Wine Quality Prediction

[Erica Thompson-Becker]

[TMU:CMTH 642]

The dataset is related to the white variants of the Portuguese “Vinho Verde” wine. Taken from the UCI Machine Learning Repository, more information can be found at https://archive.ics.uci.edu/ml/datasets/Wine+Quality (https://archive.ics.uci.edu/ml/datasets/Wine+Quality).

The goal of this project is to predict if the wine will pass or fail the quality test based on its chemical attributes. This will be achieved by using the logistic regression algorithm. Two models will be compared in this analysis, one using an unbalanced training set and one with a balanced training set.

Libraries

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.2
```

```
## corrplot 0.92 loaded
```

Data Preparation and Data Exploration

```
#Load data set
URL <- "http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv"
#read csv file
winedata <- read.csv(file = URL, sep = ";", header = TRUE, na.strings = c("","NA"))
#check first 6 rows of data
head(winedata)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
## 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   quality
## 1       6
## 2       6
## 3       6
## 4       6
## 5       6
## 6       6
```

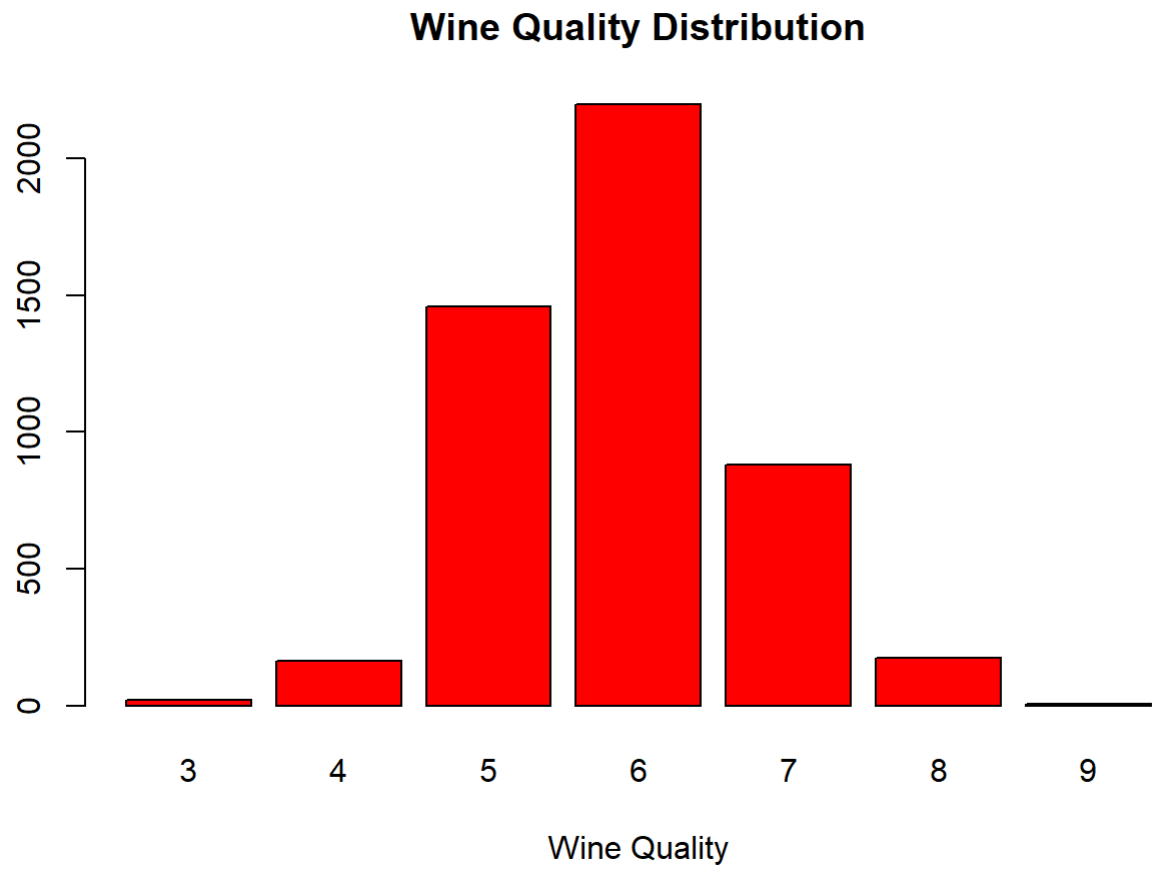
```
#check for missing values (there are no missing values)
sum(is.na(winedata))
```

