HW 8

Enter your name and EID here: Harini Shanmugam

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 = 0.8581). 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
# Make this example reproducible by setting a seed
set.seed(322)

# Choose number of folds
k = 10

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

# Create k folds from the dataset
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 <- data[folds != i, ] # all observations except in fold i
  test <- data[folds == i, ]  # observations in fold i

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

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

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

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

# Average performance
mean(perf_k)
```

```
## [1] 0.8561505
```

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 average AUC from our past assignment was 0.8581. The AUC of our pokemon_log model trained on the entire data is 0.8561505. So the logistic regression model performs at the same level as our past models and is also equally as good.

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()`)?

```
# Consider the kNN classifier with k = 5
pokemon_kNN <- knn3(Legendary ~ HP+Attack,
                data = pokemon,
                k = 5) # number of neighbors
pokemon_kNN
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.08125 0.00000 1.00000
```

```
# Find the proportion of nearest neighbors with Legendary status
predict(pokemon_kNN, pokemon, type = "prob")[,2]
```

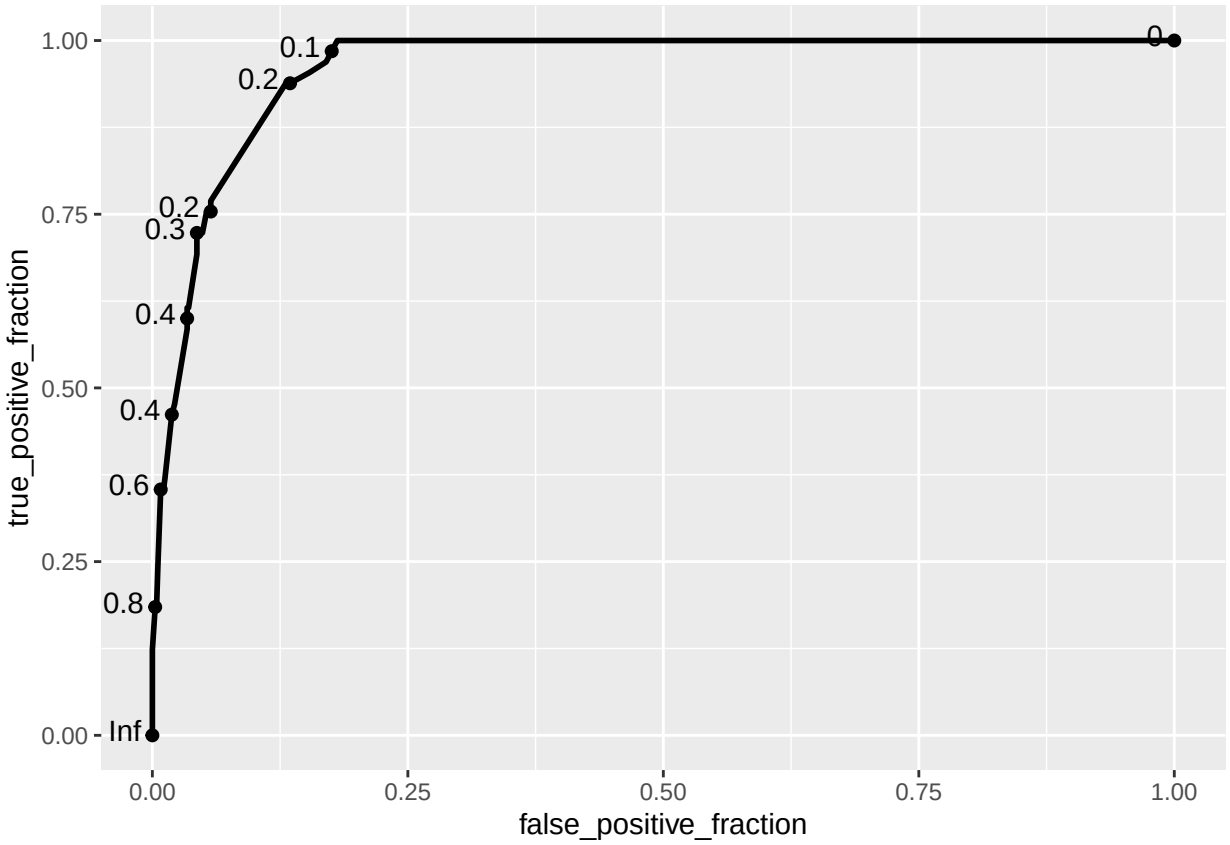
```
##   [1] 0.0000000 0.0000000 0.1111111 0.3333333 0.0000000 0.0000000 0.0000000
##   [8] 0.2000000 0.2857143 0.0000000 0.0000000 0.0000000 0.2857143 0.0000000
##  [15] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [22] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [29] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [36] 0.0000000 0.5714286 0.0000000 0.0000000 0.2500000 0.0000000 0.0000000
##  [43] 0.0000000 0.1666667 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [50] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [57] 0.0000000 0.0000000 0.0000000 0.0000000 0.1111111 0.0000000 0.0000000
##  [64] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [71] 0.0000000 0.0000000 0.0000000 0.3333333 0.2000000 0.0000000 0.0000000
##  [78] 0.3333333 0.0000000 0.0000000 0.0000000 0.1666667 0.0000000 0.0000000
##  [85] 0.0000000 0.0000000 0.1000000 0.1000000 0.0000000 0.0000000 0.0000000
##  [92] 0.0000000 0.0000000 0.0000000 0.0000000 0.2000000 0.5000000 0.0000000
##  [99] 0.2500000 0.0000000
##  [ reached getOption("max.print") -- omitted 700 entries ]
```

The model predicts, for each pokemon, if it is legendary or not. Specifically, it gives us a decimal between 0 (meaning not legendary) and 1 (meaning legendary). So the closer the decimal is to 1, or if it is greater than 0.5, then the model predicts that that specific pokemon is 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?

