BIOST 546 HW 3

My-Anh Doan

2023-02-16

```r
# set global options for code chunks
knitr::opts_chunk$set(message = FALSE, warning = FALSE, collapse = TRUE)
knitr::opts_knit$set(root.dir = rprojroot::find_rstudio_root_file())

library(knitr)
library(dplyr)
library(ggplot2)
library(caret) # classification/regression training package; confusion matrices
library(ggcorrplot) # correlation plots package
library(glmnet) # ridge and lasso glm package
library(pROC) # ROC curves package
```

As in HW2, we will 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`. This time, however, you will predict the attribute in column 2, which we denote by Y, given the columns {3, 4, . . . , 32}, which we denote by $X_1, ..., X_{30}$. The variable Y represents the diagnosis (M = malignant, B = benign).

1. Data exploration and Simple Logistic Regression

- 1a. Describe the data: sample size n, number of predictors p, and the number of observations in each class.

```r
# load data
wdbc <- read.csv("./dataset/wdbc.data", header = FALSE, stringsAsFactors = TRUE)
wdbc_set <- wdbc[, -1] %>%
    rename(diagnosis = 1)

# rename columns to predictor labels X1, ..., X30
names <- c("diagnosis")
for (i in 1:ncol(wdbc_set[ , -1])) {
  names[i + 1] <- paste("X", i, sep = "")
}
colnames(wdbc_set) <- names

# check for missing values; if returns 0, no missing data in data set
which(complete.cases(wdbc_set) == FALSE)
## integer(0)

# count number of observations in each diagnosis class
wdbc_summary <- wdbc_set %>%
  count(diagnosis)
kable(wdbc_summary, caption = "Total observations by diagnosis class")
```

Table 1: Total observations by diagnosis class

diagnosis	n
B	357
M	212

The `wdbc` data set has a sample size of $n = 569$ and $p = 30$ predictors. There are 357 observations in the benign class and 212 observations in the malignant class (as shown in **Table 1**).

- 1b. Divide the data into a training set of 400 observations and a test set.

```
set.seed(2)
random_sample <- sample(1:nrow(wdbc_set), size = 400, replace = FALSE)

# split data into training and test sets
wdbc_train <- wdbc_set[random_sample, ]
wdbc_test <- wdbc_set[-random_sample, ]
```

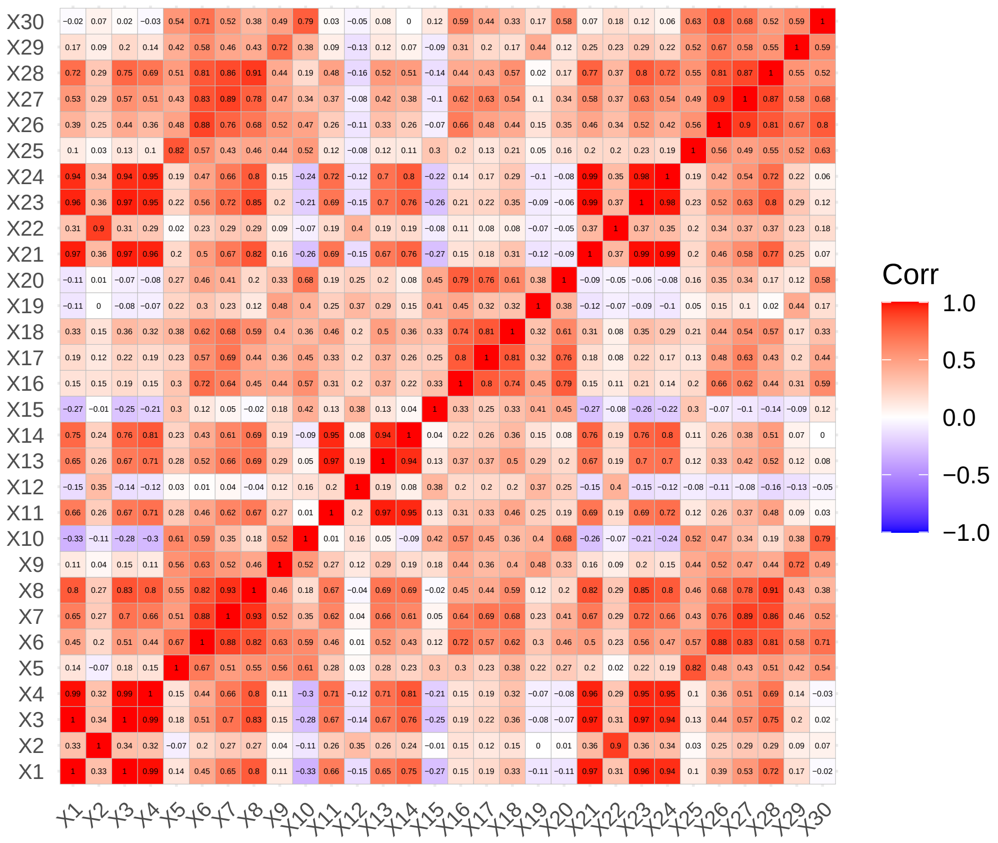
- 1c. Normalize your predictors, i.e. for each variable X_j remove the mean and make each variable's standard deviation 1. Explain why you should perform this step separately in the training set and test set.

```
# normalize predictors such that each variable has mean = 0 and sd = 1
wdbc_train_scaled <- scale(wdbc_train[ , -1])
wdbc_test_scaled <- scale(wdbc_test[ , -1])
```

We must normalize the predictors of the training and test sets separately to avoid having observations in one set affect the other set. If we were to normalize the predictors of the training and test sets together (i.e. normalize before splitting into train-test sets), the values of the test set will have been affected by the normalization of the training set observations. This renders the test set not completely brand new data for us to validate models fitted with the training set, leading to possible bias in our data.

