logistic_regression

Rishika Cherivirala

2025-04-30

```
wine_df <- read.csv("data/wine-quality-white-and-red (1).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
```

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

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

Multinomial Logistic Regression

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

set.seed(1)
```

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

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

```
## # 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
predictions <- predict(multi_model, test)

confusionMatrix(predictions, test$quality)
```

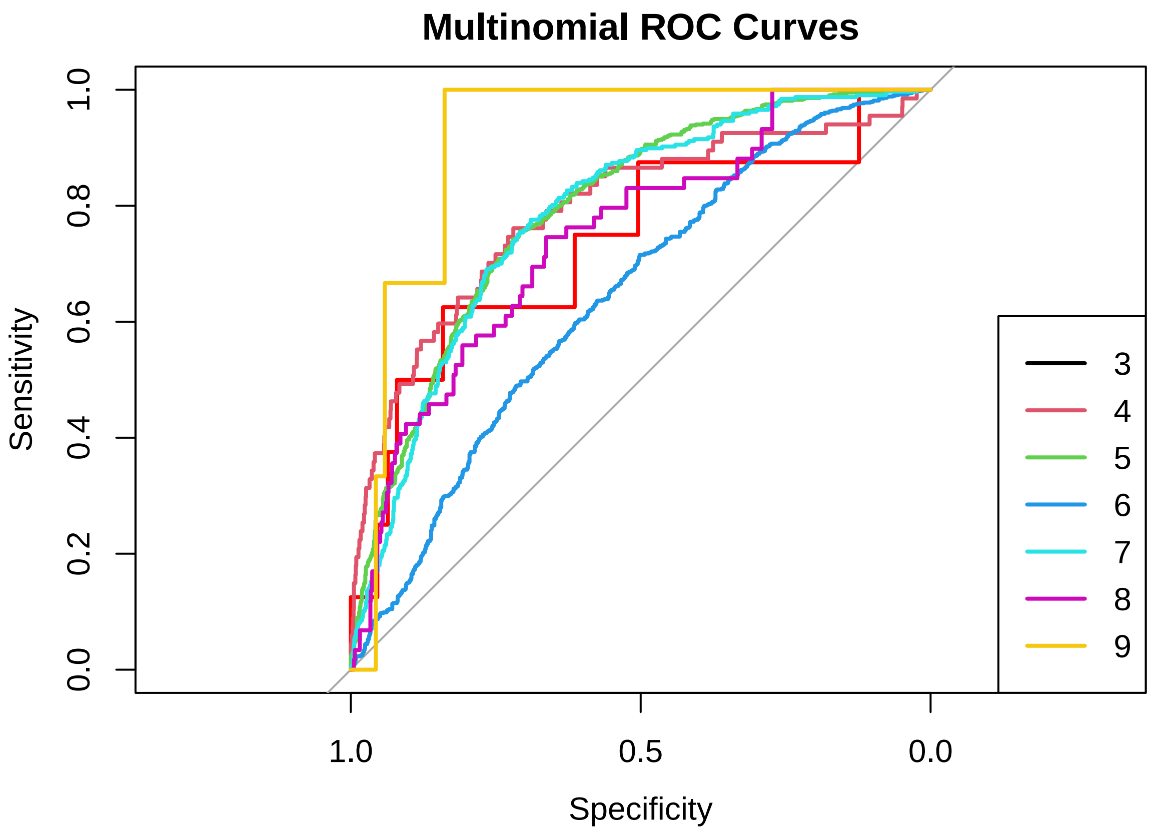
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   1   0   3   0   0   0   0
##          4   0   5   4   0   0   0   0
##          5   2  39 393 190  24   6   0
##          6   4  22 232 599 228  35   1
##          7   0   1   3  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

```
predicted_probs_multi <- predict(multi_model, test, type = "probs")

true_labels <- test$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

```
# wine_df$quality <- as.factor(wine_df$quality)
#
# control <- trainControl(method = "cv", number = 10)
#
```

```
# set.seed(1)
# multi_model_cv <- train(quality ~ ., data = wine_df, method = "multinom", trControl = control)
#
# print(multi_model_cv)
#
# predictions <- predict(multi_model_cv, newdata = test)
#
# confusionMatrix(predictions, test$quality)
```

Accuracy= 0.5477

RMSE

#Logistic Regression

Since logistic regression only has 2 output variables: 0 or 1 or "Yes" or "No". I decided to split the data in a way where wines that have a quality >= 7, are 0 and wines that have a quality < 7, are 1.

