Logistic Regression Model (Final)

2025-05-05

```r
wine_df <- read.csv("data/wine-quality-white-and-red.csv")
head(wine_df)
```

```
##    type fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1 white           7.0             0.27        0.36           20.7     0.045
## 2 white           6.3             0.30        0.34            1.6     0.049
## 3 white           8.1             0.28        0.40            6.9     0.050
## 4 white           7.2             0.23        0.32            8.5     0.058
## 5 white           7.2             0.23        0.32            8.5     0.058
## 6 white           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
```

```r
quality_counts <- table(wine_df$quality)
print(quality_counts)
```

```
##
##    3    4    5    6    7    8    9
##   30  216 2138 2836 1079  193    5
```

Binary Logistic Regression

```r
wine_df <- wine_df %>%
  mutate(quality = as.numeric(as.character(quality))) %>%
  mutate(quality_binary = ifelse(quality >= 7, 1, 0)) %>%
  mutate(quality_binary = as.factor(quality_binary))

set.seed(1)
```

```r
index <- sample(1:nrow(wine_df), 0.7 * nrow(wine_df))
train <- wine_df[index, ]
test <- wine_df[-index, ]

log_model <- glm(quality_binary ~ . - quality, data = train, family = binomial)
summary(log_model)
```

```
##
## Call:
## glm(formula = quality_binary ~ . - quality, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7802  -0.6248  -0.3639  -0.1678   3.0108
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           3.291e+02  8.006e+01   4.111 3.94e-05 ***
## typewhite            -4.512e-01  3.089e-01  -1.460 0.144182
## fixed.acidity         4.660e-01  8.131e-02   5.731 9.98e-09 ***
## volatile.acidity     -3.415e+00  4.702e-01  -7.263 3.78e-13 ***
## citric.acid          -2.504e-01  4.129e-01  -0.606 0.544203
## residual.sugar        1.931e-01  3.183e-02   6.068 1.29e-09 ***
## chlorides            -1.057e+01  3.441e+00  -3.071 0.002132 **
## free.sulfur.dioxide   1.305e-02  3.633e-03   3.592 0.000328 ***
## total.sulfur.dioxide -4.831e-03  1.647e-03  -2.933 0.003352 **
## density              -3.502e+02  8.113e+01  -4.316 1.59e-05 ***
## pH                    2.576e+00  4.279e-01   6.020 1.74e-09 ***
## sulphates             2.193e+00  3.480e-01   6.301 2.97e-10 ***
## alcohol               5.474e-01  9.714e-02   5.636 1.75e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4518.9  on 4546  degrees of freedom
## Residual deviance: 3537.4  on 4534  degrees of freedom
## AIC: 3563.4
##
## Number of Fisher Scoring iterations: 6
```

```r
predictions <- predict(log_model, test, type = "response")
predicted_classes <- ifelse(predictions > 0.5, "1", "0")
confusionMatrix(as.factor(predicted_classes), test$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1482  272
##          1   89  107
```

```
##
##                Accuracy : 0.8149
##                  95% CI : (0.7969, 0.8319)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.1583
##
##                   Kappa : 0.2763
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9433
##             Specificity : 0.2823
##          Pos Pred Value : 0.8449
##          Neg Pred Value : 0.5459
##              Prevalence : 0.8056
##          Detection Rate : 0.7600
##    Detection Prevalence : 0.8995
##       Balanced Accuracy : 0.6128
##
##        'Positive' Class : 0
##
```

