Econ 573: Problem Set 3

Part I

Exercise 2

(a) The Lasso relative to least squares, is:

The Lasso imposes a an 1 regression penalty, which shrinks the coefficients towards zero. This 1 penalty has the effect of forcing some of the coefficients to be exactly equal to zero if the tuning parameter   is sufficiently large, such that it performs a variable selection. As the tuning parameter, , increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias. Additionally, the Lasso method only improves prediction accuracy over the least squares, such that it minimizes the MSE, when the increase in bias is less than the decrease in variance. If the increase in bias is larger than the decrease in variance, then the coefficients become significantly underestimated. So, option iii is correct.

(b) Repeat (a) for ridge regression relative to least squares.

The Ridge imposes a an 2 regression penalty, which shrinks the coefficients towards zero. As the tuning parameter, , increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias. Additionally, the Lasso method only improves prediction accuracy over the least squares, such that it minimizes the MSE, when the increase in bias is less than the decrease in variance. If the increase in bias is larger than the decrease in variance, then the coefficients become significantly underestimated. So, option iii is correct.

(c) Repeat (a) for non-linear methods relative to least squares.

Nonlinear methods are able to capture more complex relationships between predictors and the response variables, making them more flexible than least squares, but they tend to have higher variance due to higher risks of overfitting. Additionally, they are able to better capture complex relationships between variables, thus reducing bias. So, the tradeoff is that if the decrease in bias is higher than the increase in variance, then the prediction accuracy of the nonlinear method in comparison to least squares will improve. So, option ii is correct.

Exercise 3

(a) iv. In the starting point where s = 0, all parameter coefficients are forced to be zero, such that the model only predicts the mean of y and has a high RSS. As s increases and the constraint is relaxed, the model is allowed to have nonzero coefficients, leading to a lower RSS. Since increasing s allows for more complex models, the training RSS will steadily decrease.

(b) ii. As s increases from the starting point where s = 0, the test RSS decreases as it includes more parameters. However, after a certain point the model encounters overfitting issues, causing the test RSS to increase again due to high variance.

(c) iii. Because increasing s, increases both the model flexibility and complexity, variance steadily increases as s increases.

(d) iv. Because model flexibility increases alongside s, bias in the model is reduced as it is able to capture more patterns in the data. So, the bias steadily decreases as s increases.

(e) v. The irreducible error represents noise in the data that can't be reduced, regardless of the complexity of the model. Therefore, it remains constant as s increases.

Exercise 10

(a)

```r
set.seed(123)

n <- 1000
p <- 20

X <- matrix(rnorm(n*p),nrow = n, ncol = p)

beta <- c(2, -1.5, 3, 0, 0, 1, -2, 0, 0, 1.5, 0, 0, 2.5, 0, 0, -1, 0, 0, 0, 0)

epsilon <- rnorm(n, mean = 0, sd = 1)

Y <- X %*% beta + epsilon

data <- data.frame(Y, X)

head(data)
```

```
##            Y          X1          X2         X3          X4         X5
## 1  2.7805591 -0.56047565 -0.99579872 -0.5116037 -0.15030748  0.1965498
## 2  2.3815253 -0.23017749 -1.03995504  0.2369379 -0.32775713  0.6501132
## 3 -3.2949558  1.55870831 -0.01798024 -0.5415892 -1.44816529  0.6710042
## 4  2.6361075  0.07050839 -0.13217513  1.2192276 -0.69728458 -1.2841578
## 5  7.2350568  0.12928774 -2.54934277  0.1741359  2.59849023 -2.0261096
## 6 -0.1556495  1.71506499  1.04057346 -0.6152683 -0.03741501  2.2053261
##           X6         X7         X8         X9        X10        X11        X12
## 1 -0.4941739 -0.6992281 -1.6180367  0.5110004  1.9315759  2.3707252  0.2677904
## 2  1.1275935  0.9964515  0.3791812  1.8079928 -0.6164747 -0.1668120 -0.3993609
## 3 -1.1469495 -0.6927454  1.9022505 -1.7026150 -0.5625675  0.9269614  0.2768838
## 4  1.4810186 -0.1034830  0.6018743  0.2874488 -0.9899631 -0.5681517 -1.2336734
## 5  0.9161912  0.6038661  1.7323497 -0.2691142  2.7312276  0.2250901  0.7535749
## 6  0.3351310 -0.6080450 -0.1468696 -0.3795247 -0.7216662  1.1319859  0.2892610
##          X13        X14        X15        X16        X17        X18         X19
## 1 -0.1506300 -1.4100699 -0.9100612 -1.3538489 -1.0921750  0.7166721  0.27822242
## 2  0.8009406 -1.7212650  0.2806627 -0.5793773 -1.3375419 -0.2835015 -1.60253773
## 3 -1.1867178  0.6654449 -1.0356704 -0.8610442  1.1092420 -1.1354637 -0.32089953
## 4  0.4306364 -0.4308067  0.2730487  0.9726783  0.8107015 -0.6328166 -1.22611415
## 5  0.2167471 -0.3290387  0.5377981  0.6191458 -0.8759813  1.3511612 -0.77665758
## 6  0.9612923  1.5633095  0.5003932  1.3854457  0.2070943 -1.9678465  0.08777636
##          X20
## 1  2.0279109
## 2 -1.4947497
## 3 -1.5729492
## 4 -0.3002123
## 5 -0.7643735
## 6 -1.1161571
```

(b)

```
set.seed(123)

train_indices <- sample(1:n, 100)

train_data <- data[train_indices, ]

test_data <- data[-train_indices, ]

dim(train_data)
```

