Predicting Victory: Evaluating Machine Learning Models for Win Percentage in League of Legends

Jake Moore
jmoore3@oxy.edu
Occidental College

1 Introduction and Problem Context

Machine learning is becoming widely used across many professional sports and has become a very valuable tool to assist teams with decision-making. Many basketball, baseball, soccer, and football teams apply machine learning algorithms to explore players' performance and strategies during games and discover winning patterns. Machine learning algorithms provide the ability to process massive amounts of player and game data so teams can predict outcomes, determine the chances of potential injuries, and even recommend lineup changes in real-time. In baseball, predictive analytics are used to evaluate pitch selection and defensive positioning, while soccer teams analyze player tracking data to improve passing accuracy and defensive coverage.

Outside the field, teams use these insights to impact decisions around player recruitment, contract negotiation, and fan engagement strategy. With data-driven insights in their hands, machine learning is revolutionizing the modern sports scene, providing teams with a competitive advantage, fostering risk minimization and maximizing efficiency. Machine learning will not only reshape the world of traditional sports but is already starting to find its place in some popular esports as well, where decision-making, strategy, and complexity of gameplay can also require some detailed analytics. A prime example of this is League of Legends, an incredibly competitive and popular online game with a steep learning curve for players looking to improve—specifically, exactly what leads to triumph or failure in each match. Certainly individual skill and team coordination are important, but the intricate relationships between in-game events and strategic decisions can be hard to identify.

This lack of clarity makes it difficult for players to learn where they went wrong and what they should have done differently. The goal of this project is to address this issue by building a machine learning model that accurately estimates win probabilities at each of a game's minutes. The model will analyze hundreds of potential variables from the game including the number of towers destroyed, gold differences, and objective control to determine key inflection points in the game and significant influencing factors on the outcome. By providing insightful details about game progress, this model will help casual players enhance their in-game decision-making and foster a more informed and strategic approach to the game. This project aims to help increase the level of play for the average player by enabling them to learn from their mistakes with data driven analysis in League of Legends.

2 Technical Background

To fully understand this project, a basic understanding of machine learning and the game of League of Legends is important. League of Legends is a multiplayer online battle arena (MOBA) game where you have two teams of five players each trying to destroy the other team's base. Each player controls a unique character, called a "champion," with special abilities. Complex aspects such as "power spikes," and champion items contribute to the high level of strategy and decision-making required in each match.

2.1 Machine Learning

The heart of this project is machine learning, a field of artificial intelligence that allows computers to learn from data. This allows a program to create a model, which produces data driven results with no human biases. For this project, supervised learning techniques are employed, specifically regression and classification algorithms. Regression models predict continuous numerical values, such as win probability, while classification models categorize data into discrete classes (e.g., win or loss). This project tests three machine learning models: Random Forest, support vector regression (SVR), and Multi-Layer Perceptrons (MLP). Random Forest, an ensemble method, combines predictions from multiple decision trees, improving accuracy and reducing overfitting which is suitable for interpreting and analyzing data like game statistics. SVR excels at classifying data with clear boundaries by finding an optimal hyperplane which works well for precise structured classifications. Finally, MLP neural networks capture complex, non-linear relationships in data, allowing the project to explore intricate patterns that may not be found with other models.

2.2 Feature Engineering

Before entering data into a model, it is important to make sure it is actually important in determining the game. Useless data not only makes the models take longer to train but can often make the models worse since trends in that data will be taken into consideration even if that data is not relevant. In the context of League of Legends, there are many crucial features, such as gold difference, kill counts, towers destroyed, and control of neutral objectives like Baron Nashor or Dragon. Beyond simple metrics some features are time dependent which adds another layer of complexity and vastly changes the importance of each feature. For example, one tower destroyed eight minutes into the game is a lot while one tower destroyed 30 minutes into the game is almost nothing. These features allow the model to capture winning trends, import turning points in the game, as well as game winning conditions. Proper feature engineering is essential as it ensures more accurate predictions.

