FDS: Pokemon Battles prediction 2025 - Team dugTRIO

Line 60: emiliano.grandis5@gmail.com, eyahouij02@gmail.com, filippovitozzi@gmail.com

Filippo Vitozzi 2265118, Emiliano Grandi 2258965, Eya Houij 2238231

1. Feature Engineering

We created a diverse set of features to summarize each battle's key factors. These features were derived from both static attributes and dynamic battle events. For example, we incorporated Pokémon base stats by computing differences between Player 1's team and Player 2's team. This captures any inherent team strength advantage. We encoded type effectiveness by using the known Pokémon type matchup chart to calculate features such as net type advantage and coverage difference (which side's team had broader type coverage). We also included a STAB (Same-Type Attack Bonus) difference to reflect if one player benefited more from using moves of the same type as their Pokémon. Additionally, we engineered features to capture battle dynamics: for instance, a round count ratio between players, an HP momentum feature tracking which side was losing health faster, and a first-strike advantage measuring which player's Pokémon tended to attack first in each round. We also modeled strategic behavior through features like smart switch difference, which quantifies the effectiveness of Pokémon switches by each player. We compared this alignment for Player 1 and Player 2 as a feature, assuming that a player whose moves better utilize their Pokémon's strengths might have a higher chance of winning. We also tracked status effects: one feature flags if Player 1's Pokémon ended the battle with a detrimental status condition, and another measures the difference in how many status conditions each side inflicted on the other. By engineering over twenty such features, we distilled the complex battle logs into a structured representation. The rationale behind this extensive feature engineering was to incorporate as much domain insight as possible – features capturing type advantages, stat differentials, and tactical momentum should help the model learn which conditions correlate with winning a battle.

2. Models Evaluated

We experiment with three classes of supervised models of increasing expressive power: a linear baseline, a lightweight ensemble, and a gradient-boosted decision tree architecture. The goal is to progressively assess how much non-linearity is required to capture the strategic dependencies encoded in the engineered features.

2.1. Logistic Regression (Baseline).

Our first approach is a regularized Logistic Regression model trained within a standardized pipeline. We perform an extensive grid search over penalties (L1/L2), inverse-regularization strengths C, and optional class weighting. Despite careful tuning and 5-fold stratified validation, the model saturates around a training accuracy of 0.796, failing to exceed the 0.80 threshold.

2.2. Soft Voting Ensemble.

To introduce limited non-linearity while retaining interpretability, we train a soft-voting ensemble combining: (i) a scaled Logistic Regression classifier, (ii) an SVM with probabilistic outputs, and (iii) a shallow Decision Tree (max depth 4). This ensemble benefits from complementary inductive biases and successfully surpasses the 0.80 accuracy barrier on the training set. However, its performance remains unstable across validation splits, suggesting that deeper nonlinear modeling is required to fully exploit the engineered feature space.

2.3. XGBoost (Final Model).

Our best-performing model is an XGBoost classifier trained with histogram-based tree construction. XGBoost builds an ensemble of decision trees using gradient boosting, allowing it to learn complex rules such as conditional combinations of stat advantages and battle momentum indicators. We performed an extensive RandomizedSearchCV over 40 configurations using 5-fold stratified cross-validation and AUC as the scoring metric. After identifying the best hyperparameter combination, we retrained the model on the full training data, applying early stopping with a patience of 80 rounds to prevent overfitting. The final model balanced predictive performance and generalization, and was ultimately used for the competition submission.