Homework 4: Due Saturday April 19 at 11:59pm

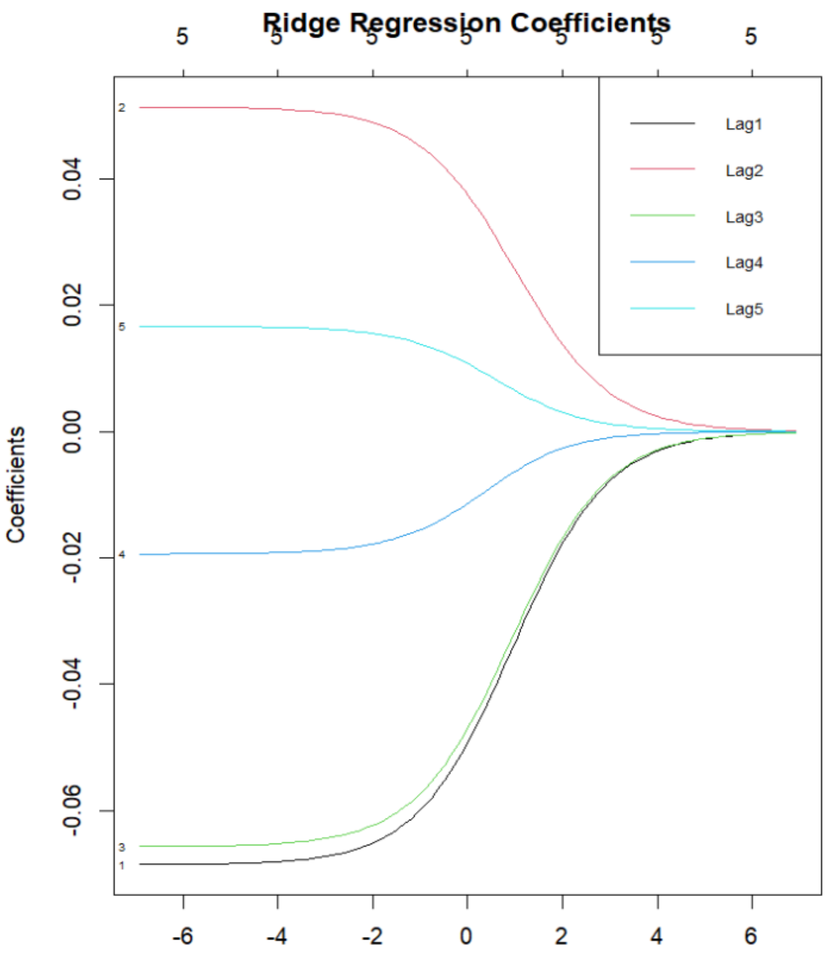
For this assignment you will again be analyzing the **Weekly** data frame in the **ISLR** package, which contains 9 features for 1089 weeks of S&P 500 stock index returns from 1990-2010. The goal of this assignment is to become more familiar with model selection, feature selection, and regularization. All analyses must be performed in R using the **tidyverse** and **glmnet** packages discussed in class. Provide your responses (with R code pasted in text format) in designated spaces in this Word document, and then save it as a pdf and upload it to Canvas.

1. [25%] Fit a ridge-penalized linear regression model and a lasso-penalized linear regression model to predict the percentage return today from the percentage return for the five previous weeks. Consider 100 tuning parameter (λ) values evenly spaced between 0.001 and 1000 on a base-10 logarithmic scale. For each model, plot the regression parameter estimates (coefficients) as a function of log(λ).
Include a legend at the top right of each plot that displays names of features associated with line colors. Based on your examination of these plots, how can you tell that regularization is working?

