Biostats A3

Ethan Scott

2023-03-26

Loading the Data

```
# Load required R packages
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
library(magrittr)

# Read data from a CSV file
data = read.csv("C:\\Users\\ethan\\Downloads\\Stat 431\\Assignment #3\\data.csv")
head(data)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

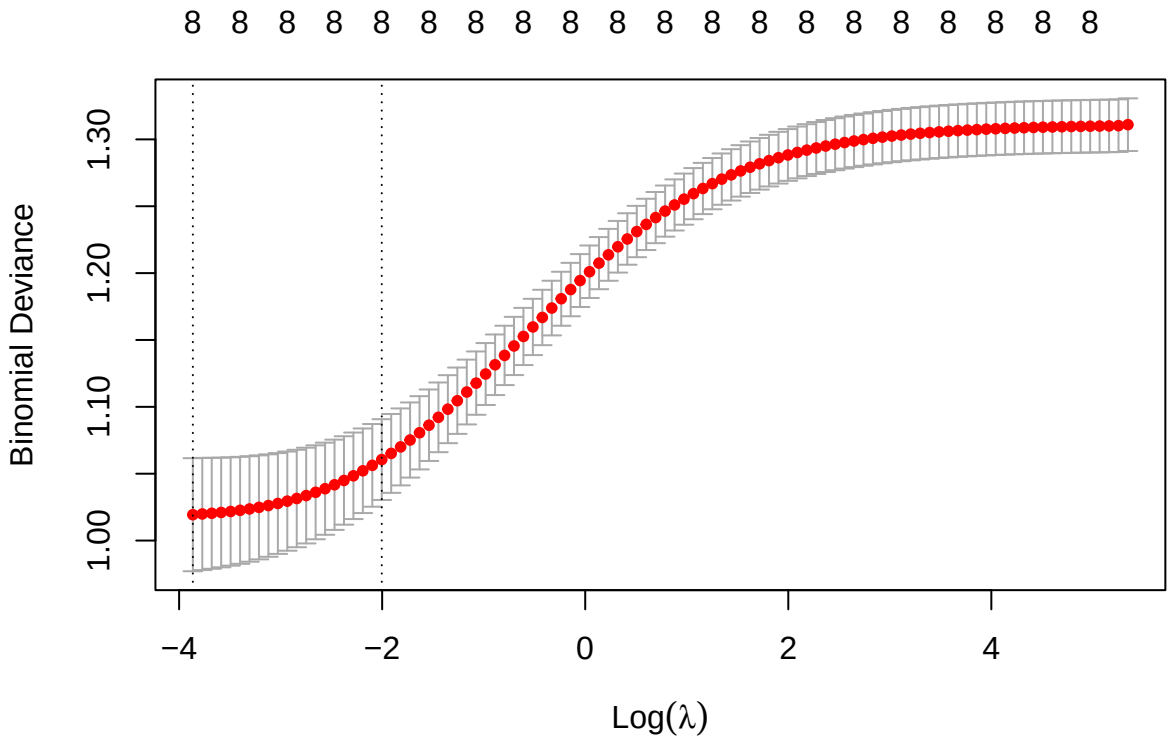
Ridge

```
# Set a random seed for reproducibility
set.seed(1)

# Split data into training and testing sets
n <- nrow(data)
```

```r
training.samples <- sample(seq_len(n), size = floor(0.75 * n))
train.data <- data[training.samples, ]
test.data <- data[-training.samples, ]

# Prepare training data by converting categorical variables to numeric using one-hot encoding
# and separate the outcome variable from the features
x <- model.matrix(Outcome~., train.data)[,-1]
y <- train.data$Outcome

# Perform cross-validation using L2 regularization and the binomial family for logistic regression
cv.ridge <- cv.glmnet(x, y, alpha = 0, family = "binomial")

# Fit a regularized logistic regression model using the lambda value that gives the smallest cross-vali
model <- glmnet(x, y, alpha =0, family = "binomial", lambda = cv.ridge$lambda.min)

# Prepare testing data and compute predicted probabilities of the outcome using the trained model
x.test <- model.matrix(Outcome ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

# Assign predicted outcome based on a threshold of 0.5
predicted.Outcome <- ifelse(probabilities >= 0.5, 1, 0)

# Print the coefficients of the model
coef(model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)              -6.8722710135
## Pregnancies               0.1017124608
## Glucose                   0.0268113310
## BloodPressure            -0.0112346586
## SkinThickness            -0.0039135403
## Insulin                  -0.0006293995
## BMI                       0.0793981585
## DiabetesPedigreeFunction  0.7683238005
## Age                       0.0142895291
```

