stat462 assignment 2

Chris Chang

2025-08-27

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```r
library(knitr)
```

Part A

In this question you are going to training a logistic regression for automatic classification of two types of pumpkin seeds, Çerçevelik and Ürgüp Sivrisi, based on geometric (measured) features of these seeds.

Q1

Load the datasets seeds_training.csv and seeds_test.csv. Since the class labels (the seed types) contain characters that R struggles with, encode the class features as a factor variable in both training and test set.

```r
# Load the datasets
seeds_train <- read.csv("seeds_training.csv")
seeds_test <- read.csv("seeds_test.csv")

# Display basic information about the datasets
cat("Training dataset dimensions:", dim(seeds_train), "\n")
```

```
## Training dataset dimensions: 2000 13
```

```r
cat("Test dataset dimensions:", dim(seeds_test), "\n")
```

```
## Test dataset dimensions: 500 13
```

```r
cat("Features:", names(seeds_train)[1:12], "\n")
```

```
## Features: Area Perimeter Major_Axis_Length Minor_Axis_Length Convex_Area Equiv_Diameter Eccentricity
```

```r
# Check the class distribution
cat("\nClass distribution in training set:\n")
```

```
##
## Class distribution in training set:
```

```r
table(seeds_train$Class)
```

```
##
##    Cercevelik Urgup Sivrisi
##          1039           961
```

```r
# Encode class features as factor variables
seeds_train$Class <- as.factor(seeds_train$Class)
seeds_test$Class <- as.factor(seeds_test$Class)

# Check which class R treats as 1
contrasts(seeds_train$Class)
```

```
##               Urgup Sivrisi
## Cercevelik                0
## Urgup Sivrisi             1
```

```r
# Display the levels to confirm encoding
cat("\nClass levels:", levels(seeds_train$Class), "\n")
```

```
##
## Class levels: Cercevelik Urgup Sivrisi
```

Q2

Fit a multiple logistic regression classifier using all of the features and record its accuracy on the test set.

```r
# Fit logistic regression using all features
logreg_all <- glm(Class ~ ., data = seeds_train, family = binomial)

# Display model summary
summary(logreg_all)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial, data = seeds_train)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        214.593346 204.305124   1.050 0.293554
## Area                 0.005708   0.002214   2.579 0.009923 **
## Perimeter            0.017973   0.062029   0.290 0.772008
## Major_Axis_Length    0.264676   0.115440   2.293 0.021862 *
## Minor_Axis_Length    0.515927   0.141661   3.642 0.000271 ***
## Convex_Area         -0.004525   0.002210  -2.048 0.040585 *
## Equiv_Diameter      -1.370730   0.231176  -5.929 3.04e-09 ***
## Eccentricity         7.366701  34.013086   0.217 0.828532
## Solidity           -78.846626 178.684652  -0.441 0.659024
## Extent               0.132721   1.494525   0.089 0.929237
## Roundness            3.469268  46.721137   0.074 0.940808
## Aspect_Ration       -4.298850   7.201718  -0.597 0.550561
## Compactness        -68.374384  91.412640  -0.748 0.454475
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2769.5  on 1999  degrees of freedom
## Residual deviance: 1130.6  on 1987  degrees of freedom
## AIC: 1156.6
##
## Number of Fisher Scoring iterations: 7
```

```r
# Make predictions on test set
logreg_all_test_pred <- predict(logreg_all, seeds_test, type = "response")
logreg_all_test_classify <- ifelse(logreg_all_test_pred > 0.5,
                                         "Urgup Sivrisi", "Cercevelik")
logreg_all_test_classify <- as.factor(logreg_all_test_classify)

# Calculate test set accuracy
logreg_all_acc <- mean(logreg_all_test_classify == seeds_test$Class)
cat("Test set accuracy (full model):", round(logreg_all_acc, 4), "\n")
```

```
## Test set accuracy (full model): 0.874
```

Q3

Provide a confusion matrix for your model based on its predictions on the test set. Why does the concept of sensitivity and specificity not make a lot of sense here?

