***The Pokemon Dataset***

Sai Tejasri Yerramsetti

Email: [ysaitejasrigrad@gmail.com](mailto:ysaitejasrigrad@gmail.com)



Introduction

**Dataset**

The Pokémon Dataset is focused on the stats and features of the Pokémon in the Pokémon

RPG games until Generation 6.

This database includes 21 variables for each of the 721 Pokémon of the first six generations,

plus the Pokémon ID and its name. These variables are briefly described here:

● Number. Pokémon ID in the Pokédex.

● Name. Name of the Pokémon.

● Type\_1. Primary type.

● Type\_2. Second type, in case the Pokémon has it.

● Total. Sum of all the base stats (Health Points, Attack, Defense, Special Attack, Special

Defense, and Speed).

● HP. Base Health Points.

● Attack. Base Attack.

● Defense. Base Defense.

● Sp\_Atk. Base Special Attack.

● Sp\_Def. Base Special Defense.

● Speed. Base Speed.

● Generation. Number of the generation when the Pokémon was introduced.

● isLegendary. Boolean that indicates whether the Pokémon is Legendary or not.

● Color. Color of the Pokémon according to the Pokédex.

● hasGender. Boolean that indicates if the Pokémon can be classified as female or male.

● Pr\_male. In case the Pokémon has Gender, the probability of its being male. The

probability of being female is, of course, 1 minus this value.

● Egg\_Group\_1. Egg Group of the Pokémon.

● Egg\_Group\_2. Second Egg Group of the Pokémon, in case it has two.

● hasMegaEvolution. Boolean that indicates whether the Pokémon is able to Mega-evolve

or not.

● Height\_m. Height of the Pokémon, in meters.

● Weight\_kg. Weight of the Pokémon, in kilograms.

● Catch\_Rate. Catch Rate.

● Body\_Style. Body Style of the Pokémon according to the Pokédex.

**Question 1**

Question:

Suppose this data is a SQL table called ‘PokemonStats’. In an SQL dialect you are most

comfortable with, find the top 3 Pokemon in terms of total stats of each type (primary type,

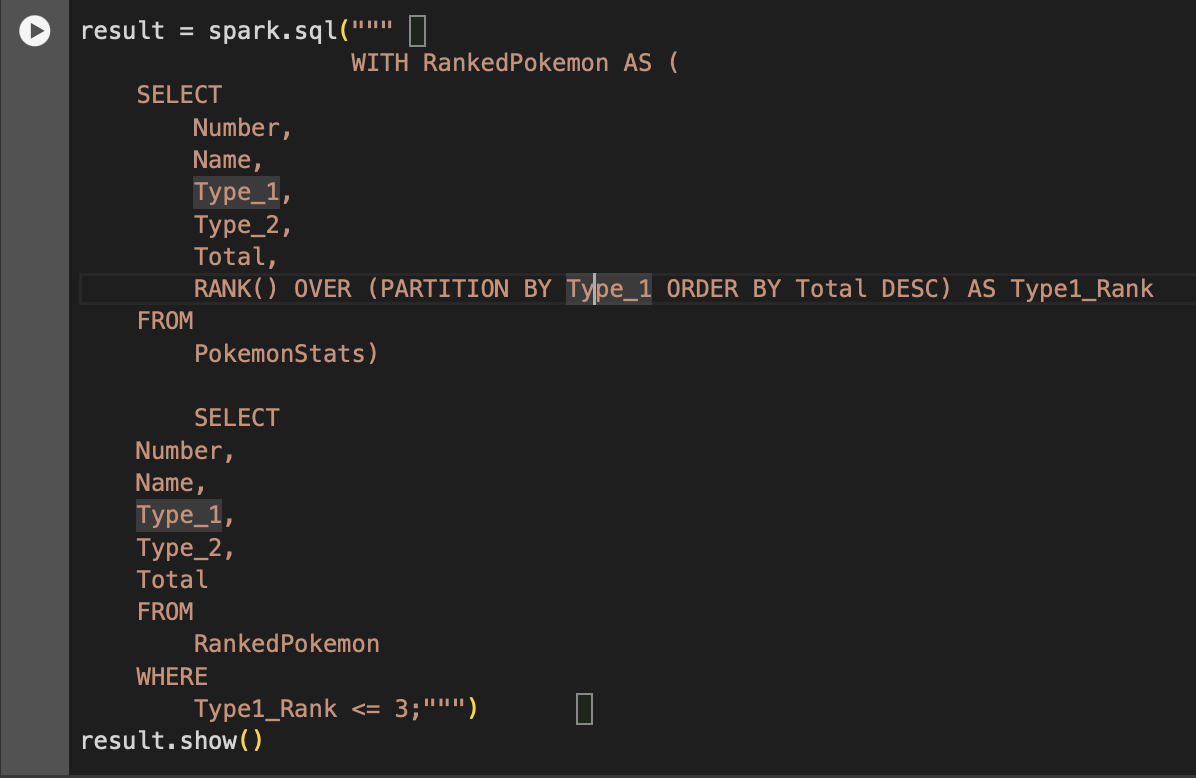
Type\_1). Your answer should include: 1) the SQL dialect you are using; 2) The SQL query used

