HW 6

Enter your name and EID here: Austine Do (ahd589)

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.

Part 1

We will work with the `pokemon` dataset:

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

# Take a look
head(mypokemon)
```

```
## # A tibble: 6 x 13
##   Number Name           Type1 Type2 Total    HP Attack Defense SpAtk SpDef Speed
##    <dbl> <chr>          <chr> <chr> <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1      1 Bulbasaur      Grass Pois~   318    45     49      49    65    65    45
## 2      2 Ivysaur        Grass Pois~   405    60     62      63    80    80    60
## 3      3 Venusaur       Grass Pois~   525    80     82      83   100   100    80
## 4      3 VenusaurMega ~ Grass Pois~   625    80    100     123   122   120    80
## 5      4 Charmander     Fire  <NA>    309    39     52      43    60    50    65
## 6      5 Charmeleon     Fire  <NA>    405    58     64      58    80    65    80
## # i 2 more variables: Generation <dbl>, Legendary <lgl>
```

Every pokemon creature has an array of features. If you would like to learn more about each of them, you can visit https://pokemondb.net/pokedex. But we will only consider very few variables.

Question 1: (2 pts)

Quickly describe the dataset: what does 1 row represent? How many columns are there? If we want to predict if a pokemon is legendary or not, what would be the outcome? What variables could NOT be considered as predictors? *Note: you're not required to add code for this question but you can if you'd like to!*

Each row in the dataset represents an individual pokemon and its corresponding features/stats. There are 13 columns in this dataset. If we want to predict a pokemon's legendary status we would use the `Legendary` variable as the outcome variable.

As R prefers to deal with binary variables coded as 0 and 1, recode the variable `Legendary`, taking a value of 1 if a pokemon is legendary and a value of 0 if not. Once your code works, overwrite `pokemon` with the dataset containing the recoded variable. What proportion of pokemons are legendary?

```r
# Re-coding the Legendary column to binary values and finding the proportion of pokemon that are legend
mypokemon <- mypokemon |>
    mutate(Legendary = ifelse(Legendary == TRUE, 1, 0))

