SDM_Assignment3_2

Sri Balaji Muruganandam

25/10/2021

Setting Working Directory

```
rm(list = ls())
setwd("G:\\SDM_Sem01\\Assignment3")
```

Importing necessary libraries

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.1
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.1
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.1.1
```

```
## Loading required package: lattice
```

```
library(class)
library(MASS)
```

Viewing the Sample data

```
head(Weekly, 5)
```

```
##   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
```

```
dim(Weekly)
```

```
## [1] 1089    9
```

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

Exploring the high level overview of the data

```
str(Weekly)
```

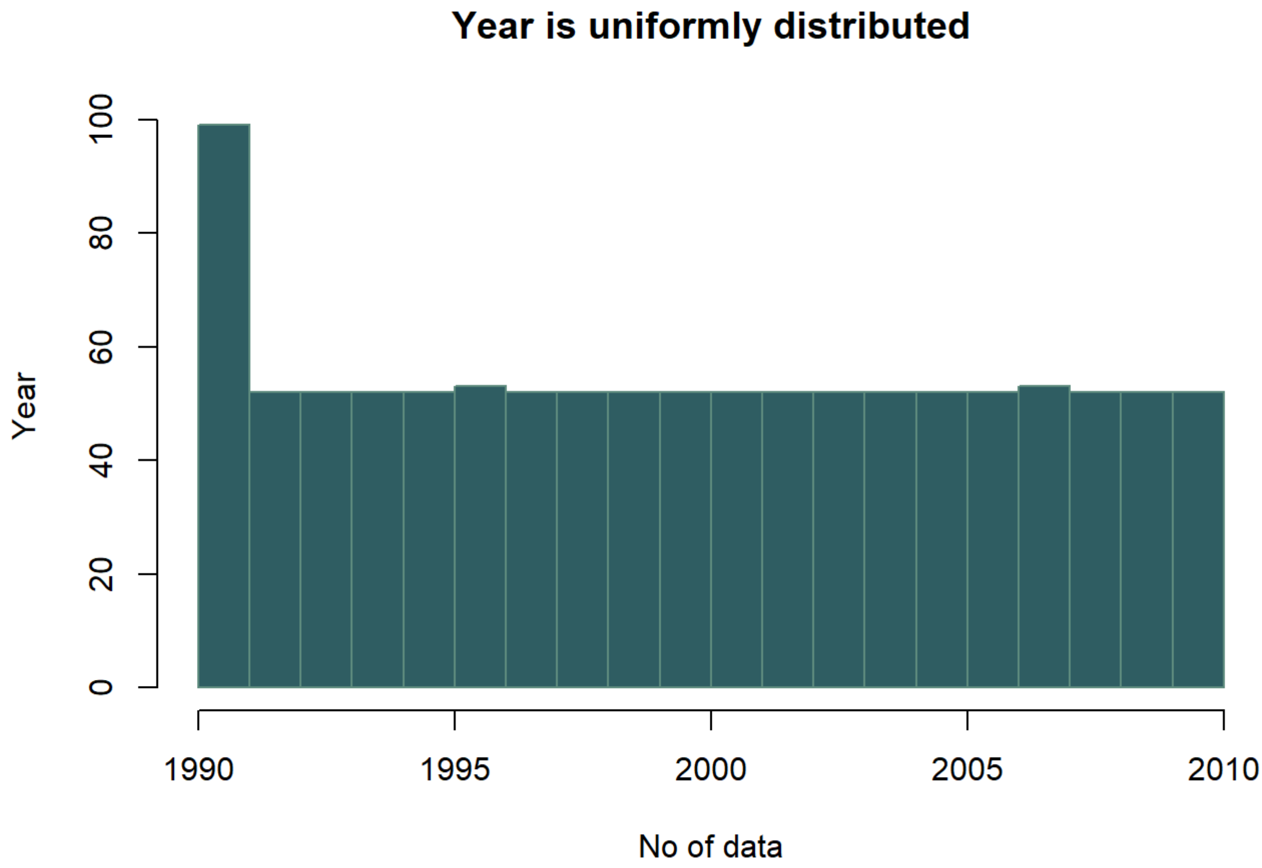
```
## 'data.frame':    1089 obs. of  9 variables:
##  $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
##  $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
##  $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
##  $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
##  $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
##  $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
##  $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
##  $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

```
summary(Weekly)
```