to answer this question; 3) The returned result.

**1. SQL Dialect:**

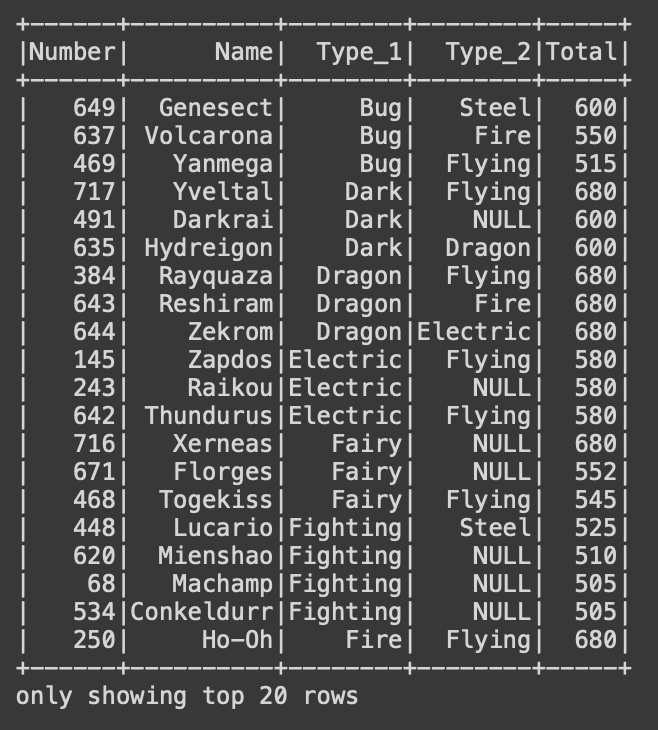
The **SQL dialect** I considered and used is the **Spark SQL**, which is a part of Apache Spark. I have decided to use Apache Spark because of its capabilities of traditional SQL with optimizations and functions suitable for big data processing.

**2.** **The SQL query:**



The above is the SQL query returns a table with columns Number, Name, Type\_1, Type\_2, and Total, showing the top 3 Pokémon for each primary type based on their total stats. The sample output you've provided illustrates this perfectly, listing Pokémon like Genesect, Volcarona, and Yanmega for the Bug type, and similarly for other types.

**3. The Returned Result**



Conclusions:

Based on the SQL logic in Python, I identified the top 3 Pokémon by total stats for each primary type. Following are the results for a few types:

**Bug Type:**

1. Genesect with a total of 600.
2. Volcarona with a total of 550.
3. Yanmega with a total of 515.

**Dark Type:**

1. Yveltal with a total of 680.
2. Darkrai and Hydreigon, both with a total of 600.
3. Umbreon with a total of 525.

**Dragon Type:**

1. Rayquaza, Reshiram, and Zekrom, each with a total of 680.
2. Kyurem with a total of 660.
3. Dragonite, Salamence, Latias, and Latios, each with a total of 600.

**Question 2**

Question:

Imagine a new Pokemon game where you are only allowed to collect ONE type of Pokemon.

