Pokémon Dataset Analysis & Legendary Prediction

William L Shepherd

August 12, 2023

1. Introduction

This is a simple project which I undertook last year (based on a DataCamps project), which aims to introduce simple data analysis and machine learning concepts using a data set of one of my favorite video-game franchises. The key objectives of the project are the following:

- To develop foundation skills in data cleaning, analysis and visualization.
- To familiarize myself with the R language.
- To develop simple machine learning models utilizing the Random Forest and Decision Tree algorithms in order to predict whether Pokémon are legendary or not.

Hyperlink to data set: https://www.kaggle.com/datasets/rounakbanik/pokemon

Hyperlink to banner picture: https://pokemonletsgo.pokemon.com/en-gb/how-to-play/

2. Implementation

2.1 Loading Libraries

```
library(datasets)
library("tidyverse")
library(ggplot2)
library(rgl)
library(plotly)
library(dplyr)
library(gridExtra)
library(RColorBrewer)
library(ggrepel)
library(caret)
library(e1071)
library(randomForest)
library(tree)
library(pROC)
library(RColorBrewer)
```

2.2 Importing Data

```
df <- read.csv("pokemon.csv")
```

2.3 Cleaning Data

Here I’m selecting the most relevant metrics which I want to analyse, alongside ones which will be used for the machine learning models, to predict whether a Pokemon is legendary or not. I’m leaving out metrics such as the following: capture_rate, abilities, experience growth, base happiness, percentage growth and type 2.

```
df <- df %>%
  select(name, type1, attack, defense, height_m, sp_attack, sp_defense, 
speed, weight_kg, generation, is_legendary, hp)
```

2.4 Displaying the Dataset’s Head

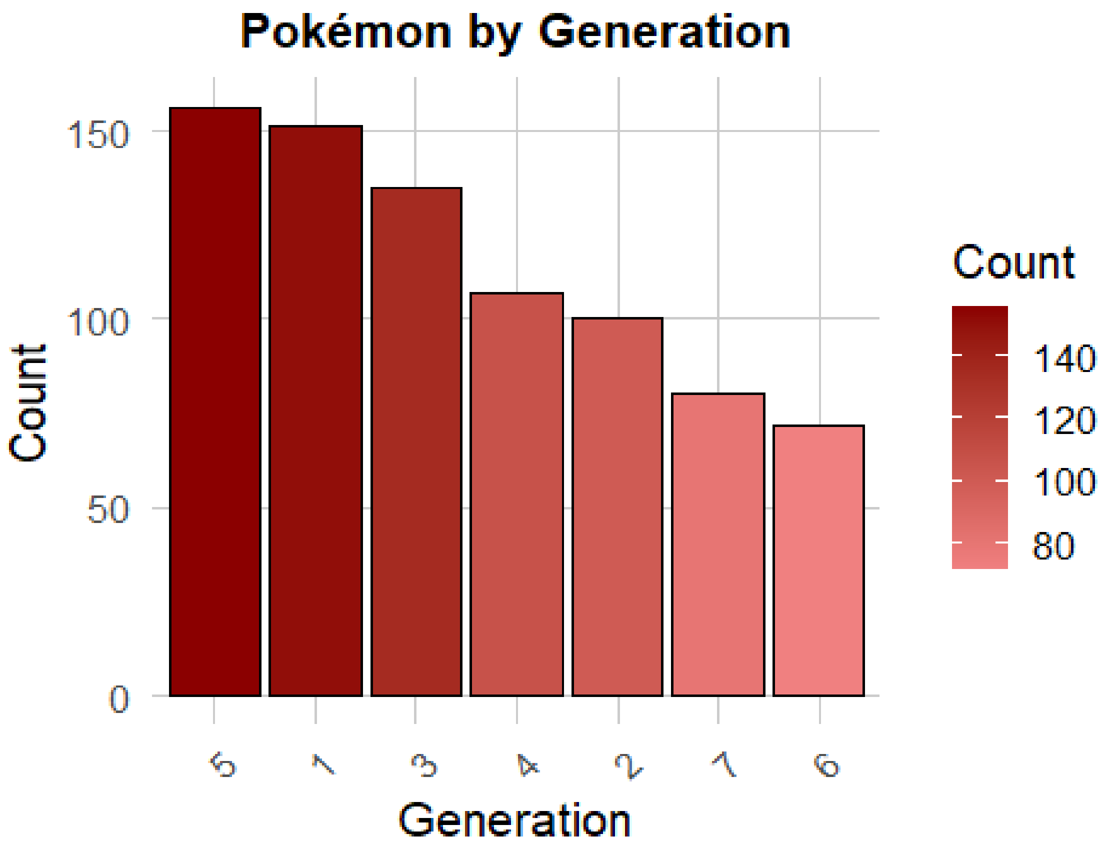
```
head(df)

##         name type1 attack defense height_m sp_attack sp_defense speed
weight_kg
## 1  Bulbasaur grass     49      49      0.7        65         65    45
6.9
## 2    Ivysaur grass     62      63      1.0        80         80    60
13.0
## 3   Venusaur grass    100     123      2.0       122        120    80
100.0
## 4 Charmander  fire     52      43      0.6        60         50    65
8.5
## 5 Charmeleon  fire     64      58      1.1        80         65    80
19.0
## 6  Charizard  fire    104      78      1.7       159        115   100
90.5
##   generation is_legendary hp
## 1          1            0 45
## 2          1            0 60
## 3          1            0 80
## 4          1            0 39
## 5          1            0 58
## 6          1            0 78
```

