LEAGUE OF LEGENDS

PROJECT CHALLENGER

PREDICTING THE OUTCOME OF A LEAGUE OF LEGENDS MATCH AFTER 15 MINUTES

Student: Pieter School (1666114)
Course: Data Science tools and techniques (portfolio)
Teacher(s): Oliver Ntenje, Witek ten Hove et al.
Date: 8-1-2025

TABLE OF CONTENTS

INTRODUCTION

This report presents the results of Project Challenger, which focuses on predicting League of Legends match outcomes based on game state data at the 15-minute mark. The goal of the project is to build a machine learning model that can analyze in-game metrics, such as gold difference, kills, minion farming, and vision control, to predict whether a team will win or lose the match.

The objective of this project is to build a predictive model that forecasts match outcomes at the 15-minute mark in League of Legends. Success will be evaluated using metrics such as accuracy, precision, recall, and ROC-AUC score.

A deep neural network (DNN) was chosen for this task due to the complex, non-linear relationships between features in the data. The project follows the CRISP-DM framework, covering each step from understanding the problem to evaluating the model's performance. The main objective is to find patterns in historical data that can provide reliable predictions, helping players, coaches, and analysts make better decisions during matches.

The model's development involved hyperparameter tuning to optimize accuracy, balancing the need for performance with the risk of overfitting. The results highlight the potential of AI tools to improve strategic planning in esports by providing data-driven insights.

This project draws inspiration from the work shared by GitHub user artoria-dev. Their contributions have provided valuable insights and guidance for the development of this work.

CONTEXT AND BACKGROUND INFORMATION

Before we dive into the project, this section provides background information on what League of Legends is, how the game is played, and most importantly, how to win. It also explains the importance of predicting match outcomes, and what role AI/Machine Learning technologies can play in this.

WHAT IS LEAGUE OF LEGENDS?

League of Legends (2025) is a competitive, team-based strategy game where two teams of five players compete to destroy the opposing team's Nexus, a core structure in their base. Each player controls a champion, a unique character with specialized abilities that contribute to their team's success.

The game is played on Summoner's Rift, a map divided into three lanes (Top, Mid, and Bot) and a jungle area filled with neutral monsters. Teams must work together to push through these lanes, destroying enemy turrets and inhibitors to gain access to the enemy's Nexus.

Figure 1: Map of Summoner's Rift

Players earn gold and experience by defeating minions, monsters, and enemy champions. Gold is used to purchase items that enhance a champion's abilities, while experience allows champions to level up and unlock stronger skills.

There are five primary roles in the game:

1. Top Lane: Focuses on durable champions who excel in one-on-one combat and can soak damage.
2. Mid Lane: Home to versatile, high-damage champions who control the center of the map.
3. Jungle: Roams the map, taking down neutral monsters and assisting lanes with surprise attacks.
4. Bot Lane (ADC): Deals sustained damage from a distance, crucial for team fights.
5. Support: Protects and assists the ADC, providing utility and vision control.

HOW DO YOU WIN?

The key to winning is strategic teamwork. Teams must secure objectives like Dragons and Baron Nashor, which provide powerful buffs that give an advantage in fights. Players must also coordinate their efforts to push lanes, destroy enemy structures, and capitalize on enemy mistakes.

A successful team balances offense and defense, makes calculated plays, and adapts to the evolving state of the game. Victory is achieved when a team successfully breaks through the enemy's defenses and destroys their Nexus, securing a win.

Ultimately, League of Legends is a game of strategy, skill, and coordination, where players must think both individually and as a team to outmaneuver their opponents.

THE IMPORTANCE OF PREDICTING MATCH OUTCOMES IN LEAGUE OF LEGENDS

League of Legends is a complex and fast-paced strategy game that requires players to make quick decisions based on constantly changing situations. Unlike traditional sports, where physical performance and well-established tactics tend to dictate outcomes, League of Legends games are highly dynamic. Victory depends on a wide range of factors, from champion picks and early-game plays to macro-level strategies and team coordination. This makes predicting match outcomes a difficult task, especially in the mid-game, when both teams have developed their initial strategies but the game is still open to big momentum swings.

Most players and analysts rely on experience and intuition to guess which team is ahead at any given point in the game. This approach can be inconsistent and biased, especially in high-pressure situations. For example, a team that appears to be winning at the start might lose due to poor decision-making later, while a team that seems behind can make a comeback with better strategy and execution. Given this unpredictability, having a data-driven, objective way to predict match outcomes could be incredibly valuable.

WHY THE 15-MINUTE MARK MATTERS

One key point in League of Legends matches is the 15-minute mark, which marks the transition from the early game (focused on lane matchups and small skirmishes) to the mid-game, where team coordination and map control become more important. By this point in the game, a lot of information is already available: things like gold differences, kill scores, turret progress, and objective control all start to hint at which team might be in a stronger position.

But just because a team has an early advantage doesn't guarantee they'll win. The mid and late game are full of uncertainty, and teams can recover from early setbacks by adjusting their strategies. That's what makes predicting match outcomes at the 15-minute mark so interesting: the game isn't over, but trends are starting to emerge.

This makes the 15-minute mark a critical turning point, where an AI model could provide valuable insights into how the rest of the game is likely to play out.

THE VALUE OF USING AI/ML TO PREDICT OUTCOMES

Applying AI and Machine Learning to predict match outcomes based on in-game data at the 15-minute mark could offer more accurate and objective predictions than human intuition. By analyzing patterns from thousands of past games, an AI model could spot trends and key performance indicators (KPIs) that are often overlooked by players and analysts.

This type of tool could be useful for several groups within the League of Legends community:

1. Players and Teams
   Real-time predictions can help teams adjust their strategies mid-game. For example, if the AI predicts that the team is at a disadvantage, they might choose to play more defensively or focus on securing specific objectives to regain control.