```r
# Plot the cross-validation errors for different values of lambda
plot(cv.ridge)
```




```
# Print the value of lambda that gives the smallest cross-validation error
cv.ridge$lambda.min
```

```
## [1] 0.02096755
```

```
# Create a confusion matrix
confusion_mat <- table(predicted.Outcome, test.data$Outcome)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(confusion_mat)
```

```
##
## predicted.Outcome   0   1
##                 0 128  32
##                 1   4  28
```

```
# Calculate classification statistics
accuracy <- sum(diag(confusion_mat)) / sum(confusion_mat)
recall <- confusion_mat[2,2] / sum(confusion_mat[2,])
precision <- confusion_mat[2,2] / sum(confusion_mat[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)
```

```r
# Print the summary statistics
print(paste0("Classification accuracy: ", round(accuracy, 3)))
```

```
## [1] "Classification accuracy: 0.812"
```

```r
print(paste0("Recall: ", round(recall, 3)))
```

```
## [1] "Recall: 0.875"
```

```r
print(paste0("Precision: ", round(precision, 3)))
```

```
## [1] "Precision: 0.467"
```

```r
print(paste0("F1 score: ", round(f1_score, 3)))
```

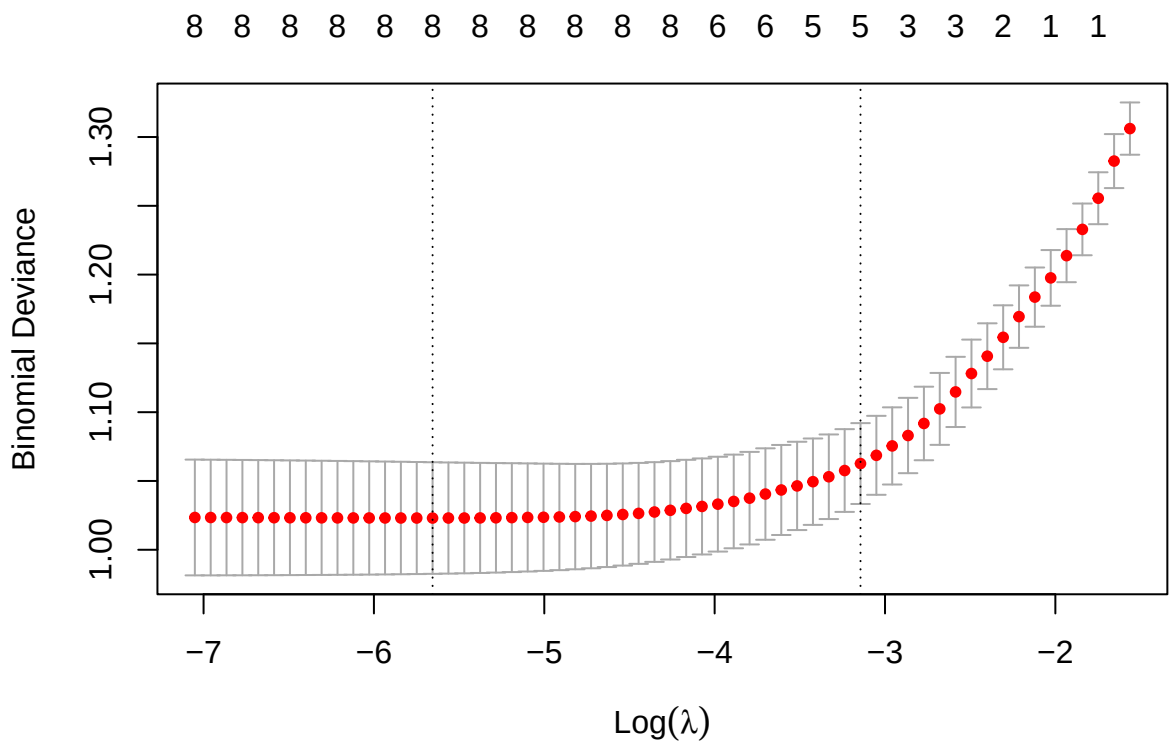
```
## [1] "F1 score: 0.609"
```

LASSO

```r
# Set a random seed for reproducibility
set.seed(1)
# Perform cross-validation using L1 regularization and the binomial family for logistic regression
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")