Similar to other Pokemon games, your goal is to have the strongest battlers and defenders for

battles and raids. Which type will you pick? Why? Please provide 2-3 supporting statistics /

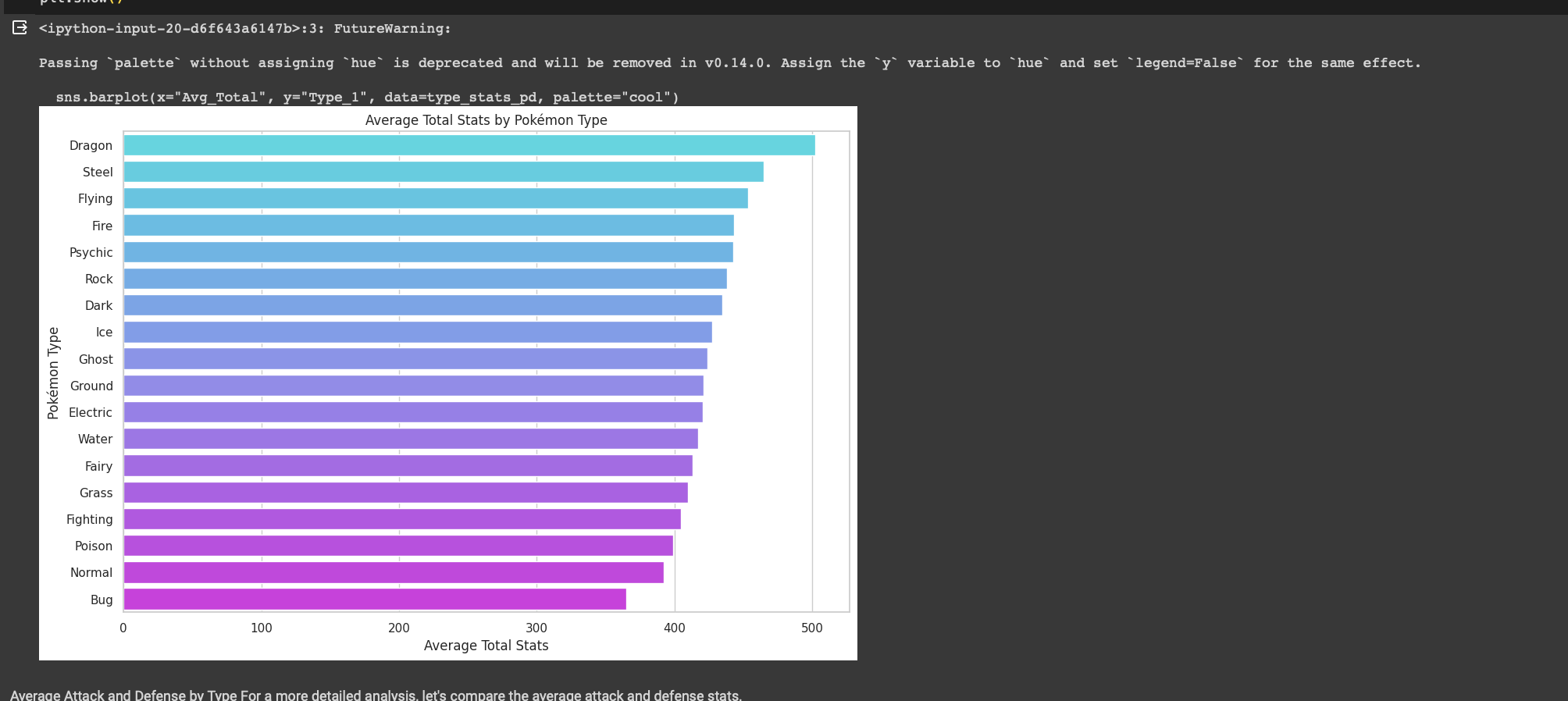
charts.

