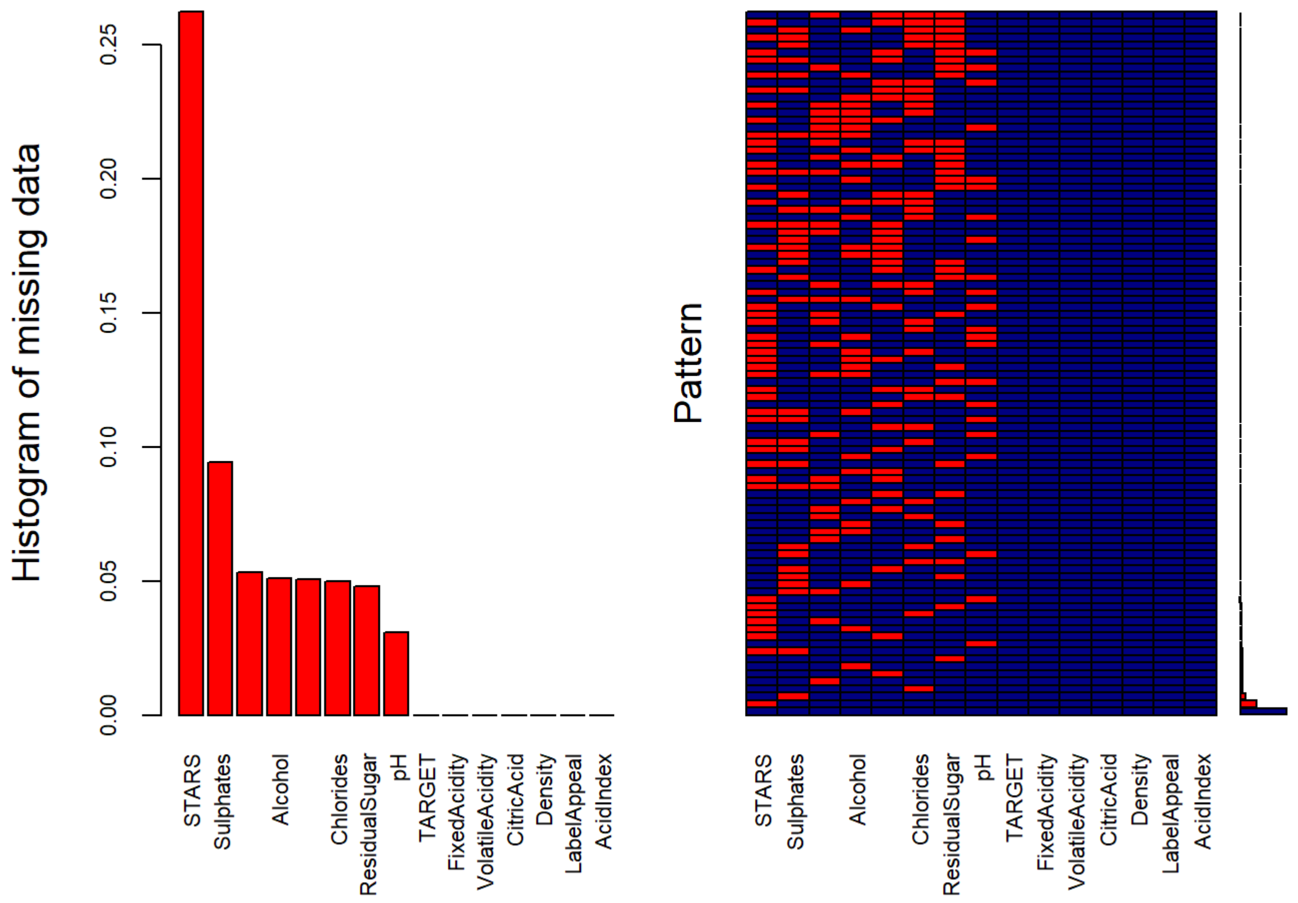
Remove Index column since it is just and indentifier and won't contribute towards predictive capability of the mdel

```
train.df = train.df[1:nrow(train.df), 2:ncol(train.df)]
```

Let's do data imputation for missing columns

Which columns are mssing and what is a missing pattern. Let's leverage VIM package to get this information

```
## Warning in plot.aggr(res, ...): not enough vertical space to display
## frequencies (too many combinations)
```