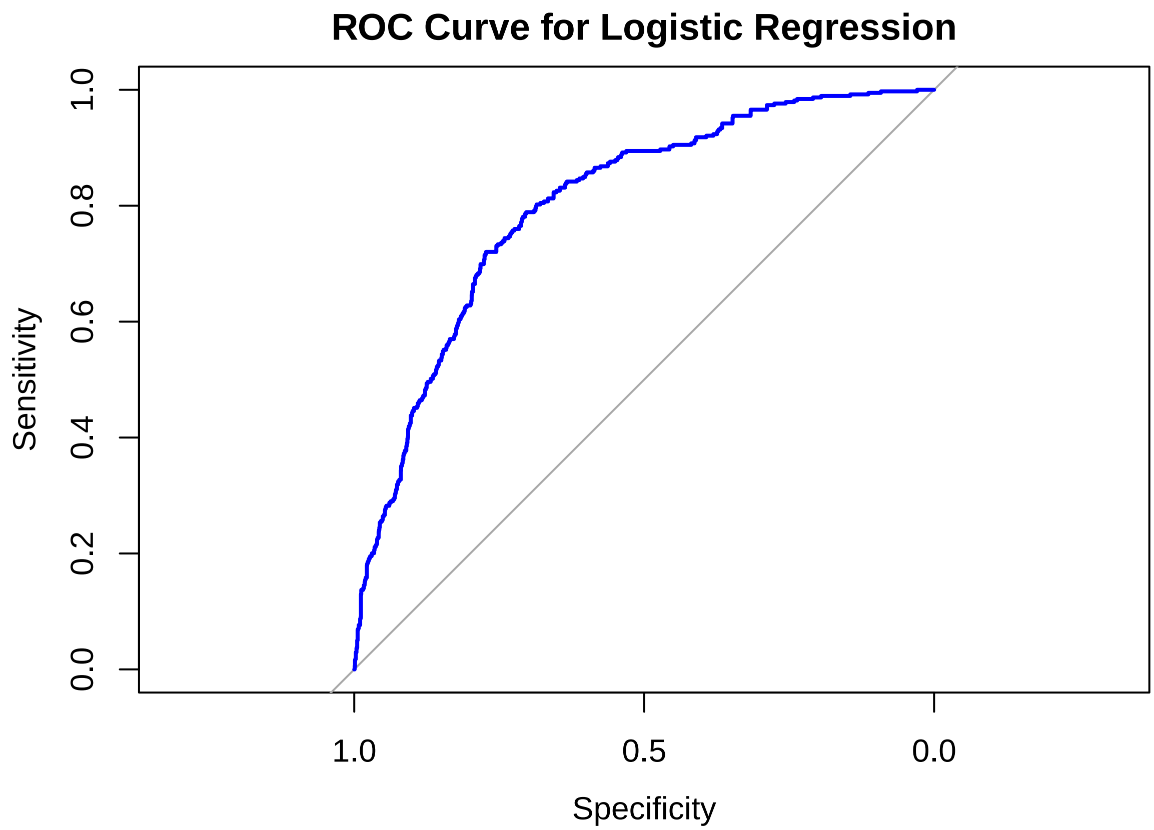
ROC Curve

```r
predictions_prob <- predict(log_model, test, type = "response")
roc_curve <- roc(test$quality_binary, predictions_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc_curve, col = "blue", main = "ROC Curve for Logistic Regression")
```




```
auc(roc_curve)
```

```
## Area under the curve: 0.8028
```

Cross Validation

```
control <- trainControl(method = "cv", number = 10)
set.seed(1)
cv_model <- train(quality_binary ~ . - quality, data = train, method = "glm", family = binomial, trCont
predictions <- predict(cv_model, newdata = test)
confusionMatrix(predictions, test$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1482  272
##          1   89  107
##
##                Accuracy : 0.8149
##                  95% CI : (0.7969, 0.8319)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.1583
```

```
##
##                   Kappa : 0.2763
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9433
##             Specificity : 0.2823
##          Pos Pred Value : 0.8449
##          Neg Pred Value : 0.5459
##              Prevalence : 0.8056
##          Detection Rate : 0.7600
##    Detection Prevalence : 0.8995
##       Balanced Accuracy : 0.6128
##
##        'Positive' Class : 0
##
```

