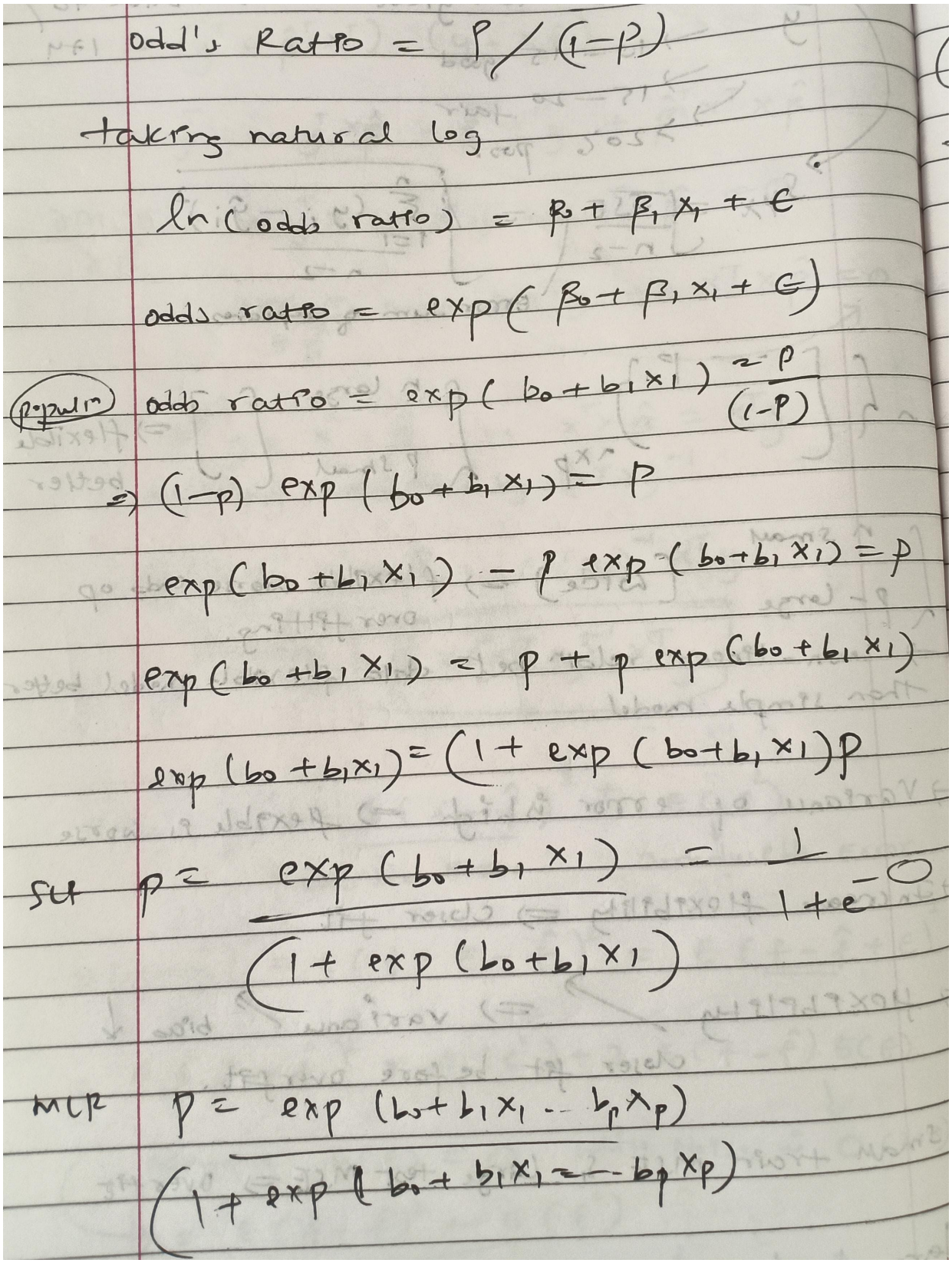
Homework Assignment 2

1. Using a little bit of algebra, prove that (4.2) is equivalent to (4.3). In other words, the logistic function representation and logit representation for the logistic regression model are equivalent.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}. \qquad (4.2)$$

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}. \qquad (4.3)$$



2. 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?

If the Bayes decision boundary is linear, LDA is expected to perform better on the training set and the test set due to its simplicity and lower risk of overfitting as it's less flexible compared to QDA and has fewer parameters due to the assumption of common covariance matrix in each class. With a linear decision boundary, LDA's assumption is more appropriate, and it is likely to generalize better to unseen data.

(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?

If the Bayes decision boundary is non-linear, QDA is expected to perform better on the training set and the test set due to its flexibility in modeling complex decision boundaries as well as the fact that it allows different covariance matrices for each class. With a non-linear decision boundary, QDA's assumption is more appropriate, and it is likely to generalize better to unseen data.

(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?

As the sample size "n" increases in the training set, the test prediction accuracy of QDA relative to LDA is expected to improve. This is because, with a larger sample size, QDA can better estimate the parameters of its more flexible model, leading to improved generalization performance under the assumption that the variance of the classifier is not a major concern, or if the assumption of a common covariance matrix for the K classes is untenable. If the sample size n increases but the data is linearly separated and assuming that the observations from each class are drawn from a Gaussian distribution then LDA might have an upper hand over QDA.

(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. Even if the Bayes decision boundary is linear, we would not necessarily achieve a superior test error rate using QDA rather than LDA. QDA is more flexible than LDA and can capture more complex decision boundaries, but this flexibility comes with the cost of increased model complexity. In cases where the true decision boundary is linear, the additional flexibility of QDA may suffer from higher variance and overfitting on the training set, resulting in a higher test error rate compared to LDA, which assumes a simpler linear boundary. LDA may achieve a superior test error rate compared to QDA, especially with a limited sample size.

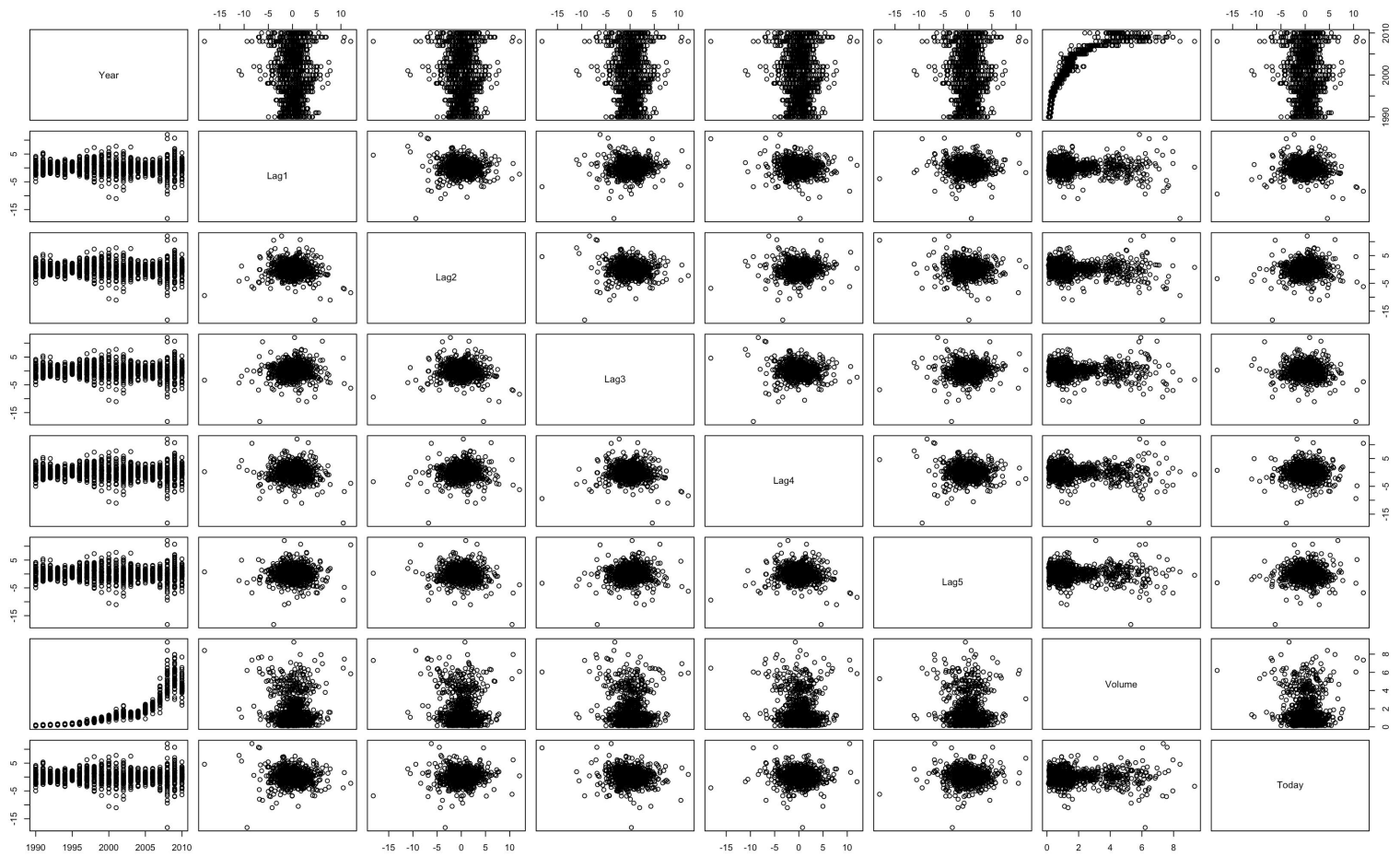
3. Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20 % on the training data and 30 % on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

