A Machine Learning System for Predicting Team Win Rates, Character Builds, and Individual Performance in Eternal Return

Zyrus Crispino
College of Engineering, Architecture, and Mathematics

De La Salle University - Dasmarinas
Dasmarinas City, Philippines
crispino.zyrus@gmail.com

***Abstract*—This paper presents a Python-based Multi-Model machine learning system specifically for the Metrics of Team and Character Performance and Prediction in *Eternal Return*. The tool employs probability, pattern ranking, and common trends when it comes to common teams and character builds that consistently rank high in every match. Additionally, it accounts for a certain team or character's additional metrics such as team performance like team kills, kills, deaths, assists, as well as the total damage to other players and damage to other monsters, otherwise known in other games as creeps.**

A distinguishing feature of the system is its capability to parse user-inputted data through the three models and output the according predictions. In one model, the user will input three names of *Eternal Return* characters to form a team, and the model will output the predicted win rate of the team. In another model, the user will simply input the name of an *Eternal Return* character and receive information on their top 3 builds. And for the last model, the user will input the name of a single character from *Eternal Return* and receive that character's individual predicted win rate, as well as show the historical win rate of the character.

The primary objective of this system is to demonstrate the capabilities of Machine Learning algorithms in providing reliable statistics regarding Team Placement, Character Builds, and individual Character Winrate that can serve as the foundation for future work regarding similar subjects.

***Index Terms*—Machine Learning, Eternal Return, Team Performance Prediction, Character Build Prediction, Win Rate Prediction, Independent Models.**

I. Introduction

The use of Machine Learning (ML) in gaming has been prevalent in recent years, though its application in game systems and AI has been present for some time now. The rise of data availability and computational power and more advanced technologies by the minute has enabled more focused ML use in game development [1]. A growing number of projects can now process video game data to build predictive systems, often using in-game statistics as the basis. This tracks, as most gameplay mechanics rely on numerical systems to generate interactive complexity [2]. Genres like Multiplayer Online Battle Arenas (MOBAs) and battle royales exemplify this. MOBAs typically feature team-based combat where players control distinct units in strategic top-down environments [3]. Battle royales, in contrast, pit players or teams against each other in survival scenarios, usually in shrinking maps [4]. Eternal Return blends both genres: it uses team-based, ability-driven combat like a MOBA while following the last-team-standing structure of a battle royale. With over 80 characters and a massive variety of item builds, the number of possible team combinations is staggering—often overwhelming for new players. However, Eternal Return provides extensive match data: win rates, item usage, combat stats, and more. This allows for ML models to predict optimal team compositions, popular character builds, and individual character performance [5].

II. Related Work

Several systems that are fundamentally similar to what my work suggests have already been developed and implemented for similar games ranging from similar genres to genres of the opposite spectrum, and some are even implemented in websites. Such projects or systems include, but are not limited to the following;

A. Building a Machine Learning Model with Linear Regression to Predict Dota 2 Teams Winning

By utilizing team compositions and in-game statistics like hero selections and performance metrics, Cholig investigates the use of linear regression to forecast match outcomes in Dota 2 [6]. The main focus is on developing a single predictive model that generates the probability of a team winning based on the lineup chosen. Despite its simplicity, the method demonstrates the fundamental use of machine learning methods in competitive multiplayer games.

However, the study does not address build optimization or individual character metrics, and it is restricted to a single model and a single prediction type (win/loss). My suggested system, on the other hand, goes beyond this by presenting three distinct models that are uniquely adapted to the mechanics of Eternal Return and can forecast not just team win rates but also individual character win rates and optimized character builds and consider more character specific variables rather than simple win rate and pick rate.

B. League of Legends Win Analysis

Using League of Legends public match data, Zijie Mei does a statistical win rate study with an emphasis on identifying elements that increase the likelihood of winning [7]. The study highlights trends among top-performing players and teams by examining champion selections, performance indicators, and game length. The essay demonstrates how MOBA game data can yield valuable insights using very simple statistical techniques.

Mei's method focuses on correlation-based insights rather than predictive modeling, despite the fact that it is instructive. By employing real machine learning algorithms to generate predicted probabilities and applying these techniques to the structurally similar but distinct game Eternal Return, my methodology expands on this kind of data exploration.