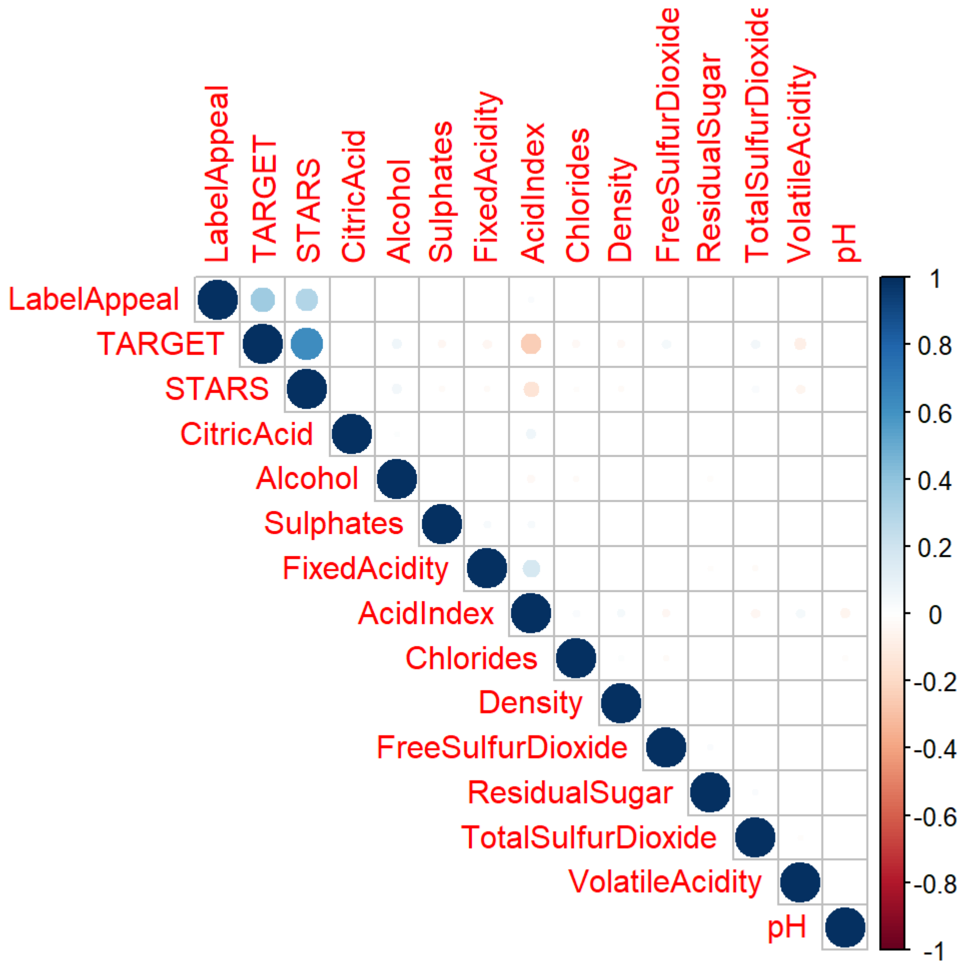
```
##
##  Variables sorted by number of missings:
##            Variable      Count
##               STARS 0.26252442
##           Sulphates 0.09456819
##  TotalSulfurDioxide 0.05330207
##             Alcohol 0.05103556
##   FreeSulfurDioxide 0.05056663
##           Chlorides 0.04986323
##       ResidualSugar 0.04814381
##                  pH 0.03087143
##              TARGET 0.00000000
##        FixedAcidity 0.00000000
##     VolatileAcidity 0.00000000
##          CitricAcid 0.00000000
##             Density 0.00000000
##          LabelAppeal 0.00000000
##           AcidIndex 0.00000000
```

From the above missing values pattern we can see that variable Stars have the highest proportion of missing values. Around 25% of the obsertvations are missing value Stars. This might also indicate wine with 0 Star ratings. All other variables have very low proportion of missing observation. This is a good news since this asserts good quality of input data.

Let’s use MICE package to imput missing values

```
##
##  iter imp variable
##   1   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   1   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   2   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   2   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   3   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   3   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   4   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   4   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   5   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   5   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   6   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   6   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   7   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   7   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   8   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   8   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   9   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   9   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   10   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
ohol  STARS
##   10   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
ohol  STARS
```

The variables which are highly correlated carry similar information and can affect model accuracy. Highly correlated variables also impacts estimation of model coefficients. Let’s figure out which variables in the training datasets are highly correlated to each other



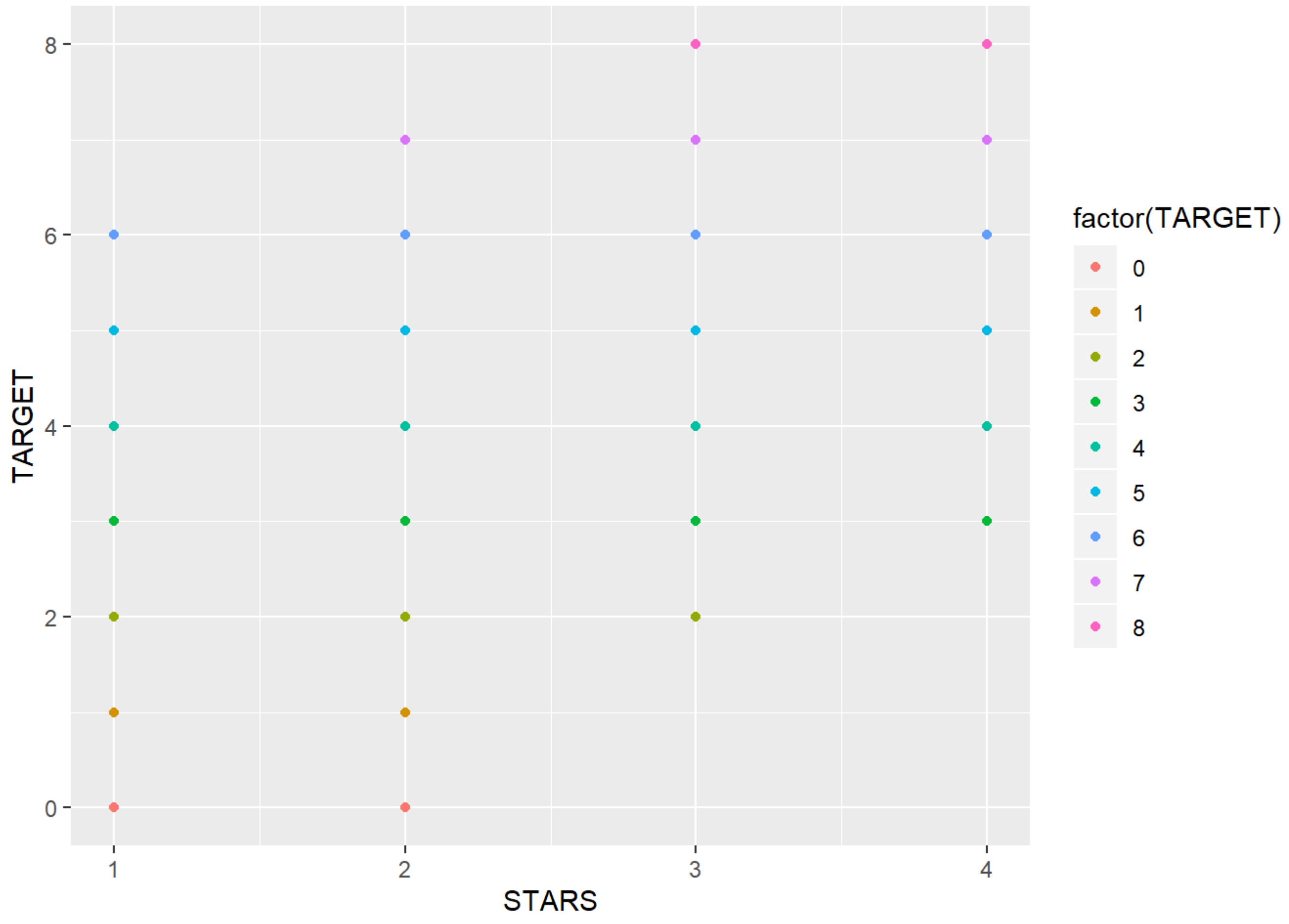
From the above correlation graph we can see that target variable TARGET is highly correlated with following variables