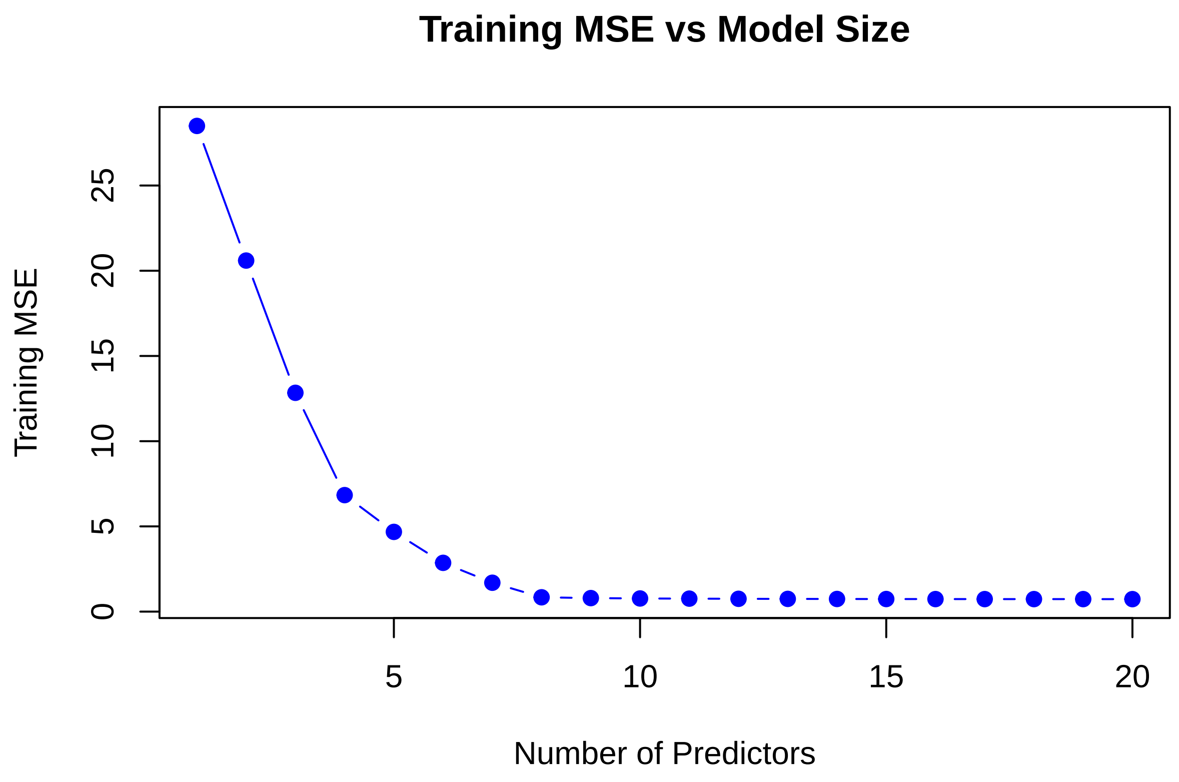
```
## [1] 100  21
```

```
dim(test_data)
```

```
## [1] 900  21
```

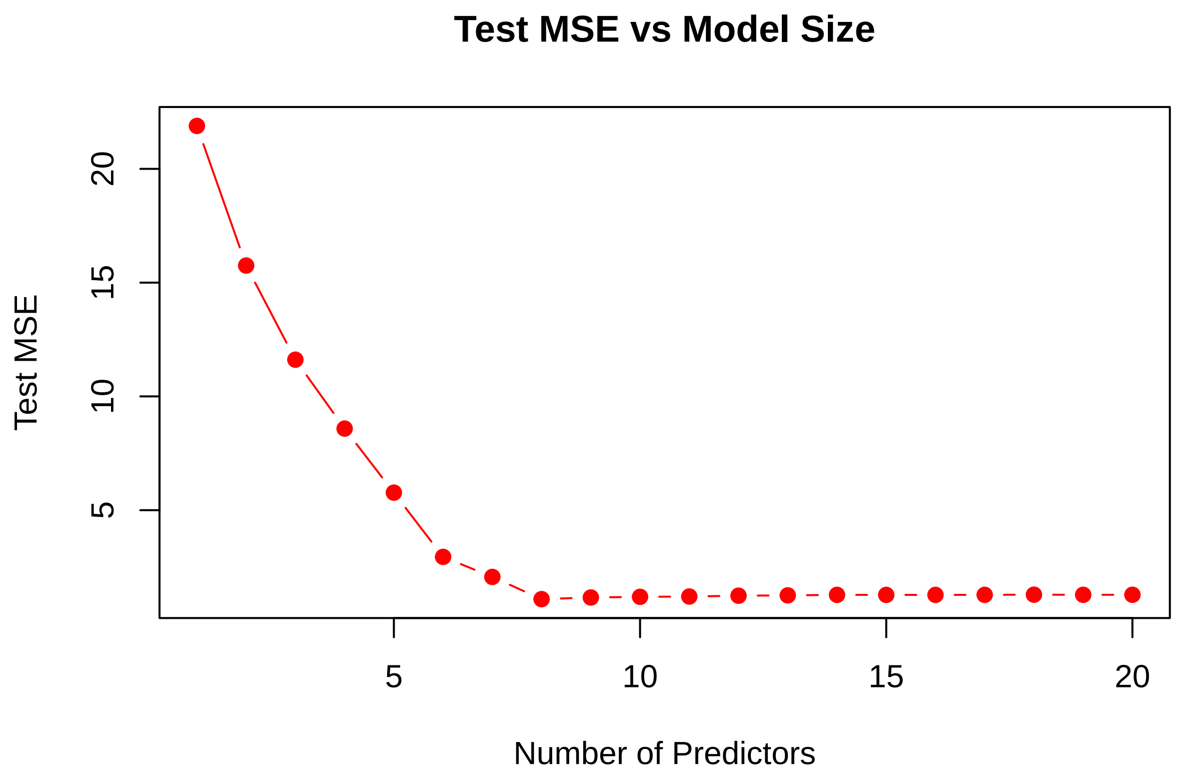
(c)

```
library(leaps)

best_subset <- regsubsets(Y ~ ., data = train_data, nvmax = 20)

train_mse <- rep(NA, 20) # Initialize vector of length 20 with NA values.

train_X <- as.matrix(train_data[-1])
train_Y <- train_data$Y

for (i in 1:20) {
  coef_i <- coef(best_subset, id = i)

  train_preds <- cbind(1, train_X[, names(coef_i)[-1]]) %*% coef_i

  train_mse[i] <- mean((train_Y - train_preds)^2)
}

 plot(1:20, train_mse, type = "b", pch = 19, col = "blue",
     xlab = "Number of Predictors", ylab = "Training MSE",
     main = "Training MSE vs Model Size")
```



(d)

```r
test_mse <- rep(NA, 20)

test_X <- as.matrix(test_data[, -1])
test_Y <- test_data$Y

for (i in 1:20) {
  coef_i <- coef(best_subset, id = i)

  selected_test_X <- test_X[, names(coef_i)[-1], drop = FALSE]

  design_matrix <- cbind(1, selected_test_X)

  test_preds <- design_matrix %*% coef_i

  test_mse[i] <- mean((test_Y - test_preds)^2)
}

plot(1:20, test_mse, type = "b", pch = 19, col = "red",
     xlab = "Number of Predictors", ylab = "Test MSE",
     main = "Test MSE vs Model Size")
```



(e) For the test set MSE, it takes its minimum value with the model that has 8 predictors. This optimal model size balances bias and variance. As expected, the test MSE rapidly decreases as we initially begin to add new predictors to the null model, reaching a minimum point where p = 8. After the model with 8 predictors, adding an extra predictors seems to cause the MSE to slightly increase due to overfitting.

