HW 6

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` and `plotROC` for this assignment.

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

Question 1: (4 pts)

We will use the `pokemon` dataset for this assignment:

```
# Upload data from GitHub
pokemon <- read_csv("https://raw.githubusercontent.com/laylaguyot/datasets/main//pokemon.csv")

# Take a look
head(pokemon)
```

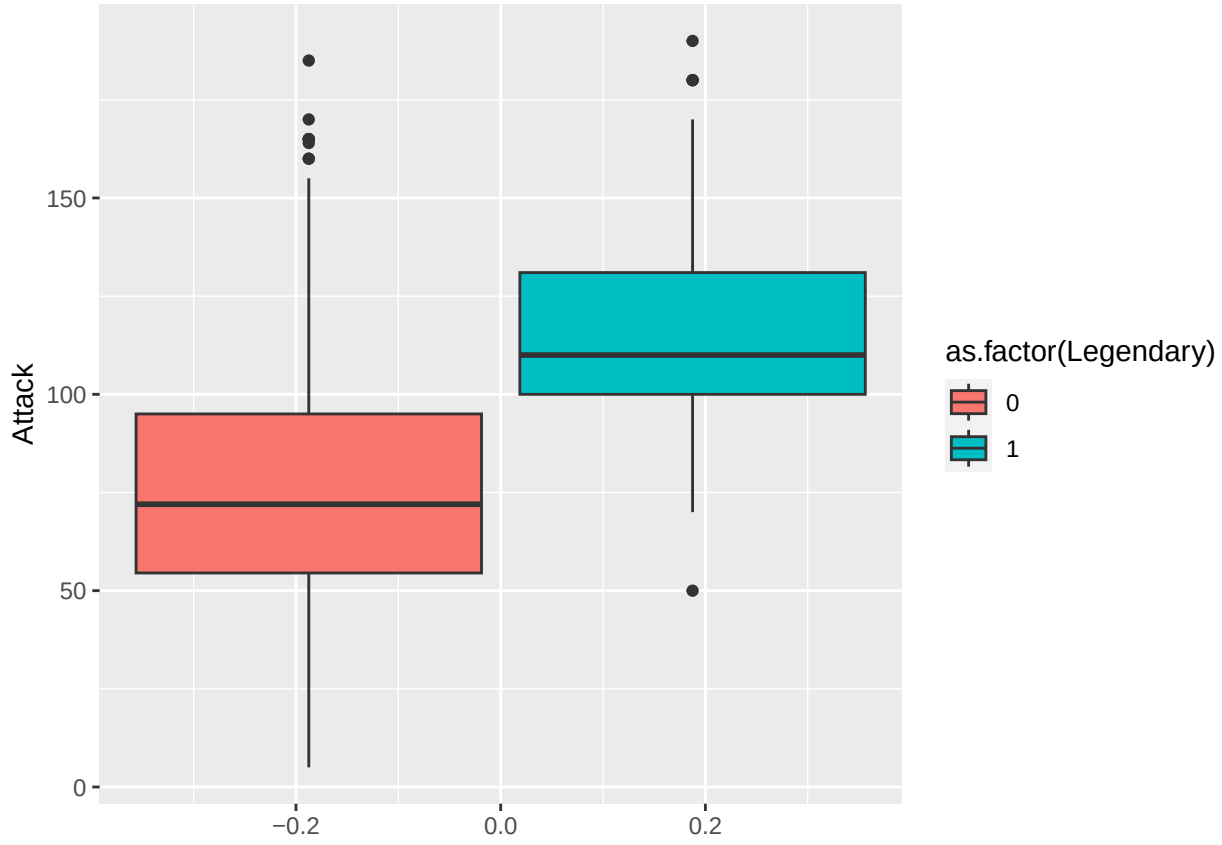
```
## # A tibble: 6 x 13
##   Number Name   Type1 Type2 Total    HP Attack Defense SpAtk SpDef Speed Gener~1
##    <dbl> <chr>  <chr> <chr> <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1      1 Bulba~ Grass Pois~   318    45     49      49    65    65    45       1
## 2      2 Ivysa~ Grass 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 <lgl>, and abbreviated variable name
## #   1: Generation
```

Recode the variable `Legendary`, taking a value of 1 if a pokemon is legendary and a value of 0 if it is not. Save the resulting data as `my_pokemon`.

