Data Mining

HW#4

학번	182STG18
이름	이하경
제출일	2019.04.10

Description

Methods of Classification

반응변수가 범주형일 때, 분류 모형이 하는 일은 하나 이상의 feature 들로 이루어진 X 를 이용해 Y 의 범주를 예측하는 Classifier C(X)를 찾는 것이라고 할 수 있다. 특히 class 자체에 대한 예측 뿐만 아니라 X 가 각 class 에 속할 확률을 추정하는 것이 유용할 수 있다.

1) Logistic Regression

Logistic Regression 은 범주형 반응변수 Y 와 설명변수 X 들의 선형결합을 Link Function 으로 연결한다. Y 가 0 과 1 로 이루어진 Binary Variable 이라고 가정할 때, $p(X) = P(Y = 1|X)$으로 다음과 같이 logit transformation 을 할 수 있고 이 때 p(X)는 항상 0 과 1 사이의 값을 가진다.

$$\frac{P(Y = 1|X)}{P(Y = 0|X)} = \log\left(\frac{p(X)}{1 - p(X)}\right) = XB, \qquad \rightarrow p(X) = \frac{\exp\{XB\}}{1 + \exp\{XB\}}$$

만약 반응변수의 class 가 3 개 이상일 경우, baseline class K 를 기준으로 K-1 개의 각 class 에 대해 linear combination 을 추정한다. Multiclass Logistic Regression 에서는 baseline 과는 비교가 가능하지만 서로 다른 class 끼리는 비교 해석할 수 없으며 class 가 많아질수록 모형이 불안정해진다.

2) LDA & QDA (Discriminant Analysis)

개별 class 에 속한 X 들의 분포가 class 간에 서로 다르다고 할 때, 판별분석은 이 분포를 정규분포로 가정하고 다음과 같이 Bayes Theorem 을 이용한다. 여기서 $f_k(x)$가 class k 에 속한 X 들의 density 이며, π_k는 class k 의 prior 라고 할 수 있다.

$$P(Y = k|X = x) = \frac{P(X = x|Y = k) \cdot P(Y = k)}{P(X = k)} = \frac{f_k(x) \cdot \pi_k}{P(X = k)}$$

판별 분석에는 판별함수를 X 와 파라미터들의 1 차 선형결합의 형태로 만드는 LDA 와 2 차 비선형 결합으로 만드는 QDA 가 있다. LDA 는 모든 class 별 정규분포의 분산이 모두 동일하다고 가정한다. 따라서 추정 파라미터의 개수는 $\mu = (\mu_1, \ldots, \mu_k)$와 σ^2으로 k+1 개이다. QDA 는 class 별 정규분포의 분산을 서로 다르다고 가정하며 이 때 추정 파라미터는 k 개의 평균과 분산, k(k-1)/2 개의 공분산으로 매우 많다.

** class 가 2 가지인 경우 Logistic 과 LDA 는 추정식의 형태가 동일하며 각 파라미터에 대한 추정 방법에 차이가 있지만 보통 서로 매우 유사한 결과를 보인다.

- Logistic Regression 은 K=2 일 때 매우 성능이 좋으며, 설명변수들이 범주형 변수를 포함할 경우에도 문제 없이 추정 가능하다.

- sample size n 이 작고 각 class 에 속한 X 의 분포가 정규분포와 유사할 때 LDA 는 Logistic Regression 보다 안정적으로 추정한다. 또한 K>2 인 multiclassification 에서도 좋은 성능을 발휘한다. 그러나 X 가 정규분포를 따른다고 가정하므로 설명변수가 numeric 으로만 이루어졌을 때 사용 가능하다.

3) KNN

HW1 에서 KNN 의 추정 방법과 Curse of Dimensionality 에 대해 논하였다. KNN 은 반응변수가 수치형 또는 범주형일 때 모두 사용할 수 있으며 범주형인 경우 각 관측치 주변 K 개 이웃의 class 빈도를 계산하여 가장 빈도가 높은 쪽으로 분류한다. (voting)

Cross-Validation

모형이 복잡할수록 표본에 포함된 관측치들에만 Overfitting 하는 문제가 발생할 수 있다. 이 때 모형은 기존 표본 이외에 새로운 관측치에 대한 일반성을 잃고 적절한 예측을 하지 못한다. 따라서 주어진 표본이 있을 때 이를 모두 사용하여 모형을 적합하지 않고, Training(또는 Training/Validation), Test Set 으로 표본을 분할하여 Training Sample 들만을 사용해 추정한 뒤 Test Data 에 대한 예측의 정확도를 평가하는 방법을 사용한다.

HW1 에서 확인하였듯이 설명변수가 많을수록 전체 공간에서 표본이 차지하는 공간의 비율이 급격하게 줄어든다. 이것은 차원이 커질수록 새로운 Test 관측치에 대해 주변에 존재하는 기존 sample 내의 Training 관측치가 존재하지 않을 가능성이 크다는 것을 의미한다.

HW4 에서는 이러한 Curse of Dimensionity 에 대해 다시 한번 다룬다. 또한 2 개의 Exercise 에서 실제 데이터에 대해 4 가지 방법론을 사용해

분류 모형을 만들고(Training Set) 예측의 정확도 또는 오분류율을 서로 비교(Test Set)한다.

