2020

DataMining

HW3

192STG11 우나영

1. Description

이번 시간에는 classification 기법으로 대표되는 logistic regression, LDA, 그리고 QDA 를 배웠다. ISL(II)의 4.5 를 읽고 3 가지 classification 기법과 더불어 chapter 2 에서 배운 KNN(K-Nearest-Neighbor)을 비교해 볼 것이다. 다음으로는 ISL(II) 4.6 의 lab 을 통해 이러한 classification 기법들을 R 에서 구현해볼 것이다. 마지막으로는 앞에서 공부했던 classification 기법의 이론과 실습을 바탕으로 예제 4, 10, 그리고 11 번을 풀어볼 것이다.

2. Implementation

Chapter 4 에서 배운 logistic regression, LDA 그리고 QDA 그리고 Chapter 2 에서 배운 KNN 의 개념을 정리한 후 6 가지 가정 아래 생성된 데이터로 각 기법들의 성능을 비교해볼 것이다. LDA 과 Logistic regression 은 linear decision boundaries 를 갖는다는 공통점을 갖는다. 예를 들어, 두개의 class 와 feature 의 개수인 p 가 1 인 data 를 고려해보자. $p_k(x) = Pr(Y = k|X = x)\, where\ k = 1, 2$라 할 때, LDA 의 log odds 는 (4.24)이고 Logistic regression 의 log odds (4.25)와 같다.

$$log\left(\frac{p_1(x)}{1-p_1(x)}\right) = log\left(\frac{p_1(x)}{p_2(x)}\right) = c_0 + c_1 x\ where\ c_0\ and\ c_1\ are\ functions\ of\ \mu_1, \mu_2\ and\ \sigma^2 \quad (4.24)$$

$$\log\left(\frac{p_1}{1-p_1}\right) = \beta_0 + \beta_1 x \quad (4.25)$$

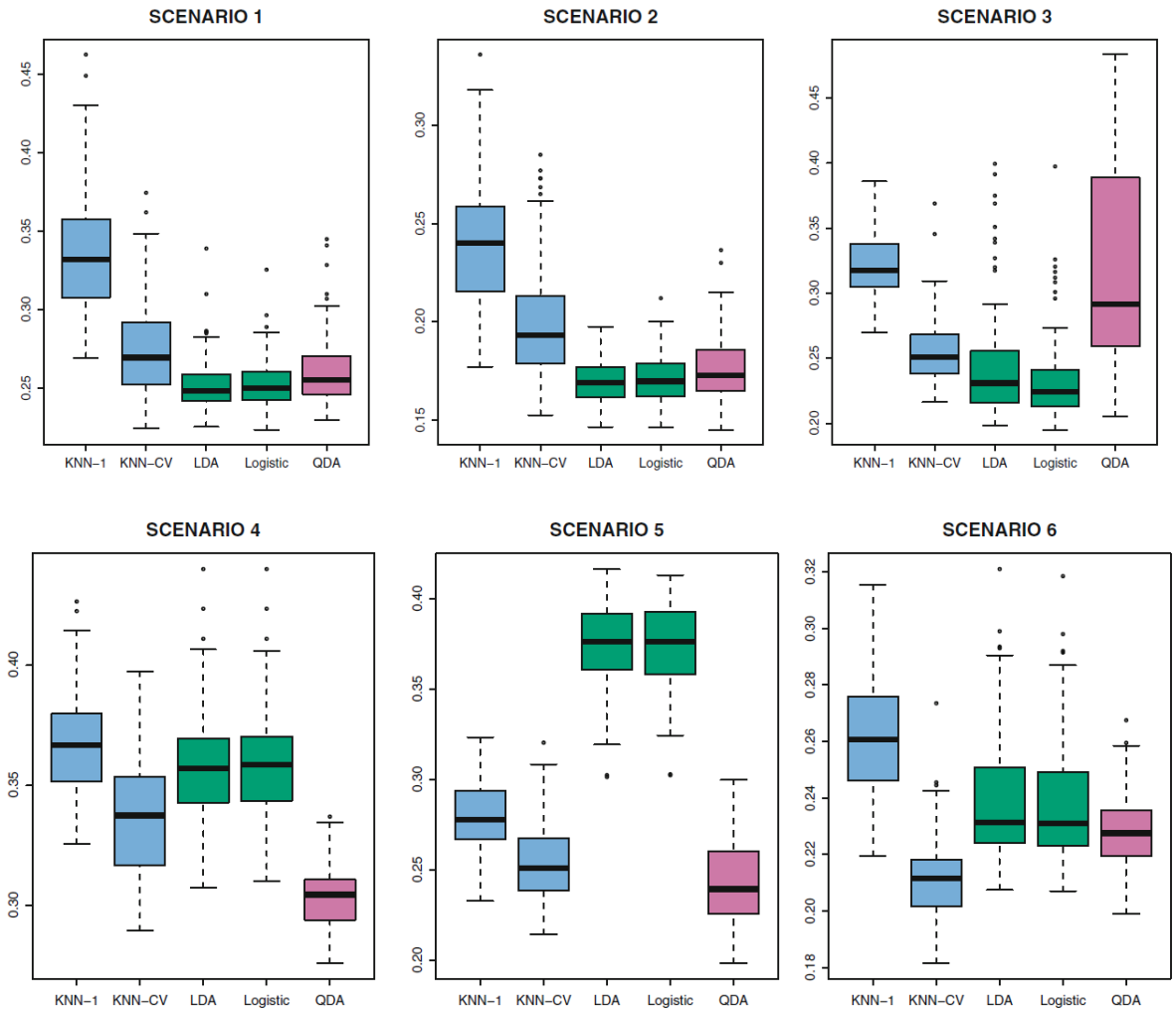
(4.24)와 (4.25) 모두 x의 linear function 으로 LDA 와 logistic regression 모두 linear decision boundary 임을 알 수 있다. 이러한 관계는 p>1 일 때도 성립한다. 두 기법의 차이점은 parameter 추정 방법이다. LDA 는 c_0 와 c_1 이 정규 분포에서 추정된 mean 과 variance 를 이용하지만 logistic regression 은 β_0 와 β_1 을 MLE(Maximum Likelihood Estimation)으로 구한다. 또한 LDA 와 logistic regression 은 대체로 비슷한 성능을 보이지만, 데이터가 모든 class 들이 공통분산(common covariance)을 갖는 Gaussian 분포로부터 생성되었다는 LDA 의 가정과 맞지 않을 때는 logistic regression 이 좀 더 뛰어난 성능을 보여주기도 한다.