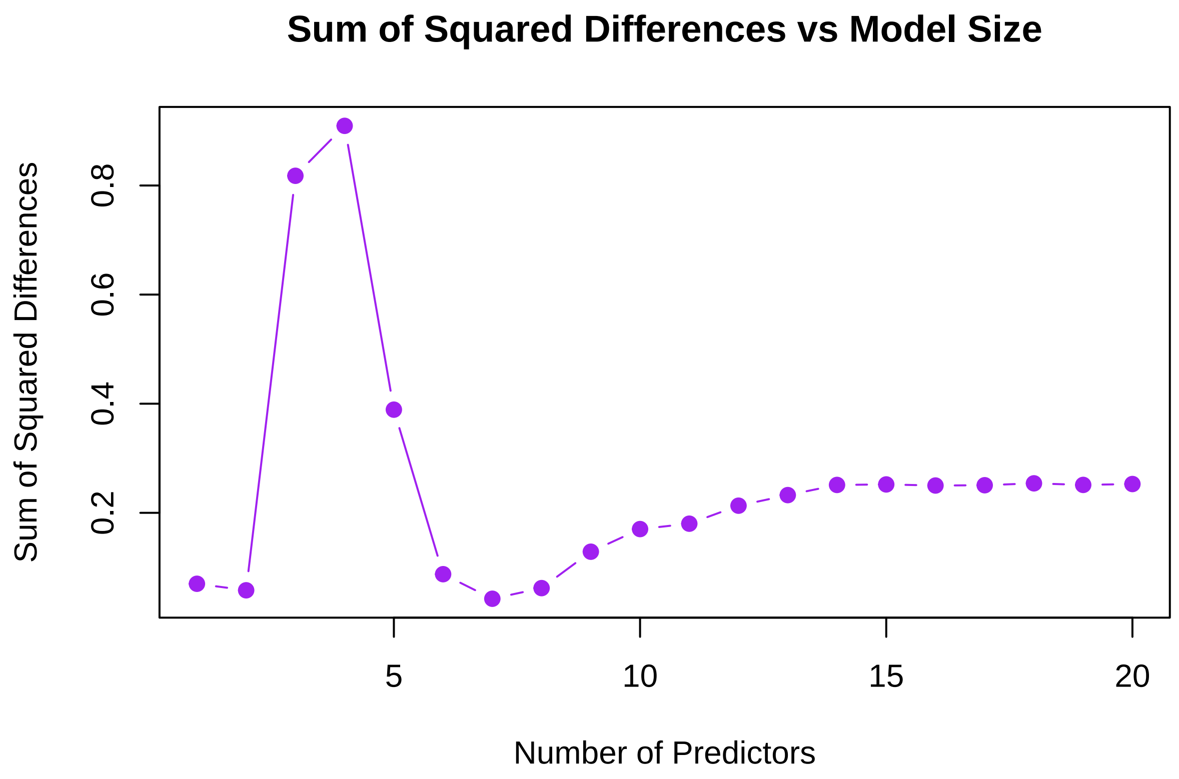
(f)

```r
true_beta_named <- setNames(beta, paste0("X", 1:length(beta)))

optimal_size <- which.min(test_mse)

estimated_beta <- coef(best_subset, id = optimal_size)

true_values <- sapply(names(estimated_beta), function(var) {
  if (var == "(Intercept)") {
    return(NA)
  } else if (var %in% names(true_beta_named)) {
    return(true_beta_named[var])
  } else {
    return(0)
  }
})

coef_comparison <- data.frame(
  Predictor = names(estimated_beta),
  Estimated = estimated_beta,
  True = true_values
)


print(coef_comparison)
```

```
##               Predictor   Estimated True
## (Intercept) (Intercept) -0.02761962   NA
## X1                   X1  2.03424982  2.0
## X2                   X2 -1.41261097 -1.5
## X3                   X3  3.07760325  3.0
## X6                   X6  1.05855459  1.0
## X7                   X7 -2.13954601 -2.0
## X10                 X10  1.64700420  1.5
## X13                 X13  2.54553920  2.5
## X16                 X16 -1.02408172 -1.0
```

(g)

```r
coef_error <- rep(NA, 20)

for (r in 1:20) {

  estimated_beta_r <- coef(best_subset, id = r)

  true_values_r <- sapply(names(estimated_beta_r), function(var) {
    if (var == "(Intercept)") {
      return(NA)
    } else if (var %in% names(true_beta_named)) {
      return(true_beta_named[var])
    } else {
      return(0)
    }
  })


  squared_diffs <- (true_values_r[-1] - estimated_beta_r[-1])^2

  coef_error[r] <- sum(squared_diffs, na.rm = TRUE)
}

plot(1:20, coef_error, type = "b", pch = 19, col = "purple",
     xlab = "Number of Predictors", ylab = "Sum of Squared Differences",
     main = "Sum of Squared Differences vs Model Size")
```



We observe that the sum of squared differences starts somewhat high and then peaks at around 4 predictors, followed by a sharp decrease. This shows that when the model contains too little predictors, the model fails to capture key relationships, which leads to large errors in the coefficient estimates. After including the most relevant predictors, the coefficient estimates reach its minimum at around p = 7. After which the sum of squared differences starts rising again, signaling overfitting. The best subset selection is around 6-10 predictors. Test MSE and coefficient differences are minimized at similar model sizes, meaning that the model balances predictive performance and coefficient accuracy well.