2.3 Model Selection and Evaluation

Choosing the appropriate type of model for the project depends on how complex the data is, the interoperability of the data, as well as computational power available. By choosing Random Forest for its robust performance and interpretability, SVR for its precision in structured classifications, and MLP for modeling non-linear relationships, this project creates an environment where the best option can be found. The high interpretability of Random Forest is in part due to easy feature analysis, which allows for seeing what features are most important. This gives great insight into what features have the most impact on winning games. By combining machine learning algorithms, feature engineering, and model evaluation, this project can help make the secrets of winning in League of Legends available to more people. It will help empower players to make informed decisions, improve their gameplay, and better understand the factors that lead to victory or defeat.

3 Prior Work

Applying machine learning to predict outcomes in sports is a rapidly growing field. With the increase in sports betting creating an accurate line is more important than ever, and machine learning models are being used to help with that. For instance, [1] applied a Random Forest algorithm to professional tennis, identifying speed of serve as a pivotal factor in predicting match outcomes. Similarly, [2] use machine learning techniques, including logistic regression and decision trees, to predict outcomes in Korean professional baseball. Their model relies on team and player statistics to forecast match results showcasing how machine learning can reveal critical patterns in player performance, and game-specific metrics. Their work helps to show the versatility of machine learning in analyzing competitive environments and improving accuracy by focusing on specific performance indicators. These two studies demonstrate that even in sports, where things like player form, weather conditions, and even how the players match up against their opponents can be predicted by a machine learning algorithm.

In esports, machine learning applications have gained significant traction, this is especially true in games like League of Legends. Riot Games implemented a "Win Probability" system during the 2023 World Championship, which calculates win probabilities based on real-time in-game data such as gold differences, objective control, champion scaling, and other key performance metrics. This tool is only available during professional games and has been proven to be a valuable tool in enhancing the viewing experience by providing actionable, data-driven insights into the flow of professional matches. This is important since compared to the games of average players, professional games are vastly more complicated and many fewer mistakes are made and even when a mistake is made they are not usually obvious. The win probability helps casual watchers understand that events that are commonplace in their own games could be pivotal to a win or loss on a professional level, helping to keep them engaged.

This project seeks to replicate Riots "Win Probability" [3]system with a few adaptations to make it more suitable for the average player. By leveraging advanced machine learning techniques such as Random Forest, Multi-Layer Perceptrons (MLP), and support vector regression (SVR), this model aims to capture the intricate dynamics of League of Legends by taking into account factors like gold difference, experience difference, towers destroyed and objectives slain. Unlike Riot Games' system, which focuses on live professional-level matches and is only available on broadcasts, this approach broadens the accessibility of machine learning driven data analytics to all players, with models trained based on data from the rank of average players.

One of the primary challenges faced by predictive models in esports is the constant evolution of the game environment. Games often undergo frequent updates, so if the characters are made stronger or weaker it would greatly affect the accuracy of the model.. For example, if a model was going to be built on a game like Overwatch, if a champion had their abilities changed to do more damage, the model wouldn't know that and we would have to be retrained. This dynamic nature of esports poses a challenge for machine learning models, as they must continuously adapt to the shifting landscape of the game. But, that is why League of Legends is a special game, and why the "Win Probability" system of riot games is something that can be repli-

cated. League of Legends is special because there are a lot more numerical data values than there is in other games, and those values have a much more direct impact on Win Percentage than other games as well. For example, gold earned is a number, and team gold is the most important statistic for determining win rate. In fact, every feature chosen for the model is a number, and almost none of them will change in importance with subsequent patches.

The growing body of research in sports and esports analytics highlights the increasing importance of data-driven decision-making in competitive environments. By synthesizing insights from both traditional sports and esports, this project can reverse engineer what big corporations and professional franchises are doing while making it both available for and tailored to average players. Machine learning techniques offer significant insight, giving a huge boost in player understanding, allowing them to improve strategies and performance. Moreover, the ability to provide feedback and predictions opens up new possibilities for strategies based on newly revealed insights into important aspects of the game.

