UNIVERSITY OF CAPE TOWN

MINOR DISSERTATION

Esports Betting Technology: Machine Learning for Match Prediction and Odds Estimation

Author:
Henri Izak David DU PLESSIS

Supervisor:
Neil WATSON

A minor dissertation submitted in partial fulfilment of the requirements for the degree of Master of Philosophy specialising in Financial Technology

in the

School of Economics

June 18, 2024

Plagiarism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the Vancouver referencing system for citation and referencing. Each contribution to, and quotation in, this minor dissertation *"Esports Betting Technology: Machine Learning for Match Prediction and Odds Estimation"*, from the work(s) of other people has been attributed, and has been cited and referenced.
3. Where I have consulted the published work of others, this is always clearly attributed.
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this minor dissertation is entirely my own work.
5. I have acknowledged all main sources of help.
6. Where the minor dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Signed by candidate

UNIVERSITY OF CAPE TOWN

Abstract

Faculty of Commerce

School of Economics

Master of Philosophy specialising in Financial Technology

Esports Betting Technology: Machine Learning for Match Prediction and Odds Estimation

by Henri Izak David DU PLESSIS

Esports betting is a rapidly growing segment of the online sports betting market. A key feature of this industry is the pricing of betting odds. This study investigated the online sports betting industry, odds compilation, and how machine learning can be used for sports prediction. The techniques used in the literature were then applied to one of the world's foremost esports: Counter-Strike. A substantial dataset of professional match data (n=11271) was collected and used to construct 142 relevant features for match prediction. Several supervised learning models, including random forests, feed-forward neural networks, and XGBoost, were trained to estimate win probabilities for both teams in each match. Betting odds were then calculated using these probabilities and compared to real-world betting odds.

A notable aspect of the research is the implementation of Microsoft's TrueSkill rating system. It served as both a benchmark and an input feature. Among the models tested, XGBoost showed the best overall performance. The highest match prediction accuracy attained was 62.7%. It was found that incorporating a large number of statistics did not significantly improve predictive accuracy when compared to models using fewer, more important features. It was also found that LAN matches and best-of-3 map formats are more predictable than their counterparts. Despite the inherent difficulty in Counter-Strike match prediction, the models could generate efficient odds which exhibited strong correlation with real-world odds (up to 85%). A betting strategy informed by the generated odds was back-tested over a six-month period and shown to be profitable.

This research therefore demonstrates how machine learning models can be used for esports match prediction, with practical applications in the online betting industry.

Acknowledgements

I would like to thank my supervisor, Neil WATSON, for his consistent guidance, feedback, and support over the past year.

I also want to thank my parents, HENRY and JANET, for their love and support throughout my life and academic journey.

Knowing that you believed in me was a powerful source of motivation.

Contents

List of Figures

List of Tables

List of Abbreviations

ADR	average damage per round
AI	artificial intelligence
AUC	area under ROC curve
API	application programming interface
BO*x*	best-of-x
CAGR	compound annual growth rate
CPL	Cyberathlete Professional League
CS	Counter-Strike
CS:GO	Counter-Strike: Global Offensive
CS2	Counter-Strike 2
CT	Counter-Terrorist
ESL	Electronic Sports League
FPS	first-person shooter
HP	health points
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
IEM	Intel Extreme Masters
IGL	in-game leader
IP	intellectual property
KAST	kill-assist-survived-traded
***k*-NN**	k-nearest neighbours
LAN	local area network
LR	logistic regression
MAE	mean absolute error
ML	machine learning
MLP	multilayer perceptron
MLB	Major League Baseball
MOBA	multiplayer online battle arena
NBA	National Basketball Association
NFL	National Football League
NHL	National Hockey League
NFTs	non-fungible tokens
NN	neural network
PC	personal computer
PGL	Professional Gamers League
RD	ratings deviation
RMSE	root mean square error
ROC	receiver operating characteristic
RTS	real-time strategy
T	Terrorist
US	United States

1 Introduction

1.1 Background and context

The rapid technological development of the 21st century has ushered in a digital transformation of most industries [1]. The sports betting industry is no different. Placing a bet on the outcome of a sports match is now only a few quick taps away, as droves of online sports betting companies compete to take your wager. Electronic banking facilitates near instantaneous deposits and withdrawals between the user and the bookmaker, and the slew of odds and other betting options are generated 24/7 by computers running statistical models and complex algorithms. All of these advancements have made betting quicker, easier, and more accessible to an ever-expanding market of bettors [2].

In a similar vein, the viability of *electronic sports* relies on the uninterrupted functioning of an ensemble of technologies: complex computer hardware and peripherals providing the physical interfaces for the gameplay, the games themselves being the top layer of a highly-advanced software stack, and the high-speed, low-latency networking infrastructure that connects each player to one another in a virtual arena [3]. With the sudden rise in popularity of this new form of competition, bookmakers have been quick to capitalise on a new market and offer betting opportunities for the outcomes of esports matches [4].

1.1.1 Online sports betting

Sports betting is a form of gambling whereby the bettor places a monetary wager on the outcome of one or more sporting events [5]. If the outcome(s) materialise, the bet is won and the bettor receives the value of their wager multiplied by the odds offered by the bookmaker. The odds for a given event are calculated by odds compilers, who analyse historical data and employ statistical techniques to accurately predict the probabilities of different outcomes [6]. The mathematical reciprocal of these probabilities determines the odds. To increase their profitability, bookmakers then decrease the odds slightly for either outcome. This difference between the compiled and offered odds is known as the bookmaker's margin [7].

It is not known precisely when the practice of sports betting began, however sources claim that the Greeks would bet on the Ancient Olympic Games. The practice later appeared in Rome, where chariot racing and gladiator battles became the subject of the bets [8]. Betting on horse races was a popular activity in the United Kingdom by the early 18th century, and soon spread throughout the developed world [9]. By 1890, the United States boasted over 300 legal horse racing tracks, with horse race betting becoming a favourable pastime of the upper class. In the US, the industry was heavily regulated, and outright banned in 1910 [10]. Regulations evolved over time, and the first government-sanctioned, brick-and-mortar sportsbook opened in Las Vegas in 1949. These businesses were known as 'Turf Clubs', and accepted bets

on a number of professional sports, including the National Football League (NFL), National Hockey League (NHL), Major League Baseball (MLB), and National Basketball Association (NBA) [11].

The era of online sports betting began in 1996, when an Austrian-based sports book used the internet to legally connect sports bettors and bookmakers. On January 17th, 1996, the first online wager was made when Jukka Honkavaara bet $50 on a football match: the first online sportsbook, Intertops (now known as Everygame), offered 1.04 odds for Tottenham Hotspurs to beat Hereford United. Though Honkavaara's winnings only amounted to a modest $2, this event was nevertheless significant, as it marked the birth of the online sports betting industry [12].

Although regulations have hampered the industry in the US significantly, by 2023 the global market value of sports betting has grown significantly to 243 billion US dollars [2]. It currently accounts for over 40% of global gambling revenue around the world - more than casinos, lotteries, and other forms of gambling [4]. Much of this growth can be attributed to the digitisation of the industry, which has led to increased access to and availability of betting opportunities. In 2018, the federal ban on sports betting in the US was lifted, and today it has been legalized in most states [13]. Furthermore, the industry is expected to grow significantly in Asia and Africa as laws surrounding sports betting become more relaxed and access to the online betting platforms improves [2]. The modern features and mechanics of the online betting industry will be further explored in the next chapter.

1.1.2 Electronic sports

The fastest-growing sports market, in terms of betting volume, are *electronic sports*, also known as esports [4]. Esports are video games played in a competitive setting [14]. The practice started in the 1970s and 80s on the floors of public arcades. The best players would rise to the top of persistent high-score leaderboards in their favourite titles like *Space Invaders*, *Pac-Man*, and *Tetris* [15]. This type of competition was indirect, but by the 1990s, multi-player fighting games like *Street Fighter* would see players pitted directly against each other. By then, the transition from arcades floors to home consoles and personal computers (PCs) was well under way, and the Local Area Network (LAN) party became the new battle arena. *Doom* and *Quake* marked the rise of first-person shooters (FPS) in this era. At the dawn of the 2000s, real-time strategy (RTS) games like *StarCraft* and *Warcraft* spawned the first professional leagues, legitimizing competitive gaming [16].

High-speed internet became ubiquitous by the 2010s, and esports went mainstream. Multiplayer online battle arena (MOBA) games like *League of Legends* and *Dota 2* rose to prominence, and now boast tournaments with multi-million dollar prize pools. The largest esports prize pool to date was *The International 2021* in Bucharest, Romania, with $40 million US being awarded to the competitors [17]. Although most are team games, some esports entail single players competing against one or more other players. In *Fortnite*, only one player can emerge victorious in a 100-player 'battle royale', where the last surviving player wins. Kyle Giersdorf won at the Fortnite World Cup Finals 2019 at the age of 16, winning $3 million US in the process.

Today there are a substantial variety of game genres that are played professionally across the globe. Viewership figures for the major tournaments already rival traditional sports, and esports continue to grow and evolve without showing any signs of slowing down [17][3]. Esports revenues are expected to reach $1.8 billion US in

2025, with a Compound Annual Growth Rate (CAGR) of 13.4% between 2020 and 2025 [17].

With a new form of competition, came a new form of gambling: esports betting. The industry initially developed in regulatory grey areas, such as the infamous CSGOLounge.com [18], which accepted wagers of hitherto unregulated virtual currencies and items, such as "skins" - tradable, in-game cosmetic items [19]. Skins and other virtual items have many of the same properties as non-fungible tokens (NFTs), often having unique identifiers, rarities, or characteristics which make them desirable in addition to their in-game utility. A significant market exists for these virtual items, with extremely rare skins being sold for thousands of dollars [18].

The industry has been formalized over time, and it is now common to see esports alongside traditional sports on legal betting sites. Controversies still remain, however, as bookmakers frequently sponsor teams and tournaments, promoting online betting to a growing, and often young, population of spectators [20]. Furthermore, the prevalence of sports betting can undermine the integrity of competition. A hallmark case of match-fixing occurred in 2015 [21], where a top American team, iBUYPOWER, purposefully lost a match they were heavily favoured to win. A number of players were later found to have received kickbacks in the form of valuable skins. This prompted Valve, the game developers, to take action to preserve the integrity of the game. All the players that were accused of "throwing" the match were banned from competing in Valve-sanctioned tournaments, effectively ending their careers as professional Counter-Strike players [22].

Counter-Strike (CS) currently dominates the esport betting market, with 56% of all wagers in this segment being placed on CS matches [23]. There are also a number of online sites which track and collect data relating to these matches, such as HTLV [24] and Liquipedia [25]. It is therefore an attractive esport to consider for this research.

1.1.3 Overview of Counter-Strike

The first version of Counter-Strike emerged in 1999 as a community-developed 'mod' (modification) for *Half-Life*, the hit FPS developed by Valve Corporation. With its unique emphasis on team-work and co-ordination, CS quickly garnered attention from players. The following year, Valve hired the developers of the mod, acquired the intellectual property (IP), and subsequently launched CS as a stand-alone title [26]. It did not take long for a competitive scene to emerge. The first tournaments were organized in Europe in 2001 by the Cyberathlete Professional League (CPL), with prize pools of ten to twenty thousand euros [27]. Since then, the game has soared in popularity over a number of iterations, and the quality and scale of competition has grown along with it. *Counter-Strike 2* was released on the 27th September 2023 and played by over 30 million players in the first month [28].

Over the last decade, CS has cemented itself as one of the major esports in the world. There are several competitive tournaments hosted by organizations like the Electronic Sports League (ESL), DreamHack, and Intel Extreme Masters (IEM). Leagues typically culminate in major in-person LAN events where the best teams compete for their share of prize pools which regularly exceed a million dollars. Valve-sponsored Major tournaments are the pinnacle of the CS calendar and attract millions of concurrent viewers on streaming platforms in addition to a packed stadium of fans. The PGL Major event, pictured below, had a prize pool of $2 million US and reached a peak viewership of 2.7 million spectators across multiple online platforms. [29][30].

At present, the sum total of all CS tournament prize pools to date is over $187 million US [25].

FIGURE 1.1: The 2021 PGL Major took place in the Avicii Arena in Stockholm, Sweden [30]

Gameplay fundamentals

Counter-Strike is a PC video game played using the mouse, keyboard, and headset. Players assume the role of a soldier in a virtual 3D environment, which can run, walk, and jump by pressing keys on the keyboard. The mouse is typically used for two purposes: to control the direction and movement of the in-game field-of-vision, and for shooting, where cross-hairs at the centre of the player's screen act as the focal point for aiming. For effective communication, players co-ordinate their actions with their team-mates primarily through a headset with a built-in microphone.

Although a number of versions have been released since its inception, the core game mode played in Counter-Strike, known as *defusal*, has remained relatively unchanged. Two teams of five players each start each round on opposite ends of a virtual arena called a *map*. One team plays the role of the Counter-Terrorists (CT), whose objective is to defend two bomb sites, A and B, from the opposing team, the Terrorists (T). The layout for the popular Counter-Strike map, Mirage, is shown in Figure 1.2; the red zones highlight the bomb sites and the green zones are the spawn areas for either team. There are seven official maps played in competitive Counter-Strike.

A map is played for best-of-24 rounds, with teams swapping sides after 12 rounds. This means that in order to win, a team must demonstrate superiority over their opponents in both attacking (as Ts) and defending (as CTs). The first team to win 13 round wins the map. In the event of a tie (12-12), the map continues in best-of-6 round *overtime* phases, with sides swapping every 3 rounds, until a winner is decided.

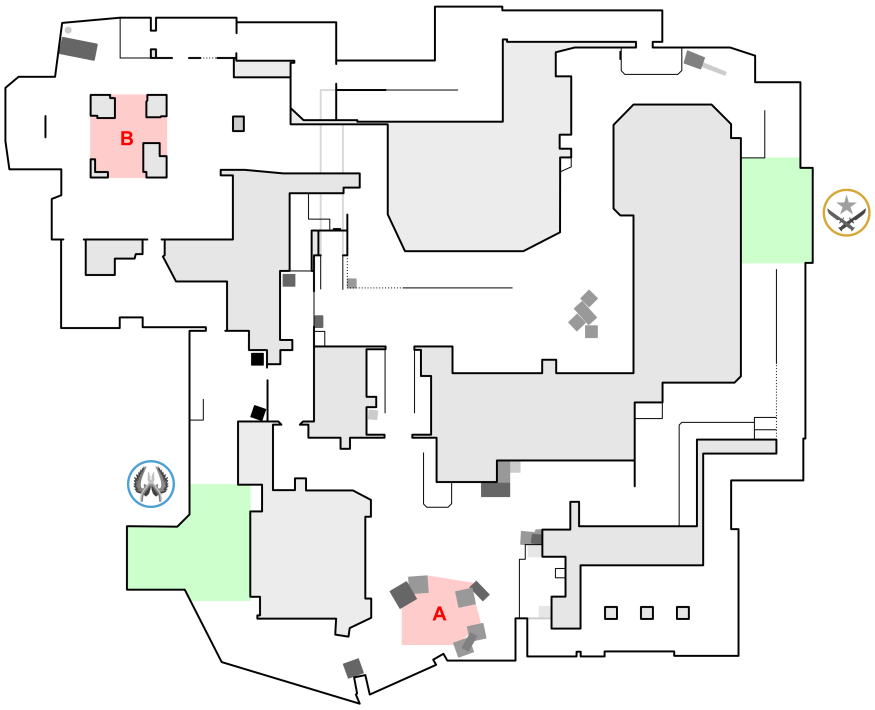


FIGURE 1.2: Map layout for 'Mirage', showing spawn areas in green and bomb sites in red [31]

Each round, spanning 1 minute and 55 seconds, is a high-stakes scenario as each player has only one life. Eliminated players have to spectate their living team-mates until the next round. Rounds can be won in four ways: by (1) eliminating the entire enemy team, or, if playing as Terrorists, by (2) detonating a bomb on one of the bomb sites. Once a bomb is planted, the round timer changes to a 40 second countdown until detonation. The Counter-Terrorists can win rounds either by (3) successfully defending both sites for the duration of the round, or by (4) retaking the bomb site and defusing the bomb after it has been planted. Players begin each round with 100 health points (HP). Damage is measured in HP and can be inflicted using a variety of weapons and grenades. There is no way to replenish HP during a round.

Strategy and team economy in Counter-Strike

Different strategies (*meta*'s) evolve for attacking and defending the various points on each map, with players learning which positions are optimal to hold with different weapons, how, when, and where to throw their *utility* (smoke, flash-bang, high-explosive, and incendiary grenades), and how to move around the map as the round unfolds. All positions on a given map are named such that players can immediately communicate critical gameplay information to their teammates over the course of a round. Success in Counter-Strike relies on efficient team-work as much as mechanical skill, as player actions need to be co-ordinated in order to optimally assist one's teammates. This can be in terms of positioning (e.g. setting up a cross-fire at a choke point[1], or moving to help bolster the defence at a bomb site which is being attacked) or in terms of utility usage (e.g. throwing an incendiary grenade at

[1] A choke point is a small part of the map near each bomb site which, if controlled, gives either team a tactical advantage.

a choke point in order to slow down an attack, or throwing a flash bang to allow a teammate to take control of a position).

Players are rewarded with in-game money for getting a kill, planting or defusing the bomb, and winning or losing a round. Winning a round is rewarded with far more money than losing a round, however this amount (the *loss bonus*) grows with each consecutive round loss. Money is spent to purchase weapons and equipment at the start of each round. If a player survives the round, they retain whatever equipment they had into the following round. Each team therefore has a monetary balance, known as the team economy, which fluctuates throughout the duration of a map. A stronger economy means you can purchase stronger weapons and more equipment. A weaker economy might relegate your team to just buying a few pistols and practically forfeiting a round in order to save funds for the following round. Sound economic strategy is thus critical to success in Counter-Strike.

At the start of a half, both sides have almost no money and so duel with only pistols. The *pistol round* is quite crucial, as it initiates economic momentum for whichever team wins it. In some exciting cases, teams will defy odds and trade the initial rounds back-and-forth, leaving the economic balance undecided. A game can become extremely tense if both teams have precarious economies near the end, as the outcome is undecided and the stakes get progressively higher.

The state of each team's economy is a good predictor of their ability to win a given round. This is the total team balance as well as their current loss bonus; teams earn progressively more money with consecutive round losses up to a cap of $3400 after 5 losses in a row - matching the round-win bonus. A team with the funds to 'full-buy' will almost always win a round against a team that is saving their money in an *eco* round where they don't purchase anything other than pistols. A *semi-buy* is a middle-ground, used to save enough funds to ensure a *full-buy* the following round, but still giving the team a fighting chance. These distinctions therefore refer to the strength of the weapons and amount of utility grenades that can be purchased in each round.

An example of the economy for two teams over the course of a map is shown in Figure 1.3. The lines corresponds to either team's balance, and a coloured icon represents a round won by the CTs (blue) or the Ts (orange) respectively. The horizontal lines are thresholds that indicate the health of the economy, indicated as eco, semi-buy, and full-buy. Note that the pistol rounds take place in rounds 1 and 13, and teams switch sides in round 13 which results in the round-win icon colour switching.

Professional Counter-Strike

The local area network (LAN) is the most competitive environment possible, as all players have the same, virtually instant network latency and there is no possibility for gameplay 'lag'. They play under the same conditions and use identical computer hardware (except for their keyboards and mice). Tournaments played "on LAN" represent the pinnacle of competitive Counter-Strike, with teams flying all over the world to attend these events. Teams qualify for LAN events either through invitation (due to their position in the world ranking), or by placing sufficiently high enough in qualification rounds that are normally played online.

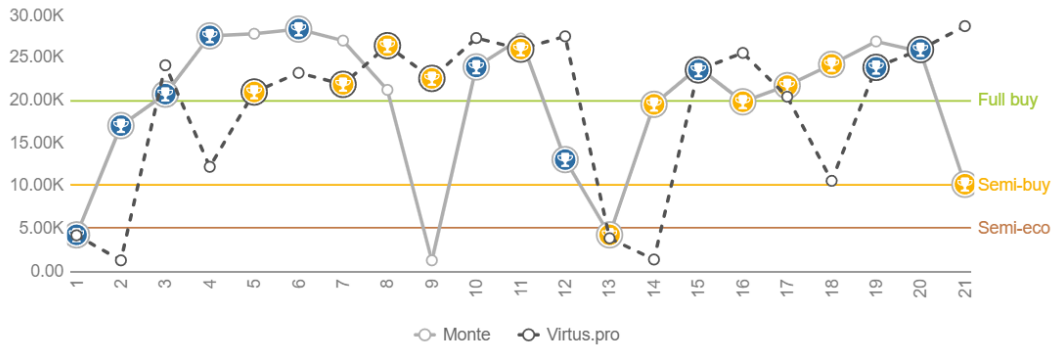


FIGURE 1.3: A graph showing the economies of two teams over the course of a map [32]

Professional CS matches are played in a best-of-1 (BO1) or best-of-3 (BO3) maps format. The grand final for bigger tournaments are occasionally contested in a best-of-5 (BO5) map series. Match lengths can vary greatly depending on the format and skill-disparity between competitors. A BO1 can be concluded in 20 minutes if one team dominates the other. In extreme cases, a BO5 grand final between two evenly matched giants can become a five-hour affair as tightly-contested maps get traded back-and-forth. The most common format is the BO3, where the match winner is determined over 2 or 3 maps which last roughly 40 minutes each.

Although CS is a methodical and tactical team-based shooter, it is highly dynamic in the short, medium, and long term. During individual rounds, players must make split-second decisions to react to events as they unfold. Over the course of a map of 24 rounds, teams need to learn and adapt to their opponent's habits in order to exploit their weaknesses. Between matches, teams will often adopt and improve strategies from other teams by reviewing their gameplay recordings, or come up with entirely new approaches to counter a certain prevailing strategy.

Teams have coaches whose role is to identify these trends mid-match and suggest appropriate countermeasures. The better-resourced teams employ analysts who review their opponent's games and develop strategies in advance. Competitive CS is mentally demanding, so teams often have sports psychologists to keep the players operating at peak performance [33]. Throughout the match, coaches must ensure their players maintain composure. As discussed earlier, the longer format matches can become a battle of mental endurance. This has led to some controversy surrounding esports players and the use of stimulants like Adderall [34].

Data in Counter-Strike

Every second of CS game generates a wealth of data; in previous versions of the game, the 'tick rate' is the frequency at which each player's game client is synchronized with the server. Professional matches were played exclusively on '128-tick', which means there were 128 snapshots of the game being generated each second, containing every minute detail of the game: each player's precise location on the map, the direction at which they are aiming, and the state they are currently in - moving, jumping, shooting, throwing a grenade, reloading, switching weapons, planting or defusing a bomb, shooting or being shot at. At any given point there can be a number of smoke grenades deployed on a map, a high-explosive grenade being

lobbed towards an enemy position, or an incendiary grenade temporarily blocking a choke point.

With the introduction of CS2's sub-tick system, the granularity of data has been further enhanced, as player movements and interactions can now be recorded in-between ticks using an interpolation system. It should be evident that there is a tremendous amount of data generated in each map played. This gameplay data is stored in a *demo file*, which can be replayed at a later stage or processed for data analysis.

A number of statistics can be generated by processing these demo files. For example, player performance can be measured by the number of kills, deaths, assists, average damage per round (ADR), and weapon accuracy. Free online tools like Leetify rate all players from the data in their match demo-files. They report statistics such as *cross-hair placement precision*, which refers to the degree deviation of the cross-hairs from an enemy who comes into view, *time-to-damage*, which is the number of milliseconds between seeing an enemy and inflicting damage, and *headshot accuracy*, which is the proportion of kills attained by lethal headshots. *Clutch* statistics refer to the percentage of rounds won where a player is last alive on his team versus 1, 2, 3, 4, or 5 opponents. All of these data points help to paint a descriptive picture of a player's contribution to their team's success or defeat. That said, some roles, such as *support* or *in-game leader* (IGL) help their team in ways that are harder to quantify.

Team performance is usually measured by their win-rates on the various maps. Teams have preferred maps, and most also have *perma-bans* which they never play. In a BO3 series, teams will decide which of the 7 maps to play through a veto process; this always follows a ban-ban-pick-pick-ban-ban-decider sequence. There are therefore trends in which maps a team will play and will avoid. A map-veto which favours one team in an otherwise equal match-up can therefore make for a favourable bet.

All these statistics are prominent features of a professional broadcast; for example, commentators will refer to the weapon accuracies when contrasting the snipers of opposing teams. Dominant teams can go on significant match or map-win streaks, which adds further tension to tight matches. On certain broadcasts, there is a round-win prediction percentage which sways depending on the round economy and number and health of players alive on each side.

1.1.4 Machine learning

Machine learning (ML) is an interdisciplinary field, spanning computer science, statistics, and data science. It refers to the development and study of statistical algorithms that can learn from data and generalize to unseen data [35]. It is fitting that the term was coined in 1959 by Arthur Samuel, who, while working at IBM, invented a program to calculate the chances of either side winning in the game of checkers [36]. In the last two decades, improvements in computational power has allowed the field to develop from a theoretical concept into a practical technology with widespread commercial use. Today, machine learning methods are employed across a plethora of industries, including health care, manufacturing, education, financial modelling, and more [37].

ML is considered a subset of artificial intelligence (AI), as many of the algorithms imitate the way that humans learn [38]. ML techniques are used for the development

of software that perform computer vision, speech recognition, natural language processing, and much more. It is normally far easier to train a system using large amounts of data than it is to explicitly program it to anticipate a desired response for all possible sets of inputs. This makes ML an effective tool for many "big data" applications [37].

There are three broad categories of machine learning paradigms, which differ primarily on the nature of the feedback available to the learning system. These are (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning. In *supervised learning*, the system learns from data with known outputs. In other words, the goal is to learn how to map a series of inputs to a series of outputs. It is commonly found in applications like image recognition. In *unsupervised learning*, there are no known output labels and the goal is to learn hidden patterns or structures in the data. An example of this is customer segmentation in marketing. Finally, in *reinforcement learning*, an agent tries to achieve a goal in a dynamic environment known as the problem space. As it navigates the problem space, it is rewarded for good responses and punished for bad ones. The system attempts to optimize the proportion of good responses. This ML paradigm applies to areas such as robotics and game playing.

For this research, the objective is to predict the probability of either team winning given a large dataset of information about two teams. The problem can be expressed as predicting whether Team A will beat Team B, with a binary answer of no, they lose (0), or yes, they win (1). Given that the input data is derived from a large dataset of match records with known outcomes, supervised learning techniques are pertinent. In the subsequent chapter algorithms which have shown promise in sports analytics will be explored.

1.2 Research problem

The techniques used by bookmakers to compile betting odds are not well-represented in the literature. Furthermore, esports betting is a high-growth segment of the market, and professional Counter-Strike matches drive a significant amount of online betting activity. Counter-Strike is a complex esport, with several factors making it difficult to predict. While existing literature has explored machine learning in traditional sports, its application in esports, particularly in Counter-Strike match prediction, remains underexplored. Fortunately, an abundance of data is generated during these matches by virtue of their digital nature, making them an ideal subject for investigation. This study aims to explore whether machine learning techniques can be used to make accurate predictions, and by extension, betting odds estimations, for professional Counter-Strike matches.

1.3 Research objective

The primary objective of this study is to develop a model that can estimate the odds of either team winning for any given match of professional Counter-Strike. This objective can be broken down into a number of smaller objectives:

1. Review the literature on the technology used in the sports betting industry, and on the application of machine learning techniques for sports prediction

2. Gather a significant dataset of historical data for professional Counter-Strike matches
3. Identify and generate relevant features from the historical match data
4. Train a selection of supervised learning models to generate probability estimates and betting odds for either team winning
5. Measure and compare the performance of each model using classification accuracy and F1-score, and identify the most relevant features for match prediction
6. Compare the betting odds generated by the models to actual historic betting odds

1.4 Scope and limitations

Although this minor dissertation is concerned with the esports betting industry, it is not feasible to create a model that can predict the outcome for every esport. This is because there are a large variety of esports with different competition rules, gameplay factors, and players. It was therefore decided to focus on one specific esport: Counter-Strike.

In Counter-Strike, there are many betting opportunities beyond just the match winner. As discussed earlier, matches are played in different formats consisting of one or more maps of gameplay. Bettors are often given the opportunity to place bets on which team will win the current map, which team will win the pistol round for a given map, or which team will simply get the first kill of a map. Although map-specific data is used for feature selection, the objective of this research is to quantify probabilities for winning the entire match. The most common match format is Best-of-3 maps, which will be considered the default for this project.

Furthermore, in recent years, live or in-play sports betting has become popular; bettors can make bets while a match is being played, and often cash-out their existing bets for some discounted return. Bookmakers offer dynamic odds which rely on live match analysis. While some research has been done in the field of real-time match prediction, this project will focus purely on predicting the overall match outcome using only historical information, i.e. before the match has begun.

Finally, bookmakers can to some extent monitor market dynamics as they know which bets are being made. They can therefore factor in the market expectations by adjusting the odds offered on different outcomes using the betting information available to them. This data is privy to the bookmakers and will thus not be included in the modelling process.

1.5 Ethics approval

This project was approved by the Commerce Research Ethics Committee with code COM/00541/2023. The approval letter has been included in Appendix B.

1.6 Structure

This dissertation is comprised of five chapters. The introduction covers the background knowledge relevant to the topic, defines the research problem, and establishes the objectives. Thereafter, existing research on the esports betting industry and data-driven match prediction is covered in the literature review.

The third chapter precisely describes the methodology employed to achieve the research objects: how the data is collected, stored, processed, analysed, and used to generate the features. The models selected and the techniques used to train them are all described in this chapter. This chapter is concluded with a description of the script which takes a match input and generates the betting odds.

The penultimate chapter is a discussion of the results obtained in the modelling process, and how well the objectives were met. Finally, a conclusion summarizes the key findings, limitations, and suggestions for areas of future research.

2 Literature Review

This chapter builds upon the topics covered in the background by reviewing relevant academic literature. The development and features of the online sports betting industry are first described, followed by an investigation into how machine learning is used for sports prediction. The specific machine learning techniques used in the literature are researched to better understand how they can be used for Counter-Strike match prediction. Additionally, various rating systems used in sports and esports are identified and discussed. The chapter concludes with a discussion of predictive analysis in esports.

2.1 The evolution of online sports betting

Technological advances have caused major shifts in gambling practices over the last two decades. The modern sports betting industry utilizes software engineering, data analysis, and real-time processing to satisfy a rapidly growing market of bettors. Bets are overwhelmingly made through the use of a smartphone app or website, where algorithms are used to set and dynamically update betting odds [39]. Prior to this revolution, bets would need to be made either in-person or with an operator over the telephone. Bookmakers would employ teams of statisticians, known as odds compilers, whose job was to spectate sports events or horse races, break them down into fundamental inputs, and calculate probabilities from those inputs. Most statistical odds compiling was done by counting the occurrences of a given event. The development of in-play betting caused a shift towards mathematical modelling for odds compilation, as humans could simply not keep up with multiple simultaneous events unfolding: "*Bookmakers needed automation, which meant models*" [6].

Betfair were one of the pioneers in the industry, leveraging technology to offer a superior value proposition to bettors in the form of a peer-to-peer betting exchange. The platform was first developed in 1998 by programmer and professional gambler Andrew Black, whose original idea was to create a betting system which operated like the US stock exchange. By late 2004, Betfair had over 300 000 customers and saw over GBP 50 million revenue per week. A 2005 study on Betfair's competitive advantage cited Moore's Law, Metcalfe's Law of Networks, and the internet as key enablers of its success. [40].

A betting exchange works by offering two types of bets: 'back' and 'lay', where a 'back' is a conventional bet on an outcome to occur, and a 'lay' is betting *against* the outcome happening. Odds are set by the bettors themselves, with a bet only going through if another participant bet on the 'lay'. In this way, no odds compilation is necessary as odds are driven purely by market forces. The exchange operator generates revenue from a much smaller commissions when compared to traditional bookmakers [41].

Another sports betting platform called EmpireBet was the focus of a recent study into sports betting through the lens of software engineering. The authors noted that the biggest drivers of evolution were advances in connectivity, the transition from trained operators to user-friendly interfaces accessible by anyone, the automation of risk management and administration tasks, integration with cloud computing services, and the adoption of third-party libraries for rapid development [42].

In-play sports betting features, also known as live betting, have been a driver of growth in the industry. This is because bettors can place multiple bets during a single sports event. Many bookmakers now also offer a 'cash-out' feature, which was first introduced to the market by William Hill in 2012 [39]. This feature allows bettors to "take profit early" if the outcome you bet on has become more likely, or to gain a fraction of your wager back if it is becoming more unlikely.

As discussed in Section 1.1.1, another driver of growth in the sports betting industry is the development of esports betting [19]. Business Research Insights reported the market size as $ 9.75 billion US in 2021, and projected it to grow to $ 35.57 billion US by 2023: a CAGR of 13.7% [43].

2.2 Machine learning for sports prediction

Over the past two decades, there has been a significant growth in the amount of data collected during sports matches, both structured and unstructured. It is no surprise that machine learning algorithms have become increasingly popular for analysis of these increasingly large and complex datasets [44].

Statistical prediction of sports outcomes, such as the match winner or final score, is of great interest to spectators, teams, and other industry stakeholders. In 2020, Horvat and Job analysed 38 research papers aimed at this particular application for various team sports, ranging from basketball, football, cricket, baseball, and American football [44]. They found that feature selection was critical for building effective models. Noting that earlier studies relied solely on expert opinions, newer studies could achieve better results using statistical techniques (such as regularization) for feature selection.

The machine learning models employed by researchers differed depending on the nature of the data and the complexity of the problem. In most of the studies, multiple supervised learning classification models were trained and compared. These included logistic regression, support vector machines, decision trees & random forests, Naive-Bayes, and gradient boosting. Neural networks (NN) were the most popular technique used for prediction, as they can effectively model non-linear relationships by capturing complex patterns in the data [45].

The models were all trained using either a chronological segmentation of the dataset, or by using k-fold cross validation. The most common way of measuring performance was using classification accuracy. The researchers found significant variance in the performance of the models depending on the sport, the quality of the data, and the feature selection process used. Figure 2.1 shows the distribution of maximum classification accuracy achieved in the 38 different papers [44].

Do the same findings apply to esports analysis? Jenny et al. [46] compared esports to traditional sports and argued that, other than the degree of physicality involved, esports display the same characteristics of traditional sports: esports are a voluntary

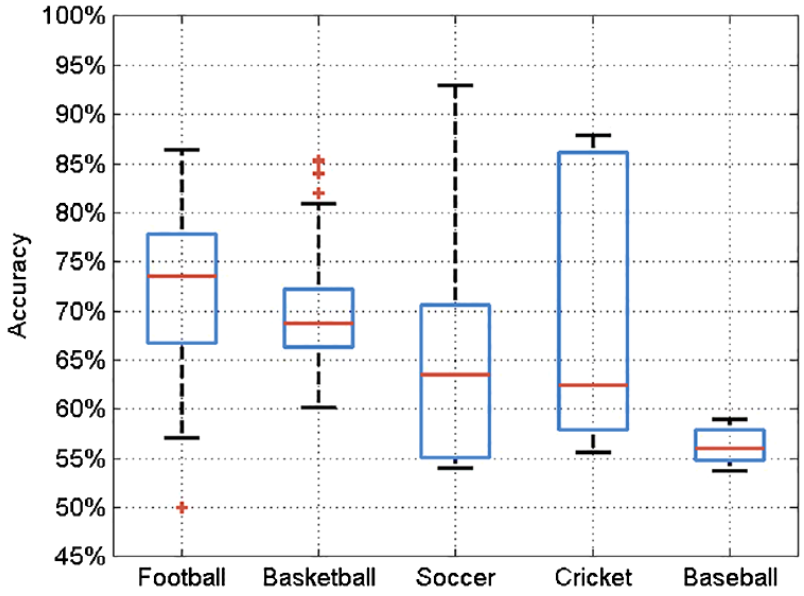


FIGURE 2.1: A boxplot of maximum classification accuracies achieved for different sports [44]

and intrinsically motivated activity, governed by rules, in which opposing players compete. Furthermore, they require skills such as strategic thinking, hand-eye coordination, and rapid decision-making. They concede that the degree of physicality differs significantly, as esports involve fine motor skills as opposed to the high levels of physical exertion typically required in traditional sports. Their research also argued that esports have a global following and are becoming increasingly institutionalized, further cementing their legitimacy. It is therefore reasonable to assume that the techniques described in [44] should, to some extent, be applicable to esports as well.

2.2.1 Machine Learning Algorithms

In this section the techniques used in the aforementioned papers are investigated. The algorithms include logistic regression, random forests, support vector machines, k-nearest neighbours, Naive Bayes, neural networks, and gradient boosting.

Logistic regression

Logistic regression (LR) [47] is a binary classification algorithm: given an input vector $\boldsymbol{x_i}$, classify the output label y_i as 0 or 1. The output value is determined by mapping a linear combination of input values onto the logistic function. This produces a probability $p(\boldsymbol{x_i})$ of a given input vector $\boldsymbol{x_i}$ (set of features) belonging to the default output class, $y_i = 1$.

$$p(\boldsymbol{x_i}) = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots}} \tag{2.1}$$

The optimal set of regression coefficients, $\beta_0, \beta_1, \ldots, \beta_n$, is obtained by maximizing the likelihood function. In LR, the likelihood function is the product of the probabilities p_i of each data point being correctly classified by the model.

$$\mathcal{L}(\beta_0, \beta_1, ..., \beta_n | y, X) = \prod_{i=1}^{N} p(y_i | \boldsymbol{x_i}, \beta_0, \beta_1, ..., \beta_n)^{y_i} \left(1 - p(y_i | \boldsymbol{x_i}, \beta_0, \beta_1, ..., \beta_n)\right)^{1-y_i} \tag{2.2}$$

where $\mathcal{L}$ is the likelihood function, y_i is the actual label of the i-th sample, $\boldsymbol{x_i}$ is the feature vector of the i-th sample, $\beta_0, \beta_1, ..., \beta_n$ are the regression coefficients, and $p(y_i | \boldsymbol{x_i}, \beta_0, \beta_1, ..., \beta_n)$ is the predicted probability of the i-th sample belonging to the positive class given the feature vector and coefficients.

Because maximizing the likelihood function is computationally challenging, the negative log-likelihood (also known as cross-entropy loss) function is minimized instead. For logistic regression, the loss function is defined as:

$$L(y, p) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right] \tag{2.3}$$

where $L(y, p)$ is the loss function, N is the number of samples, y_i is the actual label, and p_i is the predicted probability that the i-th sample belongs to the positive class.

This is an optimization process which is performed iteratively. The logistic regression model therefore fits a logistic curve (also known as a sigmoid) to the data, such that the set of regression coefficients used minimizes the cross-entropy loss function. LR is most appropriate when the class distribution is balanced and a linear relationship exists between the input variables and the output class.

Decision trees and random forests

A decision tree resembles a flowchart-structure with three elements: internal nodes which represent different split points on features, branches which represent decision rules, and leaf nodes which represent the outcome. Starting at the top-most node (root), the decision tree is recursively partitioned into progressively smaller subtrees.

There are multiple methods for determining the splitting criterion at each node, such as Information gain (the difference in entropy before and after a split) and Gini impurity. The latter quantifies the 'purity' of a node, with lower values representing more homogenous nodes. The Gini Impurity for a single node is calculated using the following formula, where p_i is the probability of an observation being classified into the i^{th} class.

$$Gini\ Impurity = 1 - \sum (p_i)^2 \tag{2.4}$$

As an objective function for determining splits, the Gini Impurity is calculated across all leaf nodes. The split that results in the greatest reduction in weighted average impurity across the child nodes is then performed.

The depth of a decision tree affects its performance and complexity. More complex relationships can be modelled by deeper trees, but these are also more prone to overfitting.

Overfitting can be mitigated using several techniques. Decision trees can be pruned back according to a cross-validated cost complexity criterion. Their maximum depth can be limited, which prevents them from becoming too complex. The minimum number of samples required to split an internal node can be increased, ensuring that there is enough data to make a decision.

Random forests [48] is an ensemble model; it combines the outputs of many, simpler decision trees which are each trained on a different random subset of the dataset. This technique is known as bagging and it helps to prevent models from over-fitting the training data. Random forests are typically more robust and have an improved accuracy when compared to a single decision tree.

Support vector machines

A support vector machine (SVM) [49] is classification model which aims to separate data points into their output classes by means of a *hyperplane*. The *margin* is the distance between the hyperplane and the nearest data point in each class. A SVM aims to maximize the margin as far as possible. If a linear hyperplane cannot separate the data well enough, the data can transformed into a higher-dimensional space using a *kernel* function.

A hyperplane in an n-dimensional space can be defined as:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.5}$$

where $\mathbf{w}$ is the normal vector to the hyperplane, $\mathbf{x}$ is a data point, and b is the bias term.

The margin, which SVMs aim to maximise, is the distance between the hyperplane and the nearest data point from either class. The points lying on the margin boundaries are called support vectors. The distance from a point $\mathbf{x}_i$ to the hyperplane is given by:

$$\frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|} \tag{2.6}$$

For a binary classification problem with labels $y_i \in \{-1, 1\}$, the optimisation problem to maximise the margin can be formulated as:

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \end{aligned} \tag{2.7}$$

As mentioned before, when the data is not linearly separable, SVMs use a *kernel* function to map the input features into a higher-dimensional space where a linear hyperplane can be used to separate the data. Commonly used kernel functions include a *linear* kernel, a *polynomial* kernel, or a *radial basis function* (RBF) kernel. A graphical representation of the effect of using a kernel function is shown in Figure 2.2.

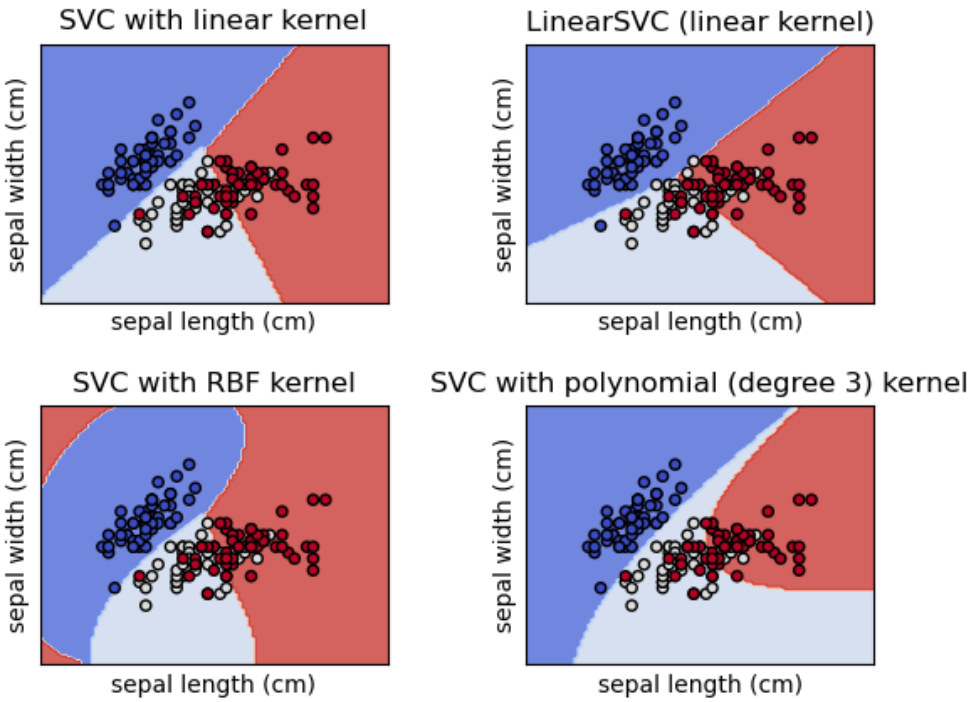


FIGURE 2.2: Hyperplanes for a SVM using different kernel functions [50]

When using a kernel function, the optimization problem becomes:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} \quad y_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad \xi_i \geq 0, \quad \forall i \end{aligned} \tag{2.8}$$

where $\phi(\mathbf{x})$ is the mapping to the higher-dimensional space, ξ_i are slack variables allowing for misclassification, and C is a regularisation parameter balancing margin maximization and classification error.

k-nearest neighbours

k-nearest neighbours (*k*-NN) [51] is a simple algorithm that classifies new observations into the majority class among its nearest neighbouring observations in the feature space. Given an input vector $\mathbf{x}_i$, the class label y_i is predicted by the most frequent class among the k closest neighbours in the training set.

For a new data point $\mathbf{x}_i$, the predicted class $\hat{y}_i$ is determined by:

$$\hat{y}_i = \arg\max_{y} \sum_{\mathbf{x}_j \in N_k(\mathbf{x}_i)} I(y_j = y) \tag{2.9}$$

where $N_k(\mathbf{x}_i)$ is the set of k nearest neighbors of $\mathbf{x}_i$, and $I(\cdot)$ is the indicator function.

Using a smaller k value makes the model more sensitive to noise, while a larger k value may smooth over useful patterns.

The distance between data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is usually measured using the Euclidean distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_{jm})^2} \tag{2.10}$$

where M is the number of features. Note that other measures of distance, such as Manhattan or Minkowski, can also be used.

k-NN differs from other models in that there is no training phase, however it can be computationally expensive during the testing phase and is often sensitive to irrelevant features.

Naive Bayes

Naive Bayes [52] is a probabilistic classifier based on Bayes' theorem. Bayes' theorem quantifies the probability of an event occurring given prior knowledge of the conditions related to the event.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.11}$$

where $P(A|B)$ is the conditional probability of event A occurring given that event B is true, and $P(A)$ and $P(B)$ are the independent probabilities of event A or B occurring.

It is "naive" as it assumes that the predictors (features) are conditionally independent given the class, i.e. the presence (or absence) of a particular feature in a class is unrelated to the presence (or absence) of any other feature in that class.

The Gaussian Naive Bayes classifier is a Naive Bayes algorithm which assumes that all continuous features are normally distributed. The probability of a feature value given a class is thus calculated using the Gaussian probability density function.

For a new data point $\mathbf{x}_i$, the predicted class $\hat{y}_i$ is determined by:

$$\hat{y}_i = \arg\max_y p(y) \prod_{j=1}^{n} p(x_{ij}|y), \tag{2.12}$$

where $p(y)$ is the prior probability of class y, and $p(x_{ij}|y)$ is the conditional probability of feature j given class y, assumed to be Gaussian:

$$p(x_{ij}|y) = \frac{1}{\sqrt{2\pi\sigma_{yj}^2}} \exp\left(-\frac{(x_{ij} - \mu_{yj})^2}{2\sigma_{yj}^2}\right), \tag{2.13}$$

where μ_{yj} and σ_{yj}^2 are the mean and variance of feature j in class y, respectively.

Neural networks

A neural network (NN) [53] consists of many densely interconnected nodes. In a feed-forward network, data moves in one direction only - forward. It traverses the network from the input layer, through multiple hidden layers of nodes, to an output layer. Figure 2.3 is an example of this structure.

Each node assigns a weight to its incoming connections, and emits an output using an activation function. This function computes an output from each weighted input and a threshold value. During training, the weights and thresholds are adjusted until convergence [45]. At convergence, these values stabilize and do not change significantly with further training.

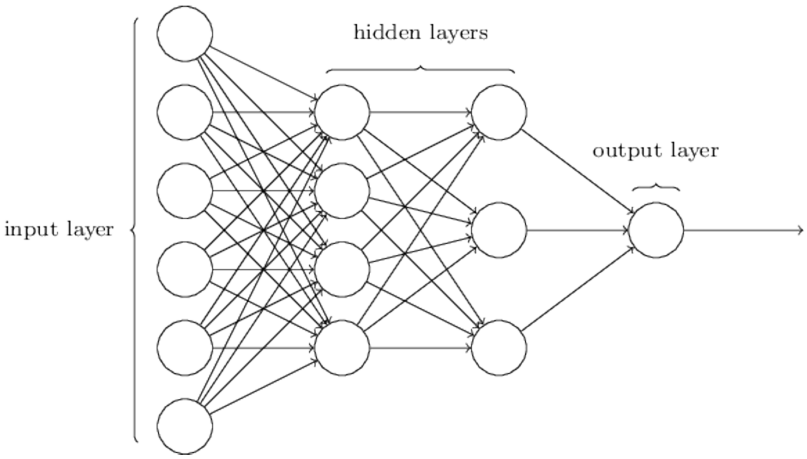


FIGURE 2.3: A simple feed-forward neural network [54]

They key parameters which affect the network's capacity to model complex relationships in the data are the depth of the network (the number of hidden layers), the width (the number of nodes in each hidden layer), and the activation function used in the hidden layers. Common activation functions include

- Rectified Linear Unit (ReLU)

$$f(x) = \max(0, x) \tag{2.14}$$

  where $f(x)$ is the activation function applied to the input x of a neuron.

- Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

  Sigmoid squashes the output to the range [0, 1], which is useful for binary classification tasks.

- Tanh (Hyperbolic tangent)

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.16}$$

  Tanh squashes the output to the range [-1, 1], which helps with zero-centered data and mitigates the vanishing gradient problem.

The activation function used impacts the training and performance of the NN. ReLU is commonly used in hidden layers due to its simplicity and effectiveness in training deep networks, while sigmoid and tanh are used in the output layer for binary classification.

There are multiple types of neural networks with different applications. Feed-forward neural networks, such as multilayer perceptrons (MLPs), work as described above.

MLPs consist of at least three layers of nodes and can be used for both classification and regression tasks.

Recurrent neural networks (RNNs) are designed for sequential data processing. They have connections that form directed cycles, thereby maintaining a hidden state that can capture information from previous time steps. The hidden state is updated iteratively using the current input and the previous hidden state. This feedback loop makes RNNs suitable for tasks involving time series data and natural language processing.

Convolutional neural networks (CNNs) are designed to process data with a grid-like topology, such as images. They consist of one or more convolutional layers, where each layer applies convolutional filters to small regions of the input, capturing local patterns and spatial hierarchies. This makes CNNs suitable for tasks like image recognition [55].

Gradient boosting

Gradient boosting [56] is an ensemble technique that differs from random forests in that each tree is built sequentially, where each new tree is trained to 'correct' the errors of the previously trained trees.

The process begins with an initial model, typically a simple predictor, such as the mean of the target values for regression problems. With each iteration, the pseudo-residuals (the difference between the predicted and actual values) are calculated and the subsequent tree is built to predict these residuals. The model is updated by adding the predictions from this tree, weighted by a learning rate λ, to the initial predictions. This iterative fine-tuning process is repeated until convergence or another stopping criterion is met.

This process is mathematically formulated as follows:

The **initial model** $F_0(x)$ is chosen to minimize the loss function over all training examples.

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{2.17}$$

where $L(y, \hat{y})$ is the loss function, y_i is the actual value, and γ is a constant (e.g., the mean of y for regression).

For each **iteration**, $m = 1$ to M (the total number of iterations):

(a) Compute the pseudo-residuals:

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\right] \tag{2.18}$$

These residuals represent the gradient of the loss function with respect to the current model's predictions.

(b) Fit a base learner (e.g., a decision tree) $h_m(x)$ to the pseudo-residuals:

$$h_m(x) \approx r_i^{(m)} \tag{2.19}$$

(c) The optimal step size γ_m is computed to minimize the loss function after adding the new learner's predictions:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \tag{2.20}$$

(d) Update the model by adding the scaled predictions of the new learner to the current model

$$F_m(x) = F_{m-1}(x) + \lambda \gamma_m h_m(x) \tag{2.21}$$

where λ is the learning rate.

After M iterations, the **final model** is given by:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \lambda \gamma_m h_m(x) \tag{2.22}$$

XGBoost (eXtreme Gradient Boosting) [57] is an advanced, scalable implementation of gradient boosting which regularly outperforms other models for classification tasks [58]. It introduces a number of improvements over gradient boosting:

XGBoost uses a second-order Taylor approximation of the loss function L to make the optimization problem more accurate. Regularisation is introduced into the objective function which penalises model complexity, thereby reducing overfitting. It handles sparse and missing data efficiently using a sparsity-aware split finding algorithm. Furthermore, tree construction is parallelised for enhanced computational efficiency.

2.2.2 Rating systems

Ratings and ranks serve as indicators of the relative performance of players or teams. These indicators are determined using different methods that vary from sport to sport, however they are always modelled on prior data. The ratings then serve as predictors for future data, and there is an implicit expectation that higher-rated teams are expected to beat lower-rated teams more often than not. A good rating system is one which represents the true skill level as accurately as possible. This section investigates the Elo, Glicko, and TrueSkill rating systems.

Elo

The Elo rating system was developed by Hungarian-American physics professor Arpad Elo [59]. Its original purpose was to rank the strength of different chess players, however it can be applied to any zero-sum game. The Elo system has been used to rate teams and players in many other sports, including football, baseball, and esports. The central idea is that the difference between two players' Elo ratings serves as a predictor of the match outcome. Each player's Elo rating is a single value, R. After each game, the winner gains points and the loser loses points. The magnitude of the loss and gain is different for each player, depending on their original Elo ratings. A mathematical description of the Elo rating system follows.

Let players A and B have initial ratings $R_A = 900$ and $R_B = 1000$. The expected score for player A and B is given by the logistic function with base 10:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} = 0.360$$

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}} = 0.640$$

As shown above, the expected outcomes are complements of each other, adding up to 1. After the match is played, the player ratings are then updated:

$$R_A^{'} = R_A + K \cdot (S_A - E_A)$$

where S_A is the actual score for player A and K is the maximum possible adjustment per game. Adjusting the K-factor will adjust the sensitivity of the system to change. Assuming that player A won the game, $S_A = 1$, and $K = 50$,

$$R_A^{'} = 800 + 50 \cdot (1 - 0.360) \approx 932$$

$$R_B^{'} = 1000 + 50 \cdot (0 - 0.640) \approx 968$$

Player A's Elo rating has increased, and Player B's has decreased. If the players were to play another game, the expected outcome has now changed to reflect their new ratings:

$$E_A^{'} = 0.448$$

$$E_B^{'} = 0.552$$

In this way, all players can be rated on the same scale and the probability of winning is easily calculated. In the graph in Figure 2.4, the expected score can be interpreted as the probability of the first player winning. For example, as the difference in Elo rating approaches 800, the probability of the player with the higher Elo rating winning approaches 100%.

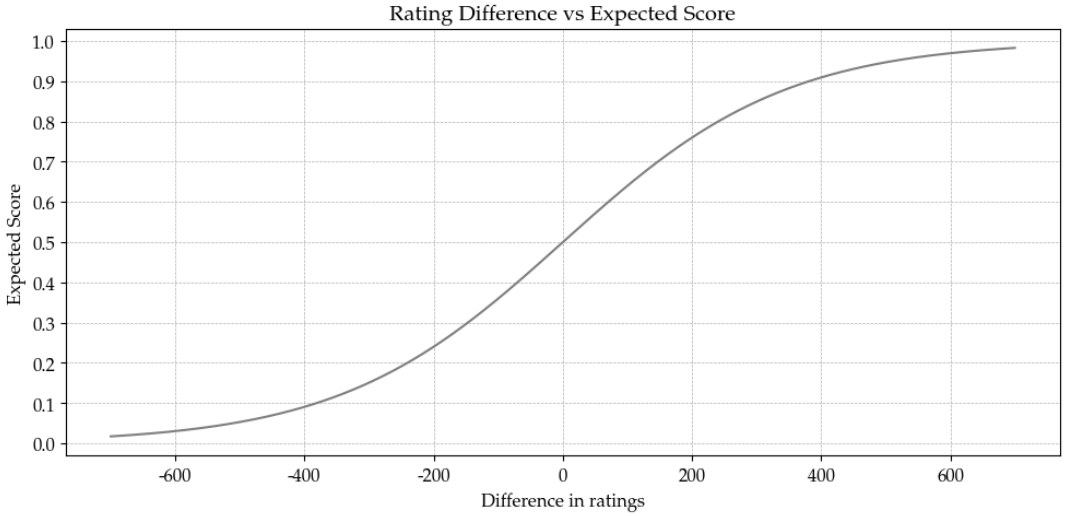


FIGURE 2.4: Difference in player Elo ratings vs expected score

Glicko

In practice, the K-factor used in the Elo system is not always held constant. For example, when trying to rank new players, it is sensible to allow their ratings to change rapidly, and decrease K over time, as more games are played and more becomes known about their skill level. The 'Glicko' system, created in 1995 by Mark Glickman [60], introduces the concept of a *Ratings Deviation* (RD), or standard deviation, to account for the uncertainty of a player's rating. A low RD indicates a higher level of certainty about the player's rating, whereas a high RD is typical for players who compete infrequently or only have a few games played.

A central idea to the Glicko system is the *rating period*. Ratings and RD are determined for each player at the start of a new rating period:

$$RD = \min\left\{\sqrt{RD_0^2 + c^2 t}, 350\right\}$$

The expected score for player A against player B is then given by:

$$E_A = \frac{1}{1 + 10^{g(RD_B)\cdot(R_B - R_A)/400}}$$

where $g(RD)$ is a function that scales the influence of the rating deviation.

The updated rating and rating deviation for player A after a series of games is calculated as:

$$R'_A = R_A + \frac{q}{(1/RD_A^2) + (1/d^2)} \sum_{i=1}^{n} g(RD_i)(S_i - E_i)$$

$$RD'_A = \sqrt{\left(\frac{1}{RD_A^2} + \frac{1}{d^2}\right)^{-1}}$$

where q is a constant, d^2 is a term derived from the expected scores and outcomes of the games played, and n is the number of games played in the rating period.

In contrast to the Elo system where both players' ratings change by the same magnitude, in the Glicko system this amount is governed by their opponent's RD.

TrueSkill™

TrueSkill™ is a skill rating system designed by Microsoft Research and published in January 2007 [61]. TrueSkill was designed specifically for multiplayer online games, with the intention of matching players together with similar skill levels such that the games were more "enjoyable, fair, and exciting". Each player's skill level is modelled as a Gaussian distribution, $\mathcal{N}(\mu, \sigma^2)$, where σ^2 is the variance of the distribution, and a smaller value reflects higher confidence in the player's skill level μ.

New players start with a default skill distribution. After each match, Bayesian inference is used to update and predict players' skill levels, using pairwise comparisons between each player in each team. These comparisons are evaluated using the cumulative distribution function of the difference of two players' TrueSkill ratings, which is also a Gaussian. The expected probability of player A beating player B can

therefore be calculated as:

$$P(A \text{ beats } B) = \Phi\left(\frac{\mu_A - \mu_B}{\sqrt{\sigma_A^2 + \sigma_B^2}}\right)$$

where (Φ) is the cumulative density function of the standard normal distribution, (μ_A) and (μ_B) are the skill estimates of players A and B respectively, and (σ_A) and (σ_B) are their corresponding uncertainties.

Over time and with more games played, the standard deviation σ of a player's skill distribution decreases, reflecting increased confidence in their skill level estimation.

TrueSkill™ expands upon the foundations of Elo and Glicko, with key advances being the ability to rate each individual in a team of players, and allowing for matches of more than two players or teams to to be modelled. For team games, TrueSkill treats the team's skill as the sum of the skills of its members. In their research paper, Herbrich et al. [61] demonstrated that TrueSkill worked well in their experiments with hundreds of thousands of Xbox Live players.

2.3 Predictive analysis in esports

Esports have the unique characteristic of being natively digital. This means that recording data from a given match is much cheaper and easier than traditional sports, and even amateur games can be recorded in fine detail. In the age of *big data*, esports are therefore well-positioned for data analysis.

This is reflected by the significant number of papers that have been written on the topic of prediction in different esports, with a particular emphasis on real-time match analysis. Hodge et al. [62] performed a comprehensive case study on live win prediction in Dota 2, and concluded that win prediction is difficult, where no single technique excels. They suggested combining techniques into ensembles to achieve better performance. These events were combined with pre-game information to construct the feature set. The ML algorithms used were logistic regression, random forests, and LightGBM, a performant gradient-boosting algorithm. They found that LightGBM outperformed both logistic regression and random forest. Dota 2 matches last 40 minutes on average but can be as short as 10 minutes. Noting a shortage of professional matches to use for analysis, their approach was to use an input data set of 5744 game replays from highly rated, but not necessarily professional players. The dataset was chronologically split, with two-thirds for training and the remainder for testing. They further tested their models on a significantly smaller set of 186 tournament matches. Despite a poor pre-match prediction accuracy of 55.8%, they found that when using features known 20 minutes into the game (the average half-way point of a match) they could predict the winner of professional games with 74.59% accuracy.

A similar study was conducted by Yang et al. [63], with an expanded dataset of 78263 Dota 2 matches. By considering player history, they could improve the pre-match feature set. This allowed the authors to achieve a 71.49% accuracy using logistic regression and 70.46% using neural networks. With live prediction, they found that a key factor affecting performance was the time of the match, with real-time features becoming increasingly informative as the match progresses. At the 40th minute of the match, their model achieved 93.73% prediction accuracy.

A number of studies have been done on different aspects of Counter-Strike gameplay. Bednárek et al. [64] described how to accurately parse data from demo files, which often contained inconsistencies or errors. They also described how player identities could be linked from the demo files to their online player profiles. The extracted demo-file data is ripe for live feature generation; Marshall et al. [65] used convolutional and recurrent neural networks to predict the probability of a player dying within the next 5 seconds. The input features were recorded from 542 matches, with 5 second windows of information containing features related to damage events. They found long short-term memory, a form of recurrent neural network, yielded the best performance.

Xenopolous et al. [66] created a framework, known as 'Win Probability Added', for valuing individual player contributions to their team's round-win percentage. They extracted over 70 million data points from 4682 demo files. Their research employed machine learning models such as logistic regression and gradient boosting (CatBoost and XGBoost) to generate the win probabilities. Another study [67] focused on modelling 'Optimal Economic Decisions' for a given round in CS. This study employed XGBoost and neural networks on a dataset derived from 6538 demo files. They found a correlation between spending errors and lower round win-rates.

Some research has been done in pre-match CS win prediction. Švec [68] used online match data to train machine learning models to predict match winners. They achieved 60% classification accuracy using graph convolutional networks, which was "disappointing" when compared to the performance attained with a comparatively simple Elo-based baseline prediction. Another study [69] aimed to create a decision support system to estimate the win probability for a given map. The researchers trained a linear regression and random forests model with player data from FACEIT, a popular third-party match-making service. They managed to attain an overall match prediction accuracy of 57.42%. A similar study [70] used k-means to cluster players together using data from their FACEIT matches. Once clustered, the player data was used to train a feed-forward neural network to determine their win probability. An accuracy of 65.11% was obtained, which exceeded their benchmark prediction made from win-rates alone.

2.4 Summary

It is evident that sports prediction using machine learning techniques is a popular field of research. The key factors affecting performance are the quality of data used, the features that can be derived, the choice of ML algorithms, and the sport in question [44]. Unpredictability is an inherent characteristic of sports, and esports are no different.

The studies surrounding Counter-Strike match prediction are noteworthy, however more work can be done to expand upon the results obtained: the quality and quantity of data, as well as the features derived therefrom, can be improved. Furthermore, studies typically measure the performance with respect to statistical metrics, such as classification accuracy, F1-score, and AUC. By expanding the scope of the study to generate betting odds, a comparison with real-world betting odds can be made, which may produce novel and valuable insights into the ever-expanding industry of online sports betting.

3 Methodology

All programming for this dissertation was written in Python [71] and the source code is accessible on GitHub.

3.1 Data Collection

HLTV is a household name in the Counter-Strike scene. Since its inception in 2002, it has evolved from a repository of CS 1.6 match recordings to a hub for all things Counter-Strike: fixtures, news articles, player interviews, and much more complement an enormous public database of match statistics.

Every professional and most semi-pro teams are tracked on HLTV; an extensive array of statistics for every match played at every tournament is reported. Furthermore, game play recordings, known as demo files, are hosted for the majority of high-level matches. These demo files can be downloaded and replayed to recreate each second of a given match.

Liquipedia is a viable alternative to HLTV; it is a collaborative online encyclopedia (a wiki) with records dating back to the very first CS tournament. Ultimately, HLTV was selected for its easy-to-navigate interface and increased depth of statistical data. Unlike Liquipedia, however, HLTV does not provide an API for retrieving the match data at scale. Thus a web-scraping solution is required. A web-scraper is a program which can automatically request, download, and parse information from web pages.

3.1.1 Requirements

The development requirements for data collection were therefore to:

1. Design a database to store desired match, map, and player data
2. Automatically download relevant match pages from HLTV
3. Parse information from the pages and store it in a database
4. Synchronize the database with the latest HLTV data

The following Python packages were identified as suitable for meeting these requirements:

- **SQLAlchemy**, for object-relational mapping to a **SQLite** database
- **ZenRows**, for efficiently downloading pages from HLTV
- **BeautifulSoup**, for reading and extracting data from downloaded pages

3.1.2 Database design

Matches

Every professional game of Counter-Strike is represented on HLTV by means of a **match** page. These list a wealth of information, such as the teams playing, the maps played, and the map veto sequence. There are also links to watch recordings of the match or download the demo file. An example of a match page is shown in Figure 3.1. The schema shown in Table 3.1 was determined after considering which data on the match page is relevant to the project.

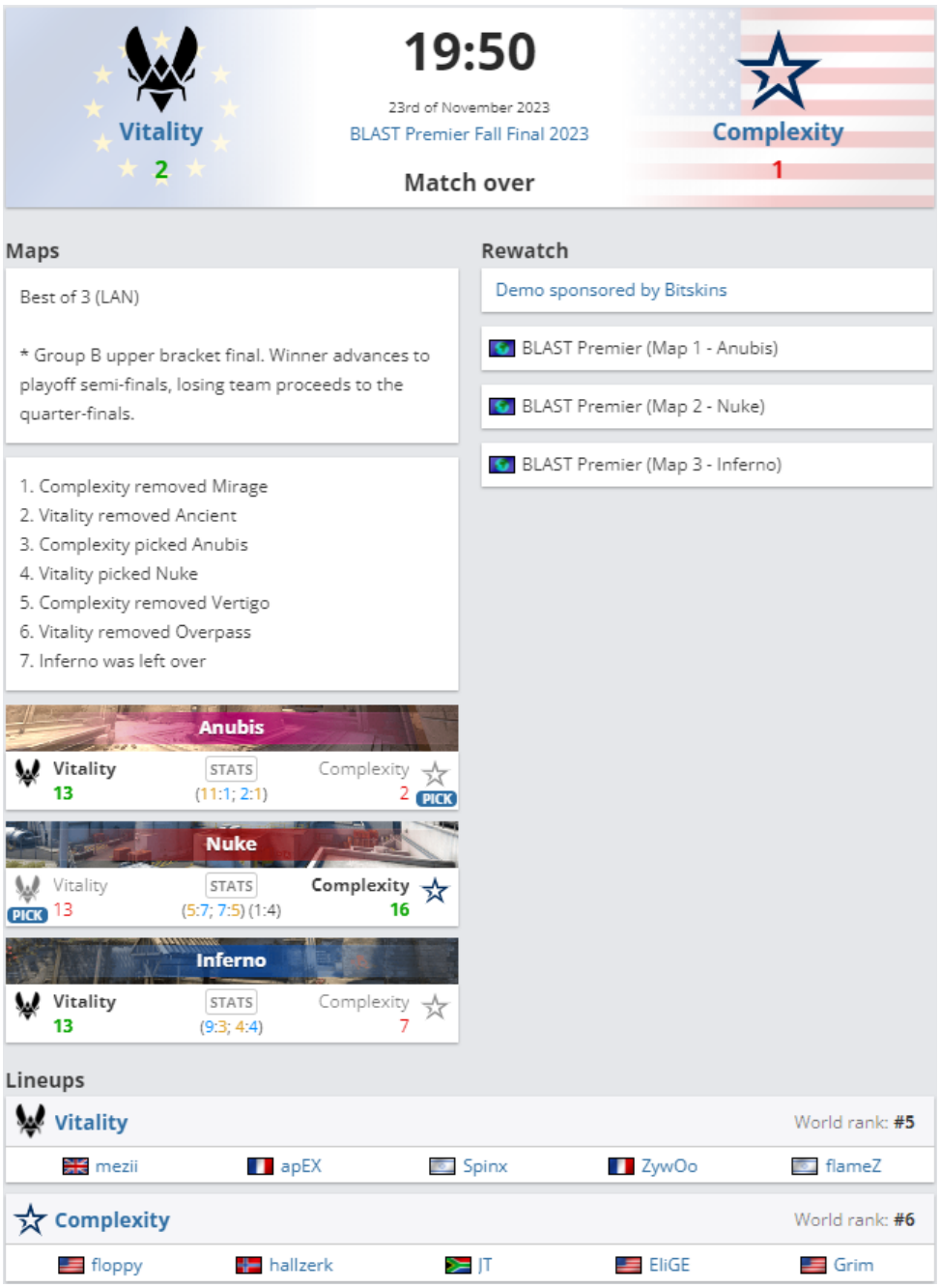


FIGURE 3.1: An HLTV match page for Vitality versus Complexity (cropped)

TABLE 3.1: Simplified schema for the `match` table

Field	Type	Description
`id`	`INTEGER`	Match ID, the primary key
`datetime`	`DATETIME`	Date and time at start of match
`team1_id`	`INTEGER`	Identifier for the first team
`team2_id`	`INTEGER`	Identifier for the second team
`team1_score`	`INTEGER`	Number of maps[1] won by first team
`team2_score`	`INTEGER`	Number of maps[1] won by second team
`event_id`	`INTEGER`	Identifier for the event
`event`	`STRING`	Name of the event
`lan`	`BOOLEAN`	Indicates if the match was played on a LAN or online
`best_of`	`INTEGER`	The match format (1/3/5)
`box_str`	`STRING`	Extra details reported on match page
`veto`	`STRING`	The order of the map veto process
`cs2`	`BOOLEAN`	Indicates if match was played in CS2 or CS:GO

The five players that make up each team are reported at the bottom of every match page in the 'Lineup' section. This information is useful to record as rosters change over time and players are sometimes substituted. It was thus decided to record the two line-ups for every match in a separate table. This schema is shown in Table 3.2.

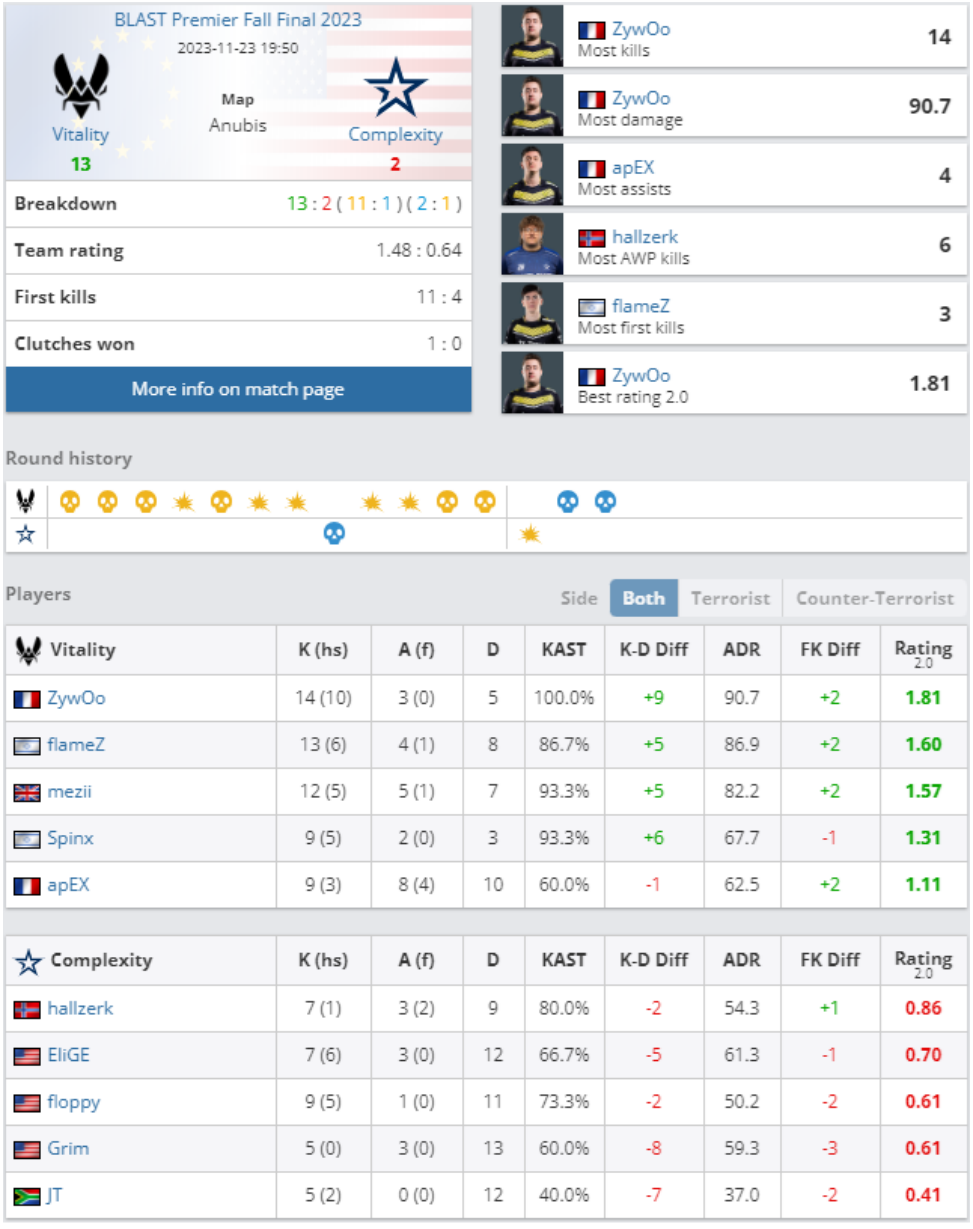
TABLE 3.2: Simplified schema for the `lineup` table

Field	Type	Description
`id`	`INTEGER`	Lineup ID, the primary key
`team_id`	`INTEGER`	Identifier for the team
`rank`	`INTEGER`	HLTV team rank at that time
`match_id`	`INTEGER`	Identifier for the associated match
`player1_id`	`INTEGER`	Identifier for the first player
`player2_id`	`INTEGER`	Identifier for the second player
`player3_id`	`INTEGER`	Identifier for the third player
`player4_id`	`INTEGER`	Identifier for the fourth player
`player5_id`	`INTEGER`	Identifier for the fifth player

[1] In a BO1, the score is the number of rounds won

Maps

Clicking on any of the individual map results listed on the match page will load the **map** page. These pages report more granular data for each map that was played in the match. A BO3 match, for example, consists of either two or three maps, all of which will have their own map page with detailed statistics. An example of a map page is shown in Figure 3.2, which was retrieved by clicking on the 'STATS' button on the corresponding tile on the match page. The schema shown in Table 3.3 was determined after considering which data on the match page is relevant to the project.



Vitality	K (hs)	A (f)	D	KAST	K-D Diff	ADR	FK Diff	Rating 2.0
ZywOo	14 (10)	3 (0)	5	100.0%	+9	90.7	+2	1.81
flameZ	13 (6)	4 (1)	8	86.7%	+5	86.9	+2	1.60
mezii	12 (5)	5 (1)	7	93.3%	+5	82.2	+2	1.57
Spinx	9 (5)	2 (0)	3	93.3%	+6	67.7	-1	1.31
apEX	9 (3)	8 (4)	10	60.0%	-1	62.5	+2	1.11

Complexity	K (hs)	A (f)	D	KAST	K-D Diff	ADR	FK Diff	Rating 2.0
hallzerk	7 (1)	3 (2)	9	80.0%	-2	54.3	+1	0.86
EliGE	7 (6)	3 (0)	12	66.7%	-5	61.3	-1	0.70
floppy	9 (5)	1 (0)	11	73.3%	-2	50.2	-2	0.61
Grim	5 (0)	3 (0)	13	60.0%	-8	59.3	-3	0.61
JT	5 (2)	0 (0)	12	40.0%	-7	37.0	-2	0.41

FIGURE 3.2: A map page for the first map played, Anubis, in the match between Vitality and Complexity (cropped)

TABLE 3.3: Simplified schema for the `map` table

Field	Type	Description
`id`	`INTEGER`	Map ID, the primary key
`match_id`	`INTEGER`	Identifier for the match
`t1_id`	`INTEGER`	Identifier for Team 1
`t2_id`	`INTEGER`	Identifier for Team 2
`map_name`	`STRING`	Name of the map
`t1_score`	`INTEGER`	Rounds won by team 1
`t2_score`	`INTEGER`	Rounds won by team 2
`t1_rating`	`FLOAT`	Team 1 HLTV rating
`t2_rating`	`FLOAT`	Team 2 HLTV rating
`t1_first_kills`	`INTEGER`	Team 1 first kills
`t2_first_kills`	`INTEGER`	Team 2 first kills
`t1_clutches`	`INTEGER`	Team 1 clutches[2]
`t2_clutches`	`INTEGER`	Team 2 clutches[2]
`t1_round_history`	`STRING`	Team 1 round outcomes
`t2_round_history`	`STRING`	Team 2 round outcomes

Each player's **performance** on a given map is reported on the table on the map page. Their performance is measured by a number of statistics, such as their kill-death ratio (KDR), average damage per round (ADR), and flashbang assists. These are described in table 3.4.

TABLE 3.4: Simplified schema for the `playerstats` table

Field	Type	Description
`player_id`	`INTEGER`	Player ID, part of the composite primary key
`map_id`	`INTEGER`	Map ID, part of the composite primary key
`kills`	`INTEGER`	Number of kills by the player
`hs`	`INTEGER`	Number of headshot kills
`assists`	`INTEGER`	Number of assists by the player
`flashes`	`INTEGER`	Number of times a thrown flashbang led to a kill
`deaths`	`INTEGER`	Number of player deaths
`kast`	`FLOAT`	Kill/Assist/Survival/Traded (KAST) percentage
`adr`	`FLOAT`	Average damage per round in hit points
`first_kd`	`INTEGER`	Number of times the player got the opening kill
`rating`	`FLOAT`	HLTV rating

[2] A clutch is when all of a player's teammates are eliminated and they still win the round

3.1.3 Downloading & processing

In order to populate the tables listed in the previous section, the web pages first needed to be downloaded and processed. By separating the scraping process into two discrete steps, downloading and processing, subsequent modifications could be made to the database and processing without the need to request any page data from HLTV a second time.

The ZenRows package was used to great effect to download the data from HLTV. Many modern websites, including HLTV, make use of Cloudflare and other services to protect against cyberattacks and other malicious traffic. This results in inconsistent responses when issuing requests programmatically. By issuing the requests via the ZenRows API, the rate of successful responses was over 99%.

Due to the sheer magnitude of pages of data, concurrency was implemented to vastly improve the rate at which pages could be downloaded and processed. A function was written which initiates ten ZenRows API client instances, loops through the list of URLs, requests each one, and downloads the HTML data in each response.

The next problem to solve was how to obtain the list of addresses for all the map- and match-pages.

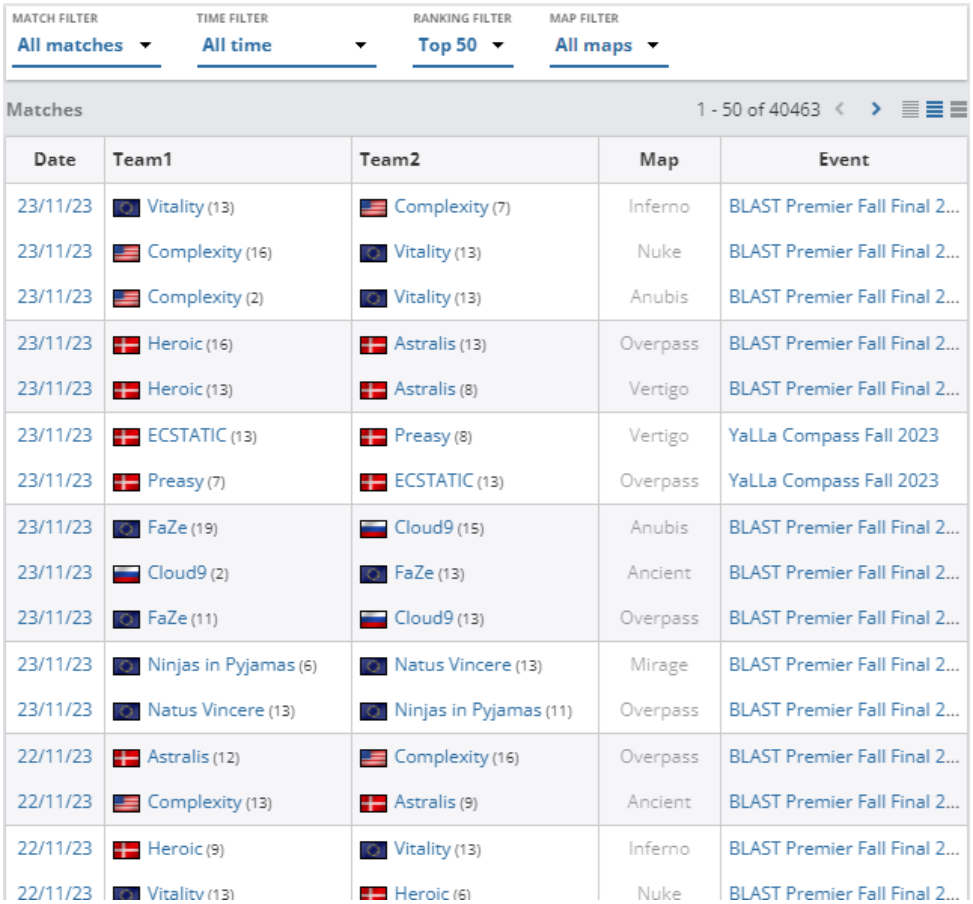
Finding the map links

HLTV conveniently has a results page that lists every single map in their database. An example of such a page is shown in Figure 3.3. The list can be filtered by event type (Majors, Big events, LAN, or online), date, ranking (where at least one of the teams playing must be ranked in the Top x teams), and Map (Inferno, Mirage, etc.). By navigating through this list, all the desired links to the map pages could be retrieved. This page will henceforth be referred to as the *map-link* page.

Two noteworthy decisions were made here: to only record data from the last five years (from 2019 to the present), and to only consider matches where at least one of the teams was ranked in the Top 50 teams as reported by HLTV's weekly rankings. This was done to narrow the scope to relatively recent matches played by professional teams.

Each map-link page reports a maximum of 50 map records, which means that the scraper needs to be able to navigate through multiple pages, depending on the amount of rows, to retrieve all the URLs. This was done by downloading the first page, then processing it to find the pagination range. This range was then used to form the list of remaining pages to download.

Once all the map-link pages had been downloaded, the list of URLs to the map pages themselves needed to be extracted. Each file was sequentially processed to isolate the map ID and URL for each map played. Further details could be collected later when the map pages themselves were scraped. This information was stored in its own table, `map_url`, along with two flags indicating if the respective map page had been downloaded and whether the map statistics had been processed and stored in the database.



Date	Team1	Team2	Map	Event
23/11/23	Vitality (13)	Complexity (7)	Inferno	BLAST Premier Fall Final 2...
23/11/23	Complexity (16)	Vitality (13)	Nuke	BLAST Premier Fall Final 2...
23/11/23	Complexity (2)	Vitality (13)	Anubis	BLAST Premier Fall Final 2...
23/11/23	Heroic (16)	Astralis (13)	Overpass	BLAST Premier Fall Final 2...
23/11/23	Heroic (13)	Astralis (8)	Vertigo	BLAST Premier Fall Final 2...
23/11/23	ECSTATIC (13)	Preasy (8)	Vertigo	YaLLa Compass Fall 2023
23/11/23	Preasy (7)	ECSTATIC (13)	Overpass	YaLLa Compass Fall 2023
23/11/23	FaZe (19)	Cloud9 (15)	Anubis	BLAST Premier Fall Final 2...
23/11/23	Cloud9 (2)	FaZe (13)	Ancient	BLAST Premier Fall Final 2...
23/11/23	FaZe (11)	Cloud9 (13)	Overpass	BLAST Premier Fall Final 2...
23/11/23	Ninjas in Pyjamas (6)	Natus Vincere (13)	Mirage	BLAST Premier Fall Final 2...
23/11/23	Natus Vincere (13)	Ninjas in Pyjamas (11)	Overpass	BLAST Premier Fall Final 2...
22/11/23	Astralis (12)	Complexity (16)	Overpass	BLAST Premier Fall Final 2...
22/11/23	Complexity (13)	Astralis (9)	Ancient	BLAST Premier Fall Final 2...
22/11/23	Heroic (9)	Vitality (13)	Inferno	BLAST Premier Fall Final 2...
22/11/23	Vitality (13)	Heroic (6)	Nuke	BLAST Premier Fall Final 2...

FIGURE 3.3: A map-link page, where each row is linked to a different map page

Downloading the maps and matches

Once the list of map URLs was recorded, the individual map pages could be downloaded. An example of a map page is shown in Figure 3.2. The majority of the metrics displayed on these pages were recorded in the database, with each map creating one record in the `map` table and ten records in the `playerstats` table. These schemas are described in Tables 3.3 and 3.4.

A match can consist of multiple maps. Each map page is therefore related to a corresponding match page. These pages can be accessed using the dark-blue button labelled "More info on match page" on the map page, as seen in Figure 3.2. An example of a match page is shown in Figure 3.1.

The links to the match pages were recorded in their own table in the database, `match-links`, in the exact same way as the map URLs were before. The match page contains valuable information such as the match format and veto process. These pages were downloaded and processed to create records in the `match` and `lineup` tables.

To summarize, there are two types of pages that display all data relevant to each match: a match page (Figure 3.1) and one or more map pages (Figure 3.2). In order to get to these pages, a list of map URLs first needed to be recorded by navigating through a series of map-links pages (Figure 3.3). Each time the scraping program is instructed to fetch new data, it starts by downloading the map-links page(s), then

TABLE 3.5: Schema for the `map_url` table

Field	Type	Description
`id`	`INTEGER`	Map ID, the primary key
`url`	`STRING`	The unique string in the map page URL
`downloaded`	`BOOLEAN`	Flag indicating if the page has been downloaded
`processed`	`BOOLEAN`	Flag indicating if the page has been processed

goes to each of the listed map pages, and finally, it downloads the corresponding match pages.

Processing and synchronization

Downloading a large number of pages can take a significant amount of time. This was mitigated in two ways, (1) by the use of tracking tables which prevent downloading the same page twice, and (2) by implementing concurrent ZenRows clients.