```r
# Create confusion matrix
logreg_full_confusion <- confusionMatrix(logreg_all_test_classify, seeds_test$Class)
print(logreg_full_confusion)
```

```
## Confusion Matrix and Statistics
```

```
##
##                 Reference
## Prediction      Cercevelik Urgup Sivrisi
##   Cercevelik           242            44
##   Urgup Sivrisi         19           195
##
##                Accuracy : 0.874
##                  95% CI : (0.8417, 0.9018)
##     No Information Rate : 0.522
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7464
##
##  Mcnemar's Test P-Value : 0.002497
##
##             Sensitivity : 0.9272
##             Specificity : 0.8159
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.9112
##              Prevalence : 0.5220
##          Detection Rate : 0.4840
##    Detection Prevalence : 0.5720
##       Balanced Accuracy : 0.8716
##
##        'Positive' Class : Cercevelik
##
```

Sensitivity and specificity are most meaningful when there's a clear distinction between positive and negative cases, often in medical or diagnostic contexts where one outcome is considered more critical than the other. In our seed classification problem, neither Çerçevelik nor Ürgüp Sivrisi represents a "positive" or "negative" case - they are simply two different types of seeds with equal importance. The concepts of sensitivity (true positive rate) and specificity (true negative rate) are less interpretable when both classes are of equal interest and there's no inherent ordering or clinical significance to the classification.

Q4

Consulting the summary of your glm model, which three features are most significant?

```r
# Extract coefficients and p-values
model_summary <- summary(logreg_all)
coefficients_df <- data.frame(
  Feature = rownames(model_summary$coefficients),
  Estimate = model_summary$coefficients[, 1],
  Std_Error = model_summary$coefficients[, 2],
  Z_value = model_summary$coefficients[, 3],
  P_value = model_summary$coefficients[, 4]
)

print(model_summary)
```

```
##
## Call:
```

```
## glm(formula = Class ~ ., family = binomial, data = seeds_train)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       214.593346 204.305124   1.050 0.293554
## Area                0.005708   0.002214   2.579 0.009923 **
## Perimeter           0.017973   0.062029   0.290 0.772008
## Major_Axis_Length   0.264676   0.115440   2.293 0.021862 *
## Minor_Axis_Length   0.515927   0.141661   3.642 0.000271 ***
## Convex_Area        -0.004525   0.002210  -2.048 0.040585 *
## Equiv_Diameter     -1.370730   0.231176  -5.929 3.04e-09 ***
## Eccentricity        7.366701  34.013086   0.217 0.828532
## Solidity          -78.846626 178.684652  -0.441 0.659024
## Extent              0.132721   1.494525   0.089 0.929237
## Roundness           3.469268  46.721137   0.074 0.940808
## Aspect_Ration      -4.298850   7.201718  -0.597 0.550561
## Compactness       -68.374384  91.412640  -0.748 0.454475
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2769.5  on 1999  degrees of freedom
## Residual deviance: 1130.6  on 1987  degrees of freedom
## AIC: 1156.6
##
## Number of Fisher Scoring iterations: 7
```

```r
# Remove intercept and sort by p-value
coefficients_df <- coefficients_df[-1, ]  # Remove intercept
coefficients_df <- coefficients_df[order(coefficients_df$P_value), ]

cat("Three most significant features based on p-values:\n")
```

```
## Three most significant features based on p-values:
```

```r
print(coefficients_df[1:3, c("Estimate", "P_value")])
```

```
##                      Estimate      P_value
## Equiv_Diameter    -1.37072986 3.040790e-09
## Minor_Axis_Length  0.51592750 2.705284e-04
## Area               0.00570849 9.922781e-03
```

```r
# Store the three most significant features
top_features <- coefficients_df$Feature[1:3]
print(top_features)
```

```
## [1] "Equiv_Diameter"    "Minor_Axis_Length" "Area"
```

Q5 a)

Train another glm model, but this time using only the three most significant features from your first model run.