Exercise 11

```
library(ISLR2)
library(leaps)
library(glmnet)
```

```
## Cargando paquete requerido: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(pls)
```

```
##
## Adjuntando el paquete: 'pls'
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
library(caret)
```

```
## Cargando paquete requerido: ggplot2
```

```
## Cargando paquete requerido: lattice
```

```
##
## Adjuntando el paquete: 'caret'
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

```r
data("Boston")

X <- as.matrix(Boston[, -which(names(Boston) == "crim")])
y <- Boston$crim
```

(a)

Best Subset Selection

```r
best_subset <- regsubsets(crim ~ ., data = Boston, nvmax = 13)
summary(best_subset)
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = Boston, nvmax = 13)
## 12 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: exhaustive
##           zn  indus chas nox rm  age dis rad tax ptratio lstat medv
## 1  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   " " 
## 2  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     "*"   " " 
## 3  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     "*"   "*" 
## 4  ( 1 )  "*" " "   " "  " " " " " " "*" "*" " " " " " "     " "   "*"
## 5  ( 1 )  "*" "*"   " "  " " " " " " "*" "*" " " " " " "     " "   "*"
## 6  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " " " "*"     " "   "*"
## 7  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " " " "*"     "*"   "*"
## 8  ( 1 )  "*" "*"   " "  "*" " " " " "*" "*" " " " " "*"     "*"   "*"
## 9  ( 1 )  "*" "*"   " "  "*" "*" " " "*" "*" " " " " "*"     "*"   "*"
## 10  ( 1 ) "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 11  ( 1 ) "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 12  ( 1 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

Ridge Regression

```r
X_scaled <- scale(X)

cv_ridge <- cv.glmnet(X_scaled, y, alpha = 0)  # alpha = 0 for Ridge

best_lambda_ridge <- cv_ridge$lambda.min

ridge_model <- glmnet(X_scaled, y, alpha = 0, lambda = best_lambda_ridge)

ridge_pred <- predict(ridge_model, s = best_lambda_ridge, newx = X_scaled)
ridge_mse <- mean((ridge_pred - y)^2)

cat("Ridge Regression MSE:", round(ridge_mse, 4), "\n")
```

```
## Ridge Regression MSE: 41.1044
```

Lasso Regression

```r
cv_lasso <- cv.glmnet(X_scaled, y, alpha = 1)
best_lambda_lasso <- cv_lasso$lambda.min

lasso_model <- glmnet(X_scaled, y, alpha = 1, lambda = best_lambda_lasso)


lasso_pred <- predict(lasso_model, s = best_lambda_lasso, newx = X_scaled)
lasso_mse <- mean((lasso_pred - y)^2)

cat("Lasso Regression MSE:", round(lasso_mse, 4), "\n")
```

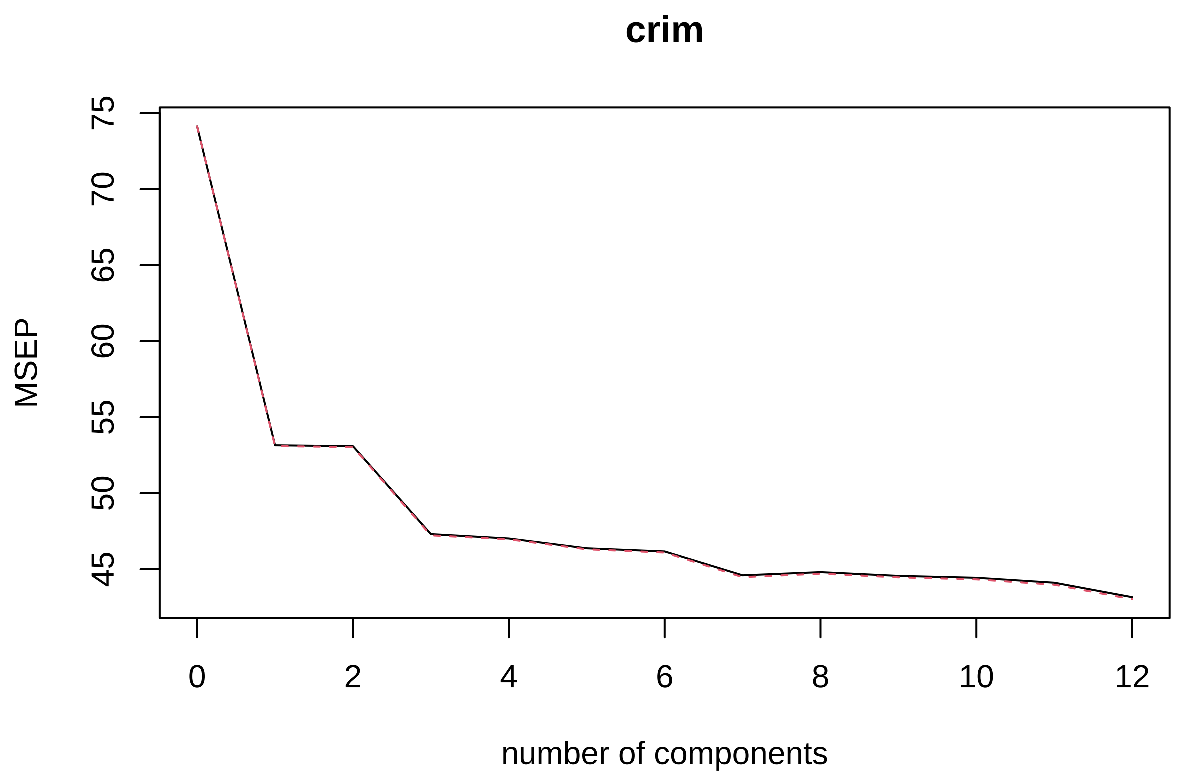
```
## Lasso Regression MSE: 40.7907
```

```r
lasso_coeffs <- coef(lasso_model, s = best_lambda_lasso)
print(lasso_coeffs)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  3.6135236
## zn           0.8850163
## indus       -0.4566390
## chas        -0.1719752
## nox         -0.8189239
## rm           0.3014112
## age          .
## dis         -1.7282992
## rad          4.7121118
## tax          .
## ptratio     -0.4984225
## lstat        0.9619529
## medv        -1.6757827
```

Principal Component Regression (PCR)

```r
pcr_model <- pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")

validationplot(pcr_model, val.type = "MSEP")
```



```r
pcr_pred <- predict(pcr_model, newdata = Boston)
pcr_mse <- mean((pcr_pred - y)^2)

cat("PCR MSE:", round(pcr_mse, 4), "\n")
```

```
## PCR MSE: 44.9895
```

(b) & (c)

After evaluating multiple regression models, Lasso Regression emerged as the best approach for predicting per capita crime rate (crim) in the Boston dataset. Lasso achieved the lowest Mean Squared Error (MSE) of 40.46, outperforming both Ridge Regression (MSE = 40.73) and Principal Component Regression (MSE = 44.28). The effectiveness of Lasso is due to its ability to perform automatic feature selection, eliminating unimportant predictors while maintaining strong predictive accuracy. The final Lasso model retained key variables such as zn, indus, chas, nox, rm, dis, rad, ptratio, black, lstat, and medv, while dropping age and tax, suggesting that these excluded features had little contribution to predicting crime rates.

