Binary Classification on the Breast Cancer Diagnostic Data

Simple Logistic Regression, Ridge Regression & LASSO Regression

Makayla Tang

2023 Feb

Abstract

In this analysis, we perform binary classification on the Breast Cancer Wisconsin (Diagnostic) Data Set in the csv file wdbc.data. The dataset describes characteristics of the cell nuclei present in n (sample size) images. Each image has multiple attributes, which are described in detail in wdbc.names. We predict the attribute in column 2, which we denote by Y, given the columns {3,4,...,32}, which we denote by X1,...,X30. The variable Y represents the diagnosis (M = malignant, B = benign).

1.Data exploration + Simple Logistic Regression

a.Describe the data: sample size n, number of predictors p, and number of observations in each class.

- n(sample size): 569
- number of predictors: 30
- number of observations in each class: M=212; B=357

```
##------------------------------------
# a.
wdbc %>% summarise(n=n())
wdbc %>% group_by(V2) %>% summarise(n=n(), )
```

b.Divide the data into training set of 400 observations, and a test set; Set the seed with set.seed(2) before you sample the training set.

```
##------------------
# b.
set.seed(2)
train_id_wdbc = sample(nrow(wdbc), 400)
train_wdbc = wdbc[train_id_wdbc, ]
test_wdbc  = wdbc[-train_id_wdbc, ]
```

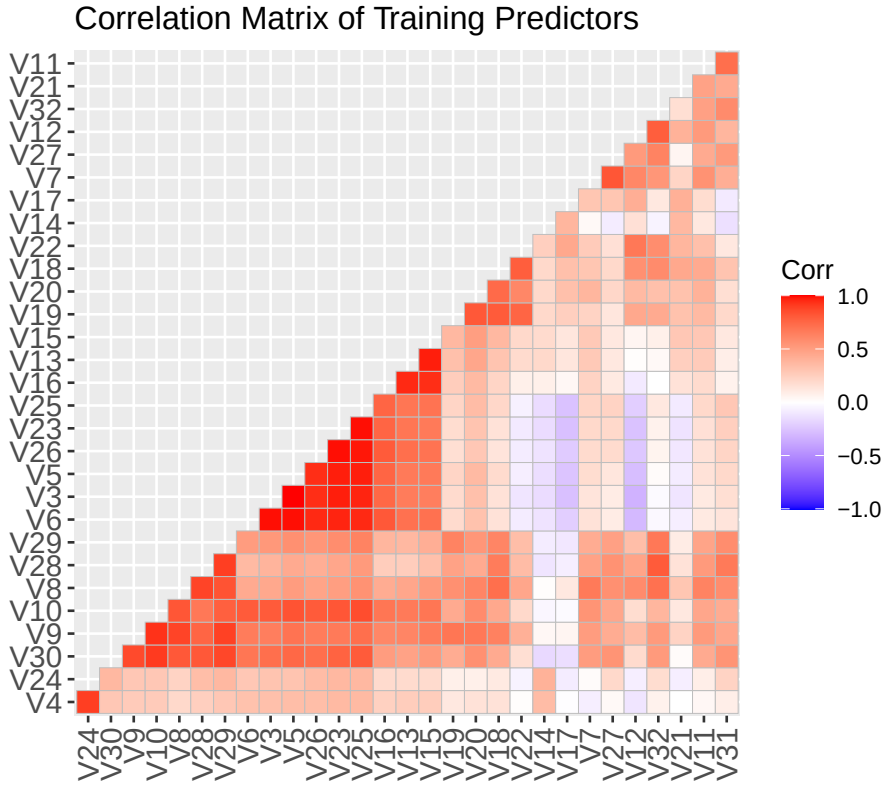
c.Normalize the predictors (for each variable Xj remove the mean and make each variable's standard deviation 1) Explain why you should perform this step separately in the training set and test set.

We performed this step separately in the training set and test set because we don't want to use information from the test set to normalize the predictors in the training set. It may cause overfitting, where the model performs well on the test set only because it was tuned to the test set. By normalizing the predictors separately in the training and test set, we can ensure that the model is not biased towards the training or test set.

```
##------------------
# c.
X_train_norm <- scale(train_wdbc[, 3:32])
sd(X_train_norm[,2])  # check sd=1
#
X_test_norm <- scale(test_wdbc[, 3:32])
sd(X_test_norm[,2])   # check sd=1
```

d.Compute the correlation matrix of your training predictors and plot it. Inspect the correlation matrix and explain what type of challenges this dataset may present?