3.1 Methods

3.2 Initial Approach

The approach to this project evolved significantly throughout its development. Initially, the goal was to create a tool that could provide real-time predictions for live games of League of Legends. This method faced two major challenges: first, the lack of readily accessible data from live games for model training, and second, the risk of violating Riot Games' Terms of Service (TOS). Riot Games' TOS prohibits third-party software from reading live game data directly, and such violations could lead to account bans. To navigate these constraints, the first iteration of the project involved recording the game screen and performing image analysis to extract relevant in-game data in real-time. The primary elements that would have been analyzed in this first approach include the kill feed (indicating kills, objectives, or turrets destroyed), the game map (providing player and enemy positions when certain conditions were met), and the scoreboard (displaying player statistics, ultimate ability status, and item purchases). This method would have allowed for real-time gameplay suggestions, an intriguing concept in the chaotic environment of League of Legends. However, the significant issue with this approach was the processing time: image analysis requires substantial computation, which results in a delay of 10-20 seconds in delivering any feedback. In League of Legends, a few seconds can dramatically alter the course of the game, making real-time recommendations not feasible.

3.3 Current Approach

In response to these limitations, the project transitioned to using Riot Games' official Developer API to access post-game data. This revised method streamlined data gathering and supported the creation of a model that would offer personalized advice to players, helping them identify areas of improvement based on the statistical analysis. This led to the current phase of the project, which involves testing three different machine learning models: Random Forest Classification (RFC); Support Vector Machines (SVM); and Multi-Layer Perceptrons (MLP) to determine the most effective model for the task.

3.4 Model Selection and Justification

During the selection process for the machine learning models to be used in this project, one critical consideration was the complexity of the data. Predicting how likely a team is to win in a game like League of Legends is challenging due to the ever changing and semi-linear nature of multiplayer games. Players' decisions, champion picks, in-game objectives, and team dynamics all contribute to the outcome making it essential to capture both linear and non-linear relationships within the data.

RFC was selected due to its high capability of dealing with high-dimensional feature space and ability to model both linear and non-linear relationships. It is an ensemble method that builds several decision trees during training and outputs the mode of their classes (classification) or mean prediction (regression) of the individual trees. This has several benefits:: it mitigates the risk of overfitting used when training on a complex dataset, and it provides feature importance, the ability to understand which features (e.g. player stats, or champion picks) are predictive of a given team performance. Furthermore, the RFC was a better fit as match data is often uncertain with very few matches having all the variables accounted for or even outliers.

Support vector regression (SVR), on the other hand, were selected for their ability to work well in high-dimensional spaces. SVRs are particularly effective in scenarios where the decision boundary is complex and non-linear, which is often the case in competitive games where player interactions and strategies vary greatly. SVRs are also known for their ability to generalize well, even in cases where the dataset has relatively few features., making them a valuable tool in predicting win probabilities when the dataset is carefully engineered for optimal performance.

Multi-Layer Perceptrons (MLP) were considered due to their capacity to learn highly complex, non-linear relationships through the use of multiple layers and neurons. MLPs have been successful in other prediction tasks, particularly when the relationships between features are intricate and

not easily captured by traditional models. In the context of League of Legends, where numerous interactions between players and in-game events influence the outcome, MLPs have the potential to model these complex relationships. However, they are more prone to overfitting and require careful tuning of hyperparameters to achieve optimal results.

By evaluating these three models, the goal was to capture a diverse range of relationships within the data, ensuring a robust prediction system that can handle both linear and non-linear interactions in a game as complex as League of Legends.