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

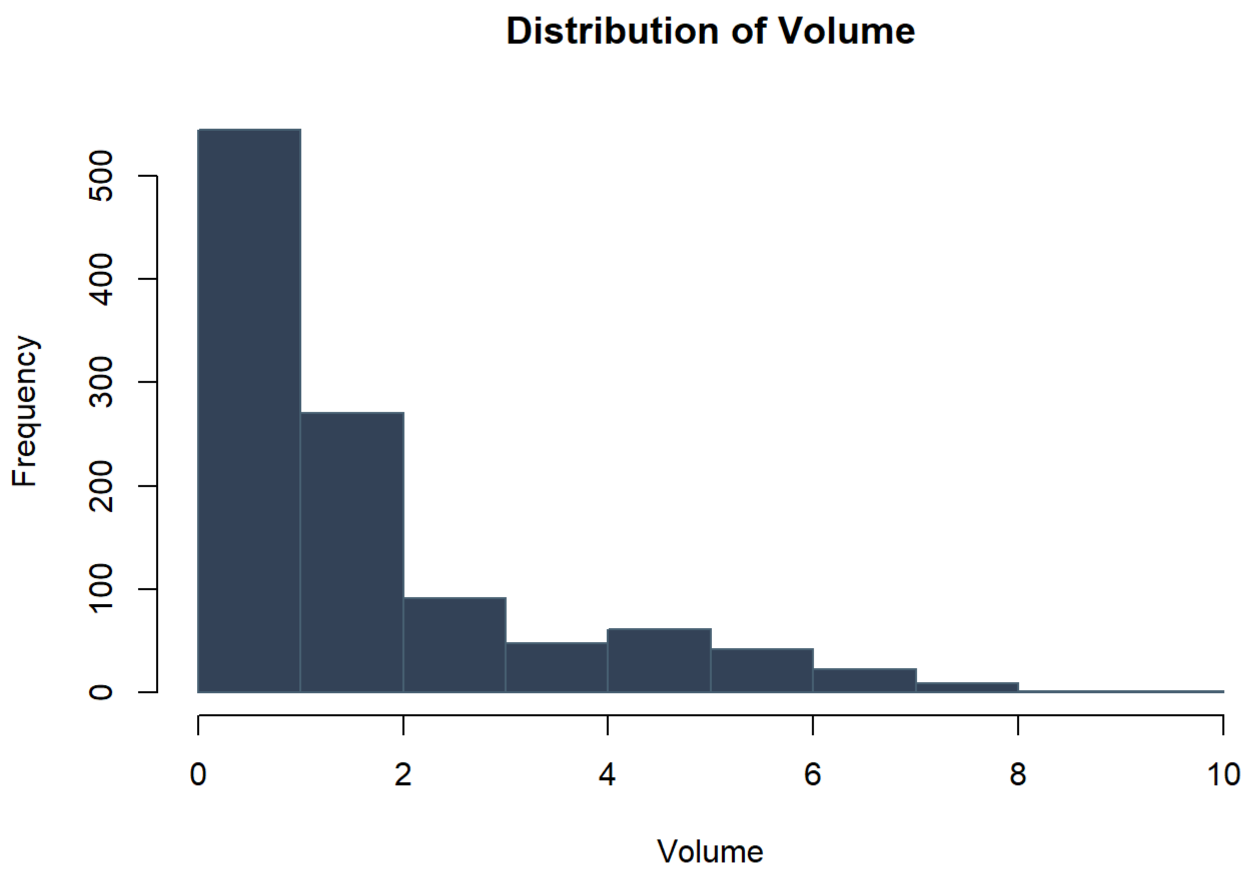
The values are uniformly distributed over the year

```
hist(Weekly$Year,breaks = 20, col="#2F5D62", border = "#5E8B7E", main = "Year is uniformly di
stributed",xlab = "No of data", ylab = "Year")
```