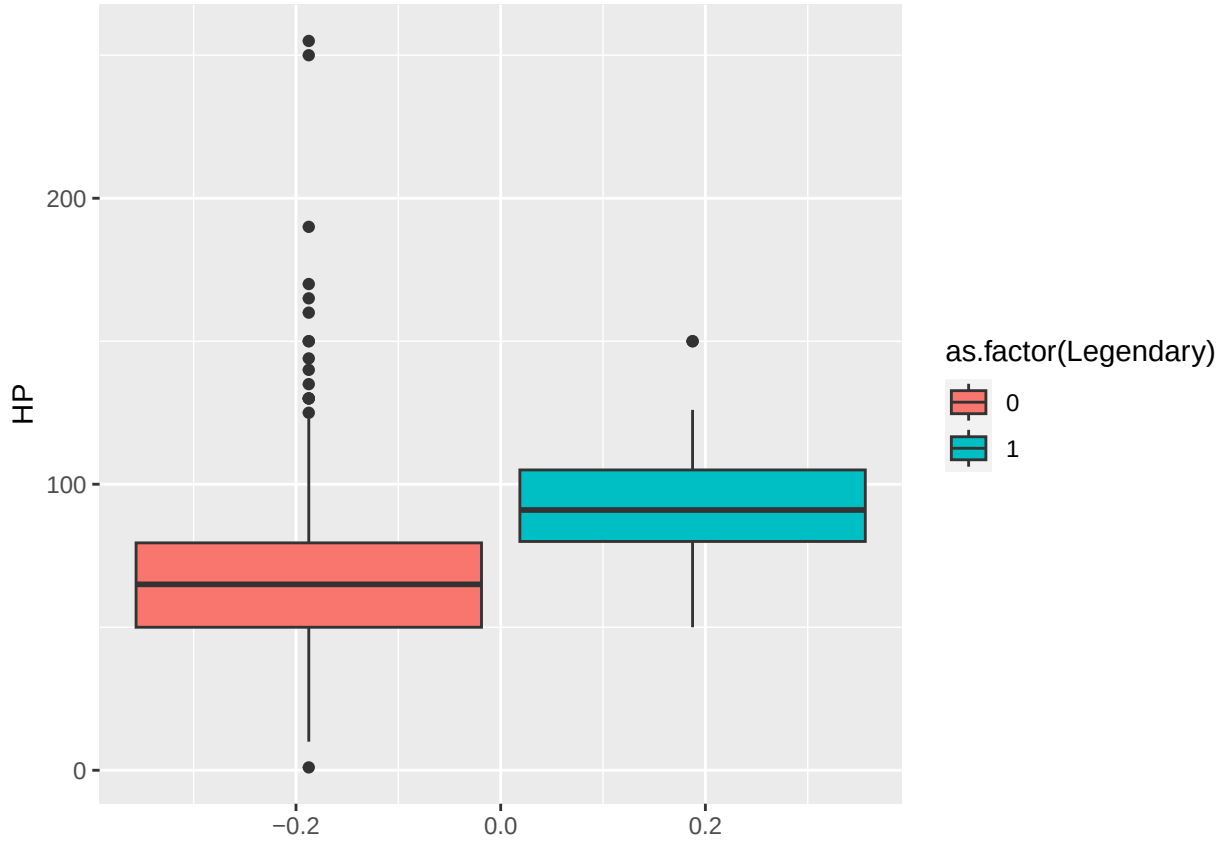
```
# recode Legendary status as binary
my_pokemon <- pokemon %>%
  mutate("Legendary" = ifelse(Legendary == FALSE, 0, 1))
```

Let's visualize how the features of `Attack` and `HP` impact the legendary status. First, visualize the distribution of `Attack` for legendary pokemons vs those that are not. Also visualize the distribution of `HP` for these two groups. *Note: consider the binary variable as a factor for your* ***`ggplot`*** *using* ***`as.factor()`****.* Comment with what you see in these visualizations.

```
# boxplot of Attack and HP based on Legendary status
my_pokemon %>%
  ggplot(aes(y = Attack, fill = as.factor(Legendary))) +
  geom_boxplot()
```



