HW 7

Enter your name and EID here: Arsh Ali & asa3683

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 × 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
## #   ¹Generation
```

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:

```
#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)

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  ~ Attack + HP, data = pokemon, 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

}

# average
mean(perf_k)
```

```
## [1] 0.8585595
```

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 (0.86) is not far from the AUC value of our pokemon_log model (0.85), indicating that the logistic regression model using cross-validation does not exhibit signs of overfitting as the training vs test AU value do not differ significantly.

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()`)?

```
# Cross validation for the kNN model
pokemon_kNN <- knn3(Legendary ~ HP + Attack,
                    data = pokemon,
                    k = 5)

# Predict the Legendary status of each pokemon
predict(pokemon_kNN, pokemon)[,2]
```

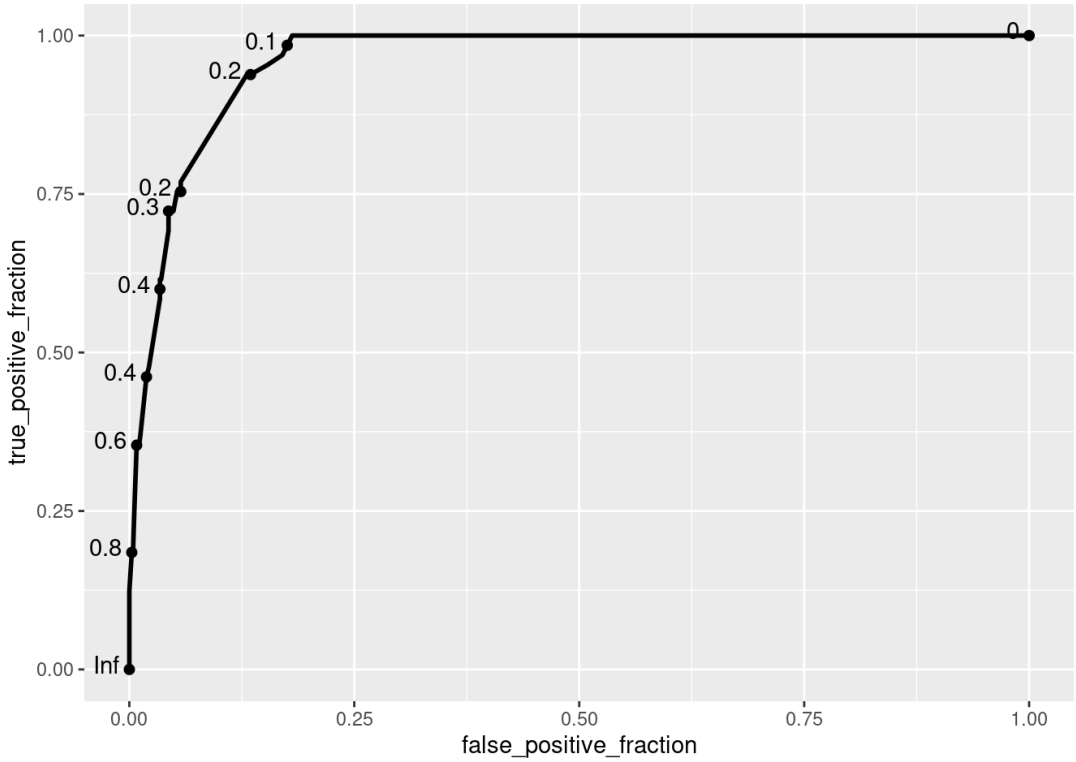
```
##   [1] 0.0000000 0.0000000 0.1111111 0.3333333 0.0000000 0.0000000 0.0000000
##   [8] 0.2000000 0.2857143 0.0000000 0.0000000 0.0000000 0.2857143 0.0000000
##  [15] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [22] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 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 output we get when using the predict function is the predicted 'Legendary' Status for each pokemon, whereby a value of zero indicates the pokemon is not legendary and a value closer to 1 indicates the pokemon likely posseses a legendary status. Overall, the model predicts the legendary status of each pokemon using a value range from zero to one.

