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 pokémon 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 pokémon 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 pokémon from the original 801 pokémon, 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:

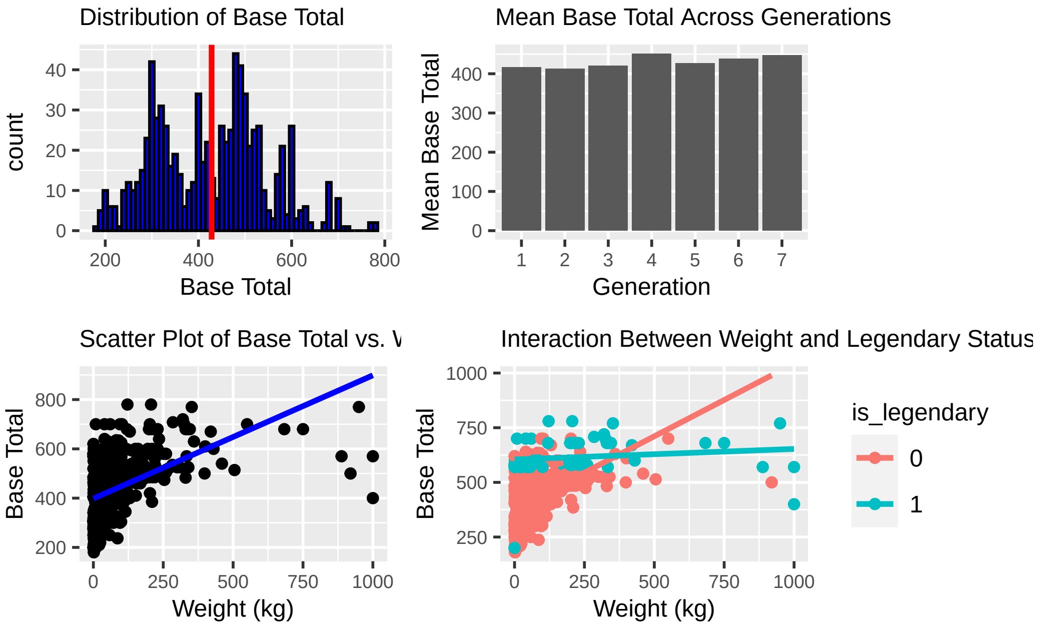
RESPONSE VARIABLE

- base_total: total base stats of the Pokemon [whole number]

PREDICTOR VARIABLES

- experience_growth: The Experience Growth of the Pokemon [whole number]
- base_egg_steps: The number of steps required to hatch an egg of the Pokemon [whole number]
- base_happiness: Base Happiness of the Pokemon [whole number]
- capture_rate: Capture Rate of the Pokemon [whole number]
- experience_growth: The Experience Growth of the Pokemon [whole number]
- generation: The numbered generation which the Pokemon was first introduced [whole number 1-7]
- height_m: Height of the Pokemon [number in meters]
- percentage_male: The percentage of the species that are male. [percentage value, blank if the Pokemon is genderless]
- pokedex_number: The entry number of the Pokemon in the National Pokedex [whole number between 1 and 801]
- type1: The Primary Type of the Pokemon (every pokémon has this)
- type2: The Secondary Type of the Pokemon (not all pokémon have this)
- weight_kg: The Weight of the Pokemon [number in kilograms]
- is_legendary: Denotes if the Pokemon is legendary.[0 = not legendary, 1 = legendary]

As you'll notice, the dataset splits the base stats of each pokémon 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.



minimum	q1	median	mean	q3	maximum
180	320	435	428.288	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:

We are conducting a multiple linear regression model to predict base_total from several other predictor variables. Any form of logistic regression would not make sense in this case as base_total is a quantitative variable, meaning we are not making classifications, and thus MLR is the model we conduct.

We first randomly split our data of 800 different pokémon 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.

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 Pokemon typing.

4. We mean-center all quantitative predictors so our intercept is interpretable.
5. We address missing data in the height/weight category:

    - There are 20 Pokemon with missing height and weight values. Per the author of the database, these 20 Pokemon have alternate regional forms where their height and weights differ from their normal form. Additionally, the other data about these Pokemon are based on the alternate form, creating a disparity between these 20 Pokemon and the rest of the Pokemon in the dataset. Thus, we decided to remove these 20 Pokemon from consideration.

6. Finally, we remove percentage_male from consideration.

    - Several Pokemon do not have a gender, leading to many missing values in the dataset. Upon further examination, we found that a disproportionate 63/70 of the legendary Pokemon in the dataset to not have a gender, due to the distinct and rare nature of the legendary status in the game, whereas most non-legendary pokémon do have a gender Thus, percentage_male and is_legendary cannot be effective predictors in conjunction, so we decide to remove percentage_male from our model.

