Pokémon Battle Win Rate Prediction

DATA1030 Final Report

Taemin Huh

https://github.com/taemin-huh/data1030-project/

1 Introduction

The Pokémon franchise is one of the highest grossing media franchises of all time. The game series sold 440 million copies worldwide, and was adapted into various media forms such as TV series, movies, comics, and trading cards.[1] Competitive Pokémon battling became an established discipline with large-scale tournaments, circuits, cash prizes and scholarships, and the next Pokémon to include in one's team has often been a popular discussion among communities.

Pokémon battles are turn-based, and each Pokémon can perform a move during its turn with the goal of reducing the opponent's Health Points (HP) to 0. Attack, Defense, Special Attack and Special Defense stats are involved in damage calculation, while Speed generally determines which Pokémon attacks first each turn. There are 18 different elemental types that interact with one another, and all Pokémons have at least one and at most two types.

This project attempts to utilize machine learning tools to predict the win rate (0-1) of a given Pokémon. To solve this regression problem, it uses two datasets published by the Weedle's Cave project on Kaggle[2]: *Pokémon* and *Combat*. *Pokémon* dataset contains 12 features of 800 Pokémons collected from the actual game, including their unique ID, name, type 1, type 2, six base stats (HP, Attack, Defense, Special Attack, Special Defense and Speed), game generation (1^{st}-6^{th}), and whether they are Legendary. *Combat* dataset contains datapoints of 50,000 Pokémon battles generated through a custom algorithm across three columns: ID's of (1) Pokémons that attacked first, (2) those that attacked after, and (3) the winners. It was further feature engineered for this project by counting the total number of wins and total number of battles for each Pokémon, then dividing the two (not included as features) to arrive at the target variable: win rate.

There are several public projects based on these datasets. J. Bouchet trained a model to predict battle outcomes based on stat differences of winning and losing Pokémons, and scored an average of 91% with 5-fold cross-validation.[3] V. Liao also created a model to predict battle outcomes, and demonstrated a peak accuracy of 96% using the random forest model.[4]

2 Exploratory Data Analysis

As shown below in Figure 1, the target variable win rate is rather evenly and symmetrically distributed with a mean value of 0.50 and median value of 0.49.

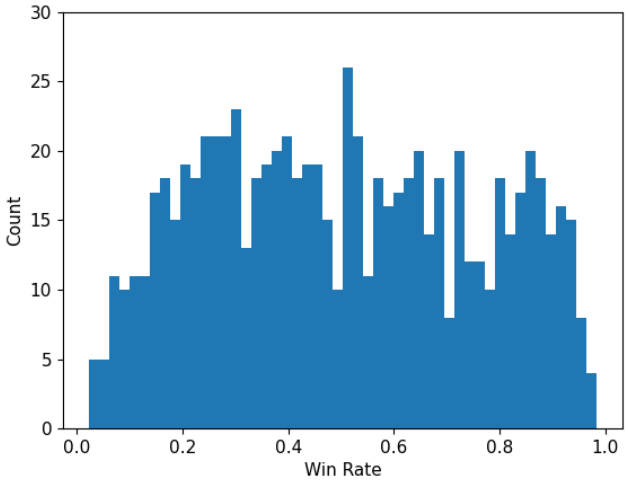

Figure 1. Histogram: win rate distribution

As illustrated by Figure 2, Speed is the feature with the strongest correlation to win rate with a 0.94 Pearson correlation coefficient, contrary to the initial expectation that Attack and Special Attack would be. This makes sense, as being able to consistently hit first each turn often translates to an extra turn before the opponent faints when exchanging blows, or a single-hit KO on the first turn.

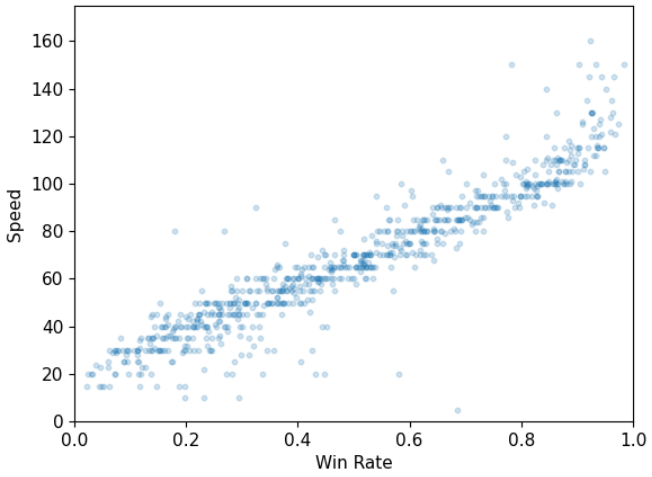

Figure 2. Scatter plot: speed stat vs. win rate

As shown in Figure 3, there is no meaningful power inflation of Pokémons over the release of newer games, with win rates evenly distributed across the six generations. This illustrates that the series have maintained the balance of Pokémons relatively well since its first game.