Results

Chapter 4 Lab: Logistic Regression, LDA, QDA, and KNN

Chapter 4의 Lab에서는 Logistic Regression, LDA 및 QDA, KNN 분류 모형(Classification)을 실제 데이터에 직접 적용해보았다.

먼저 2001년부터 2005년까지 S&P 500 주가지수의 일별 수익률(Lag1-Lag5)과 전일 거래량(Volume), 오늘 날짜의 수익률(Today)과 이 날짜의 수익률의 증감(Direction) 변수로 구성된 데이터에서 binary variable인 반응변수 Direction을 예측하기 위해 4가지 방법을 이용해 모형을 적합하고 정확도 및 오분류율을 구해보았다. 데이터 셋을 Train과 Test Set으로 나누고 각 모형 적합에는 train set을, 평가에는 test set을 이용하여 class를 예측하도록 하여 정확도 및 오분류율을 비교하는 과정을 직접 코딩해보았다. 각 모형에서 출력되는 결과가 무엇인지 직접 확인하고 의미가 무엇인지 알 수 있었다. Exercise 4.4 데이터 역시 주가에 대한 데이터로 Smarket 데이터와 거의 비슷하여 Lab이 많은 도움이 되었다.

Caravan Insurance 데이터는 보험사 고객 정보에 대한 데이터로 설명변수가 86개로 매우 많으며, Caravan 보험 가입 여부를 나타내는 반응변수인 Purchase가 Yes인 관측치가 전체의 6% 정도로 전체 데이터에서 Class간 비율이 고르지 않았다. 따라서 정확도를 계산할 때 전체 관측치에 대한 정확도보다 해당 Class에 대한 예측의 정확도를 비교하는 것이 좋다. 또한 Logistic Regression은 확률을 예측하므로 이를 기준으로 class를 구분하기 위한 적절한 threshold을 고려해야 함을 알 수 있었다. 또한 변수의 개수가 많을 때 KNN을 사용할 경우 변수 별 Scale이 달라 Scale이 큰 변수를 포함하여 KNN을 사용하면 관측치간 거리가 매우 커지는 문제가 있으므로 변수를 각각 평균 0, 분산 1로 scaling하는 것이 도움이 됨을 알게 되었다.

Exercise 4.4 Curse of Dimensionality

Fraction of the available observations to make the prediction

(Using only observations within 10% of the range of X closest to each test observation)

(a) $\mathbf{p = 1,\ X \sim Uniform[0,\ 1]}$

$$(0.65 - 0.55)^1 = (0.1)^1 = \mathbf{10\%}$$

$$(\mathrm{n} = 1000,\ \mathrm{p} = 1)\ \text{Simulation Result} = \mathbf{0.0988}$$

(b) $\mathbf{p = 2,\ (X_1, X_2) \sim Uniform[0,\ 1]^2}$

$$(0.65 - 0.55) \cdot (0.4 - 0.3) = (0.1)^2 = \mathbf{1\%}$$

$$(\mathrm{n} = 1000,\ \mathrm{p} = 2)\ \text{Simulation Result} = \mathbf{0.0097}$$

(c) $\mathbf{p = 100,\ (X_1, \ldots, X_{100}) \sim Uniform[0,\ 1]^{100}}$

$$(0.1)^{100} \cong \mathbf{0}$$

$$(\mathrm{n} = 1000,\ \mathrm{p} = 100)\ \text{Simulation Result} = \mathbf{2.10e - 95}$$

(d) Drawback of KNN:

When p is large, there are very few training observations 'near' any given test observation.

관측치 당 feature의 개수가 p개라고 할 때, 각 관측치 주변 10% 이내의 표본 공간은 전체 공간의 $(0.1)^p$으로 p가 커질수록 기하급수적으로 작아진다. n=1000개의 Uniform 난수 표본을 각각 p=1, 2, 100차원 생성하여 1000개 중 각 표본 주변의 10% 공간 이내에 포함된 다른 표본의 비율을 계산하고 평균을 구하였을 때 결과가 실제와 거의 동일했다.

이것은 임의의 test observation에 대한 response를 예측하기 위해 사용할 수 있는 공간이 전체 공간에 비해 매우 작아지므로, 주어진 표본에서 해당 test observation 주변에 존재하는 train observation이 점점 줄어든다는 것을 의미한다.

(e) Length of the Side of the Hypercube when p=1, 2 and 100

p-dimensional 전체 공간의 Volume 이 1^p라고 가정할 때, 각 test observation 을 중심으로 부피 10%만큼의 비율을 포함하기 위한 hypercube 의 side(edge)를 z 라고 하면 다음과 같이 계산된다.

$$(z)^p = 0.1, \qquad z = (0.1)^{\frac{1}{p}}$$

dimension p	side z
1	0.1000
2	0.3162
100	0.9772

따라서 p 가 커질수록 같은 fraction 을 얻기 위한 z 가 커진다.
이것은 training observation 들의 10%을 사용하기 위해서는 개별 feature 가 포함하고 있는 범위가 매우 넓어야 함을 의미한다.