2. Coaches and Analysts
   Coaches could use the model to identify which early-game factors are most likely to lead to a win or a loss. This could help improve their training routines and game preparation by focusing on specific areas that impact outcomes the most.

3. Esports Broadcasters and Fans
   For viewers, real-time predictions could make watching matches more engaging by providing win probabilities based on the current state of the game. This would give fans more context and insight into how teams are performing.

4. Game Developers (Riot Games)
   Riot Games could use this type of analysis to better understand what makes a match balanced. If certain champions, items, or strategies are shown to have overwhelming influence on match outcomes, developers could adjust the game to make it more fair and competitive.

WHY THIS PROJECT MATTERS

This project addresses one of the biggest challenges in esports analytics: turning real-time game data into meaningful insights. Predicting the outcome of a League of Legends match at any point in the game is difficult because there are so many variables to consider — champion picks, player performance, objective control, gold difference, etc. The sheer complexity of the game makes it nearly impossible to accurately predict outcomes based on human intuition alone.

By introducing AI-driven analysis, this project could make those predictions faster, more accurate, and more objective. It has the potential to change the way matches are understood and analyzed, not just for players and teams, but for the entire League of Legends ecosystem.

In a broader sense, this type of project shows how AI/ML models can be applied in fast-paced, complex environments to provide valuable insights. Whether in esports, traditional sports, or other industries, the ability to process data in real-time and make accurate predictions has huge potential to improve decision-making and optimize performance in competitive scenarios.

PROJECT BREAKDOWN

Now that we have explored the context of the project, this section will break it down using the **CRISP-DM (Cross Industry Standard Process for Data Mining)** model.

BUSINESS UNDERSTANDING

The goal of this project is to develop a machine learning model that can predict the outcome of a League of Legends match based on game state data collected at the **15-minute mark**. This prediction can provide valuable insights for players, coaches, analysts, and esports organizations, helping them make more informed decisions during matches.

The project addresses the following business questions:

- **Can the game outcome be predicted reliably at the 15-minute mark?**
- **Which in-game metrics (gold difference, kill scores, objectives, etc.) are the most important predictors?**
- **How can these predictions improve decision-making for players and analysts?**

Success will be measured by the model's ability to provide accurate and reliable win probability predictions **(aiming for 70-80% accuracy)** , along with identifying key factors that influence match outcomes.

DATA UNDERSTANDING

To make accurate predictions, it is important to understand the type of data available in League of Legends and its relevance to match outcomes. The data will be sourced from historical match records, through public APIs provided by Riot Games.

At the **15-minute mark**, some key metrics that influence match outcomes include:

- **Gold difference between teams** — a major indicator of a team's advantage.
- **Kill scores** — representing player eliminations, which can affect morale and strategy.
- **Minions killed (CS)** — reflecting how efficiently players accumulate gold through farming.
- **Tower and objective control** — indicating how well teams control the map and key resources.
- **Vision control (wards placed/cleared)** — essential for map awareness and planning.

Understanding these metrics in context is critical, as they reveal trends in match progression and help determine which features to prioritize in the model.

DATA PREPARATION

Before modeling, the raw data must be prepared to ensure it is clean, consistent, and structured. This involves several steps:

1. **Cleaning the data** to handle missing values, inconsistent formatting, and outliers.
2. **Feature engineering** to create new variables that may be more predictive, such as gold difference per minute or net kills.
3. If necessary, **encoding categorical variables** into numerical values so they can be used in machine learning algorithms.
4. **Splitting the data** into training, validation, and test sets to evaluate model performance and avoid overfitting.

Effective data preparation ensures that the model has a solid foundation to make reliable predictions.

MODELING

For this project, a deep neural network (DNN) is used to predict match outcomes based on game state data at the 15-minute mark. League of Legends data is highly complex, with many features interacting in non-linear ways, which makes DNNs a good choice over simpler models like logistic regression. The focus of this phase is to tune hyperparameters to improve the model's accuracy and generalization.

Using DNNs makes sense because they can learn hidden patterns in the data that are not immediately obvious. Tuning these hyperparameters helps the model handle non-linear relationships and improve predictions based on features like gold difference, kill score, and vision control. The final goal is to find a balance between accuracy and overfitting, ensuring the model performs well across different match scenarios.

EVALUATION

The model is evaluated using a combination of accuracy, loss, precision, recall, and ROC-AUC score to ensure a comprehensive assessment of its performance.

- Accuracy measures the overall correctness of predictions.
- Loss tracks the error during training and validation, with lower values indicating better performance.
- Precision evaluates the proportion of correctly predicted wins out of all predicted wins.
- Recall measures the model's ability to correctly identify actual wins.
- ROC-AUC Score assesses how well the model distinguishes between winning and losing teams across various thresholds.

These metrics are chosen to provide both a general performance overview (accuracy, loss) and insight into specific prediction quality (precision, recall, and ROC-AUC), ensuring the model is effective and reliable across different match scenarios.

DEPLOYMENT

While deployment is outside the scope of this project, it's important to understand how the model could be applied in a real-world setting. The model could be deployed as a web dashboard to display real-time win probabilities or as an API service that integrates with existing esports tools.

Key considerations for deployment would include:

- Real-time data handling: The model would need to process live match data efficiently.
- User-friendly interface: Ensuring the tool is accessible for players, analysts, and coaches.
- Prediction explanations: Providing insights into how the model generates predictions to build trust and usability.

Deployment would allow the project to transition from a theoretical exercise to a practical tool used in esports analytics.

MAINTENANCE

The model would require ongoing maintenance to remain relevant as League of Legends is regularly updated and game balancing changes are being made.

Maintenance would involve tracking the model's performance, retraining it with new match data to reflect changes in the game meta, and incorporating user feedback to enhance its accuracy and usability over time.