3.5 Data Collection and Database Setup

The first step to creating a reliable prediction model is to gather a substantial amount of high-quality data. For this project, data was collected from League of Legends' matches using Riot Games' API, which provides detailed information about in-game statistics and game events. The data was then stored in a PostgreSQL database, PostgreSQL was chosen as it is a highly advanced open source database tool that is perfect for this project. PostgreSQL allows for structured data storage and querying, which is important for scaling the size of the dataset. It's also important to note that the simple nature of PostgreSQL allowed for the table being used to be restructured many times without issue.

The database included one main component, a single table with a binary outcome column, and a JSON object column. The binary column held the data for which team won the game while the JSON column contained all of the match data for the game. The use of a JSON allowed for the easy grouping of information based on time into the game, so every group of data was from the same time. The flexibility of PostgreSQL permitted the data to be easily stored in this manner, which was really helpful for both storing data and pulling data back out of the database.

To populate the database, 2,000 game entries were collected from the platinum rank. These games were chosen from one rank since feature importance changes greatly from rank to rank as less mistakes are made in higher ranks. Platinum is also slightly higher ranked than the average player so most players would be able to use this model. More models would need to be trained with data from higher and lower ranks to increase accuracy. The first step in the data collection process involved retrieving the Player IDs for players in the relevant rank. After obtaining the Player IDs, the match ID for their last ranked game needs to be obtained. After the match IDs were gathered, detailed match timelines were fetched, which contained key in-game moments, such as kills, objectives secured, and item purchases. Then, the important stats deemed to influence win percentage were gathered and stored in the database as well as which team ended up winning the game.

The structure of the database allowed for easy expansion, meaning that more games could be added easily with no changes to any of the code that used the database or alterations to the database itself. The first step for making an accurate model is comprehensive and organized data. The structure of the database also allowed for easy additions of new features, although the old data would need to be removed.

3.6 Data Preprocessing and Feature Engineering

After populating the database, the next step was to preprocess the data before training. The raw JSON data was reformatted into a dataframe, which is a table like structure. This restructuring is important as it made training a model much easier. Feature engineering was one of the most important steps in this project as it greatly improved the Brier score for all three models. For instance, features such as "gold difference" and "turrets destroyed" would be included in the model training. After the model was trained and a Brier score produced, a feature would be removed, or a new one would be added. This form of testing would be continued until the Brier score would not improve.

3.7 Model Training and Evaluation

Once the dataset was prepared, and the initial parameters and features were chosen, the next step was to train the three selected machine learning models: RFC, SVR, and MLP. All three models were trained using scikit-learn since it is a versatile framework for models, and allows for easy testing with its vast array of prebuilt functions. During the training phase, the hyperparameters of each model were tested to see what would optimize the models performance. For example, the Random Forest model required careful selection of the number of trees and the maximum depth to balance model complexity and accuracy.

To evaluate the performance of the models, the Brier score was used. The Brier score is a metric that quantifies the accuracy of probabilistic predictions by measuring the mean squared difference between predicted probabilities and actual outcomes. Brier score is on a scale of zero to one, with a lower Brier score indicating better model performance. This metric is particularly suited for this task, as it allows for the evaluation of not only the accuracy of the predictions but also the confidence with which the model assigns probabilities to each outcome. In other words, it doesn't grade the models based on how many times they guessed the correct team winning, but by how accurate the percent the model gave was to the actual win percentage.

Throughout the training process, various feature combinations and hyper-parameter setups were experimented with

to minimize the Brier score. For instance, the impact of including or excluding certain features were tested and the resulting change in Brier score was analyzed to see how the changes affected the accuracy of the predictions.