Chapter 2 에서 다루었던 KNN(K-Nearest-Neighbor) 은 대표적인 non parametric classification 기법으로 $X = x$ 일 때, class 를 예측하기 위해서 x 근처 K 개의 training observation 들을 바탕으로 class 를 voting 한다. KNN 은 non-linear decision boundary 에서 앞서 살펴보았던 LDA 와 logistic regression 보다 높은 성능을 보일 것이다. 그러나 KNN 은 feature importance 를 알 수 없다. 따라서, QDA 는 non parametric 인 KNN 과 linear decision

boundary 를 갖는 LDA 그리고 logistic regression 의 타협점이 될 것이다. QDA 는 LDA 와 달리 데이터가 각각의 class 마다 covariance matrix 가 다른 Gaussian 분포로부터 생성되었음을 가정한다. LDA 와 QDA 의 가정을 수식으로 표현하면 아래와 같다.

$$LDA : X \sim N(\mu_k, \Sigma)$$

$$QDA : X \sim N(\mu_k, \Sigma_k) \; where \; \Sigma_k \; is \; a \; covariance \; matrix \; for \; the \; kth \; class$$



각각의 classification 의 특징 및 성능을 비교하기 위해 6 가지 가정 아래 만들어진 데이터로 실험을 해보았다. 각 데이터들은 training 과 test 셋으로 나뉘어 training 으로 classification 모델을 피팅하고 test set 으로 test error 를 계산하였다. 이때, KNN 은 K=1 인 경우와 cross validation 으로 K 를 구한 경우 두 가지 방법 모두 사용하였다. 또한 데이터는 feature 의 개수 p = 2 이다.

Scenario 1 은 두 개의 class 가 각각 uncorrelated 하며 normal 분포를 따르는 20 개의 training observation 을 갖는다. 이는 LDA 의 가정을 만족하기 때문에 Scenario 1 번

그래프에서 볼 수 있듯이 LDA 가 가장 성능이 뛰어나다. Logistic regression 역시 LDA 와 같은 linear decision boundary 을 가지므로 비슷한 성능을 보인다. 반면에 KNN 은 bias 를 줄이는 대신 variance 가 높기 때문에 성능이 매우 떨어진다. QDA 또한 linear decision boundary 가 true 인 데이터에서 높은 성능을 보여주지 못한다. Scenario 2 는 각각의 class 의 feature 들이 -0.5 의 correlation 을 갖는다는 점을 제외하고 scenario 1 과 똑 같은 데이터이다. 바뀐 데이터셋에서 각 기법들의 눈에 띄는 성능 변화는 없었다. Scenario 3 의 데이터는 각각의 class 마다 t 분포를 따르는 50 개의 observation 으로 이루어졌다. 이 경우에도 decision boundary 는 linear 이지만 LDA 의 가정을 만족하지 못하기 때문에 logistic regression 이 LDA 보다 성능이 더 높다. 그러나 데이터의 decision boundary 가 linear 이기 때문에 linear discriminant 를 갖는 LDA 도 다른 3 가지 QDA, KNN-1 그리고 KNN-CV 보다 높은 성능을 보인다.

Scenario 4 은 데이터가 normal 분포를 따른다. 이때, 첫 번째 class 는 feature 들끼리 correlation 이 0.5 인 반면 두 번째 class 는 feature 들끼리 correlation 이 -0.5 이다. 이러한 데이터 set up 은 QDA 가정을 만족하기 때문에 그래프에서 볼 수 있듯이 QDA 가 압도적으로 좋은 성능을 보인다. Scenario 5 의 데이터에서 각각의 class 별 observation 은 uncorrelated 한 predictor 를 갖는 normal 분포로부터 생성되었다. 하지만 response variable 은 X_1^2, X_2^2 그리고 $X_1 \times X_2$을 predictor 로 갖는 logistic function 으로부터 생성되었다. 이는 quadratic decision boundary 를 만족시키므로 QDA 가 가장 높은 성능을 보이며 KNN-1 과 KNN-CV 가 뒤따른다. 반면 linear 기법들은 매우 낮은 성능을 보인다. 마지막으로 Scenario 6 은 Scenario 5 의 response variable 을 제외하고는 모든 조건이 같다. 6 의 response 는 5 보다 복잡한 non linear function 으로부터 생성되었다. 그 결과, 가장 성능이 좋은 모델은 가장 flexible 한 KNN-CV 였으며 QDA 는 linear 기법보다 성능이 약간 더 좋았다. 하지만 주목할 사실은 KNN-1 이 가장 성능이 안 좋았다는 것이다. 이는 KNN 의 성능을 높이기 위해서 smoothness 을 조절하는 K 의 선택이 중요하다는 것을 반증한다.

