FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Predicting results of e-sports matches with matrix completion methods
Student:	Bc. Tomáš Hampl
Supervisor:	Rodrigo Augusto da Silva Alves, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

The popularity of e-sports events has been steadily increasing in recent years. For instance, the consistently rising number of professional players and leagues calls the attention of bet portals and enthusiasts. The aim of this project is to:
(1) perform a comprehensive revision of the literature on the sport and e-sport prediction learning methods;
(2) collect the data and create at least one e-sport dataset that consists of historical data of e-sport match results;
then (3) design and implement a model based on matrix completion embedding that predicts results of e-sport matches based on historical information;
moreover, (4) implement and execute baselines based found on (1) and compare the accuracy prediction with the implemented model;
finally, (5) discuss the results and analyze possible future directions.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 27 December 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Predicting results of e-sports matches with matrix completion methods

Bc. Tomáš Hampl

Department of Applied Mathematics
Supervisor: Rodrigo Augusto da Silva Alves, Ph.D.

February 15, 2024

Acknowledgements

I would like to thank my supervisor, Rodrigo Augusto da Silva Alves, Ph.D., for his valuable advice and time that he devoted to helping me with the implementation of my diploma thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on February 15, 2024

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Hampl, Tomáš. *Predicting results of e-sports matches with matrix completion methods*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Abstrakt

Obliba e-sportovních akcí v posledních letech neustále roste. To motivuje potřebu analyzovat a předpovídat profesionální zápasy v e-sportech. V této diplomové práci je navrhnut a implementován model založený na maticovém doplňování pro predikci výsledků e-sport zápasů, zejména pro hru Counter-Strike: Global Offensive. Je provedena podrobná rešerše, dle které jsou implementovány metody pro porovnání úspěšnosti našeho modelu. Úspěšnost metody je vyhodnocena z hlediska přesnosti předpovídání výsledků zápasů.

Klíčová slova maticová faktorizace, maticové doplňování, Counter-Strike: Global Offensive, e-sport, predikce zápasů, SVD, ALS

Abstract

The popularity of e-sports events has been growing steadily in recent years. This popularity motivates the need to analyse and predict professional e-sports matches. Within this thesis, a model based on matrix completion is proposed and implemented to predict the results of e-sport matches, especially for the game Counter-Strike: Global Offensive. Detailed research is carried out, according to which the methods for comparing the success of our model are implemented. The method's success is evaluated in terms of accuracy in predicting match results.

Keywords matrix factorization, matrix completion, Counter-Strike: Global Offensive, e-sports, match prediction, SVD, ALS

Contents

List of Figures

List of Tables

Introduction

In recent years, the realm of e-sports has emerged as a dynamic and rapidly evolving domain, captivating millions of enthusiasts globally. Among these competitive gaming landscapes, Counter-Strike: Global Offensive stands out as one of the most popular first-person shooter games, fostering a highly competitive environment that attracts professional players, teams, organisations, and a dedicated fan base. As the e-sports ecosystem continues to thrive, there is an increasing demand for advanced analytical tools and predictive models to enhance the understanding and anticipation of match outcomes.

This development brought much attention to the research endeavours of organisations already participating in e-sports and newly interested third-party companies that are interested in the other aspects of e-sports, like betting. Though a burgeoning field, e-sports analytics has grown substantially in recent years. Traditional sports have long embraced statistical analysis and predictive modelling to gain insights into player performance and match dynamics. Similarly, the application of advanced analytics in e-sports has the potential to enhance the viewing experience for fans and provide teams and players with valuable insights that can influence strategic decision-making. One of the most popular analytical tools recently published is Leetify, where, through a paid subscription, its users can view the statistical analysis of their matches.

Diverse models, including linear regression, decision trees, random forests, neural networks, and others, are employed to predict the outcomes of matches by analysing extracted features from the data. The main objective of this thesis is to conduct experiments on matrix completion approaches and assess their effectiveness in predicting match results for a complicated problem compared to the current state of the art. Counter-Strike: Global Offensive, known for its complex gaming mechanics, diverse team strategies, and exceptional individual player abilities, presents a distinctive obstacle for predictive modelling. The game's multifactorial nature encompasses the interaction of in-game variables, team dynamics, and tactical decision-making, making typical analytical approaches less efficient.

The primary objective of this thesis is to conduct a comparative analysis of matrix completion models and approaches with the existing state-of-the-art methods employed in e-sport forecasting. In order to acquire knowledge of the state-of-the-art techniques, it is necessary to conduct a thorough revision of the existing literature. It is necessary to gather historical data on Counter-Strike:

Global Offensive e-sport matches and tournaments. The objective is to create a robust e-sports dataset, offering a comprehensive collection that encompasses previous match performances. This dataset will serve as the cornerstone for subsequent modelling, predictive analysis, and experimentation. In order to gauge the effectiveness of the proposed model, baseline models will be implemented based on the findings from the literature review. This process enables a thorough assessment of the model using selected metrics. The last stage entails an examination of the outcomes derived from the model and comparisons with the baselines. Furthermore, the thesis will contemplate potential avenues for future research and development in the realm of e-sport forecasting.

Chapter 1

Literature revision

In this chapter, the literature revision is presented. The review is devoted to predicting the outcomes of matches in sports and e-sports. Contextually similar articles are presented, described, and compared to our approach.

Previous research papers have studied a lot of popular video game genres, such as MOBA, which stands for multiplayer online battle arena, where the researchers found success in predicting game winners. Many publications covered real-time prediction from the current game state, which was especially popular in strategy games like Dota 2 and League of Legends. Predicting the winner of the match from a time stamp of a match is not part of the contents of this diploma thesis.

Counter-Strike: Global Offensive (hereinafter CSGO) is an e-sport that is unique in its game-based nature around rounds, adding up to the final game result. This way, researchers can study both the long-term effects of decisions made by players and the short-term effects.

The great benefit of CSGO is that, as a computer video game, all matches are recorded by the server on which the match is played. This enables the creation of statistical analysis tools and the usage of machine learning to predict the outcome of the matches.

To appreciate the significance of our inquiry, it is imperative to situate it within the broader context of predicting matches, not only in e-sports. The primary objective of this chapter is to describe all kinds of approaches used to solve this problem without and within our context. Each method will be picked and described only once with the data used and the results acquired. Some of these methods will then be selected as baseline methods to create a comparison between the results of our experiments and our final model.

Predicting NBA Games with Matrix Factorization This diploma thesis is mostly inspired by Tuan Tran's 2016 thesis [8] from the Massachusetts Institute of Technology. The objective of this thesis is to expand the application of matrix factorization in order to predict the outcomes of the NBA playoffs. The thesis is influenced by FunkSVD, a collaborative filtering algorithm developed by Simun Funk during the Netflix Prize competition. The author creates latent vectors to incorporate both offensive and defensive components of the team's performance. The result is derived by computing the dot product of the offensive latent vector of one team and the defensive latent vector of the other

team. The model attained a long-term prediction accuracy of 55% for the time period spanning from 2000 to 2015 and a short-term prediction accuracy of 65% for the time period from 2014 to 2015.

A Two-Stage Real-time Prediction Method for Multiplayer Shooting E-sports This article [9], authored by Jiaxin Liu et al. from Sichuan University, focuses on predicting the ranking of players in the game Players Unknown Battlegrounds. PUBG is a multiplayer shooter game that takes inspiration from Hunger Games, with the primary goal being to be the last person standing. Upon its release, this game achieved significant acclaim and has sustained its popularity ever since. The primary aim of this article is to accurately predict the rankings of players in the game, taking into account the particular parameters of the equipment provided and the starting landing location. The authors utilised the random forest methodology and attained a 90.84% accuracy rate.

Predicting Round and Game Winners in CSGO In this article [10], Allen Rubin focuses on predicting both the victors of individual rounds by analysing the short-term decisions of the current state of the match, as well as the overall match by examining the historical data. The author retrieves the data from demo files, which are recordings made by the server during the match. The author extracted features such as economic aspects, equipment aspects, and the current score. The data were obtained from demo files within a small and stable time window, comprising 45 matches. The author employed the random forest methodology as the primary classifier, while Multi-layer Perceptron and XGBoost were utilised for comparative purposes. The random forest classifier achieved a 64% accuracy in predicting the round winner, which is higher than the 59.7% accuracy of XGBoost and the 61% accuracy of MLP. In predicting the match winner, the random forest classifier achieved an accuracy of 58.9%, the XGBoost achieved 62.4% accuracy, and the MLP achieved 52%.

Predicting the outcome of CS:GO games using machine learning Arvid Björklund et al. authored this thesis that utilises machine learning to forecast the results of CSGO matches [11]. They gathered a dataset of 6000 matches from FaceIt[1], focusing on the top 1000 ranked players in Europe. The purpose was to analyse their play style and selected characteristics. Based on the analysis, the players were clustered together, and a neural network was trained to predict the outcome of the matches based on the players' abilities alone. While intriguing, this novelty does not depict the competitive nature of CSGO, as it emphasises individual player performance. The final model demonstrated an accuracy of 65%, providing significant evidence that the individual talent of the players must be factored in the selected features in the data for our model.

Predicting Winning Team and Probabilistic Ratings in "Dota 2" and "Counter-Strike: Global Offensive" Video Games This article [12], authored by Ilya Makarov et al., utilises TrueSkill ratings to evaluate the

[1]FaceIt is a third-party organisation where professional players engage in matches against each other in order to enhance their individual abilities.

performance of individual players in teams. These ratings are determined by analysing various in-game events like kills, deaths, player movement, and view orientations. Most of the selected features pertain to situations after the bomb has been planted, specifically in relation to achieving the round objective. The post-plant features include characteristics like the count of active players, the players' health status, and similar aspects. The dataset was constructed via 162 demo files. The Bayesian rating model achieved an accuracy of 62%.

Chapter 2

Counter-Strike: Global Offensive

In this chapter, the basic concepts and principles that were used and are related to this thesis's subject are presented. The first section is devoted to the introduction of e-sports. Within this section, we will introduce what it entails, the history and the popularity behind it. In the following section, we delve into the introduction of the subject of this diploma thesis, Counter-Strike: Global Offensive. Within this section, we will provide an introduction to the game and explain its origins, gameplay, and important game rules. These rules include player roles, equipment, economy, economic strategies, map design, and other relevant information related to professional competitive play.

2.1 E-sports

Electronic sports, also known as e-sports, refers to the world of competitive and organised video gaming where individuals or teams (professional and amateur) compete against each other in various video games. These competitions can range from local tournaments for anyone to organised professional tournaments with substantial prize money, large audiences, and sponsorship deals at stake.

E-sports encompass a wide variety of video games with players competing against each other in real-time strategy (RTS), first-person shooter (FPS), multiplayer online battle arena (MOBA), sports simulation, and fighting games, among others. The most popular e-sports include Dota 2, Counter-Strike, League of Legends, and Fortnite. These games are played on all platforms, including PCs, consoles, and mobile devices.

The e-sports industry has experienced explosive growth in recent years, with millions of people watching professional e-sports events online or in person. E-sports events can attract large crowds, with some events filling stadiums and arenas with tens of thousands of spectators. The spectator's behaviour can be compared to any regular sport with a great fanbase. E-sports has also become a popular form of entertainment on streaming platforms such as Twitch and YouTube. Professional players often start streaming on these platforms after their careers end.

E-sports has developed into a professional industry, with players, coaches, and teams earning significant salaries and sponsorships. In addition, dedicated e-sports organisations, leagues, and governing bodies oversee and regulate the industry, creating a stable environment for the thriving competitive community.

Figure 2.1: Intel Extreme Masters Rio tournament, picture taken by Adela Sznajder [1].

