Biostat 212a Homework 2

Due Feb 8, 2025 @ 11:59PM

Wenqiang Ge UID:106371961

2025-02-05

Table of contents

ISL Exercise 4.8.1 (10pts)

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.

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

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

Solution:

1. Given logistic function: $\frac{p(X)}{1-p(X)} = e^{\beta_0 + \beta_1 X}$

then: $p(X) = (1 - p(X))\, e^{\beta_0 + \beta_1 X}$

$= e^{\beta_0 + \beta_1 X} - p(X) \cdot (e^{\beta_0 + \beta_1 X})$

so $p(X) + p(X) \cdot (e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$

$p(X) \cdot (1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$

Thus, we proved: $p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$

ISL Exercise 4.8.6 (10pts)

6. Suppose we collect data for a group of students in a statistics class with variables $X_1 =$ hours studied, $X_2 =$ undergrad GPA, and $Y =$ receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6, \hat{\beta}_1 = 0.05, \hat{\beta}_2 = 1$.

   (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.

   (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?

Solution:

6. We have logistic regression equation: $P(Y=1 \mid X_1, X_2) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2}}$

where: $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$

(a). For $X_1 = 40$, $X_2 = 3.5$:

$$P(Y_1 \mid X_1 = 40, X_2 = 3.5) = \frac{e^{-6 + 0.05 \times 40 + 3.5}}{1 + e^{-6 + 0.05 \times 40 + 3.5}} = \frac{e^{-0.5}}{1 + e^{-0.5}} \approx 0.378$$

So the probability of getting an A in class is 37.8%

(b). In order to have 50% chance of getting an A: $P(Y=1 \mid X_1, X_2) =$

So $\frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2}} = 0.5$, $e^{\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2} = 1$

let $X_2 = 3.5$, $\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 = 0$, so $-6 + 3.5 + 0.05 X_1 = 0$, $X_1 = 50$

The student needs to study 50 hours to have a 50% chance of getting an A in the class.

ISL Exercise 4.8.9 (10pts)

9. This problem has to do with *odds*.

   (a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

   (b) Suppose that an individual has a 16 % chance of defaulting on her credit card payment. What are the odds that she will default?

Solution:

9. (a) we have $\text{odds} = \frac{P(Y=1)}{1-P(Y=1)} = 0.37$

so $P(Y=1) = \frac{\text{odds}}{1+\text{odds}} = \frac{0.37}{1+0.37} \approx 0.27 = 27\%$

The fraction of people with an odds of 0.37 of defaulting on their credit card payment who will actually default is 27%.

(b) $\text{odds} = \frac{P}{1-P} = \frac{0.16}{1-0.16} \approx 0.19 \approx 19\%$

The odds is 0.19 (19%) that she will default.

ISL Exercise 4.8.13 (a)-(i) (50pts)

13. This question should be answered using the `Weekly` data set, which is part of the `ISLR2` 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?

    (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?

(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.

(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).

(e) Repeat (d) using LDA.

(f) Repeat (d) using QDA.

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

(h) Repeat (d) using naive Bayes.

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

Solution:

```
library(ISLR2)
library(MASS)

data("Weekly")
# Structure of the dataset
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 ...
```