위의 6 가지 실험을 통해 모든 데이터 set up 에서 압도적인 성능을 보이는 모델은 존재하지 않는다는 것을 배웠다. true decision boundary 가 linear 일 때는 linear method 인 logistic regression 과 LDA 가 가장 성능이 좋은 반면 decision boundary 가 non linear 일 때는 QDA 가정을 만족하는 경우에 QDA 가 가장 성능이 좋았고 그보다 복잡한 decision boundary 일 때는 KNN 과 같은 non parametric 모델이 가장 우수했다. 이때, KNN 의 성능을 높이기 위해서는 smoothness 를 결정하는 K 를 적절히 선택해야한다.

Chapter 3 에서 feature 와 response variable 이 non linear 인 경우 feature 들의 transformation 을 통해 regression model set up 을 할 수 있다고 배웠다. 이와 유사하게 classification 기법을 조절할 수 있다. 예를 들면, linear method 인 LDA 와 logistic regression 의

경우 feature 들의 X^2, X^3 그리고 X^4 또는 interaction term 을 추가함으로써 LDA 에 QDA 를 적용한 flexible 한 모델을 만들 수 있다.

다음으로는 ISL(II)의 4.6 Lab 코드를 수행하며 앞에서 배운 classification 기법을 R 로 구현하는 것이다. ISLR 패키지에 내장된 Smarket 데이터와 Caravan 데이터를 LDA, QDA, logistic regression 그리고 KNN 까지 다양한 classification 모델로 적합하고 예측하는 것을 배울 수 있었다. 이러한 실습 내용을 예제 4, 10, 11 번에 적용해 볼 것이다.

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.

(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 observations. 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 use to make the prediction?

만약 X 가 [0,1]에 uniform 하게 분포하는 확률변수라면, $\mathrm{P(X \le x) = F(x) = x \text{ where } 0 \le x \le 1}$. 따라서, $\mathrm{P(X \le x)} = \int_{x_i-0.05}^{x_i+0.05} f(x)dx = F(x_1 - 0.05) - F(x_1 + 0.05) = (x_i + 0.05) - (x_i - 0.05) = .1$ 또는 10% 이기 때문에 어떤 관측치 x_i 가 $[\, x_1 - 0.05, x_1 + 0.05]$ 에 있을 확률은 10%이다. 이때 $\mathrm{x} \in [0.05, 0.95]$을 가정한다.

(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?

(a)와 유사하게 만약 $X_1, and \; X_2$ 이 uniform 하게 분포하는 확률변수라면, 어떤 확률변수가 X_1 또는 X_2의 10% 근처에 속할 확률은 10%이다. 만약 X_1과 X_2가 독립이고 $\mathrm{x} \in X_1$ 그리고 $\mathrm{y} \in X_2$ 그리고 A 와 B 가 다음과 같은 X_1 그리고 X_2 의 이웃이라면 $A \coloneqq [x -$

$0.05, x + 0.05], B \coloneqq [y - 0.05, y + 0.05]$, $\Pr(\mathrm{x} \in \mathrm{A}, \mathrm{y} \in \mathrm{B}) = \Pr(\mathrm{x} \in \mathrm{A}) \Pr(\mathrm{x} \in \mathrm{B}) = 0.01 \text{ or } 1\%$. 따라서, 평균적으로 X_1 와 X_2 10% 근방에 모두 속하는 관측치는 오로지 1%에 해당된다.

(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?

(a)~(b)의 풀이를 그대로 적용하여 0.1^{100}이다.

(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.

KNN 은 curse of dimensionality 라는 가장 큰 단점을 갖고 있다. 이는 데이터의 feature 개수가 증가함에 따라 데이터의 사이가 멀어져 sparsity 현상이 나타나는 것을 말한다. 앞서 본 (a)-(c)는 이를 잘 보여준다. p=1 일 때는 test observation 과 10% 근방의 데이터가 전체 데이터의 0.1 이지만 p=2 일 때는 0.01 이며 p=100 일 때는 0.1^{100} 으로 기하급수적으로 감소한다. 따라서, KNN 은 데이터의 feature 의 수가 늘어남에 따라 test observation 의 response 를 예측하기 위해 거리가 매우 먼 training observation 을 사용하게 되고 그 결과 매우 낮은 성능을 갖게 된다.

(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?

Let the length be x then $x^p = 0.1$, which means $x = (0.1)^{1/p}$

p	$the\ length\ of\ each\ side\ x$
1	0.1
2	$0.1^{1/2} = 0.3162$
100	$0.1^{1/100} = 0.9772$