According the plot, we can see that there are several high positive correlations between the two predictors in the training set, which may lead to multicollinearity in the logistic regression model. It may cause unstable and unreliable estimates of the regression coefficients.



e.Fit a (simple) logistic regression model to predict Y given X1,. . . ,X30. Inspect and report the correlation between the variables X1 and X3; and the magnitude of their coefficient estimates beta1, beta3 wrt to the other coefficients of the model. Comment on their values and relate this to what we have seen in class.

Correlation between the variables X1 and X3 is 0.9997144.

$\beta_1 = -862.997$

$\beta_3 = 989.130$

The estimated magnitude of these two coefficients are far larger than the others. This suggests that they have a strong association with the dependent variable. The correlation between the variables X1 and X3 is closer to 1 indicating they are strongly related to each other, and this can lead to multicollinearity in the model. That is, it's difficult to interpret whether X1 or X3 affects the most as their effects may be confounded by the presence of each other.

```
##------------------
# e.
train_wdbc <- train_wdbc %>% mutate(label=ifelse(V2=="M", 1, 0))
glm_model <- glm(label ~ ., data = train_wdbc[,3:33], family = binomial(link = "logit"))
summary(glm_model)
# Compute the correlation between X1 and X3
cor(train_wdbc$V3, train_wdbc$V5)
```

f.Use the glm previously fitted and the Bayes rule to compute the pre- dicted outcome Y^ from the associated probability estimates (computed with predict) both on the training and the test set. Then compute the confusion table and prediction accuracy (rate of correctly classified obser- vations) both on the training and test set. Comment on the results.

- Accuracy on training set: 1
- Accuracy on testing set: 0.9585799

The results suggest that the logistic regression model fits the training set perfectly, achieving an accuracy of 1. This indicates that it is likely overfitting to the training set. On the other hand, when fitting on testing set, the accuracy drops to 0.959. This means that the model is not able to generalize well to new data that it has not seen before, which is a common problem with overfitting.

```
##------------------
# f.
# predict on training set
pred_train_prob <- predict(glm_model, type = "response")
pred_train_log <- ifelse(pred_train_prob > 0.5, "M", "B")
# confusion table and prediction accuracy on training set
conf_train_log <- table(pred_train_log, train_wdbc$V2)
accuracy_train_log <- sum(diag(conf_train_log)) / sum(conf_train_log)

# predict on testing set
pred_test_prob <- predict(glm_model, newdata = test_wdbc[,3:32], type = "response")
pred_test_log <- as.factor(ifelse(pred_test_prob > 0.5, "M", "B"))
# confusion table and prediction accuracy on test set
conf_test_log <- table(pred_test_log, test_wdbc$V2)
acccuracy_test_log <- sum(diag(conf_test_log)) / sum(conf_test_log)
```

```
kable(conf_train_log,
      caption = "Confusion Table for training set - Logistic Regression")
```

Table 1: Confusion Table for training set - Logistic Regression

	B	M
B	255	0
M	0	145

```
kable(conf_test_log,
      caption = "Confusion Table for testing set - Logistic Regression")
```

Table 2: Confusion Table for testing set - Logistic Regression

	B	M
B	98	3
M	4	64

2.Ridge Logistic Regression

a.From the normalized training set and validation set, construct a data matrix X (numeric) and an outcome vector y (factor).

```
##----------------------------------- Ridge
# a.
# Construct data matrix X and outcome vector y
X <- X_train_norm
y <- train_wdbc$V2
```