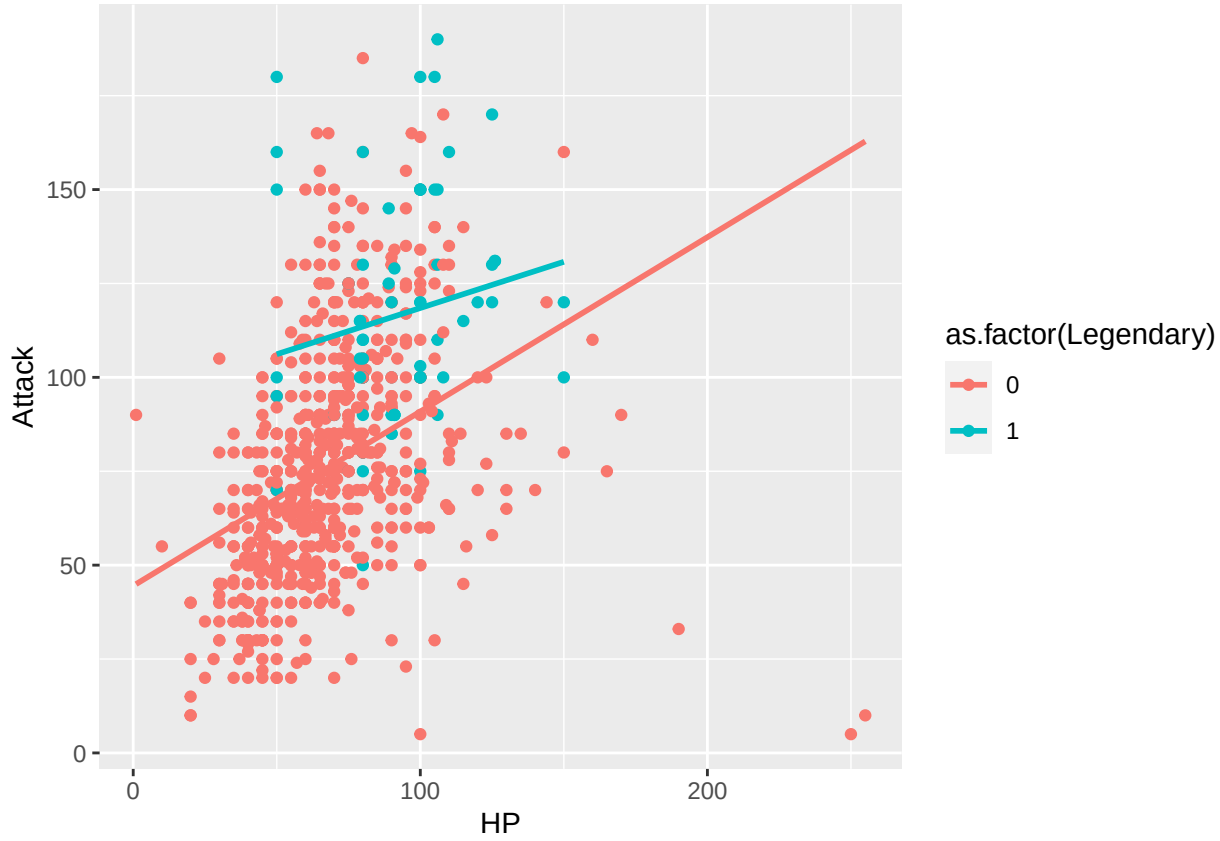

```
my_pokemon %>%
  ggplot(aes(y = HP, fill = as.factor(Legendary))) +
  geom_boxplot()
```



The boxplots showed that legendary pokemon, on average, had higher attack and HP compared to non-legendary pokemon. In addition, the variation of attack and HP was greater in non-legendary pokemon, with the variation in HP being significantly greater.

Then visualize the linear relationship between `Attack` and `HP` (hit points) for each legendary status. *Hint: color the regression lines.* Do `Attack` and `HP` seem to predict Legendary status? Comment with what you see in this visualization.

```
# linear relationship between attack and HP per legendary status, with regression lines
my_pokemon %>%
  ggplot(aes(x = HP, y = Attack, color = as.factor(Legendary))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



Both legendary and non-legendary pokemon have a positive relationship between attack and HP. However, by looking at the regression lines, non-legendary pokemon has a stronger positive relationship than legendary pokemon, which also has a positive relationship, but to a lesser degree.

Question 2: (2 pt)

Let's predict `Legendary` status using a linear regression model with `Attack` and `HP` in `my_pokemon`. Fit this model, call it `pokemon_lin`, and write its equation.

