Predicting Combat Outcomes between Pokémon using Various Machine Learning Models

Surinder Bath
College of Letters and Sciences: Cognitive Science
University of California Davis
Davis, CA, USA
sbath@ucdavis.edu

Jeanette Simo
College of Letters and Sciences: Applied Physics
University of California Davis
Davis, CA, USA
jgsimo@ucdavis.edu

Wenqian Huang
College of Engineering: Computer Science
University of California Davis
Davis, CA, USA
jywhuang@ucdavis.edu

Helen Yenson
College of Letters and Sciences: Computer Science
University of California Davis
Davis, CA, USA
hkyenson@ucdavis.edu

Yasmin Moule
College of Letters and Sciences: Cognitive Science and Applied Math
University of California Davis
Davis, CA, USA
ysmoule@ucdavis.edu

***Abstract*—Over the years, Pokémon as a franchise has been largely developing and gaining popularity through card games, TV shows, and recently Pokémon GO, a mobile app that allows the user to engage in Pokémon battles. Because the rule that the Pokémon game itself uses to determine the victor in combat is not publicly available, predicting the outcome of Pokémon battles has gained attention as a machine learning problem due to the complexity and variety of Pokémon attributes, stats, and interactions between Pokémon. Traditional models that rely on individually thought-up formulas or statistical comparisons can fall short when dealing with nuanced statistics and interactions between Pokémon during battle. Comparatively, in this paper, we implemented three machine learning models: a logistic regression model, an support vector classifier model, and a multi-layer perceptron model, to accurately predict battle outcomes, comparing their performance accuracy. Of the three models, our multi-layer perceptron model performed the best, with an accuracy score of 95.41%, a precision score of 95.72%, recall score of 94.57%, and an F1 score of 95.14%. This work conveys the significant potential of machine learning, particularly with deep learning approaches like the multi-layer perceptron model, in accurately modeling and predicting outcomes in complex, rule-based systems like Pokémon games, and could prove useful in the future for the field of esports game analytics and competitive strategy.**

***Index Terms*—machine learning, Pokémon combat outcomes, multilayer perceptron, logistic regression, support vector classifier**

I. Introduction

The application of machine learning in gaming environments has been growing over recent years due to its ability to model complex decision-making processes and generate useful insights. Although formula-based models and heuristic approaches have been previously used to predict battle outcomes, they often fall short when dealing with nuanced stat differences or less obvious Pokémon-type advantages and interactions. Additionally, as the number of available Pokémon and their associated attributes/types is ever increasing, static systems are becoming increasingly inefficient and incapable of meeting the expectations of users.

To address these issues, we propose a machine learning pipeline based on a challenge proposed on Kaggle: to create a machine learning model to predict the outcome of a Pokémon battle between two given Pokémon. We utilized two public datasets from Kaggle: (1) a dataset containing outcomes of battles between Pokémon with Pokedex IDs up to 800 and (2) a comprehensive dataset including the full base datasets(e.g. HP, Attack, Defense, Special Attack, Special Defense) and types for the Pokémon mentioned in the combat dataset. Using these datasets, we had access to a training set suitable for our machine learning models. The major contributions of this work include:

- Feature selection and an exploratory data analysis (EDA) process to identify relationships between features.
- Development of a logistic regression model, SVM model, and multi-layer perceptron model to compare performance.
- Evaluation of the aforementioned models to determine the most accurate model for predicting battle outcomes.
- Creating a demo website for users to try out our model.

II. Literature Review

A. Combat Outcome Prediction Using Player Information and Game Data

A study we reviewed by Vu, Ruta, Cen, and Liu covers a topic similar to ours, video game win predictions. It analyzes many ways game prediction classifiers predict victories in tactical games, slightly more complex compared to Pokémon battles. Initially, the researchers of the study had planned to use logistic regression as one of their models, as they similarly were dealing with a binary classification problem. However, they found that as they added more features to their models, there was an increasing gap in performance compared to other models they had implemented (including XGBoost, Neural Networks, LightGBM and Catboost, of which LightGBM performed the best). This led them to drop their logistic regression model from their research [1]. Although we still used logistic regression for our models, this led us to be cautious with the results and our expectations for this model.

B. Sports Result Predictions

Another research paper by Bunker and Thabtah we found covered the use of machine learning models to predict sport results. It outlined the process of using pregame statistics as features to predict victories in sports games, including historical matches, personal players' performance indicators, and information on the opposition. Although traditionally in sports, mathematical and statistical models are used to predict the outcome of matches, this research created a neural network model to make predictions using the previously outlined features. To evaluate the accuracy of their proposed model, they compared home wins, away wins, and draws and the number of matches in which the model had made accurate predictions using a standard classification matrix [2]. Though this research is different from ours in that it attempts to make predictions of real-life matches, which is more susceptible to a variety of factors compared to Pokémon's predefined win formulas, it provided useful information for how to explore and evaluate our models based on the limited number of features extracted.

