Hayden Baillie Final

Hayden Baillie

2025-07-23

Project Description

Abstract

This project dives into whether a Pokemon's legendary status, type and generation can predict whether its total stats exceed 500, a competitive threshold. The dataset I used for this project was gathered from Kaggle and holds detailed information on Pokemon from Generation I - Generation VI. This information includes their base stats, types, and other attributes (Barradas, 2016). Cleaning the data included removing "Mega" forms are Pokemon, which are essentially more powerful forms of certain Pokemon that have the ability to "Mega Evolve, standardizing variable names and creating new variables. I then analyzed the clean dataset using visualizations and a logistic regression to look into the relationships between the predictors and the classification of a Pokemon as"high stat". From the regression analysis, I was able to conclude that Legendary Pokemon were overwhelmingly associated with the classification of "high stat", which was expected. Looking at types, fire and steel type Pokemon showed significantly higher odds of being classified as "high stat" than any other typing. Other types and what generation a Pokemon is from were not significant predictors based on my analysis.

Citation

Barradas, A. (2016). Pokémon with stats [Data set]. Kaggle. Retrieved July 23, 2025, from Kaggle website: https://www.kaggle.com/datasets/abcsds/pokemon

Github Information

- Usernmae: haydenbaillie
- Repository Name: HaydenBaillieRFinal
- Link: github.com/haydenbaillie/HaydenBaillieRFinal

Addressing Challenges Stated in the Midterm

I was successfully able to meet most the goals that I had set in the midterm. I successfully cleaned and analyzed the data, making use of 2 different visualizations and a logistic regression to do so. I also was able to expaand a little bit on the idea that I had come up with for the midterm by making my research question a little bit more rigorous, since I received feedback that I could improve upon that.

As for future steps, one of the challenges I was not able to follow through on was merging datasets to include later Pokemon generations. I could not find any datasets that only had later generation Pokemon with the same, or even similar information. I would have had to manually compile a dataset to create this which I didn't deem worth it, since I feel like I can get a good idea of the analysis without it.

R Syntax Beyond the Course

For the most part, I stuck with the libraries and syntax that we learned in the course, including the readr, dplyr, and ggplot2 libraries. These libraries were sufficient for helping me analyze, clean, and visualize the data for the most part. One library that I used that we did not specifically talk about during class to my knowledge was the broom library, which I used to help neaten up the output of my regression to make it a little bit easier to go through and interpret.

Data Import and Cleaning

```
library(readr)
library(dplyr)
library(ggplot2)

pokemon = read_csv("Pokemon.csv")

head(pokemon)
```

```
## # A tibble: 6 x 13
##     `#` Name    `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk` `Sp. Def`
##   <dbl> <chr>   <chr>    <chr>    <dbl> <dbl>  <dbl>   <dbl>     <dbl>     <dbl>
## 1     1 Bulbas~ Grass    Poison     318    45     49      49        65        65
## 2     2 Ivysaur Grass    Poison     405    60     62      63        80        80
## 3     3 Venusa~ Grass    Poison     525    80     82      83       100       100
## 4     3 Venusa~ Grass    Poison     625    80    100     123       122       120
## 5     4 Charma~ Fire     <NA>       309    39     52      43        60        50
## 6     5 Charme~ Fire     <NA>       405    58     64      58        80        65
## # i 3 more variables: Speed <dbl>, Generation <dbl>, Legendary <lgl>
```

```
#rename columns with spaces/dots
pokemon = pokemon %>%
  rename(
    Sp_Atk = 'Sp. Atk',
    Sp_Def = 'Sp. Def',
    Type1 = 'Type 1',
    Type2 = 'Type 2'
  )
```

```
#handle missing values
pokemon = pokemon %>%
  mutate(Type2 = ifelse(is.na(Type2), "None", Type2))

#create new variable called HighStat to determine whether or not a Pokemon's total stats are >500
pokemon = pokemon %>%
  mutate(
    HighStat = ifelse(Total >= 500, TRUE, FALSE)
  )
```

```
#remove mega pokemon since they aren't part of our analysis
pokemon = pokemon %>%
  filter(!grepl("Mega", Name))

head(pokemon)
```

```
## # A tibble: 6 x 14
##     `#` Name      Type1 Type2  Total    HP Attack Defense Sp_Atk Sp_Def Speed
##   <dbl> <chr>     <chr> <chr>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
## 1     1 Bulbasaur Grass Poison   318    45     49      49     65     65    45
## 2     2 Ivysaur   Grass Poison   405    60     62      63     80     80    60
## 3     3 Venusaur   Grass Poison   525    80     82      83    100     100    80
## 4     4 Charmander Fire  None     309    39     52      43     60      50    65
## 5     5 Charmeleon Fire  None     405    58     64      58     80      65    80
## 6     6 Charizard  Fire  Flying   534    78     84      78    109      85   100
## # i 3 more variables: Generation <dbl>, Legendary <lgl>, HighStat <lgl>
```

