# 4장 분류 R 연습

## 분류

임요한

서울대학교

Aug, 2018

# “분류" 수업에서 다룰 내용

- 분류에 대한 소개
- 베이즈 분류기
- 최근접이웃방법
- 로지스틱 회귀
- 선형판별분석
- 혼동행렬, 오분류율, 민감도와 특이도, ROC 커브

# 보험자료

```
library(ggplot2)
ins=read.csv("insurance.csv",header=T)
ins=ins[,-c(3,4)]
ins$clm=as.factor(ins$clm)
head(ins)
```

```
##   clm  exposure veh_body veh_age gender area agecat veh_value
## 1   0 0.3039014    HBACK       3      F    C      2      1.06
## 2   0 0.6488706    HBACK       2      F    A      4      1.03
## 3   0 0.5694730      UTE       2      F    E      2      3.26
## 4   0 0.3175907    STNWG       2      F    D      2      4.14
## 5   0 0.6488706    HBACK       4      F    C      2      0.72
## 6   0 0.8542094    HDTOP       3      M    C      4      2.01
```

```
#ggplot(data = ins, aes(x =exposure, y = veh_value)) +
#    geom_point(alpha = 0.1, aes(color=clm))
```

```
#ggplot(data = ins, aes(x =clm, y = exposure)) +
#    geom_boxplot()
```

# Credit default자료

- predictor: 1-23
- default: 24
- 표본수=30000: 25000 = training, 5000=testing

# 알즈하이머자료

- diagnosis: “Impaird”, “Control”
- predictors: 130 variables
- 333 obs : 250 train samples, 83 test samples

```r
load("AlzheimerDisease.RData")
head(predictors,n=1)
```

```
##   ACE_CD143_Angiotensin_Converti ACTH_Adrenocorticotropic_Hormon
## 1                         2.0031                       -1.386294
##   Adiponectin Alpha_1_Antichymotrypsin Alpha_1_Antitrypsin
## 1   -5.360193                1.740466          -12.63136
##   Alpha_1_Microglobulin Alpha_2_Macroglobulin Angiopoietin_2_ANG_
## 1             -2.577022             -72.65029             1.06471
##   Angiotensinogen Apolipoprotein_A_IV Apolipoprotein_A1 Apolipopr
## 1        2.510547          -1.427116         -7.402052         -0
##   Apolipoprotein_B Apolipoprotein_CI Apolipoprotein_CIII Apolipop
## 1        -4.624044         -1.272966          -2.312635
##   Apolipoprotein_E Apolipoprotein_H B_Lymphocyte_Chemoattractant_
## 1         3.754521       -0.1573491                       2.2969
##       BMP_6 Beta_2_Microglobulin Betacellulin C_Reactive_Protein
## 1 -2.200744            0.6931472          34          -4.074542
##         CD40       CD5L Calbindin Calcitonin      CgA Clusterin_A
## 1 -0.7964147 0.09531018  33.21363   1.386294 397.6536        3.55
##   Complement_3 Complement_Factor_H Connective_Tissue_Growth_Facto
## 1    -10.36305            3.573725                      0.530628
```

# K-NN R code -1

```
library(class)
train.X=predictors[1:250,1:129]
test.X=predictors[251:333,1:129]
train.Y=as.numeric(diagnosis[1:250])
test.Y=as.numeric(diagnosis[251:333])
```

# K-NN R code -2

```
set.seed(1)
knn.pred=knn(train=train.X,test=test.X,cl=train.Y,k=5,prob=TRUE)
table(knn.pred,test.Y)
```

```
##          test.Y
## knn.pred  1  2
##        1  3 12
##        2 12 56
```

# K-NN 추가설명

- 함수 knn은 class 패키지 안에 있다.
- 다른 통계함수들은 모형을 적합한 후에, 적합한 결과를 이용해서 예측을 2단계로 따로 하는데, knn은 한번에 예측을 한다.
- 인수 : train은 훈련자료의 설명변수를 행렬이나 데이터프레임으로, test는 예측을 할 시험자료의 설명변수값들을 행렬이나 데이터프레임으로 cl은 훈련자료의 반응변수로 팩터이다. k는 예측에 이용하는 이웃의 개수이다.
- 결과는 팩터이다. 확률은 attributes를 이용해 얻을 수 있다.
- 위의 코드는 $\mathrm{K} = 1$인 경우 아래는 $\mathrm{K} = 3$인 경우이다.
- set.seed를 쓴 이유 : knn은 tie가 있는 경우 랜덤하게 tie를 깨는데, 이를 고정시키기 위해 썼다.

# K-NN - 강의록 주식예제

강의록 주식 예제 따라하기

- Credit default자료
- predictor: 1-23
- default: 24
- 표본수=30000: 25000 = training, 5000=testing

# K-NN - 크레딧 디폴트

```
creditdefault=read.csv("creditdefault.csv",header=T)
creditdefault[,3]=as.factor(creditdefault[,3])
train.X=creditdefault[1:25000,1:23]
test.X=creditdefault[25001:30000,1:23]
train.Y=creditdefault[1:25000,24]
test.Y=creditdefault[25001:30000,24]
test=creditdefault[25001:30000,]
```

```
X=train.X
head(X,n=1)
```

```
##   LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PA
## 1     20000   2         2        1  24     2     2    -1    -1
##   BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY
## 1      3913      3102       689         0         0         0
##   PAY_AMT2 PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6
## 1      689        0        0        0        0
```

X[,1], X[,2], X[,4], X[,5] 변수를 가지고 KNN을 적용하여 보자.

# 분류를 위한 보충 R 코드와 결과

# 1. 주식자료 탐색

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

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

```
dim(Smarket)
```

```
## [1] 1250    9
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500
##  Median :2003   Median : 0.039000   Median : 0.039000
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
```

# 로지스틱 회귀모형

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

```
coef(glm.fit)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938
##         Lag5       Volume
##  0.010313068  0.135440659
```

```
summary(glm.fit)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
summary(glm.fit)$coef[,4]
```

```
## (Intercept)         Lag1         Lag2         Lag3         Lag4
```

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

```
##         1         2         3         4         5         6
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.492
##         8         9        10
## 0.5092292 0.5176135 0.4888378
```

```r
contrasts(Direction)
```

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

```r
glm.pred=rep("Down",1250)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

```r
(507+145)/1250
```

```
## [1] 0.5216
```

```r
mean(glm.pred==Direction)
```

```
## [1] 0.5216
```

```
train=(Year<2005)
Smarket.2005=Smarket[!train,]
dim(Smarket.2005)
```

```
## [1] 252   9
```

```
Direction.2005=Direction[!train]
```

```
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket,family=binomial,s
glm.probs=predict(glm.fit,Smarket.2005,type="response")
glm.pred=rep("Down",252)
glm.pred[glm.probs>.5]="Up"
```

```
table(glm.pred,Direction.2005)
```

```
##             Direction.2005
## glm.pred Down  Up
##     Down   77  97
##     Up     34  44
```

```
mean(glm.pred==Direction.2005)
```

```
## [1] 0.4801587
```

```
mean(glm.pred!=Direction.2005)
```

```
## [1] 0.5198413
```

```
glm.fit=glm(Direction~Lag1+Lag2,data=Smarket,family=binomial,subset=train)
glm.probs=predict(glm.fit,Smarket.2005,type="response")
glm.pred=rep("Down",252)
glm.pred[glm.probs>.5]="Up"
```

```r
table(glm.pred,Direction.2005)
```

```
##             Direction.2005
## glm.pred Down  Up
##     Down   35  35
##     Up     76 106
```

```r
mean(glm.pred==Direction.2005)
```

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

```r
106/(106+76)
```

```
## [1] 0.5824176
```

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

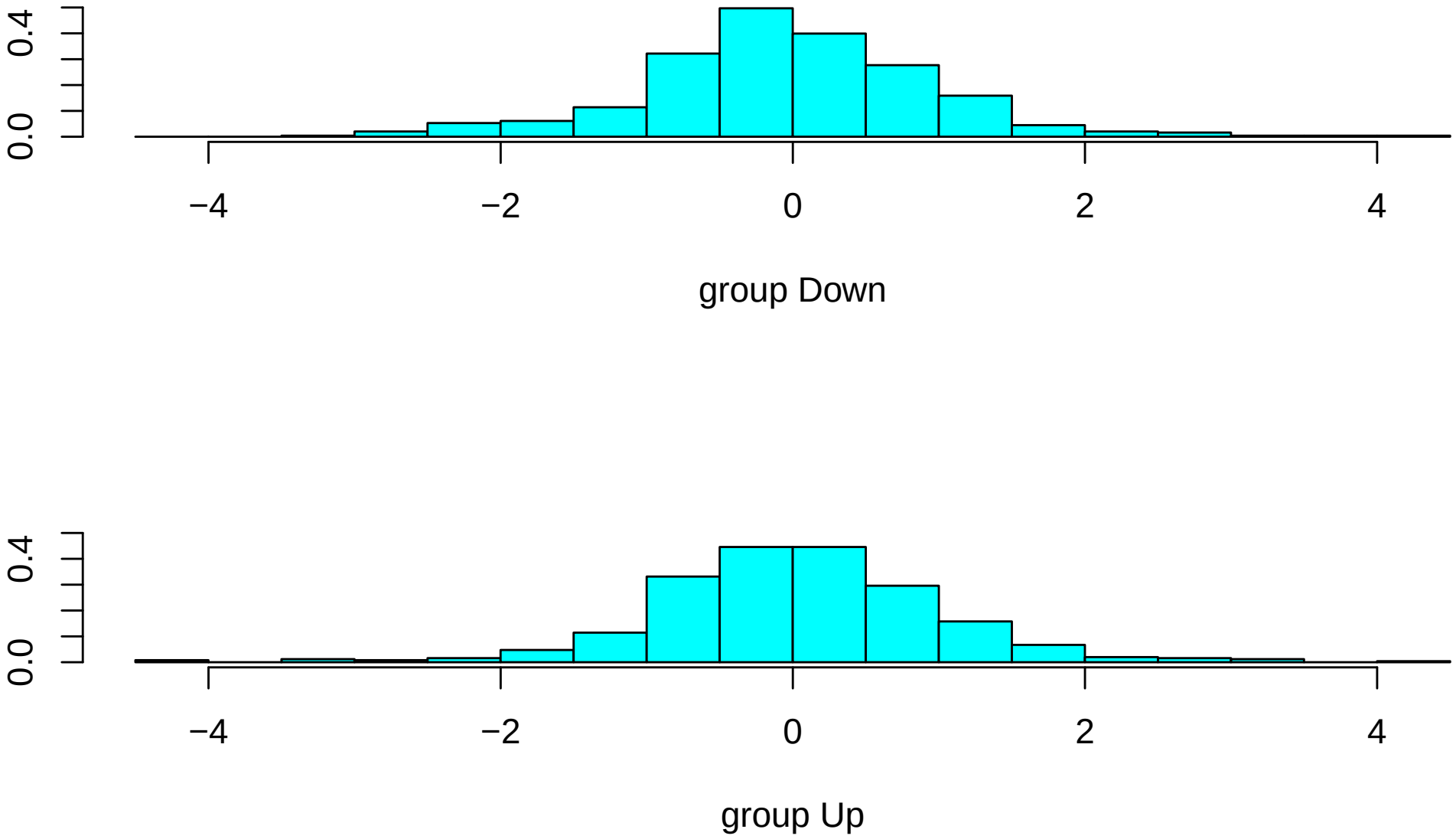
```
##         1         2
## 0.4791462 0.4960939
```

# 선형판별분석

```
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
```

```
plot(lda.fit)
```



```r
lda.pred=predict(lda.fit, Smarket.2005)
names(lda.pred)
```

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

```r
lda.class=lda.pred$class
table(lda.class,Direction.2005)
```

```
##           Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```r
mean(lda.class==Direction.2005)
```

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

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

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

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

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

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

```
##       999      1000      1001      1002      1003      1004
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.495
##      1006      1007      1008      1009      1010      1011
## 0.4872861 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.470
##      1013      1014      1015      1016      1017      1018
## 0.4744593 0.4799583 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class[1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down
## [15] Up   Up   Up   Down Up   Up
```

# 이차판별분석

```
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
```

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

```
##             Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
mean(qda.class==Direction.2005)
```

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

# 최근접 이웃 방법

```
library(class)
train.X=cbind(Lag1,Lag2)[train,]
test.X=cbind(Lag1,Lag2)[!train,]
train.Direction=Direction[train]
```

```r
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction,k=1)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```r
(83+43)/252
```

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

```r
knn.pred=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

```r
mean(knn.pred==Direction.2005)
```

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

# 2. 카라반 보험자료

```
dim(Caravan)
```

```
## [1] 5822   86
```

```
attach(Caravan)
summary(Purchase)
```

```
##    No   Yes
## 5474   348
```

```
348/5822
```

```
## [1] 0.05977327
```

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

```r
var(Caravan[,1])
```

```
## [1] 165.0378
```

```r
var(Caravan[,2])
```

```
## [1] 0.1647078
```

```r
var(standardized.X[,1])
```

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

```r
var(standardized.X[,2])
```

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

```
test=1:1000
train.X=standardized.X[-test,]
test.X=standardized.X[test,]
train.Y=Purchase[-test]
test.Y=Purchase[test]
```

```r
set.seed(1)
knn.pred=knn(train.X,test.X,train.Y,k=1)
mean(test.Y!=knn.pred)
```

```
## [1] 0.118
```

```r
mean(test.Y!="No")
```

```
## [1] 0.059
```

```r
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  873  50
##      Yes  68   9
```

```r
9/(68+9)
```

```
## [1] 0.1168831
```

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

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

```
##          test.Y
## knn.pred  No Yes
##       No  930  55
##       Yes  11   4
```

```
4/15
```

```
## [1] 0.2666667
```

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

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurre
```

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

```r
glm.pred=rep("No",1000)
glm.pred[glm.probs>.5]="Yes"
table(glm.pred,test.Y)
```

```
##          test.Y
## glm.pred  No Yes
##      No  934  59
##      Yes   7   0
```

```r
glm.pred=rep("No",1000)
glm.pred[glm.probs>.25]="Yes"
table(glm.pred,test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  919  48
##      Yes  22  11
```

```r
11/(22+11)
```

```
## [1] 0.3333333
```