- **Label Appeal**
- **Stars**
- **Acid Index**
- **Fixed Acidity**

Feature Transformations

See relationshipt between STARS and Target variable. Transform STARS Variable

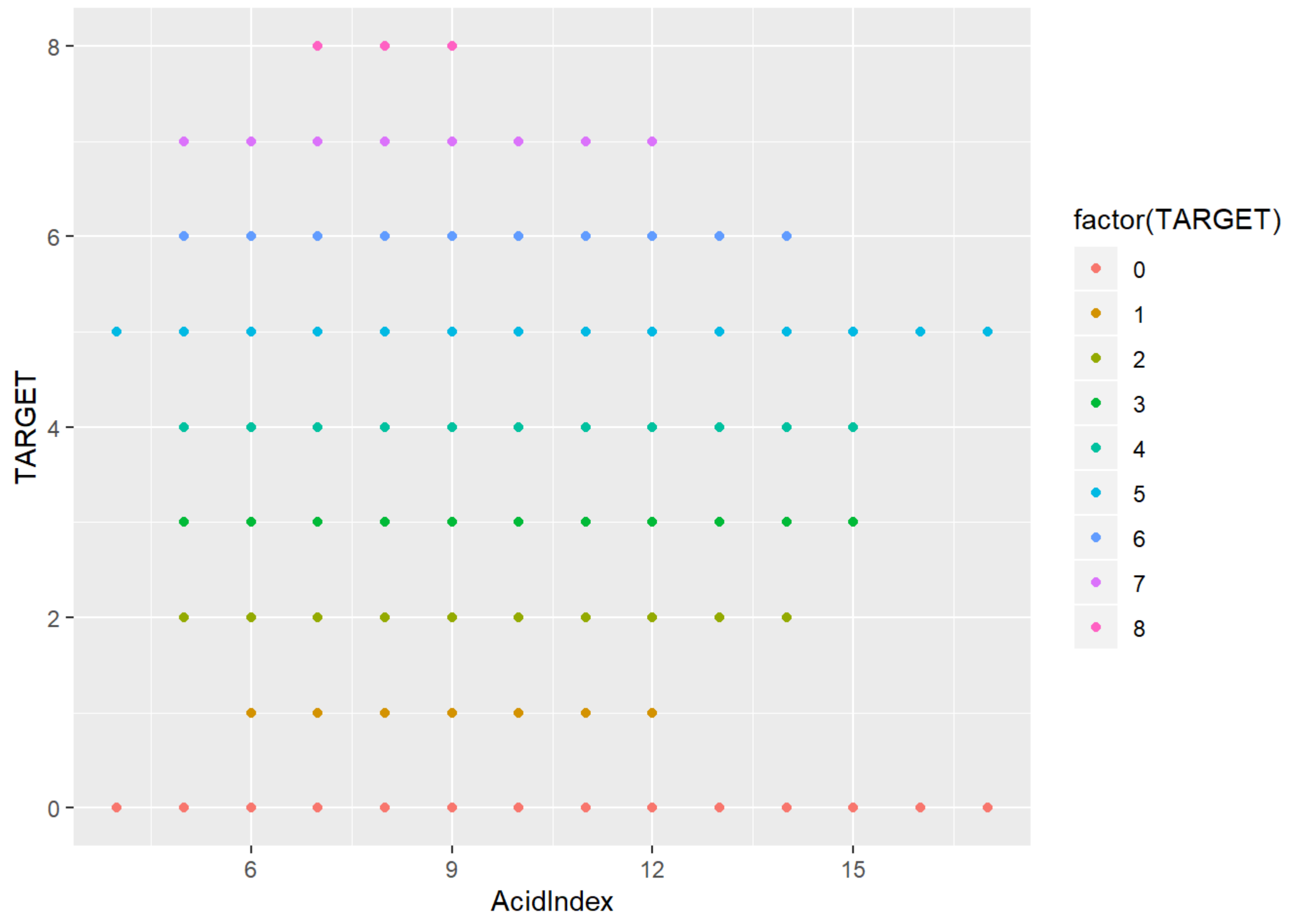
```
ggplot(train.df, aes(x= STARS, y = TARGET)) +
  geom_point(aes(colour = factor(TARGET)))
```



```
train.df$StarTran = ifelse(train.df$STARS == 1, 2, train.df$STARS)
```

See relationshipt between AcidIndex and Target variable. Transform AcidIndex variable