Summary Statistics

```r
summary(pokemon)
```

```
##        #              Name              Type1              Type2          
##  Min.   :  1.0   Length:751         Length:751         Length:751        
##  1st Qu.:189.5   Class :character   Class :character   Class :character  
##  Median :377.0   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :369.7                                                           
##  3rd Qu.:550.5                                                           
##  Max.   :721.0                                                           
##      Total             HP             Attack          Defense      
##  Min.   :180.0   Min.   :  1.00   Min.   :  5.00   Min.   :  5.00  
##  1st Qu.:325.0   1st Qu.: 50.00   1st Qu.: 55.00   1st Qu.: 50.00  
##  Median :430.0   Median : 65.00   Median : 75.00   Median : 68.00  
##  Mean   :423.3   Mean   : 68.67   Mean   : 75.98   Mean   : 71.74  
##  3rd Qu.:500.0   3rd Qu.: 80.00   3rd Qu.: 95.00   3rd Qu.: 90.00  
##  Max.   :770.0   Max.   :255.00   Max.   :180.00   Max.   :230.00  
##      Sp_Atk           Sp_Def           Speed          Generation   
##  Min.   : 10.00   Min.   : 20.00   Min.   :  5.00   Min.   :1.000  
##  1st Qu.: 45.00   1st Qu.: 50.00   1st Qu.: 45.00   1st Qu.:2.000  
##  Median : 65.00   Median : 65.00   Median : 65.00   Median :3.000  
##  Mean   : 70.21   Mean   : 70.11   Mean   : 66.61   Mean   :3.381  
##  3rd Qu.: 90.00   3rd Qu.: 85.00   3rd Qu.: 88.50   3rd Qu.:5.000  
##  Max.   :180.00   Max.   :230.00   Max.   :180.00   Max.   :6.000  
##  Legendary        HighStat      
##  Mode :logical   Mode :logical  
##  FALSE:692       FALSE:550      
##  TRUE :59        TRUE :201      
##                                 
##                                 
##                                 
```

```r
#summary of key stats by Legendary and HighStat
summary_table = pokemon %>%
  group_by(Legendary, HighStat) %>%
  summarise(
    mean_total = mean(Total),
    median_total = median(Total),
    count = n()
  )
```

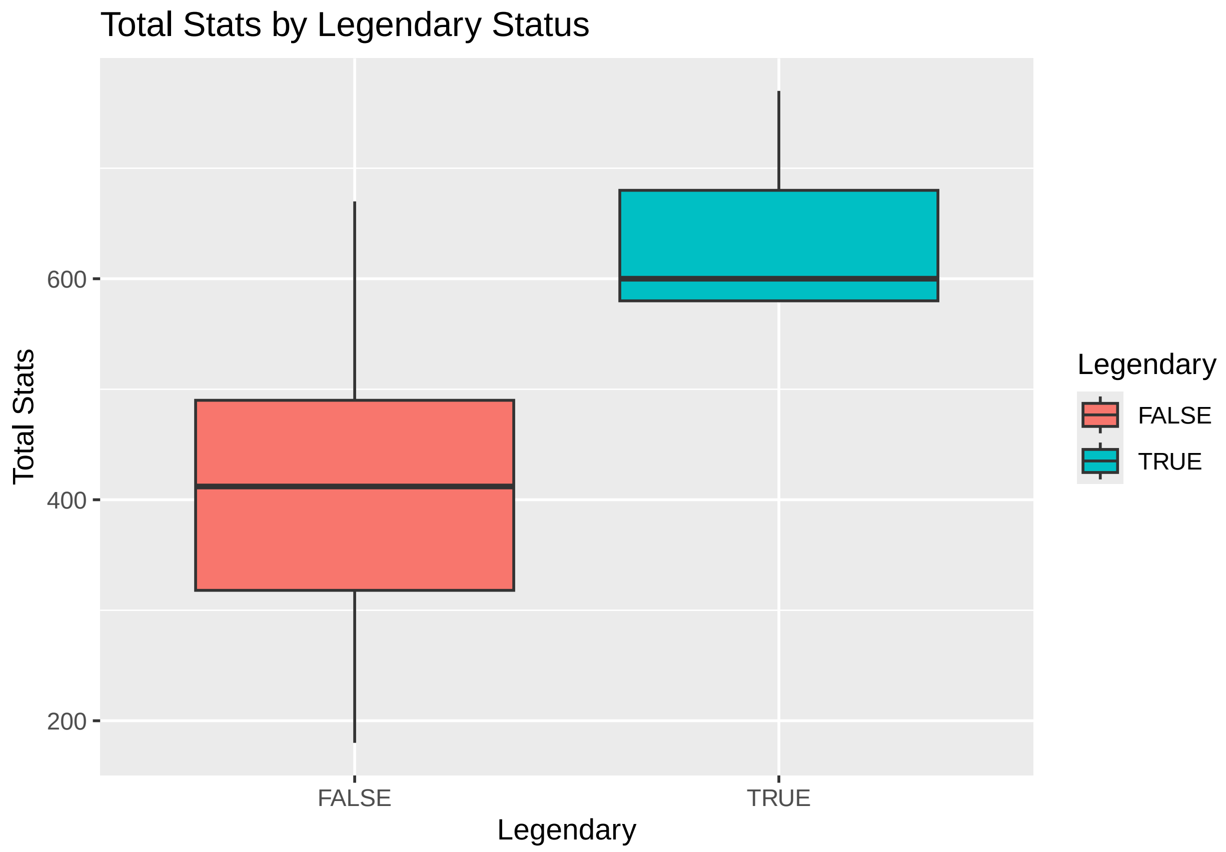
```
## `summarise()` has grouped output by 'Legendary'. You can override using the
## `.groups` argument.
```

```
summary_table
```

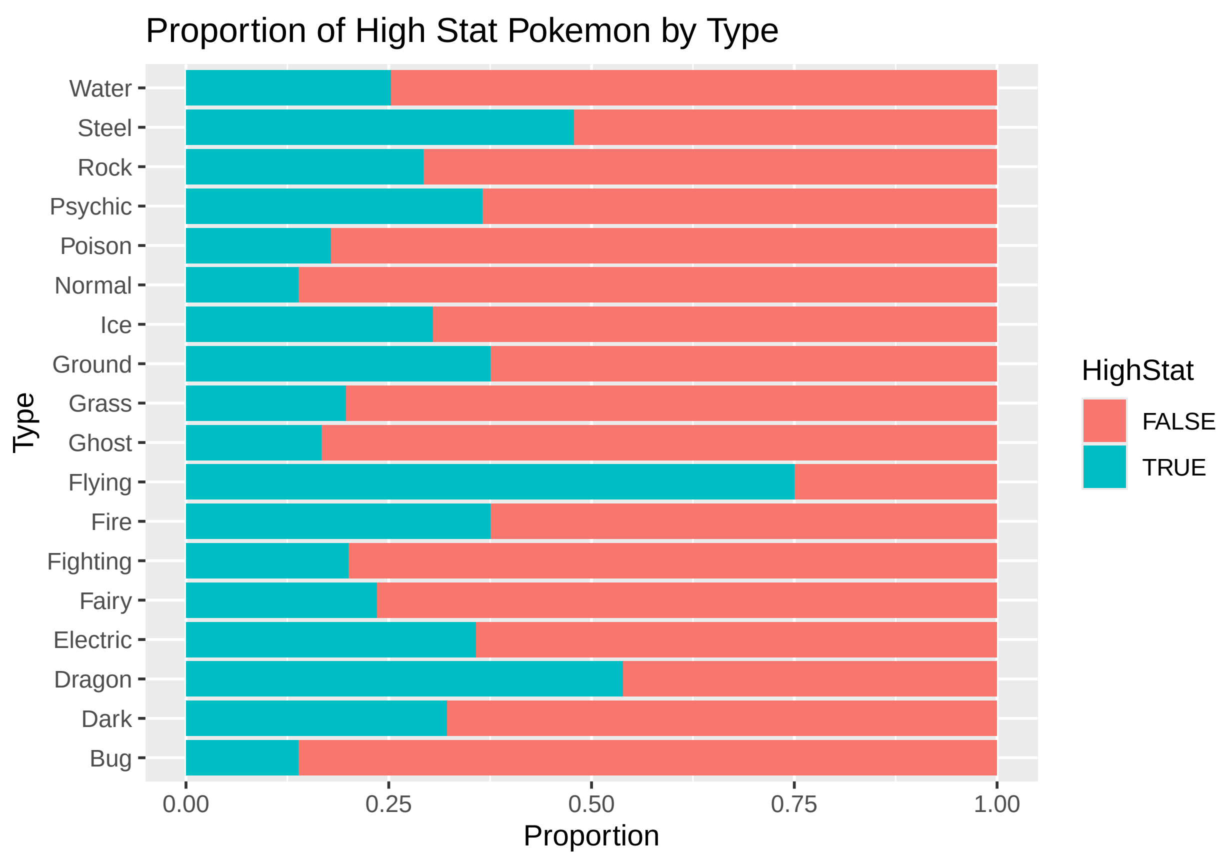
```
## # A tibble: 3 x 5
## # Groups:   Legendary [2]
##   Legendary HighStat mean_total median_total count
##   <lgl>     <lgl>         <dbl>        <dbl> <int>
## 1 FALSE     FALSE          374.          373   550
## 2 FALSE     TRUE           530.          525   142
## 3 TRUE      TRUE           627.          600    59
```

#Visualizations

```
ggplot(pokemon, aes(x = Legendary, y = Total, fill = Legendary)) +
  geom_boxplot() +
  labs(title = "Total Stats by Legendary Status", x = "Legendary", y = "Total Stats")
```



```
ggplot(pokemon, aes(x = Type1, fill = HighStat)) +
  geom_bar(position = "fill") +
  coord_flip() +
  labs(title = "Proportion of High Stat Pokemon by Type", x = "Type", y = "Proportion")
```



#Regression Analysis

```r
model = glm(HighStat ~ Legendary + Type1 + Generation, data = pokemon, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = HighStat ~ Legendary + Type1 + Generation, family = binomial,
##     data = pokemon)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -2.00507    0.41157  -4.872 1.11e-06 ***
## LegendaryTRUE  18.76187  504.92682   0.037   0.9704
## Type1Dark       0.78935    0.57152   1.381   0.1672
## Type1Dragon     0.92727    0.64311   1.442   0.1493
## Type1Electric   0.93155    0.50716   1.837   0.0662 .
## Type1Fairy      0.32493    0.73614   0.441   0.6589
## Type1Fighting   0.43958    0.61599   0.714   0.4755
## Type1Fire       0.99111    0.48944   2.025   0.0429 *
## Type1Flying     1.69002    1.46701   1.152   0.2493
## Type1Ghost     -0.34690    0.71167  -0.487   0.6259
## Type1Grass      0.15827    0.49805   0.318   0.7507
## Type1Ground     0.92788    0.55198   1.681   0.0928 .
## Type1Ice        0.64543    0.62631   1.031   0.3028
## Type1Normal    -0.15387    0.48235  -0.319   0.7497
## Type1Poison     0.34323    0.61234   0.561   0.5751
## Type1Psychic    0.26959    0.54999   0.490   0.6240
## Type1Rock       0.65440    0.52433   1.248   0.2120
## Type1Steel      1.24979    0.59777   2.091   0.0365 *
## Type1Water      0.60421    0.43103   1.402   0.1610
## Generation      0.05251    0.05877   0.893   0.3716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 872.51  on 750  degrees of freedom
## Residual deviance: 681.53  on 731  degrees of freedom
## AIC: 721.53
##
## Number of Fisher Scoring iterations: 16
```

```
library(broom)
tidy(model)
```

```
## # A tibble: 20 x 5
##    term          estimate std.error statistic    p.value
##    <chr>            <dbl>     <dbl>     <dbl>      <dbl>
##  1 (Intercept)    -2.01      0.412    -4.87   0.00000111
##  2 LegendaryTRUE  18.8     505.        0.0372 0.970
##  3 Type1Dark       0.789     0.572     1.38   0.167
##  4 Type1Dragon     0.927     0.643     1.44   0.149
##  5 Type1Electric   0.932     0.507     1.84   0.0662
##  6 Type1Fairy      0.325     0.736     0.441  0.659
##  7 Type1Fighting   0.440     0.616     0.714  0.475
##  8 Type1Fire       0.991     0.489     2.02   0.0429
##  9 Type1Flying     1.69      1.47      1.15   0.249
## 10 Type1Ghost     -0.347     0.712    -0.487  0.626
## 11 Type1Grass      0.158     0.498     0.318  0.751
## 12 Type1Ground     0.928     0.552     1.68   0.0928
## 13 Type1Ice        0.645     0.626     1.03   0.303
## 14 Type1Normal    -0.154     0.482    -0.319  0.750
## 15 Type1Poison     0.343     0.612     0.561  0.575
## 16 Type1Psychic    0.270     0.550     0.490  0.624
## 17 Type1Rock       0.654     0.524     1.25   0.212
## 18 Type1Steel      1.25      0.598     2.09   0.0365
## 19 Type1Water      0.604     0.431     1.40   0.161
## 20 Generation      0.0525    0.0588    0.893  0.372
```