IST 652 Project Pokémon

A Pokémon analysis on traditional games, Pokémon Go, and twitters

Team Members: Bing-Je Wu, and Maggie Southwick

Table of Contents

[**Introduction** 2](#_Toc27076713)

[**Dataset** 2](#_Toc27076714)

[**Question 1** 2](#_Toc27076715)

[Data Source and Cleaning 2](#_Toc27076716)

[Methods of Analysis 3](#_Toc27076717)

[Overall Description of Program 4](#_Toc27076718)

[Conclusions 4](#_Toc27076719)

[**Question 2** 6](#_Toc27076720)

[Data Source and Cleaning 6](#_Toc27076721)

[Methods of Analysis 8](#_Toc27076722)

[Overall Description of Program 8](#_Toc27076723)

[Conclusions 9](#_Toc27076724)

[**Question 3** 10](#_Toc27076725)

[Data Source and Cleaning 10](#_Toc27076726)

[Methods of Analysis 12](#_Toc27076727)

[Overall Description of Program 12](#_Toc27076728)

[Conclusions 13](#_Toc27076729)

[**Question 4** 15](#_Toc27076730)

[Data Source and Cleaning 15](#_Toc27076731)

[Methods of Analysis 16](#_Toc27076732)

[Overall Description of Program 17](#_Toc27076733)

[Conclusions 18](#_Toc27076734)

[**Overall Project Conclusion** 20](#_Toc27076735)

# **Introduction**

This project will analyze the strength, versatility and popularity of Pokémon. This analysis will consider the original Pokémon game, as well as the newer popular platform of Pokémon Go. It will draw upon data from the Pokémon database to analyze strength and versatility, and will consider data from social media platforms to assess popularity. Four research questions are as following:

1. How a legendary Pokémon was enhanced by the Mega Evolution?
2. Is there any difference in a Pokémon’s power or skill between the two platforms? How are they different?
3. Which Pokémon types have the most powerful moves? When combined with powers associated with a Pokémon’s identity, which Pokémon types are the strongest?
4. Which Pokémon types are the most popular, as measured by mentions on social media platforms?

Each of questions comes along with the sections as Data Source and Cleaning, Methods of Analysis, Overall Description of Program, and Conclusions.

# **Dataset**

Three CSV files and one JSON file are from different data source. The first dataset, main dataset, is from Kaggle, <https://www.kaggle.com/abcsds/Pokémon> . The second dataset is Pokémon GO dataset from Pokémon database, <https://pokemondb.net/go/pokedex> . The third dataset is Pokémon moves dataset from the Pokémon database, <https://pokemondb.net/move> . The fourth dataset is Tweets that contain Pokémon names, collected from Twitter. We are going to utilize those four datasets to perform our research questions and analysis.

# **Question 1**

How was a legendary Pokémon enhanced by the Mega Evolution?

## Data Source and Cleaning

The Pokémon dataset was used from Kaggle, <https://www.kaggle.com/abcsds/Pokémon> , as the main dataset to answer question 1. The Pokémon dataset is the Pokémon states collected from the console games and stored as comma-separated values (CSV) file. It has 13 fields, described in the table below.

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Description |
| # | Integer | Pokémon index, also known as Pokédex |
| Name | String | Name of each Pokémon |
| Type1 | String | Each Pokémon has a type, this determines weakness/resistance to attacks |
| Type2 | String | Some Pokémon are dual type and have second type |
| Total | Integer | Sum of all stats that come after this, a general guide to how strong a Pokémon is |
| HP | Integer/Special character | Hit points, or health, defines how much damage a Pokémon can withstand before fainting |
| Attack | Integer | The base modifier for normal attacks (e.g. Scratch, Punch) |
| Defense | String | The base damage resistance against normal attacks |
| Sp\_Atk | String | Special attack, the base modifier for special attacks (e.g. fire blast, bubble beam) |
| Sp\_Def | Integer | The base damage resistance against special attacks |
| Speed | Integer | Determines which Pokémon attacks first each round |
| Generation | Integer | Determines which Pokémon from which generation |
| Legendary | Boolean | Determines which Pokémon has the legendary attribute |

The main program, *Pokémon Project\_BJW\_MS.ipynb*, loads the Pokémon dataset with the Pandas package and stores the data as a pandas dataframe. The cleaning process primarily involved dealing with column names, missing data, data type conversion, and adding new features.

Some of the columns have been renamed in a standard format. It is better to use underline instead of blanks, periods or special signs. Four columns are renamed with the rename method from the Pandas library. Parameters were set up as follows: ‘Type 1’ as ‘Type1’, ‘Type 2’ as ‘Type2’, 'Sp. Atk' as 'Sp\_Atk' and 'Sp. Def' as 'Sp\_Def'. Because Pokémon can be dual type, some Pokémon have value on the ‘Type2’ column. For example, Bulbasaur has a ‘Grass’ type and ‘Poison’ Type. And some Pokémon have only one type. For instance, Pikachu is an ‘Electric’ type. 386 Missing values were found on the Type2 column. It means there are 386 Pokémon that have only one type. To keep the records in the dataset, the missing values were replaced with a new category, called “no value” with fill\_na function. Two data types were converted with astype method from the Pandas library. ‘Generation’ and ‘Legendary’ were converted to string type.

New features were added to answer the first research question. A ‘Mega’ field was added by using the select method from Numpy library to choose the ‘True’ or ‘False’ based on the condition. When a Pokémon name contains ‘Mega’, the select method fills in the ‘True’ value. Otherwise, it selects the ‘False’ value. ‘Total\_Atk’ was calculated by adding ‘Attack’ and ‘Sp\_Atk’ together to be one of the metrics. And, ‘Total\_Def’ was calculated by adding ‘Defense’ and ‘Sp\_Def’ together to be one of the metrics as well.

The final preprocessing task was to summarize each of the fields to understand their data types, as well as the distribution of numeric values. The pandas info and describe functions were used to perform this summary, the results of which are shown as figure 1 below.

## Methods of Analysis

The first research question is to know how the Mega Evolution effect on the legendary Pokémon. This calls for an analysis of the total power, total attack power and total defense power within the main dataset.