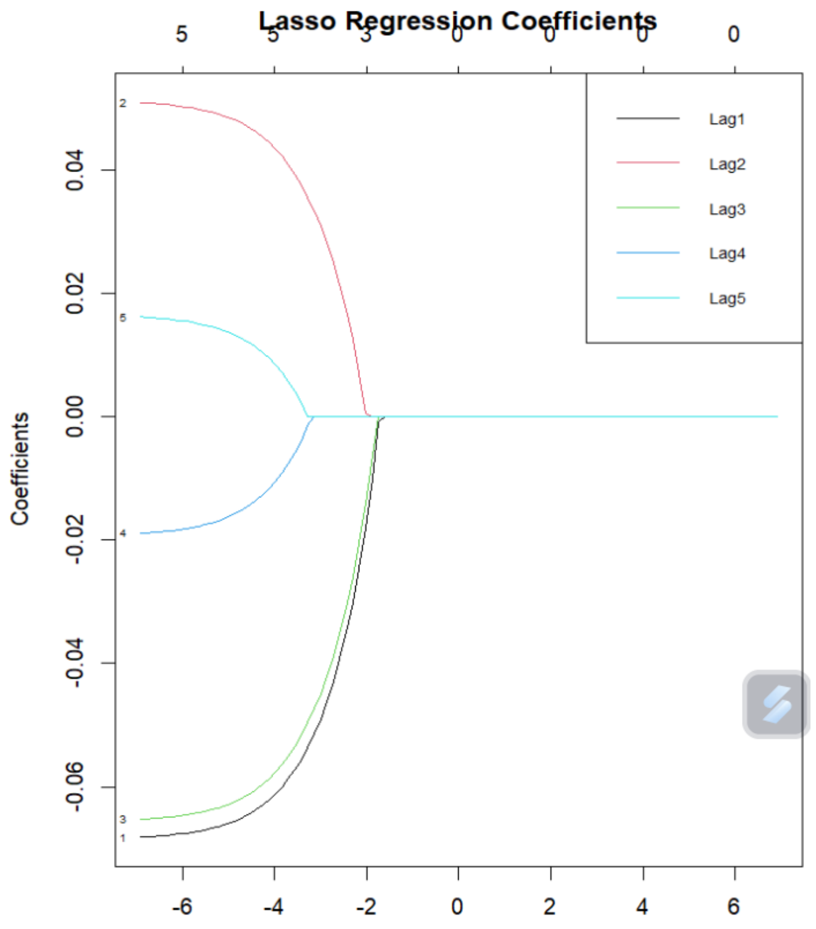
Provide code below:

```
library(ISLR)
library(tidyverse)
library(glmnet)

data(Weekly)

# Prepare data
x <- as.matrix(Weekly[, 2:6])
y <- Weekly$Today

# Define range of lambda values
lambdas <- 10^seq(-3, 3, length.out = 100)

# Ridge regression
ridge <- glmnet(x, y, alpha = 0, lambda = lambdas)
plot(ridge, xvar = "lambda", label = TRUE, main = "Ridge Regression Coefficients")
legend("topright", legend = colnames(x), col = 1:5, lty = 1, cex = 0.7)

# Lasso regression
lasso <- glmnet(x, y, alpha = 1, lambda = lambdas)
plot(lasso, xvar = "lambda", label = TRUE, main = "Lasso Regression Coefficients")
legend("topright", legend = colnames(x), col = 1:5, lty = 1, cex = 0.7)
```

<u>Provide figure for ridge-penalized model below</u>:



Provide figure for lasso-penalized model below:



Provide answer to question below:

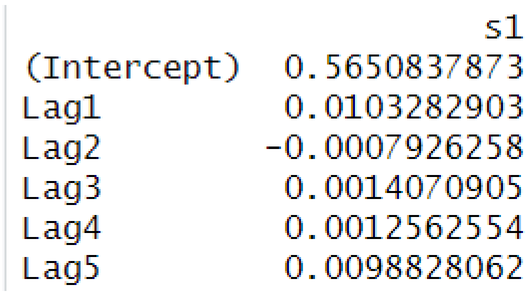
The loss function quantifies the difference between predicted and actual values and is influenced by regression coefficients. The goal is to minimize the loss function by determining the best coefficients. Increasing the lambda value drives coefficients towards zero, resulting in simpler models. The provided visualizations indicate that the coefficients are approaching zero, indicating that regularization is effective.