One of the most important moments in the development of e-sports was in 1999 when two of the biggest e-sports titles were created - first-person shooter Counter-Strike and real-time sci-fi strategy Starcraft. The popularity of Starcraft was so high that professional matches were broadcast on a paid television channel called OGN (short for OnGameNet)[13]. For comparison, professional players reached celebrity statuses like Lionel Messi or Cristiano Ronaldo for football fans. To demonstrate the fan based on the Intel Extreme Masters tournament in Rio, see 2.1.

2.2 Counter Strike: Global Offensive

Counter-Strike: Global Offensive is a multiplayer tactical first-person shooter video game. Although the gameplay is based on shooting the opponents, trying to kill them, and looking from a first-person perspective, the game also provides a significant strategic and tactical aspect of teamplay and situational executions.

CSGO is an objective-based game, where two teams of Terrorists (T-side) and Counter-terrorists (CT-side) face each other with their respective objectives. The Terrorists are trying to eliminate all of the Counter-terrorists or plant a bomb that has to explode. The Counter-terrorists are trying to eliminate all of the Terrorists, preventing them from planting the bomb or denying the terrorists from planting the bomb and surviving the round time. The objectives describe the gameplay of the teams, where the Terrorists are the attackers and the Counter-terrorists are the defenders.

2.2.1 History of CSGO

The original Counter-Strike game from the game series was launched in 1999 as a modification for Half-Life, devised by Minh Le and Jess Cliffe [14]. Valve acquired the intellectual property from the authors after recognising the mod's success. The initial official release of the standalone game, featuring improved graphics, new maps, and refined mechanics, took place in November 2000. This event was the initial major milestone in the game's history. Counter-Strike rapidly gained immense popularity, particularly within the growing e-sports scene. As of July 2001, the game had sold over 250 thousand units; by February 2003, it had sold 1,5 million units. To this day, around 15,000 players play the original game every day [15].

Valve launched Counter-Strike: Source in 2004, including an upgraded engine that brought enhanced graphics and physics. The player base reacted unfavourably to these modifications, leading to a significant number of players opting to stick with Counter-Strike. Consequently, the competitive Counter-Strike scene was predominantly centred around this game. The Counter-Strike: Source has a more limited community in terms of size. The primary issues encompassed bigger hitboxes of the player models, erratic spray patterns, and alterations to physics that diminished the game's skill ceiling, thereby catering to novice players while compromising the game's competitive nature. In order to better understand the consequences of these modifications, it is worth noting that the prize pools for Counter-Strike amounted to $13,595,060, whereas Counter-Strike: Source amounted to $3,113,222 [16].

The subsequent major milestone in the history of Counter-Strike occurred with the launch of Counter-Strike: Global Offensive in August 2012. The primary objective behind the release of CSGO was to expand its availability across other platforms, as evidenced by image 2.2, which depicts a buy menu that was supposed to support controllers and controls for controllers. Due to these factors, CSGO was developed as a means for console players to access the game Counter-Strike: Source. This made some players hesitant to switch between Counter-Strike and Counter-Strike: Source. Fortunately, Valve saw the e-sports potential of the game and focused on improving the movement, gunplay, and hitboxes to make the game more competitive in subsequent versions. In 2013, Valve organised its first Major tournament, DreamHack Winter 2013 [17], with a prize pool of $250,000. This is the point at which the e-sports industry began to expand rapidly. The majority of players who were previously engaged in Counter-Strike and Counter-Strike: Source have transitioned to playing CSGO. As of December 2015, the game boasted a player base of 800,000 individuals. The number of players continues to increase, reaching its highest point in March 2023 with 1,5 million users playing simultaneously [18].

2.2.2 Gameplay

As previously mentioned, CSGO involves two teams of five players competing on a map, with each team having specific objectives tied to the map. In competitive play, whether at an amateur or professional level, the primary objective for Terrorists is to plant a bomb at a predetermined bomb site, typically labelled A or B on the map. Their objective is to safeguard the bomb until the bomb detonates. On the contrary, the Counter-terrorists strive to prevent the

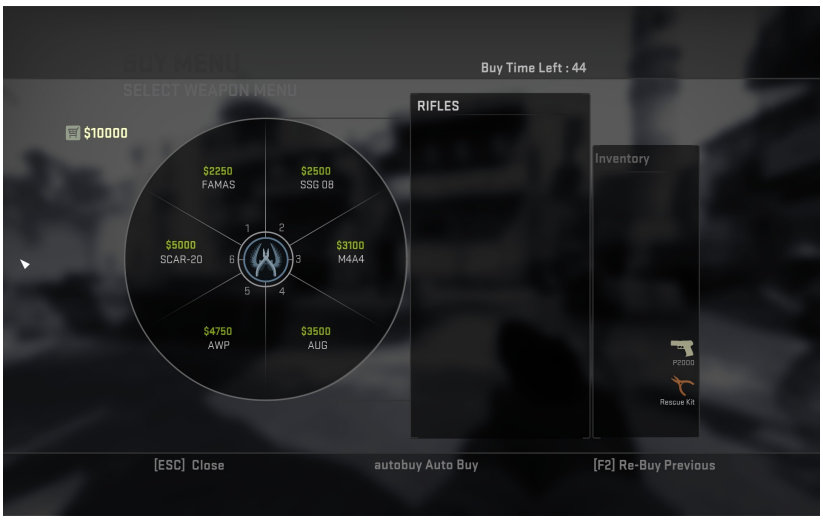

Figure 2.2: CSGO buy menu. Taken from [2].

bomb from being planted or, if it has been planted, defuse the bomb before it explodes.

Within the map's context, a match consists of a series of up to 30 rounds. At halftime, which occurs after 15 rounds, the players switch sides (Terrorists to Counter-terrorists and vice versa). The team that wins the majority of the 30 rounds, precisely 16 rounds, is declared the winner of the match. In the case of a 15-15 score, six additional rounds are played, called Overtime, which repeats itself until the winner is found. This is not true in casual play, where a score of 15-15 results in a draw.

The game always starts with both teams spawning at their spawn locations (usually on the opposite side of the map with a similar distance to the objective). There is a freeze time when nobody can move, and everyone can buy equipment. The buy time ends within 15 seconds, and the round officially starts. Counter-terrorists tend to defend their objectives, and Terrorists are trying to attack them. This behaviour is not always accurate because, in the context of hostage rescue[2], the Counter-Terrorists are the aggressors. By default, if the round time runs out, the winner is the team that successfully defends the objective (Counter-terrorists in the context of bomb defusal and Terrorists in the context of hostage rescue).

2.2.3 Player roles

The strategic aspect of the CSGO and the main objectives are based on team play. Players resolve to have different roles assigned based on their skill sets. This role can change based on the map and other situations. The most common roles are: support, AWPer, lurker, entry fragger, in-game leader, and rifler.

[2]Hostage rescue is a game mode, where the Counter-Terrorists are responsible for extracting a hostage.

Support

The support role is vital for team-oriented gameplay and fulfils its name's literal meaning. This role's primary responsibility is to support the entry fragger. The players with this role usually follow the entry fragger looking for trade kills to gain control of some part of a map or provide utility support for the entry fragger by throwing well-timed flashbangs or smoke grenades so that the entry fragger has an advantage over the defending player.

Every player, to varying extents, adopts this role, even if it is only as a secondary role. The crucial aspect of this role is the profound comprehension of the game that it requires.

AWPer

The role is named after the AWP, which is a high-risk, high-reward sniper rifle with the ability to kill enemy players with one hit. The majority of teams have a single designated player who assumes this role. Certain organisations build their entire team around one of these players, e.g. Natus Vincere, who built the team around S1mple. The players who assume this role possess exceptional skills and have the ability to influence the outcome of the match in their favour on their own. This function holds significant importance and presents considerable challenges.

Lurker

The lurker typically seeks vulnerabilities and targets isolated players who attempt to advance from their defensive positions to get behind the lurker's team or during rotations[3] to prevent player count advantages on the defender side. Typically, lurkers strategically operate near the opposing bomb site, aiming to maximise their impact by inflicting damage and impeding the enemy's ability to flank their team. For this position, having a deep understanding of maps, precise timing, and a thorough comprehension of the player's behaviour is essential.

Entry fragger

The entry fragger typically assumes the role of the vanguard, leading the charge towards the enemy position. The primary goal of this role is to eliminate the enemy players and/or create enough space for his team with the intention of providing as much information as possible to enable teammates to set up bomb site executions. This role carries significant risk with significant potential for reward. The role requires quick reflexes and excellent mechanical skills. Certain players in this role achieved legendary status, such as Adil "ScreaM" Benrlitom, who gained widespread recognition for his mechanical abilities, earning him the nickname "headshot machine". To put this into numbers, ScreaM has participated in 37066 rounds, achieving 27481 kills. His headshot ratio is an very impressive 68.1%. In comparison, the majority of players often have headshot ratios ranging from 20% to 40% [19].

[3]Rotation is the action of a player leaving one site to help the other. This action can create an advantage but also a disadvantage.

In-game leader

The IGL (short for in-game leader) takes on the role of team leader and holds authority over the majority of decision-making processes. These decisions usually encompass aspects such as team economics, strategy, rotations, pushing, and any other relevant factors that may arise during the match. The IGL typically assumes an additional function as a support player or rifler.

Rifler

The rifler serves as the fundamental support and backbone for all roles. This role represents the player's status before being moulded into one of the abovementioned roles. Occasionally, the role can refer to a player serving as a substitute for AWPer. This role closely resembles the role of the support or serves as a playmaker, with the player providing support for the entry fragger aggressive play style, primarily focusing on eliminating enemies and gaining control of the map.

2.2.4 Equipment

CSGO allows players to acquire a diverse range of weaponry and equipment through purchase [20]. The five primary weapon categories include melee weapons, pistols, heavy weapons, SMGs (abbreviated form of sub-machine guns), and rifles. The heavy category is divided into shotguns and machine guns. The rifle category is divided into assault rifles and sniper rifles.

The equipment is divided into grenades (utility) and equipment. The equipment includes a defuse kit, which expedites bomb defusing, armour that enhances protection, and a Taser.

The available grenades for purchase include HE grenades (short for high-explosive), flashbangs, smoke grenades, decoy grenades, and, depending on the side, Molotov cocktails (Terrorist side) or incendiary grenades (Counter-Terrorist side).

2.2.5 Economy

At the beginning of the match, every player receives $800 for equipment purchases [21]. After each round, the players are awarded money according to their performance in the previous round. The performance is evaluated by considering both the team's overall achievement of the main map objective and the individual's success in terms of kills and initiative in the map objectives.

The Counter-Terrorist team is rewarded with $3250 if they win the round either by eliminating the enemy or if the Terrorist team fails to complete their objective within the round time. If the Counter-Terrorist team successfully defuses the bomb, they will receive an additional $250 in addition to the base amount of $3250. The person who defused the explosive device will be awarded an additional $300. The terrorist team earns an equal amount of money for achieving victory through elimination. If the planted bomb detonates, the Terrorist team is rewarded with $3500. The person who planted the explosive device will be awarded an additional $300. In the event that the Terrorist team exceeds the round time, the surviving players will not receive any monetary

Round lost	Team
first	$1400
second	$1900
third	$2400
forth	$2900
fifth	$3400

Table 2.1: Evolution of the loss bonus by consecutive losses.

Weapon	Money award
sub-machine gun	$600
shotgun	$900
AWP	$100
knife	$1500
other	$300

Table 2.2: Money awarded for distinct weapon kills.

compensation. For the Counter-Terrorist team, this case is considered with the same rewards as for elimination victory.