2.5 Data Analysis

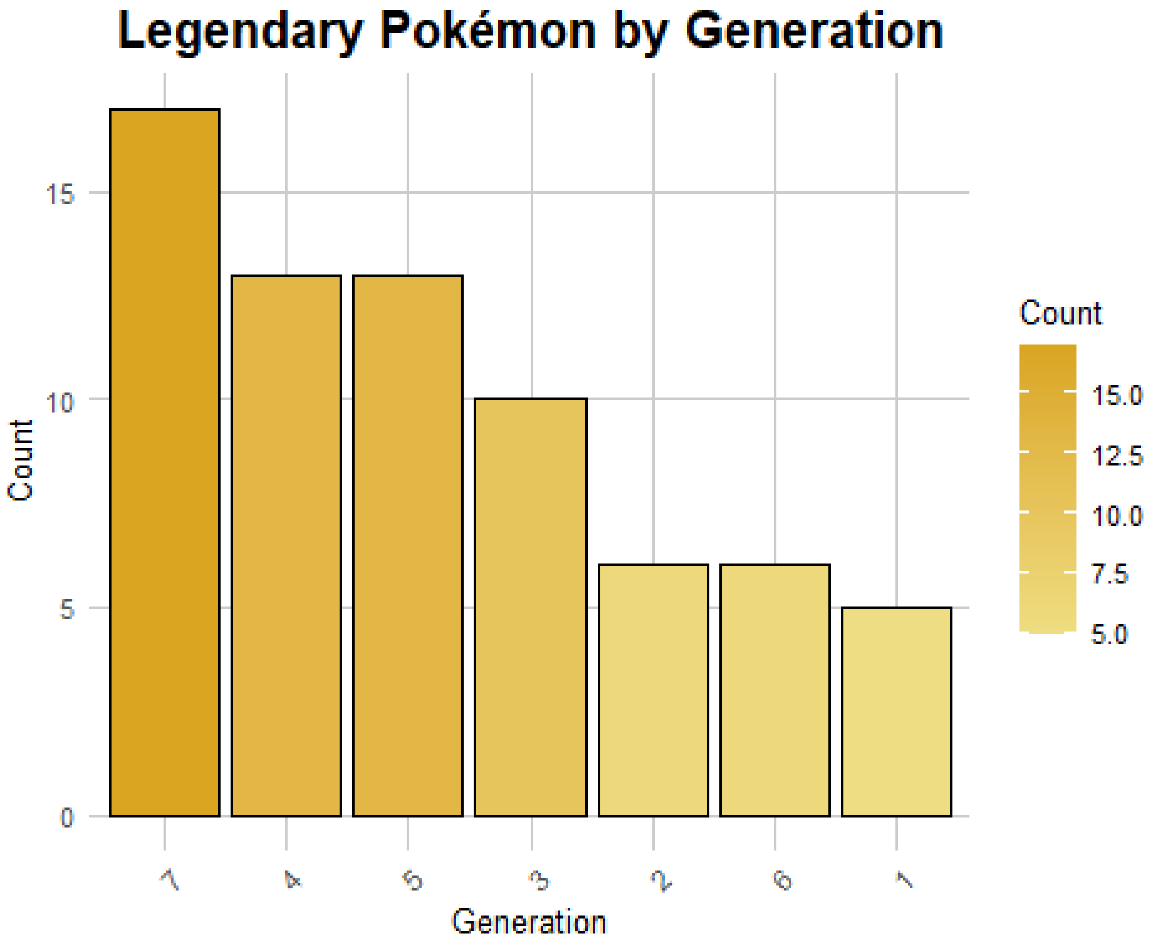
2.5.1 Displaying the Number of Pokemon per Generation

Generations in Pokémon simply refers to what generation of games a particular Pokémon was introduced in (e.g. Pikachu was introduced in Generation I, whilst Dialga was in IV). The generation with the most legendary Pokémon is the seventh, which has 17.

```r
pokemon_by_gen <- df %>%
  group_by(generation) %>%
  summarize(Count = n())

ggplot(pokemon_by_gen, aes(x = reorder(generation, -Count), y = Count)) +
  geom_bar(stat = "identity", aes(fill = Count), color = "black") +
  scale_fill_gradient(low = "lightcoral", high = "darkred") +
  labs(title = "Pokémon by Generation",
       x = "Generation",
       y = "Count") +
  theme_minimal(base_size = 15) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
        panel.grid.major = element_line(color = "grey80"),
        panel.grid.minor = element_blank())
```



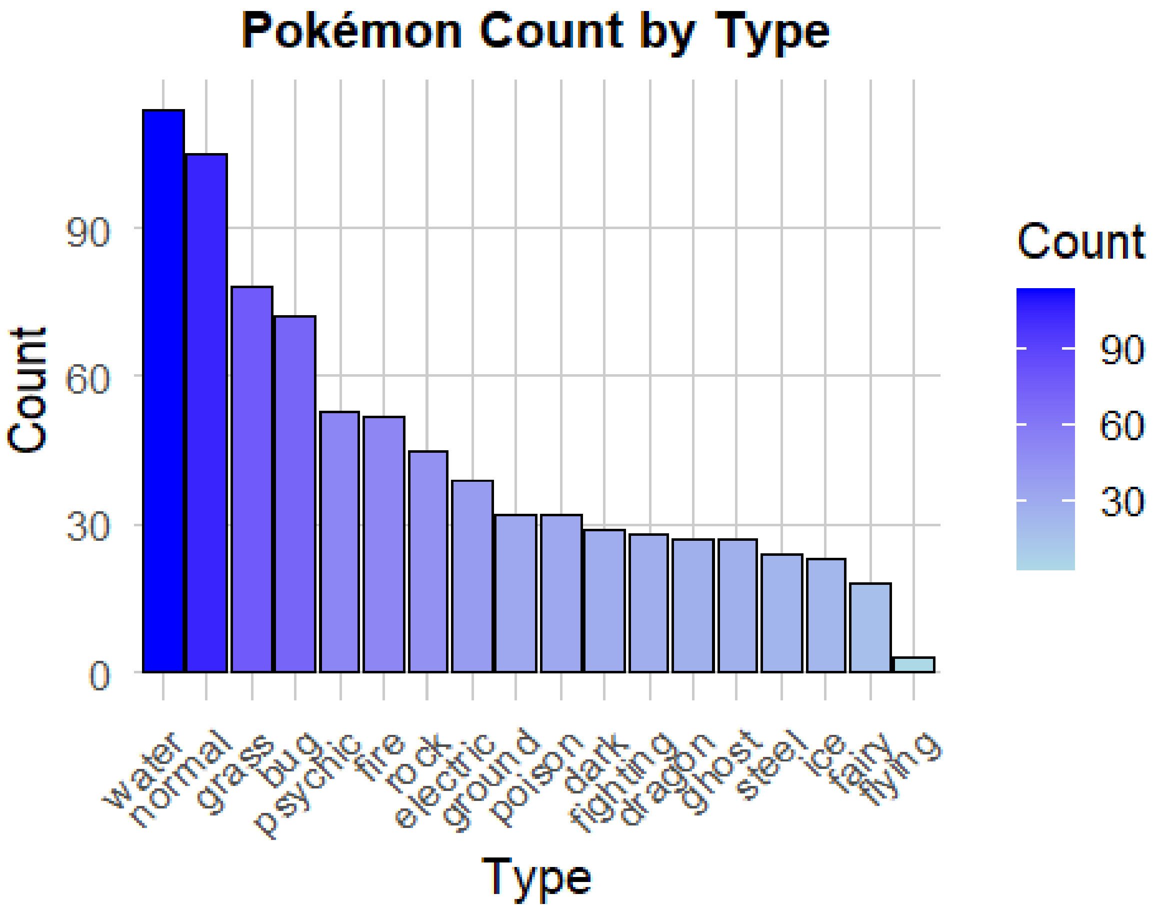
```r
pokemon_by_gen_legendary <- df %>%
  filter(is_legendary == TRUE) %>%
  group_by(generation) %>%
  summarize(Count = n())

ggplot(pokemon_by_gen_legendary, aes(x = reorder(generation, -Count), y =
Count)) +
  geom_bar(stat = "identity", aes(fill = Count), color = "black") +
  scale_fill_gradient(low = "lightgoldenrod", high = "goldenrod") +
  labs(title = "Legendary Pokémon by Generation", x = "Generation", y =
"Count") +
  theme_minimal(base_size = 10) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
        panel.grid.major = element_line(color = "grey80"),
        panel.grid.minor = element_blank())
```



