IDS575 HW2

Scott Brewer 676075252

Ono Gantsog

ESL2 Ex. 3.6:

Show that the ridge regression estimate is the mean (and mode) of the posterior distribution, under a Gaussian prior β~N(0,τI), and Gaussian sampling model y~N(Xβ,σ2I). Find the relationship between the regularization parameter λ in the ridge formula, and the variances τ and σ2.

$P(\beta|Y,X) \propto P(Y,X|\beta)P(\beta)$

Posterior ∝ (Likelihood)(Prior)

Prior: $\beta \sim N(0, \tau I)$

Likelihood: $\mathbf{Y} \sim N(\mathbf{X}\beta, \sigma^2 I)$

Gaussian Form:

Prior: $1/(\mathrm{sqrt}(2\pi\ \mathrm{sqrt}(\tau I))\ \exp((-1/(2\tau I))(\beta - 0)^2)$

Likelihood: $1/(\mathrm{sqrt}(2\pi\ \mathrm{sqrt}(\sigma^2 I))\ \exp((-1/(2\sigma^2 I))(\mathbf{Y} - \mathbf{X}\beta)^2)$

First, simplify constant terms:

Prior: $C_1 \exp((-1/(2\tau I))(\beta - 0)^2)$

Likelihood: $C_2 \exp((-1/(2\sigma^2 I))(\mathbf{Y} - \mathbf{X}\beta)^2)$

Now, plug into posterior right hand side:

$P(\beta|Y,X) \propto P(Y,X|\beta)P(\beta)$

$P(\beta|Y,X) \propto [C_2 \exp((-1/(2\sigma^2 I))(\mathbf{Y} - \mathbf{X}\beta)^2)][C_1 \exp((-1/(2\tau I))(\beta - 0)^2)]$

Now, take -log of both sides to simplify combination of terms:

$-\log(P(\beta|Y,X)) \propto \log C_2(1/(2\sigma^2 I))(\mathbf{Y} - \mathbf{X}\beta)^2 + \log C_1(1/(2\tau I))(\beta - 0)^2$

Now, collect constant terms together again:

$-\log(P(\beta|Y,X)) \propto C_3(\mathbf{Y} - \mathbf{X}\beta)^2 + C_4(\beta - 0)^2$

Now, express in vector form and simplify constants:

$-\log(\mathrm{Posterior}) \propto (\sigma^2 I)^{-1} (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + (2I)^{-1} (\beta - 0)^T(\beta - 0)$

Bring all constants to final term and simplify final term:

$-\log(\mathrm{Posterior}) \propto (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + C_5\beta^T\beta$

Note that this form matches the RSS(β) + $\lambda\beta^T\beta$ of Ridge Regression.

ISLR Ex. 3.14[a-f]:

This problem focuses on the collinearity problem.

(a) Perform the following commands in R:

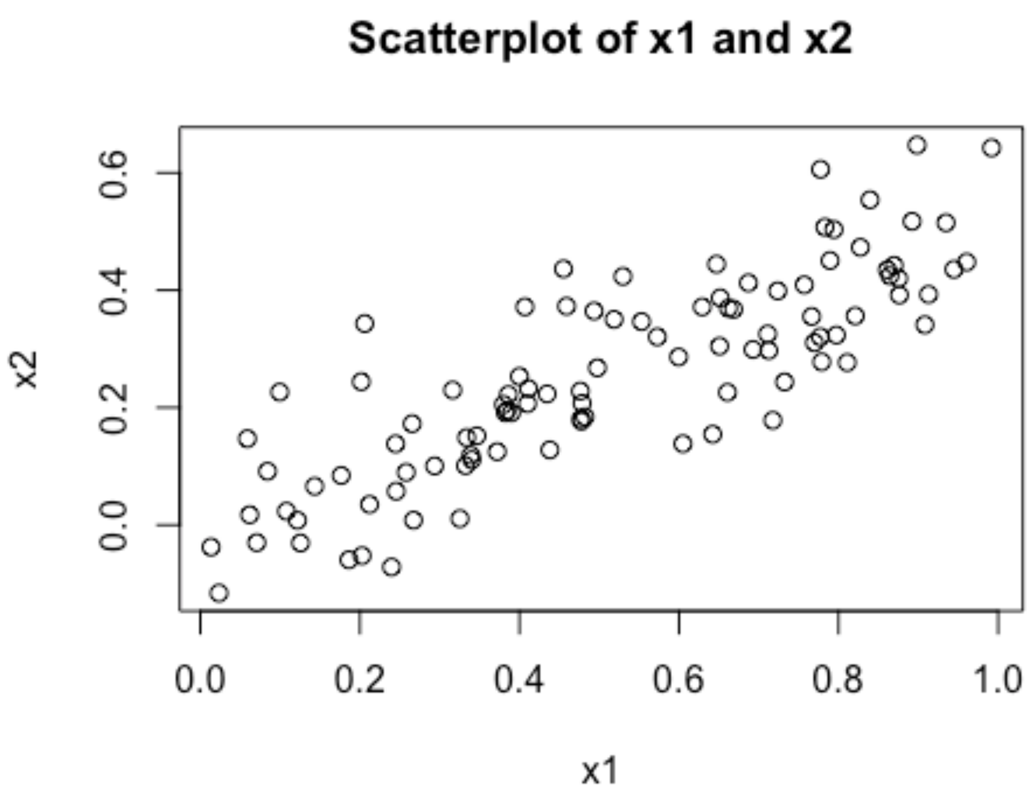
```
set.seed(1)
x1=runif(100)
x2=0.5*x1+rnorm(100)/10
y=2+2*x1+0.3*x2+rnorm(100)
```

The last line corresponds to creating a linear model in which y is a function of x1 and x2. Write out the form of the linear model. What are the regression coefficients?

