HW 7

Enter your name and EID here: Jongho Yoo (jy23294)

You will submit this homework assignment as a pdf file on Gradescope.

For all questions, include the R commands/functions that you used to find your answer (show R chunk). Answers without supporting code will not receive credit. Write full sentences to describe your findings.

We will use the packages `tidyverse`, `plotROC`, and `caret` for this assignment.

```
# Load packages
library(tidyverse)
library(plotROC)
library(caret)
```

Back to the Pokemon dataset!

Question 1: (2 pts)

Let's re-upload the data to start from fresh and recode the variable `Legendary` as 0 if a pokemon is not legendary and as 1 if it is:

```
# Upload data from GitHub
pokemon <- read_csv("https://raw.githubusercontent.com/laylaguyot/datasets/main//pokemon.csv") %>%
  mutate(Legendary = ifelse(Legendary == TRUE, 1, 0))

# Take a look
head(pokemon)
```

```
## # A tibble: 6 x 13
##   Number Name   Type1 Type2 Total    HP Attack Defense SpAtk SpDef Speed Gener~1
##    <dbl> <chr>  <chr> <chr> <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1      1 Bulba~ Grass Pois~   318    45     49      49    65    65    45       1
## 2      2 Ivysa~ Grass Pois~   405    60     62      63    80    80    60       1
## 3      3 Venus~ Grass Pois~   525    80     82      83   100   100    80       1
## 4      3 Venus~ Grass Pois~   625    80    100     123   122   120    80       1
## 5      4 Charm~ Fire  <NA>    309    39     52      43    60    50    65       1
## 6      5 Charm~ Fire  <NA>    405    58     64      58    80    65    80       1
## # ... with 1 more variable: Legendary <dbl>, and abbreviated variable name
## #   1: Generation
```

In the last assignment, you tried linear and logistic regression and (hopefully) found that these two models had a similar performance which was alright (AUC ~ 0.86). Let's see how a logistic regression would be able to predict the `Legendary` status of "new" pokemons using a 10-fold cross-validation:

```r
# define k
k = 10

# Randomly order rows in the dataset
data <- pokemon[sample(nrow(pokemon)), ]

# Create k folds from the dataset (break rows into k parts)
folds <- cut(seq(1:nrow(data)), breaks = k, labels = FALSE)

# Initialize a vector to keep track of the performance
perf_k <- NULL

# Use a for loop to get diagnostics for each test set
for(i in 1:k){
  # Create train and test sets
  train_not_i <- data[folds != i, ] # all observations except in fold i
  test_i <- data[folds == i, ]  # observations in fold i

  # Train model on train set (all but fold i)
  pokemon_log <- glm(Legendary ~ HP + Attack, data = train_not_i, family = "binomial")

  # Test model on test set (fold i)
  predict_i <- data.frame(
    predictions = predict(pokemon_log, newdata = test_i, type = "response"),
    outcome = test_i$Legendary)

  # Consider the ROC curve for the test dataset
  ROC <- ggplot(predict_i) +
    geom_roc(aes(d = outcome, m = predictions))

  # Get diagnostics for fold i (AUC)
  perf_k[i] <- calc_auc(ROC)$AUC
}

# get value
mean(perf_k)
```

```
## [1] 0.8627348
```

How does the average AUC compare to the AUC of our `pokemon_log` model trained on the entire data? What does it indicate about the logistic regression model?

The cross validation gave a value of 0.87. This value is very close to the previous AUC value of 0.86, indicating that the average AUC is very accurate to the AUC of our 'pokemon_log' model trained on the entire data.

