Logistic Regression

Yichang Liu 501777

1/10/2022

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

Logistic Regression

Logistic Regression WITHOUT TRAINING DATA

glm() needs to add "family = binomial".

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

```
##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "call"              "formula"           "terms"
## [25] "data"              "offset"            "control"
## [28] "method"            "contrasts"         "xlevels"
```

```
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
```

```
coef(glm.fit) ### coef() to get all the coefficient of the model
```

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

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

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

```
glm.pred = rep("Down",length(glm.probs))
glm.pred[glm.probs>0.5]="Up"
glm.pred[1:10]
```

```
##  [1] "Up"   "Down" "Down" "Up"   "Up"   "Up"   "Down" "Up"   "Up"   "Down"
```

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

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

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

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

Logistic Regression WITH TRAINING DATA

Firstly, define the test data(also can define the training data)

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

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

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

glm.fit use the training data

predict() – get the prediction of test data by using the model constructed by training data. ADD "type ="response"

```
glm.fit = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
              data = Smarket,
              family = binomial,
              subset = train)
glm.probs = predict(glm.fit,Smarket.2005,type = "response")
head(glm.probs)
```

```
##       999      1000      1001      1002      1003      1004
## 0.5282195 0.5156688 0.5226521 0.5138543 0.4983345 0.5010912
```

be able to use str() to get the quantitive value of dummy variable.

```
glm.pred = rep("Down", length(glm.probs))
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)  ### error rate
```

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

logistic regression to some specific points

```
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", length(glm.probs))
glm.pred[glm.probs>0.5] = "Up"
table(glm.pred,Direction.2005)
```

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

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

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

```
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
```