AFS Wine Quality Report

Ray Anthony Roderos

April 8, 2017

Contents

[1. Diagnostics 1](#_Toc479450548)

[2. Benchmarks 3](#_Toc479450549)

[3. Feature Engineering 4](#_Toc479450550)

[4. Model Selection, Optimization, Data Splitting, Training, Testing and Prediction 5](#_Toc479450551)

[5. Test for Robustness 6](#_Toc479450552)

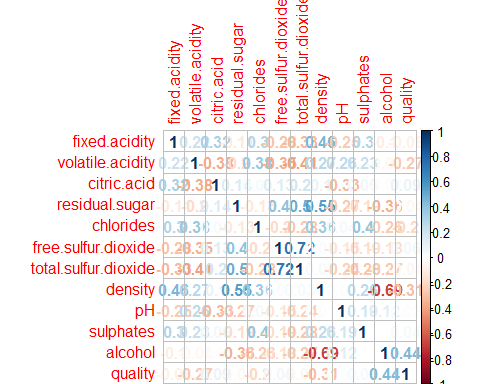
[6. Conclusion 8](#_Toc479450553)

# Diagnostics

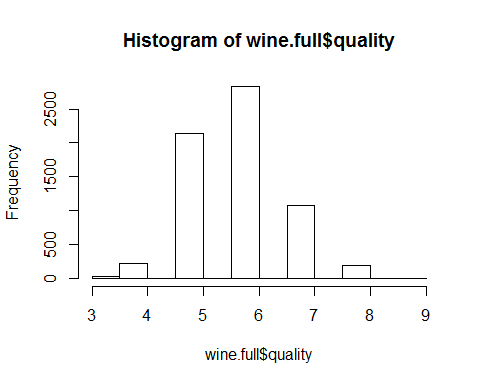
I first started with doing basic diagnostics, mainly correlations and summary statistics of each variable. My notable findings are the following:

1. The distribution of quality follows a normal distribution centered around 5.
2. There is some significant correlated pairs for alcohol vs. density, fixed.acidity vs. density, residual.sugar vs total.sulfur.dioxide, residual.sugar vs. density, residual.sugar vs. alcohol, chlorides vs. density, chlorides vs. sulphates
3. And there are high correlations with the dependent variable, particularly quality vs. alcohol, quality vs. density quality vs. volatile.acidity

corrplot(cor(wine.full),method="number")



hist(wine.full$quality)



As part of the diagnostics, I did a simple regression to quickly gauge which ones were statistically important and determined the r-squared.

From the summary results, I can see that citric.acid and chlorides were not statistically significant. RSE is at 0.7 and Rquared is 0.28 which means that the independent variables were only explaining a small part of the wine quality giving me an idea that the prediction may be hard.

diagnostic.lm <- lm(wine.full.orig$quality~.,data=wine.full.orig)  
summary(diagnostic.lm)

##   
## Call:  
## lm(formula = wine.full.orig$quality ~ ., data = wine.full.orig)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.7660 -0.4613 -0.0426 0.4668 4.9827   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.648e+01 1.201e+01 4.701 2.64e-06 \*\*\*  
## fixed.acidity 7.163e-02 1.573e-02 4.554 5.35e-06 \*\*\*  
## volatile.acidity -1.321e+00 7.815e-02 -16.902 < 2e-16 \*\*\*  
## citric.acid -1.042e-01 8.041e-02 -1.296 0.195   
## residual.sugar 4.467e-02 5.207e-03 8.578 < 2e-16 \*\*\*  
## chlorides -4.852e-01 3.360e-01 -1.444 0.149   
## free.sulfur.dioxide 5.999e-03 7.587e-04 7.907 3.08e-15 \*\*\*  
## total.sulfur.dioxide -2.485e-03 2.794e-04 -8.895 < 2e-16 \*\*\*  
## density -5.780e+01 1.226e+01 -4.715 2.47e-06 \*\*\*  
## pH 4.632e-01 9.127e-02 5.075 3.99e-07 \*\*\*  
## sulphates 7.681e-01 7.688e-02 9.990 < 2e-16 \*\*\*  
## alcohol 2.656e-01 1.690e-02 15.723 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7427 on 6485 degrees of freedom  
## Multiple R-squared: 0.2899, Adjusted R-squared: 0.2887   
## F-statistic: 240.7 on 11 and 6485 DF, p-value: < 2.2e-16