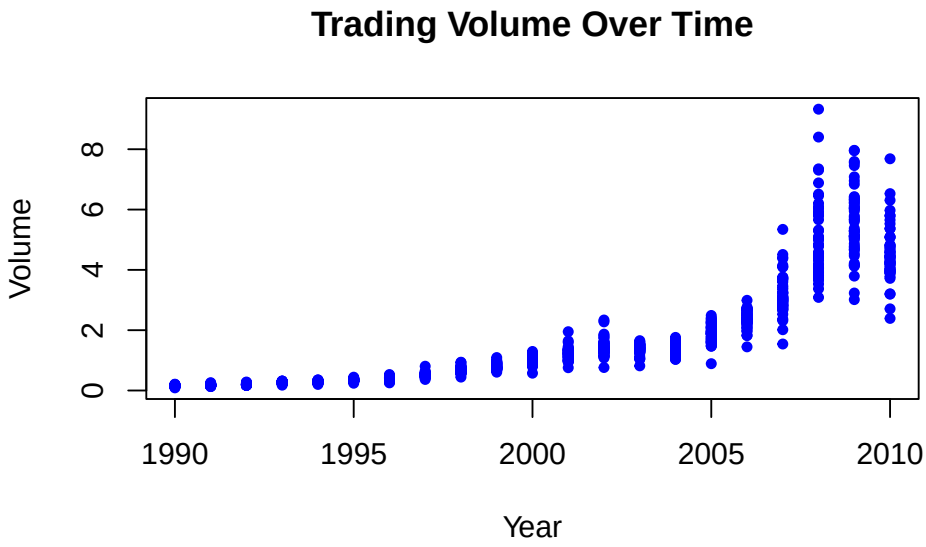
```
$ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

(a)

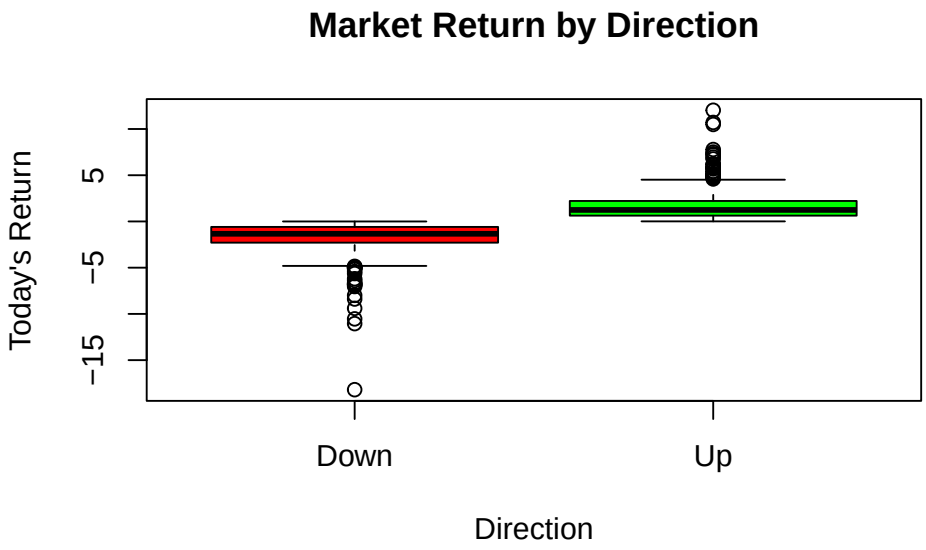
```
# Numerical summary
summary(Weekly)
```

```
     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
```

```
# Plot the Volume over time
plot(Weekly$Year, Weekly$Volume, main="Trading Volume Over Time", xlab="Year", ylab="Volu
```




```
# Boxplot of market return (Today) by Direction
boxplot(Today ~ Direction, data=Weekly, main="Market Return by Direction", ylab="Today's
```




```
# Correlation matrix (excluding categorical variables)
cor(Weekly[, -9])  # Exclude the Direction column
```

```
             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
```

Volume has increased significantly over time. Returns (Today) have a mean near zero, indicating relatively balanced ups and downs in market movement. The Direction variable (Up/Down) suggests a roughly even split, meaning the market is not strongly biased in one direction.

Upward (Up) and downward (Down) movements have different distributions. Down days have more extreme negative outliers, indicating higher risk when the market declines. The median return is slightly higher for "Up" movements, but variance is similar for both.

(b)

```r
# Fit logistic regression model
logistic_model <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                      data = Weekly, family = binomial)

# Summary of the logistic regression model
summary(logistic_model)
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
```

```
Lag4         -0.02779    0.02646  -1.050   0.2937
Lag5         -0.01447    0.02638  -0.549   0.5833
Volume       -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

$log(\frac{P(Up)}{1-P(Up)}) = 0.26686 - 0.04127 \times \text{Lag1} + 0.05844 \times \text{Lag2} - 0.01066 \times \text{Lag3} - 0.02779 \times \text{Lag4} - 0.01447 \times \text{Lag5} - 0.02274 \times \text{Volume}$.

Yes, the p-values of Lag1 and Lag3 are less than 0.05, so they are statistically significant.