**Best Type for Collecting Pokémon**

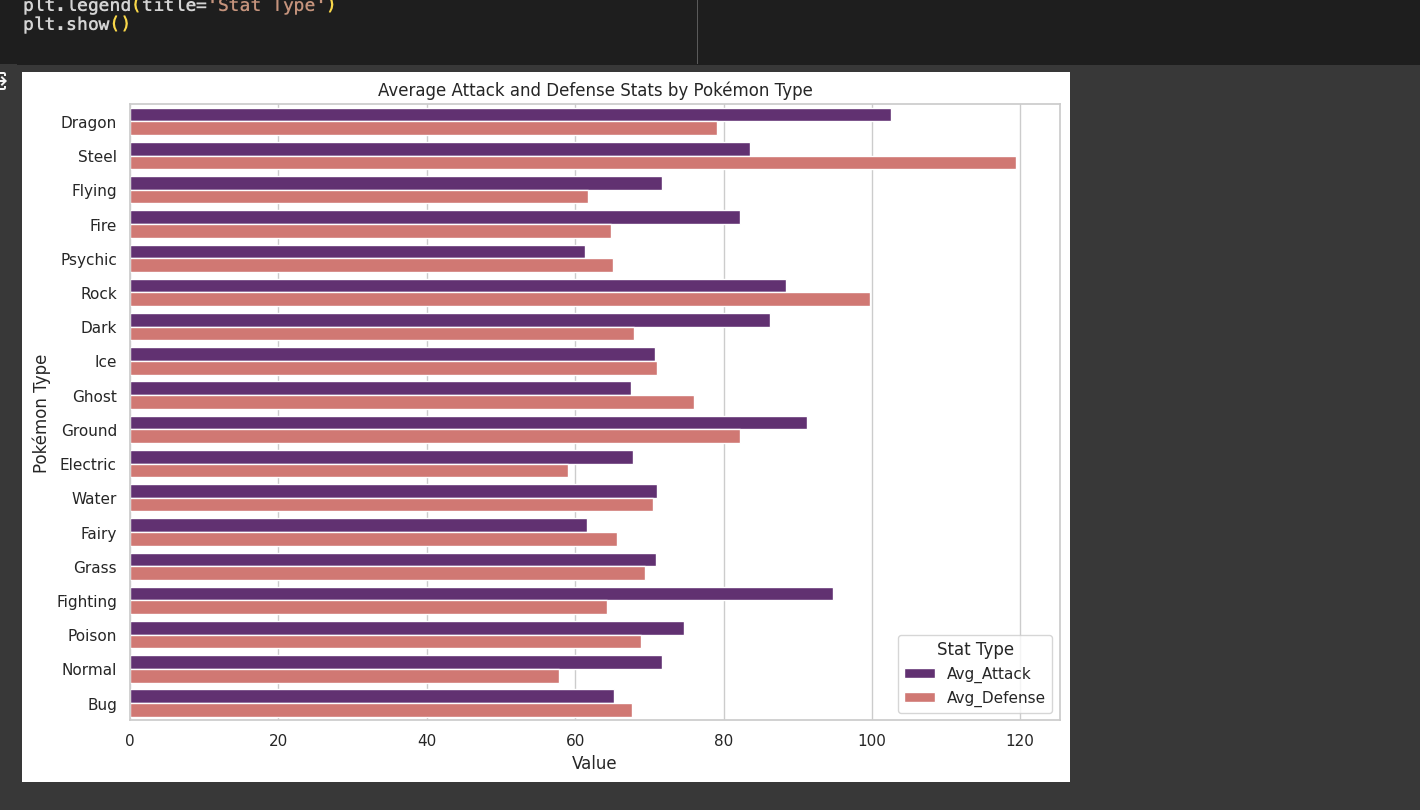
To determine the best type of Pokémon to collect for battles and raids, focusing on strength, I considered to look at overall statistics and distributions of key attributes like Total, Attack, and Defense stats across types.

By analyzing these statistics, I generated charts to visualize the data, in selecting the most suitable type. ​​

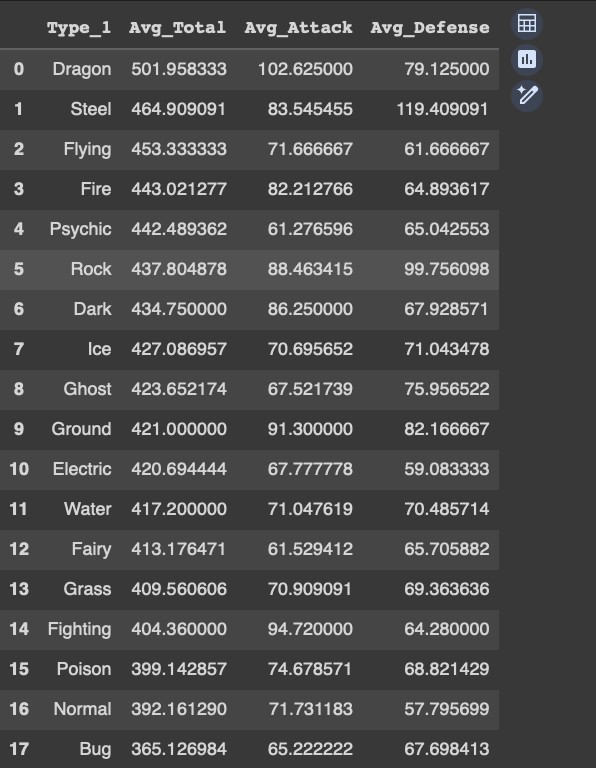
**Data visualization:**



**Mean Total Stats by Pokémon Type**: This chart shows the average total stats (a combination of HP, Attack, Defense, Special Attack, Special Defense, and Speed) for each Pokémon type. Types with higher mean total stats are generally more versatile and potentially stronger in battles.



