What makes a Pokemon legendary?

Chan Le

3/1/2020

[Articuno, The Ice Legendary Bird. One of my favourite legendary Pokemons.]

I. Introduction

A. The Pocket Monsters Universe

Pokemon (Pocket Monsters) is orginally a series of console video games developed by GameFreak, published by Nintendo, and is later adapted into numerous different media. Since its original work of Pokemon Red and Blue, Pokemon has grown into one of the most well-known media franchise of all time.

The Pokemon universe is based on the fictional creatures of the same name and their relationship with human. In the Pokemon world, Trainers capture wild Pokemons with Pokeballs and raise them as their own partners. Trainers can also battle among each others and challenge Gym Leaders to gain experience, and have one common goal of defeating the Elite Four and the Pokemon Champion, who are said to be the strongest Pokemon Trainers yet.

Up until Generation 6, there are 721 Pokemon species founded in the wild. Some of them have multiple forms (which differ in the distribution of stats), raising the number of all different versions of Pokemon species captured in this data set to 800. From this point onwards, if I mention the number of Pokemon, I also count in all possible forms for one species.

B. Base Stat Values - Pokemon species uniqueness

Principally, a Pokemon is determined by its statistics (stats). There are six main permanent stats that make up a Pokemon's overall strength:

- HP (Hit Points): determine how much damage a Pokémon can receive before fainting.

- Atk (Attack): partly determines how much damage a Pokémon deals when using a physical move.
- Def (Defense): partly determines how much damage a Pokémon receives when it is hit with a physical move.
- SpAtk (Special Attack): partly determines how much damage a Pokémon deals when using a special move.
- SpDef (Special Defense): partly determines how much damage a Pokémon receives when it is hit with a special move.
- Spe (Speed): determines the order of Pokémon that can act in battle.

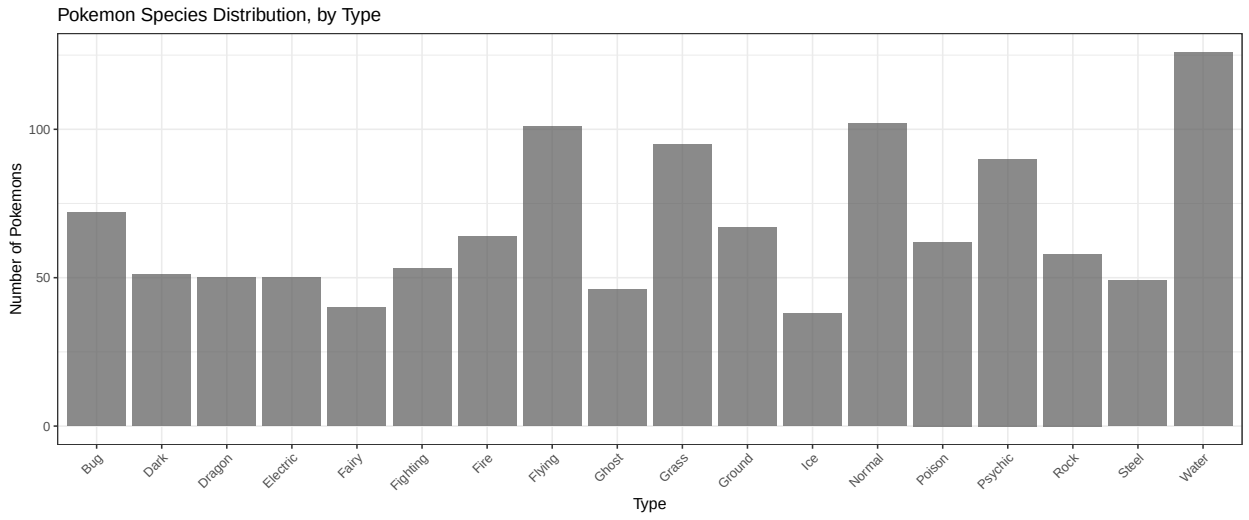
There are many elements that can affect these stats, so that hardly no two Pokemons are identical. However, each Pokemon species will have its own set of base stat values, which in turn has a great impact on a certain individual Pokemon's stats at any level. This table shows the first 5 Pokemon (species) and their base stats:

Name	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed
Bulbasaur	45	49	49	65	65	45
Ivysaur	60	62	63	80	80	60
Venusaur	80	82	83	100	100	80
Mega Venusaur	80	100	123	122	120	80
Charmander	39	52	43	60	50	65

C. Pokemon Types - Another deciding factor

Types are properties for Pokemon and their Moves. As of Generation VI, there are 18 types. One Pokemon can have one or two types. Out of 800 Pokemon species, there are 386 mono-type and 414 dual-type Pokemons. The following plots shows the distribution of Pokemon Species among different types:

```
data %>%
  select(type1, type2) %>%
  pivot_longer(everything(), names_to = "TypeNo", values_to = "Type") %>%
  filter(!Type == "Mono") %>%
  ggplot(aes(x = Type)) + geom_bar(alpha = 0.7) +
  labs(title = "Pokemon Species Distribution, by Type") +
  xlab("Type") + ylab("Number of Pokemons") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



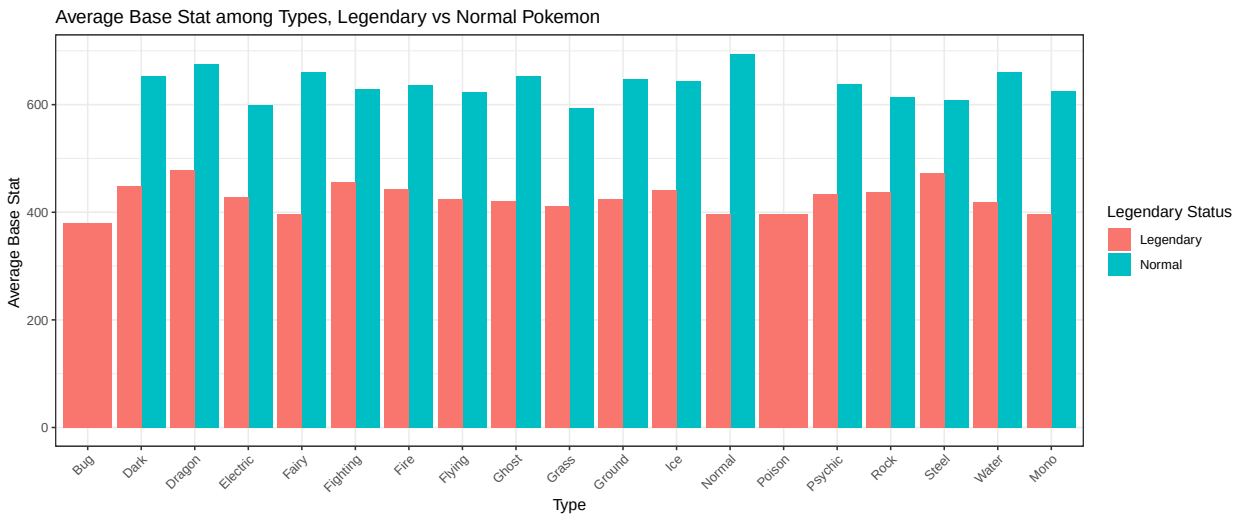
According to the plot, the Pokemon number among spieces is unevenly distributed. Water type accounts for the most Pokemon (126) with 0.1575%, where as Ice Pokemons are the rarest (38), making up only 0.0475% of all species.

D. The Legendary Pokemon

This report aims to explain what makes a Pokemon legendary. So the first question is: what are legendary Pokemons?

