Predicting a Pokémon's Strength Using Variables Other Than Base Stats

STA 210 BDN: Nate Krall, Daniel Cohen, Brian Kim

2023-11-15

Introduction and Data:

The expansive world of Pokémon, at its core, is a children's game. However, a deeper look at the numbers and statistics the game is built from reveals several intricate relationships among the different pokémons' characteristics. Each creature has its own specific set of base statistics, colloquially referred to as "base stats," including attack, special attack, defense, special defense, and speed, which indicate that pokémon's battle prowess. Summing these stats yields a pokémon's total base stats, which is the best measure of a pokémon's overall strength when all pokémon are put at an even playing field – common knowledge for any pokémon fan. We are interested in measuring a pokémon's strength without using base stats as predictors, giving us insight on how strong the relationships among pokémon's different characteristics actually are. Thus, we are looking to answer the following research question: **Can we predict a Pokémon's Base Stat Total from other variables?** In other words, we are analyzing how well variables such as the pokémon's type, capture rate, growth rate, generation, height, weight, base happiness, weaknesses, and if the pokemon is legendary or not can predict a pokémon's total base stats. We hypothesize that a multiple linear regression model including some formation of these predictor variables will be a somewhat strong predictor for base_total – thinking about the game, stronger pokémon would seem to have certain values for these predictor variables when compared to weaker ones: for example, legendary pokémon tend to be stronger in battle than non legendary pokémon, so we might expect is_legendary to be a useful predictor for base_total, for example.

We retrieved the dataset from kaggle.com, a large data science online community, and the dataset is called "The Complete Pokemon Dataset" created by Rounak Banik in 2017. The dataset was retrieved via web scraper from the website serebii.net, an all-in-one, reliable data hub for all things pokemon in 2017. Since it was formed in 2017, the dataset does not include pokémon from more recent games, but still includes a whopping 801 pokémon, meaning the dataset has 801 observations, one for each pokémon. However, note that we removed one pokemon from the original 801 pokemon, Minior, from the dataset, since it has 2 different

forms and has an uninterpretable capture rate. We noticed that while reading the .csv file, R automatically translated the *capture_rate* variable to characters because Minior's capture rate was listed as: "30 (Meteorite)255 (Core)". We decided to exclude this observation from the model because of its uninterpretable characteristics, and after, we casted *capture_rate* an integer instead of a character.

The dataset contains 23 variables, taken directly from the kaggle website for the dataset, explanations of which can be viewed in our data dictionary. For our analysis, we will be focusing on these specific variables:

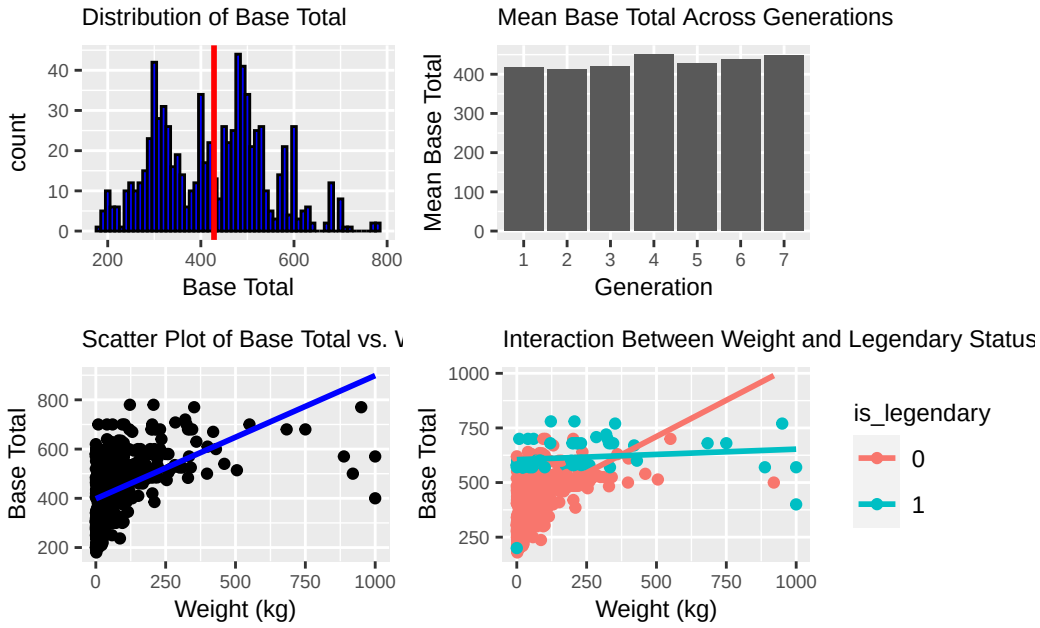
RESPONSE VARIABLE

- base_total

PREDICTOR VARIABLES

- experience_growth
- base_egg_steps
- base_happiness
- capture_rate
- experience_growth
- generation
- height_m
- percentage_male
- pokedex_number
- type1
- type2
- weight_kg
- is_legendary