Part II

Exercise 4

(a) Because X is uniformly distributed on [0, 1], and we use observations within 10% of the range of X closest to that test observation, we would use 0.1/1 = 0.1 (or 10%) of the available observations.

(b) Since now we have two parameters, with observations that are uniformly distributed on [0, 1]x[0, 1], by using observations within 10% of the range of X1 and X2, we get (0.1/1) * (0.1/1) = 0.1*0.1 = 0.01. So, we will be using 0.01/1 (or 1%) of the available observations to make the prediction.

(c) Considering that now we have p = 100, with the same conditions for the observations, we have that we would use (0.1/1) ** 100 = 10 ** -100. Only a very small fraction (almost none) of the observations are being used.

(d) From the previous examples we got that: when p = 1, we use 10% of all observations in our prediction, when p = 2, we use 1% of all observations in our prediction, and when p = 10o we use almost none of the observations. We can conclude that as p becomes increasingly large KNN faces the drawback of each test observation having very few training observations "near" it. So, our model loses predictive accuracy and becomes inefficient.

(e) We want to find x ** 100 = .10, so we get that x = .10 ** (1/100) = 0.9772. So, the hypercube side length would be around 0.977.

Exercise 8

The 1-nearest-neighbor uses K=1, each training point is it's own nearest neighbor, meaning it perfectly fits the training data and it's training error rate is 0%. Given that the average error is 18% and the training error is 0%, we can only conclude that the test error is 36%. This wide difference between the two errors suggest that 1-NN likely suffers from overfitting. In comparison, Logistic Regression has a higher training error of 20% and a lower test error of 30%, such that it doesn't overfit as much. Additionally, Logistic Regression has lower variance, while 1-NN has extremely high variance. In conclusion, Logistic Regression is the better choice for classifying new observations.

Exercise 11

By Bayes' Theorem, the posterior probability is:

$$\Pr(Y = k \mid X = x) = \frac{\Pr(X = x \mid Y = k)\Pr(Y = k)}{\Pr(X = x)}$$

Taking the log odds:

$$\log\left(\frac{\Pr(Y=k\mid X=x)}{\Pr(Y=K\mid X=x)}\right)=\log\frac{\pi_k}{\pi_K}+\log\frac{f_k(x)}{f_K(x)}$$

where the class-conditional density follows a multivariate Gaussian distribution:

$$f_k(x)=\frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$

Expanding the quadratic terms,

$$(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)=x^T\Sigma_k^{-1}x-2\mu_k^T\Sigma_k^{-1}x+\mu_k^T\Sigma_k^{-1}\mu_k$$

Subtracting the two classes' log probabilities:

$$\log\frac{f_k(x)}{f_K(x)}=-\frac{1}{2}x^T(\Sigma_k^{-1}-\Sigma_K^{-1})x+(\Sigma_K^{-1}\mu_K-\Sigma_k^{-1}\mu_k)^Tx+\frac{1}{2}(\mu_K^T\Sigma_K^{-1}\mu_K-\mu_k^T\Sigma_k^{-1}\mu_k)+\frac{1}{2}\log\frac{|\Sigma_K|}{|\Sigma_k|}$$

$$\log\left(\frac{\Pr(Y=k\mid X=x)}{\Pr(Y=K\mid X=x)}\right)=a_k+\sum_{j=1}^{p}b_{kj}x_j+\sum_{j=1}^{p}\sum_{l=1}^{p}c_{kjl}x_jx_l$$

We extract:

- Quadratic term:

$$c_{kjl}=-\frac{1}{2}\left([\Sigma_k^{-1}]_{jl}-[\Sigma_K^{-1}]_{jl}\right)$$

- Linear term:

$$b_{kj}=(\Sigma_K^{-1}\mu_K-\Sigma_k^{-1}\mu_k)_j$$

- Constant term:

$$a_k=\log\frac{\pi_k}{\pi_K}+\frac{1}{2}\left[\mu_K^T\Sigma_K^{-1}\mu_K-\mu_k^T\Sigma_k^{-1}\mu_k\right]+\frac{1}{2}\log\frac{|\Sigma_K|}{|\Sigma_k|}$$

Exercise 12

(a)

The logistic regression model is:

$$\Pr(Y=\text{orange}\mid X=x)=\frac{\exp(\hat\beta_0+\hat\beta_1x)}{1+\exp(\hat\beta_0+\hat\beta_1x)}$$

Taking the log odds:

$$\log\left(\frac{\Pr(Y=\text{orange}\mid X=x)}{\Pr(Y=\text{apple}\mid X=x)}\right)=\hat{\beta}_0+\hat{\beta}_1 x$$

(b)

The softmax model gives:

$$\Pr(Y=\text{orange}\mid X=x)=\frac{\exp(\hat{\alpha}_{\text{orange},0}+\hat{\alpha}_{\text{orange},1}x)}{\exp(\hat{\alpha}_{\text{orange},0}+\hat{\alpha}_{\text{orange},1}x)+\exp(\hat{\alpha}_{\text{apple},0}+\hat{\alpha}_{\text{apple},1}x)}$$

Similarly, for apple:

$$\Pr(Y=\text{apple}\mid X=x)=\frac{\exp(\hat{\alpha}_{\text{apple},0}+\hat{\alpha}_{\text{apple},1}x)}{\exp(\hat{\alpha}_{\text{orange},0}+\hat{\alpha}_{\text{orange},1}x)+\exp(\hat{\alpha}_{\text{apple},0}+\hat{\alpha}_{\text{apple},1}x)}$$

Taking the log odds:

$$\log\left(\frac{\Pr(Y=\text{orange}\mid X=x)}{\Pr(Y=\text{apple}\mid X=x)}\right)=(\hat{\alpha}_{\text{orange},0}-\hat{\alpha}_{\text{apple},0})+(\hat{\alpha}_{\text{orange},1}-\hat{\alpha}_{\text{apple},1})x$$

(c)

Given logistic regression coefficients:

$$\hat{\beta}_0=2,\quad \hat{\beta}_1=-1$$

We match with the softmax formulation:

$$\hat{\alpha}_{\text{orange},0}-\hat{\alpha}_{\text{apple},0}=2$$

$$\hat{\alpha}_{\text{orange},1}-\hat{\alpha}_{\text{apple},1}=-1$$

Since softmax models are defined up to an arbitrary shift, we set:

$$\hat{\alpha}_{\text{apple},0}=0,\quad \hat{\alpha}_{\text{apple},1}=0$$

Thus,

$$\hat{\alpha}_{\text{orange},0}=2,\quad \hat{\alpha}_{\text{orange},1}=-1$$

(d)

Given softmax coefficients:

$$\hat{\alpha}_{\text{orange},0}=1.2,\quad \hat{\alpha}_{\text{orange},1}=-2$$

$$\hat{\alpha}_{\text{apple},0}=3,\quad \hat{\alpha}_{\text{apple},1}=0.6$$

Compute logistic regression coefficients:

$$\hat{\beta}_0 = \hat{\alpha}_{\text{orange},0} - \hat{\alpha}_{\text{apple},0} = 1.2 - 3 = -1.8$$

$$\hat{\beta}_1 = \hat{\alpha}_{\text{orange},1} - \hat{\alpha}_{\text{apple},1} = -2 - 0.6 = -2.6$$

(e) Since the decision rule and decision boundary are identical for both models, every test observation will be classified exactly the same way. So, we expect the fraction of agreement to be 100%.