```r
# Fit sparse model
logreg_sparse <- glm(Class ~ Equiv_Diameter + Minor_Axis_Length + Area,
                     data = seeds_train, family = binomial)

# Display model summary
summary(logreg_sparse)
```

```
##
## Call:
## glm(formula = Class ~ Equiv_Diameter + Minor_Axis_Length + Area,
##     family = binomial, data = seeds_train)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       93.3898356 11.1193184   8.399  < 2e-16 ***
## Equiv_Diameter    -0.4651483  0.0683952  -6.801 1.04e-11 ***
## Minor_Axis_Length -0.1863422  0.0083172 -22.405  < 2e-16 ***
## Area               0.0012058  0.0001391   8.667  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2769.5  on 1999  degrees of freedom
## Residual deviance: 1224.8  on 1996  degrees of freedom
## AIC: 1232.8
##
## Number of Fisher Scoring iterations: 6
```

Q5 b)

Compute test set accuracy and a confusion matrix in the same way as for your first model and briefly compare the two models. Which model would you pick, and why?

```r
# Make predictions on test set
logreg_sparse_test_pred <- predict(logreg_sparse, seeds_test, type = "response")
logreg_sparse_test_classify <- ifelse(logreg_sparse_test_pred > 0.5,
                                      "Urgup Sivrisi", "Cercevelik")
logreg_sparse_test_classify <- as.factor(logreg_sparse_test_classify)

# Calculate test set accuracy
logreg_sparse_acc <- mean(logreg_sparse_test_classify == seeds_test$Class)
cat("Test set accuracy (sparse model):", round(logreg_sparse_acc, 4), "\n")
```

```
## Test set accuracy (sparse model): 0.878
```

```r
cat("Test set accuracy (full model):", round(logreg_all_acc, 4), "\n", "\n", "\n")
```

```
## Test set accuracy (full model): 0.874
##
##
```

```r
# Create confusion matrix
cat("Confusion Matrix for sparse logistic regression model:", "\n")
```

```
## Confusion Matrix for sparse logistic regression model:
```

```r
logreg_sparse_confusion <- confusionMatrix(logreg_sparse_test_classify, seeds_test$Class)
print(logreg_sparse_confusion)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Cercevelik Urgup Sivrisi
##   Cercevelik           237            37
##   Urgup Sivrisi         24           202
##
##                Accuracy : 0.878
##                  95% CI : (0.8461, 0.9054)
##     No Information Rate : 0.522
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.755
##
##  Mcnemar's Test P-Value : 0.1244
##
##             Sensitivity : 0.9080
##             Specificity : 0.8452
##          Pos Pred Value : 0.8650
##          Neg Pred Value : 0.8938
##              Prevalence : 0.5220
##          Detection Rate : 0.4740
##    Detection Prevalence : 0.5480
##       Balanced Accuracy : 0.8766
##
##        'Positive' Class : Cercevelik
##
```

```r
# Compare the two models
cat("\n=== MODEL COMPARISON ===\n")
```

```
##
## === MODEL COMPARISON ===
```

```r
cat("Full Model (12 features):\n")
```

```
## Full Model (12 features):
```

```r
cat("  - Test Accuracy:", round(logreg_all_acc, 4), "\n")
```

```
##   - Test Accuracy: 0.874
```

```r
cat("  - Number of features:", 12, "\n")
```

```
##   - Number of features: 12
```

```r
cat("\nSparse Model (3 features):\n")
```

```
##
## Sparse Model (3 features):
```

```r
cat("  - Test Accuracy:", round(logreg_sparse_acc, 4), "\n")
```

```
##   - Test Accuracy: 0.878
```

```r
cat("  - Number of features:", 3, "\n")
```

```
##   - Number of features: 3
```

```r
cat("  - Features used: Equiv_Diameter, Minor_Axis_Length, Area\n")
```

```
##   - Features used: Equiv_Diameter, Minor_Axis_Length, Area
```

```r
cat("\nAccuracy difference:", round(logreg_all_acc - logreg_sparse_acc, 4), "\n")
```