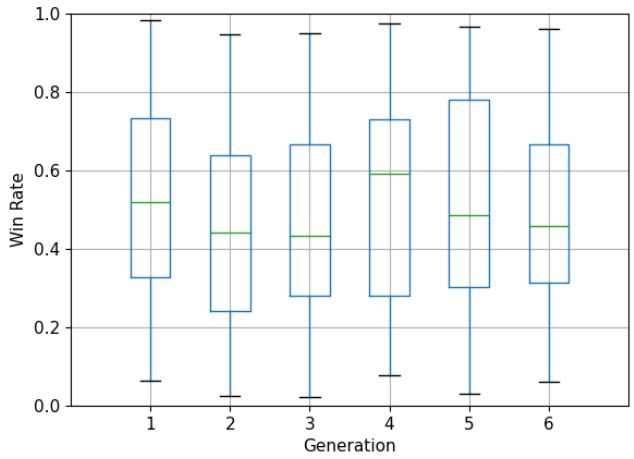

Figure 3. Box plot: win rate of Pokémons from each generation

Lastly, Figure 4 shows that Legendary Pokémons are more likely to show higher win rate (clustered in range above 0.5) than non-Legendary ones. This could be because they are a unique class of Pokémons that represent each game title and tend to have above-average base stats.

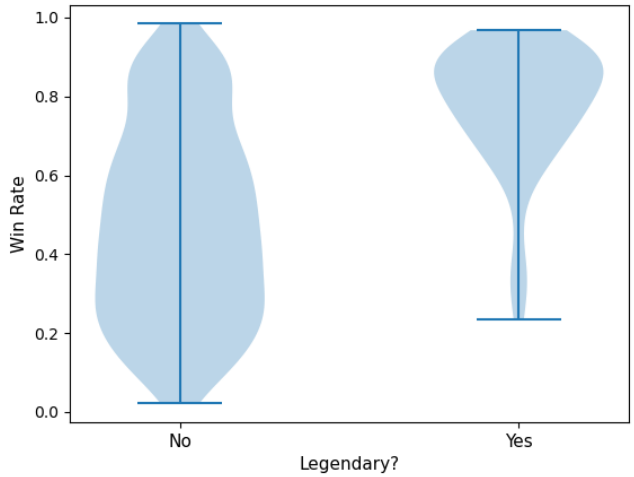

Figure 4. Violin plot: win rate of Legendary vs. non-Legendary Pokémons

3 Methods

A basic split of 60%/20%/20% was used across train/test/validation sets, as the data is independently and identically distributed without any group structure or time-series property. Given the large number of datapoints (800 in *Pokémon*, 50,000 in *Combat*), k-fold split was not deployed as it is more computationally heavy than basic split while achieving similar results when datasets are sufficiently large.

OneHotEncoder was used for the four categorical features (type 1, type 2, generation and Legendary) as none of these are ordinal or have a clear hierarchy. Type 1 is encoded into 18 features since 18 types exist, and type 2 into 19 features since not having a secondary type should be regarded as a feature on its own. Generation is encoded into six features given there are six generations, and Legendary into two given it is a Boolean.

MinMaxScaler was used for all other continuous features (HP, Attack, Defense, Special Attack, Special Defense, Speed) as the six base stats have a hard range limit of 0 to 255 – 255 is the maximum value representable by an eight-digit binary number, which was the hardware limitation of the first Pokémon games and a convention kept until today. As a result, there are a total of 51 features after preprocessing: original 14 from X_train, less unique ID and name, less total win and battle count created (directly involved in calculating win rate), plus the extra 17+18+5+1 from OneHotEncoder.