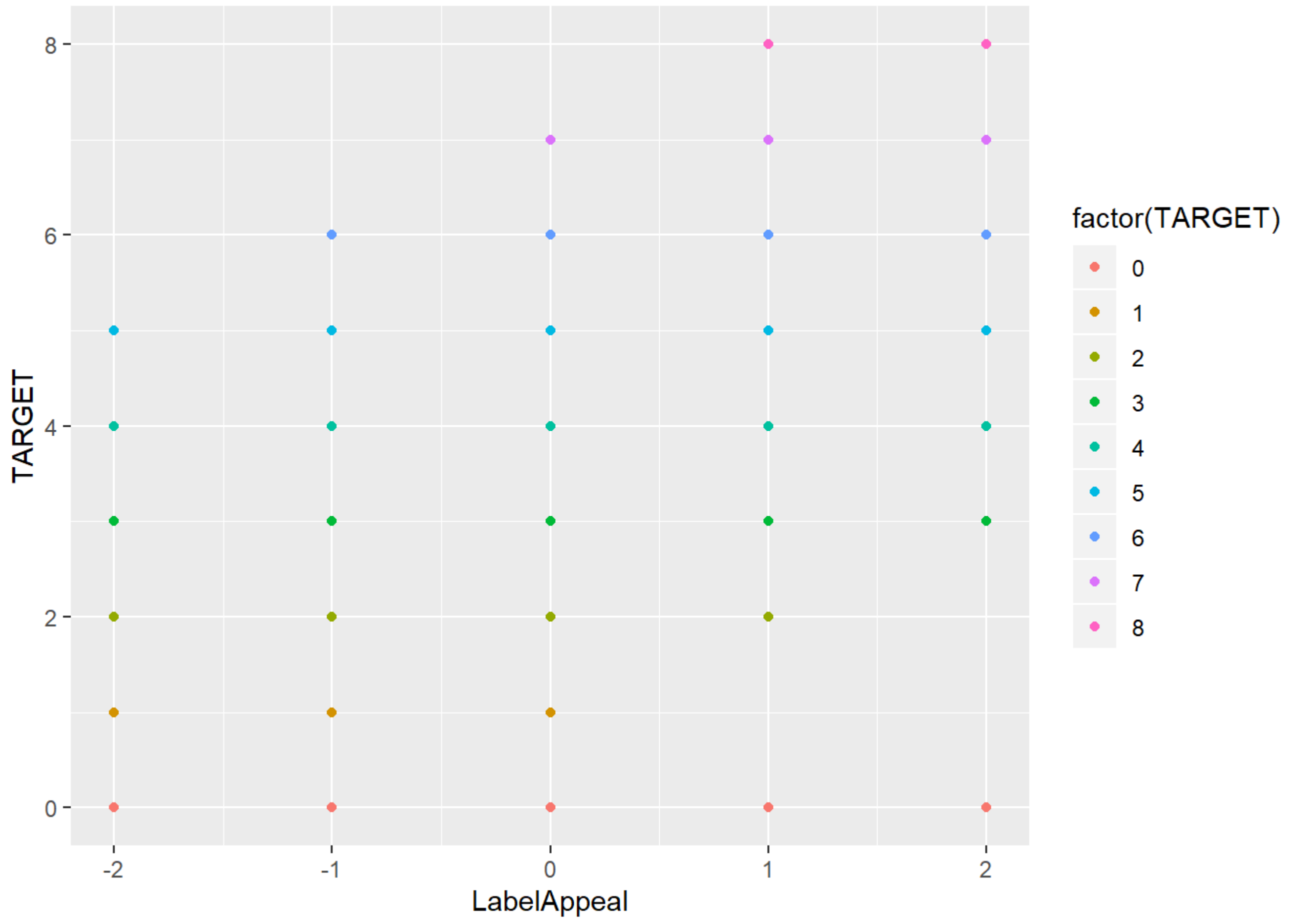
```
ggplot(train.df, aes(x= AcidIndex, y = TARGET)) +
  geom_point(aes(colour = factor(TARGET)))
```



```
train.df$AcidIndexTran = ifelse(train.df$AcidIndex <=6,  0, train.df$AcidIndex)
train.df$AcidIndexTran = ifelse((train.df$AcidIndexTran > 6 & train.df$AcidIndexTran < 12),  1,
 train.df$AcidIndexTran)
train.df$AcidIndexTran = ifelse(train.df$AcidIndexTran >= 12 ,  2, train.df$AcidIndexTran)
```

See relationship between LabelAppeal and target variable. Transform LabelAppeal Variable

```
ggplot(train.df, aes(x= LabelAppeal, y = TARGET)) +
  geom_point(aes(colour = factor(TARGET)))
```



```
train.df$LabelAppealTran = ifelse(train.df$LabelAppeal <0,  0, train.df$LabelAppeal)
```

3. Build Models

Model fitting and evaluation

For model evaluation we will use train test split technique. We will use 70% data to train the model and leverage remaining 30% to test the model

Model Evaluation Metrics

We will use following metrics for model evaluation and comparison

- **RMSE** : We wll use Root Mean Square error to evaluate Poisson and Negative binomial model. Root mean square indicates the quality of model fit. Lesser RMSE indicates better model fit and higher RMSE indicates poor model fit. RMSE can be calcualted as (sum(y-y’)$^{2)}$0.5

Where

- **y** : Actual value
- **y’** : Predicted value

- **Model Accuracy** : For multinomial model we will use accuracy as model evaluation metric. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition: Accuracy = (TP + TN)/(TP + FP + TN + FN)

Where

- **TP** : Stands for True Positives
- **TN** : Stands for True Negatives
- **FP** : Stands for False Positives
- **FN** : Stands for False Negatives

Now we are clear on our model fitting and evaluation method (Train test split) and also have model evaluation metrics (RMSE and Accuracy) which we will use to compare the model effectiveness we are all set to build different models and access it’s performance

1. Split training data into train and test

```
train.df.class = dplyr::rename(train.df, target = TARGET)
dt = sort(sample(nrow(train.df.class), nrow(train.df.class)*.7))
train.data = train.df.class[dt,]
test.data = train.df.class[-dt,]
test.values = test.data$target
test.data = test.data[1:nrow(test.data), names(test.data)[names(test.data)!= 'target']]
```