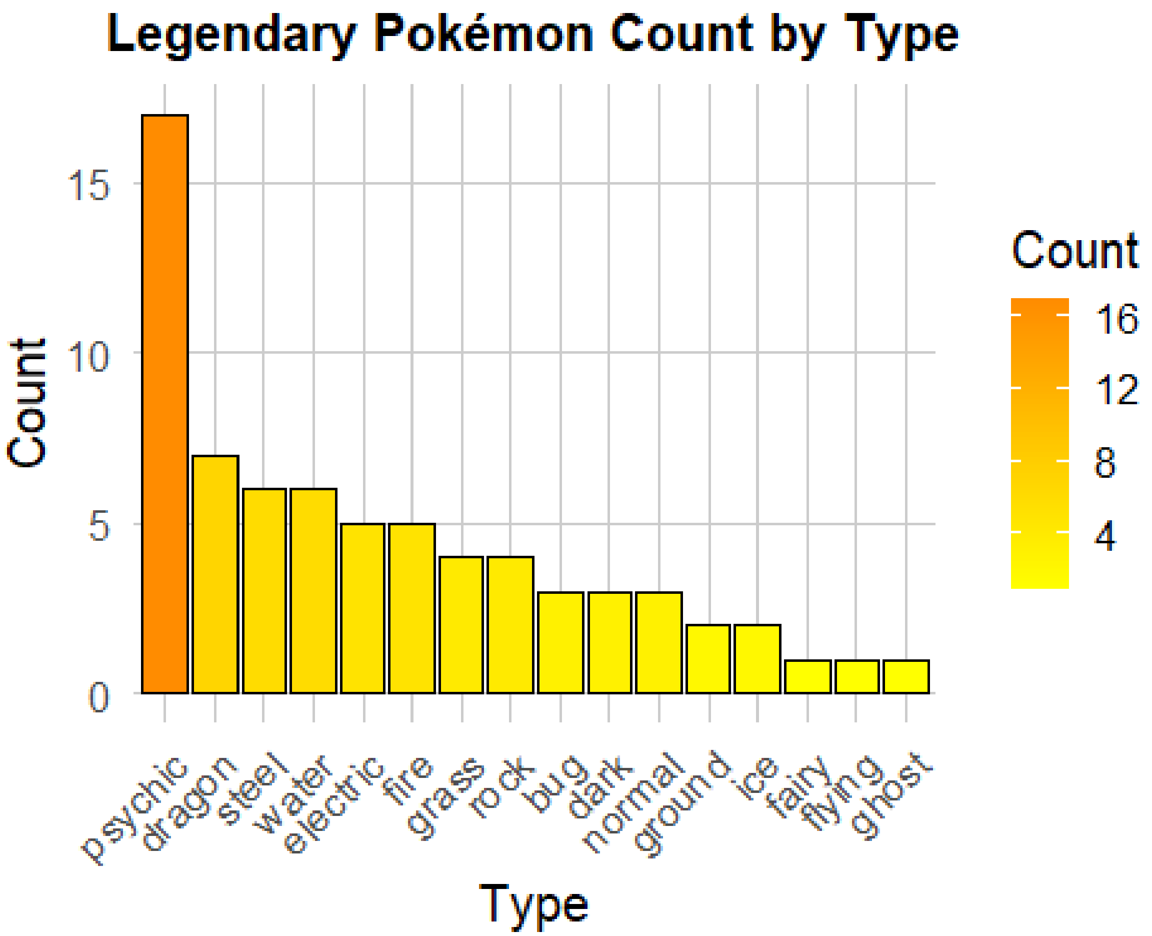
2.5.2 Displaying the Number of Pokemon per Type

As shown by the graph below, the most common type of legendary Pokemon is psychic, with as of generation 7, the absence of poison or fighting legends.

```r
pokemon_by_type <- df %>%
  group_by(type1) %>%
  summarize(Count = n())

ggplot(pokemon_by_type, aes(x = reorder(type1, -Count), y = Count)) +
  geom_bar(stat = "identity", aes(fill = Count), color = "black") +
  scale_fill_gradient(low = "lightblue", high = "blue") +
  labs(title = "Pokémon Count by Type", x = "Type", y = "Count") +
  theme_minimal(base_size = 15) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
        panel.grid.major = element_line(color = "grey80"),
        panel.grid.minor = element_blank())
```