Linear Model: y = 2 + 2*x1 + .3*x2 + error Intercept (β0) is 2 Regression coefficient for x1 (β1) is 2 Regression coefficient for x2 (β2) is 0.3

(b) What is the correlation between x1 and x2? Create a scatterplot displaying the relationship between the variables.

```
plot(x1, x2, main="Scatterplot of x1 and x2")
```



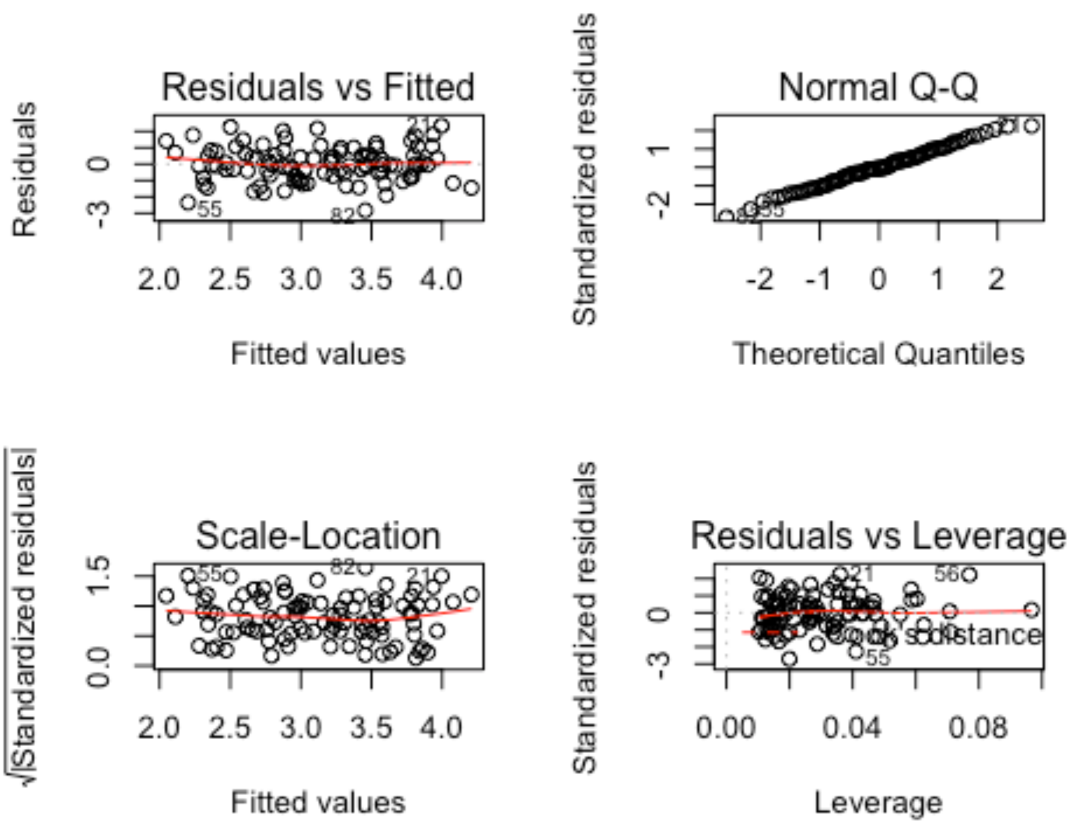
From the graph, we can see that x1 and x2 have a relatively strong positive correlation, which is confirmed with the cor() function:

```
cor(x1,x2)

## [1] 0.8351212
```

(c) Using this data, fit a least squares regression to predict y using x1 and x2. Describe the results obtained. What are $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$? How do these relate to the true β_0, β_1, and β_2? Can you reject the null hypothesis $H_0 : \beta_1 = 0$? How about the null hypothesis $H_0 : \beta_2 = 0$?

```
lr <- lm(y ~ x1 + x2)
summary(lr)

##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8311 -0.7273 -0.0537  0.6338  2.3359
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   2.1305     0.2319   9.188 7.61e-15 ***
## x1            1.4396     0.7212   1.996   0.0487 *
## x2            1.0097     1.1337   0.891   0.3754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 97 degrees of freedom
## Multiple R-squared:  0.2088, Adjusted R-squared:  0.1925
## F-statistic:  12.8 on 2 and 97 DF,  p-value: 1.164e-05

par(mfrow=c(2,2))
plot(lr)
```

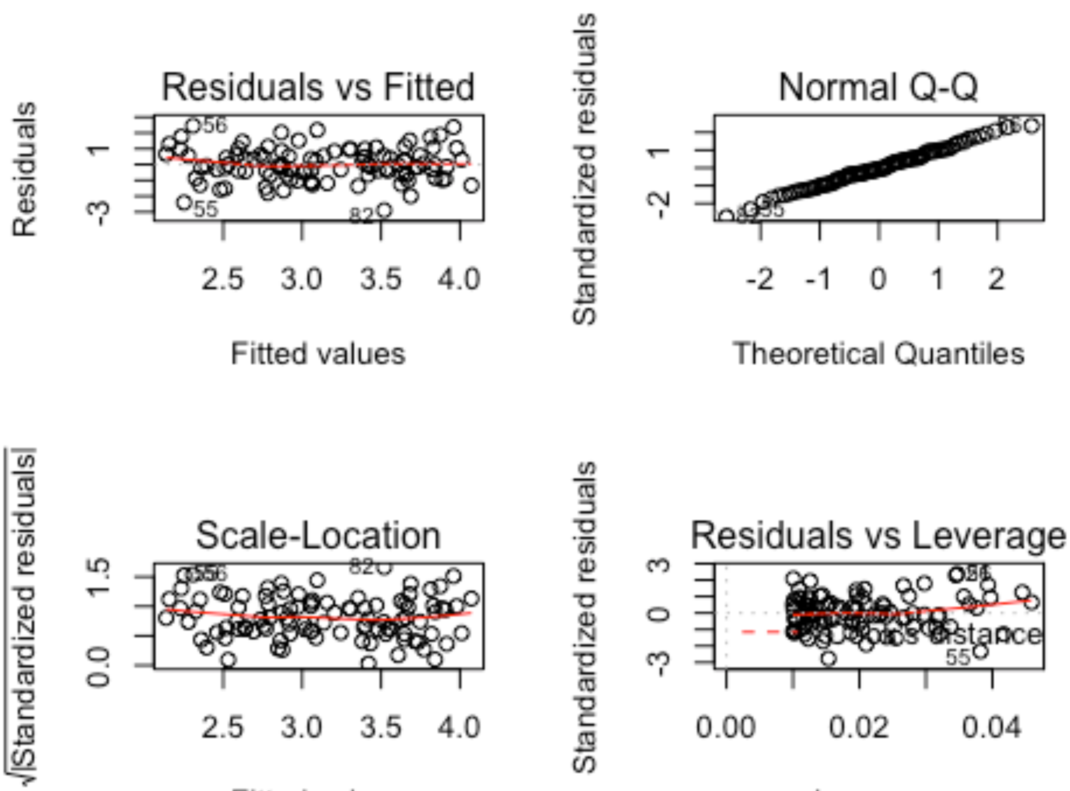


We can reject the null hypothesis for x1, because p-value for x1 = 0.049 which is less than 0.05, but we can't reject the null hypothesis for x2 because the p-value for x2=0.38.
The regression coefficients are:
β0hat = 2.13
β1hat = 1.44
β2hat = 1.01
But we know that the true coefficients are:
β0 = 2
β1 = 2
β2 = .3

Obviously fitted coefficients are similar to the true coefficients, but are also different - especially β2hat.