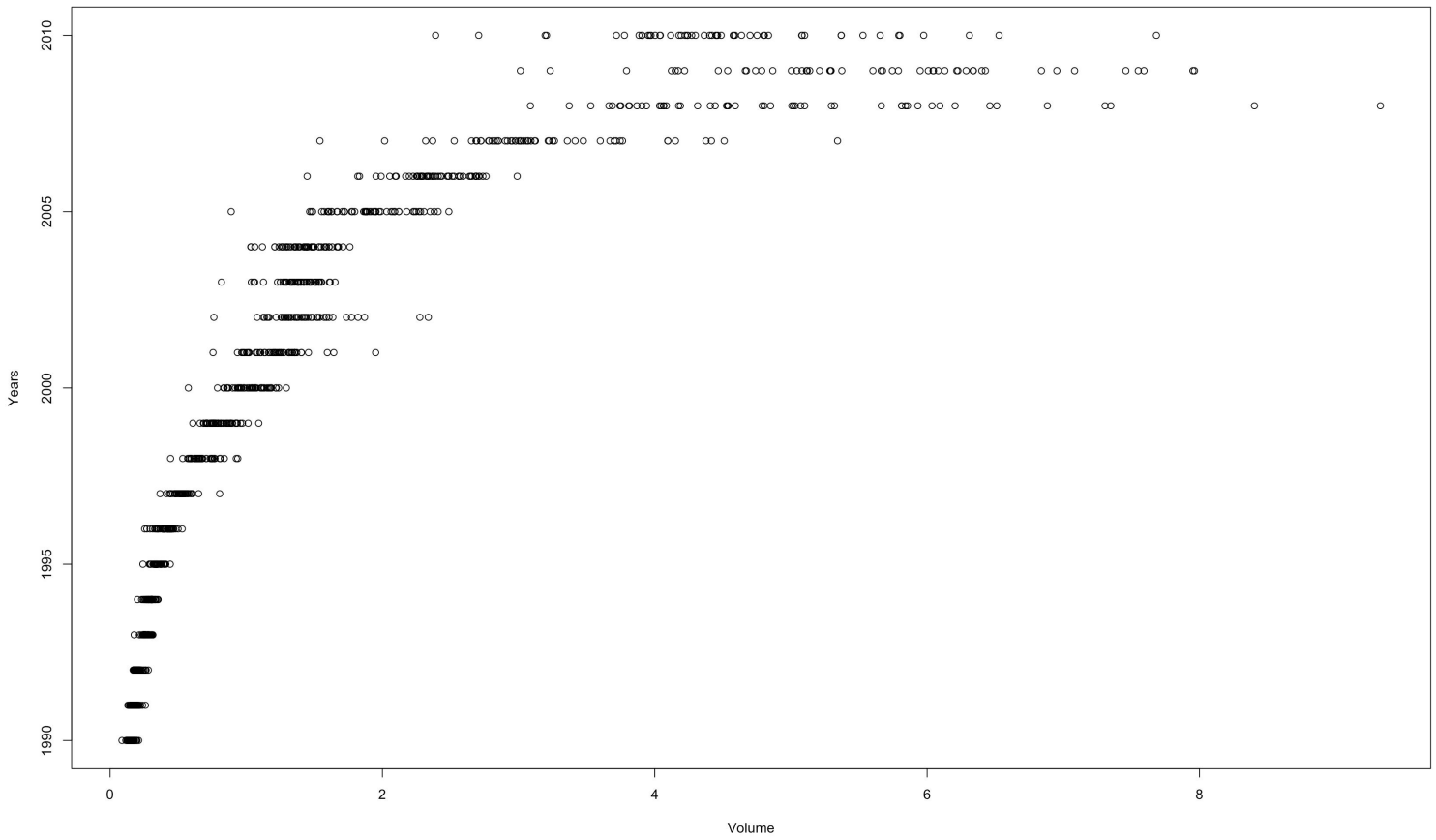
We are only provided with the average error rate for KNN (K=1). The average can be 18% given train error as 2% and test error as 34% in which case the KNN overfits the data and does not generalize well on the unseen dataset (test dataset) in this case since the logistic regression model gave 30% error on test data it seems to be better in comparison to KNN (K=1). The average can be 18% given train error as 18% and test 18% making KNN lead the scoreboard in comparison to the logistic regression model by 12% on the test dataset. Additionally, no information is given about the data distribution in train and test sets. Additionally logistic model is less flexible compared to KNN where KNN can model complex, non-linear decision boundaries, as it directly learns from the data without making strong assumptions about the underlying distribution. Based on the above-mentioned scenarios the preference for the classification model will change. Suppose there were 20 training observations and 20 testing observations in each of the two classes. The observations within each class were uncorrelated random normal variables with a different mean in each class. Since logistic regression assumes a linear decision boundary, its results will be superior to those of KNN. If we are going just based on the average error rate then the average error rate for the logistic regression model will be (20 + 30) / 2 = 25% which is more than that of KNN (18%) and thus KNN will be chosen to be used for classification of new observations.

4. 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.
(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
> names(weekly)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"      "Volume"    "Today"     "Direction"
> dim(weekly)
[1] 1089    9
> summary(weekly)
      Year           Lag1               Lag2               Lag3               Lag4               Lag5              Volume
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :  0.2380   Median :  0.2340   Median :1.00268
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
     Today           Direction
 Min.   :-18.1950   Length:1089
 1st Qu.: -1.1540   Class :character
 Median :  0.2410   Mode  :character
 Mean   :  0.1499
 3rd Qu.:  1.4050
 Max.   : 12.0260
> View(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
7	1990	-1.372	1.178	0.712	3.514	-2.576	0.1517220	0.807	Up
8	1990	0.807	-1.372	1.178	0.712	3.514	0.1323100	0.041	Up
9	1990	0.041	0.807	-1.372	1.178	0.712	0.1439720	1.253	Up
10	1990	1.253	0.041	0.807	-1.372	1.178	0.1336350	-2.678	Down
11	1990	-2.678	1.253	0.041	0.807	-1.372	0.1490240	-1.793	Down
12	1990	-1.793	-2.678	1.253	0.041	0.807	0.1357900	2.820	Up
13	1990	2.820	-1.793	-2.678	1.253	0.041	0.1398980	4.022	Up
14	1990	4.022	2.820	-1.793	-2.678	1.253	0.1643420	0.750	Up


Year
Lag1
Lag2
Lag3
Lag4
Lag5
Volume
Today


Years
Volume

```
> cor(weekly[,-9])
              Year         Lag1        Lag2        Lag3         Lag4         Lag5      Volume        Today
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923 -0.030519101  0.84194162 -0.032459894
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096 -0.06495131 -0.075031842
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482 -0.08551314  0.059166717
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175 -0.06928771 -0.071243639
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027 -0.06107462 -0.007825873
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000 -0.05851741  0.011012698
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414  1.00000000 -0.033077783
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698 -0.03307778  1.000000000
```

The correlations between the lag variables and today's returns are close to zero. In other words, there appears to be little correlation between today's returns and previous days' returns. The only substantial correlation is between Year and Volume. By plotting the data we see that Volume is increasing over time.

(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?

```
> log_reg_fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=weekly,family=binomial)
> summary(log_reg_fit)

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

> contrasts(weekly$Direction)
     Up
Down  0
Up    1
```

contrasts() function indicates that R has created a dummy variable with a 1 for Up.

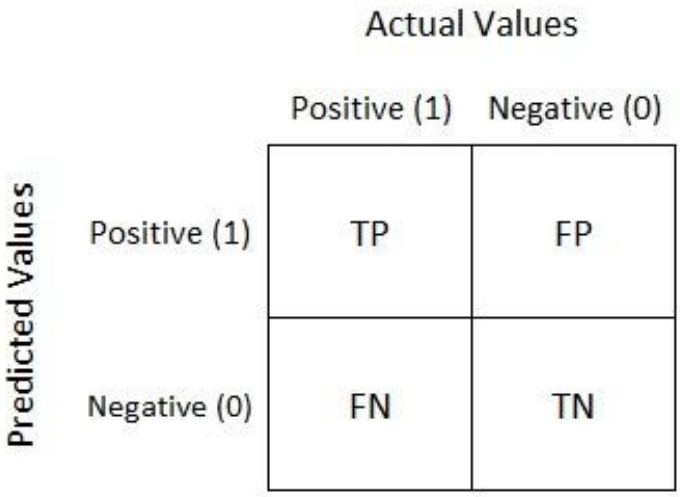
The smallest p-value here is associated with Lag2. The positive coefficient for this predictor suggests that if the market had a positive return day before yesterday, then it is more likely to go up today. However, at a value of 0.1181, the p-value is still relatively large, so there is no clear evidence of a real association between Lag2 and Direction.

(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.

```
> log_reg_fit_probs=predict(log_reg_fit,type="response")
> log_reg_fit_pred=rep("Down",dim(weekly)[1])
> log_reg_fit_pred[log_reg_fit_probs > 0.5]="Up"
> table(log_reg_fit_pred,weekly$Direction)

log_reg_fit_pred Down  Up
            Down   54  48
            Up    430 557

> mean(log_reg_fit_pred==weekly$Direction)
[1] 0.5610652
```

(54 + 557) / (54 + 557 + 430 + 48) = 0.5610652 = 56.11%

The diagonal elements of the confusion matrix indicate correct predictions, while the off-diagonals represent incorrect predictions. Hence our model correctly predicted that the market would go up on 557 days and that it would go down on 54 days, for a total of 557 + 54 = 611 correct predictions. The mean() function can be used to compute the fraction of days for which the prediction was correct. In this case, logistic regression correctly predicted the movement of the market 56.11 % of the time. At first glance, it appears that the logistic regression model is working a little better than random guessing. However, this result is misleading because we trained and tested the model on the same set of 1,089 observations. In other words, 100 - 56.11 = 43.89 % is the training error rate. As we have seen previously, the training error rate is often overly optimistic it tends to underestimate the test error rate. There are 48 incorrect occurrences when the model predicted the direction to be down but it was up and 430 occurrences when the model predicted it to be up but it was down.



Here
TP (true positives) = 557
FP (false positives) = 430
FN (false negatives) = 48
TN (true negatives) = 54

Precision measures the proportion of true positive predictions among all positive predictions made by the classifier.
Precision = TP / (TP + FP) = 557 / (557 + 430) = 0.5643 = 56.43%

Recall (True Positive Rate or Sensitivity) measures the proportion of true positive predictions among all actual positive instances.

