Yonsei University, Department of Applied Statistics

Chapter 4 실습

신보현 2014122016

January 28, 2019

ISLR package에 있는 Smarket data로 분류 모델을 실습해본다. 주식데이터로 이전 5일의 percentage returns을 기록한 lag1 ~ lag5, 전날에 거래된 주식의 개수인 Volume 그리고 시장이 up 또는 down 이었는지를 알려주는 Direction으로 구성되어 있다.

1. Logistic Regression

로지스틱 회귀를 시행하기 위해서는 glm() 함수를 이용하는데 이 때, family=binomial이라고 명시를 해주어야한다.

```
library(ISLR)
attach(Smarket)
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
head(Smarket)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
## 1 2001  0.381 -0.192 -2.624 -1.055  5.010 1.1913  0.959        Up
## 2 2001  0.959  0.381 -0.192 -2.624 -1.055 1.2965  1.032        Up
## 3 2001  1.032  0.959  0.381 -0.192 -2.624 1.4112 -0.623      Down
## 4 2001 -0.623  1.032  0.959  0.381 -0.192 1.2760  0.614        Up
## 5 2001  0.614 -0.623  1.032  0.959  0.381 1.2057  0.213        Up
## 6 2001  0.213  0.614 -0.623  1.032  0.959 1.3491  1.392        Up
```

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

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

결과를 보면 로지스틱 회귀의 각 계수 p-value가 그리 작지 않음을 확인할 수 있다. 이는 모든 변수를 사용하는 것이 아니라, 일부를 사용해야 할 수도 있음을 시사한다.
predict() 함수는 market이 올라갈지에 대한 확률을 보여준다. 다른 데이터를 설정하지 않으면 training data에 대한 확률을 보여준다. type = "response"는 $P(Y = |X)$를 보여준다. 또한 contrasts() 함수는 R이 dummy variable을 어떻게 설정했는지 보여준다.

```
glm.probs = predict(glm.fit, type="response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509
##         8         9        10
## 0.5092292 0.5176135 0.4888378
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

이제 training data에 대한 확률을 구했으니, 어떤 threshold을 정해서 이 이상이면 up이고 이하면down이라고 label을 붙이면 된다.

```
direction.label = rep("down",1250)
up.index = which(glm.probs>0.5)
direction.label[up.index] = rep("up",length(up.index))
```

이제 confusion matrix을 살펴보자.

```
table(direction.label,Direction)


##                Direction
## direction.label Down  Up
##            down  145 141
##            up    457 507


(507+145)/1250 #올바르게 분류한 비율


## [1] 0.5216
```

하지만 이는 training data에 대해서 계산한 것으로, 실제 모델을 만들 때 쓰이지 않은 데이터로 성능을 확인해봐야 한다. (생략)
새로운 데이터에 대해서 predict을 하고 싶으면 아래와 같이 실행하면 된다.

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

2. LDA

이제 Smartet data에 LDA를 적용해보자. 그 이전에, LDA의 가정이 무엇이었는지 생각해보자. 각 클래스의 데이터 분포가 정규분포를 따라야 하며 각 정규분포의 Covariance Matrix가 같아야 한다는 조건이 있었다.
우선 정규성 검정부터 해보자. Smartet data는 up, down이라는 두 개의 class만을 가지므로 각 클래스별로 데이터를 나눈 후 정규성 검정을 진행해보자.

