Logistic Regression

Jaeyoon Han

로지스틱 회귀분석 (Logistic Regression)

로지스틱 회귀분석은 앞서 공부한 선형회귀법과 같이 수치를 예측합니다. 하지만 선형회귀법은 정확한 수치를 예측하는 반면, 로지스틱 회귀분석은 0부터 1 사이의 값을 예측합니다. 정확하게 말하자면, 종속변수 Y를 직접 예측하는 것이 아닌 종속변수 Y가 어떤 범주에 속하는 *확률*을 모델링합니다. 이런 이유에서 로지스틱 회귀분석은 회귀 모델(Regression Model)이라기 보다는 분류 모델(Classification Model)로 불립니다.

예를 들어, 통장 잔고에 따른 파산 여부를 예측하고자 합니다. 각각의 변수명을 `balance`, `default` 라고 하겠습니다. 파산 여부는 `Yes` 와 `No` 로 구분할 수 있죠. 로지스틱 회귀분석을 통해 모델링을 한다면 현재 통장 잔고의 상황에서 파산일 확률을 모델링하는 것이죠. 이 내용을 수학적으로 쓴다면 조건부 확률을 이용하여 이렇게 쓸 수 있을 겁니다.

$$\Pr(\text{default} = \text{Yes} \mid \text{balance})$$

만약 이 확률값이 0.5보다 크다면 모두 파산할 것이라고 예측할 수 있습니다. 반대로 0.5보다 작다면 파산하지 않았겠죠. 여기서 로지스틱 회귀분석을 사용하는 당위성을 알 수 있습니다.

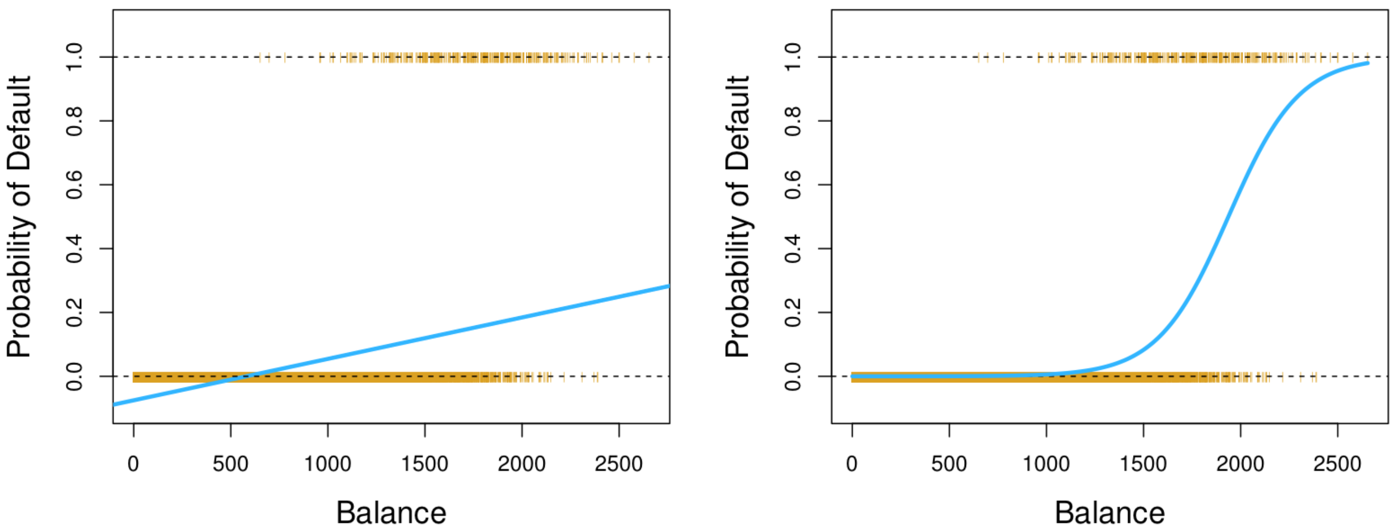
만약 이러한 확률값 모델링을 선형회귀법을 통해서 수행했다고 가정합시다. 그렇다면 굉장히 간단한 수식으로 확률값을 모델링 할 수 있을 겁니다. 하지만 문제는 확률값인데도 불구하고 그 결과값이 0과 1 사이의 값이 아닐 수 있다는 점이죠. 당장에 0 미만인 경우 해당값은 확률의 의미를 잃어버립니다. 따라서 선형적인 함수가 아닌 0과 1 사이의 값을 제공하는 다른 함수를 사용해야 합니다. 로지스틱 회귀분석에서는 아래와 같은 **로지스틱 함수(Logistic Function)**를 사용합니다.

$$p(X) = \frac{e^X}{1 + e^X}$$

여기서 $X = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$ 입니다. 위 식은 이렇게 표현할 수도 있습니다.

$$\begin{aligned}
p(X)(1 + e^X) &= e^X \\
p(X) + p(X) \cdot e^X &= e^X \\
p(X) &= e^X - p(X) \cdot e^X \\
p(X) &= e^X (1 - p(X)) \\
\frac{p(X)}{1 - p(X)} &= e^X = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n} \\
\log\left(\frac{p(X)}{1 - p(X)}\right) &= X = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n.
\end{aligned}$$

$p(X)/(1 - p(X))$는 **공산(odds)**이라 하여 0과 ∞ 사이의 값을 가집니다. 만약 $p(X)$ 값이 1에 가까워진다면 공산값이 ∞에 가까워지며, 반대로 $p(X)$ 값이 0에 가까워진다면 공산값은 0에 가까워집니다. 이 공산에 로그를 취하면 마지막 식인 **로그 공산(log-odds)**, **로짓(logit)**이 됩니다. 로지스틱 회귀모델은 선형적인 로짓을 갖습니다.