Boardly, Pokemons can be divided into two groups: normal and legendary. Legendary Pokemons are incredibly rare and often very powerful, generally featured prominently in the legends and myths of the Pokémon world. In this data set, there are 65 Pokemons counted as legendary. The plot below compares the average accumulated base stats of normal and lengendary Pokemons, with regard to the types:

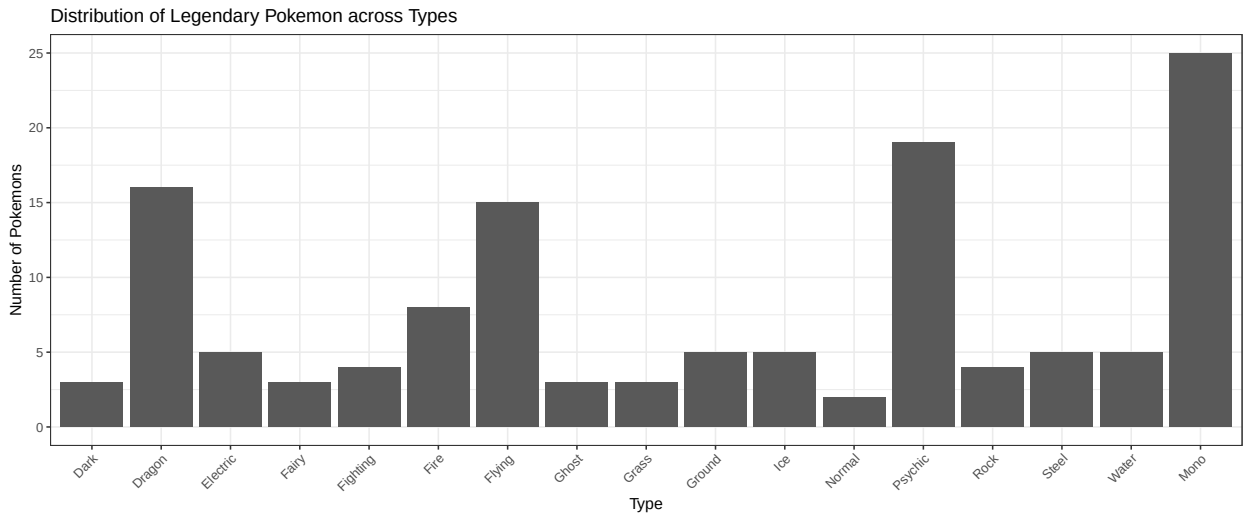
```
data %>% select(-index, -name, - gen) %>%
  pivot_longer(starts_with("type"), names_to = "TypeNo", values_to = "Type") %>%
  mutate(baseStat = hp + atk + def + spAtk + spDef + spe) %>%
  group_by(legendary, Type) %>% summarise(baseStat = mean(baseStat)) %>%
  ggplot(aes(x = Type, fill = legendary, y = baseStat)) +
  geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Average Base Stat among Types, Legendary vs Normal Pokemon") +
    xlab("Type") + ylab("Average Base Stat ") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_fill_discrete(labels = c("Legendary", "Normal"), name = "Legendary Status")
```



We can infer from this plot, that there has yet to exist a Bug or Poison type legendary Pokemon. The plot also confirms the dominant power in base stats of the legendaries across all types. A simple calculation shows that in average, the total base stats of a legendary Pokemon is 34.54% higher than that of a normal Pokemon.