Furthermore, in the event that the bomb is defused, the Terrorist team is awarded an additional $800. The maximum of money that each player can have is $16,000. Table 2.1 illustrates the correlation between consecutive losses and the corresponding increase in monetary compensation. Following a victory round, the loss bonus does not get nullified but begins to diminish in the same way it increases. For instance, if a team loses five consecutive rounds and wins a round, it will be awarded the money based on their win condition. At the same time, the players will not receive the loss bonus, and the amount for the following loss will diminish by one stage.

A variety of weapons results in varying rewards. The table 2.2 displays the monetary rewards associated with each weapon. The players frequently utilise this knowledge, particularly during safe rounds when the team is up against an eco-round, which is a round where the team conserves money and refrains from purchasing equipment. When a player kills a teammate, the player incurs a penalty of $3300.

2.2.6 Economic strategies

As previously said, CSGO incorporates a strategic aspect that necessitates player cooperation. The economic strategy is one of the crucial aspects that requires the team's cooperation, as the team's strength is determined by its least capable members. This situation pertains to the equipment rather than the player's ability. The inability to purchase equipment alongside the team can have disastrous consequences for both sides.

Economic strategies do not depend upon an individual player's wealth but rather the collective funds of the entire team. While providing equipment to the teammates is possible, this practice is only viable in the short term. Here are a few examples of the purchasing strategies:

- **Full buy** is an optimal scenario where the team can acquire all necessary equipment.

- **Half buy** is a scenario where certain players possess the funds to Full buy, while the remaining players make purchases based on the situation.
- **Force buy** is a scenario where the team operates with limited funds and decides to gamble by purchasing as much as possible to catch the other team off guard.
- **Eco round** is a scenario where the team retains most of their funds in this scenario. Occasionally, the team purchases pistols to inflict damage against the opposition, which would cause them to lose money.
- **Full save** is the worst scenario where the team becomes desperate and refrains from making purchases to save money.

2.2.7 Map design

Most Valve Active Duty Map Group maps are designed to provide balance and competitive gameplay. The maps usually revolve around two bomb sites, designated as A and B. The maps strive for symmetry to achieve a fair and balanced playing field for both teams. The critical element of the map design is ensuring equal opportunities to attack and defend, thereby minimising the inherent advantages of spawn locations.

An exemplary illustration of map design is Ancient. It is the third original map created by Valve, released in December 2020 [22]. This map was designed with a focus on professional competitive play. The response to the release of this map was predominantly positive. In 2021, the map faced criticism due to the release of agent skins, where some of these skins seamlessly blended into the background. Agent skins are typically forbidden in professional competitive play.

The design of Ancient follows the four-leaf clover design [23], such as Mirage and Dust2. The Mirage is one of the most popular maps in CSGO. The design typically consists of a central hub (mid) with four main paths extending outward, resembling the clover leaves. Every path represents a unique route, providing strategic opportunities. The map layout is depicted in figure 2.3.

2.2.8 Map selection

In competitive play, the matches are conducted using an official map pool consisting of preselected maps specifically designed for competitive play [24]. These maps tend to come from the Valve Active Duty Map Group, and every map has two bomb sites and spawn points on separate sides. A specific map-selection process is employed during tournaments and leagues for map selection, while the maps pool remains consistent for casual and professional play. The selection process commences with the banning phase, during which both teams are given the opportunity to prohibit the usage of specific maps. Following the banning phase, the teams proceed to the selection phase, where they choose the maps they want to play. When a team chooses the map, the opposing team has the choice to determine which role they wish to begin the game in, either as Terrorists or Counter-terrorists. The prevailing forms of matches in CSGO include Best of 1 (Bo1), Best of 3 (Bo3), and Best of 5 (Bo5).

Figure 2.3: Map layout of de_ancient. Taken from [3].

In casual play, the maps are not restricted to only bomb defusal scenarios, and players can play hostage rescue scenarios and other custom game modes that will be described later on.

2.2.9 Game modes

CSGO offers a variety of game modes for players to choose from [14]. Every mode possesses its own set of rules and victory conditions. To ensure comprehensiveness, we will enumerate and provide descriptions for all the available game modes.

Competitive

This game mode is widely considered the most significant and intense, as it provides unique ranking Skill Groups that players can acquire and showcase to other players. The rankings range from Silver I to Global Elite. It resembles other games and rating systems in which one gains elo[4] by winning and losing elo by defeats in the matches.

It closely resembles professional competitive play. This game mode is the classic game mode that is responsible for establishing Counter-Strike's presence in the realm of competitive gaming. Two teams of five players compete in a best-of-30 match, adhering to the standard competitive rules. Players must purchase armour, weapons, and defuse or rescue kits.

[4]Elo is a rating system used to assess players' skill.

Wingman

This game mode bears a resemblance to the competitive mode. However, it is played on more compact maps in two teams, with the winner being determined by the best out of 16 rounds. The rules and penalties remain identical to those in competitive mode. Wingman and competitive modes employ distinct rating systems.

Casual

The casual game option is designed for players who prefer a more relaxed and informal gaming experience. Engaging in the activity does not result in any form of compensation, and no negative consequences are associated. There are no repercussions for abandoning the match, either. This game mode serves as a less intense training tool for the competitive mode. Players are exempt from purchasing armour and defusal kits as the game automatically provides them to each player. The game style features a streamlined economy and turns off team damage. There is no restriction on the number of players on a team.

Deathmatch

The deathmatch mode is a fast-paced, informal game mode that enables immediate respawns and promotes conflict. Upon spawning, players are granted a finite duration of invulnerability to select weapons of their preference. The game mode is typically used as a practice tool and warm-up for other modes. The players are distributed randomly over the map, and the objective of the game mode is to accumulate the highest number of points among all players. Points are awarded to the player for eliminating opponents, where the specific weapon works as a multiplier. There are two variations of the game mode: free-for-all and team deathmatch.

Arms Race

The Arms Race mode is a game mode that is included in the war games category. It is derived from the Gun Game mode, which is renowned throughout the gaming industry. This game mode leans more towards the fun aspect than the competitive one. Each player begins the game equipped with the same weapon, and the aim of the game is to eliminate other players in order to progress through the levels. There are a total of 21 levels in the game, with the last stage being a golden knife. If a player wielding the golden knife eliminates an opponent, they are victorious.

Demolition

The Demolition mode combines elements of bomb scenarios and the Gun Game mode. Players face each other in two teams consisting of six players on a map with one bomb site. The game has ten rounds and no overtime. If both teams win five rounds, the game ends in a draw. At the beginning of the game mode, the players automatically receive a starting weapon and advance through a sequence of firearms by eliminating other players. As the player eliminates

other players, they progress towards more powerful weapons, providing their team an advantage.

Flying Scoutsman

The game mode belongs to the category of fun game modes and involves two teams consisting of 8 players. Players are limited to using either an SSG 08 sniper rifle or a knife. The game environment has reduced gravity, increasing midair acceleration, allowing players almost to fly. The removal of accuracy penalties further diminishes the realism of the gameplay. The primary objective in this game mode is typically player elimination, while certain maps feature a bombsite where the conventional bomb scenario occurs.

Retakes

The retake game mode was introduced to CSGO in late 2020 [25]. This game mode focuses on practising post-plant scenarios where Counter-terrorist attempt to eliminate Terrorists and defuse the bomb. The game mode allows a maximum of three players for the Terrorists team and four for the Counter-terrorists team. The team that wins eight rounds is declared the winner of the match. At the beginning of each round, each player has the option to select a loadout of their choosing for that particular round.

Danger Zone

The Danger Zone game mode is a Battle Royale mode introduced to the CSGO in late 2018 [26]. A maximum of 18 players can join a match in this game mode. The ultimate victor is the sole survivor. As in a typical Battle Royale game, players begin with no equipment and must scavenge for it.

Community servers

CSGO community servers refer to servers that are hosted by players or other organisations rather than being hosted by the game developer - Valve. These servers frequently provide a different experience compared to the official matchmaking systems and various game modes that are unavailable on the official servers. Counter-Strike began as a simple competitive mode and nothing else. The community initially created the majority of the game modes currently available in the game. An example of this is Deathmatch.

The community servers play a significant part in CSGO's culture. Well-known community servers include KZ Climb, bunny hop, surf, and zombie survival servers.

Third-party entities such as FaceIt [27] and Esea operate and manage their own servers. These organisations organise leagues and tournaments for players to improve their skills and prove themselves among the best, even among pro players. These servers are operated as Bo1-like matches in professional play. Furthermore, the primary benefits compared to the competitive mode are broader skill ranks, enhanced anti-cheat and 128-tick rate servers. FaceIt is situated in Europe, while Esea is situated in North America. In all other aspects, both platforms provide comparable features and experiences.

Chapter 3

Methodology

This chapter presents the fundamental concepts and principles of matrix factorization and matrix completion methods. The initial segment is dedicated to the introduction of matrix factorization and its popularity. The following section is dedicated to matrix factorization techniques upon which our model is founded. Matrix factorization is employed to acquire the latent matrices, which are subsequently utilised to predict our data matrix's unknown entries. The learning process relies on the known entries that will be described in the dataset chapter.

3.1 Matrix factorization

Matrix factorization is a highly used approach in recommender systems for collaborative filtering [28]. The technique falls under the category of latent factor models, which is based on the decomposition of a matrix into the product of two or more matrices. The purpose of this decomposition is to capture the hidden relationships and structures within the data. The data are often organised in a matrix, with one dimension representing the users and the other dimension representing the objects of interest. As we can notice in figure 3.1, we can have latent factors for both users and the objects of interest. The relationships and interactions are hidden in the latent factors of the two. We could say that these vectors describe the object or the user. The relationships or structures between the user and the object are particularly captured throughout the training process.

Matrices of latent factors typically have smaller dimensions than the original matrix. The latent vectors can have a dimension ranging up to hundreds of elements, which is determined by the complexity of the problem and the required level of precision for the task. Latent features are algorithmically derived characteristics regarding the relationship between users and objects of interest. For instance, the underlying characteristics of an online retail store determine the relationship between customers and the store's merchandise. By extracting these characteristics, the online store can utilise these vectors to suggest supplementary products depending on their relevance.

Matrix factorization offers good scalability and accuracy with fast recommendations. However, it requires longer training time and necessitates retraining if new data is introduced. An advantage of this method is its ability

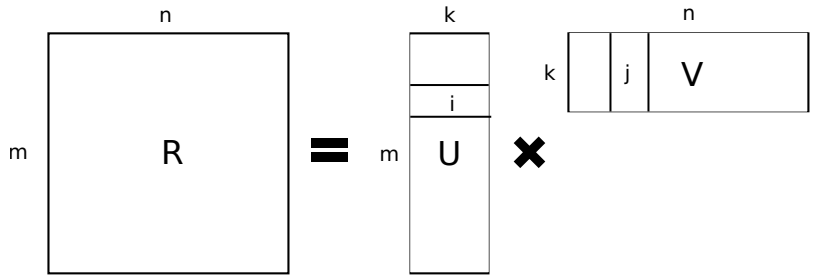

Figure 3.1: Depiction of matrix factorization, where R is rating matrix with dimensions $m \times n$, U is latent factor matrix with dimensions $m \times k$, and V is latent factor matrix with dimensions $n \times k$. The i and j represent latent factor vectors.

to integrate supplementary data. Recommender systems employ explicit and implicit feedback concepts. Explicit feedback refers to information that contains direct user input, such as a user assigning a three-star rating to a movie. Implicit feedback refers to information that indirectly reflects the user preferences obtained by observing user behaviour, such as their purchase history and browsing patterns.