Binary Logistic Regression with Variables Taken Out

```r
wine_df <- wine_df %>%
  mutate(quality = as.numeric(as.character(quality))) %>%
  mutate(quality_binary = ifelse(quality >= 7, 1, 0)) %>%
  mutate(quality_binary = as.factor(quality_binary))

set.seed(1)
index2 <- sample(1:nrow(wine_df), 0.7 * nrow(wine_df))
train2 <- wine_df[index2, ]
test2 <- wine_df[-index2, ]

log_model2 <- glm(quality_binary ~ . - quality - citric.acid - type, data = train2, family = binomial)
summary(log_model2)
```

```
##
## Call:
## glm(formula = quality_binary ~ . - quality - citric.acid - type,
##     family = binomial, data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7459  -0.6257  -0.3660  -0.1693   2.9076
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.632e+02  6.443e+01   4.085 4.41e-05 ***
## fixed.acidity         4.206e-01  7.567e-02   5.558 2.72e-08 ***
## volatile.acidity     -3.122e+00  4.241e-01  -7.361 1.83e-13 ***
## residual.sugar        1.684e-01  2.670e-02   6.308 2.83e-10 ***
## chlorides            -9.565e+00  3.272e+00  -2.923  0.00346 **
## free.sulfur.dioxide   1.432e-02  3.547e-03   4.037 5.42e-05 ***
## total.sulfur.dioxide -6.256e-03  1.388e-03  -4.506 6.62e-06 ***
## density              -2.842e+02  6.572e+01  -4.325 1.53e-05 ***
## pH                    2.458e+00  4.170e-01   5.896 3.73e-09 ***
```

```
## sulphates             2.200e+00  3.474e-01   6.333 2.41e-10 ***
## alcohol               6.110e-01  8.277e-02   7.381 1.57e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4518.9  on 4546  degrees of freedom
## Residual deviance: 3540.1  on 4536  degrees of freedom
## AIC: 3562.1
##
## Number of Fisher Scoring iterations: 6
```

```
predictions2 <- predict(log_model2, test2, type = "response")
predicted_classes2 <- ifelse(predictions2 > 0.5, "1", "0")
confusionMatrix(as.factor(predicted_classes2), test2$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1480  274
##          1   91  105
##
##                Accuracy : 0.8128
##                  95% CI : (0.7948, 0.8299)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.2206
##
##                   Kappa : 0.2683
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9421
##             Specificity : 0.2770
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.5357
##              Prevalence : 0.8056
##          Detection Rate : 0.7590
##    Detection Prevalence : 0.8995
##       Balanced Accuracy : 0.6096
##
##        'Positive' Class : 0
##
```

Multinomial Logistic Regression

```
wine_df <- read.csv("data/wine-quality-white-and-red.csv")
```

```r
wine_df$quality <- as.factor(wine_df$quality)

set.seed(1)
index3 <- sample(1:nrow(wine_df), 0.7 * nrow(wine_df))
train3 <- wine_df[index3, ]
test3 <- wine_df[-index3, ]

multi_model <- multinom(quality ~ ., data = train3)
```

```
## # weights:  98 (78 variable)
## initial  value 8848.053448
## iter  10 value 6112.967244
## iter  20 value 5795.162701
## iter  30 value 5483.743022
## iter  40 value 4942.177481
## iter  50 value 4848.170190
## iter  60 value 4825.676791
## iter  70 value 4814.244712
## iter  80 value 4812.501302
## iter  90 value 4811.705139
## iter 100 value 4810.173610
## final  value 4810.173610
## stopped after 100 iterations
```

```r
predictions3 <- predict(multi_model, test3)
confusionMatrix(predictions3, test3$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   1   0   3   0   0   0   0
##          4   0   5   4   0   0   0   0
##          5   2  39 393 190  24   6   0
##          6   4  22 232 599 228  35   1
##          7   0   1   3  71  65  18   2
##          8   0   0   0   1   0   0   0
##          9   1   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5451
##                  95% CI : (0.5227, 0.5674)
##     No Information Rate : 0.4415
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2704
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7  Class: 8
## Sensitivity          0.1250000 0.074627   0.6189   0.6957  0.20505 0.0000000
## Specificity          0.9984552 0.997876   0.8015   0.5207  0.94182 0.9994712
## Pos Pred Value       0.2500000 0.555556   0.6009   0.5343  0.40625 0.0000000
## Neg Pred Value       0.9964029 0.968058   0.8133   0.6840  0.85922 0.9697281
## Prevalence           0.0041026 0.034359   0.3256   0.4415  0.16256 0.0302564
## Detection Rate       0.0005128 0.002564   0.2015   0.3072  0.03333 0.0000000
## Detection Prevalence 0.0020513 0.004615   0.3354   0.5749  0.08205 0.0005128
## Balanced Accuracy    0.5617276 0.536251   0.7102   0.6082  0.57344 0.4997356
##                      Class: 9
## Sensitivity          0.0000000
## Specificity          0.9994864
## Pos Pred Value       0.0000000
## Neg Pred Value       0.9984607
## Prevalence           0.0015385
## Detection Rate       0.0000000
## Detection Prevalence 0.0005128
## Balanced Accuracy    0.4997432
```