The ML pipeline developed includes five types of models: two linear (multiple linear regression and SVM) and three non-linear (decision tree, random forest and XGBoost). For each model, key hyperparameters were tuned using GridSearchCV with 5-fold cross-validation, calculating the optimal combination based on mean absolute error. For linear regression, we tuned the intercept parameter (true/**false**) to determine whether to include the intercept term in the model; for SVM, kernel type (**linear**/radial basis function), regularization parameter C (0.1, 1, **10**) and gamma (**scale**/auto) to balance margin maximization and influence of individual samples; for decision tree, max depth (**5**, 10, 20) to decide how deep the tree would grow, and minimum sample split (2, **10**) to ensure splitting occurs only with sufficient samples; for random forest, number of trees (50, **100**) and max depth (5, **10**) to balance complexity and predictive power; for XGBoost, number of boosting rounds (50, **100**), learning rate (0.01, **0.1**), max depth (3, **5**), subsample size (**0.8**, 0.9) and fraction of features per tree (0.8, **0.9**) to optimize performance, efficiency and sampling randomness. Bold letters indicate which values were chosen through this process.

Mean absolute error (MAE) was chosen as the evaluation metric due to its interpretability, simplicity and relevance. Since it measures the average magnitude of prediction errors, it is intuitive for end-users to understand that an MAE of 0.05 simply represents a 5% average deviation from the actual win

rate, which is also in percentages. Moreover, in context of competitive Pokémon battles, smaller deviations (e.g. predicting 0.3 instead of 0.32) are less important than large deviations (predicting 0.3 instead of 0.7), making MAE an appropriate choice that treats all errors equally. For example, mean squared error (MSE) would square the error terms and disproportionately penalize occasional outliers (e.g. unusually strong/weak Pokémon), while MAE directly reflects the average size of such deviations to ensure that the model is tuned to provide predictions closer to true values. Thus, adopting MAE ensures a more balanced and interpretable assessment in real-world terms, making it suitable as our primary evaluation metric.

Repeated 5-fold CV with 10 repeats were used on all models to calculate uncertainties introduced by different splitting methods. The data was split into 5 folds, with each fold serving as the test set while others as train sets. This was repeated 10 times with varying random splits, and the standard deviation of MAE across all runs were measured to assess how the performance varies due to randomness in sampling. For models with inherent randomness by design such as random forest and XGBoost, 10 repeated instances were run with varying random seeds to calculate the standard deviation of MAE across the 10 runs, representing non-deterministic uncertainty involved. Splitting uncertainty ranged from ~0.00398 to ~0.00487, while non-deterministic uncertainty ranged from ~0.00025 to ~0.00028, showing that our model predictions are quite stable with data splitting method having slightly more influence on performance than inherent randomness of models.

4 Results

Figure 5 outlines the performance results of the 5 models in our pipeline. As can be seen, XGBoost demonstrated the most predictive performance with the lowest MAE of ~0.03876. This is ~44.98959 standard deviations above the baseline model's MAE of ~0.21740, representing a significant improvement in performance – baseline MAE was calculated through a naïve model using the mean of the training set targets at ~0.49942 for all predictions. The splitting uncertainty for XGBoost is also the smallest among all models at ~0.00398, with a relatively small non-deterministic uncertainty of ~0.00028, meaning it shows low variability across different splits and performs consistently even accounting for inherent randomness of the algorithm. As shown in figures 6 and 7, the model's predictions are quite close to the actual target values, indicating robust performance.

Model	Test MAE	Std Dev from Baseline	Splitting Uncertainty	Non-Deterministic Uncertainty
LinearRegression	0.048137	41.005280	0.004128	NaN
SVM	0.052230	33.929851	0.004868	NaN
DecisionTree	0.046020	39.797189	0.004306	NaN
RandomForest	0.040605	43.177033	0.004095	0.000253
XGBoost	0.038758	44.898587	0.003979	0.000278