Processing thousands of pages of match and map data is an equally pedestrian process. To speed up this step, the processing functions are executed concurrently using multiple CPU threads. A wrapper function was implemented which takes a list of downloaded map or match pages, and initialises a thread pool using the `asyncio` package. Each thread sequentially reads data from the files, parses the data from the HTML, and returns a dictionary object which is appended to a list. Once all tasks have been completed, these lists of maps or matches are inserted into the corresponding table in the database *en masse*.

Finally, in order to keep the database up-to-date with the latest matches, a synchronization feature was developed. This function performs the full process described above, but starts by filtering the map-links page by the appropriate date range, starting with the last recorded date in the database. The URLs for the new maps can then be retrieved, and all the new pages are then downloaded and processed.

3.2 Feature Engineering

The dataset spans every official match played by a Top-50 ranked team between 2019 and 2023. This range includes a total of 24372 maps across 11271 matches.

In order to be used for machine learning, this abundance of information needs to first be processed and converted into a dataset of features. The relationship between the input features and the output label is then modelled by each ML algorithm. Each feature is therefore defined from the perspective of one team against an opponent, where the target variable y is a boolean class indicating whether they won the match or not.

The majority of the features are generated from historical match records. Particular care was taken to ensure that no *data leakage* occurred. Data leakage is when the training data contains information about the target variable [72]. An example of leakage would be if a win-rate is calculated by including the match being predicted. In other words, only matches played prior to the match in question were included in the calculations for the historical features.

3.2.1 Ranks and ratings

The purpose of any given ranking system is to order the constituents by how good they are. It follows that a higher-ranked or rated team is expected to beat a lower-ranked team. Ratings therefore serve as a proxy for expected performance. That said, there are many ways to rate and rank teams.

HLTV world ranking

HLTV maintains a world ranking which is updated weekly. Each team is ranked using a points system; a team will gain points depending on their recent form and their placement in LAN tournaments. There is an element of points-decay, where older achievements become progressively more irrelevant. In the event of roster changes, teams are required to retain at least 3 'core' players, otherwise their points will be lost [73].

A team's world ranking is a good indicator of their skill level; one would expect the No. 1 rated team to consistently beat the 50th ranked team. The HLTV ranking is, however, limited in resolution: it is only updated at weekly intervals, and when new teams are formed, they will be unranked irrespective of prior evidence of the players' skill.

The HLTV ranking for both teams in a match is conveniently displayed on the match page (3.1) in the Lineup section. In the full database of matches, 168 (1.5%) featured at least one unranked team. In these cases, the rank feature was imputed. A linear regression of every team's Elo rating and TrueSkill mean rating against their HLTV world rank was performed. This linear equation could then be used to impute an HLTV rating for the 168 records.

In addition to both team's world rankings, a relative feature, the difference between two teams world ranking, was included. These three features were defined as **team1_rank**, **team2_rank**, and **rank_diff**.

Elo rating

As discussed in 2.2.2, the Elo rating has some advantages over the HLTV ranking. Instead of a team's rating being updated at weekly intervals, it can be updated for each player following every match. In the event that a player moves to a different team, or joins an entirely new team, their Elo rating is retained. This has the clear advantage that a new team formed from established players can immediately be ranked using the average Elo rating of its constituent players.

In order to calculate these Elo ratings, each match, starting from the first match played in 2019, was chronologically processed. If a player did not already have an Elo rating stored in the table, their Elo was initialized to 1000. Each player's Elo rating was then evaluated using the outcome of the match in question. The player Elo ratings after each match were stored in a separate database table, `player_elo`. Additionally, the number of matches played for a given player, M, is maintained in the table, and incremented with each match played. This parameter was used to increase the sensitivity of the rating of new players by increasing the K-factor for the first 10 matches played using the following linear equation:

$$K = \begin{cases} 50 - 4M & \text{for } M \leq 10, \\ 10 & \text{for } M > 10. \end{cases}$$

The probability of either team winning was calculated using the 'expected score' formula in 2.2.2. Thus, three features could be included: the latest average team Elo rating on the date of the match for both teams, and the expected score E for team A: **t1_elo**, **t2_elo**, and **elo_win_prob**.

TrueSkill rating

The TrueSkill rating was calculated using a similar method as above: a table in the database stores each player's TrueSkill rating for every match, and each match is processed sequentially to calculate the rating for the next match. New players are initialized with the default initial TrueSkill rating, $(\mu_0, \sigma_0) = (25, \frac{25}{3})$, where μ_0 represents the initial skill estimate and σ_0 represents the initial standard deviation of the player's skill. These values are selected to provide a neutral yet sensitive starting point on the TrueSkill scale.

TrueSkill has an advantage over the Elo rating: instead of needing to calculate the average rating of the team, it natively supports teams of multiple players.

The `trueskill` Python package conveniently abstracts all the complex computation away. New ratings could simply be calculated by calling the `rate` function using a list of player TrueSkill ratings and the match outcome. TrueSkill has four global parameters, the default μ and σ values given to new players, and β and τ. The latter can be modified to adjust the dynamics of how ratings are updated, however these were kept at their default values ($\mu = 25, \sigma = \frac{25}{3}, \beta = \frac{25}{6}, \tau = \frac{25}{300}$).

For every match-up, five features were included: the average μ and σ for either team, as well as TrueSkill's prediction of the match outcome (a value between 0 and 1). This prediction can be interpreted as the probability of the first team winning. These features were denoted as **t1_mu**, **t1_sigma**, **t2_mu**, **t2_sigma**, and **ts_win_prob**.

Roster changes

As in other team sports, Counter-Strike rosters change over time; players may get better offers from other teams, or they may be replaced. In other cases, entirely new teams are formed. Because of the high level of team-work and preparation required to succeed in Counter-Strike, teams with more time and experience playing together usually have an advantage over more newly-formed squads. To account for this experience factor, features were added for the age of the line-up, measured in days, and the number of matches they have played together.

This was implemented using another table, `lineup_age`, which contains the date, match ID, days together, and matches played together for a given team's line-up.

For each new match, a team's current line-up is compared to the line-up for their last match. If the five same players are present in both line-ups, then their "experience" is increased: the line-up age (in days) is incremented using the difference between the two match dates, and the number of matches played is incremented by 1.

If there are different line-ups in consecutive matches, it suggests that the team roster has changed. The line-up experience features are thus reset to 0 days and 0 matches played together.

Often this may just be a temporary change if one player is ill or unable to attend the match. To account for this, every time a change is detected, an older match from at least month ago is found and that line-up is compared to the latest line-up. If these line-ups match, then the line-up experience features are restored to their correct values.

In this way, four features could be added describing each team's experience: a line-up's age (in days) as well as number of matches played together. These were denoted as **t1_age**, **t1_xp**, **t2_age**, and **t2_xp**.

3.2.2 Match statistics

The features described in this section were computed using a sliding window of the last 3 months' match history.

Perhaps the best indicator of a team's form is obtained by simply asking *'how often do they win?'*. The match win-rate was calculated by counting up their match wins and dividing by the total number of matches played in the last 3 months. Two features were thus added for either team: number of matches played and match win-rate. These were denoted as **t1_mp**, **t1_wr**, **t2_mp**, and **t2_xp**.

Additionally, the number of days since a team last played a match was included to account for any "rustiness". These features were called **t1_rust** and **t2_rust**.

Occasionally, teams can go on winning streaks. Although highly correlated with win-rate, this feature was added by looking back at a batch of their past matches and counting up their sequential wins. They were denoted as **t1_ws** and **t2_ws** respectively.

If two teams have recently played against each other, the outcome is particularly relevant. Features were added to indicate the number of "head-to-head" maps played, the proportion of maps won, and the average round-win percentage. These features were included from only the perspective of the first team, as the other team's

would simply be the complement. They were denoted as **h2h_maps**, **h2h_wr**, and **h2h_rwp**.

3.2.3 Map statistics

Different teams choose to specialize on different maps, and usually have one or more maps which they avoid playing entirely. This is made possible because of the map veto process described in 1.1.3. Although there are similarities between all maps, the strategies employed and the proficiency of teams on different maps vary. For this reason it was sensible to record how regularly a team plays and wins on the different maps.

For each team and each map, it was decided to record the number of times it was played, number of times it was won, and the percentage of rounds won. These features indicate how experienced and successful a team is on each map. This introduces three features for seven maps for either team - a total of 42 features.

It should be noted that Valve has changed the map-pool over the years, occasionally removing one map and introducing another. This means that from time to time, teams will have to learn how to play a new map. To handle these cases, the function which generates these features must have knowledge of these replacements. When it detects a deprecated map in a team's match history, it ignores it as it is no longer relevant for future matches. The same features which were used to represent a team's performance on the deprecated match were used to represent their performance on the new map.

Furthermore, three features were added to compare the overall map statistics between the two teams for maps played (map_xp), map win-rate (maps_wr), and round win-rate (map_rwr). These features were generated by summing up the number of maps on which team A was better than team B. For example, if team A had more maps played for 5 of the maps, and team B on only 2 of the maps, the feature 'map_xp' was equal to $5 - 2 = 3$. Similarly, if team B had a better map win-rate on 6 of the maps, and they had an equal win-rate on the seventh, then 'map_wr' was calculated as $0 - 6 = -6$.

To summarize, 45 features were added to represent either team's competitive proficiency on a map-by-map basis.

3.2.4 Performance statistics

A number of performance metrics are generated over the course of a game of CS. The team rating, number of first kills (also referred to as "entry" kills), and the number of clutches won by each team are all reported on the map page as seen in Figure 3.1.2. Due to their substantial influence on the trajectory of a team's economy, the number of pistol rounds won was also recorded.

For each player, HLTV reports several statistics to describe a player's performance:

- a rating encompassing their overall contribution in the map, which is centred at 1.00
- the average damage inflicted per round (ADR), measured in HP

- KAST, the percentage of rounds in which they either had a kill, assisted a kill, survived the round, or died but were subsequently traded (their death was immediately avenged with a kill)

To convert these player-specific statistics to a team, they were averaged for the 5 players present on a given match's line-up.

Given that maps can be as few as 13 rounds or as many as 60 given sufficient overtime phases, each round-related statistic is divided by the number of rounds played to obtain the averages. This results in 7 statistics reported for each map of gameplay.

To obtain the distribution of these statistics over the course of the last 90 days' worth of maps, the mean and standard deviation is computed, resulting in 14 features for either team. These are **team rating, first kills per round, clutches per round, pistol round win-rate, average player rating, average player KAST**, and **average player ADR**.

3.2.5 Relative features

The majority of the features discussed this far are measurements of a single team's performance, either team A or team B for a given match. For predicting the winner of a match, it is common to compare the statistics between each team. This can be done using a relative feature. For example, the TrueSkill win probability and difference in HLTV world ranking incorporate information relating to both teams in question.

In order to compute these relative features, the difference between each pair of features was calculated and included as an extra feature. This added a significant number of features to the dataset. It is important to acknowledge that having more features is not always a good thing, if fewer features can produce the same (or better) results. This will be further explored later.

3.2.6 Miscellaneous features

Finally, three categorical features were included. **lan** indicates whether a match is played in a LAN environment, and **elim** implies a "high-stakes" elimination match; this applies to all matches where the loser would be eliminated from the tournament, including lower-bracket and play-off matches. These features were included to account for the fact that teams may play better or worse under these high-pressure circumstances. The final feature was simply the match **format**, be it BO1, BO3, or BO5.

Each feature that has been discussed in summarized in Table 3.6 below. Many of the features listed correspond to 3 features: one for either team, as well as the relative feature, which in most cases is just the difference between the value of that feature for the two teams. 142 features were generated in total.

Feature	**Description**
Ranks and Ratings	
team1_rank	HLTV world ranking for team 1
team2_rank	HLTV world ranking for team 2
rank_diff	Difference between team 1 and team 2's world rankings
t1_elo	Average Elo rating for team 1
t2_elo	Average Elo rating for team 2
elo_win_prob	Expected score for team A
t1_mu	Average TrueSkill rating (μ) for team 1
t1_sigma	Average TrueSkill rating (σ) for team 1
t2_mu	Average TrueSkill rating (μ) for team 2
t2_sigma	Average TrueSkill rating (σ) for team 2
ts_win_prob	TrueSkill's prediction of the match outcome
t1_age	Days the current lineup of team 1 has been together
t1_xp	Number of matches played together by the current lineup of team 1
t2_age	Days the current lineup of team 2 has been together
t2_xp	Number of matches played together by the current lineup of team 2
Map Statistics	
{team}_{map}_mp	Number of times this team played this map
{team}_{map}_wr	The win-rate for this team on this map
{team}_{map}_mp	The round win-rate for this team on this map
map_xp	Number of maps where team 1 has more experience than team 2
map_wr	Number of maps where team 1 has greater win-rate than team 2
map_rwr	Number of maps where team 1 has greater round-win-rate than team 2
Match Statistics	
t1_mp	Number of matches played by team 1
t1_wr	Match win-rate for team 1
t2_mp	Number of matches played by team 2
t2_wr	Match win-rate for team 2
t1_rust	Days since team 1 last played a match
t2_rust	Days since team 2 last played a match
t1_ws	Current match win streak for team 1
t2_ws	Current match win streak for team 2
h2h_maps	Number of "head-to-head" maps played
h2h_wr	Proportion of "head-to-head" maps won
h2h_rwp	Proportion of "head-to-head" rounds won
Player Performance Statistics (mean and std. dev. for each team)	
hltv_rating	Overall team rating for the map played
fk_pr	Number of first kills per round
cl_pr	Number of clutches won per round
pl_rating	Average player performance rating
pl_adr	Average damage inflicted by each player per round
pl_kast	Average percentage of rounds with kill, assist, survival, or trade
pistol_wr	Pistol round win percentage
Miscellaneous Features	
lan	Indicates if a match is played in a LAN environment
elim	Indicates if a match is an elimination match
format	Match format (1, 3, or 5)

TABLE 3.6: Features used for training the machine learning models

3.3 Exploratory Data Analysis

The dataset spans every official match played by Top-50 ranked teams between 2019 and 2023. This range includes a total of 24372 maps played in 11271 matches.

3.3.1 Descriptive statistics

The dataset is well-balanced, with 51.91% of the match records having a class label of 1, and 48.09% having a class label of 0. In other words, team 1 won 51.91% of the matches and team 2 won 48.09%.

Plotting the distribution of different features for either target class is a useful way to visualize the extent to which features correlate with the target class. From Figure 3.4, it is clear that TrueSkill makes more confident predictions than the Elo rating system, as the inter-quartile range is greater for both classes. The Elo rating system rates the majority of matches as being relatively even. The HLTV ranking difference refers to the value of team 2's rank less team 1's rank (a positive value indicates that team 1 had a higher rank than team 2). All three features exhibit larger median values for the winning class ($y = 1$).

In each boxplot, values that are located 1.5 times the inter-quartile range below the first quartile and above the third quartile are considered outliers, and appear on the plot as hollow points.

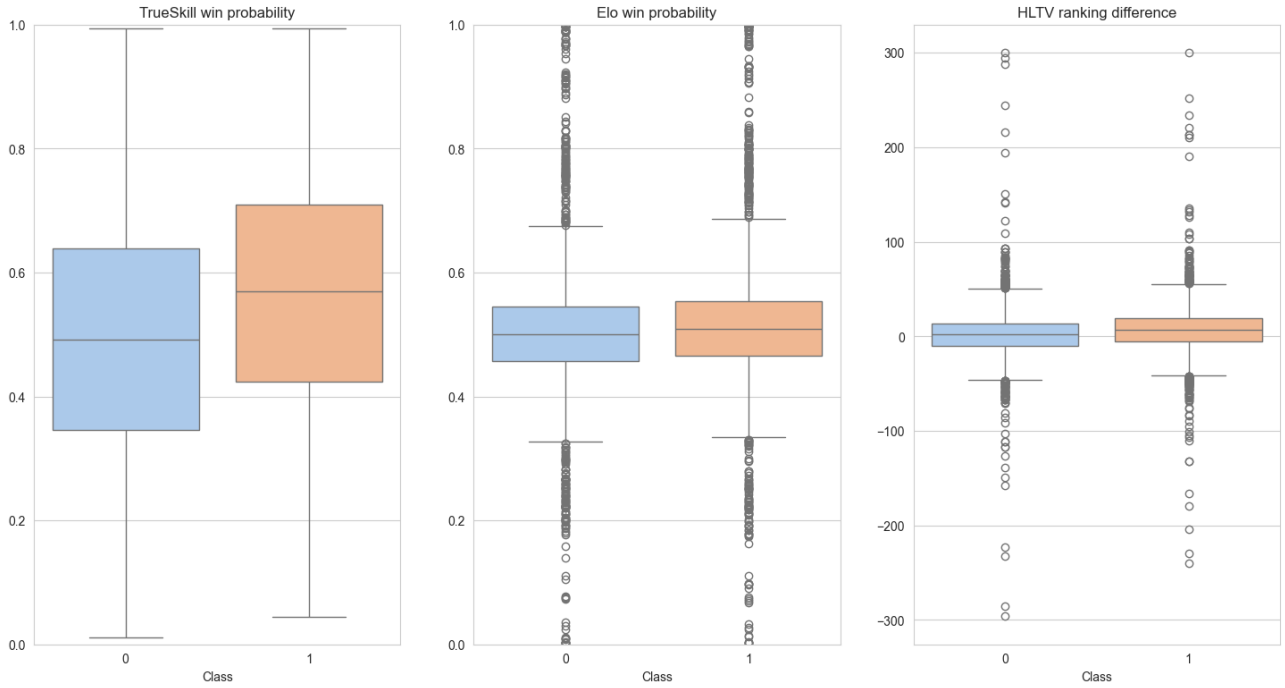


FIGURE 3.4: Boxplot of 'TrueSkill win probability', 'ELO win probability', and 'HLTV ranking difference' for either class

A similar trend is observed in Figure 3.5, which plots the distribution of various win rates. The winning team generally had a higher overall win-rate for prior matches. The number of maps where a team had a higher win-rate can range from -7 to 7 (as there are seven maps in the competitive map pool). The median value was 1 indicating that the winning team usually had at least one more map with a higher win-rate than their opponent. The head-to-head historical map win-rate feature is

also interesting, as there is a higher disparity between the classes. The conclusion that can be drawn here is that if team x beat team y more often than not in the past, they will likely continue to beat them in future. The IQR was, however, quite broad. Furthermore, for over 60% of the matches, both teams had not played each other in the last 3 months.

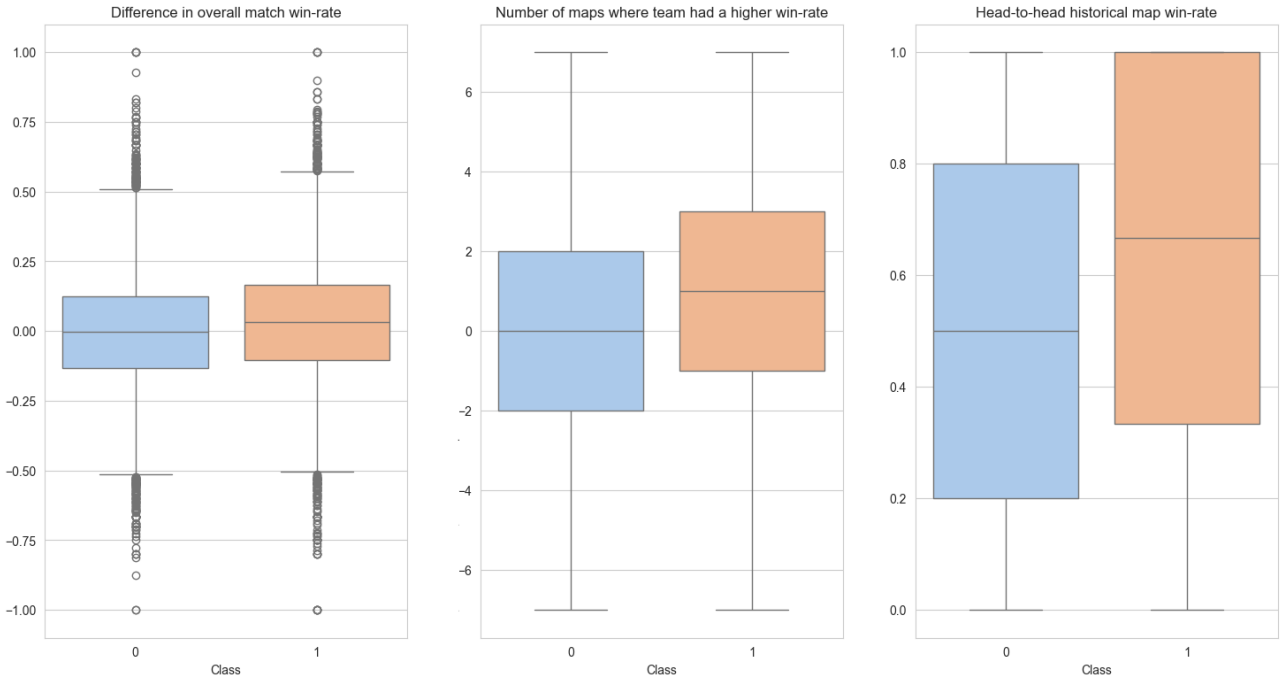


FIGURE 3.5: Boxplot of 'difference in match win-rate', 'maps with a higher win-rate', and 'head-to-head historical map win-rate' for either class

The most popular format in the dataset is BO3, with 8848 matches played (78.5%). There were a further 2234 BO1s and 142 BO5s played. The remaining 47 matches were played in BO2 maps, a format used briefly for lower-level league matches in 2019 and never again since.

2225 (19.7%) of the matches were played in a LAN environment, with the other 9046 (80.3%) being played online.

Shortly after its release in September 2023, the professional scene transitioned to CS2. This meant that only 428 matches in the database were played in the new game, with the vast majority (10843) played in CS:GO. As the dataset grows in future, the proportion will shift in favour of CS2 matches. Nevertheless, the core game has remained unchanged, and the same analysis is applicable for both iterations of Counter-Strike.

3.3.2 Feature correlation

It is useful to plot a correlation matrix to understand the relationships between pairs of variables. It can also help to identify multicollinearity, where independent variables are highly correlated. Highly correlated features essentially provide redundant information to a model, and can result in unstable and unreliable coefficient estimates. The parametric ML models described earlier can be regularised, which minimises the impact of multicollinearity.

The correlation matrix for thirty of the features is plotted in Figure 3.6. It is immediately observable that none of the features correlate strongly with the target class *win*. This is evident by the top row consisting of just pale colours.

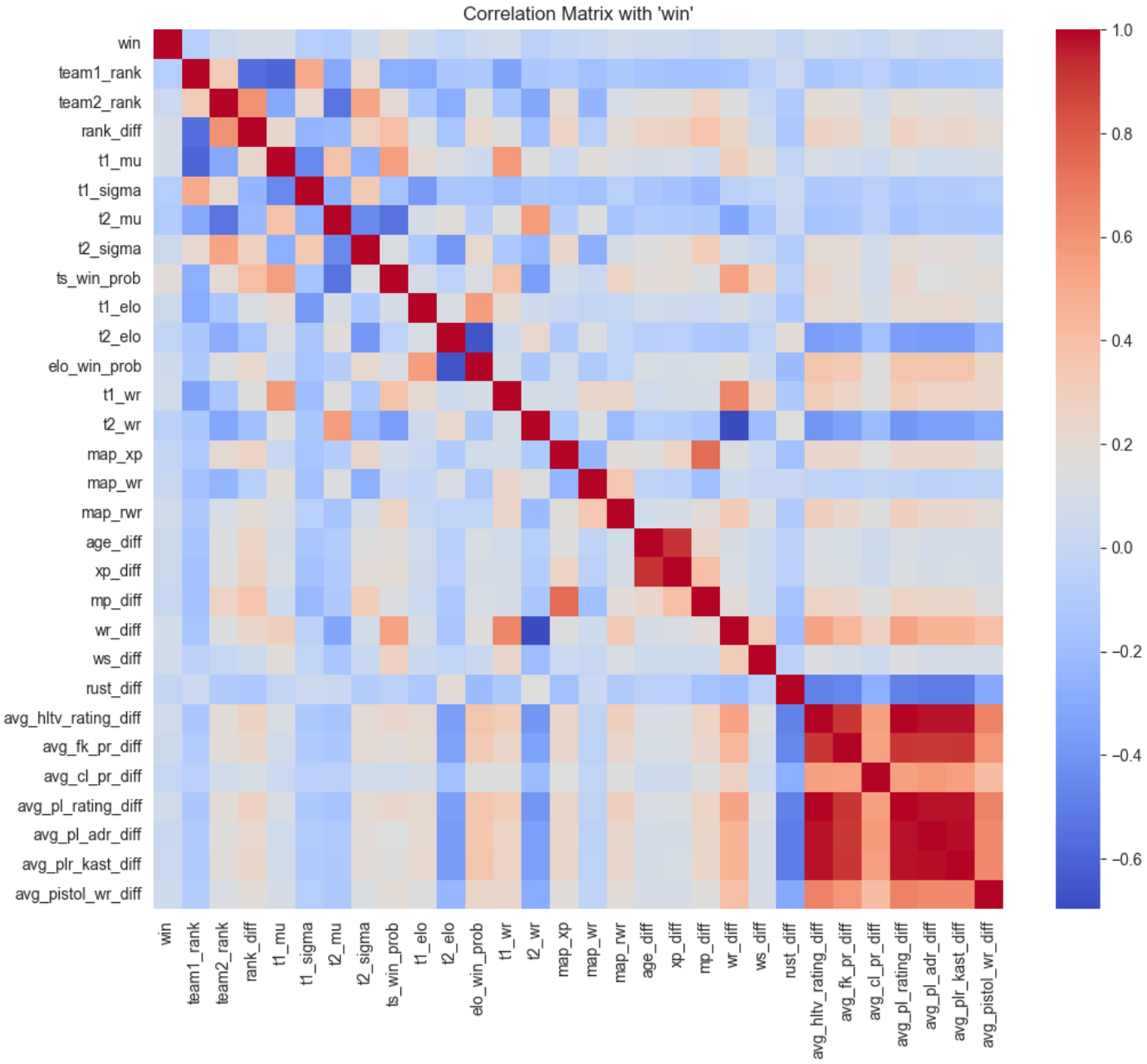


FIGURE 3.6: Feature correlation matrix heat-map

A negative correlation between 'team1_rank' and 't1_mu' exists, as well as for 'team2_rank' and 't2_mu'. This is a good indication that the TrueSkill system was implemented correctly. The best teams are ranked ascending from 1st, 2nd, 3rd - it is logical for teams with a low rank value to have a higher TrueSkill average (μ). The TrueSkill averages correlate with their respective match win-rates.

Lineup age (measured in days) and experience (measured in number of matches together) also exhibit an unsurprising correlation.

The red cluster in the bottom-right indicates a strong correlation between the performance features. This is not too surprising because generally, having a high ADR or KAST will lead to a higher player rating, and having higher player ratings leads to a higher team rating. Furthermore, the more 'clutches' and 'first kills' obtained per round, the higher your team's overall performance will be.

3.3.3 Baseline model predictors

It is useful to evaluate the performance of more basic models, with only 1 or 2 features, to form a baseline benchmark with which comparisons can be made. This will help to understand if the use of machine learning techniques offers a real benefit over a simpler analysis. To facilitate this, the most important features were identified via a feature selection process.

Feature selection refers to the selection of the most relevant features from a dataset. It is used to reduce over-fitting and simplify models. Features can be selected either based on a statistical measure (a filter method), or by evaluating the performance of models using different subsets of features (a wrapper method).

A filter method was implemented using the **`f_classif`** function from scikit-learn's **`feature_selection`** class. By computing the ANOVA F-value, or *F-statistic*, between each feature and the target class, each feature can be ranked and compared. The ANOVA F-value refers to the ratio of variance between the classes and the variance within the classes for each feature, thereby determining how well each feature can discriminate between each class, $y = 0$ or $y = 1$. If a feature has a higher ANOVA F-value, it has a stronger relationship with the target class and is potentially more useful for classification.

Other methods for feature selection were considered, such as **`SelectFdr`** which accounts for the family-wise error rate. However given the exploratory data analysis context, the filter method was deemed sufficient, and the best k features or top n^{th} percentile were identified using the **`SelectKBest`** function. The information presented in Figure 3.7 was obtained by setting $k = 15$.

