Assignment 2

Lam | ID:75381613

2024-03-27

Question 1

The fitted logistic regression model can be written as follow:

$$Pr(y = 1 \mid X_1 = x_1, X_2 = x_2) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)} = \frac{\exp(-16 + 1.4x_1 + 0.3x_2)}{1 + \exp(-16 + 1.4x_1 + 0.3x_2)}$$

a.

Given $x_1 = 5, x_2 = 36$

Using the equation gives:

```
exp(-16 + 1.4*5 + 0.3*36) / (1 + exp(-16 + 1.4*5 + 0.3*36))
```

```
## [1] 0.8581489
```

So the probability of a student getting a GPA value >= 7 in STAT318 if they study for 5 hours per week and attend all 36 classes is **0.85**.

b.

So we are given:

$$0.5 = \frac{\exp(-16 + 1.4 * x_1 + 0.3 * 18)}{1 + \exp(-16 + 1.4 * x_1 + 0.3 * 18)}$$

Set $u = 1.4 * x_1 - 10.6$

We have:

$0.5 = \frac{\exp(u)}{1+\exp(u)}$

$0.5 + 0.5 * \exp(u) = \exp(u)$

$\frac{1}{2} * \exp(u) = \frac{1}{2}$

$\exp(u) = 1$

$u = ln(1) = 0$

$\Rightarrow 1.4 * x_1 - 10.6 = 0$

$\Rightarrow x_1 = \frac{10.6}{1.4} = 7.57$

So if a student attends 18 classes, to have a 50% chance of getting a GPA value >= 7 in STAT318, they need to spend around **7.5 hours** study time per week.

Question 2

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(caret)      # Provides useful confusion matrix function
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

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

```
## Warning: package 'lattice' was built under R version 4.3.3
```

```
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(MASS)       # For LDA and QDA
```

```
## Warning: package 'MASS' was built under R version 4.3.3
```

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

```
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 4.3.3
```

```
##
## Attaching package: 'yardstick'
##
## The following objects are masked from 'package:caret':
```

```
##
##     precision, recall, sensitivity, specificity
##
## The following object is masked from 'package:readr':
##
##     spec
```

```
bank_train <- read_csv('BankTrain.csv')
```

```
## Rows: 960 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): x1, x2, x3, x4, y
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bank_test <- read_csv('BankTest.csv')
```

```
## Rows: 412 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): x1, x2, x3, x4, y
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

a.

```
glm_model <- glm(data = bank_train, y~ x1 + x3 ,family = binomial)
glm_model %>% summary
```

```
##
## Call:
## glm(formula = y ~ x1 + x3, family = binomial, data = bank_train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22041    0.11206   1.967   0.0492 *
## x1          -1.31489    0.08822 -14.905  < 2e-16 ***
## x3          -0.21738    0.02880  -7.548 4.42e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1322.01  on 959  degrees of freedom
## Residual deviance:  572.07  on 957  degrees of freedom
## AIC: 578.07
##
## Number of Fisher Scoring iterations: 6
```

Observation

- The associate P-values for our predictor x1 and x3 are both very small indicating they have significant impact in predicting y.
- The model can be written as the following equation:

$$logit(Pr(Y = 1 \mid X_1 = x_1, X_3 = x_3)) = -1.31x_1 - 0.21x_3 + 0.22$$

- The negative coefficients indicate that if we have a positive x_1 (variance of a Wavelet Transformed image) or a positive x_2 (kurtosis of a Wavelet Transformed image) then it is less likely that a given banknote is a forged banknote.

b.