III. Methodology

This project uses a three-model machine learning system, each implemented independently in Python notebooks. The models focus on team win prediction, character win prediction, and character build recommendation, respectively. All models were trained using a custom-built dataset collected via the Eternal Return API, pre-processed and evaluated using standard classification and recommendation metrics.

A. Approach

The project uses three individual models to accomplish three separate tasks. First, to identify Team Placement, second, to predict character builds, and third, to predict individual character win rate. Classifiers and regression based models were used to best identify the model that gave the best rated outputs.

Important data pre-processing operations including addressing class imbalance, transforming categorical variables, and normalizing inputs are all part of the modeling pipeline[6]. The system uses stratified 3-fold cross-validation to evaluate dependability. The top-performing models were chosen based on precision, recall, and F1 score after each model was trained and assessed using a variety of machine learning algorithms.

B. Algorithms

For the three models in this work, a variety of machine learning methods were used for both classification and regression tasks [7]. In order to identify the optimal performer for every predicting task, it was necessary to investigate performance across different algorithm families.

Here's a breakdown of the algorithm used:

C. Regression Models

Regression methods were used to approach the Character Win Rate Prediction and Team Win Rate Prediction models[8]. With the help of features including kills, fatalities, assists, team damage, and more, these models sought to estimate a continuous win rate number. We tested the following regression algorithms[9]:

1) Linear Regression
2) Ridge Regression
3) Lasso Regression
4) ElasticNet
5) Decision Tree Regressor
6) Random Forest Regressor
7) Gradient Boosting Regressor
8) Histogram-based Gradient Boosting
9) AdaBoost Regressor
10) Extra Trees Regressor
11) XGBoost Regressor[11]
12) LightGBM Regressor[12]

D. Classification Models

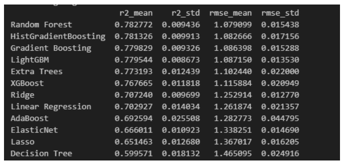
Predicting the best build (equipment + augments) for a particular character based on past match performance was the aim of the Character Build Recommendation model, which was structured as a classification problem[10]. The classification algorithms listed below were tested:

1) Random Forest (with class weights balanced)
2) Gradient Boosting
3) Logistic Regression
4) AdaBoost
5) Extra Trees
6) XGBoost[11]
7) LightGBM[12]
8) Support Vector Classifier (SVC)
9) K-Nearest Neighbors (KNN)
10) Naive Bayes

IV. Results

A. Team Win Rate Prediction (Regression)

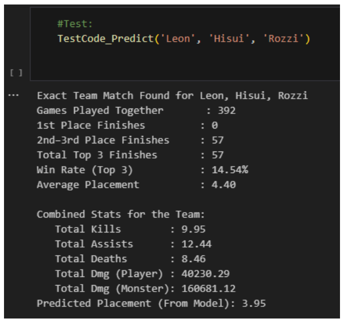
In this program, the model takes data from each game and compiles them into the proper dataframe. Taking into teams of 3 per game, and simplifying the team's stats, the model trains through the abundant dataset to learn. After testing through the models, the model that was identified to be the best was RandomForest: r2_mean: 0.782772 rmse_mean: 1.079099



```
                       r2_mean    r2_std  rmse_mean  rmse_std
Random Forest         0.782772  0.009436   1.079099  0.015438
HistGradientBoosting  0.781326  0.009913   1.082666  0.017156
Gradient Boosting     0.779829  0.009326   1.086398  0.015288
LightGBM              0.779544  0.008673   1.087150  0.013530
Extra Trees           0.773193  0.012439   1.102440  0.022000
XGBoost               0.767665  0.011818   1.115884  0.020949
Ridge                 0.707240  0.009699   1.252914  0.012770
Linear Regression     0.702927  0.014034   1.261874  0.021357
AdaBoost              0.692594  0.025508   1.282773  0.044795
ElasticNet            0.666011  0.010923   1.338251  0.014690
Lasso                 0.651463  0.012680   1.367017  0.016205
Decision Tree         0.599571  0.018132   1.465095  0.024916
```

Fig. 1. Multi Model Test

Test Program:



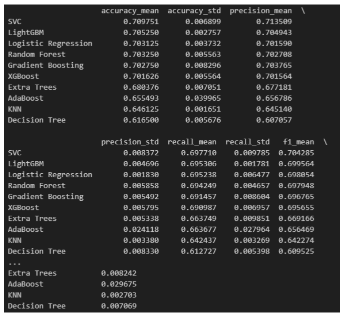
```
#Test:
TestCode_Predict('Leon', 'Hisui', 'Rozzi')
```

```
Exact Team Match Found for Leon, Hisui, Rozzi
Games Played Together       : 392
1st Place Finishes          : 0
2nd-3rd Place Finishes      : 57
Total Top 3 Finishes        : 57
Win Rate (Top 3)            : 14.54%
Average Placement           : 4.40

Combined Stats for the Team:
   Total Kills        : 9.95
   Total Assists      : 12.44
   Total Deaths       : 8.46
   Total Dmg (Player) : 40230.29
   Total Dmg (Monster): 160681.12
Predicted Placement (From Model): 3.95
```

Fig. 2. Sample Testing

B. *Character Build Recommendation (Classification)*

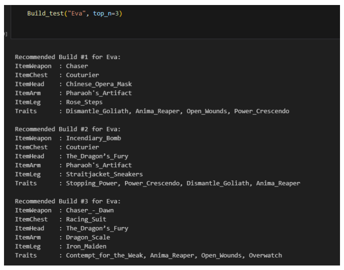
This classification model recommended the top 3 builds for a given character. It evaluated build effectiveness based on features like match outcomes, weapon types, and augment choices. After testing through the models, the model that was identified to be the best was SVC: accuracy_mean: 0.705250 precision_mean: 0.704943 recall_mean: 0.695306 f1_mean: 0.699564



```
                     accuracy_mean  accuracy_std  precision_mean  \
SVC                       0.709751      0.006899        0.713509
LightGBM                  0.705250      0.002757        0.704943
Logistic Regression       0.703125      0.003732        0.701590
Random Forest             0.703250      0.005563        0.702708
Gradient Boosting         0.702750      0.008296        0.703765
XGBoost                   0.701626      0.005564        0.701564
Extra Trees               0.680376      0.007051        0.677181
AdaBoost                  0.655493      0.039965        0.656786
KNN                       0.646125      0.001651        0.645140
Decision Tree             0.616500      0.005676        0.607057

                     precision_std  recall_mean  recall_std   f1_mean  \
SVC                       0.008372     0.697710    0.009785  0.704285
LightGBM                  0.004696     0.695306    0.001781  0.699564
Logistic Regression       0.001830     0.695238    0.006477  0.698054
Random Forest             0.005858     0.694249    0.004657  0.697948
Gradient Boosting         0.005492     0.691457    0.008604  0.696765
XGBoost                   0.005795     0.690987    0.006957  0.695655
Extra Trees               0.005338     0.663749    0.009851  0.669166
AdaBoost                  0.024118     0.663677    0.027964  0.656469
KNN                       0.003380     0.642437    0.003269  0.642274
Decision Tree             0.008330     0.612727    0.005398  0.609525
...
Extra Trees          0.008242
AdaBoost             0.029675
KNN                  0.002703
Decision Tree        0.007069
```

Fig. 3. Character Build Predictor Model Test

Test Program:



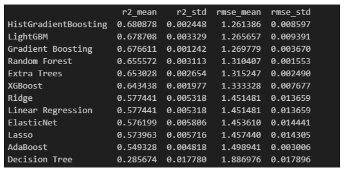
```
Build_test("Eva", top_n=3)
```

```
Recommended Build #1 for Eva:
ItemWeapon  : Chaser
ItemChest   : Couturier
ItemHead    : Chinese_Opera_Mask
ItemArm     : Pharaoh's_Artifact
ItemLeg     : Rose_Steps
Traits      : Dismantle_Goliath, Anima_Reaper, Open_Wounds, Power_Crescendo

Recommended Build #2 for Eva:
ItemWeapon  : Incendiary_Bomb
ItemChest   : Couturier
ItemHead    : The_Dragon’s_Fury
ItemArm     : Pharaoh's_Artifact
ItemLeg     : Straitjacket_Sneakers
Traits      : Stopping_Power, Power_Crescendo, Dismantle_Goliath, Anima_Reaper

Recommended Build #3 for Eva:
ItemWeapon  : Chaser_-_Dawn
ItemChest   : Racing_Suit
ItemHead    : The_Dragon’s_Fury
ItemArm     : Dragon_Scale
ItemLeg     : Iron_Maiden
Traits      : Contempt_for_the_Weak, Anima_Reaper, Open_Wounds, Overwatch
```