The program will output a series of printed lists, identifying the Pokémon that have how much difference between before mega evolution and after mega evolution in comparison of three metrics, total power, total attack, and total defense. The program will also produce some plots as an aid to the analysis. The question also investigates which Pokémon was enhanced most under mega evolution. Our target Pokémon will have a huge difference between before mega evolution and after mega evolution in comparison to three metrics.



## Overall Description of Program

The program for analysis progresses through the following steps. First, it subsets the main dataset to find the Pokémon who has Mega form and is Legendary Pokémon. The output of the subset shows a list of Pokémon that meet the requirement of the first research question.

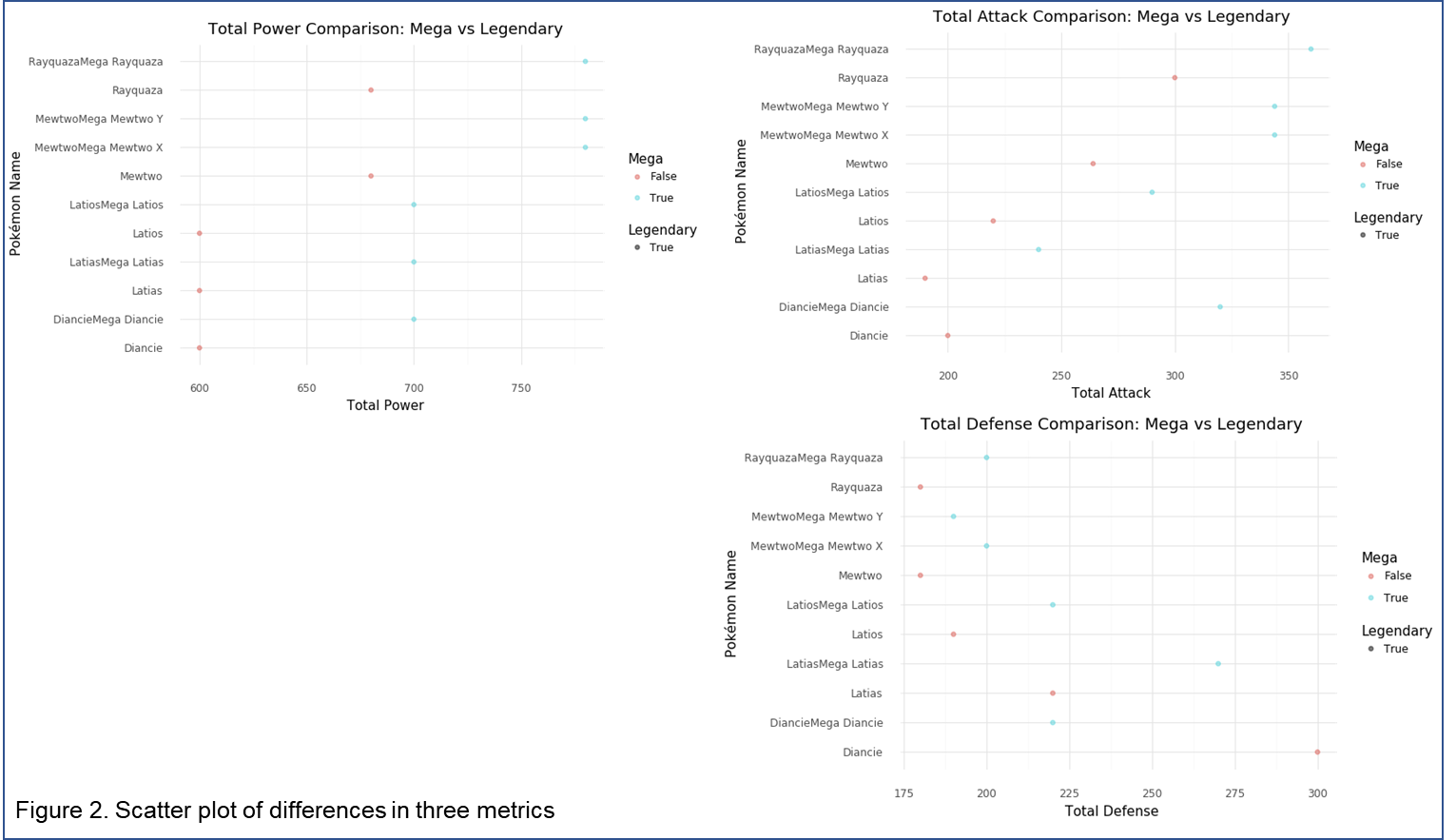
The program next redoes the subset process and gets a copy of the main dataset with the list acquired in the early steps. The program, then, uses the Plotnine package (known as ggplot2 in python) to produce the scatter plot on total power metric, total attack metric, and total defense metric.

In order to get the details analysis, the program excludes the special Pokémon who has two mega forms and iterates with for loop to calculate the level of increase on three metrics. Then, the program iterates with for loop and calculates the level of increase on three metrics for the special Pokémon who has two mega forms.

## Conclusions

Figure 2 is the visualization of the differences between before mega evolution and after mega evolution in comparison to three metrics. Those plots give us some overall understandings of the level of increases in Pokémon. It seems to have similar increases in total power. Some Pokémon get enhanced quite significant in total attack and some get the advantage of the total defense. Figure 3 provides the detail that how Pokémon’s power is enhanced in three metrics. Clearly, there is no difference in the increasing level for all of the Pokémon in total power metric. Each Pokémon get boosted the same amount of 100 total powers. And it is reasonable considering that the company wants to balance the strength of the Pokémon. For the total attack comparison, there is one Pokémon, No. 719 Diancie, which has been increased by 120 points in total attack. The Pokémon, No. 380 Latias, has the lowest amount of increase in total attack. Moving on to the total defense comparison, a surprising result shows that the Pokémon, No. 719 Diancie, which has the most amount of increase in total attack, got deducted by 80 points in total defense. It provides a piece of evidence that the company wants to make the game balanced. In the comparison of the Total Defense, we can notice that the Pokémon, No. 150 Mewtwo, has two forms after mega evolution. Each form got boosted a different amount of total defense. The Mega Mewtwo X has been increased by 20 points in total defense, and the Mega Mewtwo Y has been increased by only 10 points on total defense.