Figure 5. Summary table: performance comparison across models

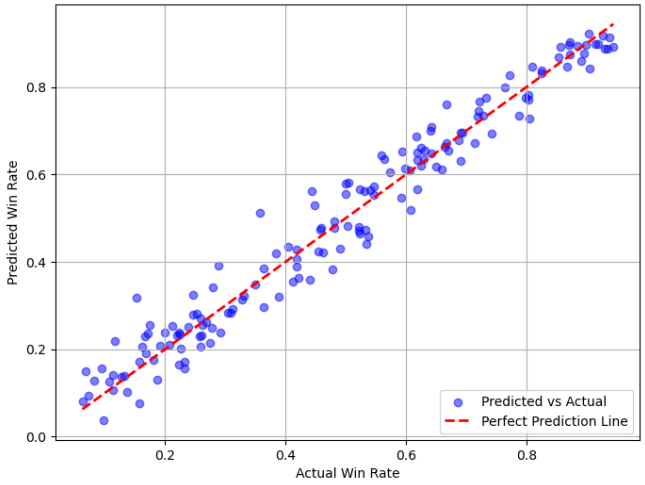

Figure 6. Scatter plot: predicted vs. actual win rate

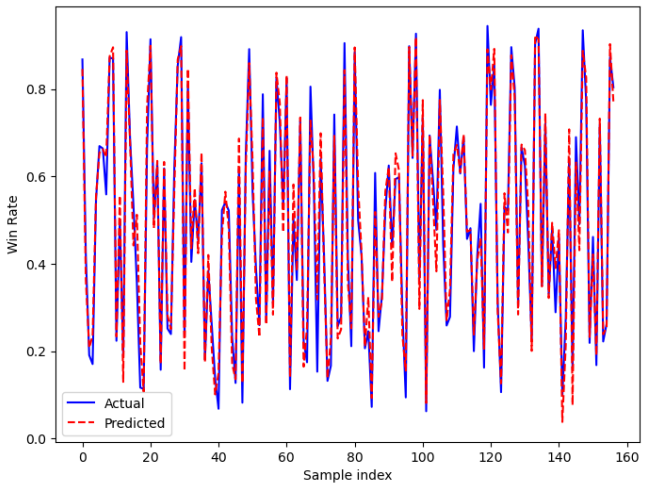

Figure 7. Line plot: predicted vs. actual win rate

Global feature importances were calculated by model type. For tree-based models (i.e. decision tree, random forest and XGBoost), the models' built-in Gini importance attribute was used to measure mean decrease in impurity, while for linear models, permutation importance was used to measure decrease in performance after feature values were randomly shuffled. Figure 8 captures the top 3 features in terms of global feature importance for each model – interestingly, feature 50 (Speed) ranks as the most important feature across all models by a wide margin, which is consistent with our preliminary findings during EDA. Feature 46 (Attack) is also notably important by ranking 2nd in 4 out of 5 models, also aligning with our EDA findings. There was no feature

that consistently ranked as the least important, as the variation in importance is miniscule in bottom ranks due to concentration of importance in higher ranks. These findings are also consistent with results in global SHAP summary plots in figure 9.

LinearRegression Global Feature Importance:

	Feature	Importance
50	Feature 50	2.184277e-01
46	Feature 46	2.034551e-02
43	Feature 43	7.043117e-03

SVM Global Feature Importance:

	Feature	Importance
50	Feature 50	2.088433e-01
46	Feature 46	2.335917e-02
10	Feature 10	2.566417e-03

DecisionTree Global Feature Importance:

	Feature	Importance
50	Feature 50	0.955395
46	Feature 46	0.042320
26	Feature 26	0.001248

RandomForest Global Feature Importance:

	Feature	Importance
50	Feature 50	0.922071
46	Feature 46	0.037326
45	Feature 45	0.010574

XGBoost Global Feature Importance:

	Feature	Importance
50	Feature 50	0.424124
43	Feature 43	0.111956
25	Feature 25	0.073364