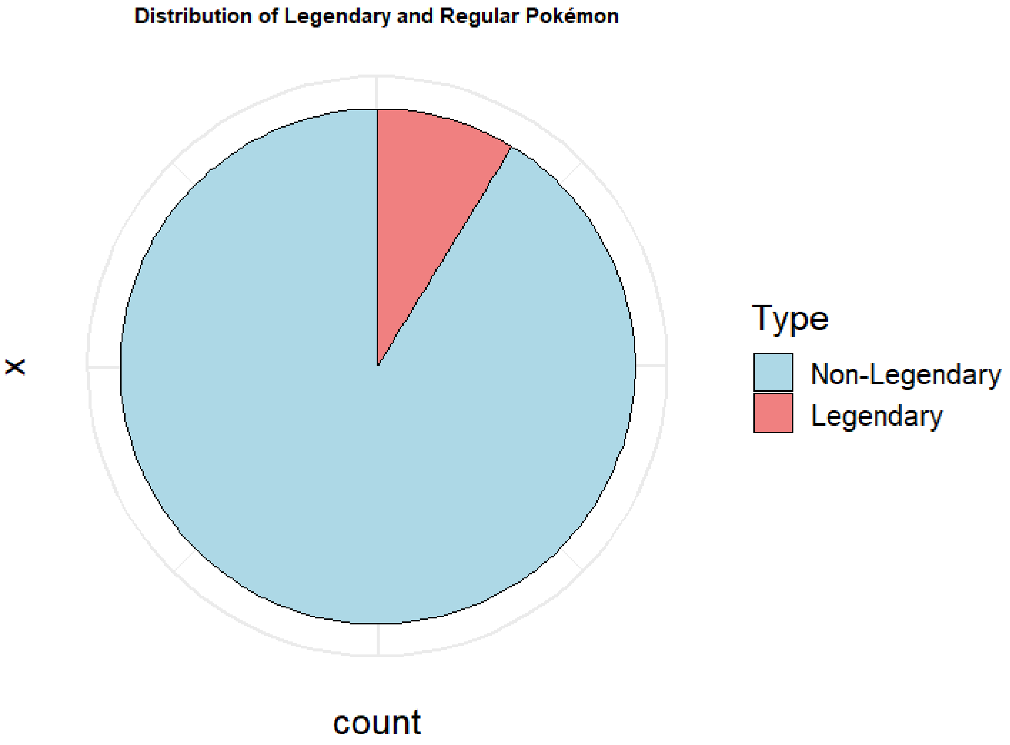
```
pokemon_by_type <- df %>%
  filter(is_legendary == 1) %>%
  group_by(type1) %>%
  summarize(Count = n())

ggplot(pokemon_by_type, aes(x = reorder(type1, -Count), y = Count)) +
  geom_bar(stat = "identity", aes(fill = Count), color = "black") +
  scale_fill_gradient(low = "yellow", high = "darkorange") +
  labs(title = "Legendary Pokémon Count by Type", x = "Type", y = "Count") +
  theme_minimal(base_size = 15) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
        panel.grid.major = element_line(color = "grey80"),
        panel.grid.minor = element_blank())
```



2.5.3 Legendary Pokemon Analysis

2.5.3.1 What Percentage of Pokemon are Legendary?

```r
num_legendary_pokemon <- df %>%
  filter(is_legendary == 1) %>%
  nrow()

legendary_pokemon <- df %>%
  filter(is_legendary == 1)

pokemon_type <- df %>%
  group_by(is_legendary) %>%
  summarize(count = n())

ggplot(pokemon_type, aes(x = "", y = count, fill = factor(is_legendary))) +
  geom_bar(stat = "identity", color = "black") +
  coord_polar("y") +  # Convert to pie chart
  scale_fill_manual(values = c("lightblue", "lightcoral"), labels = c("Non-Legendary", "Legendary")) +
  labs(title = "Distribution of Legendary and Regular Pokémon", fill = "Type") +
  theme_minimal(base_size = 15) +
  theme(axis.text.x = element_blank(),
        axis.ticks = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 13, face = "bold"))
```