```
# ROC curve
ROC <- ggplot(pokemon) +
  geom_roc(aes(d = Legendary, m = predict(pokemon_kNN, pokemon, type = "prob")[,2]),
           n.cuts = 10)
ROC
```




```
# Calculate the area under the curve
calc_auc(ROC)$AUC
```

```
## [1] 0.9600837
```

The AUC of the model is 0.9600837, which means the pokemon_kNN model is performing exceptionally 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.

```r
# Choose number of folds
k = 5

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

# Create k folds from the dataset
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 <- data[folds != i, ] # all observations except in fold i
  test <- data[folds == i, ]  # observations in fold i

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

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

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

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

# AUC
mean(perf_k)
```

```
## [1] 0.8386479
```

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

Yes, there is a fair amount of decrease in AUC when predicting `Legendary` status on "new" data. This indicates that our model is overfitting.

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).

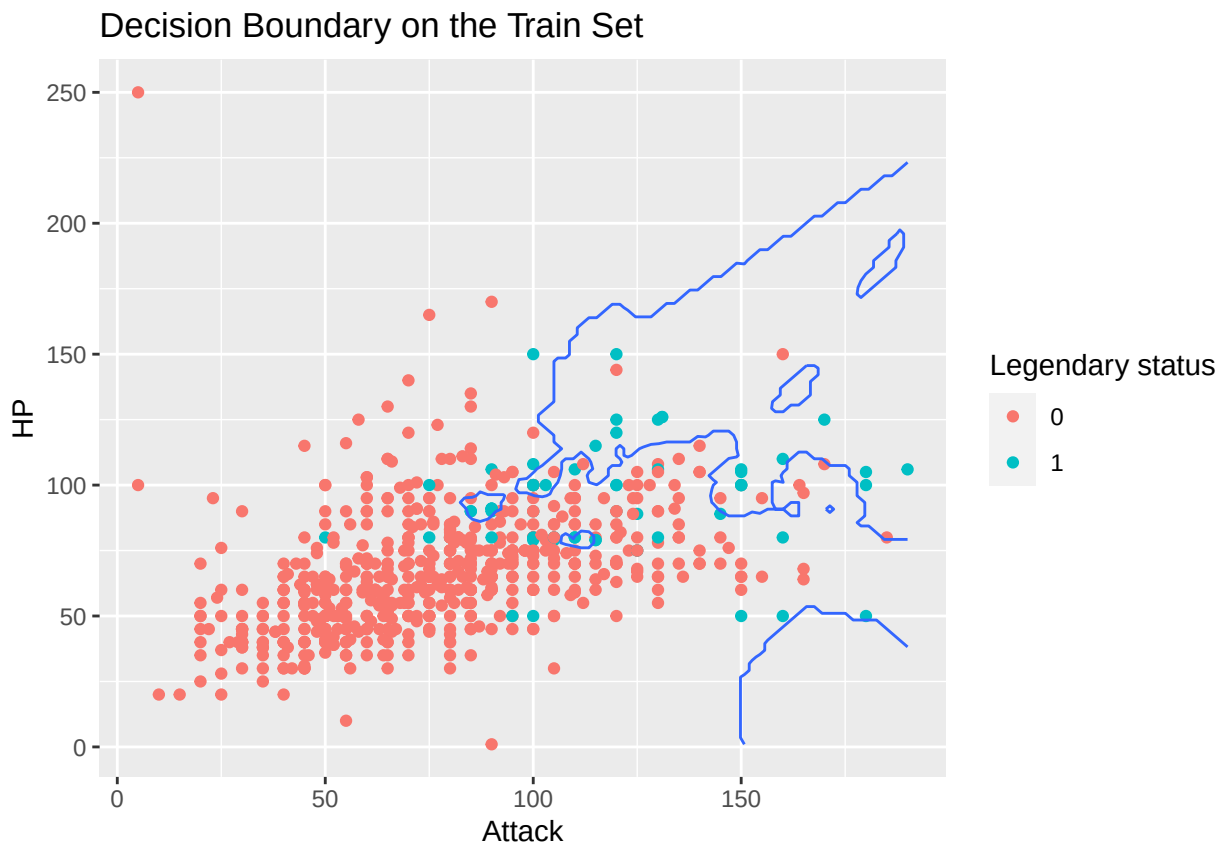
```
# Make this example reproducible by setting a seed
set.seed(322)