STEP I: ACQUIRING THE DATA

There are two ways to acquire the data that will be used for this project:

- Using pre-made datasets
- Retrieving the data using the Riot API

While using pre-made datasets will save a lot of time, since League of Legends changes a lot within a short period of time these datasets could be outdated. This is why the data will be gathered using the Riot Games API.

However, the access rates for the API are extremely limited:

- 20 requests every 1 seconds
- 100 requests every 2 minutes

Another limitation is that the API key that can be generated on the Riot Developer Portal is only valid for 24 hours. Still, this means that at a maximum around 72,000 matches can be extracted that can be used to train the model. This sample size should be big enough.

Due to the way the data is stored, multiple steps need to be taken to get to the match data:

1. Get the player's Summoner ID
   Use the Summoner-V4 API to retrieve the player's summoner ID by providing their summoner name and region.
2. Get the player's PUUID (Player Universally Unique Identifier)
   Use the Summoner-V4 API to convert the summoner ID into a PUUID. The PUUID is necessary to retrieve match history and is a global, consistent identifier across regions.
3. Retrieve the match IDs
   Use the Match-V5 API to get a list of match IDs from the player's match history using their PUUID.
4. Access detailed match data
   Use the Match-V5 API with a match ID to retrieve detailed game data.
5. Retrieve the timeline data
   To access data at the 15-minute mark, use the timeline endpoint with the match ID to get in-game events and player stats at specific timestamps.

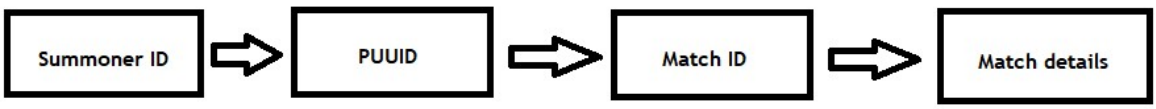

Figure 2: workflow for retrieving match details

CHOOSING THE MATCHES

To ensure that the model is able to predict the outcome of the majority of the games, the data needs to represent the rank distribution. The website League of Graphs, which tracks a variety of statistics, including rank distribution. Below is a plot of said distribution:

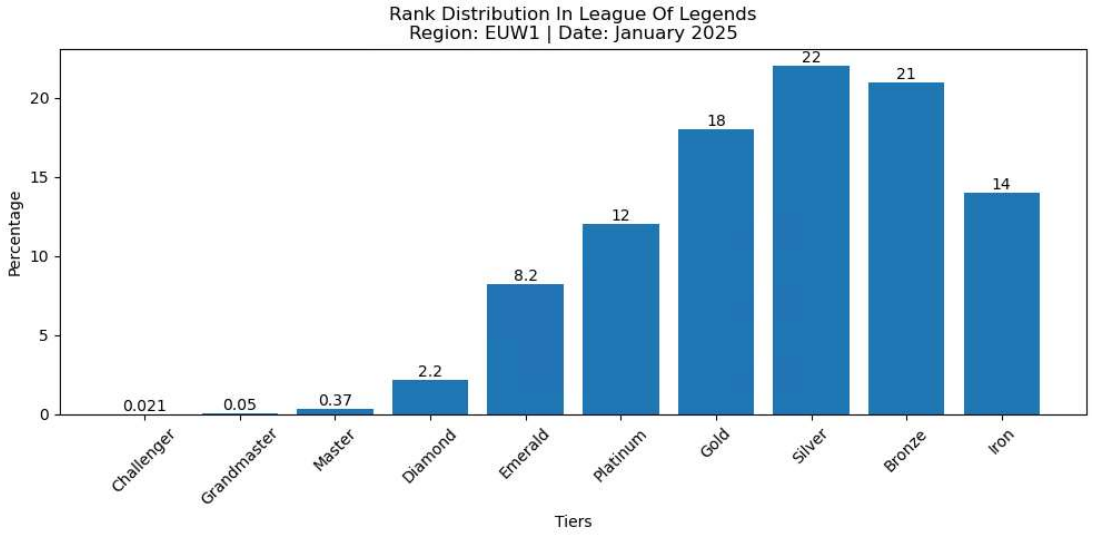

Figure 3: Rank distribution in League of Legends, Region: EUW1 Date: January 2025

Note: the Riot Games API allows for a maximum of 100 games tracked per player. So, if the dataset needs to have an accurate representation of the rank distribution, multiple players per rank need to be tracked.

Since the percentage of challenger, grandmaster and master players is so small, and they do need to be included in the model as well, the data should consist of the following amount of games per tier:

Table 1: Amount of games required

Rank	Players tracked	Games tracked	Percentage (%)
Challenger	7	700	1
Grandmaster	7	700	1
Master	7	700	1
Diamond	15	1500	2
Emerald	57	5700	8
Platinum	84	8400	12
Gold	126	12,600	18
Silver	154	15,400	22
Bronze	147	14,700	21
Iron	98	9800	14
Total	702	70,200	100