Exercise 4.10 Weekly Data Set

(a) Summary of Data

Summary Statistics of Quantitative Variables

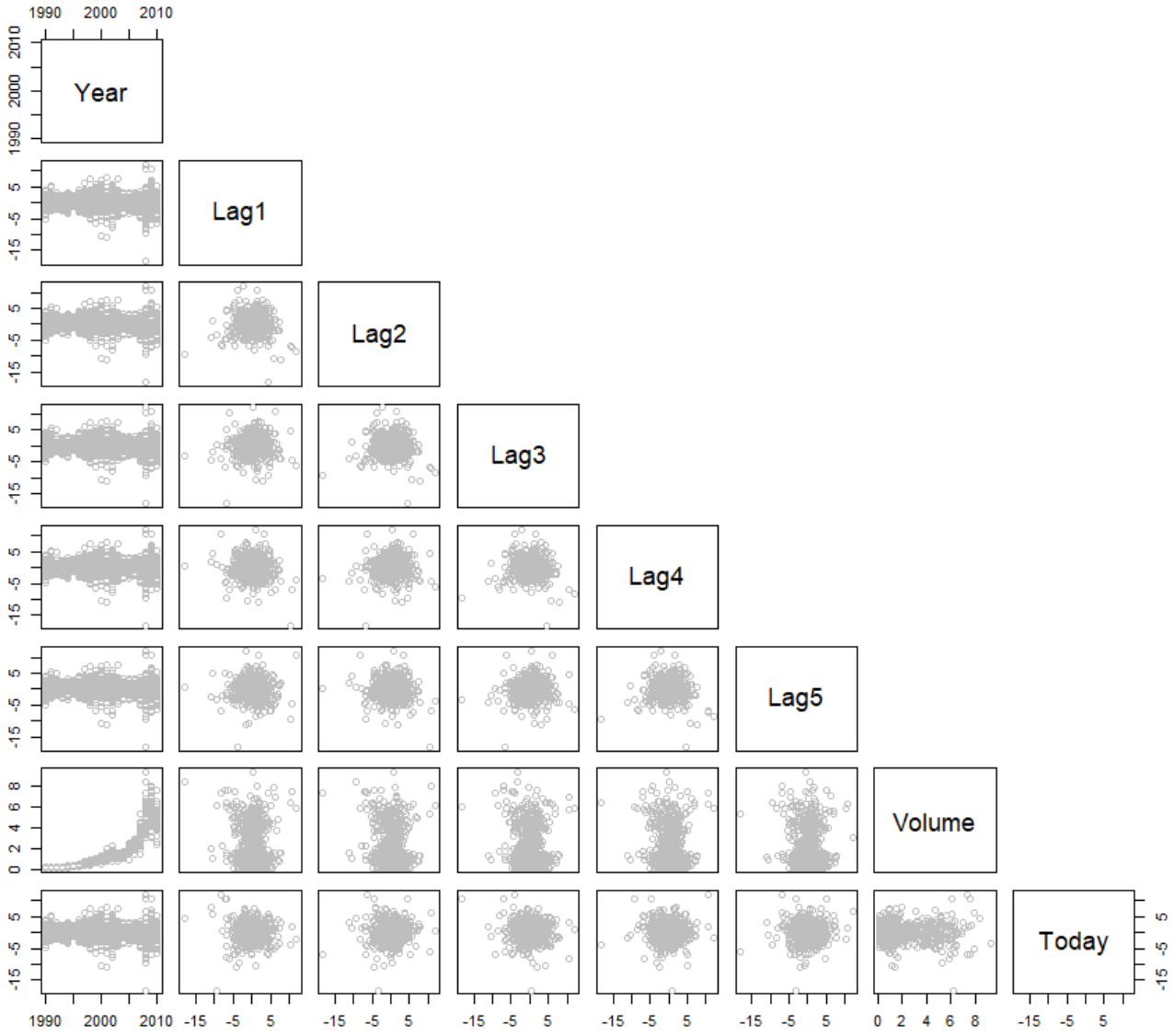
	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
Min	1990	-18.195	-18.195	-18.195	-18.195	-18.195	0.087	-18.195
Max	2010	12.026	12.026	12.026	12.026	12.026	9.328	12.026

Median	2000	0.241	0.241	0.241	0.238	0.234	1.003	0.241
Mean	2005	0.151	0.151	0.147	0.146	0.140	1.575	0.150
SD	6.033	2.357	2.357	2.361	2.360	2.361	1.687	2.357

- Lag1~Lag5 와 Today 에서 일별 수익률의 분포가 거의 동일함을 확인하였다.
- Volume 은 일부 소수의 관측치들의 값이 다른 관측치에 비해 큰 오른쪽으로 꼬리가 긴 분포이다.

Pairwise Scatterplots of Quantitative Variables

양적 변수들 간의 산점도에서 각 일별 수익률은 서로 독립적으로 분포하는 것을 알 수 있었다. Year 와 Volume 의 산점도를 보면 시간의 흐름에 따라 거래량이 점점 더 급격하게 증가하는 비선형 관계의 모습을 보인다.