To conclude the first research question, we can see how the company treats those legendary Pokémon. They want to bring in a new element, mega evolution, into the game. But, in the meantime, they also need to keep the balance of the strength of Pokémon in the Pokémon Universe. By examining the amount of total power, we can say the company has done a good job of keeping the balance of the game





# **Question 2**

Is there any difference in a Pokémon’s power or skill between the two platforms? How are they different?

## Data Source and Cleaning

The Pokémon Go dataset was scraped from the Pokémon Database website, <https://pokemondb.net/go/pokedex> , as well as the original Pokémon dataset from Kaggle used in question 1. The Pokémon Go dataset was stored as an embedded HTML table on the PokémonDB website. It has 12 fields, described in the table below. In general, the table details the various information on name, type, combat stats, catch rate, flee rate, candy, and moves.

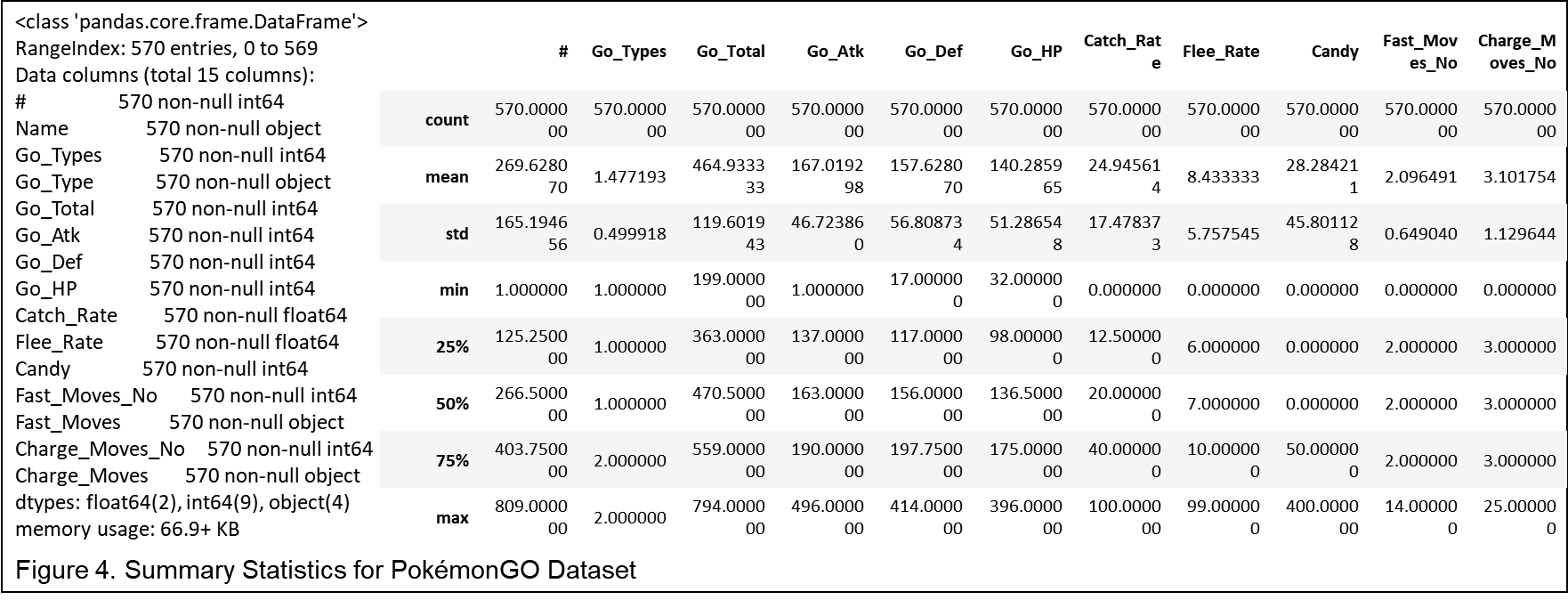
|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Data Type | Originally Stored As | Description |
| Unnamed: 0 | integer | N/A | The index number of the table |
| # | Integer | Text and Image | Pokémon index, also known as Pokédex |
| Name | String | Text | Name of each Pokémon |
| Type | String | Image Alt text | Each Pokémon has a type or two types, this determines weakness/resistance to attacks |
| Attack | String | Text | The power of the attack |
| Defense | Integer | Text | The power of the defense |
| HP | Integer/Special character | Text | Hit points, or health, defines how much damage a Pokémon can withstand before fainting |
| Catch\_Rate | String | Text | The chance of a Pokémon being caught; this can be increased by timing your PokéBall throws correctly |
| Flee\_Rate | String | Text | The chance of the Pokémon will flee after a missed capture |
| Candy | Integer | Text | The number of candies required for the Pokémon to evolve. |
| Fast Moves | String | Text | The base attack, such as Bite, Lick, etc |
| Charge Moves | String | Text | The special attack, such as Thunderbolt, Dragon Pulse, etc. |

An extra program, *Pokémon GO Pokédex Web scraping.ipynb*, was conducted to scrape the table from the web. The program uses the requests package to load the contents of the webpage on which the table is stored. It then uses the lxml HTML package to parse the contents of the web page. Since each record was stored in its own row, the program parsed each element that began with the row tag, “//tr”, and stored it as an element in Python. After confirming that we correctly read in all 11 columns for each of the 570 rows of data, the program iterated through the elements, selects the first item in each column (the headers) and stores it as a tuple, coupled with an empty list. The program appends data to each of these 11 empty lists in the following manner. It iterates through each element, or row, extracts the text content, and appends it to the empty list of the corresponding column. Checks are in place to ensure that no empty rows are appended, and that any numeric data is converted to an integer. After this iteration process is complete, the program prints the length of each column to ensure that each has a uniform length. The resulting list of tuples is then converted to a dictionary, in which the column name was the key, and the list of values in that column was the value. Finally, the dictionary was converted to a pandas dataframe. Three functions, a function of splitting a string into two for ‘Type’ column, a function of creating a bracket with Pokémon which has a special name for ‘Name’ column, and a function of breaking a string on camel case (a lowercase followed by an uppercase) for ‘Fast\_Moves’ and ‘Charge\_Moves’ column, were created for performing columns transformation with the apply method from the pandas package. Lastly, the program output a comma-separated values (CSV) file named as “pokemonGo\_list.csv”.

Now, we are back to the main program, *Pokemon\_Project\_BJW\_MS.ipynb*, the program loads in the CSV file that was exported in earlier and stores the data as a Pandas data frame. The cleaning process primarily involved dealing with column names, missing data, data type conversion, and adding new features.

Some of the columns have been renamed to differentiate from the main dataset such as ‘Go\_Type’, ‘Go\_Atk’, ‘Go\_Def’, and ‘Go\_HP’. Two missing values are on the ‘Fast\_Moves’ column and two are on the ‘Charge\_Moves’ column. In Pokémon Go, there are two Pokémon that do not have moves. To keep the records in the dataset, the missing values were replaced with a new category, called “no value” with fill\_na function. The ‘Catch\_Rate’ column has been transformed to remove the ‘%’ sign and '—' sign to be converted to float type with the replace method and the astype method. The ‘Flee\_Rate’ column has been transformed to remove the ‘%’ sign and converted to float type as well. A new feature was added to answer the second research question. The ‘Go\_Total’ has been calculated by adding the ‘Attack’, ‘Defense’, and ‘HP’ together for each of Pokémon.

The final preprocessing task was to summarize each of the fields to understand their data types, as well as the distribution of numeric values. The pandas info and describe functions were used to perform this summary, the results of which are shown as figure 4 below.



After finishing the pre-processing of the Pokémon Go dataset, the program then merges the main dataset with the Pokémon Go dataset by using an inner join to keep the Pokémon exist in both platforms.

## Methods of Analysis

The second research question seeks to understand what is the difference from Pokémon Go platform and console game platform. This calls for an analysis of the total power, total attack power, total defense power, generation, and platform. The program will output a series of visualization in comparison to three metrics, total power, total attack and total defense on both platforms. And further, the stats of the top Pokémon existing in each metric will be pulled out to analyze the performance on each platform.

## Overall Description of Program

The program for analysis progresses through the following steps. It makes a copy of the merged dataset for the Pokémon Go platform and subset the ‘#’, ‘Name’, ‘Go\_Types’, ‘Go\_Total’, ‘Go\_Atk’, ‘Go\_Def’ and ‘Generation’ columns. Another copy of the merged dataset was made for the console game platform subsetting with the ‘#’, ‘Name’, ‘Types’, ‘Total’, ‘Total\_Atk’, ‘Total\_Def’ and ‘Generation’ columns. Next, it renames the columns to make the two dataframes to have the consistent column names. The ‘Platform’ variable is created for both of the subsetted dataframes. Then, it combines the two subsetted dataframes together by rows with the concat method from the Pandas package.

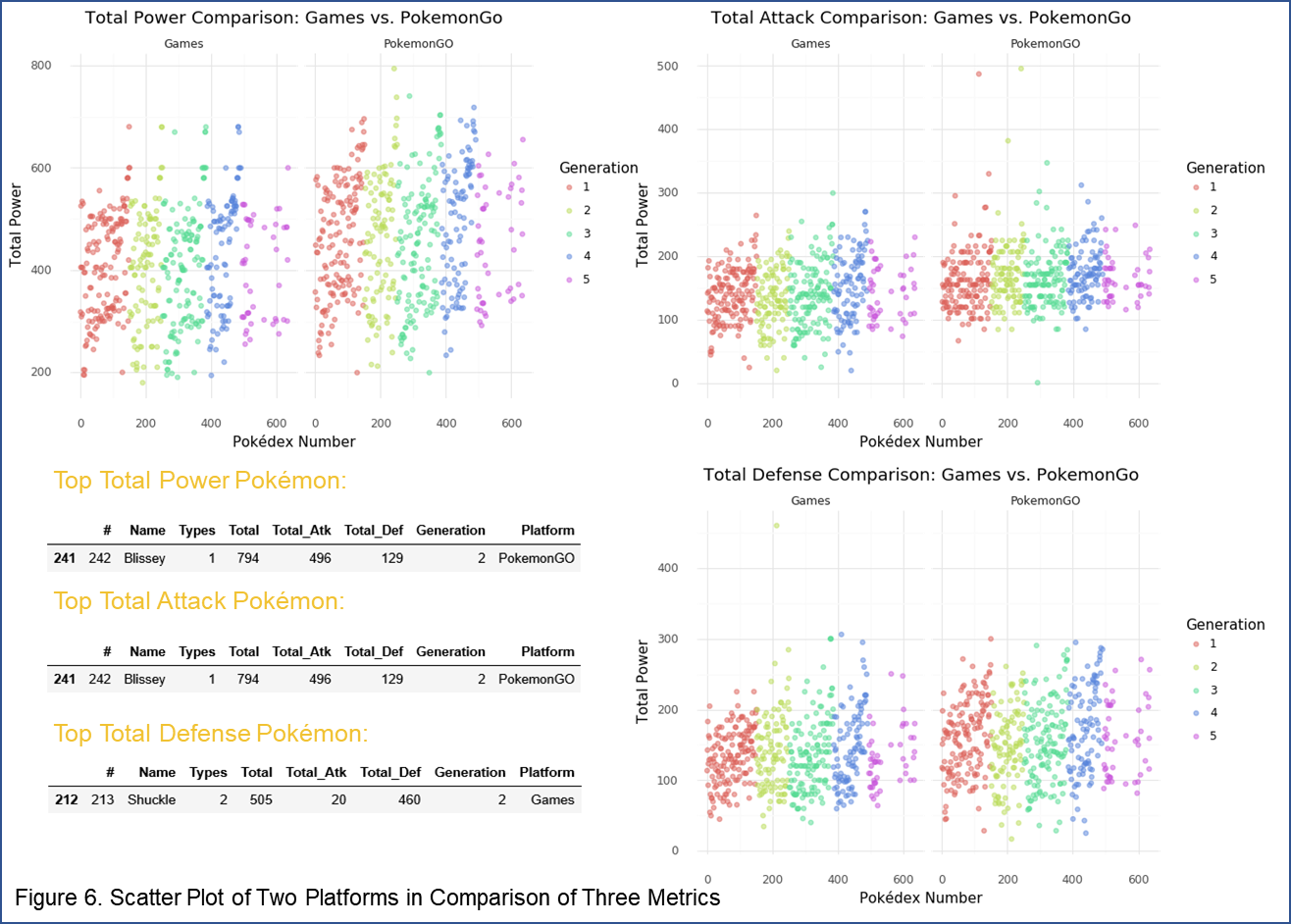
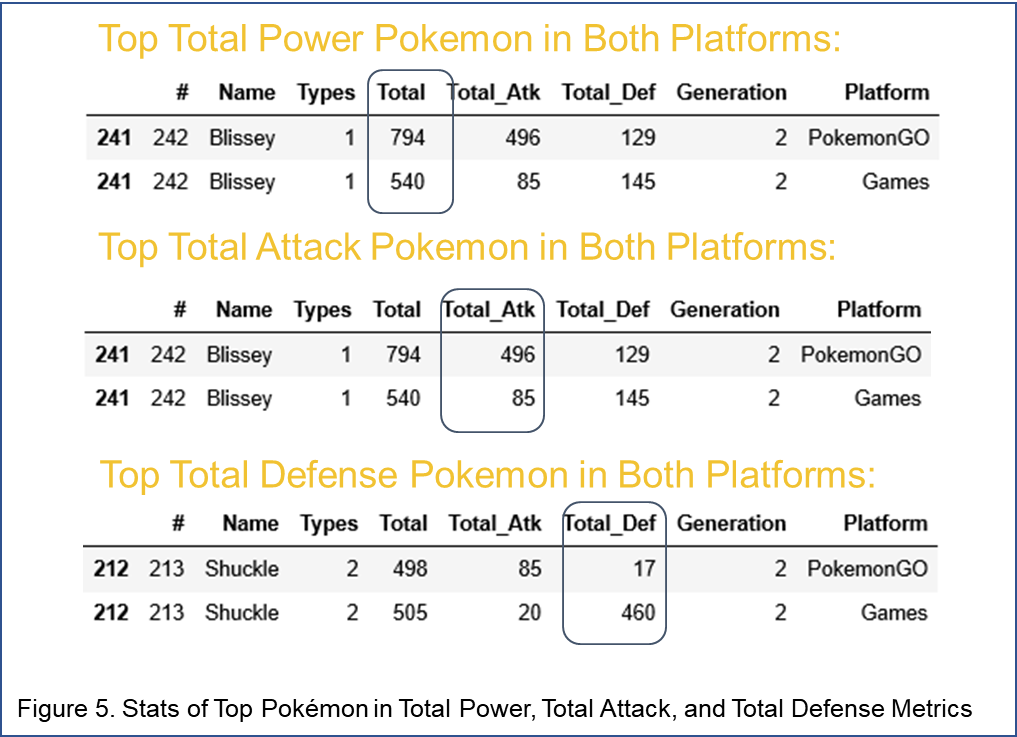
The program uses the Plotnine package to plot three scatter plots of Pokémon’s comparison in three metrics on two platforms. The ‘Platform’ variable is shown in the plot as the third dimension and the ‘Generation’ variable is shown as the fourth dimension. Each of the top outliers is identified and has been further investigated by comparing the performance on each platform.