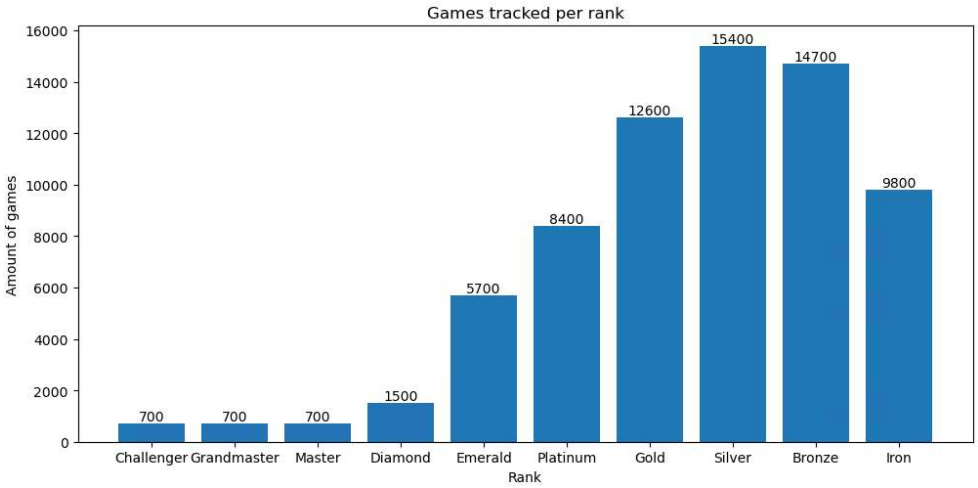

Figure 4: Final distribution of games tracked per rank

DATA LIMITATIONS AND POTENTIAL BIAS IN THE COLLECTED DATA

The dataset used in this project may be affected by differences in player behavior across ranks. In lower ranks, players are more likely to make mistakes, and higher-ranked players tend to capitalize on those mistakes more effectively. As a result, game metrics such as gold difference, experience, and kills may be less predictive in lower-ranked matches, where unexpected outplays and comebacks are more common. In contrast, matches in higher ranks are generally more consistent, making it easier to predict outcomes based on game state data.

Another potential bias comes from collecting games across multiple patches. Riot Games releases patches every two weeks to adjust champions, items, and other game mechanics. While many patches have minimal impact, some, like the Durability Patch (Patch 12.10), caused significant changes by making defensive items more cost-effective and reducing overall damage output. These changes likely altered the importance of certain features in predicting match outcomes, which may affect the model's performance on pre- and post-patch data.

To improve the model's generalizability, future datasets should aim to include more high-ranked matches and ensure that data is collected from consistent patches to reduce variability in the predictions.

OVERVIEW OF RETRIEVED DATA

The match data includes a lot more information than what is necessary to train the model. This is why for this project the following values will be gathered:

Table 2: Required features for training

Feature	Description
winningTeam	The team that won the match (blue or red).
blueTeamWardsPlaced	Number of wards placed by the blue team for vision control.
blueTeamKills	Total kills made by the blue team.
blueTeamTotalJungleMinionsKilled	Number of jungle minions killed by the blue team.
blueTeamTotalMinionsKilled	Total number of lane minions killed by the blue team.
blueTeamAvgLevel	The average level of all champions on the blue team.
blueTeamCsPerMinute	Creep Score (CS) per minute for the blue team.
blueTeamGoldPerMinute	Amount of gold earned per minute by the blue team.
blueTeamWardsDestroyed	Number of enemy wards destroyed by the blue team.
redTeamWardsPlaced	Number of wards placed by the red team for vision control.
redTeamKills	Total kills made by the red team.
redTeamTotalJungleMinionsKilled	Number of jungle minions killed by the red team.
redTeamTotalMinionsKilled	Total number of lane minions killed by the red team.
redTeamAvgLevel	The average level of all champions on the red team.
redTeamCsPerMinute	Creep Score (CS) per minute for the red team.
redTeamGoldPerMinute	Amount of gold earned per minute by the red team.

STEP 2: DATA EXPLORATION AND CLEANING

This section will cover the steps undertaken to explore the data and, if necessary, clean it.

INITIAL LOAD AND CLEANING

At first glance, the data looks good. No values are missing and they look like expected.

Below is an overview of each variable's minimum and maximum values:

Table 3: descriptive statistics

Column Name	Minimum	Maximum	Mean
winningTeam	0.00	1.00	0.52
blueTeamWardsPlaced	6.00	644.00	53.60
blueTeamKills	0.00	31.00	11.69
blueTeamTotalJungleMinionsKilled	0.00	186.00	65.81
blueTeamTotalMinionsKilled	72.80	319.80	240.43
blueTeamAvgLevel	5.98	9.44	8.48
blueTeamCsPerMinute	5.79	26.36	20.42
blueTeamGoldPerMinute	862.84	1888.92	1357.16
blueTeamWardsDestroyed	0.00	44.00	5.47
redTeamKills	0.00	27.00	11.34
redTeamTotalJungleMinionsKilled	0.00	186.00	65.95
redTeamTotalMinionsKilled	148.70	320.30	239.83
redTeamAvgLevel	7.46	9.40	8.47
redTeamCsPerMinute	13.07	26.97	20.37
redTeamGoldPerMinute	1025.54	1732.79	1347.22

The maximum value for BlueteamWardsPlaced is unusually high, so further analysis is needed to look if it's a common occurrence or an exception. Below is a table that counts the number of wards placed for the blue team, and how often this number is within a certain range:

Table 4: Number of wards placed

Number of Wards Placed	Count
800+	0
700-800	0
600-700	1
500-600	1
400-500	17
300-400	46
200-300	96
100-200	156
0-100	2485

Since it's a rare occurrence and it's mathematically possible to place this number of wards in a match, no further action is needed to resolve this.

EXPLORATORY DATA ANALYSIS

In this section the data will be explored, taking a look at the main characteristics of the dataset and comparing certain values.

Below is a pie chart displaying the win rate for each team:

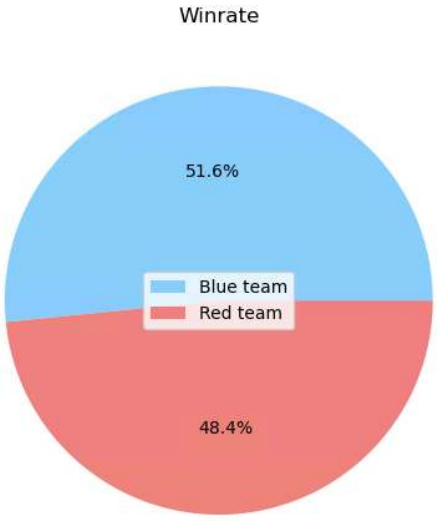

Figure 5: Winrates for each team

The actual win rate for the blue team is 51.8% as of January 2025 (League of Graphs, 2025). This means that the data is slighty off, but it's acceptable.