Fig. 4. Sample Testing

C. *Character Win Rate Prediction (Regression)*

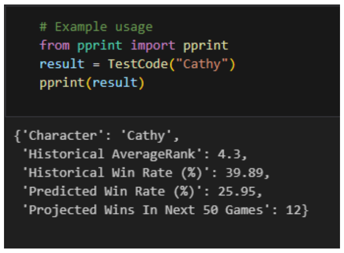
Lastly, this model predicts the individual win rate of a single character using features like personal performance stats, damage dealt, and frequency of use throughout the dataset. After testing through the models, the model that was identified to be the best was HistGradientBoosting: r2_mean: 0.680878 rmse_mean: 1.261386



```
                        r2_mean    r2_std  rmse_mean  rmse_std
HistGradientBoosting   0.680878  0.002448   1.261386  0.008597
LightGBM               0.678708  0.003329   1.265657  0.009391
Gradient Boosting      0.676611  0.001242   1.269779  0.003670
Random Forest          0.655572  0.003113   1.310407  0.001553
Extra Trees            0.653028  0.002654   1.315247  0.002490
XGBoost                0.643438  0.001977   1.333328  0.007677
Ridge                  0.577441  0.005318   1.451481  0.013659
Linear Regression      0.577441  0.005318   1.451481  0.013659
ElasticNet             0.576199  0.005806   1.453610  0.014441
Lasso                  0.573963  0.005716   1.457440  0.014305
AdaBoost               0.549328  0.004818   1.498941  0.003006
Decision Tree          0.285674  0.017780   1.886976  0.017896
```

Fig. 5. Multi Model Test Results

Test Program:



```
# Example usage
from pprint import pprint
result = TestCode("Cathy")
pprint(result)
```

```
{'Character': 'Cathy',
 'Historical AverageRank': 4.3,
 'Historical Win Rate (%)': 39.89,
 'Predicted Win Rate (%)': 25.95,
 'Projected Wins In Next 50 Games': 12}
```

Fig. 6. Sample Testing

D. *Real-World Application*

Players and teams looking to gain an advantage in Eternal Return can benefit from the machine learning models used in this project. Players can make better selections throughout the planning and drafting stages by predicting win rates based on team configurations and individual characters. By helping novice players find high-performing item and augment combinations, the Character Build concept speeds up their learning curve and enhances performance without depending entirely on trial and error. Furthermore, the framework and dataset offer a starting point for further study on team-based strategy games, allowing for further in-depth investigations into adaptive matchmaking, player behavior prediction, and synergy modeling.

V. Conclusion and Future Work

This study shows that it is feasible to predict team win rates, individual character win rates, and suggest the best character builds in Eternal Return by employing a multi-model machine learning approach. The models demonstrated encouraging outcomes in terms of prediction accuracy and usefulness after undergoing a thorough pre-processing procedure and evaluating more than ten machine learning techniques for each task. In competitive situations, athletes, strategists, and analysts can use these instruments’ insightful information to make better informed choices.

VI. Acknowledgments

First and foremost, I want to sincerely thank my instructor, Engr. Montances, Mark O., M.Sc. - ”Sir Mong,” for giving the direction and inspiration that enabled this endeavor. Even though I had been considering this undertaking for a while, I just did not have enough information or motivation to start.

Finally, I would like to extend my thanks to Eternal Return, a game I hold near and dear to my heart, and sometimes a game I wish to throw to the deepest pit there is on and off earth.