```
smarket.up = Smarket[up.index,]
smarket.down = Smarket[-up.index,]
library(MVN)


## Warning: package 'MVN' was built under R version 3.5.2
```

```
## sROC 0.1-2 loaded
```

```
mvn(smarket.down[,-9], mvnTest="mardia")
```

```
## $multivariateNormality
##              Test        Statistic               p value Result
## 1 Mardia Skewness 900.314185863058 9.25897970504432e-120     NO
## 2 Mardia Kurtosis 28.8494992633589                     0     NO
## 3             MVN             <NA>                  <NA>     NO
##
## $univariateNormality
##           Test  Variable Statistic   p value Normality
## 1 Shapiro-Wilk   Year       0.8715  <0.001      NO
## 2 Shapiro-Wilk   Lag1       0.9374  <0.001      NO
## 3 Shapiro-Wilk   Lag2       0.9633  <0.001      NO
## 4 Shapiro-Wilk   Lag3       0.9860  0.0069      NO
## 5 Shapiro-Wilk   Lag4       0.9690  <0.001      NO
## 6 Shapiro-Wilk   Lag5       0.9710  <0.001      NO
## 7 Shapiro-Wilk  Volume      0.9595  <0.001      NO
## 8 Shapiro-Wilk   Today      0.9715  <0.001      NO
##
## $Descriptives
##          n           Mean   Std.Dev     Median        Min        Max
## Year   286 2002.31118881 1.1259370 2002.00000 2001.00000 2005.00000
## Lag1   286    1.02489510 1.1164149    0.86100   -1.99800    5.73300
## Lag2   286    0.68094755 1.2779771    0.55250   -3.43000    5.73300
## Lag3   286   -0.27428322 1.2170673   -0.20400   -3.43900    3.90600
## Lag4   286   -0.15671329 1.3907876   -0.07000   -4.15400    5.73300
## Lag5   286   -0.17516084 1.3361112   -0.17550   -4.15400    5.73300
## Volume 286    1.28881948 0.3086584    1.29815    0.35607    2.77556
## Today  286    0.04253497 1.1562983   -0.00900   -4.15400    4.73400
##              25th        75th      Skew   Kurtosis
## Year   2001.00000 2003.000000 0.3259900 -1.0186065
## Lag1      0.22950    1.527750 1.0607744  2.0250079
## Lag2     -0.02400    1.304500 0.5244219  1.7446521
## Lag3     -1.01600    0.401000 0.2593818  0.7831287
## Lag4     -0.95600    0.584750 0.4794860  1.8776390
```

```
## Lag5     -0.96800    0.613000 0.4639688  2.0494333
## Volume    1.12175    1.459275 0.3118150  2.8589364
## Today    -0.56275    0.619500 0.2928441  1.8082433
```

royston으로 test를 해보아도 동일하게 다변량 정규분포가 아니라는 검정 결과가 나온다. 그렇다면 이를 Box-Cox 변환을 통해 정규 분포에 맞춰보자. 우선 해당 데이터는 다변수이고 음수값을 가지므로 양수값일 때만 사용할 수 있는 box-cox 변환은 쓸 수 없다. 그에 따라서 4장 추가 자료 조사에서 살펴보았던 YeoJohnson 방법을 이용한다. 이는 4장 추가 자료조사에서 살펴 봤듯이, 변환을 시행하고 나서도 다변량 정규분포를 통계적으로 유의미하게 따르지 않음을 mvn test에서 확인을 했다. 사실, ISLR 책에서 이 데이터에 대해 LDA를 시행하기 전에 정규성의 가정을 확인을 하지 않고 바로 LDA를 적용했다. 정규성 검정과 변환의 결과로 볼 때, 해당 데이터에 대해 LDA를 적용하는 것은 무리가 있어 보이나, 우선 책의 흐름대로 실습을 진행해보자.

lda() 함수는 MASS library에 있다. lda() 함수의 구문은 lm()의 구문과 동일하다. 2005년 이전의 데이터로만 lda를 적합시킨다.

