Universiteit Leiden

Master Computer Science

Esports Match Result Prediction for a Decision Support System in Counter-Strike: Global Offensive

Name:	Zachary Schmidt
Student ID:	s2267861
Date:	[19/07/2020]
Specialisation:	Data Science
1st supervisor:	Dr. Mike Preuss
2nd supervisor:	Dr. Rens Meerhoff

Computer Science Master's Thesis

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The aim of this thesis is to predict the results of matches in the major esports title Counter-Strike: Global Offensive (CS:GO). A CS:GO match can take place on a variety of maps, similarly to how tennis has different court materials such as cement, grass, or clay. In CS:GO, the opposing teams undergo a map selection process before a match to determine which map(s) will be played. Thus, the scope of our problem involves predicting the result of a match between two teams on each of the seven maps. After collecting and analyzing CS:GO match data for each map, clustering players based on playstyle, and modelling player and match features, we use machine learning to predict the winner of a CS:GO match based on the map(s) selected by the teams. This provides a decision support system to assist and advise professional teams in their map selection process, and in preparation for specific opponents. These techniques borrow from traditional sports data science, where modern data analytics techniques have revolutionized player performance and strategy development. Gathering sports data is historically difficult, where the data comes from complicated computer vision models at best, or from humans counting the statistics by hand at worst. Esports is facing the opposite problem- an embarrassment of riches where the data is already digitized and collected to a degree, due to the nature of games themselves being digital environments. However, esports as a whole is so young that data analytics implementations are still in their infancy. The goal of this research is to combine the developments in sports data analytics with the wealth of data inherent to esports in order to predict the result of a CS:GO match per-map.

Contents

1 Introduction

Counter-Strike: Global Offensive is a tactical first-person shooter in which two teams of 5 players each take turns attacking and defending crucial locations for many rounds. In 2019, Counter-Strike: Global Offensive (CS:GO) boasted the third largest prize pool of all esports with $21.8 million, with the highest number of professional players at 3,889, and the highest number of tournaments at 818 [1, 2]. What separates CS:GO from other modern top esports is its long history. The prize money awarded throughout Counter-Strike's history is the second largest in all of esports at $97.2 million [1]. Despite Counter Strike's 20-year history since its original release in 1999, CS:GO (2012) reached its highest player peak of all time just now in 2020 with over 1.3 million concurrent players [3].

In traditional sports, data analytics has revolutionized many aspects of modern professional sports. The process of picking up new players for a team used to be fully in the hands of gurus with an eye for talent, and this process now revolves heavily around considering the statistics touted by each player. Everything from heart rate to shot accuracy is now being tracked, which suggests where players best fit on a team or how they can best improve their performance [28]. On a macro level, team decisions and strategies are tracked and analyzed. Instead of just studying the tape, teams can now model and predict the strategies chosen by opponents [67. 68].

While the possibilities of sports data analytics have many of implications on the game in the way of player performance and team strategy, analysts are relatively starved of data due to the limitations and difficulties of recording and collecting this real-world data [29]. Tracking an individual's information such as heart rate or number of steps taken has been commodified in the modern day [33], but recording the coordinates of the ball throughout a game of football is significantly more difficult. Adding a tracking device to the ball would provide excellent data, but official-grade balls cannot be tampered with. A computer vision model could be built to track ball and player position, but this already nontrivial task is exacerbated when considering that the lighting and angles are different on different courts/fields, with different weather, at different times of the day, and so on [30-32]. Data science has revolutionized aspects of modern sports, but as with many data science fields, there are limitations regarding the quantity and diversity of collectable data.

Conversely, video games are *made* of data. Many game developers take advantage of this data availability by providing a bevy of statistics for the match and for each player when the game is finished. Additionally, nearly any statistics or information not output in-game can be parsed from the game files. This provides many potential avenues for collecting high quantities of diverse data. However, the ways in which usable data can be recorded and accessed are contingent on the ways that the developer has programmed the game. While esports data is hypothetically more abundant and accessible than traditional sports data, esports data scientists are still bottlenecked by which avenues to data that the game developer chooses to open.

Through personal interviews and discussions with experts in the field, many top CS:GO organizations and groups are just beginning to take the first steps into esports data analytics. Currently, the analysts advising professional teams have limited databases and implement rather low-level statistical analysis techniques. In the still-emerging esports world, expanding

the number and increasing the quality of human analysts takes priority over their machine counterparts. Many of these teams outsource their data analytics work to external agencies which are specialists in data, but not in esports. For the team organizations that are beginning their own foray into the data science world, their efforts are currently focusing on building data lakes containing high varieties and quantities of data. Once these lakes are filled, cleaned, and curated, teams are primarily interested in using data science to improve practice and to predict and counter an opponent's strategy. Services that provide products such as betting services, fantasy esports leagues, or match result coverage sites have larger databases of statistics, match results, and user behavior. However, the pipeline surrounding the design, implementation, and monetization of significant machine learning undertakings is still largely in the ideas phase. These groups are particularly interested in a match result predictor, which users could bet/predict against. Another interest is in player performance predictors could be used to draft fantasy teams. Esports data is inherently and relatively abundant, and the interest in implementing data science and machine learning techniques exist but are hardly realized. This work aims to contribute to bridging that gap.

2 Problem Statement

CS:GO is played in a best-of-30 rounds, where the first team to win 16 rounds wins the match. One team starts on offense (T) and the other on defense (CT). After 15 rounds, the teams switch sides. A "map" is the landscape where the match takes place, ranging from locations such as a train yard, urban Germany, a nuclear power plant, and so on. CS:GO has 7 maps in the map pool for competitive play. At professional tournaments, the standard for a match is to play a best-of-3 maps, where each map is a best-of-30 rounds. Note that while there are 7 potential maps, only 3 are played in a best-of-3. Some maps are very strategically dependent (Nuke, Overpass), some rely on good grenade usage like smokes/flashes/incendiaries (Inferno, Train), and some are more aim-based (Cache, and the famous Dust2) [27]. Figure 1 shows the aim-centric layout of Dust 2, which is evident in its long, straight sightlines. Figure 2 contrasts these long and straight sightlines by showing Nuke, which is highly strategy-dependant due to its many close corners and small rooms.

Figure 1: Layout of the map Dust 2. Each round, the defending team begins at the blue icon and defends sites A and B, and the attackers begin at the yellow icon. There are many long and straight lanes, making this map very dependent on aim [4].

Figure 2: Nuke features many more walls, small rooms, and elevations. Additionally, ramps and ladders allow players to navigate to the underground "lower" B site. This map facilitates and rewards strategy and tactics, where the attacking team uses their grenades and positioning to move the defenders around the map and away from crucial locations [5].

To decide which maps are played in the match, the teams take turns banning and picking maps in the "map veto." This map selection process can determine the winner of a match before a single round is even played, given the strengths and weaknesses of each team on each map. An example of a map veto process where Team 1 plays Team 2 in a best-of-3 is as follows:

Team 1 bans map A
Team 2 bans map B
Team 2 picks map C
Team 1 picks map D
Team 1 bans map E
Team 2 ban map F
Map G is the decider

There is deep complexity to the map veto strategy. The disparity between the terrains of CS:GO is similar to how tennis has courts with concrete, grass, or clay, and different players have different strengths on different materials. In a hypothetical tennis court veto example, if my opponent knows I'm better on clay, then I might expect them to ban clay. Then, I could secretly train on grass and ban concrete to gain an advantage. Or, my opponent could specifically choose clay knowing they're worse, but they could have studied my tapes to try to find my weaknesses in the court type that I think will be an easy win.

The overall goal of this work is to apply modern data science techniques to esports game data in order to build a decision support system which informs and advises teams and analysts. Studying the tape of an enemy team's playstyle and performance on multiple maps takes many of hours to prepare for just one match. In preparing to face an underdog, there are fewer methods for researching their strategy and performance as they have played fewer public matches against high-level opponents. Additionally, time spent analyzing lower-level opponents takes time away from analyzing higher-level opponents. So, all teams are interested in quickly predicting their team's performance against any level of opponent. These matchup predictions across all 7 maps advise the team as to which maps are best to pick and ban. To formalize our research question, how could we best provide decision support to CS:GO teams in map vetos using data analytics?

The pipeline of this work can be divided into three parts: data acquisition, model building, and conveying results. A model can only be as good as its data, so a great deal of consideration is necessary in gathering high-quality data that is as large and as clean as possible. Data comes in many forms in CS:GO. Given that our work focuses on high-level teams, the most immediate form of data is match results between top teams. Using who beat who, on what day in time, with what score, and on what maps is already excellent data for building a model which predicts the outcomes of matches before they happen. The next level of data is the statistics output by the game itself for each match, showing information about each round and each player's performance. This data can be used to build a model which predicts how players will perform and how the future of an ongoing match will go. The final and highest-level method is to parse the match data from the game itself. This includes all previous data discussed, along with a nearly endless list of additional statistics and data points. The coordinates of each player, every player action, and every interaction with the world can be pulled from the recording of the game. So, we must consider different data sources, determine which models

best predict match results per-map, and convey these results to a team.

3 Related Work

3.1 Player Ranking Systems

In the research of works to support this paper, a large amount of data science developments in esports contexts focuses on player ranks and skill matchups. Historically, metrics like Elo [6] or TrueSkill [7] are used to estimate the skill of a player and thus the skill of a team in order to build the most fair match where each team is estimated to have a 50% chance to win. A high quality matchmaking experience benefits all players of a game, so many improvements upon this system have been made. Some approaches use the outcomes of a player's matches over time to intelligently model their expected performance in the future [8]. Others model the skill of an individual player to a much deeper and more game-specific level, relative to their position, playstyle, or role of choice in order to build the best matchmaking experience.

CS:GO Player Skill Prediction [9] is a standout example of these papers, as this paper uses CS:GO data. The author of this paper engineered a massive collection of CS:GO match recordings from three separate sources. These files were parsed using CSLGO's API in order to build a multitude of features describing attributes of gameplay and playstyle for thousands of players. After these features were cleaned and analyzed, the author used these features to predict the skill of these players through machine learning. This work models player skill in a deeply complex and personal manner when compared to the traditional ranks which are mostly contingent on wins and losses.

Predicting the Outcome of CS:GO Games Using Machine Learning [10] is another highly relevant work in which the authors collected and parsed a large number of demo files, curated dozens of features for players, used clustering to group these players based on playstyle, and used a neural network to predict the winner of a CS:GO match. The authors implemented a genetic algorithm to assign weights used in clustering, which improved the strength of the clusters. Through developing more and more features per player, the authors achieved an accuracy of 65.11%.

3.2 Sports Data Science

In traditional sports, even before the modern data revolution, major scientific developments have been made regarding the understanding of how individuals behave in a team environment [59-61]. Working together in a sports team inspires societal and interpersonal research [62, 63], just as societal research leads to understanding sports team dynamics to a deeper level. While these advances in modelling and understanding individual behavior and team dynamics use traditional sports as their application, many of the core theories and principals are applicable to esports [64, 65]. Defining, modelling, and analyzing coordination between individuals is a fundamental of team sports [11-14]. Complex systems provide a framework of interactions which can be used to model sports teams as well [15, 16]. Ecological dynamics describe the relationships between individuals and their environment, and can be applied in sports contexts

[17, 18]. These fields of research provide methods of representing and understanding sports teams, so aspects of these sports team frameworks can be applied to esports teams as well.

