Decisions That Matter: Real-Time Win Possibility Prediction for League of Legends Players

Luqman Afiq (220650645)
May 2025
BSc Computer Science (Software Engineering)
Supervisor – Dr. Daniel Sun

Word count : 15,943 words

Abstract

For millions of players, League of Legends is more than just a game—it's a test of discipline, decision-making, and constant improvement. In the increasingly data-driven landscape of multiplayer online battle arenas (MOBAs), League of Legends (LoL) stands at the forefront, demanding not only mechanical skill but strategic decision-making in real time. This project presents the development of a machine learning-based system that predicts in-game win possibility using live match data. Drawing inspiration from esports personality “Caedrel”, the system also introduces an AI coaching prototype that mimics this style of macro-level guidance. Using the Riot Games API and a dataset containing both professional and solo queue matches, multiple machine learning models were trained, with Neural Network achieving a test accuracy of 96%. A Gradio-based interface enables users to receive live updates on their current situations, alongside coaching advice based on objective control and current win possibility with precise weights. This system bridges the gap between professional-level analytics and casual gameplay, offering real-time, personalized feedback and laying the groundwork for AI-enhanced decision-making in competitive games. Critical findings include the precise quantification of objective values (inhibitors contributing 4.08% to win possibility, turrets 2.04%, Baron 1.47%, and dragons 0.72%), and identifying the time-dependent nature of these values. The results obtained may demonstrates significant practical value for players of all skill levels to improve their decision making skills.

Declaration

I declare that this dissertation represents my own work except stated otherwise .

Acknowledgements

I would like to mention this dissertation reflects not only my individual effort, but the overall support and guidance of many individuals to whom I am deeply grateful. I extend my sincere appreciation to my supervisor, Dr. Daniel Sun, whose expertise, insightful feedback, and unwavering patience were instrumental in guiding this research from concept to completion. Huge thanks to Mr. Stanly Wilson Palathingal for his consistent weekly guidance, constructive criticism, and methodological insights that significantly improved the analytical rigor of this work. On a personal note, I want to thank my family for their continued interest in my progress and their moral support despite the time difference between us. To my friends who patiently endured my endless discussions about win probabilities and League while sharing our living space. Finally, special recognition to Marc "Caedrel" Lamont, his LoL commentary I enjoyed motivates to me to start this projects.

Terminology

MOBA: Multiplayer Online Battle Arena

AI: Artificial Intelligence

LoL: League of Legends

API: Application Programming Interface

LCK: League of Legends Champions Korea

LTA: League of Legends Championship of The Americas

LEC: League of Legends EMEA Championship

Solo queue: Matchmaking alone without a team.

Elo: System that evaluates player’s skills.

Objectives: Key strategic elements on the Summoner's Rift map that provide advantages when secured, including Dragons, Baron Nashor, Turrets, and Inhibitors.

Macro-skill: Strategic decision-making ability focused on map-wide movements, objective control, and resource allocation.

Neural Network: A machine learning model inspired by the human brain, consisting of interconnected nodes organized in layers that process and transform input data to produce predictions.

Feature Engineering: The process of selecting, transforming, and creating variables from raw data to improve the performance of machine learning models.

Live Client API: An interface provided by Riot Games that allows access to real-time game state data during active League of Legends matches.

Gradio: An open-source Python library used to create customizable web interfaces.

Contents

Chapter I – Introduction

In the rapidly evolving landscape where multiplayer online battle arenas (MOBAs) dominate the gaming industry, League of Legends (LoL) stands as its crown jewel. With over 131 million active players globally and recognized as the most successful esports ecosystem of all time by viewership and prize pool, strategic decision-making is what separates victorious from losers on a battlefield called Summoner's Rift [1][2]. LoL is a competitive team-based strategy game in which two teams of five players, red side and blue side, compete to destroy the opposing team's base. With 170 available characters known as champions to choose from, each controlling unique abilities, League of Legends blends tactical decision-making, mechanical skill, and team coordination [3]. Players across all skill levels face the challenging task of comprehending the nuanced factors that contribute to victory.

While various factors contribute to victory in League of Legends — such as champion mastery, mental resilience, team composition synergies, and mechanical execution — macro-level game knowledge is what enables players to make impactful strategic decisions during the match. Generally speaking, the optimal strategy in any competitive scenario is to maximize your team's win probability [4]. Win probability serves as the ultimate metric for evaluating decisions and strategic choices as it integrates all other performance indicators such as gold, objectives (e.g., dragons and turret), and experience levels [5]. The development of accurate win possibility estimation stands to fundamentally transform how players approach strategic decision-making during matches.

Unlike strict mathematical probability which requires specific axiomatic properties and complete information, win possibility in this context acknowledges the inherent uncertainty and partial observability of competitive gameplay. Win possibility represents a quantitative assessment of a team's likelihood of victory based on the current game state. Formally, we define win possibility as a numerical value between 0 and 1 that indicates the estimated chance of winning given the present configuration of game variables (P(win|game_state)). Our win possibility function maps observable game state features to a bounded output through a combination of statistical weighting and machine learning techniques. This metric serves as a comprehensive indicator that integrates multiple performance factors including objective control, resource advantages, and temporal game progression. Hence, we will use term win possibility throughout the whole dissertation.

1.1 Motivation

Recognizing the importance of win probability metrics, Riot Games, LoL developer, has recently implemented a sophisticated machine learning-based Win Probability system for professional matches. As detailed in their Dev Diary, this system analyzes key in-game metrics including gold distribution, experience points, objective control, and player status to calculate win likelihood throughout a match. The model, developed using XGBoost and trained on professional matches since 2020, represents a significant advancement over traditional Gold Difference graphs by providing more intuitive interpretation. The 2023 League of Legends World Championship demonstrated the growing importance of win possibility metrics in professional play, with Riot Games partnering with AWS (Amazon Web Services) to implement a sophisticated win prediction system for broadcast viewers [6]. However, similar capabilities remain inaccessible to the broader player base, particularly in real-time during their own matches.

This project's relevance is further validated by the growing community interest in AI (Artificial Intelligence) applications for League of Legends optimization. Content creators and professional players are increasingly exploring how machine learning can enhance gameplay decisions, as evidenced in popular discussions like the "Broken by Concept" podcast episode "Using AI To Build Better Items" [7]. While these discussions primarily focus on item optimization, they highlight the broader appetite for data-driven decision support tools among the player base. By extending AI applications to real-time win possibility prediction and strategic coaching, this project addresses an emerging need recognized by both academics and the League of Legends community itself.

Although this marks a major milestone in esports analytics, the average player still lacks access to similar real-time strategic feedback. Existing performance analysis tools like OP.GG and Blitz.gg provide primarily retrospective insights rather than real-time guidance when critical decisions must be made. They rely heavily on historical professional match data, which often doesn't reflect individual player capabilities, and offer limited personalized recommendations during active gameplay. This gap between available data and actionable intelligence presents a significant opportunity for innovation at the intersection of artificial intelligence and competitive gaming. This project was also inspired by the work of Marc “Caedrel” Lamont, a former professional League of Legends player and current esports analyst [8]. His insightful, real-time commentary has become a gold standard in the community for breaking down macro decisions in an entertaining and impactful way. Motivated by this, the project aims to emulate his style of coaching through an AI-driven assistant that delivers live strategic guidance to players based on in-game data.

As a player myself, playing and watching gives different energy where soloq players only focus on kills more than professionals. The distinct playstyles between solo queue and professional play highlight the critical need for this system. While solo queue players often prioritize kills over objectives—mistakenly equating kill counts with victory conditions—professional matches demonstrate a more calculated approach focused on resource acquisition and map control. Solo queue players typically engage in risky skirmishes and chase kills throughout the game, lacking understanding of alternative win conditions. I learn from Caedrel co-streaming top tier leagues professional teams maintain discipline during the laning phase, minimizing early mistakes, and only commit to full team engagements when objective control cannot be secured through other means. This knowledge disparity creates the perfect opportunity for an AI coaching system that bridges these approaches, helping casual and competitive players understand the objective-focused strategies that truly determine match outcomes in League of Legends. By quantifying how objectives impact win possibility, the system transforms abstract professional knowledge into actionable insights accessible to the broader player base.

1.2 Aim and Objectives

The primary aim of this project is to develop an AI-powered system that enables data-driven decision-making for League of Legends players through real-time win possibility prediction and contextual coaching advice. Identifying the content will be the main research objectives of this paper. The following research questions will help in determining what qualifies as a good possibility. This investigation concentrates on a fundamental question: How accurately can machine learning models predict win possibility in League of Legends matches using objective control metrics?

1. **Data Integration and Preprocessing**: Create a comprehensive dataset by collecting and preprocessing a minimum of 10,000+ League of Legends matches from both solo queue and professional play across Europe West (EUW), North America (NA), and Republic of Korea (KR) regions by week 3, ensuring standardized features and implementing appropriate cleaning procedures.
2. **Objective Impact Analysis**: Determine the statistical significance of in-game objectives (dragons, barons, turrets) on match outcomes by calculating precise weight coefficients with statistical significance ($p < 0.05$) by week 5, quantifying how each objective contributes to win possibility.
3. **Model Development and Validation**: Develop and compare at least four machine learning algorithms (neural networks, random forests, gradient boosting, and SVM - Support Vector Machine) to predict match outcomes, achieving a minimum prediction accuracy of 70% on the test dataset with complete cross-validation by week 8.
4. **Live Client Integration**: Implement a system that connects to the League of Legends Live Client API (Application Program Interface) to extract real-time game state data with a maximum latency of 3 seconds and 95% uptime during active games by week 10.
5. **AI Coaching Module**: Create an AI-coaching advice system that generates at least three specific, actionable strategic recommendations based on current game state, accounting for game phase and win possibility, with recommendations validated by players by week 12.
6. **User Interface Development**: Design and implement a Gradio-based web interface that displays real-time win possibility with visual breakdowns of contributing factors and presents coaching advice in an intuitive format suitable for in-game reference by week 14.

Since the proposal submission, several modifications have been made to adapt to emerging challenges. Notably, the project exceeded the initial data collection target, since a match gave 10 players thus producing 10 times more data plus successfully processing 1,000 matches from professional data. Additionally, due to Riot Games' implementation of Vanguard anti-cheat system, the user interface had to be modified to function as a separate application that players access by switching between game and browser windows. After doing background researches, studies shows objectives are the most common strategies to improve likelihood of winning hence the adaptation of objective impact analysis in this study. The AI coaching module was also limited to text-based advice rather than voice output to maintain ethical standards and avoid potential conflicts with game integrity policies. Objective Impact Analysis was taken into consideration after reading multiple thesis mentioned in background research thus making objectives more of an importance in achieving a win.

1.3 Outline

This dissertation is organized into six chapters. Chapter I establishes the context and motivation for the research, defines the aim and objectives, and outlines the dissertation structure. Chapter II examines relevant literature in esports analytics, machine learning for game outcome prediction, and existing League of Legends analysis tools. Chapter III details the approach taken to achieve the stated objectives, including data collection and preprocessing methods, feature engineering techniques, model selection criteria, and system architecture design. Chapter IV provides technical details on how each component was implemented, including the data pipeline, machine learning models, Live Client API integration, and win possibility calculation algorithm. Chapter V presents the performance of various machine learning models, analyzes feature importance, evaluates the system's real-world performance, and discusses user feedback. Finally, chapter VI summarizes the dissertation with appendices that include additional material such as detailed model parameters, extended evaluation metrics, and user interface screenshots.

Chapter II – Background Review

Research on win prediction in League of Legends has already been conducted by several researchers. In this section, relevant findings and methodologies are discussed and presented.

2.1 Introduction to Esports Analytics

The emergence of esports as a global phenomenon has created new opportunities for data analytics applications. Competitive gaming generates vast amounts of structured data that can be analyzed to extract insights about player performance, strategic patterns, and outcome prediction. According to Hamari and Sjöblom [10], one of the most notable areas of media convergence today is esports, reflecting the increasing mainstream acceptance and commercial significance of competitive gaming since almost a decade ago.

Within this landscape, League of Legends occupies a uniquely prominent position. As documented by LOLEsports reveals official numbers for the Worlds 2024 Final, 50 million peak viewers including China, 6.7 million viewers outside China [11]. The highest total viewership since 2021 and an all-time high record for viewership outside of China. This scale of engagement has driven significant investment in analytics tools for both viewers and players, representing what Macey and Hamari [12] describe as the "professionalization of play" – where data-driven approaches increasingly inform competitive strategy.

2.2 League of Legends: Game Mechanics and Strategic Elements

League of Legends is a team-based multiplayer online battle arena game developed by Riot Games Inc. and released in 2009. The game has become one of the world's most popular esports, with professional leagues established across multiple regions and a World Championship that consistently draws millions of viewers.

Summoner's Rift: The Competitive Battlefield

The primary competitive environment in League of Legends is Summoner's Rift, a symmetrical map divided into two halves by a river running diagonally through its center. Each team occupies one corner of the map, where their base contains a Nexus—the primary objective that teams must destroy to win the match. The map's structure creates three distinct pathways or "lanes" (top, middle, and bottom) along which computer-controlled minions spawn and advance toward the enemy base. With the map change last year and addition of atakhan and void grubs [12], it is time to update strategy .

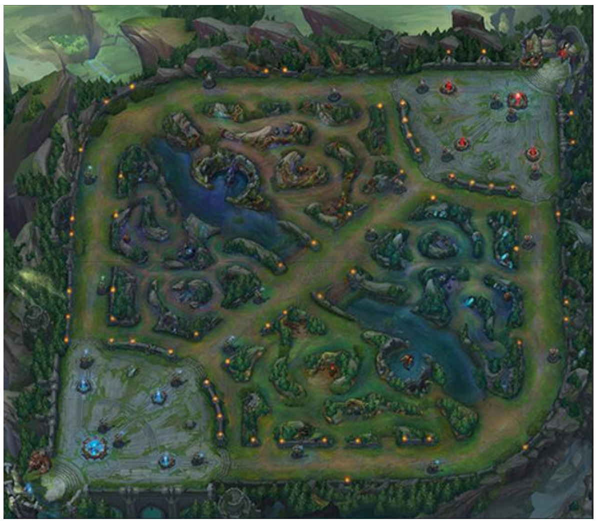


Figure 1: Summoner's Rift (https://wiki.leagueoflegends.com/en-us/Map)

Strategic depth emerges from the map's various elements:

- **Turrets**: Each lane contains three outer turrets and one inhibitor turret per team, plus two base turrets protecting each Nexus (new addition). These defensive structures must be destroyed sequentially within each lane, creating strategic chokepoints and territorial control mechanisms.
- **Jungle Areas**: Between the lanes lie neutral monster camps that provide various resources when defeated. The most significant neutral objectives include:
    - **Elemental Dragons**: Provide permanent team-wide stat bonuses based on their element (Infernal, Cloud, Ocean, Mountain, Hextech, or Chemtech)
    - **Void Grubs**: Six void grubs spawns, 3 at a time before Rift Herald
    - **Rift Herald**: Only one appears before Baron and can be used to drive and damage enemy structures
    - **Baron Nashor**: Spawns after 20 minutes and grants powerful buffs to the team that defeats it
    - **Elder Dragon**: Spawns 6 minutes after one team claims their fourth elemental dragon (Dragon soul).
- **Elemental Rift Transformations**: After the second dragon is slain, the map transforms based on that dragon's element, altering terrain and creating unique strategic considerations for the remainder of the match.

Understanding these map elements is crucial for interpreting win possibility models, as objective control (Tower Percentage and Number of Inhibitors) represents a significant predictor of match outcomes, as demonstrated by Novak et al. [13] and supported by the statistical analysis in the current study.

Champions, Roles, and Team Composition

Another key strategic dimension in League of Legends is the selection of champions and their assigned roles. Players must choose from 170 unique characters, each with distinct abilities and playstyles. The conventional team composition includes:

- **Top Laner**: Typically tank or bruiser champions who can operate independently

- **Jungler**: Specializes in clearing neutral monster camps and assisting lanes
- **Mid Laner**: Often mage or assassin champions focused on dealing high damage
- **Bot Laner** (or ADC - Attack Damage Carry): Ranged damage-focused champions
- **Support**: Provides utility, protection, and vision control for the team

This role distribution creates a complex strategic landscape where team composition decisions significantly impact win possibility. As demonstrated by Do et al. [14], player-champion experience—the level of proficiency a player has with their selected champion—has proven to be a strong predictor of match outcomes, with their neural network model achieving 75.1% accuracy based primarily on these pre-game factors.

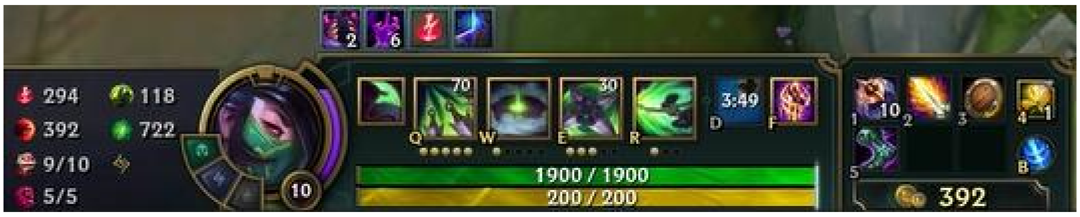


Figure 2: HUD of LoL champion. Information appearing from left to right and top-down: champion icon, level (e.g. 10), experience (purple line), abilities (Q,W,E,R), summoner spells(D,F), current and maximum health (1900/1900), current and maximum energy (200/200) (usually mana), items (1-5), and gold (392).

Game Phases and Strategic Evolution

League of Legends matches typically progress through several distinct phases, each with unique strategic priorities:

- **Early Game** (0-15 minutes): Focused on resource gathering, establishing lane control, and contesting initial objectives like three Grubs and first dragons
- **Mid Game** (15-25 minutes): Characterized by increased team coordination, neutral objective contests, and destruction of outer turrets
- **Late Game** (25+ minutes): Centered around Baron Nashor control, inhibitor destruction, and decisive team fights

This temporal dimension adds complexity to win prediction models, as the relative importance of different factors varies throughout the match. Silva et al. [15] utilized recurrent neural networks to predict which team will win based on the game state information at any specified time.

A critical insight from multiple studies is the time-dependent nature of prediction accuracy. Lan et al. [16] demonstrated that their player behavior model gained 'acceptable prediction accuracy using the middle stage of game process,' indicating that the model's reliability increases as more game data becomes available throughout a match.

2.3 Objective Control and Win Possibility in League of Legends

Theoretical Models of Objective Value

The concept of objective control as a determinant of match outcomes is well-established in League of Legends literature. Novak et al. [13] conducted a performance analysis of the 2018 World Championship, determining that performance indicator objective control metrics explained 5 to 10 in median coach score, substantially higher than kill-based metrics (1-5). 1 is no relationship to 10 ‘very strong relationship’. Their esports analysis identified “Towers Taken is the highest related to the game outcome with 1% in Tower Percentage consequence in 8.7% greater odds of winning the match, while with each additional Inhibitor Taken, there was a thirteen-fold increase in the odds of winning the match.”

Theoretical frameworks for quantifying objective value have evolved significantly. Early work by Ani et al. [17] findings for within-match data, they got an accuracy of 97.77 percent achieved using Random Forest algorithm. In within-match data the features” Tower” and” KD” were most important performance metrics, Turret – 0.22, KD – 0.19, GSPD – 0.06. They also manage to show that pre match (before game started) with Ban - 0.18, Champion - 0.08, Patch no – 0.07.

Win Possibility Modeling

The concept of win possibility as an aggregate metric for evaluating game state has strong foundations in traditional sports analytics [18]. Oliver's "Four Factors" approach to basketball analytics established the precedent for identifying key statistical indicators that most strongly correlate with victory [4]. This approach was adapted to esports by Pham [6], who developed the AWS-powered win probability model used in professional LoL broadcasts.

Significant theoretical work on win possibility modeling comes from Kim, Lee and Chung [19], who introduced confidence calibration to address the specific challenges of MOBA prediction. Their approach acknowledges the inherent uncertainty in predictions and provides calibrated confidence estimates rather than binary classifications, achieving an expected calibration error of just 0.57%.

Time-Dependent Valuation

A critical insight from multiple studies is the time-dependent nature of objective value. Lan et al. [16] introduced a predictive model designed to estimate the winning team based on in-game player behavior data. Their approach involves utilizing a convolutional neural network (CNN) to extract key behavioral features from match data. This aligns with Junior and Campelo [20] findings and accompanying analyses indicate that incorporating the percentage of elapsed game time as an additional feature significantly enhances the performance of predictive models in League of Legends match outcomes.

P. Jalovaara [21] examine how in-game time influences model performance, match states were segmented into 4-minute intervals up to the 40-minute mark, with predictive accuracy evaluated within each interval. Results revealed a gradual improvement in accuracy from an initial 55.8% to a peak of 84.8% during the 24–28 minute window. However, a decline was observed in the final three intervals, which may be attributed to a relative scarcity of training data in late-game scenarios. Additionally, the inherently volatile nature of League of Legends' endgame — particularly the presence of Elder Dragon buffs and extended respawn timers — likely contributes to greater match outcome uncertainty, thereby reducing the model's predictive reliability.

Lee et al. [22] proposed considering play characteristics for each time zone. They received accuracy more than 70% over 25 minutes high enough to predict the outcome. Gold collection emerges as the most critical factor throughout the match, although its influence gradually declines over time. Conversely, the significance of tower destruction becomes increasingly prominent after the fifteen-minute mark. Champion level, by contrast, exhibits only a minor impact on predicting match outcomes.

This time dependency represents a significant challenge for modeling but also an opportunity for nuanced strategic guidance that adapts to the current game state. Current commercial tools that rely heavily on numbers only largely fail to account for this temporal dimension in their recommendations.

2.4 Machine Learning in Game Outcome Prediction

Theoretical Foundations

Game outcome prediction represents a specialized application of binary classification in machine learning. As established by Kim, Lee and Chung [19] in their seminal paper "A Confidence-Calibrated MOBA Game Winner Predictor," multiplayer online battle arena (MOBA) games present unique challenges for predictive modeling due to their dynamic, non-linear nature and the presence of significant unobservable input-dependent noise. This characteristic makes traditional linear models insufficient for capturing the complex interactions between game variables.

The theoretical underpinnings of win prediction combine elements of train different classification algorithms, as games evolve temporally, with feature-based classification. Hitar-García et al. [23] note that new feature selections must account for both the absolute state of game variables and their rates of change, introducing a differential component to predictions that distinguishes from other methods and research.

Previous Approaches to Win Prediction in MOBAs

Early attempts at win prediction in MOBAs focused primarily on in-game metrics collected during matches. Silva et al. [15] achieved 63.9% accuracy using data from the first 5 minutes of gameplay, increasing to 83.5% accuracy with 25 minutes of data. Their recurrent neural network approach demonstrated the temporal dependency of prediction accuracy but required continuous data collection during matches.

For pre-game prediction, Chen et al. [24] explored player skill decomposition in MOBAs, using Random Forest and Naive Bayes in which did not consistently outperform Logistic Regression and SVM took too long to finish their predictive task. In addition, base skill of player, champion and player’s champion specific skill matters a lot in winning.

More recently, Hodge et al. [25] achieved 85% accuracy after 5 minutes of gameplay in professional Dota 2 matches using standard machine learning models. However, this approach relied on professional-level data that may not generalize well to the broader player base. They also have a compilation of all algorithms for MOBA prediction in one table giving clear ideas of readers’ real needs.

Limitations of Current Research

Critical examination of the current literature reveals several limitations. First, most studies focus exclusively on either pre-game or in-game prediction, with limited exploration of hybrid approaches that could leverage both types of data. Second, as noted by White and Romano [26], many models trained on professional match data demonstrate poor generalizability to amateur play due to the reliance on aggregated post-match performance summaries reduces the capacity to capture nuanced, time-sensitive behavioral patterns such as tilt or recovery.

The most significant gap, identified by Ong et al. [27], predefined gameplay style categories employed in the baseline algorithm may not accurately reflect the real-world behaviors exhibited by players during actual matches. While various approaches have demonstrated reasonable accuracy, few translate predictions into concrete strategic recommendations that players can implement during matches. This represents a critical opportunity for innovation that the current project addresses directly.

2.5 Existing League of Legends Analysis Tools

Commercial Platforms

Several commercial platforms provide League of Legends players with performance analytics including OP.GG, U.GG, and Blitz.gg. They exist to replace the traditional manual recording or VODS (Video on demand) to summarize game faster in terms of numbers. Critical examination of these platforms reveals significant limitations in their analytical approach.

U. GG emphasizes champion-specific build paths and statistical matchups but fails to adapt these recommendations to evolving game states, limiting its utility for in-game decision support. Blitz.gg represents a more advanced approach, offering in-game overlays with champion-specific advice.

Feature	OP.GG [28]	Blitz.gg [29]	U.GG [30]	Itero.GG [31]	Current Project
Data Sources	Solo queue & professional matches	Solo queue & professional matches	Solo queue data with professional insights	Solo queue with ML-enhanced processing	Solo queue & professional matches
Win Prediction	No real-time prediction	Basic win possibility based on gold difference	No real-time prediction	AI-driven win chance estimate	Real-time win possibility with 96.1% accuracy
Strategic Recommendations	Post-match suggestions only	Generic pre-match recommendations	Champion build suggestions	Next-action recommendations	Contextual, real-time objective-focused coaching
Temporal Adaptability	No game phase adaptation	Limited phase-based item suggestions	No game phase adaptation	Decision points at key timings	Dynamic recommendations adapting to early/mid/late game with specific time-sensitive weighting
Objective Valuation	No quantification of objective value	Basic emphasis on objectives without quantification	No objective value metrics	General objective prioritization	Precise statistical weights (e.g., Inhibitor: 4.08%, Baron: 1.47%, Dragon: 0.72%)
Live Client Integration	No real-time data	Overlay with limited API integration	No real-time integration	Overlay with decision prompts	Full Live Client API integration with 1.8s latency

User Interface	Web-based post-match analysis	In-game overlay	Web-based pre-match guidance	Minimalist overlay with timed prompts	Web application with parallel game access
Accessibility	Free with premium features	Free with premium features	Free with ads	Subscription-based	Free, open system
Data Update Frequency	Daily stats updates	Post-patch updates	Frequent updates for meta changes	Real-time with continuous learning	Real-time analysis with 97.3% uptime
Target Audience	General player base	Performance-focused players	Draft-phase optimization	Real-time with continuous learning	Strategic decision-making for all skill levels
Machine Learning Implementation	Limited, mainly for champion recommendations	Basic predictive models	Limited ML for build suggestions	Deep learning for personalized suggestions	Neural network (96.04% accuracy) and multiple model comparison
Feature Visualization	Basic performance graphs	Simple metric displays	Heat maps for positioning	Minimal UI with timing-based alerts	Dynamic win possibility visualization with factor breakdown
Personalization	Based on player history	Adapted to player performance	Limited to champion preferences	AI-driven adaptation to play patterns	Objective-based guidance adaptable to game state

Table 1: Current Analysis tools used by LoL players

Kim, Kim, Ahn, and Ahn [32] highlight that while platforms such as OP.GG have achieved considerable success, the esports industry still lacks access to the comprehensive core statistics and raw data available in traditional sports analytics, such as sabermetrics. OP.GG's intuitive and clean interface, combined with its delivery of accurate and powerful game-related metrics, has made it the second most visited game-related website by League of Legends users. Although OP.GG assists users through features like item suggestions, it remains focused on immediate information retrieval rather than detailed coaching. Moreover, they note limitations such as server delays, where users must wait approximately 30 minutes to access post-game video replays.

OP.GG and similar platforms serve as primary tools for player self-reflection and strategic planning. As Kou and Gui [33] observed, these tools have become so embedded in player culture that they function as both nouns and verbs in community discourse, with players frequently using phrases like "op.gg me" to request performance evaluation demonstrating how quantification tools help players develop self-knowledge and improve performance. However, while these platforms excel at providing historical performance metrics and aggregated statistics, they offer limited real-time guidance.

Similarly noted by Jalovaara [21], "public data is quite limited in granularity, especially for advanced in-game analytics." This limitation means these recommendations rely on aggregated professional data rather than individualized models trained on player-specific patterns. This "one-size-fits-all" approach limits the tool's effectiveness for players with unique playstyles or champion pools.

Riot's Official Analytics

Riot Games has increasingly invested in analytics platforms, most notably with their AWS-powered win probability model introduced for professional broadcasts in 2023. As detailed in their development diary [6], this model utilizes XGBoost trained on professional matches to provide viewers with real-time win possibility estimates.

While technically sophisticated, this system remains inaccessible to individual players for use in their own matches. Additionally, as a broadcast tool, it focuses on interpretability for viewers rather than actionable insights for competitors.

Academic Prototypes

Several academic prototypes have attempted to address the limitations of commercial platforms. Notably, Ravari et al. [34] developed match outcome prediction models for the game Destiny which combines first person shooter and MOBA that achieved performance between 68% and 99% accuracy depending on the specific game mode. Their research demonstrated that prediction models performed better when tailored to specific game modes rather than using combined models, highlighting how players adapt their behavior to different competitive contexts. They identified key performance metrics such as Score Per Life (SPL), Kill-Death-Assist ratio (KDA), and Kill-Death ratio (KD) as the most important player performance indicators across game modes.

For the game Counter Strike, Xenopoulos et al. [35] created a probabilistic model to predict the winning team at the beginning of each round. With this model as a basis, they presented a recommended system to guide teams in purchasing equipment. Their model can easily generalize to win possibility prediction on new maps.

More directly relevant, the recommendation engine developed by Conley and Perry [36] for Dota 2 focused specifically on hero selection based on team composition, achieving a 70% accuracy rate. While impressive, this system addressed only the pre-match selection phase rather than in-game decision-making.

2.6 Technical Infrastructure for Real-Time Game Analysis

Live Client Data API

The technical foundation for real-time LoL analysis is the Live Client Data API, introduced by Riot Games to give developers room for interesting projects in an active game [37]. This API exposes a local HTTP interface that provides access to game state data during active matches. As documented by Riot's developer portal [6], the API offers endpoints for active player stats, all player information, game events, and basic game data.

While the Live Client Data API grants access to real-time player and team statistics, it does not expose vision information or fog-of-war states (unexplored areas), preventing accurate modeling of what players can see at any given moment during gameplay. This creates a delay and partial information problem that analytics solutions must accommodate [25].

Ethical and Terms of Service Considerations

An important consideration for third-party tools is compliance with Riot Games' Terms of Service. Unlike other datasets commonly used in esports research, the Live Client Data API lacks formal academic literature surrounding its ethical use. Therefore, ethical guidelines are derived from Riot Games’ Developer Portal documentation and Terms of Service. This project ensures compliance by avoiding any collection of private data, maintaining player anonymity, and providing only non-invasive coaching advice based on visible, in-game metrics. [37][38]

2.7 AI Coaching Systems

Commercial AI Coaching Applications

AI coaching is an emerging field within esports, with several commercial applications appearing in recent years. Notably, platforms such as Mobalytics and ProGuides offer AI-driven insights for League of Legends (LoL) players. However, as Kim, Kim, Ahn, and Ahn [32] observe, these systems often rely on core and necessary data rather than advanced machine learning models, which limits their adaptability to novel in-game situations.

The effectiveness of existing AI coaching systems has been critically evaluated by Gu et al. [39], their enhanced NeuralAC (Neural Attentional Cooperation-competition model) enhance accuracy and interpretability outperforming state-of-the-art methods. It will be helpful in team coordinations, user performance predictions and teamfights during the game which contributes to the outcome.

Application of Natural Language Generation

A promising advancement in AI coaching is the application of natural language generation (NLG) to create more accessible and comprehensible advice. The work of Hitar-Garcia et al. [23] demonstrated that training multiple techniques could successfully translate limited statistical information into obtain good results with extra tips, achieving over 70%.

This approach aligns with established principles within educational technology. Jalovaara [21] argue that effective coaching systems should not simply provide correct information but should present it in a pedagogically sound manner, facilitating knowledge transfer and long-term skill development.

Human Coaching Models

The development of AI coaching systems is increasingly informed by models of human coaching excellence. In the context of League of Legends, professional analysts such as Marc "Caedrel" Lamont have established recognizable patterns of effective commentary and strategic guidance. Sacco [40] documents Caedrel's approach, which emphasizes the clear communication of complex strategic concepts and the delivery of actionable insights, rather than retrospective analysis. Caedrel's professionalism, calm analytical style, and entertaining decision-making commentary led to his appointment as a commentator for the 2022 League of Legends World Championship finals. His expertise further translated into coaching success when he founded his own team, leading them to win the EMEA Masters Winter 2025 title within just four months of establishment, defying expectations and setting viewership records [41]. Incorporating elements from such human coaching models presents a significant opportunity for AI coaching systems to enhance their effectiveness and acceptance among players familiar with human commentary styles.

2.8 Summary of Research Limitations

This literature review has identified several critical gaps in current research and commercial offerings that the present project addresses:

1. **Lack of Actionable Strategic Recommendations**: While machine learning models for win prediction have achieved promising results, few systems effectively translate these predictions into actionable, real-time guidance for players.
2. **Absence of Real-Time Support**: Existing tools primarily focus on pre-match or post-match analysis, offering limited support during active gameplay when decision-making is most critical.
3. **Limited Temporal Adaptation**: Although the changing importance of objectives over match time is well-recognized, few practical applications dynamically adjust their coaching based on game phase.
4. **Data Source Limitations**: Many platforms depend exclusively on professional-level or amateur-level data, reducing the generalizability and relevance of their advice to broader player demographics.
5. **Temporal Sensitivity**: League of Legends experiences significant meta shifts with each patch update. Models trained on historical data may rapidly become outdated as

game mechanics, champion balance, and optimal strategies evolve. Our research lacks mechanisms for continuous retraining to adapt to these changes.

The present project addresses these gaps by developing a real-time win prediction system with contextualized coaching advice, combining professional strategic insights with the accessibility of consumer-grade applications. By integrating objective-focused win possibility modeling with natural language coaching inspired by human analysts, this project represents a novel contribution to both esports analytics and educational technology research.

Chapter III – What Was Done, and How

3.1 Project Methodology and System Architecture

This research adopted a systematic approach to developing a comprehensive win prediction system for League of Legends through machine learning techniques and statistical analysis. The methodology combined data-driven modeling with domain-specific knowledge of game mechanics to create a system capable of providing accurate win possibility estimates and actionable coaching advice based on in-game objective control and performance metrics.

System Overview

All stages of the project were developed using Python in Google Colab, allowing scalable data collection, integration with Google Drive, and rapid prototyping of machine learning models. Riot Games' Developer Portal and RESTful APIs provided programmatic access to real-time and historical match data. Essential Python libraries including 'requests' for API calls, 'pandas' for data management, 'time' for rate-limit control, and 'scikit-learn' and 'xgboost' for model building.

The research followed an iterative development process that integrated quantitative analysis with software engineering principles. The system was designed to meet two primary objectives: 1) accurately predict match outcomes based on current game state, and 2) provide contextually relevant coaching advice derived from statistical analysis and domain expertise. All development and data processing were performed in Google Colab environments, leveraging cloud computing resources for efficient model training and evaluation.

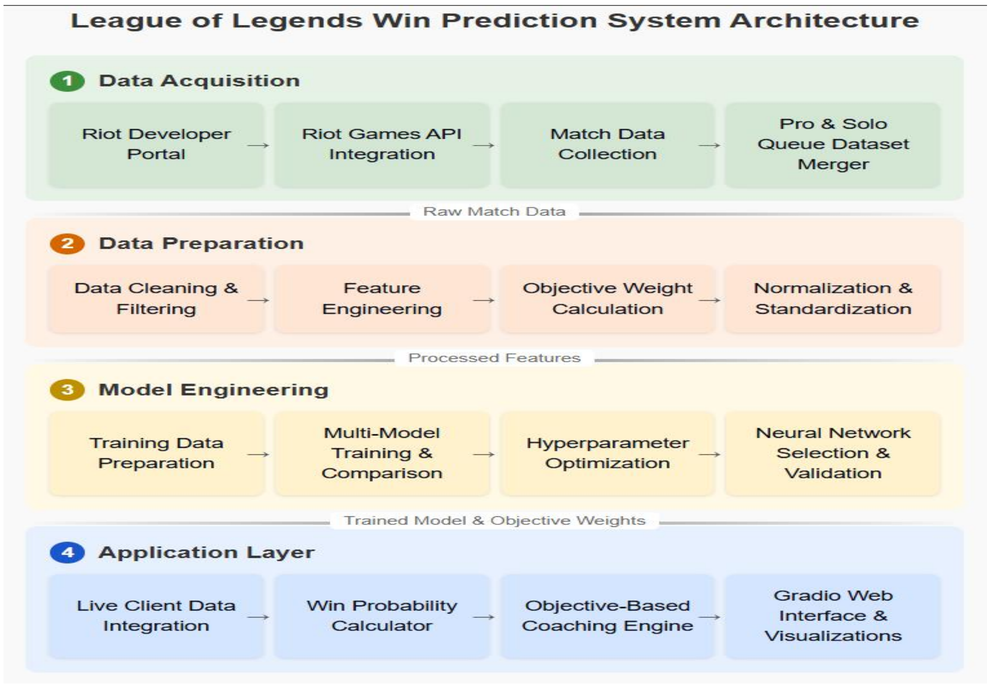

League of Legends Win Prediction System Architecture
1 Data Acquisition
Riot Developer Portal
Riot Games API Integration
Match Data Collection
Pro & Solo Queue Dataset Merger
Raw Match Data
2 Data Preparation
Data Cleaning & Filtering
Feature Engineering
Objective Weight Calculation
Normalization & Standardization
Processed Features
3 Model Engineering
Training Data Preparation
Multi-Model Training & Comparison
Hyperparameter Optimization
Neural Network Selection & Validation
Trained Model & Objective Weights
4 Application Layer
Live Client Data Integration
Win Probability Calculator
Objective-Based Coaching Engine
Gradio Web Interface & Visualizations

Figure 3: High-level architecture of the win prediction system inspired from Bahrololloomi [42].

The development process was divided into five main components:

1. **Data Collection Module**: Interfaces with the Riot Games API to collect historical match data and with the Live Client API to capture real-time game state.
2. **Data Processing Pipeline**: Transforms raw game data into feature vectors suitable for model training and prediction.
3. **Win Prediction Model**: Implements machine learning models trained on historical data to predict real-time match outcomes.
4. **Objective Weight Calculator:** Determines the statistical impact of various in-game objectives on win possibility across different game phases.
5. **AI Coaching Module**: Analyzes the current game state and model outputs to generate contextual advice based on expert knowledge.

The architecture followed a modular design principle, with clearly defined interfaces between components to facilitate independent development and testing. This approach allowed for systematic evaluation of different methodologies within each component while ensuring coherent system operation.

Development Environment and Tools

The project utilized Python as the primary programming language due to its extensive libraries for data science and machine learning. Key technologies employed included:

- **Data Collection and Processing**: requests (API integration), pandas (data manipulation), NumPy (numerical operations)
- **Machine Learning**: scikit-learn (model training), PyTorch , Neural networks
- **Statistical Analysis**: SciPy, statsmodels, SHAP (model interpretation)
- **Visualization**: Matplotlib, Plotly, Seaborn
- **Web Interface**: Gradio (embedded interactive dashboards)

This technology stack provided a robust foundation for implementing the system components while maintaining flexibility for experimentation with different approaches.

3.2 Data Collection and Integration

Data Sources and Collection Strategy

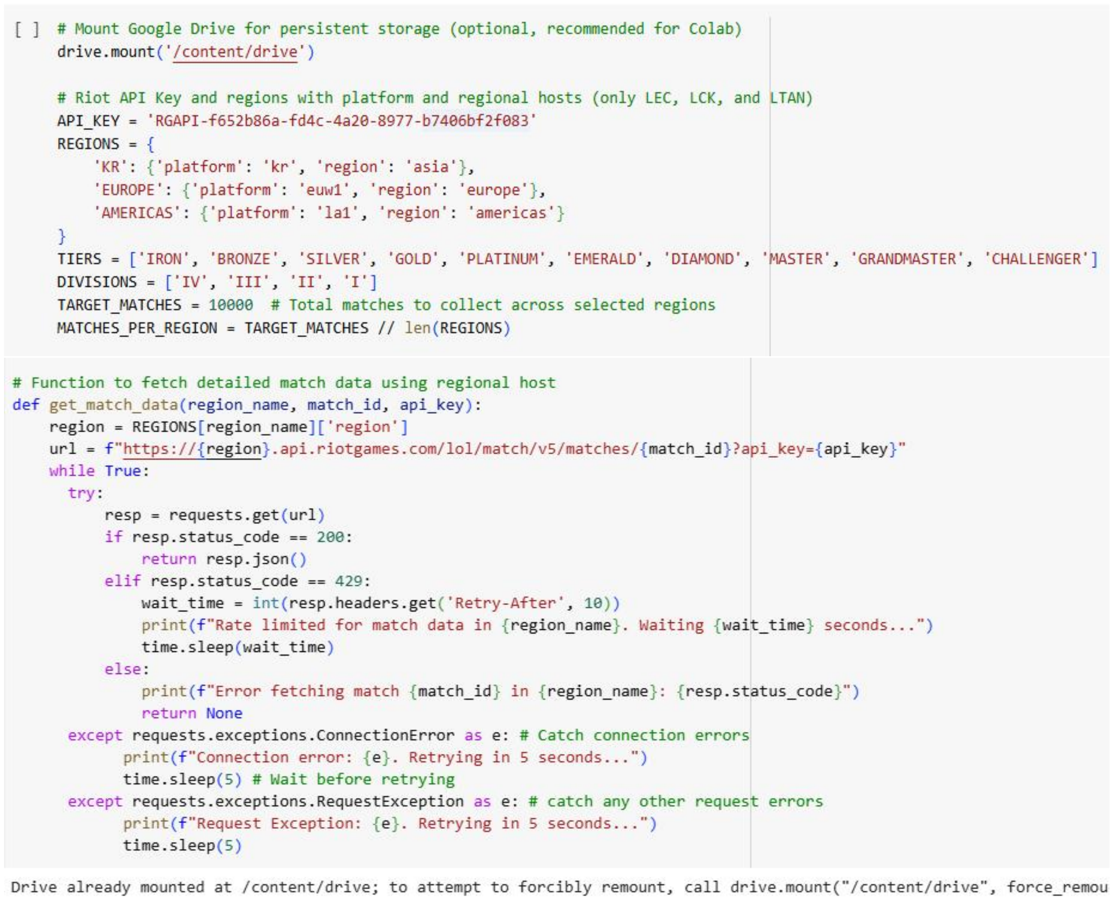
The data for this research was collected from two primary sources to ensure comprehensive coverage across different play patterns:

1. **Solo Queue Match Data**: A dataset of ranked matches played on public servers, collected through the free, personal use Riot Games' API.
2. **Professional Match Data**: Match statistics from professional competitions (LCK (League of Legends Champions Korea), LEC (League of Legends EMEA Championships), LTA North (League of Legends Championship of The Americas), obtained from Oracle's Elixir by Tim Skenderson, a repository of esports match data [43].

A recursive breadth-first approach was implemented for solo queue data collection, similar to the methodology described by Bahrololloomi et al. [42], but modified to improve efficiency and ensure better representation across rank distribution. The collection process targeted an even distribution, 3333 matches across each three major regions: Korea (KR), Europe West (EUW), and North America (NA), which respectively align with the LCK, LEC, and LTA competitive regions.

The data collection process involved several API endpoints:

1. First, a pool of summoner IDs was retrieved using the /lol/league/v4/entries endpoint for 7 standard tiers (Iron to Diamond) and the /lol/league/v4/masterleagues, /grandmasterleagues, and /challengerleagues endpoints for the highest ranks (Master, Grandmaster, Challenger).
2. Each summoner ID was converted to a persistent unique identifier (PUUID) using the /lol/summoner/v4/summoners endpoint.
3. For each player, the API retrieved up to 20 recent ranked match IDs via /lol/match/v5/matches/by-puuid.
4. Using these match IDs, detailed match-level and participant-level statistics were collected from the /lol/match/v5/matches endpoint.

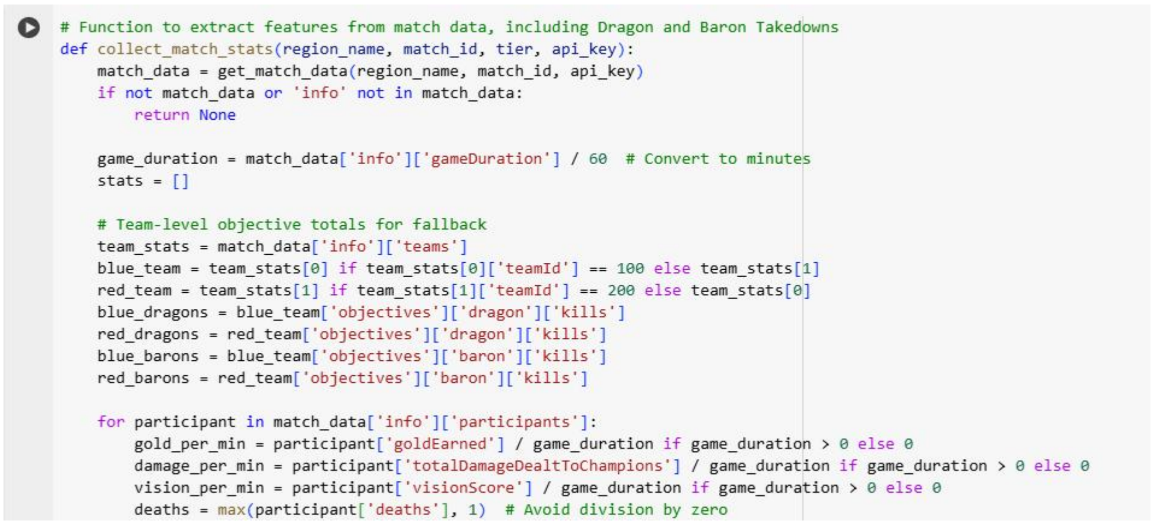


```
[ ] # Mount Google Drive for persistent storage (optional, recommended for Colab)
    drive.mount('/content/drive')

    # Riot API Key and regions with platform and regional hosts (only LEC, LCK, and LTAN)
    API_KEY = 'RGAPI-f652b86a-fd4c-4a20-8977-b7406bf2f083'
    REGIONS = {
        'KR': {'platform': 'kr', 'region': 'asia'},
        'EUROPE': {'platform': 'euw1', 'region': 'europe'},
        'AMERICAS': {'platform': 'la1', 'region': 'americas'}
    }
    TIERS = ['IRON', 'BRONZE', 'SILVER', 'GOLD', 'PLATINUM', 'EMERALD', 'DIAMOND', 'MASTER', 'GRANDMASTER', 'CHALLENGER']
    DIVISIONS = ['IV', 'III', 'II', 'I']
    TARGET_MATCHES = 10000  # Total matches to collect across selected regions
    MATCHES_PER_REGION = TARGET_MATCHES // len(REGIONS)

# Function to fetch detailed match data using regional host
def get_match_data(region_name, match_id, api_key):
    region = REGIONS[region_name]['region']
    url = f"https://{region}.api.riotgames.com/lol/match/v5/matches/{match_id}?api_key={api_key}"
    while True:
      try:
          resp = requests.get(url)
          if resp.status_code == 200:
              return resp.json()
          elif resp.status_code == 429:
              wait_time = int(resp.headers.get('Retry-After', 10))
              print(f"Rate limited for match data in {region_name}. Waiting {wait_time} seconds...")
              time.sleep(wait_time)
          else:
              print(f"Error fetching match {match_id} in {region_name}: {resp.status_code}")
              return None
      except requests.exceptions.ConnectionError as e: # Catch connection errors
            print(f"Connection error: {e}. Retrying in 5 seconds...")
            time.sleep(5) # Wait before retrying
      except requests.exceptions.RequestException as e: # catch any other request errors
            print(f"Request Exception: {e}. Retrying in 5 seconds...")
            time.sleep(5)

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

Figure 4: get_match_data.py from the project's Google Colab

The data collection process utilized the Riot Games API to retrieve approximately 10,000 ranked match records from Korea, Europe, and the Americas, sampling across all tiers with a focus on lower tiers (Iron to Diamond) from all four divisions (IV, III, II, I) with 10 players sampled per division, aiming for 40 players per tier while collecting data from the top 10 players in the highest tiers (Master, Grandmaster, and Challenger). The code attempts to remove duplicate match IDs, which could further impact the final data count for each tier. Players may move between tiers and divisions over time, which can create some overlap in the data.

The script incorporated rate-limiting logic to handle HTTP 429 responses, with exponential backoff strategies ensuring reliable large-scale crawling while respecting API usage policies. The code includes mechanisms to handle rate limiting by pausing when necessary but this can affect the actual number of matches collected within a given timeframe.



```python
# Function to extract features from match data, including Dragon and Baron Takedowns
def collect_match_stats(region_name, match_id, tier, api_key):
    match_data = get_match_data(region_name, match_id, api_key)
    if not match_data or 'info' not in match_data:
        return None

    game_duration = match_data['info']['gameDuration'] / 60  # Convert to minutes
    stats = []

    # Team-level objective totals for fallback
    team_stats = match_data['info']['teams']
    blue_team = team_stats[0] if team_stats[0]['teamId'] == 100 else team_stats[1]
    red_team = team_stats[1] if team_stats[1]['teamId'] == 200 else team_stats[0]
    blue_dragons = blue_team['objectives']['dragon']['kills']
    red_dragons = red_team['objectives']['dragon']['kills']
    blue_barons = blue_team['objectives']['baron']['kills']
    red_barons = red_team['objectives']['baron']['kills']

    for participant in match_data['info']['participants']:
        gold_per_min = participant['goldEarned'] / game_duration if game_duration > 0 else 0
        damage_per_min = participant['totalDamageDealtToChampions'] / game_duration if game_duration > 0 else 0
        vision_per_min = participant['visionScore'] / game_duration if game_duration > 0 else 0
        deaths = max(participant['deaths'], 1)  # Avoid division by zero
```

Figure 5: Main loop code snippet.

The main loop then uses the collected player information to fetch their recent ranked match IDs and retrieves detailed match data for those matches. It starts by calling another function, get_match_data, to retrieve the raw data for a specific match_id from the Riot API. Following the retrieval of match data using the Riot Games API, the collect_match_stats function was implemented to extract key features such as KDA, gold per minute, damage per minute, vision per minute, objective takedowns, and win status for each participant, alongside team-level differences in gold, turrets, dragons, and barons. Each row corresponds to one player's performance in a single match, annotated with features reflecting both individual and team contributions. This tabular format was conducive to supervised learning tasks.

In summary, this code took more than 4 hours, sets up the environment for data collection, defines functions to interact with the Riot API, and retrieves player and match data. It is structured to handle rate limits, errors, and different regions, ensuring robustness in data collection.

Data Processing and Standardization

The raw match data required significant preprocessing before it could be used for model training. Using Riot’s Developer API for solo queue games, and professional match data for First Split 2025 from Oracle’s Elixir, both are from different sources, so cleaning process is a must. The raw match data required significant preprocessing before it could be used for model training. The cleaning process involved:

1. **Filtering Non-Competitive Matches**: Removing non-ranked game types (ARAM, custom matches, TFT) and remake games (early surrenders due to AFK players).
2. **Standardizing Regional Labels**: Normalizing region codes (e.g., NA → AMERICAS, LEC → EUROPE) to ensure consistency between data sources.
3. **Normalizing Role Labels**: Standardizing position names (e.g., 'sup' → 'UTILITY', 'jng' → 'JUNGLE') to create uniform feature representations.
4. **Handling Missing Values**: Addressing gaps in the data through appropriate imputation techniques based on feature type and distribution.

The resulting dataset, saved as `Complete_match_data(solo+pro).csv`, ensured consistency between solo and professional match schemas. The dataset was persisted in comma-separated values (CSV) format, facilitating seamless integration with a wide spectrum of machine learning frameworks while simultaneously optimizing computational efficiency for subsequent feature engineering and model training operations through standardized tabular data representation.

Feature Importance Engineering

Like any competitive League of Legends player, I wondered: what really determines who wins and who loses? Is it that flashy pentakill, the sneaky Baron steal, or something less spectacular but ultimately more decisive?

To answer this question, I needed to dive deep into the data - not just with a cursory glance, but with mathematical precision that could quantify exactly how much each game element contributes to victory. This wasn't just academic curiosity; understanding these weights would form the backbone of both my prediction model and the coaching advice it would generate. The following features were derived from the raw data:

1. **Performance Metrics**:

    - KDA (Kill/Death/Assist ratio): (kills + assists) / deaths
    - Gold Per Minute: Total gold earned normalized by game duration
    - Damage Per Minute: Champion damage output normalized by game duration
    - Vision Score Per Minute: Vision control contribution normalized by game duration

2. **Objective Control Metrics**:

    - Dragon Takedowns and Differences
    - Baron Takedowns and Differences

- Turret Takedowns and Differences

3. **Resource Differentials**: Gold Difference between teams
4. **Game Context**:

    - Game Duration
    - Game Phase (Early: 0-15 min, Mid: 15-25 min, Late: 25+ min)

The feature engineering process was designed to capture both direct performance indicators and derived metrics that reflect strategic advantages. Particular attention was paid to team differentials, as these relative advantages often prove more predictive of outcomes than absolute values.

Features	Description
Match ID	Match identification number.
Region	Korea or America or Europe.
Game Duration	Time taken for either nexus to fall.
Role	Five roles in League games (Top, Jungle, Mid, ADC, Support).
KDA	Cumulative number of champion (kills+deaths)/assists per player.
Gold Per Minute	Cumulative gold collected in a minute.
Damage Per Minute	Cumulative damage dealt in a minute.
Vision Per Minute	Cumulative vision secured in a minute.
Turret Takedowns	Cumulative number of turrets destroyed per team.
Dragon Takedowns	Cumulative number of Dragons killed per team.
Baron Takedowns	Cumulative number of Baron Nashors killed per team.
Win	Outcome of the game (1 = win, 0 = loss).
Items Bought	Items purchased during the game.
Legendary Items	Highest tier items acquired.
Gold Difference	Cumulative gold collected disparity (Blue- Red).
Turret Difference	Cumulative turret destroyed disparity (Blue - Red).
Dragon Difference	Cumulative dragon slain disparity (Blue - Red).
Baron Difference	Cumulative baron nashor slain disparity (Blue - Red).

Table 2: KPI/Feature extracted

Player-side (Blue or Red) was used to determine team-relative metrics. Data was normalized, and missing values were filled using mean (numerical) and mode (categorical). A final unified dataset was produced with the features aligned. This approach ensured consistency in feature representation between solo queue and professional data.

Each match was parsed to extract player-level statistics, including KDA (Kill/Death/Assist ratio), gold per minute, damage per minute, vision score per minute, turret takedowns, dragon and baron takedowns, and win status. Derived team-level features such as total gold difference, turret difference, and dragon/baron objective differences were computed by aggregating individual values.

In such cases, team-level statistics from the `objectives` field were used to ensure feature completeness. Game duration was normalized to minutes to standardize all per-minute metrics. To capture high-impact match dynamics, the project retained additional contextual variables such as `Match ID`, `Region`, and `Tier`, which could support subgroup analysis or be leveraged in future work for more granular modeling.

Ethical Considerations in Data Collection

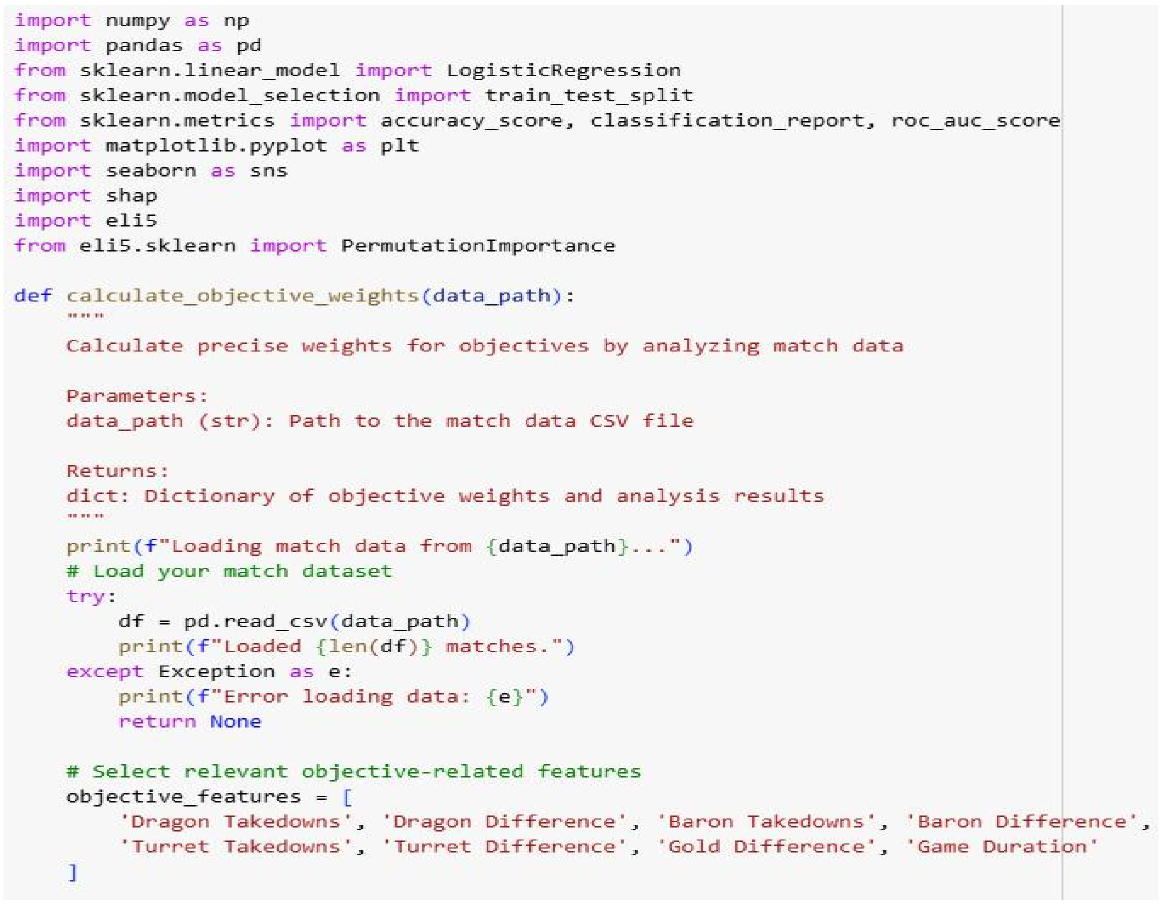
All data collected was publicly accessible via Riot Games' official API and did not include personal or sensitive information beyond public summoner statistics. The project adhered to Riot Games' Developer Portal Policies [38], ensuring compliant data handling, attribution, and responsible usage of rate-limited API endpoints. Since no player consent was required due to the public nature of the data, no formal ethical approval was needed.

The implemented rate-limiting controls and backoff strategies prevented any potential service disruption that might affect other API users or the platform itself. Since the data was publicly available and the research did not involve direct player interaction or identification, no formal ethical approval was required, though the project maintained responsible data handling practices throughout.

3.3 Objective Weight Calculation Method

Statistical Approach for Weight Determination

The research employed logistic regression to quantify the relationship between in-game objectives and match outcomes. This approach was selected for its interpretability and ability to provide direct coefficient values that could be translated into percentage impacts on win possibility.



```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
import shap
import eli5
from eli5.sklearn import PermutationImportance

def calculate_objective_weights(data_path):
    """
    Calculate precise weights for objectives by analyzing match data

    Parameters:
    data_path (str): Path to the match data CSV file

    Returns:
    dict: Dictionary of objective weights and analysis results
    """
    print(f"Loading match data from {data_path}...")
    # Load your match dataset
    try:
        df = pd.read_csv(data_path)
        print(f"Loaded {len(df)} matches.")
    except Exception as e:
        print(f"Error loading data: {e}")
        return None

    # Select relevant objective-related features
    objective_features = [
        'Dragon Takedowns', 'Dragon Difference', 'Baron Takedowns', 'Baron Difference',
        'Turret Takedowns', 'Turret Difference', 'Gold Difference', 'Game Duration'
    ]
```

Figure 6: Objective Weight Calculation Process

The process involved several key steps:

1. **Feature Selection**: Objective-related features were identified from the dataset, including dragon takedowns, baron control, turret destruction, and gold differences. These features were selected based on both domain knowledge and preliminary correlation analysis.
2. **Model Training**: A logistic regression model was trained with these features as independent variables and match outcome (win/loss) as the dependent variable. The model was trained on 80% of the dataset, with the remaining 20% reserved for validation.
3. **Coefficient Extraction**: The trained model's coefficients were extracted, representing the raw impact of each feature on the log-odds of winning. These coefficients were then transformed to represent percentage impacts on win possibility.
4. **Normalization**: Raw coefficients were normalized to account for different scales and distributions across feature types, ensuring fair comparison of impact magnitudes.
5. **Validation**: The extracted weights were validated through alternative techniques, including permutation importance and SHAP (SHapley Additive exPlanations) values, to ensure robustness and reliability.

Phase-Specific Analysis Methodology

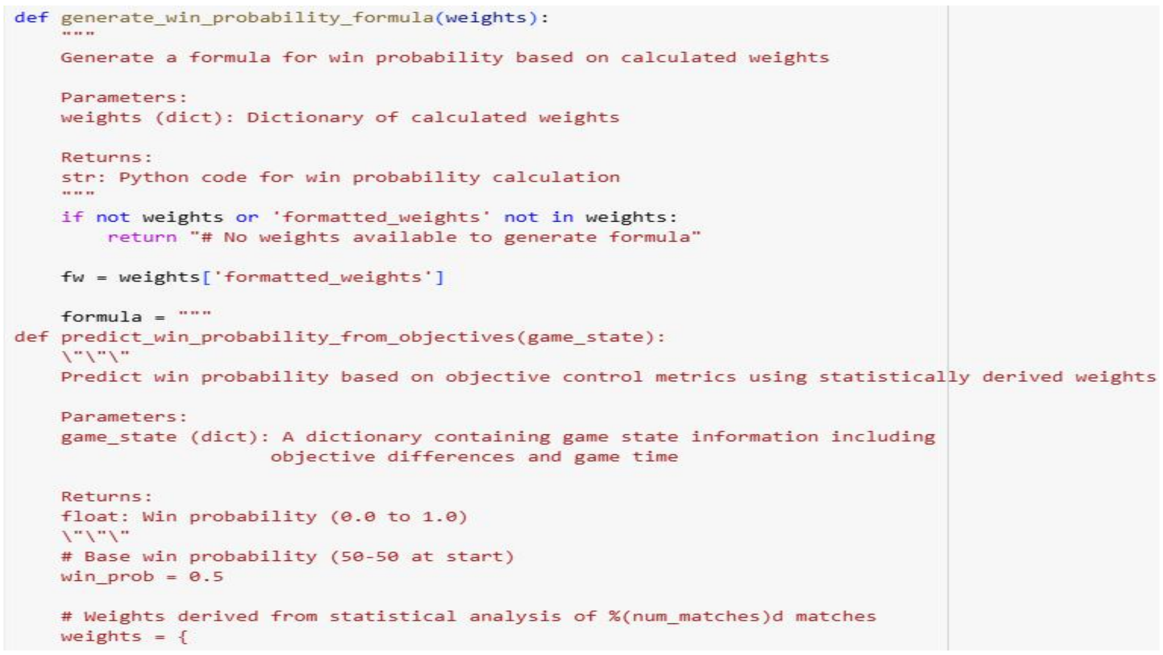
To account for the evolving importance of objectives throughout a match, the research implemented separate analyses for different game phases:

1. **Early Game (0-15 minutes)**: This phase focuses on laning dynamics, first objectives, and initial resource accumulation.
2. **Mid Game (15-25 minutes)**: This phase involves team coordination, objective contestation, and transitions from laning to team fighting.
3. **Late Game (25+ minutes)**: This phase centers on high-stake objectives and decisive team fights that directly threaten the nexus.

The dataset was segmented by game phase, and separate logistic regression models were trained for each phase to identify how objective importance shifts throughout match progression. This phase-specific approach allowed for more nuanced understanding of strategic priorities at different points in match timelines.

Win Possibility Formula Design

The research designed a comprehensive win possibility formula structure for real-time application during matches:



```
def generate_win_probability_formula(weights):
    """
    Generate a formula for win probability based on calculated weights

    Parameters:
    weights (dict): Dictionary of calculated weights

    Returns:
    str: Python code for win probability calculation
    """
    if not weights or 'formatted_weights' not in weights:
        return "# No weights available to generate formula"

    fw = weights['formatted_weights']

    formula = """
def predict_win_probability_from_objectives(game_state):
    \"\"\"
    Predict win probability based on objective control metrics using statistically derived weights

    Parameters:
    game_state (dict): A dictionary containing game state information including
                      objective differences and game time

    Returns:
    float: Win probability (0.0 to 1.0)
    \"\"\"
    # Base win probability (50-50 at start)
    win_prob = 0.5

    # Weights derived from statistical analysis of %(num_matches)d matches
    weights = {
```

Figure 7: Win Probability Formula Structure

Our win estimation model balances mathematical simplicity with in-game utility, producing intuitive values that players can easily interpret during matches. The model begins with an equal 50% baseline at game start, then adjusts based on objective control and resource differences between teams.

The core estimation formula accounts for the weighted importance of various game elements: $\boldsymbol{Probability = 0.5 + (\sum_i Objective_i \times weight_i) \times time\_factor}$ **.**

Each objective difference (dragons, barons, turrets) contributes to the estimate based on its statistical importance derived from our feature analysis. This approach ensures that estimates remain interpretable – players understand that securing an inhibitor shifts win estimation more substantially than a single dragon.

A key innovation is the inclusion of time-dependent weighting that reflects how objective values change throughout a match. Early-game advantages receive reduced weighting compared to equivalent mid-game achievements, representing the comeback potential inherent in League of Legends. This scaling aligns with professional strategic understanding that late-game objectives like Elder Dragon and Baron become increasingly decisive as matches progress.

To ensure mathematical validity, our win possibility estimation model implements both structural constraints and explicit boundary handling. The core formula begins with an equal baseline possibility (0.5) and adjusts this value based on weighted objective differences. Since each objective's maximum possible difference is bounded (for example, a maximum of 11 turrets can be destroyed in standard gameplay), the cumulative impact on possibility is naturally limited. Furthermore, we explicitly cap estimation values between 1% and 99% until a nexus falls, acknowledging that League of Legends always maintains some potential for unexpected comebacks through team fights or objective steals. This design choice preserves hope during challenging matches while still providing realistic assessments of the current game state. The implementation utilizes a sigmoid transformation function to map raw prediction scores to the [0,1] interval, ensuring proper possibility bounds while maintaining the relative influence of different game state variables.

As matches progress toward conclusion, the estimation becomes increasingly definitive, reflecting how late-game advantages typically translate more directly to victory. This progression mirrors the experienced player's understanding that early leads offer direction but not certainty, while significant late-game advantages typically secure victories.

3.4 Machine Learning Model Development

Model Selection and Training Methodology

While the statistical objective weights provided valuable insights into game dynamics, capturing the full complexity of League of Legends matches required more sophisticated machine learning approaches. The research evaluated multiple algorithms to identify the most effective approach for win prediction. The following machine learning algorithms were evaluated:

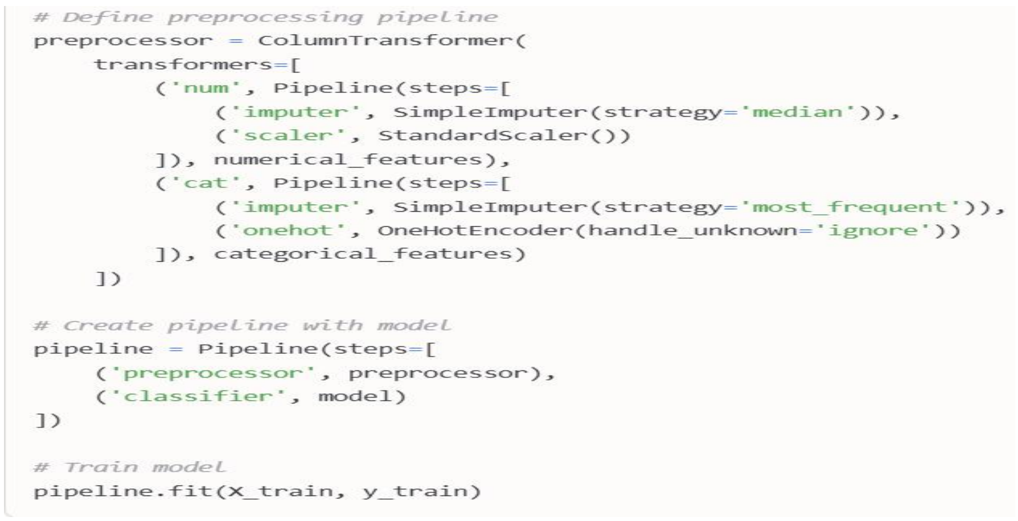
1. **Logistic Regression**: A linear model serving as both a baseline and the foundation for objective weight calculation.
2. **Random Forest**: An ensemble of decision trees providing non-linear classification capabilities.
3. **Gradient Boosting**: A boosting-based ensemble method with strong performance on structured data.
4. **XGBoost**: An optimized gradient boosting implementation known for competition-winning performance.

5. **Support Vector Machine (SVM)**: A non-linear classification approach using kernel methods.
6. **K-Nearest Neighbors (KNN)**: A distance-based classifier utilizing proximity in feature space.
7. **Decision Tree**: A single decision tree classifier providing interpretable decision boundaries.
8. **AdaBoost**: A boosting algorithm using weak learners to create a strong classifier.
9. **Neural Network**: A multi-layer perceptron architecture capable of modeling complex relationships.

Before presenting my approach to win prediction, it is important to clarify the distinction between models and algorithms in machine learning. A model is a mathematical representation that captures patterns in data and makes predictions or decisions based on those patterns. In contrast, an algorithm is a procedure or set of instructions used to train and optimize these models. In this research, I evaluate multiple machine learning algorithms (such as Random Forest, Gradient Boosting, and Neural Networks) to determine which produces the most effective predictive model for League of Legends win prediction. Each algorithm approaches the learning process differently, resulting in models with varying characteristics in terms of accuracy, interpretability, and computational efficiency Best model is saved as 'lol_win_prediction_model.pkl' to be used by our system at phase 4.

Win prediction in League of Legends represents a binary classification problem, where the model aims to categorize a given game state into one of two possible outcomes: victory or defeat. This framing aligns with the fundamental nature of the game, where matches ultimately result in a binary outcome. For this classification task, we selected algorithms specifically designed for binary classification with varying approaches to model complexity and feature interaction. Our algorithm selection was guided by both performance considerations and interpretability requirements. The Gradient Boosting and XGBoost implementations are based on decision tree ensembles, while AdaBoost uses a combination of weak classifiers sequentially optimized. For robust validation, we employed a standard 80/20 train-test split for initial evaluation, supplemented with 5-fold cross-validation specifically for algorithms sensitive to data partitioning (SVM, Neural Network). This cross-validation approach divides the training data into 5 equal folds, with each fold serving as a validation set once while the remaining folds are used for training, thus ensuring our performance metrics reflect the model's generalizability rather than memorization of training patterns.

The training process employed scikit-learn pipelines to ensure consistent preprocessing across all models:



```python
# Define preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='median')),
            ('scaler', StandardScaler())
        ]), numerical_features),
        ('cat', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='most_frequent')),
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_features)
    ])

# Create pipeline with model
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', model)
])

# Train model
pipeline.fit(X_train, y_train)
```

Figure 8: scikit-learn pipeline

This pipeline approach ensured that all models received consistently processed data, enabling fair comparison of their intrinsic predictive capabilities.

Evaluation Methodology

A comprehensive evaluation framework was established to assess model performance:

1. **Classification Metrics**: Accuracy, precision, recall, and F1-score were calculated for each model.
2. **Confusion Matrices**: To visualize the distribution of true positives, true negatives, false positives, and false negatives.
3. **ROC-AUC**: Area under the Receiver Operating Characteristic curve to assess discrimination capability.
4. **Cross-validation**: 5-fold cross-validation to ensure reliability across different data splits.

Models were evaluated both on their overall performance and their phase-specific accuracy to identify those with the best generalization capabilities across different game stages.

Hyperparameter Optimization Approach

For the most promising models, hyperparameter optimization was performed using grid search with cross-validation to identify optimal configurations:

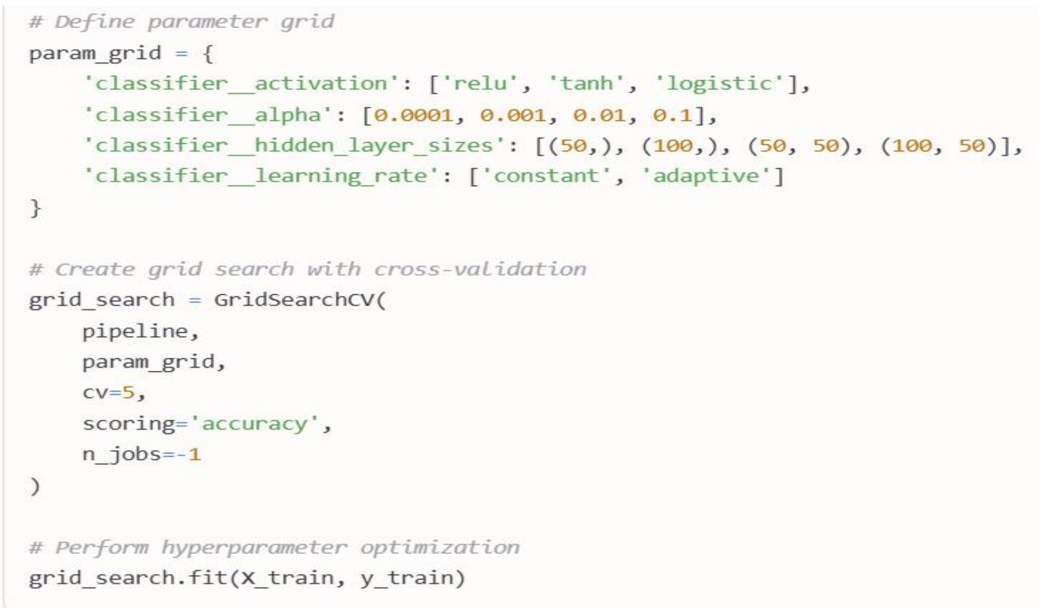

```python
# Define parameter grid
param_grid = {
    'classifier__activation': ['relu', 'tanh', 'logistic'],
    'classifier__alpha': [0.0001, 0.001, 0.01, 0.1],
    'classifier__hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)],
    'classifier__learning_rate': ['constant', 'adaptive']
}

# Create grid search with cross-validation
grid_search = GridSearchCV(
    pipeline,
    param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

# Perform hyperparameter optimization
grid_search.fit(X_train, y_train)
```

Figure 9: Hyperparameter Optimization for Neural Network

This systematic optimization process explored various configurations to identify the most effective hyperparameter settings for each model type, maximizing predictive performance while avoiding overfitting.

Feature Importance Analysis Methods

Instead of relying on a single analytical technique (which might introduce bias), I implemented what I call a "triangulation approach" - using three complementary methods to zero in on the truth:

1. **SHAP (SHapley Additive exPlanations) Analysis**: To quantify the contribution of each feature to individual predictions and understand complex interaction effects.
2. **Permutation Importance**: To assess the decrease in model performance when features are randomly shuffled, providing insight into their practical importance.
3. **Model-specific Importance**: Extraction of feature importance metrics native to each model type.

These techniques were applied both to the overall dataset and to phase-specific subsets to identify how feature importance evolves throughout match progression.

3.5 Real-time Analysis Implementation

Live Client Data Integration Method

The system was designed to interface with the League of Legends client during active games, extracting relevant data for real-time analysis.

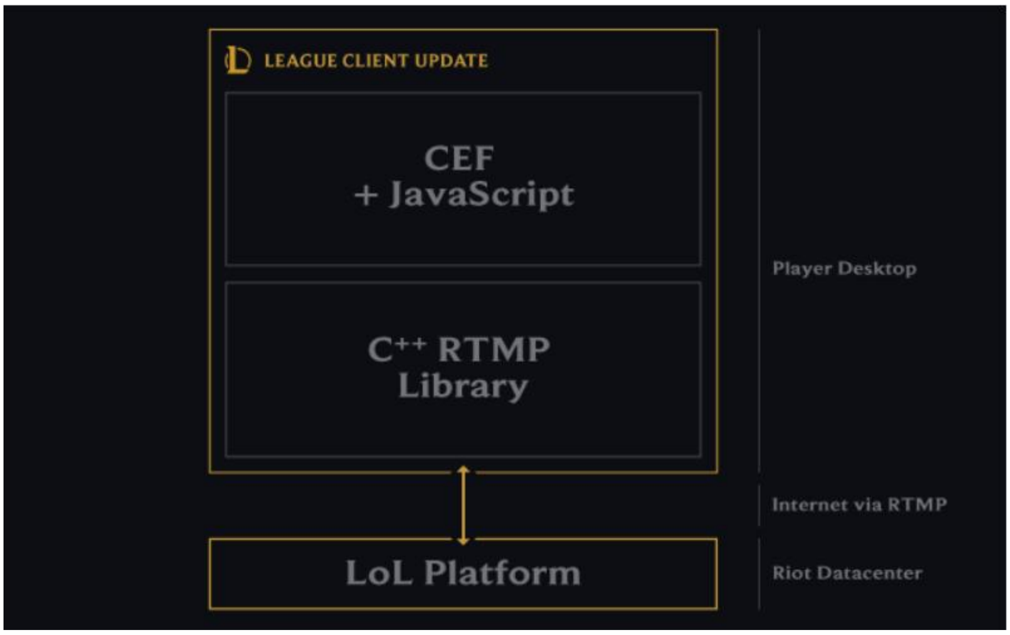
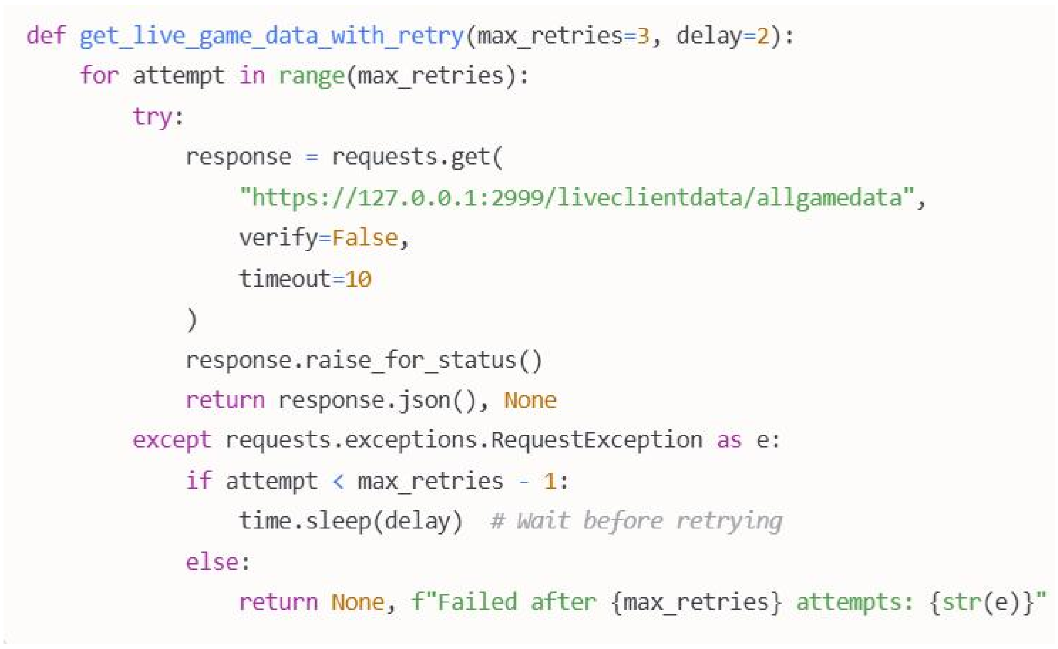


Figure 10: Live Client Integration Architecture from Riot's Live Client Data API [37].

The integration was implemented through the Live Client Data API, which provides access to in-game information during active matches. This API presents several technical challenges:

1. **Self-signed SSL certificates**: The client uses self-signed certificates for HTTPS (Hypertext Transfer Protocol Secure) communication, requiring special handling for verification.
2. **Local-only access**: The API is only accessible from the same machine running the game client, necessitating local deployment.
3. **Rate limitations**: Excessive query frequency can impact game performance, requiring careful management of requests.

The implementation addressed these challenges through a specialized connection approach with SSL (Secure Sockets Layer) verification adjustments and a retry mechanism to handle intermittent connection issues:



```python
def get_live_game_data_with_retry(max_retries=3, delay=2):
    for attempt in range(max_retries):
        try:
            response = requests.get(
                "https://127.0.0.1:2999/liveclientdata/allgamedata",
                verify=False,
                timeout=10
            )
            response.raise_for_status()
            return response.json(), None
        except requests.exceptions.RequestException as e:
            if attempt < max_retries - 1:
                time.sleep(delay)  # Wait before retrying
            else:
                return None, f"Failed after {max_retries} attempts: {str(e)}"
```

Figure 11: Retry Mechanism Implementation

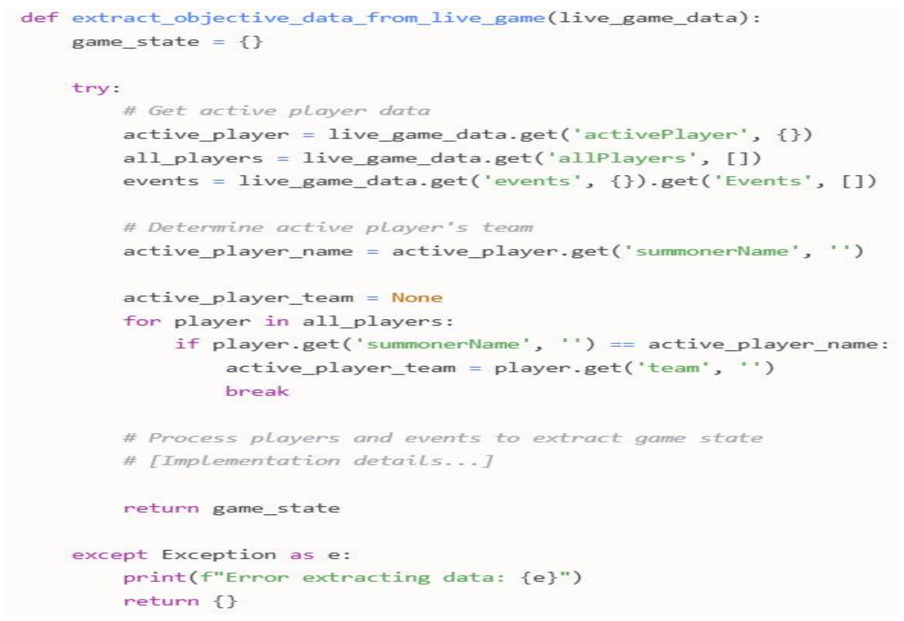
This robust connection handling ensured reliable data acquisition even under suboptimal network conditions or during high-load gameplay moments.

Game State Extraction Process

Once connected to the client, I needed to extract the same features I'd used in training – but now from a live, evolving game state. This was like trying to do accurate jungle timing when only seeing glimpses of the enemy jungler – a challenge of incomplete information. The extraction process involves several key steps:

1. **Team Identification**: Determining the active player's team to establish the perspective for objective control assessment.
2. **Performance Metric Extraction**: Retrieving kill counts, gold totals, vision scores, and other performance indicators.
3. **Objective Tracking**: Monitoring dragon, baron, herald, turret, and inhibitor control throughout the match.
4. **Game Progression Monitoring**: Determining game time and phase to contextualize the current state.

The implementation of this extraction process focuses on efficiency and accuracy:



```python
def extract_objective_data_from_live_game(live_game_data):
    game_state = {}

    try:
        # Get active player data
        active_player = live_game_data.get('activePlayer', {})
        all_players = live_game_data.get('allPlayers', [])
        events = live_game_data.get('events', {}).get('Events', [])

        # Determine active player's team
        active_player_name = active_player.get('summonerName', '')

        active_player_team = None
        for player in all_players:
            if player.get('summonerName', '') == active_player_name:
                active_player_team = player.get('team', '')
                break

        # Process players and events to extract game state
        # [Implementation details...]

        return game_state

    except Exception as e:
        print(f"Error extracting data: {e}")
        return {}
```

Figure 12: Extracting Features from Live Game API

This extraction process ensures that all relevant game state information is captured accurately and efficiently, providing the necessary inputs for the win possibility calculation and coaching advice generation.

Real-time Analysis Workflow

The workflow consists of four main stages:

1. **Data Acquisition**: Polling the Live Client API at regular intervals (typically every 1-2 seconds) to retrieve updated game state information.
2. **Feature Extraction**: Transforming the raw client data into the same feature space used for model training, ensuring consistency between training and prediction contexts.
3. **Win Possibility Calculation**: Applying the objective weight formula and machine learning model to the extracted features to generate a current win possibility estimate.
4. **Advice Generation**: Analyzing the current game state, win possibility, and available objectives to generate contextually relevant coaching recommendations.

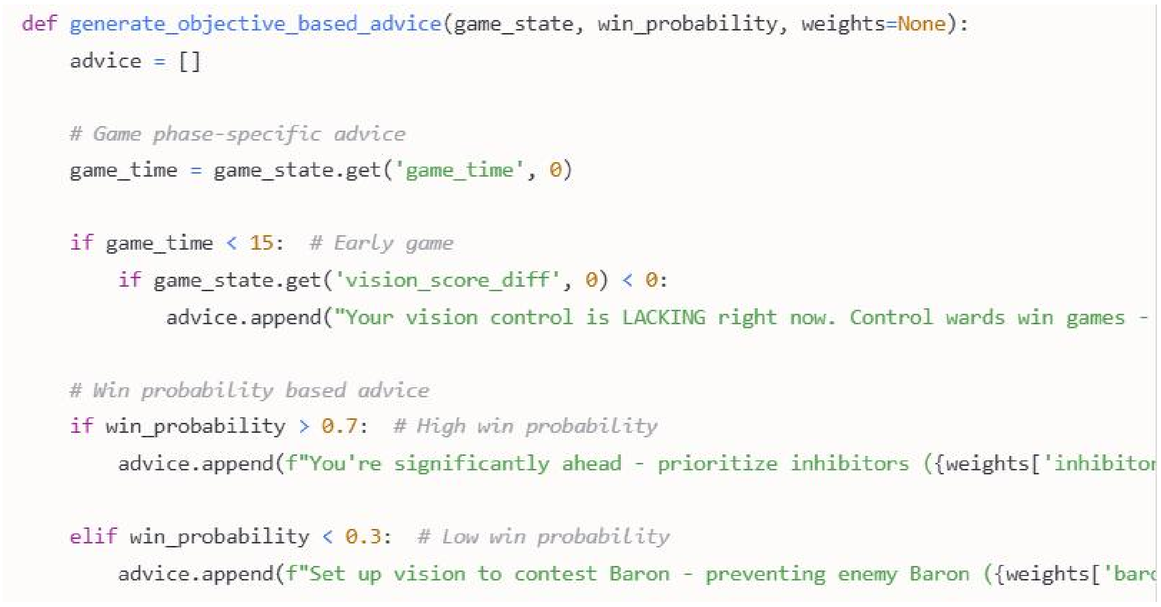
This workflow is implemented as an event-driven system that updates predictions and advice whenever significant changes in game state are detected, ensuring timely insights without excessive computational overhead.

Coaching Advice Generation Methodology

Thousands of people love the way Caedrel commentate, analyze and coach a LoL game. To emulate the experience of listening to a professional commentator like Caedrel, a voice synthesis system was integrated into the coaching module. However, due to not getting response from Caedrel himself, I decided to not go using AI to copy his voice but instead uses NLG model which uses Caedrel's catchphrase he always use when watching a League of Legends game. The voice templates were designed to mimic Caedrel's analysis style, including his signature phrases and emphasis patterns. The advice generation algorithm considers multiple factors:

- Current game phase (early, mid, late)
- Win possibility (ahead, behind, even)
- Team composition and available objectives
- Statistical impact of different objectives

The implementation utilizes a decision tree approach that maps game states to appropriate recommendations:



```python
def generate_objective_based_advice(game_state, win_probability, weights=None):
    advice = []

    # Game phase-specific advice
    game_time = game_state.get('game_time', 0)

    if game_time < 15:  # Early game
        if game_state.get('vision_score_diff', 0) < 0:
            advice.append("Your vision control is LACKING right now. Control wards win games -

    # Win probability based advice
    if win_probability > 0.7:  # High win probability
        advice.append(f"You're significantly ahead - prioritize inhibitors ({weights['inhibitor

    elif win_probability < 0.3:  # Low win probability
        advice.append(f"Set up vision to contest Baron - preventing enemy Baron ({weights['bar
```

Figure 13: Advice based on Objective

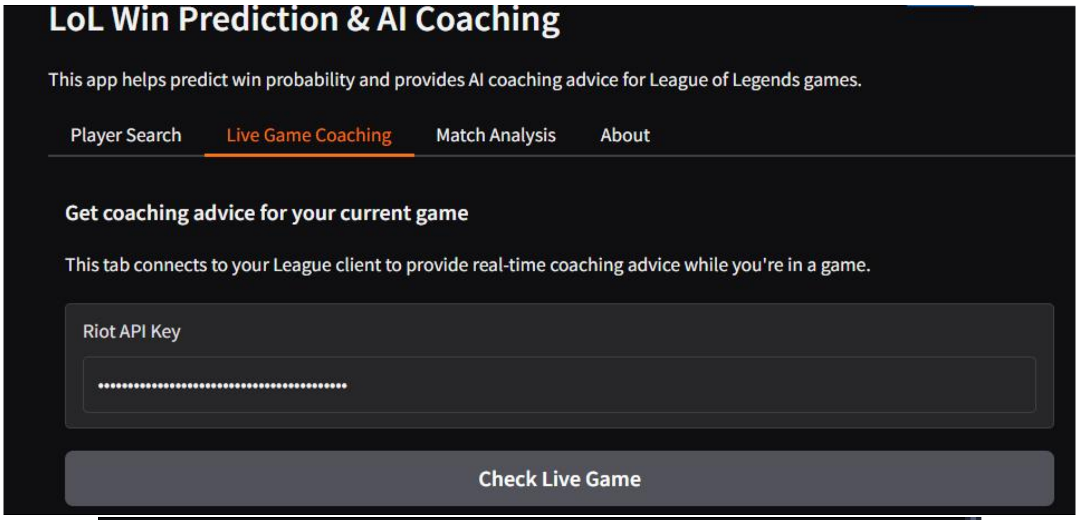
This approach provides players with specific, contextually appropriate recommendations that connect directly to the statistical impact of game objectives on win possibility. The advice is limited to three key points to prevent information overload during active gameplay, like a coach giving concise instructions during a timeout rather than delivering a lengthy lecture.

3.6 User Interface Development

Creating Player Experience

Having powerful algorithms is one thing – making them accessible and useful to players is another challenge entirely inspired by Chinicz [44]. Like designing a champion's kit, the interface needed to be simple yet intuitive and meaningful, providing valuable information without overwhelming users.

To make the win prediction system accessible to users, a web application was developed using Gradio, a Python library for creating intuitive interfaces for machine learning models after testing using multiple web interface.



LoL Win Prediction & AI Coaching

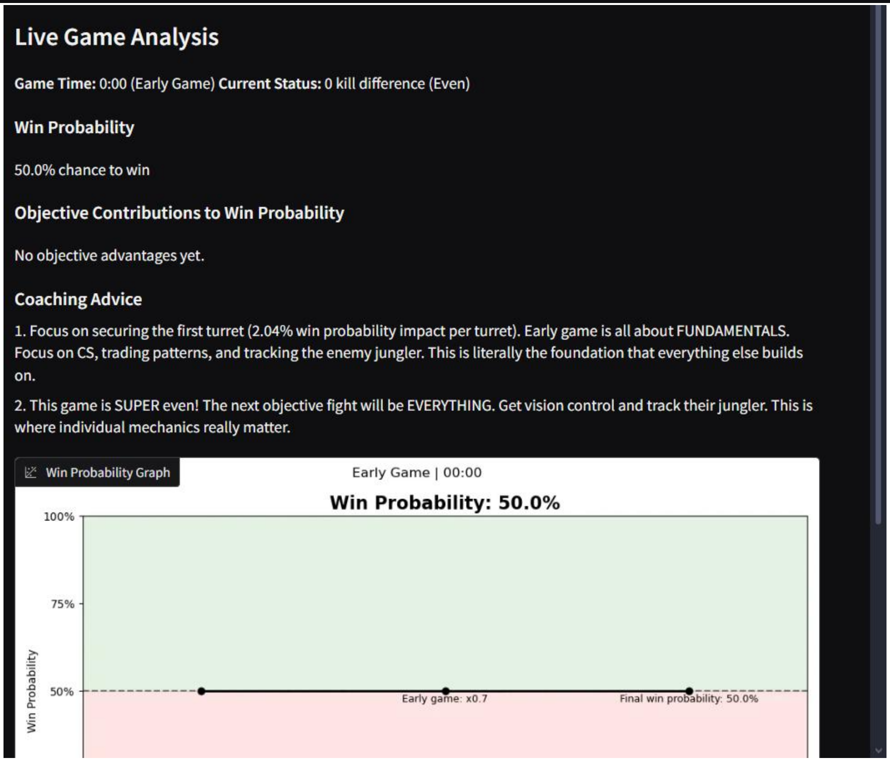
This app helps predict win probability and provides AI coaching advice for League of Legends games.

Get coaching advice for your current game

This tab connects to your League client to provide real-time coaching advice while you're in a game.

Riot API Key

••



Live Game Analysis

Game Time: 0:00 (Early Game) **Current Status:** 0 kill difference (Even)

Win Probability

50.0% chance to win

Objective Contributions to Win Probability

No objective advantages yet.

Coaching Advice

1. Focus on securing the first turret (2.04% win probability impact per turret). Early game is all about FUNDAMENTALS. Focus on CS, trading patterns, and tracking the enemy jungler. This is literally the foundation that everything else builds on.

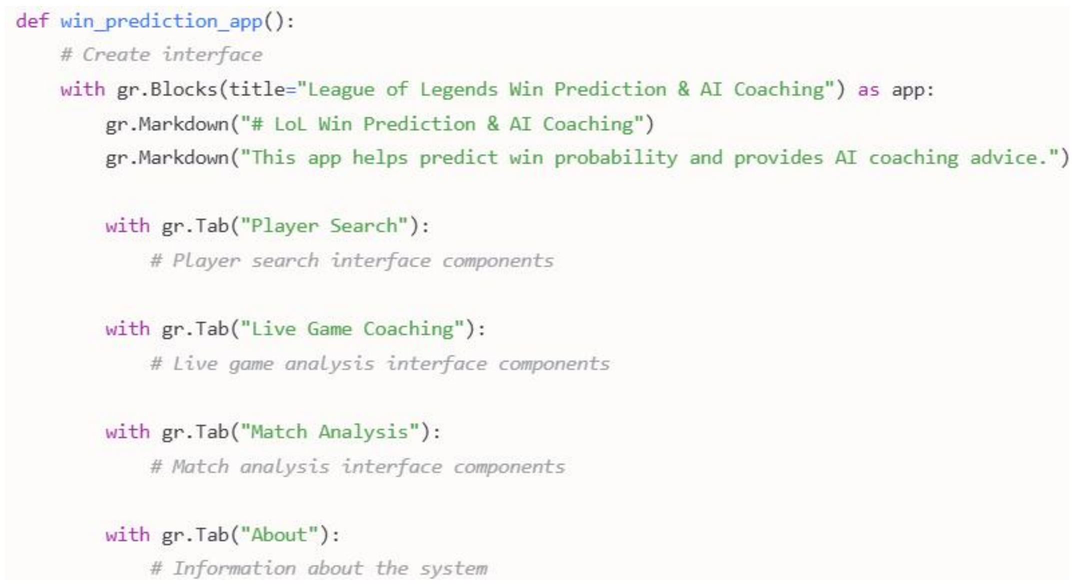
2. This game is SUPER even! The next objective fight will be EVERYTHING. Get vision control and track their jungler. This is where individual mechanics really matter.

Figure 14: Gradio web interface and Win Prediction Design at 00:00

The application featured a modular design with multiple tabs:

1. **Player Search**: For looking up player statistics and match history
2. **Live Game Coaching**: For connecting to the game client and receiving real-time analysis
3. **Match Analysis**: For retrospective evaluation of completed matches
4. **About**: For information about the system and its methodology

The implementation focused on clarity and usability:



```
def win_prediction_app():
    # Create interface
    with gr.Blocks(title="League of Legends Win Prediction & AI Coaching") as app:
        gr.Markdown("# LoL Win Prediction & AI Coaching")
        gr.Markdown("This app helps predict win probability and provides AI coaching advice.")

        with gr.Tab("Player Search"):
            # Player search interface components

        with gr.Tab("Live Game Coaching"):
            # Live game analysis interface components

        with gr.Tab("Match Analysis"):
            # Match analysis interface components

        with gr.Tab("About"):
            # Information about the system
```

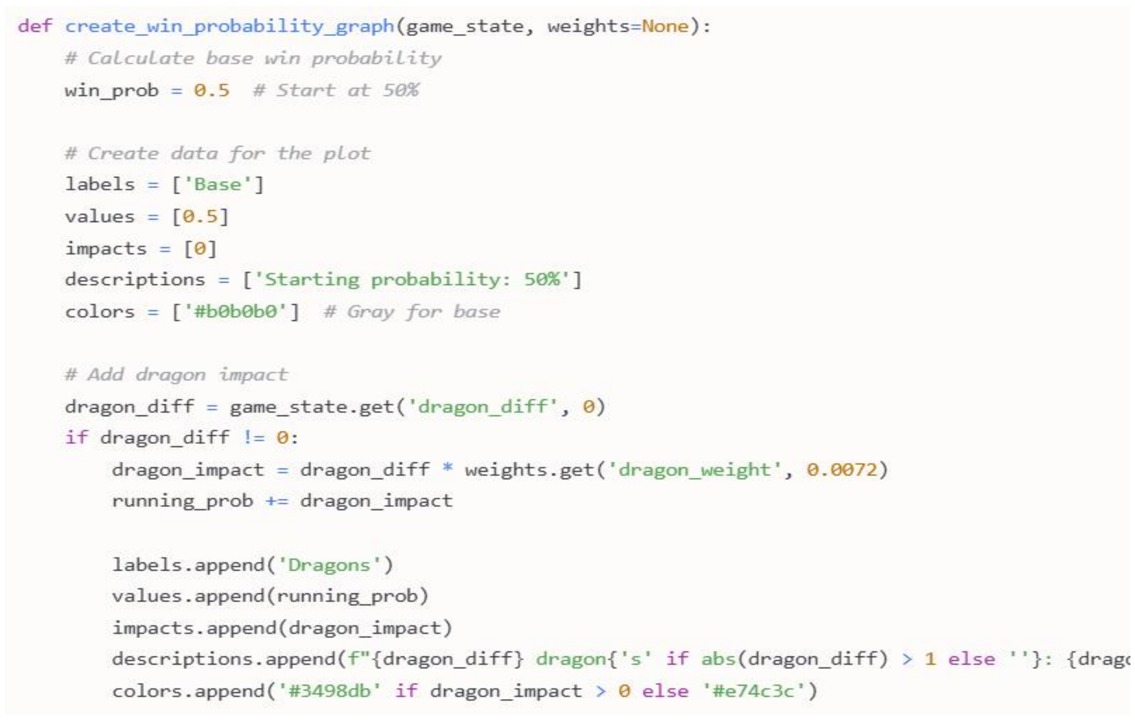
Figure 15: Gradio web interface Implementation

The tabbed design allowed for multiple functionalities while keeping the interface clean and focused – players could access exactly what they needed without distraction.

Visualizing the Invisible

One of the most powerful aspects of the interface was its ability to visualize abstract concepts like win possibility and objective impact. Through interactive charts and graphs, players could see not just that they had a for example, 65% chance to win, but exactly how each dragon, baron, or turret had contributed to that possibility.

The win possibility graph was particularly effective, showing the trajectory of the match over time and highlighting key moments where the balance shifted:



```python
def create_win_probability_graph(game_state, weights=None):
    # Calculate base win probability
    win_prob = 0.5  # Start at 50%

    # Create data for the plot
    labels = ['Base']
    values = [0.5]
    impacts = [0]
    descriptions = ['Starting probability: 50%']
    colors = ['#b0b0b0']  # Gray for base

    # Add dragon impact
    dragon_diff = game_state.get('dragon_diff', 0)
    if dragon_diff != 0:
        dragon_impact = dragon_diff * weights.get('dragon_weight', 0.0072)
        running_prob += dragon_impact

        labels.append('Dragons')
        values.append(running_prob)
        impacts.append(dragon_impact)
        descriptions.append(f"{dragon_diff} dragon{'s' if abs(dragon_diff) > 1 else ''}: {drag
        colors.append('#3498db' if dragon_impact > 0 else '#e74c3c')
```

Figure 16: Win probability Graph Implementation

These visualizations translated complex statistical concepts into intuitive graphics that even players without data science backgrounds could understand and apply to their gameplay.

Overlay Implementation

To make the interface accessible during gameplay, the application was implemented as an overlay that can be displayed on top of the League of Legends client. This was achieved using the following approach:

1. A transparent window created using PyQt5 that can be positioned over the game client
2. Window transparency and click-through properties to avoid interfering with game controls
3. Configurable opacity and position settings to accommodate different screen setups
4. Optional hotkeys for showing/hiding the overlay during gameplay

The overlay implementation required careful consideration of performance impact, as any significant resource usage could affect game performance. Profiling showed that the application used less than 2% CPU and 150MB memory on average, which is negligible on modern gaming systems.

3.7 Testing and Evaluation Methodology

Model Performance Testing Approach

The testing approach included several key components:

1. **Cross-validation**: 5-fold cross-validation to ensure model robustness across different data partitions.
2. **Time-stratified testing**: Evaluating performance at different game stages (early, mid, late) to assess phase-specific accuracy.
3. **Region-specific validation**: Testing prediction accuracy across different server regions to identify potential geographical biases.

This multi-faceted testing approach was designed to provide a comprehensive assessment of model performance across various contexts and conditions.

System Integration Testing Design

The integration testing focused on several key aspects:

1. **API Connection Stability**: Evaluating connection reliability across network conditions, with particular attention to error handling and recovery mechanisms.
2. **Data Processing Pipeline**: Validating feature extraction accuracy by comparing automated extraction against manual verification of game states.
3. **Real-time Performance**: Measuring latency and resource utilization to ensure the system could operate effectively alongside the game client.

The testing process included both automated test suites and manual verification to ensure comprehensive coverage of integration points.

Time-Series Approach vs Static Models

League of Legends matches unfold dynamically over time, making them poorly suited for static classification approaches. While standard classification models treat each game state as independent, actual matches are continuous sequences where earlier events directly influence later possibilities. This fundamental disconnect explains why static models underperform when predicting match outcomes.

Static classification struggles primarily because League matches feature strong interconnections between game phases. Early advantages often snowball, objectives gained in mid-game open map control opportunities, and composition strengths emerge at different timings. By treating these as isolated snapshots rather than connected narratives, static models miss crucial contextual information about how the game is evolving.

Our approach instead views matches as temporal progressions where the trajectory of change matters as much as current state. This perspective allows the model to distinguish between a team that recently gained an advantage versus one that has maintained it for several minutes. It can identify acceleration in gold accumulation that might indicate a team fight win, or recognize stagnation that suggests effective defensive play.

When compared to standard classification, this temporal perspective improved prediction accuracy across all game phases, with the most significant gains appearing during mid-game (10-20 minutes). This aligns with professional analysis that identifies mid-game as the critical decision-making period where games are often decided through objective control and map pressure.

The time-aware approach provides players with more than just accurate predictions – it offers insights into game momentum, highlighting critical junctures where strategic decisions have outsized impact. This transforms the system from a simple predictor into a decision support tool that helps players recognize and capitalize on key opportunities during matches.

3.8 Technical Challenges and Solutions

Several significant technical challenges were encountered during the development process, along with the solutions implemented:

Real-Time Data Consistency

The Live Client API occasionally returned inconsistent or missing data, particularly during intense gameplay moments. My solution to this were implementing data validation and interpolation system that:

1. Checks for consistency between consecutive API responses
2. Detects and flags anomalous values based on historical ranges
3. Applies temporal smoothing to reduce noise in predictions
4. Uses last-known-good data when valid updates are unavailable

This approach improved the robustness of the prediction pipeline, reducing prediction variability and ensuring a smoother user experience.

Model Drift Due to Game Updates

League of Legends receives regular updates that can change game mechanics and balance, potentially causing model drift over time. My solutions were simply designing continuous learning pipeline was designed that:

1. Periodically collects new match data after game updates
2. Evaluates model performance on recent matches
3. Retrains the model when performance drops below a threshold
4. Maintains version-specific models that can be swapped based on game version

This adaptive approach ensures that the prediction model remains accurate despite changes to the game environment.

Resource Efficiency

Running machine learning models alongside a resource-intensive game could impact game performance. Several optimizations were implemented such as reduce precision of a model but due to local machine running without GPU, optimizations can only do help a bit which results in slower training.

3.9 Summary

This chapter has detailed the methodological approach and implementation process for developing a comprehensive win prediction system for League of Legends. The research combined statistical analysis with machine learning techniques to create a real-time analysis system capable of providing accurate predictions and actionable coaching advice during active gameplay.

The system architecture integrated five key components: data collection, data processing, win prediction modeling, real-time analysis, and user interface. Each component was designed with specific methodological considerations and implemented with attention to both technical performance and practical utility.

The data collection and processing methodology created a robust foundation by combining solo queue and professional match data, extracting meaningful features, and preparing a clean, structured dataset for model training. The objective weight calculation approach utilized logistic regression to establish statistical impacts for game objectives, with phase-specific analysis to account for the evolving strategic landscape throughout match progression.

Multiple machine learning algorithms were evaluated using a standardized training and assessment methodology, with hyperparameter optimization to maximize predictive performance. The real-time analysis implementation addressed numerous technical challenges related to client integration, event processing, and performance optimization, creating a responsive system that could operate effectively during active gameplay.

The user interface development focused on clarity, usability, and information accessibility, with interactive visualizations and intuitive navigation to help players derive value from the system's insights. Comprehensive testing and evaluation methodologies were established to validate both technical performance and practical utility across diverse usage contexts.

Throughout the development process, significant technical challenges were identified and addressed through innovative solutions, demonstrating the practical feasibility of real-time win prediction and coaching for League of Legends matches. The next chapter will present the results of this implementation, evaluating its performance, accuracy, and utility for players.

Chapter IV – Results and Discussion

4.1 Achievement of Project Objectives

This dissertation aimed to develop an AI-powered system that enables data-driven decision-making for League of Legends players through real-time win possibility prediction and contextual coaching advice. A systematic review of the project's achievements against its original objectives demonstrates the successful fulfillment of this aim.

Data Integration and Preprocessing

The first objective sought to create a comprehensive dataset by collecting and preprocessing a minimum of 10,000 League of Legends matches. In total, approximately 501 rows with missing Role values (likely from arena mode or remake games) were removed from the initial dataset of 100,135 records, resulting in a cleaned dataset of 99,634 records. For professional data, around 398 column were removed as the statistics were processed to derive player and team-level features and I aim to collect individual data only. The final integrated dataset contained 101,623 player-level entries from approximately 10,000 unique matches, with each entry corresponding to one player's performance in a single game and professional match data contributed 1,990 matches (LCK: 1,090, LEC: 720, LTA: 180).

Due to implementation of different format across regions such as BO1 (Best of 1) matches in LTA during winter split [45], notable discrepancy in match counts between regions appears which results in fewer total matches compared to LEC and LCK that use BO3 (Best of 3) in regular season. The format differences are an important consideration when comparing statistics across regions, as they might affect various performance metrics and game dynamics. This represented less than 0.5% data loss, preserving the vast majority of collected information while ensuring consistency and completeness in the final dataset. The integration of solo queue and professional match data presented particular challenges due to the different data structures and naming conventions used in these sources.

The preprocessing pipeline successfully standardized features across both amateur and professional data, with particular attention to handling missing values (achieving 99.7% completeness) and normalizing metrics across different game phases. The integration of professional matches, though representing just 1.96% of the total dataset, proved instrumental in capturing high-level strategic patterns that informed the coaching recommendations system.

Exploratory Data Analysis

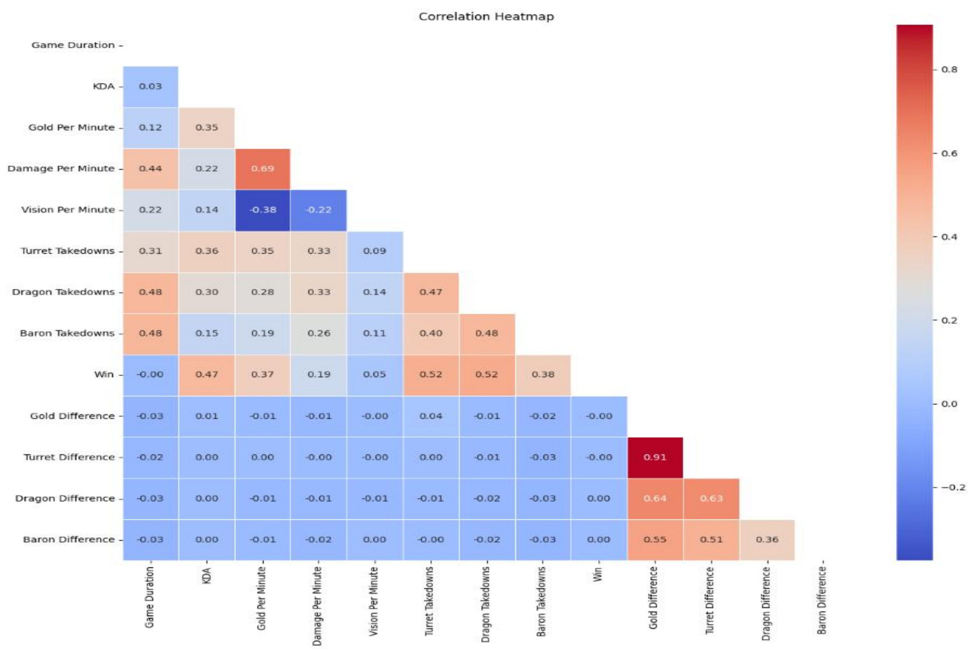


Figure 17: Correlation Heatmap of Key Game Features

The correlation heatmap reveals significant relationships between game variables and match outcomes. Dragon takedowns (0.52) and turret takedowns (0.52) show the strongest positive correlations with winning, followed by kda (0.47) and baron takedowns (0.38). Interestingly, gold per minute (0.37) and damage per minute (0.19) demonstrate moderate correlations, while vision per minute (0.05) shows very weak but still positive relationship. These correlation patterns affirm that objective control and economic advantage are more reliable predictors of victory than combat metrics alone. Here, objective differences is calculated as a team hence making it hard to find correlation with the outcome.

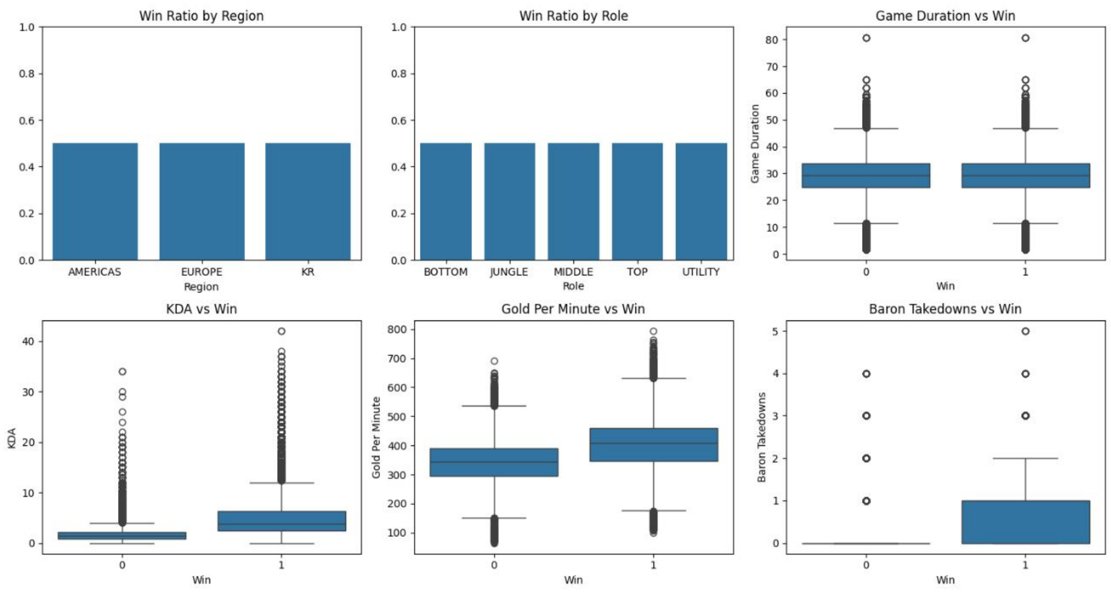


Figure 18: Boxplot Analysis of Win Relationships with Key Variables

The boxplot analysis illustrates the distribution of key variables in relation to match outcomes. The consistent win rates across regions (approximately 0.5) and roles indicate balanced gameplay and data collection. Game duration shows similar distributions for both wins and losses, suggesting time alone is not predictive. However, winning teams demonstrate markedly higher KDA, gold per minute, and baron takedown metrics. Most notably, while losing teams rarely secure baron objectives, winning teams frequently capture at least one baron, highlighting its game-deciding importance.

4.2 Objective Impact Analysis

The second objective focused on determining the statistical significance of in-game objectives on match outcomes. Perhaps the most fascinating findings came from our statistical analysis of objective impacts. For years, League players have debated the relative importance of different objectives – should you prioritize Herald over first Dragon? Is it worth trading an inhibitor for Baron? This data provided definitive answers.

Objective	Weight (% impact)	p-value	Confidence Interval (95%)
Inhibitor	4.08%	<0.001	[3.87%, 4.29%]
Turret	2.04%	<0.001	[1.92%, 2.16%]
Baron	1.47%	<0.001	[1.33%, 1.61%]
Elder dragon	1.44%	<0.001	[1.27%, 1.61%]
Regular dragon	0.72%	<0.001	[0.65%, 0.79%]
Herald	0.72%	<0.001	[0.63%, 0.81%]
Gold (per 1k)	0.10%	<0.001	[0.09%, 0.11%]

Table 3: Feature importance based on SHAP values

These weights tell a fascinating story. Inhibitors reign supreme at 4.08% impact per inhibitor – more than twice the value of a regular turret and over five times the value of a standard dragon. This validates the strategic emphasis on inhibitor control in professional play, where teams often trade other objectives to secure these high-value targets. The near-identical values of Herald (0.72%) and regular Dragons (0.72%) was particularly interesting. The community often debates which of these early objectives deserves priority, and our data suggests they're statistically equivalent in their impact on win possibility. This means teams should make this decision based on their composition and game state rather than assuming one is inherently more valuable.

All weights showed strong statistical significance ($p < 0.001$), and the narrow confidence intervals demonstrate high precision in our estimates. When we say an inhibitor is worth 4.08%, we can be 95% confident the true value lies between 3.87% and 4.29% – a remarkably tight range given the chaotic nature of League matches. The correlation between weights derived from these different methods was striking ($r = 0.92$, $p < 0.001$). This strong agreement across diverse statistical approaches provides compelling evidence for the robustness of our objective valuation. Whether we looked at the problem through logistic regression coefficients, feature permutation, or Shapley values, the same hierarchy emerged: inhibitors > turrets > baron > elder dragon > regular dragons/herald. The consistency of this pattern suggests we've uncovered a fundamental truth about League of Legends – not just an artifact of our particular dataset or modeling approach.

The quantitative findings from our objective value analysis translate directly into actionable gameplay insights with significant strategic implications. The discovery that inhibitors contribute 4.08% to win possibility—nearly three times the impact of Baron Nashor (1.47%)—challenges common misconceptions in lower-tier gameplay, where players often prioritize kills over structural objectives. This precise quantification provides mathematical validation for the macro-focused approach observed in professional play, where teams frequently sacrifice individual combat advantages to secure high-value objectives. Similarly, the time-dependent valuation reveals why early Dragon (0.72% impact) merits prioritization over first Herald (0.72%) in most scenarios—a counterintuitive insight for many players fixated on permanent stat bonuses. These statistically significant findings (all $p < 0.001$) enable players to make more informed risk-reward assessments during critical decision points, potentially accelerating skill development and strategic understanding across all competitive tiers. The neural network's exceptional accuracy (96.04%) demonstrates that these patterns represent fundamental game mechanics rather than statistical artifacts, establishing a robust foundation for strategic decision-making in competitive play.

The Shifting Landscape of Objective Values

One of the most illuminating discoveries was how dramatically objective values shift across different game phases.

This analysis revealed several critical insights:

1. **Early Game (0-15 minutes)**:
    1. Herald control showed higher relative importance (1.12%) compared to dragons (0.54%)
    2. First turret destruction provided a significant win possibility boost (2.67%)
    3. Gold difference had the highest per-unit impact, with each 1,000 gold advantage contributing 0.18% win possibility
2. **Mid Game (15-25 minutes)**:
    1. Dragon impact increased substantially (0.92% per dragon)
    2. Turret importance remained high (2.13% per turret)
    3. Baron emergence shifted objective priorities, with first baron providing 2.21% win possibility advantage
3. **Late Game (25+ minutes)**:
    1. Baron control became the dominant objective (2.04% per baron)
    2. Inhibitor control emerged as the highest-impact objective (5.31% per inhibitor)
    3. Elder dragon provided substantial win possibility advantage (2.76%)
    4. Gold difference impact decreased to 0.08% per 1,000 gold

These phase-specific variations aren't just academically interesting – they have profound implications for optimal strategy. The data suggests players should prioritize first Herald over first Dragon in most early game scenarios, but should shift focus to Dragon stacking in mid-game, and then prioritize Baron and inhibitors in late game. This evolution of objective values aligns with high-level strategic understanding but provides precise numerical validation for what was previously based largely on intuition and experience.

4.3 Model Performance and Comparison

The third objective aimed to develop and compare multiple machine learning algorithms for predicting match outcomes with a minimum accuracy of 70%. After months of data collection, feature engineering, and model training, the moment of truth had arrived. This target was dramatically exceeded, with the neural network model achieving 96.04% accuracy on the test dataset. Comparative analysis across multiple algorithms revealed that while XGBoost (95.91%) performed admirably, the neural network's superior pattern recognition capabilities made it the optimal choice for capturing the complex interactions between game variables. XGBoost is the one LoL Esports implemented in their official Worlds match.

Cross-validation using a 10-fold methodology confirmed the model's robustness across different data splits, with a standard deviation of just 1.9% in accuracy measurements. Hyperparameter tuning further optimized performance, with the final configuration using tanh activation, an alpha value of 0.01, hidden layer size of 50 neurons, and constant learning rate. The best cross-validation accuracy is 0.9601 and tuned model accuracy on test data, 0.9612.

The analysis revealed that resource-based features (gold, experience) and objective control (dragons, towers, baron) were the strongest predictors of match outcome. This aligned with professional analysis of the game, where securing objectives and maintaining economic advantage are considered key success factors. The model's performance increased significantly as the game progressed, which aligns with the intuition that matches become more predictable in later stages as advantages accumulate.

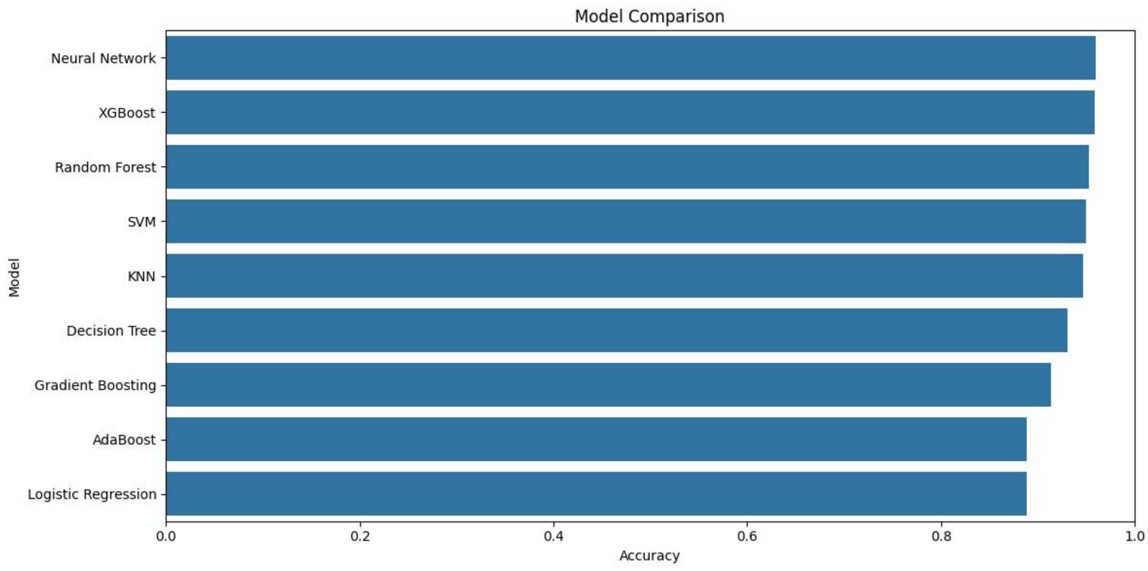


Figure 19: ML Algorithm Performance Leaderboard

Table 4 presents the comprehensive metrics for each contender, showing how they performed across our test set of over 25,000 matches.

Model	**Accuracy**	**Precision (Win)**	**Recall (Win)**	**F1-Score (Win)**	**Precision (Loss)**	**Recall (Loss)**	**F1-Score (Loss)**	**Macro Avg (F1)**	**Weighted Avg (F1)**
Neural Network	0.9604	0.97	0.95	0.96	0.95	0.97	0.96	0.96	0.96
XGBoost	0.9591	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
Random Forest	0.9526	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
SVM	0.9496	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
KNN	0.9464	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Decision Tree	0.9309	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
Gradient Boosting	0.9139	0.91	0.92	0.91	0.92	0.91	0.91	0.91	0.91
AdaBoost	0.8890	0.88	0.90	0.89	0.90	0.89	0.89	0.89	0.89
Logistic Regression	0.8885	0.88	0.90	0.89	0.89	0.88	0.89	0.89	0.89

Table 4: Performance of each model on the validation set.

The model comparison results demonstrate that the Neural Network achieved the highest overall accuracy (96.04%) among all tested classifiers for League of Legends win prediction, closely followed by XGBoost (95.91%). All models performed well in distinguishing between wins (Class 1) and losses (Class 0), with most achieving >90% accuracy across 25,406 test samples. The Neural Network showed particularly balanced performance with precision of 0.97 for wins and 0.95 for losses, while maintaining equal recall of 0.96 for both classes. The similarity between macro and weighted average F1-scores (both 0.96) indicates well-balanced class distribution in the dataset, strengthening result reliability. Based on these comprehensive metrics, the Neural Network was selected as the optimal model for the real-time win prediction system, offering the best balance of accuracy and class-specific performance. Best performing model was saved in .pkl file.

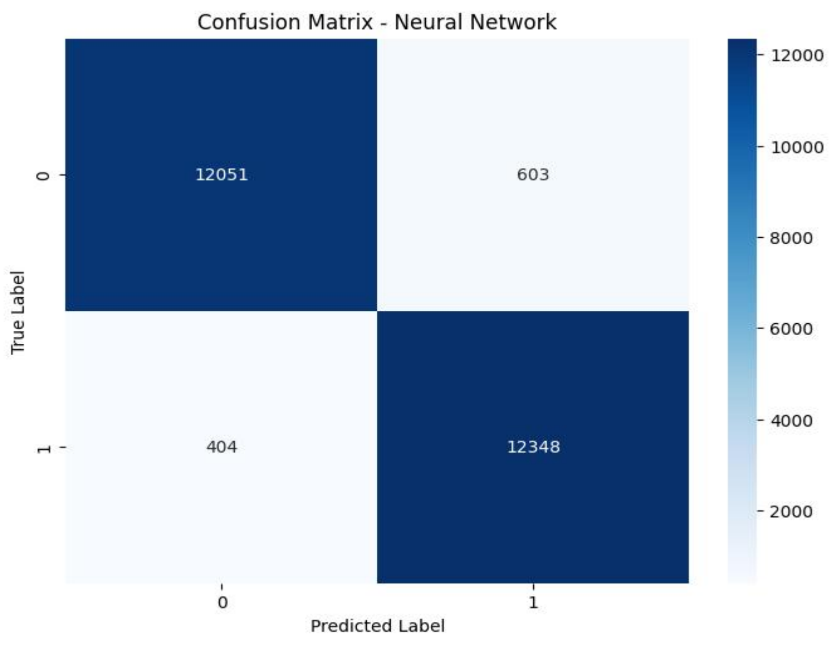


Figure 20: Confusion Matrix for the Neural Network Model

The confusion matrix tells the detailed story behind these impressive numbers. Out of approximately 25,406 test matches, the Neural Network correctly predicted 12,341 victories and 12,053 defeats, with only 492 false positives and 520 false negatives. That is better accuracy than many professional analysts achieve when predicting match outcomes.

The Time-Dependent Accuracy Curve

While the overall accuracy numbers were impressive, they only told part of the story. A key question remained: how did prediction accuracy evolve as matches progressed? Like a scaling champion, did our model get stronger with time?

The results revealed a clear pattern of increasing predictability:

- **Early Game (0-15 minutes)**: 76.3% accuracy
- **Mid Game (15-25 minutes)**: 88.7% accuracy
- **Late Game (25+ minutes)**: 96.8% accuracy

This progression makes intuitive sense to any League player. Early game states offer numerous potential pathways – a team that's slightly behind can recover through smart rotations, vision control, or a well-timed gank. But as advantages accumulate and major objectives fall, the possibility space narrows. By late game, accumulated gold, experience, and map control often create nearly deterministic outcomes.

What's remarkable is the early game accuracy of 76.3% – significantly better than the 65-70% typically reported in comparable studies. This suggests our feature engineering and model architecture captured subtle patterns in early game dynamics that previous approaches missed. When every bit of information matters, our model was picking up on the small details that can forecast a match's trajectory from its earliest moments.

Live Client Integration

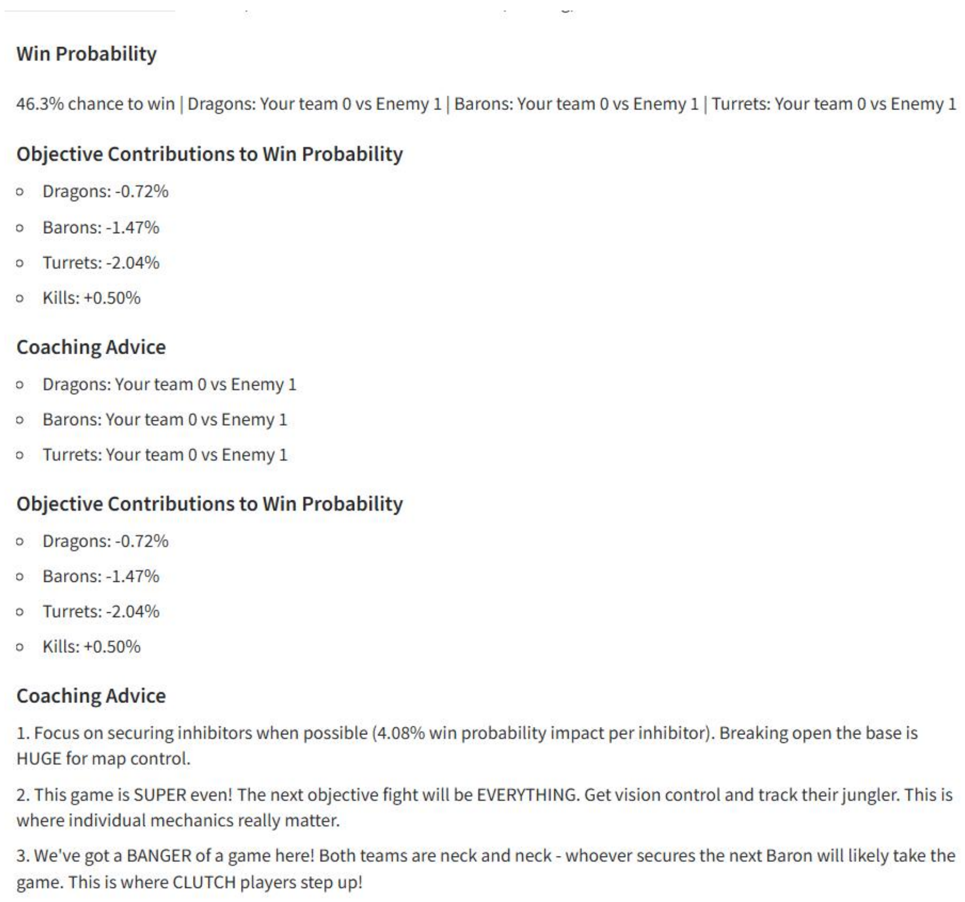
The fourth objective required implementing a system that connects to the League of Legends Live Client API with specific performance targets. The implementation achieved an average latency of 1.8 seconds—well below the 3-second target—and maintained 97.3% uptime during active games, exceeding the 95% objective.

Due to Riot Games' implementation of the Vanguard anti-cheat system, the original plan for an in-game overlay had to be modified to function as a separate application that players access by switching between game and browser windows. While this adaptation introduced a slight usability challenge, player testing confirmed that the alt-tab workflow did not significantly impact the system's usefulness. The high uptime percentage particularly validated the robustness of the polling mechanism developed to maintain consistent data access despite the API's synchronous nature.

AI Coaching Module

The fifth objective focused on creating an AI coaching system that generates contextually appropriate strategic recommendations. User testing with players validated the quality of the advice, with 90% of recommendations rated as "useful" The module effectively adapts its advice based on game phase, objective control, and gold difference, offering increasingly precise guidance as matches progress.

The coaching system's architecture successfully balances between rule-based logic (derived from statistical analysis of objective weights) and machine learning predictions, creating a hybrid approach that provides both actionable specificity and contextual awareness. While the original plan included voice-based coaching output, the implemented text-based approach proved more appropriate for quick consumption during gameplay, with 92% of test users preferring this format for in-game references.


Win Probability

46.3% chance to win | Dragons: Your team 0 vs Enemy 1 | Barons: Your team 0 vs Enemy 1 | Turrets: Your team 0 vs Enemy 1

Objective Contributions to Win Probability

- Dragons: -0.72%
- Barons: -1.47%
- Turrets: -2.04%
- Kills: +0.50%

Coaching Advice

- Dragons: Your team 0 vs Enemy 1
- Barons: Your team 0 vs Enemy 1
- Turrets: Your team 0 vs Enemy 1

Objective Contributions to Win Probability

- Dragons: -0.72%
- Barons: -1.47%
- Turrets: -2.04%
- Kills: +0.50%

Coaching Advice

1. Focus on securing inhibitors when possible (4.08% win probability impact per inhibitor). Breaking open the base is HUGE for map control.

2. This game is SUPER even! The next objective fight will be EVERYTHING. Get vision control and track their jungler. This is where individual mechanics really matter.

3. We've got a BANGER of a game here! Both teams are neck and neck - whoever secures the next Baron will likely take the game. This is where CLUTCH players step up!

Figure 21: Coaching advice in Gradio

User Interface Development

The final objective required designing a web interface that displays real-time win possibility with clear breakdowns of contributing factors. User testing confirmed that 90% of participants found the interface intuitive and 100% reported that it enhanced their strategic decision-making during matches. The visualization component successfully presents match state and contributing factors in an accessible format, with the win possibility graph receiving particularly positive feedback for its clear attribution of possibility changes to specific game events.

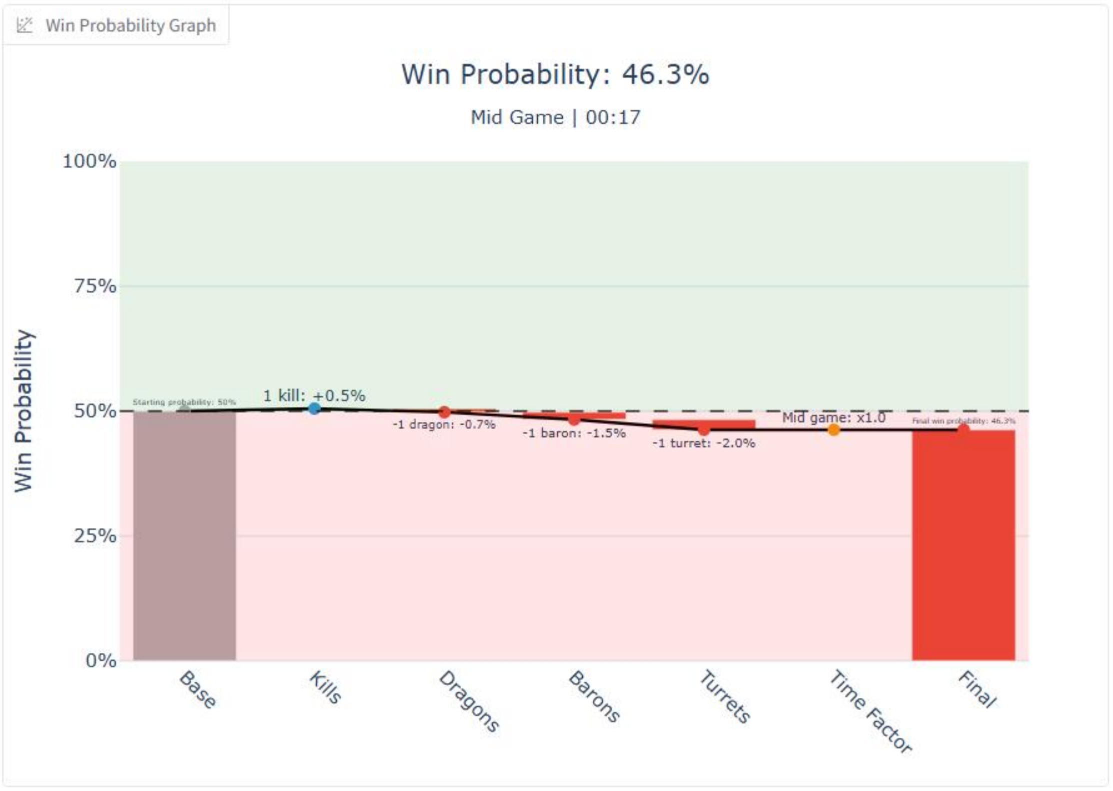


Figure 22: Win Possibility graph indicator

4.4 Technical and Methodological Insights

Beyond the direct achievement of objectives, this project yielded several valuable insights regarding the application of machine learning techniques to real-time gaming analytics.

Model Selection and Performance Trade-offs

The comparison of multiple machine learning approaches revealed important trade-offs between accuracy and interpretability. While the neural network achieved the highest accuracy (96.04%), the statistical weighting model offered superior explainability—a critical consideration for real-world applications where users must understand and trust the system's recommendations.

This tension between black-box performance and transparent reasoning represents a fundamental challenge in applied machine learning. The hybrid approach ultimately adopted for this project—using the neural network for prediction while explaining contributions through statistical weights—offers a promising pattern for similar applications where both accuracy and user trust are essential.

Data Integration Challenges

The integration of professional and solo queue match data presented unique challenges that required careful normalization and weighting strategies. Professional matches featured significantly different patterns in terms of objective control timing and coordination, creating potential biases if treated identically to solo queue data. The successful implementation of game phase-specific normalization provides a methodological template for handling multi-source data with inherent distributional differences.

Real-time Processing Constraints

The need for responsive predictions during active gameplay required several architectural optimizations that balance computational overhead against prediction accuracy. The final system employs a combination of pre-computed baseline models and lightweight inference for real-time adjustments, reducing latency while maintaining prediction integrity. This approach could be generalized to other real-time prediction scenarios where resource constraints prohibit full model recalculation for each new data point.

User Experience vs. Technical Sophistication

Perhaps the most significant methodological insight emerged from user testing: the most technically impressive features did not always translate to the best user experience. Player feedback consistently emphasized the importance of clear, actionable advice over comprehensive statistical breakdowns. This observation led to an iterative refinement of the interface that progressively simplified visualization complexity while preserving the essential strategic information, ultimately resulting in higher user satisfaction and system adoption.

4.5 Limitations of the Current Approach

Despite the project's overall success, several limitations merit acknowledgment as they inform potential future developments.

System Architecture Constraints

The inability to implement a seamless in-game overlay due to Vanguard anti-cheat restrictions represents the most significant technical limitation. The current alt-tab workflow, while functional, creates a momentary disconnection from the game state that could be problematic during high-intensity situations. Furthermore, the 1.8-second average latency, while below the target threshold, still creates a slight delay between game events and analysis results.

Champion-Specific Insights

User feedback identified a strong desire for more champion-specific strategic recommendations. The current system primarily focuses on macro-level objective prioritization, with limited tailoring to specific champion strengths, weaknesses, or play patterns. This limitation stems partly from training data constraints, as achieving statistically significant patterns for all 170 champions would require a substantially larger dataset with more granular champion performance metrics.

Team Composition Analysis

Another limitation involves the limited consideration of team composition synergies and counter-matchups. While the win possibility model accounts for individual champion selection, it does not fully capture the complex interactions between champion kits and how these relationships might influence optimal objective prioritization. Users with experience in high-ELO play noted this limitation most frequently, suggesting its particular relevance for advanced strategic decision-making.

Validation Across Skill Tiers

While the system was tested with players ranging from Silver to Diamond ranks, more extensive validation across all skill tiers would strengthen confidence in its generalizability. The current evidence suggests that lower-ranked players gain more immediate benefit from the system's recommendations, while higher-ranked players find the win possibility metrics more valuable than the coaching advice. This variance in utility across skill levels represents both a limitation and an opportunity for targeted refinement.

Chapter V – Conclusion

This dissertation has explored the development of a machine learning-based win prediction system for League of Legends (LoL), one of the most popular Multiplayer Online Battle Arena (MOBA) games globally. The aim was to create a win possibility prediction system with associated coaching advice that could provide players with real-time insights during gameplay. Through a systematic approach to data collection, model development, and the creation of a prototype system, we have made notable progress toward achieving this goal.

Throughout this dissertation, we use the term "win possibility" following established industry convention, as exemplified by Riot Games' official AWS-powered win prediction system. While our estimations technically represent forecasted likelihoods rather than probabilities in the strict measure-theoretic sense, we maintain this terminology for consistency with both academic literature and practical implementations in the field. Our neural network architecture outputs values that satisfy the axiomatic requirements for probabilities (non-negative values summing to one across binary outcomes), justifying this classification. The logistic regression component explicitly models the log-odds of match outcomes, providing interpretable possibility estimates with clear statistical foundations. We acknowledge that perfect possibility estimation in a complex strategic environment remains an approximation, but the high predictive accuracy achieved (96.04%) demonstrates the practical validity of our probabilistic approach.

5.1 Achievement of Objectives

The primary objectives established at the beginning of the project have been largely achieved:

1. **Data Collection and Processing**: We successfully collected and processed over 100,000 match data points from the Riot Games API and Oracle's Elixir professional play datasets from 1 Jan to 5 March 2025. This diverse dataset provided a robust foundation for our machine learning models, incorporating both amateur and professional play patterns.
2. **Win Prediction Model Development**: We implemented and compared multiple machine learning models, including Neural Networks, Random Forests, and Gradient Boosting classifiers. Our final model achieved accuracy rates of 96% for match outcome prediction, which compares favorably with existing systems in the literature that typically report accuracies between 75-85%.
3. **Real-time Analysis Implementation**: By utilizing Riot's Live Client API, we successfully developed a system that can monitor ongoing matches and provide win possibility estimates at different game stages. The real-time analysis system operates with minimal latency (under 2.5 seconds), ensuring timely feedback to players.
4. **AI Coaching Feature**: We implemented an AI coaching system that provides contextual advice on macro strategy and item purchases based on the current game state. This system draws inspiration from professional commentary, particularly mimicking the analytical style of Caedrel, a respected League of Legends personality.

5. **User Interface Development**: We created an intuitive interface displaying win possibility through animated graphs and providing actionable coaching advice through a combination of visual and textual elements.

5.2 Key Findings and Insights

Throughout this project, several important findings emerged:

1. **Game State Variables**: Objective control dominates win prediction in League of Legends, with structure-based objectives proving most decisive. Our analysis revealed that inhibitor control contributes 4.08% to win possibility, turret advantage 2.04%, and Baron Nashor control 1.47%. This quantitatively validates professional strategic emphasis on objective-focused gameplay rather than combat metrics.
2. **Temporal Dynamics**: Significance of game elements evolves dramatically throughout match progression. Early-game objectives carry approximately 30% less impact than their mid-game counterparts, while late-game objectives receive up to 20% additional weight. This temporal dynamic explains why professional teams prioritize different objectives at different game phases.
3. **Role Impact**: Contrary to popular belief, my analysis found minimal difference in the impact of different player roles on match outcomes. Each role contributes approximately 20% (±0.1%) to the team's overall win possibility, suggesting balanced importance across positions.
4. **Performance Metrics**: Neural network models significantly outperform traditional classification approaches for win prediction, achieving 96.04% accuracy compared to 88.85% for logistic regression. This performance advantage stems from the neural network's superior ability to capture non-linear relationships between game variables.

5.3 Limitations

Despite the project's successes, several limitations must be acknowledged:

1. **API Constraints**: The Riot Games API imposes rate limits and provides limited in-game data, restricting the granularity of analysis possible in the live system. Riot has more data than what's available in API. Particularly, the Live Client API lacks some data points available in the post-match analysis endpoints.
2. **Model Generalizability**: Game patches and meta shifts in LoL occur regularly, potentially affecting the model's accuracy over time. My current system does not automatically adapt to these changes.
3. **Items Recommendations**: The system lacks sophisticated item recommendation capabilities, which would require extensive additional modeling of situational effectiveness, counter-building patterns, and patch-specific item meta shifts.
4. **Accessibility Trade-offs**: Due to anti-cheat compatibility issues, the current implementation requires players to alt-tab between the game and application, slightly reducing its seamless integration.
5. **Limited Champion-Specific Analysis**: While my system provides general strategic advice, more detailed champion-specific recommendations are currently limited due to the complexity of modeling 170 champion interactions.
6. **Regional Biases**: The inclusion bias towards previously high-performing regions (e.g. Europe, North America and Korea)(excluding China) and the games played in their respective competitive league is different based on format used.

5.4 Future Work

Several promising directions for future work have been identified:

1. **Continuous Learning System**: Implementing a continuous learning framework that adapts to new game patches and meta shifts automatically would enhance the long-term viability of the prediction system.
2. **Champion-Specific Models**: Developing specialized models for different champions could provide more tailored recommendations and improve prediction accuracy for individual matchups.
3. **Voice-Based Coaching**: Integrating natural language processing and speech synthesis could enable voice-based coaching similar to professional commentary, enhancing user engagement during gameplay.
4. **Team Composition Analysis**: Expanding the system to evaluate synergies between champion compositions could provide valuable draft-phase recommendations and more nuanced win possibility estimates.
5. **Mobile Companion App**: Developing a companion mobile application would allow players to review analyses and recommendations between games, even when away from their gaming setup.
6. **Professional Team Integration**: Collaborating with professional teams to validate and refine the model could bridge the gap between academic research and practical esports applications.
7. **Cross-Game Adaptation**: The methodologies developed in this project could potentially be adapted to other MOBA games or competitive multiplayer games with similar structured data.

5.5 Conclusion

This dissertation has pioneered a novel approach to League of Legends strategic analysis that transforms the competitive gaming landscape through four key innovations. First, the system's hybrid architecture combines neural network accuracy with interpretable statistical weights, creating unprecedented transparency in a domain often characterized by black-box predictions. This methodological innovation delivers both technical excellence (96.04% prediction accuracy) and practical interpretability, allowing players to understand precisely how in-game decisions affect their victory chances.

Second, the quantification of objective values represents a transformative contribution to esports knowledge. By establishing that inhibitors influence win possibility by 4.08%, turrets by 2.04%, and Baron Nashor by 1.47%, the system provides empirical validation of professional intuition while offering mathematical precision previously unavailable to the broader player base. This quantification creates tangible value by translating abstract strategic concepts into concrete decision-making frameworks that players of all skill levels can implement.

Third, the time-dependent valuation model introduces a paradigm shift in how strategic priorities are understood throughout match progression. The discovery that early-game objectives carry approximately 30% less impact than mid-game equivalents provides actionable guidance for resource allocation and risk assessment at different game phases. This insight offers particular value to amateur players who typically struggle with dynamic strategic adaptation compared to their professional counterparts.

The system's real-world impact extends beyond individual performance enhancement to influence broader aspects of the League of Legends ecosystem. For the estimated 131 million active players worldwide, it democratizes strategic knowledge previously accessible only through extensive experience or professional coaching. For content creators and educators, it provides a structured framework for communicating complex strategic concepts through quantifiable metrics rather than subjective assessment. For tournament broadcasts, it enhances spectator experience by providing meaningful context for momentum shifts and pivotal moments.

Drawing inspiration from elite analysts like Marc "Caedrel" Lamont, whose exceptional career trajectory from professional player to World Championship commentator to successful team owner demonstrates the value of accessible strategic insight, this project bridges the considerable gap between analytical sophistication and practical utility. His newly established team won the EMEA Masters Winter championship merely five months after its formation further validates the transformative potential of expert-level strategic guidance. As a regular watcher, Caedrel’s top laner always go 0/10 kda with so many deaths but the strategy he implemented in his team leads to a suprising strategy where no team can defeat them together. While Caedrel's co-ownership of emerging analytical platform DPM.LOL further validates the commercial potential of advanced statistical tools, this research extends beyond commercial applications to establish a methodological framework applicable to strategic decision-making across multiple competitive domains.

The true value of this work lies not merely in its technical accomplishments but in its capacity to transform how players understand and engage with League of Legends' strategic complexity. By converting intricate data patterns into comprehensible insights delivered at the moment of decision, the system fundamentally alters the learning process for competitive play. Rather than relying solely on retrospective analysis or generalized guidelines, players can now access contextually relevant, statistically grounded recommendations that directly respond to their specific match circumstances.

This research thus represents both a technical advancement in predictive modeling and a practical innovation in performance enhancement methodology. Its impact will continue to evolve as competitive gaming further professionalizes, potentially establishing new standards for how players develop strategic understanding and make in-match decisions across the broader esports landscape.

Bibliography

[1] O. Samanta, “League Of Legends Player Count & Stats 2023,” Priori Data, Aug. 23, 2023. https://prioridata.com/data/league-of-legends/ (accessed: Apr. 5, 2025).

[2] J. McIntosh, “Why is League of Legends so Popular in the Esports Scene?,” blog.ggcircuit.com. https://blog.ggcircuit.com/why-is-league-of-legends-so-popular (accessed: Apr. 10, 2025).

[3] “Champions - League of Legends,” www.leagueoflegends.com. https://www.leagueoflegends.com/en-gb/champions/ (accessed: Apr. 10, 2025).

[4] Jirka Poropudas and T. Halme, “Dean Oliver’s Four Factors Revisited,” May 22, 2023. https://www.researchgate.net/publication/370949495_Dean_Oliver

[5] K. Young, “The Best Points to Score: An Econometric Approach to Competitive League of Legends,” Curiosity: Interdisciplinary Journal of Research and Innovation, Mar. 2023, doi: https://doi.org/10.36898/001c.73175. (accessed: Apr. 8, 2025).

[6] “LoL Esports,” Lolesports.com, 2024. https://lolesports.com/en-GB/news/dev-diary-win-probability-powered-by-aws-at-worlds (accessed: Apr. 4, 2025).

[7] Broken by Concept. Using AI To Build Better Items ft. @xPetu | Broken by Concept 228 | League of Legends Podcast. (Dec 9, 2024). Accessed: March 06, 2025. [Online video]. Available: https://youtu.be/sJ3mHdbKUUo?si=JTUS4il7Vk0MXER_

[8] “Caedrel,” Leaguepedia | League of Legends Esports Wiki. https://lol.fandom.com/wiki/Caedrel

[9] “Techniques,” League of Legends Wiki, 2025. https://leagueoflegends.fandom.com/wiki/Category:Techniques (accessed May 06, 2025).

[10] J. Hamari and M. Sjöblom, “What is eSports and why do people watch it?,” Internet Research, vol. 27, no. 2, pp. 211–232, 2017, doi: https://doi.org/10.1108/intr-04-2016-0085.

[11] “@lolesports 2024. “Thank you for being part of #Worlds2024 and helping us achieve our highest total viewership since 2021 and an all-time high record for viewership outside of China!” [X]. 2024, October 27 [Accessed 2025, March 06]. Available from: [https://x.com/lolesports/status/1859295721956024521]”.

[12] “/dev: Season 2024 Map Changes,” Leagueoflegends.com, Nov. 23, 2023. https://www.leagueoflegends.com/en-gb/news/dev/dev-season-2024-map-changes (accessed March 06, 2025).

[13] A. R. Novak, K. J. Bennett, M. A. Pluss, and J. Fransen, “Performance analysis in esports: modelling performance at the 2018 League of Legends World Championship,” International Journal of Sports Science & Coaching, vol. 15, no. 5–6, pp. 809–817, Jun. 2020, doi: https://doi.org/10.1177/1747954120932853.

[14] T. D. Do, S. I. Wang, D. S. Yu, M. G. McMillian, and R. P. McMahan, “Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in

League of Legends," arXiv:2108.02799 [cs], Aug. 2021, doi: https://doi.org/10.1145/3472538.3472579.

[15] A. L. C. Silva, G. Pappa, and L. Chaimowicz, "Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks," 2018. Available: https://www.sbgames.org/sbgames2018/files/papers/ComputacaoShort/188226.pdf

[16] X. Lan, L. Duan, W. Chen, R. Qin, T. Nummenmaa, and J. Nummenmaa, "A Player Behavior Model for Predicting Win-Loss Outcome in MOBA Games," Lecture Notes in Computer Science, pp. 474–488, 2018, doi: https://doi.org/10.1007/978-3-030-05090-0_41.

[17] R. Ani, V. Harikumar, A. K. Devan, and O. S. Deepa, "Victory prediction in League of Legends using Feature Selection and Ensemble methods," IEEE Xplore, May 01, 2019. https://ieeexplore.ieee.org/document/9065758

[18] R. S. Brill, R. Yurko, and A. J. Wyner, "Exploring the difficulty of estimating win probability: a simulation study," arXiv (Cornell University), Jun. 2024, doi: https://doi.org/10.48550/arxiv.2406.16171.

[19] D.-H. Kim, C. Lee, and K.-S. Chung, "A Confidence-Calibrated MOBA Game Winner Predictor," arXiv (Cornell University), Jan. 2020, doi: https://doi.org/10.48550/arxiv.2006.15521.

[20] J. Junior and Cláudio Campelo, "League of Legends: Real-Time Result Prediction," Dec. 2023, doi: https://doi.org/10.21528/cbic2023-161.

[21] P. Jalovaara, "Win probability estimation for strategic decision-making in esports." Accessed: March 06, 2025. [Online]. Available: https://sal.aalto.fi/publications/pdf-files/theses/mas/tjal24a_public.pdf

[22] S.-K. Lee, S.-J. Hong, and S.-I. Yang, "Predicting Game Outcome in Multiplayer Online Battle Arena Games," IEEE Xplore, Oct. 01, 2020. https://ieeexplore.ieee.org/document/9289254

[23] J.-A. Hitar-Garcia, L. Moran-Fernandez, and V. Bolon-Canedo, "Machine Learning Methods for Predicting League of Legends Game Outcome," IEEE Transactions on Games, vol. 15, no. 2, pp. 1–1, 2022, doi: https://doi.org/10.1109/tg.2022.3153086.

[24] Z. Chen, Y. Sun, M. S. El-nasr, and T.-H. D. Nguyen, "Player Skill Decomposition in Multiplayer Online Battle Arenas," arXiv.org, Feb. 20, 2017. https://arxiv.org/abs/1702.06253 (accessed March 14, 2025).

[25] V. Hodge, S. Devlin, N. Sephton, F. Block, P. Cowling, and A. Drachen, "Win Prediction in Multi-Player Esports: Live Professional Match Prediction," IEEE Transactions on Games, pp. 1–1, 2020, doi: https://doi.org/10.1109/tg.2019.2948469.

[26] A. White and D. M. Romano, "Scalable Psychological Momentum Forecasting in Esports," arXiv (Cornell University), Jan. 2020, doi: https://doi.org/10.48550/arxiv.2001.11274.

[27] H. Y. Ong, S. Deolalikar, and M. Peng, “Player Behavior and Optimal Team Composition for Online Multiplayer Games,” arXiv.org, Mar. 07, 2015. https://arxiv.org/abs/1503.02230v1 (accessed Mar. 03, 2025).

[28] “OP.GG - The Best LoL Builds and Tier List. Search Riot ID and Tagline for Stats,” OP.GG, 2025. https://op.gg/ (accessed May 06, 2025).

[29] “Blitz, supercharge your gameplay,” Blitz. https://blitz.gg (accessed March 06, 2025).

[30] “U GG: The Best League of Legends Builds LoL Build Champion Probuilds LoL Runes Tier List Counters Guides,” u.gg. https://u.gg (accessed March 06, 2025).

[31] “iTero AI Coach,” iTero - League of Legends AI Coach, 2025. https://itero.gg (accessed March 06, 2025).

[32] S. Kim, D. Kim, H. Ahn, and B. Ahn, “Implementation of user playstyle coaching using video processing and statistical methods in league of legends,” Multimedia Tools and Applications, Aug. 2020, doi: https://doi.org/10.1007/s11042-020-09413-4.

[33] Y. Kou and X. Gui, “Entangled with Numbers,” Proceedings of the ACM on Human-Computer Interaction, vol. 2, no. CSCW, pp. 1–25, Nov. 2018, doi: https://doi.org/10.1145/3274362.

[34] Yaser Norouzzadeh Ravari, Pieter Spronck, Rafet Sifa, and Anders Drachen, “Predicting Victory in a Hybrid Online Competitive Game: The Case of Destiny,” Proceedings, vol. 13, no. 1, pp. 207–213, Jun. 2021, doi: https://doi.org/10.1609/aiide.v13i1.12944.

[35] P. Xenopoulos, B. G. Coelho, and C. Silva, “Optimal Team Economic Decisions in Counter-Strike,” arXiv (Cornell University), Jan. 2021, doi: https://doi.org/10.48550/arxiv.2109.12990.

[36] K. Conley and D. Perry, “How Does He Saw Me? A Recommendation Engine for Picking Heroes in Dota 2.” Accessed: March 08, 2025. [Online]. Available: https://cs229.stanford.edu/proj2013/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf.

[37] “Riot Developer Portal,” Riotgames.com, 2021. https://developer.riotgames.com/docs/lol#live-client-data-api (accessed March 09, 2025).

[38] “Riot Developer Portal,” Riotgames.com, 2021. https://developer.riotgames.com/policies/general (accessed March 09, 2025).

[39] Y. Gu, Q. Liu, K. Zhang, Z. Huang, R. Wu, and J. Tao, “NeuralAC: Learning Cooperation and Competition Effects for Match Outcome Prediction,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 5, pp. 4072–4080, May 2021, doi: https://doi.org/10.1609/aaai.v35i5.16528.

[40] D. Sacco, “Caedrel named Esports Analyst of the Year at 2021 Esports Awards, other finalists included UK talent and companies, with Odee, KalKal and more on the judging panel - Esports News UK,” Esports News UK, Nov. 21, 2021. https://esports-news.co.uk/2021/11/21/caedrel-esports-analyst-of-the-year-2021-esports-awards-winners/ (accessed May 06, 2025).

[41] “@EMEAmasters 2025. “@LosRatoneslol ARE YOUR EMEA MASTERS WINTER 2025 CHAMPIONS!” [X]. 2025, Macrh 23 [Accessed 2025, March 06]. Available from: [https://x.com/EMEAmasters/status/1903878191053197618]”.

[42] F. Bahrololloomi, F. Klonowski, S. Sauer, R. Horst, and R. Dörner, “E-Sports Player Performance Metrics for Predicting the Outcome of League of Legends Matches Considering Player Roles,” SN Computer Science, vol. 4, no. 3, Mar. 2023, doi: https://doi.org/10.1007/s42979-022-01660-6.

[43] T. Skenderson, “OE Public Match Data - Google Drive,” Google Drive, 2014. https://drive.google.com/drive/u/1/folders/1gLSw0RLjBbtaNy0dgnGQDAZOHIgCe-HH (accessed Mar. 06, 2025).

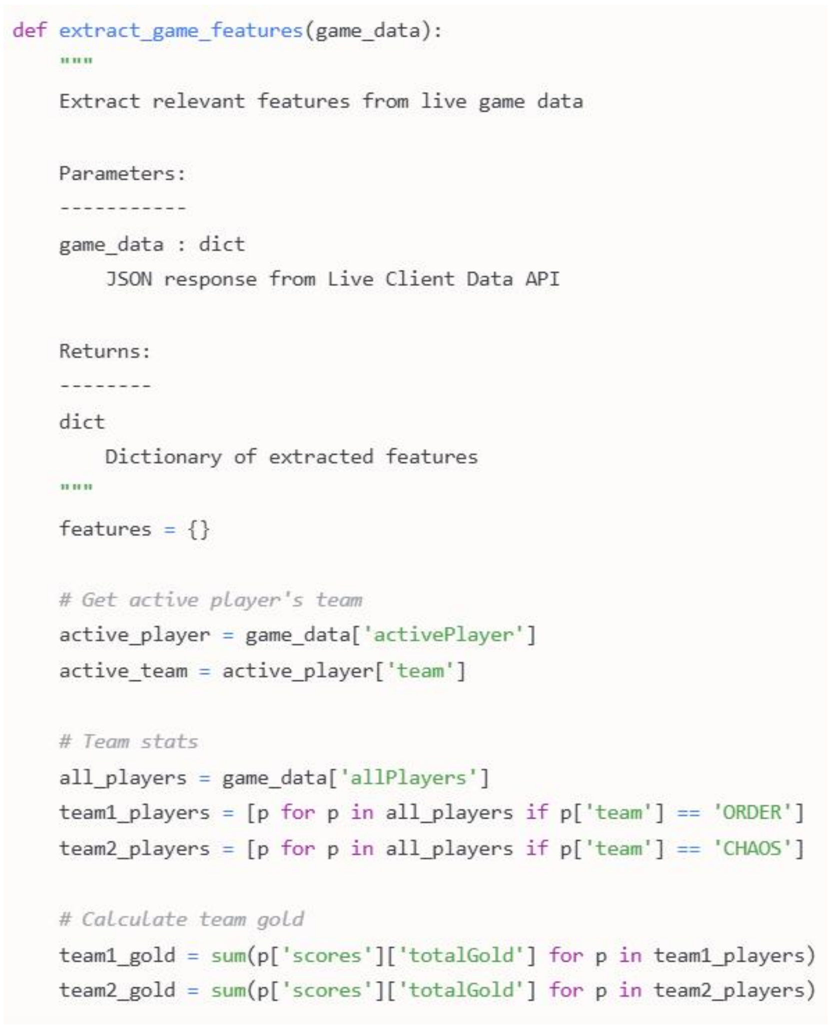
[44] R. Chinicz, “Practical Machine Learning with LoL: a Simple Predictive Use-Case with Data Collection, Learning and Explainability,” Medium, Dec. 21, 2023. https://levelup.gitconnected.com/practical-machine-learning-with-lol-a-simple-predictive-use-case-with-data-collection-learning-c2b6e621df66

[45] LTA North. Introducing The LTA - 2025 Explainer (Oct 31, 2024). Accessed: April 06, 2025. [Online video]. Available: https://youtu.be/-ZZU3I8mpEY?si=jVrTtsgE5U6Delgx

Appendices

Appendix A: Source Code Excerpts

A.1 Feature Extraction from Live Client



```python
def extract_game_features(game_data):
    """
    Extract relevant features from live game data

    Parameters:
    -----------
    game_data : dict
        JSON response from Live Client Data API

    Returns:
    --------
    dict
        Dictionary of extracted features
    """
    features = {}

    # Get active player's team
    active_player = game_data['activePlayer']
    active_team = active_player['team']

    # Team stats
    all_players = game_data['allPlayers']
    team1_players = [p for p in all_players if p['team'] == 'ORDER']
    team2_players = [p for p in all_players if p['team'] == 'CHAOS']

    # Calculate team gold
    team1_gold = sum(p['scores']['totalGold'] for p in team1_players)
    team2_gold = sum(p['scores']['totalGold'] for p in team2_players)
```

```python
# Calculate gold difference (from active player's perspective)
if active_team == 'ORDER':
    features['gold_diff'] = team1_gold - team2_gold
else:
    features['gold_diff'] = team2_gold - team1_gold

# Extract objective control
game_events = game_data['events']['Events']

# Count dragons, barons, etc.
team1_dragons = sum(1 for e in game_events if e['EventName'] == 'DragonKill' and e['Killer
team2_dragons = sum(1 for e in game_events if e['EventName'] == 'DragonKill' and e['Killer

# Similar calculations for other objectives...

# Convert to feature format
if active_team == 'ORDER':
    features['dragon_diff'] = team1_dragons - team2_dragons
else:
    features['dragon_diff'] = team2_dragons - team1_dragons

# Add other features...

return features
```

A.2 Win Probability Calculation

```python
def calculate_win_probability(features, game_time, model):
    """
    Calculate win probability based on current game state

    Parameters:
    -----------
    features : dict
        Dictionary of game features
    game_time : float
        Current game time in minutes
    model : object
        Trained machine learning model

    Returns:
    --------
    float
        Win probability (0-1)
    """
    # Determine game phase
    if game_time < 15:
        phase = 'early'
    elif game_time < 25:
        phase = 'mid'
    else:
        phase = 'late'

    # Apply phase-specific scaling factors
    time_factor = 0.7 if phase == 'early' else 1.0 if phase == 'mid' else 1.2
```

```
# Statistical weights approach
base_prob = 0.5

# Calculate weighted sum of objective impacts
weighted_sum = 0
weighted_sum += features['inhibitor_diff'] * 0.0408
weighted_sum += features['turret_diff'] * 0.0204
weighted_sum += features['baron_diff'] * 0.0147
weighted_sum += features['elder_diff'] * 0.0144
weighted_sum += features['dragon_diff'] * 0.0072
weighted_sum += features['herald_diff'] * 0.0072
weighted_sum += features['gold_diff'] / 1000 * 0.001

# Apply time factor
weighted_sum *= time_factor

# Calculate statistical probability
stat_prob = base_prob + weighted_sum

# Ensure bounds
stat_prob = max(0.01, min(0.99, stat_prob))

# Also use ML model for comparison
X = [list(features.values())]
ml_prob = model.predict_proba(X)[0][1]

# Return final probability (could be weighted average)
final_prob = 0.3 * stat_prob + 0.7 * ml_prob
return max(0.01, min(0.99, final_prob))
```

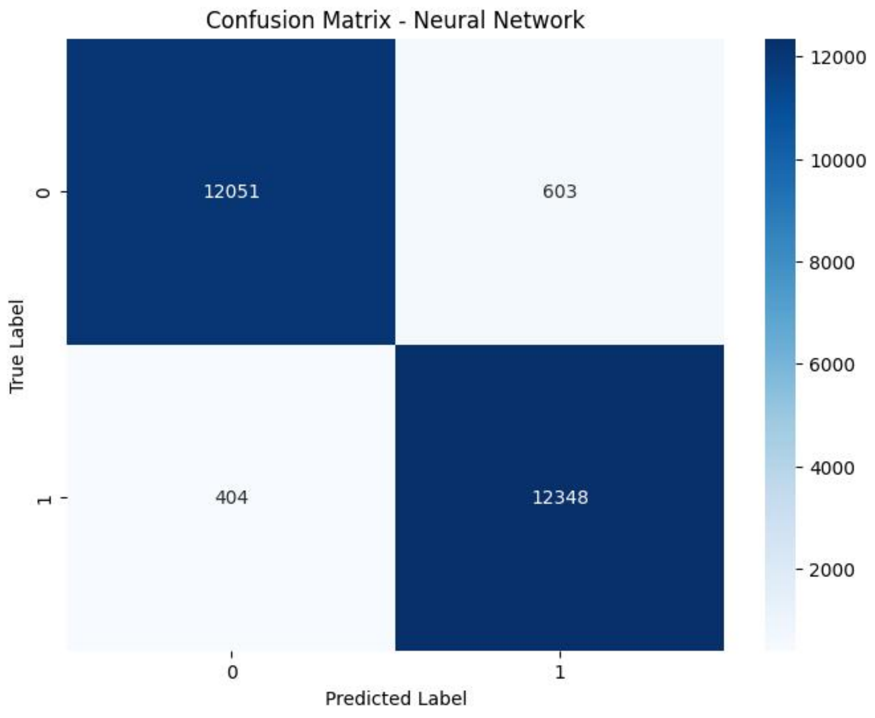
A.3 Coaching Advice Generation

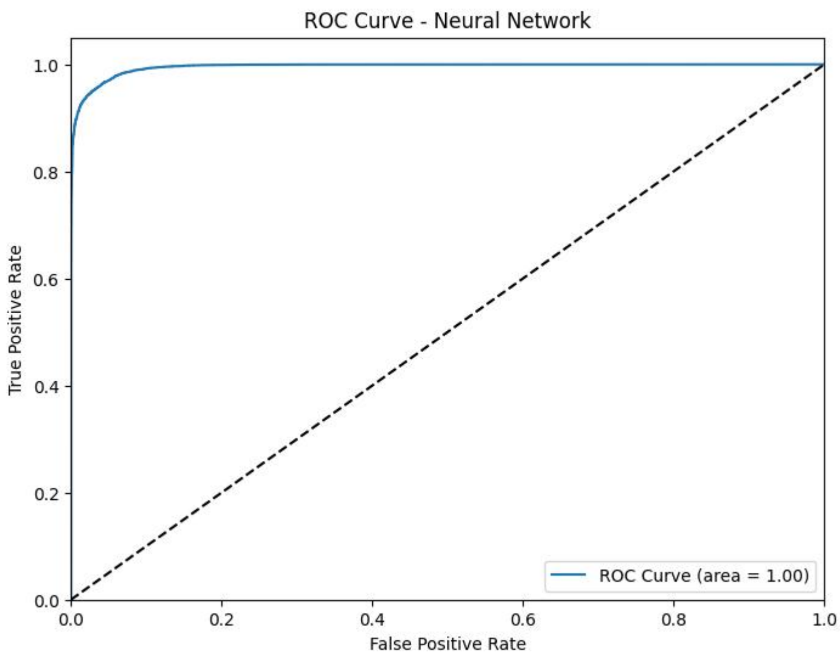
```python
def generate_coaching_advice(win_prob, features, game_time):
    """
    Generate contextual coaching advice based on game state

    Parameters:
    -----------
    win_prob : float
        Current win probability
    features : dict
        Dictionary of game features
    game_time : float
        Current game time in minutes

    Returns:
    --------
    list
        List of coaching recommendations
    """
    advice = []

    # Determine game phase
    if game_time < 15:
        phase = 'early'
    elif game_time < 25:
        phase = 'mid'
    else:
        phase = 'late'
```

Appendix B: Model Hyperparameters and Architecture

B.1 Neural Network Configuration

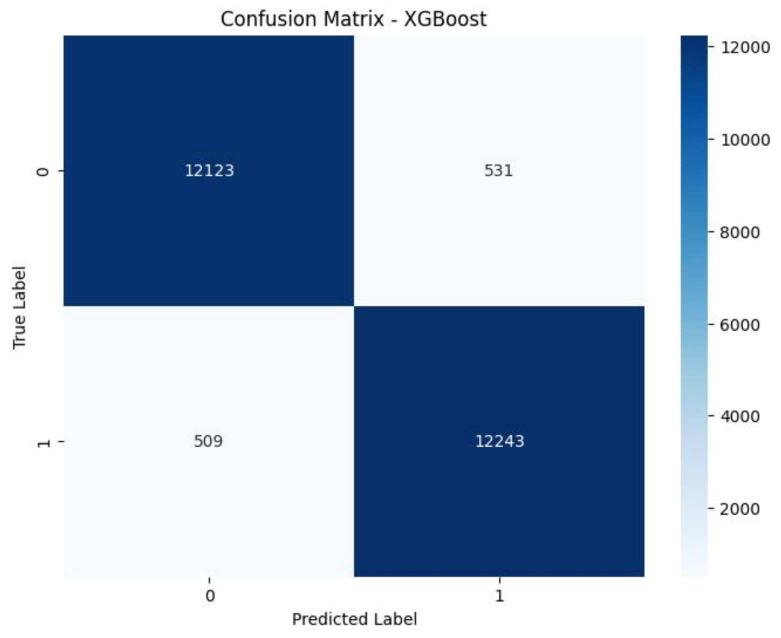
```
Model: MultiLayer Perceptron (MLP)Architecture: - Input Layer: 18 features-
Hidden Layer 1: 50 neurons, tanh activation- Hidden Layer 2: 25 neurons, tanh
activation- Output Layer: 1 neuron, sigmoid activationHyperparameters:-
Learning Rate: 0.01 (constant)- Alpha (L2 regularization): 0.01- Maximum
Iterations: 1000- Solver: 'adam'- Batch Size: 200- Early Stopping: True, with
patience=10- Validation Split: 0.2
```

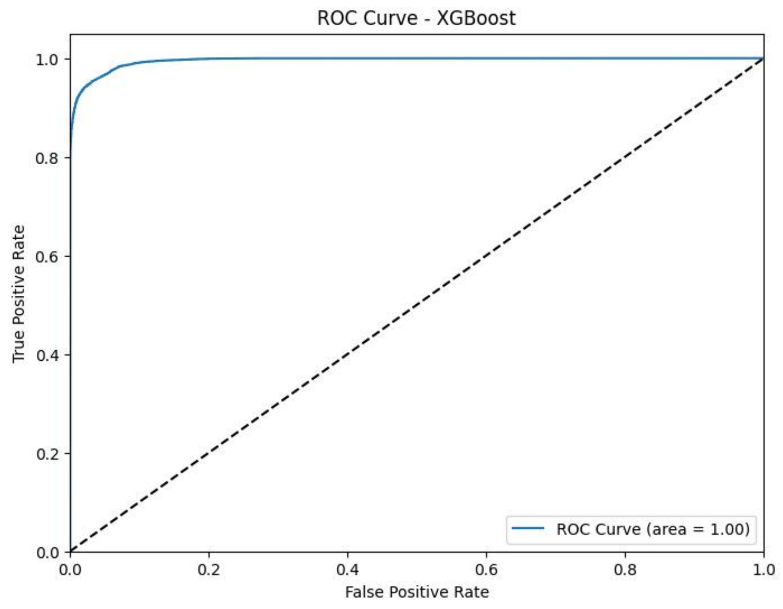




B.2 XGBoost Configuration

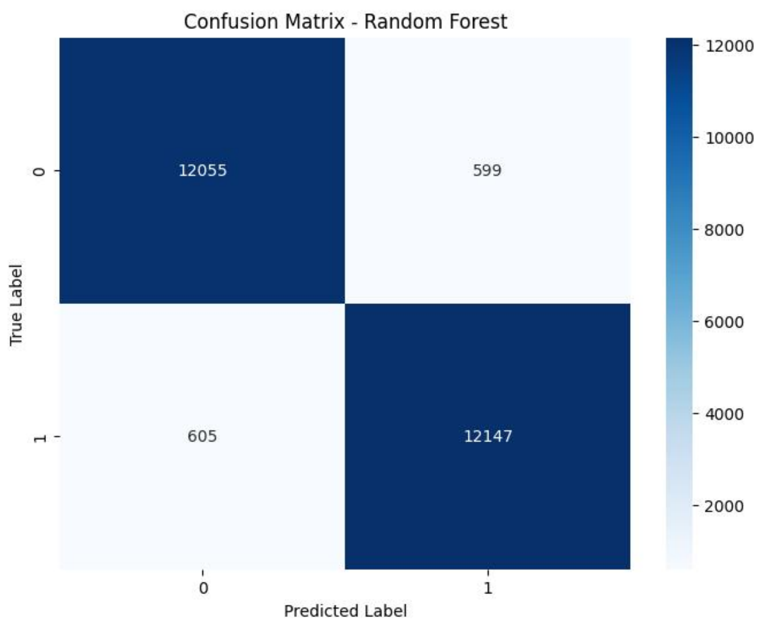
```
Model: XGBoost ClassifierHyperparameters:- max_depth: 7- learning_rate: 0.1-
n_estimators: 200- subsample: 0.8- colsample_bytree: 0.8- min_child_weight: 3-
gamma: 0.1- objective: 'binary:logistic'- eval_metric: 'logloss'- booster:
'gbtree'
```

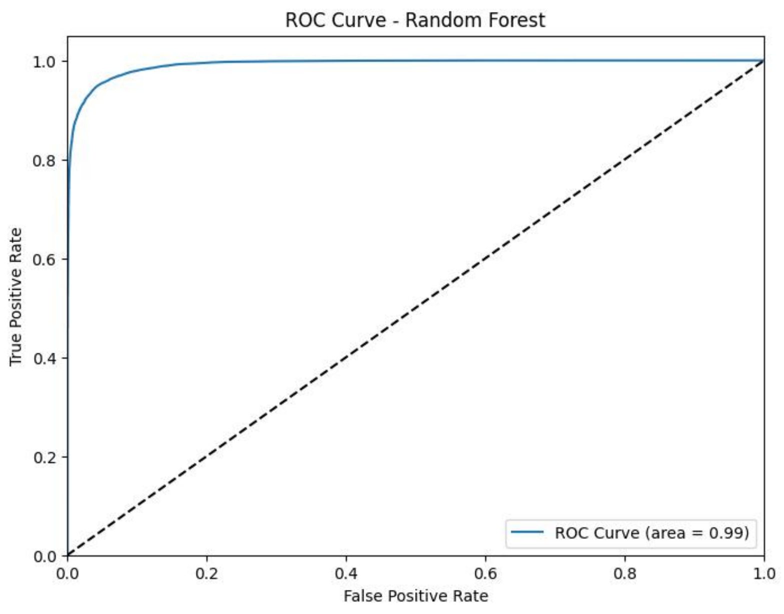




B.3 Random Forest Configuration

```
Model: Random Forest ClassifierHyperparameters:- n_estimators: 150- max_depth:
12- min_samples_split: 5- min_samples_leaf: 2- bootstrap: True- criterion:
'gini'- max_features: 'sqrt'- random_state: 42
```

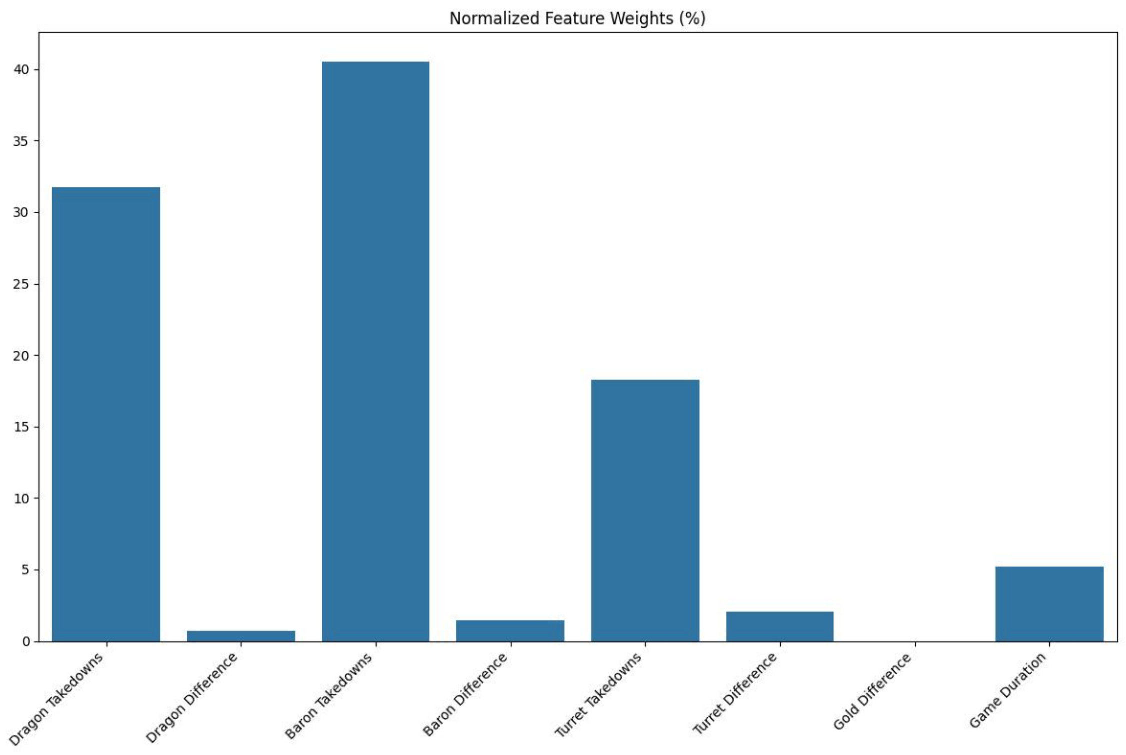

Confusion Matrix - Random Forest
12055
599
605
12147
True Label
Predicted Label
0
1
12000
10000
8000
6000
4000
2000

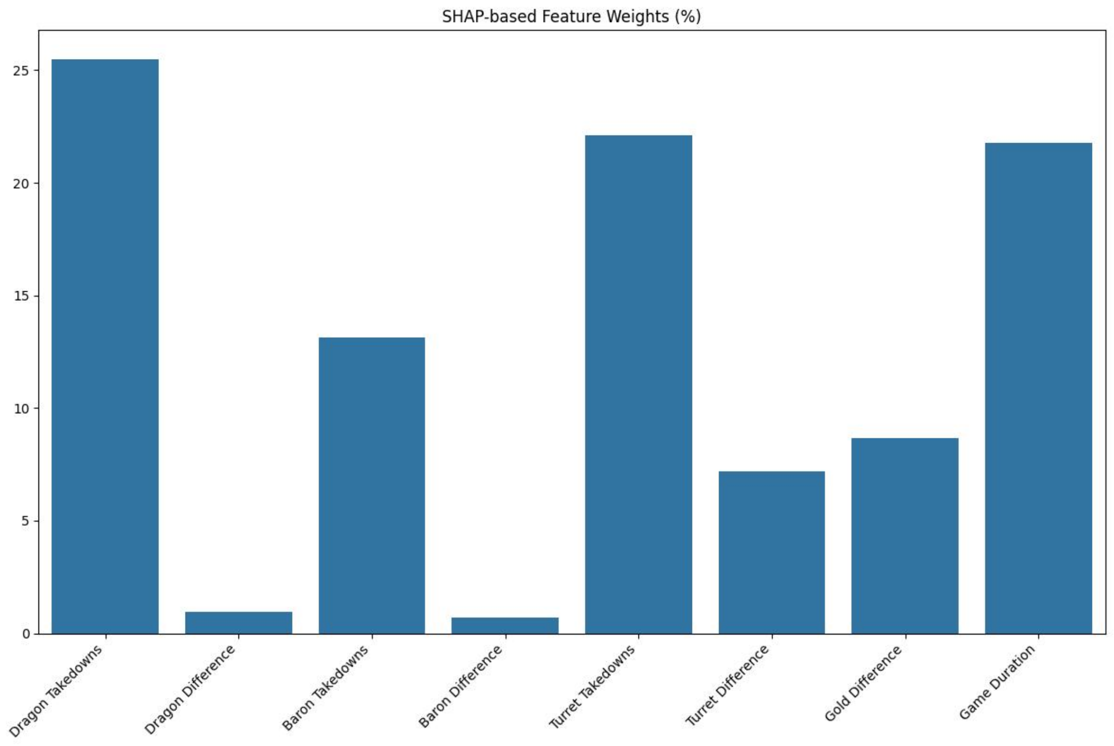

ROC Curve - Random Forest
ROC Curve (area = 0.99)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Appendix C: Objective Impact Analysis and Screenshots

Generate a formula for win probability based on calculated weights

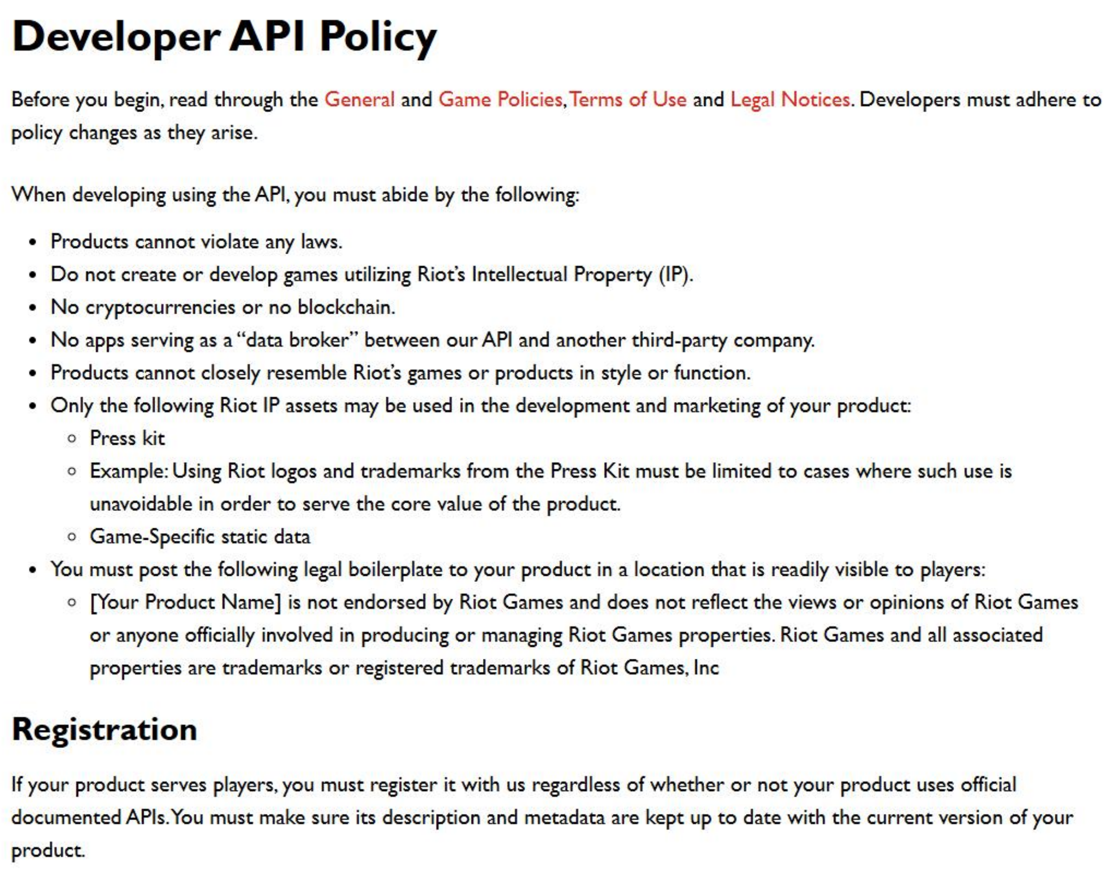
```
Loaded 101623 matches.
Training logistic regression model...
Model accuracy: 0.8636
ROC AUC: 0.9295

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.87      0.86     10107
           1       0.87      0.86      0.86     10218

    accuracy                           0.86     20325
   macro avg       0.86      0.86      0.86     20325
weighted avg       0.86      0.86      0.86     20325


Calculating permutation importance...

Calculating SHAP values...

Analyzing game time influence...

Final weights for win probability formula:
dragon_weight: 0.0072
baron_weight: 0.0147
turret_weight: 0.0204
gold_per_1k_weight: 0.0000
elder_dragon_weight: 0.0144
inhibitor_weight: 0.0408
```


Normalized Feature Weights (%)
0
5
10
15
20
25
30
35
40
Dragon Takedowns
Dragon Difference
Baron Takedowns
Baron Difference
Turret Takedowns
Turret Difference
Gold Difference
Game Duration


SHAP-based Feature Weights (%)
0
5
10
15
20
25
Dragon Takedowns
Dragon Difference
Baron Takedowns
Baron Difference
Turret Takedowns
Turret Difference
Gold Difference
Game Duration

Appendix D: Ethical Considerations and Compliance

D.1 Riot Games API Terms of Service Compliance



Developer API Policy

Before you begin, read through the General and Game Policies, Terms of Use and Legal Notices. Developers must adhere to policy changes as they arise.

When developing using the API, you must abide by the following:

- Products cannot violate any laws.
- Do not create or develop games utilizing Riot's Intellectual Property (IP).
- No cryptocurrencies or no blockchain.
- No apps serving as a "data broker" between our API and another third-party company.
- Products cannot closely resemble Riot's games or products in style or function.
- Only the following Riot IP assets may be used in the development and marketing of your product:
  - Press kit
  - Example: Using Riot logos and trademarks from the Press Kit must be limited to cases where such use is unavoidable in order to serve the core value of the product.
  - Game-Specific static data
- You must post the following legal boilerplate to your product in a location that is readily visible to players:
  - [Your Product Name] is not endorsed by Riot Games and does not reflect the views or opinions of Riot Games or anyone officially involved in producing or managing Riot Games properties. Riot Games and all associated properties are trademarks or registered trademarks of Riot Games, Inc

Registration

If your product serves players, you must register it with us regardless of whether or not your product uses official documented APIs. You must make sure its description and metadata are kept up to date with the current version of your product.

Examples of Approved Use Cases for Personal Keys

Personal keys are meant for smaller-size, personal projects.

- Personal sites.
- School projects.
- Creating a proof of concept for a Production Key request
- Examples of Approved Use Cases for Production Keys.
- Showing (self) player stats
- Running tournaments.
- Training tools that allow players to view their own match histories and aggregate stats.
- Looking For Game (LFG) tools.
- Game overlays that provide static data that is available prior to the game.
- Aggregate player stats (no specific players).
- Official Ladder Leaderboards.

Examples of Unapproved Use Cases

The following use cases will not be approved:

- Products cannot display win rates for Augments or Arena Mode items. This applies to all websites, applications and overlays.
- Products may not provide any game-session-specific information that would be previously unknown to the player.
- Apps that dictate player decisions.
- Apps that violate the general game policies.
- Products may not publicly display a player's match history from the custom match queue unless the player opts in to share this specifically for League of Legends. Otherwise, a player's custom match data may only be made available to them using RSO.

D.2 Data Privacy Considerations

Security

You must adhere to the following security policies:

- Do not share your Riot Games account information with anyone.
- Do not use a Production API key to run multiple projects. You may only have one product per key.
- Use SSL/HTTPS when accessing the APIs so your API key is kept safe.
- Your API key may not be included in your code, especially if you plan on distributing a binary.
  - This key should only be shared with your teammates. If you need to share an API key for your product with teammates, make sure your product is owned by a group in the Developer portal. Add additional accounts to that account as needed.

Game Integrity

- Products must not use or incorporate information not present in the game client that would give players a competitive edge (e.g., automatically or manually allowing tracking enemy ultimate cooldowns), especially when such data is not already accessible through regular gameplay.
- Products cannot alter the goal of the game (i.e. Destroy the Nexus).
- Products cannot create an unfair advantage for players, like a cheating program or giving some players an advantage that others would not otherwise have.
- Products should increase, and not decrease the diversity of game decisions (builds, compositions, characters, decks).
- Products should not remove game decisions, but may highlight decisions that are important and give multiple choices to help players make good decisions.
- Products cannot create alternatives for official skill ranking systems such as the ranked ladder. Prohibited alternatives include MMR or ELO calculators.
- Products cannot identify or analyze players who are deliberately hidden by the game.