```
train = which(Year<2005)
library(MASS)
lda.fit = lda(Direction~Lag1+Lag2, data=Smarket, subset=train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

위의 결과에서 Prior probabilities of groups을 살펴보자. $\hat{\pi}_1 = 0.492, \hat{\pi}_2 = 0.508$임을 확인할 수 있다. 다르게

말하면 training 관측치의 49.2%가 down이라는 것이다.
다음으로 Group means을 보자. 이는 각각의 클래스 안에서 각 예측변수의 평균이고 LDA에서 μ_k에 대한 estimates로 쓰인다. 이를 통해 지난 2일 동안의 returns이 market이 증가한다면 negative하고 전날의 returns가 market이 decline한다면 positive할 것이라고 알 수 있다.
coefficients of linear discriminants을 보자. 이는 Lag1과 Lag2의 linear combination의 estimated coefficients이다. 즉, -0.642 * Lag1 -0.514 * Lag2가 decision boundary이다. 만약, 이 값이 크다면 market increase라고, 작으면 market decline이라고 분류할 것이다.
위의 모델로 2005년 데이터를 predict해보자.

```
smarket.2005 = Smarket[-train,]
lda.pred = predict(lda.fit, smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
table(lda.pred$class,Direction[-train])
```

```
##
##        Down  Up
##   Down   35  35
##   Up     76 106
```

```
mean(lda.pred$class == Direction[-train])
```

```
## [1] 0.5595238
```

lda.pred 개체에는 class, posterior,x가 담겨있다. class는 50%라는 threshold을 적용했을 때의 분류 결과이고 posterior은 각 클래스 별로 사후 확률, x는 linear discriminants이다. 사후 확률에 대해 50%의 threshold을 적용하여 동일하게 class 개체를 확인할 수 있으며

```
sum(lda.pred$posterior[,1] >= 0.5)
```

```
## [1] 70
```

```
sum(lda.pred$posterior[,1] < 0.5)
```

```
## [1] 182
```

0.5가 아닌 좀 더 높은 기준으로 분류를 할 수도 있다.

3. QDA

마찬가지로 정규분포 가정을 확인해야 한다. 또한 각 클래스별로 동일하지 않은 공분산 행렬을 가정하기 때문에 LDA보다는 좀 더 relax한 가정이라고 할 수 있다. 사실, k개의 클래스가 동일한 공분산 행렬을 가지기는 힘들듯 하기도 하다. 구문은 lda와 동일하다.

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

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

Prior probabilities of groups, Group means은 LDA와 동일하고 달라진 점은 linear discriminants가 없다는 것인데, QDA는 quadratic form이기 때문에 없는 것이다.
다음으로 QDA로 예측을 해보자.

```
# predict by qda model
qda.pred = predict(qda.fit, smarket.2005)
table(qda.pred$class,Direction[-train])
```

```
##
##        Down  Up
##   Down   30  20
##   Up     81 121
```

```
mean(qda.pred$class==Direction[-train])
```

```
## [1] 0.5992063
```

조금 향상된 정확도를 확인할 수 있다.

4. KNN

knn() 함수는 class library에 있다. 여태까지는 모델을 fitting 하고 predict을 했지만 knn은 한 구문으로 predictions을 수행하는데 이에 네 가지 input이 필요하다.

1. training data의 x 데이터
2. test data의 x 데이터
3. training data의 y 데이터 label
4. K 값

k=1, k=3 일때의 결과를 살펴보자.

```
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[-train])
```

```
##
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
mean(knn.pred==Direction[-train])
```

```
## [1] 0.5
```

```
# when k = 3
knn.pred = knn(train_X, test_X, train.Direction, k=3)
table(knn.pred,Direction[-train])
```

```
##
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

```
mean(knn.pred==Direction[-train])
```

```
## [1] 0.5357143
```

KNN을 시행할 때의 주의할 점

KNN은 어떤 관측치가 있을 때, 가장 가까운 관측치를 고르기 때문에 거리가 굉장히 중요하다. 만약 단위가 크게 다른 변수가 존재한다면 이는 KNN 모델에 크게 영향을 미칠 것이다. 따라서 KNN을 시행하기 이전에, 표준화(standardize)을 하는 것이 매우 중요하다. R에서는 이를 scale() 함수를 통해 간단하게 할 수 있다.