Reed Ballesteros
MSDS-410-DL, Summer 2022
Dr. Mickelson
8/7/2022

Modeling Assignment 8: Modeling Dichotomous Responses

Assignment Overview

A large wine manufacturer is interested in being able 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. But, it is also important to understand what influences purchase decisions in the first place, as well as what contributes to the quality of the wine. Your task in this assignment is to model the purchase decision using Logistic Regression models.

This data set contains information on approximately 12,000 commercially available wines. A record can be considered the data associated with a bottle of wine. The explanatory variables are mostly related to the chemical properties of the wine. But, there are other variables as well. For example, the PURCHASE reflects whether or not a purchase was made of that wine. PURCHASE is the response variable for this assignment. The variable CASES then indicates the number of cases purchased. 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 a wine is to be sold at a high end restaurant. Similarly, each wine, when possible, was rated by a panel of experts as to its quality (STARS).

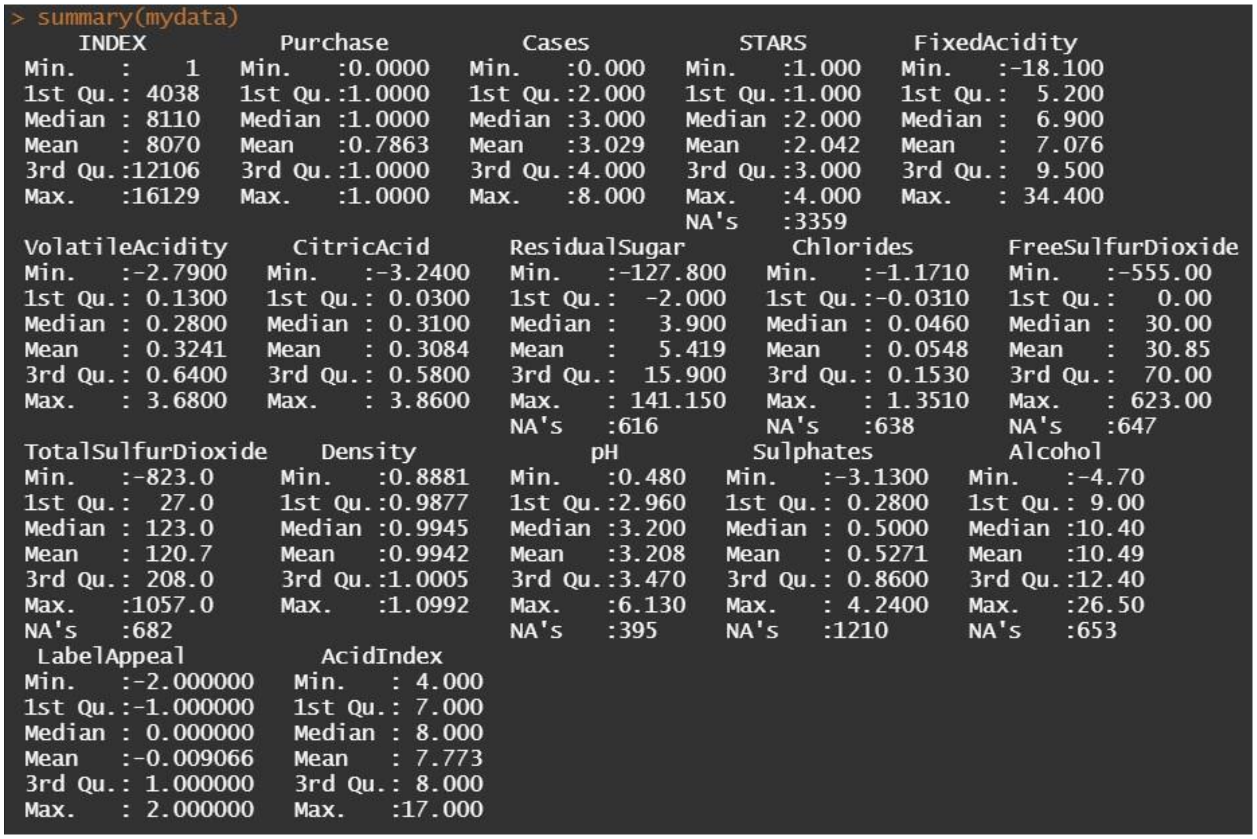
From a statistical perspective, please note the size of the sample: n is approximately 12,000 records. You should immediately be thinking, “I have tons of statistical power.” I have to be careful about statistical significance, as it is not the be all and end all. Again, you can think about randomly splitting the file into a 70% model development dataset, and into 30% validation data set, if you wish.

Assignment Tasks

1. **Use your data analysis knowledge to date, to conduct an Exploratory Data Analysis (EDA) for fitting Logistic Regression models to predict the PURCHASE decision.**

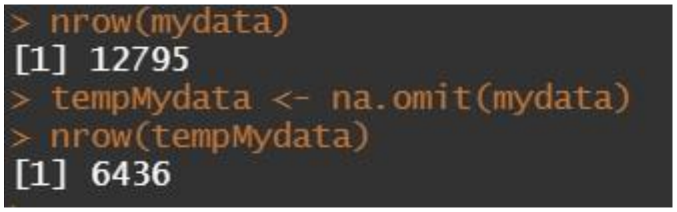
Summary

The wine dataset contains 12795 observations, or kinds of wines, with various chemical properties and other properties of each wine listed.



```
> summary(mydata)
     INDEX          Purchase           Cases            STARS        FixedAcidity
 Min.   :    1   Min.   :0.0000   Min.   :0.000   Min.   :1.000   Min.   :-18.100
 1st Qu.: 4038   1st Qu.:1.0000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:  5.200
 Median : 8110   Median :1.0000   Median :3.000   Median :2.000   Median :  6.900
 Mean   : 8070   Mean   :0.7863   Mean   :3.029   Mean   :2.042   Mean   :  7.076
 3rd Qu.:12106   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:3.000   3rd Qu.:  9.500
 Max.   :16129   Max.   :1.0000   Max.   :8.000   Max.   :4.000   Max.   : 34.400
                                                  NA's   :3359
 VolatileAcidity     CitricAcid       ResidualSugar        Chlorides       FreeSulfurDioxide
 Min.   :-2.7900   Min.   :-3.2400   Min.   :-127.800   Min.   :-1.1710   Min.   :-555.00
 1st Qu.: 0.1300   1st Qu.: 0.0300   1st Qu.:  -2.000   1st Qu.:-0.0310   1st Qu.:   0.00
 Median : 0.2800   Median : 0.3100   Median :   3.900   Median : 0.0460   Median :  30.00
 Mean   : 0.3241   Mean   : 0.3084   Mean   :   5.419   Mean   : 0.0548   Mean   :  30.85
 3rd Qu.: 0.6400   3rd Qu.: 0.5800   3rd Qu.:  15.900   3rd Qu.: 0.1530   3rd Qu.:  70.00
 Max.   : 3.6800   Max.   : 3.8600   Max.   : 141.150   Max.   : 1.3510   Max.   : 623.00
                                     NA's   :616        NA's   :638       NA's   :647
 TotalSulfurDioxide    Density             pH          Sulphates          Alcohol
 Min.   :-823.0     Min.   :0.8881   Min.   :0.480   Min.   :-3.1300   Min.   :-4.70
 1st Qu.:  27.0     1st Qu.:0.9877   1st Qu.:2.960   1st Qu.: 0.2800   1st Qu.: 9.00
 Median : 123.0     Median :0.9945   Median :3.200   Median : 0.5000   Median :10.40
 Mean   : 120.7     Mean   :0.9942   Mean   :3.208   Mean   : 0.5271   Mean   :10.49
 3rd Qu.: 208.0     3rd Qu.:1.0005   3rd Qu.:3.470   3rd Qu.: 0.8600   3rd Qu.:12.40
 Max.   :1057.0     Max.   :1.0992   Max.   :6.130   Max.   : 4.2400   Max.   :26.50
 NA's   :682                         NA's   :395     NA's   :1210      NA's   :653
  LabelAppeal          AcidIndex
 Min.   :-2.000000   Min.   : 4.000
 1st Qu.:-1.000000   1st Qu.: 7.000
 Median : 0.000000   Median : 8.000
 Mean   :-0.009066   Mean   : 7.773
 3rd Qu.: 1.000000   3rd Qu.: 8.000
 Max.   : 2.000000   Max.   :17.000
```

From running a summary over the dataset, we notice the properties STARS, ResidualSugar, Cholrides, FreeSulfurDioxide, TotalSulfurDioxide, pH, Sulphates, and Alcohol contain null values. Removing any rows or wines that contain any nulls would greatly reduce the dataset by about 50%.



```
> nrow(mydata)
[1] 12795
> tempMydata <- na.omit(mydata)
> nrow(tempMydata)
[1] 6436
```

We describe on how we handle null values for various properties below while trying to preserve as much of the wine observations as possible.

Removing INDEX Variable

As indicated in the Wine Data Dictionary, the INDEX property is just an identification variable and should not be used in any numerical functions or modelling.

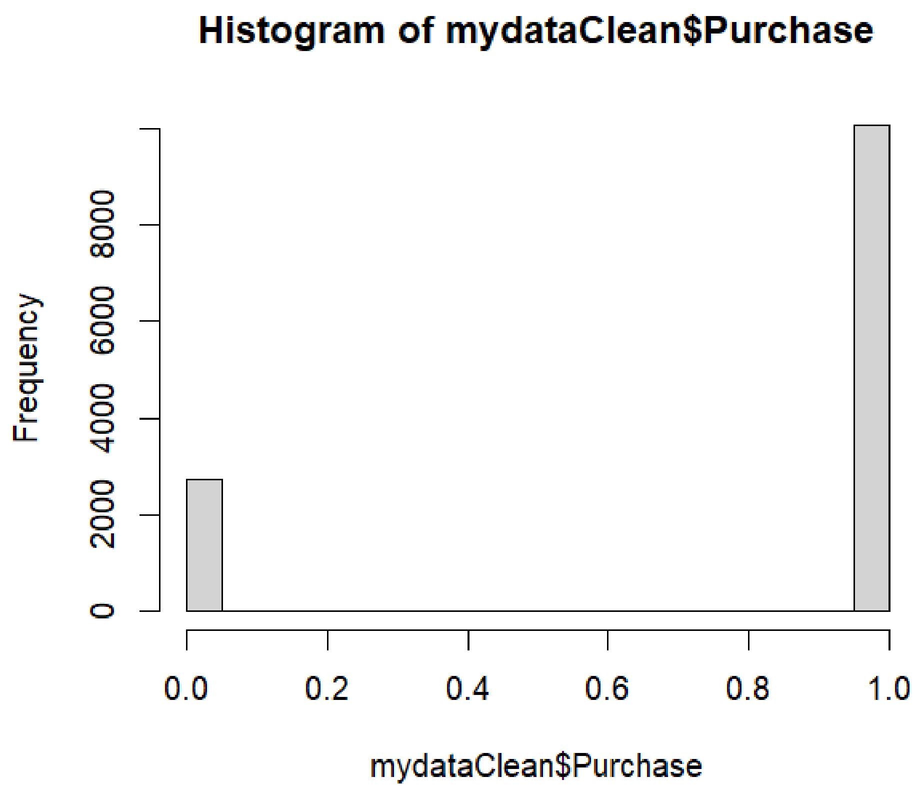
Removing Cases Variable

Ordering the cases are based after the decision to purchase a given wine. Because of this, the Cases variable would be heavily biased towards wines with Purchase = 1, as the variation of the number of cases is only based on the decision to purchase a wine. Given its after-the-fact nature, we will remove the Cases variable from the dataset.

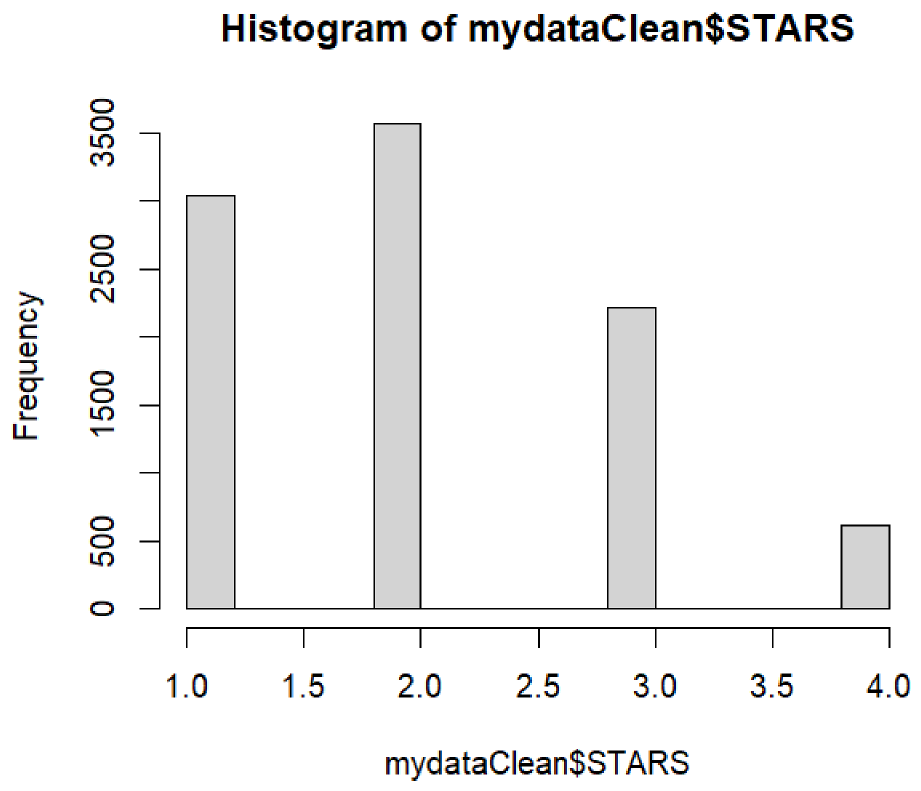
Histograms

We create the following histograms for the remaining variables in the dataset.