각각의 변수에 대해서 한 유닛 증가는 로그 공산을 해당 변수의 계수만큼 변화시킨다고 생각하시면 됩니다. 이 때 증가 변화량은 직선이 아님을 알아두셔야 합니다. 로지스틱 회귀모델은 로지스틱 함수에 적합하고 있으니까요.

로지스틱 회귀분석 실습 (Lab : Logistic Regression)

1. 대학 입학 예측 (Colleage Admissions Prediction)

대학교 4년 과정의 학점 `gpa` 와 대학원 입학 시험 점수 `gre` , 그리고 출신 대학의 명성 `rank` 를 사용해서 입학 여부 `admit` 을 예측해보려 합니다. 주어진 데이터를 임포트하겠습니다.

```
admission <- read.csv("binary.csv")
head(admission)
```

admit	gre	gpa	rank
0	380	3.61	3
1	660	3.67	3
1	800	4.00	1
1	640	3.19	4
0	520	2.93	4
1	760	3.00	2

```
str(admission)
```

```
'data.frame':   400 obs. of  4 variables:
 $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

```
summary(admission)
```

admit	gre	gpa	rank
Min. :0.0000	Min. :220.0	Min. :2.260	Min. :1.000

1st Qu.:0.0000	1st Qu.:520.0	1st Qu.:3.130	1st Qu.:2.000
Median :0.0000	Median :580.0	Median :3.395	Median :2.000
Mean :0.3175	Mean :587.7	Mean :3.390	Mean :2.485
3rd Qu.:1.0000	3rd Qu.:660.0	3rd Qu.:3.670	3rd Qu.:3.000
Max. :1.0000	Max. :800.0	Max. :4.000	Max. :4.000

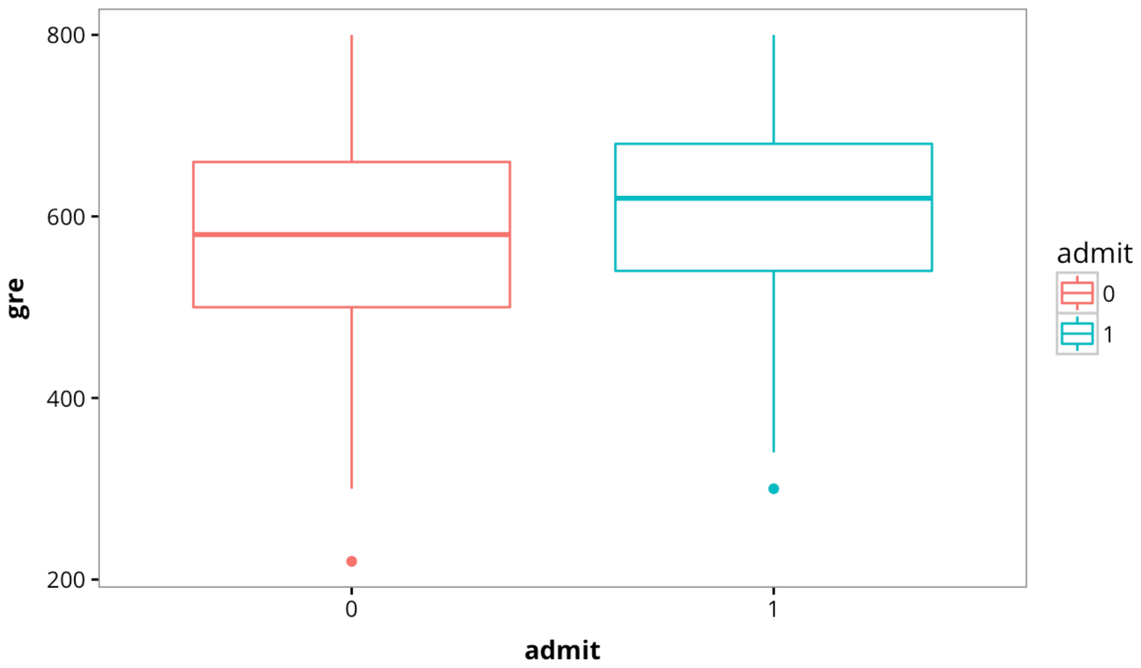
```
xtabs(~admit + rank, data = admission)
```

admit/rank	1	2	3	4
0	28	97	93	55
1	33	54	28	12

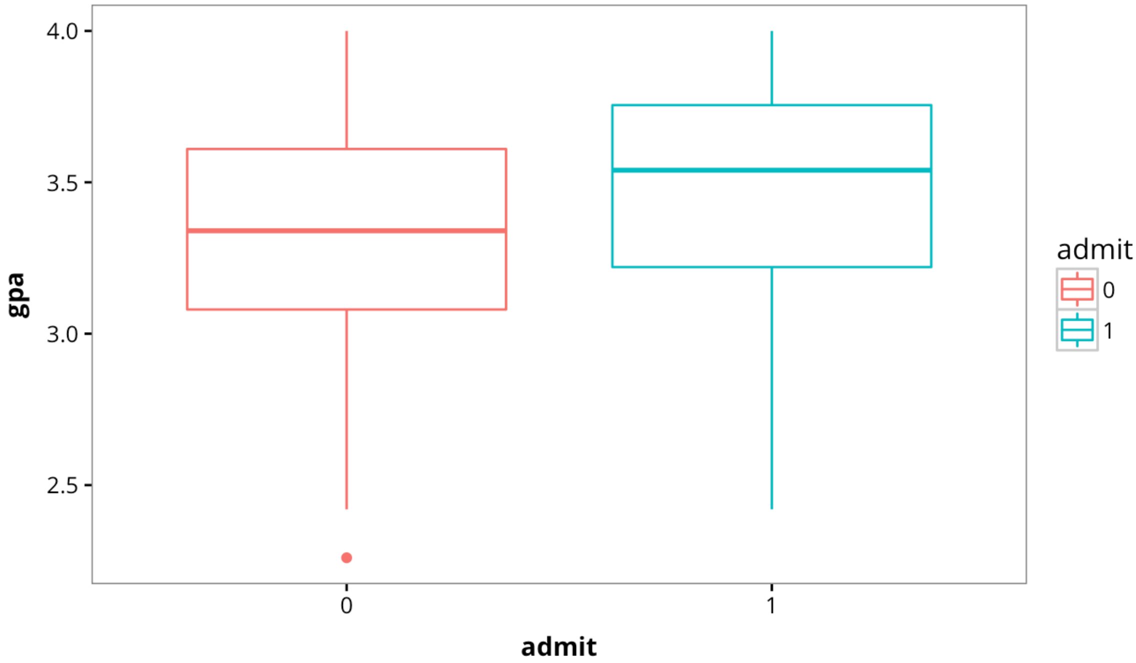
`admit` 과 `rank` 를 요인형 데이터로 바꾸도록 하겠습니다.