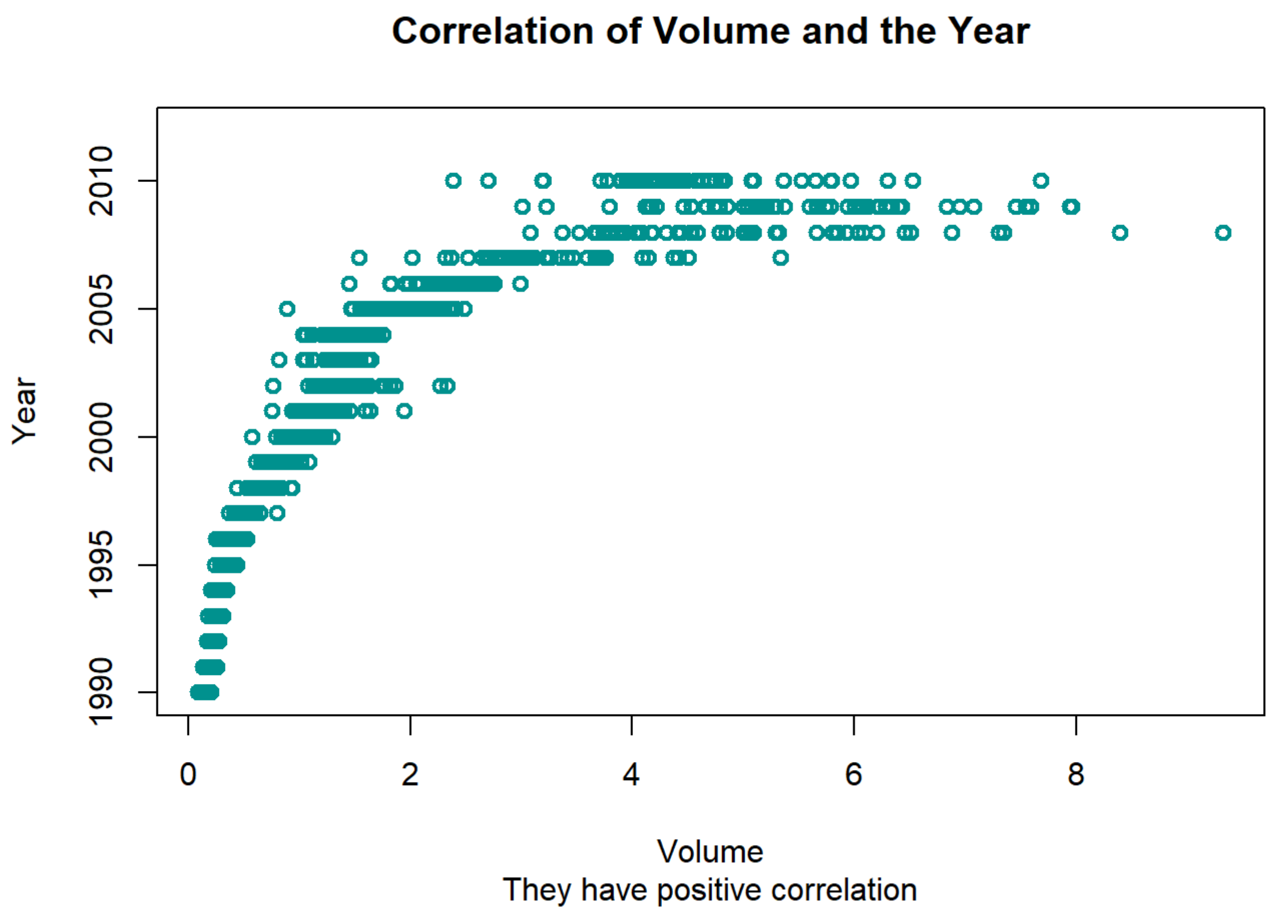
Distribution of Volume

```
hist(Weekly$Volume,breaks = 12, col="#334257", border = "#476072", main = "Distribution of Vo
lume",xlab = "Volume", ylab = "Frequency")
```



There is a positive correlation between Volume and the Year

```
plot(Weekly$Volume, Weekly$Year,col="#00918E", xlab = "Volume", ylab = "Year", ylim = c(1990,
2012), main = "Correlation of Volume and the Year", sub="They have positive correlation", lwd
= 2.3)
```