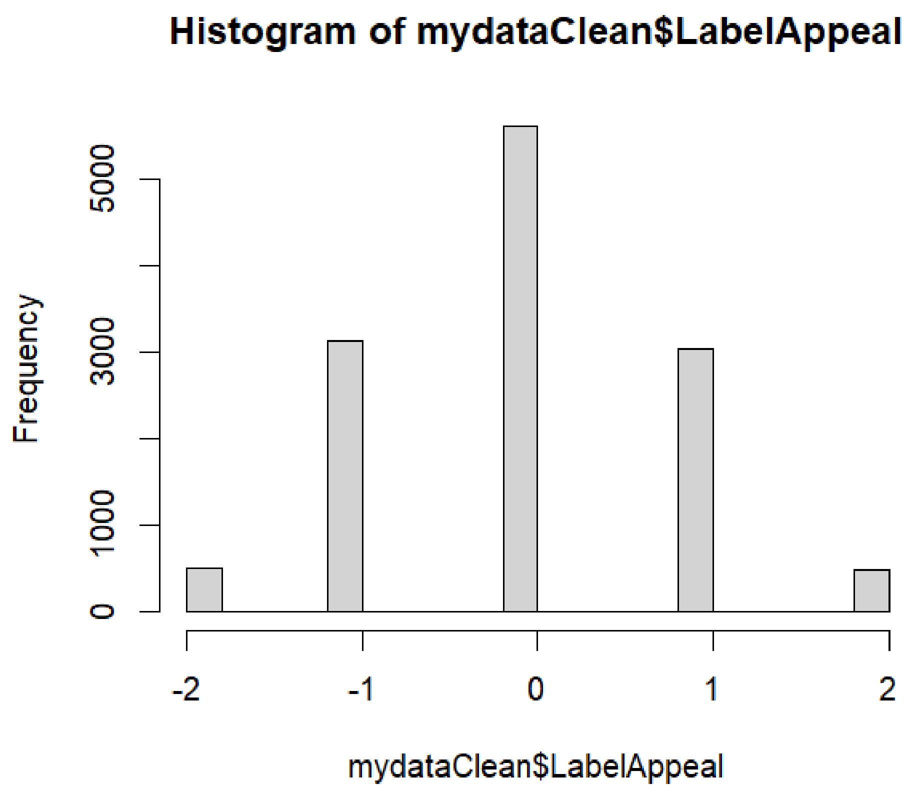
Purchase: we can see from the histogram that an overwhelming number of wines in the dataset have Purchase = 1.



STARS: we see a right-skewed distribution of stars, in which wines that are rated with 4 stars are fairly rare in the dataset.

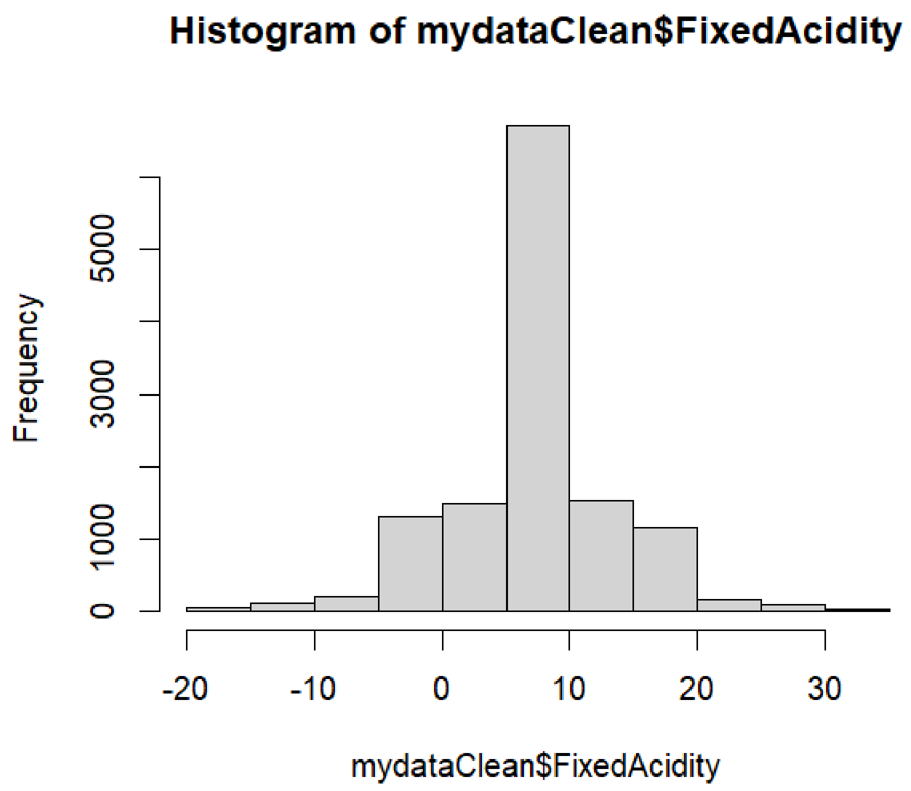


LabelAppeal: we notice that LabelAppleal is fairly normally distributed in the dataset, in which we see that people are fairly indifferent to a wine's label appeal, and that they are rarely very bad (-2) or very good (2).



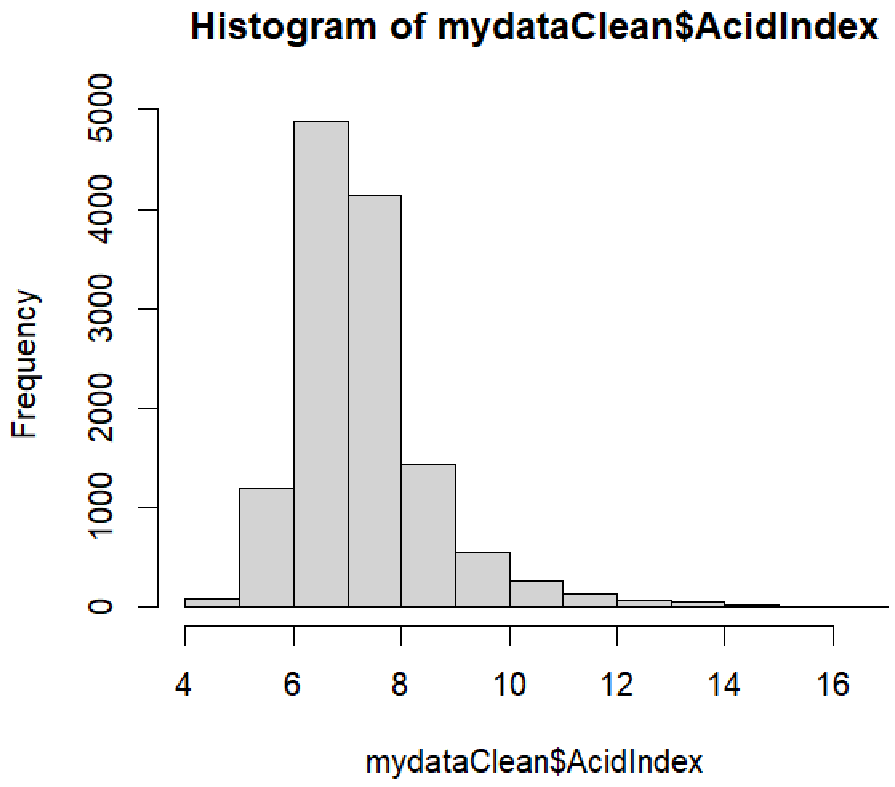
For the various chemical properties of the wines in the dataset, we find them very normally distributed with a very high peak at the mean of their respective property. This shows that their respective standard deviation is small in context to their range of values and there is not much variation in their values.

For example, let us look at the FixedAcidity property:



We see a normal distribution of values, but also a very tall and narrow one as well, showing mostly just small variations of FixedAcidity from the mean.

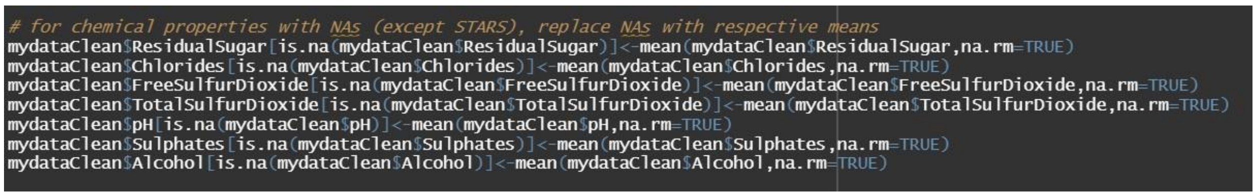
We see this pattern for practically all of the chemical properties, except for AcidIndex:



AcidIndex shows more of a right-skewed distribution, with rare outliers fairly far from the mean.

Handling NAs – Chemical Properties

Since we've observed tall and narrow normal distributions of most of the chemical properties in the dataset, we will replace chemical properties with null values with their respective means.

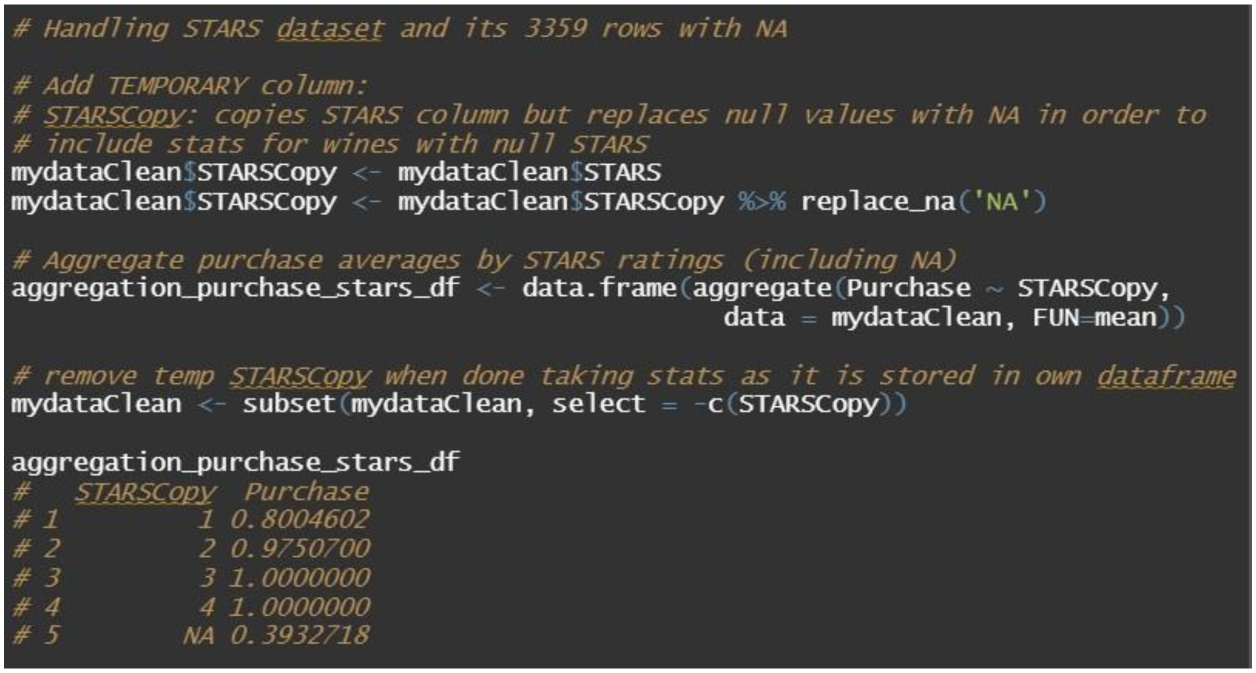


```
# for chemical properties with NAs (except STARS), replace NAs with respective means
mydataClean$ResidualSugar[is.na(mydataClean$ResidualSugar)]<-mean(mydataClean$ResidualSugar,na.rm=TRUE)
mydataClean$Chlorides[is.na(mydataClean$Chlorides)]<-mean(mydataClean$Chlorides,na.rm=TRUE)
mydataClean$FreeSulfurDioxide[is.na(mydataClean$FreeSulfurDioxide)]<-mean(mydataClean$FreeSulfurDioxide,na.rm=TRUE)
mydataClean$TotalSulfurDioxide[is.na(mydataClean$TotalSulfurDioxide)]<-mean(mydataClean$TotalSulfurDioxide,na.rm=TRUE)
mydataClean$pH[is.na(mydataClean$pH)]<-mean(mydataClean$pH,na.rm=TRUE)
mydataClean$Sulphates[is.na(mydataClean$Sulphates)]<-mean(mydataClean$Sulphates,na.rm=TRUE)
mydataClean$Alcohol[is.na(mydataClean$Alcohol)]<-mean(mydataClean$Alcohol,na.rm=TRUE)
```

Handling NAs – STARS, and New Property HasSTARS

Removing the 3359 wines in the dataset with null STARS values would reduce the dataset by 26%. We would want to preserve those wines as they could provide helpful information that could be useful for modelling.