```
admission$admit <- factor(admission$admit)
admission$rank <- factor(admission$rank)
```

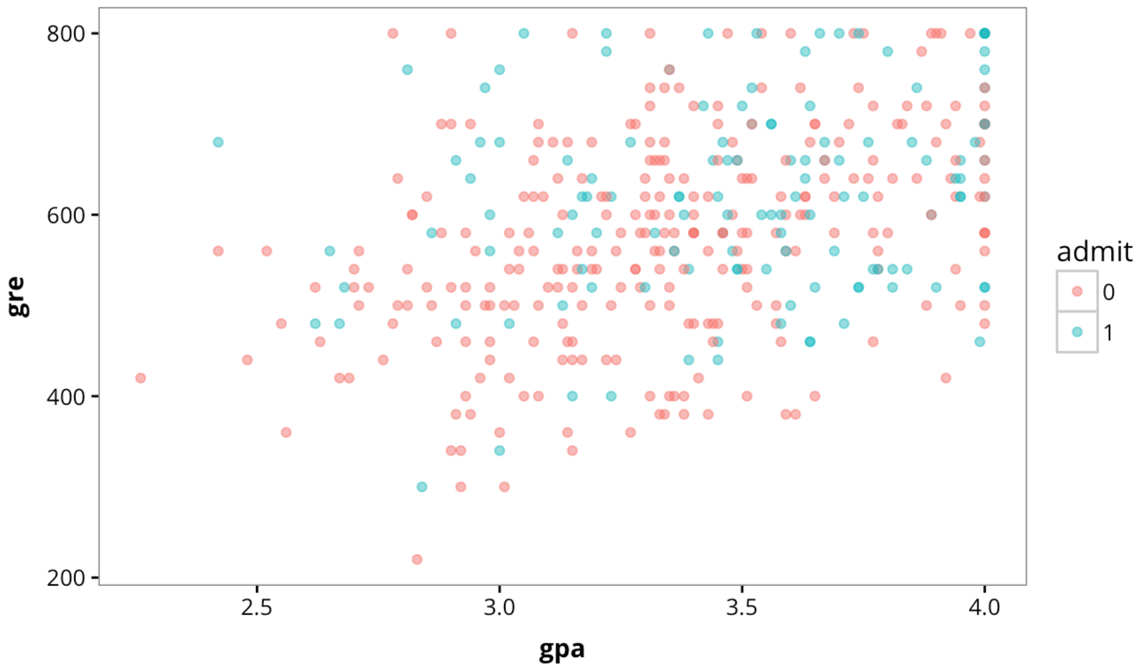
```
ggplot(data = admission, aes(x = admit, y = gre, color = admit)) + geom_boxplot()
```




```
ggplot(data = admission, aes(x = admit, y = gpa, color = admit)) + geom_boxplot()
```




```
ggplot(data = admission, aes(x = gpa, y = gre, color = admit)) + geom_point(alpha = 0.5)
```