III. Dataset Description

The dataset that we utilized contains two CSV files, one with descriptions of 800 Pokémon and their stats (Pokémon .csv) and the other CSV contains data about battles between two Pokémon based on their Pokedex IDs and the outcome of the combat (combats.csv). The first CSV contains data including the listed Pokémon 's ID, name, types, HP, attack, defense, special attack, special defense, speed, generation, and legendary status. These features are important because attack, defense, and other stats can have a large impact on the outcome of combat. There are 18 types of Pokémon in this dataset: Grass, Fire, Water, Bug, Normal, Poison, Electric, Ground, Fairy, Fighting, Psychic, Rock, Ghost, Ice, Dragon, Dark, Steel, and Flying. Pokémon types can have a large influence on the outcome of battles because types can interact with each other in different ways, with some types stronger / weaker than others. For example, in general, a Water type Pokémon will beat a Fire type Pokémon . The Pokedex IDs identify the Pokémon in the second CSV. Lastly, legendary status is important as it can point out potentially overpowered Pokémon that can lead to unexpected battle outcomes. The second CSV contains battles between the 800 Pokémon listed in Pokémon .csv, in which the outcome of the battle is decided based on a custom algorithm that was formulated by the creator of the dataset. The algorithm is a simplified version of the algorithm used in Pokémon Go. This CSV contains the values we want to predict.

IV. Exploratory Data Analysis

Before beginning with our models, we first analyzed the dataset to find relations between features that may have more influence on combat outcomes.

Fig. 1 shows the stat distribution per Pokémon numerical feature with normalization. The data is standardized and the y-axis reflects the different standardized values of each of the features, with the x-axis showing each of the different numerical features present in our dataset, including HP, Attack, Defense, Special Attack, Special Defense, and Speed. Overall, we can see from the box plots that there aren't a significant amount of outliers per feature. HP and Defense have the smallest range (lower to upper whisker) while Special Defense and Speed have the largest range. Additionally, Defense has the largest number of outliers, 20, and HP has 19 outliers with two data points peaking over 6, the largest values in our data set. From these datasets, it is shown that most of the individual Pokémon 's statistics fall within a similar range for each of the features we observed.

Most Pokémon fans know that Legendary Pokémon can be extremely powerful compared to more common Pokémon . Because of this, initially we were concerned that having the Legendary attribute would significantly skew the outcome of a Pokémon battle in favor of the Legendary Pokémon so we compared the Attack, Defense, Special Attack, and Special Defense between Legendary and Non-legendary Pokémon . In Fig. 2a and 2b, it is clear that although Legendary Pokémon have above average attack and defense compared to normal Pokémon , there is still a significant amount of overlap in their statistics; they are not outliers. Thus, we still included the Legendary Pokémon in our dataset.

Fig. 3 compares correlation between features in our dataset. Type 1, Type 2, and Generation have no correlation to other features. There is a weak correlation between Special Attack and Legendary, suggesting that Special Attack is what makes Legendary Pokémon more powerful than normal Pokémon . There is also a weak correlation between Special Attack and Special Defense, Special Attack and Speed, Special Attack and Attack, Special Defense and Defense, and Attack and Defense. Since these are all fighting features, it is likely that stronger Pokémon have high values on all fighting features, while weaker Pokémon have low values on all fighting features. Since the correlation was weak, all features were used when training our models. Based on our heatmap, we did not find any specific features that have a particularly high correlation

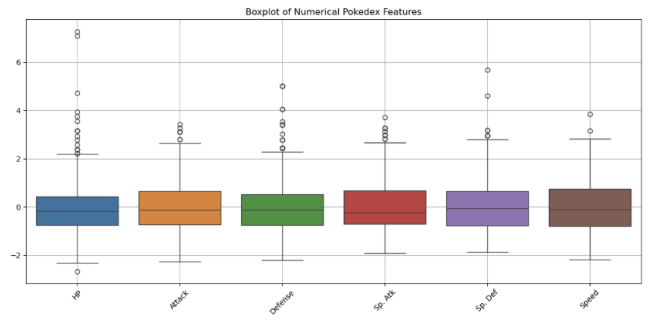

Fig. 1. Box plots for HP, Attack, Defense, Special Attack, Special Defense, and Speed

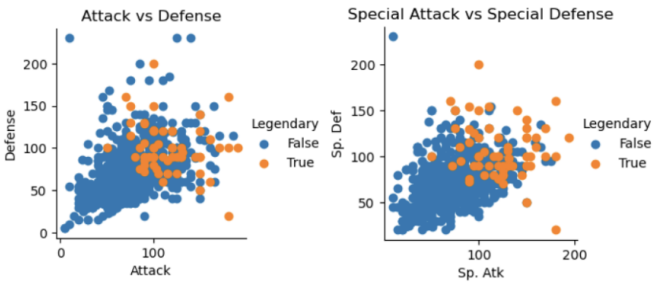

Fig. 2. (a) Attack vs Defense and (b) Special Attack vs Special Defense of Legendary and Non-legendary Pokémon