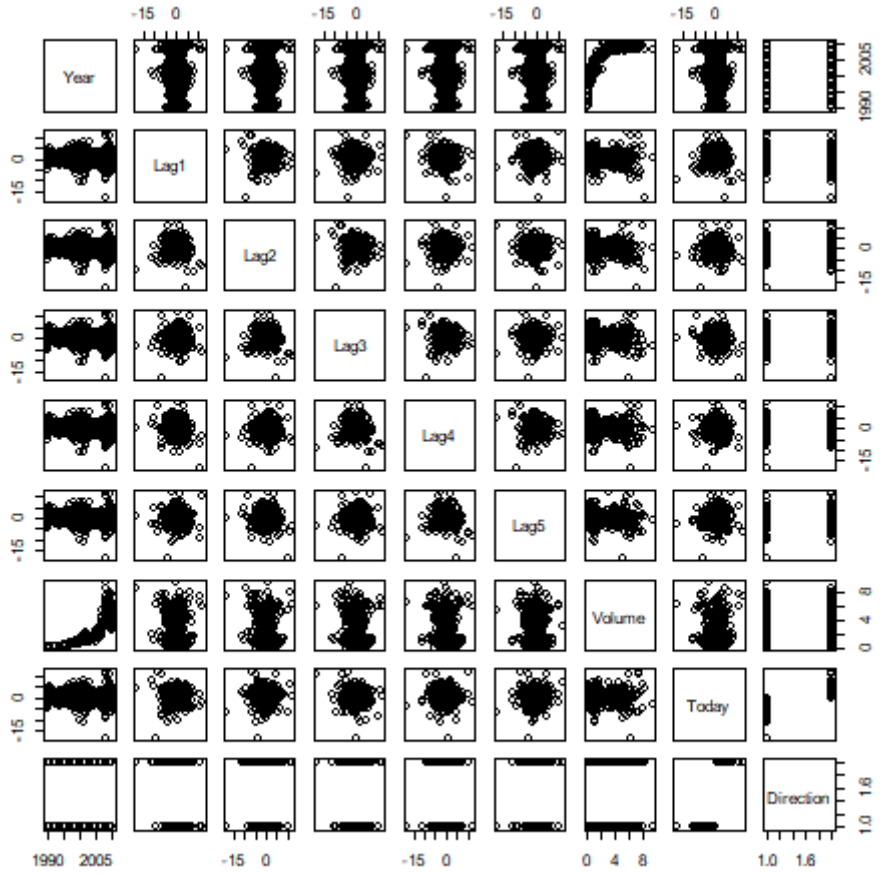
test observation 의 10% 근방의 training observation 을 포함하는 hypercube 의 한 변의 길이 x는 p 가 커짐에 따라 기하급수적으로 커진다. p=1 일 때 길이가 0.1 이었다면 p=100 일 때는 0.98 이 된다. 즉, KNN 에서 p 가 커짐에 따라 test observation 의 근방의 training observation 의 거리가 기하급수적으로 멀어짐을 알 수 있다.

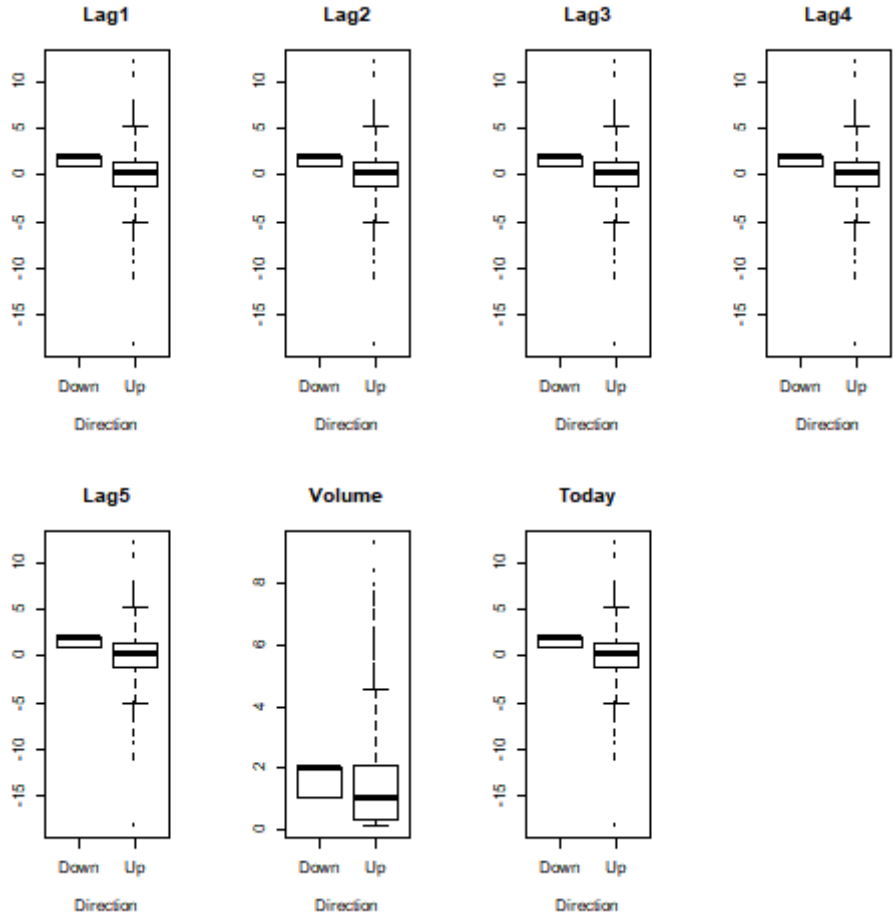
10. This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except

that it contains 1,089 weekly return 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?

	Year	*Lag 1*	*Lag 2*	*Lag 3*	*Lag 4*	*Lag 5*	*Volume*	*Today*	*Direction*
Min	1990	-18.20	-18.20	-18.20	-18.20	-18.20	0.09	-18.20	
Q1	1995	-1.15	-1.15	-1.15	-1.15	-1.15	0.33	-1.15	
Median	2000	0.24	0.24	0.24	0.24	0.24	1.00	0.24	Down : 484
Mean	2000	0.15	0.15	0.15	0.15	0.15	1.57	0.15	Up : 605
Q3	2005	1.41	1.41	1.41	1.41	1.41	2.05	1.41	
Max	2010	12.03	12.03	12.03	12.03	12.03	9.33	12.03	





통계량, paired scatter plot 그리고 boxplot 을 살펴보았을 때, year 가 커짐에 따라 volume 이 급격히 커짐을 알 수 있었다.

(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?