References

[1] The role of Machine Learning in Game Development Domain - A review of current trends and future directions. (2021, November 1). IEEE Conference Publication — IEEE Xplore. https://ieeexplore.ieee.org/document/9647261

[2] The role of Machine Learning in Game Development Domain - A review of current trends and future directions. (2021, November 1). IEEE Conference Publication — IEEE Xplore. https://ieeexplore.ieee.org/document/9647261

[3] What are MOBA games? A complete guide. (n.d.). https://www.konvoy.vc/blogs/what-are-moba-games

[4] Moba vs Battle Royale: What are some popular Battle Royale games? — Lenovo Philippines. (2023, May 28). https://www.lenovo.com/ph/en/glossary/moba-vs-battle-royale-games/

[5] Eternal Return Review The Unspoken Diamond of Battle Royales Game8. (2024, August 14). Game8 the Top Gaming and App Walkthroughs Straight From Japan! https://game8.co/articles/reviews/736

[6] Google for Developers. (2016, May 11). *Let's write a pipeline - Machine learning Recipes #4* [Video]. YouTube. https://www.youtube.com/watch?v=84gqSbLcBFE

[7] Data Professor. (2022, May 27). *Build your first machine learning model in Python* [Video]. YouTube. https://www.youtube.com/watch?v=29ZQ3TDGgRQ

[8] *What is a regression model?* (2021, June 16). IMSL by Perforce. https://www.imsl.com/blog/what-is-regression-model#:~:text=A%20regression%20model%20provides%20a,by%20a%20linear%20regression%20model.

[9] *scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation*. (n.d.). https://scikit-learn.org/stable/

[10] *Choosing the best machine learning classification model and avoiding overfitting*. (n.d.). MATLAB & Simulink. https://www.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html

[11] *Python Package Introduction — xgboost 3.0.0 documentation*. (n.d.). https://xgboost.readthedocs.io/en/stable/python/python_intro.html

[12] *Python-package Introduction — LightGBM 4.6.0.99 documentation*. (n.d.). https://lightgbm.readthedocs.io/en/latest/Python-Intro.html

[13] *NumPy*. (n.d.). https://numpy.org/

[14] *pandas - Python Data Analysis Library*. (n.d.). https://pandas.pydata.org/

[15] "Matplotlib — Visualization with Python." Available: https://matplotlib.org/

VII. Appendix

A. *Appendix: Use of AI Tools*

- **Debugging Assistance**: ChatGPT was utilized to troubleshoot and debug issues related to drop rate calculations and the overall simulation logic. It provided valuable suggestions for identifying errors and optimizing code performance.
- **Code Snippets**: The AI assisted in generating code snippets for implementing specific features, such as random number generation for simulating item drops.
- **Customization**: All code suggestions provided by ChatGPT were thoroughly reviewed, customized, and refined to meet the specific requirements of the simulation project. This process included adjusting logic, and optimizing the performance of the generated code.
- **Learning and Understanding**: ChatGPT served as a learning resource, explaining the functionality of various code components. This ensured a comprehensive understanding of the underlying logic before integrating it into the project.

B. *Appendix: Graphs and Charts*

This appendix section includes any visual representations of the data.

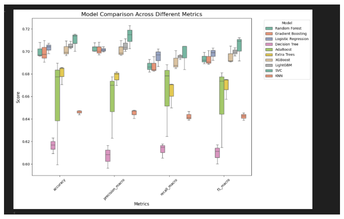


Fig. 7. Box Plot Multiple Models

C. *Appendix: Terminologies*

This section provides definitions and explanations of key terms used throughout the paper.

- **API (Application Programming Interface)** – Used to retrieve raw match data directly from Eternal Return's backend service, enabling dataset construction.
- **Augments** – Passive abilities or buffs equipped before a match that influence a character's performance.
- **Character Build** – A combination of weapon, equipment, and augments optimized for a specific character to maximize performance in matches.
- **Classification Model** – A machine learning model used to categorize data points, such as predicting the top 3 builds for a given character.
- **Creeps (Monsters)** – Non-player units in the game that players can fight for experience or items; tracked in damage statistics.
- **Cross-Validation** – A technique for evaluating ML models by dividing data into multiple training and testing subsets to ensure generalizability.
- **Eternal Return** – A multiplayer online survival game combining battle royale and MOBA elements, where players choose characters with unique abilities and compete to be the last one standing.
- **F1 Score** – A performance metric combining precision and recall to measure a model's accuracy, especially useful for imbalanced datasets.
- **Individual Winrate** – The expected win percentage of a single character, based on historical performance data.
- **Match Data** – Information collected per game session, including kills, deaths, assists, team placement, and damage dealt.
- **Regression Model** – A model used to predict continuous values, such as estimating win rates or team placements.
- **Team Winrate** – The predicted probability or percentage chance that a three-character team will win a match.