```
library(MASS)
library(dplyr)
set.seed(12345)
train <- sample_frac(admission, 0.7)
trainIdx <- as.numeric(row.names(train))
test <- admission[-trainIdx, ]
```

이제 단순하게 로지스틱 회귀분석을 해보죠. 로지스틱 회귀분석은 `glm()` 함수를 이용해 `family = binomial` 을 명시해주면 됩니다.

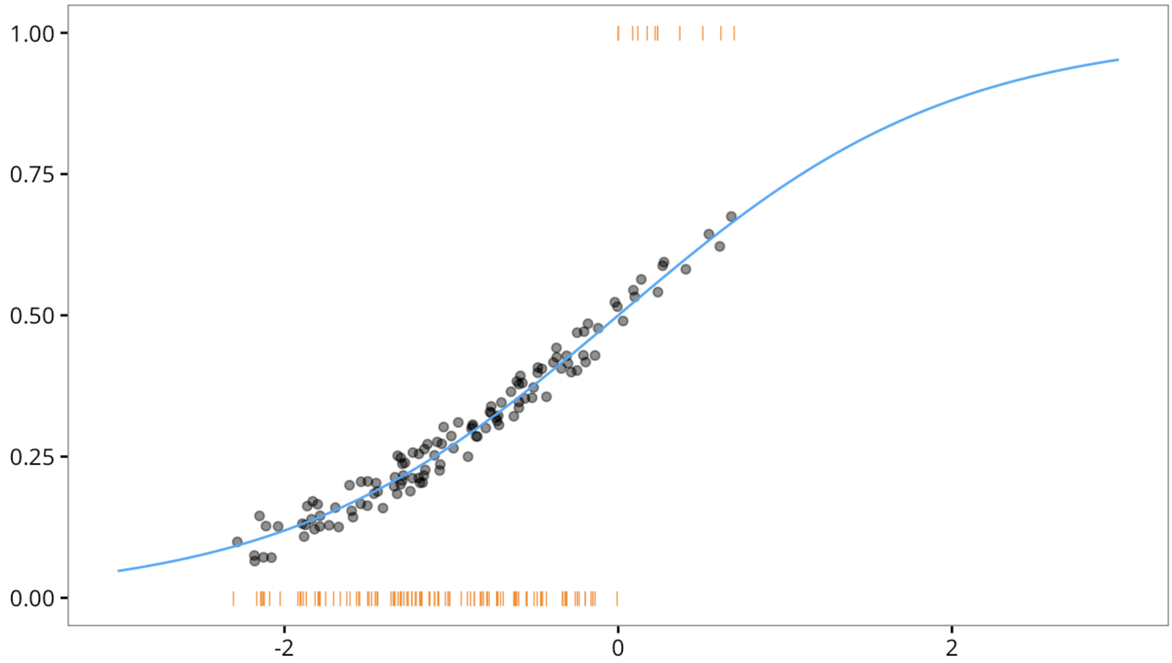
```
logit_admit <- glm(admit ~ gre + gpa + rank, data = train, family = "binomial")
summary(logit_admit)
```

```

Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial", 
    data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.4965  -0.8640  -0.6178   1.1508   1.9943  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -3.543557   1.360965  -2.604  0.00922 ** 
gre          0.001331   0.001362   0.978  0.32815    
gpa          0.806846   0.402905   2.003  0.04522 *  
rank2       -0.484687   0.359375  -1.349  0.17744    
rank3       -1.475998   0.409239  -3.607  0.00031 ***
rank4       -1.436535   0.473162  -3.036  0.00240 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 348.59  on 279  degrees of freedom
Residual deviance: 320.05  on 274  degrees of freedom
AIC: 332.05

Number of Fisher Scoring iterations: 4
```

로그공산을 중심으로 본 모델을 설명하면 다음과 같습니다..

- `gre` 에서 한 단위가 증가할 때마다, log odds는 0.001331 증가한다.
- `gpa` 에서 한 단위 증가할 때마다, log odds는 0.806846 증가한다.
- `rank` 의 경우, `rank == 1` 인 경우가 베이스라인 피처(baseline feature)이며, `rank == 2` 이면 log odds가 0.484687, `rank == 3` 이면 1.475998, `rank == 4` 이면 1.436535만큼 감소한다.

```
x <- predict(logit_admit, test)
predict_admit <- predict(logit_admit, test, type = "response")
binary_admit <- ifelse(predict_admit > 0.5, 1, 0)

sigmoid <- function(x) {
    1/(1 + exp(-x))
}

ggplot(data = data.frame(x = x, y = predict_admit, binary = binary_admit)) +
    geom_jitter(aes(x = x, y = y), width = 0.1, height = 0.1, alpha = 0.5) +
    coord_cartesian(ylim = c(0, 1), xlim = c(-3, 3)) + stat_function(fun = sigmoid,
    xlim = c(-3, 3), color = "#56A9F6") + geom_point(aes(x = x, y = binary),
    shape = "|", color = "#f18f2e", size = 2) + xlab(NULL) + ylab(NULL)
```




```
accuracy <- function(actual, predict) {
    return(sum(actual == predict)/length(actual))
}

accuracy(test$admit, binary_admit)
```

```
[1] 0.7583333
```

```
library(caret)
confusionMatrix(binary_admit, test$admit)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 80 28
         1  1 11

               Accuracy : 0.7583
                 95% CI : (0.6717, 0.8318)
    No Information Rate : 0.675
    P-Value [Acc > NIR] : 0.02973