```
## [1] 0
```

```
#check data types
str(winedata)
```

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

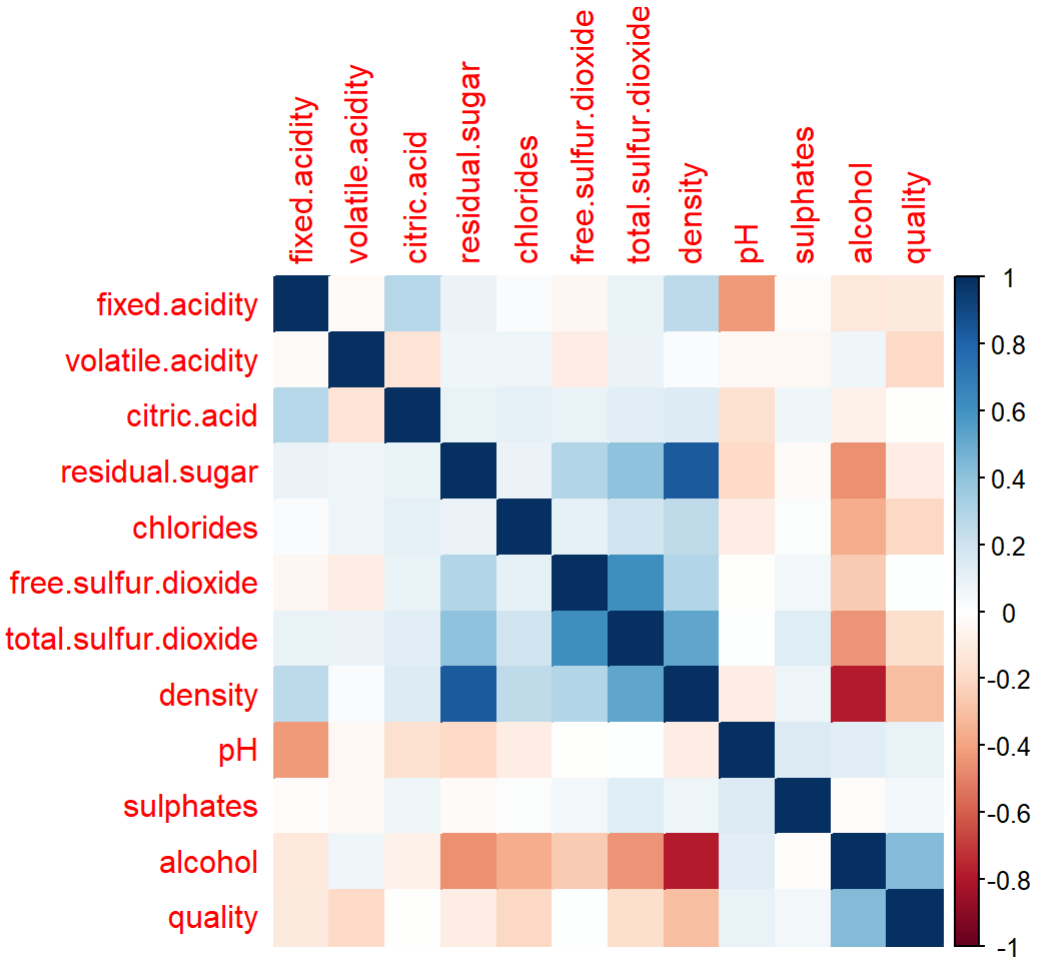
```
#check the frequency distribution of wine quality
#print the counts of the values
tab1 <- table(winedata$quality)
barplot(tab1, col = 'red', main = 'Wine Quality Distribution', xlab = 'Wine Quality')
```




```
#look at the correlation between the attributes and quality

#create correlation matrix
corr_matrix <- cor(winedata)

#visualize correlation matrix as a colour
corrplot(corr_matrix, method = 'color')
```




```
#ph, sulfates, and alcohol have a positive correlation with quality
#all others have a negative correlation with quality

#density and alcohol have a strong negative correlation
#density and residual sugar have strong positive correlation
```

This analysis uses binomial logistic regression, currently the data is separated into quality levels from 0-10. To preform the analysis the data must first be transformed into two quality levels; pass or fail. Levels 0-5 are assigned to fail, levels 6-10 are assigned to pass. Let 1 represent pass and 0 represent fail.

```
#simply done by changing values greater than 5 to be 'pass' and 5 or lower to be fail
winedata$quality <- ifelse(winedata$quality > 5, 1,0)
```

Normalizing the numeric attributes helps to ensure the scale of the values are similar, so that there is no loss of importance due to one attribute being on a higher scale. Here Min-Max Scale normalization is used.