**Mean Attack and Defense Stats by Pokémon Type:** The second chart compares the mean attack and defense stats across types, highlighting the offensive and defensive capabilities of each type. A balance of high attack and defense stats can be crucial for both dealing damage and sustaining it during battles.



Conclusion:

Based on the above charts, the types with the highest mean total stats and a good balance of attack and defense capabilities would be most suitable for collecting. **Dragon type stands out as a strong contender** due to its **high average in total stats, attack, and defense**, making it an excellent choice for battles and raids. Additionally, types like **Steel and Psychic also show strong average stats, making them competitive choices.**

**QUESTION 3**

Question:

If you want to predict whether the Pokemon is able to Mega-evolve (a.k.a. predict the field

hasMegaEvolution using other fields), which models would you use? List your top 3 models with

pros and cons for each one.

Predicting whether a Pokémon can Mega Evolve (the hasMegaEvolution field) based on other attributes can be approached with various machine learning models. Here are the top 3 model that I would like to consider is:

**1. Logistic Regression:**

Pros:

* Interpretability: Logistic regression provides easily interpretable results, allowing you to understand the relationship between each feature and the probability of mega evolution.
* Efficiency: It is computationally inexpensive, making it fast to train even on large datasets.
* Simplicity: Works well with linearly separable data and is straightforward to implement.

Cons:

* Assumption of Linearity: Assumes a linear relationship between the log odds of the dependent variable and the independent variables, which might not always hold true.
* Performance: May not perform as well with complex relationships or non-linear data as some other models.

**2.Random Forest Classifier**

Pros:

* Handling Non-linearity: Can capture non-linear relationships between features and the target variable without needing feature transformation.
* Feature Importance: Provides insights into which features are most influential in predicting the outcome, aiding in feature selection.
* Robustness: Less prone to overfitting when compared to decision trees, due to the ensemble approach.

Cons:

* Interpretability: More complex to interpret than simpler models like logistic regression due to its ensemble nature.
* Computationally Intensive: Training can be computationally more expensive, especially with a large number of trees or deep trees.
* Hyperparameter Tuning: Requires careful tuning of hyperparameters to prevent overfitting and ensure optimal performance.

**3.Gradient Boosting Machines (GBM):**

Pros:

* Performance: Often provides superior predictive accuracy and is effective on a wide range of problems, including non-linear datasets.
* Flexibility: Can handle missing data and doesn't require scaling of data. Offers support for both numerical and categorical features.
* Feature Importance: Similar to Random Forest, GBMs can provide insights into feature importance.

Cons:

* Overfitting: Without proper parameter tuning and regularization, GBMs can overfit to the training data.
* Training Time: Can be time-consuming to train, especially with large datasets and complex models.
* Complexity: More complex to tune and interpret compared to simpler models. Requires understanding of various hyperparameters to achieve the best performance.

Based on the above pros and cons I would like to experiment with logistic regression considering the variables in the dataset.

**Question 4**

Question:

Pick one model and implement it in a language you are most comfortable with (preferably

Python or R). Please do not use the ‘Catch\_Rate’ field (if you are Pokemon fan you know why

:)). How well is your model doing and what fields did you end up using?

Your answer should include: 1) The code of implementing the model (incl. feature processing,

model fitting and cross validating); 2) The formula/description of your final model along with the

performance measure. 3) In addition to the code and the model specification, if you choose to

submit a presentation/ dashboard as part of your writeup, you can present your results in any

way you like.

I would like to pick **Logistic Regression** as my model and **python** as my programming language as the size of the dataset is small.

**1. The code of implementing the model:**

For the code, I would like to share a “.ipynb” extension file (“The Pokemon Dataset.ipynb”). It contains code for questions 1,2 and 4. The file consists of detail explanations as well which includes feature processing, model fitting and cross validating.