- 1d. Compute the correlation matrix of your training predictors (command `cor`) and plot it (e.g. command `ggcorrplot` in the library `ggcorrplot`). Inspect the correlation matrix and explain what type of challenges this data set may present?

```
ggcorrplot(cor(wdbc_train_scaled),
           lab = "TRUE", lab_size = 1,
           tl.cex = 8)
```



Looking at the correlation matrix for the training predictors in the training data set, it looks like we have a number of variables/predictors that are highly correlated with each other.

For example, X_1 and X_3 are perfectly correlated with each other. There are also high correlations between X_1 and X_{21}, X_{23}, X_{24}.

This introduces multicollinearity to the data, which reduces the precision of the estimated individual predictor coefficients in our regression models.

- 1e. Fit a simple logistic regression model to predict Y given $X_1, ..., X_{30}$.
  Inspect and report the correlation between the variables X_1 and X_3; and the magnitude of their coefficient estimates $\hat{\beta}_1$, $\hat{\beta}_3$ with regard to the other coefficients of the model. Comment on their values and relate this to what we have seen in class.

```
glm_train <- glm(formula = wdbc_train$diagnosis ~ .,
                 family = "binomial",
                 data = as.data.frame(wdbc_train_scaled))

kable(summary(glm_train)$coefficients,
      digits = 3,
      caption = "Training Set: Estimates of Predictor Coefficient")
```

Table 2: Training Set: Estimates of Predictor Coefficient

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	92.461	28281.573	0.003	0.997
X1	-862.997	462270.497	-0.002	0.999
X2	-3.798	13416.446	0.000	1.000
X3	989.130	544553.105	0.002	0.999
X4	-189.766	100538.360	-0.002	0.998
X5	117.778	15574.736	0.008	0.994
X6	-302.085	41791.183	-0.007	0.994
X7	-8.086	39076.942	0.000	1.000
X8	265.443	49259.173	0.005	0.996
X9	-122.564	22339.352	-0.005	0.996
X10	96.252	28237.700	0.003	0.997
X11	559.966	61288.543	0.009	0.993
X12	3.348	9628.505	0.000	1.000
X13	-110.485	70063.531	-0.002	0.999
X14	-540.239	107218.217	-0.005	0.996
X15	31.709	12943.540	0.002	0.998
X16	73.749	13649.606	0.005	0.996
X17	52.341	57193.234	0.001	0.999
X18	90.037	20354.300	0.004	0.996
X19	-93.137	24992.864	-0.004	0.997
X20	-262.422	65894.489	-0.004	0.997
X21	-1030.876	195279.900	-0.005	0.996
X22	105.940	15834.642	0.007	0.995
X23	-502.589	175318.352	-0.003	0.998
X24	2156.153	279421.092	0.008	0.994
X25	-97.368	20531.897	-0.005	0.996
X26	5.471	30882.296	0.000	1.000
X27	106.239	52714.248	0.002	0.998
X28	162.495	25575.212	0.006	0.995
X29	241.662	25846.588	0.009	0.993
X30	-39.356	62867.795	-0.001	1.000

```r
kable(head(summary(glm_train, correlation = TRUE)$correlation)[ , 1:6],
      digits = 3,
      caption = "Training Set: Correlation Values (Partial Table)")
```

Table 3: Training Set: Correlation Values (Partial Table)

	(Intercept)	X1	X2	X3	X4	X5
(Intercept)	1.000	0.119	0.199	-0.126	0.228	-0.469
X1	0.119	1.000	0.034	-0.977	-0.057	-0.134
X2	0.199	0.034	1.000	0.015	0.383	-0.676
X3	-0.126	-0.977	0.015	1.000	-0.084	0.116
X4	0.228	-0.057	0.383	-0.084	1.000	-0.453
X5	-0.469	-0.134	-0.676	0.116	-0.453	1.000

The correlation between variables X_1 and X_3 is -0.9773678. **Table 2** shows that the estimated coefficient values for X_1 and X_3 are $\hat{\beta}_1$ = -862.9974499 and $\hat{\beta}_3$ = 989.1304951, respectively. The coefficient estimates are opposite in sign and nearly equal in magnitude to each other, which makes sense given the correlation between X_1 and X_3 is -0.9773678.

Compared to the other estimated coefficients of the model, the values of $\hat{\beta}_1$ and $\hat{\beta}_3$ are larger compared to the *non-collinear* coefficient estimates (e.g., not compared to X_{21}, X_{23}, X_{24} which are collinear with X_1).

- 1f. Use the glm previously fitted and the Bayes rule to compute the predicted outcome $\hat{Y}$ 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 observations) both on the training and test set. Comment on the results.

```
# Training predictions, confusion table, and prediction accuracy
glm_train_prob <- predict(glm_train, type = "response",
                          as.data.frame(wdbc_train_scaled))
glm_train_class <- ifelse(glm_train_prob > 0.5, "M", "B")

glm_train_matrix <- confusionMatrix(factor(glm_train_class, levels = c("B","M")),
                                    factor(wdbc_train$diagnosis, levels = c("B","M")),
                                    positive = "M")
kable(glm_train_matrix$table, caption = "glm Training Set Confusion Matrix")
```

Table 4: glm Training Set Confusion Matrix

	B	M
B	255	0
M	0	145

```
# Test predictions, confusion table, and prediction accuracy
glm_test_prob <- predict(glm_train, type = "response",
                          as.data.frame(wdbc_test_scaled))
glm_test_class <- ifelse(glm_test_prob > 0.5, "M", "B")

glm_test_matrix <- confusionMatrix(factor(glm_test_class, levels = c("B","M")),
                                   factor(wdbc_test$diagnosis, levels = c("B","M")),
                                   positive = "M")
kable(glm_test_matrix$table, caption = "glm Test Set Confusion Matrix")
```

Table 5: glm Test Set Confusion Matrix

	B	M
B	99	7
M	3	60

The prediction accuracy of the glm model on the training set is 1 while the prediction accuracy in the test set is 0.9408284.