                  Kappa : 0.3287
 Mcnemar's Test P-Value : 1.379e-06

            Sensitivity : 0.9877
            Specificity : 0.2821
         Pos Pred Value : 0.7407
         Neg Pred Value : 0.9167
             Prevalence : 0.6750
         Detection Rate : 0.6667
   Detection Prevalence : 0.9000
      Balanced Accuracy : 0.6349

       'Positive' Class : 0
```

```
ranktest <- admission %>% group_by(rank) %>% summarise(gre = mean(gre), gpa = mean(gpa))
ranktest$rankProb <- predict(logit_admit, ranktest, type = "response")
ranktest
```

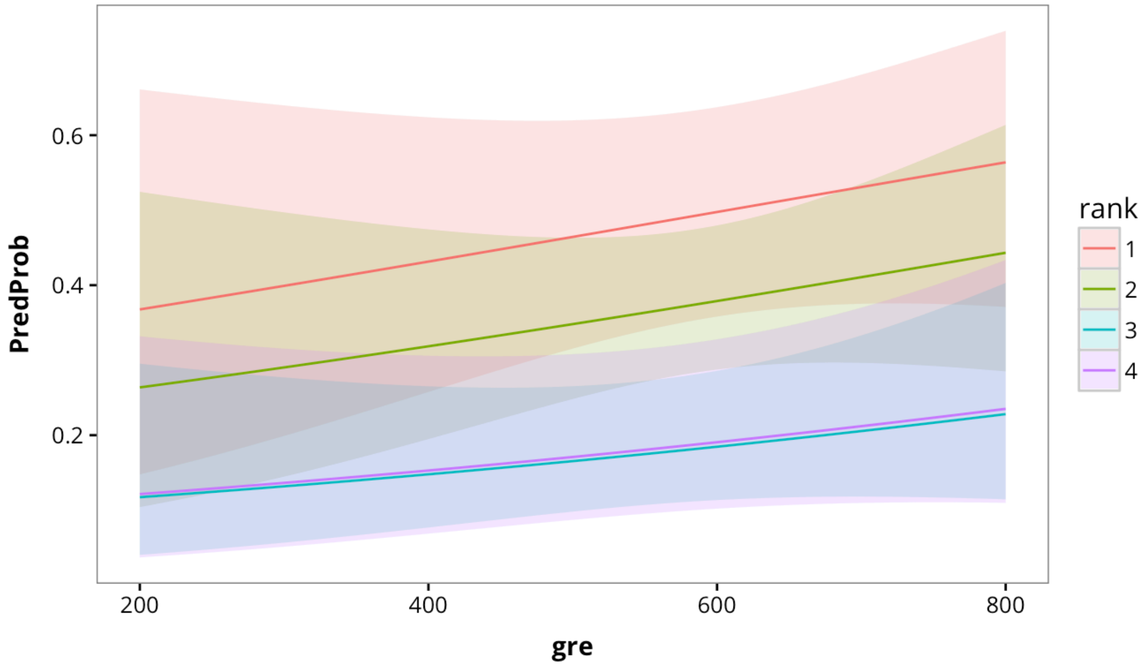
rank	gre	gpa	rankProb
1	611.8033	3.453115	0.5142845
2	596.0265	3.361656	0.3723070
3	574.8760	3.432893	0.1847747
4	570.1493	3.318358	0.1760170

=======================================

```
tmp <- with(admission, data.frame(gre = rep(seq(200, 800, length.out = 100),
    4), gpa = mean(gpa), rank = factor(rep(1:4, each = 100))))

# Get probabilities and standard errors (for plotting)
probs <- cbind(tmp, predict(logit_admit, newdata = tmp, type = "link", se = TRUE))
# Get a look at this new data
head(probs)
```

gre	gpa	rank	fit	se.fit	residual.scale
200.0000	3.3899	1	-0.5421433	0.6172925	1
206.0606	3.3899	1	-0.5340740	0.6100378	1
212.1212	3.3899	1	-0.5260046	0.6028088	1
218.1818	3.3899	1	-0.5179353	0.5956063	1
224.2424	3.3899	1	-0.5098659	0.5884315	1
230.3030	3.3899	1	-0.5017966	0.5812852	1

```
probs <- within(probs, {
    PredProb <- plogis(fit)  # fit logistic curve
    LL <- plogis(fit - (1.96 * se.fit))  # create lower limits
    UL <- plogis(fit + (1.96 * se.fit))
})

ggplot(probs, aes(x = gre, y = PredProb)) + geom_ribbon(aes(ymin = LL, ymax = UL,
    fill = rank), alpha = 0.2) + geom_line(aes(color = rank))
```



2. 주식 시장 예측 (Stock Market Prediction)

사용할 데이터는 `ISLR` 패키지에서 가져오겠습니다. 이 데이터셋은 2001년에서 2005년까지 1,250일에 걸친 S&P 500 주가지수의 수익률 퍼센테이지로 구성되어 있으며, 각 날짜에 그 날 이전의 5일의 각 거래일 `Lag1` 에서 `Lag5` 에 대한 수익률이 기록되어 있습니다. 또한 전날에 거래된 주식 수를 10억주 단위로 표시한 `Volume` , 당일의 수익률 `Today` , 당일 주가지수의 상승 여부 `Direction` 으로 구성되어 있습니다.