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?

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




```
calc_auc(ROC)
```

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

The model performance is well as the AUC value is 0.96, which also means it is more specific and sensitive (as the area under the curve is greater meaning the model is capable of distinguishing between classes - i.e those who are legendary vs those who are not legendary) and is better able to to differentiate/discriminate between positive and negative cases.

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.

```
#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)

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  ~ Attack + HP, data = pokemon)

  # Test model on test set (fold i)
  predict_i <- data.frame(
    predictions = predict(pokemon_kNN, newdata = test_i, k = 5)[,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

}

# average
mean(perf_k)
```

```
## [1] 0.961033
```

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

Here, there is no real decrease in AUC when predicting ‘Legendary’ status with this model using 10-fold cross validation. The AUC value was 0.959, which is close to the AUC value of the kNN model in question 2-3. This indicates the model does not overfit the data as the AUC values do not differ significantly, so it is balanced.

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).

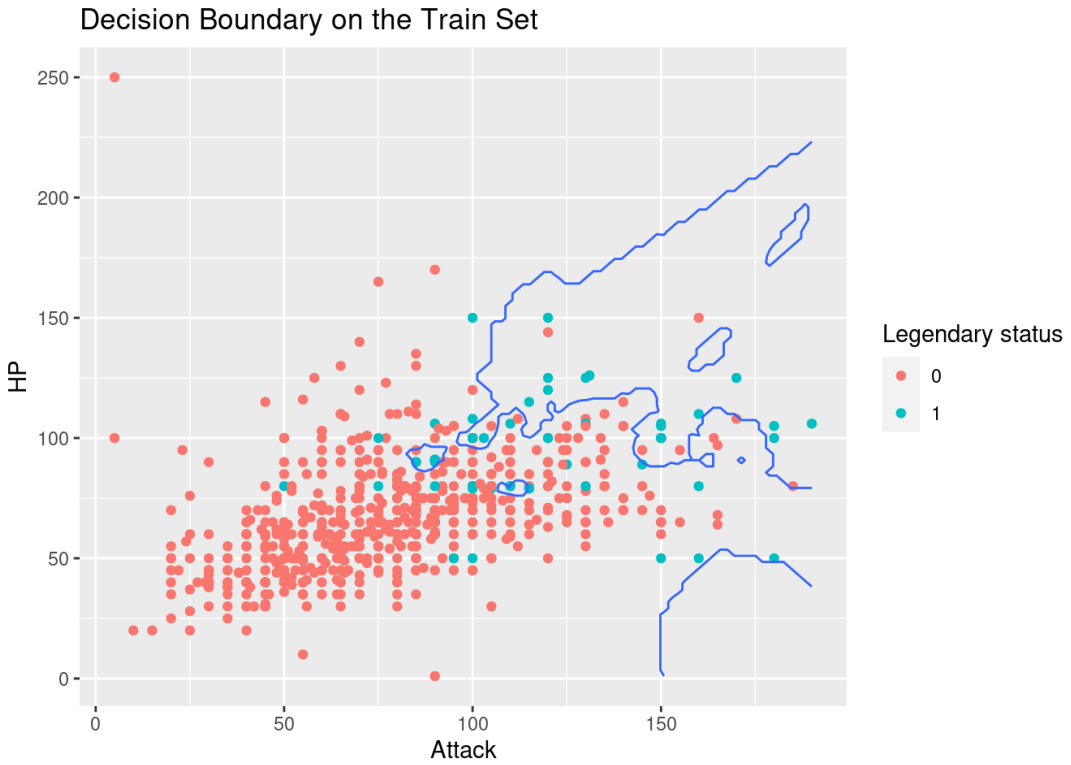
```
# Make this example reproducible by setting a seed
set.seed(322)

# 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")
```

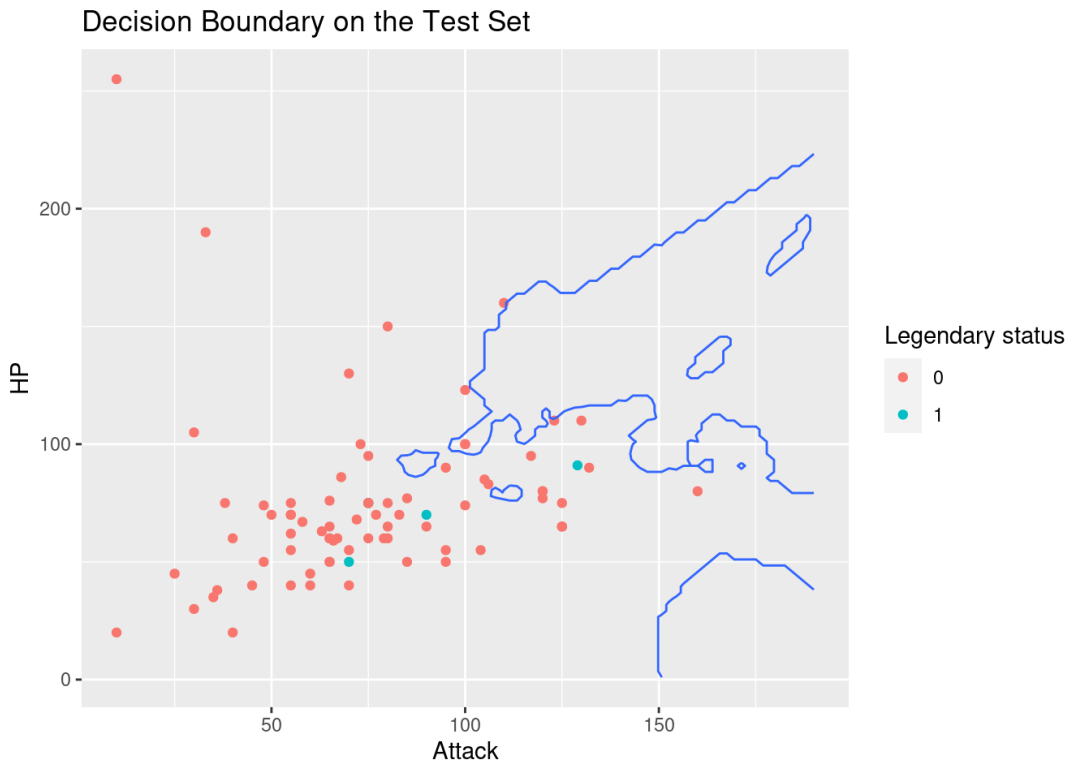


The false positive cases, where the model incorrectly predicts a pokemon to legendary when they are not, are located within the boundry indicated by the coral data points. On the other hand the false negative cases reside outside of the boundry, indicating that the model predicts that a pokemon is not legendary when they are (blue data points).

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
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 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.

The model seems to fit closely to parameters in the training dataset that are not likely to appear in the testing set (so the model is overfitting as the decision boundry for the test set differs from the train set) - the training AUC is higher than the testing AUC. This suggests that the model is overfitting to the training data, meaning it would not be able to generalize well to new data. However, the reason it may be overfitting is because the sample size is not adaquete, and thereby the train vs test set graphs look different, and in turn the model might not perform well on the test set.

Formatting: (2 pts)

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

```
##                                                  sysname
##                                                  "Linux"
##                                                  release
##                                        "5.15.0-67-generic"
##                                                  version
## "#74~20.04.1-Ubuntu SMP Wed Feb 22 14:52:34 UTC 2023"
##                                                 nodename
##                                "educcomp04.ccbb.utexas.edu"
##                                                  machine
##                                                 "x86_64"
##                                                    login
##                                                "unknown"
##                                                     user
##                                                "asa3683"
##                                           effective_user
##                                                "asa3683"
```