```
data %>% filter(legendary == "True") %>%
  select(type1, type2) %>%
  pivot_longer(everything(), names_to = "TypeNo", values_to = "Type") %>%
  ggplot(aes(x = Type)) + geom_bar() +
```

```
    labs(title = "Distribution of Legendary Pokemon across Types") +
  xlab("Type") + ylab("Number of Pokemons") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



In this plot, I add one column that shows the number of mono-type pokemon. Out of 65 legendary Pokemons, there are 25 that only have one type. Moreover, Psychic, Dragon and Flying are the top 3 types that most legendary Pokemons associate to. This might be helpful later during the process of building models.

II. The Models

A. Feature Selection using Random Forest

Based on the short descriptive analysis, I have some hypothesis about what goes into explaining a Pokemon's legendary status. Their base stats are significantly higher than the normal Pokemons, so all 6 of them should be considered. Furthermore, certain types may also affect the possibility that a Pokemon being legendary (whether it is mono-type, or has Dragon, Flying or Psychic as one of its types).

I build a random forest model. Input-wise, the base stats stay numeric, and the categorical variable `type` is dummified.

```
# Create grid search result data frame
mtryGrid <- seq(2,ncol(dataDummy),1)
ntreeGrid <- seq(500, 3000, 500)

gridSearchRF <- expand.grid(mtryGrid, ntreeGrid)
gridSearchRF <- cbind(gridSearchRF, matrix(nrow = nrow(gridSearchRF), ncol = 3))
colnames(gridSearchRF) <- c("mtry","ntree","OBB","True","False")

# Grid search for best ntree and best mtry
temp <- 1

for (i in ntreeGrid){
  for (j in mtryGrid){
```

```
    modelRF <- randomForest(legendary ~ ., data = dataDummy,
                                      mtry = j, ntree = i)
    result <- modelRF$err.rate %>% as.data.frame() %>%
      summarise_each(funs = mean) %>% as.matrix()
    gridSearchRF[temp,3:5] <- result
    temp <- temp + 1
  }
}

gridSearchRF <- gridSearchRF %>% filter(OBB == min(OBB))
gridSearchRF %>% kable() %>% kable_styling()
```

mtry	ntree	OBB	True	False
3	500	0.0519805	0.0096099	0.5312386

A grid search finds the optimal value for number of tree `ntree` = 500 and number of random candidate splitting variables at each tree node `mtry` = 500, with the minimum average out-of-bag error rate as measure.

```
modelRF <- randomForest(legendary ~ ., data = dataDummy,
                        mtry = gridSearchRF[1,1],
                        ntree = gridSearchRF[1,2], localImp = TRUE)
importance(modelRF) %>% as.data.frame() %>% mutate(variable = rownames(.)) %>%
  select(variable, MeanDecreaseAccuracy,MeanDecreaseGini) %>%
  arrange(desc(MeanDecreaseAccuracy), desc(MeanDecreaseGini)) %>% head(10) %>%
  knitr::kable() %>% kableExtra::kable_styling()
```

variable	MeanDecreaseAccuracy	MeanDecreaseGini
spAtk	20.916470	20.3010803
spe	18.346157	15.9399236
hp	16.776661	13.7611921
atk	13.888539	11.7958583
spDef	13.091262	14.0155512
def	11.542637	11.1772385
type_Flying	6.067267	1.8223348
type_Psychic	5.254547	2.4195106
type_Dragon	5.091836	2.6122767
type_Bug	4.549723	0.5058379

There are two common Importance measures among the potential predictor variables. In other words, they show how important a variable is for explaining the value of the response across all trees in the forest. The Accuracy Importance (Mean Decrease in Accuracy) measures the decrease in the accuracy on the out-of-bag samples of the forest when a varible is excluded, or permuted. The Gini Importance (Mean Decrease in Impurity) is the mean of a variable's total decrease in node impurity across all internal nodes, when the variable is excluded.

The upper table shows the top 10 most important features that contribute to the classification of Pokemon legendary status using random forest model. These will be used as imput variables for the next step, the logistic regression.

B. Logistic Regression - The Proposed Models

I build some candidate model based on a couple of insights and assumptions during my Pokemon playthroughs:

- Some Pokemon are called Physical/Special Walls. It means their Defense and Special Defense stats are correlated and potentially high.
- Pokemon Walls need to be capable of tanking hits. Maybe there are some interactions among HP and the two Defense stats.
- Some Pokemon are called Physical/Special Sweepers. It means their Attack and Special Attack stats are correlated and potentially high.
- It is crucial for Pokemon Sweepers to move first and knock out their opponent. Maybe there are some interactions among Speed and the two Attack stats.