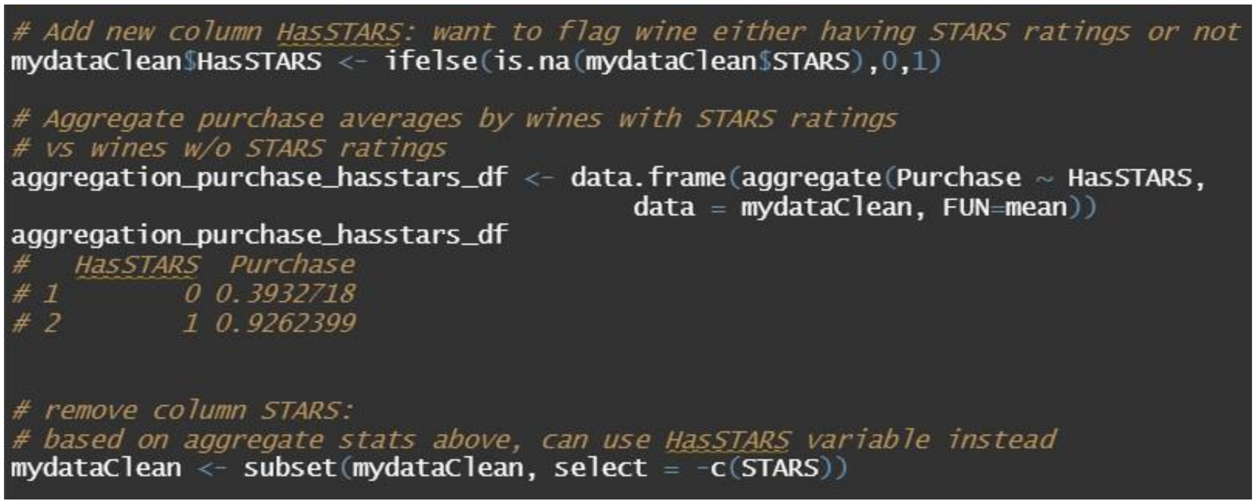
We create a temporary column in the dataset that copies the STARS data, but fills in null values with 'NA'. We can use this to aggregate purchase averages by STARS ratings, including those with no ratings. We place this data into its own dataframe and remove the temporary column.



```
# Handling STARS dataset and its 3359 rows with NA

# Add TEMPORARY column:
# STARSCopy: copies STARS column but replaces null values with NA in order to
# include stats for wines with null STARS
mydataClean$STARSCopy <- mydataClean$STARS
mydataClean$STARSCopy <- mydataClean$STARSCopy %>% replace_na('NA')

# Aggregate purchase averages by STARS ratings (including NA)
aggregation_purchase_stars_df <- data.frame(aggregate(Purchase ~ STARSCopy,
                                                      data = mydataClean, FUN=mean))

# remove temp STARSCopy when done taking stats as it is stored in own dataframe
mydataClean <- subset(mydataClean, select = -c(STARSCopy))

aggregation_purchase_stars_df
#   STARSCopy  Purchase
# 1         1 0.8004602
# 2         2 0.9750700
# 3         3 1.0000000
# 4         4 1.0000000
# 5        NA 0.3932718
```

We can see from the aggregation above that an overwhelming majority of wines with any STARS rating end up getting purchased. The purchase variation is shown with unrated wines, where only about 40% of unrated wines get purchased. Because of that, we want to preserve these unrated wines in the dataset.

With the overwhelming majority of rated wines being purchased, we will replace the STARS property with new property HasSTARS, a Boolean value which simply flags a wine with 0 for no rating, and 1 for having a rating. Let us aggregate purchase averages between wines with a rating and those without.



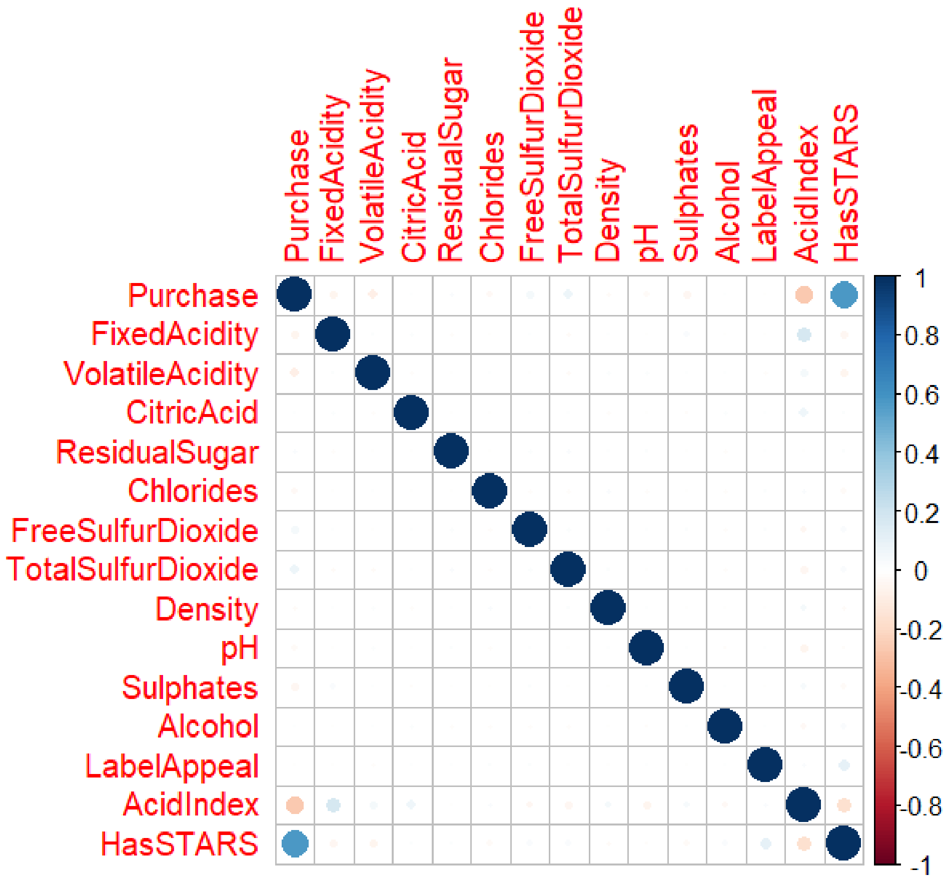
```
# Add new column HasSTARS: want to flag wine either having STARS ratings or not
mydataClean$HasSTARS <- ifelse(is.na(mydataClean$STARS),0,1)

# Aggregate purchase averages by wines with STARS ratings
# vs wines w/o STARS ratings
aggregation_purchase_hasstars_df <- data.frame(aggregate(Purchase ~ HasSTARS,
                                        data = mydataClean, FUN=mean))
aggregation_purchase_hasstars_df
#   HasSTARS  Purchase
# 1        0 0.3932718
# 2        1 0.9262399


# remove column STARS:
# based on aggregate stats above, can use HasSTARS variable instead
mydataClean <- subset(mydataClean, select = -c(STARS))
```

From the aggregation above, almost 93% of wines with any rating in the dataset get purchased, where only about 40% of unrated wines get purchased.

With these changes to the wines dataset made, we have a full non-null dataset we can use for modelling.

Correlations

Before we get into modeling, let us create a correlation matrix:



Because of the small standard deviations of the various chemical properties, the correlation matrix shows almost 0 correlations of most of them against the Purchase dependent variable. AcidIndex's right skewed distribution gives it some negative correlation to Purchase, while the HasSTARS variable has a very strong correlation to Purchase.

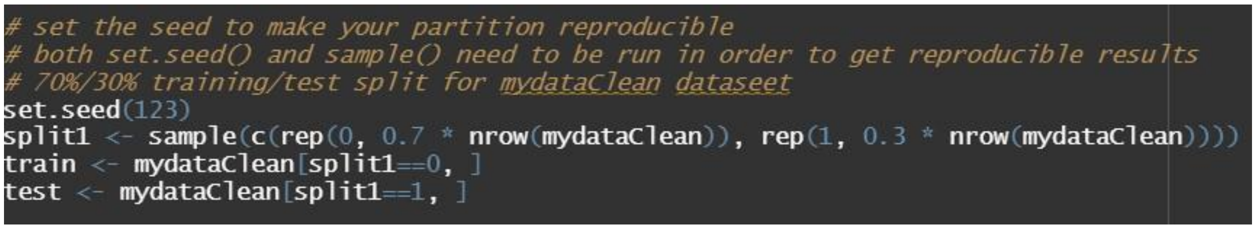
2. **There is not one perfectly correct way to approach model building. You are now charged with the task of producing your best predictive model for the PURCHASE (Y) decision. This is an open-ended modeling task. You may select the variables manually, or use an automated approach such as Forward or Stepwise. You may use continuous or categorical variables as part of the explanatory variable set. You have enough data, so you should very seriously consider taking a validation approach to this modeling endeavor, though it is not required. You need to be sure you can interpret your models, have evidence on goodness of fit, and check on assumptions via diagnostics. What criteria are you going to use select your "best" model?**

   Write a description of the technique you used to decide on your final model. Write up your final model. Report the model. Discuss the coefficients in the model, do they make sense? Report on goodness of fit and model diagnostics.

   We created three models: a full model based on the dataset I decided to used based on the EDA performed, a reduced model with only statistically significant variables, and a much more reduced model using only highly-correlated variables. I cut my my dataset into a 70% Training/30% Test split and used it towards validation and comparison among the models, which I will describe below.

   70% Training/30% Test Dataset Split

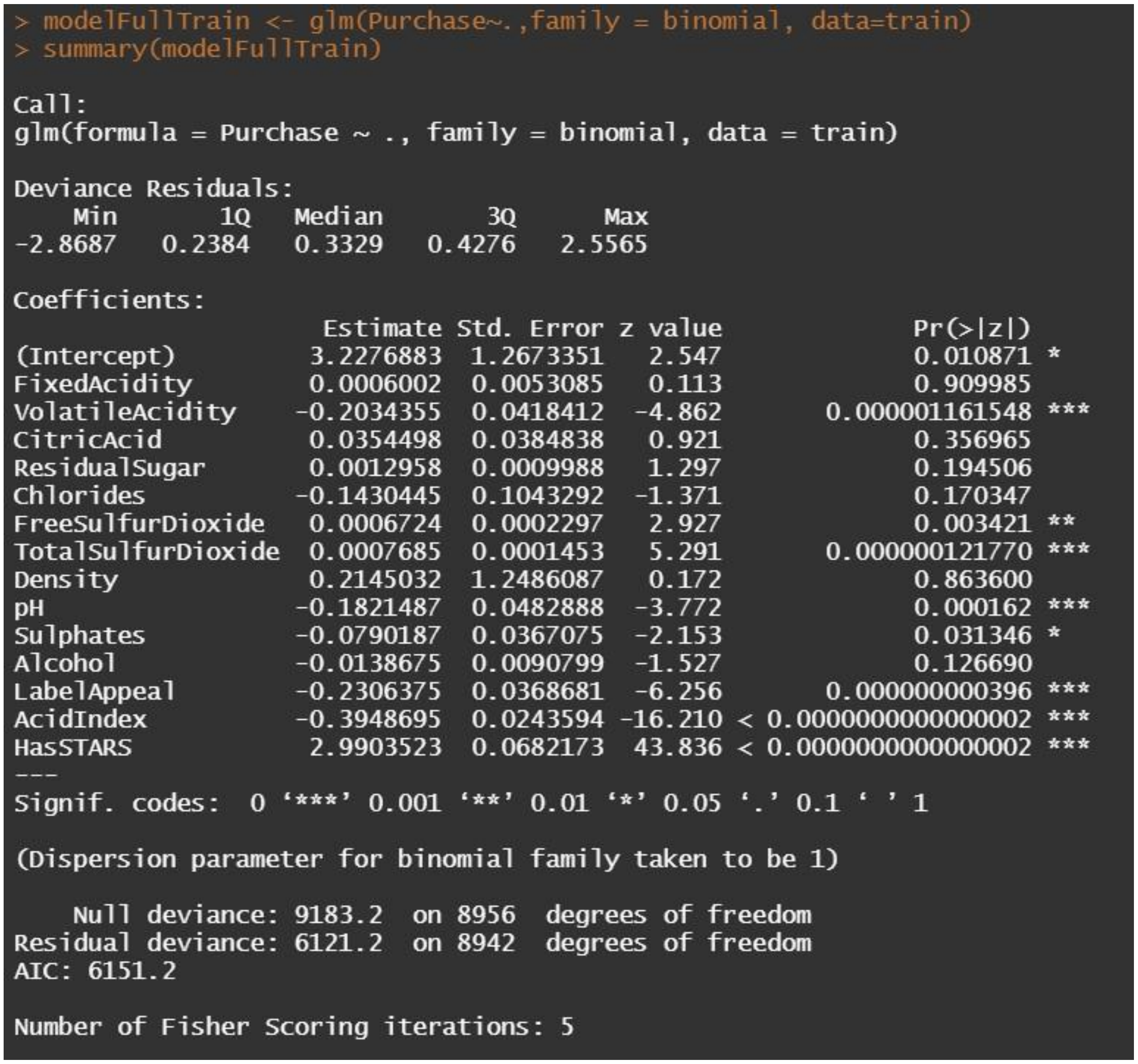
   Before modelling, we will randomly split the dataset into 70% to be used for training a model, and 30% used for testing.



```
# set the seed to make your partition reproducible
# both set.seed() and sample() need to be run in order to get reproducible results
# 70%/30% training/test split for mydataClean dataseet
set.seed(123)
split1 <- sample(c(rep(0, 0.7 * nrow(mydataClean)), rep(1, 0.3 * nrow(mydataClean))))
train <- mydataClean[split1==0, ]
test <- mydataClean[split1==1, ]
```