## Conclusions

Figure 6 visualizes the distribution of Pokémon’s in total power metric, total attack metric, and total defense metric based on their generation and platform. The plot on the upper left of figure 6 has an outlier on the top. That outlier has been identified as the Pokémon, No. 242 Blissey, on the Pokémon Go platform. Based on the plot, the Pokémon Go platform tends to have a higher total power than the console game platform. For the total attack metric, there are two obvious outliers on the Pokémon Go platform. The top one is still the Pokémon, No. 242 Blissey. This Pokémon is taking over the total power and total attack. And the Pokémon Go platform seems to have higher total attack value than the console game platform. The plot at the lower right of figure 6 shows that Pokémon Go platform and console game platform have similar total defense value. There is an extreme outlier on the console game platform. The Pokémon is identified as No.242 Shuckle.

A detailed analysis can be performed based on Figure 5. Those top outliers were pulled out to compare the performance on both of the platform. The Pokémon, No. 242 Blissey, is performing better on Pokémon Go platform than console game platform for both total power and total attack metrics. The Pokémon, No.242 Shuckle, has the highest total defense value on the other hand. Its total defense value is overwhelming higher on console game platform than Pokémon Go platform. All the top outliers are in generation 2.

In conclusion of the second research question, we can see that the stats of a Pokémon can be vary depending on platform. Each platform has its own features and element to enhance the gaming experience. For example, ‘Speed’ attribute is designed for console games to determine the attack priority and ‘Candy’ attribute is specially designed for Pokémon Go to evolve a Pokémon. Even not there 8 generations existing in the Pokémon universe, the old generation is still having its own advantage, even in the different platforms.



# **Question 3**

Which Pokémon types have the most powerful moves? When combined with powers associated with a Pokémon’s identity, which Pokémon types are the strongest?

## Data Source and Cleaning

This question was based on data pulled from the Pokémon Database website, <https://pokemondb.net/move/all> , as well as the original Pokémon dataset from Kaggle. The moves dataset is stored as an embedded html table on the PokémonDB website. It has 8 fields, described in the table below. In general, the table details the various actions that particular Pokémon may take when battling other Pokémon.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Data Type | Originally Stored As | Description |
| Name | String | Text | Title of a Pokémon move |
| Type | String | Text (highly formatted) | Identifies which of the 18 types of Pokémon that are able to use this move (i.e. Electric, Flying) |
| Category | String | Image Alt text | Identifies whether a move has a special, physical or status effect on the opponent |
| Power | Integer | Text | Impact of the move, on a scale of 0-250 |
| Accuracy | Integer/Special character | Text | Precision with which a move can be used, on a scale of 0 to infinity |
| PP (Power Points) | Integer | Text | The number of times a move can be used |
| TM (Technical Machine) | String | Text | Method by which a Pokémon is able to learn this move |
| Effect | String | Text | Describes what the move does in the context of a battle |
| Probability | Integer | Text | The likelihood that a Pokémon will be able to use this move |