Question 2: (3 pts)

Another classifier we can consider to predict `Legendary` status from `HP` and `Attack` is using the k-nearest neighbors (kNN). Fit the kNN model with 5 nearest neighbors and call it `pokemon_kNN`. What does this model predict for each pokemon (i.e., what output do we get when using the function `predict()`)?

```r
# create kNN model and predict
pokemon_kNN <- knn3(Legendary ~ HP + Attack, data = pokemon, k = 5)
predict(pokemon_kNN, pokemon) %>% as.data.frame %>% head
```

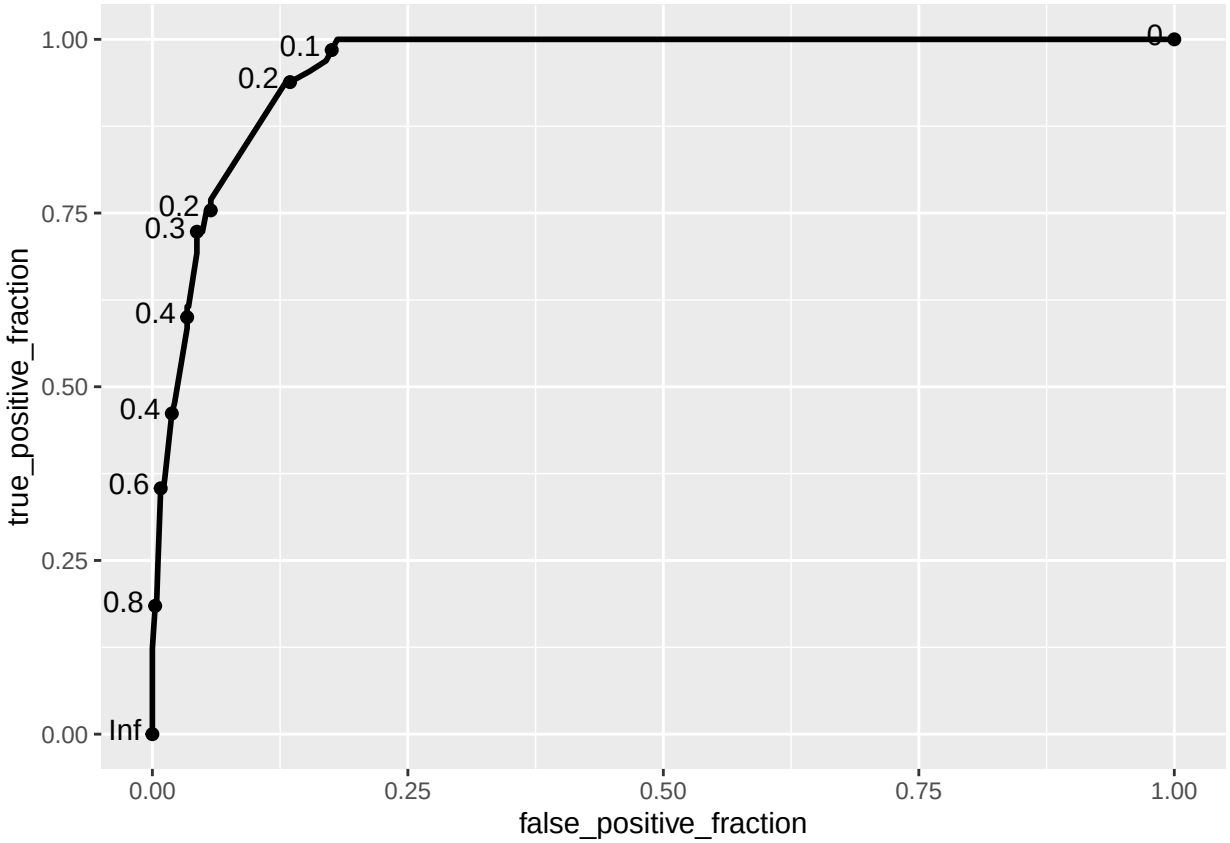
```
##           0         1
## 1 1.0000000 0.0000000
## 2 1.0000000 0.0000000
## 3 0.8888889 0.1111111
## 4 0.6666667 0.3333333
## 5 1.0000000 0.0000000
## 6 1.0000000 0.0000000
```

In each of the 5 nearest neighbors, it shows the proportions of legendary and not legendary pokemon. For example, in the first kNN, all pokemon were non legendary.

Question 3: (3 pts)

Use the `pokemon_kNN` model to build a ROC curve and compute the AUC. How well is the model performing according to the AUC?

```r
# build ROC curve
ROC <- pokemon %>%
  mutate(predictions = predict(pokemon_kNN, pokemon)[,2]) %>%
  ggplot() +
  geom_roc(aes(d = Legendary, m = predictions), n.cuts = 10)
ROC
```




```
# get AUC value
calc_auc(ROC)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9600837
```

The AUC value of 96% indicates that this kNN model performs very well.