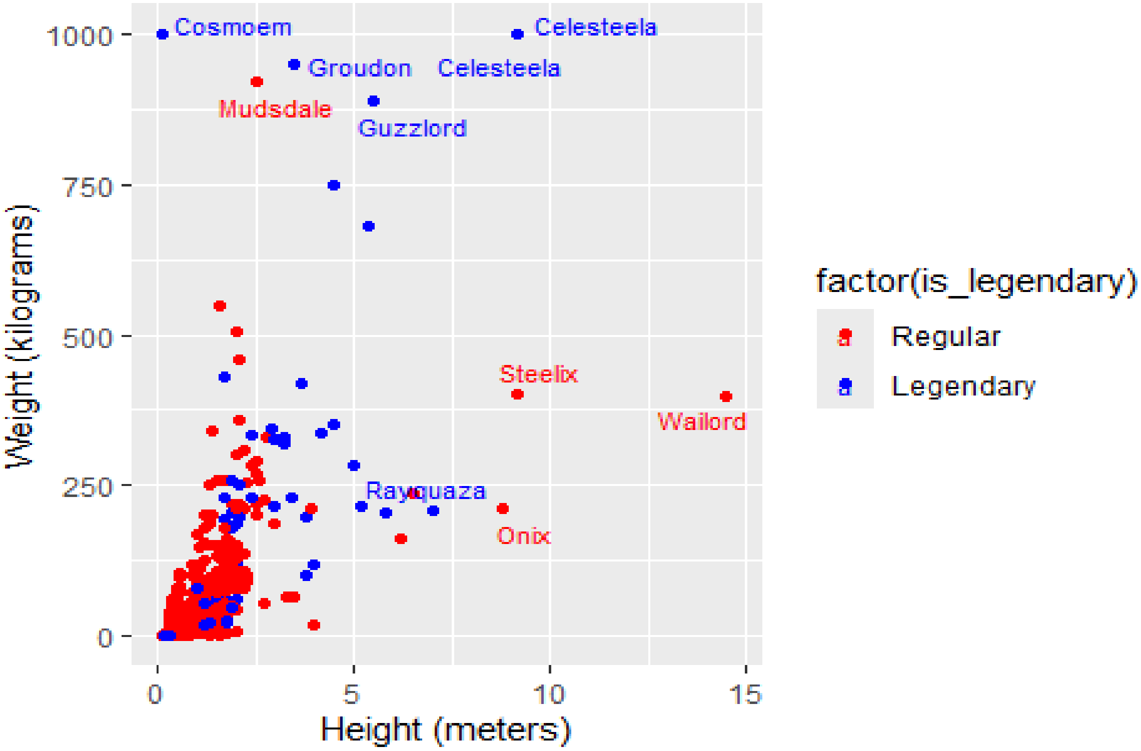


2.5.3.2 Height vs Weight

As shown by the graph below, which explores the relationship between height (m) and weight (kg); legendary Pokémon compared to their regular counterparts are generally much heavier and taller with some exceptions including: Onix, Steelix and Wailord.

```
top_5_heaviest <- df %>%
  top_n(5, weight_kg)

top_5_tallest <- df %>%
  top_n(5, height_m)

combined_top_5 <- rbind(top_5_heaviest, top_5_tallest)

ggplot(df, aes(x = height_m, y = weight_kg, color = factor(is_legendary))) +
  geom_point() +
  geom_text_repel(data = combined_top_5, aes(label = name), size = 3,
nudge_x = 0.2, nudge_y = 0.2) +
  labs(title = "", x = "Height (meters)", y = "Weight (kilograms)") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))
```

```
## Warning: Removed 20 rows containing missing values or values outside the
scale range
## (`geom_point()`).
```



2.5.3.3 Comparing Speed Against Other Metrics

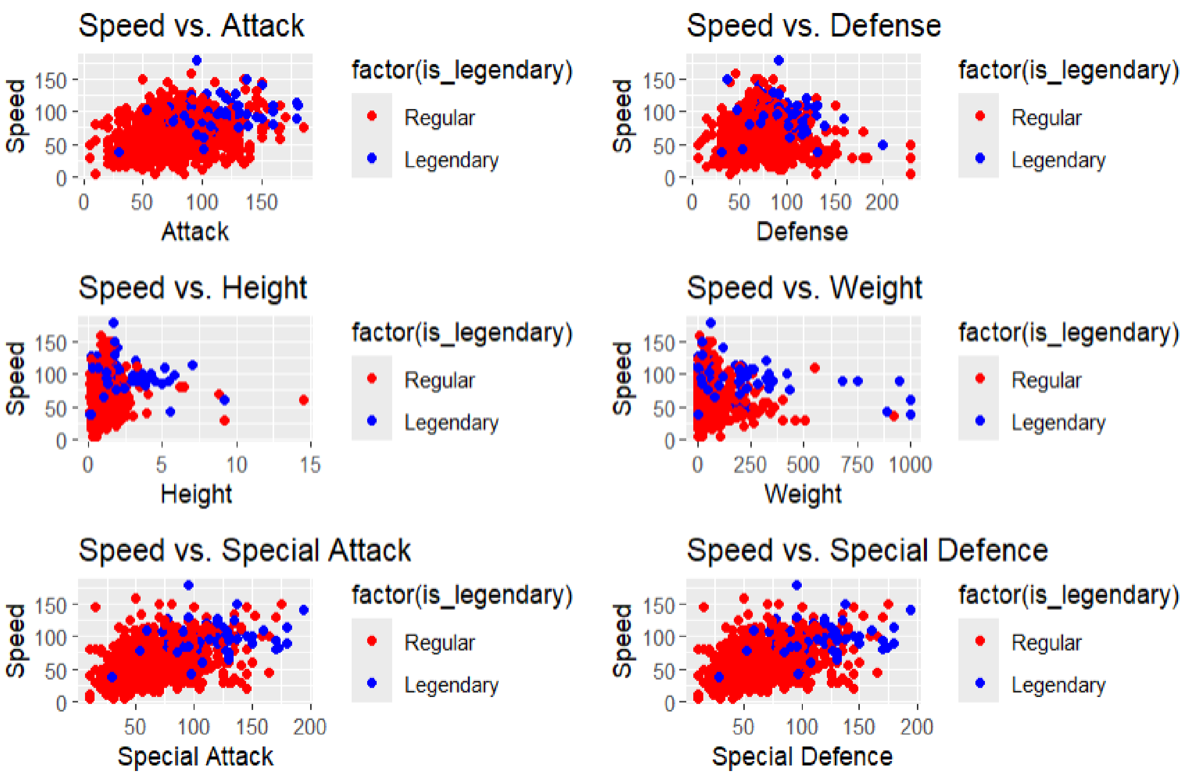
The below graphs illustrate the relationship between Speed and various other attributes (Attack, Defense, Height, Weight, Special Attack, Special Defense) for Regular and Legendary Pokémon. Each graph consists of a scatter plot where the x-axis represents the attribute and the y-axis represents Speed. The points are color-coded to distinguish between Regular and Legendary Pokémon.

The General Trends include the following:

- Legendary Pokémon exhibit a strong positive correlation between speed and both attack and special attack for legendary Pokémon, thus suggesting Legendary Pokémon with higher attack or special attack values tend to have correspondingly higher speed.
- A weak negative correlation is evident between speed and both defense and special defense for both regular and legendary Pokémon; indicating that those with higher defence stats generally have lower speed.

```
speed_vs_attack <- ggplot(df, aes(x = attack, y = speed, color =
factor(is_legendary))) +
  geom_point() +
  labs(title = "Speed vs. Attack", x = "Attack", y = "Speed") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))

speed_vs_defense <- ggplot(df, aes(x = defense, y = speed, color =
factor(is_legendary))) +
  geom_point() +
  labs(title = "Speed vs. Defense", x = "Defense", y = "Speed") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))

speed_vs_height <- ggplot(df, aes(x = height_m, y = speed, color =
factor(is_legendary))) +
  geom_point() +
  labs(title = "Speed vs. Height", x = "Height", y = "Speed") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))

speed_vs_weight <- ggplot(df, aes(x = weight_kg, y = speed, color =
factor(is_legendary))) +
  geom_point() +
  labs(title = "Speed vs. Weight", x = "Weight", y = "Speed") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))

speed_vs_spattack <- ggplot(df, aes(x = sp_attack, y = speed, color =
factor(is_legendary))) +
  geom_point() +
  labs(title = "Speed vs. Special Attack", x = "Special Attack", y =
"Speed") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))

speed_vs_spdefence <- ggplot(df, aes(x = sp_attack, y = speed, color =
factor(is_legendary))) +
  geom_point() +
  labs(title = "Speed vs. Special Defence", x = "Special Defence", y =
"Speed") +
  scale_color_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary"))

grid.arrange(speed_vs_attack, speed_vs_defense, speed_vs_height,
speed_vs_weight, speed_vs_spattack, speed_vs_spdefence, ncol = 2)
```

```
## Warning: Removed 20 rows containing missing values or values outside the
scale range
## (`geom_point()`).
## Removed 20 rows containing missing values or values outside the scale
range
## (`geom_point()`).
```