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

term	estimate	std.error	statistic	p.value

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 0.05, 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 concise 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
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:

AIC	BIC	adj.r.squared
6646.726	6694.795	0.643

AIC	BIC	adj.r.squared
6644.285	6679.244	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:

.metric	.estimator	mean	n	std_err	.config
rmse	standard	71.803	15	3.619	Preprocessor1_Model1
rsq	standard	0.634	15	0.035	Preprocessor1_Model1

Cross validation for the 2nd, reduced model:

.metric	.estimator	mean	n	std_err	.config
rmse	standard	71.437	15	3.388	Preprocessor1_Model1
rsq	standard	0.636	15	0.033	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 pokémon'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:

names	x
capture_rate	1.265
experience_growth	1.169
height_m	1.336
pokedex_number	41.169
generation	40.427
is_legendary1	1.425

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 model with generation removed.

	x
capture_rate	1.241
experience_growth	1.174
height_m	1.263
pokedex_number	1.042
is_legendary1	1.330

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

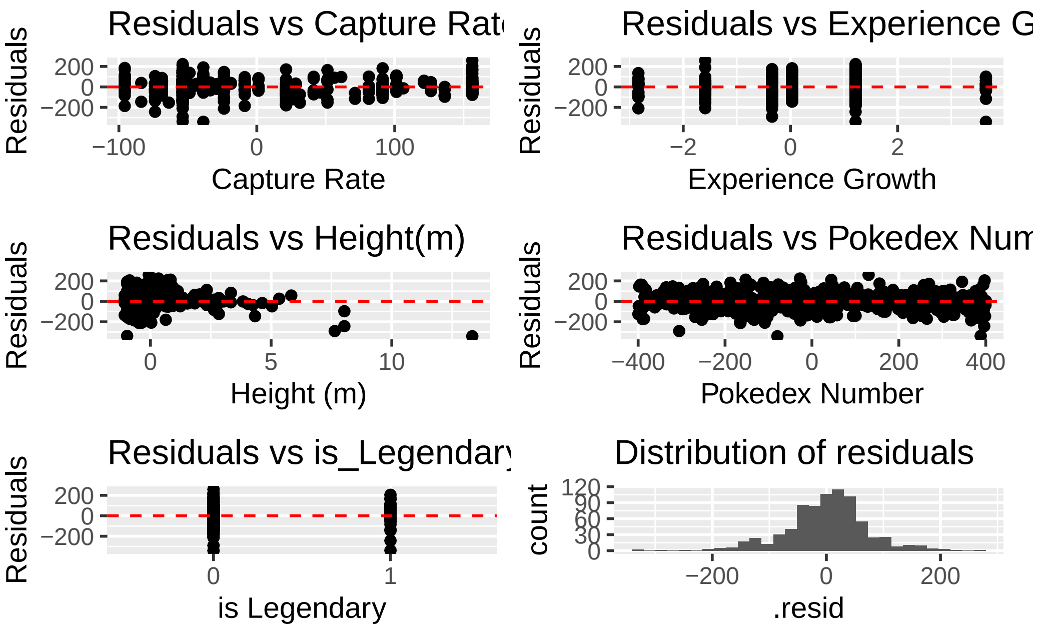
term	estimate	std.error	statistic	p.value
(Intercept)	419.982	2.774	151.379	0.000
capture_rate	-0.839	0.038	-21.811	0.000
experience_growth	2.412	2.810	0.858	0.391
height_m	28.620	2.724	10.507	0.000
pokedex_number	0.024	0.012	2.014	0.044
is_legendary1	88.596	10.629	8.336	0.000

The equation for the final model is

$$\begin{aligned} base\_total &= 419.982 - 0.839 * capture\_rate + 2.412 * experience\_growth \\ &+28.62 * height\_m + 0.024 * pokedex\_number + 88.596 * is\_legendary1 \end{aligned}$$

Results:

Model Conditions: Linearity / Constant Variance / Normality



Linearity condition - This condition is satisfied as there is no clear patterns in the residuals vs predictor variables such as a fanning pattern.

Constant Variance - The vertical spread of the residuals is constant across the plots of residuals versus fitted values, therefore this condition is satisfied.

Normality - The distribution of the residuals is approximately unimodal and symmetric, so the normality condition is satisfied. The sample size is sufficiently large » 30 so we can relax this condition.