```
# linear regression model
pokemon_lin <- lm(Legendary ~ Attack + HP, data = my_pokemon)

# summary of model
summary(pokemon_lin)
```

```
##
## Call:
## lm(formula = Legendary ~ Attack + HP, data = my_pokemon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.40650 -0.12385 -0.05025  0.01914  0.97201
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.2201775  0.0289417  -7.608 7.88e-14 ***
## Attack       0.0023563  0.0003054   7.715 3.61e-14 ***
## HP           0.0016644  0.0003882   4.288 2.03e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.254 on 797 degrees of freedom
## Multiple R-squared:  0.1392, Adjusted R-squared:  0.137
## F-statistic: 64.42 on 2 and 797 DF,  p-value: < 2.2e-16
```

$\widehat{Legendary} = (0.0023563 * Attack) + (0.0016644 * HP) - 0.2201775$

Question 3: (3 pts)

Choose a pokemon whose name starts with the same letter as yours. Take a look at its stats and, using the equation of your model from the previous question, predict the legendary status of this pokemon, "by hand":

```
# filter for Squirtle
my_pokemon %>%
  filter(Name == "Squirtle")
```

```
## # A tibble: 1 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      7 Squir~ Water <NA>    314    44     48      65    50    64    43       1
## # ... with 1 more variable: Legendary <dbl>, and abbreviated variable name
## #   1: Generation
```

```
# calculate legendary prediction value for squirtle using equation
(0.0023563 * 48) + (0.0016644 * 44) - 0.2201775
```

```
## [1] -0.0338415
```

Check your answer by using `predict()` with the argument `newdata =`:

```
# use 'predict' to automatically calculate legendary prediction value
squirtleData <- my_pokemon %>%
  filter(Name == "Squirtle") %>%
  data.frame()
predict(pokemon_lin, newdata = squirtleData)
```

```
##           1
## -0.03383984
```

Was your pokemon predicted to be legendary? Why or why not? Does it match the reality?

My pokemon was not predicted to be legendary. Legendary pokemon have a 'Legendary' value of 1, while the non-legendary have a value of 0. My pokemon had a predicted value of -0.03, which is around 0. Therefore, my pokemon is not predicted to be legendary, which matches reality, because Squirtle is not a legendary pokemon.

Question 4: (2 pts)

We can measure how far off our predictions are from reality with residuals. Use `resid()` to find the residuals of each pokemon in the dataset then find the sum of all residuals. Why does it make sense?

```
# find residuals for the linear model
my_pokemon %>%
  mutate(residuals = resid(pokemon_lin)) %>%
  summarize(sum(residuals))
```

```
## # A tibble: 1 x 1
##   `sum(residuals)`
##              <dbl>
## 1         2.78e-15
```

The sum of all residuals was 2.78e-15, which is basically 0. When you fit the linear line, half the points are above the line, and half are below the line. Therefore, when you find the difference between points above and below the line (residual), the sum will be 0.

Question 5: (2 pts)

A logistic regression would be more appropriate to predict `Legendary` status since it can only take two values. Fit this new model with `Attack` and `HP`, call it `pokemon_log`, and write its equation. *Hint: the logit form is given by the R output.*

```
# create logarithmic model of legendary prediction status based on attack and HP
pokemon_log <- glm(Legendary ~ Attack + HP, data = my_pokemon, family = "binomial")
summary(pokemon_log)
```

```
##
## Call:
## glm(formula = Legendary ~ Attack + HP, family = "binomial", data = my_pokemon)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8418  -0.3693  -0.2204  -0.1334   2.8555
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.659078   0.680595 -11.253  < 2e-16 ***
```

```
## Attack       0.032901   0.004431   7.425 1.12e-13 ***
## HP           0.025923   0.004982   5.203 1.96e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 450.90  on 799  degrees of freedom
## Residual deviance: 340.34  on 797  degrees of freedom
## AIC: 346.34
##
## Number of Fisher Scoring iterations: 6
```

$$\hat{p} = \frac{e^{-7.659078+(0.032901*Attack)+(0.025923*HP)}}{1+e^{-7.659078+(0.032901*Attack)+(0.025923*HP)}}$$

Question 6: (2 pts)

According to this new model, is the pokemon you chose in question 3 predicted to be legendary? Why or why not? *Hint: you can use predict() with the arguments* ***newdata =*** *and* ***type = "response"****.*