(c)

```r
# Predict probabilities
pred_probs <- predict(logistic_model, type="response")

# Convert probabilities to class predictions (threshold = 0.5)
pred_classes <- ifelse(pred_probs > 0.5, "Up", "Down")

# Create confusion matrix
conf_matrix <- table(Predicted = pred_classes, Actual = Weekly$Direction)

# Compute accuracy
accuracy <- mean(pred_classes == Weekly$Direction)

# Print results
print(conf_matrix)
```

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

```r
print(paste("Overall accuracy:", round(accuracy, 4)))
```

```
[1] "Overall accuracy: 0.5611"
```

True Positives (TP) = 557 ; False Positives (FP) = 430 ; True Negatives (TN)

= 54 ; False Negatives (FN) = 48

Accuracy= $\frac{TP+TN}{TotalSamples} = \frac{557+54}{54+48+430+557} = 0.5611$. This is only slightly better than random guessing (50%).

The model is biased towards predicting "Up", as indicated by the large number of false positives (FP = 430). The model fails to predict "Down" accurately, with only 54 correct "Down" predictions out of 484 actual "Down" instances.

(d)

```
# Split the dataset
train <- Weekly$Year < 2009
train_data <- Weekly[train, ]
test_data <- Weekly[!train, ]

# Fit logistic regression using Lag2
logistic_model_lag2 <- glm(Direction ~ Lag2, data=train_data, family=binomial)

# Predict on test data
test_probs <- predict(logistic_model_lag2, newdata=test_data, type="response")

# Convert probabilities to class labels
test_preds <- ifelse(test_probs > 0.5, "Up", "Down")

# Compute confusion matrix
conf_matrix_test <- table(Predicted = test_preds, Actual = test_data$Direction)

# Compute accuracy
test_accuracy <- mean(test_preds == test_data$Direction)

# Print results
print(conf_matrix_test)
```

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

```
print(paste("Test accuracy:", round(test_accuracy, 4)))
```

```
[1] "Test accuracy: 0.625"
```

(e)

```r
# Fit LDA model
lda_model <- lda(Direction ~ Lag2, data=train_data)

# Predict on test data
lda_preds <- predict(lda_model, newdata=test_data)

# Extract class predictions
lda_classes <- lda_preds$class

# Create confusion matrix
conf_matrix_lda <- table(Predicted = lda_classes, Actual = test_data$Direction)

# Compute accuracy
lda_accuracy <- mean(lda_classes == test_data$Direction)

# Print results
print(conf_matrix_lda)
```

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

```r
print(paste("LDA test accuracy:", round(lda_accuracy, 4)))
```

```
[1] "LDA test accuracy: 0.625"
```

(f)

```r
# Fit QDA model
qda_model <- qda(Direction ~ Lag2, data=train_data)

# Predict on test data
qda_preds <- predict(qda_model, newdata=test_data)

# Extract class predictions
qda_classes <- qda_preds$class

# Compute confusion matrix
conf_matrix_qda <- table(Predicted = qda_classes, Actual = test_data$Direction)

# Compute accuracy
qda_accuracy <- mean(qda_classes == test_data$Direction)

# Print results
```

```
  print(conf_matrix_qda)
```

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

```
  print(paste("QDA test accuracy:", round(qda_accuracy, 4)))
```

```
[1] "QDA test accuracy: 0.5865"
```

(g)

```
  library(class)

  # Prepare training and test data
  train_X <- train_data$Lag2
  test_X <- test_data$Lag2
  train_Y <- train_data$Direction
  test_Y <- test_data$Direction

  # Apply KNN with K=1
  knn_preds <- knn(train = matrix(train_X), test = matrix(test_X),
                   cl = train_Y, k = 1)

  # Compute confusion matrix
  conf_matrix_knn <- table(Predicted = knn_preds, Actual = test_Y)

  # Compute accuracy
  knn_accuracy <- mean(knn_preds == test_Y)

  # Print results
  print(conf_matrix_knn)
```

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

```
  print(paste("KNN (K=1) test accuracy:", round(knn_accuracy, 4)))
```

```
[1] "KNN (K=1) test accuracy: 0.5"
```

(h)

```
library(e1071)

# Fit Naive Bayes model
nb_model <- naiveBayes(Direction ~ Lag2, data=train_data)

# Predict on test data
nb_preds <- predict(nb_model, newdata=test_data)

# Compute confusion matrix
conf_matrix_nb <- table(Predicted = nb_preds, Actual = test_data$Direction)

# Compute accuracy
nb_accuracy <- mean(nb_preds == test_data$Direction)

# Print results
print(conf_matrix_nb)
```