4 Evaluation Metrics

ince the models are not classifying the teams into which team is going to win and which is going to lose, Brier score is an important tool in evaluating how good the win percentage actually is. In the context of a win probability prediction model, the Brier score quantifies the accuracy of the model's predicted probabilities by calculating the mean squared difference between the predicted probabilities and the actual binary outcomes (1 for a win, 0 for a loss). This metric is essential because without it, it's impossible to determine whether a win percentage is accurate

For example, if the model predicts a 70 percent chance of a team winning and the team wins, the squared difference between the predicted probability and actual outcome is small (0.09), indicating a highly accurate prediction. On the other hand, if the model predicts a 70 percent chance of winning but the team loses, the squared difference is much larger (0.49), signaling a poor prediction. This means that the Brier score is evaluating the model based on how accurate the percentages are, not whether the team it chose does end up winning. Brier score is on a scale of zero to 1, a model with a lower Brier score, approaching zero, indicates that the predicted probabilities are closely aligned with the actual outcomes and signifies a well-calibrated model. In contrast, a higher Brier score suggests that the model's predictions are less reliable or overconfident, and its probabilities do not reflect the actual probabilities of outcomes. For instance, if a model consistently overestimates or underestimates the win percentage, then the Brier score will be higher, resulting in a poorer evaluation of the model.

That being said, it is important to note that since the model is running on League of Legends, there are ten human beings in those games. This means that there are a lot of League of Legends' games where a team is winning and they end up losing due to unpredictable human error. This will punish the model in terms of Brier score, so there is a "floor" so to speak, where the Brier score cannot be lower than. In addition to the floor, all of the models are also going to be slightly punished by this fact as well.

4.1 Comparing to Other Metrics

Accuracy is often the first metric used to evaluate a model's performance, yet it has limitations when it comes to probabilistic models. Accuracy measures how often the model's predictions match the actual outcomes but does not take into account the model's confidence in its predictions. For example, if the model predicts that a team has a 100 percent chance to win and that team does win, the accuracy would say that the model was correct but the actual win percentage could have been 55 percent with the lead they had. This approach does not provide any information about the quality or reliability of the predicted probabilities, just whether the team that was predicted to win actually ended up winning.

Other commonly used metrics that offer valuable insights for evaluating probabilistic models are log loss and the Area Under the Curve (AUC). Log loss, for instance, penalizes predictions based on the logarithmic difference between predicted probabilities and actual outcomes which makes it sensitive to large errors in predictions.

Brier score stands out due to its intuitive interpretation of prediction quality and error. It directly measures the mean squared difference, providing a clear and straightforward indication of how well the model is predicting probabilities. This makes the Brier score particularly suitable for this project, where the goal is not just to predict outcomes but to provide an interpretable and actionable measure of prediction confidence.

Furthermore, the Brier score has the ability to accurately evaluate the calibration and accuracy of the models. Because this project has three distinct models, the Brier score allows for an objective comparison of them, not only based on their accuracy but also in terms of how well their predicted probabilities align with the actual outcomes. This enables the best-performing model to be selected while ensuring that the predicted probabilities are reliable and well-calibrated, ultimately enhancing the quality of the insights provided to players.

The Brier score is an indispensable tool for evaluating the effectiveness of probabilistic models, especially in the context of win percentages in complex environments. Its ability to measure both accuracy and calibration provides an accurate and easily interpretable metric for comparing all three types of models. By focusing on minimizing the Brier score, the model can be refined to offer more accurate and actionable insights, ultimately improving player performance and decision-making.

5 Results and Discussion

In this study, three machine learning models were tested to predict the likelihood of a team's victory in League of Legends: the Random Forest Classifier (RFC), the Multilayer Perceptron (MLP), and Support Vector Regression (SVR). All three models were evaluated using the Brier score which is a metric for assessing the accuracy of probabilistic predictions.

5.1 Random Forest Classifier (RFC)

The Random Forest Classifier (RFC) demonstrated the best performance among the three models. The RFC works by combining predictions from multiple decision trees making it a robust ensemble model capable of handling a variety of data types and relationships. In this project, RFC excelled in predicting win probabilities because it is well-suited to handle the complex, high-dimensional data typically found in League of Legends matches. The model effectively managed non-linear relationships between features and provided well-calibrated predictions, as evidenced by its lowest Brier score. The RFC's showed a high reliability in its predictions as they are consistent with the actual outcomes of the games.

5.2 Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) is a type of neural network known for its ability to model complex relationships in data. However, the MLP performed second to the RFC in this study. Although it demonstrated reasonable performance, the MLP struggled to generalize well from the training data, likely due to its higher susceptibility to overfitting and the high dimensionality of the input features. Another possible reason for the higher Brier score was the lack of data, while 2000 games may seem like a lot, much more may be required to further lower the score. The MLP was more sensitive to the choice of hyperparameters, and while it was able to capture some patterns in the data, its predictions were less accurate overall. This is especially important to note, as the computer that these models were being trained on lacked the processing power to efficiently train MLP. This means that there are combinations of hyper parameters that may have been better than what was found, but couldn't be tested due to computational limitations. The higher Brier score for the MLP indicated that the model was not as well-calibrated as the RFC, with its predicted probabilities branching out more from the actual outcomes.

5.3 Support Vector Regression (SVR)

The Support Vector Regression (SVR) model, which is typically used for predicting continuous numerical outcomes, performed the worst among the three. Although SVR is typically good for regression tasks, it struggled in this classification problem to predict probabilities of a win vs loss outcome. SVR was unable to handle the probabilistic nature of the task, resulting in less accurate predictions. The model showed a significantly higher Brier score than both the RFC and MLP, indicating poor calibration. SVR's predictions were farther from the actual outcomes which made it less suitable for this type of problem where accuracy and calibration of predicted probabilities are crucial. Similar to MLP, computing power was a limiting factor for SVR as there are a lot of combinations that could be run, and while dozens of tests were run on the model it would have taken hundreds to get the best combination.

5.4 Model Comparison and Conclusion

When comparing the performance of the three models, it is clear that the Random Forest Classifier (RFC) was the best choice for predicting win probabilities in League of Legends. Its ability to aggregate predictions from multiple decision trees and manage both linear and non-linear relationships within the data contributed to its strong performance. The Brier score for RFC was the lowest which showed that its predictions were well-calibrated and closely aligned with the actual match outcomes.

The Multilayer Perceptron (MLP), while still a powerful model, was more prone to overfitting and struggled with generalization with this data. Although it performed better than the Support Vector Regression (SVR), its higher Brier score suggested that it was not as accurate or well-calibrated as the RFC.

The Support Vector Regression (SVR), on the other hand, was the least effective model for this task. Despite its success in other regression applications, SVR struggled with the probabilistic nature of the prediction task and was unable to provide accurate or well-calibrated win probabilities. SVR received the highest Brier score of the three models.

In conclusion, the Random Forest Classifier (RFC) proved to be the most reliable model for predicting win probabilities in League of Legends, followed by the Multilayer Perceptron (MLP), and then the Support Vector Regression (SVR). The ensemble nature and robustness of the RFC made it the best choice for this application and it provided the most accurate and well-calibrated predictions

6 Ethical Considerations

It is important to address the ethical implications of providing the win probability predictions in League of Legends. Data privacy, algorithmic bias and the potential impacts on player behavior within the community should be considered.

6.1 Data Privacy

Data privacy is probably the biggest concern surrounding this project. The model relies on accessing player statistics through the Riot Games API which uses player data without explicit consent for purposes beyond game-related functionality. While Riot Games provides this data to enhance the gaming ecosystem, developers and third-party applications

must be cautious about how they use this data. There are privacy laws like the General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) that work to protect players' data. Proactive measures should be used to anonymize and aggregate player data to minimize risks to player privacy. It is important to avoid the identification of individual users. Players should have the option to opt out of data collection.