The confusion table from the training set shows that the glm model did not predict any false negatives or false positives. This is not surprising given that the model was fitted using the training data and the training set prediction accuracy of the model is 1. This is also an indication that the model is likely overfitted to the training data.

The confusion table from the test set shows that the glm model would predict more false negatives (i.e., predict `benign` when actually `malignant`; 10.45%) than false positives (i.e., predict `malignant` when actually `benign`; 2.94%).

2. Ridge Logistic Regression

- 2a.From the normalized training set and validation set, construct a data matrix X (numeric) and an outcome vector y (factor).

```
X_train <- as.matrix(wdbc_train_scaled)
y_train <- as.factor(wdbc_train$diagnosis)

X_test <- as.matrix(wdbc_test_scaled)
y_test <- as.factor(wdbc_test$diagnosis)
```

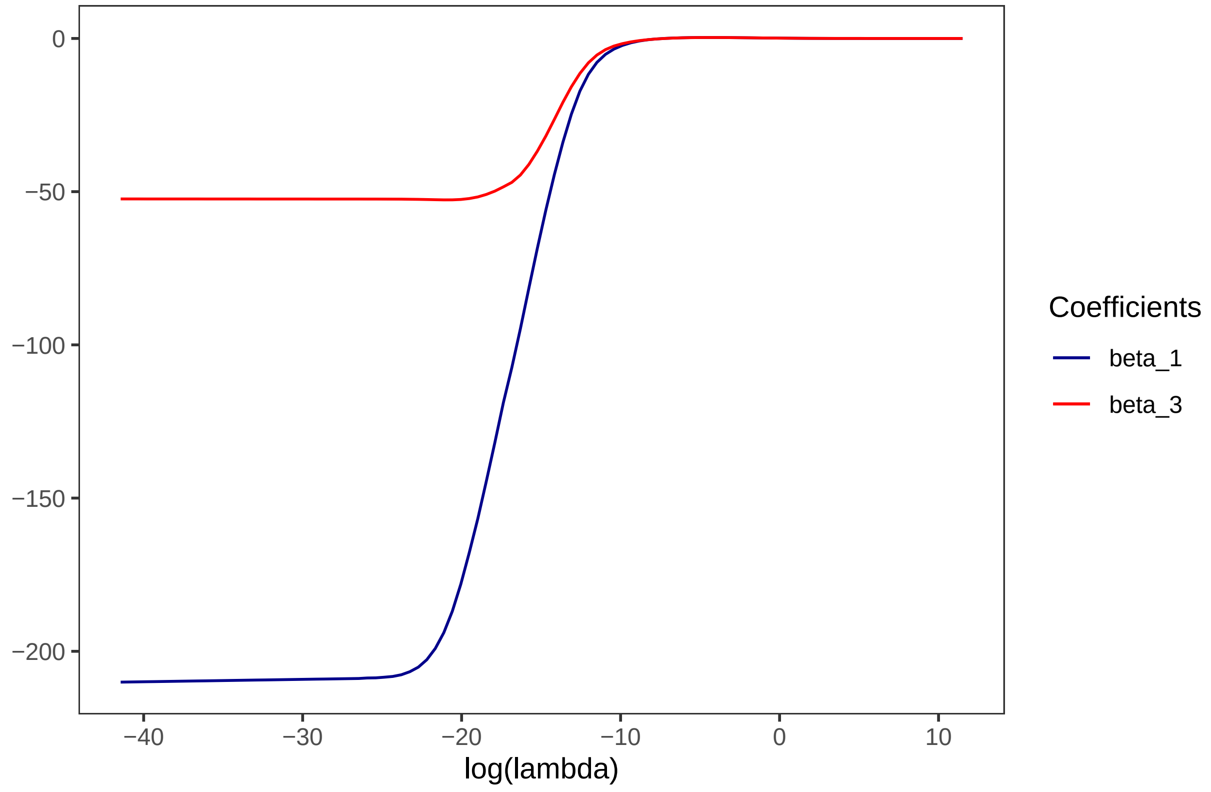
- 2b. On the training set, run a ridge logistic regression model with `glmnet` (with the argument `family = "binomial"`) to predict Y given $X_1, ..., X_{30}$. Use the following grid of values for lambda: `10^seq(5, -18, length = 100)`.

```
lambda_grid <- 10^seq(5, -18, length = 100)

ridge_train <- glmnet(X_train, y_train, alpha = 0, lambda = lambda_grid,
                      family = "binomial")
```

- 2c. Plot the values of the coefficients β_1, β_3 (y-axis) in function of `log(lambda)` (x-axis). Comment on the result.

```
ridge_dat <- data.frame(x = log(ridge_train$lambda),
                        beta_1 = ridge_train$beta[1, ],
                        beta_3 = ridge_train$beta[3, ])

ggplot(data = ridge_dat, aes(x = x)) +
  geom_line(aes(y = beta_1, color = "beta_1")) +
  geom_line(aes(y = beta_3, color = "beta_3")) +
  scale_color_manual(name = "Coefficients",
                     values = c("beta_1" = "darkblue", "beta_3" = "red")) +
  theme_bw() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank()) +
  labs(title = expression(paste("Ridge Logistic Regression Model: Values of ",
                                beta[1], " and ", beta[3], " vs log(lambda)")),
       x = "log(lambda)",
       y = element_blank())
```