3.3 Decision Support Systems

A decision support system is a program designed to support the judgements and courses of action in an organization [34, 35]. This field originates in management and business [36, 37], but is widely used in healthcare [38-41] and in the stock market [42-44] as well. Decision support systems have been implemented to assist in improving player performance [45, 46], as well as player and team strategy [47]. One work in particular [48] designed a domain-specific decision support system for esports, using data from Starcraft 2. Finally, as our decision support system is designed to estimate the likelihood that one team will beat another, there are clear use cases for this system in sports betting [49-51] and in esports betting [52-58].

3.4 AI in Counter Strike

Leetify [19] is a service released in 2019 that uses artificial intelligence to analyze a player's CS:GO gameplay and suggest areas of improvement. Leetify gathers a player's past matches, builds specific features, and compares the player's performance against its model of the average player at that player's rank. While modelling player performance and statistics at each rank is impressive, the beauty of Leetify is in the features that it constructs by parsing the match demo files. It collects the trivial statistics output immediately by the game (such as headshot percentage) which are commonly understood to be important and useful, but it also creates novel features which provide significant insights to a player and their playstyle. For example, they are able to measure crosshair placement by calculating the angle between where a player is currently aiming and the location of an enemy when they appear on screen. Some examples of these features in action are shown in Figures 3 and 4. While leetify is a significant contribution to the field of esports analytics, it focuses on the performance and mechanics of an individual player. A whole team could be analyzed one by one, but as of right now, Leetify does not offer any strategy analysis or any projections into the future.

Figure 3: Leetify showing that my past crosshair placement is worse than the benchmark for my rank, while my current form is now better than the benchmark.

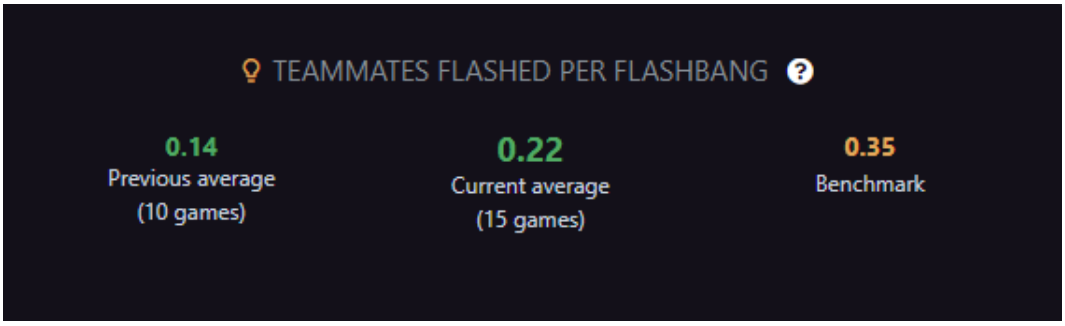

Figure 4: Try not to flash your own teammates.

Last year, the CS:GO tournament organizer Blast offered a bounty for an individual or team that developed the best match result predictor using machine learning [19]. The winner would receive a $5,000 reward and would have their model showcased during the analysis desk segments during the live broadcast of the tournament. This year in 2020, Blast have teamed up with the match analysis service Skybox to offer a bounty for an AI system that recognizes and predicts specific in-game situations in CS:GO and states likely outcomes [20]. While the instructions are vague and the data is private to registered teams, this still shows the currently wide-open possibilities for what can be accomplished with this data, and shows the hunger for data science techniques in esports analytics.

Finally, a famous and well-documented feat in the fields of both AI and esports has been OpenAI for Dota 2 [21]. The OpenAI group created an AI system which learned to contest and even defeat professional teams in Dota 2, a top and modern esport. While Dota 2 is also made by Valve, this is a 5v5 multiplayer online battle arena (MOBA) instead of a first-person shooter like CS:GO. And while the paper you are currently reading is not trying to build an AI to play CS:GO (that would be cheating), a particularly interesting aspect of the AI is that it has an ongoing estimate of each side's winrate from right when the characters are selected all the way through to the end of the match.

4 Data

When a game of CS:GO is played, all ten players connect to a CS:GO server. This server can be thought of as the "absolute truth" of the match, as it is responsible for communicating with each of the 10 players by receiving each player's inputs, processing the inputs through the physics of the game, and sending the new state of the game back to the players. This pipeline takes time, as players have varying internet connectivity speeds. This affects how soon they are able to send or receive communications to and from the server (called "ping," measured in miliseconds). While an enemy is watching a sightline, it is possible for a player to turn the corner and kill the enemy before the server even tells that enemy that the player came around the corner. For this reason, nearly all official matches at the pro level are played on LAN, Local Access Network, where players play in-person and are connected directly to the server. This allows them to play on the server's "absolute truth" of the state of the match with no communication delays.

4.1 Demo Files

The server records this absolute truth of the game, containing all players' inputs and actions. This recording is called a demo, and is saved in a DEM file format. These demo files are a few hundred megabytes for each CS:GO match, and can be played back from any first-person, third-person, or free-camera point-of-view on any CS:GO game client. This custom file format was originally designed to only be processed within the CS:GO game client to view past matches. However, Valve later released an open-source demo parsing tool.

The smallest denomination of time on the game server is called a "tick." As of the writing of this thesis, Valve public matchmaking servers operate using 64 ticks per second and private matchmaking servers use 128 ticks per second. In each tick, the server records all inputs sent from each client (player), processes these inputs using the server's logic and physics, updates its state, and sends this output to all clients. These ticks are smoothed by the server in order to ensure smooth transitions between animations for players on higher latency (ping). The data in these ticks is recorded in bursts, each consisting of 8 bits which record events and changes. An event is an interaction such as killing a player, planting a bomb, or the round ending. A change records changes in player positions or ammo amounts. Player position data is only recorded when a player moves, so this data is strictly sequential and must be processed in full rather than in parts. Given that this file type is custom-made for CS:GO demos, it is highly difficult to pull data by hand outside the game client.

Valve released a freely available demo parsing tool [22] where a user can build a parser that outputs all events from a demo file. This is used by many websites and services to parse and publish match results and player performance. Additionally, a wrapper written in Go was developed for the parser [23] to help specify which specific data to pull and in which format.

```
[T]Ex6TenZ <AK-47> [CT]tiziaN
[CT]keev <Tec-9 (HS)> [T]mistou
[T]xms /F/ <AK-47> [CT]syrsoNR
[T]Ex6TenZ <AK-47> [CT]keev
[CT]crisby <M4A1 (HS)> [T]Ex6TenZ
[CT]crisby <M4A1 (HS)> [T]ALEX
[T]to1nou <AK-47> [CT]kzy LJ
[T]xms /F/ <AK-47> [CT]crisby
Round finished: winnerSide=T  ; score=4:11
```

In this example of events in a round, we see on line 1 that Ex6TenZ is on the terrorist side and gets an AK-47 kill on tiziaN. After the T's kill all 5 CT's, the round is won by the T's and the score is now 4 T to 11 CT.

4.2 Data Source

HLTV.org is a company which tracks and records professional CS:GO matches. Their website offers match history and statistics for pro teams and their players. Along with these rather detailed statistics, each match page within a certain history offers a download link to the demo

file of the match. This makes HLTV the best public source for a high number of professional-level demo files. Unfortunately, HLTV does not offer a public API for pulling larger amounts of data. Building or implementing a scraper is the next logical step to download a significant number of these demo files, but HLTV uses CloudFlare to protect against DDoS attacks and similar malicious activity. Additionally, building a nontrivial set of demos would require well into terabytes of storage space. This many-TB database could then be parsed using Valve's Demo Parser in order to generate a working database of features. This is the optimal method by which to build a dataset for this project. But, navigating past Cloudflare, straining the website, and risking a ban from the site are a bit beyond the scope of this project when paired with the massive sizes of data used to generate a significantly smaller-sized dataset. Therefore, we explore another data source which offers a few downsides in the richness of data but significant upsides in accessibility, processing, and size.

Figure 5: Example of information and statistics available through HLTV [24].

FACEIT is a service which offers a matchmaking client, tournaments, leagues, and most notably pick-up games (pugs). Esports differs drastically from sports is in its prevalence of pick-up games. The human body can only handle a certain degree of intense physical activity playing traditional sports, and practicing often requires traveling to a specific location. In esports, when pro players have finished their team practice for the day, many opt to continue practicing and training by playing pick-up games with other top-level players. Using basketball as an example, consider a world where the best basketball players across all of North America played in mixed all-star matches all night every night. This is not entirely representative of an official match at the highest level, as players are playing on mixed teams where roles overlap, English is used instead of the players' native language, and so on. However, these are still examples of top players in the world showcasing and recording their performance. The most popular pug service is FACEIT, which offers pro-level pug systems in Europe and in North America. FACEIT tracks statistics and general details for both players and matches similarly to HLTV. And crucially, FACEIT offers a highly detailed portal for developers which includes a Data API [25] that provides free access to large amounts of data.

5 Building a Database

5.1 FACEIT Data API

An API, or Application Programming Interface, provides a set of possible requests a user can use to pull objects from a server. This is done by establishing a server-client relationship through administering an API key specific to the client. This key identifies and authenticates the user, allowing them to request data from the server using the API's methods. The FACEIT Data API provides a highly accessible and intuitive means to build a database using their data. A user registers an account with the site, builds an app to work from, and generates an API key. Armed with a key, the user now has access to a plethora of community-developed API wrappers in your language of choice, as well as a very useful in-browser interface designed for testing different calls. There are many categories of data types, and each category has different methods for pulling more specific data. The full list of categories and their methods is found in Table 1.

Category	Methods
Search	Players, teams, tournaments, championships, hubs, organizers
Games	All games, game details
Players	Details, past matches, hubs, statistics, tournaments
Matches	Details, statistics
Rankings	Global/regional ranking, player's position ranking
Teams	Details, statistics, tournaments
Tournaments	List, details, brackets, matches, teams
Championships	Details, matches, subscriptions
Hubs	Details, matches, members, roles, rules, statistics
Leaderboards	Championship, group, current hub, all time hub, seasonal hub
Organizers	Details, associated games, championships/tournaments/hubs hosted

To begin, we first examine the `Players` category in Figure 6. By expanding the `/players` player details method, we can see the parameters of this method in Figure 7.