# Fit a regularized logistic regression model using the lambda value that gives the smallest cross-vali
model <- glmnet(x, y, alpha =1, family = "binomial", lambda = cv.lasso$lambda.min)

# Prepare testing data and compute predicted probabilities of the outcome using the trained model
x.test <- model.matrix(Outcome ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

# Assign predicted outcome based on a threshold of 0.5
predicted.Outcome <- ifelse(probabilities >= 0.5, 1, 0)

# Print the coefficients of the model
coef(model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)              -7.5636864002
## Pregnancies               0.1167031225
## Glucose                   0.0306511471
## BloodPressure            -0.0130476042
## SkinThickness            -0.0035148283
## Insulin                  -0.0009342941
## BMI                       0.0898511316
## DiabetesPedigreeFunction  0.8040466706
## Age                       0.0120986058
```

```
# Plot the cross-validation errors for different values of lambda
plot(cv.lasso)
```




```
# Print the value of lambda that gives the smallest cross-validation error
cv.lasso$lambda.min
```

```
## [1] 0.003497597
```

```
# Create a confusion matrix
confusion_mat <- table(predicted.Outcome, test.data$Outcome)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(confusion_mat)
```

```
##
## predicted.Outcome   0   1
##                 0 126  30
##                 1   6  30
```

```r
# Calculate classification statistics
accuracy <- sum(diag(confusion_mat)) / sum(confusion_mat)
recall <- confusion_mat[2,2] / sum(confusion_mat[2,])
precision <- confusion_mat[2,2] / sum(confusion_mat[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the summary statistics
print(paste0("Classification accuracy: ", round(accuracy, 3)))
```

```
## [1] "Classification accuracy: 0.812"
```

```r
print(paste0("Recall: ", round(recall, 3)))
```

```
## [1] "Recall: 0.833"
```

```r
print(paste0("Precision: ", round(precision, 3)))
```

```
## [1] "Precision: 0.5"
```

```r
print(paste0("F1 score: ", round(f1_score, 3)))
```

```
## [1] "F1 score: 0.625"
```

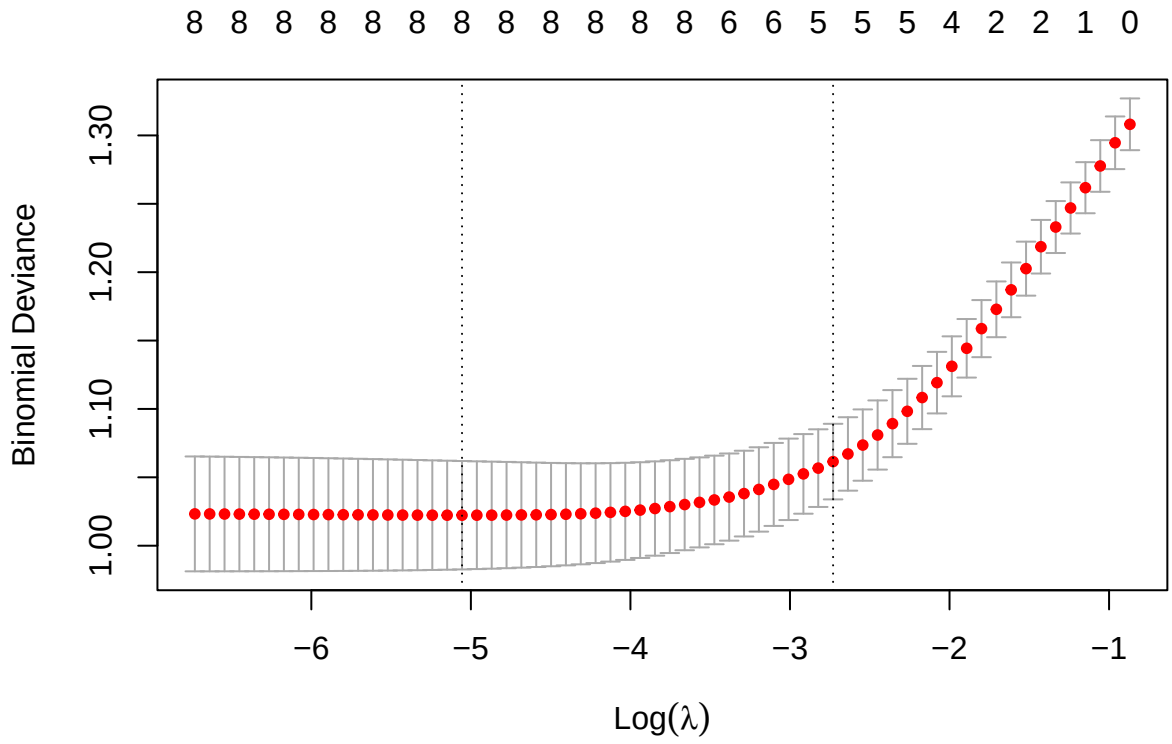
Elastic Net