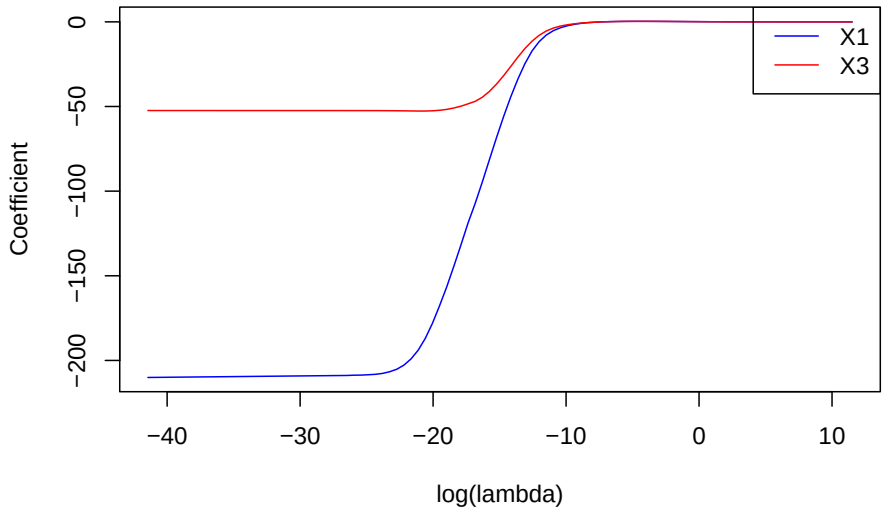
b.On the training set, run a ridge logistic regression model to predict Y given X1,...X30. Use the following grid of values for lambda: 10^seq(5,-18,length=100).

```
##------------------
# b.
grid <- 10^seq(5, -18, length=100)
ridge_model <- glmnet(X, y, alpha=0, lambda = grid, family = "binomial")
summary(ridge_model)
```

c.Plot the values of the coefficients beta1, beta3 (y-axis) in function of log(lambda) (x-axis). Comment on the result.

According to the plot, we can see that the magnitude of both coefficients decreases as log(lambda) increases. This is a common property of ridge regression, where the penalty term of the ridge regression shrinks the magnitude of the coefficients towards zero. The plot also shows that the coefficient for X1 starts out larger than the coefficient for X3, then towards zero as log(lambda) increases. This suggests that X1 may be more important for predicting the outcome than X3, at least for small values of lambda.

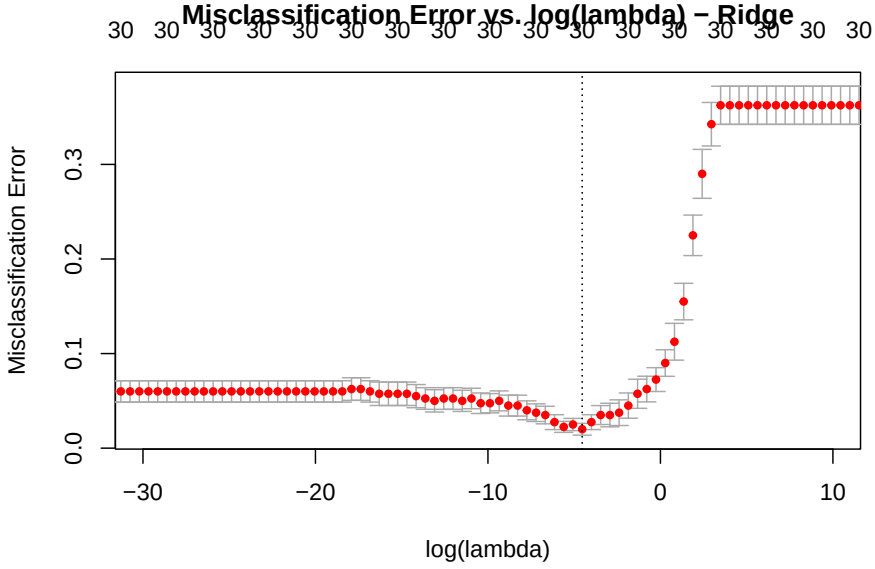
```
##------------------
# c.
coef1 <- ridge_model$beta[1,]
coef3 <- ridge_model$beta[3,]
```



d.Apply 10-fold cross-validation with the previously defined grid of values for lambda. Report the value of lambda that minimizes the CV misclassification error. Refer to it as the optimal lambda. Plot the misclassification error (y-axis) in function of log(lambda) (x-axis). Note: 10-fold cross-validation is the default option in cv.glmnet.

optimal lambda = 0.0022

```
##------------------
# d.
set.seed(2)
X <- as.matrix(X)
cv.out <- cv.glmnet(x=X, y=y, alpha = 0, family = "binomial", type.measure = "class",
                    lambda = 10^seq(5,-18,length=100))
# Get optimal lambda
best_lam_ridge <- cv.out$lambda.min
# 0.002154435
```



e.Report the number of coefficients betaj that are different from 0 for the ridge model with the optimal lambda. Comment on the results.