Simon Funk popularised matrix factorization in 2006 during the Netflix Prize competition for the best collaborative filtering algorithm, in which participants were challenged to predict user ratings for films solely based on previous ratings without any further information about the users or films. The prize of the competition was $1,000,000. The training data contained 480,189 users, 17,770 movies, and 100,480,507 ratings.

3.2 Matrix factorization methods

Figure 3.1 depicts the main idea behind matrix factorisation. Although the complete recovery of the matrix R from the latent factor matrices U and V is achievable, it is not part of this thesis, and we will work with approximations only. The complete recovery of the matrix R is based on the completion capacity of the utilised algorithm and the number of known entries [29].

Various methods exist for calculating latent factors, but one of the most effective approaches is to employ matrix factorization techniques that utilise singular value decomposition (SVD). The input of these latent factor models is the interaction matrix, which contains observed interactions between the user and the object of interest. Various input matrices exist, each tailored to a particular use case. The most common characteristic of these matrices is their scale, where the matrix can contain millions of users and objects of interest. Often, these matrices have a very low number of observed entries, mainly because there are few users that would interact with every possible object. Typically, the user interacts with only a few objects. Based on this fact, these matrices are often called sparse matrices.

Methods of matrix factorization are trying to find latent factors by decom-

posing the interaction matrix R into matrices U and V so that equation 3.1 would apply.

$$R = UV^T \tag{3.1}$$

If the matrix R has dimensions of $m \times n$, then the matrix U has dimensions of $m \times k$ and matrix V has dimensions of $n \times k$, where $k < min\{m, n\}$. The selected k is usually much smaller than the original dimensions of the interaction matrix R.

Let us have a latent feature vector p_u, which represents the row of the user matrix U, where $p_u \in \mathbb{R}^k$. This vector is associated with user u, who it describes. Similarly, a latent feature vector q_i, where $q_i \in \mathbb{R}^k$. This vector is associated with item i, which it describes.

As mentioned before, the latent factor vectors represent the relationship between the user u and item i. Equation 3.2 is used to predict unknown values of the rating matrix R, where q_i and p_u are latent factor vectors related to item and user.

$$\hat{r}_{u,i} = q_i^T p_u \tag{3.2}$$

This calculation is an approximation of the unknown value based on the latent factor vectors learned on observed entries. Algorithms utilise this fact in order to find matrices U and V. To learn the latent factor vectors, the algorithms typically minimise squared error on the set of known ratings Ω.

$$\min_{p_u \in U; q_i \in V} \Sigma_{(u,i) \in \Omega} (r_{ui} - q_i^T p_u)^2 \tag{3.3}$$

Typically, the algorithms use regularized squared error to avoid overfitting the observed entries by regularizing the learned parameters, whose magnitude is penalized as

$$\min_{p_u \in U; q_i \in V} \Sigma_{(u,i) \in \Omega} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2), \tag{3.4}$$

where the λ constant controls the extent of regularisation. The optimisation problem is different for each approach. The most commonly used approaches will be described in the following subsections.

3.2.1 Singular value decomposition

Singular value decomposition (from now on, SVD) is a well-established technique for matrix factorization and identifying latent factors. As stated in [4], every matrix M with dimensions $s \times d$, where $d \geq s$, and rank r can be decomposed into a product of three matrices as follows:

$$M = U\Sigma V^T \tag{3.5}$$

where the matrix U is an orthogonal unitary matrix with dimensions $s \times s$ containing vectors $u \in \mathbb{R}^s$, the matrix V is an orthogonal unitary matrix with dimensions $d \times d$ containing vectors $v \in \mathbb{R}^d$ and the diagonal matrix Σ with the singular values in decreasing order on its diagonal. The depiction of the decomposition can be seen in figure 3.2.

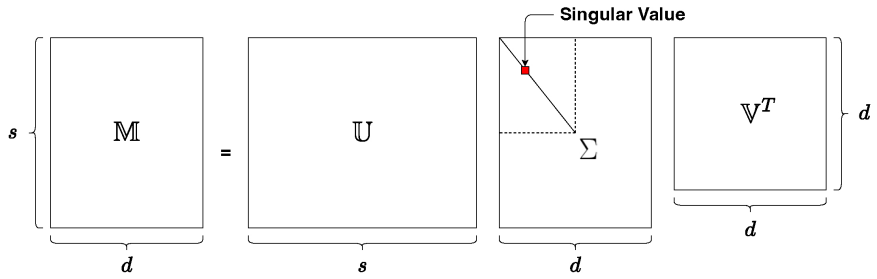

Figure 3.2: Depiction of singular value decomposition. The matrices with their dimensions are described in the chapter 3.2.1. The picture is taken from [4].

The main disadvantage of this approach is that the matrix M has to be complete to calculate the decomposition of the matrix M. There cannot be any unknown entries. There are multiple options for how to deal with this scenario. One of the options to complete the matrix M is to fill the unknown entries with zeros. However, this action will render many and, in some cases, most of the entries insignificant, considerably distorting the data. The dimensions of matrix R can be of millions, where the problem can become uncomputable because of the scale. Another option is filling the unknown entries with imputation to complete the matrix R. However, the imputation can be computationally expensive based on the algorithm used and, as stated before, the increase in the pure volume of data can become incomputable. The imputation can create inaccuracies, which may considerably distort the data.

3.2.2 Funk SVD

The traditional SVD, as mentioned before, needs the matrix to be complete, creating the need for large storage and significant computational power. In order to solve these problems, Simon Funk proposed Funk-SVD [30] in 2006 during the Netflix Prize competition. Funk-SVD is a modification of the traditional SVD.

The Funk-SVD does not create any assumptions and does not put any requirements on the matrices that the original matrix decomposes into, unlike SVD. The matrix M with dimensions $m \times n$ can be decomposed into matrices U with dimensions $m \times k$ and V with dimensions $n \times k$. Funk-SVD uses stochastic gradient descent to calculate the latent factors, which will be introduced in the subsequent sections.

3.2.3 Stochastic gradient descent

As mentioned before, traditional SVD has serious disadvantages. Based on this fact, several other methods were created, which find the decomposition of the matrix in other, typically non-exact, ways by which the disadvantages would be eliminated.

One of these methods is stochastic gradient descent (from now on SGD), which Simon Funk popularized during the Netflix Prize competition. The main benefit of this method is the ease of implementation and mainly the lower com-

putational demandingness, which makes this algorithm faster and less memory dependent.

The optimisation equation 3.4 contains two parts: squared loss and regularisation. Squared loss is the difference between the existing entry and the prediction by the model for the entry. Regularisation is used because the matrix is sparse, and the number of known entries is typically minimal. It is used to prevent overfitting of the model based on the known entries. The algorithm goes through all of the known entries and predicts, using the latent factor vectors, the expected value and then computes the associated prediction error as follows:

$$e_{ui} = r_{ui} - q_i^T p_u \tag{3.6}$$

where e_{ui} is the prediction error, r_{ui} is the known entry of the rating matrix, and the $q_i^T p_u$ is the prediction. Then, it updates the associated latent factor vectors p_u and q_i by a magnitude proportional to γ in the opposite direction of the gradient as follows:

$$q_i \leftarrow q_i + \gamma(e_{ui} p_u - \lambda q_i) \tag{3.7}$$

$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u) \tag{3.8}$$

where γ is the learning coefficient. These update equations were obtained using partial differentiation of equation 3.4. Some articles [31] incorporate a fixed value of 2 into the learning coefficient, although it is possible to neglect this constant.

This method is prevalent because of the simplicity of implementation and computational speed. The input for the algorithm is a sparse rating matrix R. The algorithm outputs the latent factor matrices U and V. The algorithm starts by initialising the latent factor matrices U and V. Different approaches exist for the initialisation from complete zeros through uniform distribution around zero up to completely random values. The algorithm can loop for a specific number of steps, or some convergent rule can be used as a stop condition. Then, for every value in the matrix, if the value is not unknown, calculate the error as in 3.6 and update the latent factor vectors q_i as in 3.7 and p_u as in 3.8.

3.2.4 Alternating least squares

The alternating least squares (from now on, ALS) method shares the error function that it minimises with SGD, but because we have two unknowns (p_u and q_i), the equation 3.4 is not convex, and the global minimum cannot be easily found. However, if we fix one of the unknowns, the optimisation problem becomes quadratic and can be solved optimally.

The essence of ALS is inherent in its name. The term “alternating” refers to the repetition of two steps as follows:

- Fix the matrix U and find the optimal matrix V.
- Fix the matrix V and find the optimal matrix U.

Type	Reality - group i	Prediction - group j
True Positive (TP)	1	1
False Negative (FN)	1	0
False Positive (FP)	0	1
True Negative (TN)	0	0

Table 3.1: Confusion matrix groups.

The term "least squares" refers to the process of solving the least-squares problem. This problem is being addressed for the currently not fixed matrix, allowing the algorithm to determine the optimal matrix for the non-fixed matrix. This action ensures that each step decreases the equation 3.4 with each step until the method converges.

Compared to SGD, the ALS algorithm has greater implementation difficulty and may exhibit worse performance. However, as stated earlier, the rating matrices can include both explicit feedback directly gained from the users and implicit feedback inferred from the observations of user behaviour. Matrices that include implicit feedback cannot be considered sparse matrices. That is why iterating over all of the known entries, as the SGD algorithm does, is not feasible. The ALS is capable of managing this particular situation.

Another significant benefit of ALS is the ease of parallelization [32]. The ALS algorithm computes latent factor vectors q_i independently of the other item vectors and p_u independently of the other user vectors. Since each vector can be computed independently, massively parallel systems could calculate each vector on a different thread, which would result in a substantial speedup.

3.3 Metrics

To effectively evaluate various algorithms and their implementations, it is imperative to establish specific measures for evaluating these methodologies. Given that our objective is to determine the victor of the match played between two teams, our work can be classified as binary classification.

Various methodologies and metrics exist for assessing the effectiveness of binary classification models. To assess the effectiveness of the match winner prediction, we will utilise a confusion matrix, accuracy, precision, and recall.

A confusion matrix in binary classification is a tabular representation of the model's classification ability, presenting a comprehensive view of how well or wrong the model is making predictions for both classes. It is a widely used tool in the field of machine learning and statistical analysis for the assessment of algorithms.

For binary classification, the confusion matrix is organised into four cells. Each cell has indexes i and j, where $i, j \in \{0, 1\}$. These indexes represent the number of observations that are in a group i and are predicted to be in a group j. Table 3.1 further explains how these indexes affect their cell. The meaning of these cells is as follows:

- **True Positive** (TP): Case where the model correctly predicts the positive class.

- **False Negative** (FN): Case where the model incorrectly predicts the negative class when the true class is positive (Type 2 error).
- **False Positive** (FP): Case where the model incorrectly predicts the positive class when the true class is negative (Type 1 error).
- **True Negative** (TN): Case where the model correctly predicts the negative class.

Accuracy refers to the ability of the model to classify data points correctly. For classification tasks, the statistical metric assesses the model's capacity to recognise or dismiss cases precisely. Accuracy is the proportion of correctly identified cases, including true positives and negatives, out of the total number of cases studied. The formula is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision, also known as positive predictive value, is calculated by dividing the number of true positives by the total of the number of true positives and false positives. It signifies the model's ability to identify only the relevant cases. The formula is as follows:

$$Precision = \frac{TP}{TP + FP}$$

Recall, also known as sensitivity, is calculated by dividing the number of true positives by the sum of the number of true positives and the number of false negatives. It signifies the model's ability to find all relevant cases. The formula is as follows:

$$Recall = \frac{TP}{TP + FN}$$

Chapter 4

Dataset

The primary objective of this chapter is to offer a thorough and detailed depiction of the dataset. In this chapter, we comprehensively analyse the dataset used in this diploma thesis. We offer comprehensive information about the data source, the methodology used for data collection, the range of data covered, the storage mechanism, and the procedures utilised for normalisation. We provide a description of the contents and fundamental attributes of the dataset.