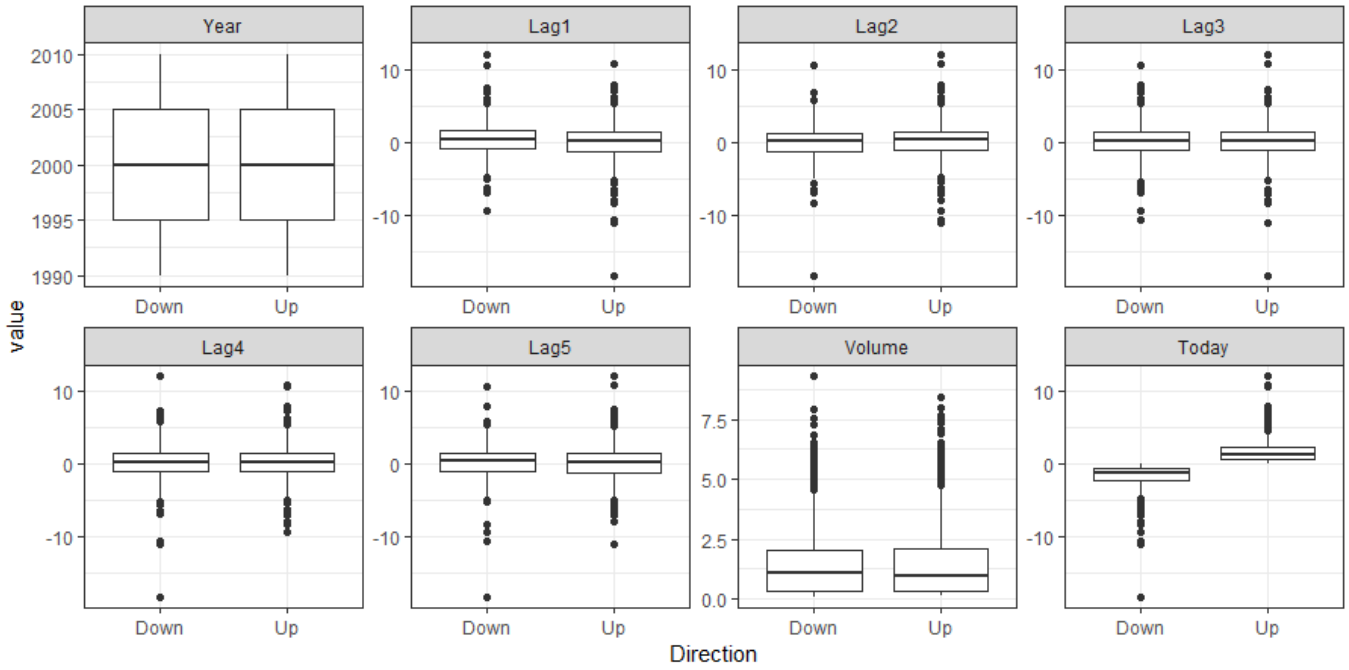


Frequency of Qualitative Variable: Direction

Down	Up
484	605

- Class 별 사전 비율이 서로 다르며 Up 에 해당하는 관측치가 전체의 55%정도로 조금 더 많다.

Pairwise Boxplots Direction (Response) vs Predictors



- 반응변수의 값에 따라 다른 변수들의 분포에 차이가 있는지 보았을 때 Year 부터 Volume 까지의 모든 설명변수들의 분포에 큰 차이가 없는 것처럼 보였다.

(b) Logistic Regression Model using Full Data Set

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

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.6949	-1.2565	0.9913	1.0849	1.4579

Coefficients:

	Estimate	Std. Error	z-value	P(>|z|) (p-value)
(Intercept)	0.2669	0.0859	3.1056	** 0.0019
Lag1	-0.0413	0.0264	-1.5626	0.1181
Lag2	0.0584	0.0269	2.1754	* 0.0296
Lag3	-0.0161	0.0267	-0.6024	0.5469
Lag4	-0.0278	0.0265	-1.0501	0.2937
Lag5	-0.0145	0.0264	-0.5485	0.5833
Volume	-0.0227	0.0369	-0.6163	0.5377

(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

- **Lag1**(전일자의 수익률)만이 회귀계수 검정의 p-value 가 0.05 이하로 유의하게 나타났다.

(c) Confusion Matrix & Accuracy

56.11%	TRUE	
Fitted	Down	Up
Down	**54**	48
Up	430	**557**
	TNR=54/484=**11.15%**	**TPR**=557/605=**92.07%**

전체 데이터 셋에 대한 분류의 정확도는 56.11%로 계산되었다.

오차 행렬을 보면 모형에서 대부분의 추정치를 Up 으로 분류한 것을 알 수 있다. Sensitivity (TPR)의 경우 92.07%으로 실제 Up 인 관측치들 중 Up 으로 맞게 분류된 비율은 매우 높으나 Specificity (TNR)은 11.15%으로 실제 Down 중 Down 으로 맞게 분류한 비율이 매우 낮음을 알 수 있다.

(d) Split Data Set & Re-Fit using Train Set with only Lag2 Variable

전체 데이터 중 Year가 1990 년부터 2008 년까지인 985 개의 관측치를 train set 으로, 2009 년과 2010 년의 관측치 104 개를 test set 으로 분할하였다. train set 만을 이용해 Lag2 변수만을 사용한 로지스틱 모형을 적합한 후 test data 에 대해 예측한 결과 오차행렬은 다음과 같다.