Full Model

We create a logistic regression model with the following coefficients:

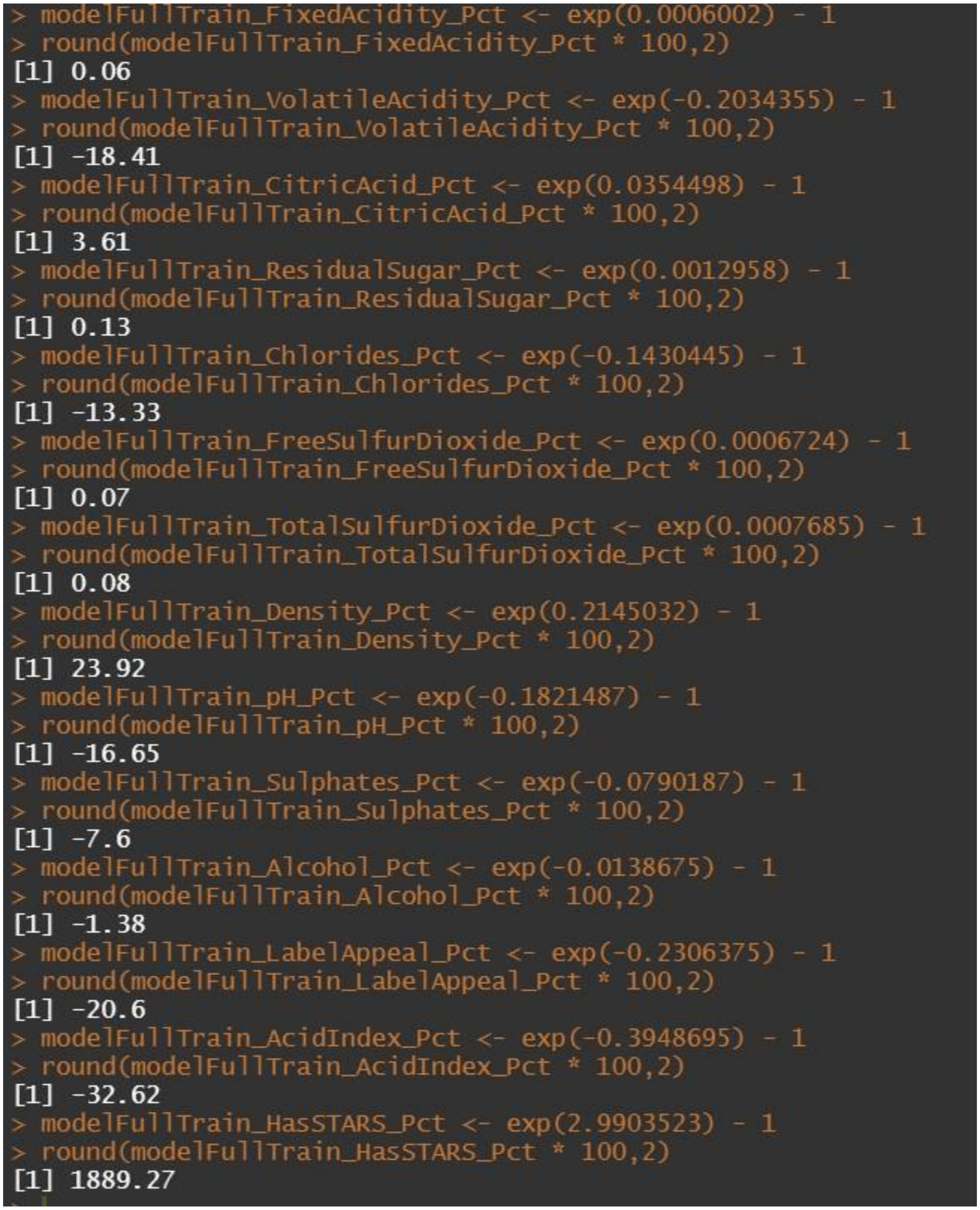


```
> modelFullTrain <- glm(Purchase~.,family = binomial, data=train)
> summary(modelFullTrain)

Call:
glm(formula = Purchase ~ ., family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8687   0.2384   0.3329   0.4276   2.5565

Coefficients:
                    Estimate Std. Error z value             Pr(>|z|)
(Intercept)        3.2276883  1.2673351   2.547             0.010871 *
FixedAcidity       0.0006002  0.0053085   0.113             0.909985
VolatileAcidity   -0.2034355  0.0418412  -4.862       0.000001161548 ***
CitricAcid         0.0354498  0.0384838   0.921             0.356965
ResidualSugar      0.0012958  0.0009988   1.297             0.194506
Chlorides         -0.1430445  0.1043292  -1.371             0.170347
FreeSulfurDioxide  0.0006724  0.0002297   2.927             0.003421 **
TotalSulfurDioxide 0.0007685  0.0001453   5.291       0.000000121770 ***
Density            0.2145032  1.2486087   0.172             0.863600
pH                -0.1821487  0.0482888  -3.772             0.000162 ***
Sulphates         -0.0790187  0.0367075  -2.153             0.031346 *
Alcohol           -0.0138675  0.0090799  -1.527             0.126690
LabelAppeal       -0.2306375  0.0368681  -6.256       0.000000000396 ***
AcidIndex         -0.3948695  0.0243594 -16.210 < 0.0000000000000002 ***
HasSTARS           2.9903523  0.0682173  43.836 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9183.2  on 8956  degrees of freedom
Residual deviance: 6121.2  on 8942  degrees of freedom
AIC: 6151.2

Number of Fisher Scoring iterations: 5
```

We see high p-values for FixedActiviy, CitiricAcid, ResidualSugar, Chlorides, Density, and Alcohol, which indicate that they might not be statistically significant, and could be possible to reduce the model by not including them and without affecting the purchase outcome much.

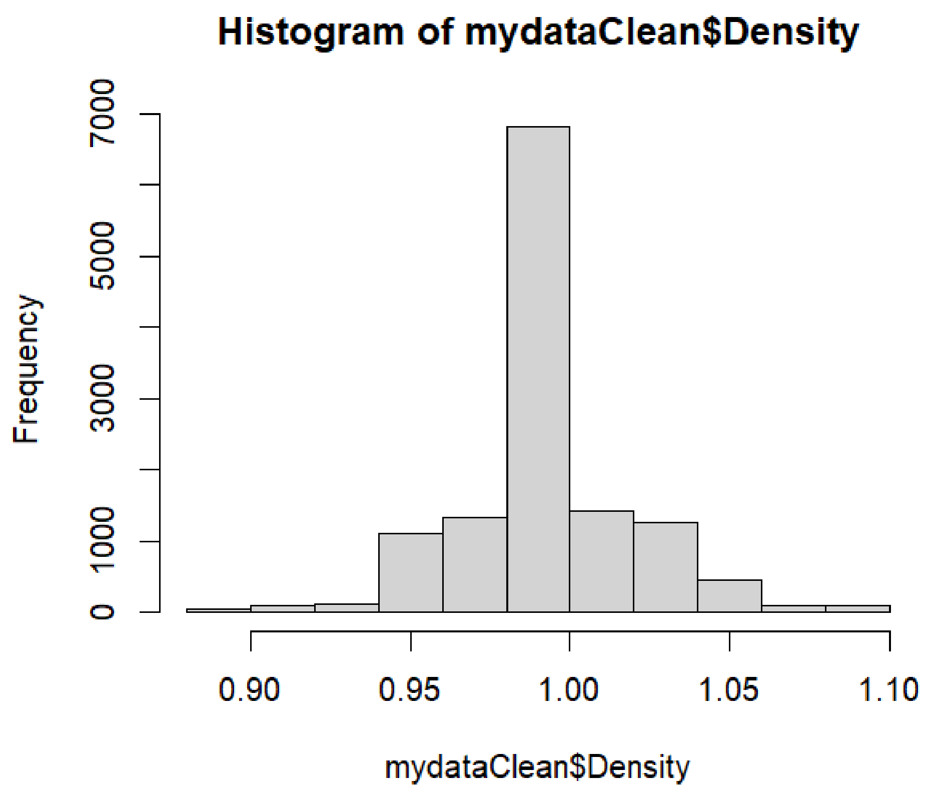
The coefficients in the summary for the full model above are the expected change in log odds of having the outcome per unit change in X.

Converting log odds into percentages using exp(coefficient) -1:



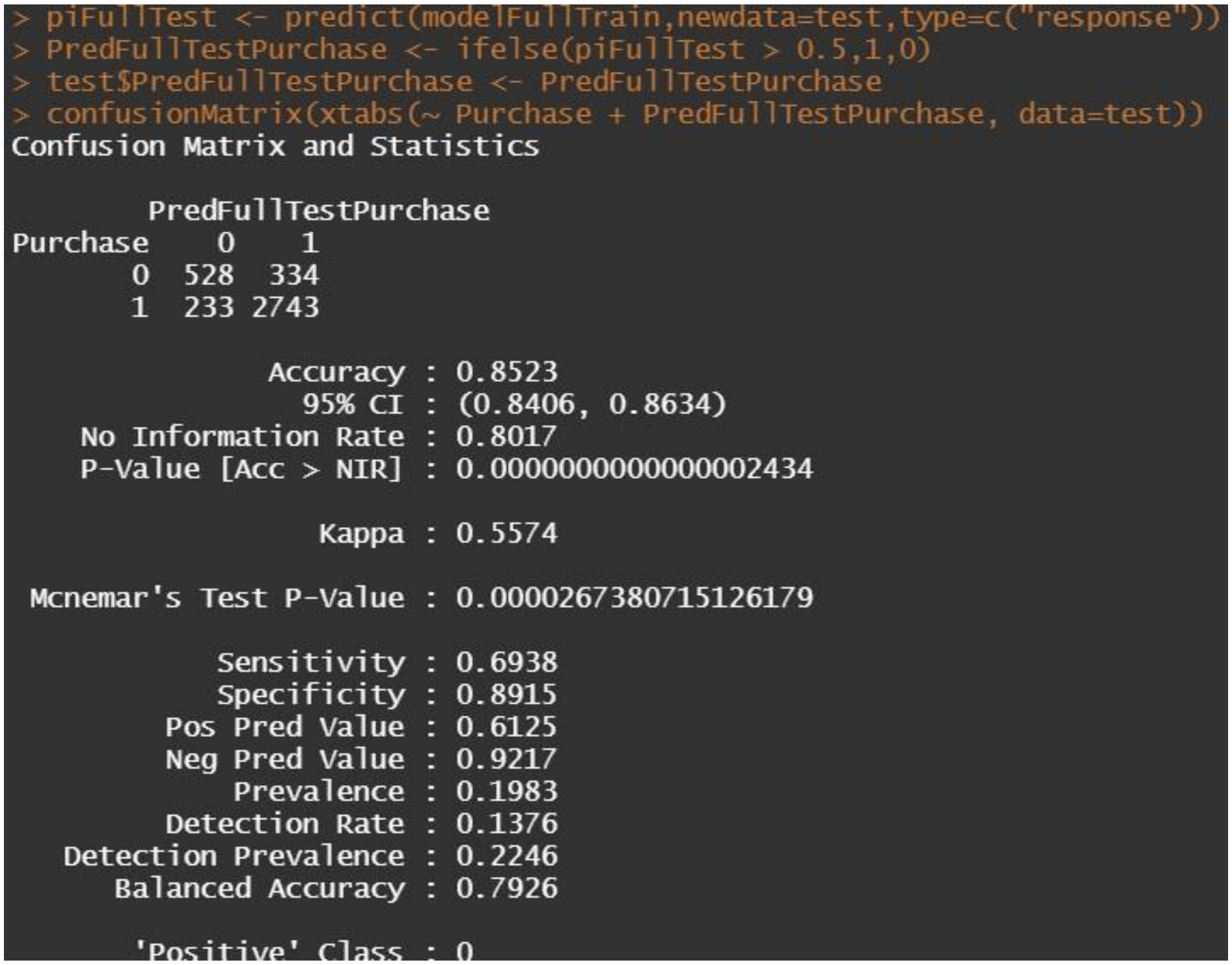
```
> modelFullTrain_FixedAcidity_Pct <- exp(0.0006002) - 1
> round(modelFullTrain_FixedAcidity_Pct * 100,2)
[1] 0.06
> modelFullTrain_VolatileAcidity_Pct <- exp(-0.2034355) - 1
> round(modelFullTrain_VolatileAcidity_Pct * 100,2)
[1] -18.41
> modelFullTrain_CitricAcid_Pct <- exp(0.0354498) - 1
> round(modelFullTrain_CitricAcid_Pct * 100,2)
[1] 3.61
> modelFullTrain_ResidualSugar_Pct <- exp(0.0012958) - 1
> round(modelFullTrain_ResidualSugar_Pct * 100,2)
[1] 0.13
> modelFullTrain_Chlorides_Pct <- exp(-0.1430445) - 1
> round(modelFullTrain_Chlorides_Pct * 100,2)
[1] -13.33
> modelFullTrain_FreeSulfurDioxide_Pct <- exp(0.0006724) - 1
> round(modelFullTrain_FreeSulfurDioxide_Pct * 100,2)
[1] 0.07
> modelFullTrain_TotalSulfurDioxide_Pct <- exp(0.0007685) - 1
> round(modelFullTrain_TotalSulfurDioxide_Pct * 100,2)
[1] 0.08
> modelFullTrain_Density_Pct <- exp(0.2145032) - 1
> round(modelFullTrain_Density_Pct * 100,2)
[1] 23.92
> modelFullTrain_pH_Pct <- exp(-0.1821487) - 1
> round(modelFullTrain_pH_Pct * 100,2)
[1] -16.65
> modelFullTrain_Sulphates_Pct <- exp(-0.0790187) - 1
> round(modelFullTrain_Sulphates_Pct * 100,2)
[1] -7.6
> modelFullTrain_Alcohol_Pct <- exp(-0.0138675) - 1
> round(modelFullTrain_Alcohol_Pct * 100,2)
[1] -1.38
> modelFullTrain_LabelAppeal_Pct <- exp(-0.2306375) - 1
> round(modelFullTrain_LabelAppeal_Pct * 100,2)
[1] -20.6
> modelFullTrain_AcidIndex_Pct <- exp(-0.3948695) - 1
> round(modelFullTrain_AcidIndex_Pct * 100,2)
[1] -32.62
> modelFullTrain_HasSTARS_Pct <- exp(2.9903523) - 1
> round(modelFullTrain_HasSTARS_Pct * 100,2)
[1] 1889.27
>
```

Based on the full model, if a wine has any kind of STARS rating, the odds of it being purchased increases by an overwhelming 1889%. The other chemical properties show that they might influence the purchase outcome by certain percentages as well. For example, an additional unit of Density could increase the odds of a wine being purchased by almost 24%. That being said, in the dataset, the distribution of Density is as follows:



We mentioned earlier, many of these chemical properties have very tall narrow normal distributions with very small standard deviations from the mean. In the case of Density, even though an extra unit of Density can increase the odds of a wine being purchased by almost 24%, there is not much variation in density such that there will unlikely be some wine with a very different value of Density to even affect the purchase outcome. Such is the case with most of the chemical properties in the dataset, and that we've also seen almost 0 correlation between many of these chemical properties to the Purchase outcome. While these chemical properties percentages look to have some influence, they really don't due to their respective normal distributions.