(https://vineroutes.com/wine-rating-system) shows the scale for wine quality.

```
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_model <- glm(quality_binary ~ . - quality, data = train2, family = binomial)
summary(log_model)
```

```
##
## Call:
## glm(formula = quality_binary ~ . - quality, family = binomial,
##     data = train2)
##
## 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
```

```
predictions2 <- predict(log_model, 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 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
##
```

```
log2_index <- sample(1:nrow(wine_df), 0.7 * nrow(wine_df))
log2_train <- wine_df[log2_index, ]
log2_test <- wine_df[-log2_index, ]

log_model2 <- glm(quality_binary ~ . - quality - type - citric.acid, data = log2_train, family = binomi
```

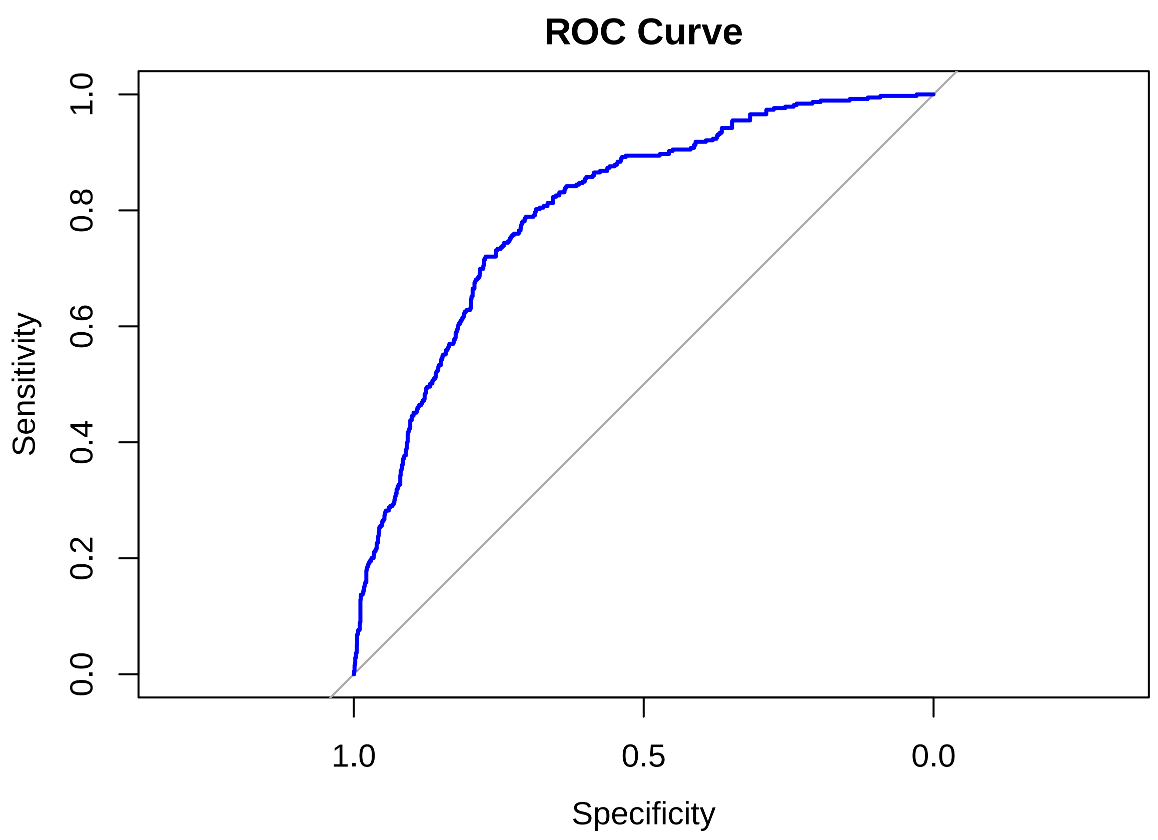
```r
log_predictions2 <- predict(log_model2, log2_test, type = "response")

log_predicted_classes2 <- ifelse(log_predictions2 > 0.5, "1", "0")
```