**2. The formula/ description of Logistic Regression:**

The final model implemented is a logistic regression model designed to predict whether a Pokémon is legendary based on various features, excluding 'Catch\_Rate'. Logistic regression is suitable for binary classification tasks, such as this one, where the target variable is binary (legendary or not legendary). The logistic regression model estimates probabilities using a logistic function, which is defined as:

​

Where:

* P(Y=1) is the probability that the Pokémon is hasMegaEvolution.
* e is the base of the natural logarithm.
* X1, X2,X3, ….Xn represent the feature variables included in the model (such as 'Total', 'HP', 'Attack', 'Defense', 'Sp\_Atk', 'Sp\_Def', 'Speed', 'Generation', and one-hot encoded variables for 'Type\_1' and 'Type\_2').
* β0, β1, …, βn are the coefficients for the intercept and each feature in the model, which are learned during the training process.

**Final Model Features**

The model includes the following features, after preprocessing

* Numerical features: 'Total', 'HP', 'Attack', 'Defense', 'Sp\_Atk', 'Sp\_Def', 'Speed', 'Generation', and 'Weight\_kg' (with outliers handled).
* Categorical features: 'Type\_1' and 'Type\_2', which are one-hot encoded.

**Performance Measures for Logistic Regression:**

(0.7862068965517242,

0.7452107279693486,

array([[109, 28],

[ 3, 5]]),

' precision recall f1-score support\n\n False 0.97 0.80 0.88 137\n True 0.15 0.62 0.24 8\n\n accuracy 0.79 145\n macro avg 0.56 0.71 0.56 145\nweighted avg 0.93 0.79 0.84 145\n')

**Overview of Metrics:**

Accuracy: 78.62%

Cross-Validation Accuracy: 74.52%

Confusion Matrix:

True Negatives (TN): 109

False Positives (FP): 28

True Positives (TP): 5

False Negatives (FN): 3

**Classification Report Details:**

* Precision for Non-Mega Evolvable (False): 97%. Indicates a high likelihood that Pokémon predicted not to Mega Evolve indeed cannot.
* Recall for Non-Mega Evolvable (False): 80%. Shows that 80% of the actual non-Mega Evolvable Pokémon were correctly identified.
* F1-Score for Non-Mega Evolvable: 88%. Reflects a good balance between precision and recall for the majority class.
* Precision for Mega Evolvable (True): 15%. Suggests a low accuracy in predictions when the model predicts a Pokémon can Mega Evolve.
* Recall for Mega Evolvable (True): 62%. Indicates that the model was able to identify 62% of all actual Mega Evolvable Pokémon, suggesting sensitivity towards the minority class but with many false positives.
* F1-Score for Mega Evolvable: 24%. A lower F1-score for the minority class indicates an imbalance in model performance favoring the majority class.

**My Insights**

In the course of my analysis, I employed logistic regression to predict a specific binary outcome: whether a Pokémon is capable of Mega Evolution. This process involved the application of SMOTE (Synthetic Minority Over-sampling Technique) to address the issue of class imbalance and the selection of both numerical and categorical features for model training. The numerical features included were 'Total', 'HP', 'Attack', 'Defense', 'Sp\_Atk', 'Sp\_Def', 'Speed', 'Generation', and 'Weight\_kg', with outliers appropriately managed. The categorical features, 'Type\_1' and 'Type\_2', underwent one-hot encoding.

The results of this approach yielded a notably high true negative rate, underscoring the presence of significant class imbalance within the dataset. Recognizing the importance of thorough exploratory data analysis (EDA) and feature engineering in the model development process, I considered additional steps to enhance the model's predictive accuracy. These steps included visualizing each column against the target variable (hasMegaEvolution) to identify patterns and check for skewness, applying transformations (e.g., logarithmic, polynomial) to normalize feature distributions, and assessing the relevance of each feature through statistical tests such as t-tests and ANOVA.

Given more time, these methodologies would be systematically applied to refine the dataset further and potentially uncover more insightful relationships between the features and the target variable. This belief stems from an understanding that meticulous EDA and feature engineering often play a more crucial role in predictive modeling than the mere application of complex algorithms. Efforts were also made to address missing data and outliers, ensuring the model was trained on clean and representative data.

This comprehensive approach highlights the iterative nature of data science projects, where initial findings lead to further investigations and model adjustments, ultimately aiming for the most accurate and robust predictive model possible.