2. Poisson regression model with all variables

```
model.poss.all = glm(target~  ., family=poisson, data=train.data)
summary(model.poss.all)
```

```
##
## Call:
## glm(formula = target ~ ., family = poisson, data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2267  -0.6251   0.0430   0.5538   2.9064
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         2.161e+00  2.345e-01   9.213  < 2e-16 ***
## FixedAcidity       -1.319e-03  9.843e-04  -1.340 0.180293
## VolatileAcidity    -3.735e-02  7.776e-03  -4.803 1.56e-06 ***
## CitricAcid          3.474e-03  7.027e-03   0.494 0.621092
## ResidualSugar       1.214e-04  1.807e-04   0.672 0.501759
## Chlorides          -5.214e-02  1.918e-02  -2.719 0.006547 **
## FreeSulfurDioxide   1.119e-04  4.097e-05   2.732 0.006298 **
## TotalSulfurDioxide  7.891e-05  2.653e-05   2.974 0.002937 **
## Density            -4.094e-01  2.280e-01  -1.796 0.072572 .
## pH                 -1.771e-02  9.048e-03  -1.957 0.050299 .
## Sulphates          -1.679e-02  6.506e-03  -2.581 0.009849 **
## Alcohol             3.662e-03  1.627e-03   2.250 0.024430 *
## LabelAppeal         1.866e-01  1.407e-02  13.265  < 2e-16 ***
## AcidIndex          -1.109e-01  7.002e-03 -15.842  < 2e-16 ***
## STARS               6.594e-01  1.569e-02  42.028  < 2e-16 ***
## StarTran           -5.011e-01  2.166e-02 -23.136  < 2e-16 ***
## AcidIndexTran       9.482e-02  2.340e-02   4.053 5.06e-05 ***
## LabelAppealTran    -6.712e-02  2.029e-02  -3.308 0.000941 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 15939  on 8955  degrees of freedom
## Residual deviance: 10603  on 8938  degrees of freedom
## AIC: 33003
##
## Number of Fisher Scoring iterations: 5
```

```
rmse.poss.all = model.fit.evaluate.rmse(model.poss.all, test.data, test.values)
print(rmse.poss.all)
```

```
## [1] 163.9909
```

2. Poisson regression model with selected variables

```
model.poss.sel = glm(target~  LabelAppeal + STARS + AcidIndex + FixedAcidity, family=poisson, da
ta=train.data)
summary(model.poss.sel)
```

```
##
## Call:
## glm(formula = target ~ LabelAppeal + STARS + AcidIndex + FixedAcidity,
##     family = poisson, data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8899  -0.6853   0.1378   0.6380   2.8229
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.1815724  0.0453155  26.074   <2e-16 ***
## LabelAppeal   0.1384440  0.0072780  19.022   <2e-16 ***
## STARS         0.3438128  0.0067139  51.209   <2e-16 ***
## AcidIndex    -0.1010355  0.0054112 -18.672   <2e-16 ***
## FixedAcidity -0.0014938  0.0009843  -1.518    0.129
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 15939  on 8955  degrees of freedom
## Residual deviance: 11256  on 8951  degrees of freedom
## AIC: 33630
##
## Number of Fisher Scoring iterations: 5
```

```
rmse.poss.sel = model.fit.evaluate.rmse(model.poss.sel, test.data, test.values)
print(rmse.poss.sel)
```

```
## [1] 162.8619
```

From the RMSE analysis of Poisson Regression model with all variables and Poisson regression model with selected variables, we can see that even though the RMSE of Poisson regression model with all variable is lesser, there is not much difference. Going forward we will stick to model with selected variables. This is done to maintain model simplicity and to make model more interpretable by removing many correlated variables.

3. Model Dispersion test

```
dispersiontest(model.poss.sel)
```

```
##
##  Overdispersion test
##
## data:  model.poss.sel
## z = -7.5819, p-value = 1
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##  0.8980863
```

The model dispersion test is 0.89. This indicates under dispersion. For Poisson assuption to hold true we need mean and variance of the target variable same. If mean is greater than variance it results into over dispersion. if mean is lesser than variance it results in under dispersion.

For Poisson model violation of above assumption (Mean = Var) will result into inaccurate calculation of standard errors for model coefficients. The magnitude of model coefficients will be same however inaccurate standard errors will result in to inaccurate calucation of confidence interval of the model coefficients and will in turn result into inaccurate inference.

Negative binomial regression is well suited for overdispersion. For underdispersion, Negative binomial regression is not reccomend. However let’s try negative Binomial Regression on the above dataset and compare the model paramters

4. Negative Binomial Regression

```
negbio = glm.nb(target~  LabelAppeal + STARS + AcidIndex + FixedAcidity, data=train.data)
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached

## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
summary(negbio)
```

```
##
## Call:
## glm.nb(formula = target ~ LabelAppeal + STARS + AcidIndex + FixedAcidity,
##     data = train.data, init.theta = 48373.9056, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8898  -0.6853   0.1378   0.6380   2.8229
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.1815770  0.0453170  26.074   <2e-16 ***
## LabelAppeal   0.1384433  0.0072783  19.021   <2e-16 ***
## STARS         0.3438171  0.0067142  51.208   <2e-16 ***
## AcidIndex    -0.1010372  0.0054114 -18.671   <2e-16 ***
## FixedAcidity -0.0014938  0.0009843  -1.518    0.129
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(48373.91) family taken to be 1)
##
##     Null deviance: 15939  on 8955  degrees of freedom
## Residual deviance: 11256  on 8951  degrees of freedom
## AIC: 33632
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  48374
##           Std. Err.:  66925
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -33620.27
```

```
rmse.nb = model.fit.evaluate.rmse(negbio, test.data, test.values)
print(rmse.nb)
```

```
## [1] 162.8619
```

5. Robust Poisson regression model

In general it is a good idea to fit robust Poisson regression model to get accurate estimation for Std Errors. Since dataset indicates under dispersion it is a good idea to fit robust Poisson regression model and check if we see any difference in the std error estimation for model regression coefficients

```
model.poss.sel.robust = glm(target~  LabelAppeal + STARS + AcidIndex + FixedAcidity, family=quas
ipoisson, data=train.data)
summary(model.poss.sel.robust)
```

```
##
## Call:
## glm(formula = target ~ LabelAppeal + STARS + AcidIndex + FixedAcidity,
##     family = quasipoisson, data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8899  -0.6853   0.1378   0.6380   2.8229
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.181572   0.042681  27.684   <2e-16 ***
## LabelAppeal   0.138444   0.006855  20.196   <2e-16 ***
## STARS         0.343813   0.006324  54.370   <2e-16 ***
## AcidIndex    -0.101036   0.005097 -19.824   <2e-16 ***
## FixedAcidity -0.001494   0.000927  -1.611    0.107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 0.8871091)
##
##     Null deviance: 15939  on 8955  degrees of freedom
## Residual deviance: 11256  on 8951  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
rmse.poss.sel.robust = model.fit.evaluate.rmse(model.poss.sel.robust, test.data, test.values)
print(rmse.poss.sel.robust)
```

```
## [1] 162.8619
```