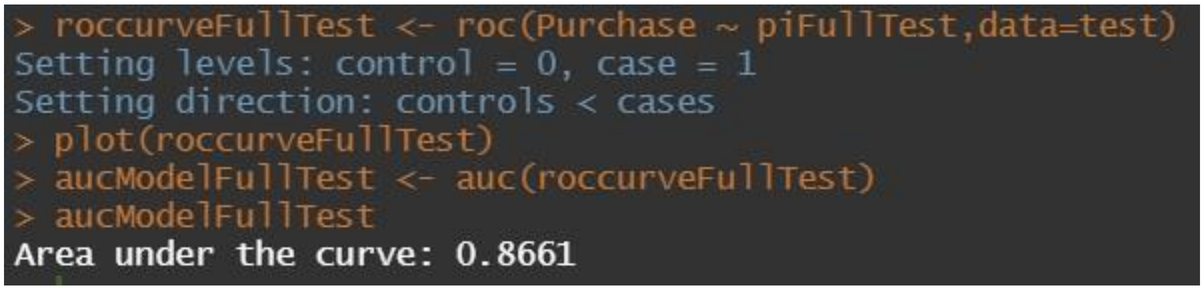
Let us observe how the full model predicts against the test set:



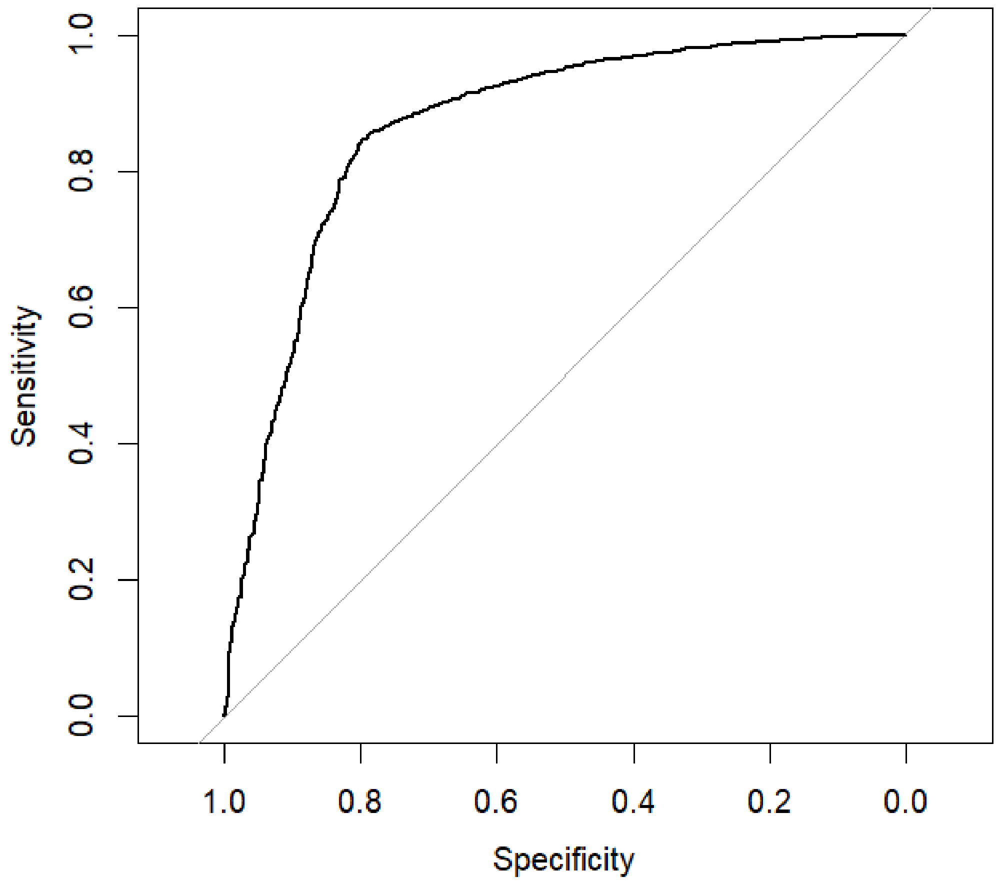
```
> piFullTest <- predict(modelFullTrain,newdata=test,type=c("response"))
> PredFullTestPurchase <- ifelse(piFullTest > 0.5,1,0)
> test$PredFullTestPurchase <- PredFullTestPurchase
> confusionMatrix(xtabs(~ Purchase + PredFullTestPurchase, data=test))
Confusion Matrix and Statistics

        PredFullTestPurchase
Purchase    0    1
       0  528  334
       1  233 2743

               Accuracy : 0.8523
                 95% CI : (0.8406, 0.8634)
    No Information Rate : 0.8017
    P-Value [Acc > NIR] : 0.0000000000000002434

                  Kappa : 0.5574

 Mcnemar's Test P-Value : 0.0000267380715126179

            Sensitivity : 0.6938
            Specificity : 0.8915
         Pos Pred Value : 0.6125
         Neg Pred Value : 0.9217
             Prevalence : 0.1983
         Detection Rate : 0.1376
   Detection Prevalence : 0.2246
      Balanced Accuracy : 0.7926

       'Positive' Class : 0
```

Based on the confusion matrix above, the full model achieved 85.2% accuracy, with 61.3% positive precision (Pos Pred Value), 92.1% negative precision (Neg Pred Value), and 69.4% recall (Sensitivity).

The ROC curve of the full model has an area under the curve (AUC) of 86.6%.

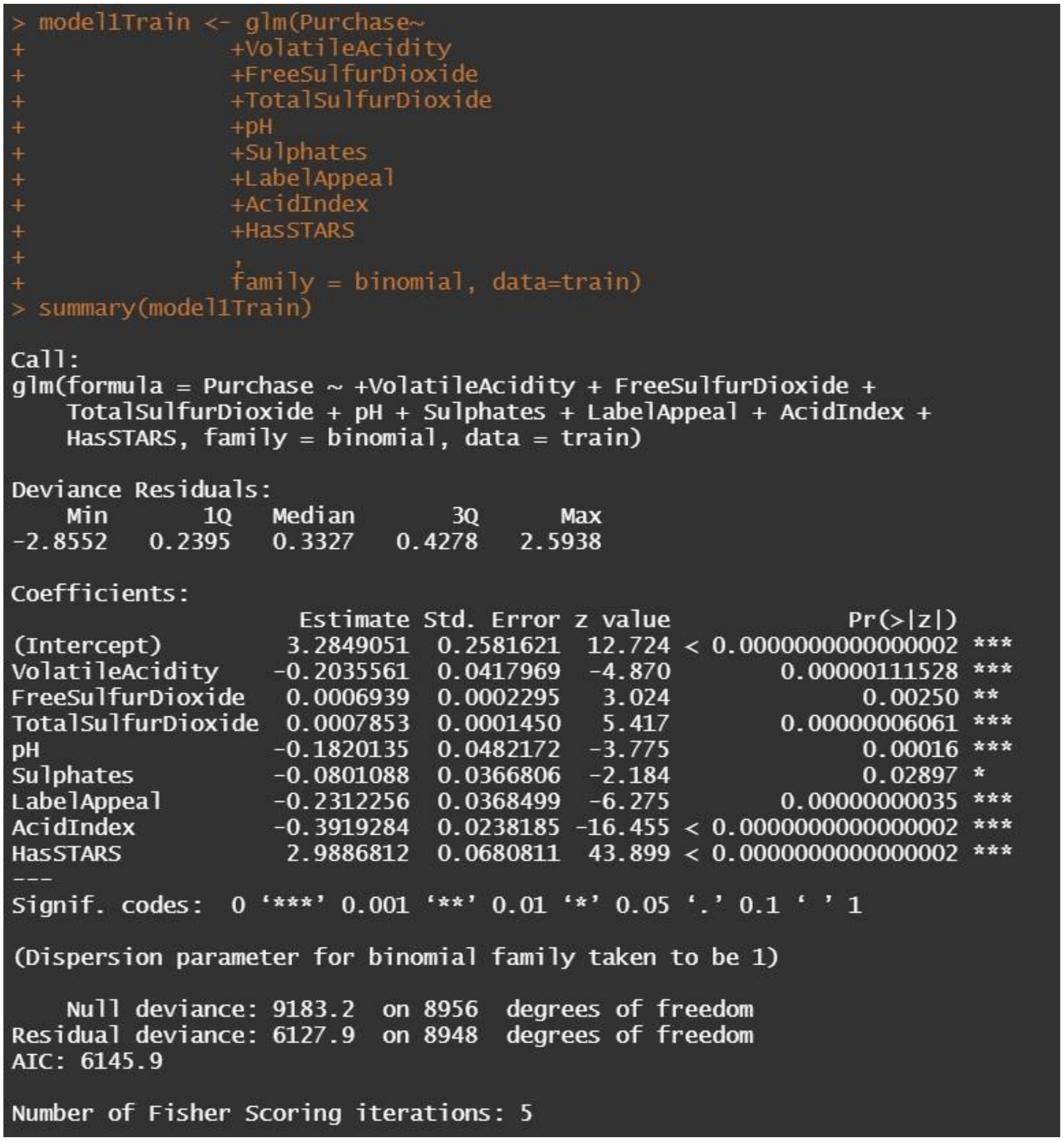


```
> roccurveFullTest <- roc(Purchase ~ piFullTest,data=test)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot(roccurveFullTest)
> aucModelFullTest <- auc(roccurveFullTest)
> aucModelFullTest
Area under the curve: 0.8661
```



Reduced Model: Using Only Statistically Significant Variables

We reduce the model by removing variables from the full model that are not statistically significant: FixedAcidity, CitricAcid, ResidualSugar, Chlorides, Density, and Alcohol. With them removed, we create the following logistic regression model:

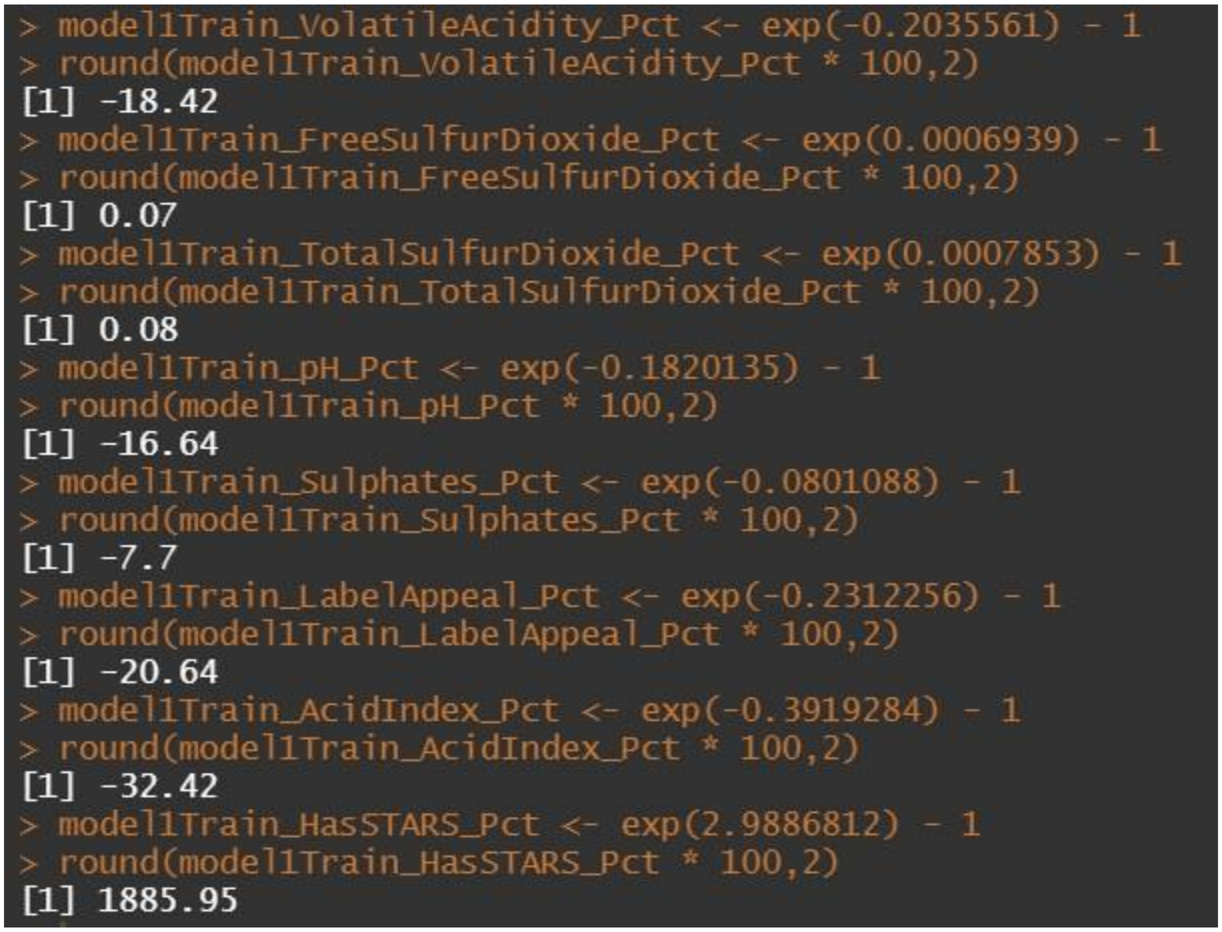


```
> model1Train <- glm(Purchase~
+               +VolatileAcidity
+               +FreeSulfurDioxide
+               +TotalSulfurDioxide
+               +pH
+               +Sulphates
+               +LabelAppeal
+               +AcidIndex
+               +HasSTARS
+               ,
+               family = binomial, data=train)
> summary(model1Train)

Call:
glm(formula = Purchase ~ +VolatileAcidity + FreeSulfurDioxide +
    TotalSulfurDioxide + pH + Sulphates + LabelAppeal + AcidIndex +
    HasSTARS, family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8552   0.2395   0.3327   0.4278   2.5938

Coefficients:
                    Estimate Std. Error z value             Pr(>|z|)
(Intercept)        3.2849051  0.2581621  12.724 < 0.0000000000000002 ***
VolatileAcidity   -0.2035561  0.0417969  -4.870         0.00000111528 ***
FreeSulfurDioxide  0.0006939  0.0002295   3.024              0.00250 **
TotalSulfurDioxide 0.0007853  0.0001450   5.417         0.00000006061 ***
pH                -0.1820135  0.0482172  -3.775              0.00016 ***
Sulphates         -0.0801088  0.0366806  -2.184              0.02897 *
LabelAppeal       -0.2312256  0.0368499  -6.275         0.00000000035 ***
AcidIndex         -0.3919284  0.0238185 -16.455 < 0.0000000000000002 ***
HasSTARS           2.9886812  0.0680811  43.899 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9183.2  on 8956  degrees of freedom
Residual deviance: 6127.9  on 8948  degrees of freedom
AIC: 6145.9

Number of Fisher Scoring iterations: 5
```

The reduced model only includes statistically significant variables.

The coefficients in the summary for the reduced model above are the expected change in log odds of having the outcome per unit change in X.

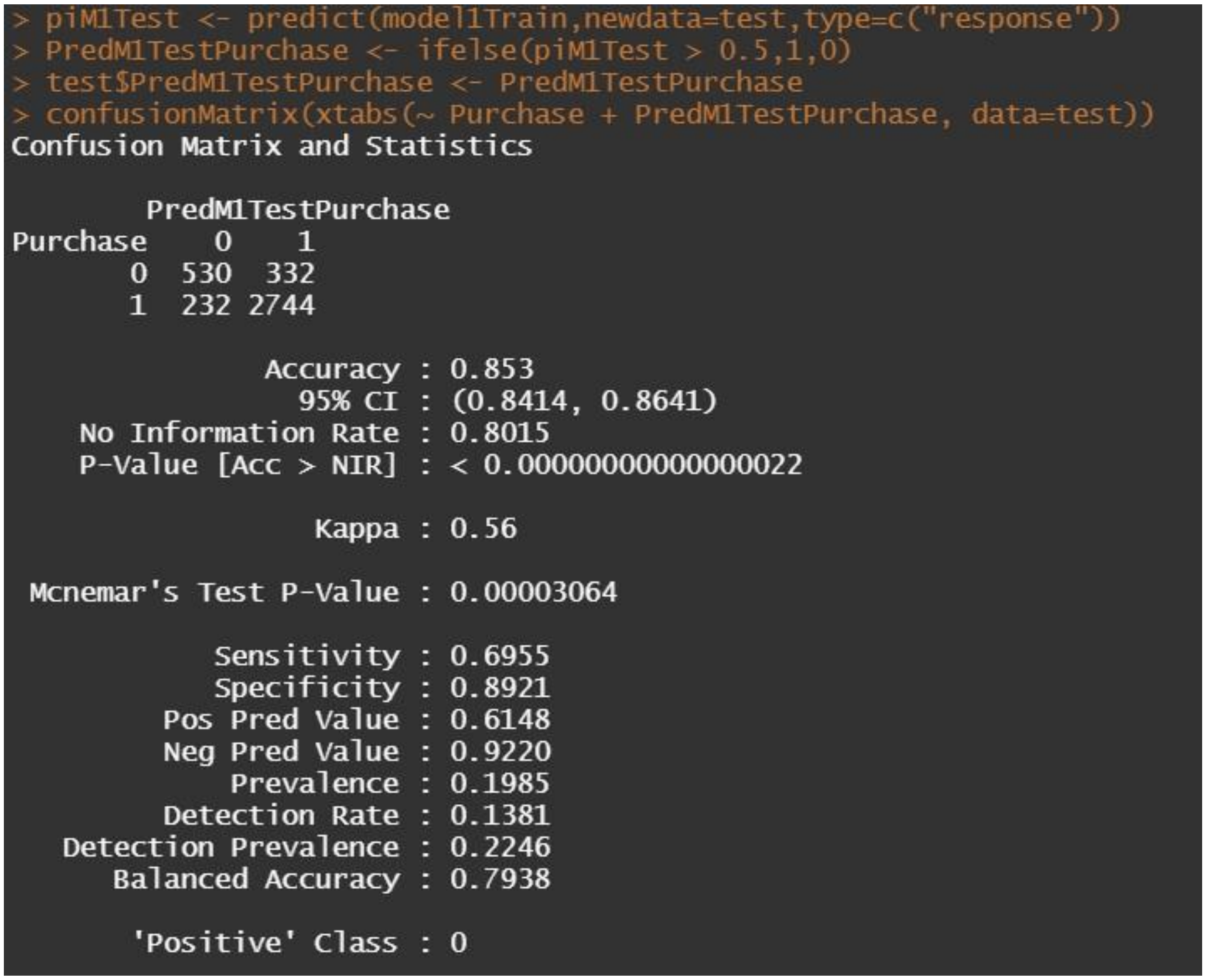
Converting log odds into percentages using exp(coefficient) -1:



```
> model1Train_VolatileAcidity_Pct <- exp(-0.2035561) - 1
> round(model1Train_VolatileAcidity_Pct * 100,2)
[1] -18.42
> model1Train_FreeSulfurDioxide_Pct <- exp(0.0006939) - 1
> round(model1Train_FreeSulfurDioxide_Pct * 100,2)
[1] 0.07
> model1Train_TotalSulfurDioxide_Pct <- exp(0.0007853) - 1
> round(model1Train_TotalSulfurDioxide_Pct * 100,2)
[1] 0.08
> model1Train_pH_Pct <- exp(-0.1820135) - 1
> round(model1Train_pH_Pct * 100,2)
[1] -16.64
> model1Train_Sulphates_Pct <- exp(-0.0801088) - 1
> round(model1Train_Sulphates_Pct * 100,2)
[1] -7.7
> model1Train_LabelAppeal_Pct <- exp(-0.2312256) - 1
> round(model1Train_LabelAppeal_Pct * 100,2)
[1] -20.64
> model1Train_AcidIndex_Pct <- exp(-0.3919284) - 1
> round(model1Train_AcidIndex_Pct * 100,2)
[1] -32.42
> model1Train_HasSTARS_Pct <- exp(2.9886812) - 1
> round(model1Train_HasSTARS_Pct * 100,2)
[1] 1885.95
```

Compared to the full model, the percentages for these variables in the reduced model do not change much.

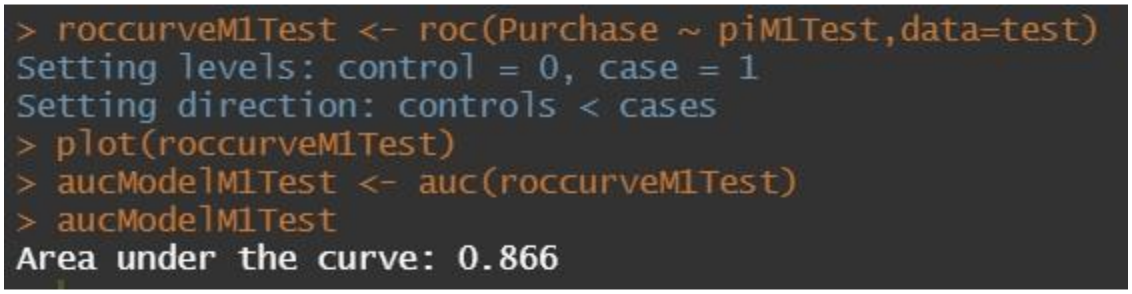
Let us observe how the reduced model predicts against the test set:



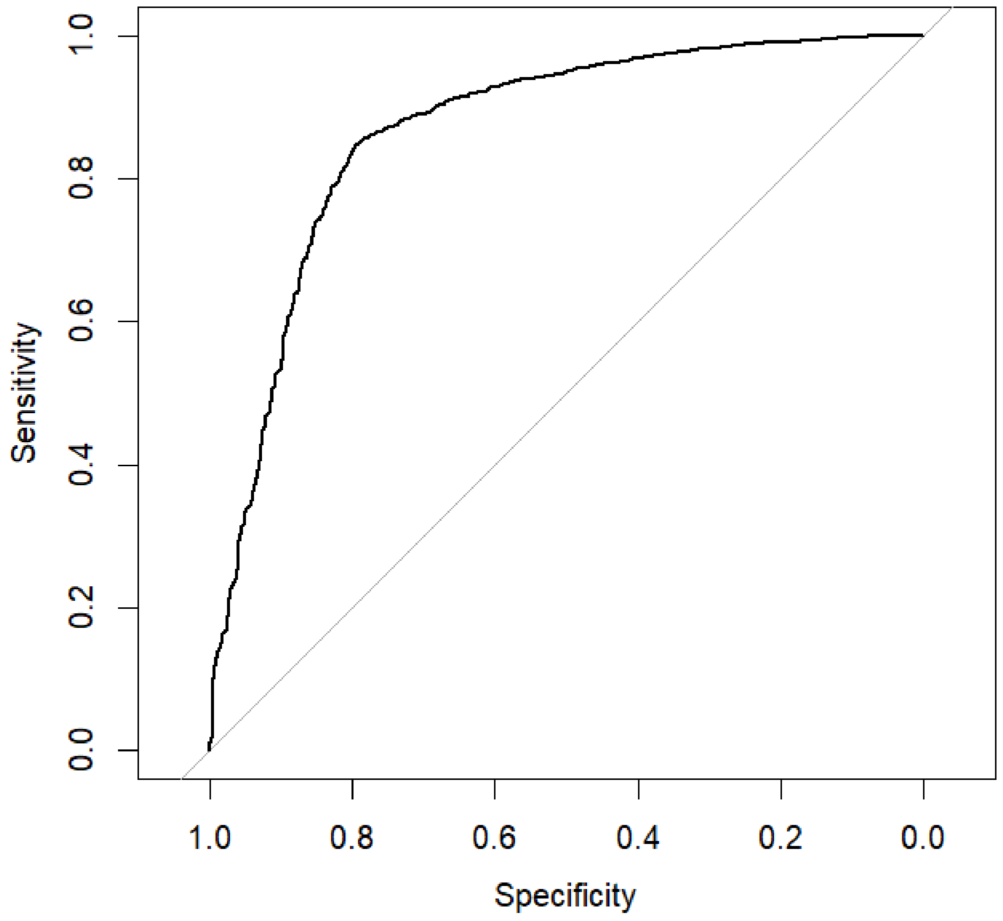
```
> piM1Test <- predict(model1Train,newdata=test,type=c("response"))
> PredM1TestPurchase <- ifelse(piM1Test > 0.5,1,0)
> test$PredM1TestPurchase <- PredM1TestPurchase
> confusionMatrix(xtabs(~ Purchase + PredM1TestPurchase, data=test))
Confusion Matrix and Statistics

        PredM1TestPurchase
Purchase    0    1
       0  530  332
       1  232 2744

               Accuracy : 0.853
                 95% CI : (0.8414, 0.8641)
    No Information Rate : 0.8015
    P-Value [Acc > NIR] : < 0.0000000000000022

                  Kappa : 0.56

 Mcnemar's Test P-Value : 0.00003064

            Sensitivity : 0.6955
            Specificity : 0.8921
         Pos Pred Value : 0.6148
         Neg Pred Value : 0.9220
             Prevalence : 0.1985
         Detection Rate : 0.1381
   Detection Prevalence : 0.2246
      Balanced Accuracy : 0.7938

       'Positive' Class : 0
```

Based on the confusion matrix above, the reduced model achieved 85.3% accuracy, with 61.5% positive precision (Pos Pred Value), 92.5% negative precision (Neg Pred Value), and 69.6% recall (Sensitivity), a slight improvement over the full model.