Predictors	*Estimate*	*Std. Error*	*z value*	*Pr(|z|)*
(intercept)	0.267	0.086	3.106	0.002
Lag1	-0.041	0.026	-1.563	0.118
Lag2	0.058	0.027	2.175	0.030
Lag3	-0.016	0.027	-0.602	0.547
Lag4	-0.028	0.026	-1.050	0.294
Lag5	-0.014	0.026	-0.549	0.583
Volume	-0.023	0.037	-0.616	0.538

유의수준 $\alpha = 0.05$이라 할 때, 통계적으로 유의미한 feature 는 p-value 가 0.05 보다 작은 (intercept)와 Lag2 이다.

(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.

Predicted/Observed	*Down*	*Up*
Down	54	48
Up	430	557

Confusion matrix 는 실제 관측치와 모델의 예측치를 한 눈에 파악할 수 있도록 요약한 표이다. cut off 는 0.5 이다. accuracy = $\frac{54+557}{1089} = 0.5611$이며 모델이 Up 으로 예측한 것 중 실제 Up 인 비율은 $\frac{557}{(430+557)} = 0.5643$이다.

(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).

cut off : 0.5

Predicted/Observed	*Down*	*Up*
Down	9	5
Up	34	56

$\text{accuracy} = 0.625$

(e) LDA

cut off : 0.5

Predicted/Observed	*Down*	*Up*
Down	9	5
Up	34	56

$\text{accuracy} = 0.625$

(f) QDA

cut off : 0.5

Predicted/Observed	*Down*	*Up*
Down	0	0
Up	43	61

$\text{accuracy} = 0.5865$

(g) KNN with K = 1

cut off : 0.5

Predicted/Observed	*Down*	*Up*
Down	21	30
Up	22	31

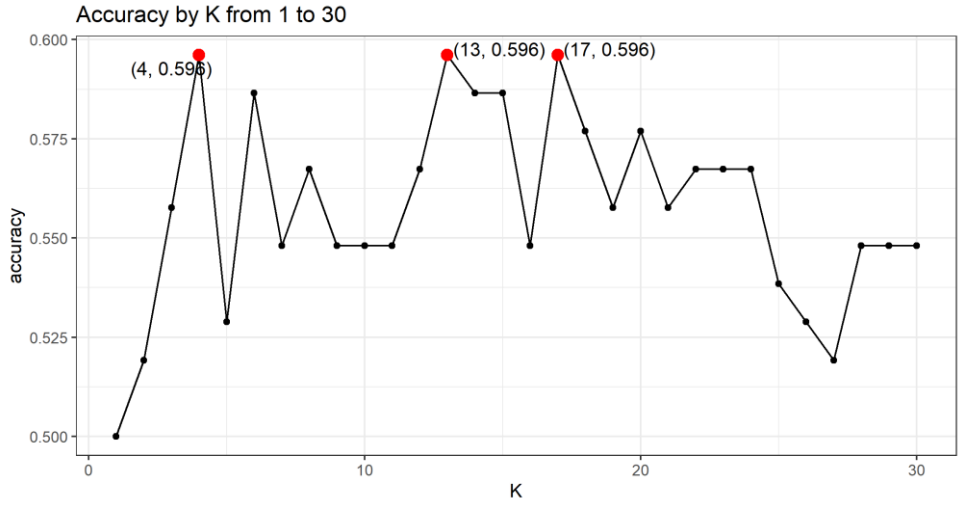
$$\text{accuracy} = 0.5$$

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

LDA 와 logistic regression 이 accuracy 0.625 로 가장 우수한 성능을 보여준다.

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

LDA 와 logistic regression 의 다양한 실험을 해보았지만, 결과적으로 앞서 (d)와 (e)번에서 다룬 Lag2 변수를 단독으로 transformation 없이 수행한 것이 가장 성능이 좋았다. 한편, KNN 의 최적의 K 값을 다음과 같이 구해 그래프로 표현해보았다. 그 결과 가장 좋은 성능을 보이는 K 값은 4, 13, 17 이었다. 그러나 K 값이 optimal 일 때도 accuracy 가 0.59 로 LDA 와 logistic regression 의 accuracy 인 0.625 를 넘어서지 못했다.



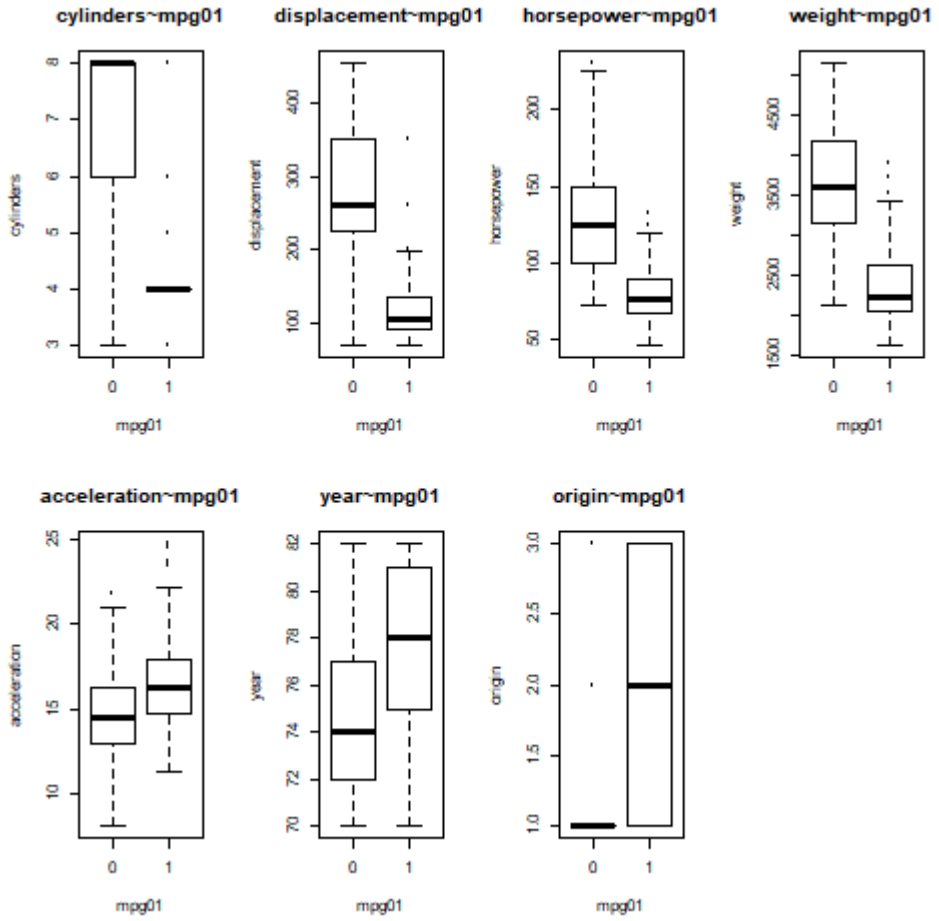
11. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