4.1 HLTV

The dataset was acquired from a website called HLTV. This website is focused on providing comprehensive coverage of competitive Counter-Strike [33]. The website includes news articles, tournament schedules, results of matches, live scores of matches, statistics of matches, statistics of players, screenshots, images, and a forum for the community.

The website was established in 2002 to serve as a platform for hosting Half-Life TV IP addresses and demo files. These demo files were utilised for viewing Counter-Strike matches during the game's early stages. Since then, the website has evolved to incorporate news, results, statistics, and additional features.

HLTV is one of the most popular and highly respected sources of information for the CSGO community, drawing in millions of monthly visitors. The website is a crucial resource for individuals who want to stay updated on competitive CSGO, access previous match results, and find other relevant information about CSGO in the world of e-sports.

We utilise HLTV as our primary source because it is the largest and most extensive database of Counter-Strike statistics worldwide, encompassing matches, teams, and players. It is a well-known and reliable platform. An additional advantage is the availability of downloadable demo files containing data related to the match recorded by the game server. The server records every tick[5] of the game, capturing all available data. Furthermore, it includes details regarding the server's characteristics. Users can extract all of the information for research purposes. An inherent limitation of the HLTV platform is the lack of an API (application programming interface). An illustration of the data found in the

[5]Tick is the moment when the game updates the status. The most common servers in professional play are 128-tick servers, which means that the server updates 128 times a second.

Figure 4.1: Example of the player detail page. Taken from [5].

Figure 4.2: Example of the match detail page. Taken from [6].

player detail section of the website can be observed in figure 4.1, whereas an illustration of the match detail page can be observed in figure 4.2.

4.2 Data collection

We utilised web scraping techniques to collect the data from the previously mentioned website to obtain the necessary data for our tests, baselines, and model. Web scraping is an automated technique employed to gather data from websites. The process entails retrieving the HTML code of the web page and parsing the information to extract the relevant information. The data encompasses specific information such as the name of the event, scores of the teams, statistics of the players, and statistics about the maps played.

The web scraper was implemented in Ruby as a basic command line inter-

face and includes several functionalities for retrieving accessible data from the website. The available functions of the web scraper encompass the following:

- match lookup by identification
- player lookup by identification
- team lookup by identification
- top-players lookup
- top-teams lookup
- list of actual results
- list of live matches
- list of upcoming matches

The extracted information from a match contains the match identifier, final score, date of the match, team one identifier, team two identifier, and maps that were played. For each team, the information contains team identifiers, team names, and player information. The information for each player contains the player identifier, player name, number of kills, number of deaths, ADR (short for average damage per round), KAST (short for kills, assists, survived, and trades), and HLTV rating. The extracted information for each map played contains the name of the map, final score, first-half sides (which team starts where), first-half score, second-half sides, second-half score, and overtime score.

Furthermore, for each team, the team details were scraped, containing the team identifier, name of the team, world ranking, number of weeks where the team was in the top 30, average age of the players, coach, and player information. The same was done for each player, and the information contains the player identifier, real name, origin, nickname, team identifier of which the player is part of, team name, age, HLTV rating, DPR (short for deaths per round), KAST, impact (impact of the player based on multi-kills, opening kills, and clutches), ADR (average damage per round), and KPR (kills per round).

The most relevant extracted data are stored in CSV (short for Comma-Separated Values) files with a semicolon used as the delimiter. There are three files that are most relevant: matches, maps, and players.

Furthermore, to gain access to the demo files provided to the users from HLTV, we needed to scrape the links leading to the storage of the demo file. This was done by visiting each match page and using web scraper techniques. Because of the use of Cloudflare, which blocks the activity of automated bots, we were unable to obtain the demo files directly. With the links extracted, we used an application called Demo Downloader for HLTV created by David Balderson. This application is not associated with HLTV but is able to bypass the Cloudflare protection. The application contains many functions related to demo files. For example, automatic downloads of the demo files of any team. In addition, users can filter the matches by date or event. Furthermore, users can choose individual maps. The last function is the ability to specify what demo files to download by presenting links to the files. This feature was used to obtain the demo files.

4.2.1 Demo files

As previously stated, the demo file refers to the server's recording of the match. It is essentially a tape that captures all the events and actions that occurred during the match. The information contained in the demo file is as follows:

- **Player actions** - movements (including type of movement - walk, run, and crouch), aiming, shooting, throwing grenades, reloading, and all other in-game activities.
- **Camera perspective** - camera perspective of each player in the match, how each player sees and how they navigate the map.
- **Round events** - round start, round end, bomb plants, bomb defuses, kills and deaths.
- **Player positions** - player coordinates are being stored in every tick.
- **Economy aspects** - player's purchases of equipment and economic aspects (how much money the player spent and how much he is left with, for example).
- **Grenades** - types of grenades, trajectories and their impact with information on what player threw the grenade and what player was hurt for what damage.
- **Game statistics** - scores, time and relevant details to the match itself.
- **Server information** - tick rate, location, timing, and other relevant information.

To parse the demo files to extract the data from the demo files, we used a node.js library called Demofile. This library provides support for several functionalities, such as game events (e.g. player death), user messages (e.g. HUD text), console variables, entity changes, server classes, data tables, string tables, and streaming from GOTV broadcasts. The extracted data from the demo files comprises information on utility utilisation (specifically grenades) and economic characteristics (economic strategies).

In addition, an application called CS Demo Manager was used to double-check the extracted information from the demo files. This application extracts information from the demo file, like player information, including statistics, movement, economic decisions and grenade trajectories. The application has other functions like heat map generation, video generation of the highlights, and demo filtering, which were not utilised.

4.2.2 Feature possibilities

From the previous subsection, you can get the idea that much more information could be used for analytical purposes and feature creation. The problem with many of the statistics is how to classify and evaluate them all together. For example, we could assign the player to a specific location based on the player's position. The problem is that the roles in CSGO do not work as in football or other sports. In football, every player can be assigned a position on the field,

Figure 4.3: Modified cut out of the map de_inferno to showcase the possible placement of smoke grenades. The picture is taken from [7].

which could clarify important factors of that role. However, this is not possible to do in CSGO because of the dynamic character of the game and how fluently the players and roles can interchange based on the state of the game or certain situations. In the thesis [8], where the author was predicting NBA matches, he used the vectors based on the player roles. This could be possible to recreate, but the roles are not distinct enough to be properly utilised.

Another example of this is how to decide if the smoke grenade placement is useful or not. The naive approach would be to say if the smoke grenade obstructed the vision of the enemy, the grenade was useful. Notice the figure 4.3 where you can see the cutoff of bombsite A from Inferno with multiple differently coloured crosses. The standard execution of the bombsite takeover would utilise all of the places for smoke grenades depicted by the crosses. However, in the context of singular smoke, how would we decide what position is the best for the smoke grenade? If the smoke was thrown at the Red Cross, the Terrorist team could cross from Mid to Short safely, rendering the Counter-Terrorist player on the Long ineffective. However, if the Long player was killed before the grenade was thrown, the smoke grenade would be ineffective and better used elsewhere. The smoke grenade could be even counterproductive because, depending on the information the team has, one of the Terrorist players could sneak through the Long to Counter-Terrorist spawn location to catch the rotating player off guard or even attack the bombsite from a different angle. On the other hand, every smoke grenade that, in theory, can prevent the enemy team from attacking or gaining intel is beneficial to the team who threw it.

Player positioning on the bombsites and different defending positions are other features that could be interesting to study. Each site has so-called default positions. These positions are the most common and usually the strongest for the Counter-Terrorist players to stay in. These positions are usually pre-aimed and checked by the Terrorist players when entering the bomb site. Another way to deal with these positions is by utilising grenades, which can neutralise

the positions before the team enters the bomb site. This is why many of the Counter-Terrorist players hold so-called "off" angles that are unusual. Sometimes, the Counter-Terrorist players risk walking through the smoke grenades deployed by the Terrorist in order to catch the Terrorist team off guard.

The game is complex, and the benefits of these implicit features that could be extracted from player behaviour would need to be studied and properly analysed to decide what metric to use. It would be interesting to see the impact of these features.

4.3 Attributes of collected data

From January 2020 to September 2022, we have gathered a total of 20,141 professional matches. These matches include 40,775 maps, which in turn consist of 1,083,801 rounds played by 1,753 distinct teams composed of 5,654 unique players. The map pool has 11 maps: de_dust2, de_inferno, de_mirage, de_ancient, de_vertigo, de_nuke, de_overpass, de_tuscan, de_cobblestone, de_train, and de_cache.

The rating matrix has dimensions 1753×1753, and the number of observed entries is 10,995, which puts the rate at 0.003. This rate is very low. Ideally, the percentage of observed entries would be at least five times higher. There are different ideas about what is the ideal rate of known entries for matrix completion. Both articles [29] and [34] introduce ways how to approximate the necessary number of entries based on algorithms completion capacity and matrix rank. Some sources mention that the matrix must have at least between 5 % and 20 % known entries for a good approximation.

From tables 4.1 and 4.2, which contain information about the unique teams that participated in the specific map, we can notice that Tuscan, Cobblestone and Cache contain very few entries. For this reason, these maps will not be included in the experiments.

Table 4.3 contains the win rates on the map for a team starting as Counter-Terrorist. This table represents the first baseline to which we will compare our experiments and models.

Some interesting facts about the dataset are as follows:

- 645 teams played only one match.
- 1477 teams played less than five matches.
- 1101 teams played more than five matches.
- 749 teams played more than ten matches.
- 257 teams played more than 50 matches.
- 148 teams played more than 100 matches.
- Team SKADE played 460 matches, which is the most from the time period.
- On average, each team had 7.9 players during the time period.

Map name	Unique CT teams	Unique T teams
Dust2	856	849
Inferno	1002	966
Ancient	475	425
Vertigo	707	651
Nuke	892	795
Overpass	823	747
Tuscan	5	4
Cobblestone	8	5
Train	471	458
Cache	4	4

Table 4.1: Counts of unique teams by side (Counter-Terrorists and Terrorists) per map.

Map name	Number of unique teams
Dust2	611
Inferno	709
Mirage	694
Ancient	327
Vertigo	451
Nuke	616
Overpass	549
Tuscan	5
Cobblestone	5
Train	334
Cache	4

Table 4.2: Counts of unique teams per map.

Map name	Win rate of CT side
Dust2	49.1%
Inferno	49.3%
Mirage	52.3%
Vertigo	48.7%
Nuke	53.6%
Overpass	52.9%

Table 4.3: Win rate of the Counter-Terrorist side for selected maps.

Player name	Kills	Deaths	ADR	KAST	Rating
ju1ces	69	66	68.0	75.5	1.10
DannyG	74	74	73.2	66.7	1.03
chelleos	69	75	71.1	55.9	0.98
nettik	60	74	75.6	62.7	0.93
ADK	62	73	75.9	61.8	0.92
HUGHMUNGUS	59	55	82.4	65.4	1.11
Jinxx	52	44	71.6	73.1	1.05
apocdud	54	54	81.6	66.7	1.02
Mayker	53	49	69.4	64.1	1.00
SkulL	49	62	68.1	62.8	0.93

Table 4.4: Statistics of a match between teams ROOSTER and CHIEFS.