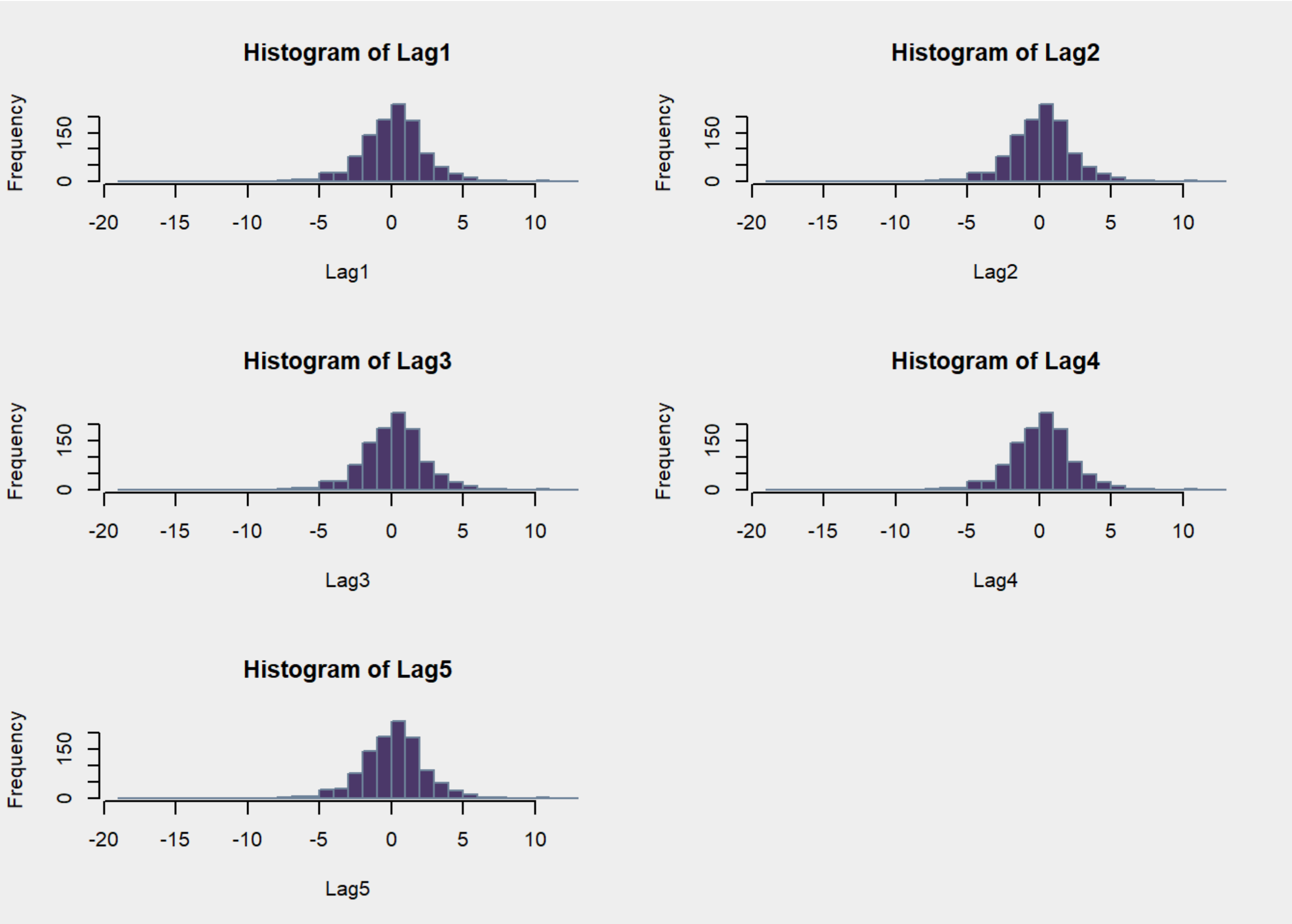



```
table(Weekly$Direction)
```

```
##
## Down   Up
##  484  605
```

Lags are normally distributed

```
par(mfrow = c(3,2),bg = '#EEEEEE')
hist(Weekly$Lag1,breaks = 25, col="#4B3869", border = "#6D8299", main = "Histogram of Lag1",
 xlab = "Lag1", ylab = "Frequency")
hist(Weekly$Lag2,breaks = 25, col="#4B3869", border = "#6D8299", main = "Histogram of Lag2",
 xlab = "Lag2", ylab = "Frequency")
hist(Weekly$Lag3,breaks = 25, col="#4B3869", border = "#6D8299", main = "Histogram of Lag3",
 xlab = "Lag3", ylab = "Frequency")
hist(Weekly$Lag4,breaks = 25, col="#4B3869", border = "#6D8299", main = "Histogram of Lag4",
 xlab = "Lag4", ylab = "Frequency")
hist(Weekly$Lag5,breaks = 25, col="#4B3869", border = "#6D8299", main = "Histogram of Lag5",
 xlab = "Lag5", ylab = "Frequency")
```



b) Use the full data to perform 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? Comment on these.