```
library(ISLR)
str(Smarket)
```

```
'data.frame':	1250 obs. of  9 variables:
 $ Year     : num  2001 2001 2001 2001 2001 ...
 $ Lag1     : num  0.381 0.959 1.032 -0.623 0.614 ...
 $ Lag2     : num  -0.192 0.381 0.959 1.032 -0.623 ...
 $ Lag3     : num  -2.624 -0.192 0.381 0.959 1.032 ...
 $ Lag4     : num  -1.055 -2.624 -0.192 0.381 0.959 ...
 $ Lag5     : num  5.01 -1.055 -2.624 -0.192 0.381 ...
 $ Volume   : num  1.19 1.3 1.41 1.28 1.21 ...
 $ Today    : num  0.959 1.032 -0.623 0.614 0.213 ...
 $ Direction: Factor w/ 2 levels "Down","Up": 2 2 1 2 2 2 1 2 2 2 ...
```

```
summary(Smarket)
```

Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
Min. :2001	Min. :-4.922000	Min. :-4.922000	Min. :-4.922000	Min. :-4.922000	Min. :-4.92200	Min. :0.3561	Min. :-4.922000	Down:602
1st Qu.:2002	1st Qu.:-0.639500	1st Qu.:-0.639500	1st Qu.:-0.640000	1st Qu.:-0.640000	1st Qu.:-0.64000	1st Qu.:1.2574	1st Qu.:-0.639500	Up :648
Median :2003	Median : 0.039000	Median : 0.039000	Median : 0.038500	Median : 0.038500	Median : 0.03850	Median :1.4229	Median : 0.038500	NA
Mean :2003	Mean : 0.003834	Mean : 0.003919	Mean : 0.001716	Mean : 0.001636	Mean : 0.00561	Mean :1.4783	Mean : 0.003138	NA
3rd Qu.:2004	3rd Qu.: 0.596750	3rd Qu.: 0.596750	3rd Qu.: 0.596750	3rd Qu.: 0.596750	3rd Qu.: 0.59700	3rd Qu.:1.6417	3rd Qu.: 0.596750	NA
Max. :2005	Max. : 5.733000	Max. : 5.733000	Max. : 5.733000	Max. : 5.733000	Max. : 5.73300	Max. :3.1525	Max. : 5.733000	NA

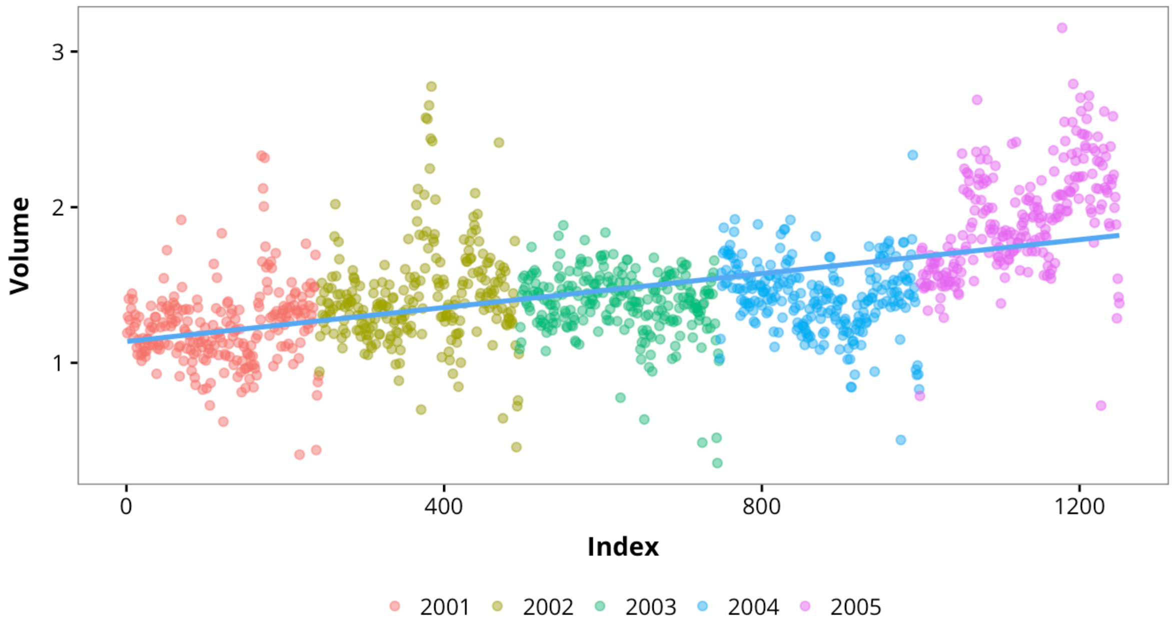
데이터에서 칼럼간 상관관계를 확인해보도록 하죠. 마지막 칼럼은 요인형이므로 수치형 데이터에 대해서만 계산할 수 있는 상관계수를 계산할 수 없습니다.