Recall = TP / (TP + FN) = 557 / (557 + 48) = 0.9206 = 92.06%

False Positive Rate (FPR) measures the proportion of actual negative instances that are incorrectly classified as positive by the classifier.
FPR = FP / (FP + TN) = 430 / (430 + 54) = 0.8884 = 88.84%

(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_set<-weekly[(weekly$Year<2009),]
> dim(train_set)
[1] 985   9
> test_set<-weekly[!(weekly$Year<2009),]
> dim(test_set)
[1] 104   9
> log_reg_fit_till_2008=glm(Direction~Lag2,data=train_set,family=binomial)
> summary(log_reg_fit_till_2008)

Call:
glm(formula = Direction ~ Lag2, family = binomial, data = train_set)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
---
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: 1350.5  on 983  degrees of freedom
AIC: 1354.5

Number of Fisher Scoring iterations: 4
```

```
> log_reg_fit_till_2008_probs=predict(log_reg_fit_till_2008,data.frame(Lag2=test_set$Lag2),type="response")
> log_reg_fit_till_2008_pred=rep("Down",dim(test_set)[1])
> log_reg_fit_till_2008_pred[log_reg_fit_till_2008_probs > 0.5]="Up"
> table(log_reg_fit_till_2008_pred,test_set$Direction)

log_reg_fit_till_2008_pred Down Up
                      Down    9  5
                      Up     34 56
> mean(log_reg_fit_till_2008_pred==test_set$Direction)
[1] 0.625
```

Based on the test dataset, our model correctly predicted that the market would go up in 56 days and that it would go down in 9 days, for a total of 9 + 56 = 65 correct predictions. In this case, logistic regression correctly predicted the movement of the market 62.5 % of the time. There are 5 incorrect occurrences when the model predicted the direction to be down but it was up and 34 occurrences when the model predicted it to be up but it was down.

Here
TP (true positives) = 56

FP (false positives) = 34
FN (false negatives) = 5
TN (true negatives) = 9

Precision = TP / (TP + FP) = 56 / (56 + 34) = 0.6222 = 62.22%

Recall = TP / (TP + FN) = 56 / (56 + 5) = 0.9180328 = 91.80%

FPR = FP / (FP + TN) = 34 / (34 + 9) = 0.7907 = 79.07%

(e) Repeat (d) using LDA.

```
> lda_fit_till_2008=lda(Direction~Lag2,data=train_set,family=binomial)
> summary(lda_fit_till_2008)
        Length Class  Mode
prior   2      -none- numeric
counts  2      -none- numeric
means   2      -none- numeric
scaling 1      -none- numeric
lev     2      -none- character
svd     1      -none- numeric
N       1      -none- numeric
call    4      -none- call
terms   3      terms  call
xlevels 0      -none- list
> lda_fit_till_2008_pred=predict(lda_fit_till_2008,data.frame(Lag2=test_set$Lag2),type="response")
> table(lda_fit_till_2008_pred$class,test_set$Direction)

      Down Up
  Down    9  5
  Up     34 56
> mean(lda_fit_till_2008_pred$class==test_set$Direction)
[1] 0.625
```

Based on the test dataset, the predictions are the same as we got for the logistic regression model.

(f) Repeat (d) using QDA.

```
> qda_fit_till_2008=qda(Direction~Lag2,data=train_set,family=binomial)
> summary(qda_fit_till_2008)
        Length Class  Mode
prior   2      -none- numeric
counts  2      -none- numeric
means   2      -none- numeric
scaling 2      -none- numeric
ldet    2      -none- numeric
lev     2      -none- character
N       1      -none- numeric
call    4      -none- call
terms   3      terms  call
xlevels 0      -none- list
> qda_fit_till_2008_pred=predict(qda_fit_till_2008,data.frame(Lag2=test_set$Lag2),type="response")
> table(qda_fit_till_2008_pred$class,test_set$Direction)

      Down Up
  Down    0  0
  Up     43 61
> mean(qda_fit_till_2008_pred$class==test_set$Direction)
[1] 0.5865385
```

Based on the test dataset, our model correctly predicted that the market would go up on 61 days and that it would go down on 0 days, for a total of 61 + 0 = 61 correct predictions. In this case, Quadratic Discriminant Analysis (QDA) correctly predicted the movement of the market 58.65 % of the time. There are 0 incorrect occurrences when the model predicted the direction to be down but it was up and 43 occurrences when the model predicted it to be up but it was down.

Here
TP (true positives) = 61
FP (false positives) = 43
FN (false negatives) = 0
TN (true negatives) = 0

Precision = TP / (TP + FP) = 61 / (61 + 43) = 0.5865 = 58.65%

Recall = TP / (TP + FN) = 61 / (61 + 0) = 1.00 = 100%

FPR = FP / (FP + TN) = 43 / (43 + 0) = 1.00 = 100%

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

```
> knn_fit_till_2008_pred=knn((data.frame(Lag2=train_set$Lag2)),(data.frame(Lag2=test_set$Lag2)),train_set$Direction,k=1)
> table(knn_fit_till_2008_pred,test_set$Direction)

knn_fit_till_2008_pred Down Up
                  Down   21 30
                  Up     22 31
> mean(knn_fit_till_2008_pred==test_set$Direction)
[1] 0.5
```

Based on the test dataset, our model correctly predicted that the market would go up on 61 days and that it would go down on 0 days, for a total of 61 + 0 = 61 correct predictions. In this case, K-Nearest Neighbors (KNN) correctly predicted the movement of the market 50% of the time. There are 30 incorrect occurrences when the model predicted the direction to be down but it was up and 22 occurrences when the model predicted it to be up but it was down.

Here
TP (true positives) = 31
FP (false positives) = 22
FN (false negatives) = 30
TN (true negatives) = 21

Precision = TP / (TP + FP) = 31 / (31 + 22) = 0.5849 = 58.49%

Recall = TP / (TP + FN) = 31 / (31 + 30) = 0.5082 = 50.82%

FPR = FP / (FP + TN) = 22 / (22 + 21) = 0.5116 = 51.16%

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

Based on the test results given by logistic regression, LDA, QDA and KNN (K=1), we can see that both logistic regression and LDA share the same results for all predictions. The precision given by logistic / LDA was 62.22% which was the highest among all the other competitors. This suggests that LR and QDA have better precision in correctly identifying positive instances compared to KNN. The recall value and false positive rate were highest for QDA (100%), indicating that it identifies all positive instances correctly and logistic / LDA was second in the position with 91.80% recall and 79.07% FPT whereas KNN stood in the last with close to random guess predictions with only 50.82% recall and 51.16% FPR. Based on the provided results and without considering other factors such as computational complexity or model assumptions, it appears that logistic regression and LDA perform the best among the methods considered as they achieve a good balance between precision, recall, and false positive rate, making it a favorable choice for this dataset.

(h) 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.

The data was scaled and a few interactions were added before performing the train test split.

```
> weekly<-read.csv("./Weekly.csv")
> weekly$Direction<-as.factor(weekly$Direction)
> names(weekly)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"      "Volume"    "Today"     "Direction"
> weekly$Year_scaled<-scale(weekly$Year)
> weekly$Lag1_scaled<-scale(weekly$Lag1)
> weekly$Lag2_scaled<-scale(weekly$Lag2)
> weekly$Lag3_scaled<-scale(weekly$Lag3)
> weekly$Lag4_scaled<-scale(weekly$Lag4)
> weekly$Lag5_scaled<-scale(weekly$Lag5)
> weekly$Volume_scaled<-scale(weekly$Volume)
> weekly$Today_scaled<-scale(weekly$Today)
> weekly$Lag2_Lag1<-weekly$Lag2*weekly$Lag1
> weekly$Lag3_Lag1<-weekly$Lag3*weekly$Lag1
> weekly$Lag4_Lag1<-weekly$Lag4*weekly$Lag1
> weekly$Lag5_Lag1<-weekly$Lag5*weekly$Lag1
> weekly$Volume_Lag1<-weekly$Volume*weekly$Lag1
> names(weekly)
 [1] "Year"          "Lag1"          "Lag2"          "Lag3"          "Lag4"          "Lag5"          "Volume"        "Today"
 [9] "Direction"     "Year_scaled"   "Lag1_scaled"   "Lag2_scaled"   "Lag3_scaled"   "Lag4_scaled"   "Lag5_scaled"   "Volume_scaled"
[17] "Today_scaled"  "Lag2_Lag1"     "Lag3_Lag1"     "Lag4_Lag1"     "Lag5_Lag1"     "Volume_Lag1"
> train_set<-weekly[(weekly$Year<2009),]
> dim(train_set)
[1] 985  22
> test_set<-weekly[!(weekly$Year<2009),]
> dim(test_set)
[1] 104  22
> log_reg_fit_till_2008=glm(Direction~Lag2_scaled+Lag3_scaled+Volume_scaled,data=train_set,family=binomial)
> summary(log_reg_fit_till_2008)

Call:
glm(formula = Direction ~ Lag2_scaled + Lag3_scaled + Volume_scaled,
    family = binomial, data = train_set)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.18953    0.06715   2.822  0.00477 **
Lag2_scaled   0.12165    0.06884   1.767  0.07720 .
Lag3_scaled  -0.03962    0.06823  -0.581  0.56145
Volume_scaled -0.10084    0.08832  -1.142  0.25357
---
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: 1349.1  on 981  degrees of freedom
AIC: 1357.1

Number of Fisher Scoring iterations: 4


> log_reg_fit_till_2008_probs=predict(log_reg_fit_till_2008,test_set[,c("Lag1_scaled","Lag2_scaled","Lag3_scaled","Lag4_scaled","Lag5_scaled","Volume_scal
ed","Today_scaled")],type="response")
> log_reg_fit_till_2008_pred=rep("Down",nrow(test_set))
> log_reg_fit_till_2008_pred[log_reg_fit_till_2008_probs>0.5]="Up"
> table(log_reg_fit_till_2008_pred,test_set$Direction)

log_reg_fit_till_2008_pred Down Up
                      Down   24 26
                      Up     19 35
> mean(log_reg_fit_till_2008_pred==test_set$Direction)
[1] 0.5673077
```

```
> lda_fit_till_2008=lda(Direction~Lag2_scaled+Lag3_scaled+Volume_scaled,data=train_set)
> summary(lda_fit_till_2008)
        Length Class  Mode
prior   2      -none- numeric
counts  2      -none- numeric
means   6      -none- numeric
scaling 3      -none- numeric
lev     2      -none- character
svd     1      -none- numeric
N       1      -none- numeric
call    3      -none- call
terms   3      terms  call
xlevels 0      -none- list
> lda_fit_till_2008_pred=predict(lda_fit_till_2008,test_set[,c("Lag1_scaled","Lag2_scaled","Lag3_scaled","Lag4_scaled","Lag5_scaled","Volume_scaled","Toda
y_scaled")],type="response")
> table(lda_fit_till_2008_pred$class,test_set$Direction)