(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.

Value	0	1
# of observations	196	196

(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.



boxplot 을 살펴보았을 때, displacement, horsepower, weight 그리고 cylinder 이 mpg01 을 예측함에 있어 유의할 것으로 파악된다.

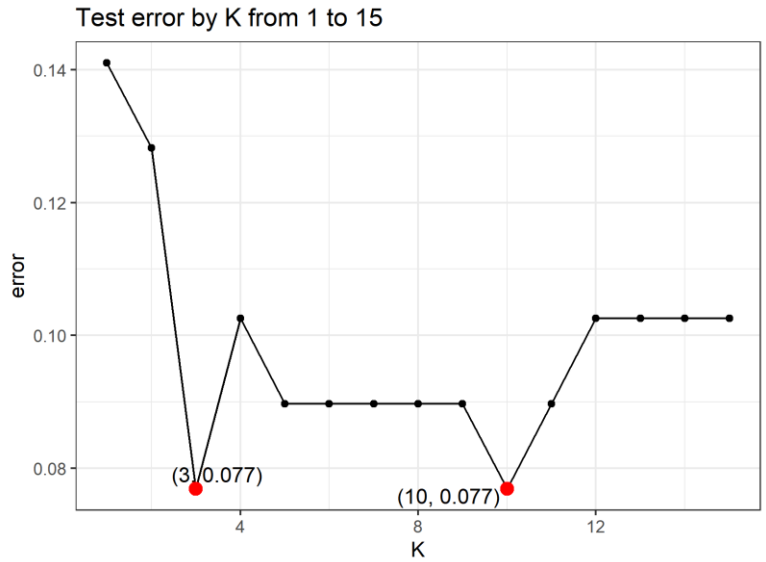
(c) Split the data into a training set and a test set.

Train 과 test 를 8:2 비율로 나누었다.

Value of mpg01		0	1	*Total*
Number of obs	training	161	153	314
	test	35	43	78

(d)~(g) Perform LDA/QDA/logistic regression/KNN 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?

Method	*Test Error*
LDA	0.1026
QDA	0.1154
Logistic Regression(cut off = 0.5)	0.0897
KNN when K=3, 10	0.077



displacement, horsepower, weight 그리고 cylinder 를 이용하여 mpg01 을 예측하는 가장 성능이 좋은 모델은 예측 오차가 7%로 가장 낮은 K=3 또는 10 일때의 KNN 이며 가장 성능이 나쁜 모델은 예측 오차가 11%인 QDA 이다.

3. Discussion

이번 과제를 통해 Classification 의 기법으로 대표되는 logistic regression, LDA, QDA 그리고 KNN 의 특징을 배울 수 있었다. Logistic regression 과 LDA 는 parameter 추정 방법에는 차이가 있으나 linear decision boundary 를 사용한다는 공통점이 있다. LDA 는

데이터가 공통분산을 갖는 Gaussian distribution 을 따른다는 가정을 하기 때문에 이 가정을 따르지 않는 linear decision boundary 데이터의 경우 logistic regression 이 성능이 좀 더 뛰어나다. QDA 는 데이터가 class 마다 각자의 covariance matrix 를 갖는 Gaussian distribution 을 가정하는 quadratic decision boundary 방법이다. 하지만 true decision boundary 가 복잡한 non linear 인 경우에는 non parametric 인 KNN 을 사용해주는 것이 좋다. 이때, KNN 은 성능을 최대화하기 위해 smoothness 를 control 하는 K 의 선택이 중요하다. 다음으로는 6 가지 가정 하에 생성된 데이터들을 이러한 classification 기법으로 피팅 후 test error 를 계산하여 성능을 비교해보았다. 그 결과 4 가지 서로 다른 classification method 의 특징들을 명확하게 살펴볼 수 있었다. ISL(II)의 Lab 4.6 은 4 가지 classification 기법의 R 실행 코드를 소개한다. 이를 바탕으로 ISL(II) 예제 4, 10, 11 을 풀 수 있었다. 예제 4 는 curse of dimensionality 를 다룬다. 이는 KNN 의 대표적인 단점으로 데이터의 feature 개수 p 가 증가함에 따라 데이터의 sparsity 가 커져 test observation 의 근방의 데이터들이 멀어지는 현상을 말한다. 예제 10 과 11 에서 패키지 ISLR 에 내장된 데이터들을 4 가지 classification 기법으로 모델링 및 예측하고 결과를 비교 분석했다. 위의 과정을 통해 4 가지 classification 기법의 특징에 대해 깊이 이해할 수 있었다.