Figure 8. Table: top 3 global feature importances for all models

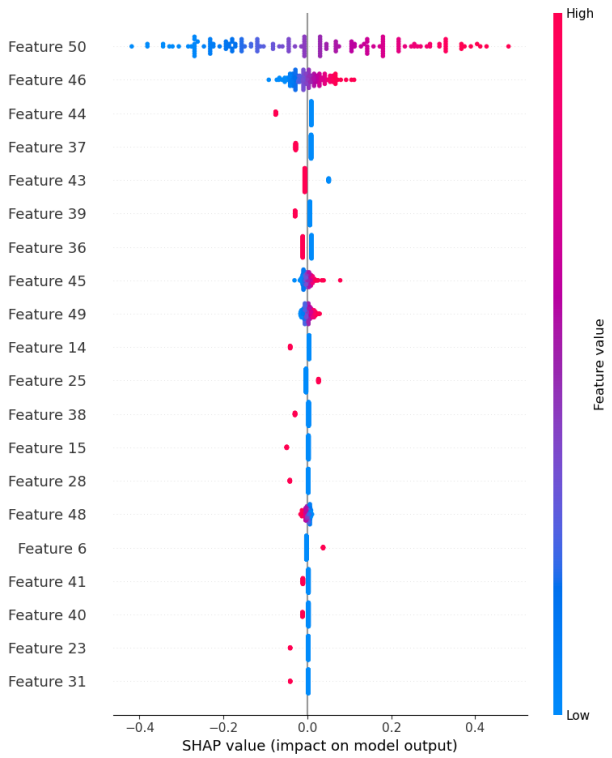

Figure 9a. Global SHAP summary plot: linear regression

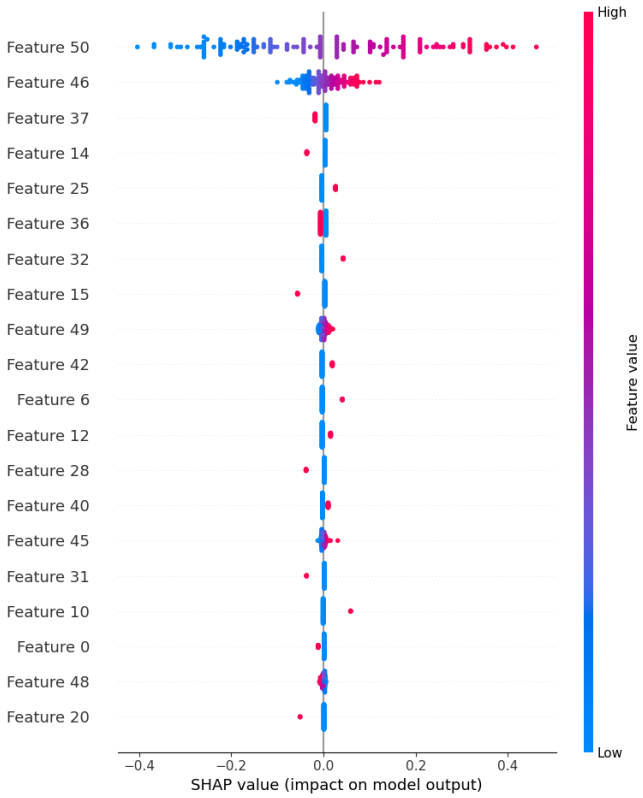

Figure 9b. Global SHAP summary plot: SVM

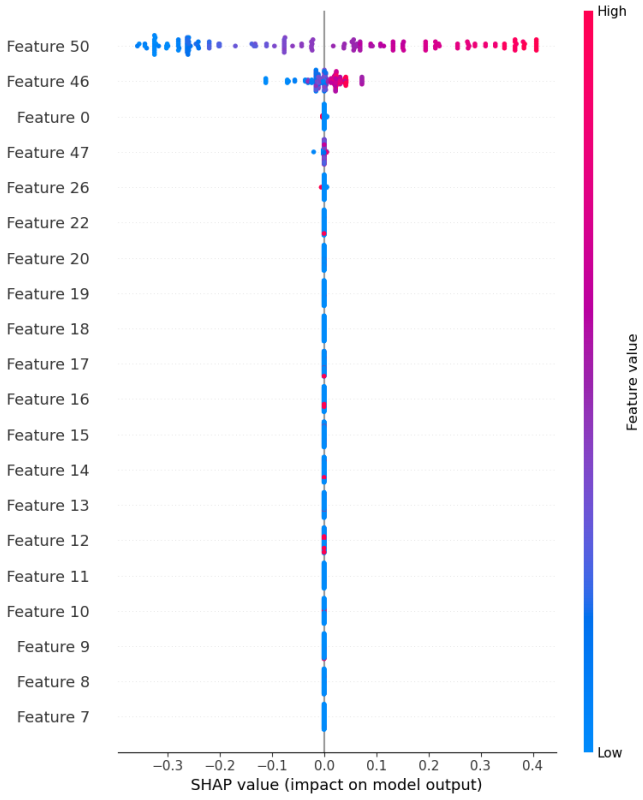

Figure 9c. Global SHAP summary plot: decision tree

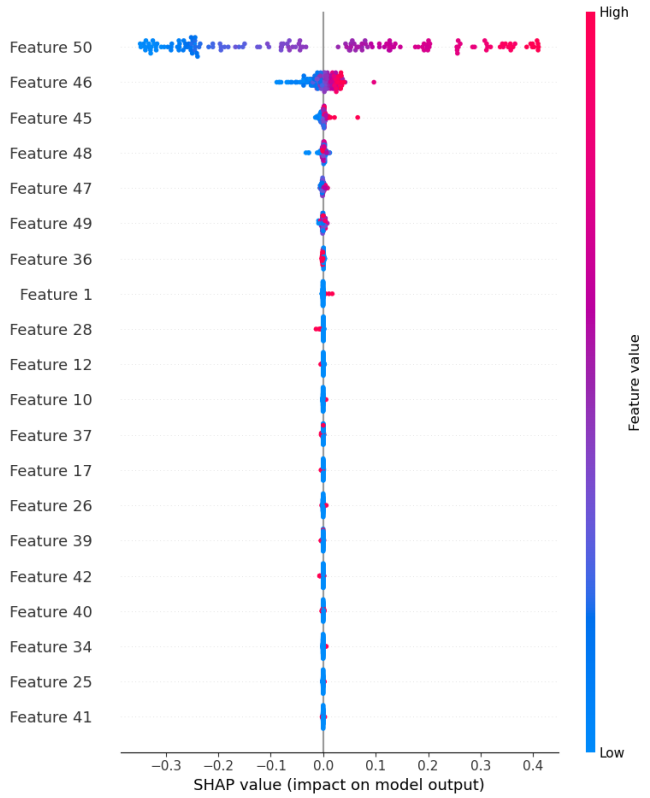

Figure 9d. Global SHAP summary plot: random forest

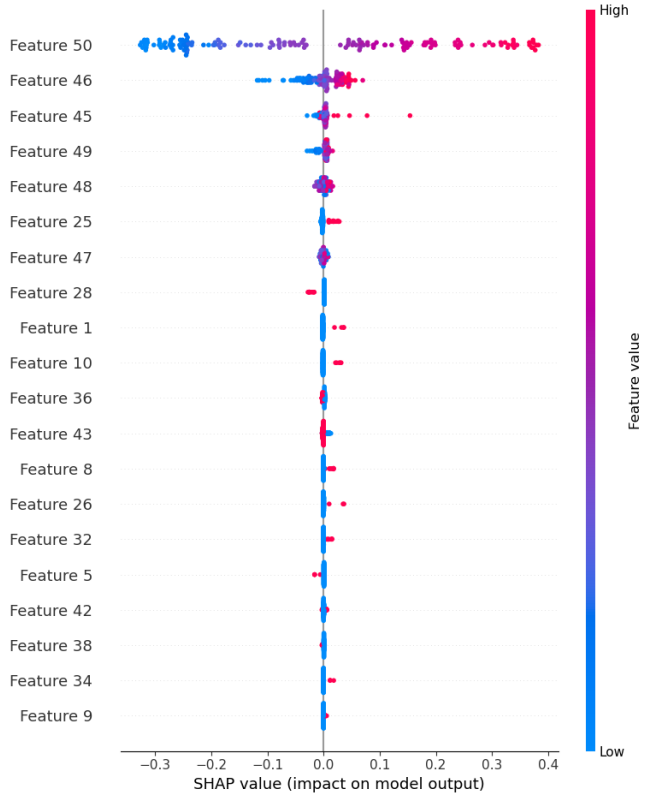

Figure 9d. Global SHAP summary plot: XGBoost

Local feature importances were calculated through SHAP values of two specific instances: a typical instance (index 11) and an edge-case instance (index 119). Former was selected by identifying a representative case with closest prediction to the median prediction, while the latter by selecting one with the largest prediction (argmax) to represent outliers, for balanced coverage. As can be seen from the SHAP force plots in figure 10, Speed still consistently ranks as the most locally important feature with the longest width in every plot, with Attack making frequent appearances in the top list. Thus, it can be concluded that the Speed stat is arguably the most important feature that affects Pokémon battle win rates both locally and globally, with Attack also playing a meaningful role.