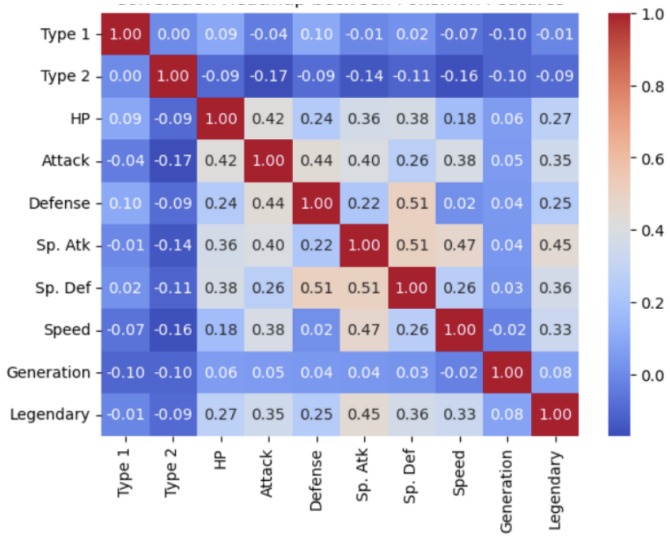

Fig. 3. Correlation Heatmap between Features

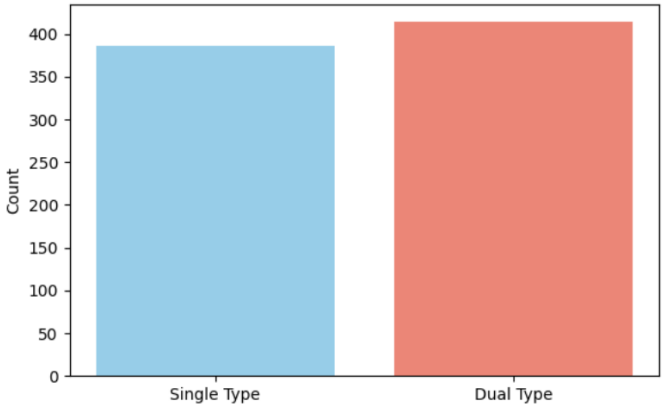

Fig. 4. Count of Single vs Dual Type Pokémon

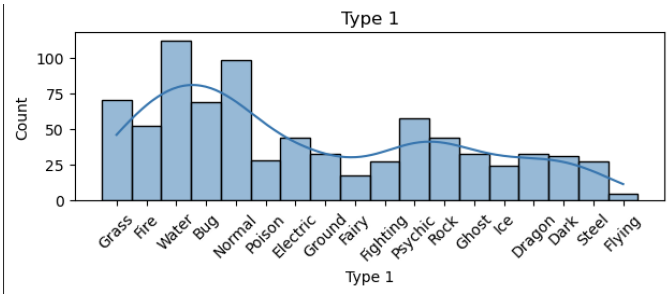

Fig. 5. Pokémon Counts for each Type 1

compared to others so we will be including all of them for our model, besides legendary category Pokémon . It seems that all the features interact to decide the outcome of a combat. In Fig. 3 above, we encoded Type 1 and 2 using Integer Encoding for simplicity. To avoid error from the accidental ordering of unordered data caused by Integer Encoding, we used one hot encoding for type when training our models. As shown in Fig. 4, a significant portion of our dataset was dual type so it was important to accommodate for this in our models, as Pokémon type interactions can also have a significant impact on combat outcomes. To account for Pokémon with two types, we could encode Type 1 and Type 2 separately and then combine them into vectors. For Pokémon with only one type, we can represent missing the Type 2 as a “None” before one-hot encoding. Since there are 18 types, this caused the dimensionality of our data to increase significantly.

Additionally, after comparing the counts for single vs dual type Pokémon , we compared the counts for each different Type of Pokémon separating between Type 1 and Type 2 Pokémon as shown in Fig. 5 and Fig. 6. From those graphs we found that Water and Normal Type Pokémon were more highly represented in out dataset compared to other Type 1 Pokémon , and Flying is the most common Type 2 in our dataset. It is important to note that some Pokémon types are clearly underrepresented in the dataset compared to others, which could lead to the model lacking accuracy when dealing with combats that have those types, like Fairy or Ice types.

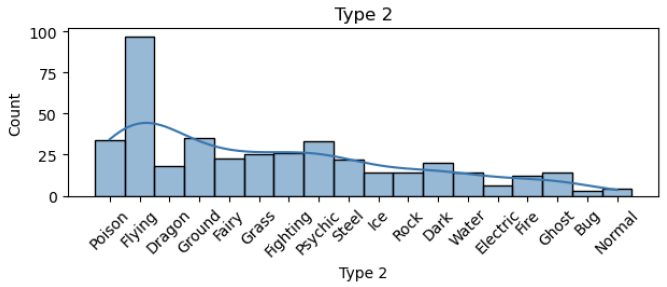

Fig. 6. Pokémon Counts for each Type 2

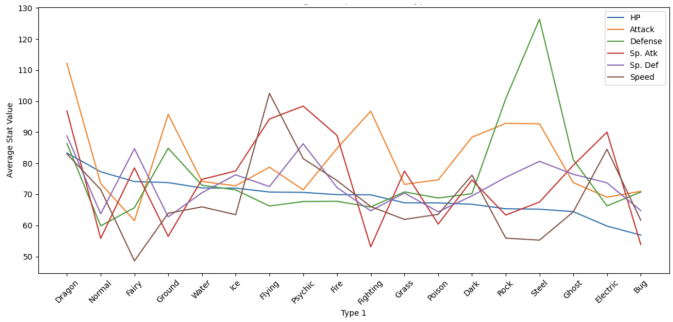

Fig. 7. Average Stats per Type

Furthermore, we noted down the number of wins between Pokémon with obvious Type advantage interactions. In Fire Type vs Water Type battles, Fire wins 544 times while water wins 387 times. This is odd as usually Water types should have the advantage in these combats. It may be due to other factors such as secondary types or overall combat stats. In Electric vs Ground interactions, Electric types won 21 times, with Ground types winning 197 times. This is inline with our expectations as Ground types have a Type advantage here. Lastly, when comparing Grass type vs Water type combats, Grass types win 602 times, while Water types win 606 times. This shows the dataset to be inconclusive for this matchup, but in the Pokémon universe Grass types should beat Water types.