```
# Obtain the predicted probability on the training set
glm_predicted_prob <- predict(glm_model, type='response')
glm_predicted_prob %>% head()
```

```
##           1           2           3           4           5           6
## 0.033262089 0.010027942 0.812170652 0.025108954 0.006453763 0.007459787
```

```
# Obtain predictions on training set using .5 boundary
pred_training <- ifelse(glm_predicted_prob > 0.5, yes = 1, no = 0)
pred_training %>% head()
```

```
## 1 2 3 4 5 6
## 0 0 1 0 0 0
```

```
table(pred_training, bank_train$y)
```

```
##
## pred_training   0   1
##             0 455  45
##             1  71 389
```

```
# Create a function to obtain predictions for different boundary and return a confusion matrix
get_confusion_matrix <- function(glm_model ,boundary, test_x = NULL, test_y = NULL){
  pred_probability <- predict(glm_model, test_x, type='response')
  predictions <- ifelse(pred_probability > boundary, 1, 0)
  predictions = factor(predictions)
  test_y <- factor(test_y)
  confusion_matrix <- confusionMatrix(data = predictions, reference = test_y, positive= "1")
  return(list(confusion_matrix = confusion_matrix, predictions = predictions))
}
```

i. Using 0.5 as decision boundary So if prediction probability of an observation is > 0.5 then it will be classified as forged banknote and if $<= 0.5$ it will be classified as a genuine banknote.

```r
test_data <- bank_test %>% dplyr::select(x1, x3)

# Obtain confusion matrix using boundary 0.5
matrix_0.5 <- get_confusion_matrix(glm_model, 0.5, test_data, bank_test$y)
matrix_0.5$confusion_matrix$table
```

```
##           Reference
## Prediction   0   1
##          0 204  24
##          1  32 152
```

From the confusion matrix:

- **TP**: 152 - the model correctly predicted 152 instances of the positive class (forged banknote).
- **TN**: 204 - the model correctly predicted 204 instances of negative class (genuine banknote).
- **FP**: 32- the model predicted positive but the true classes are not.
- **FN**: 24- the model predicted negative but the true classes are not.

```r
# model accuracy
(204+152) / 412
```

```
## [1] 0.8640777
```

The accuracy of the model is 0.86 meaning that our model has correctly classify the banknotes 86% of the time.

```r
matrix_0.3 <- get_confusion_matrix(glm_model, 0.3, test_data, bank_test$y)
matrix_0.6 <- get_confusion_matrix(glm_model, 0.6, test_data, bank_test$y)

matrix_0.3$confusion_matrix$table
```

ii. 0.3 and 0.6 decision boundaries

```
##           Reference
## Prediction   0   1
##          0 183   5
##          1  53 171
```

```r
matrix_0.6$confusion_matrix$table
```

```
##           Reference
## Prediction   0   1
##          0 210  35
##          1  26 141
```

```
accuracy_0.3 = (183 + 171) / 412
accuracy_0.6 = (210 + 141) / 412

paste('Accuracy of model using threshold 0.3 is: ', accuracy_0.3)
```

```
## [1] "Accuracy of model using threshold 0.3 is:  0.859223300970874"
```

```
paste('Accuracy of model using threshold 0.6 is: ', accuracy_0.6)
```

```
## [1] "Accuracy of model using threshold 0.6 is:  0.851941747572815"
```

- The accuracy of the model using 0.3 and 0.6 are not big of a difference (0.8592 vs 0.8519).
- The TP of model_0.6 is lower than model_0.3
- The TN of model_0.6 is higher than model_0.3
- Model_0.6 has lower FP and higher FN than model_0.3

```
paste('Precision: ', matrix_0.3$confusion_matrix$byClass['Precision'])
```

```
## [1] "Precision:  0.763392857142857"
```

```
paste('Recall: ', matrix_0.3$confusion_matrix$byClass['Recall'])
```

```
## [1] "Recall:  0.971590909090909"
```

```
paste('Precision: ', matrix_0.6$confusion_matrix$byClass['Precision'])
```

```
## [1] "Precision:  0.844311377245509"
```

```
paste('Recall: ', matrix_0.6$confusion_matrix$byClass['Recall'])
```

```
## [1] "Recall:  0.801136363636364"
```

So in general, the **Recall** of model_0.3 is very high **0.97** this means that the model has correctly captured 97% of the actual positive class (in context forged note). But there is a trade off, the precision of the model also decreases, that means the model also produces more False Positive.

In context, the model using threshold 0.3 is will correctly capture 97% of the forged bank note. But also will more often wrongly predict genuine notes as forged. The model will be preferred in a situation which it is very important to capture all forged notes. Such as in a bank when customer deposit cash to the bank. Yes, the model will sometimes catches genuine notes as forged but at least we will capture all (97%) the forged notes.

Question 3