Ridge Logistic Regression Model: Values of β_1 and β_3 vs log(lambda)

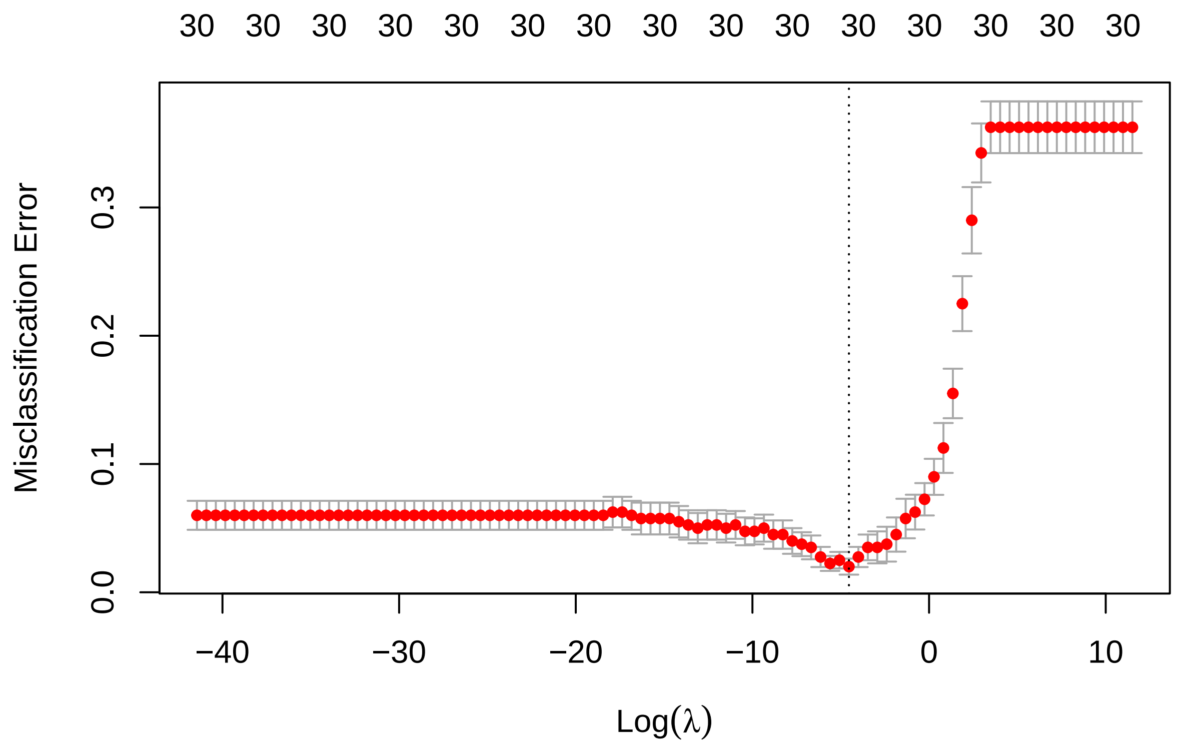


When applying the ridge logistic regression model to the training set, we see that the values of coefficient β_1 are much lower than the values of coefficient β_3 for log(λ) $<$ -10. This indicates that β_1 is more penalized as λ increases compared to β_3 when log(λ) $<$ -10.

The values for both coefficients approaches zero at approximately log(λ) $=$ -10 and plateaus there for log(λ) $>$ -10.

- 2d. 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. We will refer to it as the optimal lambda. Plot the misclassification error (y-axis) in function of `log(lambda)` (x-axis). Use `cv.glmnet` with the arguments `family = "binomial"` and `type.measure = "class"`.

```
set.seed(2)

ridge_train_cv <- cv.glmnet(X_train, y_train, alpha = 0, nfolds = 10,
                            lambda = lambda_grid, family = "binomial",
                            type.measure = "class")
plot(ridge_train_cv)
```



```r
optimal_train_lambda <- ridge_train_cv$lambda.min
```

The optimal value of λ that minimizes the CV misclassification error is 0.0107227.

- 2e. Report the number of coefficients β_j that are different from 0 for the ridge model with the optimal lambda. Comment on the results.

```r
sum(predict(ridge_train, s = optimal_train_lambda, type = "coefficients")[-1, ] != 0)
## [1] 30
```

There are 30 non-zero coefficients for the ridge model using the optimal λ. This makes sense given that coefficients will shrink towards zero asymptotically as λ approaches ∞ (i.e., coefficients will not equal zero when using ridge regression to shrink coefficients).

- 2f. Use the regularized glm previously fitted (with the optimal lambda) and the Bayes rule to compute the predicted outcome $\hat{Y}$ from the associated probability estimates on both the training and test sets. Then compute the confusion table and prediction accuracy both on the training and test set. Comment on the results. Use the command `predict` with argument `type = "response"`.

```r
ridge_opt_train <- glmnet(X_train, y_train, alpha = 0,
                          lambda = optimal_train_lambda,
                          family = "binomial")
```

```r
ridge_train_prob <- predict(ridge_opt_train, type = "response", X_train)
ridge_train_class <- ifelse(ridge_train_prob > 0.5, "M", "B")

ridge_train_matrix <- confusionMatrix(factor(ridge_train_class, levels = c("B","M")),
                                      y_train, positive = "M")

kable(ridge_train_matrix$table,
      caption = "Ridge Logistic Regression Training Set Confusion Matrix")
```