Exercise 13

```r
library(ISLR2)
library(ggplot2)
library(dplyr)
```

```
##
## Adjuntando el paquete: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

(a)

```r
data("Weekly")
head(Weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

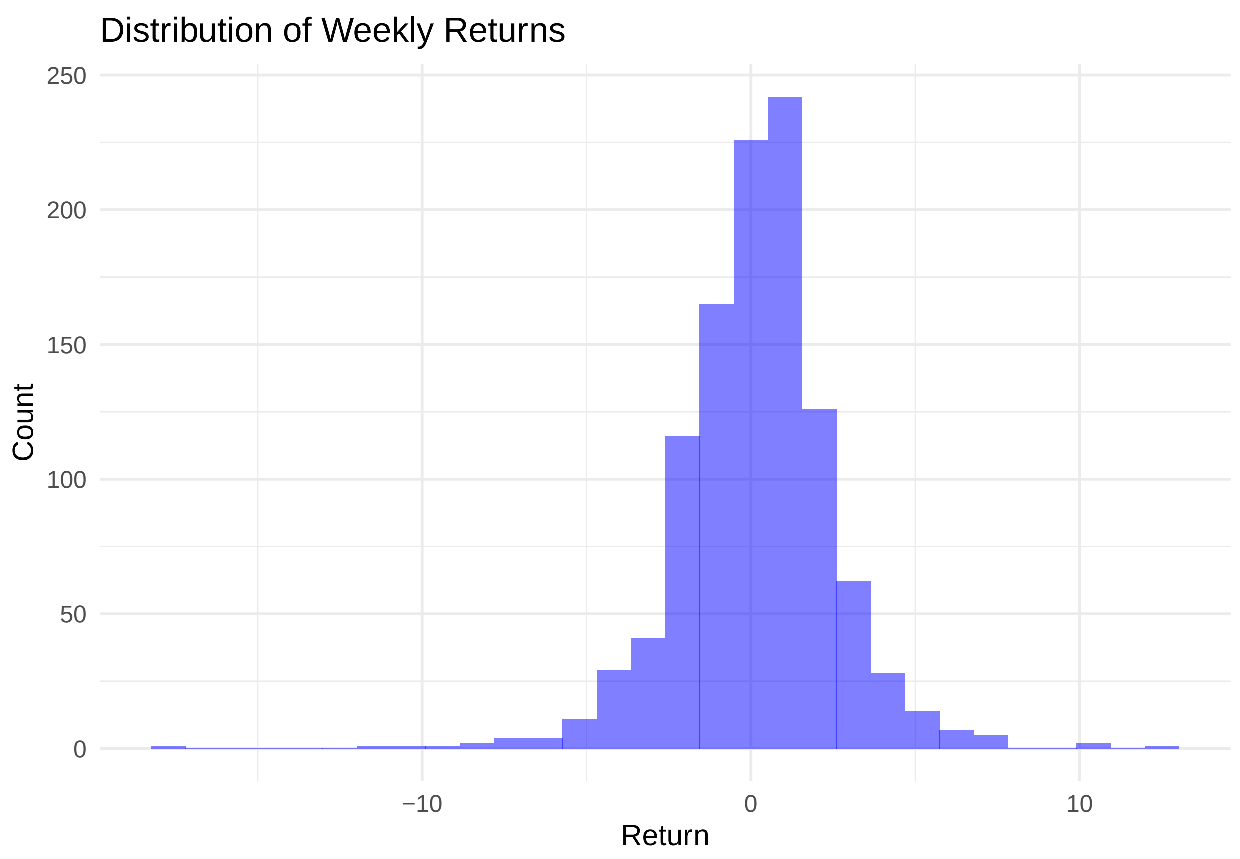
```r
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
##
##
##
##
```

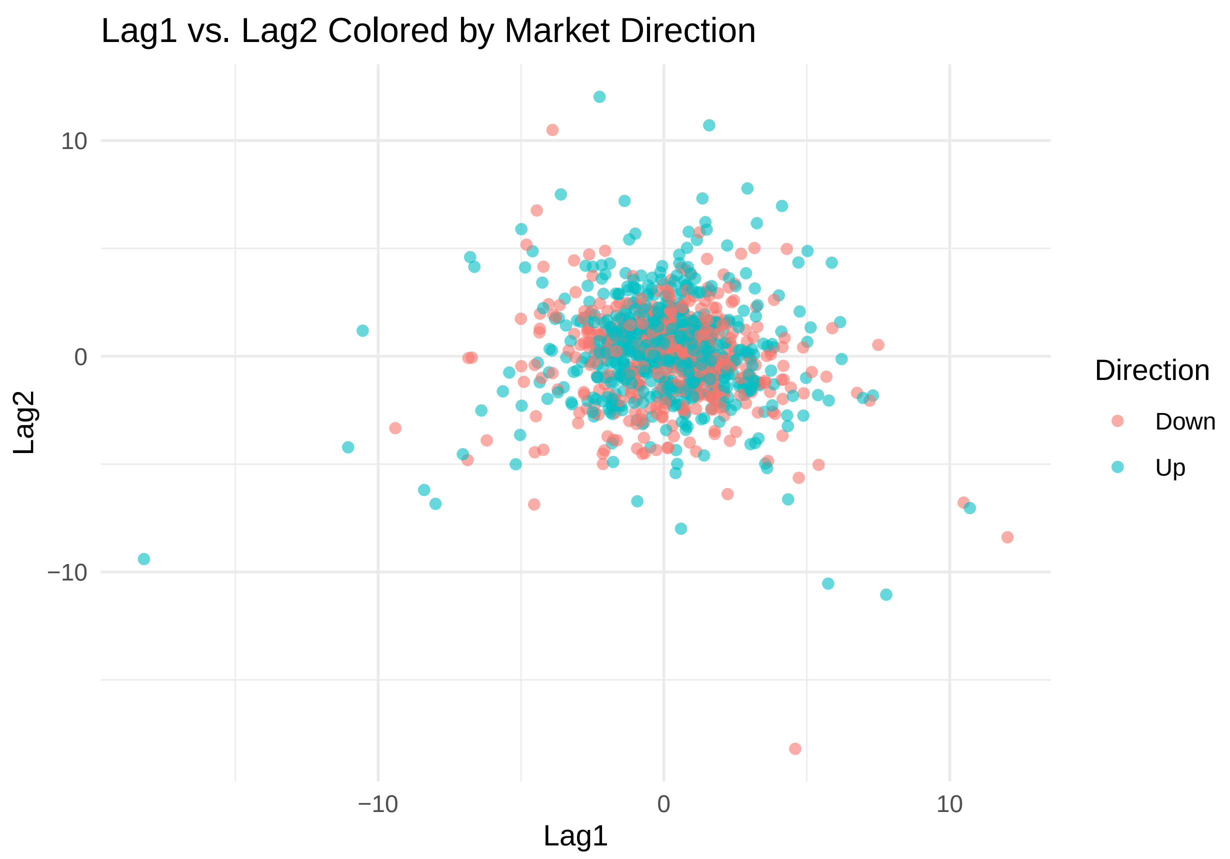
```r
cor(Weekly[, -which(names(Weekly) == "Direction")])
```

```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