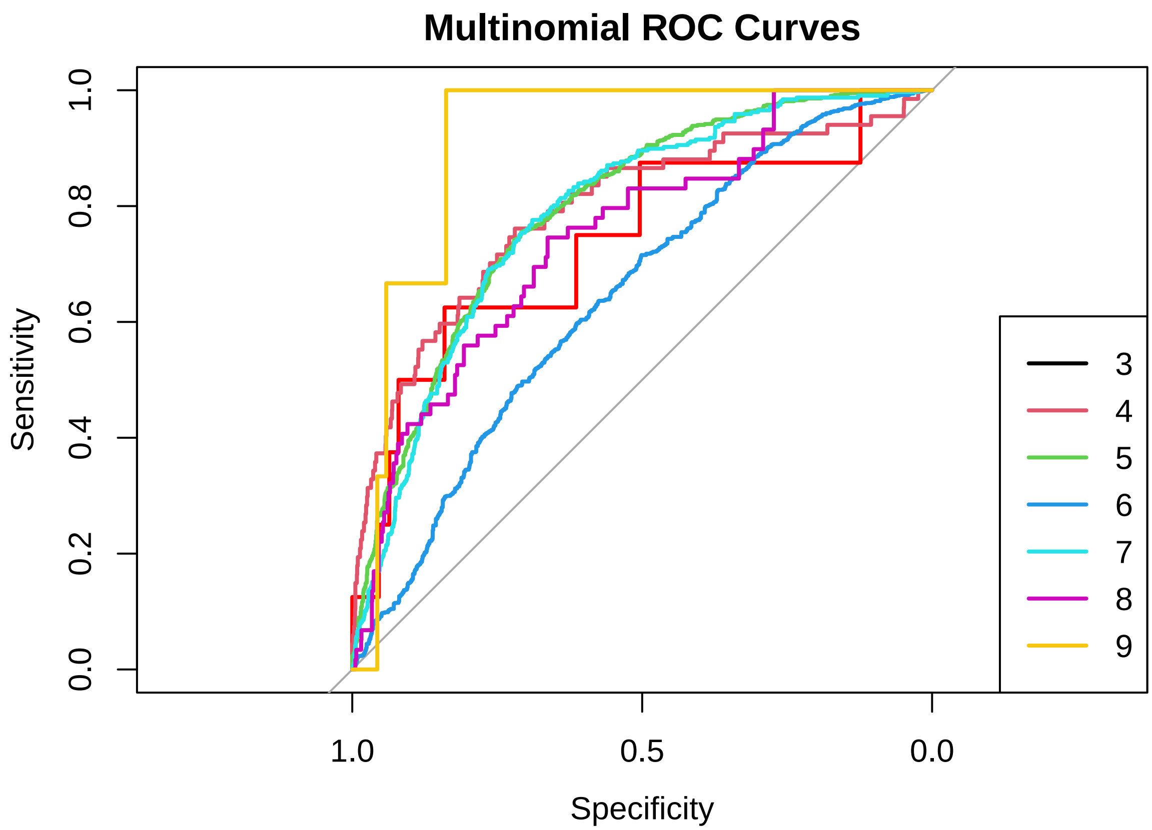
ROC Curve

```r
predicted_probs_multi <- predict(multi_model, test3, type = "probs")
true_labels <- test3$quality

roc_list <- lapply(levels(true_labels), function(class_label) {
  binary_labels <- ifelse(true_labels == class_label, 1, 0)
  roc(binary_labels, predicted_probs_multi[, class_label])
})
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
plot(roc_list[[1]], col = "red", main = "Multinomial ROC Curves", lwd = 2)
for (i in 2:length(roc_list)) {
  lines(roc_list[[i]], col = i, lwd = 2)
}
legend("bottomright", legend = levels(true_labels), col = 1:length(roc_list), lwd = 2)
```




```
sapply(roc_list, auc)
```

```
## [1] 0.7366117 0.7912033 0.7940301 0.6434688 0.7860067 0.7477525 0.9121726
```

Cross Validation

```
train3$quality <- as.factor(train3$quality)
test3$quality <- as.factor(test3$quality)
control2 <- trainControl(method = "cv", number = 10)
set.seed(1)
```

```r
multi_model_cv <- train(quality ~ ., data = train3, method = "multinom", trControl = control2)
predictions3 <- predict(multi_model_cv, newdata = test3)
confusionMatrix(predictions3, test3$quality)
```