Question 4: (4 pts)

You should find that the `pokemon_kNN` model performs pretty well! Much better than the logistic regression anyway. Perform a 10-fold cross-validation with the `pokemon_kNN` model using the same folds as defined in the first question.

```
# define k
k = 10

# Randomly order rows in the dataset
data <- pokemon[sample(nrow(pokemon)), ]

# Create k folds from the dataset (break rows into k parts)
folds <- cut(seq(1:nrow(data)), breaks = k, labels = FALSE)
```

```r
# Initialize a vector to keep track of the performance
perf_k <- NULL

# Use a for loop to get diagnostics for each test set
for(i in 1:k){
  # Create train and test sets
  train_not_i <- data[folds != i, ] # all observations except in fold i
  test_i <- data[folds == i, ]  # observations in fold i

  # Train model on train set (all but fold i)
  pokemon_kNN <- knn3(Legendary ~ HP + Attack, data = train_not_i, k = 5)

  # Test model on test set (fold i)
  predict_i <- data.frame(
    predictions = predict(pokemon_kNN, newdata = test_i)[,2],
    outcome = test_i$Legendary)

  # Consider the ROC curve for the test dataset
  ROC <- ggplot(predict_i) +
    geom_roc(aes(d = outcome, m = predictions))

  # Get diagnostics for fold i (AUC)
  perf_k[i] <- calc_auc(ROC)$AUC
}

# get value
mean(perf_k)
```

```
## [1] 0.8079364
```

Do you see a real decrease in AUC when predicting `Legendary` status on "new" data? What does it indicate about our model?

Yes, the AUC value slightly decreases from 0.867 to 0.797, indicating that this new data is not as good at predicting legendary status.

Question 5: (3 pts)

Let's focus on the `pokemon_kNN` model trained on a random 9/10 of the data and then tested on the remaining 1/10. We plot the decision boundary: the blue boundary classifies points inside of it as *Legendary* and points outside as *Not Legendary*. Locate where the false positive cases and the false negative cases are (indicate if they are inside/outside the decision boundary and what they mean).