The *Pokémon\_Moves\_Data\_Scrape.ipynb* program uses the requests package to load the contents of the webpage on which the table is stored. It then uses the lxml HTML package to parse the contents of the web page. Since each record was stored in its own row, the program parsed each element that began with the row tag, “//tr”, and stored it as an element in Python. After confirming that, we correctly read in all 9 columns for each of the 804 rows of data, the program iterated through the elements, selects the first item in each column (the column name) and stores it as a tuple, coupled with an empty list. The program appends data to each of these 9 empty lists in the following manner. It iterates through each element, or row, extracts the text content, and appends it to the empty list of the corresponding column. Checks are in place to ensure that no empty rows are appended, and that any numeric data is converted to an integer. After this iteration process is complete, the program prints the length of each column to ensure that each has a uniform length. The resulting list of tuples is then converted to a dictionary, in which the column name was the key, and the list of values in that column was the value. Finally, the dictionary was converted to a pandas dataframe, at which point it was further cleaned.

The cleaning process primarily involved dealing with missing data. Most significantly, the web scraping process described above, as well as a similar process using beautiful soup, were both unsuccessful in obtaining the alt text stored for the image representing move category. Ultimately, since this field was unattainable, and not necessary for the research question, it was removed from the data frame.

Missing values in numeric fields were represented either by dashes or blanks. For all fields except for probability, a missing value was replaced with a zero. Missing values in probability field were replaced with “NaN,” or not a number, since replacing with zero would inaccurately convey that there was no chance of any Pokémon using that move. The accuracy field was primarily comprised of integers, but it periodically had an infinity symbol. A regular expression was applied to substitute the infinity symbol for the integer value of infinity.

The final preprocessing task was to summarize each of the fields to understand their data types, as well as the distribution of numeric values. The pandas info and describe functions were used to perform this summary, the results of which are shown below.

A screenshot of a cell phone

Description automatically generated

## Methods of Analysis

Research Question 3 seeks to understand which types of Pokémon have the most impressive moves. This calls for an analysis of the power, accuracy and power points (PP) of the moves in our dataset. Probability, tech machine and description of effect fall outside the score of this analysis.

The main program, *Pokemon\_Project\_BJW\_MS.ipynb,* will output a series of printed lists, identifying the Pokémon that have moves with the highest measures of center for power, accuracy and power points. Power and power points will be summarized with mean. Since some moves have infinite accuracy, this will be summarized using median to avoid skewed results. The program will also produce a list that identifies which types have more than 1 move with infinite accuracy.

The question also investigates which types of Pokémon are the strongest when considering both the power associated with a Pokémon on its own combined with the power associated with a given move. The question will draw upon the original Pokémon dataset used in question 1 to identify which types of Pokémon have the highest total power overall. This total power score is calculated by combining the scores of all attributes: HP, (special) attack, (special) defense and speed.

The output will be a ranked dataset of the Pokémon types with the highest Total power on average. The program will then combine this ranked dataset with the moves data and reprint the Moves lists, including the overall rank of the strength of the Pokémon type. The program will also output visuals in the form of bar graphs that compare Pokémon character strength with power, accuracy and power points of moves.

## Overall Description of Program

The main program for analysis, *Pokemon\_Project\_BJW\_MS.ipynb,* progresses through the following steps. It groups the moves dataset by Pokémon Type and calculates the average of each numeric measure of move strength, which it stores in a data frame named compareMeans. Next, it groups by type and summarizes each by Median, which it stores in a dataframe named compare Median. These two tables are merged to create a final result set, named compareMixed, which contains the average Power and PP and median Accuracy by Pokémon Type. The program creates new data frames for Power, PP and Accuracy, sorted in descending order, such that the Types with the strongest move measures are listed first. It then iterates through each sorted data frame using the pandas iterrows function to print out the top 3 Types for each category.

The program next reads in the Pokémon dataset and calculates the highest average total power by Pokémon Type. It uses the Pandas rank function to rank each Pokémon type from most to least powerful. It joins the ranked and summarized list to the compareMixed dataset containing average/median move strength by type. It then re-iterates through the list to identify the overall strength of the top 3 Types in each move category.

In order to facilitate visual analysis, the program uses the pandas loc function and if condition logic to assign a color based on the strength rating of Pokémon types. The top 6 were assigned to be represented in red, the bottom 6 in yellow and the middling 8 in red. These colors were used in resulting visuals as a way to identify overall strength of Pokémon when comparing move strength.

The program first created a bar graph comparing average overall strength of Pokémon characters, sorted in order of most powerful to least powerful. It produced similar graphs for strength of moves attributed to types, by power, accuracy and power point measures of center. Each graph used the color scheme previously described to identify the most and least powerful Pokémon characters.

## Conclusions

The Pokémon with the most powerful moves on average are Fire, Dragon and Water. Of these, Dragon is the most powerful on its own, ranking number 1 out of 18 Pokémon types in average Power. Figure 8 shows that Dragon types have highly accurate moves as well, with a group median of 100 out of 100. To balance the scale, Figure 9 shows that their moves have the lowest power points on average, which means that they can use them less often than other Pokémon types.

This pattern is evident throughout the Pokémon types: more powerful characters, in red, tend to have power points, or ability to reuse moves in a battle. Weaker characters, such as Poison, normal, bug and grass, have a stronger ability to reuse their moves. This characteristic helps even out the competition between weak and strong characters, and offers a balance to moves that are especially powerful or weak.

Figure 10 demonstrates that stronger Pokémon tend to have less accurate moves. Surprisingly, weaker Pokémon also have the least accurate moves. It also illustrates that accuracy tends to be skewed high: all but 2 Pokémon have a median accuracy of over 80%. Eight have a median accuracy of 100%, and there are four types that have at least 2 moves with infinite accuracy. Since accuracy does not elevate weaker players, or players with weaker moves, this is not a move characteristic that evens the playing field in the way that the Power Points characteristic does.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# **Question 4**

Which Pokémon types are the most popular, as measured by mentions on social media platforms?

## Data Source and Cleaning

This data used the Twitter developer API and Python tweepy package to query Twitter for a list of search terms. The search term list was sourced from the data set of Pokémon characters provided by Kaggle.