Table 6: Ridge Logistic Regression Training Set Confusion Matrix

	B	M
B	254	5
M	1	140

```r
# Test predictions, confusion table, and prediction accuracy
ridge_test_prob <- predict(ridge_opt_train, type = "response", X_test)
ridge_test_class <- ifelse(ridge_test_prob > 0.5, "M", "B")

ridge_test_matrix <- confusionMatrix(factor(ridge_test_class, levels = c("B","M")),
                                     y_test, positive = "M")
kable(ridge_test_matrix$table,
      caption = "Ridge Logistic Regression Test Set Confusion Matrix")
```

Table 7: Ridge Logistic Regression Test Set Confusion Matrix

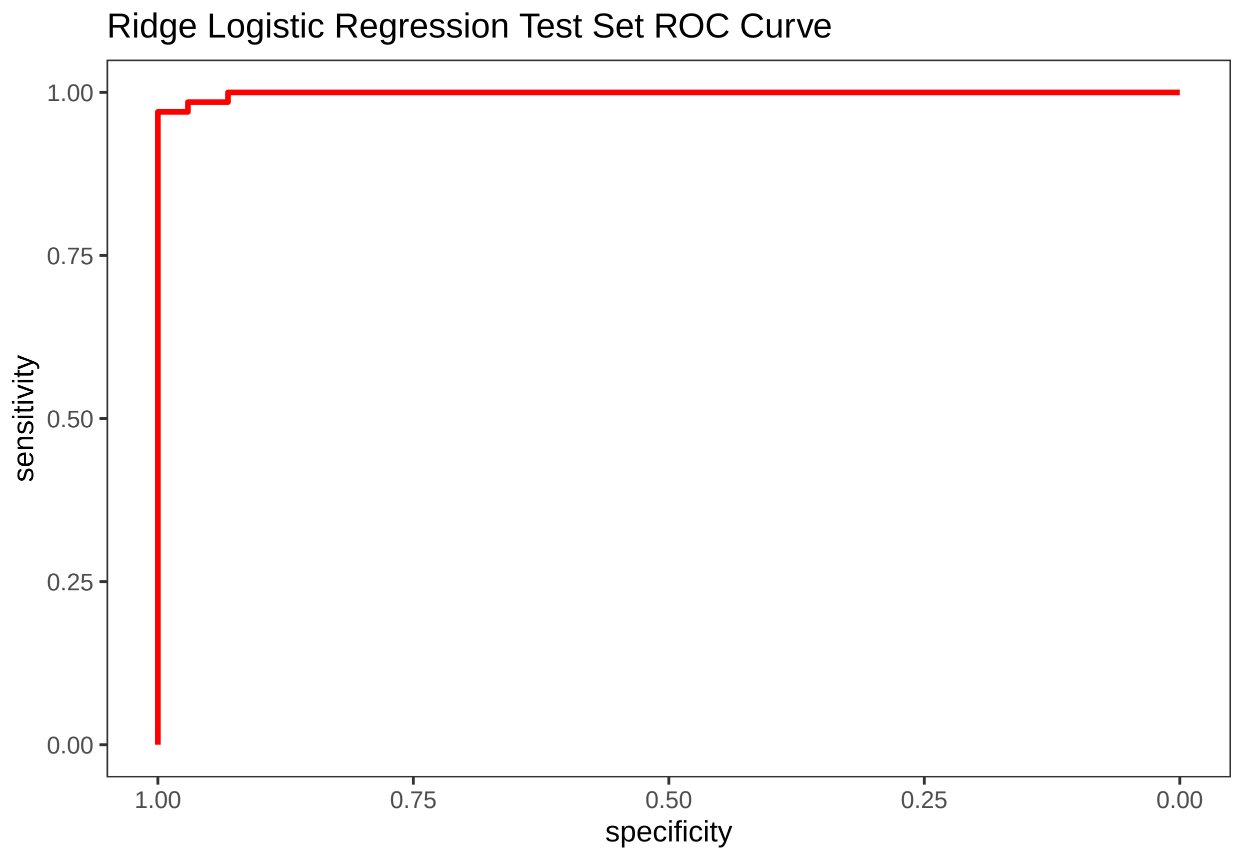
	B	M
B	102	2
M	0	65

The prediction accuracy of the ridge logistic regression model on the training set is 0.985 while the prediction accuracy in the test set is 0.9881657.

The confusion table from the training and test sets show that the ridge logistic regression model would predict more false negatives (i.e., predict `benign` when actually `malignant`; 3.45% and 2.99%, respectively) than false positives (i.e., predict `malignant` when actually `benign`; 0.39% and 0%, respectively).

- 2g. Plot the ROC curve, computed on the test set.

```r
ridge_roc_score <- roc(response = y_test, predictor = ridge_test_prob)

ggroc(ridge_roc_score, linetype = 1, size = 1, color = "red") +
  ggtitle("Ridge Logistic Regression Test Set ROC Curve") +
  theme_bw() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank())
```



- 2h. Compute an estimate of the area under the ROC curve (AUC).

```
ridge_roc_score$auc
## Area under the curve: 0.9985
```

The area under the ROC curve is 0.9985367.

