Classifying Legendary Pokemons According to their in-game Stats

Dylan Tartarini

INTRO

Pokemon is a classic turn-based RPG: a pokemon has to defeat the opposing one according to the player's indications, and the outcome of a match will be established both by pokemon's intrinsic characteristics as well as strategic components like the abilities of a certain specie of pokemon, the player's strategy and the moveset given to the creature. Overall, the game got more and more complex since its debut and the competitive meta-game is often revised and improved by Pokemon's creators, the members of Game Freak.

One thing that remained the same over the years is one simple fact: creatures which are classified as *Legendaries* are, apart from a few exceptions, far stronger than normal pokemons. The features of these particular creatures make them so strong that strategy alone usually isn't enough to win a duel against them, and they are usually banned from official competitions.

Using a dataset freely available on Kaggle (https://www.kaggle.com/cristobalmitchell/pokedex) and on my GitHub (https://github.com/DylanTartarini1996), we want to establish whether or not a given pokemon should be classified as Legendary, with a comparison between two methodologies: **Logistic Regression** and **Decision Trees**. Some knowledge of both R and Statistics is assumed.

SETUP

Loading needed packages..

```
library(ggplot2)

library(gplots)

library(readr)

library(rpart)

library(tree)

library(ROCR)

library(rpart.plot)

library(readxl)

library(plyr)

library(tidyr)

library(naniar)

library(dplyr)

library(ggrepel)

library(gridExtra)
```

```
library(colortools)
```

Downloading Dataset..

```
pkdf <- read_excel("C:/Users/Utente/Google Drive/pkmn analysis/pokedex.xlsx")
pkdf<- data.frame(pkdf)
colnames(pkdf)

##  [1] "national_number"    "gen"                "english_name"
##  [4] "japanese_name"      "primary_type"       "secondary_type"
##  [7] "classification"     "percent_male"       "percent_female"
## [10] "height_m"           "weight_kg"          "capture_rate"
## [13] "base_egg_steps"     "hp"                 "attack"
## [16] "defense"            "sp_attack"          "sp_defense"
## [19] "speed"              "abilities_0"        "abilities_1"
## [22] "abilities_2"        "abilities_hidden"   "against_normal"
## [25] "against_fire"       "against_water"      "against_electric"
## [28] "against_grass"      "against_ice"        "against_fighting"
## [31] "against_poison"     "against_ground"     "against_flying"
## [34] "against_psychict"   "against_bug"        "against_rock"
## [37] "against_ghost"      "against_dragon"     "against_dark"
## [40] "against_steel"      "against_fairy"      "is_sublegendary"
## [43] "is_legendary"       "is_mythical"        "evochain_0"
## [46] "evochain_1"         "evochain_2"         "evochain_3"
## [49] "evochain_4"         "evochain_5"         "evochain_6"
## [52] "gigantamax"         "mega_evolution"     "mega_evolution_alt"
## [55] "description"

nrow(pkdf)

## [1] 898
```

Dataset is composed by 898 observations: that is to say, we got 898 creatures in our dataset.

```
sum(pkdf$is_legendary)

## [1] 20

sum(pkdf$is_sublegendary)

## [1] 45

sum(pkdf$is_mythical)

## [1] 20
```

In the Dataset, there are 20 creatures classified as legendaries, 45 as sub-legendaries and 20 as mythical: for the sake of simplicity we consider them all as *Legendary*.

```
pkdf <- pkdf %>% mutate(Legendary = case_when(is_legendary == 1 ~ 1L,
                                              is_sublegendary == 1 ~ 1L,
                                              is_mythical == 1 ~ 1L))
sum(pkdf$Legendary, na.rm = T)

## [1] 85

pkdf[is.na(pkdf)] = 0L
```

Before going through with the analysis, it might be useful to define a new column, *statTOT* as the sum of the Stats of a given pokemon. Then, we select only the columns we plan to utilize in our analysis.

```
pkdf <- pkdf %>% mutate(statTOT = hp + attack + sp_attack + defense + sp_defense + speed)

pkdf <- select(pkdf, national_number, english_name, gen, primary_type, secondary_type, statTOT, hp,
attack, defense,
               sp_attack, sp_defense, speed, Legendary)
head(pkdf)
```

```
##   national_number english_name gen primary_type secondary_type statTOT hp
## 1               1    Bulbasaur   I        grass         poison     318 45
## 2               2      Ivysaur   I        grass         poison     405 60
## 3               3     Venusaur   I        grass         poison     525 80
## 4               4   Charmander   I         fire              0     309 39
## 5               5   Charmeleon   I         fire              0     405 58
## 6               6    Charizard   I         fire         flying     534 78
##   attack defense sp_attack sp_defense speed Legendary
## 1     49      49        65         65    45         0
## 2     62      63        80         80    60         0
## 3     82      83       100        100    80         0
## 4     52      43        60         50    65         0
## 5     64      58        80         65    80         0
## 6     84      78       109         85   100         0
```