```
#define the normalization function
normalize <- function(x){
  (x - min(x))/(max(x) - min(x))
}

#normalize the numeric attributes and create a new dataframe with the normalized data set
norm_data <- cbind(as.data.frame(lapply(winedata[1:11], normalize)),quality = winedata$quality)

head(norm_data)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1     0.3076923        0.1862745   0.2168675     0.30828221 0.1068249
## 2     0.2403846        0.2156863   0.2048193     0.01533742 0.1186944
## 3     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
## 4     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
## 5     0.3269231        0.1470588   0.1927711     0.12116564 0.1454006
## 6     0.4134615        0.1960784   0.2409639     0.09662577 0.1216617
##   free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
## 1          0.14982578            0.3735499 0.2677848 0.2545455 0.2674419
## 2          0.04181185            0.2853828 0.1328321 0.5272727 0.3139535
## 3          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
## 4          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
## 5          0.15679443            0.4106729 0.1636784 0.4272727 0.2093023
## 6          0.09756098            0.2041763 0.1540389 0.4909091 0.2558140
##     alcohol quality
## 1 0.1290323       1
## 2 0.2419355       1
## 3 0.3387097       1
## 4 0.3064516       1
## 5 0.3064516       1
## 6 0.3387097       1
```

Logistic Regression Analysis

To test the algorithm the data is split into two groups, a training set and a test set. The training set trains the algorithm using the attributes to find the likeliness of passing. The test set uses the attributes and determines if the wine sample will pass or not, this output is then compared to the actual values. Using this comparison we can determine the effectiveness of the algorithm.

With unbalanced train set

```
#there are much more pass values than fail values
sum(norm_data$quality == 1)
```

```
## [1] 3258
```

```
sum(norm_data$quality == 0)
```

```
## [1] 1640
```

```
#split data into train and test sets based on set.seed
set.seed(123)
ind <- sample(2, nrow(norm_data),replace=TRUE,prob = c(0.7,0.3))
train <- norm_data[ind==1,]
test <- norm_data[ind==2,]

table(train$quality)
```

```
##
##    0    1
## 1159 2280
```

```
table(test$quality)
```

```
##
##   0   1
## 481 978
```

```
prop.table(table(train$quality))
```

```
##
##         0         1
## 0.3370166 0.6629834
```

```
#proportion of fail to pass is much too low
#we want to make sure that the train sample is more even to increase the accuracy of the algorithm
```

```
#preform logistic regression on ill proportioned data
#define the logistic regression
lrm <- glm( quality ~ ., family = "binomial", data = train)

summary(lrm)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1430  -0.9049   0.4454   0.8047   2.4319
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            0.02301    0.44882   0.051  0.95911
## fixed.acidity          1.30125    0.88517   1.470  0.14154
## volatile.acidity      -6.45925    0.50186 -12.870  < 2e-16 ***
## citric.acid            0.53956    0.60018   0.899  0.36865
## residual.sugar        12.58979    2.11259   5.959 2.53e-09 ***
## chlorides              0.66289    0.65810   1.007  0.31381
## free.sulfur.dioxide    2.04640    0.93150   2.197  0.02803 *
## total.sulfur.dioxide  -0.67855    0.61984  -1.095  0.27364
## density              -17.59240    4.49851  -3.911 9.20e-05 ***
## pH                     1.52679    0.47210   3.234  0.00122 **
## sulphates              1.65123    0.36513   4.522 6.12e-06 ***
## alcohol                3.97870    0.69746   5.705 1.17e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4395.3  on 3438  degrees of freedom
## Residual deviance: 3485.7  on 3427  degrees of freedom
## AIC: 3509.7
##
## Number of Fisher Scoring iterations: 5
```

```
#calculate predicted values
predicted <-predict(lrm, test, type = "response")
```

```
#display confusion matrix to test the effectiveness of the model
#convert defaults with a probability greater than 0.5
predicted_quality <- ifelse(predicted>= 0.5,1,0)

#create confusion matrix with a table
ConfMatrix <- table(actual = test$quality,
                    predicted = predicted_quality)
ConfMatrix
```

```
##       predicted
## actual   0   1
##      0 243 238
##      1  99 879
```

```
#evaluate accuracy, sensitivity and specificity
#Values
TP <- as.integer(ConfMatrix[2,2]) #True Positive
FP <- as.integer(ConfMatrix[1,2]) #False Positive
TN <- as.integer(ConfMatrix[1,1]) #True Negative
FN <- as.integer(ConfMatrix[2,1]) #False Negative

#Accuracy
#the number of correctly predicted wine quality
#divided by the total amount of wine tested

accuracy <- (TP + TN)/(TP + FN + FP + TN)
accuracy
```

```
## [1] 0.7690199
```

```
#Sensitivity
#number of correct predictions of passing wine
#divided by the total number of actually passing wine

sensitivity <- TP / (TP+FN)
sensitivity
```

```
## [1] 0.898773
```

```
#Specificity
#the number of correctly predicted wines that fail
#by the number of wines that actually fail
specificity <- TN / (TN+FP)
specificity
```

```
## [1] 0.5051975
```

Looking at these evaluation metrics we can see that the model does a relatively good job of correctly predicting the outcome of the wine quality test. Although, the sensitivity is high (89.9%), while the specificity is low (50.5%). This means that the model is good at correctly predicting that a wine will pass but bad at correctly determining if the wine will fail the test. This is most likely due to the unbalanced training set that was used. This will be explored in the next section.

With Balanced Train set

By using a balanced number of pass/fail cases, the model should do a better job at classifying the data into the correct categories.

```
#use simple downsampling to reduce the number of values of passes to have an equal number of fail and pass values

#separate train values into pass = 1, fail = 0
train_fail <- train[train$quality == 0,]
train_pass <-  train[train$quality == 1,]
#take a random sample of the pass values to create a balanced train set
train_p <- train_pass[sample(nrow(train_pass),1140),]

#combine the fail and pass values
train2 <- rbind(train_fail,train_p)

table(train2$quality)
```

```
##
##    0    1
## 1159 1140
```

```r
#preform logistic regression on balanced data
#define the logistic regression
lrm2 <- glm( quality ~ ., family = "binomial", data = train2)
#print the summary
summary(lrm2)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "binomial", data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7511  -0.9003  -0.2143   0.8676   2.7623
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -0.24548    0.53549  -0.458 0.646654
## fixed.acidity          2.03294    1.04445   1.946 0.051604 .
## volatile.acidity      -6.81709    0.62276 -10.947  < 2e-16 ***
## citric.acid           -0.04386    0.72266  -0.061 0.951599
## residual.sugar        13.75408    2.52275   5.452 4.98e-08 ***
## chlorides             -0.27224    0.95557  -0.285 0.775724
## free.sulfur.dioxide    1.27257    1.06070   1.200 0.230239
## total.sulfur.dioxide  -0.47969    0.73636  -0.651 0.514762
## density              -20.78417    5.45530  -3.810 0.000139 ***
## pH                     2.25142    0.55824   4.033 5.50e-05 ***
## sulphates              1.47348    0.42034   3.505 0.000456 ***
## alcohol                3.34466    0.83613   4.000 6.33e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3186.9  on 2298  degrees of freedom
## Residual deviance: 2494.3  on 2287  degrees of freedom
## AIC: 2518.3
##
## Number of Fisher Scoring iterations: 4
```

```r
#calculate predicted values
predicted2 <-predict(lrm2, test, type = "response")
```

```r
#display confusion matrix to test the effectiveness of the balanced model
#convert defaults with a probability greater than 0.5
predicted_quality2 <- ifelse(predicted2>= 0.5,1,0)

#create confusion matrix with a table
ConfMatrix2 <- table(actual = test$quality,
                     predicted = predicted_quality2)
ConfMatrix2
```

```
##       predicted
## actual   0   1
##      0 359 122
##      1 274 704
```

```r
#evaluate accuracy, sensitivity and specificity
#Values
TP2 <- as.integer(ConfMatrix2[2,2]) #True Positive
FP2 <- as.integer(ConfMatrix2[1,2]) #False Positive
TN2 <- as.integer(ConfMatrix2[1,1]) #True Negative
FN2 <- as.integer(ConfMatrix2[2,1]) #False Negative

#Accuracy
#the number of correctly predicted wine quality
#divided by the total amount of wine tested

accuracy2 <- (TP2 + TN2)/(TP2 + FN2 + FP2 + TN2)
accuracy2
```

```
## [1] 0.7285812
```

```
#Sensitivity
#number of correct predictions of passing wine
#divided by the total number of actually passing wine

sensitivity2 <- TP2 / (TP2+FN2)
sensitivity2
```

```
## [1] 0.7198364
```

```
#Specificity
#the number of correctly predicted wines that fail
#by the number of wines that actually fail
specificity2 <- TN2 / (TN2+FP2)
specificity2
```

```
## [1] 0.7463617
```

Comparing the balanced evaluation metrics to the unbalanced metrics, the accuracy is relatively the same, but the sensitivity is lower and the specificity is higher in the balanced model, this is what was expected. This model does a better job at correctly classifying a failed wine, but a moderately good job at correctly classifying a passing wine. Overall the balanced model does a better job at classifying the wine correctly into the fail or passing categories.