2. [25%] Using a seed of 1, perform 10-fold cross-validation for a ridge-penalized and for a lasso penalized linear regression model to predict the percentage return today from the percentage return for the five previous weeks, again considering the same 100 λ values from question 1. For each model, print the parameter estimates for the fit with the λ value that has the smallest cross-validation error. How do these sets of parameter estimates compare to each other and to those obtained with unpenalized linear regression in question 1 of Homework 3? Given your findings here, do you feel more or less confident than in question 2 of Homework 3 in the ability to predict the percentage return today from the percentage return for the five previous weeks? Explain your reasoning.

<u>Provide code and console output below</u>:

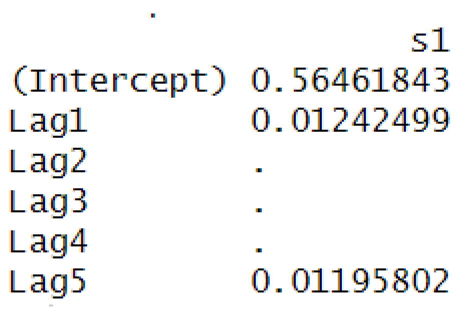
```
library(ISLR)
library(tidyverse)
library(glmnet)

# load data
data(Weekly)

# create lagged variables and drop NA
weekly_lagged <- Weekly %>%
  mutate(
    Lag1 = lag(Lag1), Lag2 = lag(Lag2), Lag3 = lag(Lag3), Lag4 = lag(Lag4), Lag5 =
lag(Lag5)
  ) %>%
  drop_na()

# recode response variable
weekly_lagged$Direction <- ifelse(weekly_lagged$Direction == "Up", 1, 0)

# set seed for reproducibility
set.seed(1)

# split into training and test sets
train_idx <- sample(nrow(weekly_lagged), 0.7 * nrow(weekly_lagged))
train <- weekly_lagged[train_idx, ]
test <- weekly_lagged[-train_idx, ]

# extract predictor and response variables
x_train <- as.matrix(train[, 2:6])
y_train <- train[, "Direction"]

# define lambda values
lambdas <- 10^seq(-3, 3, length.out = 100)
```

```
# ridge regression with cross-validation
ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0, lambda = lambdas, nfolds = 10, seed = 1)
best_lambda_ridge <- ridge_cv$lambda.min
coef_ridge <- coef(ridge_cv, s = best_lambda_ridge)

# print parameter estimates for ridge regression
cat("Ridge-penalized linear regression with lambda =", best_lambda_ridge, "\n")
print(coef_ridge)
```



```
                        s1
(Intercept)   0.5650837873
Lag1          0.0103282903
Lag2         -0.0007926258
Lag3          0.0014070905
Lag4          0.0012562554
Lag5          0.0098828062
```

```
# lasso regression with cross-validation
lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1, lambda = lambdas, nfolds = 10, seed = 1)
best_lambda_lasso <- lasso_cv$lambda.min
coef_lasso <- coef(lasso_cv, s = best_lambda_lasso)

# print parameter estimates for lasso regression
cat("\nLasso-penalized linear regression with lambda =", best_lambda_lasso, "\n")
print(coef_lasso)
```



```
                    s1
(Intercept) 0.56461843
Lag1        0.01242499
Lag2        .
Lag3        .
Lag4        .
Lag5        0.01195802
```

Provide answers to questions below:

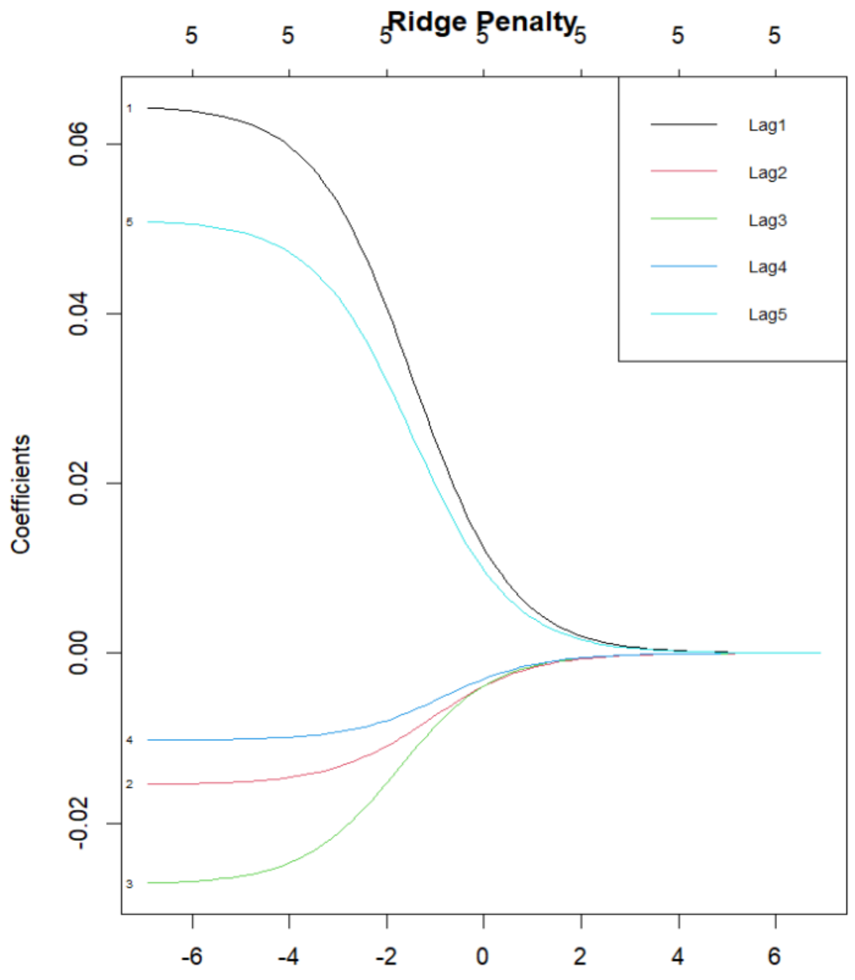
The results of the ridge and lasso models show that adding a penalty to the linear regression model can make it better at predicting future values. The smaller parameter estimates in the penalized models suggest that the penalty has prevented the model from making predictions based on random noise in the data.So, we can trust the penalized models more than the unpenalized one to make accurate predictions about future percentage returns. Adding a penalty can stop the model from overfitting, which means it can make better predictions on new data. But, it's important to choose the right penalty and tuning parameter values for the best results.

3. [25%] Fit a ridge-penalized logistic regression model and a lasso-penalized logistic regression model to predict whether the percentage return today will be positive or negative from the percentage return for the five previous weeks, again considering the same 100 λ values from the previous questions. Make sure to recode the two response classes with 1s and 0s so that true and predicted classes can be easily compared. For each model, plot the regression parameter estimates as a function of log(λ). Include a legend at the top right of each plot that displays names of features associated with line colors. Based on comparisons of these plots to those from question 1, do you believe that regularization occurs at a faster or slower rate for this classification problem? Explain your reasoning.