Exploratory Data Analysis

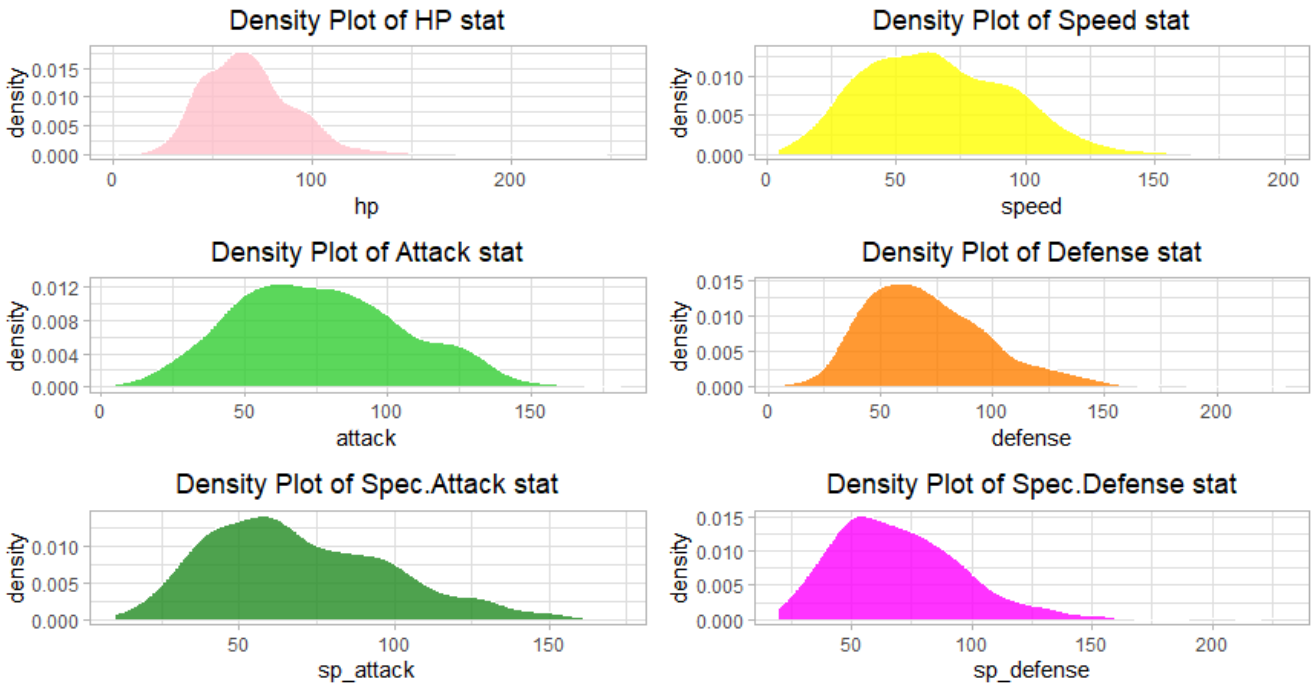
Pokemon is a classic turn-based RPG: a pokemon has to defeat the opposing one according to the player's indications, and the outcome of a match will be established both by pokemon's intrinsic characteristics as well as strategic components like the abilities of a certain specie of pokemon, the player's strategy and the moveset given to the creature. Overall, the game got more and more complex since its debut and the meta-game is often revised and improved by Pokemon's creators, the members of Game Freak.

Here, we are not considering any feature like the moveset of a pokemon, their peculiar abilities, or any other strategic component of the game. The only features we are interested in are measurable ones, and we want to determine whether or not a particular pokemon should be classified as Legendary according to only its *brute strength*.

In game, "brute strength" is given by a pokemon's *statistics*: Hit Points , Attack, Defense, Special Attack and Special Defense. Lastly, Speed is the characteristic which decides whoever moves first during a particular turn. We can visualize distribution of those statistics using a **Density Plot**:

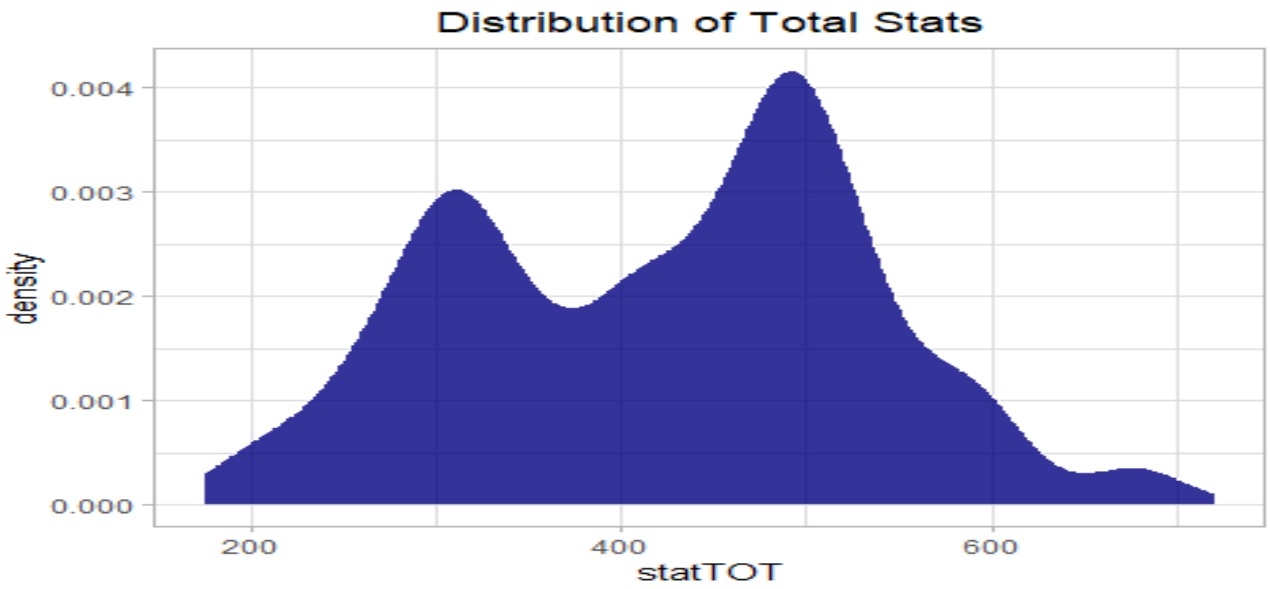
```
density_hp <- ggplot(data = pkdf, aes(hp)) +
  geom_density(col="white",fill="pink", alpha=0.8) +
  ggtitle("Density Plot of HP stat")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

density_speed <- ggplot(data = pkdf, aes(speed)) +
  geom_density(col="white",fill="yellow1", alpha=0.8) +
  ggtitle("Density Plot of Speed stat")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

density_atk <- ggplot(data = pkdf, aes(attack)) +
  geom_density(col="white",fill="limegreen", alpha=0.8) +
  ggtitle("Density Plot of Attack stat")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))
```

```
density_def <- ggplot(data = pkdf, aes(defense)) +
  geom_density(col="white",fill="darkorange1", alpha=0.8) +
  ggtitle("Density Plot of Defense stat")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

density_spatk <- ggplot(data = pkdf, aes(sp_attack)) +
  geom_density(col="white",fill="forestgreen", alpha=0.8) +
  ggtitle("Density Plot of Spec.Attack stat")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

density_spdef <- ggplot(data = pkdf, aes(sp_defense)) +
  geom_density(col="white",fill="magenta", alpha=0.8) +
  ggtitle("Density Plot of Spec.Defense stat")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

grid.arrange(density_hp, density_speed, density_atk, density_def, density_spatk, density_spdef,
ncol=2)
```



Clearly, the peaks of a Density Plot help display where values are concentrated over the interval. We can observe that it is quite rare observing Stats higher than 150. Plotting the Density Plot of the total stat distribution..

```
ggplot(data = pkdf, aes(statTOT))+
  geom_density(col = "white", fill = "navy", alpha = 0.8)+
  ggtitle("Distribution of Total Stats")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))
```



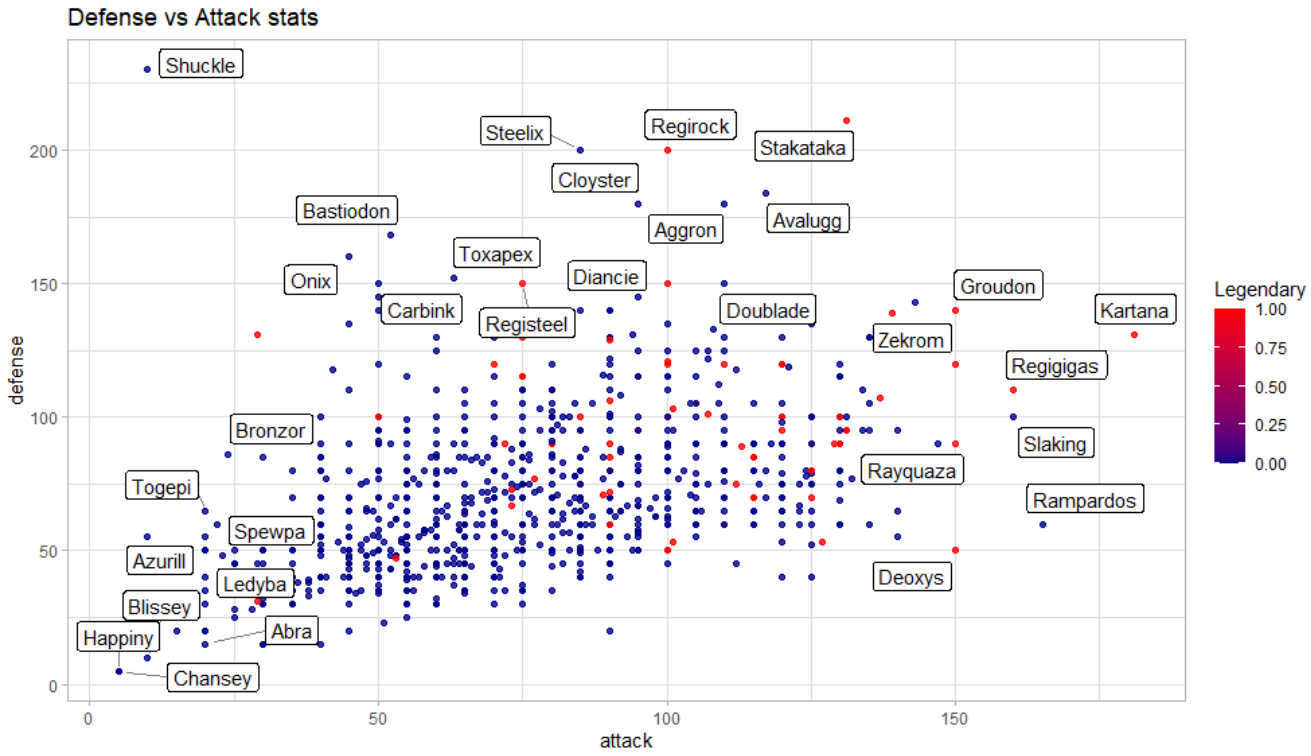
.. we can see that only a few pokemon have a stat total exceeding 600, while the great majority of the observations lies somewhere between 300 and 500.