There are 31 (including intercept) coefficients that are different from 0 for the ridge model with the optimal lambda. It means that the penalty term was not strong enough to push any of the coefficients to zero. This suggests that all the predictors have somehow association with the response variable and are important for making accurate predictions.

```
##------------------
# e.
ridge.coef = predict(ridge_model, type = "coefficients", s = best_lam_ridge)
length(ridge.coef[ridge.coef!=0])
```

f.Use the regularized glm previously fitted (with the optimal lambda) – and the Bayes rule – to compute the predicted outcome Y^ from the associated probability estimates; both on the training and the test set. Then compute the confusion table and prediction accuracy both on the training and test set. Comment on the results.

- Accuracy on training set: 0.9925
- Accuracy on testing set:0.9822485

The model accuracy for training data and the one for testing data are similar and very high. It indicates that a very good model performance.

```
##------------------
# f.
# predict on training set
best_ridge_model <- glmnet(X, y, alpha=0, lambda = best_lam_ridge, family = "binomial")
pred_train_ridge <- predict(best_ridge_model, newx = as.matrix(X_train_norm), type = "response")
pred_train_ridge <- as.factor(ifelse(pred_train_ridge > 0.5, "M", "B"))

# confusion table and prediction accuracy on training set
conf_train_ridge <- table(pred_train_ridge, train_wdbc$V2)
accuracy_train_ridge <- sum(diag(conf_train_ridge)) / sum(conf_train_ridge)

# predict on test set
pred_test_ridge <- predict(best_ridge_model, newx = as.matrix(X_test_norm), type = "response")
pred_test_ridge <- as.factor(ifelse(pred_test_ridge > 0.5, "M", "B"))

# confusion table and prediction accuracy on test set
conf_test_ridge <- table(pred_test_ridge, test_wdbc$V2)
acccuracy_test <- sum(diag(conf_test_ridge)) / sum(conf_test_ridge)
```

```
kable(conf_train_ridge,
      caption = "Confusion Table for training set - Ridge")
```

Table 3: Confusion Table for training set - Ridge

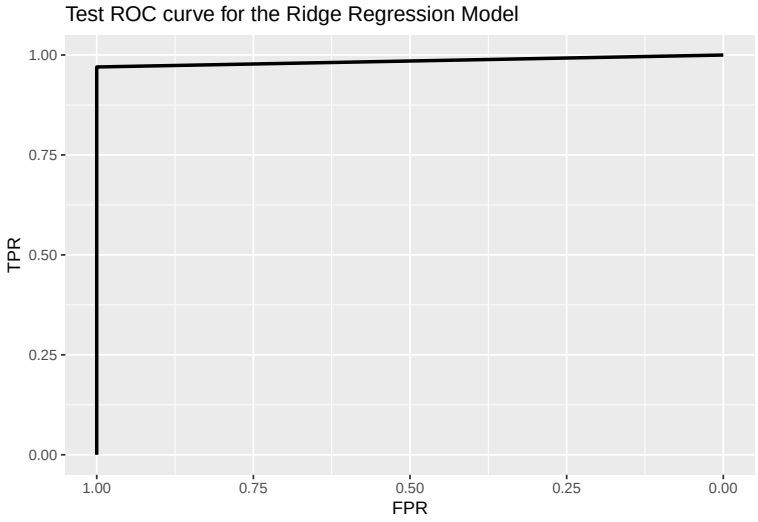
	B	M
B	254	5
M	1	140

```
kable(conf_test_ridge,
      caption = "Confusion Table for testing set - Ridge")
```

Table 4: Confusion Table for testing set - Ridge

	B	M
B	102	2
M	0	65

g.Plot the ROC curve, computed on the test set.



h.Compute an estimate of the Area under the ROC Curve (AUC).

The AUC score is 0.9822485.