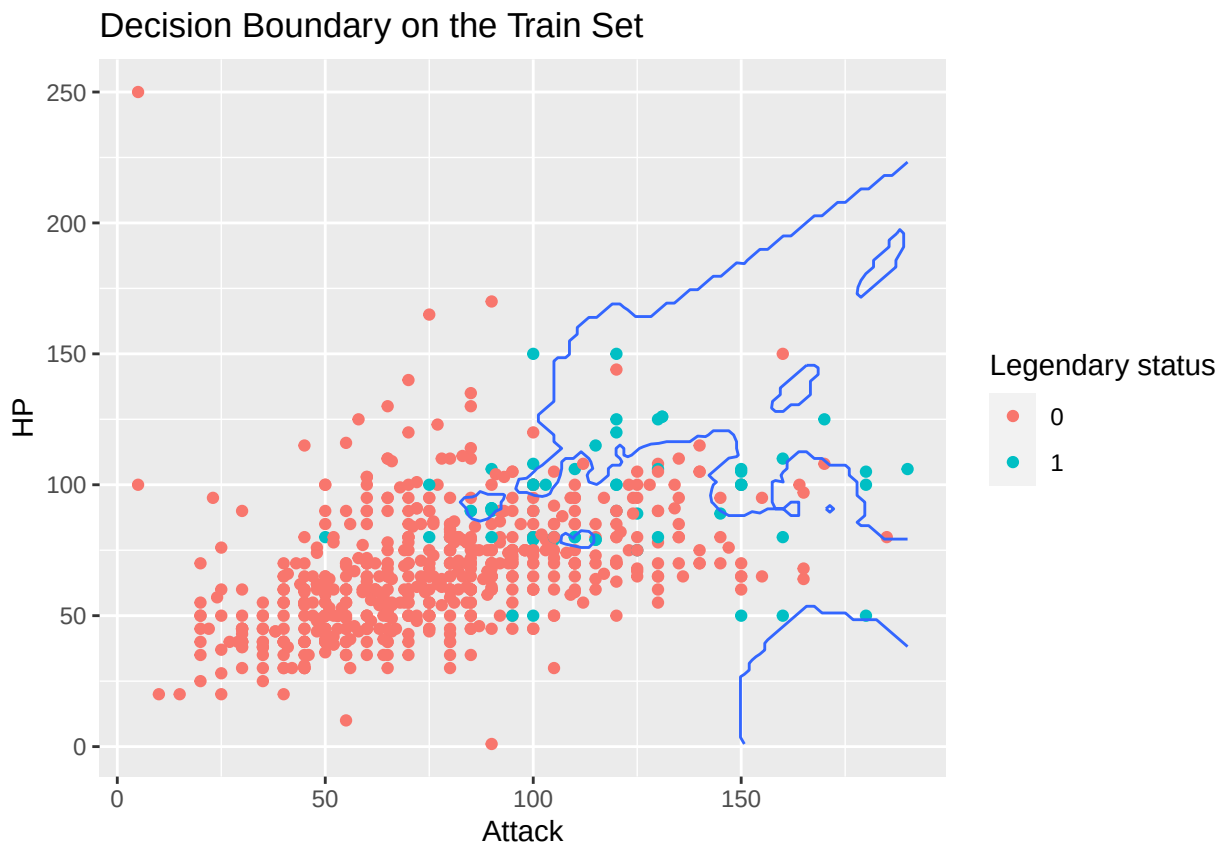
```r
# Make this example reproducible by setting a seed
set.seed(322)

# Split data into train and test sets
train <- pokemon %>% sample_frac(0.9)
test <- pokemon %>% anti_join(train, by = "Name")
```

```r
# Fit the model on the train data
pokemon_kNN <- knn3(Legendary ~ Attack + HP,
                    data = train,
                    k = 5)

# Make a grid for the graph to layout the contour geom
grid <- data.frame(expand.grid(Attack = seq(min(pokemon$Attack),
                                            max(pokemon$Attack),
                                            length.out = 100),
                               HP = seq(min(pokemon$HP),
                                        max(pokemon$HP),
                                        length.out = 100)))

# Use this grid to predict legendary status
grid %>%
  mutate(p = predict(pokemon_kNN, grid)[,2]) %>%
  ggplot(aes(Attack, HP)) +
  # Only display data in the train set
  geom_point(data = train,
             aes(Attack, HP, color = as.factor(Legendary))) +
  # Draw the decision boundary
  geom_contour(aes(z = p), breaks = 0.5) +
  # Labels
  labs(title = "Decision Boundary on the Train Set",
       color = "Legendary status")
```