The ROC curve of the full model has an area under the curve (AUC) of 86.6%.

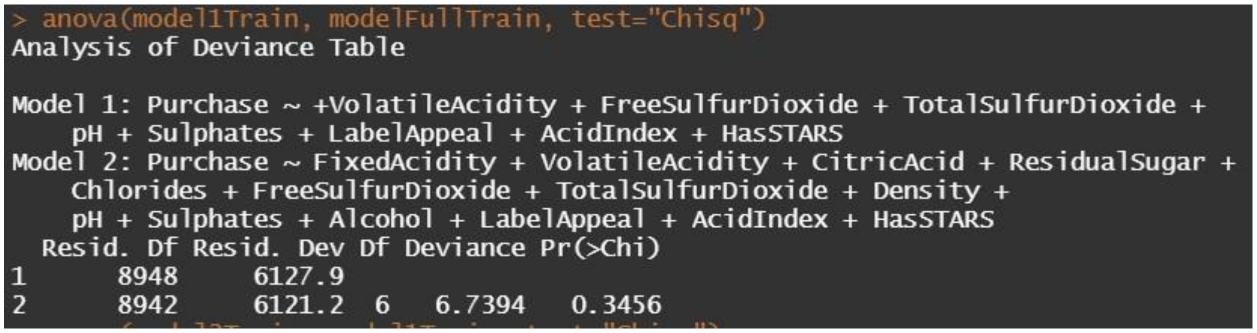


```
> roccurveM1Test <- roc(Purchase ~ piM1Test,data=test)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot(roccurveM1Test)
> aucModelM1Test <- auc(roccurveM1Test)
> aucModelM1Test
Area under the curve: 0.866
```



Based on the observations above, the reduced model slightly seems to perform better than the full model using the test set.

We can perform an ANOVA test to compare the full model to the reduced model:

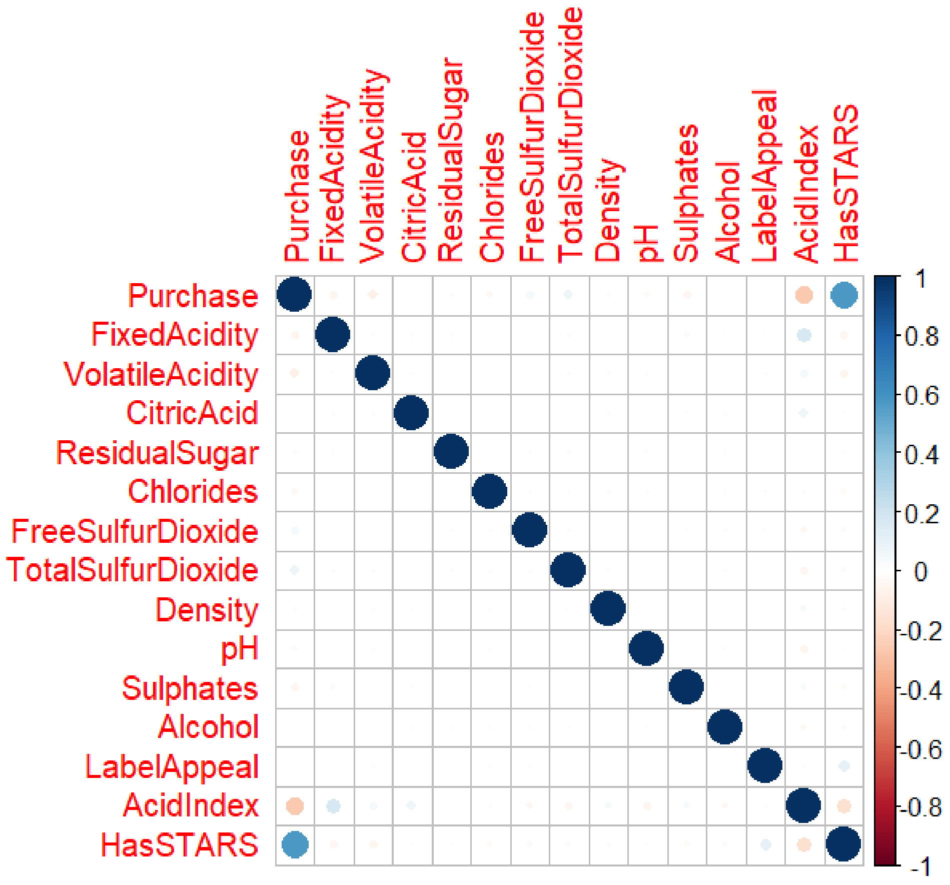


```
> anova(model1Train, modelFullTrain, test="Chisq")
Analysis of Deviance Table

Model 1: Purchase ~ +VolatileAcidity + FreeSulfurDioxide + TotalSulfurDioxide +
    pH + Sulphates + LabelAppeal + AcidIndex + HasSTARS
Model 2: Purchase ~ FixedAcidity + VolatileAcidity + CitricAcid + ResidualSugar +
    Chlorides + FreeSulfurDioxide + TotalSulfurDioxide + Density +
    pH + Sulphates + Alcohol + LabelAppeal + AcidIndex + HasSTARS
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      8948     6127.9
2      8942     6121.2  6   6.7394   0.3456
```

The high p-value also suggests using the simplified reduced model over the full model.

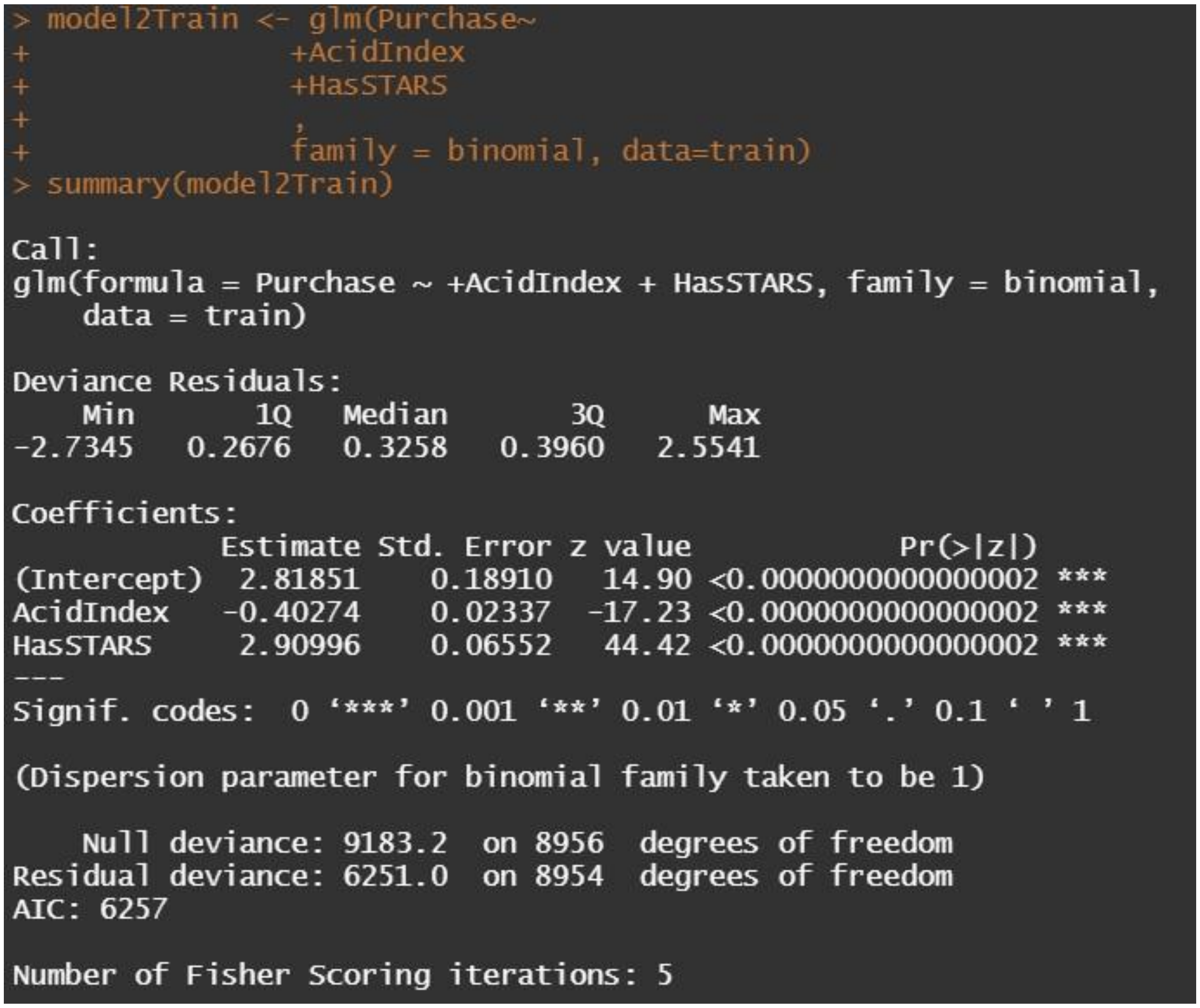
Much More Reduced Model: Using Only Highly-Correlated Variables

Recall the correlation matrix:



Based on the matrix above, with most chemical properties against Purchase seem to have almost no correlation to the Purchase outcome, we see some correlation with AcidIndex and HasSTARS variables.

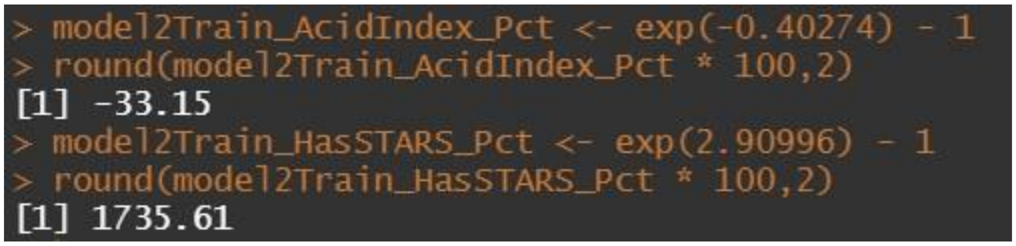
Let us create a much more reduced model using only those variables:



```
> model2Train <- glm(Purchase~
+               +AcidIndex
+               +HasSTARS
+               ,
+               family = binomial, data=train)
> summary(model2Train)

Call:
glm(formula = Purchase ~ +AcidIndex + HasSTARS, family = binomial,
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7345   0.2676   0.3258   0.3960   2.5541

Coefficients:
            Estimate Std. Error z value            Pr(>|z|)
(Intercept)  2.81851    0.18910   14.90 <0.0000000000000002 ***
AcidIndex   -0.40274    0.02337  -17.23 <0.0000000000000002 ***
HasSTARS     2.90996    0.06552   44.42 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9183.2  on 8956  degrees of freedom
Residual deviance: 6251.0  on 8954  degrees of freedom
AIC: 6257

Number of Fisher Scoring iterations: 5
```

The coefficients in the summary for the reduced model above are the expected change in log odds of having the outcome per unit change in X.