(d) Now fit a least squares regression to predict y using only x1. Comment on your results. Can you reject the null hypothesis H0 :β1 =0?

```
lr1 <- lm(y ~ x1)
summary(lr1)

##
## Call:
## lm(formula = y ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89495 -0.66874 -0.07785  0.59221  2.45560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1124     0.2307   9.155 8.27e-15 ***
## x1            1.9759     0.3963   4.986 2.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.055 on 98 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.1942
## F-statistic: 24.86 on 1 and 98 DF,  p-value: 2.661e-06

par(mfrow=c(2,2))
plot(lr1)
```



The regression coefficients are now:
β0hat = 2.11
β1hat = 1.98
Which are very similar to the true coefficients.
Additionally, we can reject the null hypothesis because the p value for x1 is very small.

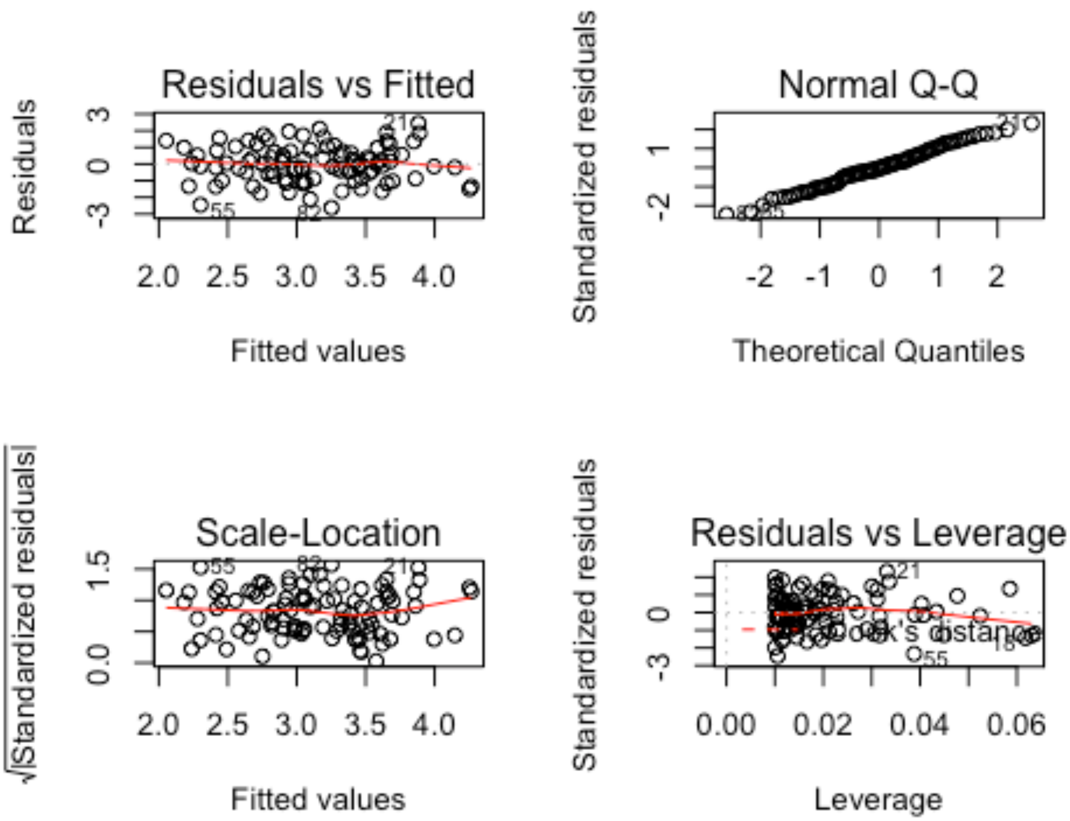
(e) Now fit a least squares regression to predict y using only x2. Comment on your results. Can you reject the null hypothesis H0 :β1 =0?

```r
lr2 <- lm(y ~ x2)
summary(lr2)
```

```
##
## Call:
## lm(formula = y ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## x2            2.8996     0.6330    4.58 1.37e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

```r
par(mfrow=c(2,2))
plot(lr2)
```



The regression coefficients are now:
β0hat = 2.39
β2hat = 2.90
Which are different than the true coefficients.
Additionally, we can reject the null hypothesis because the p value for x2 is very small.

(f) Do the results obtained in (c)–(e) contradict each other? Explain your answer.

The results from parts c-e don't contradict each other. Instead, they are showing that there is a degree of collinearity between x1 and x2. We can calculate VIF to determine the degree of collinearity:

```r
library(car)
vif(lr)

##       x1       x2
## 3.304993 3.304993
```

These results indicate that there is collinearity between x1 and x2, but as the VIF values are less than the 5-10 range, it is not yet problematic.

ISLR Ex. 4.5:

We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

For a linear Bayes decision boundary, we expect LDA to perform better than QDA on both the training and test sets, with performance on the training set being better than the performance on the test set. This is due to a linear decision boundary being indicative of common covariance matrices between the two classes, which is an assumption of LDA, but not QDA. So, while this assumption holds, LDA should have both low variance and low bias.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

For a non-linear Bayes decision boundary, we expect QDA to perform better than LDA on both the training and test sets, with performance on the training set being better than the performance on the test set. This is due to the non-linear decision boundary being indicative of unequal covariance matrices between the two classes, which is an assumption of QDA, but not LDA. So, while this assumption holds, QDA should have both low variance and low bias.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

Generally, as the sample size increases, we expect test prediction accuracy of QDA to increase relative to LDA. This expectation is due to the inherent variation difference between the models due to thier assumptions about covariance matrices among classes between the LDA and QDA. Since LDA assumes each class has a common covariance matrix, the number of matrices to calculate is a factor of p varaiables squared, whereas a different covariance matrix for each class yields a factor of K*p^2 covariance matrices, which greatly increases the variance of QDA vs LDA. With this background, we would generally prefer LDA vs QDA for smaller data sets (assuming that the data at least roughly adheres to LDA's assumptions). But as data size increases, the advantage of lower variability inherent in LDA is lost since the variablity of any series of numbers evens out due to thoeries of central tendency (assuming no extreme outliers). So, once data is sufficiently large, the increased flexibility of QDA is worth the the increased variability since the relative magnitudes of variability have gotten closer.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

False. While it is true that QDA is a more flexible model than LDA, that flexibility comes with the potential to overfit on training data, so while QDA may generate better training error rates, it's inherently greater variability will always generate worse test error rates that LDA assuming a linear decision boundary.

ISLR Ex. 4.6:

Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, β^0 = −6, β^1 = 0.05, β^2 = 1.

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

```
hrs <- 40
gpa <- 3.5
#log(p/(1-p)) = b0 + b1*hours + b2*gpa
log.odds <- -6 + .05*hrs + 1*gpa
odds <- exp(log.odds)
prob <- odds/(1+odds)
prob

## [1] 0.3775407
```

The student has a probability of 0.38 of getting an A in the class.

(b) How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class?