In order to create the list of search terms, the Pokemon\_Tweets\_Collection.ipynb program read the Pokémon dataset using a csv reader. It stored the data as a list of dictionaries, before iterating through said list to extract a list of Pokémon names. It filtered out all Pokémon names which contained “Mega” because the names had an irregular format and were often variations of other names already on the list. The resulting, clean and unduplicated list of names was modified by adding a hashtag (‘#’) symbol to the beginning of the name. This specifies that the search should return tweets in which a Pokémon was mentioned in a hashtag, instead of in plain context in the rest of the tweet. The hashtags signify that the user is seeking to highlight the identify of a Pokémon and is more formal than simply including the name in a tweet.

The app login function stores twitter API credentials, handles the login process and connects to the tweepy API. The search function queries statuses for a given search term using the tweepy cursor and returns a specified number tweets that meet the search criteria in json format. The save\_to\_DB function saves tweets to a specified MongoDB database and collection. The program uses the described functions to iterate through the list of Pokémon name hashtags and search for tweet that contains that text. Since the list of Pokémon hashtags is long, the program searches segments of the list at a time and waits for the search limit of the Twitter API to reset. Once the tweets are loaded into the MongoDB, the program uses the pymongo package to save all documents in the collection to a list of documents. It removed the tweet id in the process. It then used json.dump to save the result set to a JSON file.

The main program, *Pokemon\_Project\_BJW\_MS.ipynb,* uses json.load to open the JSON file and prints a confirmation stating the number of records it contains. It then iterates through the list of json-formatted tweets and extracts each tweet’s list of hashtags, nested in a list of entities. It appends each item in the list of hashtags to a list, hashes. Since each tweet can have more than one hashtag, there are additional words that are not in the original list of search terms. The program therefore iterates through the hashtag list, and if a hashtag is in the original list of search terms, it is added to a new, trimmed down list, called hash\_names. Since some Pokémon names are included in more than one tweet, the list has duplicate entries. A dictionary is created to count the number of times each hashtagged Pokémon name occurred. Finally, the original list of search terms is processed to eliminate any Pokémon names with a space: hashtags do not include spaces and so cannot be tweeted exactly the way they appear in the dataset. This also eliminates variants of a Pokémon character, such as Mega Pokémon. The program strips the names of white space and detects whether the name contains a space. If there is no space, the Pokémon name is one word only and is added to the list of plain names. The plain\_names list is then parsed through hash\_names to determine which Pokémon have been tweeted about and which have not. If a Pokémon name is in hash\_names, it is added to the list, included. If it has not, it is added to the list, missing.

The program also a Popular column to the original Pokémon dataset, which provides a Boolean indicator of whether a Pokémon was mentioned in a tweet. It first creates a conditions list to test whether or not a name in the Pokémon dataset is included in the list of tweeted names. If a Pokémon is included in the tweeted list, it is assigned a value of True. Otherwise it is assigned a value of false. It also creates a smaller subset of data containing only name, type and Popular for the first and primary research question.

## Methods of Analysis

The primary research question is which Pokémon types are the most popular, as measured by mentions on social media platforms? Follow-up questions revolve around understanding characteristics of popular Pokémon, including legendary status and generation. These questions use the following fields: Pokémon popularity, type, legendary status and generation. The program also calculates the number of Pokémon in each category (i.e. number of Popular Fire Pokémon),

The expected output begins with a series of printed statements and progresses to various visuals. Upon adding the popularity indicator to the original dataset, the program prints a statement indicating the percentage of Pokémon that have been mentioned in a tweet. Next, the program produces a stacked bar chart showing the popularity breakdown of each Pokémon type. The program then creates a normalized stacked bar chart in which popularity is expressed as a percent of the group. The visual is supplemented with a printed statement indicating the 5 most popular and 5 least popular Pokémon types, based on the percent that were mentioned in a tweet.

Follow-up questions take the same approach: both are assessed using a printed statement and an accompanying visual. When analyzing legendary status and popularity, the program prints out a statement identifying whether legendary or non-legendary Pokémon are most popular, and lists the percentage of popularity for each. It then produces a stacked bar chart to compare the proportion of popularity for legendary versus non-legendary Pokémon. When analyzing generational popularity, the program prints a statement listing the most popular generation and percentage popularity, as well as the last popular and accompanying percentage. It then produces a stacked bar chart of proportion popularity, sorted in order of most to least popular generations.

## Overall Description of Program

The program reads in and processes the twitter data as previously described in the pre-processing steps. In addition to adding a popularity column to the Pokémon dataset that filters out variant Pokémon, it creates a list of Pokémon that were included in a tweet, and Pokémon that were not. It calculates the length of each list, determines the percent of total for each, and prints a statement identifying the percent of Pokémon that were mentioned in a tweet.

To perform the analysis of popularity by type, the program creates a duplicated list of Pokémon by type and popularity. That is to say, if a Pokémon has two types, it is listed one under each type. The program groups the resulting data frame by type and popularity and counts the number in each. It uses matplotlib to create a stacked bar chart that plots the number of popular Pokémon in each type, and plots the number of unpopular Pokémon by type on top, as a second layer. To facilitate a comparison that accounts for the overall size of a type group, the program calculates the number of Pokémon of each type, adds this number to the data frame containing counts by type and popularity, then calculates the percent of group by dividing the subgroup count by group count. The two values (popular and unpopular) for each type add up to 100%. This normalized dataset is plotted in the same manner as the raw dataset: percent popular by type is plotted first, followed by percent unpopular by type. The program then iterates through the data frame used to create the bar plot and prints a statement containing the types and percent popularity of the 5 most popular and 5 least popular types. The program also exports a .csv file containing the counts of each type/popularity subgroup. This result set is used in R to create a Sankey diagram, illustrating the types of Pokémon that supply the popular and unpopular groups. The program also used the methodology described in question 3 to plot the average Pokémon power by type to understand whether there was any connection between Pokémon power and popularity.