4.4 Data normalization

Normalising the data points to a consistent scale is essential because of the methods employed. Linear models utilising error functions such as MSE (Mean Squared Error) or RMSE (Root Mean Squared Error) are not invariant to changes in the scale of the data. Algorithms such as SGD also experience similar phenomena. Retaining features with variant ranges, such as ADR (average damage per round) or economic features with values that can exceed hundreds or even thousands, would introduce high variance in the model. Consequently, certain features may become unnecessarily important. This problem is not relevant for models like linear regression, which are invariant to the linear transformation of scaling. However, it could help with interpretability.

The features derived from the results of the matches, like win ratio and loss ratio in their raw form, are scaled from 0 to 1. However, player statistics are in a completely different value range. For this reason, player kills, player deaths and other similar statistics were recalculated to fit into the range from 0 to 1. The problem with the player statistics is that it would be unfair because the number of rounds can vary from map to map. For example, the match between ROOSTER and CHIEFS during ESL Australia & NZ Championship Season 10 played on Mirage consisted of 78 rounds where players had statistics that can be seen in table 4.4. In theory, the maximal number of kills that a player can reach is $num\_of\_rounds \times 5$, so in the previously mentioned match, one of the players could reach 390 kills, which would be absurd. The average number of kills per match is 27. For this reason, most of the player's statistics, which could potentially reach absurd numbers, are converted into a "per round" type of statistics by $player\_statistic \div number\_of\_rounds$ and then rescaled using min-max normalization.

4.4.1 Min-Max normalization

Min-max normalization, commonly known as min-max scaling, is a widely used linear scaling transformation [35]. This approach is straightforward and performs feature scaling by individually adjusting each feature to a specified range. The specified range in our case is between 0 and 1. An advantage of this transformation is that it preserves the relationship between the data. The formula is as stated:

Map name	Testing set	Training set	Percentage
Dust2	929	4797	16.2%
Inferno	1197	6446	15.7%
Mirage	1127	5467	13.9%
Vertigo	706	3532	16.7%
Overpass	829	4226	16.4%
Overall	6436	34321	15.8%

Table 4.5: Split of the data into training and testing sets per map.

$$x_i = \frac{x_i - min(x}{max(x) - min(x)} \tag{4.1}$$

4.5 Data partitioning for training and testing

In order to assess the quality of each model, it is necessary to partition the available data into distinct sets for training, validation, and testing purposes. The training set is utilised for model training, whilst the test set is employed for model evaluation. Throughout the evaluation process, we collect measures as outlined in chapter 3.3 that facilitate the comparison of individual models. The validation set is employed to assess the model's performance unbiasedly during training. This assessment is crucial for determining the optimal hyperparameters and/or implementing regularisation techniques such as early stopping. Prior to the actual assessment of the model, the training and validation sets are merged to utilise all the available data.

In our scenario, the date serves as the fundamental attribute of our data. The data is partitioned into training, validation, and test sets based on the date. The training set accounts for 85% of the data, whereas the test set accounts for 15%. In addition, the training set is split into a training set and a validation set at a ratio of 90% and 10%, respectively. The test set comprises the most recent data, whereas the training set comprises the oldest data.

To avoid the need for frequent ratio/index calculations, we determined the optimal date for dividing our data into training and testing sets. The ideal date is May 1, 2022. Table 4.5 illustrates the proportions of training and testing set data for every map employed in the dataset. The final row of the table represents the overall division of the data.

Chapter 5

Experiments and results

In this concluding chapter, we present and analyse the conducted experiments and their corresponding results based on the data described in the previous chapter. The experiments are conducted methodically to demonstrate the performance of matrix completion models compared to the state-of-the-art models identified in the literature revision chapter. The models are assessed using the metrics outlined in chapter 3.3.

Articles that have used matrix factorization to predict the outcome of matches, whether football, basketball or in another context, take advantage of the fact that the matrix has two dimensions. The authors can build a rating matrix in a way where entry $R[i, j]$ represents the home match between the two teams and $R[j, i]$ represents the away match. This cannot be used in CSGO because there is a lack of both the concept of a league and a home and away-match.

The absence of a league concept poses a concern since it allows clubs to compete irrespective of their skill level. It is not uncommon for a newly formed team consisting of highly rated FaceIt players to compete against a team that has been playing at the highest level for several years. These teams that consist of FaceIt players frequently disband, and individuals consistently depart from them. It is common for a player to be kicked out if they fail to perform. This fact contributes to the presence of noisy data, which might have a negative impact on forecasting. Simultaneously, it is possible that these teams will never encounter each other in the future. In football and other structured sports leagues, teams engage in a minimum of two matches against each other per season, and there is also the possibility of additional encounters in tournaments that take place simultaneously with the regular season.

Most events in the professional competitive scene are organised in a tournament format, often featuring a preliminary group stage. Certain events directly place teams straight into the main tournament bracket through a random draw. In tournaments featuring a group stage, teams compete against each other in a single match where the best-of-1 format determines the winner. Consequently, based on their position within the group, they will be incorporated into the main tournament bracket. Frequently, these tournaments are conducted using a double-elimination format, where a team that loses in the main tournament bracket is then placed in the losers bracket. Nevertheless, they still possess an opportunity to advance to the finals. The main tournament bracket can often

be played in the best-of-1 concept, depending on the organisation that oversees the event.

The closest concept in CSGO to the home and away-match concept is to utilise only the first half of the match. By utilising solely the first portion of the match, we can construct the rating matrix R in such a manner that the element $R[i, j]$ relates to the Counter-Terrorist team, whereas the element $R[j, i]$ refers to the Terrorist team.

The experiments were performed with the Funk-svd Python library, which is freely accessible on GitHub[6]. This library implements the Funk SVD method, as described in chapter 3.2.2. The sole distinction is in the inclusion of biases and a global average rating, which serve to enhance the accuracy of the model. The model incorporates biases to accommodate the users' and items' systematic tendencies (preferences) [36]. Bias calculation closely resembles the calculation of latent components. This approach is straightforward yet extensively employed to enhance the accuracy of models.

5.1 First-half only latest model

This approach exclusively utilises each team's most recent map outcome to construct the rating matrix R. The rating matrix is constructed such that the $R[i, j]$ entry represents the performance of team i as the Counter-Terrorists against team j, while $R[j, i]$ represents the performance of team j as the Terrorists against team i.

This experiment was our first and mainly served to validate its feasibility. In Table 5.2, it is possible to see selected results divided by individual maps and overall, where the map was ignored. The maximum number of rounds is a constant that is equal to 15. As previously stated in the game description, this constant is set by the game mode. For testing purposes, the victor was determined based on the higher number of rounds won out of the 15 rounds in the first half. The prediction was generated by multiplying the latent factors and subsequently applying a threshold to determine the outcome. The optimal threshold for thresholding was determined using the Scipy Python package based on accuracy. The use of the optimal threshold resulted in improvements in accuracy and recall compared to the average of the possible range, as seen in table 5.1.

This experiment was our initial endeavour primarily aimed at validating the feasibility of this task. The entries of matrix R were computed as the win ratio of the team. The formula is as follows:

$$R[i, j] = \frac{number\_of\_rounds\_won}{maximum\_number\_of\_rounds}.$$

The optimal learning rate and regularization combination were determined algorithmically by exhaustively testing all feasible combinations. Afterwards, the model was evaluated on various reasonable counts of latent factors. Table 5.2 shows the result achieved for each map. The final row in the table demonstrates the model's capacity for generalisation.

[6] https://github.com/gbolmier/funk-svd/tree/master

n	Acc w/	Acc w/o	Rec w/	Rec w/o	Prec w/	Prec w/o
5	0.588	0.582	0.863	0.649	0.592	0.623
10	0.570	0.565	0.962	0.640	0.570	0.608
20	0.580	0.568	0.921	0.631	0.580	0.613
50	0.573	0.556	0.916	0.606	0.576	0.606
100	0.561	0.556	0.988	0.622	0.563	0.602

Table 5.1: Demonstration of using the optimal threshold in comparison to the usual thresholding method. The variable n denotes the number of latent factors. Acc is an abbreviation for accuracy, Rec is an abbreviation for recall, and Prec is an abbreviation for precision. Symbol $w/$ denotes the optimal threshold used, and symbol w/o denotes the contrary.

Map name	n	Accuracy	Precision	Recall
Dust2	20	0.561	0.561	0.996
Inferno	10	0.569	0.555	0.685
Mirage	20	0.604	0.638	0.856
Vertigo	5	0.598	0.603	0.804
Nuke	10	0.649	0.651	0.993
Overpass	5	0.582	0.581	0.916
Overall	10	0.589	0.593	0.861

Table 5.2: Achieved results on the first-half "only latest" dataset by FunkSVD. The n represents the number of latent factors.

Map name	n	Accuracy	Precision	Recall
Dust2	100	0.557	0.511	0.876
Inferno	10	0.516	0.421	0.248
Mirage	100	0.540	0.555	0.415
Vertigo	50	0.531	0.521	0.789
Nuke	100	0.510	0.561	0.873
Overpass	10	0.546	0.597	0.383
Overall	20	0.507	0.502	0.437

Table 5.3: Achieved results on the first-half "decay" dataset by FunkSVD. The n represents the number of latent factors.

5.2 First-half decay model

In this experiment, instead of solely utilising the most recent match, all of the accessible data for the team is employed. The data is sorted by date in descending order, and then, the rating for each team is computed using a decay function. The decay is determined by the formula

$$decay = power(alpha - rate, iter),$$

where *alpha* represents the multiplier, *rate* indicates the rate at which the multiplier should decrease, and *iter* denotes the number of iterations. Otherwise, the strategy is similar to the one described in the previous subsection. The thresholding is performed in a similar fashion. The table 5.3 illustrates achieved results.

Map name	Accuracy	Precision	Recall
Dust2	0.510	0.602	0.466
Inferno	0.543	0.540	0.483
Mirage	0.554	0.649	0.581
Vertigo	0.547	0.605	0.484
Nuke	0.591	0.703	0.598
Overpass	0.556	0.637	0.591
Overall	0.555	0.610	0.622

Table 5.4: Achieved results by logistic regression on the first-half dataset.

Map name	Accuracy	Precision	Recall
Dust2	0.570	0.635	0.615
Inferno	0.531	0.521	0.582
Mirage	0.576	0.640	0.696
Vertigo	0.551	0.601	0.526
Nuke	0.598	0.716	0.610
Overpass	0.564	0.636	0.624
Overall	0.558	0.614	0.619

Table 5.5: Achieved results by random forest classifier on the first-half dataset.

5.3 First-half results

The implementation of logistic regression and random forest classifier from the Sklearn Python library was utilised for the purposes of testing. We created specific data from the general match information to compare these algorithms with the previously mentioned matrix factorization models. The feature vectors for each team comprise data on the average number of victories and losses, the most recent match rating, and the decayed rating of all prior matches for both sides, Counter-Terrorists and Terrorists. The results obtained from logistic regression can be examined in table 5.4, whilst the results obtained from the random forest classifier can be examined in table 5.5.

All of the first-half models surpassed the most fundamental benchmark, which is the win rate on a map. Surprisingly, the thresholded matrix factorization model proved highly competitive compared to logistical regression and random forest classifier. The matrix factorization model has outperformed logistical regression on all maps, including the collective set of maps. The model performed better than the random forest classifier in certain cases.

Surprisingly, the first-half latest model outperformed the first-half decay model significantly. To a certain extent, it is logical because the data has the potential to interfere with the assessment. In principle, certain teams could accumulate high ratings through several victories, only to decline and start losing. If the losses happen immediately before the test set, it could negatively impact the predictions. Another factor to consider is that the team could have participated in events that were below their skill level, resulting in effortless triumphs.