```
gpa <- 3.5
prob <- .5
log.odds <- log(prob/(1-prob))
hrs <- (log.odds + 6 - 1*gpa)/.05
hrs

## [1] 50
```

According to the model, the same student would have to study 50 hrs to have a 50% chance of getting an A in the class.

ISLR Ex. 4.10:

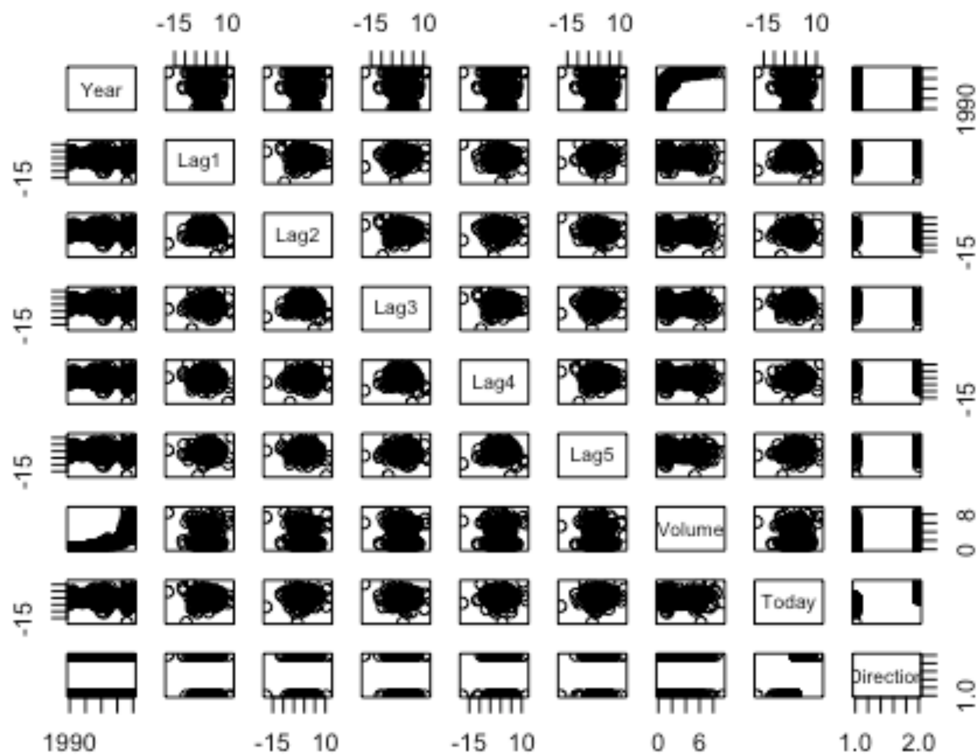
This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

```
library(ISLR)
df <- Weekly
```

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
names(df)

## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"

dim(df)

## [1] 1089    9

summary(df)

##       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
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260

cor(df[,-9])

##               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

plot(df)
```



(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data = df, family='
binomial')
summary(glm.fit)

##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
```

```
## 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
```

Lag2 is statistically significant with a p value of .0296 and Lag1 is almost statistically significant with a p value of .1181.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
glm.prob <- predict(glm.fit, newdata = df, type = 'response')
glm.pred <- ifelse(glm.prob>.5, 'Up', 'Down')
table(glm.pred,df$Direction)

##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557

mean(glm.pred==df$Direction)

## [1] 0.5610652
```

The confusion matrix is a table with actual values crossed with predicted values resulting in four quadrants (for a binary class), with the diagonal indicating True Negative and True Positive Predictions, and the off diagonal values indicating False Negatives and False Positives. The confusion matrix is telling us that the logistic model has an accuracy of 55%.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
train <- which(df$Year<2009)
glm.fit <- glm(Direction~Lag2, data = df, subset = train, family='binomial')
glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
```

```
glm.pred <- ifelse(glm.prob>.5, 'Up', 'Down')
table(glm.pred,df$Direction[-train])

##
## glm.pred Down Up
##     Down    9  5
##     Up     34 56

mean(glm.pred==df$Direction[-train])

## [1] 0.625
```

(e) Repeat (d) using LDA.

```
library(MASS)
train <- which(df$Year<2009)
lda.fit <- lda(Direction~Lag2, data = df, subset = train)
lda.pred <- predict(lda.fit, newdata = df[-train,])
table(lda.pred$class,df$Direction[-train])

##
##        Down Up
##   Down    9  5
##   Up     34 56

mean(lda.pred$class==df$Direction[-train])

## [1] 0.625
```

(f) Repeat (d) using QDA.

```
library(MASS)
train <- which(df$Year<2009)
qda.fit <- qda(Direction~Lag2, data = df, subset = train)
qda.pred <- predict(qda.fit, newdata = df[-train,])
table(qda.pred$class,df$Direction[-train])

##
##        Down Up
##   Down    0  0
##   Up     43 61

mean(qda.pred$class==df$Direction[-train])

## [1] 0.5865385
```

(g) Repeat (d) using KNN with K = 1.

```
library(class)
train <- which(df$Year<2009)
train.X <- as.data.frame(df[train,which(names(df)=='Lag2')])
test.X <- as.data.frame(df[-train,which(names(df)=='Lag2')])

train.Y <- df[train,]$Direction
test.Y <- df[-train,]$Direction
```

```r
set.seed(1)
knn.pred <- knn(train.X,test.X,train.Y,k=1)
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```r
mean(knn.pred==test.Y)
```

```
## [1] 0.5
```

(h) Which of these methods appears to provide the best results on this data?

LDA and Logistic Regression both predict Direction on test data (2009 and 2010) with an accuracy of 62.5% vs QDA with 58.7% and KNN=1 at 50%. So, for this data, Logistic Regression or LDA would be preferred.

(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

KNN with predictor Lag1 and k=1 to k=25