      Down Up
  Down   24 26
  Up     19 35
> mean(lda_fit_till_2008_pred$class==test_set$Direction)
[1] 0.5673077
> qda_fit_till_2008=qda(Direction~Lag2_scaled+Lag3_scaled+Volume_scaled,data=train_set)
> summary(qda_fit_till_2008)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means    6     -none- numeric
scaling 18     -none- numeric
ldet     2     -none- numeric
lev      2     -none- character
N        1     -none- numeric
call     3     -none- call
terms    3     terms  call
xlevels  0     -none- list
> qda_fit_till_2008_pred=predict(qda_fit_till_2008,test_set[,c("Lag1_scaled","Lag2_scaled","Lag3_scaled","Lag4_scaled","Lag5_scaled","Volume_scaled","Toda
y_scaled")],type="response")
> table(qda_fit_till_2008_pred$class,test_set$Direction)

      Down Up
  Down   36 47
  Up      7 14
> mean(qda_fit_till_2008_pred$class==test_set$Direction)
[1] 0.4807692
> k_values<-c(1,3,5,7,9,10)
> for (k_clust in k_values) {cat("K =",k_clust,",Training Error =",mean(knn(train_set[,c("Lag1_scaled","Lag2_scaled","Lag3_scaled","Lag4_scaled","Lag5_sca
led","Volume_scaled","Today_scaled")],test_set[,c("Lag1_scaled","Lag2_scaled","Lag3_scaled","Lag4_scaled","Lag5_scaled","Volume_scaled","Today_scaled")],t
rain_set$Direction,k=k_clust)!=test_set$Direction),"\n")}
K = 1 ,Training Error = 0.2019231
K = 3 ,Training Error = 0.1057692
K = 5 ,Training Error = 0.08653846
K = 7 ,Training Error = 0.1057692
K = 9 ,Training Error = 0.125
K = 10 ,Training Error = 0.1153846
```

Here we can see that Logistic and LDA performed the same and had the same validation error (43.27%), QDA's validation error was (51.93%) and for KNN (1 to 10) the error ranges between 8.6% to 20.19%. The best model can be considered to be KNN with K =5 where the validation error was 8.6%.

```
> log_reg_fit_till_2008=glm(Direction~Lag2_scaled+Lag2_Lag1+Lag3_Lag1+Volume_scaled,data=train_set,family=binomial)
> summary(log_reg_fit_till_2008)

Call:
glm(formula = Direction ~ Lag2_scaled + Lag2_Lag1 + Lag3_Lag1 +
    Volume_scaled, family = binomial, data = train_set)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.185317   0.067540   2.744  0.00607 **
Lag2_scaled    0.119463   0.069806   1.711  0.08702 .
Lag2_Lag1      0.005287   0.007241   0.730  0.46532
Lag3_Lag1      0.011627   0.009309   1.249  0.21169
Volume_scaled -0.108674   0.088857  -1.223  0.22132
---
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: 1346.8  on 980  degrees of freedom
AIC: 1356.8

Number of Fisher Scoring iterations: 4
```

```
> log_reg_fit_till_2008_probs=predict(log_reg_fit_till_2008,test_set[,c("Lag2_scaled","Lag2_Lag1","Lag3_Lag1","Volume_scaled")],type="response")
> log_reg_fit_till_2008_pred=rep("Down",nrow(test_set))
> log_reg_fit_till_2008_pred[log_reg_fit_till_2008_probs>0.5]="Up"
> table(log_reg_fit_till_2008_pred,test_set$Direction)

log_reg_fit_till_2008_pred Down Up
                      Down   23 26
                      Up     20 35
> mean(log_reg_fit_till_2008_pred==test_set$Direction)
[1] 0.5576923
> lda_fit_till_2008=lda(Direction~Lag2_scaled+Lag2_Lag1+Lag3_Lag1+Volume_scaled,data=train_set)
> summary(lda_fit_till_2008)
        Length Class  Mode
prior   2      -none- numeric
counts  2      -none- numeric
means   8      -none- numeric
scaling 4      -none- numeric
lev     2      -none- character
svd     1      -none- numeric
N       1      -none- numeric
call    3      -none- call
terms   3      terms  call
xlevels 0      -none- list
> lda_fit_till_2008_pred=predict(lda_fit_till_2008,test_set[,c("Lag2_scaled","Lag2_Lag1","Lag3_Lag1","Volume_scaled")],type="response")
> table(lda_fit_till_2008_pred$class,test_set$Direction)

       Down Up
  Down   22 25
  Up     21 36
> mean(lda_fit_till_2008_pred$class==test_set$Direction)
[1] 0.5576923
> qda_fit_till_2008=qda(Direction~Lag2_scaled+Lag2_Lag1+Lag3_Lag1+Volume_scaled,data=train_set)
> summary(qda_fit_till_2008)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means    8     -none- numeric
scaling 32     -none- numeric
ldet     2     -none- numeric
lev      2     -none- character
N        1     -none- numeric
call     3     -none- call
terms    3     terms  call
xlevels  0     -none- list
> qda_fit_till_2008_pred=predict(qda_fit_till_2008,test_set[,c("Lag2_scaled","Lag2_Lag1","Lag3_Lag1","Volume_scaled")],type="response")
> table(qda_fit_till_2008_pred$class,test_set$Direction)