ROC Curve

```r
predictions_prob <- predict(log_model, test2, type = "response")

roc_curve <- roc(test2$quality_binary, predictions_prob)
```

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

## Setting direction: controls < cases
```

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



```r
auc(roc_curve)
```

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

AUC = 0.8028. That means the model does an okay job in predicting it and is not completely due to random chance.

Cross Validation

```r
control2 <- trainControl(method = "cv", number = 10)

set.seed(1)
cv_model <- train(quality_binary ~ . - quality, data = wine_df, method = "glm", family = binomial, trCo

print(cv_model)
```

```
## Generalized Linear Model
##
## 6497 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5847, 5847, 5847, 5847, 5848, 5847, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8174517  0.2741793
```

```r
predictions <- predict(cv_model, newdata = test2)

confusionMatrix(predictions, test2$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1490  275
##          1   81  104
##
##                Accuracy : 0.8174
##                  95% CI : (0.7996, 0.8344)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.09824
##
##                   Kappa : 0.2766
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9484
##             Specificity : 0.2744
##          Pos Pred Value : 0.8442
##          Neg Pred Value : 0.5622
##              Prevalence : 0.8056
##          Detection Rate : 0.7641
##    Detection Prevalence : 0.9051
##       Balanced Accuracy : 0.6114
##
```

```
##        'Positive' Class : 0
##
```

Since the data is a little skewed with a lot more samples landing in the 0 category than the 1 category, which might affect the accuracy of the model, I'm going to try adding weights to the regression to try to make it more accurate

Weighted Logistic Regression

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

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

weights <- ifelse(train3$quality_binary == 1, 0.8, 0.2)

log_model_weighted <- glm(quality_binary ~ . - quality, data = train3, family = binomial, weights = wei
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
predictions_weighted <- predict(log_model_weighted, test3, type = "response")
predicted_classes_weighted <- ifelse(predictions_weighted > 0.5, "1", "0")

confusionMatrix(as.factor(predicted_classes_weighted), as.factor(test3$quality_binary))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1124   91
##          1  447  288
##
##                Accuracy : 0.7241
##                  95% CI : (0.7037, 0.7438)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3505
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7155
##             Specificity : 0.7599
##          Pos Pred Value : 0.9251
##          Neg Pred Value : 0.3918
##              Prevalence : 0.8056
##          Detection Rate : 0.5764
##    Detection Prevalence : 0.6231
##       Balanced Accuracy : 0.7377
##
##        'Positive' Class : 0
##
```

```
predictions_prob2 <- predict(log_model_weighted, test3, type = "response")
```

```
# For normal logistic regression
roc_normal <- roc(test2$quality_binary, predictions_prob)
```