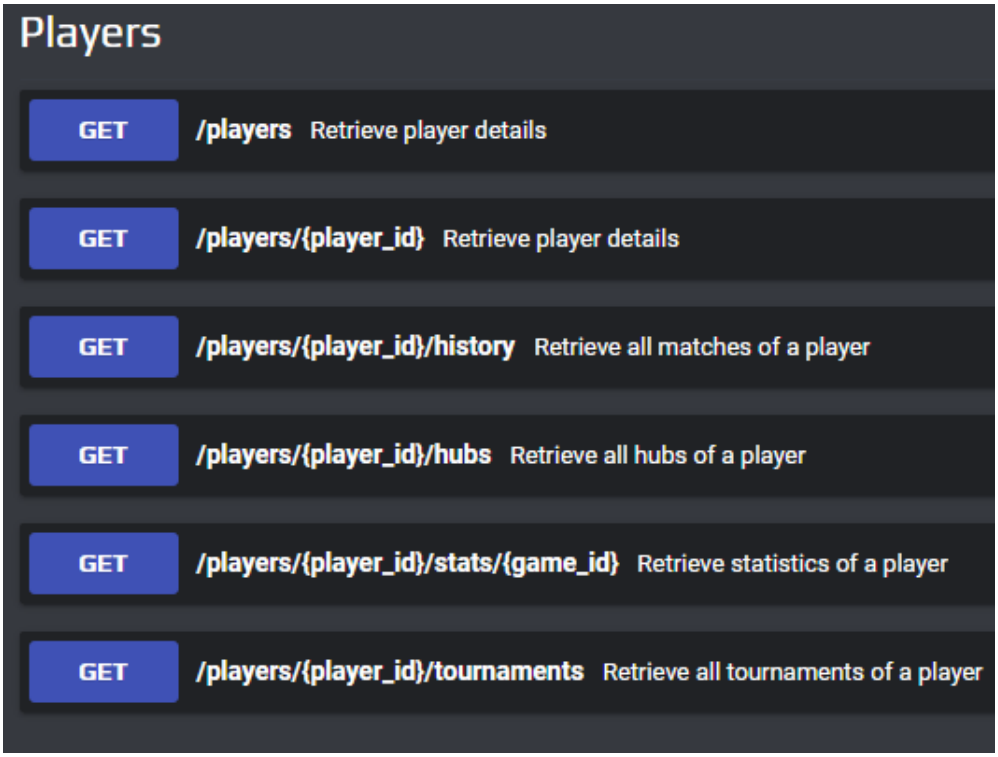

Figure 6: FACEIT Data API `Players` category [25].

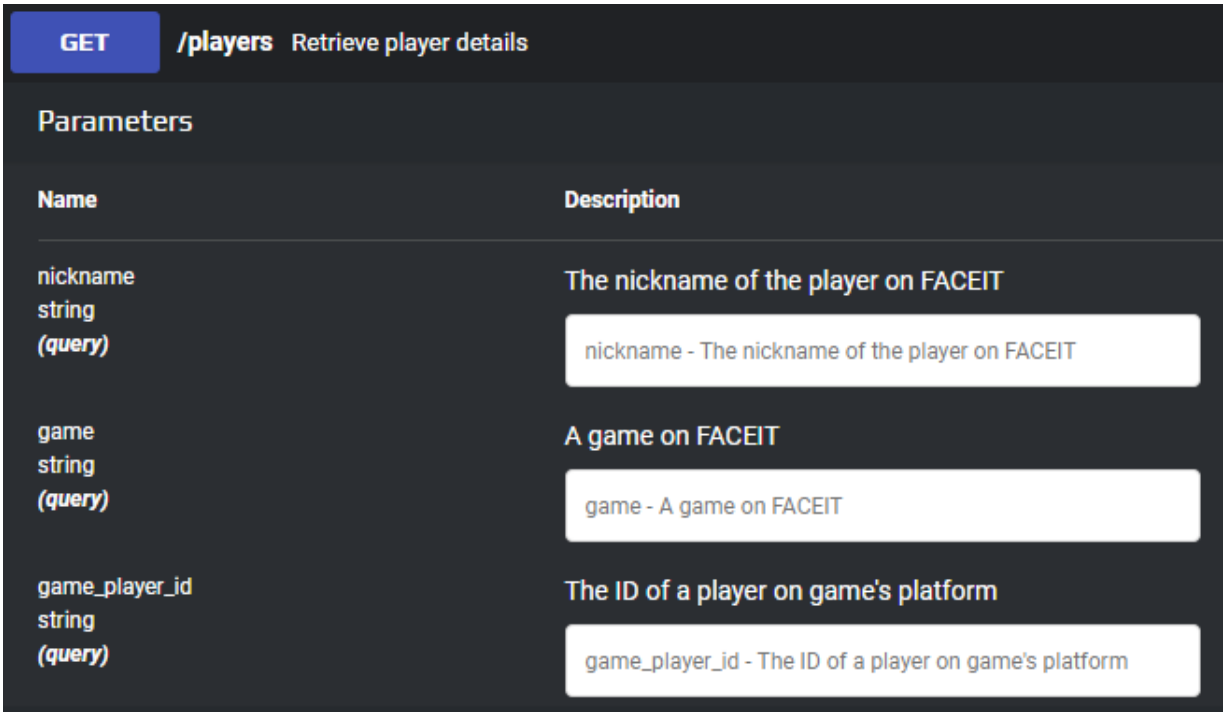

Figure 7: FACEIT Data API `/players` player details method parameters [25].

The data is returned in the form of JSON blocks, as the data can be complexly structured. JSON (JavaScript Object Notation) is a method for structuring data with varying levels of complexity and depth, and can be processed by any programming language. This is useful for storing information on players and matches in CS:GO due to their variable size and structure. By entering the nickname "s1mple" in the `/players` method, we receive a JSON block of details about this player (with some sections removed for visual clarity).


```
{
  "player_id": "ac71ba3c-d3d4-45e7-8be2-26aa3986867d",
  "nickname": "s1mple",
  "country": "ua",
  "games": {
    "csgo": {
      "game_profile_id": "b02abbdd-6019-4f17-b385-34bc071f35c8",
      "region": "EU",
      "regions": {
        "EU": {
          "selected_ladder_id": "40c6f7d2-32a1-4cb1-984b-d025bb695286"
        }
      },
      "skill_level_label": "10",
      "game_player_id": "76561198034202275",
      "skill_level": 10,
      "faceit_elo": 3356,
      "game_player_name": "s1"
    }
  },
  "settings": {
    "language": "ru"
  },
```


This provides crucial data, most notably the player ID. Additionally we see some skill rankings such as `skill_level` and `faceit_elo`. This data was pulled using an API wrapper written in Python [26]. A combination of pandas and NumPy were used to build data frames from the JSON blocks pulled from the API. Jupyter Notebook was used so that API calls could be made once and stored locally, as to not overload the server. If a user makes more than 10,000 requests within 1 hour, the user is blocked from further calls until the next o'clock hour.

We now build two datasets: one for player data, and one for match data. Focusing on the player dataset first, in order to get player matches or statistics, we must start with a player ID. There are many ways to get player IDs. As this work aims to focus on the top level of CS:GO, we pull the top players from the regional ranking in the `Rankings` section. We are able to iterate through each batch of 100 players at a time in order to generate a list of player IDs. For each player ID in the player ID list, we can call the player statistics method from the player object. This provides a list of general lifetime statistics, and then more detailed statistics

which are separated by map. Due to the hierarchical structure of JSON, per-map statistics are essentially dataframes nested within dataframes, so some digging is necessary to access all of the data relevant to us. The lifetime statistics are rather immediate, and converting them to a dataframe leaves us with columns for each statistic as well as additional columns containing entire JSON blocks of individual map data. We can iterate through each of these cells containing JSON blocks and append their contents to a running list of per-map stats, and then convert this list to a dataframe which contains the player ID, the map, and the stats of this player on this map. We can pull additional lifetime stats from these more detailed statistics by iterating through each of these maps for each player and combining our findings. This gives us a playerstats dataframe with the following features:

Players: Win%, Recent Results, Current Win Streak, Longest Win Streak, Number of Matches, Kills Per Death (K/D), Headshot (HS)%

Player per map: Matches, Rounds, Wins, Kills, Deaths, Assists, Triple-kills, Quad-kills, Penta-kills, MVPs, K/D, Total Headshots

There are some metrics such as number of multi-kill rounds or total headshots that are available in the per-map dataset. We can add these missing metrics to the player lifetime dataset by grouping the per-map dataset by player and combining the data from all of theplayer's maps. Headshots are included in these statistics because in CS:GO, headshots deal much more damage to players and often yield a one-hit-kill. Headshot % tracks which percentage of a player's shots are headshots, and Total Headshots tracks how many headshots the player hits in total. Finally, MVPs are a distinction awarded to 1 player each round for having the highest "impact" in a round in the form of planting the bomb, getting the most kills, and similar achievements.

Next, we build the match dataset. Using the `Hubs`, we can pull all the recent matches in FACEIT in order. This returns a multitude of match IDs and details, which can then be used to pull the match stats for each match ID using the Matches object. In the same way as with players, we can pull 100 match IDs at a time until we collect 10,000, and for each ID, we can pull the match stats. We first note that while these IDs are recently recorded matches, perhaps due to an error from the API, only 6,693 of these matches returned their statistics. This JSON block is tiered as well, where we need to extract specific cells from each block. Through expanding dataframes and collecting the relevant data, we build a match database with the following features:

Matches: Number of Rounds, Score, Map, Winner

Match per team: First Half Score, Second Half Score, Final Score, Player IDs

Per player: Kills, Deaths, K/D, Kills per Round (K/R), Triple-kills, Quad-kills, Penta-kills, MVPs, Headshots, HS%, Match Result (win or loss)

The statistics of each player in this match is already captured in their lifetime stats on the given map, so these are removed from the dataset. Additionally, the most recent matches aren't guaranteed to include all of the top players. By checking for which player IDs exist in

the match dataset but do not exist in the player dataset, we get a short list of 63 player IDs to pull statistics for and add to our player dataset. This yields 10,063 players and 6,693 matches in total.

For our player dataset, our data takes different forms. Some are percentages, some are totals, and some are averages. Additionally, in CS:GO, matches can be of different lengths. If a player achieves 25 kills in a match that was a 16-0, getting 1.56 kills per round is quite impressive. But if a player gets 25 kills in the world record for the longest CS:GO match of all time, which went to 6 overtimes for a total of 88 rounds finishing 46-42 [66], 0.28 kills per round is suddenly much less impressive. Additionally, if one player has played on Overpass for 500 pugs and another has played 5, the total number of 3k's would be radically disproportionate. By normalizing our data for how a player tends to perform per round instead of per match, we get the true representation of their performance. To do this, we divide the following features by number of rounds played: Kills, Deaths, Assists, Total Headshots, Triple Kills, Quadro Kills, Penta Kills, and MVPs. Metrics such as Average K/D or Average Headshot % are naturally already normalized. Recent Results is a list stored in the form of a string, such as '[1, 0, 0, 1, 0]'. By using a literal evaluation in Python, this is translated to a list of integers which can then give us the player's win percentage in their past 5 games.