```r
library(class)
train <- which(df$Year<2009)

train.X <- as.data.frame(df[train,which(names(df)=='Lag1')])
test.X <- as.data.frame(df[-train,which(names(df)=='Lag1')])

train.Y <- df[train,]$Direction
test.Y <- df[-train,]$Direction

acc <- vector(length=25)
for (k in 1:25){
  set.seed(1)
  knn.pred <- knn(train.X,test.X,train.Y,k=k)
  acc[k] <- mean(knn.pred==test.Y)
}
#plot(1:25,acc, type='l')
k.opt <- min(which(acc==max(acc)))
set.seed(1)
knn.pred <- knn(train.X,test.X,train.Y,k=k.opt)
print(paste('Optimal k =',k.opt))
```

```
## [1] "Optimal k = 17"
```

```r
table(knn.pred,test.Y)
```

```
##          test.Y
## knn.pred Down Up
##     Down   14 17
##     Up     29 44

mean(knn.pred==test.Y)

## [1] 0.5576923
```

KNN with predictor all predictors except Today and Year and k=1 to k=25

```
library(class)
train <- which(df$Year<2009)

train.X <- df[train,2:7] # Select all but Year and Today
test.X <- df[-train,2:7]

train.Y <- df[train,]$Direction
test.Y <- df[-train,]$Direction

acc <- vector(length=25)
for (k in 1:25){
  set.seed(1)
  knn.pred <- knn(train.X,test.X,train.Y,k=k)
  acc[k] <- mean(knn.pred==test.Y)
}
#plot(1:25,acc, type='l')
k.opt <- min(which(acc==max(acc)))
set.seed(1)
knn.pred <- knn(train.X,test.X,train.Y,k=k.opt)
print(paste('Optimal k =',k.opt))

## [1] "Optimal k = 11"

table(knn.pred,test.Y)

##          test.Y
## knn.pred Down Up
##     Down   21 19
##     Up     22 42

mean(knn.pred==test.Y)

## [1] 0.6057692
```

LDA with Lag1, Lag2 as predictors and polynomials

```
library(MASS)
train <- which(df$Year<2009)
lda.fit <- lda(Direction~Lag1+Lag2, data = df, subset = train)
lda.pred <- predict(lda.fit, newdata = df[-train,])
lda.class <- ifelse(lda.pred$posterior[,2]>.5,'Up','Down')
table(lda.pred$class,df$Direction[-train])
```

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

```
mean(lda.pred$class==df$Direction[-train])
```

```
## [1] 0.5769231
```

Logistic regression with Lag1, Lag2 as predictors

```
train <- which(df$Year<2009)
glm.fit <- glm(Direction ~  Lag1 + Lag2, data = df, subset = train, family='b
inomial')
glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
glm.pred <- ifelse(glm.prob>.5, 'Up', 'Down')
table(glm.pred,df$Direction[-train])
```

```
##
## glm.pred Down Up
##     Down    7  8
##     Up     36 53
```

```
mean(glm.pred==df$Direction[-train])
```

```
## [1] 0.5769231
```

```
glm.fit$formula
```

```
## Direction ~ Lag1 + Lag2
```

Logistic Regression with Lag2 as predictors trying increasing polynomials 1-5 power

```
train <- which(df$Year<2009)
acc <- vector(length = 6)
for (i in 1:6) {
  glm.fit <- glm(Direction~poly(Lag2,i), data = df, subset = train, family='b
inomial')
  glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
  glm.pred <- ifelse(glm.prob>.5, 'Up', 'Down')
  table(glm.pred,df$Direction[-train])
  acc[i] <- mean(glm.pred==df$Direction[-train])
}
#plot(1:10,acc, type='l')
pwr.opt <- which(acc==max(acc))
print(paste('Optimal power =',pwr.opt))
```

```
## [1] "Optimal power = 1" "Optimal power = 2" "Optimal power = 6"
```

```
acc[pwr.opt]
```

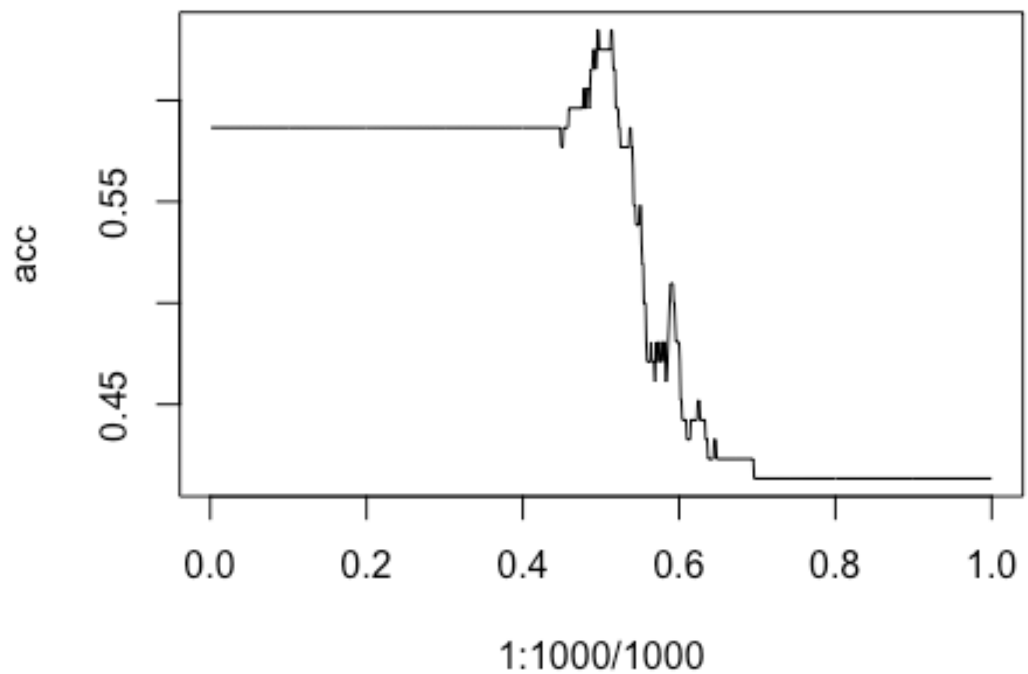
```
## [1] 0.625 0.625 0.625
```

Logistic regression with Lag1, Lag2 as one predictors in the form of Lag1*Lag2

```
train <- which(df$Year<2009)
glm.fit <- glm(Direction ~  Lag1*Lag2, data = df, subset = train, family='bin
omial')
glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
glm.pred <- ifelse(glm.prob>.5, 'Up', 'Down')
table(glm.pred,df$Direction[-train])