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

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

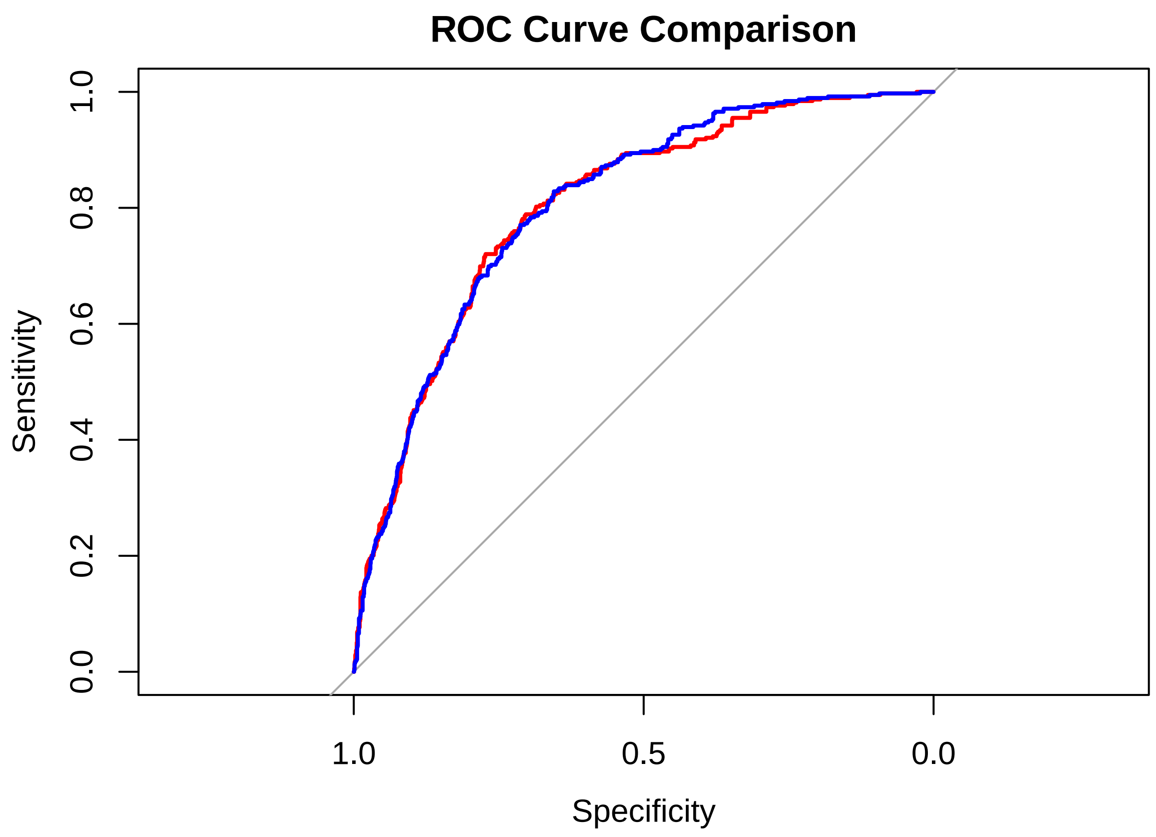
```
plot(roc_normal, col = "red", main = "ROC Curve Comparison")
auc(roc_normal)
```

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

```
# For weighted logistic regression
roc_weighted <- roc(test3$quality_binary, predictions_prob2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(roc_weighted, col = "blue", add = TRUE)
```



```r
auc(roc_weighted)
```

```
## Area under the curve: 0.8053
```

The ROC curves for weighted/normal are basically the same. So the model's discriminatory power did not really improve.

Balances Logistic Regression

```r
minority_class <- wine_df %>% filter(quality_binary == 1)
majority_class <- wine_df %>% filter(quality_binary == 0)

set.seed(1)
majority_class_undersampled <- majority_class %>%
  sample_n(nrow(minority_class))

balanced_df <- bind_rows(minority_class, majority_class_undersampled)

set.seed(1)
index_balanced <- sample(1:nrow(balanced_df), 0.7 * nrow(balanced_df))
train_balanced <- balanced_df[index_balanced, ]
test_balanced <- balanced_df[-index_balanced, ]
```

```r
log_model_balanced <- glm(quality_binary ~ . - quality, data = train_balanced, family = binomial)

predictions_balanced <- predict(log_model_balanced, test_balanced, type = "response")
predicted_classes_balanced <- ifelse(predictions_balanced > 0.5, "1", "0")

confusionMatrix(as.factor(predicted_classes_balanced), as.factor(test_balanced$quality_binary))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 301  72
##          1 104 290
##
##                Accuracy : 0.7705
##                  95% CI : (0.7391, 0.7999)
##     No Information Rate : 0.528
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5418
##
##  Mcnemar's Test P-Value : 0.01945
##
##             Sensitivity : 0.7432
##             Specificity : 0.8011
##          Pos Pred Value : 0.8070
##          Neg Pred Value : 0.7360
##              Prevalence : 0.5280
##          Detection Rate : 0.3924
##    Detection Prevalence : 0.4863
##       Balanced Accuracy : 0.7722
##
##        'Positive' Class : 0
##
```

```r
# For normal logistic regression
roc_normal <- roc(test2$quality_binary, predictions_prob)
```