       Down Up
  Down   33 46
  Up     10 15
> mean(qda_fit_till_2008_pred$class==test_set$Direction)
[1] 0.4615385
> k_values<-c(1,3,5,7,9,10)
> for (k_clust in k_values) {cat("K =",k_clust,",Training Error =",mean(knn(train_set[,c("Lag2_scaled","Lag2_Lag1","Lag3_Lag1","Volume_scaled")],test_set
[,c("Lag2_scaled","Lag2_Lag1","Lag3_Lag1","Volume_scaled")],train_set$Direction,k=k_clust)!=test_set$Direction),"\n")}
K = 1 ,Training Error = 0.5384615
K = 3 ,Training Error = 0.4519231
K = 5 ,Training Error = 0.5576923
K = 7 ,Training Error = 0.5
K = 9 ,Training Error = 0.5576923
K = 10 ,Training Error = 0.4903846
```

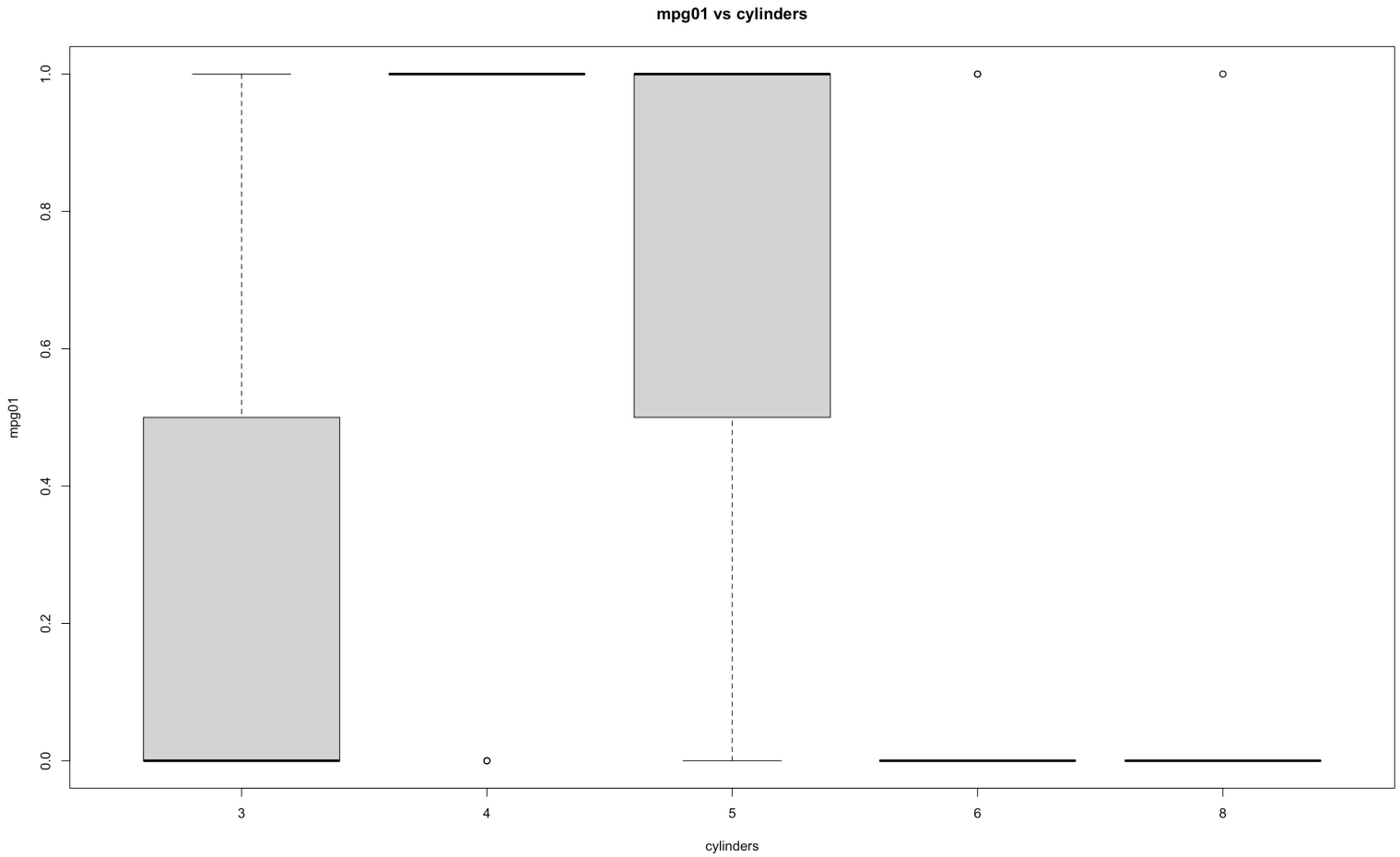
Here we can see that Logistic and LDA performed the same and had the same validation error (44.23%), QDA's validation error was (53.85%) and for KNN (1 to 10) the error ranges between 49% to 55.77%. It can be said that the interactions are not of much help to the model and none of the models performed better than the model fitted on scaled features.

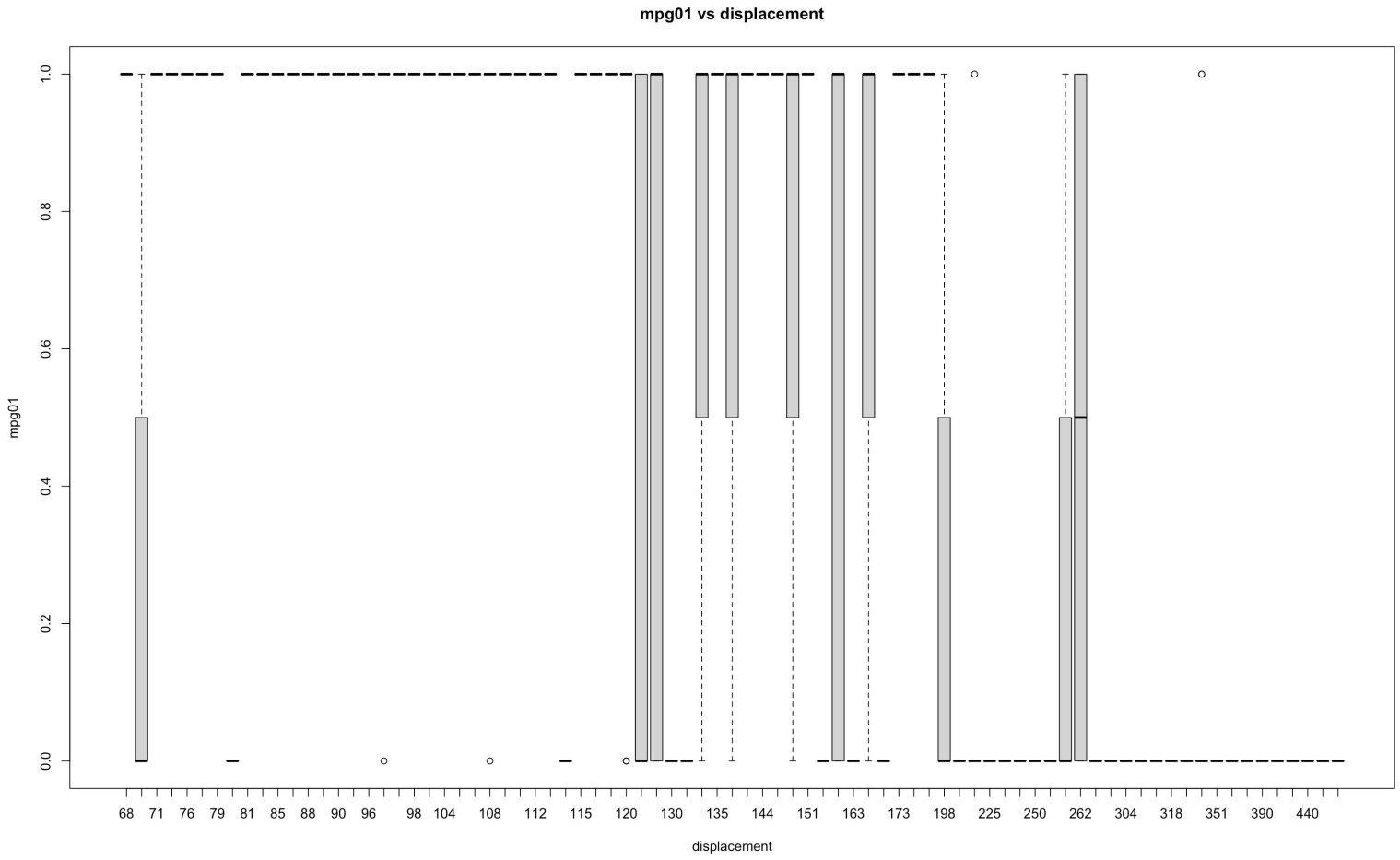
5. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

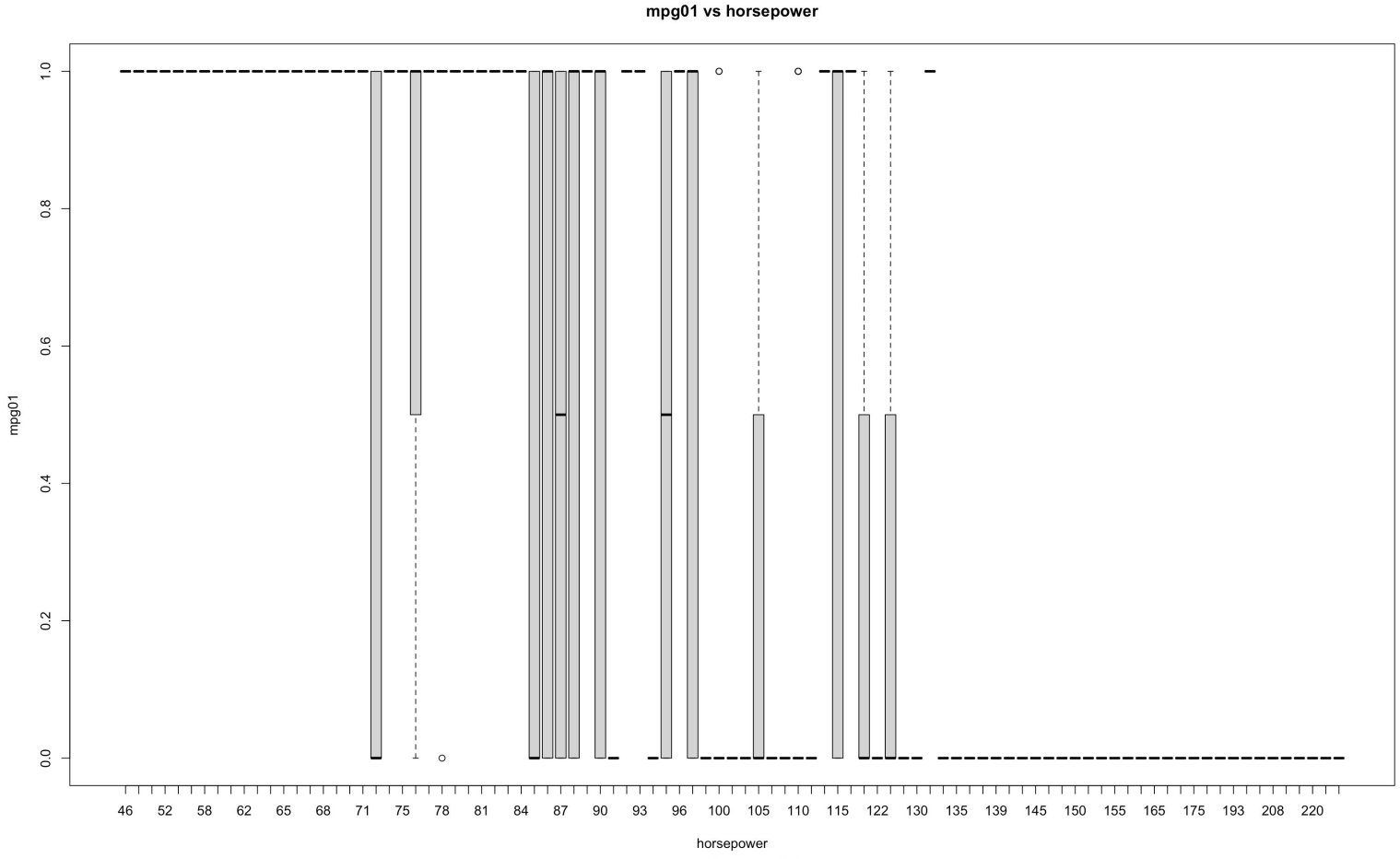
(a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

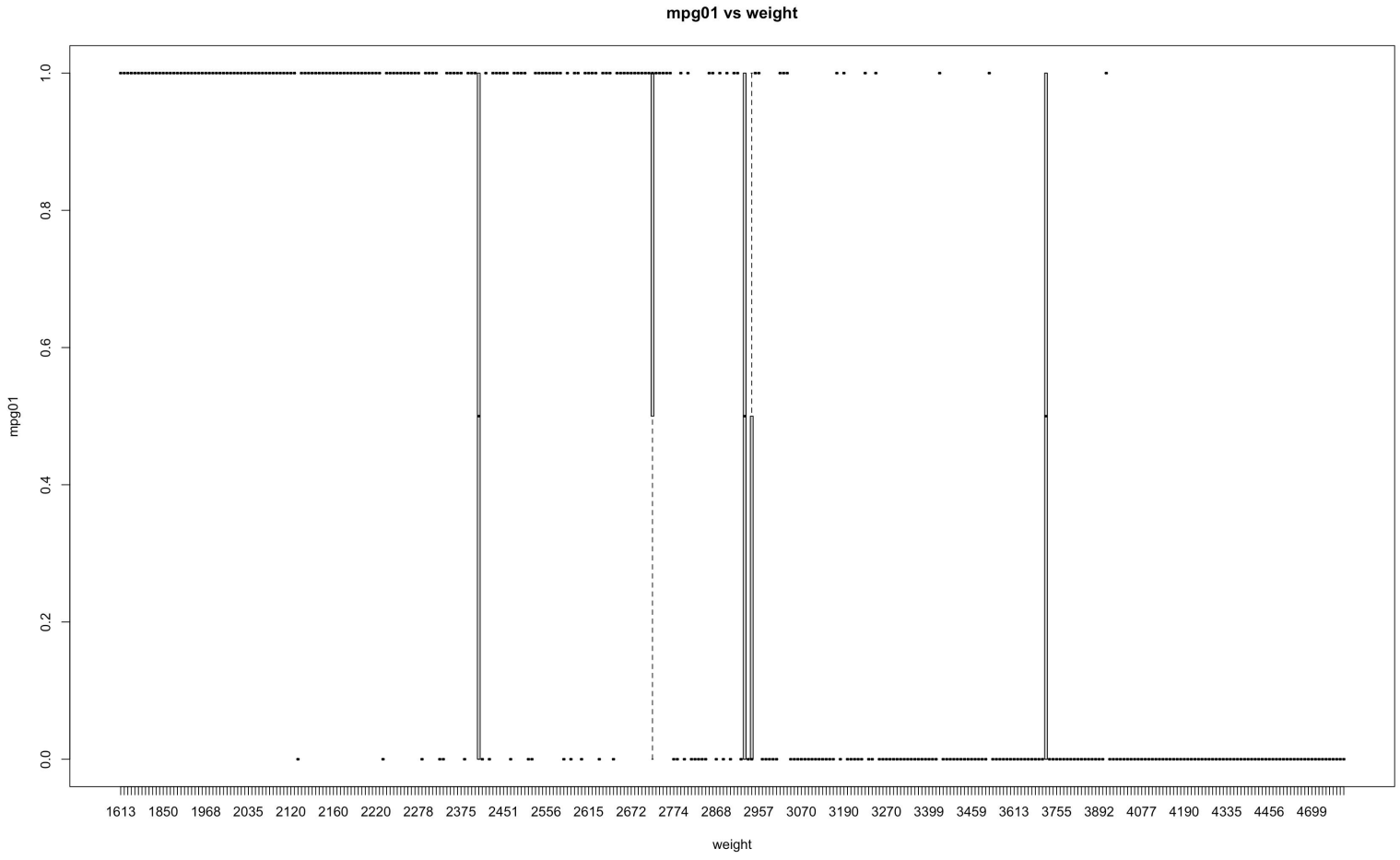
```
> auto_df<-read.csv("./Auto.csv")
> names(auto_df)
[1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"       "acceleration" "year"         "origin"       "name"
> dim(auto_df)
[1] 397   9
> summary(auto_df)
      mpg          cylinders      displacement    horsepower           weight      acceleration        year           origin
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Length:397         Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
 1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.0   Class :character   1st Qu.:2223   1st Qu.:13.80   1st Qu.:73.00   1st Qu.:1.000
 Median :23.00   Median :4.000   Median :146.0   Mode  :character   Median :2800   Median :15.50   Median :76.00   Median :1.000
 Mean   :23.52   Mean   :5.458   Mean   :193.5                      Mean   :2970   Mean   :15.56   Mean   :75.99   Mean   :1.574
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0                      3rd Qu.:3609   3rd Qu.:17.10   3rd Qu.:79.00   3rd Qu.:2.000
 Max.   :46.60   Max.   :8.000   Max.   :455.0                      Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
     name
 Length:397
 Class :character
 Mode  :character