5.2 Exploratory Data Analysis

5.2.1 Player Statistics

To begin this section, quite the interesting statistic is in the lifetime statistics of a player, called "Total Headshots %". This statistic is the summation of every headshot percent recorded for this player for every match, resulting in data such as 155,692%. Dividing this number by their 3,250 matches, you find that their average headshot percent is 48%. However, there is already another statistic in the same lifetime statistics section for this player called "Average Headshot %" which equals 48%. The "Total Headshots %" statistic may be used for calculations elsewhere or may just not have been removed from this section. Another similar variable is "Total K/D Ratio," which itself is not a ratio but a summation of every individual K/D ratio of this player, which seems to indicate that this particular player gets 4,333 kills for every death. Luckily "Average K/D Ratio" is present as well.

	faceit_elo	Quadro Kills	Kills	Deaths	Rounds	Triple Kills	Penta Kills	MVPs	Headshots	Assists
count	10000.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00
mean	2978.19	589.64	44579.32	39585.18	57001.01	2699.14	68.62	6773.18	20561.75	7855.94
std	244.79	383.68	27332.68	24785.18	34939.18	1692.63	47.32	4202.72	13139.71	4869.64
min	2703.00	2.00	151.00	179.00	255.00	6.00	0.00	15.00	64.00	32.00
25%	2797.00	311.00	24150.50	21012.50	30728.50	1445.00	35.00	3660.00	10834.00	4236.00
50%	2918.00	501.00	39116.00	34888.00	50485.00	2331.00	57.00	5903.00	17838.00	6920.00
75%	3094.00	784.00	59676.00	52971.50	76488.00	3601.00	91.00	9031.00	27443.50	10485.00
max	6909.00	3637.00	213112.00	170579.00	249154.00	15090.00	507.00	35682.00	107145.00	36324.00

Figure 8: Player Lifetime Statistics dataset summary

Figure 8 is our statistical summary of the relevant categories from our player lifetime statistics dataset. We observe first that a few players have a significantly high ELO. All other columns are a total over the player's career on FACEIT, so until we divide these by the player's number of matches, we do not get an accurate representation of the true values in this section. It is nice to see that even the player with the fewest matches played out of the 10,063 players still has 2 Quadro-kills.

	Rounds	Average Headshots %	Kills/R	Deaths/R	Quadro Kills/R	Triple Kills/R	Penta Kills/R	Assists/R	MVPs/R	Headshots/R
count	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00	10063.00
mean	57001.01	0.46	0.79	0.69	0.01	0.05	0.00	0.14	0.12	0.36
std	34939.18	0.06	0.06	0.03	0.00	0.01	0.00	0.02	0.02	0.06
min	255.00	0.26	0.56	0.44	0.00	0.02	0.00	0.05	0.06	0.19
25%	30728.50	0.43	0.75	0.67	0.01	0.04	0.00	0.13	0.11	0.32
50%	50485.00	0.46	0.78	0.69	0.01	0.05	0.00	0.14	0.12	0.36
75%	76488.00	0.50	0.82	0.71	0.01	0.05	0.00	0.15	0.13	0.40
max	249154.00	0.74	1.18	0.79	0.04	0.12	0.01	0.24	0.46	0.74

Figure 9: Player Lifetime Statistics dataset normalized per-round

Now that all features have been normalized per-round in Figure 9, we are able to glean meaningful insight. The best headshot percent is 74%, and the worst is 24%. Kills and Deaths per round for each player have values that are very close together, but some high-performing players have many kills and others have few deaths. Interestingly, for every ace (Penta-kill), we expect approximately 10 Quadro-kills and 45 Triple-kills. MVPs show little deviation, and as a metric which highlights a variety of accomplishments (getting kills, planting the bomb, or defusing the bomb) which are often distributed somewhat evenly among players and thus displays a low variance. Lastly, we note that the Average Headshots per round is lower than the Average Headshot %. This can be explained by the fact that players sometimes have rounds where they don't fire at all, thus lowering their Average Headshots per round but not impacting their Average Headshot %.

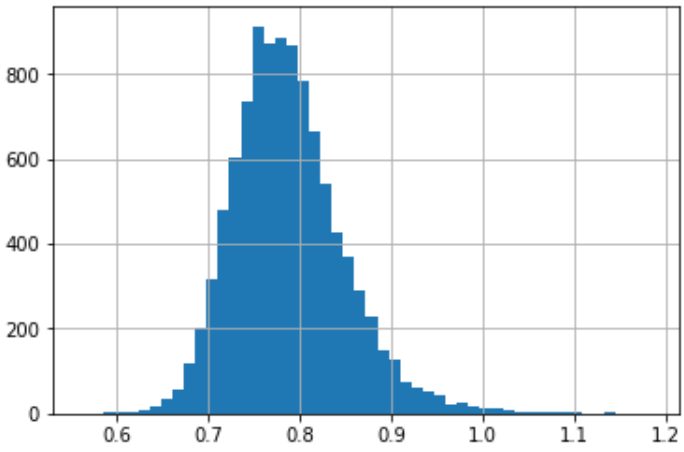

Figure 10: Kills per Round

Taking a closer look at Kills per Round in Figure 10, this distribution has a slight positive skew. A few stand-out players get more kills than others, but they may have higher deaths per

round as a result.

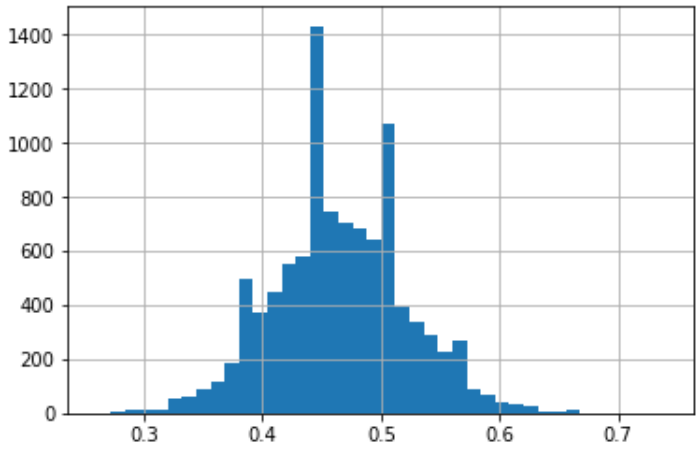

Figure 11: Average Headshot Percentage

When observing Average Headshot Percentages in Figure 11, we notice significant spikes at particular locations. Average Headshot Percentage is rounded by FACEIT, which can explain these particularly high-frequency values.

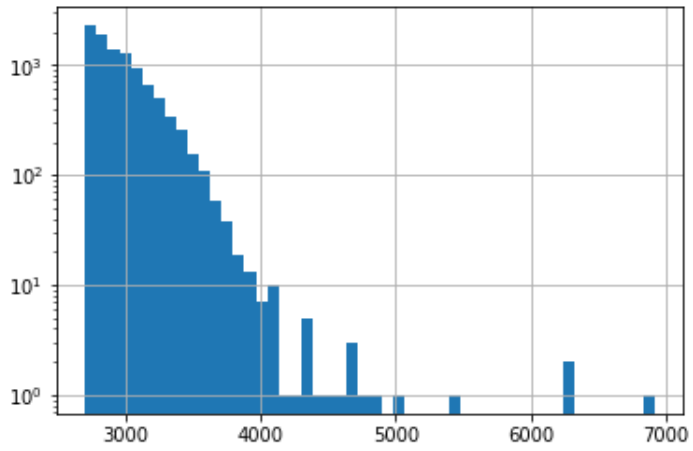

Figure 12: FACEIT Elo (log-scaled)

These 10,063 players are the top players in FACEIT, so they reside in the top echelons of Elo as shown in Figure 12. Most interesting is that our log scale highlights that there is a single-digit number of players with more than 5,000 Elo.

5.2.2 Map Statistics

Examining the distribution of maps played in Table 1, the pug favorite Mirage is the most played map by a wide margin. Strategically-focused maps such as Nuke and Overpass find themselves much closer to the bottom of the list. We also observe older maps that there are not in the current map pool, such as Cobblestone or Season. Finally, we note that aim maps appear in players' match history, so these must be excluded from our datasets and models.

Map	Frequency	Percentage
de_mirage	6506910	30.08
de_cache	3967091	18.34
de_inferno	2975454	13.76
de_dust2	2255996	10.43
de_train	1976287	9.14
de_overpass	1893029	8.75
de_cbble	977654	4.52
de_nuke	690307	3.19
de_vertigo	252150	1.17
aim_map	66278	0.31
aim_redline	23572	0.11
aim_map2_go	16496	0.08
aim_map_cl	11318	0.05
de_season	7366	0.03
aim_9h_ak	5438	0.03
aim_ak47_v2	3104	0.01
de_trailerpark	2	0.00

Table 1: Frequency of maps played in our match dataset.

For each map, by dividing the number of matches played by the number of rounds played, we obtain the average number of rounds per match on each map in Table 2. Since the winning team finishes the game at 16 rounds, by subtracting 16 from the average number of rounds, we see the average score of the losing team on each map. Vertigo matches result in slightly closer scores, with Cobble and Train being a bit more one-sided.

Map	Rounds	Mean Loss Score
de_vertigo	26.82	10.82
de_inferno	26.67	10.67
de_dust2	26.61	10.61
de_overpass	26.58	10.58
de_mirage	26.55	10.55
de_cache	26.45	10.45
de_nuke	26.42	10.42
de_train	26.40	10.40
de_cbble	26.25	10.25

Table 2: Mean number of rounds played per map, and mean score of the losing team.

5.2.3 Linear Regressions

Linear regressions describe the relationship between a response variable and its explanatory variables. Using a linear regression, we can analyze the impact of each statistic as they relate to winning matches. The linear regression output is displayed in Table 3.

Variable	**Coefficient**
ELO	0.11
Average K/D	0.13
Kills/Round	-0.66
Deaths/Round	-0.46
Average HS %	-0.63
HS/Round	0.92
3k/Round	2.60
4k/Round	-1.07
5k/Round	-13.7
Assists/Round	0.65
MVPs/Round	0.67

Table 3: Linear regression predicting Win Rate %.

A player's recent results and current win streak have minor impacts on leading to wins. Elo and Average K/D have high positive impacts on a player's win percentage. Surprisingly, Kills Per Round and Deaths Per Round both have a negative relationship with win percentage. As Average K/D is a linear combination of Kills/Round and Deaths/Round, this can explain our model attempting to disregard these metrics. Average Headshots % has a negative coefficient, while Headshots/Round has a positive coefficient. This could be to say that players who take fewer fights and thus fewer shots win fewer games compared to players who prioritize engaging in more fights. A player's average Triple-Kills Per Round has a positive correlation with wins, but Quad-Kills and Aces are both negative. It could be that players that go for Quad-Kills and Aces are not team-players by chasing after kills, or it could be that players only face the opportunity for a Quad-Kill or Ace after the rest of their team is already dead. However, it is more likely that these variables are so infrequent already that their impact is very slight. While -13.2 seems largely negative, we remember that the average number of aces per round is 0.001. Finally, assists and MVPs have moderate positive coefficients. This is a surprise in the case of MVPs. This metric is somewhat arbitrary considering how the game logic decides who to award, but intuitively, being a team player or a high performer that is acknowledged with an MVP could have a nontrivial relationship with winning matches.

5.2.4 Headshots

We briefly deviate from our main research to take a cursory look into the age-old question: whether or not Headshot Percentage is a relevant or meaningful metric. On its own it does describe a player's skill and playstyle, but in this section we analyze whether or not aiming for the head correlates with greater success in any of our other meaningful statistics. Using linear models, we see in Figures 13 and 14 that a player's Headshot Percentage is nearly irrelevant

to their K/D and their number of multi-kills. A slightly negative relationship with assists in Figure 15 is interesting, but a headshot will often kill the enemy immediately so no assist could be possible.