##
## glm.pred Down Up
##     Down    7  8
##     Up     36 53

mean(glm.pred==df$Direction[-train])

## [1] 0.5769231

glm.fit$formula

## Direction ~ Lag1 * Lag2
```

Logistic regression with Lag2 as predictor for finding cutoff point for classification

```
train <- which(df$Year<2009)
acc <- vector(length=1000)
for (i in 1:1000) {
  glm.fit <- glm(Direction ~ Lag2, data = df, subset = train, family='binomia
l')
  glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
  glm.pred <- ifelse(glm.prob>i/1000, 'Up', 'Down')
  acc[i] <- mean(glm.pred==df$Direction[-train])
}

cutoff.opt <- which(acc==max(acc))/1000
print(paste('Optimal cutoff =',cutoff.opt, 'with accuracy',acc[cutoff.opt*100
0]))

## [1] "Optimal cutoff = 0.496 with accuracy 0.634615384615385"
## [2] "Optimal cutoff = 0.497 with accuracy 0.634615384615385"
## [3] "Optimal cutoff = 0.513 with accuracy 0.634615384615385"
## [4] "Optimal cutoff = 0.514 with accuracy 0.634615384615385"

glm.fit$formula

## Direction ~ Lag2

plot(1:1000/1000,acc,type='l')
```



Subset selection for Logistic regression

```
train <- which(df$Year<2009)
glm.fit <- glm(Direction ~  .-Year-Today, data = df, subset = train, family='
binomial')
glm.fit <- step(glm.fit, direction='both')
```

```
## Start:  AIC=1356.33
## Direction ~ (Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume +
##     Today) - Year - Today
##
##          Df Deviance    AIC
## - Lag3    1   1342.6 1354.6
## - Lag4    1   1343.5 1355.5
## - Lag5    1   1344.0 1356.0
## <none>        1342.3 1356.3
## - Lag2    1   1344.6 1356.6
## - Volume  1   1345.1 1357.1
## - Lag1    1   1346.9 1358.9
##
## Step:  AIC=1354.61
## Direction ~ Lag1 + Lag2 + Lag4 + Lag5 + Volume
##
##          Df Deviance    AIC
```

```
## - Lag4    1   1343.6 1353.6
## - Lag5    1   1344.3 1354.3
## <none>        1342.6 1354.6
## - Lag2    1   1345.1 1355.1
## - Volume  1   1345.2 1355.2
## + Lag3    1   1342.3 1356.3
## - Lag1    1   1347.3 1357.3
##
## Step:  AIC=1353.64
## Direction ~ Lag1 + Lag2 + Lag5 + Volume
##
##          Df Deviance    AIC
## - Lag5    1   1345.1 1353.1
## <none>        1343.6 1353.6
## - Volume  1   1345.9 1353.9
## - Lag2    1   1346.0 1354.0
## + Lag4    1   1342.6 1354.6
## + Lag3    1   1343.5 1355.5
## - Lag1    1   1348.0 1356.0
##
## Step:  AIC=1353.14
## Direction ~ Lag1 + Lag2 + Volume
##
##          Df Deviance    AIC
## - Volume  1   1347.0 1353.0
## <none>        1345.1 1353.1
## + Lag5    1   1343.6 1353.6
## - Lag2    1   1347.8 1353.8
## + Lag4    1   1344.3 1354.3
## + Lag3    1   1344.9 1354.9
## - Lag1    1   1349.4 1355.4
##
## Step:  AIC=1352.96
## Direction ~ Lag1 + Lag2
##
##          Df Deviance    AIC
## <none>        1347.0 1353.0
## + Volume  1   1345.1 1353.1
## + Lag5    1   1345.9 1353.9
## + Lag4    1   1346.4 1354.4
## - Lag2    1   1350.5 1354.5
## - Lag1    1   1350.5 1354.5
## + Lag3    1   1346.9 1354.9
```

```r
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = df,
##     subset = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6149  -1.2565   0.9989   1.0875   1.5330
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.21109    0.06456   3.269  0.00108 **
## Lag1        -0.05421    0.02886  -1.878  0.06034 .
## Lag2         0.05384    0.02905   1.854  0.06379 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1347.0  on 982  degrees of freedom
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4

glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
glm.pred <- ifelse(glm.prob>.5, 'Up', 'Down')
table(glm.pred,df$Direction[-train])

##
## glm.pred Down Up
##     Down    7  8
##     Up     36 53

mean(glm.pred==df$Direction[-train])

## [1] 0.5769231

glm.fit$formula

## Direction ~ Lag1 + Lag2
```

Logistic regression with Lag1, Lag2 as predictors and plot for finding cutoff point

```
train <- which(df$Year<2009)
acc <- vector(length=200)
for (i in 1:1000) {
  glm.fit <- glm(Direction ~ Lag1+Lag2, data = df, subset = train, family='bi
nomial')
  glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
  glm.pred <- ifelse(glm.prob>i/1000, 'Up', 'Down')
  acc[i] <- mean(glm.pred==df$Direction[-train])
}

cutoff.opt <- which(acc==max(acc))/1000
```

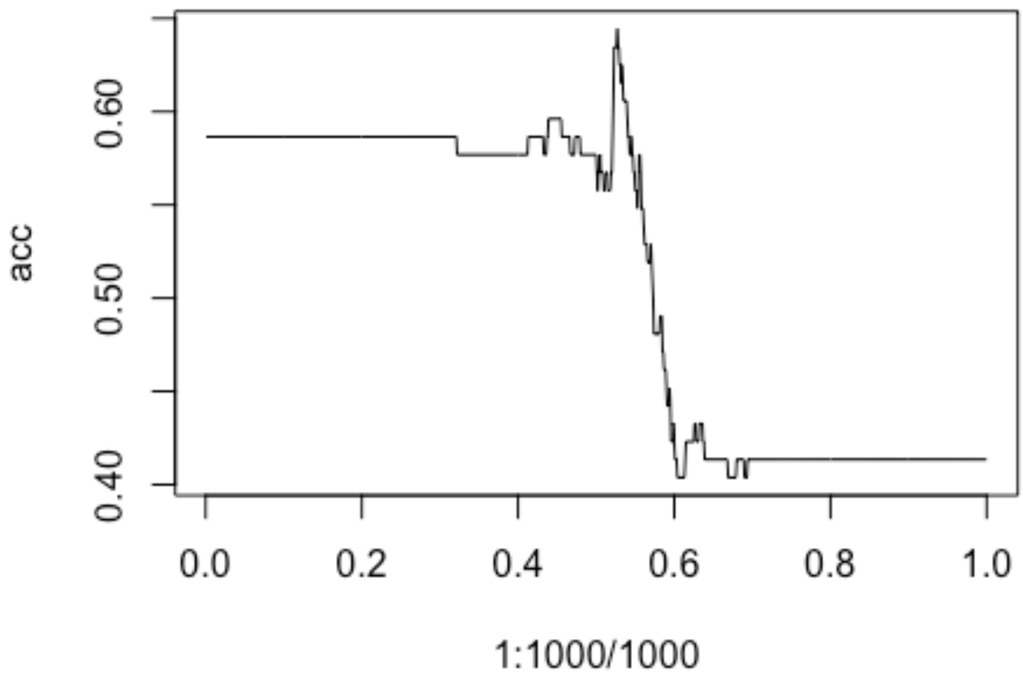
```r
print(paste('Optimal cutoff =',cutoff.opt, 'with accuracy',acc[cutoff.opt*100
0]))
```