6.2 Algorithmic Bias

Another potential risk is algorithmic bias. The features, such as specific in-game statistics, used in the model, may inadvertently select certain strategies or style of play. For example, if the model weighs certain actions more heavily than others, there could be a preference for aggressive strategies which then might alter a player's creativity and there's a risk of missing out on unique styles of play. It's important for the model to consider diverse styles of play that contribute to the vibrant community of League of Legends.

6.3 Impact on Player Behavior

Another ethical concern surrounds player behavior. The main purpose of a game like League of Legends is to foster creativity, spontaneity, and enjoyment. By providing real-time predictions about win probabilities, the model may encourage players to focus more on actions based on the model's suggestions, rather than exploration and enjoyment of the game. As Soren Johnson stated, "Given the opportunity, players will optimize the fun out of games." If players focus on model-driven decisions over creative play then the game could become more mechanical and less engaging or joyful.

6.4 Reinforcing Toxicity

A reliance on predictive tools could also exacerbate toxicity within the League of Legends community which is already known for its contentious environment. If players are provided with real-time insights into their mistakes, the model could unintentionally become a source of blame. This is a highly competitive game where emotions run high so providing direct feedback about what caused a loss could lead to negative interactions among players. For example, players could use the information to accuse teammates of making mistakes which would only fuel the toxic atmosphere inherent in competitive matches. To mitigate this, the model should not only provide win probability predictions but also encourage positive reinforcement and highlight areas for improvement without assigning blame. Future iterations of the model should incorporate data that promotes teamwork and cooperation. Instead of just focusing on individual performance, the model could also consider collaborative strategies among teammates.

6.5 Design Considerations

To address these ethical concerns, the model should be designed as support for casual players rather than an omnipotent guide. It could offer insights into game dynamics, allowing players to better understand the intricacies of the game without prescribing every decision. By emphasizing that the model is an aid for learning and enhancing gameplay, rather than a replacement for player creativity and strategic thinking, players would feel empowered to make their own decisions.

Additionally, warnings should be included stressing the importance that the model not be used for negative purposes, such as blaming teammates for losses. Features that encourage positive interaction, such as rewarding cooperative play or highlighting effective teamwork, will help foster a healthy and supportive environment within the community. This model should help promote a respectful gaming experience and help ease some frustrations for more casual players.

6.6 Conclusion

The goal of enhancing League of Legends gameplay is important, it is also crucial to maintain the integrity of the gaming experience. By considering the ethical implications such as privacy, bias, and player behavior, this predictive tool can not only benefit players but also uphold the core values of enjoyment, creativity, and community. Ensuring that the model is optional and designed to promote positive interactions will help prevent the game from becoming overly mechanized and will allow players to retain the freedom to experiment with new strategies and playstyles. This approach will help balance technological advancements with the ethical responsibility of maintaining a fair, enjoyable, and inclusive gaming environment for all players.

7 Replication Instructions

To replicate the League of Legends win rate prediction project, follow the steps below. Ensure you have the required environment, software, and dependencies installed. This project utilizes Python 3.10.6, with the primary libraries being pandas (v1.5.0), numpy (v1.23.3), scikit-learn (v1.2.0), psycopg2 (v2.9.7), and matplotlib (v3.5.3). The dataset is stored in a PostgreSQL database that you will need to set up and populate based on the project's specifications. The project was developed and tested on a Windows

11 environment but should work on macOS and Linux systems with minimal adjustments.

7.1 Set up the environment

Install Python 3.10.6 from the official Python website, then use a virtual environment to manage dependencies. Create and activate a virtual environment by running the following:

Listing 1: Creating the environment

```
python -m venv venv
source venv/bin/activate  # For macOS/Linux
venv\Scripts\activate
```

7.2 Install Dependencies

Install the required python Packages using pip

Listing 2: Creating the environment

```
pip install pandas==1.5.0 numpy==1.23.3
    scikit-learn==1.2.0 psycopg2==2.9.7
    matplotlib==3.5.3
```

7.3 Set up database

Step 1: Install PostgreSQL and pgAdmin Download and Install PostgreSQL:

Go to the PostgreSQL website and download the installer for your operating system. The installer includes both the PostgreSQL server and pgAdmin. Set Up PostgreSQL:

During installation, you'll be prompted to set a password for the postgres user (superuser). Remember this password. Launch pgAdmin:

Open pgAdmin after installation. You'll be asked to log in using the password you set earlier. Step 2: Create a Database Connect to the Server:

In pgAdmin, expand the "Servers" menu on the left sidebar. Right-click on the PostgreSQL server and select Connect. Enter your password if prompted. Create a New Database:

Right-click on Databases in the left sidebar and select Create ¿ Database. In the dialog box, enter the name of your database (e.g., lolPrediction) in the Database field. Click Save. Step 3: Create a Table Navigate to the Database:

Expand the Databases menu in the left sidebar. Click on the newly created database (lolPrediction) to select it. Create a New Table:

Expand Schemas ¿ public ¿ Tables in the database tree. Right-click on Tables and select Create ¿ Table. Define Table Name and Columns:

In the dialog box, provide a name for your table (e.g., matches). Switch to the Columns tab. Click the + button to add a new column. For each column, specify the Name, Data Type, and other settings (e.g., whether it's a primary key, allows NULL values, etc.). Example columns for a matches table:

matchId (Primary Key, integer, Not NULL) team (varchar(50), Not NULL) kills (integer, Not NULL) deaths (integer, Not NULL) assists (integer, Not NULL) Save the Table:

Once all columns are added, click Save to create the table.

7.4 Code Overview

The architecture of this project is designed to predict the likelihood of a team's victory in League of Legends matches using machine learning models. The system is modular and consists of several key components that work together seamlessly to fetch, preprocess, train, and evaluate predictive models.

At the core of the system is the data access module, which is responsible for retrieving match data from a PostgreSQL database using SQLAlchemy. This data consists of various metrics, such as gold difference, experience difference, and in-game events, which are stored in JSON format. Once the data is fetched, it is passed to the data preprocessing module. This module processes the raw data by extracting relevant features from the JSON structures and organizing them into a clean format suitable for machine learning. The features include game statistics like team gold, experience points, and events like kills and towers, with the target variable being the winning team. The system employs multiple machine learning models—Random Forest, Multilayer Perceptron (MLP), and Support Vector Regressor (SVR)—for training and predicting the win probabilities of the teams. These models are implemented in separate files, each configured with adjustable parameters for hyperparameter tuning. The models are trained on the preprocessed data, and the features are scaled using StandardScaler to ensure that all features are on the same scale, which is crucial for models like MLP and SVR.

After training the models, the system evaluates their performance by calculating the Brier score, a metric that quantifies the accuracy of probabilistic predictions. The Brier score compares the predicted probabilities to the actual outcomes (whether the team won or not). The trained models and feature scalers are then serialized using joblib, allowing them to be saved to disk for future use without needing to retrain them.

The architecture is designed to handle large datasets efficiently by separating concerns into distinct modules, ensuring that each part of the system is easily maintainable and scalable. Error handling is incorporated throughout the

system, especially when dealing with database access, data integrity, and model training. Overall, the system is flexible, allowing for future expansion with additional models, hyperparameters, or improvements to the data pipeline.

8 Paper Rubric

References

[1] Gao, Zijiana. *Random forest model identifies serve strength as a key predictor of tennis match outcome*. 2021. URL: https://content.iospress.com/articles/journal-of-sports-analytics/jsa200515.

[2] SeoYeong-Jin. *A Win/Lose prediction model of Korean professional baseball using machine learning technique*. 2019. URL: https://koreascience.kr/article/JAKO201910537995254.page.

[3] Xu, Davide. *Riot Games explains Win Probability for LoL Worlds 2023*. 2023. URL: https://www.esports.net/news/lol/riot-games-explains-win-probability-for-lol-worlds-2023/.