mypokemon |>
    summarize(percent_legendary = sum(Legendary) / n())
```

```
## # A tibble: 1 x 1
##   percent_legendary
##               <dbl>
## 1            0.0812
```

A little more than 8% of pokemon in this dataset are classified as legendary

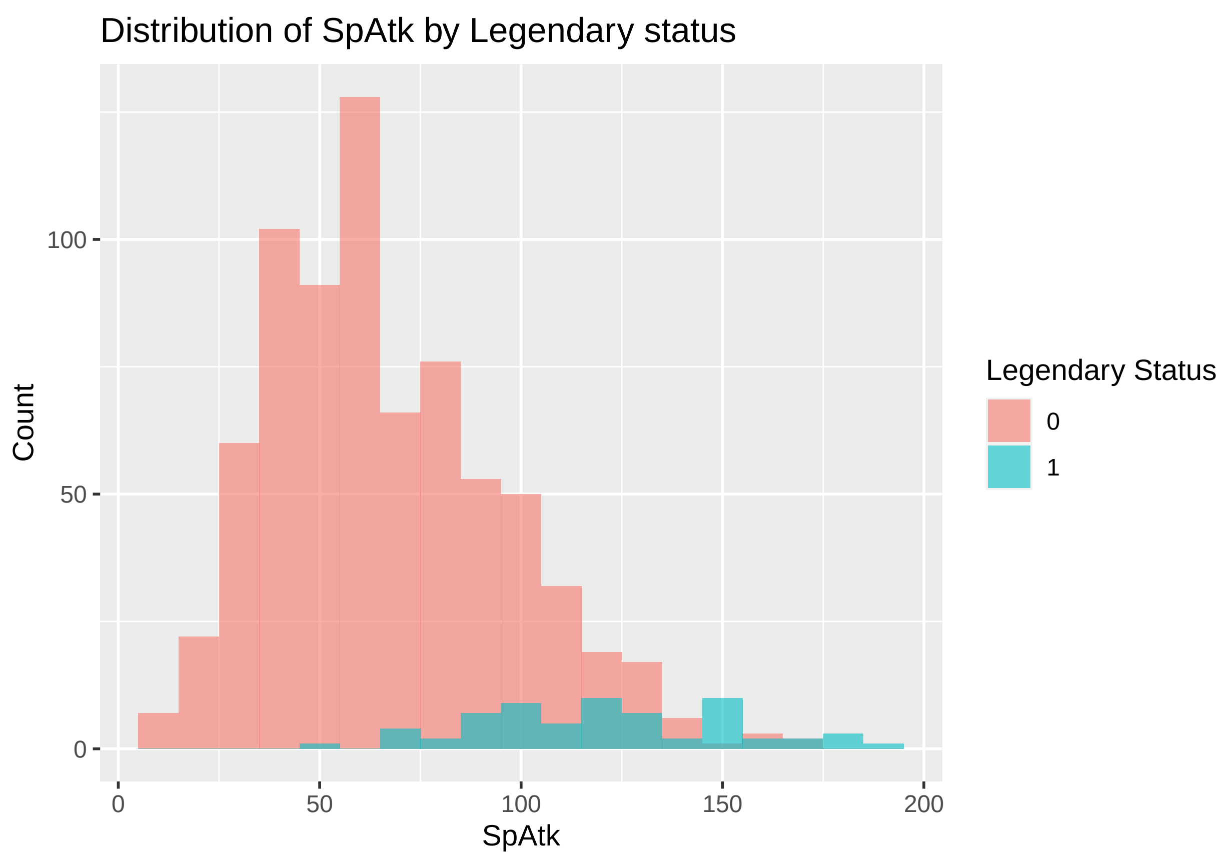
---

Question 2: (3 pts)

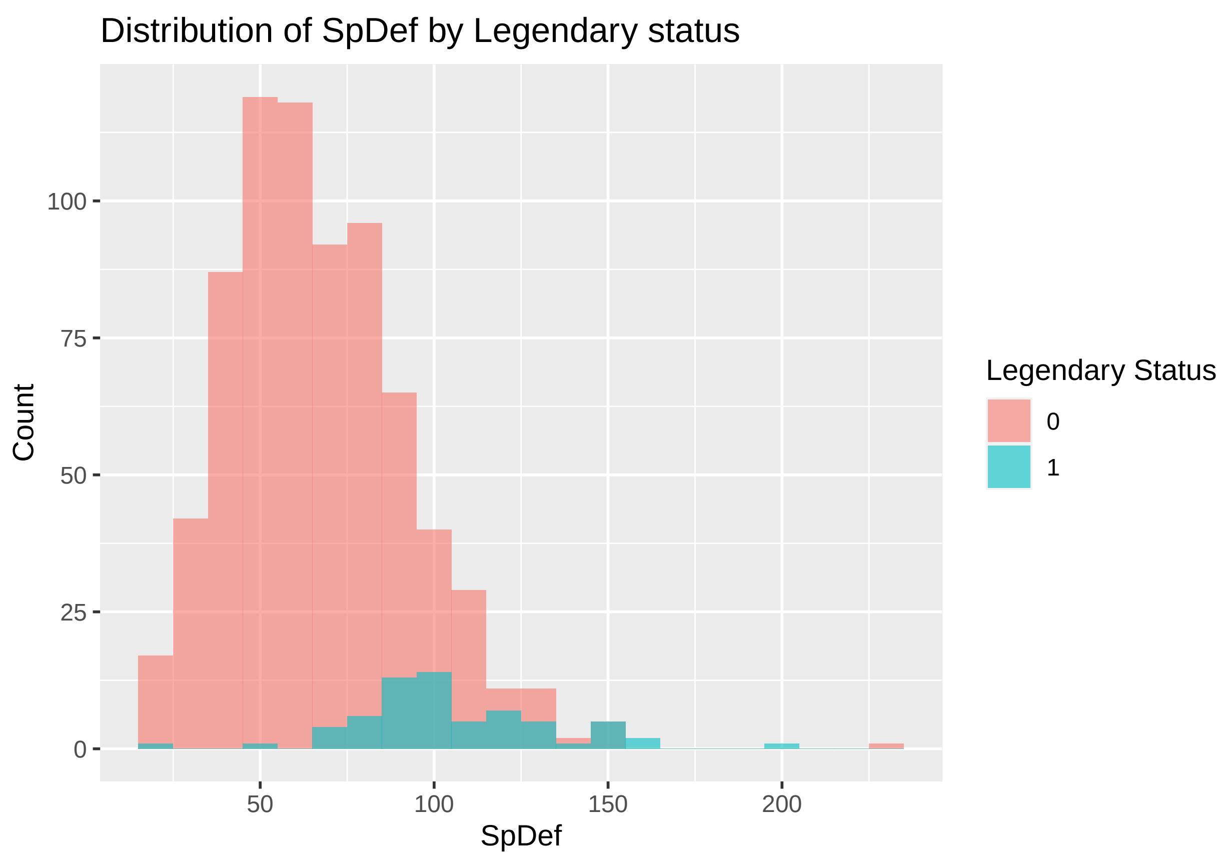
Are legendary pokemons special? Legendary pokemons are incredibly rare and often very powerful. Let's investigate if the features of special attack (`SpAtk`, the strength of special moves: the higher this feature is, the more damage will be done to opponents) and special defense (`SpDef`, the ability to take attacks from other pokemons: the higher this feature is, the fewer points will be lost when attacked) can predict the legendary status of a pokemon.

Visualize how the features of `SpAtk` and `SpDef` impact the legendary status. First, make two visualizations to 1) compare the distribution of `SpAtk` for legendary pokemons vs those that are not, 2) compare the distribution of `SpDef` for these two groups. *Note: if you'd like to add color to your viz, consider the binary variable as a factor for your `ggplot` using `as.factor()` for a discrete scale of color.* Comment with what you see in these viz.