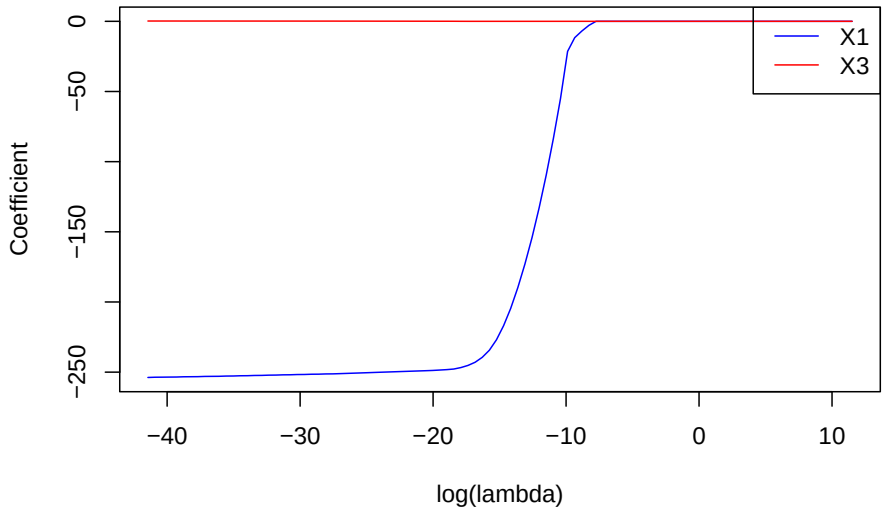
3. Lasso Logistic Regression

a.On the training set, run a ridge logistic regression model to predict Y given X1,...X30. Use the following grid of values for lambda: 10^seq(5,-18,length=100).

```
##---------------------------------- LASSO
# a.
lasso_model <- glmnet(X, y, alpha=1, lambda = grid, family = "binomial")
summary(lasso_model)
```

b.Plot the values of the coefficients beta1, beta3 (y-axis) in function of log(lambda) (x-axis). Comment on the result.

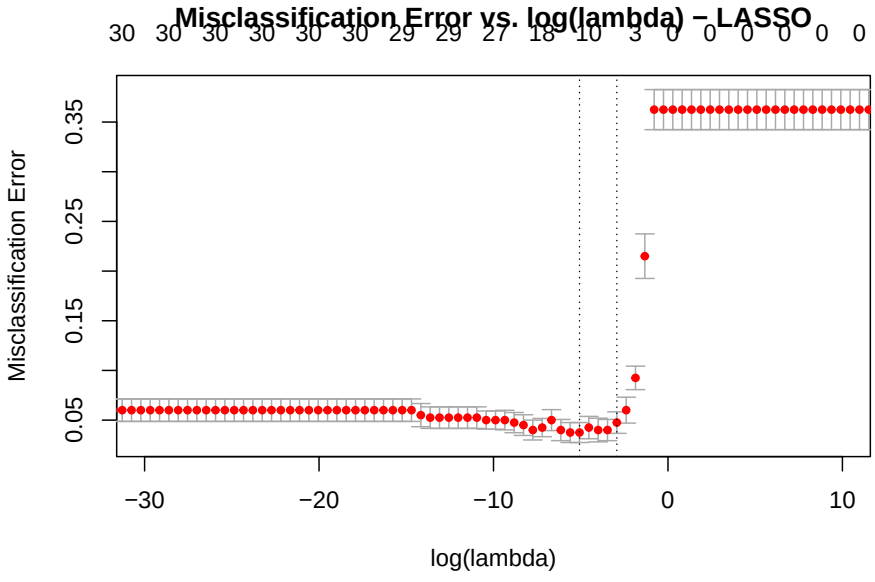
Based on the plot, it appears that the magnitude of the coefficient beta1 decreases as log(lambda) increases, while the coefficient beta3 remains constant and equal to zero, regardless of the value of log(lambda). This suggests that predictor X1 is more important for predicting the outcome than X3 because the coefficient beta1 is sensitive to the regularization parameter lambda, indicating that its contribution to the model is reduced as the regularization strength increases. In contrast, the coefficient beta3 remains at zero, indicating that X3 is not contributing to the model, regardless of the value of lambda.



c.Apply 10-fold cross-validation with the previously defined grid of values for lambda. Report the value of lambda that minimizes the CV misclassification error. Refer to it as the optimal lambda. Plot the misclassification error (y-axis) in function of log(lambda) (x-axis). Note: 10-fold cross-validation is the default option in cv.glmnet.

optimal lambda = 0.0063

```
##------------------
# c.
set.seed(2)
cv.out_LA <- cv.glmnet(x=X, y=y, alpha = 1, family = "binomial", type.measure = "class",
                       lambda = 10^seq(5,-18,length=100))
# Get optimal lambda
best_lam_lasso <- cv.out_LA$lambda.min
# 0.006280291
```



d.Report the number of coefficients betaj that are different from 0 for the ridge model with the optimal lambda. Comment on the results.

There are 16 (including intercept) coefficients that are different from 0 for the LASSO model with the optimal lambda. This suggests that there are half of the total coefficients have been shrunk to exactly zero by the L1 penalty, while the remaining 16 coefficients are considered the most important predictors for the model.

```
##------------------
# d.
lasso.coef = predict(lasso_model, type = "coefficients", s = best_lam_ridge)
length(lasso.coef[lasso.coef!=0])
```

e.Use the regularized glm previously fitted (with the optimal lambda) – and the Bayes rule – to compute the predicted outcome Y^ from the associated probability estimates; both on the training and the test set. Then compute the confusion table and prediction accuracy both on the training and test set. Comment on the results.

- Accuracy on training set: 0.98
- Accuracy on testing set: 0.988

The model accuracy for training data and the one for testing data are similar and very high. It indicates that a very good model performance.

```
##------------------
# e.
# predict on training set
best_lasso_model <- glmnet(X, y, alpha=1, lambda = best_lam_ridge, family = "binomial")
pred_train_lasso <- predict(best_ridge_model, newx = as.matrix(X_train_norm), type = "response")
pred_train_lasso <- as.factor(ifelse(pred_train_lasso > 0.5, "M", "B"))

# confusion table and prediction accuracy on training set
conf_train_lasso <- table(pred_train_lasso, train_wdbc$V2)
accuracy_train_lasso <- sum(diag(conf_train_lasso)) / sum(conf_train_lasso)

# predict on test set
pred_test_lasso <- predict(best_lasso_model, newx = as.matrix(X_test_norm), type = "response")
pred_test_lasso <- as.factor(ifelse(pred_test_lasso > 0.5, "M", "B"))

# confusion table and prediction accuracy on test set
conf_test_lasso <- table(pred_test_lasso, test_wdbc$V2)
acccuracy_test_lasso <- sum(diag(conf_test_lasso)) / sum(conf_test_lasso)
```

```
kable(conf_train_lasso,
      caption = "Confusion Table for training set - LASSO")
```

Table 5: Confusion Table for training set - LASSO

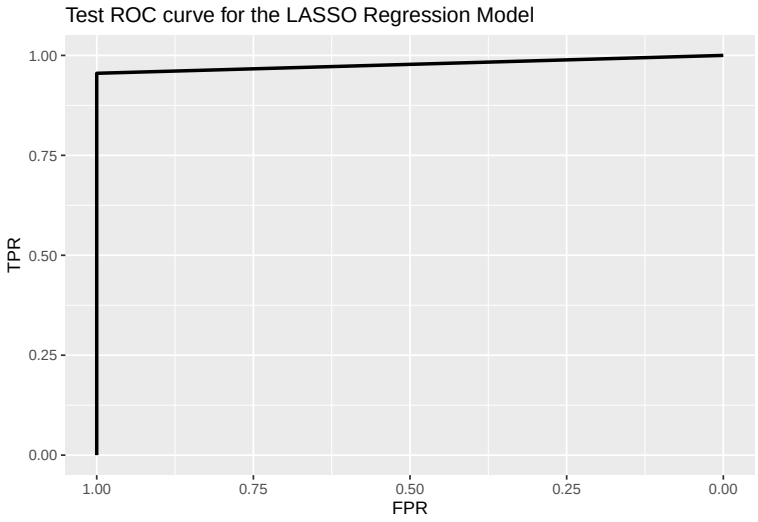
	B	M
B	254	5
M	1	140

```
kable(conf_test_lasso,
      caption = "Confusion Table for testing set - LASSO")
```

Table 6: Confusion Table for testing set - LASSO

	B	M
B	102	3
M	0	64

f.Plot the ROC curve, computed on the test set.



g.Compute an estimate of the Area under the ROC Curve (AUC).

The AUC score is 0.9851.

Discuss the performance

Accuracy (%)	training set	testing set
Logistic Regression	100	95.86
Ridge Regression	99.25	98.22
LASSO Regression	98.0	98.80

This results suggest that logistic regression fits the training set perfectly with accuracy 100%, but it overfits and generalizes poorly to new data, as the accuracy of testing set drops to 95.86%.

In comparison, the Ridge and LASSO regression models fits well on the testing set, with accuracies of testing set 98.22% and 98.8% respectively, while maintaining reasonable accuracy of training set. It indicates that regularization techniques such as Ridge and LASSO can reduce overfitting and improve the generalization performance of the model.

In addition, Ridge regression increase the testing accuracy from 95.86% to 98.22%, while LASSO regression further improves the test accuracy to 98.80%. However, LASSO has a slightly lower training accuracy than Ridge regression, indicating that it have reduced the model complexity more aggressively.

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
## load the libraries
library(ggplot2)
library(caret) # confusionMatrix
library(pROC)  # ROC & AUC
library(dplyr)
library(ggcorrplot)
library(glmnet)  # ridge
library(knitr)

##-------------------------------------
# load the dataset
wdbc <- read.csv("wdbc.data", header = FALSE)
##-------------------------------------
# a.
wdbc %>% summarise(n=n())
wdbc %>% group_by(V2) %>% summarise(n=n(), )
##------------------
# b.
set.seed(2)
train_id_wdbc = sample(nrow(wdbc), 400)
train_wdbc = wdbc[train_id_wdbc, ]
test_wdbc  = wdbc[-train_id_wdbc, ]
##------------------
# c.
X_train_norm <- scale(train_wdbc[, 3:32])
sd(X_train_norm[,2])  # check sd=1
#
X_test_norm <- scale(test_wdbc[, 3:32])
sd(X_test_norm[,2])   # check sd=1
##------------------
# d.
# Compute the correlation matrix
cor <- cor(train_wdbc[, 3:32])
# Plot the correlation matrix using ggcorrplot
ggcorrplot(cor, type = "lower", hc.order = TRUE,
           ggtheme = ggplot2::theme_gray,
           title = "Correlation Matrix of Training Predictors") +
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
##------------------
# e.
train_wdbc <- train_wdbc %>% mutate(label=ifelse(V2=="M", 1, 0))
glm_model <- glm(label ~ ., data = train_wdbc[,3:33], family = binomial(link = "logit"))
summary(glm_model)
# Compute the correlation between X1 and X3
cor(train_wdbc$V3, train_wdbc$V5)
##------------------
# f.
# predict on training set
pred_train_prob <- predict(glm_model, type = "response")
```

```r
pred_train_log <- ifelse(pred_train_prob > 0.5, "M", "B")
# confusion table and prediction accuracy on training set
conf_train_log <- table(pred_train_log, train_wdbc$V2)
accuracy_train_log <- sum(diag(conf_train_log)) / sum(conf_train_log)

# predict on testing set
pred_test_prob <- predict(glm_model, newdata = test_wdbc[,3:32], type = "response")
pred_test_log <- as.factor(ifelse(pred_test_prob > 0.5, "M", "B"))
# confusion table and prediction accuracy on test set
conf_test_log <- table(pred_test_log, test_wdbc$V2)
acccuracy_test_log <- sum(diag(conf_test_log)) / sum(conf_test_log)
kable(conf_train_log,
      caption = "Confusion Table for training set - Logistic Regression")
kable(conf_test_log,
      caption = "Confusion Table for testing set - Logistic Regression")
##----------------------------------- Ridge
# a.
# Construct data matrix X and outcome vector y
X <- X_train_norm
y <- train_wdbc$V2
##------------------
# b.
grid <- 10^seq(5, -18, length=100)
ridge_model <- glmnet(X, y, alpha=0, lambda = grid, family = "binomial")
summary(ridge_model)
##------------------
# c.
coef1 <- ridge_model$beta[1,]
coef3 <- ridge_model$beta[3,]
# Plot coefficients vs log(lambda)
plot(log(grid),coef1, type = "l", col = "blue", xlab = "log(lambda)", ylab = "Coefficient")
lines(log(grid), coef3, col = "red")
legend("topright", legend = c("X1", "X3"), col = c("blue", "red"), lty = 1)
##------------------
# d.
set.seed(2)
X <- as.matrix(X)
cv.out <- cv.glmnet(x=X, y=y, alpha = 0, family = "binomial", type.measure = "class",
                    lambda = 10^seq(5,-18,length=100))
# Get optimal lambda
best_lam_ridge <- cv.out$lambda.min
# 0.002154435
# Plot misclassification error as function of log(lambda)
plot(cv.out, xlab = "log(lambda)", main = "Misclassification Error vs. log(lambda) - Ridge", xlim=c(-30
##------------------
# e.
ridge.coef = predict(ridge_model, type = "coefficients", s = best_lam_ridge)
length(ridge.coef[ridge.coef!=0])
##------------------
# f.
# predict on training set
best_ridge_model <- glmnet(X, y, alpha=0, lambda = best_lam_ridge, family = "binomial")
pred_train_ridge <- predict(best_ridge_model, newx = as.matrix(X_train_norm), type = "response")
```

```r
pred_train_ridge <- as.factor(ifelse(pred_train_ridge > 0.5, "M", "B"))