As you'll notice, the dataset splits the base stats of each pokemon into the individual stats, but we only need to know about the base_total variable, which is included in the csv file. Each variable describes the pokémon at hand in a different way. Some, like is_legendary, may prove to be extremely important in our regression model, while with others, like name and Japanese name, we can remove them from consideration as they are simply unique identifiers.




```
# A tibble: 1 x 6
  minimum    q1 median  mean    q3 maximum
    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
1     180   320    435  428.   505     780
```

p1: Distribution of Base total: We can see from the distribution of the base total, that is seems to be roughly trimodal with three peaks (one around 300, 400, and 475). The base total points vary from about 180 to 780 with a mean at around 428.4. The median is 435 which is higher than the mean by a little bit which could mean that there are more extreme low base total Pokemon bringing the mean down a little bit. The IQR is 185 (505 - 320) and looking at the data there might be a potential outlier around 780 base total.

p2: Mean Base Total Across Generations: This is the graph showing the relationship between different generation Pokemon and their mean base total. We can see from this graph that the mean base total of generation 4 Pokemon were the highest but overall we can't see any distinct relationships between the generation of the Pokemon and their mean base total. One can maybe say there could be a very slight positive linear relationship between the generation of the Pokemon and their mean base total as we obsere that as generation increases the mean base total tends to increase slightly.

p3: Scatter Plot of Base Total vs Weight: From the scatter plot I can see that there is no apparent correlation between the weight and the base total. There may be a very low positive correlation between these two variables but most of the points are focused around

when the weight of the Pokemon is less than 100kg so it's hard to tell the relationship. There seems to be a few outliers near 900-1000kg.

p4: Interaction Between Weight and Legendary Status: This graph is showing the relationship between the weight of the Pokemon and the base total for legendary and non legendary Pokemon. The red line is for non legendary Pokemons (is_legendary = 0) and the blue line is for legendary Pokemon (is_legendary = 1). Something interesting we can note is that legendary Pokemon all tend to have a much higher and consistent base total for all weights. Across all weights most base total for legendary Pokemons stay between 550- 700. Among the legendary Pokemon there doesn't seem to be any correlation between the weight and the base total. Among the non-legendary Pokemon however, there seems to be a very slight positive correlation between the weight of the Pokemon and the base total. As the weight of the non-legendary Pokemon goes up, the Base total seems to go up slightly although with the cluster of points under 100 kg Pokemon, it is hard to conclude, and regardless, the correlation does not appear to be strong between these variables.

Note that from this graph we can tell that Legendary Pokemon appear to tend to have much higher base stat totals than non-legendary Pokemon on average, which is something that will definitely impact our multiple linear regression model.

Methodology:

> **!** Important
>
> Before you submit, make sure your code chunks are turned off with `echo: false` and there are no warnings or messages with `warning: false` and `message: false` in the YAML.

We first randomly split our data of 800 different pokemon into 75% training and 25% testing data so we can both train and evaluate our model.

Next, we take the training data through a recipe to ready it for our analysis. We took it through the following steps:

1. We update the role of "name" to be an ID. Name is simply a label for each observation.
2. We remove irrelevant, non-predicting information like abilities, classification, and Japanese name.
    - There are hundreds of different abilities a Pokemon can have, and very little overlap of abilities between Pokemon, so we do not need the abilities variable. The classification of a Pokemon is almost unique for every Pokemon (there is very little overlap), and it mainly just groups Pokemon by their evolution line, yielding classification unwanted.

3. We remove all against_* variables and all type variables.

    - these two variables convey the same information. Since there are 18 types, including typing as a predictor would convolute the other predictor variables, leaving our model extremely difficult to interpret, so we continue our analysis without considering typing.

4. We mean-center all quantitative predictors so our intercept is interpretable.

After bringing the training data through our recipe, we fit the data under a multiple linear regression model, which is outputted below:

term	estimate	std.error	statistic	p.value
(Intercept)	420.025	3.586	117.121	0.000
base_egg_steps	-0.001	0.001	-0.688	0.492
base_happiness	0.275	0.186	1.481	0.139
capture_rate	-0.800	0.044	-18.063	0.000
experience_growth	8.931	3.201	2.790	0.005
height_m	36.300	4.253	8.536	0.000
pokedex_number	0.301	0.085	3.540	0.000
weight_kg	0.023	0.042	0.564	0.573
generation	-31.754	10.048	-3.160	0.002
is_legendary1	78.688	25.909	3.037	0.002

As you can see from above base_egg_steps, base_happiness, and weight_kg are three candidates for variables to remove from the model, since we notice their p-values are all greater than 0.1, meaning they are potentially statistically insignificant predictors. Thus, the next step in our method is compare two models through a series of tests: the one model being our original model with all predictors after running our feature engineering, and one with base_egg_steps, base_happiness, and weight_kg removed to select a model that is either consice with enough comprehensiveness or as comprehensive as we can make it.