Lastly, we observed average statistics per Types of Pokémon , to review any advantages of certain types that may give them an edge over their opponents. Fig. 7 shows the average statistics such as Attack, Defense, etc. per Pokémon Type 1. From the graph, we observed that Pokémon of Steel Type on average had a significantly higher Defense stats with lower than average Speed stats. Comparatively, Pokémon of Dragon Type had the highest average Attack overall compared to the other listed Pokémon types. Overall, this graph shows how different types of Pokémon can have different strengths and weaknesses which can all influence combat outcomes.

V. Proposed Methodology

A. Overview of Proposed Methodology

In this section, we present the methodology used for predicting the outcome of Pokémon battles using machine learning techniques. We utilized three different models, Support Vector Machine (SVM), Multilayer Perceptron (MLP), and logistic regression models. We chose these models to attempt to capture the complexities of the data, from possibly simpler linear relationships to more complex non-linear patterns. Each model was trained and evaluated separately, and their prediction accuracy was compared to identify the most accurate model for this task. The models predict the outcome of battles between two Pokémon based on their individual attributes (e.g. stats, type advantages, and legendary status). The detailed methodology of each model is described below.

B. Logistic Regression

We started with Logistic Regression because it is one of the most widely-used and interpretable models in binary classification tasks. It's also been a core part of our class discussions, especially in connection with Maximum Likelihood Estimation (MLE). Since we wanted a solid and explainable baseline to compare against, Logistic Regression was a natural first choice.

Logistic Regression works by applying a linear transformation to the input features and then using a sigmoid function to convert the output into a probability between 0 and 1. This makes it ideal for understanding how each feature contributes to the final prediction. It also allows us to interpret the importance of each feature by examining the learned coefficients.

Before training the model, we created input features by calculating the difference in base stats between the two Pokémon in each battle (first Pokémon minus second Pokémon). The features we selected include: HP, Attack, Defense, Special Attack, Special Defense, Speed, and a binary feature indicating whether the Pokémon is Legendary. These seven features capture a broad spectrum of strengths and weaknesses.

Because Logistic Regression is sensitive to feature magnitudes, we standardized all features using StandardScaler. We then split the data into training and test sets with an 80/20 ratio and trained the model using scikit-learn.

Logistic Loss Function

During training, the Logistic Regression model optimizes the logistic loss, also known as negative log-likelihood (NLL). This loss penalizes incorrect predictions more strongly when the model is confident but wrong. The objective is to assign high probabilities to the correct class while avoiding overconfidence in wrong predictions.

The logistic loss function is defined as:

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (1)$$

Here, $\hat{y}_i$ is the predicted probability for sample i, and y_i is the actual label (0 or 1). By minimizing this loss, the model learns parameters that best explain the observed outcomes under the assumption of a Bernoulli distribution.

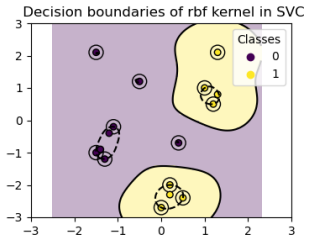

Fig. 8. SVC: RBF Kernel Example

C. Support Vector Classifier Model

A Support Vector Classifier (SVC) is a type of Support Vector Machine (SVM), which is a supervised learning algorithm. SVMs solve an optimization problem. In the linear case, SVMs minimize $||w||^2/2$ subject to $S \times (wx^T + \gamma I) \leq I$ [3]. Here, w are the model weights, x is the input, $S \in \{-1, 1\}$ in our case, depending on whether the point is above or below the boundary line, and γ is a bias term. In simpler terms, the SVC model maximizes the distance between the data and the boundary line. The model by design focuses mainly on points that fall close to the boundary line, rather than focusing on points that clearly are a part of a particular category.

When the data is not linearly separable, a "kernel trick" is used. A kernel translates the data into a different vector space, typically of higher dimension than the original space, in which the data is linearly separable [3]. Then, it applies the same algorithm as before. A common non-linear kernel is the radial basis function (RBF), also known as the Gaussian kernel, which measures similarity between two data points in infinite dimensions and approaches classification by majority vote [4]. Its kernel function is $K(x_1, x_2) = \exp(-\gamma * ||x_1 - x_2||^2)$, where the larger gamma is, the less influence each individual training sample has on the decision boundary [2]. In addition, the larger the Euclidean distance between x_1, x_2, the closer the kernel function is to zero, and the more dissimilar the points. A nice visual example of the RBF kernel can be seen on the scikit-learn page, shown in Fig. 8.

We chose to make a SVC model because the data has a very large number of features after One-Hot encoding the labels. In fact, including Type 2 Pokémon features, our input data has 86 different features. SVC models are effective in high dimensional spaces [5]. Thus, we wanted to see how well they would handle this classification task.