62.5%	TRUE	
Fitted	Down	Up
Down	**9**	5
Up	34	**56**
	TNR=9/43=**20.93%**	**TPR**=56/61=**91.07%**

(e) Repeat (d) using LDA

62.5%	TRUE	
Fitted	Down	Up
Down	**9**	5
Up	34	**56**
	TNR=9/43=**20.93%**	**TPR**=56/61=**91.07%**

- LDA 모형의 예측의 결과가 로지스틱 회귀 모형과 동일하다.

(f) Repeat (d) using QDA

58.65%	TRUE	
Fitted	Down	Up
Down	**0**	0
Up	43	**60**
	TNR=0%	**TPR=100%**

- QDA 모형의 정확도는 58.65%지만 오차행렬을 보면 test 데이터에 대해 모든 관측치를 Up 으로 분류한 것을 알 수 있다.

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

50.00%	TRUE	
Fitted	Down	Up
Down	**21**	30
Up	22	**31**
	TNR=21/43=**48.84%**	**TPR**=31/61=**50.82%**

- K=1 으로 가장 가까운 하나의 이웃만을 사용한 KNN 예측 결과는 50%로 랜덤으로 예측하는 것과 동일하다.

(h) Best Model Performance

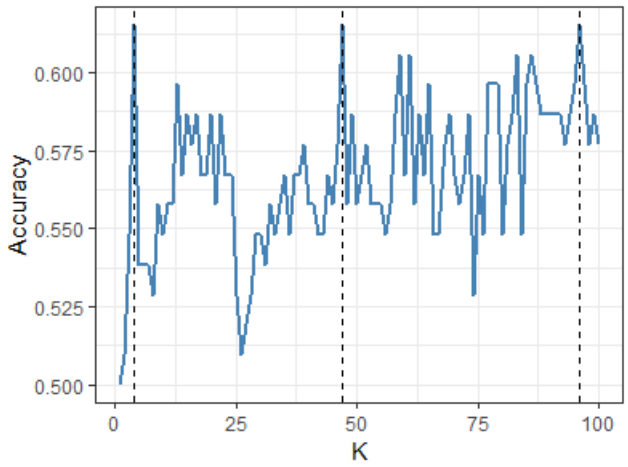
(d)-(g)에서 사용한 4 가지 모형 중 test 데이터에 대해 분류 예측의 정확도가 가장 높은 모형은 62.5%의 로지스틱 회귀 모형과 LDA 이다. 104 개 각각의 관측치에 대한 두 예측 값을 비교하였을 때 100% 일치하였다.

(i) Find the Best Model for each Method - Combinations of Predictors & Transformations & Interactions

Method	Predictor	Accuracy
(Stepwise) Logistic	**Lag1** + **Lag2**	58.65
Logistic	**Lag2** + $(\mathbf{Lag2})^2$	63.46
LDA	$(\mathbf{Lag1})^2$ + **Lag2**	**64.42**
(K=4) KNN		61.54

먼저 train 데이터에 대해 Lag1~Lag5 과 Volume 모든 변수를 사용한 Full Model 에 대해 Stepwise Regression 을 통해 AIC 를 줄이는 모형을 찾아보았을 때 Lag1 과 Lag2 변수가 최종 선택되었으나 test 데이터에 대한 정확도는 58.65%로 Lag2 만을 사용한 모형보다 좋지 못했다. Lag2 의 이차항을 도입하였을 때 63.46%로 상승하였다.

LDA 모형에서는 Lag1 의 이차항과 Lag2 를 사용한 모형의 정확도가 64.42%로 모든 모형 중 가장 높았다.



KNN 에서는 사용하는 이웃의 개수 K 를 1:50 까지 조정하여 각 모형에 대한 예측의 정확도를 평가하였다. 왼쪽에서 K 에 따른 정확도가 K=4, K=47 또는 96 일 때 가장 높은 것을 확인할 수 있으며 이 때의 정확도는 61.54%이다.

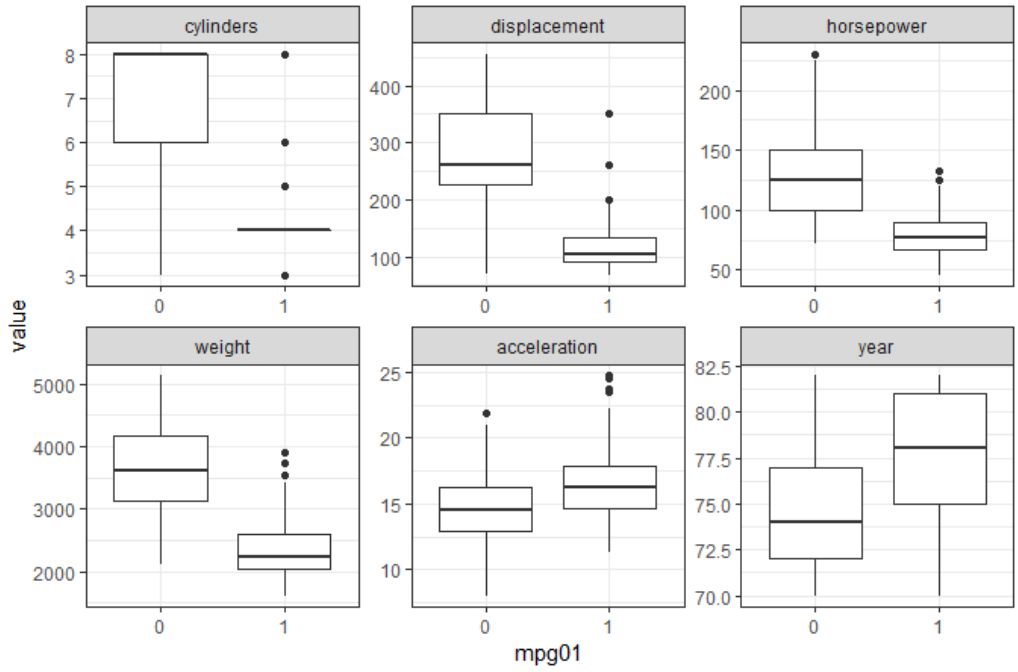
Exercise 4.11 Auto Data Set

(a) Create a Binary Variable mpg01

$$\mathbf{mpg01} = \begin{cases} \mathbf{1}, & \mathbf{if\ mpg} \geq \mathbf{median(mpg)} \\ \mathbf{0}, & \mathbf{if\ mpg} < \mathbf{median(mpg)} \end{cases}$$

(b) Graphical Investigation (mpg01 and other features)

mpg01 은 범주형(binary)이므로 다른 수치형 변수들과의 관계를 그림으로 확인할 때 scatterplot 보다는 boxplot 이 적절하다.



mpg01 의 값에 따른 변수들의 분포가 차이가 있는지 확인하였을 때, mpg01 이 1 인 관측치의 cylinders, displacement, horsepower, weight 가 0 인 관측치들에 비해 작은 것으로 보인다. acceeration, year 는 mpg01 이 1 인 관측치들이 0 인 관측치들에 비해 값이 큰 것으로 보인다. 6 개의 변수 모두 mpg01 을 예측하는 데 유용할 것이라고 생각하였다.

	mpg01	
origin	0	1
1	173	72
2	14	54
3	9	70

origin 변수는 3 가지 label 을 가진 범주형 변수로 범주형 변수들 간의 scatterplot 이나 boxplot 은 유용하지 않다. 따라서 contingency table 을 이용하였다.

origin 이 1(American)인 관측치는 주로 mpg01 이 0 에 밀집되어 있으며 2(European)와 3(Japanese)인 관측치는 대부분 mpg01 이 1 에 해당한다. 이 역시 mpg 값에 따라 차이가 명확히 보이므로 mpg01 의 예측에 유용한 변수라고 생각하였다.

(c) Split Train & Test Data Set

시드를 고정하고 전체 데이터를 랜덤하게 80%의 Train 데이터(313 개)와 Test 데이터(79 개)로 분할하였다.

(d) LDA

LDA 는 설명변수들의 정규분포를 가정하므로 name, origin, year 변수를 제외하였고 나머지 numeric 변수들이 mpg01 을 예측하는 데 유용할 것이라고 판단하였으므로 cylinders, displacement, horsepower, acceleration, weight 5 가지 변수를 사용해 train 데이터에 대한 LDA 를 적합하였다. 오차행렬과 오분류율은 다음과 같다.

5.06%	TRUE	
Fitted	0	1
0	**32**	1
1	3	**43**
	TNR=91.43%	TPR=97.71%

(e) QDA

10.13%	TRUE	
Fitted	0	1
0	**31**	4
1	4	**40**
	TNR=88.57%	TPR=90.91%

LDA 와 동일한 변수를 사용한 QDA 모형은 예측력이 LDA 에 비해 떨어졌다.

(f) Logistic Regression

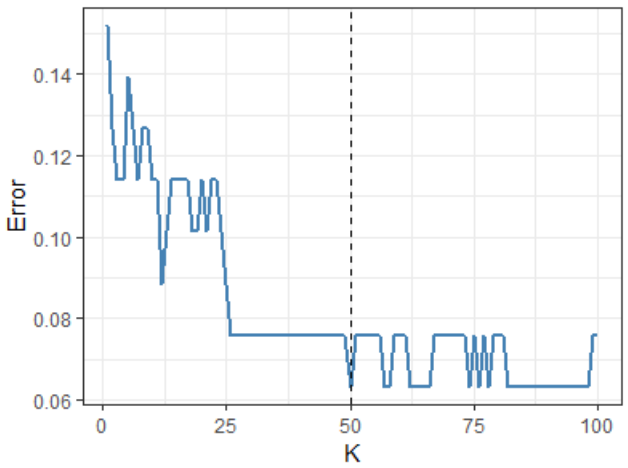
10.13%	TRUE	
Fitted	0	1
0	**33**	6
1	2	**38**
	TNR=94.29%	TPR=86.36%

Logistic Regression 은 범주형, 수치형 설명변수들을 모두 사용 가능하므로 origin 과 year 을 포함한 총 7 개의 변수를 사용하였다. test 데이터에 대한 예측력은 QDA 와 동일했으나 실제 0 을 0 으로 분류한 TNR 이 조금 더 높고 TPR 은 조금 떨어졌다.

(g) KNN with several values of K

K=50

6.33%	TRUE	
Fitted	0	1
0	**32**	2
1	3	**42**
	TNR=91.43%	TPR=95.45%



KNN 역시 관측치 간 거리를 이용하므로 범주형 설명변수 origin 은 제외하였다.

K=1:100 의 KNN 을 이용해 test 데이터에 대해 예측하였을 때 K 가 커짐에 따라, 즉 더 많은 이웃을 사용함에 따라 오분류율이 감소하는 경향을 보였다. 가장 작은 오분류율은 K 가 50 일 때 등 6.33%으로 계산되었다.

- 따라서 mpg01 예측 모형 중에서는 LDA 가 가장 예측의 정확도가 높았다.

Discussion

HW4에서는 범주형 변수의 클래스 예측에 4가지 방법론을 적용해 테스트 데이터에 대한 예측력을 서로 비교하였다.

본 과제에서 다룬 데이터는 모두 반응변수가 1 또는 0인 2-Class problem이다. Exercise 10에서는 로지스틱 모형과 LDA 모형이 성능이 좋았으며, Exercise 11에서는 LDA 모형의 예측 성능이 가장 좋았다. binary classification에서 두 방법론이 유사한 결과를 보이며, 성능 또한 뛰어나다는 것을 알 수 있었다. Exercise 11에서는 모형 적합에 사용된 training size가 313개로 작은 편이고 설명변수들의 정규분포 가정에 무리가 없다고 생각되었다.

실제로 다양한 데이터 및 설명변수들에 따라 적절한 모형이 다를 수 있으며 모든 경우에 성능이 뛰어난 방법론은 정해져 있지 않다. 따라서 설명변수의 type에 따라 적절한 방법론을 여러 가지 고려해보고 하나의 방법론 내에서도 반응변수의 class를 잘 예측할 수 있는 설명변수들의 조합을 찾는 등 좋은 예측 모형을 찾기 위해서는 다양한 시도가 필요하다.

[Appendix] R code

Exercise 4.4

```
meanProp <- function(X) {
  n <- nrow(X) ; p <- ncol(X) ; prop <- matrix(nrow = n, ncol = p)
  for (i in 1:n) for(j in 1:p)
    prop[i,j] <- mean(between(X[,p], X[i,p]-0.05, X[i,p]+0.05))
  prop <- apply(prop, 1, prod)
  return(mean(prop))
}

n <- 1000

# (a) p = 1
set.seed(1)
X1 <- matrix(runif(n), ncol = 1)
meanProp(X1)

# (b) p = 2
set.seed(1)
X2 <- matrix(runif(n*2, 0, 1), ncol = 2)
meanProp(X2)

# (c) p = 100
set.seed(1)
X100 <- matrix(runif(n*100, 0, 1), ncol = 100)
meanProp(X100)

# (e)
(0.1) # p = 1
(0.1)^(1/2) # p = 2
(0.1)^(1/100) # p = 100
```

Exercise 4.10

```
# (a)
dim(Weekly)
apply(Weekly[,-9], 2, quantile, probs = c(0, 0.5, 1))
apply(Weekly[,-9], 2, mean)
apply(Weekly[,-9], 2, sd)
table(Weekly[,9])
cor(Weekly[,-9])
pairs(Weekly[,-9], upper.panel = NULL, col = 'gray')

gather(Weekly, var, value, -Direction) %>%
  mutate(var = factor(var, levels = names(Weekly)[-9])) %>%
  ggplot() + theme_bw() +
  geom_boxplot(aes(x = Direction, y = value, group = Direction)) +
  facet_wrap(~var, ncol = 4, scales = 'free')

# (b) GLM Full
glmFit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
              data = Weekly, family = binomial)
summary(glmFit) # Lag2

# (c)
glmProb <- predict(glmFit, type = 'response')
glmPred <- ifelse(glmProb > 0.5, 'Up', 'Down')
table(glmPred, Weekly$Direction)
mean(glmPred == Weekly$Direction)

# (d) Split & only Lag2
train <- Weekly %>% filter(Year %in% (1990:2008))
test <- Weekly %>% filter(Year %in% (2009:2010))

glmFit.d <- glm(Direction ~ Lag2, data = train, family = binomial)
summary(glmFit.d)

glmProb.d <- predict(glmFit.d, newdata = test, type = 'response')
glmPred.d <- ifelse(glmProb.d > 0.5, 'Up', 'Down')
table(glmPred.d, test$Direction)
mean(glmPred.d == test$Direction)

# (g) KNN
trainX <- as.matrix(train$Lag2)
testX <- as.matrix(test$Lag2)
trainY <- as.matrix(train$Direction)

set.seed(1)
knnPred <- knn(trainX, testX, trainY, k = 1)
table(knnPred, test$Direction)
mean(knnPred == test$Direction)

# (i)
glmFit <- step(glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                   data = Weekly, family = binomial))
glmProb.i <- predict(glmFit.i, test, type = 'response')
glmPred.i <- ifelse(glmProb.i > 0.5, "Up", "Down")
mean(glmPred.i == test$Direction)

glmFit.i <- update(glmFit.i, ~ Lag1*Lag2)
glmProb.i <- predict(glmFit.i, test, type = 'response')
glmPred.i <- ifelse(glmProb.i > 0.5, "Up", "Down")
mean(glmPred.i == test$Direction)

glmFit.i <- update(glmFit.i, ~ poly(Lag2, 2))
glmProb.i <- predict(glmFit.i, test, type = 'response')
glmPred.i <- ifelse(glmProb.i > 0.5, "Up", "Down")
mean(glmPred.i == test$Direction)

ldaFit.i <- update(ldaFit, ~ I(Lag1^2) + Lag2)
ldaPred.i <- predict(ldaFit.i, newdata = test)$class
mean(ldaPred.i == test$Direction)

# KNN
p <- 100
for (i in 2:p) {
  set.seed(1)
  tmp <- knn(trainX, testX, trainY, k = i)
  knnPred <- data.frame(knnPred, tmp)
}
colnames(knnPred) <- paste0('p', 1:p)
```

```
# (e) LDA
ldaFit <- lda(Direction ~ Lag2, data = train)
ldaPred <- predict(ldaFit, newdata = test)$class
table(ldaPred, test$Direction)
mean(ldaPred == test$Direction)

# (f) QDA
qdaFit <- qda(Direction ~ Lag2, data = train)
qdaPred <- predict(qdaFit, newdata = test)$class
table(qdaPred, test$Direction)
mean(qdaPred == test$Direction)
```

```
knnAcc <- NULL
for(i in 1:p) knnAcc <- c(knnAcc, mean(knnPred[,i] == test$Direction))
which.max(knnAcc) ; knnAcc[which.max(knnAcc)]
ggplot()+ theme_bw() +
geom_line(aes(1:p, knnAcc), col = 'steelblue', size = 1) +
  geom_vline(aes(xintercept = which(knnAcc == max(knnAcc))), lty = 2) +
  labs(x = 'K', y = 'Accuracy')
```

Exercise 4.11

```
# (a)
myAuto <- Auto %>%
  mutate(mpg01 = as.numeric(mpg >= median(mpg))) %>%
  select(-name, -mpg)

# (b)
# scatterplot
gather(myAuto, var, value, -mpg01) %>%
  mutate(var = factor(var, levels = names(myAuto)[-8])) %>%
  ggplot() + theme_bw() + theme(legend.position = '') +
  geom_point(aes(x = factor(mpg01), y = value), color = 'gray50') +
  facet_wrap(~var, ncol = 4, scales = 'free') +
  labs(x = 'mpg01')

# origin은 categorical이라서 scatterplot X

# boxplot
gather(myAuto, var, value, -mpg01) %>%
  mutate(var = factor(var, levels = names(myAuto)[-8])) %>%
  filter(var != 'origin') %>%
  ggplot() + theme_bw() + theme(legend.position = '') +
  geom_boxplot(aes(x = factor(mpg01), y = value, group = mpg01)) +
  facet_wrap(~var, ncol = 3, scales = 'free') +
  labs(x = 'mpg01')

# contingency
table(myAuto$origin, myAuto$mpg01)

# (c)
myAuto$origin <- as.factor(myAuto$origin)

set.seed(1)
tr <- sample(1:nrow(myAuto), nrow(myAuto)*0.8)
te <- setdiff(1:nrow(myAuto), tr)
train <- myAuto[tr,]
test <- myAuto[te,]

# (d) LDA
(ldaFit <- lda(mpg01 ~ . -origin -year, data = train))
ldaPred <- predict(ldaFit, test)$class
table(ldaPred, test$mpg01)
mean(ldaPred != test$mpg01)

# (e) QDA
```

```
(qdaFit <- qda(mpg01 ~ . -origin -year, data = train))
qdaPred <- predict(qdaFit, test)$class
table(qdaPred, test$mpg01)
mean(qdaPred != test$mpg01)

# (f) GLM
summary(glmFit <- glm(mpg01 ~ ., data = train, family = binomial))
glmProb <- predict(glmFit, test)
glmPred <- (ifelse(glmProb > 0.5, 1, 0))
table(glmPred, test$mpg01)
mean(glmPred != test$mpg01)

# (g) KNN
trainX <- data.matrix(train[,-(7:8)])
testX <- data.matrix(test[,-(7:8)])
trainY <- data.matrix(train[,8])

set.seed(10)
knnPred <- knn(trainX, testX, trainY, k = 1)
table(knnPred, test$mpg01)
mean(knnPred != test$mpg01)

p <- 100
for (i in 2:p) {
  set.seed(10)
  tmp <- knn(trainX, testX, trainY, k = i)
  knnPred <- data.frame(knnPred, tmp)
}
colnames(knnPred) <- paste0('p', 1:p)

knnError <- NULL
for (i in 1:p) knnError <- c(knnError, mean(knnPred[,i] != test$mpg01))
which.min(knnError)
knnError[knnError == min(knnError)]

table(knnPred[,50], test$mpg01)
mean(knnPred[,50] != test$mpg01)

ggplot() + theme_bw() +
  geom_line(aes(1:p, knnError), col = 'steelblue', size = 1) +
  geom_vline(aes(xintercept = which.min(knnError)), lty = 2) +
    labs(x = 'K', y = 'Error')
```