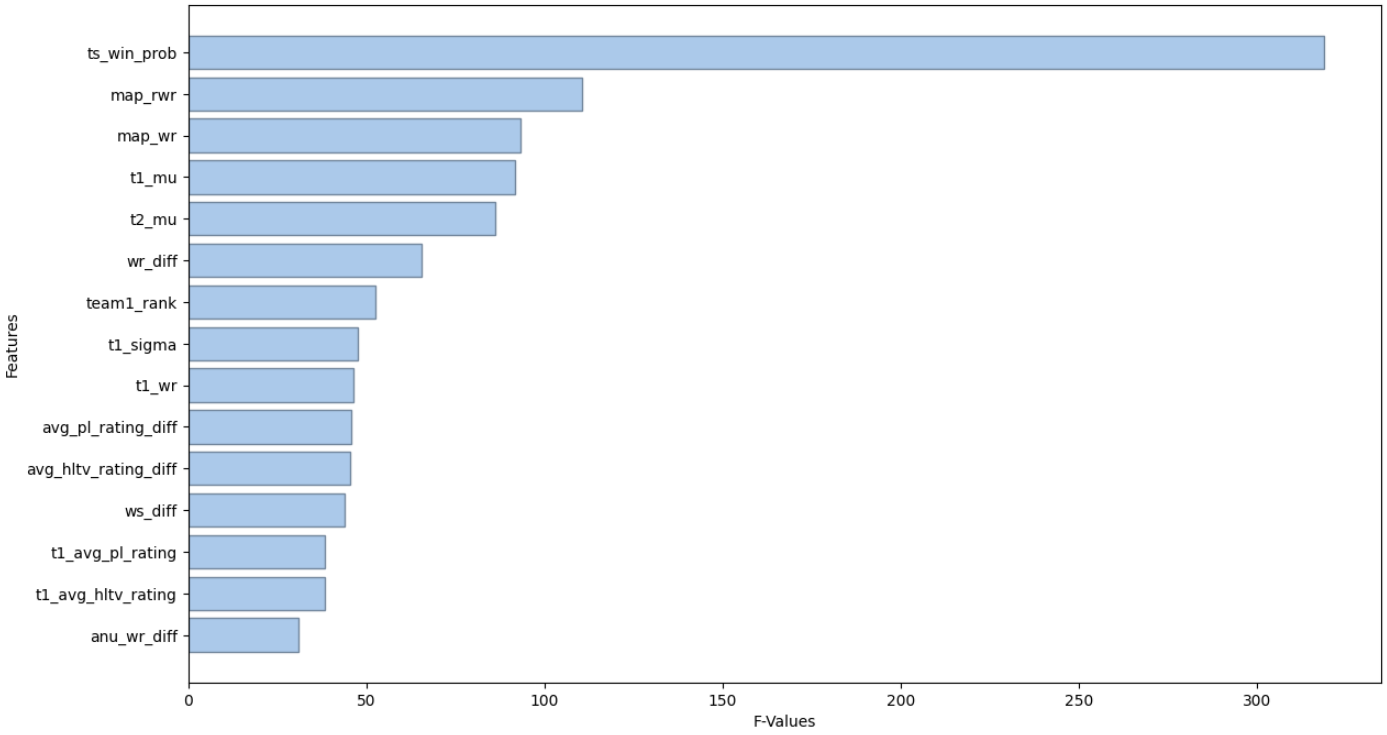


FIGURE 3.7: Barplot of best features by F-value

The TrueSkill features appear to be the best predictors of the match outcome, with the win probability and mean μ values for either team exhibiting the highest scores. The map win-rate differential, round win-rate differential, and HLTV world ranking all follow, but with a large drop-off in score.

From these insights, we can establish three baseline models:

1. the TrueSkill win probability model
2. the HLTV world ranking model, where the team with the higher rank is always predicted to win
3. the map win-rate model, where the team with the higher win-rate on more maps is always predicted to win. If both teams are equal, then the team with the higher overall round win-rate on more maps is predicted to win.

TrueSkill achieved a classification accuracy of 57.00% on the full dataset of 11271 matches. The confusion matrix is shown in Table 3.7.

From the matrix, a number of useful statistics can be computed. The true positive rate (TPR, also known as sensitivity or recall) is the proportion of actual positives correctly identified by the model. The true negative rate (TNR, also known as specificity) is the proportion of actual negatives correctly identified by the model. Precision refers to the proportion of positive classifications that were correct. The F1-score is a balanced measure of precision and recall. These metrics will be discussed in more depth later. The TrueSkill model achieved a true positive rate of 0.62, a true negative rate of 0.516, a precision of 0.58, and a F1-score of 0.60.

TrueSkill can evaluate the probability of a team winning, which means a receiver operating characteristic curve can be constructed by varying the threshold at which

	Predicted: No	**Predicted: Yes**
Actual: No	2516	2363
Actual: Yes	1999	3267

TABLE 3.7: Confusion matrix for the TrueSkill model

this probability is enough to make a positive classification. The area under the receiver operating characteristic curve (AUC ROC) is a measure of classifier's ability to distinguish between the two classes. As shown in Figure 3.8, the TrueSkill model is indeed a better predictor than random chance, with the area under the curve, $AUC = 0.5997$. The probabilities generated by TrueSkill may also be converted into betting odds. This aspect will be discussed later.

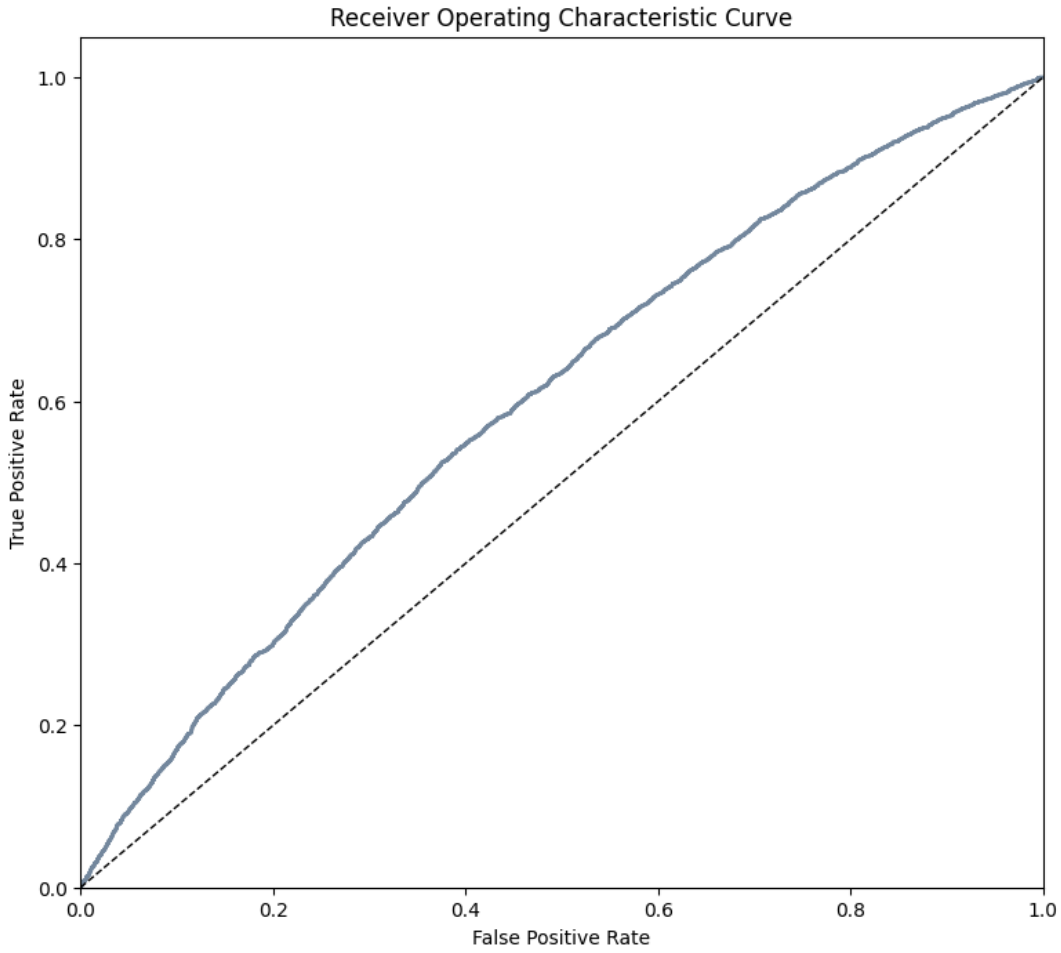


FIGURE 3.8: ROC curve for the TrueSkill model

The **HLTV world ranking model**, which assumes the higher-ranked team will always win the match, correctly classified the winner for 56.28% of the matches, achieving a slightly higher F1-score than TrueSkill of 0.61. The final baseline model, the **map win-rate model**, achieved only 54.95% classification accuracy and a lower F1-score of 0.579. As neither of these last two models produce classification probabilities, no receiver operating characteristic curve can be constructed.

3.4 Modelling

After considering the methods used in the literature, the following supervised learning models were selected and trained to predict the match winner:

- Logistic regression
- Random forests
- Support vector machine
- *k*-nearest neighbours
- Gaussian Naive Bayes
- Multilayer perceptron
- XGBoost

3.4.1 Data segmentation

To accurately assess the performance of any ML model, it must be tested on data which has not been used during training. The data should therefore be split into a training set and a testing set. Because the match data is chronological time-series data, it would not be appropriate to randomise the data prior to splitting, which is a common practice. It is more appropriate to segment the data chronologically.

A portion of the earliest match data was excluded from both training and testing. These matches were used to generate features for the later matches. This served two main purposes: the initialisation of Elo and TrueSkill ratings, and for providing historical data for most of the match- and performance-history features. This initialisation set amounted to the first 10% of the dataset, which was 1127 matches spanning a period of five months. This ensured that the features generated for the first match in the training set could draw upon accurate player ratings and sufficient match history for each team.

The remaining 90% of the data shall henceforth be referred to as the *full dataset*. It was split with a ratio of 80:20 for the training and testing sets. The test set consisted of every match played in 2023. This amounted to 8116 match records for training and 2029 for testing.

Filters were later applied to the full dataset in order to identify if certain types of matches were easier to predict than others. The different scenarios tested were:

1. the full dataset,
2. a dataset which excluded all Best-of-1 (single map) matches,
3. a dataset which excluded matches without a team ranked in the Top-30,
4. a dataset which excluded matches without a team ranked in the Top-15, and
5. a dataset which excluded all matches which were played online (i.e. LAN matches only)

The same process was performed for each dataset: data segmentation, feature scaling, feature selection, hyperparameter tuning, and performance evaluation.

3.4.2 Feature scaling

Some models, such as those which use gradient descent for optimisation, are sensitive to the magnitude of features. This means that each numeric feature should be *scaled* prior to training. There are two methods of scaling features, namely *normalisation* and *standardisation*. In normalisation, each feature is scaled to fit into the range $[0, 1]$ by subtracting the minimum value and dividing by the range, $(max - min)$. Although normalised data preserves the distribution of the original data, it is more sensitive to outliers.

In standardisation, the data is transformed to have a zero mean and unit variance using the familiar equation:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Where z_i is the new value, x_i is the original value, μ and σ are the mean and standard deviation of x.

Both techniques were implemented using the `MinMaxScaler` and `StandardScaler` classes from scikit-learn. During testing, it was found that normalisation degraded model performance significantly when compared to both the unscaled and standardised data. This is because normalisation is sensitive to outliers, which were present in many of the features. It was therefore decided to scale the features using standardisation.

3.4.3 Hyperparameter tuning

Each ML model has configuration settings which dictate the structure and computational performance of the model. These settings are known as *hyperparameters*. An example of a hyperparameter is the maximum depth of the decision trees used in random forests, or the choice of kernel used in a SVM.

When tuning these parameters to find the optimal configuration, it is a good practice to partition a portion of the training set into a validation set. The performance of models with different configurations can be compared by testing them on the validation set.

A technique known as k-fold cross-validation splits the training data into k folds (partitions). The training is then performed repeatedly with a different fold used as the validation set for each iteration. This approach provides a proxy for estimating the out-of-sample performance, ensuring that a balanced measure of performance is attained for each configuration of hyperparameters.

The `GridSearchCV` class from scikit-learn was utilized to test several different hyperparameters grids (configurations) using k-fold cross-validation. For many of the hyperparameters, a broad initial range of values was used. This range was progressively narrowed down in an iterative process until the configuration yielding the greatest average classification accuracy on the validation set was found.

The hyperparameters for each method and the range of values considered are explained in the following subsections.

Logistic regression

Table 3.8 presents the hyperparameters of the logistic regression model and the respective ranges of values explored during the tuning process.

Hyperparameter	Values Tested
1/Regularisation strength (C)	10^{-5} - 10^{2}
Regularisation type	None, L1 (Lasso), L2 (Ridge)
Solver	liblinear, lbfgs, saga

TABLE 3.8: Logistic regression hyperparameters and the range of values tested

Regularisation introduces a penalty term to the loss function which helps control the size of the coefficients. This can mitigate the risk of overfitting by preventing the model from relying too heavily on any single predictor variable. There are two *types* of regularisation, Lasso and Ridge.

C is the inverse of the *regularisation strength* parameter λ. Higher values of C therefore correspond to less regularisation, while lower values enforce stronger regularisation.

The *solver* is the algorithm used to minimise the loss function.

Random forests

Table 3.9 presents the hyperparameters of the random forests model and the respective ranges of values used in the tuning process.

Hyperparameter	Values Tested
Number of estimators	20 - 500
Maximum depth	2 - None
Minimum samples for split	2 - 10
Minimum samples per leaf	1 - 10
Maximum features	1 - None

TABLE 3.9: Random forests hyperparameters and the range of values tested

The *number of estimators* is the number of decision trees to include in the ensemble. Increasing this value can improve performance but also increases computational cost.

The *maximum depth* limits the maximum depth of each decision tree. Deeper trees can capture more complex relationships but may overfit the data.

The *minimum samples for split* are the amount of samples needed at a node in order to split it further. The *minimum samples per leaf* are the lowest amount of samples that can exist at a leaf node. The greater these values, the simpler the trees become.

The *maximum features* refers to the maximum amount of features considered when looking for the best split. Limiting the number of features reduces the risk of overfitting.

Support vector machine

Table 3.10 presents the hyperparameters of the support vector machine and the respective ranges of values explored during the tuning process.

Hyperparameter	Values Tested
Kernel	'linear', 'poly', 'rbf', 'sigmoid'
Regularisation (C)	10^{-3} - 10^{3}
Degree (for 'poly' kernel)	2 - 5
Gamma (for 'rbf' kernel)	10^{-5} - 10^{0}

TABLE 3.10: Support Vector Classifier hyperparameters and the range of values tested

The *kernel* function defines the shape of the decision boundary.

For a support vector classifier, Ridge regularisation is implemented by means of a penalty term for the complexity of the decision boundary. The *regularisation* parameter C is the inverse of the strength of regularisation, where lower values of C correspond to stronger regularisation, while higher values correspond to weaker regularisation

For the polynomial kernel, the *degree* specifies the polynomial order, and for the RBF kernel, the *gamma* parameter defines the kernel width.

k-nearest neighbours

Table 3.11 presents the hyperparameters of the *k*-nearest neighbours classifier and the respective ranges of values explored during the tuning process.

Hyperparameter	Values Tested
Number of neighbours	1 - 30
Weight function	'uniform', 'distance'
Algorithm	'auto'
Power parameter (p)	1 - 2

TABLE 3.11: *k*-nearest neighbours hyperparameters and the range of values tested

The *number of neighbours* are the number of neighbouring data points which are considered for classification. A higher value can reduce noise but may also smooth out decision boundaries.

The *weight function* assigns a weight to each neighbour based on its distance from the query point. 'uniform' gives equal weight to all neighbours, while 'distance' weights closer neighbours more heavily.

The *algorithm* parameter specifies the algorithm used for computing nearest neighbors. 'auto' selects the best approach based on the dataset.

The *power* parameter (p) is used for the Minkowski metric, which calculates the distance between data points. The Manhattan distance is used when $p = 1$, and the Euclidean distance is used when $p = 2$.

Gaussian Naive Bayes

The Gaussian Naive Bayes classifier is a relatively simple model and has only a single hyperparameter, *var_smoothing*, which is a small positive value added to the variance of each Gaussian used in the model. Adding this smoothing value can prevent numerical instabilities if the variance of a feature approaches 0. The range of values tested was 10^{-10} - 10^{-5}.

Multilayer perceptron

Table 3.12 presents the hyperparameters of the MLP classifier and the respective ranges of values explored during the tuning process.

Hyperparameter	Values Tested
Hidden layer sizes	*see below*
Activation function	'identity', 'logistic', 'tanh', 'relu'
Solver	'lbfgs', 'sgd', 'adam'
Learning rate	'constant', 'invscaling', 'adaptive'
Alpha	0.0001, 0.001, 0.01, 0.1
Batch size	'auto', 20 - 400
Max iterations	200 - 10000

TABLE 3.12: Multilayer perceptron hyperparameters and the range of values tested

The *hidden layer sizes* are the number of neurons in each hidden layer. Different configurations of hidden layers can affect the model's ability to capture complex patterns. The hidden-layer topologies tested include wide (200,200), narrow (20,20), bottleneck (200, 50, 200), and hierarchical (200, 100, 50). The amount of neurons in each hidden layer in each topology was varied from 1 to 400. The maximum number of hidden layers tested was five.

The *activation function* defines the function used for activating neurons in the hidden layers, where 'identity' is a linear activation, 'logistic' is the sigmoid function, 'tanh' is the hyperbolic tangent, and 'relu' is the rectified linear unit.

The *solver* is the optimization algorithm used to update the weights. 'lbfgs' is an quasi-Newton method, 'sgd' is stochastic gradient descent, and 'adam' is a stochastic gradient-based optimizer.

The *learning rate* controls the step size used when updating the weights. 'constant' uses a fixed learning rate (0.001), 'invscaling' gradually decreases the learning rate, and 'adaptive' maintains the learning rate unless the training loss ceases to improve.

The *alpha* parameter is the regularisation term, similar to λ in logistic regression and SVMs.

The *batch size* is the number of samples processed before the model is updated. Smaller batch sizes allow the model to learn faster but can be more unstable, while larger batch sizes provide more stable updates but may take longer to converge.

The *max iterations* parameter is the maximum number of iterations the solver will run for. This allows "early stopping" if convergence is not reached after max_iter. For stochastic solvers ('sgd', 'adam'), this is the number of times each data point is used (epochs).

XGBoost

Table 3.13 presents the hyperparameters of the XGBoost classifier and the respective ranges of values explored during the tuning process.

Hyperparameter	Values Tested
Number of estimators	50 - 10000
Learning rate	0.001, 0.01, 0.1, 0.2, 0.3
Max depth	2 - 10
Subsample	0.5 - 1.0
Colsample by tree	0.5 - 1.0
Gamma	0, 0.1, 0.2, 0.3

TABLE 3.13: XGBoost hyperparameters and the range of values tested

The *number of estimators* is the number of trees in the model. More trees can increase the model's performance but also the risk of overfitting.

The *learning rate* (eta) controls the step size at each iteration while minimising the loss function. Lower learning rates make the model more robust to overfitting by shrinking the feature weights.

The *max depth* parameter defines the maximum depth of a tree. Deeper trees can model more complex relationships but are more prone to overfitting.

The *subsample* parameter is the fraction of samples to be used for fitting each tree, while *colsample by tree* is the fraction of features to be used for each tree. These parameters prevent overfitting by adding randomness.

The *gamma* parameter specifies the minimum reduction in loss needed to split a node, thereby controlling the complexity of the trees.

3.4.4 Performance evaluation

Once each model has been trained, each match in the test set is classified into either class, $y = 1$ (team 1 wins), or $y = 0$ (team 1 loses, i.e. team 2 wins).

The classification accuracy refers to the percentage of matches that was correctly classified by each model. It is reported as **Test ACC** in the results tables in the following chapter.

Precision is a measure of the accuracy of positive predictions. It is the percentage of matches that were actually won out of the set of positive predictions. **Recall**, also known as sensitivity, measures the proportion of true positives that are correctly classified by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-score is another measure of predictive performance which balances both precision and recall.

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

Finally, the **area under receiver operating characteristic curve** (AUC ROC) is a measurement of the classification performance of a model at different threshold settings. The ROC curve plots the True Positive Rate (recall) against the False Positive Rate as the discrimination threshold is varied from 0 to 1. A random classifier will therefore, on average, have a straight line from (0,0) to (1,1) with a corresponding AUC ROC of 0.5. An AUC ROC of 1.0 indicates that the model can perfectly distinguish the positive and negative classes.

3.4.5 Betting odds generation

Each of the models generate the probability $p(y)$ of an observation belonging to each class. These probabilities are then used to generate the betting odds. Odds can be represented in a number of ways. For this research, all odds are denoted as decimal odds, which is a decimal number representing the payout per unit wagered.

The decimal odds are the reciprocal of the probability of the event occurring as shown in the following equation:

$$(O_A, O_B) = \left(\frac{1}{p(y=1)}, \frac{1}{p(y=0)} \right)$$

Where O_A and O_B are the decimal odds for team A and B to win the match, and $p(y)$ is the probability of team A winning $(y = 1)$ or team B winning $(y = 0)$. In practice, these odds represent the *fair* odds which exclude the effect of the bookmaker margin: a reduction in the decimal odds to increase bookmaker profitability.

3.4.6 Betting odds evaluation

In order to compare the generated betting odds with real betting odds provided by bookmakers, the average odds for 347 matches across 11 tournaments in the test set (2023) were downloaded from an online archive at `oddsportal.com`. To remove the bookmaker margin, the odds were then converted back into probabilities and normalized such that the sum of each pair of probabilities added up to 1.

The correlation, mean absolute error (MAE), and root mean square error (RMSE) between the generated probabilities and the bookmaker probabilities are reported for each model to quantify the discrepancy between the odds from the models and the bookmaker. These are formulated as follows:

Given that the generated probabilities are P_g and the bookmaker probabilities are P_b, with n being the total number of probability comparisons, the **correlation** is defined as:

$$r = \frac{\sum_{i=1}^{n}(P_{g_i} - \bar{P_g})(P_{b_i} - \bar{P_b})}{\sqrt{\sum_{i=1}^{n}(P_{g_i} - \bar{P_g})^2 \sum_{i=1}^{n}(P_{b_i} - \bar{P_b})^2}}$$

where:

- P_{g_i} and P_{b_i} are the i-th generated and bookmaker probabilities, respectively.
- $\bar{P_g}$ and $\bar{P_b}$ are the means of the generated and bookmaker probabilities.

The **mean absolute error** (MAE) is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|P_{g_i} - P_{b_i}|$$

And finally, the **root mean square error** (RMSE) is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_{g_i} - P_{b_i})^2}$$

3.4.7 Betting simulation

Finally, a betting simulation was performed to test each model's real-world performance using the following basic strategy: the bookmaker odds for each match was first converted into a probability. If the model generated a higher probability for either team to win than what was offered by the bookmaker, a one unit bet was placed on that team.

For example, consider a scenario where the bookmaker odds for team A : team B is 1.28 : 3.36. These odds correspond to $p(y = 1) = 0.78$ and $p(y = 0) = 0.30$ (the sum is greater than 1 because of the bookmaker margin).

If the model then predicts that $p(y = 1) = 0.65$ and $p(y = 0) = 0.35$ for this match, it would place a one unit bet on team B, despite it having a lower probability of winning.

On the other hand, if it had generated $p(y = 1) = 0.75$ and $p(y = 0) = 0.25$, it has determined that neither team is worth betting on.

The proportion of matches bet on, win-percentage, loss-percentage, and resulting profit or loss for each model is reported in the Results.

4 Results and Discussion

4.1 Model Performance

The performance of each machine learning model on each dataset is presented in the following tables. The classification accuracy on the training set, as well as the classification accuracy, precision, recall, F1 score, and ROC AUC on the test set, are reported. All values are reported as percentages unless otherwise indicated. The best result achieved for each metric is highlighted in **bold**.

4.1.1 Full dataset

The full dataset consists of 10145 matches, with 8116 used for training and 2029 for testing. Each model was first trained using all 142 features. The results obtained for this setup are tabulated in Table 4.1.

Model	Train Acc.	Accuracy	Precision	Recall	F1 Score	ROC AUC
TrueSkill	56.8	57.9	57.2	62.4	59.7	61.6
Logistic regression	60.0	58.5	56.9	69.3	62.5	62.9
Random forests	64.3	58.3	56.7	69.9	62.6	62.5
Support vector machine	59.5	59.4	**58.3**	66.3	62.0	63.0
k-nearest neighbours	58.9	55.9	55.3	61.2	58.1	57.8
Gaussian Naive Bayes	56.6	55.7	57.4	44.5	50.1	59.0
Multilayer perceptron	59.9	59.7	59.0	63.8	61.3	62.1
XGBoost	61.0	**60.0**	**58.5**	**69.6**	**63.5**	**64.1**

TABLE 4.1: Performance of ML models trained with all features on complete dataset

The ML models did not significantly outperform the TrueSkill baseline model, with the improvement in classification accuracy varying from -2% to 2%. XGBoost and MLP had the best overall performance with a classification accuracy of 60.0% and 59.7% respectively. XGBoost had the highest recall, F_1 score, and AUC ROC, indicating slightly better ability to discriminate between classes.

The optimal set of hyperparameters for each model was found using grid search and is captured in Table 4.2 below.

Hyperparameter	**Description**	**Value Selected**
	Logistic regression	
C	Inverse of regularization strength	0.005
penalty	Type of regularization	"l2"
solver	Optimization algorithm	"saga"
	Random forests	
n_estimators	Number of trees in the forest	34
max_depth	Maximum depth of the tree	6
min_samples_split	Min samples to split a node	2
min_samples_leaf	Min samples to be at a leaf node	4
max_features	Number of features for best split	16
	Support vector machine	
C	Regularization parameter	0.1
kernel	Kernel type	"linear"
	k-nearest neighbours	
n_neighbors	Number of neighbors to use	50
weights	Weight function in prediction	"uniform"
algorithm	Algorithm to compute nearest neighbors	"auto"
p	Power parameter for Minkowski metric	1 (Manhattan)
	Gaussian Naive Bayes	
var_smoothing	Variance smoothing	1e-9
	Multilayer perceptron	
hidden_layer_sizes	Neurons in hidden layers	(5,4)
activation	Activation function	"relu"
solver	Solver for weight optimization	"sgd"
alpha	L2 penalty (regularisation term)	0.001
learning_rate	Learning rate for weight updates	"constant"
	XGBoost	
n_estimators	Number of gradient boosted trees	208
learning_rate	Step size shrinkage	0.025
max_depth	Maximum depth of a tree	2
subsample	Subsample ratio of the training instances	0.5
colsample_bytree	Subsample ratio of columns for each tree	0.5
gamma	Min loss reduction for partition on a leaf node	0.1

TABLE 4.2: Hyperparameters and range of values tested for each ML model implemented

As a consequence of the large number of features, the models were prone to overfit the training data. This was especially pronounced with random forests, XGBoost, and MLP, which when initially configured could achieve near perfect accuracy on the training set, but otherwise performed poorly on the test set. In the case of random forests and XGBoost, this was mitigated by reducing the maximum depth of the trees to only 2 splits. For MLP, the number of nodes per hidden layer had to be quite low (5,4) to avoid overfitting.

The weakest performers were *k*-nearest neighbours and Gaussian Naive Bayes, exhibiting worse performance than the TrueSkill base model.

The ROC curve for TrueSkill and the best performing ML model, XGBoost, are plotted in Figure 4.1. It is evident that both models are better at predicting match outcomes than random chance, however XGBoost is slightly superior with an AUC of 0.641 vs 0.624.