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

```
print(paste("Naive Bayes test accuracy:", round(nb_accuracy, 4)))
```

```
[1] "Naive Bayes test accuracy: 0.5865"
```

(i)

```
# Create a comparison table
model_comparison <- data.frame(
  Model = c("Logistic Regression", "LDA", "QDA", "KNN (K=1)",
            "Naive Bayes"),
  Accuracy = c(test_accuracy, lda_accuracy, qda_accuracy,
               knn_accuracy, nb_accuracy)
)

# Print comparison results
print(model_comparison)
```

```
               Model  Accuracy
1 Logistic Regression 0.6250000
2                 LDA 0.6250000
3                 QDA 0.5865385
4           KNN (K=1) 0.5000000
5         Naive Bayes 0.5865385
```

The Logistic Regression and LDA appear to have the best results on this data, and they both have 0.625 accuracy.

Bonus question: ISL Exercise 4.8.13 Part (j) (30pts)

(j) 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.

Solution:

(j) Logistic Regression with multiple predictors

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

# Predictions
test_probs_extended <- predict(logistic_model_extended, newdata=test_data, type="response
test_preds_extended <- ifelse(test_probs_extended > 0.5, "Up", "Down")

# Confusion Matrix
conf_matrix_logistic_extended <- table(Predicted = test_preds_extended, Actual = test_dat
logistic_accuracy_extended <- mean(test_preds_extended == test_data$Direction)

print(conf_matrix_logistic_extended)
```

```
         Actual
Predicted Down Up
     Down   31 44
     Up     12 17
```

```
print(paste("Extended Logistic Regression Accuracy:", round(logistic_accuracy_extended, 4
```

```
[1] "Extended Logistic Regression Accuracy: 0.4615"
```

Logistic Regression with interaction terms

```
logistic_model_interaction <- glm(Direction ~ Lag2 * Volume,
                                  data=train_data, family=binomial)
```

```r
# Predictions
test_probs_interaction <- predict(logistic_model_interaction,
                                  newdata=test_data, type="response")
test_preds_interaction <- ifelse(test_probs_interaction > 0.5, "Up", "Down")

# Confusion Matrix
conf_matrix_logistic_interaction <- table(Predicted = test_preds_interaction,
                                          Actual = test_data$Direction)
logistic_accuracy_interaction <- mean(
  test_preds_interaction == test_data$Direction
  )

print(conf_matrix_logistic_interaction)
```

```
         Actual
Predicted Down Up
     Down   20 25
     Up     23 36
```

```r
print(paste("Logistic Regression with Interaction Accuracy:",
            round(logistic_accuracy_interaction, 4)))
```

```
[1] "Logistic Regression with Interaction Accuracy: 0.5385"
```

LDA with More Predictors

```r
library(MASS)
lda_model_extended <- lda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                          data=train_data)

# Predictions
lda_preds_extended <- predict(lda_model_extended, newdata=test_data)$class

# Confusion Matrix
conf_matrix_lda_extended <- table(Predicted = lda_preds_extended, Actual =
                                    test_data$Direction)
lda_accuracy_extended <- mean(lda_preds_extended == test_data$Direction)

print(conf_matrix_lda_extended)
```

```
         Actual
Predicted Down Up
     Down   31 44
     Up     12 17
```

```
print(paste("Extended LDA Accuracy:", round(lda_accuracy_extended, 4)))
```

```
[1] "Extended LDA Accuracy: 0.4615"
```

QDA with More Predictors

```
qda_model_extended <- qda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                          data=train_data)

# Predictions
qda_preds_extended <- predict(qda_model_extended, newdata=test_data)$class

# Confusion Matrix
conf_matrix_qda_extended <- table(Predicted = qda_preds_extended, Actual =
                                    test_data$Direction)
qda_accuracy_extended <- mean(qda_preds_extended == test_data$Direction)

print(conf_matrix_qda_extended)
```

```
         Actual
Predicted Down Up
     Down   33 49
     Up     10 12
```

```
print(paste("Extended QDA Accuracy:", round(qda_accuracy_extended, 4)))
```

```
[1] "Extended QDA Accuracy: 0.4327"
```

Tuning K for KNN

```
library(class)