> str(auto_df)
'data.frame':   397 obs. of  9 variables:
 $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
 $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
 $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
 $ horsepower  : chr  "130" "165" "150" "150" ...
 $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
 $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
 $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
 $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel sst" ...
> unique(auto_df$horsepower)
 [1] "130" "165" "150" "140" "198" "220" "215" "225" "190" "170" "160" "95"  "97"  "85"  "88"  "46"  "87"  "90"  "113" "200" "210" "193" "?"
[24] "100" "105" "175" "153" "180" "110" "72"  "86"  "70"  "76"  "65"  "69"  "60"  "80"  "54"  "208" "155" "112" "92"  "145" "137" "158" "167"
[47] "94"  "107" "230" "49"  "75"  "91"  "122" "67"  "83"  "78"  "52"  "61"  "93"  "148" "129" "96"  "71"  "98"  "115" "53"  "81"  "79"  "120"
[70] "152" "102" "108" "68"  "58"  "149" "89"  "63"  "48"  "66"  "139" "103" "125" "133" "138" "135" "142" "77"  "62"  "132" "84"  "64"  "74"
[93] "116" "82"
> auto_df<-auto_df[auto_df$horsepower!="?",]
> auto_df$horsepower<-as.integer(auto_df$horsepower)
> str(auto_df)
'data.frame':   392 obs. of  9 variables:
 $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
 $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
 $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
 $ horsepower  : int  130 165 150 150 140 198 220 215 225 190 ...
 $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
 $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
 $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
 $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel sst" ...


> summary(auto_df)
      mpg          cylinders      displacement     horsepower        weight      acceleration        year           origin
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
 1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
 Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804   Median :15.50   Median :76.00   Median :1.000
 Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
 Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
     name
 Length:392
 Class :character
 Mode  :character



> mpg_median <- median(auto_df$mpg)
> auto_df$mpg01 <- ifelse(auto_df$mpg > mpg_median, 1, 0)
```

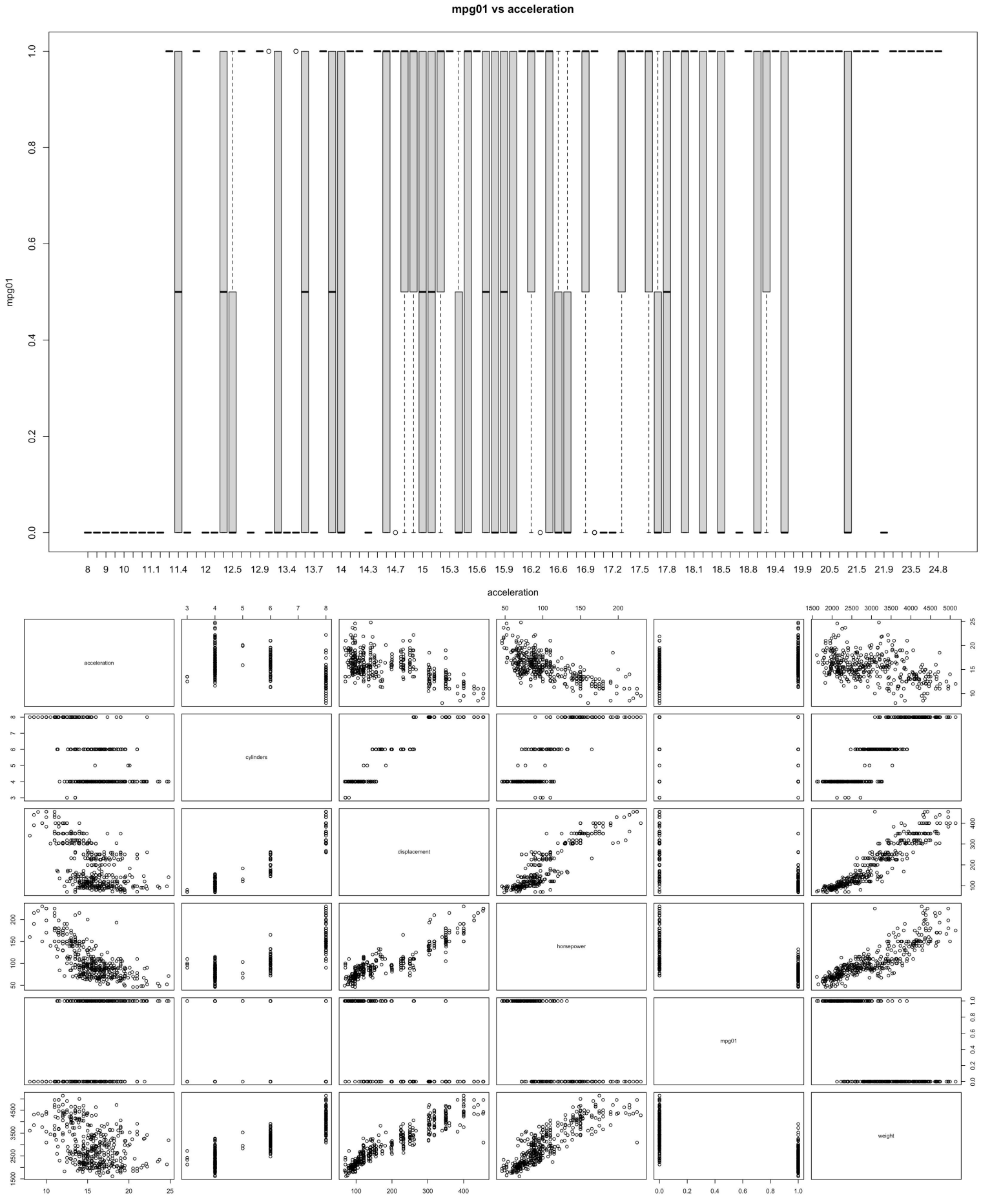
(b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.






mpg01 vs horsepower
mpg01
0.0
0.2
0.4
0.6
0.8
1.0
46 52 58 62 65 68 71 75 78 81 84 87 90 93 96 100 105 110 115 122 130 135 139 145 150 155 165 175 193 208 220
horsepower


mpg01 vs weight
mpg01
0.0
0.2
0.4
0.6
0.8
1.0
1613 1850 1968 2035 2120 2160 2220 2278 2375 2451 2556 2615 2672 2774 2868 2957 3070 3190 3270 3399 3459 3613 3755 3892 4077 4190 4335 4456 4699
weight



There is no strong correlation between mpg01 and other variables that strongly suggest that they affect mpg01. We have used a box plot and scatter plot to graphically view the interaction between mpg01 and other factors in the dataset.

(c) Split the data into a training set and a test set.

```
> mpg_median <- median(auto_df$mpg)
> auto_df$mpg01 <- ifelse(auto_df$mpg > mpg_median, 1, 0)
> boxplot(mpg01~cylinders,data=auto_df,main="mpg01 vs cylinders")
> boxplot(mpg01~displacement,data=auto_df,main="mpg01 vs displacement")
> boxplot(mpg01~horsepower,data=auto_df,main="mpg01 vs horsepower")
> boxplot(mpg01~weight,data=auto_df,main="mpg01 vs weight")
> boxplot(mpg01~acceleration,data=auto_df,main="mpg01 vs acceleration")
> pairs(auto_df[,c("acceleration","cylinders","displacement","horsepower","mpg01","weight")])
> set.seed(1)
> auto_df_train_indices<-sample(1:nrow(auto_df),0.8*nrow(auto_df))
> train_set<-auto_df[auto_df_train_indices,]
> test_set<-auto_df[-auto_df_train_indices,]
> dim(train_set)
[1] 313  10
> dim(test_set)
[1] 79 10
```

Here the data is split into 80% for train and 20% for test. We have used a random sample from the dataset and the train and test dataset observations are 313 and 79 respectively.

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
> lda_fit=lda(mpg01~cylinders+displacement+horsepower+weight+acceleration+year+origin,data=train_set)
> summary(lda_fit)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means   14     -none- numeric
scaling  7     -none- numeric
lev      2     -none- character
svd      1     -none- numeric
N        1     -none- numeric
call     3     -none- call
terms    3     terms  call
xlevels  0     -none- list
> lda_fit_pred=predict(lda_fit,test_set[,c("cylinders","displacement","horsepower","weight","acceleration","year","origin")],type="response")
> table(lda_fit_pred$class,test_set$mpg01)

    0  1
  0 35  0
  1  7 37
> mean(lda_fit_pred$class==test_set$mpg01)
[1] 0.9113924
> mean(lda_fit_pred$class!=test_set$mpg01)
[1] 0.08860759
```