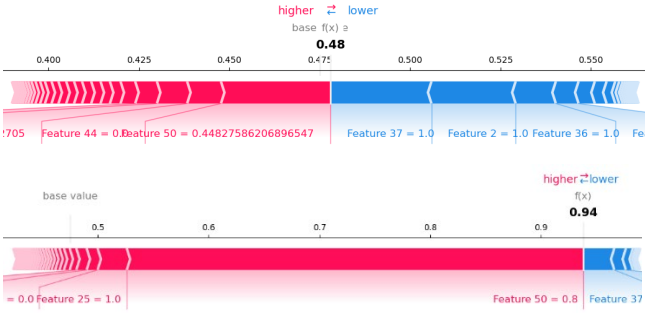

Figure 10a. Local SHAP force plot: linear regression typical case (upper) vs. edge case (lower)

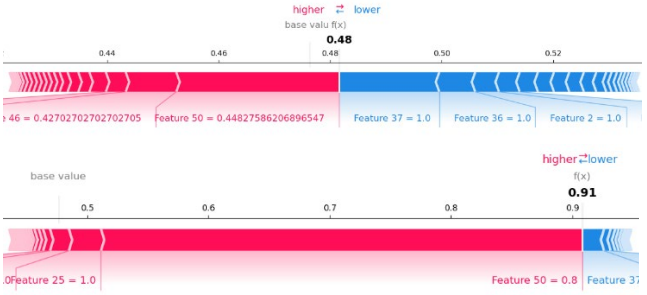

Figure 10b. Local SHAP force plot: SVM

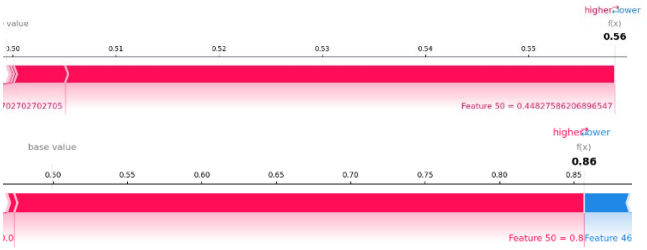

Figure 10c. Local SHAP force plot: decision tree

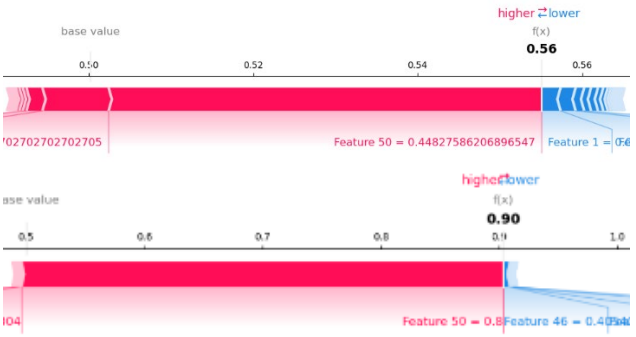

Figure 10d. Local SHAP force plot: random forest

Figure 10e. Local SHAP force plot: XGBoost

5 Outlook

The project could be improved by sourcing additional game data, as there are other significant factors such as 25 unique Pokémon natures, level, held items, abilities, move sets and field effects that affect the outcome of a battle. Incorporating these would enable a more extensive analysis and better-informed prediction for win rates. Moreover, data on each Pokémon's competitive play pick rate would greatly enhance our analysis – pick rate is one of the two most important measures (along with win rate) to assess competitive viability in many gaming statistics platforms such as OP.GG[5], as popular picks usually indicate meta-relevant strategies that perform well in the current environment. Lastly, the analysis could benefit from combining strengths of multiple models through stacking to improve prediction accuracy, creating partial dependence plots to show how a predicted outcome changes as a single feature changes, or even implementing neural network models when given enough data.

6 References

[1] The Pokémon Company. (2022, March). Pokémon in Figures. Retrieved October 21, 2022, from https://corporate.Pokémon.co.jp/en/aboutus/figures/

[2] Weedle's Cave. (2017). Pokémon- Weedle's Cave. Retrieved October 21, 2022, from https://www.kaggle.com/datasets/terminus7/Pokémon-challenge

[3] Bouchet, J. (2017, September 27). Pokémon Battles. Retrieved October 21, 2022, from https://www.kaggle.com/code/jonathanbouchet/Pokémon-battles

[4] Liao, V. (2017, December 18). Visualizing Data and Predicting Pokémon Fights. Retrieved October 21, 2022, from https://www.kaggle.com/code/vforvince1/visualizing-data-and-predicting-Pokémon-fights/notebook

[5] OP.GG. (2024). OP.GG: Champion Statistics and Analytics. Retrieved September 17, 2024, from https://www.op.gg/