4. Appendix

```
#---- 4.6 Lab : Logistic Regression, LDA, QDA, and KNN ----#

## 4.6.1 The Stock Market Data

library(ISLR)

names(Smarket)

dim(Smarket)

summary(Smarket)

pairs(Smarket)

cor(Smarket[,-9])

attach(Smarket)

plot(Volume)


## 4.6.2 Logistic Regression

glm.fits = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = binomial, data = Smarket)

summary(glm.fits)

coef(glm.fits)

summary(glm.fits)$coef

glm.probs = predict(glm.fits, type = "response")

glm.probs[1:10]

contrasts(Direction)

glm.pred = rep("Down", 1250)

glm.pred[glm.probs > .5] = "Up"

table(glm.pred, Direction)

(507+145)/1250

mean(glm.pred == Direction)

train = (Year < 2005)

Smarket.2005 = Smarket[!train, ]

dim(Smarket.2005)

Direction.2005 = Direction[!train]

glm.fits = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Smarket, family = binomial, subset = train)

glm.probs = predict(glm.fits, Smarket.2005, type = "response")

glm.pred = rep("Down", 252)

glm.pred[glm.probs > 0.5] = "Up"

mean(glm.pred == Direction.2005)

table(glm.pred, Direction.2005)

mean(glm.pred != Direction.2005)

glm.fits = glm(Direction ~ Lag1 + Lag2, data = Smarket, family = binomial, subset = train)

glm.probs = predict(glm.fits, Smarket.2005, type = "response")

glm.pred = rep("Down", 252)

glm.pred[glm.probs>.5] = "Up"

table(glm.pred, Direction.2005)

mean(glm.pred == Direction.2005)

106/(106+76)

predict(glm.fits, newdata = data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)), type = "response")


## Linear Discriminant Analysis

library(MASS)

lda.fit = lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

lda.fit

lda.pred = predict(lda.fit, Smarket.2005)

names(lda.pred)

lda.class = lda.pred$class

table(lda.class, Direction.2005)

mean(lda.class == Direction.2005)

sum(lda.pred$posterior[,1]>=.5)

sum(lda.pred$posterior[,1]<.5)

lda.pred$posterior[1:20, 1]

lda.class[1:20]

sum(lda.pred$posterior[,1]>.9)


## Quadratic Discriminant Analysis

qda.fit = qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

qda.fit

qda.class = predict(qda.fit, Smarket.2005)$class

table(qda.class, Direction.2005)

mean(qda.class == Direction.2005)


## 4.6.5 K-Nearest Neighbors

library(class)

train.X = cbind(Lag1, Lag2)[train,]

test.X = cbind(Lag1, Lag2)[!train,]

train.Direction = Direction[train]

set.seed(1)

knn.pred = knn(train.X, test.X, train.Direction, k = 1)

table(knn.pred, Direction.2005)

(83+43)/252

knn.pred = knn(train.X, test.X, train.Direction, k = 3)

table(knn.pred, Direction.2005)

mean(knn.pred == Direction.2005)


## 4.6.6 An Application to Caravan Insurance Data
```

```
dim(Caravan)

attach(Caravan)

summary(Purchase)

348/5822

standardized.X = scale(Caravan[, -86])

var(Caravan[,1])

var(Caravan[,2])

var(standardized.X[,1])

var(standardized.X[,2])

test = 1:1000

train.X = standardized.X[-test,]

test.X = standardized.X[test,]

train.Y = Purchase[-test]

test.Y = Purchase[test]

set.seed(1)

knn.pred = knn(train.X, test.X, train.Y,
k=1)

mean(test.Y!=knn.pred)

mean(test.Y!="No")

table(knn.pred, test.Y)

9/(68+9)

knn.pred = knn(train.X, test.X, train.Y, k =
3)

table(knn.pred, test.Y)

5/26

knn.pred = knn(train.X, test.X, train.Y, k =
5)

table(knn.pred, test.Y)

4/15

glm.fits = glm(Purchase ~., data = Caravan,
family=binomial, subset = -test)

glm.probs = predict(glm.fits,
Caravan[test, ], type = "response")

glm.pred = rep("No", 1000)

glm.pred[glm.probs > .5] = "Yes"

table(glm.pred, test.Y)

glm.pred = rep("No", 1000)

glm.pred[glm.probs>0.25] = "Yes"

table(glm.pred, test.Y)

11/(22+11)


#--- Exercise 4, 10, 11 ---#

## Exercise 10


## 10-a

names(Weekly)

summary(Weekly)

png('weekly.png')

plot(Weekly)

dev.off()

sum.week <- summary(Weekly)

write.csv(sum.week, 'weekly.csv')

attach(Weekly)

png('boxplot.png')

par(mfrow=c(2,4))

boxplot(Direction, Lag1, main = "Lag1", xlab
= "Direction",  names=c("Down","Up"))

boxplot(Direction, Lag2, main = "Lag2", xlab
= "Direction",  names=c("Down","Up"))

boxplot(Direction, Lag3, main = "Lag3", xlab
= "Direction",  names=c("Down","Up"))

boxplot(Direction, Lag4, main = "Lag4", xlab
= "Direction",  names=c("Down","Up"))

boxplot(Direction, Lag5, main = "Lag5", xlab
= "Direction",  names=c("Down","Up"))

boxplot(Direction, Volume, main = "Volume",
xlab = "Direction",  names=c("Down","Up"))

boxplot(Direction, Today, main = "Today",