This might be a clue that *strength is positively related with rarity* in the Pokemon world, and that looking for the strongest creatures is the right road to follow when trying to distinguish between Legendaries and normal pokemons.

In fact, it can be shown that while for non-legendaries (excluding some anomalies) it exists a positive linear relationship between Attack and Defense stats, this is not the case for Legendaries, who tend to follow a more complex pattern. The same can be seen for the relationship between Special Attack and Special Defense.
In the following **Scatter Plots**, Legendaries are depicted in red, while non-legendaries are in blue.

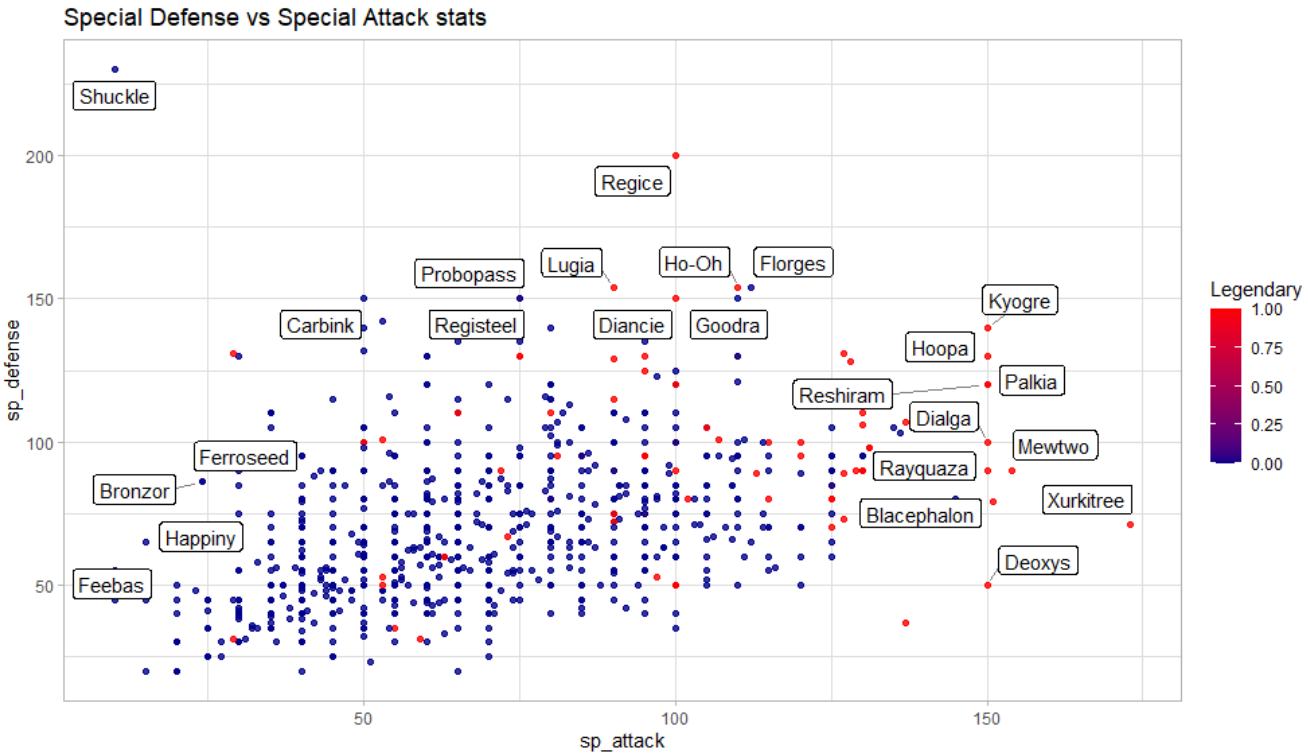
```
ggplot(data=pkdf, aes(x = attack,  y= defense))  +
  geom_point(aes(color=Legendary), alpha=0.8) +
  scale_color_gradient(low="darkblue", high="red") +
  ggtitle("Defense vs Attack stats") +
  theme(plot.title = element_text(hjust = 0.5))+
  theme_light()+
  geom_label_repel(data=subset(pkdf,attack > 150 | defense >150 | attack < 25),
                   aes(label=english_name), box.padding = 0.35, point.padding = 0.5,
                   segment.color = 'grey50')
```

```
## Warning: ggrepel: 17 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
ggplot(data=pkdf, aes(x = sp_attack,  y= sp_defense))  +
  geom_point(aes(color=Legendary), alpha=0.8) +
  scale_color_gradient(low="darkblue", high="red") +
  ggtitle("Special Defense vs Special Attack stats") +
  theme(plot.title = element_text(hjust = 0.5))+
  theme_light()+
  geom_label_repel(data=subset(pkdf, sp_attack > 150 | sp_defense >150 | sp_attack < 25),
                   aes(label=english_name), box.padding = 0.35, point.padding = 0.5,
                   segment.color = 'grey50')
```

```
## Warning: ggrepel: 16 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



So, apart from a few anomalies, *Legendaries tend to have higher Attack, Defense, Special Attack and Special Defense than normal pokemon*. You'll probably want to choose Deoxys over Caterpie when it comes to winning a duel.

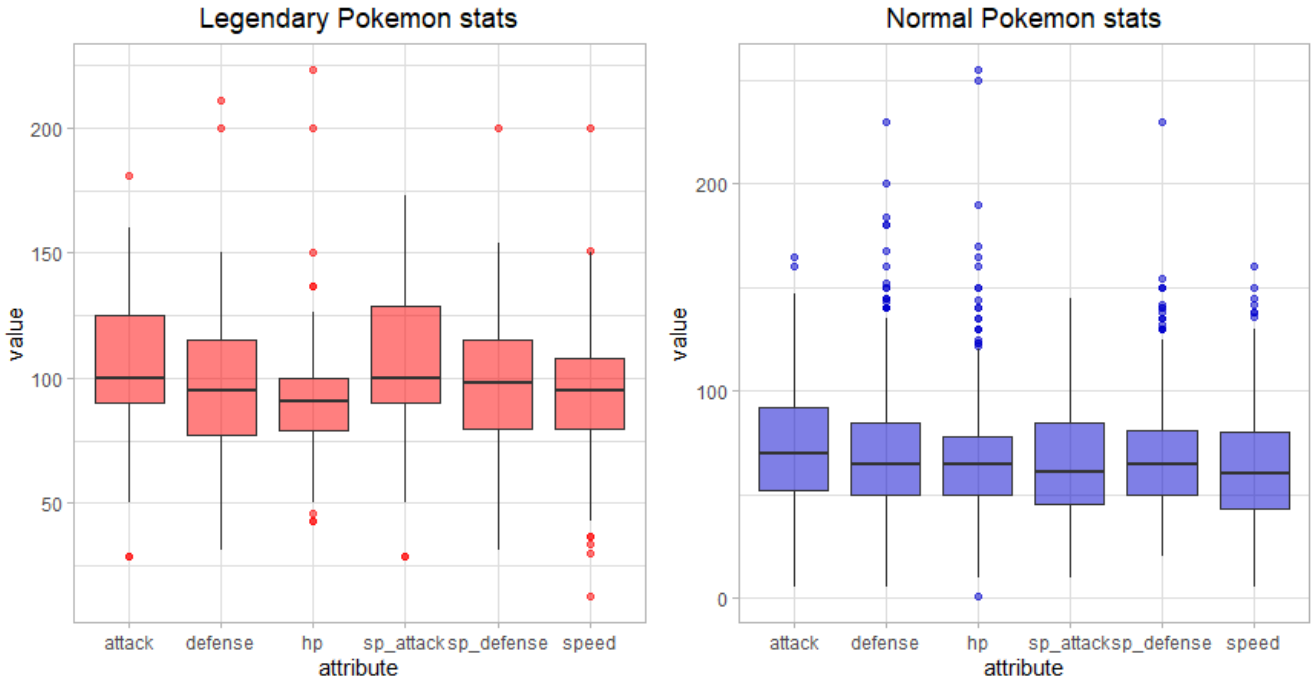
The last piece of data visualization here consists in a couple **Box Plots** establishing what we already guessed.

Box Plots are a way to show spreads and centers of a data set. Measures of spread include the interquartile range and the mean of the data set. Measures of center include the mean or average and median (the middle of a data set).

The black line indicates the median. The box indicates the interquartile range which represents the 50% data in the middle, while. the whiskers extend from either side of the box, representing the ranges for the bottom 25% and the top 25% of the data values, excluding outliers. We can therefore see that **the median of all the Stats of Legendaries is higher that non-legendary pokemons**.