Splitting the data into training and the test data

```
set.seed(23)
random_index = sample(c(1:nrow(Weekly)), size = round(8/10 * nrow(Weekly)), replace = FALSE)
data1 = Weekly[,-c(1,8)]
train_data <- data1[random_index,]
test_data <- data1[-random_index,]
train_data = data.frame(train_data)
test_data = data.frame(test_data)
data1 = data.frame(data1)

y_train_data <- as.numeric(train_data$Direction)-1
y_test_data <- as.numeric(test_data$Direction)-1
dim(train_data)
```

```
## [1] 871   7
```

```r
dim(test_data)
```

```
## [1] 218   7
```

Modelling Logistic Regression

```r
model = glm(Direction~., data = data1, family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = Direction ~ ., family = "binomial", data = data1)
##
## 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
```

The predictor Lag2 has a coefficient of 0.05844 and seems to be significant for classification

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.

Prediction for train data

```
train_predict = predict(model, newdata = train_data, type = "response")
y_predict_train = round(train_predict)
```

Predicting for new values

```
test_predict = predict(model, newdata = test_data, type = "response")
y_predict_test = round(test_predict)
```

Calculating the error

```
train_error <- sum(abs(y_predict_train- y_train_data))/length(y_train_data)
test_error <- sum(abs(y_predict_test- y_test_data))/length(y_test_data)
print(train_error)
```

```
## [1] 0.4362801
```

```
print(test_error)
```

```
## [1] 0.4495413
```

Computing the confusion matrix

```
conf <- confusionMatrix(as.factor(y_predict_test), as.factor(y_test_data))
names(conf)
```

```
## [1] "positive" "table"    "overall"  "byClass"  "mode"     "dots"
```

```
conf$table
```

```
##           Reference
## Prediction   0   1
##          0   8   8
##          1  90 112
```

```
conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   5.504587e-01   1.620925e-02   4.818293e-01   6.176972e-01   5.504587e-01
## AccuracyPValue  McnemarPValue
##   5.280360e-01   2.786256e-16
```

From the result it is shown that 112 UP’s and 8 DOWN’s are predicted well.

Whereas 8 UP's are incorrectly predicted as DOWN's and 90 DOWN's are incorrectly predicted as UP's

Accuracy of the model is 5.504587e-01

Calculating the accuracy using formula

```
false_result = which(y_predict_test==y_test_data)
accuracy=length(false_result) / length(y_test_data)
print(accuracy)
```

```
## [1] 0.5504587
```

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

Creating new dataset for the training and testing data based on the new conditions

```
data2 = Weekly[,c(1,3,9)]
random_index = which(data2$Year %in% c(1990:2008))
knn_index = Weekly$Year <= 2008
train_data2 <- data2[random_index,]
test_data2 <- data2[-random_index,]

knn_train = Weekly[knn_index,"Lag2",drop=F]
knn_test = Weekly[knn_index,"Lag2",drop=F]
knn_y_train = Weekly[knn_index,"Direction",drop=T]
knn_y_test = Weekly[knn_index,"Direction",drop=T]

train_data2 = data.frame(train_data2[,-c(1)])
test_data2 = data.frame(test_data2[,-c(1)])
data2 = data.frame(data2[,-c(1)])


y_train_data2 <- as.numeric(train_data2$Direction)-1
y_test_data2 <- as.numeric(test_data2$Direction)-1
dim(train_data2)
```

```
## [1] 985   2
```

```
dim(test_data2)
```

```
## [1] 104   2
```

```
head(data2,4)
```

```
##     Lag2 Direction
## 1  1.572      Down
## 2  0.816      Down
## 3 -0.270        Up
## 4 -2.576        Up
```

```
head(train_data2,2)
```

```
##    Lag2 Direction
## 1 1.572      Down
## 2 0.816      Down
```

```
head(test_data2,2)
```

```
##       Lag2 Direction
## 986 -1.698      Down
## 987  6.760      Down
```

Modelling Logistic Regression with the newly generated data

```
model2 = glm(Direction~., data = data2, family = "binomial")
summary(model2)
```

```
## 
## Call:
## glm(formula = Direction ~ ., family = "binomial", data = data2)
## 
## Deviance Residuals: 
##    Min      1Q  Median      3Q     Max  
## -1.564  -1.267   1.008   1.086   1.386  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  0.21473    0.06123   3.507 0.000453 ***
## Lag2         0.06279    0.02636   2.382 0.017230 *  
## ---
## 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: 1490.4  on 1087  degrees of freedom
## AIC: 1494.4
## 
## Number of Fisher Scoring iterations: 4
```