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

# 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 false positive cases are the handful of red dots to the right/inside of the decision boundary. This means that the model predicted that those pokemon are legendary, but in reality they are not

The false negative cases are the handful of blue dots in the middle and left/outside of the decision boundary. This means that the model predicted that those pokemon are not legendary, but in reality they are

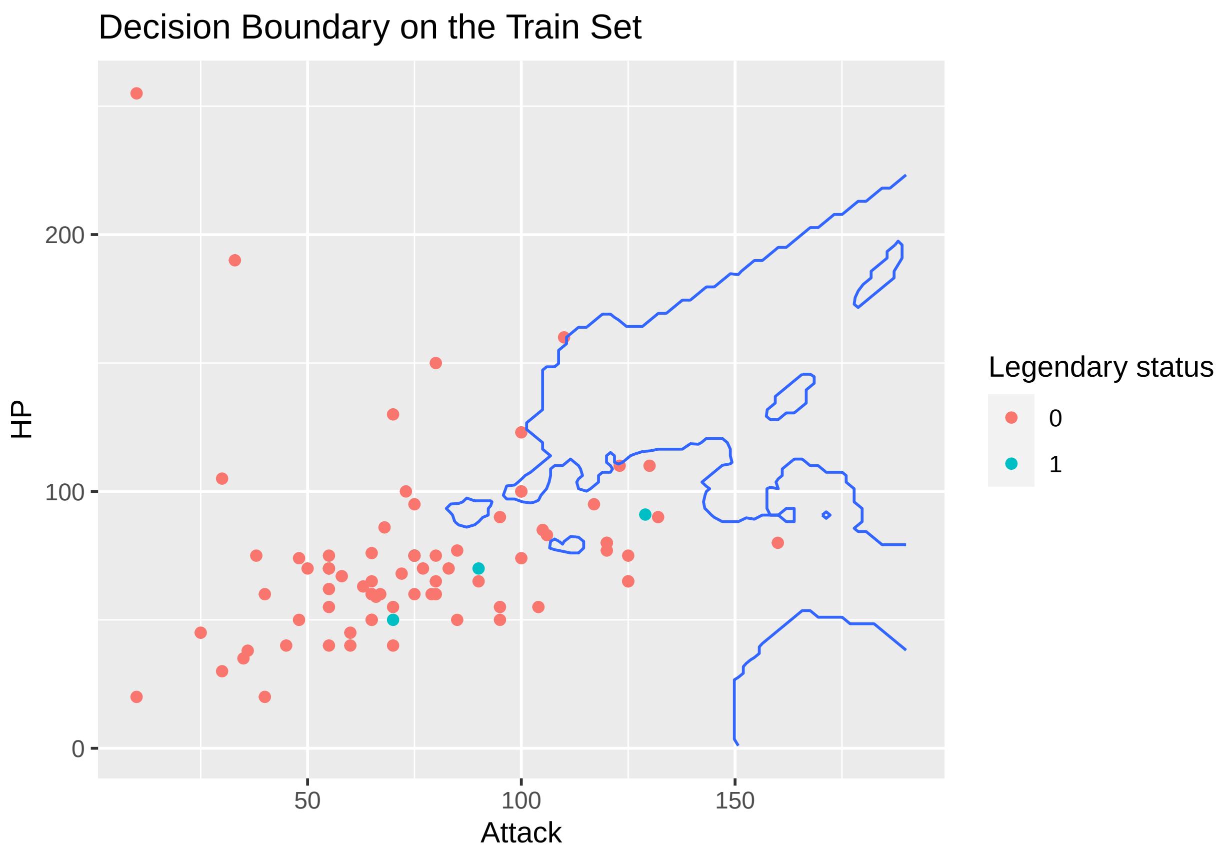
---

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.*

```
# Use this grid to predict legendary status using test set
grid %>%
  mutate(p = predict(pokemon_kNN, grid)[,2]) %>%
  ggplot(aes(Attack, HP)) +
  # Only display data in the test 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 Train 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.

The kNN model might not have performed very well on the test set because it might have overfitted to the data points in the training set. Essentially the model would have memorized the training set and tried to fit the test set the same way even though the sample being tested is different.

Formatting: (2 pts)

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

```
##                                                                                                                      sysna
##                                                                                                                     "Darwi
##                                                                                                                      relea
##                                                                                                                     "21.6
```

```
##                                                                                      versi
## "Darwin Kernel Version 21.6.0: Wed Aug 10 14:28:35 PDT 2022; root:xnu-8020.141.5~2/RELEASE_ARM64_T81(
##                                                                                      nodena
##                                                                       "Harinis-Air.attlocal.ne
##                                                                                      machi
##                                                                                      "arm6
##                                                                                        log
##                                                                                       "roo
##                                                                                         us
##                                                                             "harinishanmuga
##                                                                                 effective_us
##                                                                             "harinishanmuga
```