The program turns next to analysis of legendary status. It groups the Pokémon dataset by legendary status, adding the ‘total’ field to record the total number of legendary and non-legendary Pokémon. It next groups by legendary status and popularity type, calculating the size of each subgroup. It then joins the two resulting data frames together and calculates each subgroup’s percentage of the larger group (i.e. the number of popular-legendary Pokémon is x% of the number of legendary Pokémon). The program prints a statement summarizing the results, identifying whether legendary or non-legendary Pokémon has a larger percentage of popular Pokémon. Finally, it uses matplotlib to plot a stacked bar plot, with popular Pokémon plotted on the bottom in blue, and unpopular Pokémon on top in red.

It follows a similar process to analyze popularity of generations. It calculates the total number of Pokémon in each generation and stores it in a data frame. It then calculates the number of popular and unpopular Pokémon in each generation and stores it in another data frame. The total size of each generation is added to the data frame containing generation and popularity counts. The program then calculates the proportion of popular versus unpopular for each generation. It prints a statement confirming the most and least popular generations, identifying of each that are popular. It creates a stacked bar plot, plotted in order of most to least popular generation. Percent popular is plotted first on the bottom in blue, with percent unpopular plotted on top in red.

## Conclusions

A screenshot of a cell phone

Description automatically generated

The initial analysis that examined number of popular Pokémon by type revealed that no single type is tweeted about significantly more than any other type. The Sankey diagram in Figure 11B illustrates that Popular Pokémon are “sourced” from each of the 18 Pokémon types. Types that have a large amount of popular Pokémon also tend to have more Pokémon in general, as illustrated in Figure 11A. The normalized comparison in Figure 12A offers additional insight. When comparing the percent of a given type that are popular, the differences in type popularity become clearer. Over 40% of Normal, Fairy and Dragon Pokémon were mentioned in a tweet, while less than 25% of Steel and Fighting Pokémon were mentioned. Water Pokémon, which appeared to have the most popular Pokémon in the first graph, is in the middle 50% of the group.

An initial visual analysis reveals no strong connection between the average power of a Pokémon type and the percentage of that type that is popular. The most popular types, Normal and Fairy, are among the least powerful Pokémon. The most powerful Pokémon, Dragon, is the third most popular. However, the second most powerful, Steel, is the least popular.

A smaller percent of legendary Pokémon were mentioned in a tweet than their non-legendary counterparts, as seen in Figure 13. This is ironic, given that the term ‘legendary’ implies that there is something particularly special about these Pokémon that gives them a well-known reputation, and presumably, popularity. That said, there are only 38 legendary Pokémon in the clean data set of 705 Pokémon. The title ‘legendary’ may refer to the fact that they do not appear often and are harder to find, which could explain why the group is tweeted about with less frequency than non-legendary Pokémon. This question could benefit from a deeper textual analysis that examines the sentiment surrounding mentions of these Pokémon. If sentiment is one of surprise or excitement at the opportunity to interact with them, then this would confirm the idea that legendary Pokémon, though rare, are popular when discovered.

The analysis of generational popularity shows that there is not a huge difference in popularity between generations. The general pattern is that the older generations, 1-3, have a higher percentage of popular Pokémon than new generations. This is likely due to the fact that these Pokémon have been in circulation longer and have had more time to build a reputation and following. Anecdotally, since Pokémon games have been revived in recent years after their introduction to American culture 15-20 years ago, it’s likely that older Pokémon are more popular among users because of nostalgia and familiarity.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# **Overall Project Conclusion**

The four research questions offered a rich insight into the world of Pokémon and the characters that fill it. They foster an understanding of the most powerful characters and types of Pokémon, and their influence in various gaming platforms and in popular culture. Two themes have emerged in this process of analysis. The first is that the creators of the Pokémon universe carefully craft these characters to enhance game play. The second is that the older generations of Pokémon stand apart from the rest.

Pokémon are carefully designed to foster competition, rather than stifle it with an indomitable player or with characteristics that do not fit with the context. While there certainly are stronger and weaker Pokémon, there is no one character that is unbeatable. Mega Evolution offers a boost in power, but the amount increased is the same across the board for all Pokémon. Similarly, while certain types of Pokémon are stronger and have more powerful moves, the creators of the game maintain balance by limiting the number of times they can use their moves. Finally, the authors determine Pokémon characteristics based on the platform and fine tune them to encourage competition in a given context. For example, characters on the Pokémon Go platform have a higher attack power than those on gaming consoles. This may be because Pokémon Go was designed to encourage people to get outside and compete with each other in real life, and augmented attack power could fuel competition and encourage this behavior. It is this attention to detail that has allowed this game to evolve into its own mini-universe and influence culture over the course of multiple decades.

While there have been eight Pokémon generations, or sets of new characters, the first three generations maintain a special place in the Pokémon universe. In the contexts of both gaming consoles and Pokémon Go, characters from generations 1-3 are stronger than those from newer generations, 4-8. These three oldest generations were also the most popular, with generation 1 leading the way: almost 40% of Pokémon belonging to that generation were mentioned in a tweet in the 7 days prior to the analysis. The power of moves was not compared based on generation, which offers an excellent opportunity for future analysis. The popularity of the earlier generations could be attributed to their comparative strength, but there is no strong relationship between popularity and strength. It is far more likely that the company intentionally promotes the older generations. Many of its users have followed the game since its beginning and have probably developed a loyalty to its original characters, as reflected in their tweets. The Pokémon brand and visual identify are based around some of these initial characters, such as Pikachu and Charizard. It would behoove them to cater to this public base by empowering these Pokémon to excel in every platform.

Given the opportunity to further this study, we would employ machine learning models to glean more insight into the relationship between Pokémon characteristics and popularity. The analysis would take into account power, gaming platform, mega evolution and moves to predict a character’s popularity on Twitter, or even Facebook. Future research would expand its scope by universalizing the search terms (i.e. searching for Pokémon names in all lower-case or upper-case characters), collecting more than 1 tweet for each Pokémon, and looking for associations between gaming platforms and particular Pokémon. Furthermore, it would perform a sentiment analysis on the text of the tweet. For example, if a Pokémon is more powerful on one platform versus another, it could analyze the attitude of tweets that include the Pokémon name and a particular platform.

The rich history and culture surrounding the Pokémon universe offers ample opportunity for further research and analysis.