Map name	n	Accuracy	Precision	Recall
Dust2	100	0.546	0.534	0.787
Inferno	50	0.516	0.508	0.574
Mirage	10	0.528	0.525	0.844
Vertigo	50	0.531	0.515	0.707
Nuke	10	0.551	0.517	0.815
Overpass	20	0.564	0.540	0.676
Overall	10	0.571	0.563	0.474

Table 5.6: Achieved results on the full-match "only latest" dataset by FunkSVD. The n represents the number of latent factors.

Map name	n	Accuracy	Precision	Recall
Dust2	20	0.580	0.614	0.726
Inferno	100	0.577	0.577	0.977
Mirage	100	0.542	0.549	0.821
Vertigo	20	0.578	0.589	0.862
Nuke	10	0.613	0.637	0.815
Overpass	10	0.562	0.579	0.737
Overall	10	0.569	0.568	0.998

Table 5.7: Achieved results on the full-match "decay" dataset by FunkSVD. The n represents the number of latent factors.

5.4 Full-match only latest model

In this experiment, we have abandoned the concept of pitting the Counter-Terrorist team against the Terrorist team. The structure has been maintained for the sake of consistency. Consequently, the value in the rating matrix R at index $[i, j]$ represents the team that initially played as Counter-Terrorists, while the value at index $[j, i]$ indicates the team that initially played as Terrorists. The victor is now determined by the actual winner of the map rather than the one who has won the most rounds. The entry for each team in the rating matrix is calculated using the same method as in the first-half model. The results achieved from the matrix factorization model can be examined in table 5.6.

5.5 Full-match decay model

As mentioned before, this experiment no longer utilises the concept of home and away match. Similarly, as in the first-half "only latest" model, the data are sorted in descending order based on date. The decay is then utilised to calculate the team rating. However, thresholding is no longer needed, as each map has a distinct winner. The formula remains unchanged from the one previously described. The achieved results can be examined in table 5.7.

Map name	Accuracy	Precision	Recall
Dust2	0.534	0.636	0.467
Inferno	0.549	0.544	0.519
Mirage	0.583	0.658	0.657
Vertigo	0.561	0.620	0.504
Nuke	0.609	0.693	0.691
Overpass	0.580	0.645	0.655
Overall	0.574	0.628	0.633

Table 5.8: Achieved results by logistic regression on the full-match dataset.

Map name	Accuracy	Precision	Recall
Dust2	0.555	0.632	0.561
Inferno	0.529	0.520	0.563
Mirage	0.585	0.652	0.681
Vertigo	0.549	0.597	0.530
Nuke	0.612	0.698	0.689
Overpass	0.560	0.630	0.630
Overall	0.564	0.620	0.621

Table 5.9: Achieved results by random forest classifier on the full-match dataset.

5.6 Full-match results

As previously stated, we derived specific data from the general match information in order to compare these techniques. We ensured that we incorporated solely the identical category of data accessible to the matrix factorization model. The feature vectors remain unchanged from the previous mention, but now they incorporate the data relevant to the second half.

Modelling only the second half is not meaningful as the data does not exhibit a direct correlation with the match outcome. The second half is highly reliant on the first half of the match. For instance, if the team achieves a score of 15 rounds during the first half, they would only require one additional round to secure a victory and could afford to lose a maximum of 14 rounds without facing defeat in the match. Throughout the experiments, we attempted to replicate the modelling approach used for the first half in order to analyse the second half. However, the outcomes were confusing due to the influence of the winning metric.

The findings of the current model, to some extent, deteriorated compared to the results attained by the first-half models, as we had anticipated. Nevertheless, the decay model yielded better results than the latest model. This behaviour is logical as numerous teams experience a strong start but then struggle to keep it up. CSGO is renowned for its comeback stories.

5.7 Matrix completion model

Let us have a matrix X with dimensions $num\_of\_teams \times num\_of\_featuers$, representing every map's team feature. This matrix is calculated beforehand and is fixated. In addition, let us have a rating matrix R, which consists of the

observed entries. The rating matrix R follows the previously defined $CT \times T$ structure. To predict the unknown entries of the rating matrix R, we have to calculate the following formula:

$$\hat{R} = XM_M X^T, \tag{5.1}$$

where matrix M is denoted as:

$$M = U_M {V_M}^T, \tag{5.2}$$

where U_M and V_M are embeddings for which applies $U_M \in \mathbb{R}^{n \times k_1}$ and $V_M \in \mathbb{R}^{n \times k_1}$. Here, k_1 is a hyperparameter that affects a number of latent factors, while n is the number of teams.

We are using the logistic activation function, also called Sigmoid, to train this model to obtain the prediction. The forward step of the model looks as follows:

$$OUT = \sigma((X[ct,:]U_M) \times (V_M X[t,:]^T)), \tag{5.3}$$

where σ represents the activation function, $X[ct,:]$ and $X[t,:]$ represent respective team feature vectors, and U_M and V_M are trained embeddings.

5.7.1 Data

For this particular model, custom datasets were created that include feature vectors for each team. We created three distinct datasets for our experimental analysis. Each dataset consists of three categories: general information, player information, and map-specific information.

The general information provides an overview of the total number of matches won and lost and the number of rounds won and lost for both the Counter-Terrorist and Terrorist sides. Player information may vary based on the dataset type. It can include current players' statistics, such as kills, deaths, ADR, and ratings, as well as more comprehensive information from demo files, such as utility usage and other statistics. The map-specific data included details on the number of matches won and lost, and the number of rounds won and lost for both the Counter-Terrorist and Terrorist sides on that particular map.

Dataset "by date"

This dataset includes numerous records for each team, depending on the number of matches played by the team and the corresponding dates. If the match occurred on January 10th, then all prior matches of the team to that date would be relevant to this entry. It included all the data that was previously described.

Dataset "cutoff date"

The dataset has a singular entry for every team. A cutoff date was chosen, and data were generated for each team up to the designated date. It included all the data that was previously described.

Dataset "general"

This dataset was created in a similar manner to the "by date" dataset. However, it does not incorporate individual players. Player-specific statistics are converted into more general statistics. These statistics are typically team-wide averages.

5.7.2 Hyperparameter selection and training

Different optimizers are differently robust to bad hyperparameter initialization. ADAM is more robust than SGD, but SGD, with the correct learning rate, regularization and momentum tuning, can outperform ADAM both in speed and accuracy. ADAM is a much safer option as it will often achieve convergence even with bad hyperparameters.

Initialization can have the same impact as learning rate and momentum for convergence. We tested different initialization approaches, mostly ones that work well with SGD, like Xavier/Glorot initialization. In addition, we experimented with truncated normal distribution, uniform distribution and completely random initialization. We experimented with different activation functions, but Sigmoid converged the best. Most of the hyperparameters were selected by trial-and-error approach within the reasonable ranges. Figure 5.1 contains the example of the evolution of training accuracy during the training.

The main results were obtained by combination as follows:

- **Optimizer** - Adam
- **Learning rate** - 1e-6
- **Regularization** - 1e-3
- **Initialization** - Xavier/Glorot
- **Activation function** - Sigmoid
- **Loss function** - BCELoss

5.7.3 Experiments and results

Tables from 5.10 to 5.15 contain achieved results on different types of data as described in subsection 5.7.1. Our model was comparable with random forest classifier and logistical regression. In some cases it achieved better results than the before mentioned algorithms.

5.7.4 Extension of the model

During experiments with the matrix completion model, the idea emerged to use another matrix embedding to capture the rest of the information not captured by the matrix M.

The unknown entries of the rating matrix R are approximated as

$$\hat{R} = XM_M X^T + Z. \tag{5.4}$$

For matrix Z applies

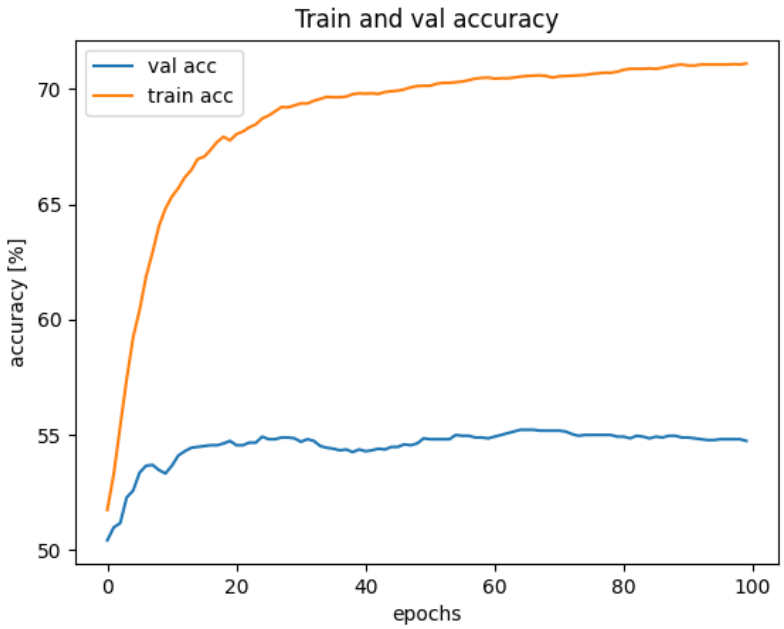

Figure 5.1: Training and validation accuracy without early stopping to showcase the choice of hyperparameters 5.7.2.

Map name	n	Accuracy	Logistic Regression	Random forest
Dust2	75	0,575	0,535	0,558
Inferno	50	0,511	0,503	0,552
Mirage	75	0,554	0,513	0,528
Nuke	150	0,548	0,597	0,613
Overpass	5	0,548	0,579	0,572
Vertigo	75	0,565	0,549	0,549

Table 5.10: Achieved results by our model on the first-half "by date" dataset compared to logistic regression and random forest classifier.

Map name	n	Accuracy	Logistic Regression	Random forest
Dust2	5	0,561	0,543	0,545
Inferno	75	0,533	0,518	0,520
Mirage	75	0,557	0,555	0,577
Nuke	20	0,565	0,587	0,595
Overpass	5	0,598	0,529	0,544
Vertigo	75	0,557	0,566	0,564

Table 5.11: Achieved results by our model on the first-half "cutoff" dataset compared to logistic regression and random forest classifier.

Map name	n	Accuracy	Logistic Regression	Random forest
Dust2	100	0,570	0,539	0,579
Inferno	5	0,532	0,506	0,523
Mirage	20	0,538	0,533	0,560
Nuke	5	0,575	0,607	0,613
Overpass	5	0,543	0,561	0,564
Vertigo	5	0,570	0,556	0,556

Table 5.12: Achieved results by our model on the first-half "general" dataset compared to logistic regression and random forest classifier.

Map name	n	Accuracy	Logistic Regression	Random forest
Dust2	75	0,586	0,576	0,589
Inferno	10	0,586	0,558	0,598
Mirage	100	0,527	0,518	0,527
Nuke	20	0,559	0,570	0,527
Overpass	200	0,505	0,524	0,557
Vertigo	200	0,570	0,559	0,549

Table 5.13: Achieved results by our model on full-match "by date" dataset compared to logistic regression and random forest classifier.

Map name	n	Accuracy	Logistic Regression	Random forest
Dust2	200	0,614	0,598	0,602
Inferno	20	0,654	0,583	0,583
Mirage	50	0,585	0,523	0,515
Nuke	200	0,553	0,587	0,556
Overpass	20	0,516	0,513	0,546
Vertigo	100	0,573	0,561	0,574

Table 5.14: Achieved results by our model on full-match "cutoff" dataset compared to logistic regression and random forest classifier.