The test error for logistic regression seems to be 8.9%

(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
> qda_fit=qda(mpg01~cylinders+displacement+horsepower+weight+acceleration+year+origin,data=train_set)
> summary(qda_fit)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means   14     -none- numeric
scaling 98     -none- numeric
ldet     2     -none- numeric
lev      2     -none- character
N        1     -none- numeric
call     3     -none- call
terms    3     terms  call
xlevels  0     -none- list
> qda_fit_pred=predict(qda_fit,test_set[,c("cylinders","displacement","horsepower","weight","acceleration","year","origin")],type="response")
> table(qda_fit_pred$class,test_set$mpg01)

     0  1
  0 36  2
  1  6 35
> mean(qda_fit_pred$class==test_set$mpg01)
[1] 0.8987342
> mean(qda_fit_pred$class!=test_set$mpg01)
[1] 0.1012658
```

The test error for logistic regression seems to be 10.13%

(f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
> log_reg_fit=glm(mpg01~cylinders+displacement+horsepower+weight+acceleration+year+origin,data=train_set,family=binomial)
> summary(log_reg_fit)

Call:
glm(formula = mpg01 ~ cylinders + displacement + horsepower +
    weight + acceleration + year + origin, family = binomial,
    data = train_set)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -16.813754   6.108476  -2.753 0.005914 **
cylinders     -0.141080   0.440246  -0.320 0.748621
displacement   0.006259   0.012693   0.493 0.621932
horsepower    -0.035162   0.026005  -1.352 0.176324
weight        -0.004820   0.001266  -3.806 0.000141 ***
acceleration   0.030651   0.151320   0.203 0.839483
year           0.424998   0.083323   5.101 3.39e-07 ***
origin         0.411418   0.385541   1.067 0.285919
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 433.83  on 312  degrees of freedom
Residual deviance: 135.29  on 305  degrees of freedom
AIC: 151.29

Number of Fisher Scoring iterations: 7

> log_reg_fit_probs=predict(log_reg_fit,test_set[,c("cylinders","displacement","horsepower","weight","acceleration","year","origin")],type="response")
> log_reg_fit_pred=rep(0,nrow(test_set))
> log_reg_fit_pred[log_reg_fit_probs>0.5]=1
> table(log_reg_fit_pred,test_set$mpg01)

log_reg_fit_pred  0  1
               0 38  2
               1  4 35
> mean(log_reg_fit_pred==test_set$mpg01)
[1] 0.9240506
> mean(log_reg_fit_pred!=test_set$mpg01)
[1] 0.07594937
```

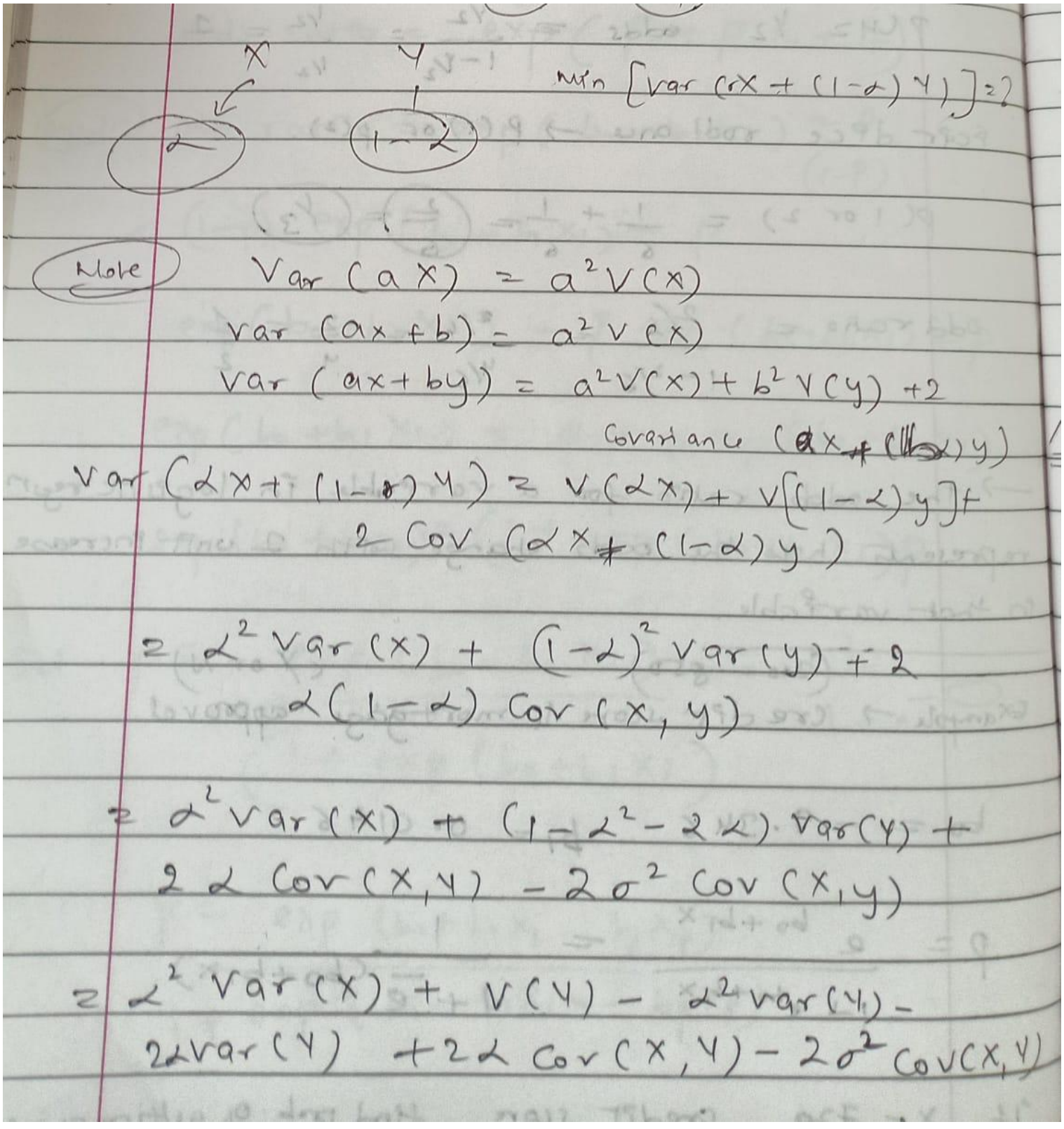
The test error for logistic regression seems to be 7.6%

(g) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```
> set.seed(1)
> k_values<-c(1,3,5,7,9,10)
> for (k_clust in k_values) {cat("K =",k_clust,",Training Error =",mean(knn(train_set[,c("cylinders","displacement","horsepower","weight","acceleratio
n","year","origin")],test_set[,c("cylinders","displacement","horsepower","weight","acceleration","year","origin")],train_set$mpg01,k=k_clust)!=test_set$mp
g01),"\n")}
K = 1 ,Training Error = 0.1265823
K = 3 ,Training Error = 0.1012658
K = 5 ,Training Error = 0.1012658
K = 7 ,Training Error = 0.1139241
K = 9 ,Training Error = 0.1139241
K = 10 ,Training Error = 0.1265823
```

The best value based on the test error rate seems to be 3 where the error is 10.12%.

6. Using basic statistical properties of the variance, as well as single variable calculus, derive (5.6). In other words, prove that α given by (5.6) does indeed minimize Var(αX + (1 - α)Y).



Page No
Date

(A) $= \alpha^2 V(X) + \alpha^2 V(Y) - 2\alpha^2 Cov(X,Y) - 2\alpha V(Y) + 2\alpha Cov(X,Y) + V(Y)$

Need to minimize — We have to take derivative & set it to zero. Use α as its unknown.

$$\frac{\partial A}{\partial \alpha} = 2\alpha V(X) + 2\alpha V(Y) - 4\alpha Cov(X,Y) - 2V(Y) + 2Cov(X,Y) + 0 = 0$$

÷ by 2 on both sides.

$$\alpha V(X) + \alpha V(Y) - 2\alpha Cov(X,Y) - V(Y) + Cov(X,Y) = 0$$

$$\alpha [V(X) + V(Y) - 2Cov(X,Y)] = V(Y) - Cov(X,Y)$$

Page 29 Net 5

$$\alpha = \frac{V(Y) - Cov(X,Y)}{V(X) + V(Y) - 2Cov(X,Y)}$$

Outlier → y_i is far from the predicted

$\left|\frac{e_i}{\text{std } e_i}\right| > 3 \Rightarrow$ outlier

Z-score

$$Z = \frac{X - \mu}{\sigma} \quad > 3 \text{ or } < -3$$

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

7. We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.
(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

The probability that the first bootstrap observation is not the jth observation from the original sample is given by the complement of the probability that it is the jth observation. Assuming the bootstrap sample is obtained by sampling with replacement, the probability of selecting any specific observation from the original sample is 1/n. Therefore, the probability that the first bootstrap observation is not the jth observation is (1 - 1/n).

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

For the second bootstrap observation, the probability that the second bootstrap observation is not the jth observation from the original sample is also (1 - 1/n), as each observation has the same probability of being selected in each draw.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is (1 - 1/n)^n.

The probability that the jth observation is not in the bootstrap sample after n draws is the product of the probability that it is not selected in each draw, which is (1 - 1/n) for each draw. Therefore, the overall probability is (1 - 1/n) * (1 - 1/n) * ... * (1 - 1/n) = (1 - 1/n)^n.

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

When n = 5, the probability that the jth observation is in the bootstrap sample is approximately (1 - 1/5)^5 = (4/5)^5=0.32768

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

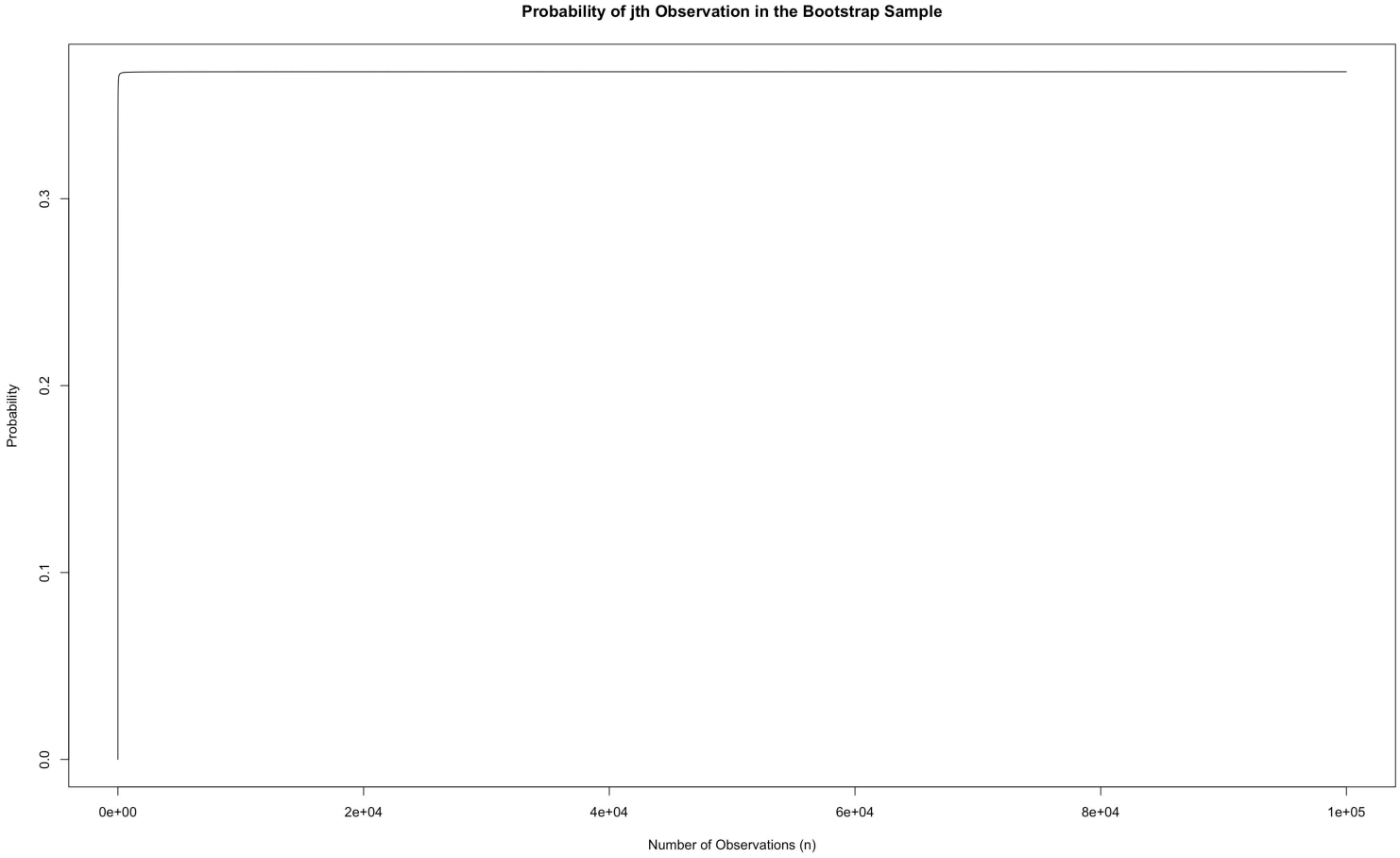
When n = 100, the probability that the jth observation is in the bootstrap sample is approximately (1 - 1/100)^100=0.3660323

(f) When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?

When n = 10,000, the probability that the jth observation is in the bootstrap sample is approximately (1 - 1/10000)^10000=0.367861

(g) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```
> n_range_values<-1:100000
> n_range_values_prob <- (1 - 1/n_range_values)^n_range_values
> plot(n_range_values, n_range_values_prob, type = "l", xlab = "Number of Observations (n)", ylab = "Probability", main = "Probability of jth Observation
in the Bootstrap Sample")
```