```
box_plot_attr <- select(pkdf, primary_type, Legendary, hp, defense, attack, sp_attack, sp_defense,
speed)
box_plot_attr_leg <- filter(box_plot_attr, Legendary == 1)
box_plot_attr_nor <- filter(box_plot_attr, Legendary == 0)
box_plot_attr_leg_long <- gather(box_plot_attr_leg, attribute, value, -c(primary_type, Legendary))
box_plot_attr_nor_long <- gather(box_plot_attr_nor, attribute, value, -c(primary_type, Legendary))

bp_leg <- ggplot(data = box_plot_attr_leg_long, aes(attribute, value)) +
  geom_boxplot(fill="red", alpha = 0.5, outlier.color = "red", outlier.fill = "red") +
  ggtitle("Legendary Pokemon Stats")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

bp_nor <- ggplot(data = box_plot_attr_nor_long, aes(attribute, value)) +
```

```
  geom_boxplot(fill = "mediumblue", alpha = 0.5, outlier.color = "mediumblue", outlier.fill =
"mediumblue") +
  ggtitle("Normal Pokemon Stats")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))

grid.arrange(bp_leg, bp_nor,ncol=2)
```



We also notice that there are a lot more outliers in the HP stat of non-legendary pokemons as compared to Stats. But on the other hand the same HP stat has the least variation as compared to other Stats. Of course, because of their higher number in the dataset, non-legendaries show (Stats-wise) more outliers.

CLASSIFICATION

After Exploratory Data Analysis, we now know where to start for classifying Legendary and non-legendary pokemons employing their Stats as relevant variables.
The next step will be to run both a *Logistic Regression Model* and a *Decision Tree Model*, to see which of them performs better in the classification task.

LOGIT

Logistic Regression is used to determine the **probability** of an event happening or not. It is used when the dependent variable is categorical. Classical examples of Logistic Regression usage are determining the sex of a person (M/F) according to some features like weight and height or deciding to concede a loan (Yes/No) according to the credit history of a client.
We will use it to predict whether or not a certain pokemon should be classified as Legendary utilizing its Stats as predictors.