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

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

```r
plot(roc_normal, col = "red", main = "ROC Curve Comparison", lwd = 2)

auc_normal <- auc(roc_normal)
cat("AUC for Normal Logistic Regression:", auc_normal, "\n")
```

```
## AUC for Normal Logistic Regression: 0.8028011
```

```r
# For weighted logistic regression
roc_weighted <- roc(test3$quality_binary, predictions_prob2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
lines(roc_weighted, col = "blue", lwd = 2)

auc_weighted <- auc(roc_weighted)
cat("AUC for Weighted Logistic Regression:", auc_weighted, "\n")
```

```
## AUC for Weighted Logistic Regression: 0.8052633
```

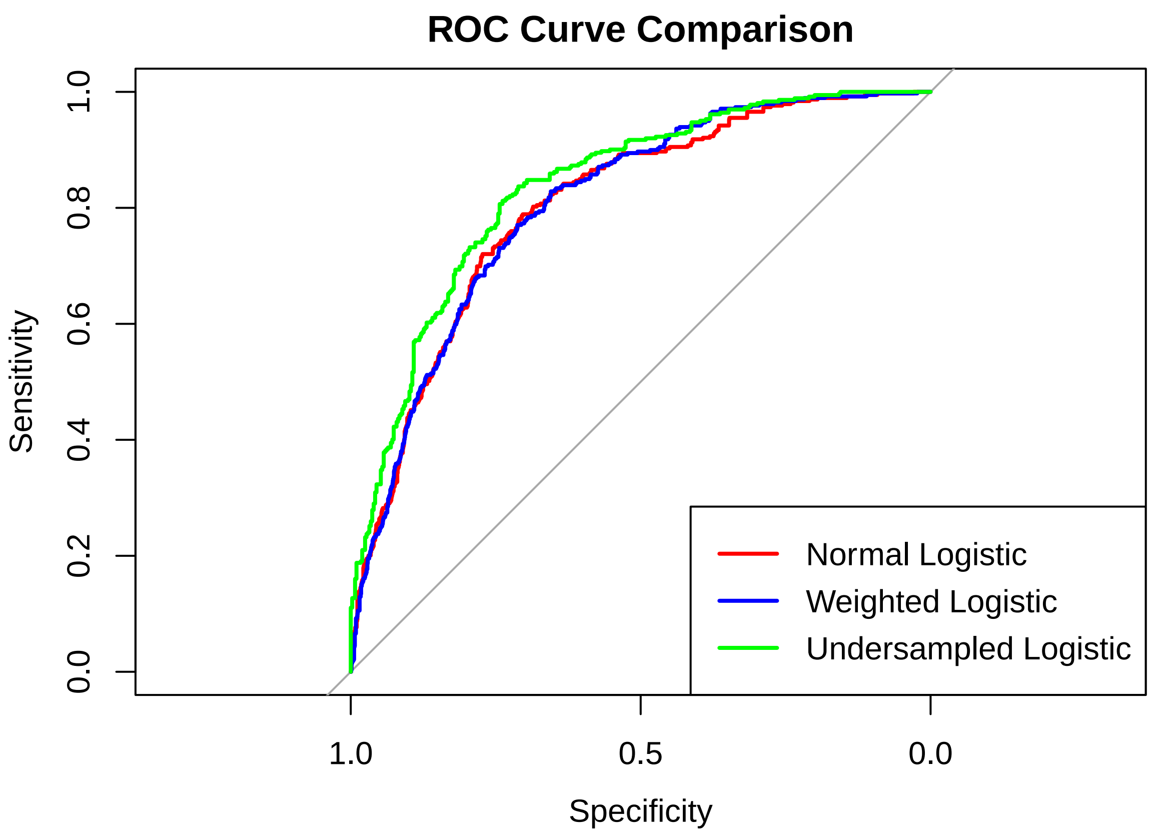
```r
# For balanced logistic regression
roc_balanced <- roc(test_balanced$quality_binary, predictions_balanced)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
lines(roc_balanced, col = "green", lwd = 2)

auc_balanced <- auc(roc_balanced)
cat("AUC for Undersampled Logistic Regression:", auc_balanced, "\n")
```

```
## AUC for Undersampled Logistic Regression: 0.8355296
```

```r
# Add Legend to the Plot
legend("bottomright",
       legend = c("Normal Logistic", "Weighted Logistic", "Undersampled Logistic"),
       col = c("red", "blue", "green"),
       lwd = 2)
```