```
cor(Smarket[-ncol(Smarket)])
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
Year	1.0000000	0.0296996	0.0305964	0.0331946	0.0356887	0.0297880	0.5390065	0.0300952
Lag1	0.0296996	1.0000000	-0.0262943	-0.0108034	-0.0029859	-0.0056746	0.0409099	-0.0261550
Lag2	0.0305964	-0.0262943	1.0000000	-0.0258967	-0.0108535	-0.0035579	-0.0433832	-0.0102500
Lag3	0.0331946	-0.0108034	-0.0258967	1.0000000	-0.0240510	-0.0188083	-0.0418237	-0.0024476
Lag4	0.0356887	-0.0029859	-0.0108535	-0.0240510	1.0000000	-0.0270836	-0.0484142	-0.0068995
Lag5	0.0297880	-0.0056746	-0.0035579	-0.0188083	-0.0270836	1.0000000	-0.0220023	-0.0348601
Volume	0.5390065	0.0409099	-0.0433832	-0.0418237	-0.0484142	-0.0220023	1.0000000	0.0145918
Today	0.0300952	-0.0261550	-0.0102500	-0.0024476	-0.0068995	-0.0348601	0.0145918	1.0000000

`Year` 와 `Volume` 사이의 Positive correlation을 제외하고는 높은 상관관계는 보이지 않음을 알 수 있습니다.

```
ggplot(data = Smarket, aes(x = 1:nrow(Smarket), y = Volume)) + geom_point(aes(colour = factor(Year)),
    alpha = 0.5) + geom_smooth(fill = NA, method = "lm", colour = "#56A9F6") +
    xlab("Index") + theme(legend.position = "bottom", legend.title = element_blank(),
    legend.key = element_blank())
```



트레이닝 데이터로 2001년부터 2004년의 데이터를, 테스트 데이터로 2005년 데이터를 사용하도록 하겠습니다.

```
library(dplyr)
train <- filter(Smarket, Year != 2005)
test <- filter(Smarket, Year == 2005)
```

로지스틱 회귀는 `glm()` 함수를 이용해 `family = binomial` 을 명시해주면 됩니다.

```
stock <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = train,
    family = binomial)
summary(stock)
```

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

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.302  -1.190   1.079   1.160   1.350

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.191213   0.333690   0.573    0.567
Lag1        -0.054178   0.051785  -1.046    0.295
Lag2        -0.045805   0.051797  -0.884    0.377
Lag3         0.007200   0.051644   0.139    0.889
Lag4         0.006441   0.051706   0.125    0.901
Lag5        -0.004223   0.051138  -0.083    0.934
Volume      -0.116257   0.239618  -0.485    0.628

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1383.3  on 997  degrees of freedom
Residual deviance: 1381.1  on 991  degrees of freedom
AIC: 1395.1

Number of Fisher Scoring iterations: 3
```

결과가 매우 형편없습니다. 모델에서 전체적으로 각 변수에 대한 회귀계수들의 z-value가 매우 작고, 이에 대한 p-value 역시 매우 높습니다. 그나마 `Lag1` 의 p-value가 가장 낮구요. 추정된 회귀계수는 음수인데, 이 말은 즉슨 전날의 수익률이 양수이면 오늘 주가지수가 상승할 가능성이 낮다는 이야기입니다. 위 모델을 가지고 예측을 하더라도 좋은 결과는 얻기 힘들 것으로 보입니다.

```
stock_pred <- predict(stock, test, type = "response")
predictedStock <- rep("Down", nrow(test))
predictedStock[stock_pred > 0.5] <- "Up"
table(predictedStock)
```

Down	Up
174	78

```
table(predictedStock, test$Direction)
```

predictedStock/	Down	Up
Down	77	97
Up	34	44

```
accuracy(test$Direction, predictedStock)
```

```
[1] 0.4801587
```

`hatvalues()` : 레버리지 관측치 확인

이 중에서 가장 낮은 p-value를 보이는 네 개의 변수를 선택해서 다시 모델링을 해보도록 하겠습니다.

```
stock2 <- glm(Direction ~ Lag1 + Lag2 + Lag3, data = train, family = binomial)
summary(stock2)
```

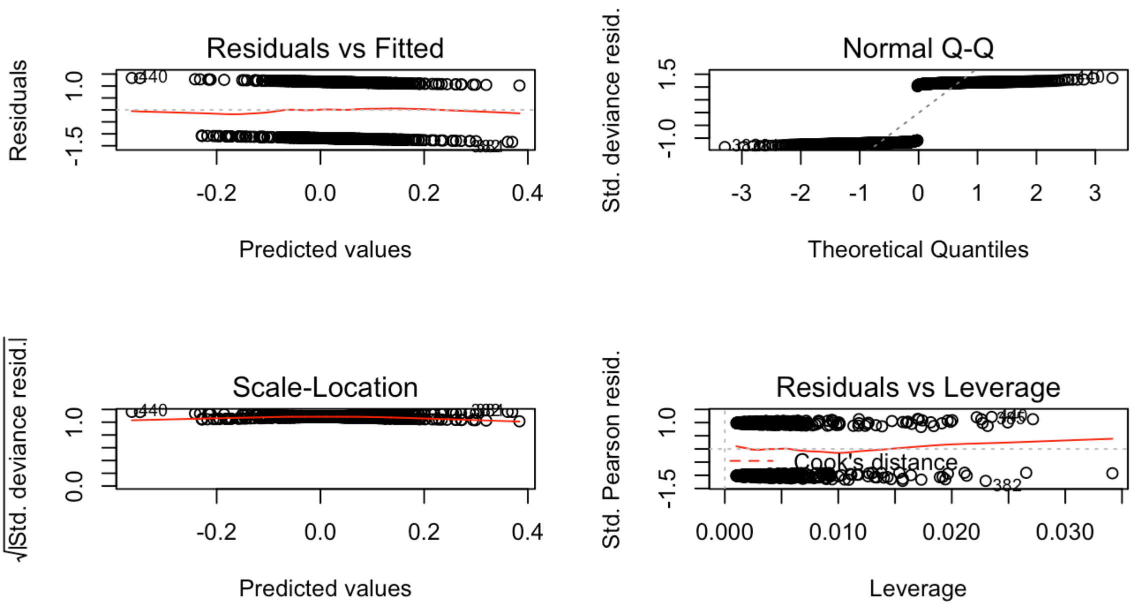
```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3, family = binomial,
    data = train)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.338   -1.189    1.072    1.163    1.335