# Benchmarks

As a benchmark, I did a PLS-gaussian regression and prediction with default values using original features and checked their RMSE and MAE. The figures here would serve as my baseline.

ptm <- proc.time() #start time  
  
#benchmark  
model1 <- plsRglm(train.y,train.x[,1:11],3, modele="pls-glm-gaussian")

#Making the preidction  
  
pred\_y1 <-predict(model1,test.x[,1:11])  
  
#Checking MSE  
error1 <- (pred\_y1-test.y)  
  
rmse <- function(error)  
{  
 sqrt(mean(error^2))  
}  
  
mae <- function(error)  
{  
 mean(abs(error))  
}  
  
accuracy <- function(error){  
 1-mean(abs(error))  
}  
  
#Checking RMSE and MAE  
error1 <- (test.y-pred\_y1)  
mae(error1)

## [1] 0.5659358

rmse(error1)

## [1] 0.7474929

accuracy(error1)

## [1] 0.4340642

proc.time() - ptm

## user system elapsed   
## 2.53 0.00 3.69

The benchmarks are pretty low. I think it is best to add more features and change the number of components. Fortunately, the PLS model works well with datasets with a lot of features.

# Feature Engineering

The following are what I did:

1. To start on engineering the features, I determined that I should take into account interaction effects of various features. Based from internet research, I created new features that are combinations of the original features that have similar properties. Namely the following:
2. Acids - fixed acidity, volatile acidity and citric acidity
3. Dioxides - free sulfur dioxides and total sulfur dioxides.
4. Aside from these, I also made combinations of features which highlight the ones that have the most correlation with wine quality. Those are volatile acidity, chlorides, density and alcohol.
5. I also added log transformations of the original features to take into account relative changes.
6. I added wine type as a category.
7. And I rescaled the wine quality from 3-8 to 1-6 for the algorithm to read it better.

I tested the model without the features that were not statistically significant (citric acid and chlorides) and the results didn’t improve significantly. I decided to retain them.

I chose not to remove outliers in the observations to preserve the original data and an initital run without the outliers did not show much improvement.

I also chose not to rescale the predictors since the function will do it for me.

# Model Selection, Optimization, Data Splitting, Training, Testing and Prediction

From the original 11 predictors there are now 54, expanding the feature space which is my initial intention.

For the PLS model, I chose the gaussian family since wine quality is distributed normally and the it showed the highest accuracy. I also took into account the computational cost, the original PLS algorithm took too long to execute (700+ secs) while the others not only took too long to compute but also did worse.

For the number of components in the PLS model, I made a for loop function executing the model from 1 to 54 components and checked the summary statistics of each and found k= 14 with the lowest RMSE and with the biggest jump in Rsquared.

For data splitting, I ran a function that randomly split the data into 80% training and 20% test for white wine and red wine separetely then combined them after to have a proportional share of white and red wines for both sets.

I then trained the model for 80% and tested on the 20% and checked their RMSE,MAE and Accuracy.

##-----------------------------------------------------------------------------##  
#Checking the Metrics  
  
#Creat functions to compute RMSE, MAE and Accuracy  
error <- (pred\_y-test.y)  
  
rmse <- function(error)  
{  
 sqrt(mean(error^2))  
}  
  
mae <- function(error)  
{  
 mean(abs(error))  
}  
  
accuracy <- function(error){  
 1-mean(abs(error))  
}  
  
#Compute MAE, RMSE and Accuracy of Prediction  
mae(error)

## [1] 0.5568502

rmse(error)

## [1] 0.7359359

accuracy(error)

## [1] 0.4431498

As you can see, I was able to improve the MSE and RMSE to .55 and .73 and accuracy to 0.44, still low but an improvement over the benchmark.