D. Multilayer Perceptron Model

Our second model we developed was a Multilayer Perception (MLP) model. MLP models are neural networks which are non-cyclic, where input values operate on an input layer, are fed through activation functions and weights on a hidden layer with many neurons, then lastly result in an output layer.

Mathematically, each neuron in an MLP network calculates output based on $wx = w_0 + w_1x_1 + w_2x_2 + ...$ or the sum of weights multiplied by input features. When feeding information through the hidden layer and output layer, the output of a neuron is also adjusted by an activation function; this allows the results through multiple neurons to handle non-linear data, rather than giving similar results compared to a single linear transformation. For example, an MLP network using a sigmoid activation function would calculate its output as $\hat{y} = \sigma(W_{jk}\sigma(W_{ij}X_i))$.

On a larger scale, classification-based MLP networks solve an optimization problem and minimize the loss with respect to a specific label: $E(w) = -\frac{1}{2}\sum_{i=1}^{n}[y_i - \hat{y}]^2$. The MLP network initially performs a forward pass of calculating the predicted output based on initial weights and input features. Then, the MLP performs a backward pass, as each weight is updated based on the derivative of the loss function.

MLP networks are more effective at processing non-linear data, and in conjunction with activation functions such as 'sigmoid' and 'logistic' are especially effective when applied to binary classification problems. The high amount of input processing and the low amount of layers lead to a model which can fit well to the data without risking overfitting or running into the vanishing gradient problem. Lastly, the model's use of a momentum term when undergoing training helps prevent converging to local minima and speeds up convergence time.

E. Handling the High Dimensional Data

Our different models used different methods for handling the highly dimensional data. First, the logistic regression model used the lowest-dimensional data, with an input vector size (1x7), since logistic regression does not perform as well with highly dimensional data. We did this by defining each Pokémon attribute (like HP) as a different between the attribute of the first pokémon and the second pokémon. Generation and Type information was also left out.

The second method involved an input vector of size (1x50). This included one hot encoding for Type 1 for each pokémon, then creating one variable for each attribute per pokémon (i.e. HP1, HP2). This method was used on our SVM model.

Finally, the highest dimensional method used an input vector of size (1x86). First, both Type 1 and Type 2 were encoded using one hot encoding, then one variable was created for each attribute for both pokémon like above. This method was used for MLP.

These methods don't just differ in dimensionality, they also have different focuses. The first method focuses on the differences in pokémon stats. This method suggests that the actual value of the stat does not matter, only the difference between stats. Since some type information is encoded in the stats as well, this model has some ability to handle different types despite discarding that information. The second method counts the stats of each pokémon separately, suggesting that the strength of a pokémon may increase non-linearly with respect to the actual value of the attribute. It ignores Type 2 information in favor of accuracy on Type 1 interactions. Finally, the third model includes all the information and has the potential to learn the most type interaction rules for pokémon. However, because of the high dimensionality, a lot of training data is necessary for the model to learn the rules.

The dataset may be too small for this method to be as effective as it could be given a larger dataset.

VI. Experimental Results and Evaluations

A. Logistic Regression

The model achieved the following results on the test set:

- Accuracy: 89.03
- F1 Score: 0.883
- Log Loss: 0.365

Confusion Matrix:

- True Negatives: 4750
- False Positives: 497
- False Negatives: 600
- True Positives: 4153

These results suggest that our model did a fairly good job distinguishing which Pokémon is likely to win. The F1 score is high, meaning there's a good balance between precision and recall. The log loss value indicates that the model's predicted probabilities are reasonably well-calibrated—neither too overconfident nor too vague.

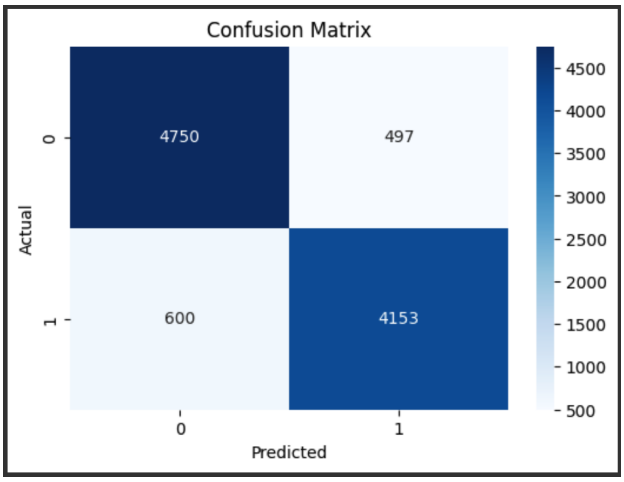

Fig. 9. Logistic Regression performance: confusion matrix and classification metrics.

We also analyzed the learned coefficients to understand which features had the most influence. As expected, Speed had the highest positive weight, which aligns with our intuition that faster Pokémon tend to attack first and often win. Attack was the second most important feature. Interestingly, the coefficient for Legendary status was small and slightly negative, showing that just being a Legendary Pokémon doesn't always lead to a win.

Overall, Logistic Regression served as a strong and interpretable baseline model. It allowed us to understand how individual features relate to the outcome and gave us valuable insights.