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.032230   0.063377   0.509    0.611
Lag1        -0.055523   0.051709  -1.074    0.283
Lag2        -0.044300   0.051674  -0.857    0.391
Lag3         0.008815   0.051495   0.171    0.864

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1383.3  on 997  degrees of freedom
Residual deviance: 1381.4  on 994  degrees of freedom
AIC: 1389.4

Number of Fisher Scoring iterations: 3
```

```
stock_pred <- predict(stock2, test, type = "response")
predictedStock2 <- rep("Down", nrow(test))
predictedStock2[stock_pred > 0.5] <- "Up"
accuracy(test$Direction, predictedStock2)
```

```
[1] 0.5912698
```

59.1%의 정확도를 보입니다. 아까보다 10% 가량 좋아졌지만 아직은 부족한 수치죠. 이 모델을 개선하는 방법은 선형회귀법에서 했던 방법들과 크게 다르지 않습니다. 가장 좋은 방법은 레버리지가 관측되는 데이터와 이상치를 삭제하는 것입니다. 우선 모델에 대해서 더 알아보도록 하죠.

```
par(mfrow = c(2, 2))
plot(stock2)
```



마지막 그래프를 통해서 레버리지가 높은 관측치들이 다수 존재하는 것을 알 수 있습니다. 이 점들이 몇 번째 인스턴스인지 확인할 때는 `hatvalues()` 함수를 사용합니다. 이 때 레버리지가 0.025보다 높은 점들을 찾아보겠습니다.

```r
leverage <- hatvalues(stock2)
leveragePoint <- which(leverage > 0.025)

cleanedStock <- train[-leveragePoint, ]
stock3 <- glm(Direction ~ Lag1 + Lag2 + Lag3, data = cleanedStock, family = binomial)
summary(stock3)
```

```


Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3, family = binomial,
    data = cleanedStock)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.329  -1.191   1.080   1.162   1.333

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.03369    0.06347   0.531    0.596
Lag1        -0.04177    0.05264  -0.794    0.427
Lag2        -0.05517    0.05240  -1.053    0.292
Lag3         0.01348    0.05207   0.259    0.796

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1379.1  on 994  degrees of freedom
Residual deviance: 1377.3  on 991  degrees of freedom
AIC: 1385.3

Number of Fisher Scoring iterations: 3
```

```r
stock_pred <- predict(stock3, test, type = "response")
predictedStock3 <- rep("Down", nrow(test))
predictedStock3[stock_pred > 0.5] <- "Up"
accuracy(test$Direction, predictedStock3)
```

```
[1] 0.5833333
```

```r
library(caret)
confusionMatrix(predictedStock2, test$Direction)
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down  Up
      Down   39  31
      Up     72 110

               Accuracy : 0.5913
                 95% CI : (0.5278, 0.6526)
    No Information Rate : 0.5595
    P-Value [Acc > NIR] : 0.1707

                  Kappa : 0.1369
 Mcnemar's Test P-Value : 8.104e-05

            Sensitivity : 0.3514
            Specificity : 0.7801
         Pos Pred Value : 0.5571
         Neg Pred Value : 0.6044
             Prevalence : 0.4405
         Detection Rate : 0.1548
   Detection Prevalence : 0.2778
      Balanced Accuracy : 0.5657

       'Positive' Class : Down
```

```r
confusionMatrix(predictedStock3, test$Direction)
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down  Up
      Down   32  26
      Up     79 115

               Accuracy : 0.5833
                 95% CI : (0.5198, 0.6449)
    No Information Rate : 0.5595
    P-Value [Acc > NIR] : 0.2431

                  Kappa : 0.1095
 Mcnemar's Test P-Value : 3.881e-07

            Sensitivity : 0.2883
            Specificity : 0.8156
         Pos Pred Value : 0.5517
         Neg Pred Value : 0.5928
             Prevalence : 0.4405
         Detection Rate : 0.1270
   Detection Prevalence : 0.2302
      Balanced Accuracy : 0.5519

       'Positive' Class : Down
```

Accuracy는 약 1% 정도 감소했지만, `Up` 인 경우의 예측률이 약 81.6%로 굉장히 높게 측정되었습니다. 이와 반대로 `Down` 인 경우는 예측률이 약 6% 더 낮아졌죠. `Up` 에 대해서는 적절히 일반화되었다고 볼 수 있지만, 반대로 `Down` 인 경우에는 과적합되었다고 생각할 수 있습니다. 실제로 `leveragePoint` 에 해당하는 데이터가 `Down` 이 더 많습니다.