```
logit_model <- glm(Legendary ~ attack + defense + sp_attack + sp_defense + speed + hp,
                   data = pkdf, family = "binomial")
```

With the function *summary()*, we can verify that all the variables used are significant.

```
summary(logit_model) #taking the estimates from the model..

##
## Call:
## glm(formula = Legendary ~ attack + defense + sp_attack + sp_defense +
##     speed + hp, family = "binomial", data = pkdf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8120  -0.2754  -0.0754  -0.0135   5.0048
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -19.051993   1.784443 -10.677  < 2e-16 ***
## attack        0.023400   0.007010   3.338 0.000844 ***
## defense       0.039618   0.006923   5.722 1.05e-08 ***
## sp_attack     0.039740   0.006775   5.865 4.48e-09 ***
## sp_defense    0.028992   0.006963   4.163 3.13e-05 ***
## speed         0.036711   0.006666   5.507 3.65e-08 ***
## hp            0.028176   0.007114   3.961 7.48e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 562.47  on 897  degrees of freedom
## Residual deviance: 270.77  on 891  degrees of freedom
## AIC: 284.77
##
## Number of Fisher Scoring iterations: 8
```

We can also determine how much the probability of a pokemon being Legendary increases when we raise one of its Stats by one point:

```
atk_percent_increase <- exp(0.023400)-1
print(paste("increasing ATK by 1 point increases probability of a pkmn of being legendary by",
(round(atk_percent_increase*100, digits=2)),"%"))

## [1] "increasing ATK by 1 point increases probability of a pkmn of being
legendary by 2.37 %"

def_percent_increase <- exp(0.039618)-1
print(paste("increasing DEF by 1 point increases probability of a pkmn of being legendary by",
(round(def_percent_increase*100, digits=2)),"%"))

## [1] "increasing DEF by 1 point increases probability of a pkmn of being
legendary by 4.04 %"

spatk_percent_increase <- exp(0.039740)-1
print(paste("increasing Sp.ATK by 1 point increases probability of a pkmn of being legendary by",
(round(spatk_percent_increase*100, digits=2)),"%"))

## [1] "increasing Sp.ATK by 1 point increases probability of a pkmn of being
legendary by 4.05 %"

spdef_percent_increase <- exp(0.028992)-1
print(paste("increasing Sp.DEF by 1 point increases probability of a pkmn of being legendary by",
(round(spdef_percent_increase*100, digits=2)),"%"))

## [1] "increasing Sp.DEF by 1 point increases probability of a pkmn of being
legendary by 2.94 %"
```

```r
speed_percent_increase <- exp(0.036711)-1
print(paste("increasing SPEED by 1 point increases probability of a pkmn of being legendary by",
(round(speed_percent_increase*100, digits=2)),"%"))
```

```
## [1] "increasing SPEED by 1 point increases probability of a pkmn of being
legendary by 3.74 %"
```

```r
hp_percent_increase <- exp(0.028176)-1
print(paste("increasing HP by 1 point increases probability of a pkmn of being legendary by",
(round(hp_percent_increase*100, digits=2)),"%"))
```

```
## [1] "increasing HP by 1 point increases probability of a pkmn of being
legendary by 2.86 %"
```

Then, we use the model to obtain *predictions* for the probability of a creature of being Legendary and the resulting true and false positive rates:

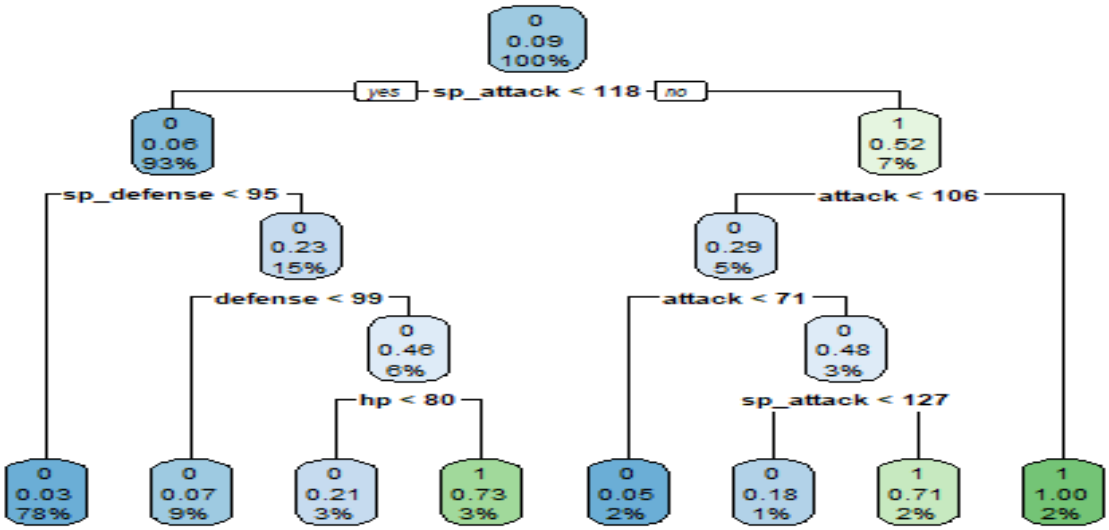
```r
predict_logit <- predict(logit_model, pkdf, type = "response")