```r
# Set a random seed for reproducibility
set.seed(1)
# Perform cross-validation using L1 and L2 regularization and the binomial family for logistic regressi
cv.elnet <- cv.glmnet(x, y, alpha = 0.5, family = "binomial")

# Fit an elastic net logistic regression model using the lambda value that gives the smallest cross-val
model <- glmnet(x, y, alpha = 0.5, family = "binomial", lambda = cv.elnet$lambda.min)
# Prepare testing data and compute predicted probabilities of the outcome using the trained model
x.test <- model.matrix(Outcome ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

# Assign predicted outcome based on a threshold of 0.5
predicted.Outcome <- ifelse(probabilities >= 0.5, 1, 0)

# Print the coefficients of the model
coef(model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)             -7.4172873976
## Pregnancies              0.1132654412
## Glucose                  0.0298613940
## BloodPressure           -0.0125664773
```

```
## SkinThickness            -0.0035265608
## Insulin                  -0.0008542789
## BMI                       0.0874545339
## DiabetesPedigreeFunction  0.7915706356
## Age                       0.0124793018
```

```
# Plot the cross-validation errors for different values of lambda
plot(cv.elnet)
```




```
# Print the value of lambda that gives the smallest cross-validation error
cv.elnet$lambda.min
```

```
## [1] 0.006373761
```

```
# Create a confusion matrix
confusion_mat <- table(predicted.Outcome, test.data$Outcome)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(confusion_mat)
```

```
##
```

```
## predicted.Outcome   0   1
##                 0 127  30
##                 1   5  30
```

```
# Calculate classification statistics
accuracy <- sum(diag(confusion_mat)) / sum(confusion_mat)
recall <- confusion_mat[2,2] / sum(confusion_mat[2,])
precision <- confusion_mat[2,2] / sum(confusion_mat[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the summary statistics
print(paste0("Classification accuracy: ", round(accuracy, 3)))
```

```
## [1] "Classification accuracy: 0.818"
```

```
print(paste0("Recall: ", round(recall, 3)))
```

```
## [1] "Recall: 0.857"
```

```
print(paste0("Precision: ", round(precision, 3)))
```

```
## [1] "Precision: 0.5"
```

```
print(paste0("F1 score: ", round(f1_score, 3)))
```

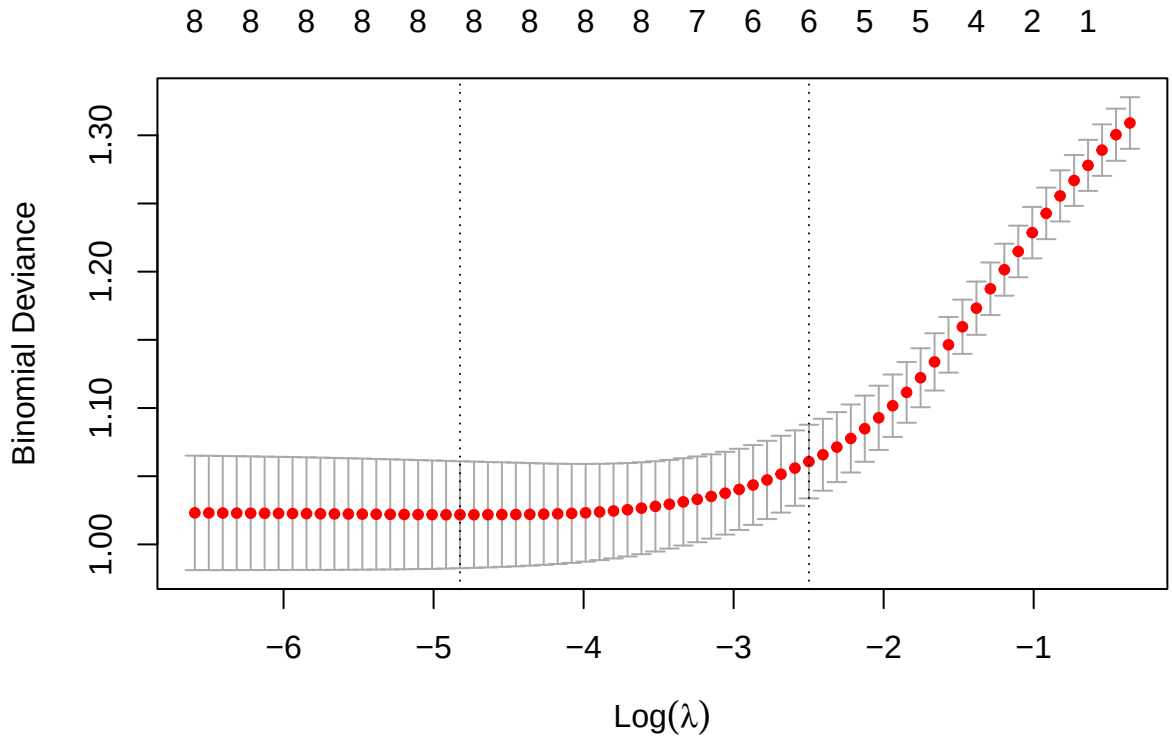
```
## [1] "F1 score: 0.632"
```

