HW9

```
library(dplyr)
library(ROCR)

getMSE <- function(y, yhat){
  MSE <- sum((y-yhat)^2)/length(y)
  return(MSE)
}
```

#1

```
iris_data <- iris %>%
  mutate(Species_binary = c("setosa" = 1, "versicolor" = 0, "virginica" = 0)[Species]) %>%
  select(-Species)
```

Species가 setosa이면 1, 아니면 0으로 정의한 column을 추가하고 기존 Species column을 제외시킨 iris_data라는 data를 만들었다.

다음은 iris_data 중 70%는 train data로, 나머지 30%는 test data로 나누는 작업이다.

```
set.seed(123)
ratio = 0.7
train_index <- caret:: createDataPartition(iris_data$Species_binary, p = ratio, list = FALSE)

train <- iris_data[train_index,]
test <- iris_data[-train_index,]
```

아래 과정을 통해 0과 1값을 갖는 Species를 response, 나머지 변수를 모두 predictor로 하는 multiple logistic regression으로 적합시켰다.

```
fit1 <- glm(Species_binary~., data = train, family = "binomial")

train_probs = predict(fit1, train, type="response")
train_pred = rep(0,nrow(train))
train_pred[train_probs > 0.5] = 1
table(train_pred, train$Species_binary)
```

```
##
## train_pred  0  1
##          0 69  0
##          1  0 36
```

```
mean(train_pred==train$Species_binary)
```

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

threshold를 0.5로 했을 때 train accuracy는 1이 나왔다.

```
test_probs = predict(fit1, test, type = "response")
test_pred = rep(0,nrow(test))
test_pred[test_probs > 0.5] = 1
table(test_pred, test$Species_binary)
```

```
##
## test_pred  0  1
##         0 31  0
##         1  0 14
```

```
mean(test_pred==test$Species_binary)
```

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

threshold를 0.5로 했을 때 test accuracy 또한 1이 나왔다.

```
pred <- prediction(test_pred, test$Species_binary)
auc_ROCR <- performance(pred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]]
auc_ROCR
```

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

auc는 1이 나왔다.

#2

```
data <- read.csv("https://stats.idre.ucla.edu/stat/data/poisson_sim.csv")
data <- within(data, {
  prog <- factor(prog, levels=1:3, labels=c("General", "Academic", "Vocational"))
  id <- factor(id)
})

set.seed(123)
ratio = 0.7
train_index <- caret:: createDataPartition(data$num_awards, p = ratio, list = FALSE)

train <- data[train_index,]
test <- data[-train_index,]
```

1번과 동일하게 7:3 비율로 랜덤하게 train set와 test set로 나누어주었다.

```
fit2 <- glm(num_awards ~ prog + math, family="poisson", train)

train_pred = predict(fit2, train, type="response")
getMSE(train$num_awards, train_pred)
```

```
## [1] 0.6179338
```

num_awards를 response로, prog와 math를 predictor로 하여 multiple possion regression으로 적합하였다.

train set의 MSE는 0.6179338이 나온다.

num_awards는 정수를 가지는 값이므로 train set prediction 값을 반올림을 하여 정수로 나타낸 후 MSE를 다시 구해보았다.

```
train_pred <- round(train_pred)
getMSE(train$num_awards, train_pred)
```

```
## [1] 0.7234043
```

정수값을 가지도록 바꾸었더니 train MSE가 0.7234043이 나왔다.

```
table(train_pred, train$num_awards)
```

```
##
## train_pred  0  1  2  3  4
##          0 67 14  2  1  0
##          1 20 17  3  6  1
##          2  1  2  4  1  1
##          3  0  1  0  0  0
```

```
mean(train_pred==train$num_awards)
```

```
## [1] 0.6241135
```

train accuracy는 0.6241135이다.

```
test_pred = predict(fit2, test, type="response")
getMSE(test$num_awards, test_pred)
```

```
## [1] 1.113296
```

test MSE는 1.113296이 나왔다.

test set prediction 또한 반올림을 통해 정수로 나타내고 MSE를 다시 구해보았다.

```
test_pred <- round(test_pred)
getMSE(test$num_awards, test_pred)
```

```
## [1] 1.169492
```

test MSE가 1.169492이 나왔다.

```
table(test_pred, test$num_awards)
```

```
##
## test_pred  0  1  2  3  5  6
##         0 26  8  0  0  0  0
##         1  9  7  3  1  1  0
##         2  1  0  1  0  1  1
```

```
mean(test_pred==test$num_awards)
```

```
## [1] 0.5762712
```

test accuracy는 0.5762712가 나왔다.