```
# Fit LDA model
lda_model <- lda(y ~ x1 + x3, data=bank_train)
lda_model
```

a. Linear Discriminant Analysis

```
## Call:
## lda(y ~ x1 + x3, data = bank_train)
##
## Prior probabilities of groups:
##         0         1
## 0.5479167 0.4520833
##
## Group means:
##          x1       x3
## 0  2.322977 0.938296
## 1 -1.870594 2.114927
##
## Coefficients of linear discriminants:
##            LD1
## x1 -0.55425154
## x3 -0.07209638
```

```
# Compute predictions
lda_pred <- predict(lda_model, test_data)
lda_pred %>% names()
```

```
## [1] "class"     "posterior" "x"
```

```
# Compute confusion matrix
confusionMatrix(factor(lda_pred$class), factor(bank_test$y), positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 203  22
##          1  33 154
##
##                Accuracy : 0.8665
##                  95% CI : (0.8298, 0.8978)
##     No Information Rate : 0.5728
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7294
##
##  Mcnemar's Test P-Value : 0.1775
##
##             Sensitivity : 0.8750
##             Specificity : 0.8602
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.9022
##              Prevalence : 0.4272
##          Detection Rate : 0.3738
##    Detection Prevalence : 0.4539
##       Balanced Accuracy : 0.8676
##
##        'Positive' Class : 1
##
```

```
qda_model <- qda(y ~ x1 + x3, data=bank_train)
qda_model
```

b. Quadratic Discriminant Analysis

```
## Call:
## qda(y ~ x1 + x3, data = bank_train)
##
## Prior probabilities of groups:
##         0         1
## 0.5479167 0.4520833
##
## Group means:
##          x1       x3
## 0  2.322977 0.938296
## 1 -1.870594 2.114927
```

```
qda_pred <- predict(qda_model, test_data)
qda_pred %>% names()
```

```
## [1] "class"     "posterior"
```

```
confusionMatrix(factor(qda_pred$class), factor(bank_test$y), positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 208  18
##          1  28 158
##
##                Accuracy : 0.8883
##                  95% CI : (0.8539, 0.9171)
##     No Information Rate : 0.5728
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7735
##
##  Mcnemar's Test P-Value : 0.1845
##
##             Sensitivity : 0.8977
##             Specificity : 0.8814
##          Pos Pred Value : 0.8495
##          Neg Pred Value : 0.9204
##              Prevalence : 0.4272
##          Detection Rate : 0.3835
##    Detection Prevalence : 0.4515
##       Balanced Accuracy : 0.8895
##
##        'Positive' Class : 1
##
```

c. Results, comparison and recommendation

- The accuracy of the LDA model is **0.86** and of QDA model is **0.88**.
- Comparing to our logistic regression model (using $\theta = 0.5$ from question 2 LDA model is performing equally well and QDA performs slightly better by correctly classify banknotes 88% of the time.
- It will depend on what we want to achieve to select the best model for this problems. If our goal is to accurately predict or classify banknotes we do not care about interpret-ability of the model then QDA is our best options. However, if we do care about interpretation of the model logistic regression is easier to interpret.
- It is important to keep in mind that we can adjust our decision boundary θ to adjust the model sensitivity and specificity based on our domain knowledge to the problem. Luckily, all of our model can easily be done so to adapt to the problem.
- For this problem I would recommend QDA model, this model will much likely to be used in bank and to detect forged notes during transactions. So it will be important to have a high recall or sensitivity. QDA has sensitivity of 0.89 and also highest in accuracy compared to all model. Moreover, we can fine tuning the model by adjusting the threshold to increase the sensitivity to match our problem just like we did with the logistic model in previous question.