```
# Set a random seed for reproducibility
set.seed(1)
#alpha =0.3
# Perform cross-validation using L1 and L2 regularization and the binomial family for logistic regressi
cv.elnet <- cv.glmnet(x, y, alpha = 0.3, family = "binomial")

# Fit an elastic net logistic regression model using the lambda value that gives the smallest cross-val
model <- glmnet(x, y, alpha = 0.3, family = "binomial", lambda = cv.elnet$lambda.min)

# Prepare testing data and compute predicted probabilities of the outcome using the trained model
x.test <- model.matrix(Outcome ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

# Assign predicted outcome based on a threshold of 0.5
predicted.Outcome <- ifelse(probabilities >= 0.5, 1, 0)

# Print the coefficients of the model
coef(model)
```

Same as above but changing alpha to 0.3 and 0.7 to see changed

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                -7.3575625210
## Pregnancies                 0.1119613756
## Glucose                     0.0294675128
## BloodPressure              -0.0125114600
## SkinThickness              -0.0036960927
## Insulin                    -0.0008349982
## BMI                         0.0867540678
## DiabetesPedigreeFunction    0.7953916376
## Age                         0.0129065276
```

```
# Plot the cross-validation errors for different values of lambda
plot(cv.elnet)
```




```
# Print the value of lambda that gives the smallest cross-validation error
cv.elnet$lambda.min
```

```
## [1] 0.008035861
```

```
# Create a confusion matrix
confusion_mat <- table(predicted.Outcome, test.data$Outcome)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(confusion_mat)
```

```
##
## predicted.Outcome   0   1
##                 0 127  30
##                 1   5  30
```

```r
# Calculate classification statistics
accuracy <- sum(diag(confusion_mat)) / sum(confusion_mat)
recall <- confusion_mat[2,2] / sum(confusion_mat[2,])
precision <- confusion_mat[2,2] / sum(confusion_mat[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the summary statistics
print(paste0("Classification accuracy: ", round(accuracy, 3)))
```

```
## [1] "Classification accuracy: 0.818"
```

```r
print(paste0("Recall: ", round(recall, 3)))
```

```
## [1] "Recall: 0.857"
```

```r
print(paste0("Precision: ", round(precision, 3)))
```

```
## [1] "Precision: 0.5"
```