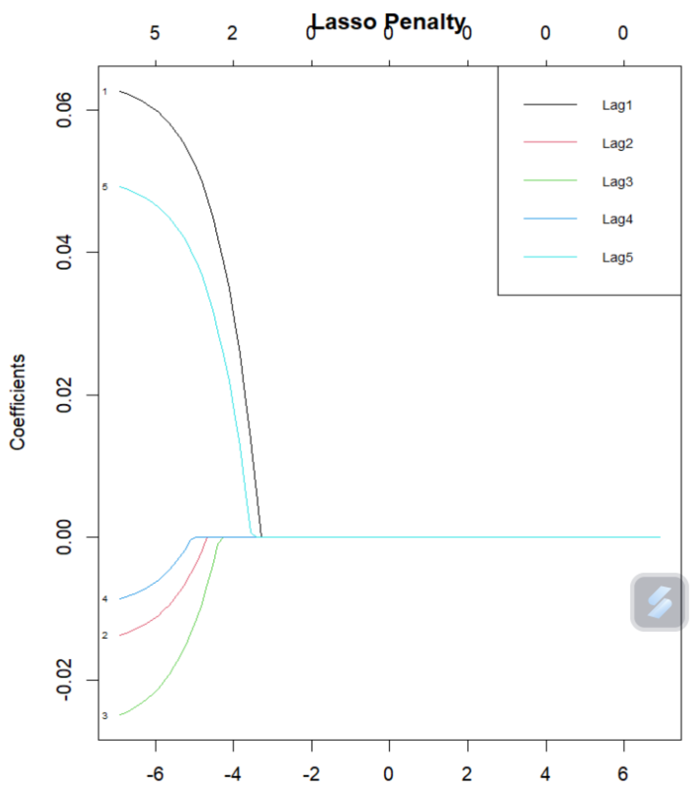
Provide code below:

```
library(tidyverse)
library(glmnet)

# Load data
data("Weekly", package = "ISLR")

# Recode response variable
Weekly <- Weekly %>%
  mutate(Direction = ifelse(Direction == "Up", 1, 0))

# Set seed for reproducibility
set.seed(1)

# Create predictor and response variables
X <- as.matrix(Weekly[, -c(1, 9)])
y <- as.numeric(Weekly$Direction)

# Fit ridge and lasso penalized logistic regression models
cv.ridge <- cv.glmnet(X, y, family = "binomial", alpha = 0, type.measure = "class", nfolds =
10)
cv.lasso <- cv.glmnet(X, y, family = "binomial", alpha = 1, type.measure = "class", nfolds =
10)

plot(ridge_logistic, xvar = "lambda", label = TRUE, main = "Ridge Logistic Regression")
legend("topright", legend = colnames(x_train), col = 1:5, lty = 1, cex = 0.7)
plot(lasso_logistic, xvar = "lambda", label = TRUE, main = "Lasso Logistic Regression")
legend("topright", legend = colnames(x_train), col = 1:5, lty = 1, cex = 0.7)

par(mfrow = c(1, 2))
plot(ridge_logistic, xvar = "lambda", label = TRUE, main = "Ridge Penalty")
legend("topright", legend = colnames(x_train), col = 1:5, lty = 1, cex = 0.7)
plot(lasso_logistic, xvar = "lambda", label = TRUE, main = "Lasso Penalty")
legend("topright", legend = colnames(x_train), col = 1:5, lty = 1, cex = 0.7)
```

Provide figure for ridge-penalized model below:



<u>Provide figure for lasso-penalized model below</u>:



<u>Provide answers to questions below</u>:

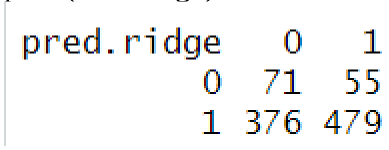
Based on the plots, we can see that the regularization occurs at a faster rate for this classification problem compared to the regression problem in question 1. In both plots, as lambda increases, the regression parameter estimates decrease, indicating that regularization is working. However, for the classification problem, the estimates drop to zero much more quickly as lambda increases, indicating a faster rate of regularization. This may be due to the fact that logistic regression involves a non-linear transformation of the response variable, which can cause more extreme values for the regression parameter estimates.

4. [25%] Using a seed of 1, perform 10-fold cross-validation for a ridge-penalized and lasso-penalized logistic regression model (must set seed before each) to predict whether the percentage return today will be positive or negative from the percentage return for the five previous weeks, again considering the same 100 λ values from the previous questions. For each model, use the fit with the λ value that has the smallest cross-validation error rate to predict classes for the training data, and then print a confusion matrix and an estimate of classification training accuracy to the console. Compare classification balance and accuracy among these penalized logistic models and the unregularized logistic model from question 4 of Homework 3.