### RMSE

```r
actual_normal <- as.numeric(as.character(test2$quality_binary))

rmse_normal <- sqrt(mean((predictions_prob - actual_normal)^2))
cat("RMSE for Normal Logistic Regression:", rmse_normal, "\n")
```

```
## RMSE for Normal Logistic Regression: 0.3557835
```

```r
actual_weighted <- as.numeric(as.character(test3$quality_binary))

rmse_weighted <- sqrt(mean((predictions_prob2 - actual_weighted)^2))
cat("RMSE for Weighted Logistic Regression:", rmse_weighted, "\n")
```

```
## RMSE for Weighted Logistic Regression: 0.427233
```

```r
actual_balanced <- as.numeric(as.character(test_balanced$quality_binary))

rmse_balanced <- sqrt(mean((predictions_balanced - actual_balanced)^2))
cat("RMSE for Undersampled Logistic Regression:", rmse_balanced, "\n")
```

```
## RMSE for Undersampled Logistic Regression: 0.4077837
```

```r
actual_normal2 <- as.numeric(as.character(log2_test$quality_binary))
rmse_log2 <- sqrt(mean((log_predictions2 - actual_normal2)^2))
cat("RMSE for Specific Predictors Logistic Regression:", rmse_log2, "\n")
```

```
## RMSE for Specific Predictors Logistic Regression: 0.3555901
```

```r
AIC(log_model, log_model_weighted, log_model_balanced, log_model2)
```

```
## Warning in AIC.default(log_model, log_model_weighted, log_model_balanced, :
## models are not all fitted to the same number of observations
```

```
##                    df       AIC
## log_model          13 3563.4361
## log_model_weighted 13  951.6473
## log_model_balanced 13 1953.6609
## log_model2         11 3573.4232
```

```r
# checking for multicollinearity

vif_model <- lm(quality ~ ., data = wine_df)
vif(vif_model)
```

```
##                 type        fixed.acidity     volatile.acidity
##             7.234696             5.089161             2.183694
##          citric.acid       residual.sugar            chlorides
##             1.622161             9.769547             1.660215
##  free.sulfur.dioxide total.sulfur.dioxide              density
##             2.240663             4.050840            22.561563
##                   pH            sulphates              alcohol
##             2.580317             1.572390             5.670002
##       quality_binary
##             1.238910
```