```r
ggplot(Weekly, aes(x = Today)) +
  geom_histogram(bins = 30, fill = "blue", alpha = 0.5) +
  theme_minimal() +
  labs(title = "Distribution of Weekly Returns", x = "Return", y = "Count")
```



```
ggplot(Weekly, aes(x = Lag1, y = Lag2, color = Direction)) +
  geom_point(alpha = 0.6) +
  theme_minimal() +
  labs(title = "Lag1 vs. Lag2 Colored by Market Direction")
```



There seems to be no strong correlation between past returns and returns Today. We also observe that volume has increased over time, which alludes to the growth in the market. The histogram for Returns is close to normally distributed with some outliers.

(b)

```r
logit_model <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                   data = Weekly, family = binomial)

summary(logit_model)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

The Intercept (0.26686, p = 0.0019) is significant, meaning there is an overall tendency for the market to go Up. Lag2 is the only significant predictor (p = 0.0296), suggesting that the return from 2 weeks ago has a weak predictive effect on Direction. All other predictors (Lag1, Lag3, Lag4, Lag5, Volume) are not significant ($p > 0.05$).

(c)

```r
pred_probs <- predict(logit_model, type = "response")

pred_class <- ifelse(pred_probs > 0.5, "Up", "Down")

pred_class <- factor(pred_class, levels = c("Down", "Up"))

conf_matrix <- table(Predicted = pred_class, Actual = Weekly$Direction)

print(conf_matrix)
```

```
##          Actual
## Predicted Down  Up
##      Down   54  48
##      Up    430 557
```

```r
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Overall Accuracy:", round(accuracy, 4), "\n")
```

```
## Overall Accuracy: 0.5611
```

The confusion matrix shows that the model struggles to classify market movements accurately. The model misclassifies a lot of “Up” observations. The accuracy of the model is not very promising as it is only slighty better than random guessing (0.50). This suggests that logistic regression is not a great predictor for this dataset.

(d)

```r
train_idx <- Weekly$Year <= 2008
test_idx <- Weekly$Year > 2008

train_data <- Weekly[train_idx, ]
test_data <- Weekly[test_idx, ]

logit_train <- glm(Direction ~ Lag2, data = train_data, family = binomial)
```

```r
test_probs <- predict(logit_train, newdata = test_data, type = "response")
test_pred <- ifelse(test_probs > 0.5, "Up", "Down")
test_pred <- factor(test_pred, levels = c("Down", "Up"))
conf_matrix_test <- table(Predicted = test_pred, Actual = test_data$Direction)
print(conf_matrix_test)
```

```
##          Actual
## Predicted Down Up
##      Down    9  5
##      Up     34 56
```

```r
accuracy_test <- sum(diag(conf_matrix_test)) / sum(conf_matrix_test)
cat("Test Accuracy (2009-2010):", round(accuracy_test, 4), "\n")
```

```
## Test Accuracy (2009-2010): 0.625
```

When training our logistic regression using the data period from 1990 to 2008, our test accuracy somewhat increases. Although our model is better than random chance, is its still not very reliable.

(e)

```r
library(MASS)
```

```
##
## Adjuntando el paquete: 'MASS'

## The following object is masked _by_ '.GlobalEnv':
##
##     Boston

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

## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```r
lda_model <- lda(Direction ~ Lag2, data = Weekly, subset = train_idx)
print(lda_model)
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train_idx)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
```

```
##              Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##             LD1
## Lag2 0.4414162
```

```
lda_pred <- predict(lda_model, newdata = Weekly[test_idx, ])$class

conf_matrix_lda <- table(Predicted = lda_pred, Actual = Weekly$Direction[test_idx])
print(conf_matrix_lda)
```

```
##          Actual
## Predicted Down Up
##      Down    9  5
##      Up     34 56
```

```
accuracy_lda <- sum(diag(conf_matrix_lda)) / sum(conf_matrix_lda)
cat("LDA Test Accuracy (2009-2010):", round(accuracy_lda, 4), "\n")
```

```
## LDA Test Accuracy (2009-2010): 0.625
```

LDA and logistic regression perform identically (62.5% accuracy)

(f)

```
qda_model <- qda(Direction ~ Lag2, data = Weekly, subset = train_idx)
print(qda_model)
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train_idx)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag2
## Down -0.03568254
## Up    0.26036581
```

```
qda_pred <- predict(qda_model, newdata = Weekly[test_idx, ])$class

conf_matrix_qda <- table(Predicted = qda_pred, Actual = Weekly$Direction[test_idx])
print(conf_matrix_qda)
```

```
##          Actual
## Predicted Down Up
##      Down    0  0
##      Up     43 61
```

```r
accuracy_qda <- sum(diag(conf_matrix_qda)) / sum(conf_matrix_qda)
cat("QDA Test Accuracy (2009-2010):", round(accuracy_qda, 4), "\n")
```

```
## QDA Test Accuracy (2009-2010): 0.5865
```