```
## [1] "Optimal cutoff = 0.527 with accuracy 0.644230769230769"
```

```r
glm.fit$formula
```

```
## Direction ~ Lag1 + Lag2
```

```r
plot(1:1000/1000,acc,type='l')
```



LDA with Lag1, Lag2 and plot for finding cutoff point

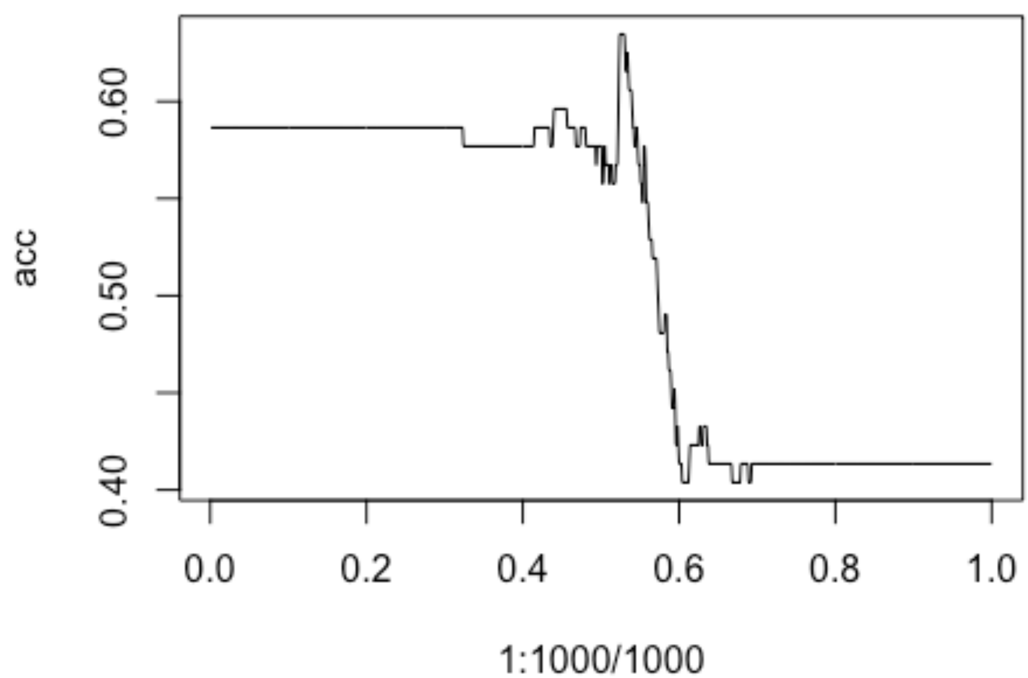
```r
library(MASS)
train <- which(df$Year<2009)
acc <- vector(length = 1000)
for (i in 1:1000) {
  lda.fit <- lda(Direction~Lag1+Lag2, data = df, subset = train)
  lda.pred <- predict(lda.fit, newdata = df[-train,])
  lda.class <- ifelse(lda.pred$posterior[,2]>i/1000,'Up','Down')
  acc[i] <- mean(lda.class==df$Direction[-train])
}
cutoff.opt <- which(acc==max(acc))/1000
print(paste('Optimal cutoff =',cutoff.opt, 'with accuracy',acc[cutoff.opt*100
0]))
```

```
## [1] "Optimal cutoff = 0.524 with accuracy 0.634615384615385"
## [2] "Optimal cutoff = 0.525 with accuracy 0.634615384615385"
## [3] "Optimal cutoff = 0.526 with accuracy 0.634615384615385"
## [4] "Optimal cutoff = 0.527 with accuracy 0.634615384615385"
## [5] "Optimal cutoff = 0.528 with accuracy 0.634615384615385"
## [6] "Optimal cutoff = 0.529 with accuracy 0.634615384615385"
## [7] "Optimal cutoff = 0.53 with accuracy 0.634615384615385"
```

```
lda.fit$formula
```

```
## NULL
```

```
plot(1:1000/1000,acc,type='l')
```



Subset selection on powers of Lag1 and Lag2 with Logistic regression

```
formula <- as.formula(Direction~Lag1+I(Lag1^2)+I(Lag1^3)+I(Lag1^4)+
                        Lag2+I(Lag2^2)+I(Lag2^3)+I(Lag2^4))
glm.fit <- glm(formula, data = df, subset = train, family='binomial')
glm.fit <- step(glm.fit,direction = 'both')
```

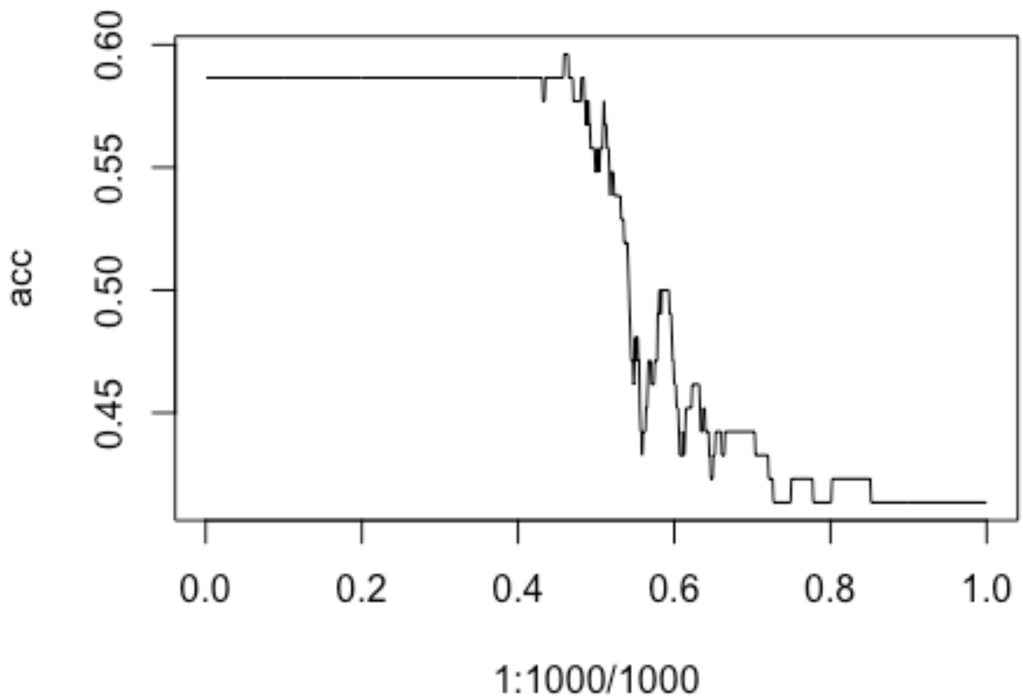
```
## Start:  AIC=1358.84
## Direction ~ Lag1 + I(Lag1^2) + I(Lag1^3) + I(Lag1^4) + Lag2 +
##     I(Lag2^2) + I(Lag2^3) + I(Lag2^4)
##
```