From the plot, we can observe that as the number of observations n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample approaches a limit of approximately 0.3678776 (estimated based on (1 - 1/100000)^100000).

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

```
> store=rep(NA, 10000)
> for(i in 1:10000){
store[i]=sum(sample(1:100, rep=TRUE)==4)>0
}
> mean(store)
 0.6411
```

Comment on the results obtained.

The result suggests that there is approximately a 64.11% chance that the fourth observation is included in a bootstrap sample of size 100. This indicates a relatively high likelihood of sampling the jth observation in such bootstrap samples given the small sample size and the random nature of bootstrap sampling.

8. In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that uses income and balance to predict default.

```
> View(default_df)
> default_df$default<-as.factor(default_df$default)
> log_reg_fit=glm(default~income+balance,data=default_df,family=binomial)
> summary(log_reg_fit)

Call:
glm(formula = default ~ income + balance, family = binomial,
    data = default_df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.
ii. Fit a multiple logistic regression model using only the training observations.
iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.
iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
> train_indices<-sample(nrow(default_df),0.5*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.0264
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred   No  Yes
             No  4814  118
             Yes   14   54
```

Validation Error = (14 + 118) / (14 + 118 + 4814 + 54) = 0.0264 = 2.64%

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
> train_indices<-sample(nrow(default_df),0.7*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.02533333
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred   No  Yes
             No  2893   59
             Yes   17   31
> train_indices<-sample(nrow(default_df),0.8*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.022
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred   No  Yes
             No  1933   33
             Yes   11   23
> train_indices<-sample(nrow(default_df),0.9*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.028
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred  No Yes
             No  959  25
             Yes   3  13
```

We tried 3 different variations of split which are 70% train and 30% test, 80% train and 20% test and 90% train and 10% test. 80/20 gave the least validation error (2.2%) followed by 70/30 (2.53%) and 90/10 gave the highest validation error (2.8%).

(e) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

```
> train_indices<-sample(nrow(default_df),0.5*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance+student,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.0254
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred   No  Yes
             No  4821  104
             Yes   23   52
> train_indices<-sample(nrow(default_df),0.7*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance+student,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.02666667
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred   No  Yes
             No  2895   67
             Yes   13   25
> train_indices<-sample(nrow(default_df),0.8*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance+student,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.0205
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred   No  Yes
             No  1940   36
             Yes    5   19
> train_indices<-sample(nrow(default_df),0.9*nrow(default_df))
> train_set<-default_df[train_indices,]
> validation_set<-default_df[-train_indices,]
> log_reg_fit=glm(default~income+balance+student,data=train_set,family=binomial)
> log_reg_fit_probs<-predict(log_reg_fit,validation_set,type="response")
> log_reg_fit_pred<-ifelse(log_reg_fit_probs>0.5,"Yes","No")
> cat("Validation Error =",mean(log_reg_fit_pred!=validation_set$default),"\n")
Validation Error = 0.031
> table(log_reg_fit_pred,validation_set$default)

log_reg_fit_pred  No Yes
             No  962  23
             Yes   8   7
```

Based on the above experiments where the train and validation split used was 50/50, 70/30, 80/20 and 90/10 we found that for 80/20 split including the student dummy variable helped achieve the lowest validation error (2.05%) better compared to the model where the student dummy variable was not used. We can see a decrease of (2.2 - 2.05) = 0.15% compared to the model where the student dummy variable was excluded from the model. Comparing the 50/50 split with the previous two models we can see that the addition of student variable gives a 2.54 % validation error whereas excluding gives a closer value of 2.53% in predicting the default cases. For this split excluding the student, the variable makes more sense than its addition as it reduces the total number of parameters as well as validation error.

9. We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

(a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

```
> set.seed(1)
> log_reg_fit=glm(default~income+balance,data=default_df,family=binomial)
> summary(log_reg_fit)$coef
                 Estimate   Std. Error    z value      Pr(>|z|)
(Intercept) -1.154047e+01 4.347564e-01 -26.544680 2.958355e-155
income       2.080898e-05 4.985167e-06   4.174178  2.990638e-05
balance      5.647103e-03 2.273731e-04  24.836280 3.638120e-136
> summary(log_reg_fit)$coef[,"Std. Error"]
 (Intercept)       income      balance
4.347564e-01 4.985167e-06 2.273731e-04
```

(b) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
> library(boot)
> boot.fn<-function(df,indices) {
+   log_reg_fit<-glm(default~income+balance,data=df[indices,],family=binomial)
+   return(coef(log_reg_fit))
+ }
```

(c) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

```
> boot_results<-boot(default_df,boot.fn,R=1000)
> summary(boot_results)
          Length Class      Mode
t0           3  -none-     numeric
t          3000  -none-     numeric
R             1  -none-     numeric
data          4  data.frame list
seed        626  -none-     numeric
statistic     1  -none-     function
sim           1  -none-     character
call          4  -none-     call
stype         1  -none-     character
strata    10000  -none-     numeric
weights   10000  -none-     numeric
> names(boot_results)
 [1] "t0"        "t"         "R"         "data"      "seed"      "statistic" "sim"       "call"      "stype"     "strata"    "weights"
> boot_results

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = default_df, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -2.914609e-02 4.492830e-01
t2*  2.080898e-05  1.259753e-07 4.988746e-06
t3*  5.647103e-03  1.268814e-05 2.347490e-04
```

(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

```
> cat("Standard Errors from glm() function:",summary(log_reg_fit)$coef[,"Std. Error"],"\n")
Standard Errors from glm() function: 0.4347564 4.985167e-06 0.0002273731
> cat("Standard Errors from bootstrap function:","\n")
Standard Errors from bootstrap function:
> boot_results

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = default_df, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -2.914609e-02 4.492830e-01
t2*  2.080898e-05  1.259753e-07 4.988746e-06
t3*  5.647103e-03  1.268814e-05 2.347490e-04
> summary(log_reg_fit)

Call:
glm(formula = default ~ income + balance, family = binomial,
    data = default_df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8

> summary(log_reg_fit)$coef[,"Std. Error"]
  (Intercept)       income      balance
4.347564e-01 4.985167e-06 2.273731e-04
```

Results for estimated standard errors from our logistic model are beta0 as 4.347e-01, beta1 as 4.985e-06 and beta2 as 2.273e-04 and from our bootstrap method we got beta0 as 4.348e-01, beta1 as 4.985e-06 and beta2 as 2.274e-04. The fact that the estimated standard errors from both the logistic model and the bootstrap method are very similar means that we can trust both methods to give us accurate estimates of how uncertain our coefficient estimates are. It indicates that the logistic regression model's assumptions are likely met, and the bootstrap method effectively captures the variability in the coefficient estimates due to sampling variation. The consistency between the two approaches enhances our confidence in the accuracy of the standard error estimates for the logistic regression coefficients.