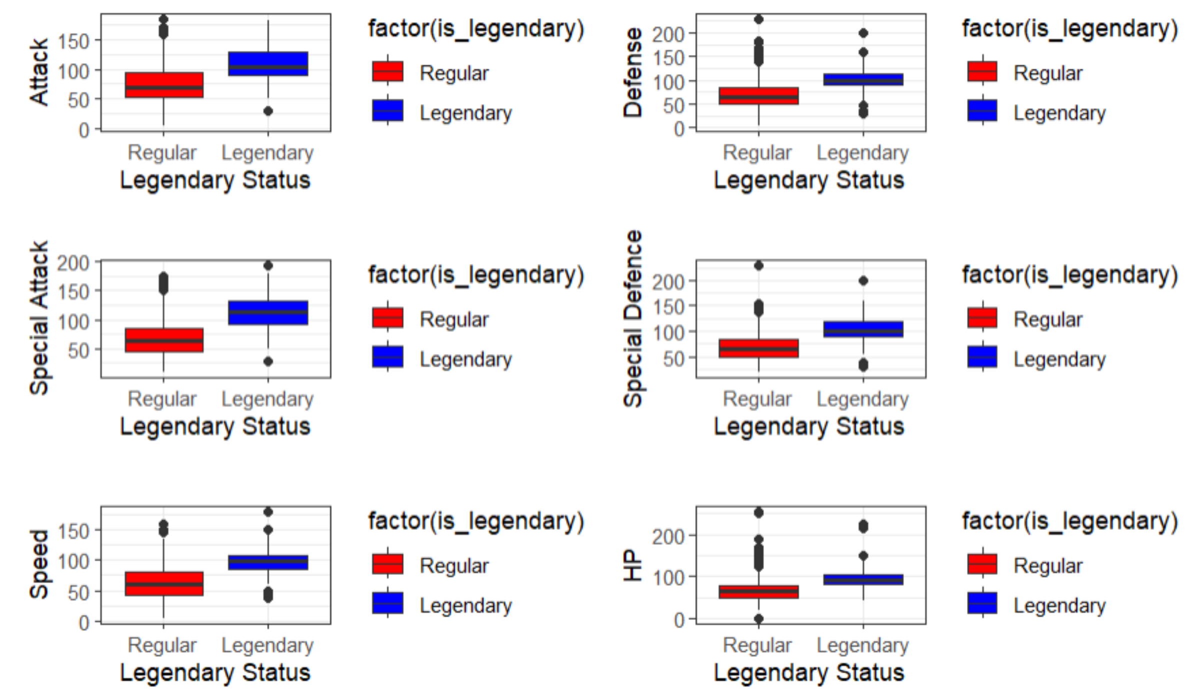
2.5.3.4 Boxplots of all Metrics

The following box plot diagrams illustrate the distribution of various attributes for both regular and legendary Pokémon. Each plot consists of two boxes, one for Regular Pokémon (red) and one for Legendary Pokémon (blue). The boxes represent the interquartile range (IQR), while the lines extending from the boxes indicate the range of the data excluding outliers. The dots represent individual data points.

Overall, the plots suggest that Legendary Pokémon tend to have higher values for most attributes compared to Regular Pokémon. This is particularly evident for Attack, Special Attack, and Speed, where the median and upper quartile of Legendary Pokémon are significantly higher.

```r
attack <- ggplot(na.omit(df), aes(x = factor(is_legendary), y = attack, fill
= factor(is_legendary))) +
 geom_boxplot() +
 labs(title = "", x = "Legendary Status", y = "Attack") +
 scale_x_discrete(labels = c("Regular", "Legendary")) +
 scale_fill_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary")) +
 theme_bw()

defence <- ggplot(na.omit(df), aes(x = factor(is_legendary), y = defense,
fill = factor(is_legendary))) +
 geom_boxplot() +
 labs(title = "", x = "Legendary Status", y = "Defense") +
 scale_x_discrete(labels = c("Regular", "Legendary")) +
 scale_fill_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary")) +
 theme_bw()

sp_attack <- ggplot(na.omit(df), aes(x = factor(is_legendary), y =
sp_attack, fill = factor(is_legendary))) +
 geom_boxplot() +
 labs(title = "", x = "Legendary Status", y = "Special Attack") +
 scale_x_discrete(labels = c("Regular", "Legendary")) +
 scale_fill_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary")) +
 theme_bw()

sp_defence <- ggplot(na.omit(df), aes(x = factor(is_legendary), y =
sp_defense, fill = factor(is_legendary))) +
 geom_boxplot() +
 labs(title = "", x = "Legendary Status", y = "Special Defence") +
 scale_x_discrete(labels = c("Regular", "Legendary")) +
 scale_fill_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary")) +
 theme_bw()
```

```
speeed <- ggplot(na.omit(df), aes(x = factor(is_legendary), y = speed, fill
= factor(is_legendary))) +
 geom_boxplot() +
 labs(title = "", x = "Legendary Status", y = "Speed") +
 scale_x_discrete(labels = c("Regular", "Legendary")) +
 scale_fill_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary")) +
 theme_bw()

hpp <- ggplot(na.omit(df), aes(x = factor(is_legendary), y = hp, fill =
factor(is_legendary))) +
 geom_boxplot() +
 labs(title = "", x = "Legendary Status", y = "HP") +
  scale_x_discrete(labels = c("Regular", "Legendary")) +
  scale_fill_manual(values = c("1" = "blue", "0" = "red"), labels =
c("Regular", "Legendary")) +
  theme_bw()

grid.arrange(attack, defence, sp_attack, sp_defence, speeed, hpp, ncol = 2)
```