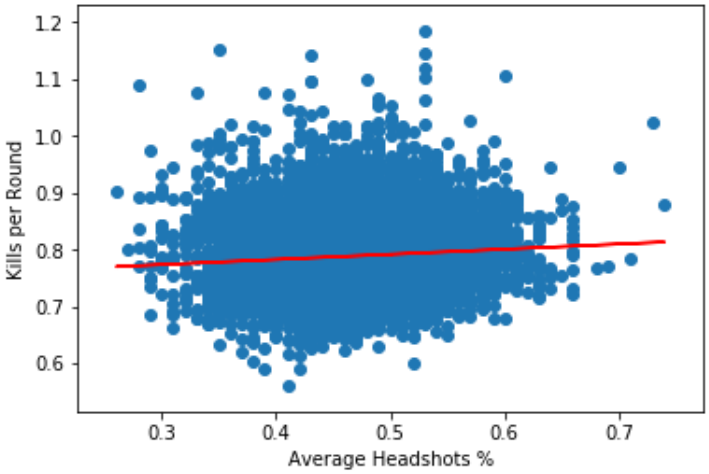

Figure 13: Linear regression using Average Headshot % to predict Average Kills/Round.

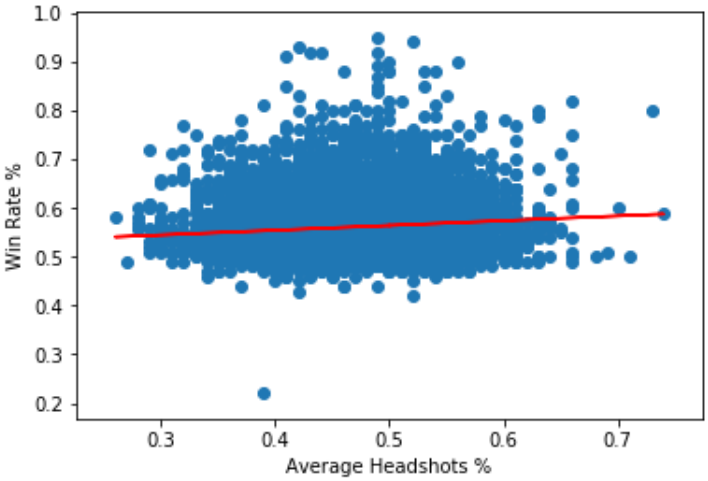

Figure 14: Linear regression using Average Headshot % to predict Win Rate %.

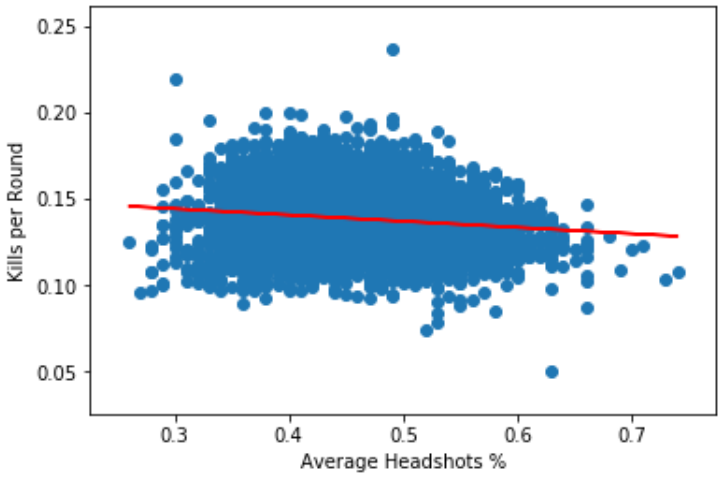

Figure 15: Linear regression using Average Headshot % to predict Average Assists/Round

6 Experimental Approach

Armed with our data, our task is to develop a decision support system which could advise teams in their map veto by estimating their team's likelihood of winning against another team on each map. First, we decide which player features to include, and we may choose to cluster players based on playstyle. Next, we implement a machine learning algorithm which models team compositions of players and models how two team compositions match up. Our model then outputs its expectation for team A to beat team B in the form of a percentage. We compare the performance of different models in order to determine the optimal configuration. Finally, we showcase the effectiveness of our decision support system by applying it to data from the grand finals of the world championship.

Our first step in our experiment design is to consider the types and combinations of player data that we wish to model. Some examples of choices are using the player's lifetime stats only, using the player's stats on only this specific map, or using a player's stats on all maps at once. We also have latent statistics such as Elo or Win Rate which measure player performance over many matches. This is different from our in-server statistics such as Kills and Deaths that are accumulated over individual matches. Since Elo and Recent Results are measurements of player performance themselves, a pitfall for our model could be that it simply predicts "the team with the higher Elo and Win Rate will win." So, we must analyze the model performance across different combinations of data, and investigate the impact that including latent performance statistics will have on our model.

Another option for preparing player data is to cluster players based on their performance. Using k-means clustering, we are able to group together players that exhibit similar performance. This shows us what a "playstyle" is in terms of our data, which players have similar playstyles to others, and which players have more unique playstyles. Clustering also serves to reduce the dimensionality of our data by labeling the playstyle of each player using a single digit, providing streamlined data to our models. Dimensionality reduction and playstyle definitions are beneficial and interesting, but they come at the cost of lossy compression of our data. By representing a player's full list of statistics with a single digit, many details and nuances of this

player's performance are lost. In k-means clustering, we choose the number of clusters k, thus choosing the level of error we are comfortable with incurring. The levels of error are shown in an elbow plot, where we iterate through numbers of clusters k and plot the error incurred by each k. Having a high value of k captures more aspects of our data with less error, but a higher k becomes less and less useful for describing patterns and reducing dimensionality. A low k yields exponentially larger error and fails to capture the significant patterns in our data. So, the common approach to selecting k is to choose the value of k at the bottom of the "elbow" in our curve. After this elbow, the decreases in error are marginal, thus not worth trading more dimensionality for. While the elbow plot indicates the hypothetically optimal value of k in terms of compression, we may also arbitrarily select any number of k for comparison. So in selecting and curating player data for our model, we may choose to cluster the data and may choose different numbers of clusters k.

Once we have selected the input player data, how do we model combinations of players on a team? How do two team compositions match up against each other? To understand which aspects of a team and the opposing team contribute to a win or a loss, we implement a random forest. This machine learning algorithm is able to model complex relationships within the data while requiring only 1 parameter to be tuned, the number of trees. For consistency between our experiments, we set the number of trees to be 10,000 for each experiment. For each match in our dataset, we pull the statistics of each player on each team to use as input data. We use 70% of our data for training and 30% for testing our model performance on unseen data. After receiving player statistics and (potentially) map information as input, the random forest returns an estimation of the percent chance that it believes team A will beat team B.

After a model is built, we measure its performance. In typical binary classification where we estimate if a match should be a win or a loss using accuracy as the performance metric, the model's estimation of 51% is rounded up to "win." In this measure of accuracy, model outputs of 51% and 99% are both classified as a win, while 50.1% to 49.9% is the difference between a win and a loss. Many of our estimations are expected to be near 50%, so the true result of one match being win or loss does not make the model's estimation inaccurate. In other words, if we are attempting to estimate the fairness of an unfair coin that we only get to flip once, the result of just one flip does not indicate the correctness of our estimation. Accuracy can provide a cheap and immediate way of comparing which models are better than others, so this metric is still temporarily useful. Ultimately we are interested in modelling the confidence of our model's estimations. While we only observe one flip of the unfair coin (the match result), we can isolate estimations from our set of 6,693 matches that are significantly in favor of one team in order to analyze the confidence of the model. This answers questions such as "are win estimations near 70% actually wins near 70% of the time?"

The final goal of this work is to develop a full pipeline where a team can receive predictions for per-map results against an opponent. Using the estimations of the percent-chance to win on each map, a team would be advised to pick the maps with the highest estimations to win, and to ban the maps with the lowest estimations to win. Using this decision support, a team can alter and improve their approach to the map veto and their methods of preparation for the given opponent. We implement a use-case for our decision support system to investigate its effectiveness.

7 Experimental Analysis

7.1 Experiment 1: Predicting Team Wins Using Player Lifetime Stats

For our initial experiment, we test the effectiveness of using only lifetime statistics. These are a combination of a player's performance throughout their match history regardless of map, so this will serve as a baseline and a framework for subsequent experiments. These statistics from each player used in this model are as follows:

Average Kills per Round, Average Deaths per Round, Average Triple Kills per Round, Average Quadro Kills per Round, Average Penta Kills per Round, Average Assists per Round, Average MVPs per Round, Average Number of Headshots per Round, Average K/D Ratio, and Average Headshot Percentage

These provide a general indication of this player's overall playstyle and performance. So, is a model using this data capable of predicting who will win a match? We are interested in measuring how much greater than 50% the accuracy is, as 50% is the accuracy achieved by randomly guessing. We do expect that a majority matches will be estimated to be close matches, so we also do not expect results significantly greater than 50%. After combining player performance across dozens, hundreds, or thousands of their matches, our data is not fully able to approximate each player's performance in a single match. Instead we analyze the matchup of their team composition against their opponent's. A related work which used a neural network to predict match results using 30 player features instead of 10 achieved an accuracy of 63.33% [10]. So results near 50% are unsatisfactory, while results near 60% are more significant.

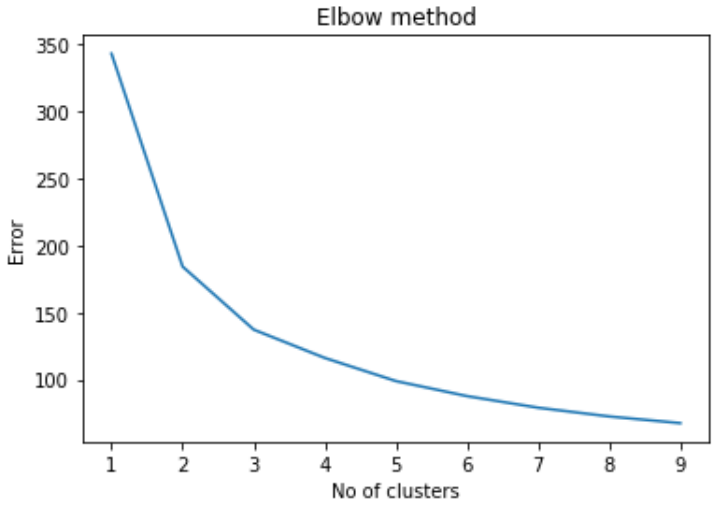

Figure 16: Elbow plot for our lifetime statistics of 10,063 players

We begin by clustering our players. We plot the results of our elbow method in Figure 16 by measuring the error incurred at each choice of k. A sharper elbow indicates that k-means clustering is an effective method for our data, but our elbow is rather smooth. The smoothness

of our curve makes it more difficult to choose a value for k, and indicates that clustering might not be an effective method for this set of features. The rightmost point of our slight bend is at $k = 5$, so for this initial experiment, we choose $k = 5$ clusters. We observe the cluster centers in Figure 17.