3. Lasso Logistic Regression

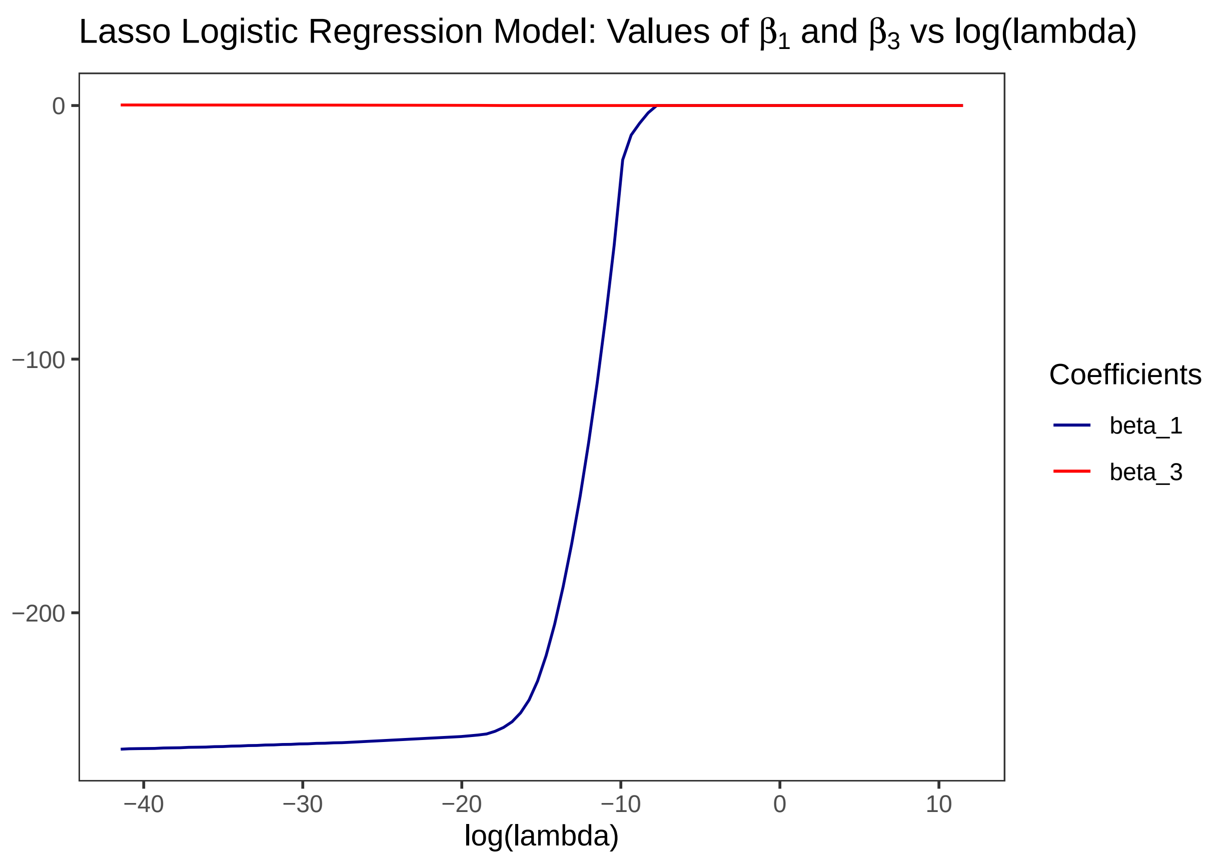
Repeat the sub-problems 2b to 2h using a lasso logistic regression model instead of a ridge logistic regression model.

- 3a. On the training set, run a lasso logistic regression model with `glmnet` (with the argument `family = "binomial"`) to predict Y given $X_1, ..., X_{30}$. Use the same grid of values for lambda as the ridge logistic regression model: `10^seq(5, -18, length = 100)`.

```
lasso_train <- glmnet(X_train, y_train, alpha = 1, lambda = lambda_grid,
                      family = "binomial")
```

- 3b. Plot the values of the coefficients β_1, β_3 (y-axis) in function of `log(lambda)` (x-axis). Comment on the result.

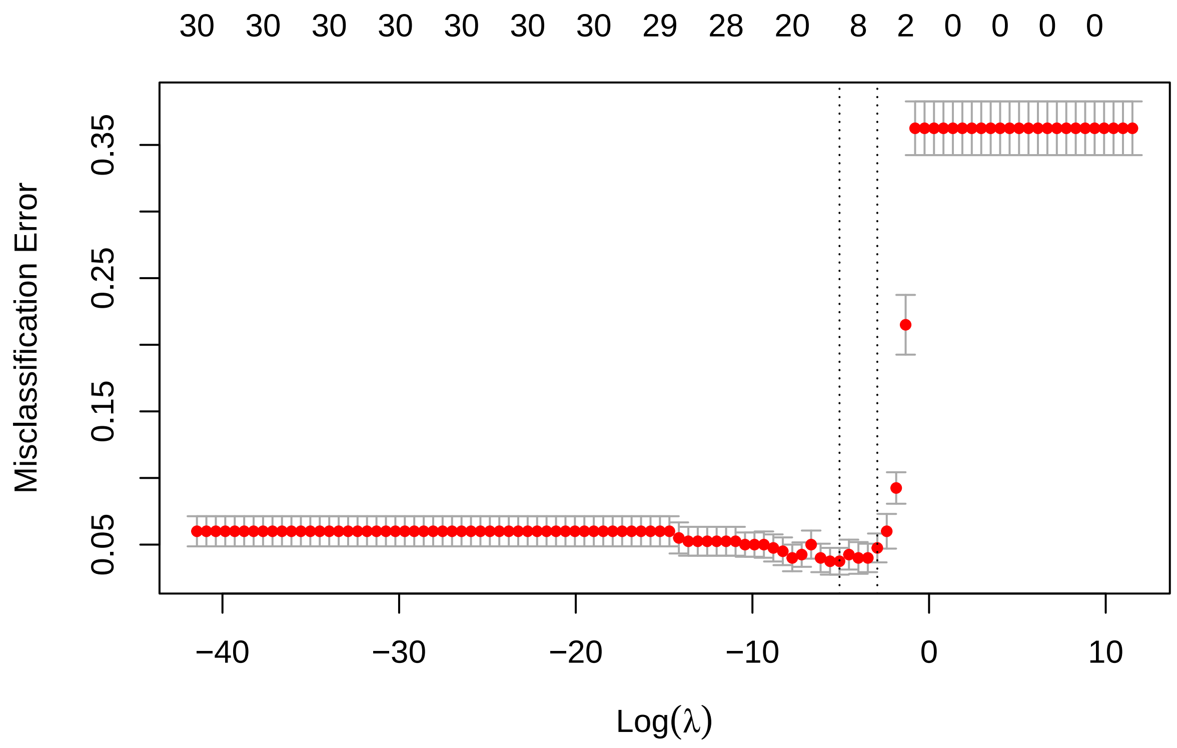
```
lasso_dat <- data.frame(x = log(lasso_train$lambda),
                        beta_1 = lasso_train$beta[1, ],
                        beta_3 = lasso_train$beta[3, ])

ggplot(data = lasso_dat, aes(x = x)) +
  geom_line(aes(y = beta_1, color = "beta_1")) +
  geom_line(aes(y = beta_3, color = "beta_3")) +
  scale_color_manual(name = "Coefficients",
                     values = c("beta_1" = "darkblue", "beta_3" = "red")) +
  theme_bw() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank()) +
  labs(title = expression(paste("Lasso Logistic Regression Model: Values of ",
                                beta[1], " and ", beta[3], " vs log(lambda)")),
       x = "log(lambda)",
       y = element_blank())
```