Along with the "favourable" types, the candidate models are display in the following table:

```
modelLogRFormular <-
  data.frame( ModelName = paste0("mod",1:8),
    Formular = c("legendary ~ hp + atk + def + spAtk + spDef + spe",
                 "legendary ~ hp + atk + def + spAtk + spDef + spe + type_Mono",
                 "legendary ~ hp + atk*def + spAtk*spDef + spe + type_Mono",
                 "legendary ~ hp + atk*spAtk + def*spDef + spe + type_Mono",
                 "legendary ~ atk*spAtk + def*hp + hp*spDef + spe + type_Mono",
                 "legendary ~ hp + atk*def + spAtk*spDef + spe + type_Mono +
                 type_Psychic + type_Dragon + type_Flying",
                 "legendary ~ hp + atk*spe + def*spDef + spAtk*spe + type_Mono +
                 type_Psychic + type_Dragon + type_Flying",
                 "legendary ~ hp + atk + def + spAtk + spDef + spe + type_Psychic +
                 type_Dragon + type_Flying + type_Mono"))

modelLogRFormular %>% kable() %>% kable_styling(full_width = FALSE) %>% column_spec(1, width = "4cm")
```

ModelName	Formular
mod1	legendary ~ hp + atk + def + spAtk + spDef + spe
mod2	legendary ~ hp + atk + def + spAtk + spDef + spe + type_Mono
mod3	legendary ~ hp + atk*def + spAtk*spDef + spe + type_Mono
mod4	legendary ~ hp + atk*spAtk + def*spDef + spe + type_Mono
mod5	legendary ~ atk*spAtk + def*hp + hp*spDef + spe + type_Mono
mod6	legendary ~ hp + atk*def + spAtk*spDef + spe + type_Mono + type_Psychic + type_Drag
mod7	legendary ~ hp + atk*spe + def*spDef + spAtk*spe + type_Mono + type_Psychic + type_
mod8	legendary ~ hp + atk + def + spAtk + spDef + spe + type_Psychic + type_Dragon + type

I let R run through the models and compare them based on AICc score:

```
modelList <- list()

# Run all models
for (i in 1:nrow(modelLogRFormular)){
  modelList[[i]] <- glm(as.character(modelLogRFormular[i,2]),
                        family = binomial(link = "logit"), data = dataDummy)
}
```

```r
# Rename models
names <- paste("mod",1:length(modelList),sep = "")

# Compare candidate models
aictab(cand.set = modelList, modnames = names, sort = TRUE)
```

```
##
## Model selection based on AICc:
##
##       K   AICc Delta_AICc AICcWt Cum.Wt     LL
## mod6 13 177.17       0.00   0.71   0.71 -75.35
## mod3 10 179.22       2.05   0.26   0.97 -79.47
## mod5 11 185.72       8.55   0.01   0.98 -81.69
## mod4 10 185.73       8.57   0.01   0.99 -82.73
## mod8 11 186.49       9.32   0.01   1.00 -82.08
## mod7 14 189.16      11.99   0.00   1.00 -80.31
## mod2  8 189.60      12.43   0.00   1.00 -86.71
## mod1  7 191.03      13.86   0.00   1.00 -88.44
```