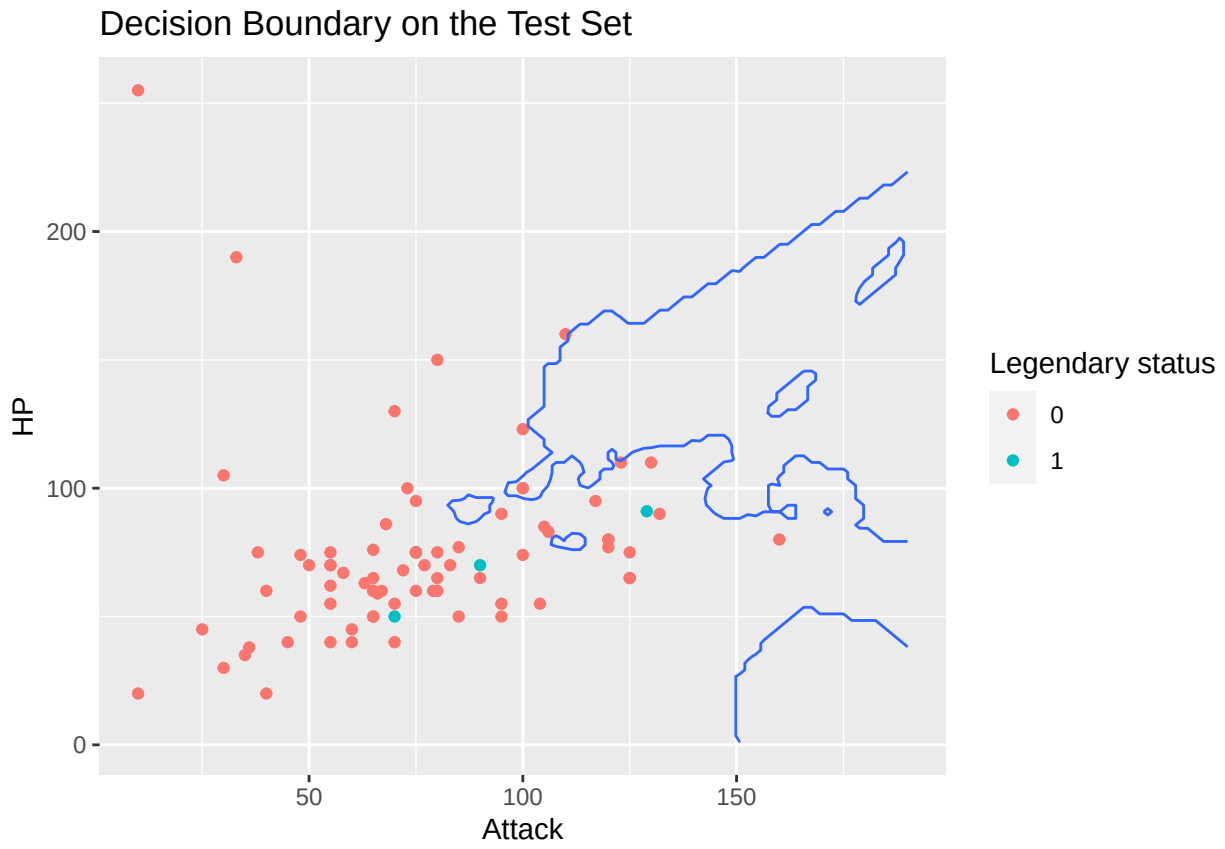


The regions inside the blue borders are classified as legendary, while the regions outside the blue borders are classified as non-legendary. Therefore, any red dots within the blue border are false positives, and any blue dots outside the blue borders are false negatives. False positive means that they were classified as legendary, but actually were non-legendary. Conversely, false negatives were classified as non-negative, but actually were legendary.

Question 6: (3 pts)

Now, represent the same decision boundary but with the test set. *Hint: use the last piece of the code from the previous question.*

```
# decision boundary with test set
grid %>%
  mutate(p = predict(pokemon_kNN, grid)[,2]) %>%
  ggplot(aes(Attack, HP)) +
  # Only display data in the train set
  geom_point(data = test,
             aes(Attack, HP, color = as.factor(Legendary))) +
  # Draw the decision boundary
  geom_contour(aes(z = p), breaks = 0.5) +
  # Labels
  labs(title = "Decision Boundary on the Test Set",
       color = "Legendary status")
```



Comparing the decision boundary for the train set and for the test set, describe why the kNN model might not perform very well on the test set.

kNN sees which groups are present near the data point in question, and classifies it according to the group with the most points near the data point. In this test set, most points are non-legendary, and only 3 data points are legendary. Therefore, when using kNN, whichever data points you look at or however you group them, there will be more non-legendary cases causing all points to be classified as non-legendary.

Formatting: (2 pts)

Comment your code, write full sentences, and knit your file!

```
##                                                                                                                  sysn
##                                                                                                                "Darw
##                                                                                                                 relo
##                                                                                                                "21.3
##                                                                                                                 vers
## "Darwin Kernel Version 21.3.0: Wed Jan  5 21:37:58 PST 2022; root:xnu-8019.80.24~20/RELEASE_ARM64_T8
##                                                                                                              noden
##                                                                                  "Stevens-MacBook-Pro-2.loc
##                                                                                                                 mach
##                                                                                                                 "arm
##                                                                                                                   l
##                                                                                                                  "r
##                                                                                                                    
##                                                                                                     "steveny
##                                                                                                   effective_u
##                                                                                                     "steveny
```