K/D	K/R	A/R	D/R	3k/R	4k/R	5k/R	HS%	HS/R	MVP/R
1.22	0.79	0.14	0.7	0.05	0.01	0.001	0.52	0.41	0.12
1.68	0.91	0.14	0.64	0.06	0.02	0.002	0.47	0.43	0.16
1.12	0.73	0.14	0.71	0.04	0.009	0.0009	0.45	0.33	0.11
1.27	0.79	0.14	0.68	0.05	0.01	0.001	0.42	0.33	0.11
1.41	0.85	0.14	0.67	0.05	0.01	0.001	0.48	0.4	0.13

Figure 17: k-means clustering centers where $k = 5$.

The second cluster represents a high-performing player with the highest K/D of 1.68, where we see the highest Kills/Round with a low number of Deaths/Round. The first and fourth centers represent players with very similar K/D ratios (1.22 vs 1.27) as well as other similar stats, but the first cluster differentiates itself by representing "better aimers" with their increased Headshots per Round and increased Headshot Percentage. The fifth cluster represents the "secondary stars" with a slightly lower K/D and slightly fewer multi-kills per round compared to the high-performing second cluster. The third cluster represents the low-performers, with the lowest K/D, most Deaths per Round, and fewest multi-kills. These clusters capture noticeable patterns between different players' playstyles. To the uninitiated, it's easy to predict that a team of all cluster 2 top-performers would be the optimal composition. Just as having an entire soccer team of high-scoring forwards does not a winning team make, there is an importance in understanding the dynamics and roles involved in team structure and supporting each other. The goal of machine learning is to model and understand the patterns in strengths and weaknesses of different team compositions in relation to each other.

To model these team dynamics and predict the winner of a match, we implement a random forest model. To prepare the clustered data, for each player ID in a match, we pull the cluster label the player belongs to. As the order of players on a team does not matter, we can instead count how many players belong to each one of the five clusters per team. This involves creating an array with length 5, where each column index represents a cluster. We then sum the number of players in each cluster on each team into one array, and repeat for the opposing team. The result is 10 single-digit features, and the final 11th binary column representing whether team A or team B won the match. Of these 6,693 rows, we split 70% of our data into training data and 30% into testing data. Using 10,000 trees, our model outputs the percent chance it believes that team A will beat team B. By rounding the output percentages to an absolute win or loss, we can compare this vector to the true result in order to calculate our accuracy. This implementation yields an accuracy of 50.38%. Unfortunately this is hardly better than random guessing, so we attempt a random forest model using the same data without clustering.

Since our data is no longer clustered, each player is represented by their 10 lifetime features. In this case we do not wish to combine the stats of the players on each team as was done with the cluster groups above. Given a hypothetical team average of 5 Kills per Round, there is a

significant difference between one team where each player averages 1 Kill per Round compared to a team where one player averages 5 Kills per Round and the others average 0. So, we keep the statistics of each player separate. The match dataset contains 10 columns of players in each match, so pulling each player's 10 lifetime features results in 100 columns with a 101st column representing the win or loss. Splitting our data with the same 70/30 train/test split over 10,000 trees, we achieve an accuracy of 51%. This is not significantly better than 50%, but does improve upon our clustered model. This indicates that these general lifetime statistics across a player's entire career are not capable enough data for making meaningful predictions.

7.2 Experiment 2: Predicting Team Wins Using Player Stats on All Maps

For our second experiment, we test the effectiveness of using each player's map-specific statistics. Comparing these results to the previous results, we investigate if and to what degree our prediction accuracy changes when considering the players' performance on each map individually. Each map contains the 10 features used above, so after normalizing per round for each of the 9 maps, we have a total of 90 features per player ID. If a player has never played a map, their stats for each feature of that map are set to 0. It would be better practice to ignore this player and thus the match, but eliminating all matches from our dataset in which all 10 players have not played all 9 maps would significantly reduce the number of matches for analysis. We begin by clustering players based on their performance across all maps.

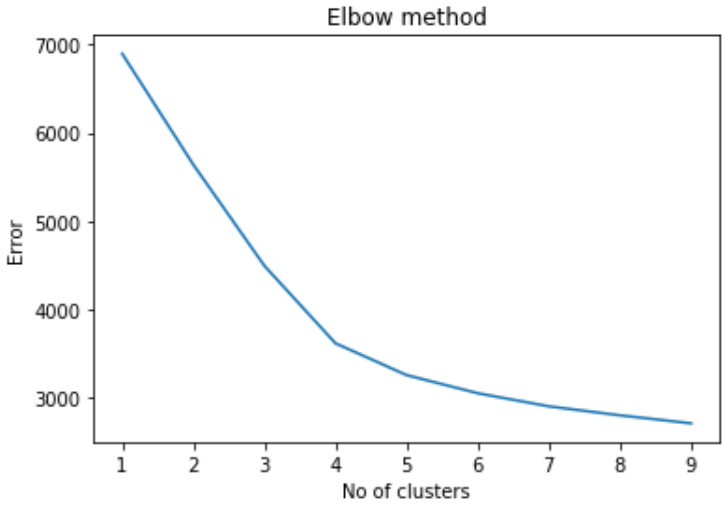

Figure 18: Elbow plot for our per-map statistics of 10,063 players

We see again in Figure 18 that our elbow is rather smooth, with $k = 5$ as a decent choice for k. Gleaning meaning from these cluster centers is much more difficult given their number of points, but clustering reduces the dimensionality of our team data from 900 columns to 10 columns. To prepare the match dataset for prediction with random forests, we begin by pulling all 10 players' new cluster label as well as which team won. Additionally, we now include the map name that is being played to get a per-map prediction. Random forests are not designed to handle strings, so we use one-hot encoding to represent the map name in a binary format.

With 10,000 trees, our random forest achieved an accuracy of 51.84%. This is a better result on clustered data compared to the result from Experiment 1, indicating that a player's playstyle can be better understood when it is separated by map.

Finally, we use the non-clustered version of our players' per-map data. Each of our 10 players has 10 features for each of the 9 maps, so after adding the map label and the winner indicator, our dataset has 910 columns and 6,693 rows. This model achieved an accuracy of 54.38%. We now achieve a more significant result, indicating that our model is beginning to recognize matches that strongly favor one team over the other. Again, our model should estimate that many matches are near 50% for both teams, which pulls our accuracy closer to 50%. By correctly identifying matches which are more strongly in favor of one team, our accuracy increases from 50%. So by including a player's map-specific performance and not clustering, we are able to output nontrivial estimations of the likelihood that team A will win against team B on map X.

7.3 Experiment 3: Predicting Team Wins Using Latent Player Statistics

With our first two experiments, we explored the viability of predicting match results using player performance data that is directly observable in the server. We did not use the latent statistics that are player performance measurements themselves, such as Elo or Win Rate, as these statistics could yield a model that simply predicts "the team with the higher Elo and Win Rate will win." However, the recent performance of players/teams and their win rates on each map are highly insightful and beneficial to a prediction model. Now that we have shown in Experiment 2 that it is possible to build a robust model using only in-server performance, we analyze the performance of a model that uses only the latent statistics Elo, Current Win Streak, Recent Results, and Win Rate Percentage to estimate the likelihood that Team A will win against Team B regardless of map. In a Random Forest with 10,000 trees, our model achieves an accuracy of 53.88%. This is a significant result using only 4 features to predict the winner of a match. Our first experiment struggled to predict the winner of a match regardless of map, but using these latent statistics, we are able to more accurately make these predictions.

7.4 Experiment 4: Exploring the Search Space of Possible Models

In designing these previous experiments, many individual decisions had to be made. For the lifetime statistics of all 10 players in a match, does a Random Forest model perform better by concatenating all 10 of each of the 10 players' statistics totaling 100 columns, or is it more efficient to sum each team's statistics together totaling 20 columns? Is clustering our players beneficial, and if so, how do numbers of clusters greater than $k = 5$ compare? Should we pull the player's per-map performance across all 9 maps, or should we only pull their stats for the current map being estimated? How do lifetime and per-map models improve when combined with the player's latent statistics such as recent performance and Elo? We answer these questions by performing a search of these combinations of decisions using random forests with 10,000 trees.

Model	Lifetime	Per-map
A: $k = 5$ clusters	50.4	51.84
B: $k = 10$ clusters	50.95	51.91
C: $k = 30$ clusters	50.4	52.59
D: Separate stats per player	51	54.38
E: Summed player stats per team	52.04	54.53
F: Adding map name	54.63	53.64
G: Latent stats only	53.88	
H: Adding separate latent stats	56.18	55.38
I: Adding latent stats summed per team	56.23	55.93
J: This map only		54.78
K: This map only with map name		54.86
L: This map only with latent stats		**57.72**
M: This map only with map name and latent stats		57.42

Table 4: Results of the random forest algorithm on different choices of input data.

In Table 4, the Lifetime column uses the data set implemented in Experiment 1, and the Per-map column uses the data from Experiment 2. In models A, B, and C, we compare the choices of 5, 10, and 30 clusters. We observe that increasing the number of clusters has a slightly positive impact, as more of our data is captured as we increase the number of clusters. Model D lists all of each player's stats per team in one row as was done in Experiments 1 and 2 above. Model E uses the same data as Model D but sums together these values for each team. This removes many of the nuances compared to considering each player separately, but results in a noticeable increase in performance in the lifetime stats compared to model D. This could be the result of the random forest's preference for fewer columns and more streamlined data as shown by higher performance on clustered lifetime data of Models A, B, and C compared to D as well. Model F uses the same data as Model D but with the addition of the map name, and this yields an increase of 4.53%. This indicates that our model is able to estimate how well players and teams of players perform on each of the given maps. There is a 0.74% decrease between Model D and Model F for Per-map statistics, but adding the name of the map in Model F is somewhat redundant since the data for these players on this map is already represented in the data of Model D. Model G is our result from Experiment 3. Model F showed that we can make match predictions using in-server statistics alone, and model G showed that we can make match predictions using latent stats alone. Model H combines the data used in Models F and G to achieve our highest model performance yet with 56.18% for Lifetime stats and 55.38% for Per-map stats. The data for Model H is a combination of the latent stats from Experiment 3 with the data from Experiments 1 and 2, and model I sums each team's stats from model H together.

Previously, with all map data present in the Per-map column of Models A through I, a player's performance on Dust 2 could hypothetically impact their predicted performance on Cache. Now, rather than using the players' Per-map statistics from all maps at once, we instead pull only the stats of each player on this specific map for Models J through M. So, if the map is Mirage, we pull this player's statistics from only Mirage and ignore all other maps. Models J and K use this version of the data to parallel Models D and F respectively, resulting in comparatively similar performance. Model L combines the single-map data used in Model J

with the latent statistics for that map used in Model H. Model M uses the same data and includes the name of the map as well, reaching a similar accuracy. Models L and M are our best-performing models, with Model L achieving 57.72% accuracy.