Next, a heatmap will display the correlation between different features. A higher correlation means that this feature is likely to influence the other. This can be used to see what feature influences the chances of winning for a team. In the heat map the variables of the blue team are used, since the red team wins if the blue team does not.

(note: for the heatmap the features that will be used in the model have been engineered already. A further explanation of these features will follow after.)

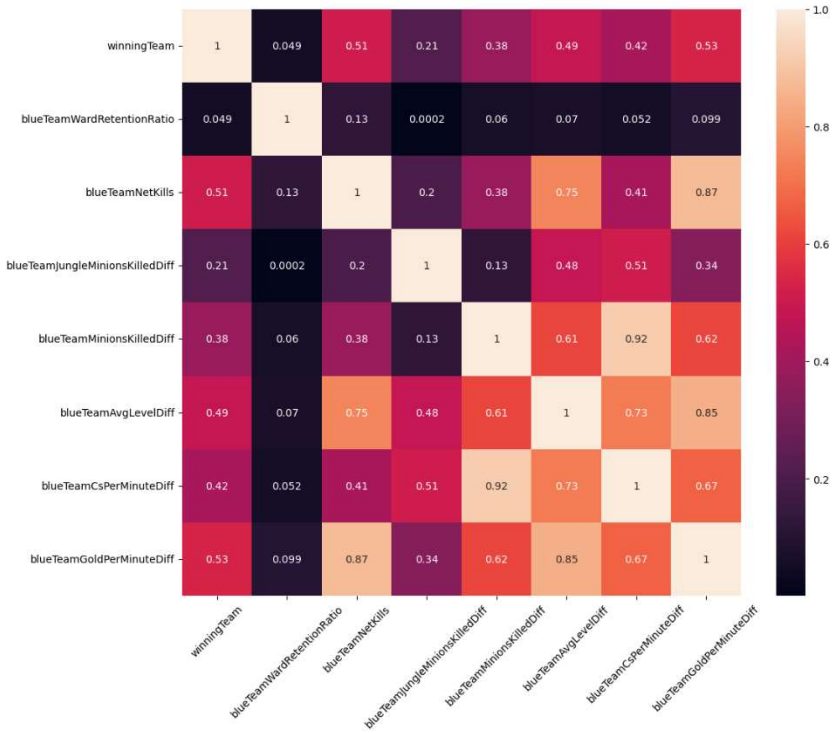

Figure 6: Heatmap of selected features

Table 5: Features and their correlations

Feature	Highest Correlation With	Lowest Correlation With
winningTeam	GoldPerMinute	WardRetentionRatio
WardRetentionRatio	NetKills	JungleMinionsKilled
NetKills	GoldPerMinute	WardRetentionRatio
JungleMinionsKilled	CsPerMinute	WardRetentionRatio
MinionsKilled	CsPerMinute	WardRetentionRatio
AvgLevel	GoldPerMinute	WardRetentionRatio
CsPerMinute	MinionsKilled	WardRetentionRatio
GoldPerMinute	NetKills	WardRetentionRatio

The table shows that GoldPerMinute is the most important factor related to winningTeam, which makes sense since gold is essential for item purchases that give teams an advantage. Gold is primarily earned through kills and farming, as seen in the strong correlations with NetKills and CsPerMinute. This indicates that consistent farming and securing kills while

minimizing deaths are key to increasing gold income and improving the likelihood of winning a match.

Interestingly, WardRetentionRatio has the weakest correlation with most features, suggesting that vision control may have an indirect impact on game outcomes. This doesn't mean it's unimportant though—it likely helps teams make better decisions, even if it doesn't directly affect core metrics like kills or gold.

In summary, GoldPerMinute, NetKills, and CsPerMinute are the most important metrics for predicting match outcomes. Teams that focus on farming efficiently and securing kills tend to perform better overall.

STEP 3: FEATURE ENGINEERING

This section provides an overview of the features that will be used to train the model. These features are meant to simplify the model for learning.

The following features, as used above in the heatmap will be used in the model:

Table 6: Engineered features

Feature	Calculation
blueTeamWardRetentionRatio	(blueTeamWardsPlaced - redTeamWardsDestroyed) / blueTeamWardsPlaced
blueTeamNetKills	blueTeamKills - redTeamKills
blueTeamJungleMinionsKilledDiff	blueTeamTotalJungleMinionsKilled - redTeamTotalJungleMinionsKilled
blueTeamMinionsKilledDiff	blueTeamTotalMinionsKilled - redTeamTotalMinionsKilled
blueTeamAvgLevelDiff	blueTeamAvgLevel - redTeamAvgLevel
blueTeamCsPerMinuteDiff	blueTeamCsPerMinute - redTeamCsPerMinute
blueTeamGoldPerMinuteDiff	blueTeamGoldPerMinute - redTeamGoldPerMinute

These variables capture differences in performance between the blue team and red team at the 15-minute mark, which is exactly the information the model needs to predict the match outcome.

Now that all this is set up, the model can be trained.

STEP 4: TRAINING THE MODEL

This section covers the training process of the model. As described in the project breakdown, a deep neural network (DNN) will be used to predict match outcomes based on game state data at the 15-minute mark. The model will take features such as gold difference, kill score, minion farming efficiency, and vision control to learn patterns that indicate whether a team is likely to win or lose the match.

A critical part of this process involves setting and tuning hyperparameters, which are the predefined settings that control how the model learns. Unlike model parameters, which the model adjusts during training, hyperparameters must be defined before training begins and directly affect the model's performance. These include settings such as the optimizer type, number of neurons in each layer, activation functions, batch size, and number of epochs.