```r
# Visualizing the distribution of SpAtk and SpDef by Legendary Status
mypokemon |>
    group_by(Legendary) |>
    ggplot(aes(x = SpAtk, fill = as.factor(Legendary))) +
        geom_histogram(binwidth = 10, position = 'identity', alpha = 0.6) +
        labs(title = 'Distribution of SpAtk by Legendary status',
             x = 'SpAtk',
             y = 'Count',
             fill = 'Legendary Status')
```



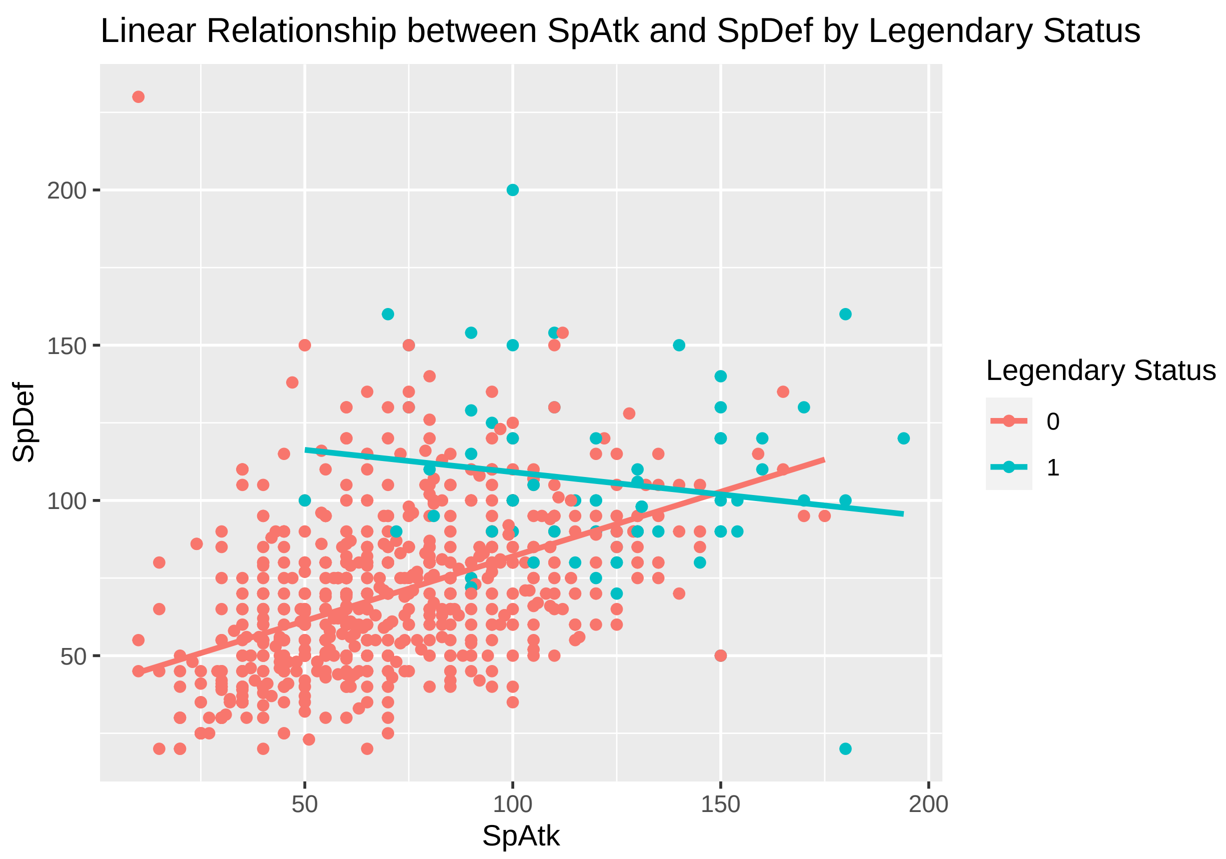
```
mypokemon |>
    group_by(Legendary) |>
    ggplot(aes(x = SpDef, fill = as.factor(Legendary))) +
        geom_histogram(binwidth = 10, position = 'identity', alpha = 0.6) +
        labs(title = 'Distribution of SpDef by Legendary status',
             x = 'SpDef',
             y = 'Count',
             fill = 'Legendary Status')
```



For the distribution of SpAtk, it appears that the median SpAtk on the non-Legendary pokemons is much lower than that of the Legendary pokemons and the non-Legendary pokemons SpAtk is positively skewed while the Legendary pokemons SpAtk seems to be more evenly distributed around the median. For the distribution of SpDef, it appears to be the same where the median SpDef of non-Legendary pokemons is much lower than Legendary pokemons and the non-Legendary pokemons SpDef is positively skewed while the Legendary pokemons SpDef seems to be more evenly distributed around the median. For both distributions, the legendary pokemons generally had a distribution range higher than the non-legendary pokemons.

Second, make a visualization with the linear relationship between `SpAtk` and `SpDef` for each legendary status. *Hint: color the regression lines.* Do these variables seem to predict Legendary status? Comment with what you see in this viz.