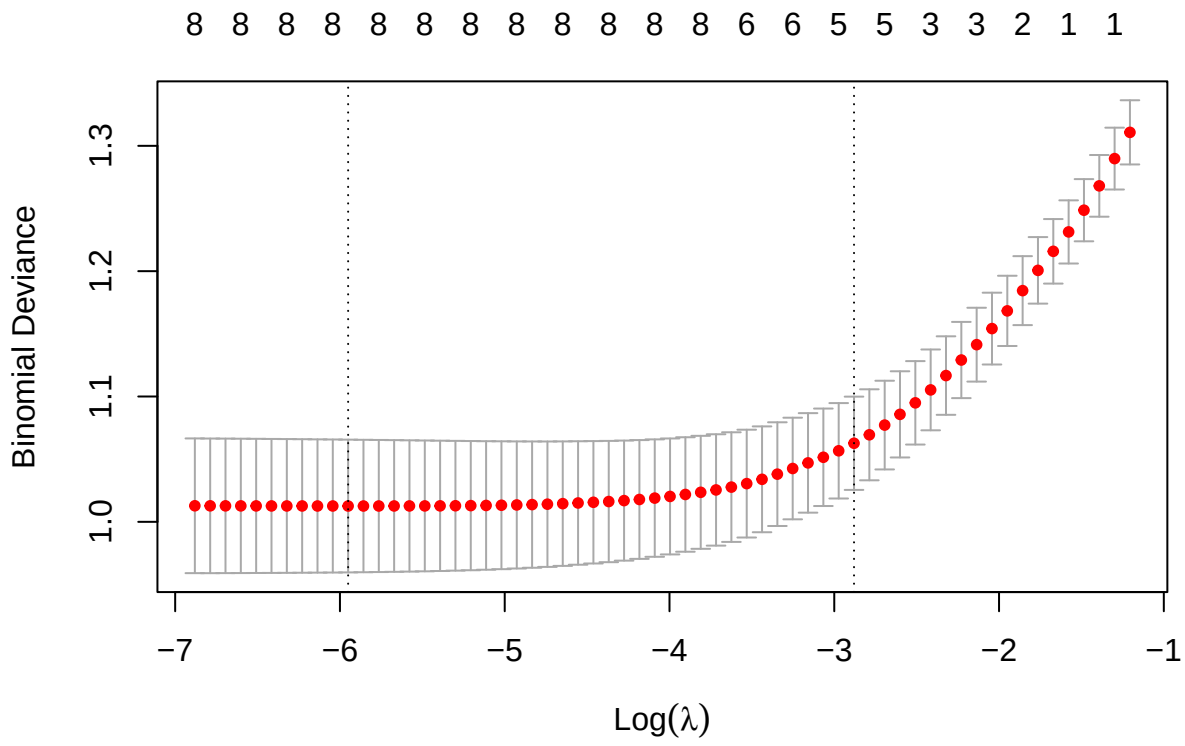
```r
print(paste0("F1 score: ", round(f1_score, 3)))
```

```
## [1] "F1 score: 0.632"
```

```r
##############################################################################
#With alpha 0.7
# Perform cross-validation using L1 and L2 regularization and the binomial family for logistic regressi
cv.elnet <- cv.glmnet(x, y, alpha = 0.7, family = "binomial")

# Fit an elastic net logistic regression model using the lambda value that gives the smallest cross-val
model <- glmnet(x, y, alpha = 0.7, family = "binomial", lambda = cv.elnet$lambda.min)

# Prepare testing data and compute predicted probabilities of the outcome using the trained model
x.test <- model.matrix(Outcome ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

# Assign predicted outcome based on a threshold of 0.5
predicted.Outcome <- ifelse(probabilities >= 0.5, 1, 0)

# Print the coefficients of the model
coef(model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                -7.670697016
## Pregnancies                 0.119310267
## Glucose                     0.030989977
## BloodPressure              -0.013813699
## SkinThickness              -0.004014275
## Insulin                    -0.001026494
## BMI                         0.092429550
## DiabetesPedigreeFunction    0.840802880
## Age                         0.012710730
```

```
# Plot the cross-validation errors for different values of lambda
plot(cv.elnet)
```




```
# Print the value of lambda that gives the smallest cross-validation error
cv.elnet$lambda.min
```

```
## [1] 0.002605215
```

```
# Create a confusion matrix
confusion_mat <- table(predicted.Outcome, test.data$Outcome)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(confusion_mat)
```

```
##
## predicted.Outcome   0   1
##                 0 125  30
##                 1   7  30
```

```r
# Calculate classification statistics
accuracy <- sum(diag(confusion_mat)) / sum(confusion_mat)
recall <- confusion_mat[2,2] / sum(confusion_mat[2,])
precision <- confusion_mat[2,2] / sum(confusion_mat[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the summary statistics
print(paste0("Classification accuracy: ", round(accuracy, 3)))
```

```
## [1] "Classification accuracy: 0.807"
```

```r
print(paste0("Recall: ", round(recall, 3)))
```

```
## [1] "Recall: 0.811"
```

```r
print(paste0("Precision: ", round(precision, 3)))
```

```
## [1] "Precision: 0.5"
```

```r
print(paste0("F1 score: ", round(f1_score, 3)))
```

```
## [1] "F1 score: 0.619"
```