Prediction for train data

```
train_predict2 = predict(model2, newdata = train_data2, type = "response")
y_predict_train2 = round(train_predict2)
```

Predicting for new values

```
test_predict2 = predict(model2, newdata = test_data2, type = "response")
y_predict_test2 = round(test_predict2)
```

Calculating the error

```
train_error2 <- sum(abs(y_predict_train2- y_train_data2))/length(y_train_data2)
test_error2 <- sum(abs(y_predict_test2- y_test_data2))/length(y_test_data2)
print(train_error2)
```

```
## [1] 0.4446701
```

```
print(test_error2)
```

```
## [1] 0.375
```

With the newly generated data, the training error is 0.4446701 and the testing error is 0.375

Computing the confusion matrix

```
conf2 <- confusionMatrix(as.factor(y_predict_test2), as.factor(y_test_data2))
names(conf2)
```

```
## [1] "positive" "table"    "overall"  "byClass"  "mode"     "dots"
```

```
conf2$table
```

```
##           Reference
## Prediction  0  1
##          0  9  5
##          1 34 56
```

```
conf2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.250000e-01   1.414056e-01   5.246597e-01   7.180252e-01   5.865385e-01
## AccuracyPValue  McnemarPValue
##   2.439500e-01   7.339821e-06
```

From the result it is shown that 56 UP’s and 9 DOWN’s are predicted well.

Whereas 5 UP’s are incorrectly predicted as DOWN’s and 34 DOWN’s are incorrectly predicted as UP’s

Accuracy of the model is 6.250000e-01

e) Repeat (d) using LDA.

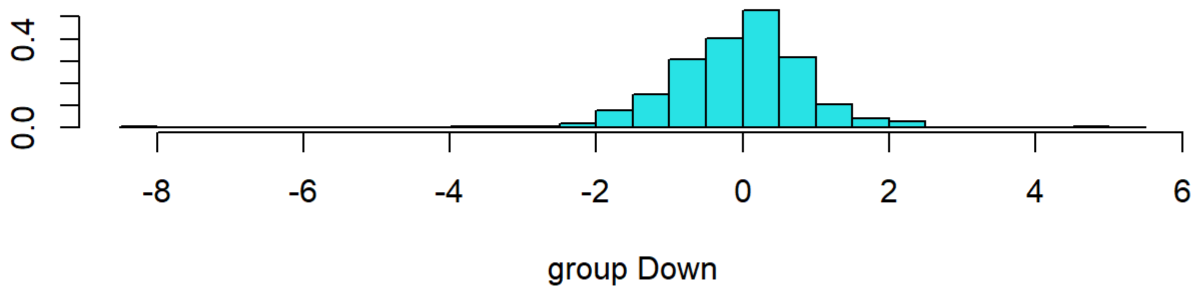
Modelling LDA

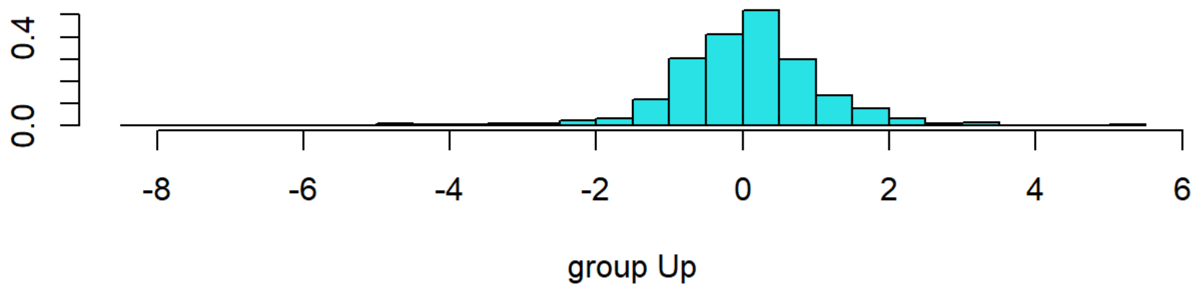
```
lda_model = lda(Direction~., data = train_data2)
summary(lda_model)
```

```
##         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    3      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

Plotting the values of LDA

```
plot(lda_model)
```





Predicting for test data

```
lda_predict_train = predict(lda_model, newdata = train_data2)
lda_predict_test = predict(lda_model, newdata = test_data2)
res_train = as.numeric(lda_predict_train$class)-1
res_test = as.numeric(lda_predict_test$class)-1
```

Calculating the train and test errors