I then analyzed the results of the prediction. The prediction showed that it had a narrower spectrum of answers (such as a wine quality of 5 will give predicted answers between 4 and6) but it gave negative values.

I also rounded the predicted answers and checked if it matches the actual wine quality and also created a confusion matrix.

acc <- round(pred\_y) == test.y  
length(acc[acc==TRUE])/length(acc)

## [1] 0.5612953

table(test.y,round(pred\_y))

##   
## test.y -2 1 2 3 4 5  
## 1 1 0 0 3 3 0  
## 2 0 0 2 27 10 1  
## 3 0 0 2 228 198 1  
## 4 0 1 1 87 431 46  
## 5 0 0 0 6 145 67  
## 6 0 0 0 0 23 12  
## 9 0 0 0 0 1 1

The model, once predicted values have been rounded, had 0.56 accuracy. Not as good but much better than the benchmark and the previous predicted wine quality values. I can see that many of the misclassified values are on 3 and 5.

# Test for Robustness

With no further changes on the feature engineering, rescaling, data splitting, PLS model and optimal number of components, I then tested the robustness of the model by running it in a for loop with different seeds.

table <- matrix(,nrow=5,ncol=4)  
colnames(table) <- c("MAE","RMSE","Accuracy","Computation Time")  
  
  
for (i in 1:5){  
   
 #Start Computation Time  
ptm <- proc.time()  
  
set.seed(i)  
  
##-----------------------------------------------------------------------------##  
#Creating the training and test set with different seeds  
  
index.white <- createDataPartition(white.full$quality, p = .8,   
 list = FALSE,   
 times = 1)  
  
train.white <- white.full[index.white,]  
test.white <- white.full[-index.white,]  
  
index.red <- createDataPartition(red.full$quality, p = .8,   
 list = FALSE,   
 times = 1)  
  
train.red <- red.full[index.red,]  
test.red <- red.full[-index.red,]  
  
train <- rbind(train.white,train.red)  
test <- rbind(test.white,test.red)  
  
  
train.y <- train[,12]  
train.x <- train[,-12]  
test.y <- test[,12]  
test.x <- test[,-12]  
  
##-----------------------------------------------------------------------------##  
   
 #Training the model  
model <- plsRglm(train.y,train.x,14, modele="pls-glm-gaussian")  
  
  
#Making the preidction  
  
pred\_y <-predict(model,test.x)  
  
#Checking MSE  
error <- (pred\_y-test.y)  
  
rmse <- function(error)  
{  
 sqrt(mean(error^2))  
}  
  
mae <- function(error)  
{  
 mean(abs(error))  
}  
  
accuracy <- function(error){  
 1-mean(abs(error))  
}  
  
##-----------------------------------------------------------------------------##  
  
table[i,1] <- mae(error)  
table[i,2] <- rmse(error)  
table[i,3] <- accuracy(error)  
  
  
##-----------------------------------------------------------------------------##  
  
time <-proc.time() - ptm #check the computation time  
time <- round(as.numeric(time[3]),2)  
table[i,4] <- time  
  
}

table

## MAE RMSE Accuracy Computation Time  
## [1,] 0.5627468 0.7238797 0.4372532 47.94  
## [2,] 0.5454933 0.7134348 0.4545067 49.31  
## [3,] 0.5568502 0.7359359 0.4431498 49.86  
## [4,] 0.5773156 0.7421463 0.4226844 46.86  
## [5,] 0.5444345 0.7086755 0.4555655 49.42

Looking at the table, looks like there is little variation, my model proves robust.

# Conclusion

In conclusion, I would say that my PLS-Gaussian model is not the best model for this dataset given the pretty low accuracy. The Rsquared is low since we’re predicting something as subjective as wine quality which is determined by preferences of experts. I also understand that PLS works best with lots of features with many of them highly collinear or with too few observations. That is why I expanded the feature space as much as I can but for every feature I added, the added accuracy was marginal. I will recommend trying using other linear regression techniques with L1 and L2 regularization or getting more data on more features of the wine.