logit_prediction <- prediction(predict_logit, pkdf$Legendary)

performance_logit <- performance(logit_prediction,"tpr","fpr")  #true and false positive rates..
```

DECISION TREE

A decision tree is a tool used in classification and prediction. It uses an easy to understand diagram which presents the test and outputs two outcomes. This pattern is continued until there is a reasonable prediction outcome for the test.

The main strength of Trees is that they are very easy to understand, way more than Logistic Regression and other classification methods, thanks to their visual representation of the problem they are trying to solve: some people also thinks that they are a good proxy of human-like decision processes!

```r
tree_model <- rpart(Legendary ~ attack + defense + sp_attack + sp_defense + speed + hp,
                    data = pkdf, method = "class")
rpart.plot(tree_model)
```



We can use the Tree to obtain predictions, just like we did with Logit:

```
predict_tree <- predict(tree_model, pkdf, type = "prob")

tree_prediction <- prediction(predict_tree[,2], pkdf$Legendary)

performance_tree <- performance(tree_prediction,"tpr","fpr")
```

Model Comparison

Now, the only remaining task to do is to compare the two models according to their performances. One non-systematic way to do so is to check out some of the models's predictions and to see whether the models are giving the same answers.

For example….

```
print(paste("Caterpie has",(round(predict_logit[10]*100, digits=2)),"% probability of being
classified as Legendary according to Logit model"))
```

```
## [1] "Caterpie has 0 % probability of being classified as Legendary according to
Logit model"
```

```
print(paste("Caterpie has",(round(predict_tree[10,2]*100, digits=2)),"% probability of being
classified as Legendary according to the Decision Tree"))
```

```
## [1] "Caterpie has 2.87 % probability of being classified as Legendary according
to the Decision Tree"
```

```
print(paste("Arcanine has",(round(predict_logit[59]*100, digits=2)),"% probability of being
classified as Legendary according to Logit model"))
```

```
## [1] "Arcanine has 27.06 % probability of being classified as Legendary
according to Logit model"
```

```
print(paste("Arcanine has",(round(predict_tree[59,2]*100, digits=2)),"% probability of being
classified as Legendary according to the Decision Tree"))
```

```
## [1] "Arcanine has 2.87 % probability of being classified as Legendary according
to the Decision Tree"
```

```
print(paste("Mewtwo has",(round(predict_logit[150]*100, digits=2)),"% probability of being
classified as Legendary according to Logit model"))
```

```
## [1] "Mewtwo has 97.28 % probability of being classified as Legendary according
to Logit model"
```

```
print(paste("Mewtwo has",(round(predict_tree[150,2]*100, digits=2)),"% probability of being
classified as Legendary according to the Decision Tree"))
```

```
## [1] "Mewtwo has 100 % probability of being classified as Legendary according to
the Decision Tree"
```

```
print(paste("Deoxys has",(round(predict_logit[386]*100, digits=2)),"% probability of being
classified as Legendary according to Logit model"))
```

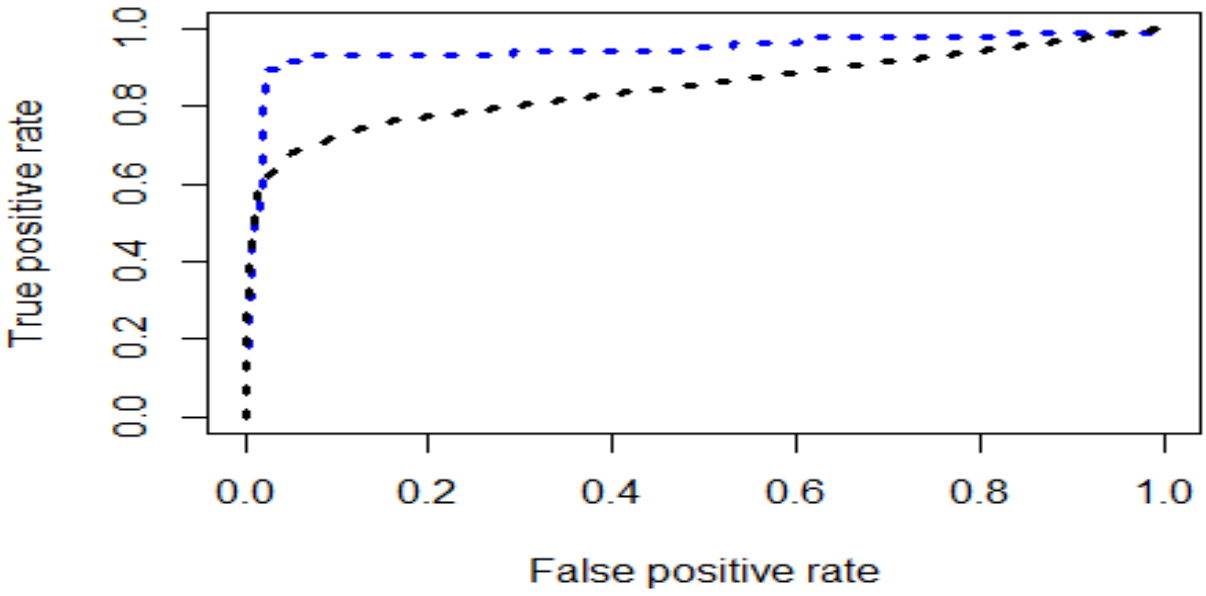
```
## [1] "Deoxys has 68.24 % probability of being classified as Legendary according
to Logit model"
```