```
lda_result_train = which(res_train==y_train_data2)
lda_train_error=length(lda_result_train) / length(y_train_data2)
print(lda_train_error)
```

```
## [1] 0.5543147
```

```
lda_result_test = which(res_test==y_test_data2)
lda_test_error=length(lda_result_test) / length(y_test_data2)
print(lda_test_error)
```

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

The train error is 0.5543147 and the test error is 0.625 when predicted using LDA.

f) Repeat (d) using KNN with k=1.

Predicting using KNN with k = 1

```
set.seed(23)
knn_model = knn(knn_train,knn_test,knn_y_train,k=1)
```

Calculating Error

```
error_knn<- mean(knn_model != knn_y_test)
print(error_knn)
```

```
## [1] 0.04060914
```

Calculating Accuracy

```
false_result_knn = which(knn_model == knn_y_test)
knn_accuracy = length(false_result_knn)/length(knn_y_test)
print(knn_accuracy)
```

```
## [1] 0.9593909
```

g) Which method appears to provide the best results?

KNN model provides the best prediction with an accuracy of 0.9593909. Other models provide less accuracy than KNN model.

h) Experiment with different combinations of predictors, including possible transformations and interactions, for each method. 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.

Performing Logistic Regression

Now considering only the return in week 1, 2 and 3

Prediction for train data

```
model_test = glm(Direction~Lag1+Lag2+Lag3, data = data1, family = "binomial")
train_predict = predict(model_test, newdata = train_data, type = "response")
y_predict_train = round(train_predict)
```

Predicting for new values

```
test_predict = predict(model, newdata = test_data, type = "response")
y_predict_test = round(test_predict)
```

Calculating the error

```
train_error <- sum(abs(y_predict_train- y_train_data))/length(y_train_data)
test_error <- sum(abs(y_predict_test- y_test_data))/length(y_test_data)
print(train_error)
```

```
## [1] 0.445465
```

```
print(test_error)
```

```
## [1] 0.4495413
```

Computing the confusion matrix

```
conf2 <- confusionMatrix(as.factor(y_predict_test), as.factor(y_test_data))
names(conf2)
```

```
## [1] "positive" "table"    "overall"  "byClass"  "mode"     "dots"
```

```
conf2$table
```

```
##           Reference
## Prediction   0   1
##          0   8   8
##          1  90 112
```

```
conf2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   5.504587e-01   1.620925e-02   4.818293e-01   6.176972e-01   5.504587e-01
## AccuracyPValue  McnemarPValue
##   5.280360e-01   2.786256e-16
```

The train error is 0.445465 and the test error is 0.4495413 for the above model

Modelling LDA

Considering only the columns Lag2, Lag4 Lag5 along with the Volume

```
lda_model = lda(Direction~Lag2+Lag4+Lag5+Volume, data = train_data)
summary(lda_model)
```

```
##         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
```

Predicting for test data

```
lda_predict_train = predict(lda_model, newdata = train_data)
lda_predict_test = predict(lda_model, newdata = test_data)
res_train = as.numeric(lda_predict_train$class)-1
res_test = as.numeric(lda_predict_test$class)-1
```

Calculating the train and test errors

```
lda_result_train = which(res_train==y_train_data)
lda_train_error=length(lda_result_train) / length(y_train_data)
print(lda_train_error)
```

```
## [1] 0.5579793
```

```
lda_result_test = which(res_test==y_test_data)
lda_test_error=length(lda_result_test) / length(y_test_data)
print(lda_test_error)
```

```
## [1] 0.5550459
```

The train error is 0.5579793 and the test error is 0.5550459 when predicted using LDA.

Modelling KNN with various k values (k = 3,5,7,9,11,13,15)

```
kvalues = c(3,5,7,9,11,13,15)
error_knn = c()
knn_accuracy = c()
for(i in 1:7) {
  predict_knn <- knn(knn_train,knn_test,knn_y_train,k=i)
  error_knn[i]<- mean(predict_knn != knn_y_test)

  false_result_knn = which(predict_knn == knn_y_test)
  knn_accuracy[i] = length(false_result_knn)/length(knn_y_test)
}
```

Error at each k value

```
print(error_knn)
```

```
## [1] 0.04060914 0.27208122 0.26700508 0.33299492 0.34213198 0.34923858 0.36040609
```