```
##              Df Deviance    AIC
## - I(Lag1^4)  1   1340.8 1356.8
## - I(Lag1^2)  1   1341.1 1357.1
## - I(Lag1^3)  1   1341.6 1357.6
## - Lag1       1   1341.6 1357.6
## - I(Lag2^3)  1   1341.7 1357.7
## <none>            1340.8 1358.8
## - I(Lag2^4)  1   1344.1 1360.1
## - Lag2       1   1344.3 1360.3
## - I(Lag2^2)  1   1346.0 1362.0
##
## Step:  AIC=1356.84
## Direction ~ Lag1 + I(Lag1^2) + I(Lag1^3) + Lag2 + I(Lag2^2) +
##     I(Lag2^3) + I(Lag2^4)
##
##              Df Deviance    AIC
## - I(Lag1^2)  1   1341.3 1355.3
## - Lag1       1   1341.6 1355.6
## - I(Lag2^3)  1   1341.7 1355.7
## - I(Lag1^3)  1   1342.0 1356.0
## <none>            1340.8 1356.8
## - I(Lag2^4)  1   1344.1 1358.1
## - Lag2       1   1344.3 1358.3
## + I(Lag1^4)  1   1340.8 1358.8
## - I(Lag2^2)  1   1346.0 1360.0
##
## Step:  AIC=1355.32
## Direction ~ Lag1 + I(Lag1^3) + Lag2 + I(Lag2^2) + I(Lag2^3) +
##     I(Lag2^4)
##
##              Df Deviance    AIC
## - I(Lag2^3)  1   1342.1 1354.1
## - I(Lag1^3)  1   1342.1 1354.1
## - Lag1       1   1342.1 1354.1
## <none>            1341.3 1355.3
## - I(Lag2^4)  1   1344.2 1356.2
## + I(Lag1^2)  1   1340.8 1356.8
## + I(Lag1^4)  1   1341.1 1357.1
## - Lag2       1   1345.1 1357.1
## - I(Lag2^2)  1   1346.0 1358.0
##
## Step:  AIC=1354.06
## Direction ~ Lag1 + I(Lag1^3) + Lag2 + I(Lag2^2) + I(Lag2^4)
##
##              Df Deviance    AIC
## - I(Lag1^3)  1   1342.8 1352.8
## - Lag1       1   1342.9 1352.9
## <none>            1342.1 1354.1
## - I(Lag2^4)  1   1345.0 1355.0
## - Lag2       1   1345.2 1355.2
```

```
## + I(Lag2^3)  1   1341.3 1355.3
## + I(Lag1^2)  1   1341.7 1355.7
## + I(Lag1^4)  1   1341.9 1355.9
## - I(Lag2^2)  1   1346.1 1356.1
##
## Step:  AIC=1352.82
## Direction ~ Lag1 + Lag2 + I(Lag2^2) + I(Lag2^4)
##
##               Df Deviance    AIC
## <none>            1342.8 1352.8
## - I(Lag2^4)  1   1345.8 1353.8
## - Lag2       1   1345.9 1353.9
## + I(Lag1^3)  1   1342.1 1354.1
## + I(Lag2^3)  1   1342.1 1354.1
## - Lag1       1   1346.2 1354.2
## + I(Lag1^2)  1   1342.8 1354.8
## + I(Lag1^4)  1   1342.8 1354.8
## - I(Lag2^2)  1   1347.0 1355.0

form.opt <- glm.fit$formula
```

Cutoff selection with Lag1 and Lag2 powers formula from above

```
train <- which(df$Year<2009)
acc <- vector(length=1000)
for (i in 1:1000) {
  glm.fit <- glm(form.opt, data = df, subset = train, family='binomial')
  glm.prob <- predict(glm.fit, newdata = df[-train,], type = 'response')
  glm.pred <- ifelse(glm.prob>i/1000, 'Up', 'Down')
  acc[i] <- mean(glm.pred==df$Direction[-train])
}

cutoff.opt <- which(acc==max(acc))/1000
print(paste('Optimal cutoff =',cutoff.opt, 'with accuracy',acc[cutoff.opt*100
0]))

## [1] "Optimal cutoff = 0.459 with accuracy 0.596153846153846"
## [2] "Optimal cutoff = 0.46 with accuracy 0.596153846153846"
## [3] "Optimal cutoff = 0.461 with accuracy 0.596153846153846"
## [4] "Optimal cutoff = 0.462 with accuracy 0.596153846153846"
## [5] "Optimal cutoff = 0.463 with accuracy 0.596153846153846"
## [6] "Optimal cutoff = 0.464 with accuracy 0.596153846153846"

glm.fit$formula

## Direction ~ Lag1 + Lag2 + I(Lag2^2) + I(Lag2^4)

plot(1:1000/1000,acc,type='l')
```



Overall, Logistic regression and LDA remained the top models. We were able to increase the accuracy slightly over the initial value of each (62.5%) up to 64.4% with Logistic Regression using Lag1 and Lag2 predictors and a cutoff on posterior probabilities of .527.

While some of the transformations had greater than 50% accuracies, none performed as well as just using Lag1 and Lag2 as predictors.