5. Compare Negative Binomial, Poisson Regression and Robust Poisson regression models Coefficients and Std Errors

```
pos.coef = coef(model.poss.sel)
negbinom.coef = coef(negbio)
pos.stderr = se.coef(model.poss.sel)
negbinom.stderr = summary(negbio)$coefficients[, 2]
pos.robust.coef = coef(model.poss.sel.robust)
pos.robust.stderr = se.coef(model.poss.sel.robust)
df.analysis = cbind(pos.coef, negbinom.coef, pos.stderr, negbinom.stderr,pos.robust.coef, pos.ro
bust.stderr)
head(df.analysis,10)
```

```
##                  pos.coef negbinom.coef   pos.stderr negbinom.stderr
## (Intercept)   1.181572373   1.181576993 0.0453155026    0.0453170403
## LabelAppeal   0.138443992   0.138443327 0.0072780254    0.0072782946
## STARS         0.343812766   0.343817102 0.0067138721    0.0067141589
## AcidIndex    -0.101035473  -0.101037181 0.0054112043    0.0054113857
## FixedAcidity -0.001493798  -0.001493843 0.0009842588    0.0009842964
##              pos.robust.coef pos.robust.stderr
## (Intercept)      1.181572373      0.0426810726
## LabelAppeal      0.138443992      0.0068549152
## STARS            0.343812766      0.0063235592
## AcidIndex       -0.101035473      0.0050966224
## FixedAcidity    -0.001493798      0.0009270386
```

From the above table we can see that both model coefficient and std errors for Poisson and Negative Binomial regression model are same. This can be due to the fact that under-dispersion in the dataset is not that severe to impact the accuracy of Poisson regression model

From the above table we can see that model coefficients for Poisson Regression and Robust Poisson Regression models are same, however the estimates for Std Errors are different for them. This is expected since dataset has under dispersion. Std Errors estimations for regression coefficient of Poisson regression model will not be accurate. We need to rely on Std Error estimation for Robust Poisson regression model which is more suited for datasets exhibiting under dispersion or over dispersion. If we need to use these coefficients for inference it is better to rely on Std Error estimated of Robust Poisson regression model to calcualte confidence interval rather than normal Poisson regression modle for increasing accuracy of inference

6. Fit zero inflated Poisson Regression model

```
model.poss.zip= zeroinfl(target~  LabelAppeal + STARS + AcidIndex + FixedAcidity|STARS, dist='po
isson', data=train.data)
summary(model.poss.zip)
```

```
##
## Call:
## zeroinfl(formula = target ~ LabelAppeal + STARS + AcidIndex + FixedAcidity |
##     STARS, data = train.data, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -2.1289 -0.5022  0.0764  0.4987  2.1263
##
## Count model coefficients (poisson with log link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.3432168  0.0492539  27.271  < 2e-16 ***
## LabelAppeal   0.2272545  0.0076291  29.788  < 2e-16 ***
## STARS         0.1189978  0.0075242  15.815  < 2e-16 ***
## AcidIndex    -0.0391114  0.0061213  -6.389 1.67e-10 ***
## FixedAcidity  0.0001247  0.0010072   0.124    0.901
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.4956     0.1278   19.53   <2e-16 ***
## STARS        -2.8342     0.1141  -24.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 13
## Log-likelihood: -1.506e+04 on 7 Df
```

```
rmse.poss.zip = model.fit.evaluate.rmse(model.poss.zip, test.data, test.values)
print(rmse.poss.zip)
```

```
## [1] 89.21323
```

We can see that Zero inflated Poisson regression model has greately improved the RMSE and indicates better fit for the model. This is due to the fact that dataset has unusually higher number of samples with zeros. This make this dataset better fit for Zero Inflated Poisson regression model. Note that we need to specify parameter for Zero Inflated Poisson regression model indicating what feature can contribute to more occurence of records with zero target variable. Obvious choice here is STARS variable. It is intuitive to think that if the number of STARS are more it will contribute to increased count of wine sample purchase. We have fitted our model indicating STARS as the parameter which impact number of records with zero target variable and we can see that it is resulting into lesser RMSE indicating better model fit compare to regular Poisson regression model.

7. Fit zero inflated Negative Binomial Regression model

```
model.nb.zip= hurdle(target~  LabelAppeal + STARS + AcidIndex + FixedAcidity|STARS, dist='negbi
n', data=train.data)
summary(model.nb.zip)
```

```
##
## Call:
## hurdle(formula = target ~ LabelAppeal + STARS + AcidIndex + FixedAcidity |
##     STARS, data = train.data, dist = "negbin")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.9993 -0.5194  0.0778  0.4930  2.0641
##
## Count model coefficients (truncated negbin with log link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.2131011  0.0491121  24.701  < 2e-16 ***
## LabelAppeal   0.2452836  0.0078590  31.211  < 2e-16 ***
## STARS         0.1123598  0.0076063  14.772  < 2e-16 ***
## AcidIndex    -0.0216747  0.0059830  -3.623 0.000292 ***
## FixedAcidity  0.0004453  0.0010340   0.431 0.666746
## Log(theta)   11.8343661  3.7448485   3.160 0.001577 **
## Zero hurdle model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.19463    0.08961  -24.49   <2e-16 ***
## STARS        2.40909    0.07114   33.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta: count = 137911.3157
## Number of iterations in BFGS optimization: 28
## Log-likelihood: -1.501e+04 on 8 Df
```

```
rmse.nb.zip = model.fit.evaluate.rmse(model.nb.zip, test.data, test.values)
print(rmse.nb.zip)
```

```
## [1] 89.84987
```

We can see that Zero Inflated Poisson regression model has lower RMSE compared to Zero Inflated Negative Binomial model. This is due to the fact that dataset has under dispersion and Negative Binomial model is not reccomended when there is under dispersion in the dataset.