```
##
## Accuracy difference: -0.004
```

```r
# Compare AIC values
cat("\nModel Complexity Comparison:\n")
```

```
##
## Model Complexity Comparison:
```

```r
cat("Full Model AIC:", round(logreg_all$aic, 2), "\n")
```

```
## Full Model AIC: 1156.64
```

```r
cat("Sparse Model AIC:", round(logreg_sparse$aic, 2), "\n")
```

```
## Sparse Model AIC: 1232.79
```

Part B

Colors can be coded via their RGB (red, green, blue) value in the form (r,g,b), where r, g, and b are integers between 0 and 255. For example, (255,0,0) is pure red, and (128,200,128) is a shade of green.

In this exercise we will map (r,g,b) values to their color names. For example, we want (255,0,0) to be classified as red.

To make things a bit easier, we focus on the part of colour space where g=0, i.e. there is no green component. This means the feature space is all combinations (r,0,b), where r and b are between 0 and 255. For orientation, here is a visualisation of some of these colors, with each circle having the 3 color of its r-b-coordinate. We can clearly see that the coordinate (255,0,0) corresponds to bright red. Our goal is to create a classification algorithm that takes an r value, and a b value, and outputs the name of the color this corresponds to. Thankfully, we have a dataset where some of these labels have been entered.

Q1

Load the dataset colors_train.csv. How many classes are there in the dataset?

```r
# Load the color training dataset
colors_train <- read.csv("colors_train.csv")
colors_test <- read.csv("colors_test.csv")

# Display basic information about the dataset
cat("Color training dataset dimensions:", dim(colors_train), "\n")
```

```
## Color training dataset dimensions: 400 3
```

```r
cat("Column names:", names(colors_train), "\n")
```

```
## Column names: r b color
```

```r
# Check how many classes are in the dataset
unique_colors <- unique(colors_train$color)
cat("\nNumber of classes:", length(unique_colors), "\n")
```

```
##
## Number of classes: 5
```

```r
cat("Color classes:", paste(unique_colors, collapse = ", "), "\n")
```

```
## Color classes: pink, purple, red, brown, blue
```

```r
# Check class distribution
cat("\nClass distribution:\n")
```

```
##
## Class distribution:
```

```r
table(colors_train$color)
```

```
##
##    blue   brown    pink purple     red
##      77      29      81    150      63
```

```r
# Convert color to factor
colors_train$color <- as.factor(colors_train$color)
colors_test$color <- as.factor(colors_test$color)
```

Q2

Fit a QDA algorithm to this classification problem and compute the confusion matrix.

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Confusion Matrix and Statistics
##
##           Reference
## Prediction blue brown pink purple red
##     blue     15     0    0      0   0
##     brown     0    10    0      1   0
##     pink      0     0   22      4   0
##     purple    0     0    0     31   0
##     red       0     0    0      0  17
##
## Overall Statistics
##
##                Accuracy : 0.95
##                  95% CI : (0.8872, 0.9836)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.935
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: blue Class: brown Class: pink Class: purple
## Sensitivity                 1.00       1.0000      1.0000        0.8611
## Specificity                 1.00       0.9889      0.9487        1.0000
## Pos Pred Value              1.00       0.9091      0.8462        1.0000
## Neg Pred Value              1.00       1.0000      1.0000        0.9275
## Prevalence                  0.15       0.1000      0.2200        0.3600
## Detection Rate              0.15       0.1000      0.2200        0.3100
## Detection Prevalence        0.15       0.1100      0.2600        0.3100
## Balanced Accuracy           1.00       0.9944      0.9744        0.9306
##                      Class: red
## Sensitivity                1.00
## Specificity                1.00
## Pos Pred Value             1.00
## Neg Pred Value             1.00
## Prevalence                 0.17
## Detection Rate             0.17
## Detection Prevalence       0.17
## Balanced Accuracy          1.00
```