Our best model included each player's statistics for only the given map, which was an improvement over models which used the player's statistics across all 9 maps at once. This is less complete data, but it allows the algorithm to focus on only the most relevant information. This model does not incur information loss inherent in clustering, and includes both in-server performance as well as the player's overall skill ratings. Including the map label informing the model which map these statistics pertain to had virtually no impact on the model's accuracy. This indicates that with the data available to our model, having high performance and a better team composition is more relevant to a win than a particular combination of statistics on this specific map. We also note that in some cases, separating our data per player yields better results, while other times combining the team's data performs better. This is a limitation of the random forest model at our number of trees, where in some models such as the non-clustered model built in Experiment 2 (Model D Per-map), the number of trees is just over ten times the number of columns. We also observe that different numbers of clusters on different types of data yielded different results, and while these forests can run much faster than the others, they boast the lowest accuracy.

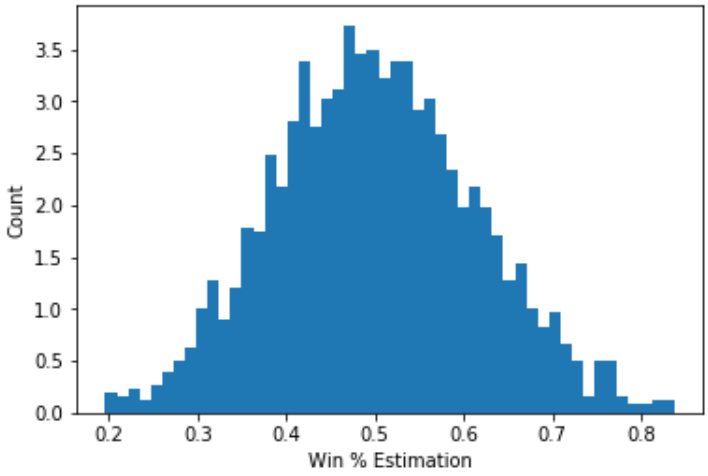

Figure 19: Distribution of estimations from our best model L.

To measure the performance of our best model, we must ensure that the expectations of our model follow a similar distribution to the true data. We observe the distribution of Model L predictions in Figure 19. In the set of cases where our model predicts X% chance to win, a distribution of the true results in this set of matches approaching X% indicates robustness of our model. By selecting all cases in which our model predicted $\geq$ 70% chance to win, we observe that 64% were truly wins. Selecting all cases where our model predicted $\leq$ 30% to win, 31.2% of these matches were a win. By repeating this process for all matches within a 5% range, we can compare the distributions of our model estimations to the true win percentage distributions in Figure 20. Estimations between 35% and 40% are very close to the true percentage of wins in this batch. We note that the estimations near 50% deviate from the ideal red line, which is due to grouping 45%-50% and 50%-55% into separate batches. The greatest deviations from the ideal result lie at the extreme ends near 30% and 70%, where our model estimations are slightly more modest in strongly one-sided matches. These predictions to win or to lose closely resemble the distributions of the true results, so our model accurately represents the expectations for a team to win or lose a match.

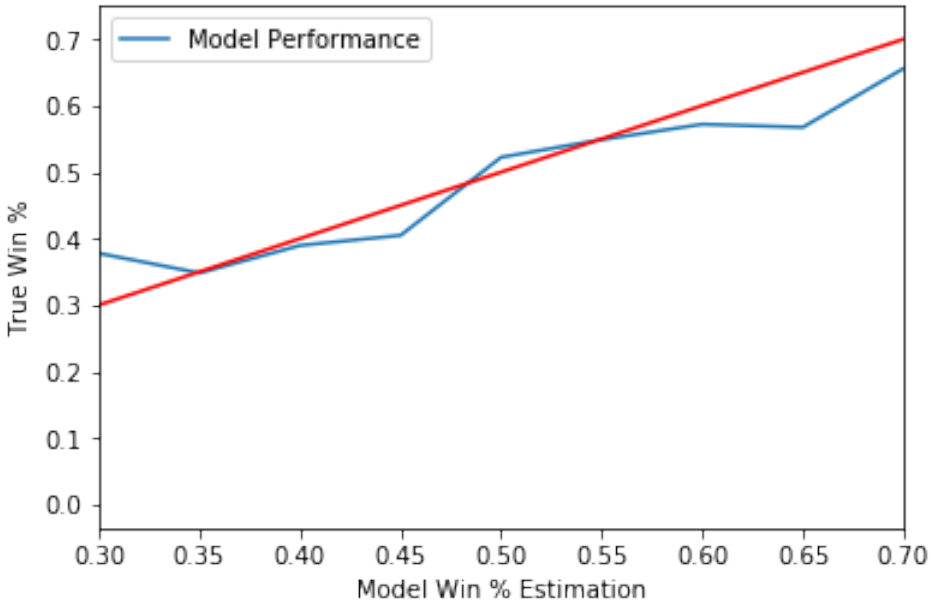

Figure 20: Closeness of our model prediction distribution to the true win percentage distribution at each batch of estimations. The blue line represents our model performance from each 5% batch, and the red line represents the ideal result.

8 Use Case

In this section we apply our decision support system to a real-world example. A Major is the most prestigious CS:GO tournament, and FACEIT hosted a Major in London in September 2018. Since this tournament was hosted by FACEIT, the data on these players and matches is available through their API. On September 23rd, Natus Vincere (NAVI) played against Astralis in the best-of-3 Major grand finals. The map veto went as follows:

Astralis bans Train
NAVI bans Cache
Astralis picks Nuke
NAVI picks Overpass
NAVI bans Dust2
Astralis bans Mirage
The decider is Inferno

Astralis won this series cementing one of the most dominant Major performances in history with a score of 16-6 on Nuke and 16-9 on Overpass. Armed with our model, could the map veto have gone better for NAVI? Our model outputs the percent chance that it believes NAVI will beat Astralis on each map. Intuitively, the dominant strategy for NAVI is to ban the maps with their lowest estimation to win, and to pick maps with their highest estimation to win. Of course, much more strategy is at play when experts plan their team's veto process, but our quantitative analysis may provide decision support that could inform and improve these experts in their approach to the veto.

For a true implementation, we would ideally re-train our model on pro team data only. However, there is not enough data from this one tournament to build a sturdy model, so we use the best pre-trained model built previously. First, we pull the IDs of the 5 players on each team. Using their IDs, we pull each of their individual map statistics for each of the seven maps in the map pool at the time of the major. Next, we run these 2 teams through our algorithm for each of the 7 maps. Our model win rate estimations for NAVI are displayed in Table 5.

Map	**Win Prediction**
Overpass	53 %
Nuke	50 %
Train	47 %
Dust2	46 %
Cache	45 %
Inferno	41 %
Mirage	38 %

Table 5: Win estimations for NAVI against Astralis in the FACEIT Major London 2018 Grand Finals.

So, our model would suggest the following map veto:

Astralis bans Overpass
NAVI bans Mirage
Astralis picks Inferno
NAVI picks Nuke
NAVI bans Cache
Astralis bans Train
The decider is Dust2

We now compare our model's map veto suggestions to the real veto. Our model estimates that NAVI has the greatest chance to win against Astralis on Overpass, and in the real veto, NAVI does select Overpass for their map pick. Cache is famously the map that NAVI never plays, making that their permanent ban. Our model did suggest that NAVI should ban Cache, and NAVI first-banned it in the real veto. We estimate that Nuke has 50/50 odds, and Astralis were the ones to select this map in reality. Train is suggested as a good map for Astralis to ban as well, and Astralis do ban this map in the first round. Our model performs very closely to the true veto until we reach the choice for the deciding map. NAVI have an 8% higher predicted chance of winning on Dust2 than on Mirage, so banning Mirage instead of Dust2 may have given NAVI a more favorable deciding map. The win prediction for Astralis between Mirage and Inferno is only 3%, so in such a small margin, this decision is much better left to the players and analysts.

The picks and bans suggested by our model correspond rather closely with the true map veto that occurred, indicating that our quantitative model performs similarly to the qualitative predictions of human experts. Including the detail that NAVI will first-ban Cache no matter what, our model suggests a veto even closer to the true veto. Unfortunately for NAVI, our model could not have suggested a potentially game-winning alternative veto, but it does advise to Astralis that picking Mirage as the decider could be stronger than picking Inferno. Given the closeness to the decisions made in the original veto, we conclude that our model is effective in predicting per-map results. In this demonstration of its advisory capabilities, we show that our model is effective in supporting the map veto decisions of players, coaches, and analysts.

9 Conclusion

In this thesis, we developed a decision support system which estimates a CS:GO team's chance to win against an opponent across each of the 7 maps in order to advise professional teams in their map veto. Our best-performing model used the lifetime statistics of each player on each team on the specified map and achieved an overall accuracy of 57.42%. We discussed that accuracy is not indicative of model performance as many of our win estimations are expected to be near 50%, so we showed that our model achieved an accuracy near 70% in the set of cases where our estimations of a win or loss were near 70%. By demonstrating the use-case of our model, we showcased the accuracy and effectiveness of the decision support this system provides to experts.

We discussed the state of the art in esports data, machine learning in esports, and the ex-

tents to which esports analytics are being implemented by CS:GO teams. We compared the methods and insights in esports analytics to their potential application in sports data. We collected and curated game data from the FACEIT CS:GO Data API, clustered players based on performance, modelled team matchups, and predicted the outcomes of matches based on the map. We showed that a number of features can be collected and curated from non-demo data. We found that higher headshot percentages do not correlate with more kills or wins. Through machine learning, we were able to model players' playstyles, how well different combinations of playstyles work together on a team, and how one team's combination of playstyles performs against another. We showed that combining the player's in-server performance statistics with their latent statistics such as win percentage and Elo contribute to higher model accuracy. This proves that our model does not simply predict that the higher Elo team will win, and instead it models player performance and team cohesion in its decision making.

There are many promising avenues for improvement upon this thesis. A large volume of data from professional CS:GO teams would improve the performance and effectiveness of this decision support system dramatically, but currently no publicly available data source exists. Players and experts have first-hand understandings of the differences between maps, but very little academic research exists which qualitatively or quantitatively defines these map differences. Aside from these industry-wide impediments, we list improvements specific to this thesis below:

- **Demo files:** This work showcased what can be accomplished immediately using pre-processed statistics. With a large number of demo files, the number of features that can be extracted regarding performance, playstyle, and matches is significantly greater.
- **Data source:** FACEIT provides an interesting and large dataset for free, but data from the matches of professional teams is more valuable to this project design. Professional teams are always playing with the same 5 players, against top-level opponents doing the same.
- **Data size:** More data improves the performance and robustness of machine learning models, so downloading more match and player statistics would result in stronger models.
- **Additional models:** Implementing different and more complex models than random forests could yield even higher accuracies.
- **Model performance measurement:** Deeper statistical investigation of the confidence of our model results could better reveal the robustness of our results.
- **Time series:** The skill of players and teams changes by the day. Our method uses lifetime statistics and a few metrics regarding recent form, but recent performance is much more significant than older data.
- **Other games:** Given relatively similar player data and match data from other esports or from traditional sports, this system's pipeline could be used to model player playstyles, how well different combinations of playstyles work together on a team, and how one team's combination of playstyles performs against another. Additionally, the per-map aspect of this thesis could be used to model a sports team's performance differences between home games and away games for example.