Using the prepared dataset, the following hyperparameters will be used as a baseline for the model:

Table 7: Baseline hyperparameters

Hyperparameter	Value	Explanation
Number of Layers	2	Provides enough depth to capture complex relationships without overfitting.
Neurons per Layer	32	Balanced size for capturing features without overloading the model.
Activation Function	ReLU	Helps prevent vanishing gradients and speeds up convergence.
Optimizer	Adam	Efficient and widely used for deep learning models.
Learning Rate	0.001	Common starting point for the Adam optimizer.
Loss Function	Binary Cross-Entropy	Best suited for binary classification tasks like predicting win/loss.
Batch Size	20	Helps with generalization and allows faster convergence.
Epochs	50	Gives the model enough time to learn while avoiding overfitting.
Regularization	L2(0.01)	Prevents overfitting by penalizing large weights.

HYPERPARAMETER TUNING

To ensure that the model is optimized, GridSearch is used to systematically test different combinations of hyperparameters and identify the combination that yields the best performance on the validation data. This process ensures that the model generalizes well to new data and is not overfitted to the training set.

The table below shows the hyperparameter values selected for the GridSearch process:

Table 8: GridSearch parameters

Optimizer	Activation Function	Neurons	Batch Size	Epochs
Adam	ReLU	16	10	50
RMSprop	Tanh	32	20	100
Sigmoid	Sigmoid	64	50	150
		128		

To further enhance the training process, EarlyStopping is implemented to prevent the model from overfitting or continuing to train beyond the point of improvement. EarlyStopping monitors the validation loss and stops training when no further improvements are observed, thereby saving computational resources and ensuring the model remains efficient.

After running the GridSearch, the best hyperparameters are the following:

- Optimizer: Adam
- Activation function: Sigmoid
- Neurons: 32
- Batch size: 20
- Epochs: 150

(note: the maximum amount of epochs is set to 150, while this might be the best option out of these three the optimal amount may be higher)

Expanding the GridSearch to include regularization and number of layers would result in training too many models, increasing the time and computational resources required. To avoid this, instead of running a full GridSearch with these additional hyperparameters, a simpler approach will be used. A few models will be trained with different combinations of regularization and layers to find which setup works best.

The table below shows the configurations that will be tested:

Table 9: Extra hyperparameters

Regularization	Layers
L1(0.001)	1
L1(0.01)	2
L1(0.1)	3
L2(0.001)	
L2(0.01)	
L2(0.1)	

Regularization helps the model avoid overfitting by adding a penalty to the model's weights. Both L1 regularization (which pushes some weights to zero) and L2 regularization (which reduces the size of all weights) will be tested to see which performs better.

It is true that the best hyperparameters found through GridSearch could change when additional settings like regularization and number of layers are considered. However, testing every possible combination would increase the number of models to train by a lot. To be specific, with 216 combinations from the initial GridSearch, multiplied by 6 regularization options and 3 layer configurations, this would require training 3888 models.

Since training 216 models already took a **really long time**, training 3888 models would most likely take a couple weeks. This is clearly not practical.

Instead, a simpler approach is taken by manually testing a few configurations for regularization and layers, which keeps training time manageable while still improving the model’s performance.

Plotted in a graph, the results after an hour look like this:

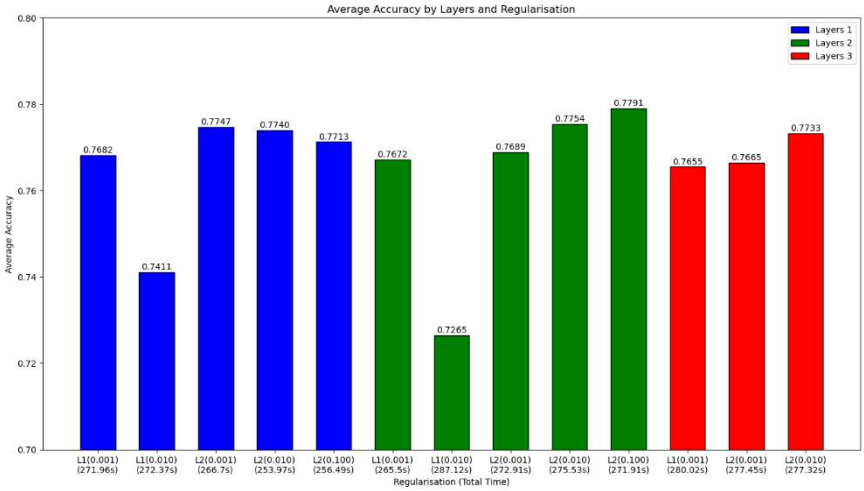

Figure 7: Results of layer and regularization testing

Judging from these results, the combination of 2 layer(s) and L2(0.100) gives the highest accuracy and will be used to train the final model.

FINAL MODEL

After tuning all the hyperparameters, the final model will use the following hyperparameters:

Table 10: Final model hyperparameters

Hyperparameter	Value
Number of Layers	2
Neurons per Layer	32
Activation Function	Sigmoid
Optimizer	Adam
Learning Rate	0.001
Loss Function	Binary Cross-Entropy
Batch Size	20
Epochs	150
Regularization	L2(0.1)

With these hyperparameters the model will be trained and the results will be covered in the next section.

STEP 5: MODEL EVALUATION

This section evaluates both models using a combination of accuracy, loss, precision, recall, and ROC-AUC score.

ACCURACY REPORTS

Derived directly from the code, upon training the model for the first time the baseline model has an accuracy of **74.11%**, and the final model achieved an accuracy of **76.48%,** making it slightly higher than the baseline model. Since the aim was between 70% and 80% accuracy, these results are really acceptable.