According to the AICc score comparison, model 6 performs the best. Model 3 is behind, as it is identical to model 6, only minus the three Pokemon species type dummy variables. Let's have a closer look at the result of model 6.

```r
summary(modelList[[6]])
```

```
##
## Call:
## glm(formula = as.character(modelLogRFormular[i, 2]), family = binomial(link = "logit"),
##     data = dataDummy)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.28721  -0.10879  -0.01665  -0.00123   2.16431
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.181e+01  4.233e+00  -7.516 5.65e-14 ***
## hp            5.453e-02  1.356e-02   4.022 5.77e-05 ***
## atk           9.730e-02  2.716e-02   3.582 0.000341 ***
## def           1.220e-01  3.098e-02   3.938 8.21e-05 ***
## spAtk         9.125e-03  2.616e-02   0.349 0.727229
## spDef         9.645e-03  2.730e-02   0.353 0.723886
## spe           5.525e-02  1.167e-02   4.733 2.21e-06 ***
## type_Mono     1.207e+00  5.106e-01   2.364 0.018090 *
## type_Psychic  4.742e-01  5.646e-01   0.840 0.400977
## type_Dragon  -6.199e-01  6.527e-01  -0.950 0.342260
## type_Flying   1.541e+00  6.127e-01   2.515 0.011906 *
## atk:def      -7.618e-04  2.511e-04  -3.034 0.002411 **
## spAtk:spDef   3.749e-04  2.708e-04   1.385 0.166169
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 450.90  on 799  degrees of freedom
## Residual deviance: 150.71  on 787  degrees of freedom
## AIC: 176.71
##
## Number of Fisher Scoring iterations: 9
```

Looking at the result, I am quite surprise that Special Attacks and Special Defense stats have virtually no significant contribution to the determination of Pokemon legendary status, as with the two Psychic and Dragon types. I proceed to remove the insignificant predictor variables, leaving us with the best model yet:

```
modelFinal <- glm(legendary ~ hp + atk*def + spe + type_Mono + type_Flying,
                  family = binomial(link = "logit"), data = dataDummy)
```

Now the other two dummy variables `type_Mono` and `type_Flying` become insignificant. Let's drop them and there we have our final model:

```
modelFinal <- glm(legendary ~ hp + atk*def + spe,
                  family = binomial(link = "logit"), data = dataDummy)
```

C. Logistic Regression - Post-hoc Analysis

1. Result Interpretation

Let's call the result of the final model.

```
summary(modelFinal)
```

```
##
## Call:
## glm(formula = legendary ~ hp + atk * def + spe, family = binomial(link = "logit"),
##     data = dataDummy)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.47929  -0.23991  -0.06429  -0.01264   3.10978
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.156e+01  2.464e+00  -8.751  < 2e-16 ***
## hp           5.034e-02  8.809e-03   5.714 1.10e-08 ***
## atk          6.322e-02  1.390e-02   4.549 5.38e-06 ***
## def          9.118e-02  1.474e-02   6.184 6.25e-10 ***
## spe          5.935e-02  8.420e-03   7.049 1.81e-12 ***
## atk:def     -5.084e-04  1.225e-04  -4.151 3.30e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 450.90  on 799  degrees of freedom
## Residual deviance: 237.92  on 794  degrees of freedom
## AIC: 249.92
##
## Number of Fisher Scoring iterations: 8
```

The beta coefficients in the case of logistic regression can be interpreted as odd ratios. It measures the variation of the genuine probabity of getting the binary response label through changes in the predictors. For example, the regression coefficient for `hp` is 0.05034. It means that `hp` is increased by one unit, the odds of a Pokemon being legendary will also increase by 1.0516286 times. Overall, as expected, the higher base stats a Pokemon has, the higher the chance that it is a legendary Pokemon. This chance is slightly reduced, though, when the Attack and the Defense stats are highly correlated (accounting for the significant interaction term).

2. Confusion Matrix and relevant metrics

```r
# Check accuracy for one model
predictProb <- predict(modelFinal, type = "response")

predictLabel <- ifelse(predictProb < 0.5, FALSE, TRUE)

table(Predict = predictLabel, Actual = dataDummy$legendary) %>% data.frame()
```

```
##   Predict Actual Freq
## 1   FALSE  False  722
## 2    TRUE  False   13
## 3   FALSE   True   38
## 4    TRUE   True   27
```