B. Support Vector Classifier Model

The model was created using the SVC model from Scikit-learn. Data was One-Hot Encoded, and the data from the two battling Pokémon was concatenated to create the input data set. The winners were encoded as 0 and 1, where 0 refers to the second Pokémon winning and 1 refers to the first Pokémon winning. SVC models were created with different kernels, regularization parameters, and kernel coefficients, and compared using the scikit-learn module Grid Search. Each model was improved until reaching a stopping tolerance. In our models, the default tolerance $1e-3$ was used.

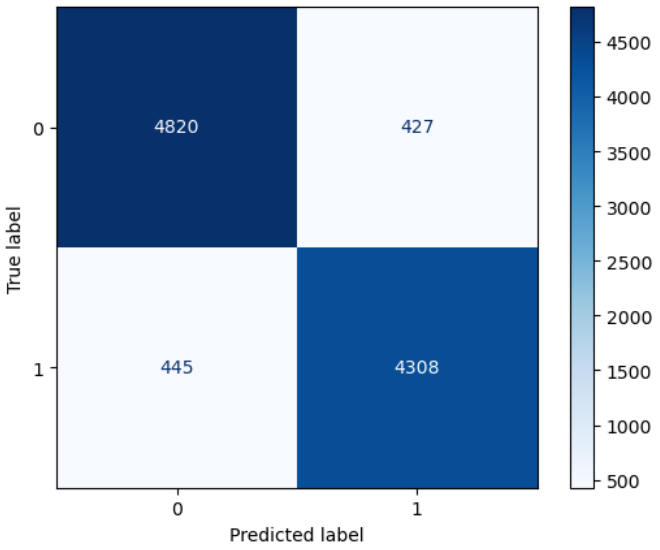

Fig. 10. SVM: Confusion Matrix

We also compared models that trained all features compared to models that trained on a subset of the features (leaving out Type 2 and Generation). Training on a subset of features reduced the dimensionality of the data significantly (changed from 86 input features to 50 input features). We found that, in general, training on a subset of features produced the best accuracy. The best model we created has an accuracy of 0.9128 and an F1 score of 0.9128. The confusion matrix can be seen in Fig 10 and is as follows:

- True Negatives: 4820
- False Positives: 427
- False Negatives: 445
- True Positives: 4308

Despite performing Grid Search, the best model we created uses the default parameters for the kernel 'rbf'. These parameters are as follows: C = 1.0 and gamma ='scale' (that is, $\gamma = \frac{1}{n\_features * Var(X)}$).

Interestingly, the best-performing model created using all the features has a linear kernel, which suggests that the features have a linear relationship in higher dimensions.

C. Multilayer Perceptron Model

Similar to the SVM model, the MLP model was developed using the MLPClassifier model from Scikit-learn. Data was first pre-processed, then the type of "solver" function was selected, then hyperparameter tuning was conducted across neuron count, activation function, and maximum iterations,

and finally the hyperparameter tuned model was compared to the standard default parameters.

For data pre-processing, the HP, Attack, Defense, Special Attack, and Special Defense parameters were all normalized via Min-Max normalization. Type 1 and Type 2 data was one-hot encoded, while Legendary status was converted to a binary 0/1 score. The output y was encoded as a binary 0/1 score as well, with 0 corresponding to the first Pokémon winning while 1 corresponded to the second Pokémon winning.

Scikit-learn's "solver" parameter indicated which Stochastic learning strategy was used during training. The default solver "adam" was a stochastic gradient based-learning method proposed by Kingma, Diederik, and Jimmy Ba and was recommended for all datasets [6]. This was supported by our experimental model processing, as the "adam" solver outperformed compared to the plain stochastic gradient descent "sgd" solver.

Hyperparameter tuning was conducted via Scikit-learn's GridSearch module. The MLPClassifier model by default initialized with a 'relu' activation function, 100 neurons in a single layer, and a maximum iteration count of 200. Grid search compared neuron counts of 50, 100, and 150 across a single layer, activation functions 'relu', 'logistic', and 'tanh';, and maximum iterations of 200 and 400. It found that a logistic activation function with 50 neurons in a layer and 200 maximum iterations was the most ideal, with an accuracy score of 0.934725.

This made sense as 'Logistic' activation functions are especially effective at binary classification. Lowering the amount of neurons and the maximum iterations also likely curbed the model overfitting. However, the hyperparameter tuned model was still slightly below the model initialized with standard parameters, which had an accuracy score of 0.9541.

The confusion matrix of our best MLP model was as follows:

- True Negatives: 5046
- False Positives: 201
- False Negatives: 258
- True Positives: 4495

D. Comparative Evaluations

In this section, we will give a comparative analysis of the performance of Logistic Regression, Support Vector Machine (SVC), and Multi-Layer Perceptron (MLP) models in predicting Pokémon combat outcomes. The models were evaluated based on four key metrics: Precision, Recall, F1 Score, and Accuracy.

1) Precision Score Analysis: Precision is defined as the ratio of true positive predictions to the total number of positive predictions (true positives and false positives combined). Mathematically, it can be shown as:

$$Precision = \frac{TP}{TP + FP}$$

where TP represents true positives (correctly predicted win outcomes) and FP represents false positives (incorrectly predicted win outcomes).