Map name	n	Accuracy	Logistic Regression	Random forest
Dust2	200	0,634	0,574	0,608
Inferno	5	0,565	0,567	0,586
Mirage	200	0,554	0,517	0,527
Nuke	75	0,548	0,560	0,542
Overpass	200	0,532	0,515	0,557
Vertigo	5	0,591	0,584	0,571

Table 5.15: Achieved results by our model on a full-match "general" dataset compared to logistic regression and random forest classifier.

$$Z = U_Z V_Z, \tag{5.5}$$

where U_Z and V_Z are embeddings for which $U_Z \in \mathbb{R}^{n \times k_2}$ and $V_Z \in \mathbb{R}^{n \times k_2}$ applies. Here, k_2 is a hyperparameter that affects a number of latent factors while n is the number of teams.

To make the forward step of this model, the steps would be as follows:

$$inductive = (X[ct, :]U) \times (VX[t, :]^T) \tag{5.6}$$

$$noninductive = U_Z \cdot V_Z \tag{5.7}$$

$$OUT = \sigma(inductive + noninductive) \tag{5.8}$$

However, we had trouble making this model converge, second-guessing everything from learning rate, initialization, vanishing gradient, insufficient dataset, and model complexity, but the model still did not converge.

5.8 Future work

The potential for further advancement in this thesis is boundless. Enhancing the process of extracting and analysing data from demo files would result in significant advancements. Several features pose challenges in evaluation, and determining what is correct or wrong is contingent upon several factors. If heat maps could be produced with precision, it would enable the utilisation of alternative collaborative filtering techniques to get further insights by analysing commonalities and, conversely, disparities in the patterns of teams' and players' behaviour. Another viable suggestion could involve analysing patterns in the data and leveraging biases to emphasise specific trends. Perhaps employing time series methodologies could aid us in identifying trends in teams that are performing very well and using the outcome as bias.

Conclusion

The objective of this diploma thesis was to utilise matrix completion algorithms in order to predict the outcomes of e-sport matches, with a specific emphasis on the game Counter-Strike: Global Offensive (CSGO). In order to accomplish this objective, it was necessary to initially conduct a comprehensive review of the available literature, gather the dataset, devise a model, carry out and execute initial experiments, and analyse the outcomes and potential future avenues.

The thorough literature review enabled us to comprehend the strategies employed and their respective methodologies. Many publications and school projects employ methodologies like logistic regression and random forest classifiers to tackle these types of tasks. Simultaneously, during the literature review, I encountered information regarding the extraction of characteristics from e-sports games and their subsequent processing into usable features for the model.

While conducting the literature review, we encountered various methodologies for collecting data in the CSGO context. The majority of the authors utilised demo files to extract the required features for their task. Typically, the feature set was limited to the specific context of research. The publications usually presented results from less than one hundred demo files. For the purpose of providing more information, a tournament consisting of 32 teams generates approximately 90 demonstration files. However, we comprehend the rationale behind this action. The CSGO's dynamic environment evolves constantly. Due to the absence of established leagues, teams from various regions with distinct skill levels encounter each other on a daily basis. Therefore, utilising data from a single event for testing is logical. Nevertheless, we aimed to undertake sparse matrix approximation and predict the unknown entries of the rating matrix utilising methods of matrix completion.

We initiated our experiments with rudimentary ones in order to gain a thorough comprehension of the data. Subsequently, we introduced a model based on matrix completion, which utilises past data to predict future results. The model underwent training using several datasets. We then assessed the performance of the models using the metrics outlined in the theoretical section of this thesis and conducted an analysis of the results, interpreted them, and presented them visually using tables.

Ultimately, we have successfully accomplished our objectives of creating

and executing a model based on matrix completion principles to predict the results of e-sport matches based on historical data. While the method did not definitively surpass the state-of-the-art methods and has considerable room for improvement, the results are not excessively poor.

Bibliography

[1] Adela-Sznajder. PRESS GALLERY: Intel Extreme Masters Rio 2023 [online]. April 2023, [Cited 2024-01-20]. Available from: https://photos.eslgaming.com/PRESS-GALLERY-Intel-Extreme-Masters-Rio-2023-

[2] Pruim, R. Classic Buy Menu [online]. May 2013, [Cited 2024-01-20]. Available from: https://gamebanana.com/mods/27461

[3] EVO_SHOCKER. Callout map for Ancient [online]. December 2020, [Cited 2024-01-20]. Available from: https://www.reddit.com/r/csgo/comments/k6vpk2/i_made_a_callout_map_for_ancient_using_what_ive/

[4] Liu, B.; Pejó, B.; et al. Privacy-Preserving Federated Singular Value Decomposition. *Applied Sciences*, volume 13, no. 13, 2023, ISSN 2076-3417, doi:10.3390/app13137373. Available from: https://www.mdpi.com/2076-3417/13/13/7373

[5] HLTV.org. HLTV Oleksandr s1mple Kostyliev Counter-Strike Statistics [online]. January 2024, [Cited 2024-01-20]. Available from: https://www.hltv.org/stats/players/7998/s1mple

[6] HLTV.org. HLTV ESL Challenger Melbourne 2024 Oceania Open Qualifier [online]. January 2024, [Cited 2024-01-20]. Available from: https://www.hltv.org/stats/matches/mapstatsid/170325/dxa-vs-making-history

[7] unCURTYous. Dust II & Inferno map [online]. June 2016, [Cited 2024-01-20]. Available from: https://www.csgowallpapers.com/wallpaper/1204-dust-ii---inferno-map

[8] Tran, T. *Predicting NBA Games with Matrix Factorization*. Graduate theses, Massachusetts Institute of Technology, 2016. Available from: https://dspace.mit.edu/handle/1721.1/106385

[9] *A Two-Stage Real-time Prediction Method for Multiplayer Shooting E-Sports*, volume 46, AIS Electronic Library, 2020. Available from: https://aisel.aisnet.org/iceb2020/46

[10] Rubin, A. Predicting Round and Game Winners in CSGO. Jan 2022, doi: 10.31219/osf.io/u9j5g. Available from: osf.io/u9j5g

[11] Björklund, A.; Visuri, W. J.; et al. *Predicting the outcome of CS:GO games using machine learning*. Dissertation thesis, CHALMERS UNIVERSITY OF TECHNOLOGY, 2018. Available from: https://api.semanticscholar.org/CorpusID:70208061

[12] Makarov, I.; Savostyanov, D.; et al. Predicting Winning Team and Probabilistic Ratings in "Dota 2" and "Counter-Strike: Global Offensive" Video Games. In *Analysis of Images, Social Networks and Texts*, edited by W. M. van der Aalst; D. I. Ignatov; M. Khachay; S. O. Kuznetsov; V. Lempitsky; I. A. Lomazova; N. Loukachevitch; A. Napoli; A. Panchenko; P. M. Pardalos; A. V. Savchenko; S. Wasserman, Cham: Springer International Publishing, 2018, ISBN 978-3-319-73013-4, pp. 183–196.

[13] Liquipedia. OGN - Liquipedia PUBG Wiki[online]. May 2023, [Cited 2024-01-20]. Available from: https://liquipedia.net/pubg/OGN

[14] Corporation, V. Counter-Strike: Global Offensive - About [online]. 2024, [Cited 2024-01-20]. Available from: https://blog.counter-strike.net/index.php/about/

[15] steamcharts. Counte-Strike - Steam charts [online]. January 2012, [Cited 2024-01-20]. Available from: https://steamcharts.com/app/10

[16] Earnings, E. Counte-Strike vs Counter-Strike: Source - Prize Pools & Players Compared [online]. 2012, [Cited 2024-01-20]. Available from: https://www.esportsearnings.com/comparisons/zzvv-cs-vs-css

[17] MIRAA. DH WINTER WITH $250K TOURNAMENT [online]. September 2013, [Cited 2024-01-20]. Available from: https://www.hltv.org/news/11368/dh-winter-with-250k-tournament

[18] SteamDB. Counter-Strike Steam Charts [online]. 2012, [Cited 2024-01-20]. Available from: https://steamdb.info/app/730/charts/

[19] HLTV. Adil ScreaM Benrlitom Counter-Strike Statistics [online]. 2024, [Cited 2024-01-20]. Available from: https://www.hltv.org/stats/players/7390/ScreaM

[20] Wiki, C.-S. Weapons [online]. January 2020, [Cited 2024-01-20]. Available from: https://counterstrike.fandom.com/wiki/Category:Weapons

[21] Wiki, C.-S. Money [online]. February 2023, [Cited 2024-01-20]. Available from: https://counterstrike.fandom.com/wiki/Money

[22] Professeur. PROS COMMENT ON STATE OF ANCIENT [online]. August 2021, [Cited 2024-01-20]. Available from: https://www.hltv.org/news/32125/pros-comment-on-state-of-ancient

[23] MrPrisen. Ancient [online]. January 2024, [Cited 2024-01-20]. Available from: https://counterstrike.fandom.com/wiki/Ancient

[24] GmbH, E. G. ESL Pro Tour CS:GO Game Specific Rules [online]. January 2022, [Cited 2024-01-20]. Available from: https://drive.google.com/file/d/1z4Ep_tbjDw8MZzV7dZBwbyr6Lhv-pqdq/view

[25] Wiki, C.-S. Retakes [online]. February 2022, [Cited 2024-01-20]. Available from: https://counterstrike.fandom.com/wiki/Retakes

[26] Wiki, C.-S. Danger Zone [online]. January 2024, [Cited 2024-01-20]. Available from: https://counterstrike.fandom.com/wiki/Danger_Zone

[27] IT, F. ABOUT FACEIT [online]. 2017, [Cited 2024-01-20]. Available from: https://corporate.faceit.com

[28] Koren, Y.; Bell, R.; et al. Matrix Factorization Techniques for Recommender Systems. *Computer*, volume 42, no. 8, 2009: pp. 30–37, doi: 10.1109/MC.2009.263.

[29] Suh, C. Information Theory of Matrix Completion. *CoRR*, volume abs/1402.4225, 2014, 1402.4225. Available from: http://arxiv.org/abs/1402.4225

[30] Funk, S. Netflix Update: Try This at Home [online]. December 2006, [Cited 2024-01-20]. Available from: https://sifter.org/~simon/journal/20061211.html

[31] Zhang, Y. An Introduction to Matrix factorization and Factorization Machines in Recommendation System, and Beyond. 2022, 2203.11026.

[32] Zhou, Y.; Wilkinson, D.; et al. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM '08, Berlin, Heidelberg: Springer-Verlag, 2008, ISBN 9783540688655, p. 337–348, doi:10.1007/978-3-540-68880-8_32. Available from: https://doi.org/10.1007/978-3-540-68880-8_32

[33] HLTV.org. HLTV About us [online]. January 2024, [Cited 2024-01-20]. Available from: https://www.hltv.org/about/

[34] Candès, E. J.; Recht, B. Exact Matrix Completion via Convex Optimization. *CoRR*, volume abs/0805.4471, 2008, 0805.4471. Available from: http://arxiv.org/abs/0805.4471

[35] Patro, S. G. K.; Sahu, K. K. Normalization: A Preprocessing Stage. *CoRR*, volume abs/1503.06462, 2015, 1503.06462. Available from: http://arxiv.org/abs/1503.06462

[36] Jannach, D.; Lerche, L.; et al. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, volume 25, 2015: pp. 427–491. Available from: https://api.semanticscholar.org/CorpusID:15621670

Appendix A

Contents of attachments

Visualisation of the contents of the enclosed media.

- `README.txt` ..... the description of this directory
- `src` ..... the source code directory
  - `README.txt` ..... the description for source code directory
    - `*.py` ..... the source code
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the Diploma thesis in PDF format
  - `thesis.ps` ..... the Diploma thesis in PS format
- `thesis` ..... the thesis source code
  - `images` ..... the images for the thesis
  - `*.tex` ..... the source code of the thesis