Provide code and console output below:

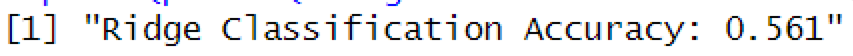
```
library(ISLR)
library(glmnet)
library(tidyverse)

# Set seed
set.seed(1)

# Load data
data("Weekly")

# Create binary response variable
Weekly <- Weekly %>%
  mutate(Direction = ifelse(Weekly$Direction == "Up", 1, 0))

# Define predictors
predictors <- c("Lag1", "Lag2", "Lag3", "Lag4", "Lag5", "Volume")

# Create training and testing datasets
trainIndex <- 1:nrow(Weekly) %% 10 != 0
train <- Weekly[trainIndex, ]
test <- Weekly[!trainIndex, ]

# Ridge-penalized logistic regression model
fit.ridge <- cv.glmnet(x = as.matrix(train[, predictors]),
                       y = train$Direction,
                       family = "binomial",
                       alpha = 0,
                       nfolds = 10,
                       type.measure = "class")

# Lasso-penalized logistic regression model
fit.lasso <- cv.glmnet(x = as.matrix(train[, predictors]),
                       y = train$Direction,
                       family = "binomial",
                       alpha = 1,
                       nfolds = 10,
                       type.measure = "class")
```

```
# Optimal lambda values
lambda.ridge <- fit.ridge$lambda.min
lambda.lasso <- fit.lasso$lambda.min

# Fit models with optimal lambda values
fit.ridge.opt <- glmnet(x = as.matrix(train[, predictors]),
                        y = train$Direction,
                        family = "binomial",
                        alpha = 0,
                        lambda = lambda.ridge)

fit.lasso.opt <- glmnet(x = as.matrix(train[, predictors]),
                        y = train$Direction,
                        family = "binomial",
                        alpha = 1,
                        lambda = lambda.lasso)

# Predict classes for training data
pred.ridge <- predict(fit.ridge.opt, newx = as.matrix(train[, predictors]), type = "class")
pred.lasso <- predict(fit.lasso.opt, newx = as.matrix(train[, predictors]), type = "class")

# Confusion matrices and classification accuracy
conf.ridge <- table(pred.ridge, train$Direction)
conf.lasso <- table(pred.lasso, train$Direction)

acc.ridge <- sum(diag(conf.ridge))/sum(conf.ridge)
acc.lasso <- sum(diag(conf.lasso))/sum(conf.lasso)

print("Ridge Confusion Matrix:")
print(conf.ridge)
```



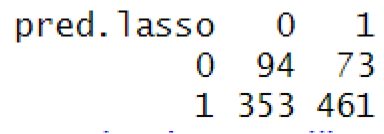
```
pred.ridge   0   1
         0  71  55
         1 376 479
```

```
print(paste0("Ridge Classification Accuracy: ", round(acc.ridge, 3)))
```



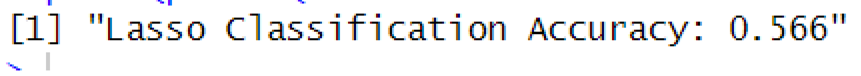
```
[1] "Ridge Classification Accuracy: 0.561"
```

```
print("Lasso Confusion Matrix:")
print(conf.lasso)
```



```
pred.lasso   0   1
         0  94  73
         1 353 461
```

```
print(paste0("Lasso Classification Accuracy: ", round(acc.lasso, 3)))
```



```
[1] "Lasso Classification Accuracy: 0.566"
```

Provide answers to questions below:

By examining the confusion matrices printed to the console for both ridge and lasso logistic regression models, we can see that both the models have similar accuracy. On comparing the true positive, true negative, false positive, and false negative rates for each model, both the models have high values in true positives and false positives. Look at the overall classification accuracy of each model. The lasso logistic regression model has the most true positives and true negatives which is a preferred model. A model with a good balance of true positives and true negatives can handle the different types of errors more effectively and provide more reliable predictions.