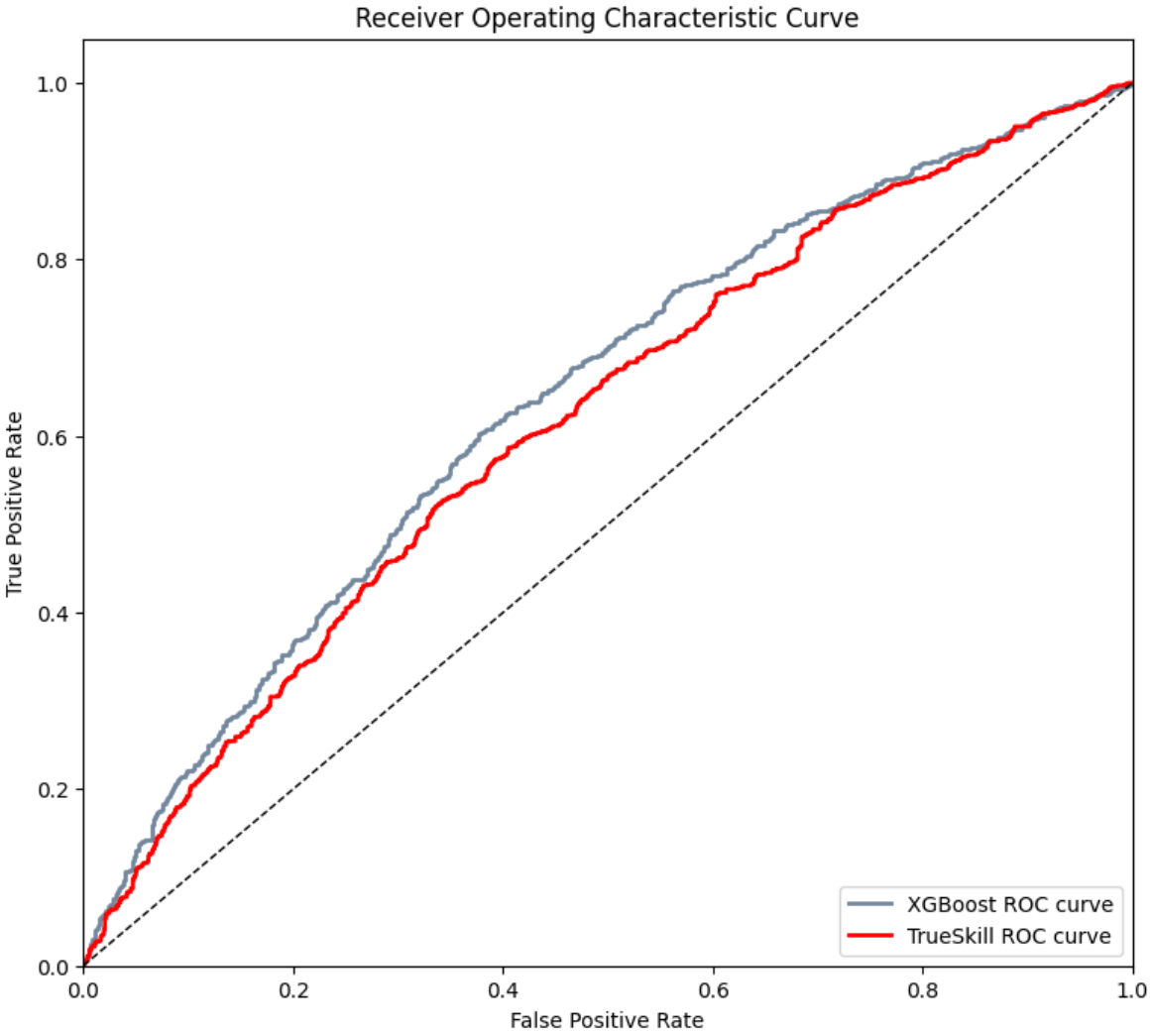


FIGURE 4.1: Receiver operating characteristic curve for TrueSkill and XGBoost

Feature importance

Feature selection methods such as filter and wrapper methods have already been discussed in section 3.3.3. Some models can also be used to determine feature importance. In a regularised logistic regression model, the magnitude of the coefficients for each feature are indicative of their importance given a large enough regularization penalty.

For SVMs with linear kernels, the weights assigned to each feature indicate their importance. With tree-based methods like random forests and XGBoost, the features yielding the largest average reduction in the loss function are more important. Figure 4.2 shows the feature importances determined by XGBoost.

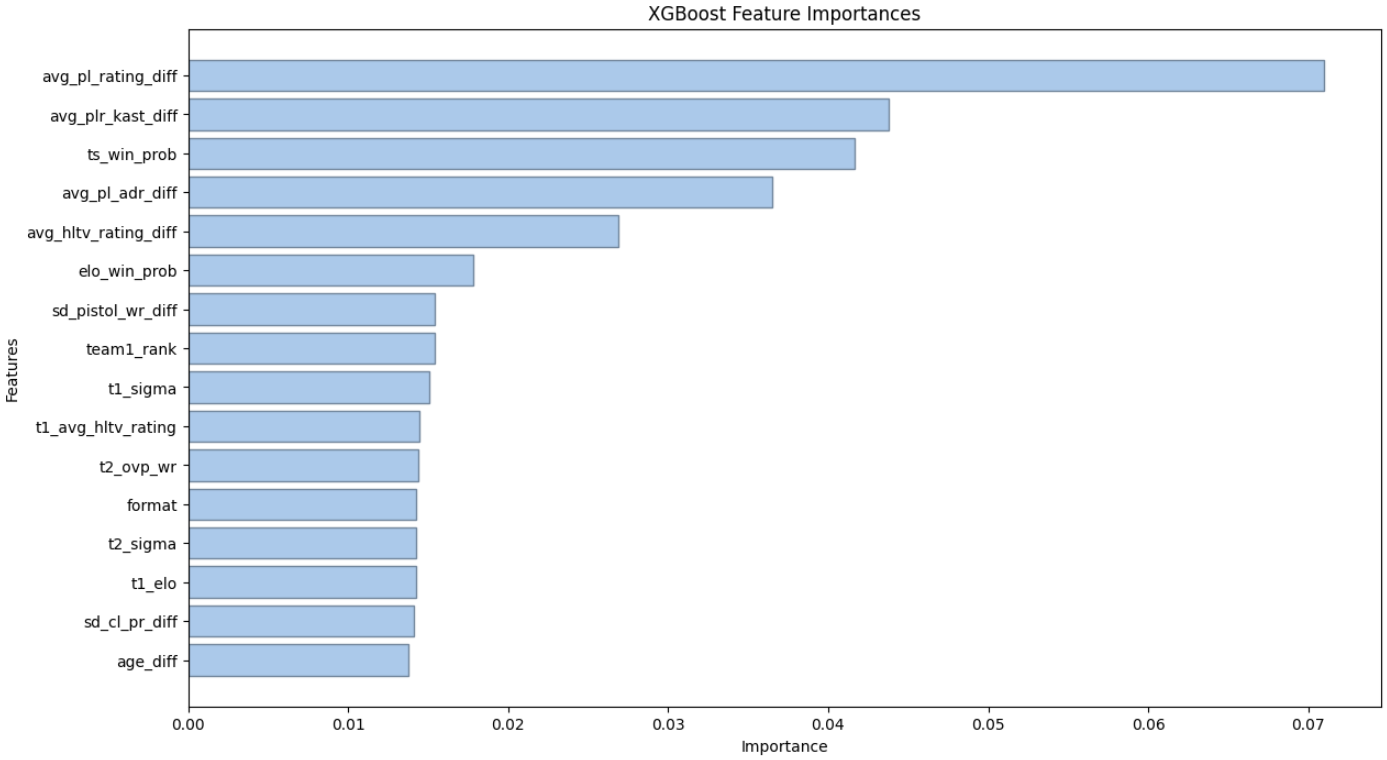


FIGURE 4.2: Feature importance from XGBoost

In contrast to the filter methods in section 3.3.3, XGBoost considers the difference in average player statistics (rating, KAST, ADR, and HLTV rating) between the two teams to be highly relevant measures. Both TrueSkill and Elo win probabilities are taken into account as well, whereas previously the Elo measure was discarded.

With logistic regression, the feature importances differed considerably. As seen in Figure 4.3, the TrueSkill win probability and μ, σ values for either team were very important, followed by the difference in average map win-rate, round win-rate per map. HLTV world ranking, and win-streak.

The results reported in Table 4.3 below were obtained after reducing the number of features to 18, an 87% reduction in the data provided to each model. This smaller set of features was determined by considering which features appeared in the feature importance lists produced in the previous analysis as well as by the f-classif filter method discussed in the Methodology (3.3.3). The number of matches in the training and testing sets remained the same.

The reduction in the number of features did not significantly degrade performance. In fact, the Recall, F_1 score, and ROC AUC all increased on average - likely due to a reduction in noise. This strongly suggests that many of the features do not meaningfully contribute to an accurate match prediction and can safely be ignored.

Occam's razor recommends searching for an explanation constructed from the smallest possible set of elements. This begs the question: how many features can be removed from the dataset without incurring a considerable loss in performance?

Another test was performed by reducing the total number of features to just seven features: difference in average player rating, TrueSkill win probability and both teams' μ-values, the difference in their HLTV world ranking, the first team's HLTV

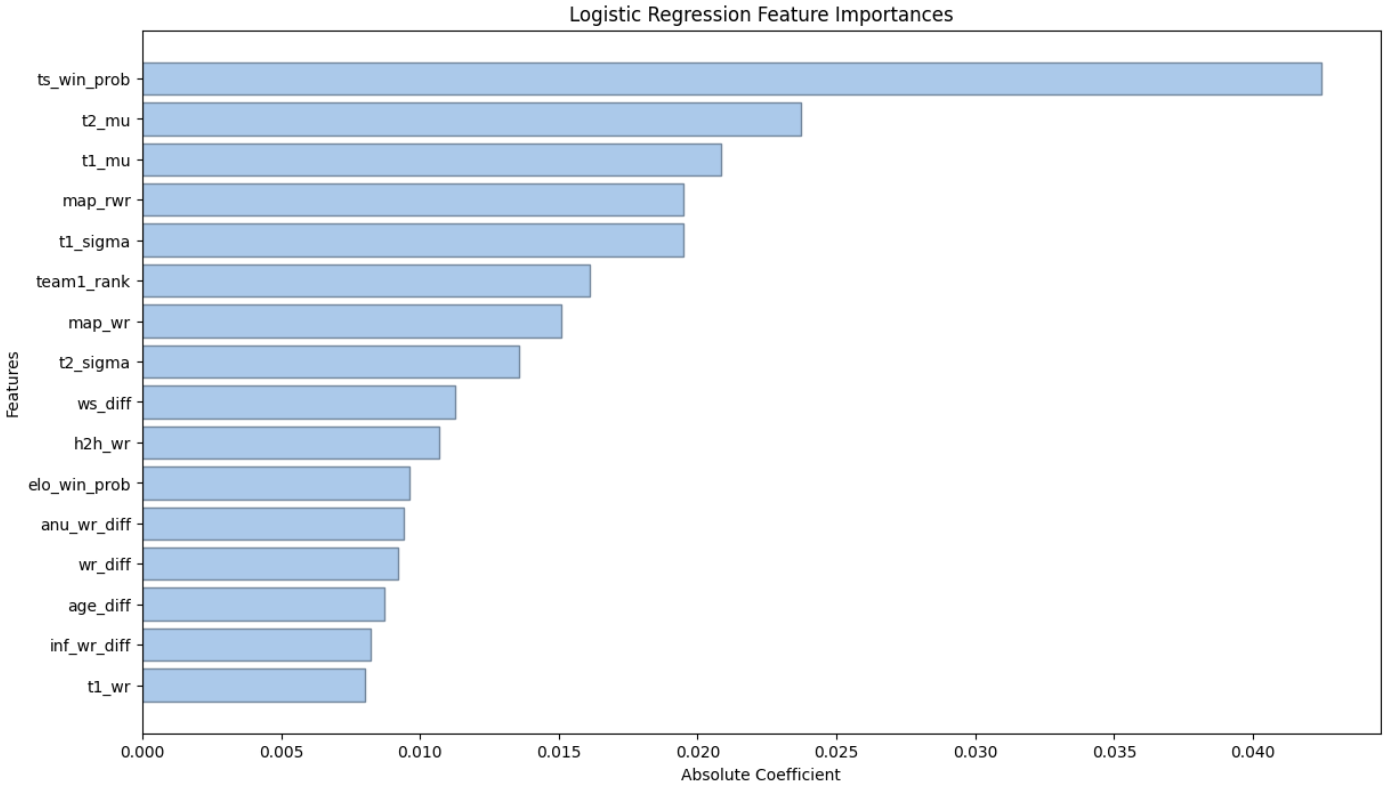


FIGURE 4.3: Feature importance from logistic regression coefficients

Model	**Train Acc.**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
TrueSkill	56.8	57.9	57.2	62.4	59.7	61.6
Logistic regression	58.4	59.1	57.7	67.7	62.3	62.2
Random forests	63.3	59.1	57.7	68.3	62.5	63.3
Support vector machine	59.7	57.4	55.6	**74.0**	**63.5**	62.8
k-nearest neighbours	63.9	53.9	53.4	60.6	56.7	55.8
Gaussian Naive Bayes	58.3	57.9	56.9	65.2	60.8	61.0
Multilayer perceptron	59.8	**59.8**	**58.6**	66.4	62.3	63.7
XGBoost	60.0	59.3	57.9	68.6	62.8	**64.2**
Mean Change	+0.45	-0.12	-0.59	+3.74	+1.54	+0.23

TABLE 4.3: Performance of ML models trained with 18 selected features on the complete dataset

world ranking, and the number of maps where the first team had a higher round win-rate than their opponent. The results are shown below in Table 4.4.

When trained with fewer features, the mean classification accuracy across the seven models increased. The best accuracy was achieved by XGBoost once more, although it degraded slightly from 60.0% in the full model to 59.5% in the seven-feature model. This result suggests that many of the models were over-fitting the training data to some degree, and secondly that the addition of over a hundred features yields only a minor improvement in prediction accuracy.

These results are graphically summarised in Figure 4.4.

Model	Train Acc.	Accuracy	Precision	Recall	F1 Score	ROC AUC
TrueSkill	56.8	57.9	57.2	62.4	59.7	61.6
Logistic regression	57.7	58.8	58.0	63.6	60.7	61.1
Random forests	62.1	59.0	57.6	**67.9**	62.3	62.3
Support vector machine	57.5	58.7	57.7	65.1	61.2	61.8
k-nearest neighbours	59.3	57.7	56.9	63.3	59.9	60.8
Gaussian Naive Bayes	58.3	57.9	56.9	65.2	60.8	61.0
Multilayer perceptron	57.1	56.9	57.0	56.5	56.7	58.4
XGBoost	59.2	**59.5**	**58.3**	66.6	**62.2**	**62.7**
Mean Change	-1.29	+0.14	+0.04	+0.51	+0.53	-0.47

TABLE 4.4: Performance of ML models trained with 7 selected features on the complete dataset

4.1.2 Filtered datasets

Excluding BO1 matches

A common conception in the Counter-Strike community is that best-of-1 matches are more unpredictable and upsets are more likely. This is a reasonable expectation considering that fewer round wins are required to win the match, which can lead to more volatile outcomes. One would therefore expect an improvement in predictive accuracy when ignoring BO1 matches.

The dataset used for this analysis consisted of 8320 matches in total split using the same 80:20 ratio. 6656 matches were used to train the models, and their performance was evaluated by testing them on 1664 matches. The 18 selected features were determined using the feature importance graphs obtained earlier.

Model	Train Acc.	Accuracy	Precision	Recall	F1 Score	ROC AUC
TrueSkill	58.8	59.2	59.3	62.3	60.8	63.7
Logistic regression	61.3	61.7	60.6	69.9	64.9	65.6
Random forests	65.5	61.3	60.1	**70.6**	64.9	66.0
Support vector machine	61.3	61.7	**60.8**	68.8	64.6	65.5
k-nearest neighbours	61.2	60.9	59.9	69.2	64.2	65.7
Gaussian Naive Bayes	60.4	59.4	58.5	68.2	63.0	63.3
Multilayer perceptron	62.1	61.1	60.2	68.8	64.2	65.9
XGBoost	63.7	**61.9**	**60.8**	69.9	**65.0**	**66.3**

TABLE 4.5: Performance of ML models trained with all features on dataset which excludes BO1 matches

The baseline TrueSkill model's classification accuracy improved slightly from 57.9% to 59.2%, a modest 1.3% improvement. Almost all of models achieved over 60% accuracy. XGBoost once again achieved the best performance, followed by SVM and logistic regression. These results empirically support the notion that Best-of-1's are more difficult to predict.

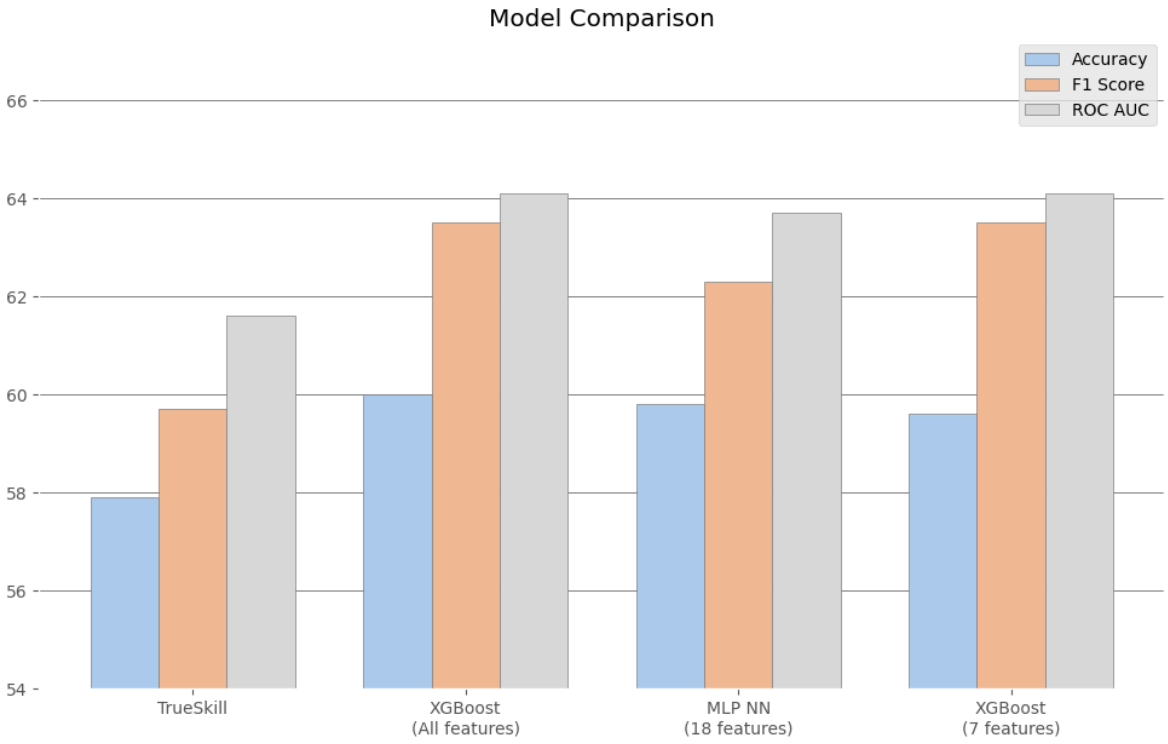


FIGURE 4.4: Comparison of best accuracy, F1-score, and AUC ROC attained for models trained with different amounts of features

LAN-only

LAN tournaments are often prestigious events, and it is usually only the top teams who play in these matches. It is thus useful to understand the classification performance for these types of matches. Applying this filter reduces the number of matches significantly. Furthermore, in 2020 and a large part of 2021, almost no LAN matches were played due to the COVID-19 pandemic. The dataset therefore contains only 2130 matches, of which 1704 were used for training and 426 for testing. Note that these include all match formats, including BO1.

Model	Train Acc.	Accuracy	Precision	Recall	F1 Score	ROC AUC
TrueSkill	54.8	59.9	59.0	64.8	61.7	62.8
Logistic regression	58.1	60.1	59.6	62.4	61.0	63.5
Random forests	63.8	62.4	**61.4**	67.1	64.1	63.9
Support vector machine	58.0	**62.7**	61.1	**70.0**	**65.2**	**64.2**
k-nearest neighbours	59.2	58.0	56.9	65.7	61.0	62.8
Gaussian Naive Bayes	57.0	59.9	59.5	61.5	60.5	61.7
Multilayer perceptron	57.3	59.2	58.3	64.3	61.2	61.1
XGBoost	67.2	60.3	59.7	63.4	61.5	61.8

TABLE 4.6: Performance of ML models trained with selected features on dataset with only LAN matches included

SVM performed the best under these circumstances followed by random forests. XGBoost and the MLP did not perform as well as before. This was the first time an ML model achieved an accuracy in excess of 62%. This dataset was, however, far

smaller than the prior datsets. The best model achieved a 2.8% improvement over the baseline TrueSkill model, which was more significant than previously obtained. These results empirically support the idea that LAN matches are more predictable than online matches.

Rating cut-off

As discussed previously, the full dataset includes all matches played where at least one of the team's was ranked in the Top-50 on HLTV's world ranking. Does varying this constraint meaningfully impact the overall predictability of matches? To test this hypothesis, the rank threshold was adjusted to only consider matches played where at least one of the teams were ranked in the Top-30 and Top-15 respectively. The results are tabulated in Tables 4.7 and 4.8 below.

Top-30

This filter removed a sizeable number of matches from the dataset, leaving 7115 for analysis. 5692 were used for training the models and 1423 for testing.

Model	Train Acc.	Accuracy	Precision	Recall	F1 Score	ROC AUC
TrueSkill	57.4	58.0	57.8	62.4	60.0	61.5
Logistic regression	60.4	59.3	57.6	73.7	**64.6**	63.3
Random forests	62.9	58.1	56.6	**73.0**	63.7	61.7
Support vector machine	61.3	58.6	57.7	67.0	62.0	62.8
k-nearest neighbours	64.7	52.8	53.0	58.4	55.5	54.2
Gaussian Naive Bayes	58.2	57.2	57.8	56.3	57.0	60.8
Multilayer perceptron	61.7	59.0	**60.7**	53.3	56.8	62.0
XGBoost	62.9	**59.5**	58.0	71.6	64.1	**64.3**

TABLE 4.7: Performance of ML models trained with selected features on dataset consisting only of Top-30 matches

The results obtained were very similar to the full dataset, with the baseline benchmark only improving its accuracy by 0.1%. XGBoost and MLP once again proved to be the best performing models, which was expected given the earlier results. That being said, the best accuracy attained actually decreased slightly, from 59.8% to 59.5%.

Top-15

The ranking threshold was further constrained to only include matches with at least one Top-15 team. This dataset had 3756 matches, of which 3004 were used for training and 752 for testing.

The results here mirror what was found with the LAN-only filter. Indeed, there is significant overlap between these two datasets, as teams which attend LAN events also tend to be higher-ranked. XGBoost obtained the highest classification ability once more, 62%. It is noteworthy that each model (except Gaussian Naive Bayes) exhibited significantly better recall than precision. This means that the models tend towards positive classifications: true positives were often identified, however, more false positives also occurred.

Model	Train Acc.	Accuracy	Precision	Recall	F1 Score	ROC AUC
TrueSkill	58.2	58.5	59.3	62.6	60.9	61.6
Logistic regression	62.5	57.6	56.9	73.5	64.1	62.3
Random forests	61.8	58.6	57.1	**79.9**	**66.6**	62.4
Support vector machine	63.3	58.1	57.0	77.1	65.5	63.7
k-nearest neighbours	66.3	51.6	52.7	61.3	56.7	54.0
Gaussian Naive Bayes	59.0	58.4	**61.9**	50.3	55.5	62.1
Multilayer perceptron	64.4	60.2	59.5	71.9	65.1	61.9
XGBoost	66.5	**62.0**	61.8	69.1	65.2	**65.8**

TABLE 4.8: Performance of ML models trained with selected features on dataset consisting only of Top-15 matches

One explanation for the improvement in model performance is that there is a greater proportion of mis-matches (Top-15 vs a lower ranked team) present in this subset of the data, which should be easier to predict.

4.1.3 Summary

Machine learning techniques outperform the baseline classification performance, albeit usually only by a few percentage points. XGBoost was found to consistently achieve better performance than the other methods employed, notably obtaining 62% accuracy on the dataset which excluded BO1s. SVM and MLP also delivered good performance, sometimes beating XGBoost. In the LAN-only subset, SVM achieved 62.7% accuracy.

The results suggest that LAN matches are more predictable than online matches, and that matches involving a higher-ranked team are generally easier to predict than a match with lower ranked teams. Additionally, BO3 and BO5 matches are easier to predict than BO1 matches.

Furthermore, it was found that incorporating many features did not measurably improve the predictive power of the models. Key features could be identified and selected using filter and wrapper techniques, but also from feature importance metrics produced by the models themselves. Training using a smaller subset of only the most relevant features reduced the tendency for models to overfit the training data while simultaneously reducing the computational requirements. These smaller models produced almost identical performance to the fully-featured models.

4.2 Betting odds and simulation

The odds generated by the various models all exhibited a high correlation with the bookmaker odds, as shown in Table 4.9. As explained in 3.4.6, the correlation, mean absolute error (MAE), and root-mean squared error (RMSE) for each model is reported. These measure the discrepancy between real odds offered by bookmakers and the odds generated by the models. Bet % refers to the proportion of matches which were deemed to be worth betting on. The P/L column is the total number of betting units won or lost by the end of the betting period.

Model	**Correlation**	**MAE**	**RMSE**	**Bet %**	**Win %**	**Loss %**	**P/L**
TrueSkill	0.727	0.104	0.137	79.02	45.45	54.55	-1.77
Logistic regression	0.815	0.099	0.121	82.76	39.58	60.42	27.80
Random forests	0.815	0.101	0.124	81.61	39.08	60.92	25.98
Support vector machine	0.803	0.100	0.123	82.47	39.02	60.98	23.72
k-nearest neighbours	0.740	0.105	0.130	81.90	40.70	59.30	**32.78**
Gaussian Naive Bayes	0.774	0.102	0.135	80.17	**51.25**	**48.75**	15.27
Multilayer perceptron	0.801	0.097	0.120	**84.20**	38.23	61.77	16.70
XGBoost	**0.855**	**0.088**	**0.108**	81.90	38.60	61.40	21.38

TABLE 4.9: Comparison of generated odds to real-world betting odds and per-model profit/loss

The betting performance is of particular interest, as every ML model was profitable despite having a low percentage of winning bets relative to their classification accuracy. This counter-intuitive result is explained by the models assigning a higher probability (but still less than 50%) to the underdog than what was implied by the bookmaker odds. This behaviour is evident by the number of underdog bets specified in Table 4.10.

Bet Type	Bets	Wins	Losses	Profit
Underdog	240	89	151	29.54
Overdog	39	24	15	3.15
No Bet	68	0	0	0.00

TABLE 4.10: Summary of *k*-NN betting results

The models effectively preferred to bet on the teams they predicted to lose. The large payouts from winning underdog bets offset the more frequent losses, resulting in an overall profitable strategy.

Every model considered over 80% of the matches as worth betting on, with bookmakers offering favourable odds; odds for one of the teams being greater than what the model generated. The MLP produced the most avid betting activity, with 84% of matches being bet on.

XGBoost, the model which repeatedly produced the most accurate classifications, shared the greatest overlap with the real-world odds, with a correlation of 0.855, MAE of 0.088 and RMSE of 0.108.

The most profitable model, *k*-NN, disagreed with the bookmaker odds the most; it had the lowest correlation of 0.740 and highest errors of all the ML models.

The cumulative profit/loss for *k*-NN is plotted below. Figure 4.5 shows the profit/loss on a match-by-match basis. Figure 4.6 shows the profit/loss over time. Note that the horizontal lines indicate periods in which no matches occurred.

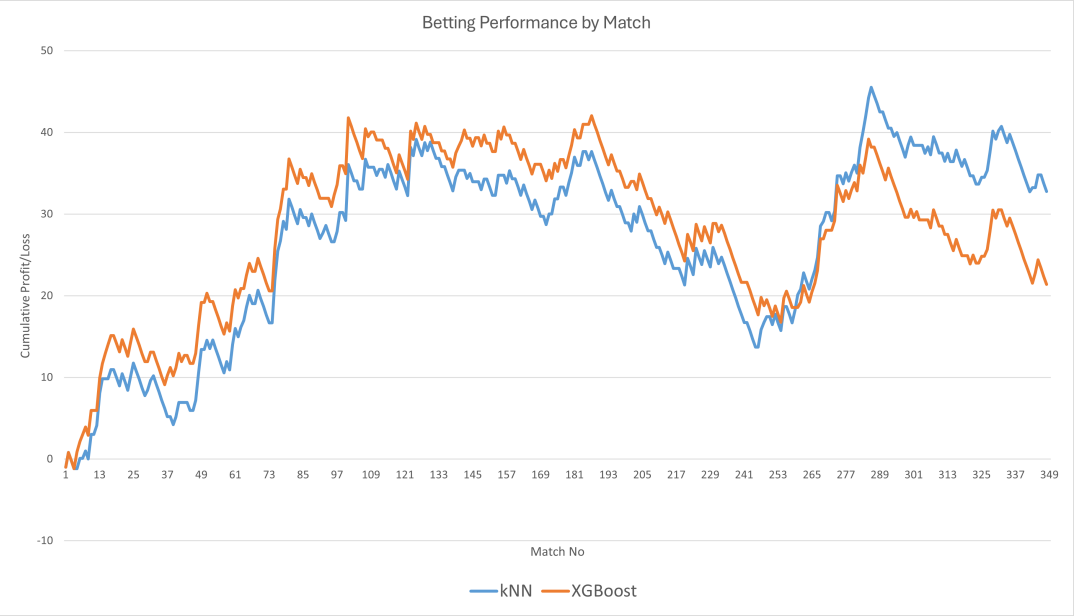


FIGURE 4.5: Cumulative profit/loss for *k*-NN and XGBoost (by match)

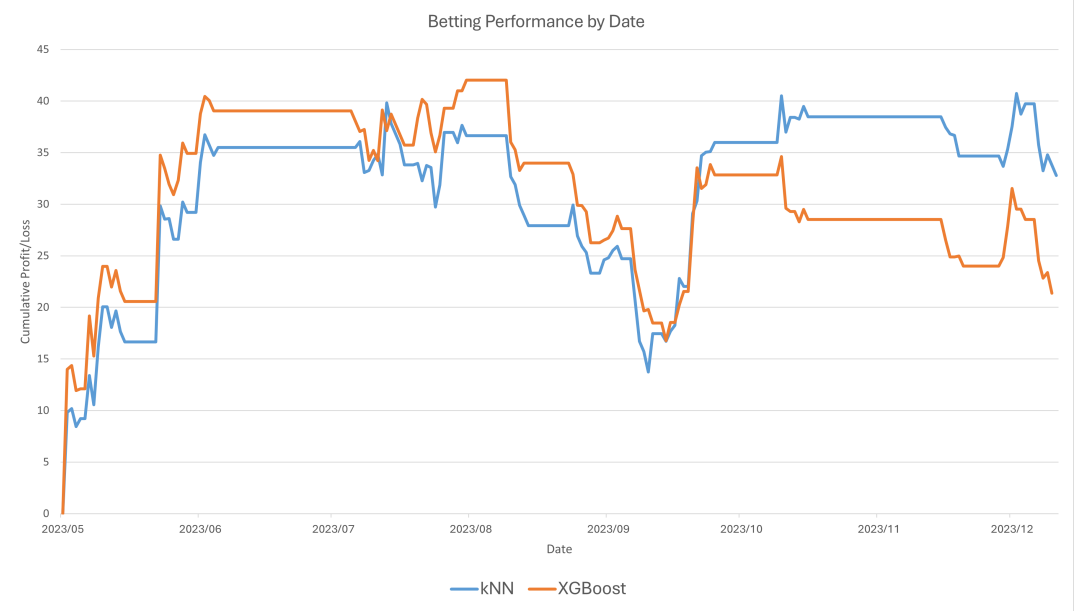


FIGURE 4.6: Cumulative profit/loss for *k*-NN and XGBoost (by date)

It is interesting to note that the model which deviated most from the bookmaker odds was the most profitable. *k*-NN only shared a 74% correlation with the bookmaker odds. It should also be noted that XGBoost outperformed *k*-NN until the 260th match played, as shown by the intersection of the graphs in Figure 4.5. This suggests that short-term results or fluctuations may not be indicative of a model's true performance, and a longer betting period may lead to different outcomes. The raw betting data for the *k*-NN model is included in Appendix A.