# confusion table and prediction accuracy on training set
conf_train_ridge <- table(pred_train_ridge, train_wdbc$V2)
accuracy_train_ridge <- sum(diag(conf_train_ridge)) / sum(conf_train_ridge)

# predict on test set
pred_test_ridge <- predict(best_ridge_model, newx = as.matrix(X_test_norm), type = "response")
pred_test_ridge <- as.factor(ifelse(pred_test_ridge > 0.5, "M", "B"))

# confusion table and prediction accuracy on test set
conf_test_ridge <- table(pred_test_ridge, test_wdbc$V2)
acccuracy_test <- sum(diag(conf_test_ridge)) / sum(conf_test_ridge)
kable(conf_train_ridge,
      caption = "Confusion Table for training set - Ridge")
kable(conf_test_ridge,
      caption = "Confusion Table for testing set - Ridge")
##------------------
# g.
# calculate AUC score
# should convert to numeric
pred_test_ridge <- ifelse(pred_test_ridge == "M", 1, 0)
roc_score_ridge = roc(response = test_wdbc$V2, predictor = pred_test_ridge)

## plot the ROC curve
ggroc(roc_score_ridge, linetype=1, size = 1) +
        xlab("FPR") + ylab("TPR") +
        ggtitle("Test ROC curve for the Ridge Regression Model")
##------------------
# h.
roc_score_ridge
# 0.9822485
##------------------------------------ LASSO
# a.
lasso_model <- glmnet(X, y, alpha=1, lambda = grid, family = "binomial")
summary(lasso_model)
##------------------
# b.
coef1_La <- lasso_model$beta[1,]
coef3_La <- lasso_model$beta[3,]
# Plot coefficients vs log(lambda)
plot(log(grid), coef1_La, type = "l", col = "blue", xlab = "log(lambda)", ylab = "Coefficient")
lines(log(grid), coef3_La, col = "red")
legend("topright", legend = c("X1", "X3"), col = c("blue", "red"), lty = 1)
##------------------
# c.
set.seed(2)
cv.out_LA <- cv.glmnet(x=X, y=y, alpha = 1, family = "binomial", type.measure = "class",
                   lambda = 10^seq(5,-18,length=100))
# Get optimal lambda
best_lam_lasso <- cv.out_LA$lambda.min
# 0.006280291
# Plot misclassification error as function of log(lambda)
```

```r
plot(cv.out_LA, xlab = "log(lambda)", main = "Misclassification Error vs. log(lambda) - LASSO", xlim=c(
##------------------
# d.
lasso.coef = predict(lasso_model, type = "coefficients", s = best_lam_ridge)
length(lasso.coef[lasso.coef!=0])
##------------------
# e.
# predict on training set
best_lasso_model <- glmnet(X, y, alpha=1, lambda = best_lam_ridge, family = "binomial")
pred_train_lasso <- predict(best_ridge_model, newx = as.matrix(X_train_norm), type = "response")
pred_train_lasso <- as.factor(ifelse(pred_train_lasso > 0.5, "M", "B"))

# confusion table and prediction accuracy on training set
conf_train_lasso <- table(pred_train_lasso, train_wdbc$V2)
accuracy_train_lasso <- sum(diag(conf_train_lasso)) / sum(conf_train_lasso)

# predict on test set
pred_test_lasso <- predict(best_lasso_model, newx = as.matrix(X_test_norm), type = "response")
pred_test_lasso <- as.factor(ifelse(pred_test_lasso > 0.5, "M", "B"))

# confusion table and prediction accuracy on test set
conf_test_lasso <- table(pred_test_lasso, test_wdbc$V2)
acccuracy_test_lasso <- sum(diag(conf_test_lasso)) / sum(conf_test_lasso)
kable(conf_train_lasso,
      caption = "Confusion Table for training set - LASSO")
kable(conf_test_lasso,
      caption = "Confusion Table for testing set - LASSO")
##------------------
# f.
# calculate AUC score
# should convert to numeric
pred_test_lasso <- ifelse(pred_test_lasso == "M", 1, 0)
roc_score_lasso = roc(response = test_wdbc$V2, predictor = pred_test_lasso)

## plot the ROC curve
ggroc(roc_score_lasso, linetype=1, size = 1) +
        xlab("FPR") + ylab("TPR") +
        ggtitle("Test ROC curve for the LASSO Regression Model")
##------------------
# g.
roc_score_lasso
# 0.9851
```