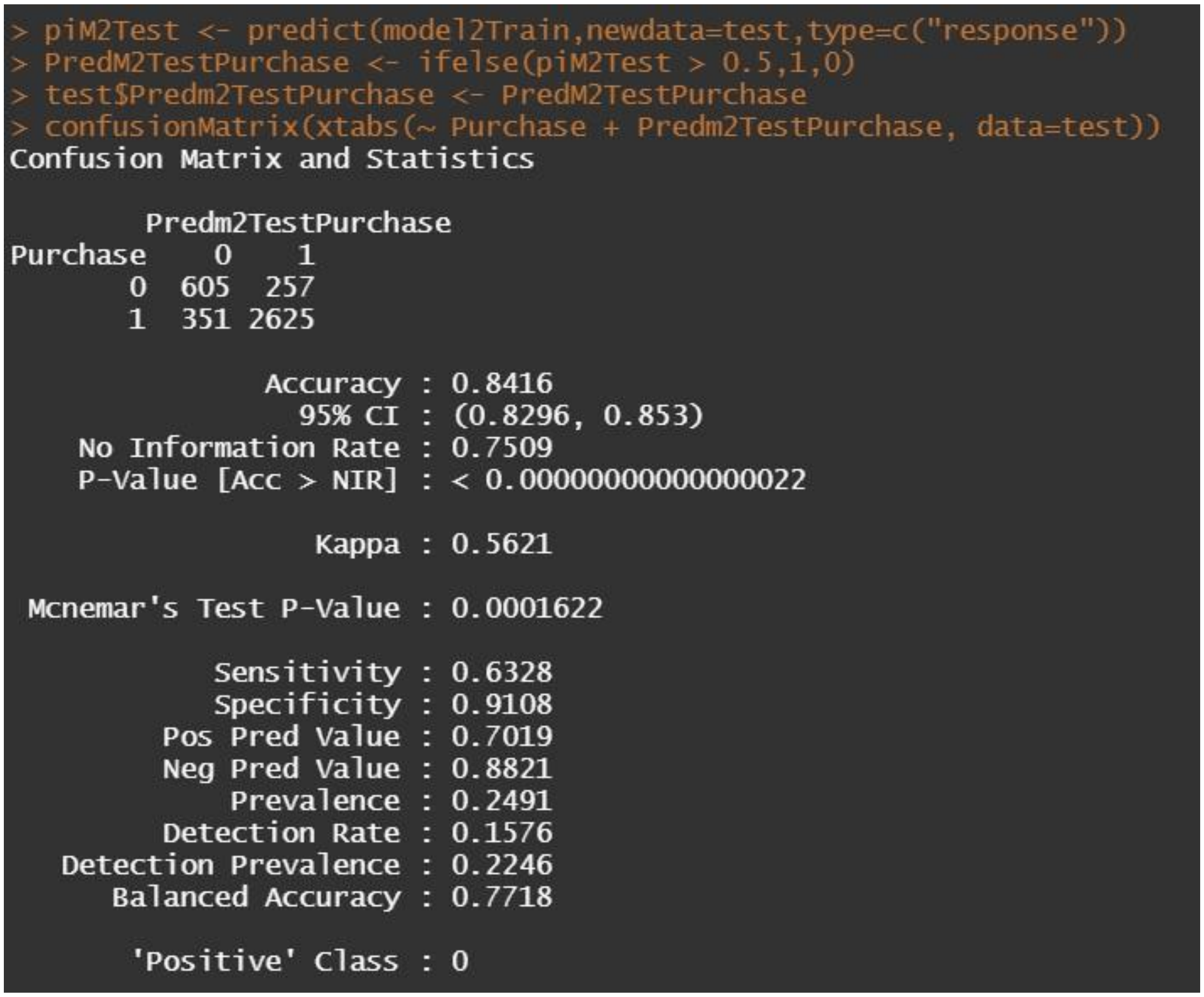
Converting log odds into percentages using exp(coefficient) - 1:



```
> model2Train_AcidIndex_Pct <- exp(-0.40274) - 1
> round(model2Train_AcidIndex_Pct * 100,2)
[1] -33.15
> model2Train_HasSTARS_Pct <- exp(2.90996) - 1
> round(model2Train_HasSTARS_Pct * 100,2)
[1] 1735.61
```

While the HasSTARTS odds percentage has somewhat dropped to almost 1736% it is still an overwhelming influence on the Purchase outcome.

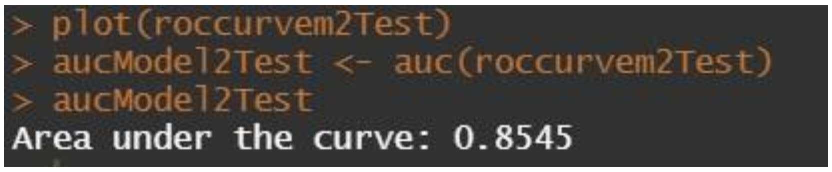
Let us observe how the much-reduced model predicts against the test set:

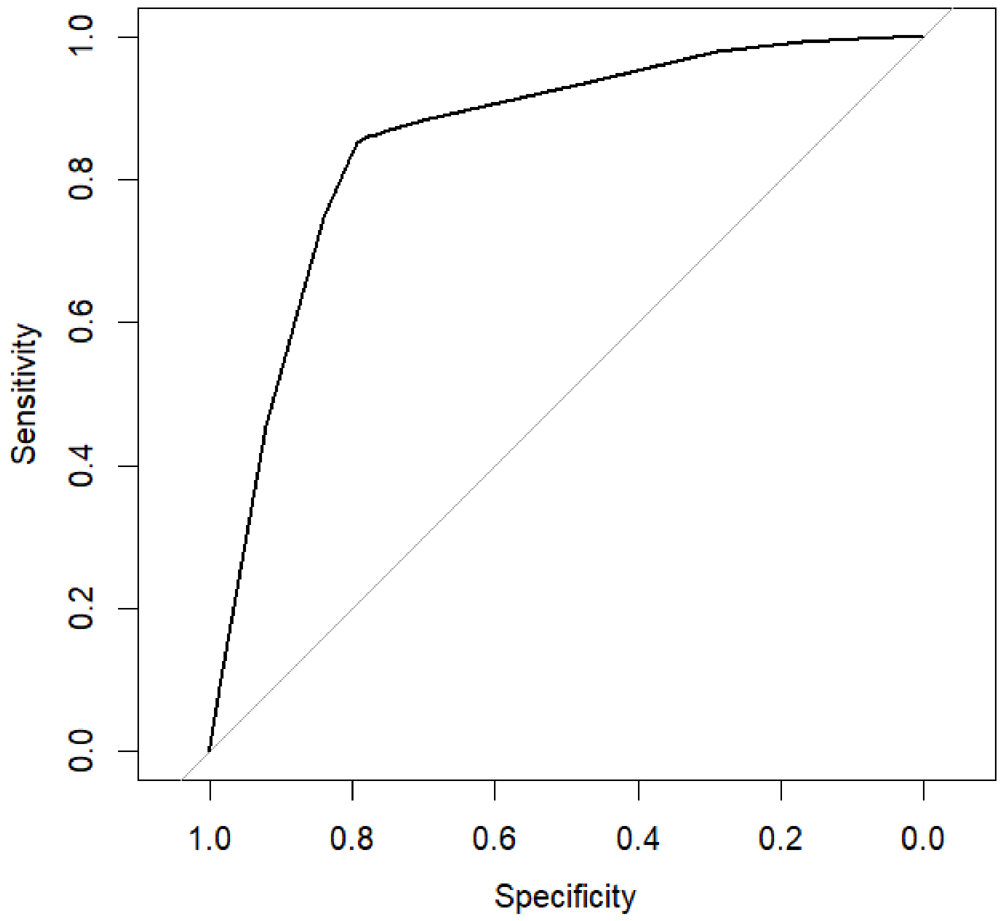


```
> piM2Test <- predict(model2Train,newdata=test,type=c("response"))
> PredM2TestPurchase <- ifelse(piM2Test > 0.5,1,0)
> test$Predm2TestPurchase <- PredM2TestPurchase
> confusionMatrix(xtabs(~ Purchase + Predm2TestPurchase, data=test))
Confusion Matrix and Statistics

        Predm2TestPurchase
Purchase    0    1
       0  605  257
       1  351 2625

               Accuracy : 0.8416
                 95% CI : (0.8296, 0.853)
    No Information Rate : 0.7509
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5621

 Mcnemar's Test P-Value : 0.0001622

            Sensitivity : 0.6328
            Specificity : 0.9108
         Pos Pred Value : 0.7019
         Neg Pred Value : 0.8821
             Prevalence : 0.2491
         Detection Rate : 0.1576
   Detection Prevalence : 0.2246
      Balanced Accuracy : 0.7718

       'Positive' Class : 0
```

Based on the confusion matrix above, this model achieved 84.2% accuracy, with 70.2% positive precision (Pos Pred Value), 88.2% negative precision (Neg Pred Value), and 63.2% recall (Sensitivity), which doesn’t perform as well overall compared to the other reduced model.

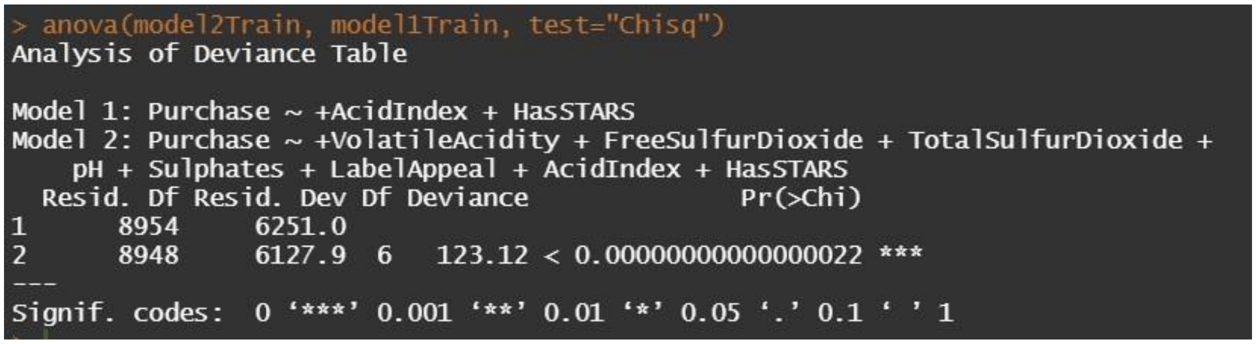
The ROC curve of the full model has an area under the curve (AUC) of 85.5%, again not performing as well as the other reduced model.


```
> plot(roccurvem2Test)
> aucModel2Test <- auc(roccurvem2Test)
> aucModel2Test
Area under the curve: 0.8545
```



After running tests with three models, I would choose the second model as my final model for purchasing a wine to stock, as it performs slightly better than both the full model and much-reduced model. While the much-reduced model is very simple to maintain and is close to the performance of the second model, the second model is not too complex of a model and would want other chemical properties of a wine to be evaluated if a wine does not have a rating.

We can perform an ANOVA test between the two reduced models:



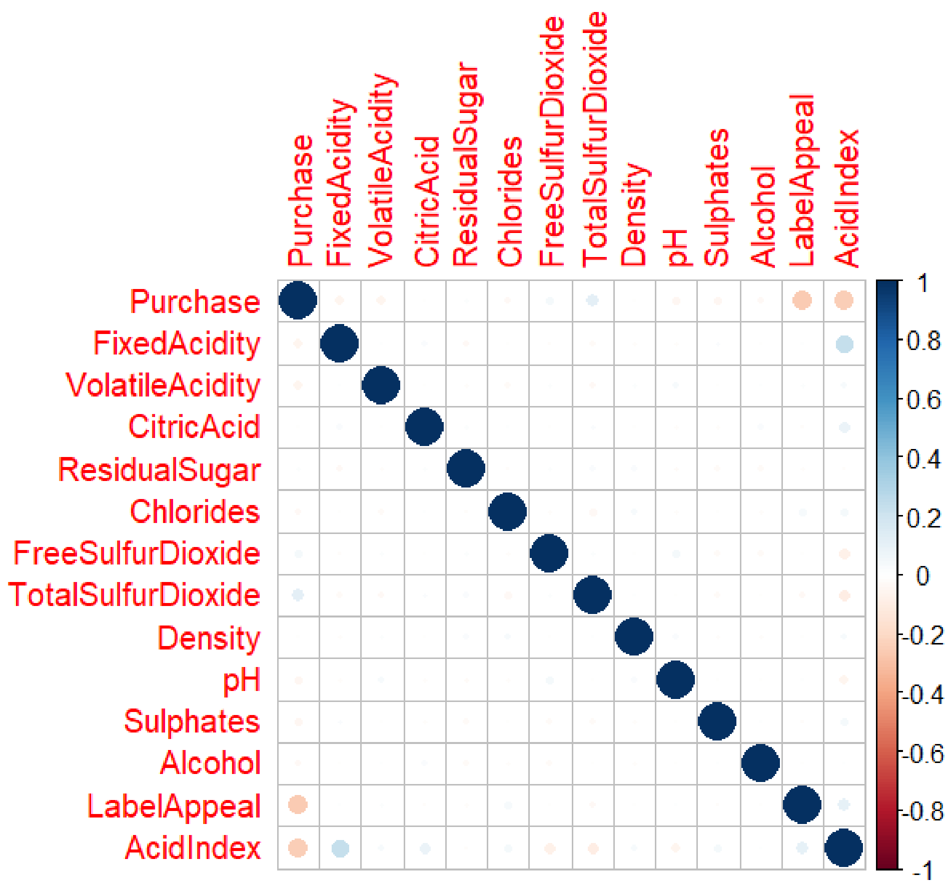
```
> anova(model2Train, model1Train, test="Chisq")
Analysis of Deviance Table

Model 1: Purchase ~ +AcidIndex + HasSTARS
Model 2: Purchase ~ +VolatileAcidity + FreeSulfurDioxide + TotalSulfurDioxide +
    pH + Sulphates + LabelAppeal + AcidIndex + HasSTARS
  Resid. Df Resid. Dev Df Deviance            Pr(>Chi)
1      8954     6251.0
2      8948     6127.9  6   123.12 < 0.00000000000000022 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The low p-value for the larger reduced model suggests to use that model over the much-reduced model.

3. **What conclusions do you draw from having conducted this analysis? What did you learn about the wine world through your modeling endeavor? What actions can you recommend to anyone involved in this field? How did your perspective on modeling change? Discuss anything else you wish to discuss.**

   We find that rated wines have a huge influence over unrated wines. Out in the real world, from James Sucking to Wine Enthusiast, any rating, even low-rated ones, can make the difference for a store or restaurant to stock, as it gives regular customers some digestible information to use for selecting a wine at their price point. Not everyone has sommelier-level knowledge or are chemists, so people put trust into these ratings to help make sure they are getting value for the money spent on them.

   That being said, there could be an opportunity to model unrated wines. After removing rated wines from the dataset, we have the following correlation matrix:



For unrated wines, we see a negative correlation with AcidIndex and LabelAppeal to Purchase, as well as see some positive correlation with TotalSulfurDioxide. In the case of LabelAppeal, it would be an interesting study to figure out why it has a negative correlation to Purchase.

While I might have had my own personal thoughts about what factors might play into a wine purchase, going through this thorough exercise helps show why ratings are such a big influence on wine. It would have been nice to find some ground-breaking revelation from it, but this process is very beneficial which can be carried over to other kinds of logistic regression modelling in the future.