xlab = "Direction",  names=c("Down","Up"))

dev.off()


## 10-b

glm.fit.weekly <- glm(Direction ~ Lag1 +
Lag2+ Lag3 + Lag4+ Lag5+ Volume, data =
Weekly, family = binomial)

summary(glm.fit.weekly)

sum.glm <- summary(glm.fit.weekly)$coef

write.csv(sum.glm, 'glm.csv', row.names =
FALSE)


## 10-c

glm.probs = predict(glm.fit.weekly, type =
"response")

glm.pred = rep("Down", nrow(Weekly))

glm.pred[glm.probs > 0.5] = "Up"

table(glm.pred, Direction)

557/(430+557)

mean(glm.pred == Direction)


## 10-d

training = (Year<2009)

train_data = Weekly[training,]

test_data = Weekly[!training,]

glm.train.fit <- glm(Direction~Lag2, data =
train_data, family = binomial)

glm.probs <- predict(glm.train.fit,
test_data, type = "response")

glm.pred <- rep("Down", nrow(test_data))
```

```
glm.pred[glm.probs > 0.5] <- "Up"

table(glm.pred, test_data$Direction)

mean(glm.pred == test_data$Direction)


## 10-e

lda.fit = lda(Direction ~ Lag2, data =
train_data)

lda.fit

lda.pred = predict(lda.fit, test_data)

lda.class = lda.pred$class

table(lda.class, test_data$Direction)

mean(lda.class == test_data$Direction)


## 10-f

qda.fit = qda(Direction ~ Lag2, data =
train_data)

qda.fit

qda.class = predict(qda.fit,
test_data)$class

table(qda.class, test_data$Direction)

mean(qda.class == test_data$Direction)


## 10-g

train <- (Year < 2009)

train.X = as.matrix(Lag2[train])

test.X = as.matrix(Lag2[!train])

train.Direction = Direction[train]

test.Direction = Direction[!train]

set.seed(1)

knn.pred = knn(train.X, test.X,
train.Direction, k = 1)

table(knn.pred, test.Direction)

mean(knn.pred == test.Direction)


## 10-i

set.seed(1886)

result <- c()

for(i in 1:30){

  knn.pred <- knn(train.X, test.X,
train.Direction, k = i)

  result[i] <- mean(knn.pred ==
test.Direction)

}

df <- data.frame(K = 1:30, accuracy =
result)

best <- which(df$accuracy ==
max(df$accuracy))

library(ggrepel)

df %>% ggplot(aes(K, accuracy)) +
geom_line() + geom_point() + theme_bw() +
labs(title = "Accuracy by K from 1 to 30")+

  geom_text_repel(aes(label = paste0("(", K,
", ", round(accuracy, 3), ")")) , data =
df[best, ]) + geom_point(data=df[best,],
size = 3, col = "red")

ggsave('best.png')


## 11-a

attach(Auto)

Auto$mpg01 <- as.factor(ifelse(mpg >
median(mpg), 1, 0))

table(Auto$mpg01)


## 11-b

summary(Auto)

png('scattter.png')

plot(Auto[,-c(9,10)])

dev.off()

png('boxplotagina.png')

par(mfrow=c(2,4))

boxplot(cylinders~mpg01,main="cylinders~mpg0
1")

boxplot(displacement~mpg01,main="displacemen
t~mpg01")

boxplot(horsepower~mpg01,main="horsepower~mp
g01")

boxplot(weight~mpg01,main="weight~mpg01")

boxplot(acceleration~mpg01,main="acceleratio
n~mpg01")

boxplot(year~mpg01,main="year~mpg01")

boxplot(origin~mpg01,main="origin~mpg01")

dev.off()


## 11-(c)

set.seed(1886)

train <- sample(1:nrow(Auto),
round(nrow(Auto)*0.8,0), replace = FALSE)

training <- Auto[train, ]

test <- Auto[-train,]

table(training$mpg01)

table(test$mpg01)


## 11-(d)

lda.fit = lda(mpg01 ~
displacement+horsepower+weight+cylinders,
data = training)

lda.fit

lda.pred = predict(lda.fit, test)

lda.class = lda.pred$class
```

```
table(lda.class, test$mpg01)

mean(lda.class != test$mpg01)


## 11-(e)

qda.fit = qda(mpg01 ~
displacement+horsepower+weight+cylinders,
data = training)

qda.pred = predict(qda.fit, test)

qda.class = qda.pred$class

mean(qda.class != test$mpg01)


## 11-(f)

glm.train.fit <- glm(mpg01 ~
displacement+horsepower+weight+cylinders,
data = training, family = binomial)

glm.probs <- predict(glm.train.fit, test,
type = "response")

glm.pred <- rep("0", nrow(test))

glm.pred[glm.probs > 0.5] <- "1"

mean(glm.pred != test$mpg01)


## 11-(g)

standardized.X = scale(Auto[, -c(9,10)])

train.X = standardized.X[train,c(2,3,4,5)]

test.X = standardized.X[-train,c(2,3,4,5)]

train.mpg01 = mpg01[train]

test.mpg01 = mpg01[-train]


set.seed(1886)

result <- c()

for(i in 1:15){

  knn.pred <- knn(train.X, test.X,
train.mpg01, k = i)

  result[i] <- mean(knn.pred != test.mpg01)

}

df <- data.frame(K = 1:15, error = result)

best <- which(df$error == min(df$error))

df %>% ggplot(aes(K, error)) + geom_line() +
geom_point() + theme_bw() + labs(title =
"Test error by K from 1 to 15") +

  geom_text_repel(aes(label = paste0("(", K,
", ", round(error, 3), ")")) , data =
df[best, ]) + geom_point(data=df[best,],
size = 3, col = "red")

ggsave('best2.png')
```