Independence - The independence condition is **not** satisfied. We established earlier there is little variation in base total over generations, analogous to time in Pokemon, ridding the worry for a serial effect. However, due to the evolutionary nature of Pokemon, there is non-independent data in our data set. Most Pokemon evolve from a different pokemon/evolve into a new Pokemon as they gain experience, and these Pokemon among the same evolutionary line are not independent of each other. For example, Bulbasaur, Ivysaur, and Venusaur are all among the same evolutionary line and they often have the same or similar typing and generation while often maintaining correlated base stat totals and other information. Thus, our data does not satisfy the independence condition.

Limitations

Due to the non-independence of some observations in our dataset, our model will not be as good of a predictor compared to one over completely independent Pokemon as it assumes independent error terms when that is not the case. Instead of manually removing all but one pokémon in the each evolutionary line, we decide to keep the data as is with the non-independent limitation in mind as it better reflects the point of our model. If we decided to alter our data set to make it pass the independence condition, we would be removing more than half of our observations, and we would draw far less relevant conclusions in regards to our research question. We want to see how well different variables can predict the base stat total of *any given pokémon*, not just the pokémon at the top of their evolutionary chain. Thus, we accept that the lack of independence negatively affects the model, but we carry on with conclusions and results as is.

Results

Here are the RMSE and R^2 given from K-fold cross validation:

.metric	.estimator	mean	n	std_err	.config
rmse	standard	71.950	15	3.206	Preprocessor1_Model1
rsq	standard	0.634	15	0.033	Preprocessor1_Model1

Here are the RMSE and R^2 given from the training set:

.metric	.estimator	.estimate
rsq	standard	0.641

.metric	.estimator	.estimate
rmse	standard	71.085

Here are the RMSE and R^2 given from the test set:

.metric	.estimator	.estimate
rsq	standard	0.577

.metric	.estimator	.estimate
rmse	standard	80.689

When we compare the R^2 values, that of the training model was .641 while the testing model had a R^2 value of .577. This is to be expected since the training data has been trained on so it should be a better predictor than the testing model. When we compare the RMSE for the training and testing data, we can see that the RMSE for the training set was 71.1 while the RMSE of the testing set was 80.69. Since lower RMSE values indicate a better fit to the model and we can see that the RMSE of the testing set was greater than that of the training set which means the training data once again is a better predictor and this is once again to be expected since training data was trained on so it should predict the model better. Since the difference in the R^2 and the RMSE values between the training and the testing data wasn't too significant, this shows that our model doesn't over fit the data.

Overall, for our initial interpretations of our final model, we find it to be a relatively strong predictor for base_total of a Pokemon given the relatively low RMSE value and the relatively high R^2 value. Thus, we have confirmed our hypothesis that in fact, we *can* predict the base_total of a Pokemon with decent accuracy by using different variables.

Discussion + Conclusion:

Summary of Findings

Our research aimed to predict a Pokémon's base stat total. Our analysis revealed statistically significant and statistically insignificant relationships between certain characteristics such as Pokémon's capture rate, whether it is legendary, and generation with the pokemon's overall strength (base total). Our final model which included capture rate, experience growth, height, pokedex number, and legendary status, was not only a good predictor of base_total, accounting for approximately 64.074% of the variability in Base Stat Total in the training data, but also was a concise model, only including statistically significant predictors. The relatively low RMSE value of 72.81 also suggested a satisfactory level of prediction accuracy, as base total ranges from 200 to 800.

Limitations and Improvement Suggestions

The dataset was drawn in 2017 does not include pokémon in games released after this date. This means that this model should not be extrapolated past this date as the Pokemon games evolve over time and might affect our applicability of these results to future editions. Additionally, as we talked more in detail about in the previous section, the independence condition being violated limits the predictive power of this model, as evolution chains are not including in our model. While we removed variables like base_egg_steps and base_happiness due to their high p-values, further exploration could determine if any interaction effects or non-linear relationships exist that we might have overlooked.

Reliability and Validity Concerns

The linear regression model assumes linearity, independence, homoscedasticity, and normality of residuals, which we proved held for our dataset. Even though our model satisfied these conditions, any violation in future datasets of future Pokemon games could make the model unreliable. Additionally, the data was scraped from a fan-run website, there might be biases or errors in how the information was recorded, affecting the validity.

Future Work

To continue our research in the future, we could could include data from newer Pokémon games to keep our model relevant over time. Additionally, analyzing how a Pokémon's evolutionary stage affects its Base Stat Total could be very interesting, especially due to the great variance in Pokemon evolution. For examples, the starter pokémon are generally pretty strong compared to other pokémon their level at all stages in their evolution. On the other hand, Magicarp is possibly the weakest pokémon in the game, but evolves to become Gyrados, one of the strongest pokémon in the game. Additionally, we could have more investigation into the interactive effects between variables, like how the combination of type and legendary status impacts base stats, could provide a deeper understanding of the underlying dynamics.