8. Fit Multinomial Logistic Regression model with selected variables

Since this is a Multinomial Regression model, we will use Accuracy as a model evaluation metric as stated in the model evaluation criteria above

```
train.data.class = train.data
train.data.class$target = factor(train.data.class$target)
model.multi= multinom(target~  LabelAppeal + STARS + AcidIndex + FixedAcidity,  data=train.data.
class)
```

```
## # weights:  54 (40 variable)
## initial  value 19678.343315
## iter  10 value 15272.975080
## iter  20 value 14105.755485
## iter  30 value 13120.534943
## iter  40 value 11900.468673
## iter  50 value 11651.438339
## iter  60 value 11638.797590
## iter  70 value 11635.425597
## final  value 11635.394124
## converged
```

```
summary(model.multi)
```

```
## Call:
## multinom(formula = target ~ LabelAppeal + STARS + AcidIndex +
##     FixedAcidity, data = train.data.class)
##
## Coefficients:
##    (Intercept) LabelAppeal     STARS  AcidIndex FixedAcidity
## 1  -4.12164444  -3.3624742 0.6573153 -0.2058376 -0.017017269
## 2  -1.27465702  -2.1461083 1.5720766 -0.3317468 -0.010355005
## 3  -0.29848320  -0.9321725 2.1682663 -0.3495449 -0.012751877
## 4  -0.08134995   0.2701936 2.8057220 -0.4873023 -0.008308202
## 5  -1.78517431   1.3367197 3.4349859 -0.5586506 -0.014705256
## 6  -4.58144235   2.3610247 4.0847651 -0.6738233 -0.002462028
## 7  -9.48640573   3.5425672 4.9847748 -0.8009662 -0.007358204
## 8 -22.89648967   6.2804363 6.3619335 -0.5964125  0.009640870
##
## Std. Errors:
##   (Intercept) LabelAppeal      STARS  AcidIndex FixedAcidity
## 1   0.6653281  0.15845542 0.21872827 0.07142347  0.014277094
## 2   0.3336031  0.07826628 0.09863049 0.03898681  0.007753479
## 3   0.2436938  0.05270857 0.08059825 0.02887085  0.005979224
## 4   0.2519086  0.05211286 0.08064641 0.03073973  0.006080751
## 5   0.3176997  0.06653866 0.08741593 0.03874698  0.007320162
## 6   0.4823453  0.09930873 0.10387394 0.05772491  0.010223001
## 7   1.0429964  0.20357731 0.17959752 0.11805314  0.018903049
## 8   3.8523193  1.05465625 0.64080980 0.28175333  0.047933614
##
## Residual Deviance: 23270.79
## AIC: 23350.79
```

```
accuracy.multi = model.fit.evaluate.mcr(model.multi, test.data, test.values)
print(accuracy.multi)
```

```
## [1] 0.4469914
```

From the above Accuracy metric we can say that with the Accuracy score of 0.44 we can predict wine sample purchase count with 44% accuracy. This is not a great accuracy score. This is expected since Multinomial regression model is not relevant for this problem statement. Multinomial regression model can be used to predict un-ordered target class with multiple values. However in this problem statement we are dealing with target variable which is wine sample purchase count. The target variable is ordered here and that makes Poisson regression model more applicable here compare to Multinomial Logistic Regression model

9. Fit multiple linear regression model with all the variables

```
model.multilr= lm(target~  .,  data=train.data)
summary(model.multilr)
```

```
##
## Call:
## lm(formula = target ~ ., data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8162 -0.9679  0.0981  0.9870  4.4728
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         5.311e+00  5.682e-01   9.347  < 2e-16 ***
## FixedAcidity       -3.680e-03  2.393e-03  -1.538 0.124083
## VolatileAcidity    -1.111e-01  1.877e-02  -5.919 3.35e-09 ***
## CitricAcid          9.282e-03  1.713e-02   0.542 0.587834
## ResidualSugar       3.338e-04  4.376e-04   0.763 0.445602
## Chlorides          -1.653e-01  4.625e-02  -3.573 0.000355 ***
## FreeSulfurDioxide   3.235e-04  9.970e-05   3.245 0.001179 **
## TotalSulfurDioxide  2.261e-04  6.395e-05   3.535 0.000409 ***
## Density            -1.335e+00  5.528e-01  -2.416 0.015724 *
## pH                 -4.743e-02  2.175e-02  -2.181 0.029225 *
## Sulphates          -4.907e-02  1.586e-02  -3.094 0.001979 **
## Alcohol             1.262e-02  3.939e-03   3.204 0.001359 **
## LabelAppeal         4.141e-01  2.923e-02  14.170  < 2e-16 ***
## AcidIndex          -2.723e-01  1.502e-02 -18.136  < 2e-16 ***
## STARS               1.626e+00  3.390e-02  47.981  < 2e-16 ***
## StarTran           -8.752e-01  5.419e-02 -16.151  < 2e-16 ***
## AcidIndexTran       2.125e-01  5.815e-02   3.654 0.000260 ***
## LabelAppealTran     4.821e-02  4.773e-02   1.010 0.312421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.396 on 8938 degrees of freedom
## Multiple R-squared:  0.4736, Adjusted R-squared:  0.4726
## F-statistic:   473 on 17 and 8938 DF,  p-value: < 2.2e-16
```

```
rmse.multilr = model.fit.evaluate.rmse(model.multilr, test.data, test.values)
print(rmse.multilr)
```

```
## [1] 88.32893
```