2.6 Utilizing Machine Learning to Predict Which Pokemon are Legendary?

2.6.1 Splitting the Data into Training and Testing

Training Data: 534 Pokémon

Testing Data: 267 Pokémon

```
training_data <- sample(1:nrow(df), 2 * nrow(df) / 3)
testing_data <- setdiff(1:nrow(df), training_data)

legendary_test <- df$is_legendary[testing_data]
```

2.6.2 Decision Tree Algorithm

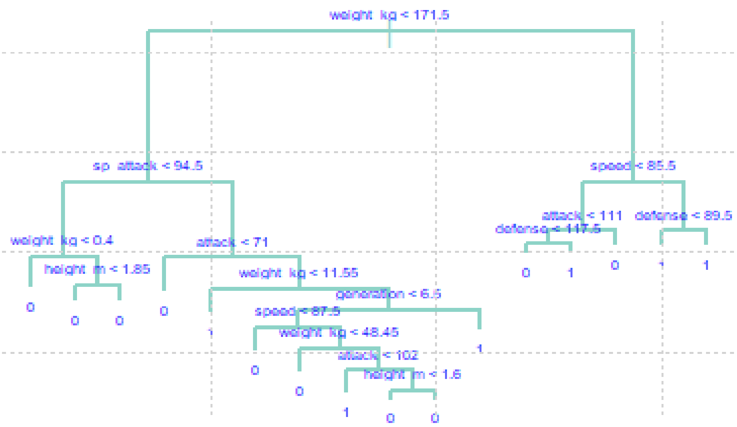
“Decision tree learning is a supervised learning approach used in statistics, data mining and machine learning. In this formalism, a classification or regression decision tree is used as a predictive model to draw conclusions about a set of observations. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity” (https://en.wikipedia.org/wiki/Decision_tree_learning).

The decision tree for predicting whether Pokémon are legendary begins at the root with the special attack attribute. It determines whether the Pokémon’s special attack stat is greater or less than 71.5. If it’s less than 71.5, the Pokémon is automatically deemed to not be legendary. If it is, the tree branches off into different attributes including defense and speed. Special attack is the most important metric, as it forms the root of the tree, it’s the first gate which determines whether a Pokémon should be evaluated further. Defense and speed are also important metrics for how this model determines if a Pokémon is legendary.

Ultimately, this systematic breakdown navigates through various metrics, with each one acting as a gate leading to the Pokémon’s classification. Each decision point in the tree filters the Pokémon through key criteria, with only those that meet all the thresholds being deemed legendary.

Accuracy of Decision Tree Model: 0.895

```
set.seed(200)

df$is_legendary <- as.factor(df$is_legendary)

pokemon_tree <- tree(is_legendary ~ ., data = df[training_data, ], na.action
= na.omit)

## Warning in tree(is_legendary ~ ., data = df[training_data, ], na.action =
## na.omit): NAs introduced by coercion

colors <- brewer.pal(3, "Set3")
plot(pokemon_tree, col = colors, lwd = 2, main = "Decision Tree for
Legendary Pokémon Prediction")
text(pokemon_tree, pretty = 0, col = "blue", cex = 0.45)
grid()
```



```
predictions <- predict(pokemon_tree, newdata = df[testing_data, ], type =
"class")

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by
coercion

confusion_matrix <- table(predictions, legendary_test)

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

cat("Accuracy:", accuracy, "\n")

## Accuracy: 0.906367
```

2.6.3 Random Forest Algorithm

“Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that works by creating a multitude of decision trees during training. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the output is the average of the predictions of the trees (https://en.wikipedia.org/wiki/Random_forest).”

```
df$is_legendary <- as.factor(df$is_legendary)

pokemon_rf <- randomForest(is_legendary ~ ., data = df[training_data, ],
importance = TRUE, na.action = na.omit, type = "classification")

print(pokemon_rf)
```

```
##
## Call:
##  randomForest(formula = is_legendary ~ ., data = 
df[training_data,      ], importance = TRUE, type = "classification", 
na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 6.76%
## Confusion matrix:
##     0  1 class.error
## 0 466  6  0.01271186
## 1  29 17  0.63043478
```

The confusion matrix indicates that:

- 477 instances were correctly classified as class 0.
- 4 instances were incorrectly classified as class 0 (false negatives).
- 29 instances were incorrectly classified as class 1 (false positives).
- 12 instances were correctly classified as class 1.

The class error rates for each class are also provided:

- Class 0: 0.00845666 (approximately 0.83%)
- Class 1: 0.65909091 (approximately 68.29%)

Accuracy = (True Positives + True Negatives) / (Total Instances)