5 Conclusion

5.1 Key Findings

Professional Counter-Strike matches are difficult to predict. Incorporating more than a hundred statistics into a machine learning model only yields a 0.5% improvement over a model with just seven of the most important features. The best accuracy attained for match prediction was 62.7% which was achieved from the dataset consisting of only matches played in a LAN environment. For matches played in a BO3 or BO5 maps format, 61.9% of the match outcomes could be correctly predicted. When considering the full dataset of matches played in any format, both online or at LAN, and by at least one Top-50 ranked team, the best accuracy decreased to 59.9%.

The machine learning models which produced the best results were XGBoost configured to have a large number of estimators with a low maximum tree depth, MLPs with a small number of hidden layers and nodes per layer, and SVMs with a linear kernel. The performance attained by these models were a 2-3% improvement over the base model, depending on the measure (classification accuracy, F_1-score, and AUC ROC).

The base model was a rating system known as TrueSkill, where each player or team's strength is modelled as a Gaussian distribuion. Player TrueSkill ratings are adjusted after every match played. These ratings alone provide good predictions of match outcomes without the need for machine learning techniques, and performed better than both HLTV's world ranking and an Elo rating system. The TrueSkill-related features were consistently ranked as most important by the feature selection techniques employed.

The betting simulation comprised 347 matches played in the second half of 2023. Every model found at least 79% of the matches were worth betting on. In these cases, the win-probability generated was higher than what the bookmaker odds implied for one of the teams. A 1 unit bet was subsequently placed on the undervalued team.

This simple betting strategy proved successful to varying degrees for each of the machine learning models. Although all were profitable, the ML model which generated the most profit was $k - NN$. It returned 32.78 units over the seven-month betting period, which translates to 0.11 units per match. A more advanced betting strategy may be able to leverage the ML-generated predictions to an even greater extent. Varying the period and number of matches bet on will likely yield a different outcome.

These results prove that ML techniques can be used to generate effective betting odds for professional Counter-Strike matches. Furthermore, they may even outperform the odds offered by bookmakers, as demonstrated by the simple betting strategy simulated during the aforementioned betting period.

5.2 Significance of the study

The key findings of this research bears significance to a number of stakeholders:

- Betting Industry
  Both **bookmakers** and **bettors** can benefit from this research; bookmakers can improve the accuracy of their odd-setting, while bettors can potentially improve their profitability by implementing the insights into their own betting strategies. This research could contribute to bettors making more informed decisions, thereby promoting responsible betting.
- Viewer Experience
  Commentators use statistics to support their narratives. **Broadcasters** can enhance their streams by incorporating data-driven insights, potentially improving viewer engagement. **Counter-Strike fans** may find entertainment value from a deeper understanding of match predictability.
- Tournament Organizers
  Predictive information can assist **tournament organizers** to structure their tournaments and schedules in order to maximize the competitive fairness of their tournaments. They could also be used to design fair qualification systems or choose which teams to invite.
- Academia
  This research contributes to the understanding of esports betting and predictive modelling of professional Counter-Strike. **Future researchers** may find it to be a valuable resource for further study into these topics.
- Esports Community
  Professional teams (**players**, **coaches**, and **analysts**) may find the outputs from the statistical modelling valuable.

5.3 Limitations

Odds compilation is a private industry. The exact mechanisms used by bookmakers to price odds are not publicised. This makes it difficult to ascertain the extent to which statistical models influence the odds. Other factors, such as bookmaker risk management strategies and user betting behaviour, may also be used to generate the odds.

The data used to train the models was limited to match information and statistics. External factors such as player injuries, team morale, or technical issues were therefore unaccounted for. There is inherent unpredictability to any sport; the accuracy of predictive models are thus subject to the quality of the available data and the modelling techniques employed.

The feature engineering, although extensive, was not comprehensive. The historical features were generated using a three-month sliding window of a team's past performance. For newly formed line-ups, this data is not as informative as established teams. This could have been mitigated by considering a team "core" consisting of 3 or 4 players. Furthermore, 3 months may be too large or too small a window; varying this time-window was not explored. Incorporating features that more accurately

describe the trends in a team's performance may lead to stronger predictions. An example of this was the 'winning streak' feature. Additional feature selection methods, as well as dimensionality reduction techniques, could have also been explored.

The list of machine learning techniques implemented was not exhaustive. For example, a recurrent neural network such as long short-term memory (LSTM) may be more appropriate for chronological data. Additionally, an ensemble model taking into account the outputs of multiple models may achieve better results than any individual model. These hypotheses were not tested.

The data segmentation method employed does not accurately reflect a real-world scenario, where all known match data would be used to train the model and the test would be a future fixture. An approach which better resembles real-world use would be to incrementally increase the training dataset, each time using all of the data for training, leaving only 1 match for testing. In the next iteration, this match would be added to the training set and the next match becomes the test set. This approach would be more repesentative of a real-world model, however at the cost of being significantly more computationally expensive.

Finally, for testing each model's betting performance, the betting strategy used was fairly basic. More complex strategies could have been explored, such as the use of a dynamic wager which depends on the odds differential, or perhaps combining the predictions from different models. The number of matches tested was also a fairly low amount (347) - more odds data could have been collected and tested.

5.4 Future work

Given the limitations described, future work could expand on the findings made here in a number of ways:

1. An investigation of how external factors, such as player injuries, team morale, and even social media sentiments, can influence match prediction
2. There is room for more sophisticated feature engineering techniques. As mentioned above, adjusting the sliding window of data to find the optimal period to balance recency and relevance, as well as including features that describe the team's performance trajectory or trends.
3. Investigating more advanced machine learning techniques, such as LSTMs, ensemble methods, or even transformer models. These techniques may be able to outperform the models used in this research.
4. Application of an iterative data segmentation method, which more closely mirrors how a model would be used in real-world applications.
5. Collaboration with bookmakers and odds compilers to understand the methods employed in the betting industry. This may help to foster transparency and innovation in the field.
6. The development and testing of more complex betting strategies on a larger dataset of matches would make the findings more robust.

A Raw betting data

The generated odds used in this table were generated using the *k*-nearest neighbours algorithm.

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
05/08	monte : faze	2.68 : 1.44	1.96 : 2.04	monte	L	-1	-1
05/08	pain : fluxo	1.43 : 2.80	1.80 : 2.24	fluxo	W	1.8	0.8
05/08	g2 : mongolz	1.10 : 6.44	1.53 : 2.89	mongolz	L	-1	-0.2
05/08	gamerl : col	1.75 : 2.06	1.43 : 3.33	gamerl	L	-1	-1.2
05/08	forze : grayhound	1.35 : 3.07	1.41 : 3.44			0	-1.2
05/08	apeks : liquid	2.27 : 1.60	2.16 : 1.86	apeks	W	1.27	0.07
05/08	nip : mouz	1.90 : 1.85	2.16 : 1.86			0	0.07
05/08	og : ence	1.93 : 1.82	1.93 : 2.08	og	W	0.93	1
05/08	faze : pain	1.34 : 3.16	1.55 : 2.82	pain	L	-1	0
05/08	col : g2	4.03 : 1.22	2.75 : 1.57	col	W	3.03	3.03
05/08	apeks : grayhound	1.38 : 2.94	1.47 : 3.14			0	3.03
05/08	ence : nip	1.75 : 2.07	2.04 : 1.96	nip	W	1.07	4.1
05/08	fluxo : monte	4.97 : 1.17	2.97 : 1.51	fluxo	W	3.97	8.07
05/08	gamerl : mongolz	1.42 : 2.75	1.64 : 2.56	mongolz	W	1.75	9.82
05/08	forze : liquid	2.27 : 1.59	2.29 : 1.77			0	9.82
05/08	mouz : og	1.73 : 2.05	1.77 : 2.29			0	9.82
05/09	col : pain	2.14 : 1.68	1.83 : 2.20	col	W	1.14	10.96
05/09	grayhound : mongolz	1.91 : 1.85	2.00 : 2.00			0	10.96
05/09	forze : monte	2.35 : 1.56	2.12 : 1.90	forze	L	-1	9.96
05/09	nip : og	1.75 : 2.01	2.34 : 1.75	og	L	-1	8.96
05/09	ence : faze	2.47 : 1.52	2.00 : 2.00	ence	W	1.47	10.43
05/09	apeks : g2	3.83 : 1.24	2.08 : 1.93	apeks	L	-1	9.43
05/09	fluxo : liquid	5.85 : 1.12	2.82 : 1.55	fluxo	L	-1	8.43
05/09	gamerl : mouz	2.75 : 1.43	2.20 : 1.83	gamerl	W	1.75	10.18
05/10	gamerl : og	2.57 : 1.49	2.24 : 1.80	gamerl	W	1.57	11.75
05/10	mongolz : monte	4.55 : 1.18	2.34 : 1.75	mongolz	L	-1	10.75
05/10	col : liquid	2.74 : 1.42	2.12 : 1.90	col	L	-1	9.75
05/10	faze : forze	1.39 : 3.04	1.69 : 2.44	forze	L	-1	8.75
05/10	grayhound : nip	4.47 : 1.17	2.34 : 1.75	grayhound	L	-1	7.75
05/10	apeks : pain	1.68 : 2.12	1.64 : 2.56	apeks	W	0.68	8.43
05/11	forze : gamerl	1.65 : 2.18	2.04 : 1.96	gamerl	W	1.18	9.61
05/11	monte : pain	1.59 : 2.33	1.57 : 2.75	monte	W	0.59	10.2
05/11	grayhound : liquid	4.78 : 1.17	2.75 : 1.57	grayhound	L	-1	9.2
05/13	navi : gamerl	1.32 : 3.29	1.51 : 2.97	gamerl	L	-1	8.2
05/13	9ine : liquid	1.83 : 1.92	2.44 : 1.69	liquid	L	-1	7.2
05/13	furia : monte	1.66 : 2.15	2.12 : 1.90	monte	L	-1	6.2
05/13	fnatic : nip	2.13 : 1.67	2.50 : 1.67	nip	L	-1	5.2
05/13	heroic : faze	1.78 : 1.97	1.83 : 2.20			0	5.2
05/13	itb : apeks	2.45 : 1.52	2.16 : 1.86	itb	L	-1	4.2
05/13	vitality : g2	1.96 : 1.79	1.93 : 2.08	vitality	W	0.96	5.16
05/13	bne : ence	2.78 : 1.42	1.86 : 2.16	bne	W	1.78	6.94
05/13	liquid : navi	2.56 : 1.46	2.68 : 1.59			0	6.94

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
05/13	fnatic : monte	2.00 : 1.76	2.24 : 1.80			0	6.94
05/13	vitality : ence	1.54 : 2.39	1.62 : 2.62			0	6.94
05/13	heroic : apeks	1.34 : 3.13	1.53 : 2.89	apeks	L	-1	5.94
05/13	furia : nip	1.79 : 1.97	1.93 : 2.08			0	5.94
05/13	9ine : gamerl	1.62 : 2.23	2.12 : 1.90	gamerl	W	1.23	7.17
05/13	bne : g2	4.44 : 1.19	2.04 : 1.96	bne	W	3.44	10.61
05/13	faze : itb	1.25 : 3.79	1.75 : 2.34	itb	W	2.79	13.4
05/14	navi : nip	1.51 : 2.49	1.59 : 2.68			0	13.4
05/14	fnatic : gamerl	1.68 : 2.13	2.34 : 1.75	gamerl	W	1.13	14.53
05/14	ence : itb	1.30 : 3.36	1.64 : 2.56	itb	L	-1	13.53
05/14	apeks : bne	1.74 : 2.03	2.00 : 2.00	bne	W	1.03	14.56
05/14	heroic : liquid	1.39 : 2.87	1.90 : 2.12	liquid	L	-1	13.56
05/14	monte : vitality	3.21 : 1.33	2.20 : 1.83	monte	L	-1	12.56
05/14	furia : g2	3.31 : 1.33	1.83 : 2.20	furia	L	-1	11.56
05/14	9ine : faze	2.99 : 1.37	2.12 : 1.90	9ine	L	-1	10.56
05/15	ence : nip	1.55 : 2.37	1.96 : 2.04	nip	W	1.37	11.93
05/15	faze : bne	1.24 : 3.87	1.90 : 2.12	bne	L	-1	10.93
05/15	fnatic : g2	4.10 : 1.22	2.34 : 1.75	fnatic	W	3.1	14.03
05/15	monte : navi	2.96 : 1.38	1.86 : 2.16	monte	W	1.96	15.99
05/15	liquid : itb	1.31 : 3.29	1.62 : 2.62	itb	L	-1	14.99
05/15	apeks : gamerl	1.66 : 2.16	1.86 : 2.16	gamerl	W	1.16	16.15
05/16	faze : navi	1.81 : 1.94	1.67 : 2.50	faze	W	0.81	16.96
05/16	fnatic : itb	1.45 : 2.64	1.72 : 2.39	itb	W	1.64	18.6
05/16	apeks : nip	2.45 : 1.52	2.12 : 1.90	apeks	W	1.45	20.05
05/18	heroic : faze	1.61 : 2.24	1.90 : 2.12	faze	L	-1	19.05
05/19	monte : gamerl	1.53 : 2.42	1.69 : 2.44			0	19.05
05/19	liquid : apeks	1.48 : 2.61	1.80 : 2.24	apeks	W	1.61	20.66
05/18	vitality : itb	1.12 : 5.85	1.55 : 2.82	itb	L	-1	19.66
05/20	heroic : gamerl	3.76 : 1.25	1.69 : 2.44	heroic	L	-1	18.66
05/20	vitality : apeks	1.23 : 3.96	1.62 : 2.62	apeks	L	-1	17.66
05/21	gamerl : vitality	3.30 : 1.33	2.29 : 1.77	gamerl	L	-1	16.66
05/29	furia : og	1.75 : 2.01	1.77 : 2.29			0	16.66
05/29	g2 : nouns	1.11 : 6.18	1.64 : 2.56	nouns	W	5.18	21.84
05/29	heroic : 9z	1.17 : 4.61	1.67 : 2.50	9z	W	3.61	25.45
05/29	fnatic : mouz	2.25 : 1.61	2.16 : 1.86	fnatic	W	1.25	26.7
05/29	ence : col	1.29 : 3.42	1.62 : 2.62	col	W	2.42	29.12
05/29	liquid : astralis	1.87 : 1.87	1.86 : 2.16	liquid	L	-1	28.12
05/29	cloud9 : grayhound	1.17 : 4.71	1.59 : 2.68	grayhound	W	3.71	31.83
05/29	g2 : og	1.26 : 3.74	1.59 : 2.68	og	L	-1	30.83
05/29	furia : nouns	1.16 : 5.10	1.64 : 2.56	nouns	L	-1	29.83
05/30	mouz : heroic	3.15 : 1.34	2.24 : 1.80	mouz	L	-1	28.83
05/30	fnatic : 9z	1.45 : 2.73	2.04 : 1.96	9z	W	1.73	30.56
05/30	liquid : grayhound	1.12 : 5.81	1.55 : 2.82	grayhound	L	-1	29.56
05/30	faze : ence	1.66 : 2.16	1.77 : 2.29			0	29.56
05/31	og : 9z	1.44 : 2.68	2.04 : 1.96	9z	L	-1	28.56
05/31	cloud9 : astralis	1.52 : 2.45	1.83 : 2.20	astralis	W	1.45	30.01
05/30	furia : mouz	1.87 : 1.87	1.83 : 2.20	furia	L	-1	29.01
05/31	col : cloud9	5.35 : 1.14	2.56 : 1.64	col	L	-1	28.01
05/31	faze : liquid	1.60 : 2.27	1.96 : 2.04	liquid	L	-1	27.01
05/31	mouz : og	1.69 : 2.12	1.67 : 2.50	mouz	W	0.69	27.7
05/31	g2 : heroic	1.84 : 1.91	2.24 : 1.80	heroic	W	0.91	28.61
06/01	ence : astralis	2.01 : 1.76	2.39 : 1.72	astralis	L	-1	27.61
06/01	faze : cloud9	1.75 : 2.03	2.04 : 1.96	cloud9	L	-1	26.61
06/02	astralis : mouz	1.54 : 2.40	1.62 : 2.62			0	26.61

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
06/03	g2 : faze	1.64 : 2.25	2.29 : 1.77	faze	W	1.25	27.86
06/03	heroic : mouz	1.31 : 3.34	1.64 : 2.56	mouz	W	2.34	30.2
06/03	ence : faze	2.21 : 1.65	2.29 : 1.77			0	30.2
06/04	mouz : ence	2.12 : 1.68	1.83 : 2.20	mouz	L	-1	29.2
06/07	vitality : imperial	1.08 : 7.85	1.47 : 3.14	imperial	W	6.85	36.05
06/07	g2 : cloud9	1.60 : 2.26	2.34 : 1.75	cloud9	L	-1	35.05
06/07	heroic : col	1.10 : 6.51	1.47 : 3.14	col	L	-1	34.05
06/07	faze : astralis	1.63 : 2.29	1.75 : 2.34			0	34.05
06/08	imperial : g2	4.90 : 1.16	2.50 : 1.67	imperial	L	-1	33.05
06/08	vitality : cloud9	1.54 : 2.39	1.69 : 2.44			0	33.05
06/08	col : astralis	4.68 : 1.19	2.34 : 1.75	col	W	3.68	36.73
06/09	heroic : faze	1.70 : 2.12	1.90 : 2.12	faze	L	-1	35.73
06/09	imperial : col	1.61 : 2.24	1.62 : 2.62			0	35.73
06/09	vitality : faze	1.70 : 2.13	1.72 : 2.39			0	35.73
06/10	heroic : imperial	1.19 : 4.47	1.55 : 2.82	imperial	L	-1	34.73
06/11	g2 : vitality	2.01 : 1.76	2.75 : 1.57	vitality	W	0.76	35.49
06/11	heroic : vitality	2.06 : 1.72	2.29 : 1.77			0	35.49
07/12	vitality : eg	1.08 : 7.38	1.38 : 3.67	eg	L	-1	34.49
07/13	nip : col	1.48 : 2.58	1.64 : 2.56	col	W	1.58	36.07
07/13	heroic : big	1.11 : 6.12	1.77 : 2.29	big	L	-1	35.07
07/14	navi : astralis	1.51 : 2.55	2.16 : 1.86	astralis	L	-1	34.07
07/14	faze : og	1.18 : 4.60	1.55 : 2.82	og	L	-1	33.07
07/14	g2 : liquid	1.33 : 3.19	1.96 : 2.04	liquid	W	2.19	35.26
07/15	vitality : col	1.15 : 5.12	1.41 : 3.44	col	L	-1	34.26
07/15	eg : nip	4.54 : 1.18	3.24 : 1.45	eg	L	-1	33.26
07/15	col : nip	2.23 : 1.62	2.00 : 2.00	col	L	-1	32.26
07/18	vitality : nip	1.12 : 6.90	1.57 : 2.75	nip	W	5.9	38.16
07/19	heroic : navi	1.40 : 2.87	1.86 : 2.16	navi	L	-1	37.16
07/16	big : astralis	2.98 : 1.39	2.39 : 1.72	big	W	1.98	39.14
07/16	navi : big	1.31 : 3.27	1.53 : 2.89	big	L	-1	38.14
07/18	heroic : navi	1.30 : 3.35	1.72 : 2.39	navi	L	-1	37.14
07/20	faze : liquid	1.46 : 2.61	1.93 : 2.08	liquid	W	1.61	38.75
07/17	og : g2	4.12 : 1.22	2.75 : 1.57	og	L	-1	37.75
07/17	faze : g2	2.07 : 1.71	1.83 : 2.20	faze	W	1.07	38.82
07/19	liquid : faze	2.51 : 1.53	2.44 : 1.69	liquid	L	-1	37.82
07/20	astralis : col	1.59 : 2.27	1.90 : 2.12	col	L	-1	36.82
07/21	og : big	2.01 : 1.75	2.29 : 1.77			0	36.82
07/21	eg : g2	8.20 : 1.06	3.24 : 1.45	eg	L	-1	35.82
07/22	liquid : astralis	1.67 : 2.17	1.77 : 2.29			0	35.82
07/22	vitality : big	1.11 : 6.17	1.49 : 3.06	big	L	-1	34.82
07/23	navi : g2	2.07 : 1.70	2.56 : 1.64	g2	L	-1	33.82
07/23	mouz : mongolz	1.14 : 5.35	1.55 : 2.82	mongolz	L	-1	32.82
07/26	itb : nip	2.72 : 1.43	2.16 : 1.86	itb	W	1.72	34.54
07/26	fnatic : col	1.83 : 1.92	1.64 : 2.56	fnatic	W	0.83	35.37
07/26	og : 9ine	2.05 : 1.71	2.29 : 1.77			0	35.37
07/26	monte : imperial	1.61 : 2.29	1.64 : 2.56			0	35.37
07/26	astralis : liquid	1.79 : 1.95	2.16 : 1.86	liquid	L	-1	34.37
07/26	apeks : big	1.60 : 2.28	1.55 : 2.82	apeks	W	0.6	34.97
07/26	grayhound : furia	3.56 : 1.26	2.20 : 1.83	grayhound	L	-1	33.97
07/26	nip : mouz	1.72 : 2.07	1.83 : 2.20			0	33.97
07/26	fnatic : 9ine	2.00 : 1.78	2.16 : 1.86			0	33.97
07/26	astralis : apeks	1.52 : 2.47	1.75 : 2.34	apeks	L	-1	32.97
07/27	monte : furia	2.31 : 1.57	2.20 : 1.83	monte	W	1.31	34.28
07/27	itb : mongolz	1.56 : 2.36	1.57 : 2.75			0	34.28

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
07/27	col : og	1.82 : 1.92	1.72 : 2.39	col	L	-1	33.28
07/27	liquid : big	1.28 : 3.70	1.72 : 2.39	big	L	-1	32.28
07/27	imperial : grayhound	1.53 : 2.42	1.62 : 2.62			0	32.28
07/27	mongolz : furia	3.49 : 1.28	2.39 : 1.72	mongolz	W	2.49	34.77
07/28	apeks : og	1.56 : 2.36	1.64 : 2.56			0	34.77
07/28	9ine : liquid	2.51 : 1.50	2.62 : 1.62			0	34.77
07/28	nip : imperial	1.41 : 2.77	2.20 : 1.83	imperial	L	-1	33.77
07/28	cloud9 : fnatic	1.48 : 2.54	1.80 : 2.24	fnatic	W	1.54	35.31
07/29	gamerl : monte	2.77 : 1.41	2.12 : 1.90	gamerl	L	-1	34.31
07/29	g2 : astralis	1.66 : 2.13	1.67 : 2.50			0	34.31
07/29	ence : 9ine	1.42 : 2.77	1.96 : 2.04	9ine	L	-1	33.31
07/29	heroic : mongolz	1.09 : 6.81	1.69 : 2.44	mongolz	L	-1	32.31
07/29	navi : mouz	1.61 : 2.24	2.00 : 2.00	mouz	W	1.24	33.55
07/29	vitality : og	1.15 : 5.10	1.51 : 2.97	og	L	-1	32.55
07/30	faze : nip	1.33 : 3.21	1.75 : 2.34	nip	L	-1	31.55
07/30	monte : heroic	3.34 : 1.29	3.06 : 1.49	monte	L	-1	30.55
07/30	gamerl : mongolz	2.17 : 1.65	1.75 : 2.34	gamerl	W	1.17	31.72
07/30	fnatic : ence	2.27 : 1.60	2.12 : 1.90	fnatic	L	-1	30.72
07/30	cloud9 : 9ine	1.52 : 2.43	1.93 : 2.08	9ine	L	-1	29.72
07/30	mouz : vitality	2.78 : 1.41	2.82 : 1.55			0	29.72
07/31	navi : og	1.36 : 3.04	1.51 : 2.97	og	L	-1	28.72
07/31	g2 : faze	2.28 : 1.59	2.20 : 1.83	g2	W	1.28	30
07/31	astralis : nip	1.61 : 2.25	1.77 : 2.29			0	30
07/31	fnatic : gamerl	1.40 : 2.86	2.00 : 2.00	gamerl	W	1.86	31.86
07/31	cloud9 : monte	1.66 : 2.16	1.72 : 2.39			0	31.86
07/31	faze : navi	1.52 : 2.45	1.86 : 2.16	navi	W	1.45	33.31
08/01	mouz : astralis	2.07 : 1.71	2.34 : 1.75			0	33.31
08/01	cloud9 : gamerl	1.27 : 3.56	1.64 : 2.56	gamerl	L	-1	32.31
08/01	vitality : g2	1.49 : 2.53	1.69 : 2.44	g2	W	1.53	33.84
08/01	astralis : navi	2.22 : 1.71	1.93 : 2.08	astralis	W	1.22	35.06
08/01	ence : heroic	2.90 : 1.42	2.89 : 1.53	ence	W	1.9	36.96
08/01	vitality : cloud9	1.51 : 2.46	1.77 : 2.29	cloud9	L	-1	35.96
08/04	heroic : astralis	1.53 : 2.45	1.57 : 2.75			0	35.96
08/04	ence : vitality	2.68 : 1.44	2.20 : 1.83	ence	W	1.68	37.64
08/05	g2 : astralis	1.71 : 2.07	1.83 : 2.20			0	37.64
08/05	ence : g2	2.13 : 1.67	2.00 : 2.00	ence	L	-1	36.64
08/06	apeks : gamerl	1.74 : 2.03	2.04 : 1.96	gamerl	W	1.03	37.67
08/16	cloud9 : fnatic	1.39 : 2.89	1.55 : 2.82	fnatic	L	-1	36.67
08/16	heroic : falcons	1.12 : 5.72	1.47 : 3.14	falcons	L	-1	35.67
08/16	mibr : vitality	7.30 : 1.07	2.62 : 1.62	mibr	L	-1	34.67
08/16	liquid : ence	2.49 : 1.50	1.96 : 2.04	liquid	L	-1	33.67
08/16	navi : furia	1.35 : 3.10	1.72 : 2.39	furia	L	-1	32.67
08/16	9ine : g2	4.19 : 1.20	2.08 : 1.93	9ine	L	-1	31.67
08/17	faze : virtuspro	1.64 : 2.24	2.08 : 1.93	virtuspro	W	1.24	32.91
08/17	vitality : navi	1.40 : 2.89	1.72 : 2.39	navi	L	-1	31.91
08/17	gamerl : heroic	4.23 : 1.20	2.56 : 1.64	gamerl	L	-1	30.91
08/18	cloud9 : ence	1.73 : 2.02	1.83 : 2.20			0	30.91
08/18	g2 : virtuspro	1.47 : 2.55	2.08 : 1.93	virtuspro	L	-1	29.91
08/18	heroic : ence	1.76 : 1.98	1.75 : 2.34	heroic	L	-1	28.91
08/19	vitality : g2	1.73 : 2.02	1.83 : 2.20			0	28.91
08/19	ence : vitality	2.05 : 1.71	1.83 : 2.20	ence	L	-1	27.91
08/20	furia : riders	1.34 : 3.10	1.62 : 2.62	riders	W	2.1	30.01
08/30	grayhound : gamerl	3.17 : 1.33	2.08 : 1.93	grayhound	L	-1	29.01
08/30	astralis : nip	1.38 : 2.92	1.86 : 2.16	nip	W	1.92	30.93