# Function to evaluate KNN for different K values
knn_evaluate <- function(k) {
  knn_preds <- knn(train=as.matrix(train_data[, c("Lag2")]),
                   test=as.matrix(test_data[, c("Lag2")]),
                   cl=train_data$Direction, k=k)

  conf_matrix_knn <- table(Predicted = knn_preds, Actual = test_data$Direction)
  knn_accuracy <- mean(knn_preds == test_data$Direction)

  return(list(conf_matrix=conf_matrix_knn, accuracy=knn_accuracy))
}

# Experiment with different values of K
```

```r
knn_results <- lapply(c(1, 3, 5, 7, 10, 15, 20), knn_evaluate)

# Print results for each K
for (i in 1:length(knn_results)) {
  print(paste("KNN with K =", c(1, 3, 5, 7, 10, 15, 20)[i]))
  print(knn_results[[i]]$conf_matrix)
  print(paste("Accuracy:", round(knn_results[[i]]$accuracy, 4)))
}
```

```
[1] "KNN with K = 1"
          Actual
Predicted Down Up
     Down   21 30
     Up     22 31
[1] "Accuracy: 0.5"
[1] "KNN with K = 3"
          Actual
Predicted Down Up
     Down   15 19
     Up     28 42
[1] "Accuracy: 0.5481"
[1] "KNN with K = 5"
          Actual
Predicted Down Up
     Down   15 21
     Up     28 40
[1] "Accuracy: 0.5288"
[1] "KNN with K = 7"
          Actual
Predicted Down Up
     Down   15 20
     Up     28 41
[1] "Accuracy: 0.5385"
[1] "KNN with K = 10"
          Actual
Predicted Down Up
     Down   17 19
     Up     26 42
[1] "Accuracy: 0.5673"
[1] "KNN with K = 15"
          Actual
Predicted Down Up
     Down   20 20
     Up     23 41
[1] "Accuracy: 0.5865"
```

```
[1] "KNN with K = 20"
          Actual
Predicted Down Up
     Down   20 20
     Up     23 41
[1] "Accuracy: 0.5865"
```

Naive Bayes with More Predictors

```
library(e1071)
nb_model_extended <- naiveBayes(Direction ~ Lag1 + Lag2 +
                                  Lag3 + Lag4 + Lag5 + Volume, data=train_data)

# Predictions
nb_preds_extended <- predict(nb_model_extended, newdata=test_data)

# Confusion Matrix
conf_matrix_nb_extended <- table(Predicted = nb_preds_extended, Actual =
                                   test_data$Direction)
nb_accuracy_extended <- mean(nb_preds_extended == test_data$Direction)

print(conf_matrix_nb_extended)
```

```
          Actual
Predicted Down Up
     Down   42 56
     Up      1  5
```

```
print(paste("Extended Naive Bayes Accuracy:", round(nb_accuracy_extended, 4)))
```

```
[1] "Extended Naive Bayes Accuracy: 0.4519"
```

Comparing All Models

```
# Create a comparison table
model_comparison <- data.frame(
  Model = c("Logistic Regression", "logistic_model_extended",
            "Logistic Regression (Interaction)",
            "LDA", "LDA (Extended)", "QDA", "QDA (Extended)",
            "KNN (K=1)","KNN (K=3)", "KNN (K=5)", "KNN (K=7)",
            "KNN (K=10)", "KNN (K=15)", "KNN (K=20)",
            "Naive Bayes", "Naive Bayes (Extended)"),
  Accuracy = c(test_accuracy, logistic_accuracy_extended,
               logistic_accuracy_interaction,
               lda_accuracy, lda_accuracy_extended,
```

```
                qda_accuracy, qda_accuracy_extended,
                knn_results[[1]]$accuracy, knn_results[[2]]$accuracy,
                knn_results[[3]]$accuracy, knn_results[[4]]$accuracy,
                knn_results[[5]]$accuracy, knn_results[[6]]$accuracy,
                knn_results[[7]]$accuracy,
                nb_accuracy, nb_accuracy_extended)
)