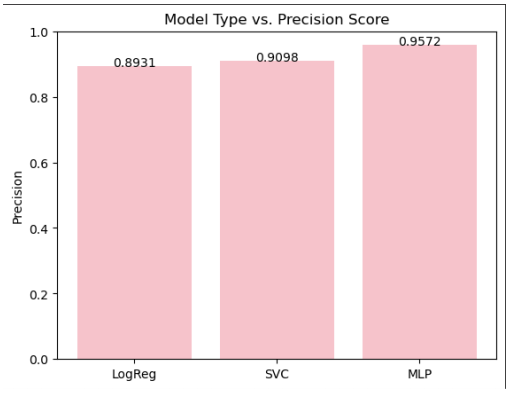

Fig. 11. Model Type vs. Precision Score

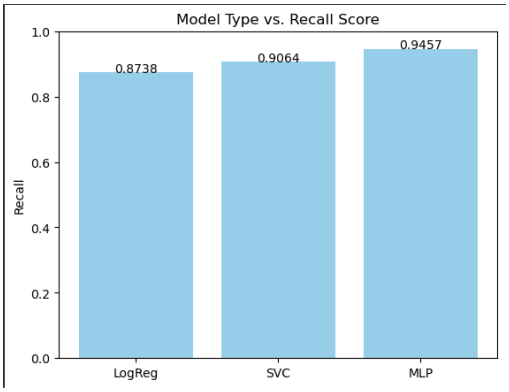

Fig. 12. Model Type vs. Recall Score

As shown in Fig. 11, the MLP model achieved the highest precision score of 0.9572, indicating its stronger ability to correctly identify positive outcomes (e.g. win predictions) with minimal false positive. SVC performed slightly worse with a precision of 0.9098, while the logistic regression model had the lowest precision at 0.8931. This suggests that MLP is particularly effective when minimizing false win predictions.

2) Recall Score Analysis: Recall, also known as sensitivity, measures the proportion of actual positive cases that are correctly predicted by the model. It is defined as the ratio of true positive predictions to the total number of actual positive instances. Mathematically, this can be shown as

$$Recall = \frac{TP}{TP + FN}$$

where TP again represents true positives and FN represents false negatives(actual win outcomes that were incorrectly predicted to be a loss).

Fig. 12 shows that the MLP model also had the highest recall score, with a score of 0.9457. This signifies that MLP has a strong ability in identifying a large proportion of the actual win combat outcomes, minimizing false negatives. SVC again performed well with a recall of 0.9064, and logistic regression had the lowest recall score of the three models with an 0.8738. The higher recall of MLP implies its robustness in capturing true win instances, ensuring that most of the actual

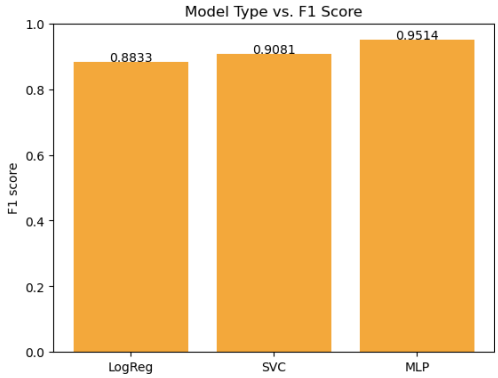

Fig. 13. Model Type vs. F1 Score

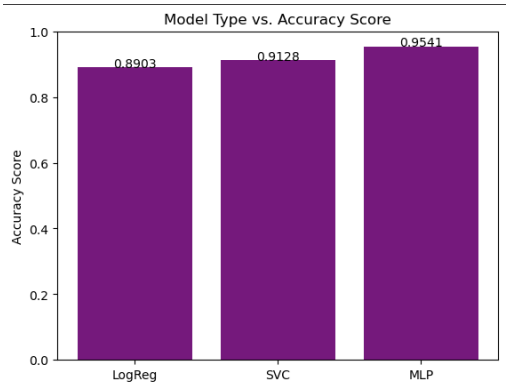

Fig. 14. Model Type vs. Accuracy Score

wins are actually predicted as wins, which is crucial for our model.

3) F1 Score Analysis: The F1 score provides a measure of a model's accuracy and is especially useful in the cases of imbalanced datasets. Mathematically, it can be shown as

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

From Fig. 13, MLP again was the top performer with an F1 score of 0.9514, suggesting it's consistent and balanced performance across both precision and recall. SVC had an F1 score of 0.9081, while logistic regression had an F1 score of 0.8833. The superior F1 score of MLP suggests that it has the strongest overall effectiveness in classifying Pokémon combat outcomes, as it achieves a strong balance between avoiding both false positives and false negatives.

4) Accuracy: The last evaluative measure we used was accuracy scoring. Accuracy is defined as the ratio of correct predictions to all predictions. Mathematically, the equation of accuracy is

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

where TP represents true positives, TN represents true negatives (correctly predicted losses) FP represents false positives, and FN represents false negatives.

Fig. 14 shows that the MLP model achieved the highest overall accuracy of 0.9541, signifying its ability to correctly predict a large proportion of both win and loss outcomes. SVC followed with an accuracy of 0.9128, and logistic regression had the lowest accuracy of the three models at 0.8903. The highest accuracy of MLP solidifies that it has the strongest predictive abilities of the three models.