```r
# Visualizing the linear relationship between `StAtk` and `SpDef` by Legendary status
mypokemon |>
    ggplot(aes(x =  SpAtk, y = SpDef, color = as.factor(Legendary))) +
        geom_point() +
        geom_smooth(aes(group = Legendary), method = 'lm', se = FALSE) +
        labs(title = 'Linear Relationship between SpAtk and SpDef by Legendary Status',
             x = 'SpAtk',
             y = 'SpDef',
             color = 'Legendary Status')
```



These variables seem to do an okay job at predicting the legendary status. Based on this graph I would expect the residuals for each linear regression line to be fairly high and I also expect the linear model to produce an okay performance in predicting the the legendary status of the pokemon.

Question 3: (2 pts)

Which regression model would be more appropriate (linear or logistic) to predict `Legendary` status?

A logistic regression model would be more appropriate to predict `Legendary` status since Legendary status is a binary/categorical variable

Fit this new model with `SpAtk` and `SpDef` as predictors and call it `pokemon_reg`.

```
# Creating the logistic regression model based on SpAtk and SpDef
pokemon_reg <- glm(Legendary ~ SpAtk + SpDef, data = mypokemon, family = 'binomial')
```

Choose a pokemon which name starts with the same letter as yours. Predict the legendary status of this pokemon using `pokemon_reg`. Do you think your pokemon was predicted to be legendary? Does the prediction match the reality?

```r
# Selecting 'Arceus' as my pokemon and testing the model's prediction
austine_pokemon <- mypokemon[mypokemon$Name == 'Arceus',]

predict(pokemon_reg, type = 'response', newdata = austine_pokemon)
```

```
##        1
## 0.457906
```

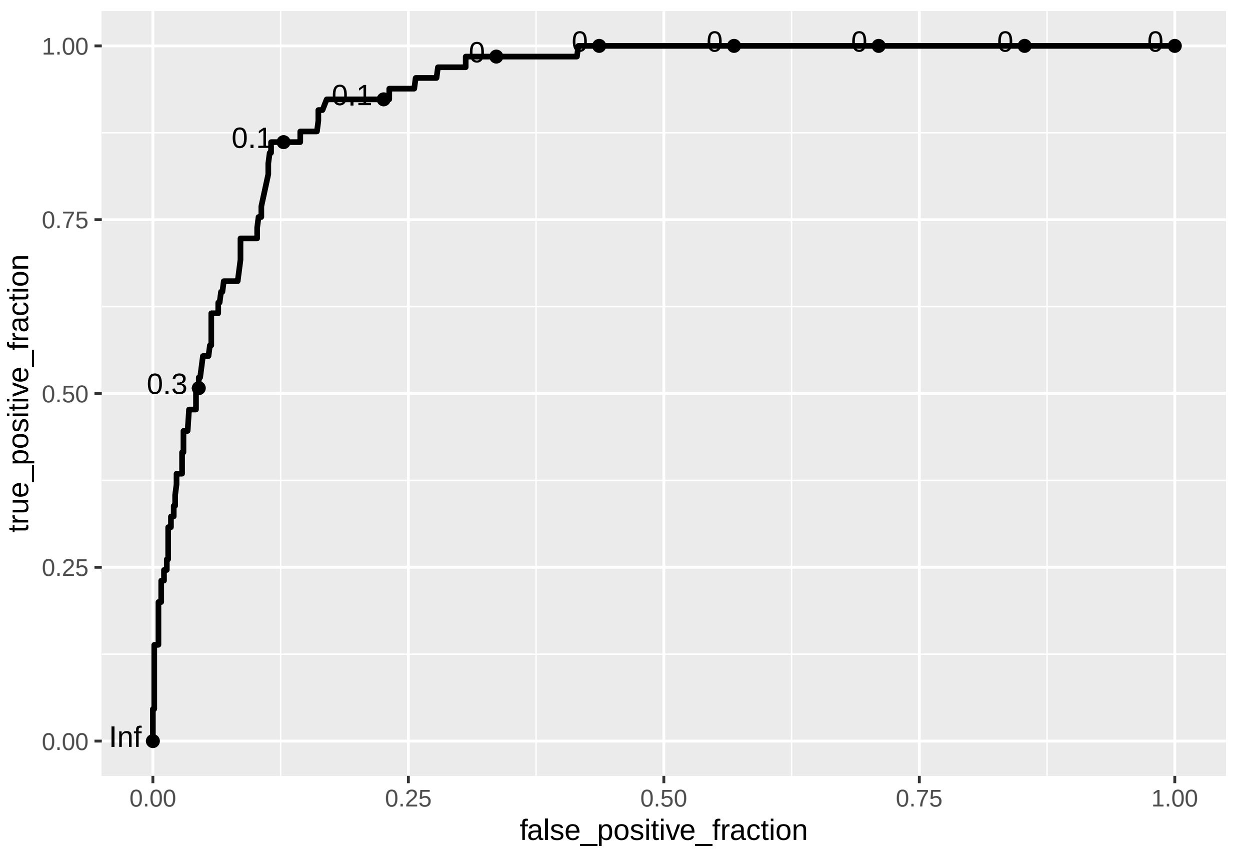
No I don't think the model predicts my pokemon to be legendary because the probability is only about 46% which is quite low. This doesn't match expectations because in reality my pokemon Arceus is classified as a legendary pokemon

---

Question 4: (3 pts)

Evaluate the performance of the `pokemon_reg` model when predicting legendary status for all pokemons. What does the performance indicate? Comment on its value.