References

[1] https://www.esportsearnings.com/games

[2] https://www.esportsearnings.com/history/2019/games

[3] https://steamcharts.com/app/730All

[4] https://i.redd.it/qfo2xaiw0b451.png

[5] https://www.pinnacle.com/Cms_Data/Contents/Guest/Media/esports2017/Article-Images/CSGO/2019/Evergreen/2019-CSGO-Betting-Knowledge-Maps/inarticle-map-knowledge-nuke.jpg

[6] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. 2009. Elo Rating System. Alpha Press.

[7] Schölkopf, B., Platt, J., Hofmann, T. (2007). TrueSkill™: A Bayesian Skill Rating System.

[8] Nikolakaki S. M., Dibie O., Beirami A., Peterson N., Aghdaie N., Zaman K. : Competitive Balance in Team Sports Games

[9] Nama, B. 2019. CS:GO Player Skill Prediction

[10] Björklund A., Lindevall F., Svensson P., Visuri W. J., 2018. Predicting the outcome of CS:GO games using machine learning

[11] Turvey, M. T. (1990). Coordination. American Psychologist, 45(8), 938–953. https://doi.org/10.1037/0003-066X.45.8.938

[12] de Poel H. J. (2016). Anisotropy and Antagonism in the Coupling of Two Oscillators: Concepts and Applications for Between-Person Coordination. Frontiers in psychology, 7, 1947. https://doi.org/10.3389/fpsyg.2016.01947

[13] Vesper, C., van der Wel, R. P., Knoblich, G., Sebanz, N. (2011). Making oneself predictable: Reduced temporal variability facilitates joint action coordination. Experimental brain research, 211(3-4), 517-530.

[14] Bartlett, R., Button, C., Robins, M., Dutt-Mazumder, A., Kennedy, G. (2012). Analysing team coordination patterns from player movement trajectories in soccer: methodological considerations. International Journal of Performance Analysis in Sport, 12(2), 398-424.

[15] Passos, P., Davids, K., Araújo, D., Paz, N., Minguéns, J., Mendes, J. (2011). Networks as a novel tool for studying team ball sports as complex social systems. Journal of Science and Medicine in Sport, 14(2), 170-176.

[16] Balague, N., Torrents, C., Hristovski, R. et al. Overview of complex systems in sport. J

Syst Sci Complex 26, 4–13 (2013). https://doi.org/10.1007/s11424-013-2285-0

[17] Vilar, L., Araújo, D., Davids, K. et al. The Role of Ecological Dynamics in Analysing Performance in Team Sports. Sports Med 42, 1–10 (2012). https://doi.org/10.2165/11596520-000000000-00000

[18] Bruno Travassos, Keith Davids, Duarte Araújo T. Pedro Esteves (2013) Performance analysis in team sports: Advances from an Ecological Dynamics approach, International Journal of Performance Analysis in Sport, 13:1, 83-95, DOI: 10.1080/24748668.2013.11868633

[19] https://leetify.com/

[20] https://csgo.ai/

[21] Greg Brockman, Brooke Chan, Przemyslaw Debiak, Christy Dennison, David Farhi, Rafal Józefowicz, Jakub Pachocki, Michael Petrov, Henrique Pondé, Jonathan Raiman, Szymon Sidor, Jie Tang, Filip Wolski, and Susan Zhang. OpenAI Five, Jun 2018.
URL https://blog.openai.com/openai-five/.

[22] https://github.com/ValveSoftware/csgo-demoinfo

[23] https://github.com/markus-wa/demoinfocs-golang

[24] https://www.hltv.org/stats/players/7998/s1mple

[25] https://blog.faceit.com/launching-the-faceit-developer-portal-5740fb48ac26

[26] https://github.com/LaughingLove/faceit_api.py

[27] https://www.pinnacle.com/en/esports-hub/betting-articles/cs-go/introduction-to-the-csgo-map-pool/ryw24drkxby7m275

[28] Bowerman, Maurice S. "Monitoring a sports draft based on a need of a sports team and the best available player to meet that need." U.S. Patent No. 8,485,876. 16 Jul. 2013.

[29] Buchheit, M., Simpson, B. M. (2017). Player-tracking technology: half-full or half-empty glass?. International journal of sports physiology and performance, 12(s2), S2-35.

[30] Wisbey, B., Montgomery, P. G., Pyne, D. B., Rattray, B. (2010). Quantifying movement demands of AFL football using GPS tracking. Journal of science and Medicine in Sport, 13(5), 531-536.

[31] Xu, M., Orwell, J., Lowey, L., Thirde, D. (2005). Architecture and algorithms for tracking football players with multiple cameras. IEE Proceedings-Vision, Image and Signal Processing, 152(2), 232-241.

[32] Edgecomb, S. J., Norton, K. I. (2006). Comparison of global positioning and computer-

based tracking systems for measuring player movement distance during Australian football. Journal of science and Medicine in Sport, 9(1-2), 25-32.

[33] Diaz, K. M., Krupka, D. J., Chang, M. J., Peacock, J., Ma, Y., Goldsmith, J., ... Davidson, K. W. (2015). Fitbit®: An accurate and reliable device for wireless physical activity tracking. International journal of cardiology, 185, 138-140.

[34] Sprague, R. H., Watson, H. J. (Eds.). (1986). Decision support systems: putting theory into practice. Englewood Cliffs, NJ: Prentice-Hall.

[35] Bonczek, R. H., Holsapple, C. W., Whinston, A. B. (2014). Foundations of decision support systems. Academic Press.

[36] Turban, E. (1993). Decision support and expert systems: management support systems. Prentice Hall PTR.

[37] Turban, E., Sharda, R., Delen, D. (2010). Decision support and business intelligence systems (required). Google Scholar.

[38] Berner, E. S. (2007). Clinical decision support systems (Vol. 233). New York: Springer Science+ Business Media, LLC.

[39] Kawamoto, K., Houlihan, C. A., Balas, E. A., Lobach, D. F. (2005). Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. Bmj, 330(7494), 765.

[40] Sim, I., Gorman, P., Greenes, R. A., Haynes, R. B., Kaplan, B., Lehmann, H., Tang, P. C. (2001). Clinical decision support systems for the practice of evidence-based medicine. Journal of the American Medical Informatics Association, 8(6), 527-534.

[41] Hunt, D. L., Haynes, R. B., Hanna, S. E., Smith, K. (1998). Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review. Jama, 280(15), 1339-1346.

[42] Luo, Y., Liu, K., Davis, D. N. (2002). A multi-agent decision support system for stock trading. IEEE network, 16(1), 20-27.

[43] Gottschlich, J., Hinz, O. (2014). A decision support system for stock investment recommendations using collective wisdom. Decision support systems, 59, 52-62.

[44] Kuo, R. J., Chen, C. H., Hwang, Y. C. (2001). An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. Fuzzy sets and systems, 118(1), 21-45.

[45] Robertson, S., Bartlett, J. D., Gastin, P. B. (2017). Red, amber, or green? Athlete monitoring in team sport: the need for decision-support systems. International journal of sports physiology and performance, 12(s2), S2-73.

[46] Xing-bin, C. D. W. Z. (2009). The Sports Training Decision Support System Based on Data Mining [J]. Microcomputer Information, 12.

[47] Yu, L., Ling, P., Zhang, H. (2010, December). Study on the decision support system of techniques and tactics in net sports and the application in beijing olympic games. In 2010 Second WRI Global Congress on Intelligent Systems (Vol. 1, pp. 170-174). IEEE.

[48] Bonner, J., Woodward, C. J. (2012, November). On domain-specific decision support systems for e-sports strategy games. In Proceedings of the 24th Australian Computer-Human Interaction Conference (pp. 42-51).

[49] Cantinotti, M., Ladouceur, R., Jacques, C. (2004). Sports betting: Can gamblers beat randomness?. Psychology of addictive behaviors, 18(2), 143.

[50] Gray, P. K., Gray, S. F. (1997). Testing market efficiency: Evidence from the NFL sports betting market. The Journal of Finance, 52(4), 1725-1737.

[51] LaPlante, D. A., Schumann, A., LaBrie, R. A., Shaffer, H. J. (2008). Population trends in Internet sports gambling. Computers in Human Behavior, 24(5), 2399-2414.

[52] Owens Jr, M. D. (2016). What's in a name? eSports, betting, and gaming law. Gaming Law Review and Economics, 20(7), 567-570.

[53] Grove, C., Krejcik, A. (2015). eSports betting: It's real, and bigger than you think. Anaheim, CA: Eilers Research.

[54] Griffiths, M. D. (2017). The psychosocial impact of professional gambling, professional video gaming eSports. Casino Gaming International, 28, 59-63.

[55] Dos Reis, V. (2017). QA: The rise of esports betting and the challenges the industry faces. Gaming Law Review, 21(8), 630-633.

[56] Holden, J. T., Ehrlich, S. C. (2017). Esports, skins betting, and wire fraud vulnerability. Gaming Law Review, 21(8), 566-574.

[57] Esq, S. C., Smith, J. (2017). eSports Betting The Past and Future.

[58] Greer, N., Rockloff, M., Browne, M., Hing, N., King, D. L. (2019). Esports Betting and Skin Gambling: A Brief History. Journal of Gambling Issues, (43).

[59] Helsen, W. F., Starkes, J. L., Hodges, N. J. (1998). Team sports and the theory of deliberate practice. Journal of Sport and Exercise psychology, 20(1), 12-34.

[60] Grehaigne, J. F., Godbout, P., Bouthier, D. (1997). Performance assessment in team sports. Journal of teaching in Physical Education, 16(4), 500-516.

[61] Gréhaigne, J. F., Godbout, P. (1995). Tactical knowledge in team sports from a constructivist and cognitivist perspective. Quest, 47(4), 490-505.

[62] Lusher, D., Robins, G., Kremer, P. (2010). The application of social network analysis to team sports. Measurement in physical education and exercise science, 14(4), 211-224.

[63] Passos, P., Davids, K., Araújo, D., Paz, N., Minguéns, J., Mendes, J. (2011). Networks as a novel tool for studying team ball sports as complex social systems. Journal of Science and Medicine in Sport, 14(2), 170-176.

[64] Lee, D., Schoenstedt, L. J. (2011). Comparison of eSports and traditional sports consumption motives. ICHPER-SD Journal Of Research, 6(2), 39-44.

[65] Hallmann, K., Giel, T. (2018). eSports–Competitive sports or recreational activity?. Sport management review, 21(1), 14-20.

[66] https://medium.com/dreamteam-media/the-three-longest-cs-go-maps-ever-9901474a83d1

[67] Dong, J. S., Shi, L., Jiang, K., Sun, J. (2015, December). Sports strategy analytics using probabilistic reasoning. In 2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS) (pp. 182-185). IEEE.

[68] Min, B., Kim, J., Choe, C., Eom, H., McKay, R. B. (2008). A compound framework for sports results prediction: A football case study. Knowledge-Based Systems, 21(7), 551-562.