```
print(paste("Deoxys has",(round(predict_tree[386,2]*100, digits=2)),"% probability of being
classified as Legendary according to the Decision Tree"))
```

```
## [1] "Deoxys has 100 % probability of being classified as Legendary according to
the Decision Tree"
```

Both models seem to work fairly well.
One more systematic way to compare model performances is to plot their true and false positive rates, like we do in the graph below, where Logit is depicted in blue and the Decision Tree is in black, it should be easy to establish that Logit Regression performs better in this particular classification task.

```
plot(performance_logit,col="blue",lty=3, lwd=3)      #Logit model performs better
plot(performance_tree,col="black",lty=3, lwd=3, add=TRUE)
```



Bonus: New Generation

As any fan will know, new creatures are added to the dataset (the *PokeDex*) any few years for renewing the franchise, and they come in "*Generations*": the last one added is Generation 8, and it brought to the table 89 new pokemons. So, it can make sense adding the new creatures to the old dataset used for training models and see how the new observations are classified. We will use this knowledge to split our data in train and test set, and see if our conclusions about model performances change using a smaller test set.

```
train <- pkdf[1:809, ] #pkmn up to Gen8

gen8 <- pkdf[810:898,]
```

LOGIT (New Gen)
Same as before, we train the Logit Model..

```
logit_model_train <- glm(Legendary ~ attack + defense + sp_attack + sp_defense + speed + hp,
                     data = train, family = "binomial")       #training logit model using Stats
variables of the training set
```

..then, we use the model to obtain *predictions* for the new generation of creatures:

```
predict_logit_gen8 <- predict(logit_model_train, gen8, type = "response")   #predicting the outcomes
for the test set

gen8_logit_prediction <- prediction(predict_logit_gen8, gen8$Legendary) #comparing predictions with
actual values

performance_logit_gen8 <- performance(gen8_logit_prediction,"tpr","fpr") #True and False Positive
rates
```

DECISION TREE (New Gen)

Once again, we train the Tree Model on the training set..

```
tree_model_train <- rpart(Legendary ~ attack + defense + sp_attack + sp_defense + speed + hp,
                     data = train, method = "class")
```

Like we did with Logit Model, we use the Tree we trained on the previous generations of pokemons on the eight generation..

```
predict_tree_gen8 <- predict(tree_model_train, gen8, type = "prob")  #predicting the outcomes for
the test set

gen8_prediction <- prediction(predict_tree_gen8[,2], gen8$Legendary)  #comparing predictions with
actual values

performance_tree_gen8 <- performance(gen8_prediction,"tpr","fpr")  #True and False Positive rates
```

Model Comparison

The first pokemon introduced in the 8th generation is Grookey, which is, by tradition, one of the first pokemons the player gets to start the game; it is, of course, not a legendary.

```
print(paste("Grookey has",(round(predict_logit_gen8[1]*100, digits=2)),"% probability of being
classified as Legendary according to Logit model"))
```

```
## [1] "Grookey has 0 % probability of being classified as Legendary according to
Logit model"
```

```
print(paste("Grookey has",(round(predict_tree_gen8[1,2]*100, digits=2)),"% probability of being
classified as Legendary according to the Decision Tree"))
```

```
## [1] "Grookey has 2 % probability of being classified as Legendary according to
the Decision Tree"
```

The 84th pokemon of the 8th generation instead is Zarude, a creature with a Stat Total of 600:

```
print(paste("Zarude has",(round(predict_logit_gen8[84]*100, digits=2)),"% probability of being
classified as Legendary according to Logit model"))
```

```
## [1] "Zarude has 68.19 % probability of being classified as Legendary according
to Logit model"
```

```
print(paste("Zarude has",(round(predict_tree_gen8[84,2]*100, digits=2)),"% probability of being
classified as Legendary according to the Decision Tree"))
```

```
## [1] "Zarude has 100 % probability of being classified as Legendary according to
the Decision Tree"
```

Finally, we compare the true and false positive rates of the two models: from the graph it should be easy to establish once again that Logit Regression performs better in this particular classification task

```
plot(performance_logit_gen8,col="blue",lty=3, lwd=3)      #Logit model performs better also when
adding a new generation
plot(performance_tree_gen8,col="black",lty=3, lwd=3, add=TRUE)
```