Accuracy = (12 + 477) / 522 ≈ 0.9377

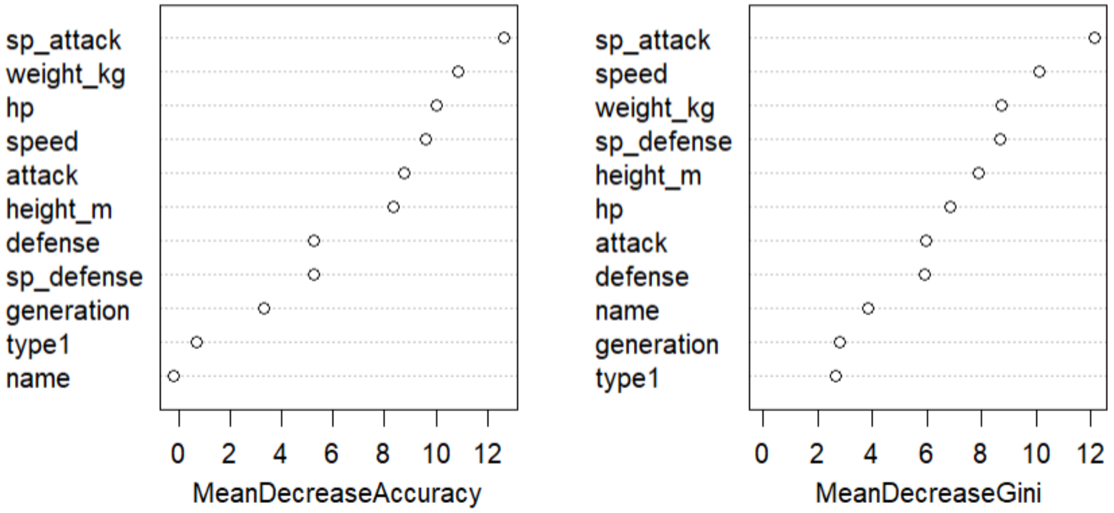
Accuracy of Random Forest Model: 0.937

2.6.3.1 Most Important Metrics

- *Mean Decrease Accuracy*: a metric that measures how much a model’s accuracy decreases when a variable is removed.
- *Mean Decrease Gini*: measures how important a variable in a random forest model, by quantifying how much it contributes to the homogeneity of the model’s nodes.

As shown by the output below, the most crucial three metrics used by the Random Forest model to successfully predict whether a Pokémon are special attack, weight and speed.

```
varImpPlot(pokemon_rf)
```



3. Conclusion

In conclusion, as shown by the ROC Curve graph below, the Random Forest model (blue) is clearly more accurate than the Decision Tree model (red). The closer the curve is to the top left corner, the more accurate and efficient the model is. The Random Forest model achieved higher sensitivity and specificty compared to the other one, which thus indicates that it’s more efficient at correctly classifying both legendary and regular Pokémon.

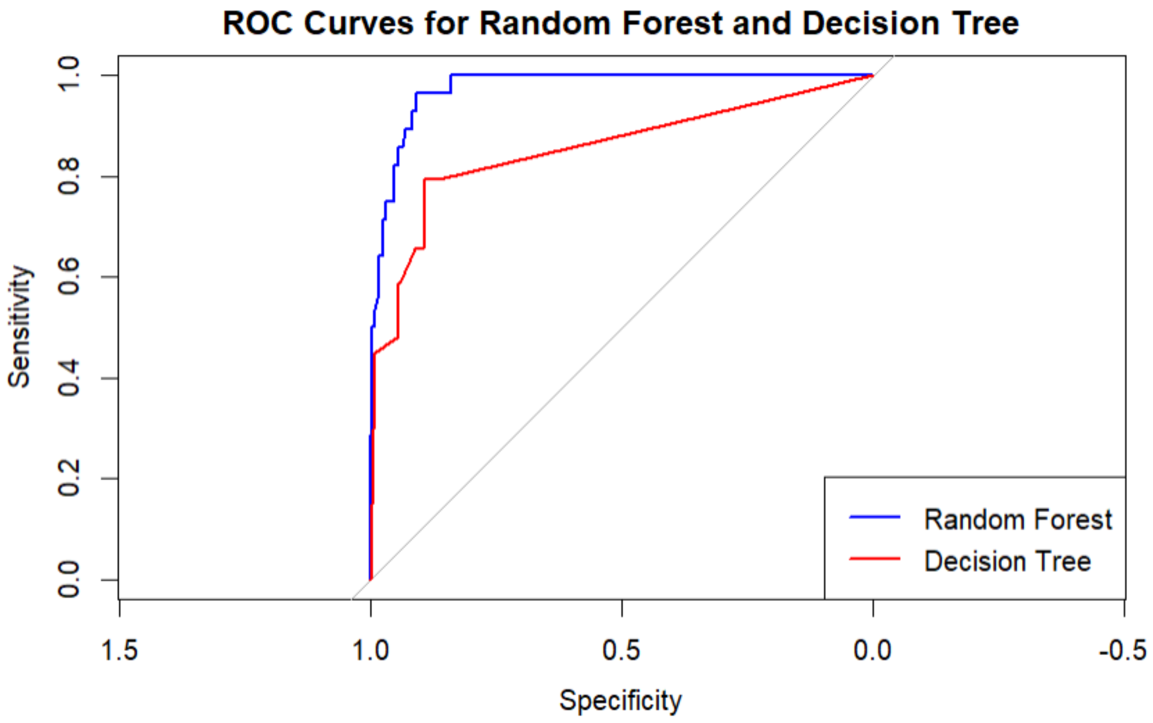
The higher accuracy, is due to the nature of the Random Forest algorithm, which combined multiple trees which produced more reliable predictions; the Decision Tree algorithm is much more simplistic and lacks the robustness as Random Forest. Through comparing two different models, it highlighted the importance of model selection and the impact of algorithm complexity on predictive performance.

The most crucial metrics across both models in determining Legendary Pokémon are the following:

- Special Attack
- Weight (KG)
- Speed

Overall this simplistic project explored and introduced the fundamental concepts of data analysis, visualization and machine learning with R. It acted as a simplistic gateway offering practical, hands-on experience in applying these techniques. The skills and insights gained here lay a solid foundation for more advanced explorations in the exciting fields of data analysis and data science.

```r
# Predict probabilities for both models
rf_prob <- predict(pokemon_rf, df[testing_data, ], type = "prob")[, 2]
tree_prob <- predict(pokemon_tree, df[testing_data, ], type = "vector")[, 2]

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by
coercion

# Generate ROC curves
rf_roc <- roc(df$is_legendary[testing_data], rf_prob)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

tree_roc <- roc(df$is_legendary[testing_data], tree_prob)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# Plot ROC curves
plot(rf_roc, col = "blue", main = "ROC Curves for Random Forest and Decision
Tree")
lines(tree_roc, col = "red")
legend("bottomright", legend = c("Random Forest", "Decision Tree"), col =
c("blue", "red"), lwd = 2)
```


ROC Curves for Random Forest and Decision Tree
Sensitivity
Specificity
Random Forest
Decision Tree