VII. Conclusion

Across our evaluations, the MLP model consistently outperformed both the SVC and logistic regression models in predicting Pokémon combat outcomes. This suggests its superior predictive power for the task of predicting Pokémon combat outcomes. While SVC also performed well, MLP stands out as the most effective model among the three for this application. The strong performance of MLP can be attributed to its ability to learn complex non-linear relationships within the dataset, which may be prevalent in the complex dynamics of Pokémon combats. While in our model, we only considered unique base stats of each Pokémon and their elemental type interactions, there is much more complexity including environmental factors, individually held items, and more. The combined effects of these elements can cause combat outcomes to not be easily captured in linear boundaries, which explains why our logistic regression model didn't perform as well as the other two models. This underscores the necessity of non-linear modeling approaches like MLP to accurately capture the complex nature of not just Pokémon battles, but also other complex rule-based systems.

Our final model does many things right and some things wrong. First, we will discuss where it fails. The model fails when a pokémon goes up against an evolution of itself or against an evolution in general. It also fails for rare pokémon like Mewtwo. To us as players, these failures are painfully obvious. Mewtwo is well known as one of the strongest Pokémon in the game. People spend days trying to evolve their Pokémon so that it can get stronger. So in the end, where our model matters is not predicting these obvious outcomes, but in predicting the outcome of more evenly matched fights (like Pikachu vs Sandshrew), thus it has decent utility.

VIII. Discussion

Despite the strong performance of our MLP model, we acknowledge that our research had certain limitations. Our current dataset and model focused primarily on core combat stats and elemental types. A notable challenge that we encountered was the under-representation of Pokémon types within our dataset. Although the underrepresented types are less common in the Pokémon world, this class imbalance could have led our model to not accurately generalize or make accurate predictions in combats involving those specific types. Furthermore, other key influences like status conditions(e.g. burn, poison, paralysis), and player-specific strategies were not integrated, which leads our model to be a much simpler

simulation of combats compared to the full complexity of actual Pokémon game combats.

Further work could address these limitations by building on our current study, and addressing the data imbalance by sampling more from underrepresented types. Integrating more dynamic features like status effects would also be an interesting addition to our current research.

This work also illustrates a common problem with building Pokémon related models, there is not much data available. Our dataset itself is an approximation of the true battle outcomes. Then, we use our machine learning model to approximate our dataset. As a result, our results are twice-removed from the ground truth. Due to the lack of available data, there is no improving upon this fact. Most of the errors our model generates come from the data itself not being fully accurate to the Pokémon world.

Finally, there are lots of significant implications of our research. Firstly, for the competitive Pokémon gaming community, a highly accurate predictive model would serve as powerful tool for team optimization and game strategy. It could allow player to build more effective teams, simulate battle outcomes, and simulate game play. Secondly, our research contributes to the field of game AI and data analytics, demonstrating the strong capabilities that AI has in understanding and predicting outcomes in more complex games. It highlights that even in a seemingly simple game like Pokémon, the underlying mechanics can require sophisticated machine learning models for accurate prediction. Lastly, our research highlights a general failure in machine learning models, the failure to perform well for the minority. While for our model, the minority was Fairy type, in the real world, not performing well for the minority can have serious consequences.

IX. Github Link and Demo Website Link

https://github.com/JeanetteGS/ECS171FinalProject
https://pokemonwebsite-rso8.onrender.com/
https://www.youtube.com/watch?v=yVun3zFHvI8

Our website generally takes about 5 minutes to load if starting up.

X. Project Roadmap and Team Contributions

We followed the ML pipeline outlined in lecture, starting with collecting and cleaning the dataset, then performing exploratory data analysis (EDA) to understand relationships in the data. After that, we deliberated on which models were best suited for the task, ultimately deciding on logistic regression, SVC, and MLP models. Wenqian worked on the logistic regression model, Jeanette and Yasmin worked on the SVC model, and Helen worked on the MLP model. After working on the models, Jeanette did the comparative evaluations. Surinder worked on the final project report. Yasmin also worked on the demo website. Our timeline was as follows: data cleaning/analysis (1 week), developing, testing and training our models (2 weeks), and creating demos and writing the final report (3 weeks).

References

[1] Q. H. Vu, D. Ruta, L. Cen and M. Liu, "A combination of general and specific models to predict victories in video games," 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 5683-5690, doi: 10.1109/BigData52589.2021.9671285.

[2] Rory P. Bunker, Fadi Thabtah, A machine learning framework for sport result prediction, Applied Computing and Informatics, Volume 15, Issue 1, 2019, Pages 27-33, ISSN 2210-8327, https://doi.org/10.1016/j.aci.2017.09.005

[3] Suthaharan, S. (2016). Support Vector Machine. In: Machine Learning Models and Algorithms for Big Data Classification. Integrated Series in Information Systems, vol 36. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7641-3_9

[4] scitkit learn. (n.d.). Plot classification boundaries with different SVM kernels. scikit. https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

[5] scikit learn. (n.d.-a). 1.4. Support Vector Machines. scikit. https://scikit-learn.org/stable/modules/svm.html

[6] scikit learn. (n.d.). Compare Stochastic learning strategies for MLPClassifier. scikit. "https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mlp_training_curves.html"