```r
# ROC and AUC logistic regression model metrics
ROC <- mypokemon |>
        mutate(probability = predict(pokemon_reg, type = 'response')) |>
        ggplot() +
            geom_roc(aes(d = Legendary, m = probability), n.cuts = 10)
ROC
```



```r
calc_auc(ROC)$AUC
```

```
## [1] 0.9307378
```

The AUC of the ROC curve is fairly high at about 93% which indicates that the model does well in predicting the legendary status of all the pokemon in the dataset

Now, let's perform a 10-fold cross-validation for the `pokemon_reg` model. Adjust the following code (remember to get rid of `eval=FALSE` to actually run this code when knitting):

```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 <- mypokemon[sample(nrow(mypokemon)), ]

# 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 dataset
for(i in 1:k){
  # Create train and test datasets
  train_not_i <- data[folds != i, ] # all observations except in fold i
  test_i <- data[folds == i, ]  # all observations in fold i

  # Train model on train data (all but fold i)
  train_model <- glm(Legendary ~ SpAtk + SpDef, data = train_not_i, family = 'binomial')

  # Performance listed for each test data (fold i)
  perf_k[i] <- calc_auc(
                  ggplot(test_i) +
                      geom_roc(aes(
                          d = Legendary,
                          m = predict(train_model, type = "response", newdata = test_i)))
                  )$AUC

}
perf_k
```

```
##  [1] 0.9859155 0.9360000 0.9314888 0.9146667 0.9054054 0.9617117 0.9288194
##  [8] 0.9680000 0.9391892 0.8026316
```

```r
# Average performance over all k folds and variation
mean(perf_k)
```

```
## [1] 0.9273828
```

```r
sd(perf_k)
```

```
## [1] 0.05021559
```

How does the average performance compare to the performance of the `pokemon_reg` model for the entire data? What does it indicate about the model?

The average performance (0.927) is slightly lower than the performance of the `pokemon_reg` model for the entire data (0.931). This cross validation using k-folds indicates that the performance of the model is consistently good with low variance in performance and will likely generalize well to new data.

Question 5: (3 pts)

Let's consider another model to predict `Legendary` status based on `SpAtk` and `SpDef`: the k-nearest neighbors (kNN). Fit the kNN model with 5 nearest neighbors and call this model `pokemon_kNN`.

```r
# Creating a k-nearest neighbors model to predict a categorical variable
pokemon_kNN <- knn3(Legendary ~ SpAtk + SpDef,
                    data = mypokemon,
                    k = 5)

predict(pokemon_kNN, mypokemon) |>
    as.data.frame()
```

```
##            0         1
## 1  1.0000000 0.0000000
## 2  1.0000000 0.0000000
## 3  0.6250000 0.3750000
## 4  0.8000000 0.2000000
## 5  1.0000000 0.0000000
## 6  1.0000000 0.0000000
## 7  0.8750000 0.1250000
## 8  0.6250000 0.3750000
## 9  0.3333333 0.6666667
## 10 1.0000000 0.0000000
## 11 1.0000000 0.0000000
## 12 1.0000000 0.0000000
## 13 0.6000000 0.4000000
## 14 1.0000000 0.0000000
## 15 1.0000000 0.0000000
## 16 1.0000000 0.0000000
## 17 1.0000000 0.0000000
## 18 1.0000000 0.0000000
## 19 1.0000000 0.0000000
## 20 1.0000000 0.0000000
## 21 1.0000000 0.0000000
## 22 1.0000000 0.0000000
## 23 1.0000000 0.0000000
## 24 1.0000000 0.0000000
## 25 1.0000000 0.0000000
## 26 1.0000000 0.0000000
## 27 1.0000000 0.0000000
## 28 1.0000000 0.0000000
## 29 1.0000000 0.0000000
## 30 1.0000000 0.0000000
## 31 1.0000000 0.0000000
## 32 1.0000000 0.0000000
## 33 1.0000000 0.0000000
## 34 1.0000000 0.0000000
## 35 1.0000000 0.0000000
## 36 1.0000000 0.0000000
## 37 1.0000000 0.0000000
## 38 1.0000000 0.0000000
## 39 1.0000000 0.0000000
## 40 1.0000000 0.0000000
## 41 1.0000000 0.0000000
## 42 0.8000000 0.2000000
## 43 1.0000000 0.0000000
## 44 0.8000000 0.2000000
## 45 1.0000000 0.0000000
## 46 1.0000000 0.0000000
## 47 1.0000000 0.0000000
## 48 1.0000000 0.0000000
## 49 1.0000000 0.0000000
## 50 1.0000000 0.0000000
##  [ reached 'max' / getOption("max.print") -- omitted 750 rows ]
```

What does the model predict for each pokemon (i.e., what output do we get when using the function `predict()`)?

The model predicts the probability of a specific pokemon's legendary status for all pokemon in the dataset.

Evaluate the performance of the `pokemon_kNN` model when fitted on the entire dataset. How does this performance compare to the performance of the `pokemon_reg` model?

```r
# Using the AUC performance metric to evaluate the `pokemon_kNN` model
calc_auc(
  ggplot(mypokemon) +
    geom_roc(aes(
      d = Legendary,
      m = predict(pokemon_kNN, mypokemon)[,2]))
  )$AUC
```

```
## [1] 0.9560963
```

Based on the AUC of the ROC curve, it appears that the `pokemon_kNN` model performs better than the `pokemon_reg` by about 2-3%.

Now, let's also perform a 10-fold cross-validation for the `pokemon_kNN` model. Copy/paste the code from the previous question and adjust it for performing the cross-validation with the `pokemon_kNN` model.

```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 <- mypokemon[sample(nrow(mypokemon)), ]

# 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 dataset
for(i in 1:k){
  # Create train and test datasets
  train_not_i <- data[folds != i, ] # all observations except in fold i
  test_i <- data[folds == i, ]  # all observations in fold i

  # Train model on train data (all but fold i)
  train_model <- knn3(Legendary ~ SpAtk + SpDef, data = train_not_i, k = 5)

  # Performance listed for each test data (fold i)
  perf_k[i] <- calc_auc(
                  ggplot(test_i) +
                      geom_roc(aes(
                          d = Legendary,
                          m = predict(train_model, newdata = test_i)[,2]))
                  )$AUC

}
perf_k
```

```
##  [1] 0.9522692 0.9466667 0.8952569 0.7986667 0.8569820 0.8693694 0.7786458
##  [8] 0.6586667 0.9493243 0.6217105
```

```r
# Average performance over all k folds and variation
mean(perf_k)
```

```
## [1] 0.8327558
```

```r
sd(perf_k)
```

```
## [1] 0.1181422
```

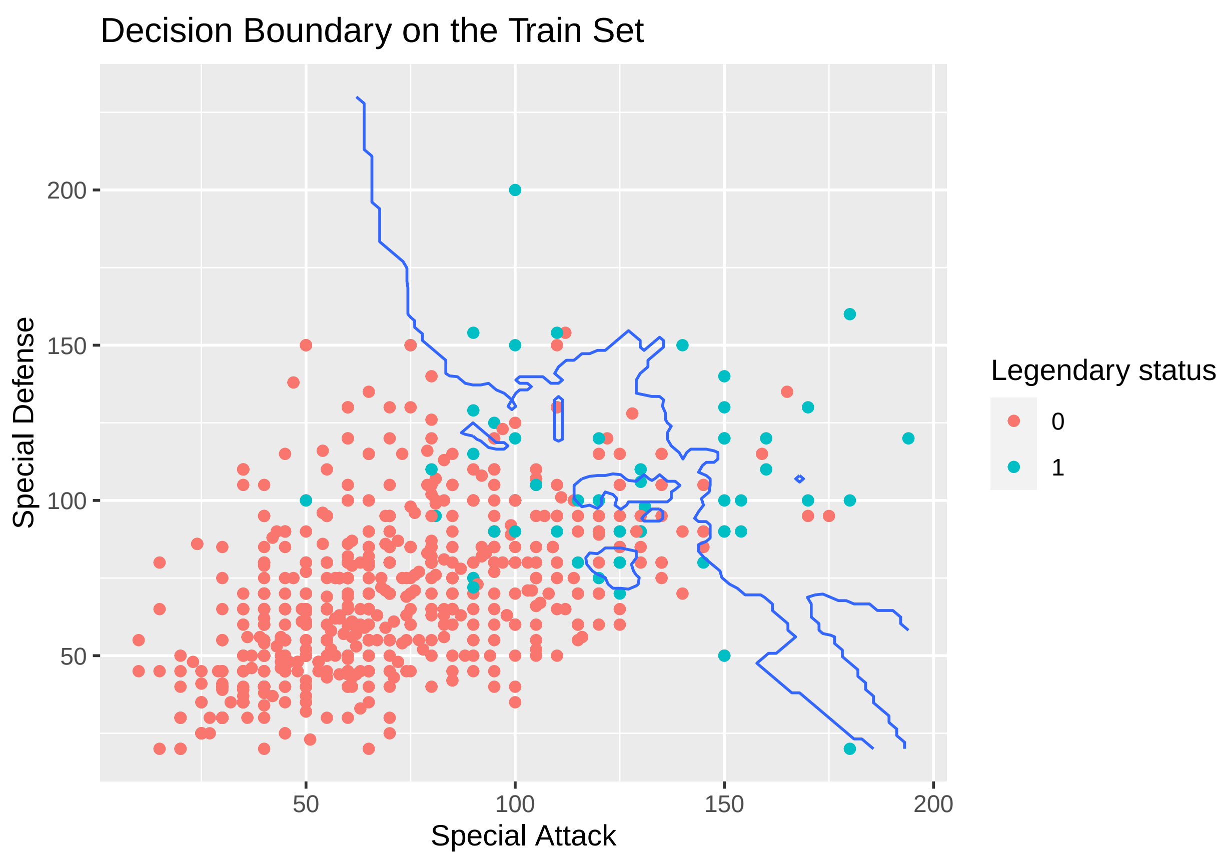
You should observe a huge decrease in average performance. What does it indicate about our model?

This indicates that our model might be overfitting since the average performance is lower and the variance of model performance is high.

Question 6: (2 pts)

Let's focus on the `pokemon_kNN` model trained on a random 90% of the data and then tested on the remaining 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).

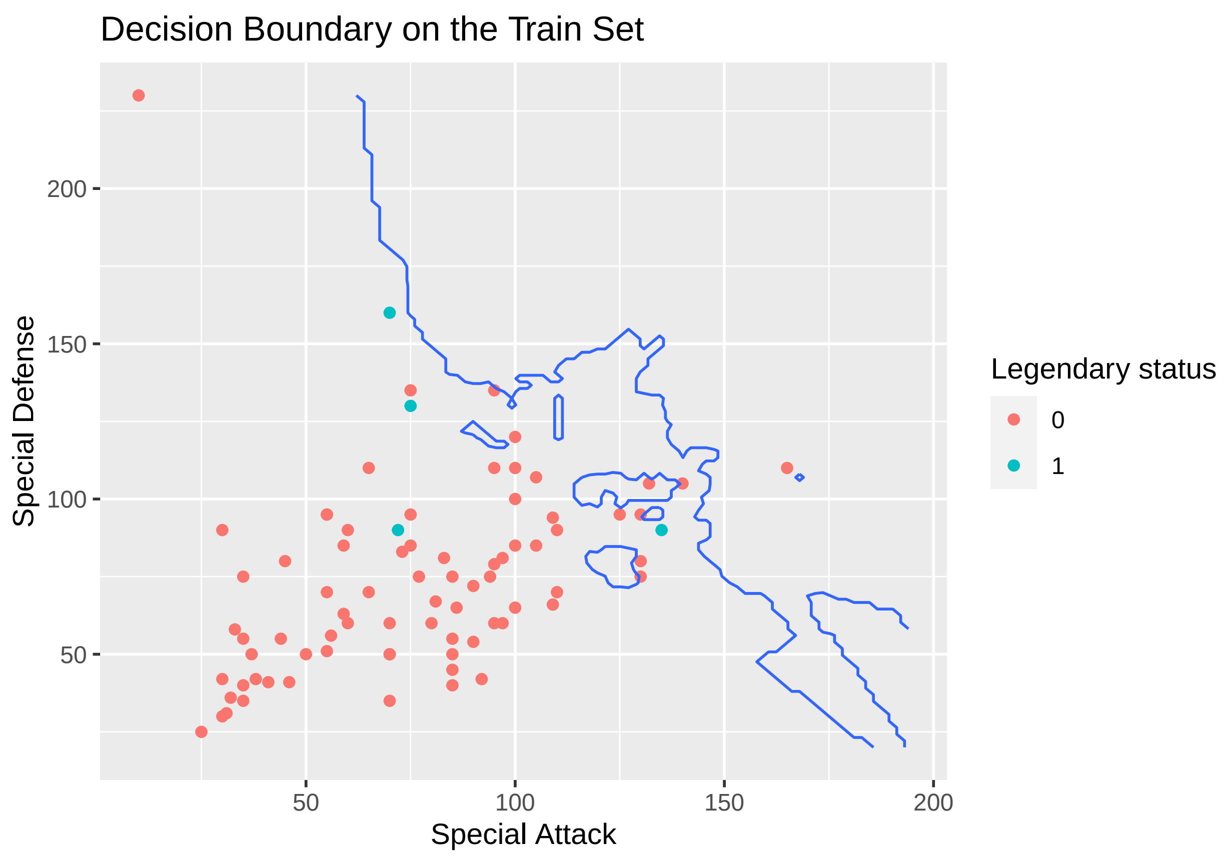
```r
# Decision boundary plot with the training data

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

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

# Fit the model on the train data
pokemon_kNN <- knn3(Legendary ~ SpAtk + SpDef,
                data = train,
                k = 5)

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

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



The false positive cases are located within the decision boundary and this means that the model predicted them to be legendary but in reality they were not legendary while the false negatives are located outside the decision boundary and this means that the model predicted them to be not legendary but in reality they were actually legendary

Now, represent the same decision boundary but with the test set. *Hint: copy/paste and adjust the code above.*

```r
# Decision boundary plot with the test data

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

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

# Fit the model on the train data
pokemon_kNN <- knn3(Legendary ~ SpAtk + SpDef,
                data = train,
                k = 5)

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

# Use this grid to predict legendary status
grid |>
  mutate(p = predict(pokemon_kNN, grid)[,2]) |>
  ggplot(aes(SpAtk, SpDef)) +
  # Only display data in the train set
  geom_point(data = test,
             aes(SpAtk, SpDef, color = as.factor(Legendary))) +
  # Draw the decision boundary
  geom_contour(aes(z = p), breaks = 0.5) +
  # Labels
  labs(x = "Special Attack",
       y= "Special Defense",
       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.

A potential and likely reason the kNN model may not perform well on the test set is because of overfitting on the training data. This causes the model to be too specific, capture noise in the data, and capture specific pattern in the data that may not generalize well to new test data and it seems to be the case for this model.

Part 2

Recall the context about the Internet clothing retailer Stitch Fix wanting to develop a new model for selling clothes to people online (see HW 1, HW2, and HW 4). Their basic approach is to send people a box of 5–6 items of clothing and allow them to try the clothes on. Customers keep (and pay for) what they like while mailing back the remaining clothes. Stitch Fix then sends customers a new box of clothes a month later.

You built a model to predict the clothes each customer would be more likely to keep.

Question 7: (2 pts)

What is the goal of cross-validation and why is it important in the context of building a clothing recommendation model for Stitch Fix?

The goal of cross-validation is to assess the performance and generalization ability of a model. In the context of building a clothing recommendation model, cross-validation is crucial for ensuring the model's robustness and its ability to generalize well to unseen data. By validating the model on multiple subsets of the dataset, cross-validation helps identify potential issues such as overfitting or underfitting, enabling the selection of a model that performs well on diverse data. This process enhances the reliability and effectiveness of clothing recommendations for Stitch Fix customers.

Question 8: (2 pts)

After Stitch Fix sent a few boxes to some customers, the customers returned all of the clothes in the box. What would be some important steps to take regarding your model?

Reevaluating and refining the recommendation model based on the insights gained, adjusting the model's parameters, incorporating additional features into the model as necessary, and testing the model versions against each other and repeating the steps to iteratively develop and improve the accuracy of the recommendation model are absolutely important steps to take regarding the model.

Formatting: (1 pt)

Knit your file! You can knit into html and once it knits in html, click on `Open in Browser` at the top left of the window that pops out. **Print** your html file into pdf from your browser.

Is it working? If not, try to decipher the error message: look up the error message, consult websites such as stackoverflow or crossvalidated.

Finally, remember to select pages for each question when submitting your pdf to Gradescope.