Since density has a large vif, I'm going to remove it from the model.

```r
set.seed(1)

# Remove 'quality' from wine_df
wine_df2 <- subset(wine_df, select = -quality)

# Train-test split
index <- sample(1:nrow(wine_df2), 0.7 * nrow(wine_df2))
train <- wine_df2[index, ]
test <- wine_df2[-index, ]

# Fitting models
train$quality_binary <- factor(train$quality_binary)
test$quality_binary <- factor(test$quality_binary, levels = levels(train$quality_binary))

model_full <- multinom(quality_binary ~ ., data = train)
```

```
## # weights:  14 (13 variable)
## initial  value 3151.740230
## iter  10 value 1851.185132
## iter  20 value 1777.845727
## iter  30 value 1776.496685
## iter  40 value 1775.585843
## iter  50 value 1774.485533
## final  value 1768.720837
## converged
```

```
model_nodensity <- multinom(quality_binary ~ . - density, data = train)
```

```
## # weights:  13 (12 variable)
## initial  value 3151.740230
## iter  10 value 1877.564741
## iter  20 value 1778.332902
## final  value 1778.329532
## converged
```

```
model_nodensity_sugar <- multinom(quality_binary ~ . - density - residual.sugar, data = train)
```

```
## # weights:  12 (11 variable)
## initial  value 3151.740230
## iter  10 value 1851.411800
## iter  20 value 1793.858082
## final  value 1793.857645
## converged
```

```
model_top <- multinom(quality_binary ~ alcohol + fixed.acidity + volatile.acidity + sulphates, data = t
```

```
## # weights:  6 (5 variable)
## initial  value 3151.740230
## iter  10 value 1823.342739
## final  value 1823.300931
## converged
```

```
# Predicting models
pred_full <- factor(predict(model_full, newdata = test), levels = levels(test$quality_binary))
pred_nodensity <- factor(predict(model_nodensity, newdata = test), levels = levels(test$quality_binary)
pred_nodensity_sugar <- factor(predict(model_nodensity_sugar, newdata = test), levels = levels(test$qua
pred_top <- factor(predict(model_top, newdata = test), levels = levels(test$quality_binary))

# Evaluation matrix
confusionMatrix(pred_full, test$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1481  272
##          1   90  107
```

```
##
##                Accuracy : 0.8144
##                  95% CI : (0.7964, 0.8314)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.1727
##
##                   Kappa : 0.2752
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9427
##             Specificity : 0.2823
##          Pos Pred Value : 0.8448
##          Neg Pred Value : 0.5431
##              Prevalence : 0.8056
##          Detection Rate : 0.7595
##    Detection Prevalence : 0.8990
##       Balanced Accuracy : 0.6125
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(pred_nodensity, test$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1483  279
##          1   88  100
##
##                Accuracy : 0.8118
##                  95% CI : (0.7937, 0.8289)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.2563
##
##                   Kappa : 0.257
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9440
##             Specificity : 0.2639
##          Pos Pred Value : 0.8417
##          Neg Pred Value : 0.5319
##              Prevalence : 0.8056
##          Detection Rate : 0.7605
##    Detection Prevalence : 0.9036
##       Balanced Accuracy : 0.6039
##
##        'Positive' Class : 0
##
```

```r
confusionMatrix(pred_nodensity_sugar, test$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1483  275
##          1   88  104
##
##                Accuracy : 0.8138
##                  95% CI : (0.7958, 0.8309)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.1879
##
##                   Kappa : 0.2687
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9440
##             Specificity : 0.2744
##          Pos Pred Value : 0.8436
##          Neg Pred Value : 0.5417
##              Prevalence : 0.8056
##          Detection Rate : 0.7605
##    Detection Prevalence : 0.9015
##       Balanced Accuracy : 0.6092
##
##        'Positive' Class : 0
##
```

```r
confusionMatrix(pred_top, test$quality_binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1484  281
##          1   87   98
##
##                Accuracy : 0.8113
##                  95% CI : (0.7932, 0.8284)
##     No Information Rate : 0.8056
##     P-Value [Acc > NIR] : 0.2752
##
##                   Kappa : 0.2522
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9446
##             Specificity : 0.2586
##          Pos Pred Value : 0.8408
##          Neg Pred Value : 0.5297
##               Prevalence : 0.8056
##           Detection Rate : 0.7610
##     Detection Prevalence : 0.9051
##        Balanced Accuracy : 0.6016
##
##         'Positive' Class : 0
##
```

```
# AIC
AIC(model_full)
```

```
## [1] 3563.442
```

```
AIC(model_nodensity)
```

```
## [1] 3580.659
```

```
AIC(model_nodensity_sugar)
```

```
## [1] 3609.715
```

```
AIC(model_top)
```

```
## [1] 3656.602
```

```
# calculating the rmse values too

calculate_rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

rmse_full <- calculate_rmse(as.numeric(test$quality_binary), as.numeric(pred_full))
rmse_nodensity <- calculate_rmse(as.numeric(test$quality_binary), as.numeric(pred_nodensity))
rmse_nodensity_sugar <- calculate_rmse(as.numeric(test$quality_binary), as.numeric(pred_nodensity_sugar
rmse_top <- calculate_rmse(as.numeric(test$quality_binary), as.numeric(pred_top))

rmse_full
```

```
## [1] 0.4308608
```

```
rmse_nodensity
```

```
## [1] 0.4338261
```

```
rmse_nodensity_sugar
```

```
## [1] 0.4314555
```

```r
rmse_top
```

```
## [1] 0.4344168
```

The RMSE values are basically the same.