10. Fit multiple linear regression model with selected variables and cubic transformation of Stars variable

```
model.multilr.sel= lm(target~  LabelAppeal + poly(STARS,3) + AcidIndex + FixedAcidity,  data=tra
in.data)
summary(model.multilr.sel)
```

```
##
## Call:
## lm(formula = target ~ LabelAppeal + poly(STARS, 3) + AcidIndex +
##     FixedAcidity, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5693 -0.9937  0.0878  0.9859  4.8195
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.972428   0.090692  54.827  < 2e-16 ***
## LabelAppeal       0.435210   0.017485  24.891  < 2e-16 ***
## poly(STARS, 3)1  98.424916   1.487213  66.181  < 2e-16 ***
## poly(STARS, 3)2 -21.415300   1.407379 -15.216  < 2e-16 ***
## poly(STARS, 3)3   7.821419   1.404827   5.568 2.66e-08 ***
## AcidIndex        -0.246016   0.011679 -21.066  < 2e-16 ***
## FixedAcidity     -0.003878   0.002405  -1.613    0.107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.405 on 8949 degrees of freedom
## Multiple R-squared:  0.4665, Adjusted R-squared:  0.4662
## F-statistic:  1304 on 6 and 8949 DF,  p-value: < 2.2e-16
```

```
rmse.multilr.sel = model.fit.evaluate.rmse(model.multilr.sel, test.data, test.values)
print(rmse.multilr.sel)
```

```
## [1] 89.21323
```

Interestingly we are getting similar (close enough) RMSE between Zero Inflated Possion Regression model and Multiple Linear regression model with selected transformed Stars variable. This is possible here because distribution of target variable was not strictly Poisson. The distribution looked more kind of bimodal with skew on the right side. We can say that both Zero inflated Poisson regression model perform as good as multiple linear regression model with quadratic term of Stars variable.

Model Selection

From the above model fit we can conclude following

- **For Inference we can select Robust Poisson Regression model. We are selecting this model for inference even though it doesn't have great RMSE. We are doing this because Robust Poisson regression model is interpretable comapred to zero inflated Poisson regression model and std error estimation is more robust for Robust Poisson regression model. This is more important here becuase dataset have under dispersion and we need to stick to Robust Poisson regression model for inference rather that simple Poisson regressio model**
- **For prediction we can select Zero Inflated Poisson Regression model. We are selecting this model because it has lower RMSE. This not only indicates better model fit but also makes this model perfect fit for the given dataset because dataset has unusual high number of wine sample observation with zero count. Zero inflated Poisson regression model can handle such datasets well and can greately improves prediction accuracy by taking into acount parameters which can influence the occurence of zero records while making prediction**

1. Model Inference

```
summary(model.poss.sel.robust)
```

```
##
## Call:
## glm(formula = target ~ LabelAppeal + STARS + AcidIndex + FixedAcidity,
##     family = quasipoisson, data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8899  -0.6853   0.1378   0.6380   2.8229
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.181572   0.042681  27.684   <2e-16 ***
## LabelAppeal   0.138444   0.006855  20.196   <2e-16 ***
## STARS         0.343813   0.006324  54.370   <2e-16 ***
## AcidIndex    -0.101036   0.005097 -19.824   <2e-16 ***
## FixedAcidity -0.001494   0.000927  -1.611    0.107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 0.8871091)
##
##     Null deviance: 15939  on 8955  degrees of freedom
## Residual deviance: 11256  on 8951  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

From the above model summary we can say that following predicters are statistically significant while predicting target sample wine sale count

- **Label Appeal: Positive coefficient indicates one unit increase in Label Appeal incrases the wine sample count**

- **STARS: Positive coefficient indicates that one unit increase in STARS will increase the wine sample sale count**
- **Acid Index: Negative coefficient indicates that one unit increae in Acid index will reduce the wine sample sale count**
- **Fixed Acidity: Negative coefficient indicates that one unit increae in Fixed Acidity will reduce the wine sample sale count**

This goes with our intuition as well. Above coefficients indicate that better Label Appeal and Stars count increases the wine sample sale count

We can say that people don't like wine with high alcoholic content. As Acid Index and Fixed Acidity increases it negatively impacts wine sample sale count

The coefficient for Stars is 0.34. This indicates that with all other coefficient held constant one unit increase in Stars increases **log** count of wine sale by 0.34

Calculate impact of Stars count on wine sale

```
exp(0.34)
```

```
## [1] 1.404948
```

This analysis indicates that with all other coefficeint held constant, one unit increase in Stars count increases wine sample sale count by 40%. This is a significant impact and that's why Stars count is very important for wine sale

Calculate impact of Lable Appeal on wine sale

```
exp(0.13)
```

```
## [1] 1.138828
```

This analysis indicates that with all other coefficeint held constant, one unit increase in Label Appeal increases wine sample sale count by 13%.

2. Model Prediction

For prediction we will use Zero Inflated Poisson Regression model due to it's lower RMSE and better model fit

5. Select Models

1. Read evaluation data and impute missing columns. Perform prediction

```
testing = read.csv("./Data/wine-evaluation-data.csv", stringsAsFactors = FALSE, na.strings=c("N
A","NaN", " ", ""))

comp.data.test <- mice(testing,m=2,maxit=10,meth='pmm',seed=500)
```

```
##
##  iter imp variable
##   1   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   1   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   2   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   2   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   3   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   3   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   4   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   4   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   5   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   5   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   6   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   6   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   7   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   7   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   8   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   8   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   9   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   9   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alco
hol  STARS
##   10   1  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
ohol  STARS
##   10   2  ResidualSugar  Chlorides  FreeSulfurDioxide  TotalSulfurDioxide  pH  Sulphates  Alc
ohol  STARS
```

```
## Warning: Number of logged events: 1
```

```
testing = complete(comp.data.test)

out = predict(model.poss.zip, testing)
testing$TARGET = round(out)
write.table(testing, "./Data/PredictedOutcome.csv", row.names = FALSE, sep=",")
```

6. Appendix

```r
model.fit.evaluate.mcr <- function(model, test.data, test.values) {
  output = predict(model, test.data)
  mat = table(output, test.values)
  mcr = sum(diag(mat))/sum(mat)
  return(mcr)
}

model.fit.evaluate.rmse <- function(model, test.data, test.values) {
  output = predict(model, test.data)
  rmse = (sum((output - test.values)^2)) ^ 0.5
}
```