Q3

Visualise the decision boundaries in a suitable way

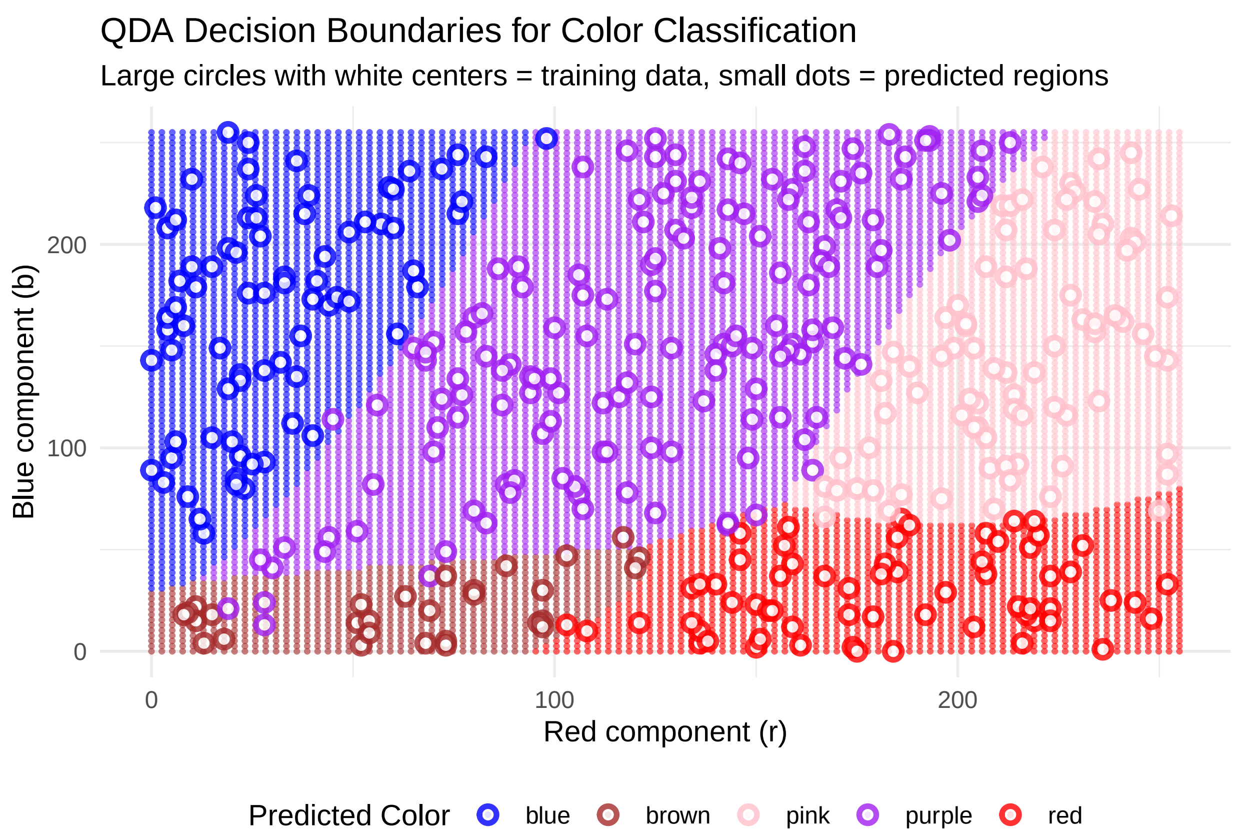
```r
library(ggplot2)

# Create a grid of points to visualize decision boundaries
r_seq <- seq(0, 255, length.out = 100)
b_seq <- seq(0, 255, length.out = 100)
grid <- expand.grid(r = r_seq, b = b_seq)

# Predict colors for the grid
grid_pred <- predict(qda_model, grid)
grid$predicted_color <- grid_pred$class

# Create the decision boundary plot
p1 <- ggplot() +
  geom_point(data = grid, aes(x = r, y = b, color = predicted_color),
             size = 0.5, alpha = 0.6) +
  geom_point(data = colors_train, aes(x = r, y = b, color = color),
             size = 2, alpha = 0.8, shape = 21, fill = "white", stroke = 1.5) +
  scale_color_manual(values = c("blue" = "blue", "brown" = "brown",
                                "pink" = "pink", "purple" = "purple", "red" = "red")) +
  labs(title = "QDA Decision Boundaries for Color Classification",
       subtitle = "Large circles with white centers = training data, small dots = predicted regions",
       x = "Red component (r)",
       y = "Blue component (b)",
       color = "Predicted Color") +
  theme_minimal() +
  theme(legend.position = "bottom")

print(p1)
```



Q4

Test your algorithm on (200,0,200). What colour is this being called by your algorithm?

```r
# Test the algorithm on the specific point (200, 0, 200)
test_point <- data.frame(r = 200, b = 200)

# Make prediction
test_pred <- predict(qda_model, test_point)
predicted_color <- test_pred$class

cat("Prediction for point (200, 0, 200):\n")
```

```
## Prediction for point (200, 0, 200):
```

```r
cat("Predicted color:", as.character(predicted_color), "\n")
```

```
## Predicted color: pink
```