# Print comparison results
print(model_comparison)
```

```
                              Model  Accuracy
1                 Logistic Regression 0.6250000
2            logistic_model_extended 0.4615385
3  Logistic Regression (Interaction) 0.5384615
4                                 LDA 0.6250000
5                       LDA (Extended) 0.4615385
6                                 QDA 0.5865385
7                       QDA (Extended) 0.4326923
8                            KNN (K=1) 0.5000000
9                            KNN (K=3) 0.5480769
10                           KNN (K=5) 0.5288462
11                           KNN (K=7) 0.5384615
12                          KNN (K=10) 0.5673077
13                          KNN (K=15) 0.5865385
14                          KNN (K=20) 0.5865385
15                         Naive Bayes 0.5865385
16              Naive Bayes (Extended) 0.4519231
```

Bonus question: ISL Exercise 4.8.4 (30pts)

4. When the number of features p is large, there tends to be a deterioration in the performance of KNN and other *local* approaches that perform prediction using only observations that are *near* the test observation for which a prediction must be made. This phenomenon is known as the *curse of dimensionality*, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

curse of dimensionality

   (a) Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on $[0, 1]$. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10 % of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$, we will use observations in the range $[0.55, 0.65]$. On average, what fraction of the available observations will we use to make the prediction?

   (b) Now suppose that we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2. We assume that (X_1, X_2) are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to

predict a test observation's response using only observations that are within 10 % of the range of X_1 *and* within 10 % of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$, we will use observations in the range $[0.55, 0.65]$ for X_1 and in the range $[0.3, 0.4]$ for X_2. On average, what fraction of the available observations will we use to make the prediction?

(c) Now suppose that we have a set of observations on $p = 100$ features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

(d) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

(e) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For $p = 1, 2,$ and 100, what is the length of each side of the hypercube? Comment on your answer.

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment, when $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube.

Solution:

(a) Since the feature X is uniformly distributed in the range$[0, 1]$, We consider a test observation and use only those within 10% of its range.

If a test point is at $X = 0.6$, we use observations in the range: $[0.55, 0.65]$. The total range is 1, so the fraction of data used is$\frac{selected\ range}{total\ range} = \frac{0.65-0.55}{1} = 0.1$. Therefore, we use 10% of the data.

(b) We now have two features, (X_1, X_2), both uniformly distributed on $[0, 1] \times [0, 1]$.

To predict the response, we use only observations within 10% of both X_1 and X_2.

When $X_1 = 0.6,\ X_2 = 0.35$, we will use: $X_1 \in [0.55, 0.65],\ X_2 \in [0.3, 0.4]$.

The fraction of data used is $\frac{selected\ range}{total\ range} = \frac{0.10*0.10}{1} = 0.01$. Therefore, we use 1% of the data.

(c)

Now, we have 100 features, all uniformly distributed in $[0, 1]$. We use only the observations within 10% of each feature's range.

For each feature X_i, we select observations within: $[X_i - 0.05,\ X_i + 0.05]$ (assuming the test observation is not too close to the boundaries).

Since the features are independent, the fraction of observations in each dimension is 0.1. The fraction of data used is $(0.1)^{100}$, which is very small number.

Therefore, in 100D space, almost none of the training data is "close" to the test point, making KNN ineffective.

(d)

From previous results, in 1D, we use 10% of the data; In 2D, we use 1%; In 100D, we use $(0.1)^{100}$, which is practically zero.

Because most points are far away, and distances between them are almost uniform, and even with millions of points, find "near" neighbors is unlikely. What's more, searching for neighbors in high-dimensional space is inefficient.

Therefore, KNN performs poorly in high dimensions because there are too few nearby training points, making predictions unreliable.

(e)

We define a p-dimensional hypercube centered at a test point that contains 10% of the total training data.

Volume of the hypercube in $p-dimensional$ space is: $Fraction\ of\ Data\ Used = (Side\ Length)^p$

Let s be the side length of the hypercube. To contain 10% of data, we solve: $s^p = 0.1$.

For $p = 1$: $s^1 = 0.1$, $s = 0.1$. In 1D, we take 10% of the range.

For $p = 2$: $s^2 = 0.1$, $s = \sqrt{0.1} \approx 0.316$. In 2D, the square has side length around 0.316, much larger than in 1D.

For $p = 100$: $s^{100} = 0.1$, $s = 0.1^{\frac{1}{100}} = 10^{-0.01} \approx 0.977$. In 100D, the side length is about 0.977, meaning almost the entire space is included.

In high dimensions, the hypercube must be almost the entire space to contain just 10% of the data. This confirms why KNN is ineffective in high dimensions: Everything is "far" apart, so the notion of "nearest neighbors" breaks down.

Thus, KNN works well in low dimensions because neighbors are meaningful. In high dimensions, the space becomes too sparse, making KNN ineffective.