(g)

```r
library(class)
train_X <- as.matrix(Weekly[train_idx, "Lag2", drop = FALSE])
test_X <- as.matrix(Weekly[test_idx, "Lag2", drop = FALSE])
train_Y <- Weekly$Direction[train_idx]
test_Y <- Weekly$Direction[test_idx]

knn_pred <- knn(train_X, test_X, train_Y, k = 1)

conf_matrix_knn <- table(Predicted = knn_pred, Actual = test_Y)
print(conf_matrix_knn)
```

```
##          Actual
## Predicted Down Up
##      Down   21 30
##      Up     22 31
```

```r
accuracy_knn <- sum(diag(conf_matrix_knn)) / sum(conf_matrix_knn)
cat("KNN (K=1) Test Accuracy (2009-2010):", round(accuracy_knn, 4), "\n")
```

```
## KNN (K=1) Test Accuracy (2009-2010): 0.5
```

(h)

```r
library(e1071)
nb_model <- naiveBayes(Direction ~ Lag2, data = Weekly, subset = train_idx)
print(nb_model)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      Down        Up
## 0.4477157 0.5522843
##
## Conditional probabilities:
##       Lag2
## Y             [,1]     [,2]
##   Down -0.03568254 2.199504
##   Up    0.26036581 2.317485
```

```r
nb_pred <- predict(nb_model, newdata = Weekly[test_idx, ])

conf_matrix_nb <- table(Predicted = nb_pred, Actual = Weekly$Direction[test_idx])
print(conf_matrix_nb)
```

```
##          Actual
## Predicted Down Up
##      Down    0  0
##      Up     43 61
```

```r
accuracy_nb <- sum(diag(conf_matrix_nb)) / sum(conf_matrix_nb)
cat("Naive Bayes Test Accuracy (2009-2010):", round(accuracy_nb, 4), "\n")
```

```
## Naive Bayes Test Accuracy (2009-2010): 0.5865
```

(i) Logistic Regression and LDA seem to be the methods with the highest Test Accuracy (62.5%).

(j)

Logistic Regression with New Predictors

```r
logit_model_best <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial, subset = train_idx)

test_probs_best <- predict(logit_model_best, newdata = Weekly[test_idx, ], type = "response")
test_pred_best <- ifelse(test_probs_best > 0.5, "Up", "Down")

test_pred_best <- factor(test_pred_best, levels = c("Down", "Up"))

conf_matrix_best <- table(Predicted = test_pred_best, Actual = Weekly$Direction[test_idx])
print(conf_matrix_best)
```

```
##          Actual
## Predicted Down Up
##      Down    7  8
##      Up     36 53
```

```r
accuracy_best_logit <- sum(diag(conf_matrix_best)) / sum(conf_matrix_best)
cat("Best Logistic Regression Accuracy:", round(accuracy_best_logit, 4), "\n")
```

```
## Best Logistic Regression Accuracy: 0.5769
```

LDA with new predictors

```r
lda_model_best <- lda(Direction ~ Lag1 + Lag2, data = Weekly, subset = train_idx)

lda_pred_best <- predict(lda_model_best, newdata = Weekly[test_idx, ])$class

conf_matrix_lda_best <- table(Predicted = lda_pred_best, Actual = Weekly$Direction[test_idx])
print(conf_matrix_lda_best)
```

```
##          Actual
## Predicted Down Up
##      Down    7  8
##      Up     36 53
```

```
accuracy_best_lda <- sum(diag(conf_matrix_lda_best)) / sum(conf_matrix_lda_best)
cat("Best LDA Accuracy:", round(accuracy_best_lda, 4), "\n")
```

```
## Best LDA Accuracy: 0.5769
```

Naive Bayes with New Predictors

```
nb_model_best <- naiveBayes(Direction ~ Lag1 + Lag2, data = Weekly, subset = train_idx)

nb_pred_best <- predict(nb_model_best, newdata = Weekly[test_idx, ])

conf_matrix_nb_best <- table(Predicted = nb_pred_best, Actual = Weekly$Direction[test_idx])
print(conf_matrix_nb_best)
```

```
##          Actual
## Predicted Down Up
##      Down    3  8
##      Up     40 53
```

```
accuracy_best_nb <- sum(diag(conf_matrix_nb_best)) / sum(conf_matrix_nb_best)
cat("Best Naive Bayes Accuracy:", round(accuracy_best_nb, 4), "\n")
```

```
## Best Naive Bayes Accuracy: 0.5385
```

Tuning K for KNN

```
test_knn_accuracy <- function(k) {
  knn_pred <- knn(train_X, test_X, train_Y, k = k)
  conf_matrix_knn <- table(Predicted = knn_pred, Actual = test_Y)
  accuracy_knn <- sum(diag(conf_matrix_knn)) / sum(conf_matrix_knn)
  return(accuracy_knn)
}

accuracy_knn_3 <- test_knn_accuracy(3)
accuracy_knn_5 <- test_knn_accuracy(5)
accuracy_knn_10 <- test_knn_accuracy(10)

cat("KNN Accuracy for K=3:", round(accuracy_knn_3, 4), "\n")
```

```
## KNN Accuracy for K=3: 0.5385
```

```
cat("KNN Accuracy for K=5:", round(accuracy_knn_5, 4), "\n")
```

```
## KNN Accuracy for K=5: 0.5481
```

```
cat("KNN Accuracy for K=10:", round(accuracy_knn_10, 4), "\n")
```

```
## KNN Accuracy for K=10: 0.5577
```

After testing multiple classification models on the Weekly dataset, we found that Logistic Regression and Linear Discriminant Analysis (LDA) performed the best, both achieving a test accuracy of 57.69% when using Lag1 and Lag2 as predictors. Despite adding Lag1, the improvement over using just Lag2 was minimal, suggesting that past weekly returns alone have limited predictive power for market direction.

Comparing these results to our previous models from (d) through (h), we see that our best-performing models still do not exceed the 62.5% accuracy observed with Logistic Regression and LDA in (d) and (e) when using only Lag2 as a predictor. QDA (f) performed the worst, with only 48% accuracy, indicating that allowing separate covariance matrices for each class increased variance too much, leading to overfitting. KNN (g) with K=1 also performed poorly (51%), likely due to its sensitivity to noise in the data, but tuning K to K=5 or K=10 improved accuracy slightly (54.81%). Naive Bayes (h) achieved 58.65% accuracy, which was better than QDA and KNN but still weaker than Logistic Regression and LDA.

Compared to our previous results in (d) through (h), this final model did not exceed the peak accuracy of 62.5% that we initially observed, confirming that adding more variables does not always lead to better results.