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
08/30	vitality : orks	1.04 : 10.41	1.53 : 2.89	orks	L	-1	29.93
08/30	furia : grayhound	1.22 : 4.05	1.69 : 2.44	grayhound	L	-1	28.93
08/31	orks : astralis	10.09 : 1.04	2.04 : 1.96	orks	L	-1	27.93
08/31	riders : gamerl	2.25 : 1.61	2.50 : 1.67			0	27.93
08/31	vitality : nip	1.20 : 4.29	1.38 : 3.67	nip	L	-1	26.93
08/31	gamerl : astralis	3.24 : 1.32	2.20 : 1.83	gamerl	L	-1	25.93
09/01	nip : furia	1.79 : 1.96	1.86 : 2.16			0	25.93
09/01	grayhound : orks	1.44 : 2.69	2.08 : 1.93	orks	L	-1	24.93
09/02	astralis : furia	1.43 : 2.68	1.69 : 2.44	furia	L	-1	23.93
09/02	gamerl : nip	2.40 : 1.54	2.24 : 1.80	gamerl	W	1.4	25.33
09/02	grayhound : furia	5.52 : 1.13	2.97 : 1.51	grayhound	L	-1	24.33
09/03	riders : vitality	7.26 : 1.08	2.97 : 1.51	riders	L	-1	23.33
09/03	gamerl : furia	2.38 : 1.55	2.44 : 1.69			0	23.33
09/03	big : monte	1.80 : 1.94	2.04 : 1.96			0	23.33
09/06	heroic : rooster	1.01 : 17.64	1.83 : 2.20	rooster	L	-1	22.33
09/06	eg : ence	8.48 : 1.06	2.56 : 1.64	eg	L	-1	21.33
09/06	mouz : mibr	1.20 : 4.26	1.41 : 3.44	mibr	W	3.26	24.59
09/06	rooster : monte	11.18 : 1.03	2.68 : 1.59	rooster	L	-1	23.59
09/07	ence : mibr	1.13 : 5.42	1.55 : 2.82	mibr	L	-1	22.59
09/07	heroic : big	1.21 : 4.21	1.62 : 2.62	big	W	3.21	25.8
09/07	eg : mouz	5.38 : 1.13	2.50 : 1.67	eg	L	-1	24.8
09/07	monte : mibr	1.80 : 1.95	1.75 : 2.34	monte	L	-1	23.8
09/08	heroic : mouz	1.42 : 2.72	1.93 : 2.08	mouz	W	1.72	25.52
09/08	rooster : eg	4.22 : 1.21	2.75 : 1.57	rooster	L	-1	24.52
09/09	mibr : mouz	3.01 : 1.37	2.89 : 1.53	mibr	L	-1	23.52
09/09	monte : heroic	3.39 : 1.30	2.75 : 1.57	monte	W	2.39	25.91
09/09	eg : mibr	2.62 : 1.46	2.34 : 1.75	eg	L	-1	24.91
09/10	big : ence	3.17 : 1.35	2.62 : 1.62	big	L	-1	23.91
09/10	monte : mibr	1.81 : 1.93	1.69 : 2.44	monte	W	0.81	24.72
09/10	navi : 5yclone	1.10 : 6.51	1.53 : 2.89	5yclone	L	-1	23.72
09/13	apeks : m80	1.28 : 3.53	1.93 : 2.08	m80	L	-1	22.72
09/13	faze : 9z	1.13 : 5.57	1.49 : 3.06	9z	L	-1	21.72
09/13	imperial : fnatic	2.43 : 1.52	2.00 : 2.00	imperial	L	-1	20.72
09/13	m80 : 5yclone	2.20 : 1.63	1.53 : 2.89	m80	L	-1	19.72
09/14	apeks : navi	3.00 : 1.36	2.08 : 1.93	apeks	L	-1	18.72
09/14	faze : fnatic	1.29 : 3.46	1.72 : 2.39	fnatic	L	-1	17.72
09/14	9z : imperial	2.26 : 1.60	1.93 : 2.08	9z	L	-1	16.72
09/14	apeks : imperial	1.86 : 1.88	2.12 : 1.90			0	16.72
09/15	5yclone : fnatic	3.15 : 1.34	2.50 : 1.67	5yclone	L	-1	15.72
09/15	m80 : 9z	2.49 : 1.49	2.20 : 1.83	m80	L	-1	14.72
09/16	fnatic : apeks	1.55 : 2.39	2.29 : 1.77	apeks	L	-1	13.72
09/16	5yclone : imperial	2.40 : 1.54	2.56 : 1.64			0	13.72
09/16	9z : apeks	3.14 : 1.34	2.08 : 1.93	9z	W	2.14	15.86
09/17	navi : faze	1.89 : 1.87	1.77 : 2.29	navi	W	0.89	16.75
09/17	5yclone : 9z	2.09 : 1.70	2.68 : 1.59	9z	W	0.7	17.45
09/17	liquid : virtuspro	2.60 : 1.47	2.62 : 1.62			0	17.45
09/20	g2 : lynn-vision	1.07 : 7.61	1.69 : 2.44	lynn-vision	L	-1	16.45
09/20	col : 9ine	2.27 : 1.60	1.86 : 2.16	col	W	1.27	17.72
09/20	cloud9 : eternal-fire	1.34 : 3.14	1.75 : 2.34	eternal-fire	L	-1	16.72
09/20	virtuspro : g2	2.43 : 1.52	2.12 : 1.90	virtuspro	L	-1	15.72
09/21	cloud9 : col	1.23 : 3.96	1.57 : 2.75	col	W	2.96	18.68
09/21	eternal-fire : 9ine	1.83 : 1.91	1.86 : 2.16			0	18.68
09/21	liquid : lynn-vision	1.27 : 3.60	1.80 : 2.24	lynn-vision	L	-1	17.68
09/21	eternal-fire : virtuspro	2.65 : 1.45	1.93 : 2.08	eternal-fire	L	-1	16.68

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
09/22	cloud9 : liquid	1.48 : 2.57	1.75 : 2.34	liquid	W	1.57	18.25
09/22	lynn-vision : 9ine	2.88 : 1.39	2.50 : 1.67	lynn-vision	W	1.88	20.13
09/23	virtuspro : liquid	1.67 : 2.15	1.53 : 2.89	virtuspro	W	0.67	20.8
09/23	eternal-fire : cloud9	3.01 : 1.36	1.90 : 2.12	eternal-fire	W	2.01	22.81
09/23	lynn-vision : liquid	4.27 : 1.21	2.82 : 1.55	lynn-vision	L	-1	21.81
09/24	col : g2	4.56 : 1.18	2.29 : 1.77	col	L	-1	20.81
09/24	eternal-fire : liquid	2.21 : 1.62	1.62 : 2.62	eternal-fire	W	1.21	22.02
09/24	fnatic : monte	1.69 : 2.10	2.29 : 1.77	monte	W	1.1	23.12
09/26	astralis : eternal-fire	1.48 : 2.56	1.69 : 2.44	eternal-fire	W	1.56	24.68
09/26	virtuspro : 9z	1.17 : 4.85	1.59 : 2.68	9z	W	3.85	28.53
09/26	mouz : furia	1.57 : 2.33	1.55 : 2.82	mouz	W	0.57	29.1
09/26	col : monte	1.70 : 2.08	2.04 : 1.96	monte	W	1.08	30.18
09/27	big : eternal-fire	2.15 : 1.66	2.34 : 1.75			0	30.18
09/27	riders : 9z	1.93 : 1.81	2.24 : 1.80	9z	L	-1	29.18
09/27	faze : mouz	1.66 : 2.16	2.75 : 1.57	mouz	W	1.16	30.34
09/27	vitality : monte	1.14 : 5.35	1.45 : 3.24	monte	W	4.35	34.69
09/28	navi : eternal-fire	1.49 : 2.53	1.59 : 2.68			0	34.69
09/28	ence : riders	1.17 : 4.74	1.45 : 3.24	riders	L	-1	33.69
09/29	g2 : mouz	1.56 : 2.35	1.96 : 2.04	mouz	W	1.35	35.04
09/29	monte : navi	2.74 : 1.43	2.39 : 1.72	monte	L	-1	34.04
09/30	ence : mouz	1.73 : 2.05	2.08 : 1.93	mouz	W	1.05	35.09
09/30	navi : mouz	1.85 : 1.91	2.75 : 1.57	mouz	W	0.91	36
10/01	navi : apeks	2.38 : 1.58	1.72 : 2.39	navi	L	-1	35
10/16	mouz : vertex	1.23 : 4.15	1.90 : 2.12	vertex	W	3.15	38.15
10/16	vitality : betboom	1.41 : 2.90	1.53 : 2.89	betboom	W	1.9	40.05
10/16	faze : gamerl	1.41 : 2.87	1.75 : 2.34	gamerl	W	1.87	41.92
10/16	ence : lynn-vision	1.33 : 3.32	1.55 : 2.82	lynn-vision	W	2.32	44.24
10/16	fnatic : cloud9	2.28 : 1.60	2.24 : 1.80	fnatic	W	1.28	45.52
10/16	monte : col	1.79 : 2.00	2.20 : 1.83	col	L	-1	44.52
10/16	g2 : grayhound	1.22 : 4.12	1.45 : 3.24	grayhound	L	-1	43.52
10/16	navi : mouz	4.36 : 1.21	3.44 : 1.41	navi	L	-1	42.52
10/16	betboom : gamerl	1.66 : 2.27	1.77 : 2.29			0	42.52
10/16	apeks : vertex	1.27 : 3.65	1.96 : 2.04	vertex	L	-1	41.52
10/16	vitality : faze	1.56 : 2.42	1.51 : 2.97	vitality	L	-1	40.52
10/16	monte : grayhound	1.33 : 3.33	1.38 : 3.67			0	40.52
10/17	lynn-vision : cloud9	4.52 : 1.21	2.29 : 1.77	lynn-vision	L	-1	39.52
10/17	ence : fnatic	1.46 : 2.69	1.39 : 3.55	ence	W	0.46	39.98
10/17	apeks : gamerl	1.94 : 1.84	1.77 : 2.29	apeks	L	-1	38.98
10/17	col : g2	2.62 : 1.49	2.24 : 1.80	col	L	-1	37.98
10/17	navi : faze	4.04 : 1.23	1.96 : 2.04	navi	L	-1	36.98
10/17	col : cloud9	2.47 : 1.54	2.04 : 1.96	col	W	1.47	38.45
10/18	monte : fnatic	1.98 : 1.82	1.64 : 2.56	monte	W	0.98	39.43
10/18	gamerl : faze	2.81 : 1.43	2.20 : 1.83	gamerl	L	-1	38.43
10/18	mouz : betboom	1.70 : 2.13	1.72 : 2.39			0	38.43
10/18	g2 : ence	1.70 : 2.13	1.86 : 2.16			0	38.43
10/18	monte : col	1.85 : 1.93	1.86 : 2.16			0	38.43
10/18	ence : faze	1.87 : 1.92	1.75 : 2.34	ence	L	-1	37.43
10/20	betboom : col	1.98 : 1.82	2.24 : 1.80	col	W	0.82	38.25
10/20	mouz : faze	2.40 : 1.57	1.77 : 2.29	mouz	L	-1	37.25
10/21	g2 : col	1.35 : 3.22	1.55 : 2.82	col	W	2.22	39.47
10/21	col : faze	2.68 : 1.46	2.12 : 1.90	col	L	-1	38.47
10/22	faze : nip	1.18 : 4.54	1.53 : 2.89	nip	L	-1	37.47
11/22	cloud9 : navi	1.65 : 2.15	1.69 : 2.44			0	37.47
11/22	vitality : heroic	1.36 : 2.98	1.93 : 2.08	heroic	L	-1	36.47

Date	Match	Book Odds	Gen. Odds	Bet	W/L	Amount	P/L
11/22	col : astralis	1.97 : 1.78	1.83 : 2.20	col	W	0.97	37.44
11/22	nip : navi	2.99 : 1.37	2.29 : 1.77	nip	L	-1	36.44
11/23	faze : cloud9	1.49 : 2.53	1.55 : 2.82			0	36.44
11/23	heroic : astralis	2.39 : 1.54	1.86 : 2.16	heroic	W	1.39	37.83
11/23	vitality : col	1.45 : 2.67	1.93 : 2.08	col	L	-1	36.83
11/23	cloud9 : heroic	1.40 : 2.83	1.90 : 2.12	heroic	L	-1	35.83
11/24	navi : col	1.93 : 1.84	2.24 : 1.80	col	W	0.84	36.67
11/24	vitality : cloud9	1.57 : 2.33	1.86 : 2.16	cloud9	L	-1	35.67
11/25	faze : col	1.33 : 3.18	1.55 : 2.82	col	L	-1	34.67
11/25	vitality : faze	2.09 : 1.70	2.29 : 1.77			0	34.67
11/26	gamerl : cloud9	3.75 : 1.27	2.16 : 1.86	gamerl	L	-1	33.67
12/05	mibr : betboom	1.82 : 1.96	2.00 : 2.00			0	33.67
12/05	astralis : virtuspro	1.95 : 1.83	2.68 : 1.59	virtuspro	W	0.83	34.5
12/06	furia : spirit	2.24 : 1.63	2.24 : 1.80			0	34.5
12/06	gamerl : betboom	1.92 : 1.84	2.44 : 1.69	betboom	W	0.84	35.34
12/06	cloud9 : mibr	1.35 : 3.17	1.67 : 2.50	mibr	W	2.17	37.51
12/07	betboom : cloud9	3.66 : 1.26	2.20 : 1.83	betboom	W	2.66	40.17
12/08	virtuspro : spirit	1.88 : 1.87	2.39 : 1.72	spirit	L	-1	39.17
12/07	astralis : furia	1.75 : 2.03	2.08 : 1.93	furia	W	1.03	40.2
12/07	spirit : furia	1.54 : 2.39	1.43 : 3.33	spirit	W	0.54	40.74
12/08	mibr : spirit	2.62 : 1.46	2.50 : 1.67	mibr	L	-1	39.74
12/09	betboom : virtuspro	2.96 : 1.37	2.12 : 1.90	betboom	L	-1	38.74
12/09	virtuspro : spirit	1.77 : 2.00	2.20 : 1.83	spirit	W	1	39.74
12/10	cloud9 : ence	1.51 : 2.47	1.83 : 2.20	ence	L	-1	38.74
12/13	vitality : navi	1.36 : 3.05	1.59 : 2.68	navi	L	-1	37.74
12/13	faze : heroic	1.21 : 4.18	1.57 : 2.75	heroic	L	-1	36.74
12/13	mouz : g2	2.13 : 1.68	2.08 : 1.93	mouz	L	-1	35.74
12/13	ence : navi	2.29 : 1.60	2.24 : 1.80	ence	L	-1	34.74
12/14	cloud9 : vitality	2.83 : 1.40	2.29 : 1.77	cloud9	L	-1	33.74
12/14	heroic : mouz	2.73 : 1.43	1.69 : 2.44	heroic	L	-1	32.74
12/14	faze : g2	1.49 : 2.52	1.45 : 3.24	faze	W	0.49	33.23
12/14	cloud9 : mouz	1.58 : 2.31	1.59 : 2.68			0	33.23
12/15	navi : g2	2.55 : 1.50	2.04 : 1.96	navi	W	1.55	34.78
12/15	faze : mouz	1.43 : 2.72	1.49 : 3.06			0	34.78
12/16	vitality : navi	1.29 : 3.55	1.41 : 3.44	navi	L	-1	33.78
12/16	faze : vitality	2.30 : 1.59	1.69 : 2.44	faze	L	-1	32.78

B Research Ethics Approval Letter

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA · UNIVERSITEIT VAN KAAPSTAD

2023/12/12

COM/00541/2023

RE: Research Ethics Committee Project Approval Letter

Dear Henri du Plessis,

Your application for ethics review of your project titled

Machine Learning for Professional Counter-Strike Match Prediction

has been reviewed and evaluated by the

Commerce Research Ethics Committee.

You may proceed with your research project titled:

Machine Learning for Professional Counter-Strike Match Prediction

Please note that should:

(i) any serious or adverse effects to participants occur and/or,

(ii) aspect(s) of your current project change and/or

(iii) any unforeseen events that might affect continued ethical acceptability of the project occur then you should immediately report this to the approving REC. You may be required to submit an amendment to this application, in order to determine whether the changed aspects increase the ethical risks of your project.

Based on the information supplied your application has been successful and is approved.

Please note the following additional conditions associated with this approval:

(i)

Regards,

Commerce Research Ethics Committee.

Bibliography

1. Hillyer M. Here's how technology has changed the world since 2000. 2020 Nov. Available from: https://www.weforum.org/agenda/2020/11/heres-how-technology-has-changed-and-changed-us-over-the-past-20-years/
2. Torrance J, O'Hanrahan M, Carroll J, and Newall P. The structural characteristics of online sports betting: a scoping review of current product features and utility patents as indicators of potential future developments. Addiction Research & Theory 2023; 0:1–15. DOI: 10.1080/16066359.2023.2241350
3. Taylor TL. Raising the Stakes: E-Sports and the Professionalization of Computer Gaming. The MIT Press, 2012 Mar. DOI: 10.7551/mitpress/8624.001.0001
4. Gray A. The size and increase of the global sports betting market. 2020 Aug. Available from: https://www.sportsbettingdime.com/guides/finance/global-sports-betting-market/
5. Etuk R, Xu T, Abarbanel B, Potenza MN, and Kraus SW. Sports betting around the world: A systematic review. Journal of Behavioral Addictions 2022; 11:689–715. DOI: 10.1556/2006.2022.00064
6. Trenhaile M. How Bookmakers Create their Odds, from a Former Odds Compiler. 2017 Jun. Available from: https://tradematesports.medium.com/how-bookmakers-create-their-odds-from-a-former-odds-compiler-5b36b4937439
7. Cain M, Law D, and Peel D. The favourite-longshot bias, bookmaker margins and insider trading in a variety of betting markets. Bulletin of Economic Research 2003; 55:263–73
8. Badillo K. History of Sports Betting. 2023 Nov. Available from: https://playtoday.co/blog/history-of-sports-betting/
9. Lamb J. History of sports betting. 2023 Oct. Available from: https://www.sportingpost.com/sports-betting/history-of-sports-betting/
10. Gray A. The History of sports betting legislation in the USA. 2020 Nov. Available from: https://www.sportsbettingdime.com/guides/legal/sports-betting-history-part-i/
11. Gray A. The History of sports betting legislation in the USA. 2020 Nov. Available from: https://www.sportsbettingdime.com/guides/legal/sports-betting-history-part-ii/
12. Avramovic M. Exploring the Roots: The Gripping History of Online Gambling. 2024 Jan. Available from: https://playtoday.co/blog/guides/history-of-online-gambling/
13. Linnehan R. Where is sports betting legal? State by state legal tracker. 2024 Jan. Available from: https://www.sportsbettingdime.com/guides/legal/sports-betting-state-by-state-legal-tracker/
14. Borowy M and Jin DY. Pioneering eSport: The Experience Economy and the Marketing of Early 1980s Arcade Gaming Contests. International Journal of

Communication 2013; 7. Available from: https://ijoc.org/index.php/ijoc/article/view/2296
15. Greig C. The ongoing rise of the esport Industry - Michigan Journal of Economics. 2024 Jan. Available from: https://sites.lsa.umich.edu/mje/2024/01/15/the-ongoing-rise-of-the-esport-industry/
16. Mozur P. For South Korea, E-Sports is national pastime. 2014 Oct. Available from: https://www.nytimes.com/2014/10/20/technology/league-of-legends-south-korea-epicenter-esports.html
17. Geyser W. The incredible growth of eSports. 2023 Oct. Available from: https://influencermarketinghub.com/esports-stats/
18. Hardenstein TS. Skins in the game: Counter-strike, esports, and the shady world of online gambling. UNLV Gaming LJ 2017; 7:117
19. Greer N, Rockloff M, Browne M, Hing N, and King DL. Esports Betting and Skin Gambling: A Brief History. Journal of Gambling Issues 2019. DOI: 10.4309/jgi.2019.43.8
20. Bräutigam T. Is it a problem that esports betting sites are sponsoring teams? 2019 Jul. Available from: https://archive.esportsobserver.com/is-it-a-problem-that-esports-betting-sites-are-sponsoring-teams/
21. Lewis R. New evidence points to match-fixing at highest level of American Counter-Strike. 2018 Aug. Available from: https://dotesports.com/general/news/match-fixing-counter-strike-ibuypower-netcode-guides
22. Integrity and Fair Play. 2015 Jan. Available from: https://blog.counter-strike.net/index.php/2015/01/11261/
23. Taylor-Hill G. Abios Report: Huge Esports Betting Lead for Counter-Strike. 2024 Jan. Available from: https://www.esportsbets.com/news/abios-report-esports-betting-2023
24. HLTV: About. 2024 Jan. Available from: https://www.hltv.org/about
25. Liquipedia Statistics Portal. 2024 Jan. Available from: https://liquipedia.net/counterstrike/Portal:Statistics
26. The Making of Counter-Strike. Retro gamer :84–87. Available from: https://issuu.com/michelfranca/docs/retro_gamer____119
27. Mitchell F. Esports Essentials: The impact of the Counter-Strike Majors. 2018 Sep. Available from: https://archive.esportsobserver.com/esports-essentials-counter-strike-majors/
28. Trahan P. How many people play CS2? Counter-Strike 2 Player count in 2023. 2023 Oct. Available from: https://www.dexerto.com/counter-strike-2/how-many-people-play-cs2-player-count-record-2071859/
29. PGL Major Stockholm 2021 - Liquipedia Counter-Strike Wiki. Available from: https://liquipedia.net/counterstrike/PGL/2021/Stockholm
30. Ispas V. PGL Major Stockholm 2021 broke all viewership records in CS:GO history. 2021 Nov. Available from: https://press.pglesports.com/163642-pgl-major-stockholm-2021-broke-all-viewership-records-in-csgo-history
31. SmooveBwainPways. Steam Community Guide: Printable Counter-Strike Maps. Available from: https://steamcommunity.com/sharedfiles/filedetails/?id=230110279
32. ESL Challenger Jönköping 2023 Monte vs Virtus.pro HLTV Map page

33. Pickard J and Redbull. What's the role of sports psychology in esports? 2019 Jan. Available from: https://www.redbull.com/za-en/mia-stellberg-sports-psychology-interview-esports-prism
34. Hamstead C. 'Nobody talks about it because everyone is on it': Adderall presents esports with an enigma. 2020 Feb. Available from: https://www.washingtonpost.com/video-games/esports/2020/02/13/esports-adderall-drugs/
35. Alpaydin E. Machine learning. Mit Press, 2021
36. Samuel AL. Some studies in machine learning using the game of checkers. IBM Journal of research and development 2000; 44:206–26
37. Jordan MI and Mitchell TM. Machine learning: Trends, perspectives, and prospects. Science 2015; 349:255–60. DOI: 10.1126/science.aaa8415. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8415
38. What is Machine Learning? | IBM. Available from: https://www.ibm.com/topics/machine-learning
39. Killick EA and Griffiths MD. In-Play Sports Betting: a Scoping Study. International Journal of Mental Health and Addiction 2019; 17:1456–95. DOI: 10.1007/s11469-018-9896-6
40. Davies M, Pitt L, Shapiro D, and Watson R. Betfair.com:: Five Technology Forces Revolutionize Worldwide Wagering. European Management Journal 2005; 23:533–41. DOI: https://doi.org/10.1016/j.emj.2005.09.008. Available from: https://www.sciencedirect.com/science/article/pii/S0263237305000976
41. Koning R and Velzen B. Betting Exchanges: The Future of Sports Betting? International Journal of Sport Finance 2009 Feb; 4:42–62
42. Marinkovic G, Lukic N, and Medvidovic N. Online sports betting through the prism of software engineering. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. Virtual Event, USA: Association for Computing Machinery, 2020 :1455–1465. DOI: 10.1145/3368089.3417049. Available from: https://doi.org/10.1145/3368089.3417049
43. Business Research Insights. ESports Betting Market Share & Growth Analysis Report 2031. Accessed: date-of-access. 2023. Available from: https://www.businessresearchinsights.com/market-reports/esports-betting-market-100204
44. Horvat T and Job J. The use of machine learning in sport outcome prediction: A review. WIREs Data Mining and Knowledge Discovery 2020; 10:e1380. DOI: https://doi.org/10.1002/widm.1380
45. Islam M, Chen G, and Jin S. An overview of neural network. American Journal of Neural Networks and Applications 2019; 5:7–11
46. Jenny SE, Manning RD, Keiper MC, and Olrich TW. Virtual(ly) Athletes: Where eSports Fit Within the Definition of "Sport". Quest 2017; 69:1–18. DOI: 10.1080/00336297.2016.1144517
47. Hosmer DW and Lemeshow S. Applied Logistic Regression. 2nd. New York: John Wiley & Sons, 2000 :31–117
48. Breiman L. Random Forests. Machine Learning 2001; 45:5–32. DOI: 10.1023/A:1010933404324. Available from: https://doi.org/10.1023/A:1010933404324
49. Cortes C and Vapnik V. Support-vector networks. Machine Learning 1995; 20:273–97. DOI: 10.1007/BF00994018. Available from: https://doi.org/10.1007/BF00994018

50. 1.4. Support vector machines. Available from: https://scikit-learn.org/stable/modules/svm.html
51. Cover T and Hart P. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 1967; 13:21–7. DOI: 10.1109/TIT.1967.1053964
52. Duda RO and Hart PE. Pattern Classification and Scene Analysis. New York: John Wiley & Sons, 1973
53. Bishop CM. Neural Networks for Pattern Recognition. Oxford: Oxford University Press, 1995 :114–204
54. Alsudani S. Enhancing Thyroid Disease Diagnosis through Emperor Penguin Optimization Algorithm. Wasit Journal for Pure sciences 2023 Dec; 2:66–79. DOI: 10.31185/wjps.230
55. Albawi S, Mohammed TA, and Al-Zawi S. Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*. 2017 :1–6. DOI: 10.1109/ICEngTechnol.2017.8308186
56. Friedman JH. Greedy function approximation: A gradient boosting machine. Annals of Statistics 2001; 29:1189–232. DOI: 10.1214/aos/1013203451
57. Chen T and Guestrin C. XGBoost: A Scalable Tree Boosting System. CoRR 2016; abs/1603.02754. eprint: 1603.02754. Available from: http://arxiv.org/abs/1603.02754
58. Natekin A. Topic 10. Gradient Boosting. Available from: https://www.kaggle.com/code/kashnitsky/topic-10-gradient-boosting
59. Elo AE. The Proposed USCF Rating System: Its Development, Theory, and Applications. Chess Life 1967 Aug; 22:242–7
60. Glickman M. The Glicko system. 2016 Sep. Available from: http://www.glicko.net/glicko/glicko.pdf
61. Herbrich R, Minka T, and Graepel T. TrueSkill(TM): A Bayesian Skill Rating System. *Advances in Neural Information Processing Systems 20*. MIT Press, 2007 :569–76. Available from: https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/
62. Hodge VJ, Devlin S, Sephton N, Block F, Cowling PI, and Drachen A. Win Prediction in Multiplayer Esports: Live Professional Match Prediction. IEEE Transactions on Games 2021; 13:368–79. DOI: 10.1109/TG.2019.2948469
63. Yang Y, Qin T, and Lei YH. Real-time eSports Match Result Prediction. 2016. arXiv: 1701.03162 [stat.AP]
64. Bednárek D, Krulis M, Yaghob J, and Zavoral F. Data Preprocessing of eSport Game Records - Counter-Strike: Global Offensive. 2017 Jan :269–76. DOI: 10.5220/0006475002690276
65. Marshall S, Mavromoustakos Blom P, and Spronck P. Enabling Real-Time Prediction of In-game Deaths through Telemetry in Counter-Strike: Global Offensive. 2022 Nov :1–10. DOI: 10.1145/3555858.3555859
66. Xenopoulos P, Doraiswamy H, and Silva CT. Valuing Player Actions in Counter-Strike: Global Offensive. CoRR 2020. Available from: https://arxiv.org/abs/2011.01324
67. Xenopoulos P, Coelho BG, and Silva CT. Optimal Team Economic Decisions in Counter-Strike. CoRR 2021. Available from: https://arxiv.org/abs/2109.12990
68. Švec O. Predicting Counter-Strike Game Outcomes with Machine Learning. 2022
69. Schmidt Z and Meerhoff R. Esports Match Result Prediction for a Decision Support System in Counter-Strike : Global Offensive. 2020. Available from: https://api.semanticscholar.org/CorpusID:232402954

70. Björklund A, Visuri WJ, Lindevall F, and Svensson P. Predicting the outcome of CS:GO games using machine learning. 2018. Available from: https://api.semanticscholar.org/CorpusID:70208061
71. Python Software Foundation. Python Language Reference, version 3.9. 2024. Available from: https://docs.python.org/3/reference/
72. Alexisbcook. Data Leakage. Available from: https://www.kaggle.com/code/alexisbcook/data-leakage
73. HLTV World ranking. Available from: https://www.hltv.org/ranking/teams/2024/january/22