```
print(min(error_knn))
```

```
## [1] 0.04060914
```

Accuracy at each k value

```
print(knn_accuracy)
```

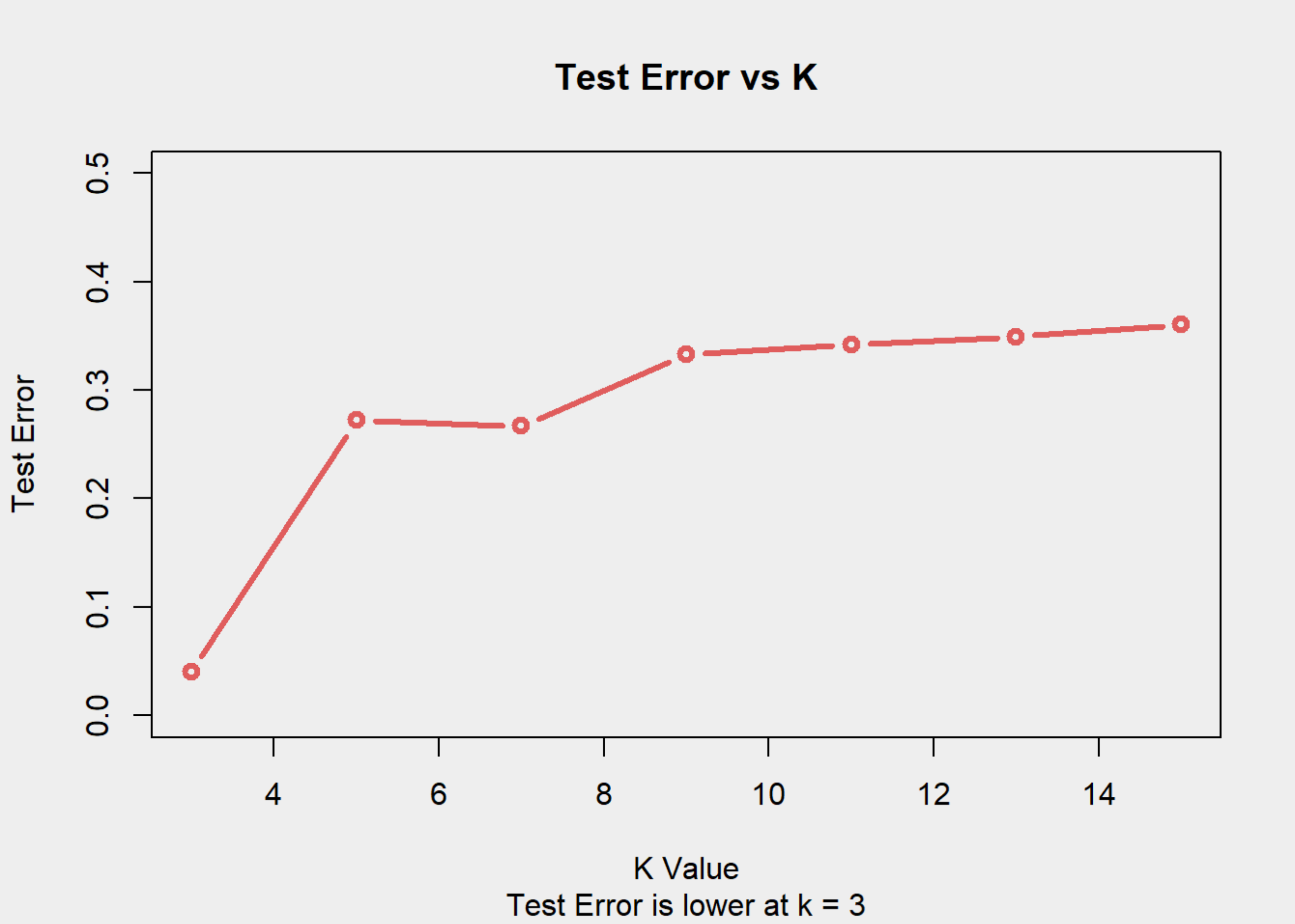
```
## [1] 0.9593909 0.7279188 0.7329949 0.6670051 0.6578680 0.6507614 0.6395939
```

```
print(max(knn_accuracy))
```

```
## [1] 0.9593909
```

Plotting K and the error value corresponding to each k value

```
par(bg = '#EEEEEE')
plot(kvalues,error_knn, col="#E05D5D", type = "b", xlab = "K Value", ylab = "Test Error", yli
m = c(0,0.5), main = "Test Error vs K", sub="Test Error is lower at k = 3", lwd = 3.0)
```



The error is low when the value of k =3