When applying the lasso logistic regression model to the training set, we see that the values of β_3 is reduced to zero for all values of λ. The values of β_1 is lower than β_3 when log(λ) $>$ -10 and β_1 approaches zero between $-10 < log(\lambda) < -5$.

- 3c. 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. We will refer to it as the optimal lambda. Plot the misclassification error (y-axis) in function of `log(lambda)` (x-axis). Use `cv.glmnet` with the arguments `family = "binomial"` and `type.measure = "class"`.

```
set.seed(2)

lasso_train_cv <- cv.glmnet(X_train, y_train, alpha = 1, nfolds = 10,
                            lambda = lambda_grid, family = "binomial",
                            type.measure = "class")
plot(lasso_train_cv)
```



```
optimal_lasso_lambda <- lasso_train_cv$lambda.min
```

The optimal value of λ that minimizes the CV misclassification error is 0.0062803.

- 3d. Report the number of coefficients β_j that are different from 0 for the ridge model with the optimal lambda. Comment on the results.

```
sum(predict(lasso_train, s = optimal_lasso_lambda, type = "coefficients")[-1, ] != 0)
## [1] 11
```

There are 11 non-zero coefficients for the lasso logistic regression model using the optimal λ. Unlike the ridge logistic regression model, the lasso logistic regression model is able to reduce coefficients to zero.

- 3e. Use the regularized glm previously fitted (with the optimal lambda) and the Bayes rule to compute the predicted outcome $\hat{Y}$ from the associated probability estimates on both the training and test sets. Then compute the confusion table and prediction accuracy both on the training and test set. Comment on the results. Use the command `predict` with argument `type = "response"`.

```
lasso_opt_train <- glmnet(X_train, y_train, alpha = 1,
                          lambda = optimal_lasso_lambda,
                          family = "binomial")

lasso_train_prob <- predict(lasso_opt_train, type = "response", X_train)
```

```
lasso_train_class <- ifelse(lasso_train_prob > 0.5, "M", "B")

lasso_train_matrix <- confusionMatrix(factor(lasso_train_class,
                                             levels = c("B","M")),
                                      y_train, positive = "M")

kable(lasso_train_matrix$table,
      caption = "Lasso Regression Training Set Confusion Matrix")
```

Table 8: Lasso Regression Training Set Confusion Matrix

	B	M
B	253	6
M	2	139

```
# Test predictions, confusion table, and prediction accuracy
lasso_test_prob <- predict(lasso_opt_train, type = "response", X_test)
lasso_test_class <- ifelse(lasso_test_prob > 0.5, "M", "B")

lasso_test_matrix <- confusionMatrix(factor(lasso_test_class,
                                            levels = c("B","M")),
                                     y_test, positive = "M")
kable(lasso_test_matrix$table,
      caption = "Lasso Regression Test Set Confusion Matrix")
```