```
# calculate legendary prediction value for squirtle using log log model
squirtleDataLog <- my_pokemon %>%
  filter(Name == "Squirtle") %>%
  data.frame()
predict(pokemon_log, newdata = squirtleData, type = "response")
```

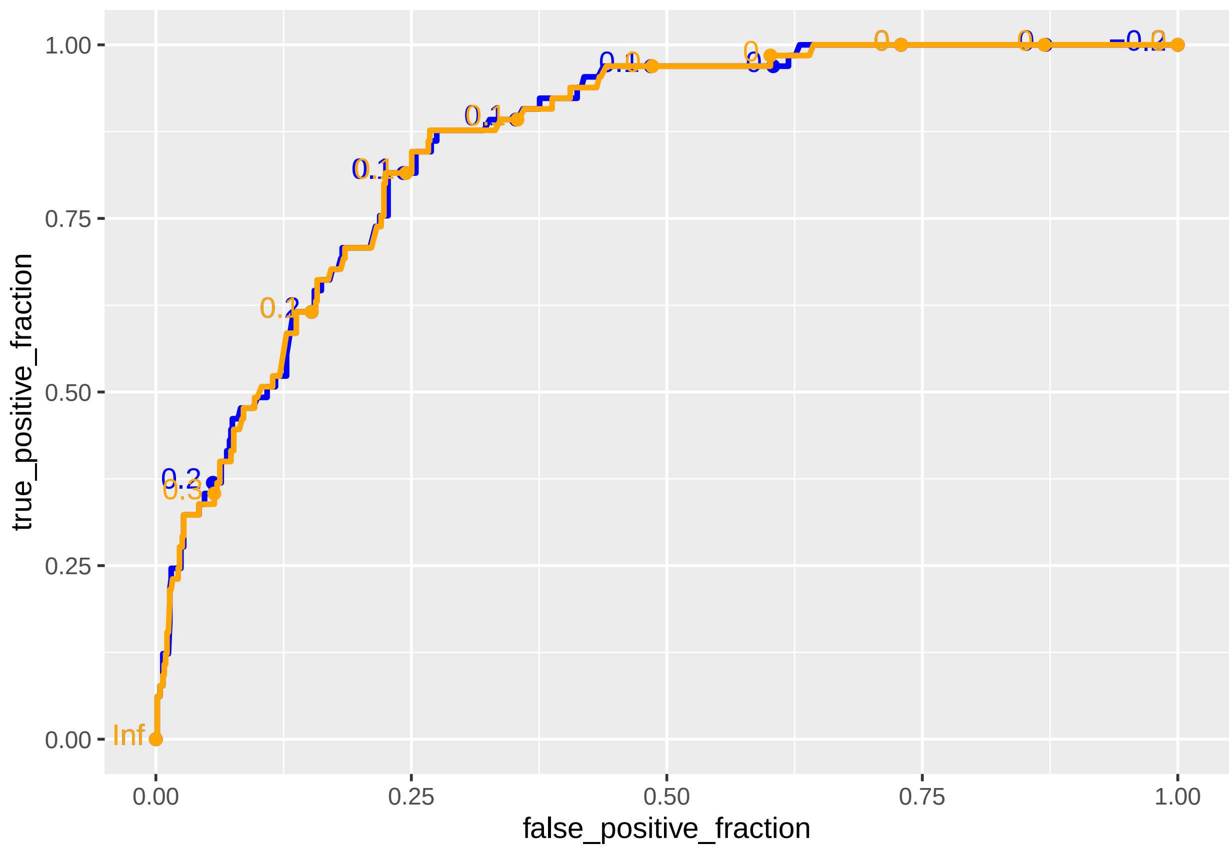
```
##           1
## 0.007109134
```

My pokemon was also predicted to not be legendary according to this new model. This new model gave a predicated 'Legendary' value of 0.007109134. Since this is approximately 0, it would mean the prediction is non-Legendary.

Question 7: (3 pts)

Let's compare the performance of these two models using ROC curves. On the same plot, represent the ROC curve for predicting `Legendary` status based on the predictions from the linear regression in blue and another ROC curve based on the predictions from the logistic regression in orange.

```
# ROC plots for linear and log models
ROC <- my_pokemon %>%
  mutate(predictionsLin = predict(pokemon_lin, type = "response")) %>%
  mutate(predictionsLog = predict(pokemon_log, type = "response")) %>%
  ggplot() +
  geom_roc(aes(d = Legendary, m = predictionsLin), color = "blue", n.cuts = 10) +
  geom_roc(aes(d = Legendary, m = predictionsLog), color = "orange", n.cuts = 10)
ROC
```



How do these two models compare?

The ROC curves of the two models are very similar. This indicates that both models predict Legendary status relatively similarly, and one model is not particularly better than the other.

Formatting: (2 pts)

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

```
##                                                                                              sysn
##                                                                                             "Darw
##                                                                                              rele
##                                                                                             "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_T81
##                                                                                             noden
##                                                                                   "Stevens-MBP-2.l
##                                                                                              mach
##                                                                                              "arm
##                                                                                                lo
```

```
##                                                                                           "re
##                                                                                             u
##                                                                                       "steveny
##                                                                                     effective_u
##                                                                                       "steveny
```