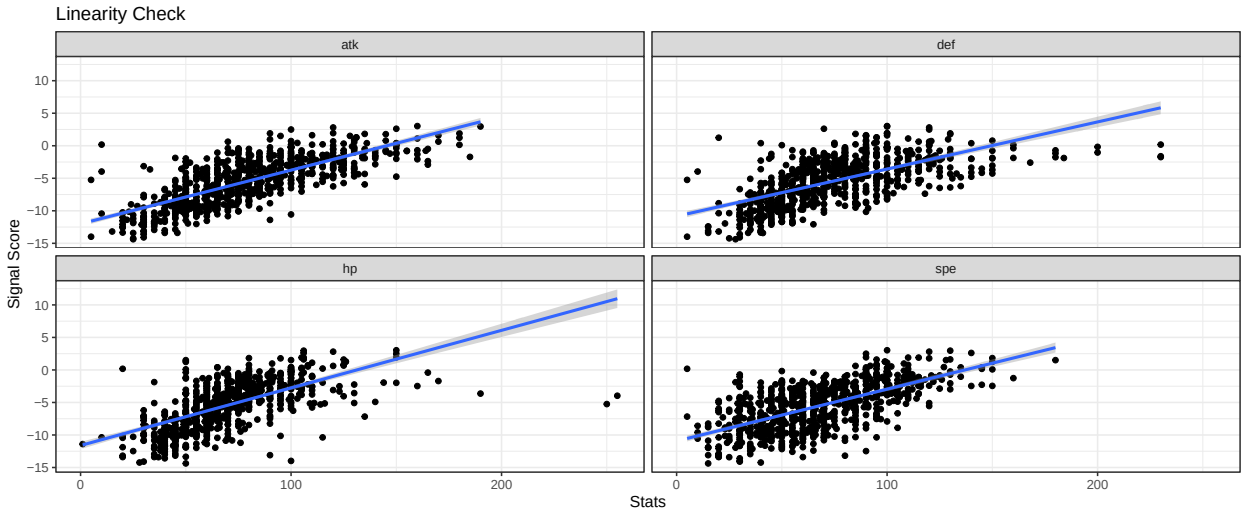
According to the confusion matrix, out of 800 cases, we have 12 False Positive and 40 False Negative cases.

- **Misclassification Rate** = 13.05% (Overall, how often is the model wrong?)
- **Test Sensitivity** = 0.42% (When it's actually true, how often does the model predict true?)
- **Test Specificity** = 0.98% (When it's actually true, how often does the model predict true?)

3. Linearity check

The relationship between the continuous predictor variables and the signal score (logit) of the response should be linear. I check this by plotting each individual predictor with the signal:

```r
dataDummy %>% select(hp, atk, def, spe) %>%
  mutate(signal = log(predictProb/(1 - predictProb))) %>%
  pivot_longer(-signal, names_to = "stat", values_to = "value") %>%
  ggplot(aes(x = value, y = signal)) + geom_point() + geom_smooth(method = "gam") +
  facet_wrap(.~stat, ncol = 2, nrow = 2) + labs(title = "Linearity Check") +
    xlab("Stats") + ylab("Signal Score")
```



Indeed, there is a general linearity among all explanatory variables when compared to the signal score of the response (apart from some outliers in the variable `hp` and `def`, but I guess still acceptable).

4. Residual Check

Based on this plot, it is possible that there are still some unknown trends in the residual that could explain the response variable. Nevertheless, most of the residuals land around the value -1 (which I guess is a good thing but don't know the explanation to).

```
residualCheck <- data.frame(predictLabel, Residual = modelFinal$residuals) %>%
  filter((Residual < 10) , (Residual > -10))

residualCheck %>% ggplot(aes(x = 1:nrow(residualCheck), color = predictLabel, y = Residual)) +
    geom_point() + coord_cartesian(ylim = c(-5,5)) + labs(title = "Residual Plot") +
    xlab("Index") + ylab("Residual") +
    scale_color_discrete(labels = c("Normal", "Legendary"), name = "Legendary Status")
```