Table 9: Lasso Regression Test Set Confusion Matrix

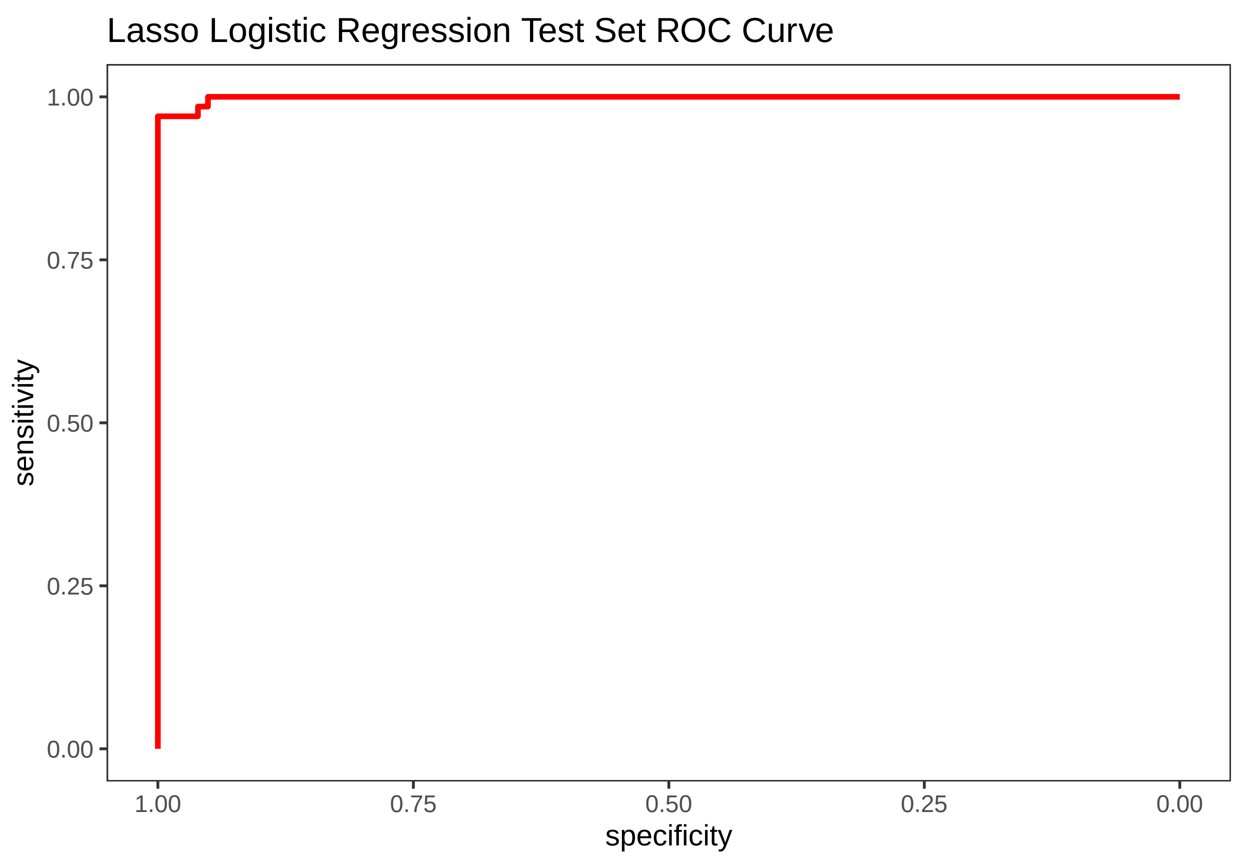
	B	M
B	102	2
M	0	65

The prediction accuracy of the logistic regression model on the training set is 0.98 while the prediction accuracy in the test set is 0.9881657.

The confusion table from the training and test sets show that the regression model would predict more false negatives (i.e., predict `benign` when actually `malignant`; 4.14% and 2.99%, respectively) than false positives (i.e., predict `malignant` when actually `benign`; 0.78% and 0%, respectively).

- 3f. Plot the ROC curve, computed on the test set.

```
lasso_roc_score <- roc(response = y_test, predictor = lasso_test_prob)

ggroc(lasso_roc_score, linetype = 1, size = 1, color = "red") +
  ggtitle("Lasso Logistic Regression Test Set ROC Curve") +
  theme_bw() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank())
```



- 3g. Compute an estimate of the area under the ROC curve (AUC).

```
lasso_roc_score$auc
## Area under the curve: 0.9987
```

The area under the ROC curve is 0.9986831.

4. Discussion

Discuss the performance of the simple glm, ridge glm, and lasso glm on the Breast Cancer Wisconsin Data Set in terms of prediction accuracy (on the training and test set) and model interpretability.

```r
kable(data.frame(Model = c("simple", "ridge", "lasso"),
           `Training accuracy` = c(glm_train_matrix$overall[1],
                                   ridge_train_matrix$overall[1],
                                   lasso_train_matrix$overall[1]),
           `Test accuracy` = c(glm_test_matrix$overall[1],
                               ridge_test_matrix$overall[1],
                               lasso_test_matrix$overall[1]),
           `Test AUC` = c(roc(response = y_test,
                              predictor = glm_test_prob)$auc,
                          ridge_roc_score$auc,
                          lasso_roc_score$auc),
           `Non-zero coefficients` = c(sum(coef(glm_train)[-1] != 0),
                                       sum(predict(ridge_train,
                                                   s = optimal_train_lambda,
                                                   type = "coefficients")[-1, ] != 0),
                                       sum(predict(lasso_train,
                                                   s = optimal_lasso_lambda,
                                                   type = "coefficients")[-1, ] != 0)),
           check.names = FALSE),
      caption = "glm Model Performance Summary")
```

Table 10: glm Model Performance Summary

Model	Training accuracy	Test accuracy	Test AUC	Non-zero coefficients
simple	1.000	0.9408284	0.9469564	30
ridge	0.985	0.9881657	0.9985367	30
lasso	0.980	0.9881657	0.9986831	11

From **Table 10**, we can see that the simple glm model had the greatest training prediction accuracy (1.000), followed by the ridge model (0.985). However, the simple glm model had the lowest test prediction accuracy of the three models (0.941). The ridge and lasso glm models are both tied for the greatest test prediction accuracy (0.988). If going by test prediction accuracy alone, both the ridge and lasso glm models would be sufficient over the simple glm model.

To select between the lasso and ridge glm models, one can look at the test ROC curve AUC values and the overall model interpretability. The lasso glm model has the greater test ROC curve AUC of the two models (0.9986831 vs 0.9985367) as well as less number of coefficients included in the model (11 non-zero coefficients vs 30 non-zero coefficients). The greater ROC curve AUC value indicates better prediction accuracy/model performance while the lower number of non-zero coefficients indicates better/greater model interpretability. In other words, the lasso glm model is a more simple model compared to the ridge glm model while also being more accurate.