FDS: Pokemon Battles prediction 2025

Giannicola Mongelli Francesco Nardiello Karim Ciacciarelli

1 Feature Engineering

First, we did a data cleaning phase, where the only inconsistency we identified was that not all Pokémon were at level 100. To ensure uniformity across the dataset, we corrected this issue by retaining only those entries where the Pokémon level was equal to 100. During the feature engineering phase, unlike analyses based solely on static attributes such as Pokémon types or base stats, the process focuses on a dynamic examination of the battle timeline, analyzing each turn in detail to compute damage, healing, KOs, switches, and status conditions. We also observed that features derived from base statistics had limited influence on model performance, whereas dynamic features provided much greater predictive power.
We observed that the choice of features had a far greater impact on model accuracy than the choice of the model itself. In fact, during the feature engineering phase, we noticed that after a certain point, adding more features led to a decrease in accuracy. In our opinion, this effect can be explained by the introduction of redundant or irrelevant features, which increase noise and reduce the model's ability to generalize effectively.

1.1 Features Importance

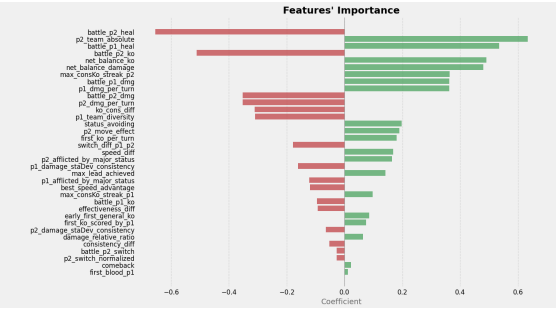

Figure 1: Features' importance

To analyze and select the most relevant features, we compute and visualize the relative impact of each factor. The *Figure 1* highlights the features that the machine learning model considers most influential in determining the battle outcome. Green bars extending to the right indicate factors that increase Player 1's probability of winning, whereas red bars extending to the left represent those that reduce it.

2 Machine learning models

We primarily relied on Logistic Regression and Random Forest (Bagging), focusing on these models due to their stability and reliable performance. We also experimented XGBoost, and the corresponding code is available on GitHub repo. In the two submissions where Logistic Regression was used, the main difference lay in the feature engineering: for each, we applied a distinct feature engineering strategy to better capture the predictive patterns in the data. The data was split into a training set (70%) and a validation set (30%), and we set up a GridSearch to identify the best hyperparameters. Logistic Regression was placed inside a pipeline with StandardScaler, as the model is sensitive to feature scaling and standardization is essential for optimal performance. For the GridSearch, we used ROC-AUC as the scoring metric, considering it more reliable than accuracy for evaluating discriminatory ability, and applied 5-fold cross-validation. Once the optimal model was selected, we evaluated its performance on the validation set.

3 Used Strategies

We explored several strategies to improve model performance. One approach involved creating features based on Pokémon types and their interactions (for example, Water being strong against Fire). However, this strategy, along with the inclusion of base Pokémon statistics, did not lead to a significant improvement in predictive accuracy.
We also observed that the choice of machine learning model had relatively little impact on performance. Instead, the feature engineering process proved to be the key factor influencing accuracy. We noticed that adding too many features eventually led to a decline in model performance, likely due to overfitting caused by redundant or irrelevant variables. To gain a clearer understanding of how the model behaved, we complemented our evaluation with additional visualizations, including the confusion matrix.