Below are the accuracy reports for both models:

Table 11: Classification report baseline model

Class (Label)	Precision	Recall	F1-Score	Support
0	0.82	0.68	0.74	229
1	0.68	0.82	0.74	192
Accuracy			0.74	421
Macro Avg	0.75	0.75	0.74	421
Weighted Avg	0.75	0.74	0.74	421

Table 12: Classification report final model

Class (Label)	Precision	Recall	F1-Score	Support
0	0.83	0.71	0.77	229
1	0.71	0.83	0.76	192
Accuracy			0.76	421
Macro Avg	0.77	0.77	0.76	421
Weighted Avg	0.77	0.76	0.77	421

What do the columns mean?

Precision: The percentage of correct positive predictions out of all predictions for that class (How precise is the model when it predicts a certain class?).

Recall: The percentage of actual samples correctly predicted as that class (How well does the model catch all instances of a class?).

F1-Score: The harmonic mean of Precision and Recall, balancing both metrics (A score that considers both false positives and false negatives).

Support: The number of true samples in each class (How many samples actually belong to each class?).

LOSS

Here is a graph that shows the loss plotted throughout the training process. Note the loss function converging to a minimum. Since the loss only slightly decreases after the first epochs, time-wise it might be better to run less epochs next time the model is trained.

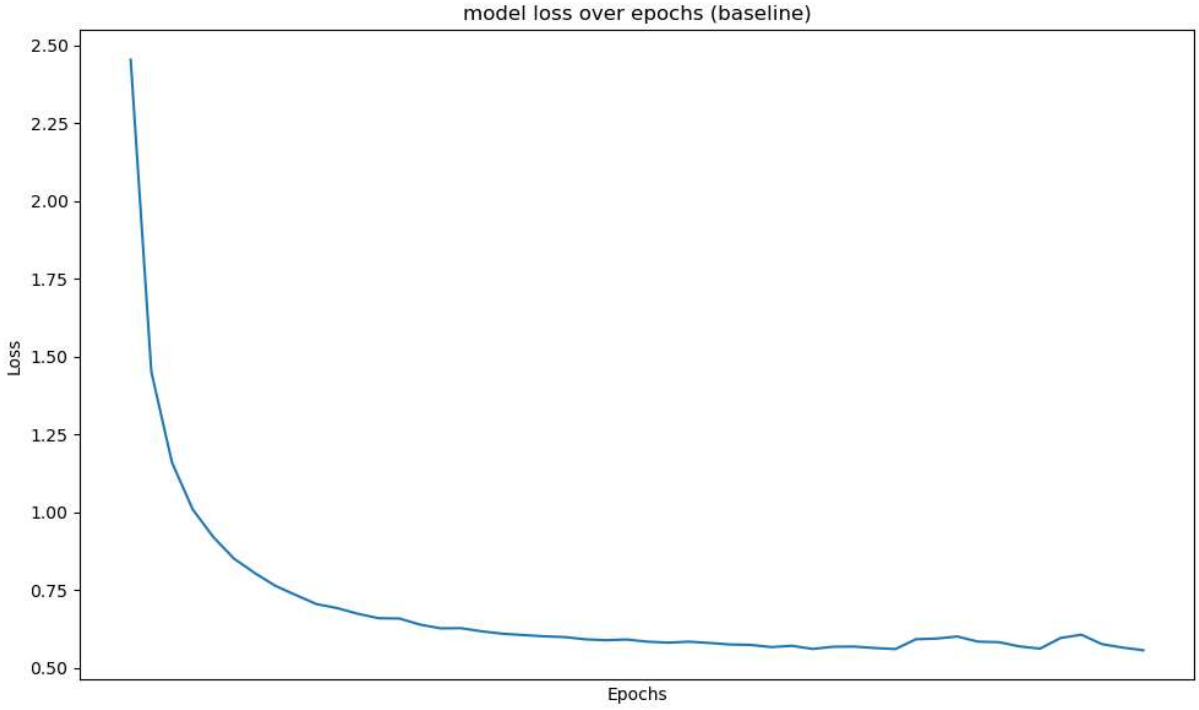

Figure 8: Model loss over epochs (baseline model)

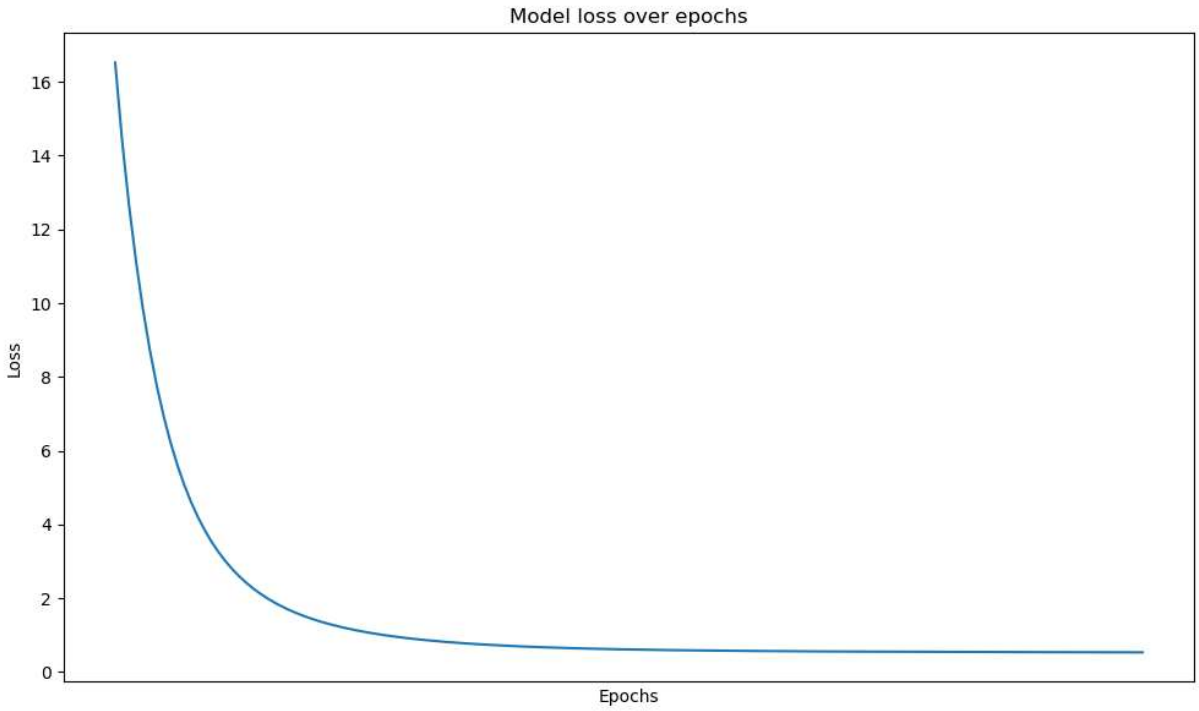

Figure 9: Model loss over epochs (final model)

ROC AND AUC CURVE

The next and final form of evaluation is a ROC/AUC curve.

The ROC (Receiver Operating Characteristic) curve shows how well the model distinguishes between two classes across different thresholds. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR).

The AUC (Area Under the Curve) score summarizes the ROC curve as a single number:

- AUC = 1.0: Perfect model.
- AUC = 0.5: Random guessing.
- AUC > 0.7: Indicates a good model.

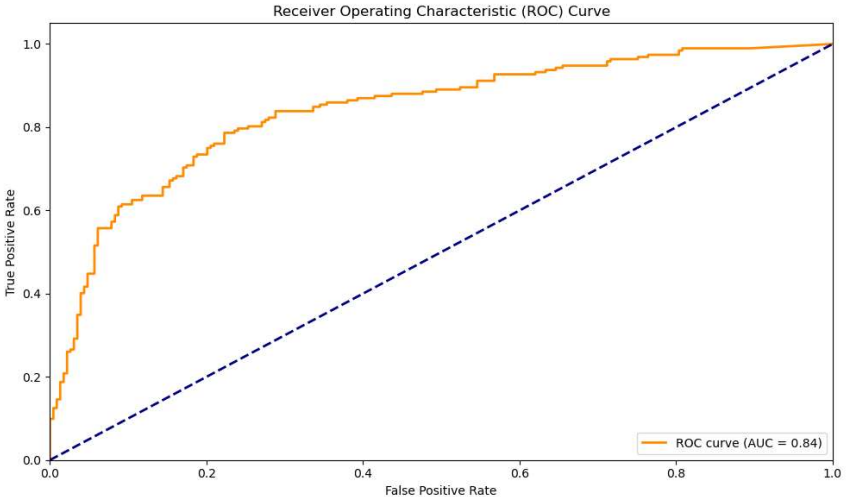

Figure 10: ROC-curve baseline model

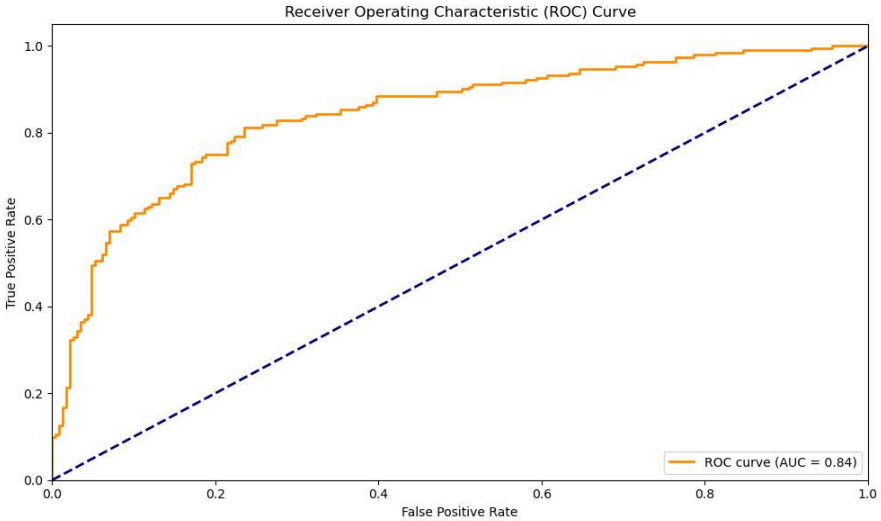

Figure 11: ROC-curve final model

Both achieve an AUC-score of 0.84, which means that the models performed well on this evaluation metric.

Overall both models performed really well, with their accuracy reaching the aim of 70-80%.

CONCLUSION

Project Challenger shows how machine learning can be used to predict League of Legends match outcomes based on game data from the 15-minute mark. The model focuses on key metrics like gold difference, kills, and vision control to provide insights at a critical point in the game.

The project highlights the importance of hyperparameter tuning in improving model performance. Starting with a GridSearch, additional tuning of regularization and layers helped refine the model for better accuracy and generalization. The final model successfully predicts win probabilities across various match scenarios.

This project demonstrates the potential for AI-driven tools to support decision-making in esports. Providing real-time, data-based predictions can be valuable for players, coaches, and analysts. Future work could explore more features, larger datasets, or different machine learning methods to further improve accuracy.

BIBLIOGRAPHY

artoria-dev. (n.d.). *Data Science LoL*. GitHub. Retrieved January November 22nd, 2024, from https://github.com/artoria-dev/data-science-lol

League of Graphs. (n.d.). *Blue Team vs. Red Team - League of Legends*. Retrieved December 17, 2024, from https://www.leagueofgraphs.com/stats/blue-vs-red/euw

League of Legends. (n.d.). *League of Legends, how to play*. Retrieved December 17, 2024, from https://www.leagueoflegends.com/en-au/how-to-play/