We run the training data through a very similar recipe for the second model except that base_egg_steps, base_happiness, and weight_kg are removed. The output of running multiple linear regression for this model is shown below:

term	estimate	std.error	statistic	p.value
(Intercept)	421.680	3.105	135.814	0.000
capture_rate	-0.806	0.044	-18.408	0.000
experience_growth	0.000	0.000	2.581	0.010
height_m	35.826	3.510	10.207	0.000
pokedex_number	0.284	0.084	3.396	0.001

term	estimate	std.error	statistic	p.value
generation	-30.017	9.896	-3.033	0.003
is_legendary1	57.550	12.883	4.467	0.000

We then conducted two tests to decipher which model is best to select and use for our final model: 1 comparing the AIC, BIC, and adjusted r squared for the models, and another comparing the results of V-fold cross validation for the two models. The output for the first test is shown below:

```
# A tibble: 1 x 3
    AIC   BIC adj.r.squared
  <dbl> <dbl>         <dbl>
1 6647. 6695.         0.643
```

```
# A tibble: 1 x 3
    AIC   BIC adj.r.squared
  <dbl> <dbl>         <dbl>
1 6644. 6679.         0.643
```

From this test, we note that both AIC and BIC are lower for the second, reduced model than the first model. The adjusted r squared values are very similar in size. With this information, it would be logical to select the second model, as it produces more preferable values of AIC and BIC while maintaining a very similar adjusted r squared value.

However, we run one more test, v-fold cross validation, to compare the models once again to have another point of evidence to base our decision off of. The results of this second test are below:

Cross validation results for the 1st, full model:

```
# A tibble: 2 x 6
  .metric .estimator   mean     n std_err .config
  <chr>   <chr>       <dbl> <int>   <dbl> <chr>
1 rmse    standard   71.8      15  3.62   Preprocessor1_Model1
2 rsq     standard    0.634    15  0.0347 Preprocessor1_Model1
```

Cross validation for the 2nd, reduced model:

```
# A tibble: 2 x 6
  .metric .estimator   mean     n std_err .config
  <chr>   <chr>       <dbl> <int>   <dbl> <chr>
1 rmse    standard   71.4      15  3.39   Preprocessor1_Model1
2 rsq     standard    0.636    15  0.0334 Preprocessor1_Model1
```

We notice that the RMSE value from the cross validation of the second model is lower than the RMSE value from the cross validation of the first full model, while the r squared values from both models have negligible difference, leading us to believe that the second model performs better in terms of predicting a pokemon's base stat total.

From the results of these two tests, **we can confidently select model number 2, the reduced model**.

The final check of our methodology is to examine the VIF values for our predictors to confirm there is no collinearity present in our model:

```
# A tibble: 6 x 2
  names                x
  <chr>            <dbl>
1 capture_rate      1.27
2 experience_growth 1.17
3 height_m          1.34
4 pokedex_number   41.2
5 generation       40.4
6 is_legendary1     1.42
```

A VIF value greater than 10 for a variable indicates concerning collinearity. We notice that pokedex_number and generation both have VIF values > 40, while all other variables have very low VIF values, meaning pokedex_number and generation appear to depend heavily on each other. This makes perfect sense – as generations of pokemon are simply intervals of pokedex numbers, or in other words, generation divides all values 1-800 of pokedex_number into different intervals (for example, generation 1 is pokedex numbers 1-151). Thus, we should remove 1 of the 2 variables before continuing with our final model. We choose to delete generation, since it is simply a far more discrete form of pokedex_number.

Our final model is the same as the second mode with generation removed.

```
# A tibble: 5 x 2
  names                x
  <chr>            <dbl>
1 capture_rate      1.24
2 experience_growth 1.16
```

```
3 height_m           1.33
4 pokedex_number     1.06
5 is_legendary1      1.37
```

As we can see, the VIF values are all now satisfactorily low for the final model, which we will display below:

term	estimate	std.error	statistic	p.value
(Intercept)	421.164	3.122	134.898	0.000
capture_rate	-0.824	0.044	-18.860	0.000
experience_growth	8.689	3.163	2.747	0.006
height_m	36.439	3.529	10.326	0.000
pokedex_number	0.034	0.013	2.495	0.013
is_legendary1	64.941	12.740	5.097	0.000

Results: