INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




OddAssist - Um sistema de recomendação de apostas em eSports

CARLOS MANUEL MAIA COELHO
Outubro de 2023

P.PORTO

OddAssist

An eSports betting recommendation system

Carlos Manuel Maia Coelho

1170527

Dissertation to obtain the Master's degree in Artificial Intelligence Engineering

Supervisor: Doctor Isabel Cecília Correia da Silva Praça Gomes Pereira, Coordinator Professor, Polytechnic of Porto - School of Engineering

Júri:

President:

Doctor António Constantino Lopes Martins, Assistant Professor, Polytechnic of Porto - School of Engineering

Vocals:

Doctor Isabel Cecília Correia da Silva Praça Gomes Pereira, Coordinator Professor, Polytechnic of Porto - School of Engineering

Doctor Paulo Sérgio dos Santos Matos, Assistant Professor, Polytechnic of Porto - School of Engineering

Porto, October 2023

«My method is different. I do not rush into actual work.
When I get an idea I start at once building it up in my imagination».

Nikola Tesla

Abstract

It is globally accepted that sports betting has been around for as long as the sport itself. Back in the 1st century, circuses hosted chariot races and fans would bet on who they thought would emerge victorious. With the evolution of technology, sports evolved and, mainly, the bookmakers evolved. Due to the mass digitization, these houses are now available online, from anywhere, which makes this market inherently more tempting. In fact, this transition has propelled the sports betting industry into a multi-billion-dollar industry that can rival the sports industry.

Similarly, younger generations are increasingly attached to the digital world, including electronic sports – eSports. In fact, young men are more likely to follow eSports than traditional sports. Counter-Strike: Global Offensive, the videogame on which this dissertation focuses, is one of the pillars of this industry and during 2022, 15 million dollars were distributed in tournament prizes and there was a peak of 2 million concurrent viewers. This factor, combined with the digitization of bookmakers, make the eSports betting market extremely appealing for exploring machine learning techniques, since young people who follow this type of sports also find it easy to bet online.

In this dissertation, a betting recommendation system is proposed, implemented, tested, and validated, which considers the match history of each team, the odds of several bookmakers and the general feeling of fans in a discussion forum.

The individual machine learning models achieved great results by themselves. More specifically, the match history model managed an accuracy of 66.66% with an expected calibration error of 2.10% and the bookmaker odds model, with an accuracy of 65.05% and a calibration error of 2.53%.

Combining the models through stacking increased the accuracy to 67.62% but worsened the expected calibration error to 5.19%. On the other hand, merging the datasets and training a new, stronger model on that data improved the accuracy to 66.81% and had an expected calibration error of 2.67%.

The solution is thoroughly tested in a betting simulation encapsulating 2500 matches. The system's final odd is compared with the odds of the bookmakers and the expected long-term return is computed. A bet is made depending on whether it is above a certain threshold. This strategy called positive expected value betting was used at multiple thresholds and the results were compared.

While the stacking solution did not perform in a betting environment, the match history model prevailed with profits form 8% to 90%; the odds model had profits ranging from 13% to 211%; and the dataset merging solution profited from 11% to 77%, all depending on the minimum expected value thresholds.

Therefore, from this work resulted several machine learning approaches capable of profiting from Counter Strike: Global Offensive bets long-term.

Keywords: sport betting, supervised learning, sentiment analysis, ensemble methods; counter-strike: global offensive

Resumo

É globalmente aceite que as apostas desportivas existem há tanto tempo quanto o próprio desporto. Mesmo no primeiro século, os circos hospedavam corridas de carruagens e os fãs apostavam em quem achavam que sairia vitorioso, semelhante às corridas de cavalo de agora. Com a evolução da tecnologia, os desportos foram evoluindo e, principalmente, evoluíram as casas de apostas. Devido à onda de digitalização em massa, estas casas passaram a estar disponíveis *online*, a partir de qualquer sítio, o que torna este mercado inerentemente mais tentador. De facto, esta transição propulsionou a indústria das apostas desportivas para uma indústria multibilionária que agora pode mesmo ser comparada à indústria dos desportos.

De forma semelhante, gerações mais novas estão cada vez mais ligadas ao digital, incluindo desportos digitais – *eSports*. *Counter-Strike: Global Offensive*, o videojogo sobre o qual esta dissertação incide, é um dos grandes impulsionadores desta indústria e durante 2022, 15 milhões de dólares foram distribuídos em prémios de torneios e houve um pico de espectadores concorrentes de 2 milhões. Embora esta realidade não seja tão pronunciada em Portugal, em vários países, jovens adultos do sexo masculino, têm mais probabilidade de acompanharem *eSports* que desportos tradicionais. Este fator, aliado à digitalização das casas de apostas, tornam o mercado de apostas em *eSports* muito apelativo para a exploração técnicas de aprendizagem automática, uma vez que os jovens que acompanham este tipo de desportos têm facilidade em apostar online.

Nesta dissertação é proposto, implementado, testado e validado um sistema de recomendação de apostas que considera o histórico de resultados de cada equipa, as cotas de várias casas de apostas e o sentimento geral dos fãs num fórum de discussão – HLTV. Deste modo, foram inicialmente desenvolvidos 3 sistemas de aprendizagem automática.

Para avaliar os sistemas criados, foi considerado o período de outubro de 2020 até março de 2023, o que corresponde a 2500 partidas. Porém, sendo o período de testes tão extenso, existe muita variação na competitividade das equipas. Deste modo, para evitar que os modelos ficassem obsoletos durante este período de teste, estes foram re-treinados no mínimo uma vez por mês durante a duração do período de testes.

O primeiro sistema de aprendizagem automática incide sobre a previsão a partir de resultados anteriores, ou seja, o histórico de jogos entre as equipas. A melhor solução foi incorporar os jogadores na previsão, juntamente com o *ranking* da equipa e dando mais peso aos jogos mais recentes. Esta abordagem, utilizando regressão logística teve uma taxa de acerto de 66.66% com um erro expectável de calibração de 2.10%.

O segundo sistema compila as cotas das várias casas de apostas e faz previsões com base em padrões das suas variações. Neste caso, incorporar as casas de aposta tendo atingido uma taxa de acerto de 65.88% utilizando regressão logística, porém, era um modelo pior calibrado que o modelo que utilizava a média das cotas utilizando *gradient boosting machine,* que exibiu uma taxa de acerto de 65.06%, mas melhores métricas de calibração, com um erro expectável de 2.53%.

O terceiro sistema, baseia-se no sentimento dos fãs no fórum HLTV. Primeiramente, é utilizado o GPT 3.5 para extrair o sentimento de cada comentário, com uma taxa geral de acerto de 84.28%. No entanto, considerando apenas os comentários classificados como conclusivos, a

taxa de acerto é de 91.46%. Depois de classificados, os comentários são depois passados a um modelo *support vector machine* que incorpora o comentador e a sua taxa de acerto nas partidas anteriores. Esta solução apenas previu corretamente 59.26% dos casos com um erro esperado de calibração de 3.22%.

De modo a agregar as previsões destes 3 modelos, foram testadas duas abordagens. Primeiramente, foi testado treinar um novo modelo a partir das previsões dos restantes (*stacking*), obtendo uma taxa de acerto de 67.62%, mas com um erro de calibração esperado de 5.19%. Na segunda abordagem, por outro lado, são agregados os dados utilizados no treino dos 3 modelos individuais, e é treinado um novo modelo com base nesse conjunto de dados mais complexo. Esta abordagem, recorrendo a *support vector machine,* obteve uma taxa de acerto mais baixa, 66.81% mas um erro esperado de calibração mais baixo, 2.67%.

Por fim, as abordagens são postas à prova através de um simulador de apostas, onde sistema cada faz uma previsão e a compara com a cota oferecia pelas casas de apostas. A simulação é feita para vários patamares de retorno mínimo esperado, onde os sistemas apenas apostam caso a taxa esperada de retorno da cota seja superior à do patamar.

Esta cota final é depois comparada com as cotas das casas de apostas e, caso exista uma casa com uma cota superior, uma aposta é feita. Esta estratégia denomina-se de apostas de valor esperado positivo, ou seja, apostas cuja cota é demasiado elevada face à probabilidade de se concretizar e que geram lucros a longo termo. Nesta simulação, os melhores resultados, para uma taxa de mínima de 5% foram os modelos criados a partir das cotas das casas de apostas, com lucros entre os 13% e os 211%; o dos dados históricos que lucrou entre 8% e 90%; e por fim, o modelo composto, com lucros entre os 11% e os 77%.

Assim, deste trabalho resultaram diversos sistemas baseados em machine learning capazes de obter lucro a longo-termo a apostar em *Counter Strike: Global Offensive*.

Palavras-chave: apostas desportivas, aprendizagem supervisionada, análise de sentimento, métodos ensemble, counter-strike: global offensive

Acknowledgements

First, to everyone who contributed to the development of this work, whether by giving suggestions or simply lending an ear.

To my supervisor, Doctor Isabel Cecília Correia da Silva Praça Gomes Pereira, for the guidance and support throughout the development of this work, as well as time invested in advising and revising this work.

To previous and current co-workers at Unilabs, for the comprehension and input throughout the duration of this project.

To family and friends, without your support, I would be as stable as the new Counter Strike 2 release.

Index

List of Figures

List of Tables

Acronyms

List of acronyms

AE	Auto-Encoder
ADP	Adaptive Dynamic Programming
AI	Artificial Intelligence
API	Application Programming Interface
BAE	Bookmakers' Odds Average
BERT	Bidirectional Encoder Representations from Transformers
BOE	Bookmakers' Odds Encodings
BS	Brier Score
boX	Best-Of-X series
CART	Classification And Regression Tree
CNN	Convolutional Neural Network
CS:GO	Counter-Strike: Global Offensive
eSports	Electronic Sports
EV	Expected Value
GAE	Guess x Accuracy Encoding
GBM	Gradient Boosting Machine
GRU	Gated Recurrent Unit
LAN	Local Area Network
LL	Logarithmic Loss
LLM	Large Language Model
MDP	Markov Decision Process
NBC	Naïve Bayes Classifier
NN	Neural Network

PBE	Players Binary Encoding
POL	Polynomial classifier
KNN	K Nearest Neighbours
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
OP	OddsPedia
RBF	Radial Basis Function
RFC	Random Forest Classifier
RFR	Random Forest Regression
RNN	Recurrent Neural Network
ROBERTa	Robustly Optimized BERT Approach
SVM	Support Vector Machine
SVR	Support Vector Regression
TBE	Team Binary Encoding

1 Introduction

This document details the process of developing a recommendation tool for pre-game sports betting. What was thought to be a selfish field where knowledge sharing was not commonplace, turned out to be a subject rising in popularity amongst scholars. Still, there are many challenges that arise from acquiring and curating the data needed for the project, to designing, training, and testing a machine learning model that would compete with the bookmakers.

On this chapter the scope of the project is contextualized. The problem, approach and proposed solution are also defined as well as presented the structure of the rest of the document.

1.1 Contextualization

The proposed solution "OddAssist" is self-proposed by the author and is a module of BetaCS – a sports betting recommendation platform born in 2021 as a fantasy betting line-up optimizer. This project focuses on a subset of sports betting – competitive video games, namely Counter-Strike: Global Offensive (CS:GO). This focus is due to the latter being a sport where teams face each other multiple time per year resulting in a lot of head-to-head data available when compared to traditional sports such as football, or even other videogames. Furthermore, the discussion about the matches is largely centralized in a single forum – HLTV [1] – which can immensely boost the sentiment analysis tasks' capabilities.

Nevertheless, it is a niche and there is a lack of datasets readily available to continuously train the models. Henceforth, a sizeable component of the project focuses on extracting, compiling and merging the data.

1.2 Problem Statement

It is widely accepted that sports betting has been around for as long as sports themselves. Dating back to the first century, circuses would host chariot races and the fans would bet on the outcome, similarly to current day horse racing bets [2], [3]. It should not come as a surprise that as sports and technology evolved, so did its betting scene. One of the biggest factors was the digitalization of the industry through online bookmakers, propelling it to a multi-billion-dollar industry expected to grow to over 180 billion dollars by 2030, being comparable to the sports industry itself [4]–[6].

The house always wins. Mathematically, that's a fact. The odds are meticulously created considering factors such the prior form of the players, statistics, historical precedents, and experts' opinion [7]. A search on Google Scholar for the query 'Sports betting machine learning' yielded nearly 34 000 results [8], therefore, considering the amount of public research on the topic, it is highly likely that the multi-billion-dollar industry, having more resources and more data available, also has their own private machine learning driven approaches to predict the outcome of a match.

Furthermore, the bookmakers not only takes a cut form the odds but can also adjust them to attract punters, meaning that if the initial estimate is decent and the odds are not fixed, the house always has a margin of profit, regardless of the outcome [7], [9], [10].

Moreover, it has been reported that these online platforms are increasingly using artificial intelligence to keep the bettors addicted, by offering special promotions and targeted advertisement. As stated by Brian, a digital marketer for the gambling industry [11]:

> *"It's like a science, it's not just random advertising messages, the whole thing is personalised, and data-driven customer profiles are constructed from gamblers' behaviour."*

However, this does not mean that it is impossible to beat the odds and turn a profit from this kind of venture. There is always a winning outcome, so the house winning, and a punter winning are not mutually exclusive. The keys are figuring out which (if any) odd is worthwhile, realising that the correct odd might not always win and to not be baited in by special promotions or juicy multipliers. By the nature of being data driven, machine learning is resilient to the latter.

Nevertheless, even if there was a model that could perfectly predict the outcome of every match at a given time, that accuracy is only valid while the teams maintain their current form, players, and supporting staff, meaning that a model could quickly become obsolete. A human who watches the sport could probably guess the impact a player has on a team considering context, but how can that intricate knowledge be replicated by artificial intelligence?

More recently, a new kind of sport – electronic sports (eSports) – has been rising in popularity amongst young men, even surpassing established traditional titles [12]. Esports are the highest level of competition within videogames and Counter-Strike: Global Offensive (CS:GO) is one of the pillars of this industry, with over $15 million in combined prize pools and over 2 million peak concurrent viewers during 2022 [13].

The digitalization of the bookmakers as well as the growth of eSports combines into a very tempting market for the younger generation, which is exactly where OddAssist comes in – eSport betting recommendation.

1.2.1 Objectives

The main goal is for OddAssist be an eSports betting recommendation system that could be integrated into the BetaCS platform and would result in long-term profit. As briefly explained above, there are several mechanisms in place to ensure the bookmaker profits regardless of the outcome of a match. To develop a solution capable of exploiting these mechanisms, it is crucial to understand them as well as research how they have been taken advantage of in the past.

The proposed solution is a system that will compute the odds for each side of a given matchup, which will then be compared to the bookmakers' and, if the latter odds are favourable, a bet would be made accordingly, in a process known as positive expected value betting [14].

As previously mentioned, bookmakers use a combination of: **prior form** – how well each player has been performing recently; **statistics** – metrics that characterize a player's strengths and weaknesses; **historical precedents** – how well players mesh together, team synergy, rivalries, etc; and **experts' opinion** – which as the name suggests, is a subjective view on a matchup.

OddAssist will aim to replicate the process of the odd creation, using different models to emulate the methods used by traditional bookmakers. The proposed solution will therefore be an ensemble approach composed of 3 different models – match history model (I); bookmakers' odds compiler model (II); and discussion sentiment analysis model (III) – which are then fed into a fourth model to compute the final odd. As exposed in Table 1, model (I) will be responsible for considering prior form, statistics and historical precedent; on the other hand, expert's opinions will be captured by models (II) and (III), the first will compile the odds from different bookmakers which, in this field, can be considered as experts; in contrast, model (III) will capture the general fan sentiment going into a match, which might give nuance to the model.

Table 1 – Metrics and models used for the generation of odds by bookmakers and OddAssist

Traditional Bookmaker	OddAssist
Prior form	Model I
Statistics	Model I
Historical precedent	Model I
Expert's opinions	Model II and Model III

There are multiple valid approaches for a machine learning model to learn from the data. Consider the goal of having a model learn to contemplate prior form of a team. One approach would be to feed that information directly to the model through a feature like "last 5 matches win rate". Other option would be to overvalue more recent results when training the model, although the latter would become obsolete faster. Therefore, a more detailed insight into how each model tackles learning these interactions will be discussed on later chapters.

Furthermore, as stated previously the target market will not be traditional sports betting, but rather competitive video games' betting, focusing on CS:GO. This was chosen as the case study because in CS:GO HLTV is unanimously used by the community and contains information about all the matches that have ever happened at a professional level. Furthermore, and within each

match thread, the discussion is open to the fans. In other words, HLTV is a discussion forum organized by match, whereas in other sports the discussion is usually taken to Twitter [15] or Reddit [16] which would need further data processing to associate an opinion to specific match.

This focus potentiates the sentiment analysis task but comes at the expense of readily available datasets and existing state-of-the-art solutions. Therefore, the study of the game CS:GO as well as its betting scene is of utmost importance to grasp how transferable the research made on traditional sports betting is. Thus, briefly, the success of the development of OddAssist is reliant on the following tasks:

- Study how odds are calculated by the bookmakers;
- Study approaches that have successfully beaten bookmakers;
- Study similarities between CS:GO and traditional sports;
- Study of data retrieval techniques;
- Study of the literature of machine learning techniques;
- Study of the literature of sentiment analysis;
- Study of the literature of sport match outcome prediction models based on historic data and/or bookmakers' predictions;
- Experimentation with multiple pre-processing and processing techniques for each model;
- Evaluation of the performance of the tested models and techniques.

1.2.2 Approach

There is no denying that Artificial Intelligence is increasingly popular, boasting an ever-expanding number of machine learning models and techniques. When this document was first created, in late 2022, OpenAI had just launched their new prototype of a chatbot – ChatGPT [17] – that displayed unheard levels of context recognition and explainability in its line of thinking, taking the world by storm. As of June 2023, the state of the art has further evolved, with more companies launching their own competitors as well as OpenAI improving its models. As such, it is of utmost importance that a state of the art analysis takes place prior to the development of the machine learning models. Nevertheless, the goal of this project is not to merely replicate work that has previously been proven to work.

Rather, the goal is to improve it further by analysing the problem, designing a solution curated to the domain and comparing it to previously established approaches – quoting Steven Jeffes, a marketing and business expert:

> *"Innovation is the unrelenting drive to break the status quo and develop anew where few have dared to go."*

Henceforth, the quality of the final solution is dependent on a deep understanding of the domain as well as machine learning techniques that are the most suitable to the problem at hand. As such, the analysis will consist of literature that responds any of the following research questions:

- RQ1 – How do bookmakers generate their odds?
- RQ2 – Which approaches already exist that consistently beat bookmakers?
- RQ3 – How comparable is CS:GO to traditional sports?
- RQ4 – Which machine learning models or techniques have been used to predict the outcome of a matchup using match history?
- RQ5 – How have bookmaker odds been used to predict the outcome of a match and which machine learning models or techniques were used?
- RQ6 – Which machine learning models or techniques have been used to analyse fan sentiment?
- RQ7 – How to represent a team over time?

The research conducted was inspired by the systematic review approach and is represented in Figure 1. Firstly, research questions were created and searched in multiple scientific databases, namely IEEE Xplore [18], Science Direct [19], Web of Science [20] and b-on [21]. From those results, it was selected few documents with the highest number of citations, prioritizing staple references such as [22]. Surveys and reviews were also prioritized due to exploring vast number of concepts. Lastly, recency was also considered as an inclusion criterion. Within the selected documents, some of the concepts and references were individually explored both through the databases and academic search engines (arXiv [23] and Google Scholar [8]), resulting in a concept and reference snowballing and a wider knowledge pool.

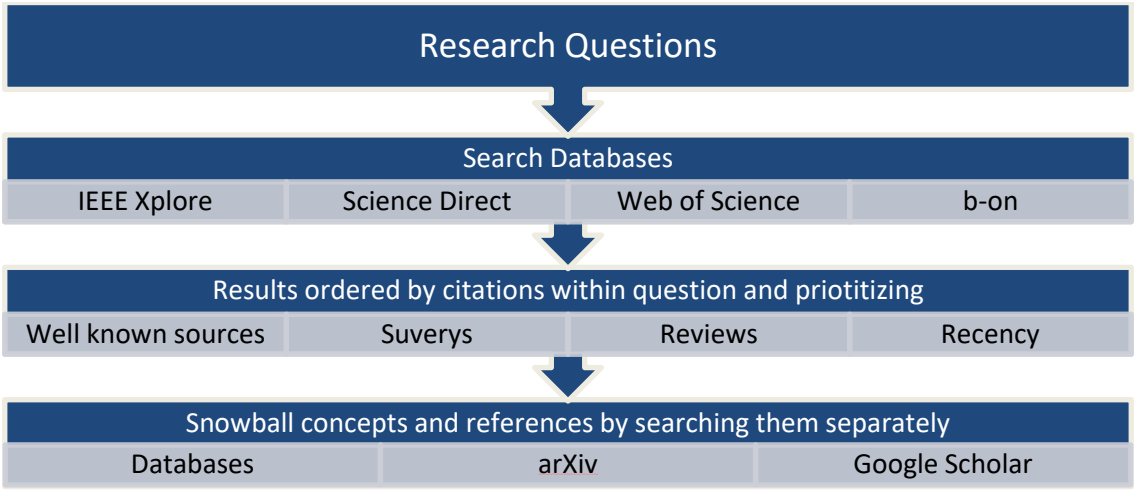


Figure 1 – Research methodology

1.2.3 Contributions

OddAssist is, to the best of the author's knowledge, unique in its approach of using a combination of previous outcomes, bookmakers odds and sentiment analysis to predict the outcome of a match, therefore pioneering research on how to combine these different types of data and its results. Furthermore, it also contributes to the respective fields by providing:

- An analysis of existing approaches for predicting the outcome of a game using either match history, bookmakers' odds or fan sentiment;
- An analysis and employment of different data extraction techniques used to create the datasets;
- A literature review on machine learning;
- Comparison between different pre-processing and processing experiments with the promising techniques reviewed in the literature;
- The results of experiments with neural networks that use multiple input layers feeding into each other, creating explicit feature interaction.

Henceforth, regardless of the success of the proposed approach, the work still contributes to the fields of sports betting and sentiment analysis by providing an overview on what has been made and whether it worked. Similarly, the literature review on machine learning provides a comprehensive breakdown of both older and state-of-the-art techniques. All of these can be used as the basis for future works.

1.3 Dissertation Structure

In this subsection of the Introduction chapter, the structure and content of this document is briefly detailed. It is divided in 5 chapters, structured with the intent of facilitating the reading of each chapter.

The first and present chapter is an introduction describing the context in which the project emerged. Additionally, the problem statement is detailed and outlined the objectives, approach and contributions of the work.

In the second chapter, the knowledge domain is contextualized. More specifically, within sports betting, the process of calculating and exploiting an odd is detailed. Core concepts CS:GO's competitive scene are also explained and outlined similarities with traditional sports.

The third chapter delves into the state of the art of machine learning, detailing how each technique works at a high level and explored key concepts for the construction of a reliable machine learning model.

Chapter number four is a review of similar work previously done related to sports match outcome prediction and sentiment analysis.

In the fifth chapter, the methods and materials used by this work are exposed, covering web scraping techniques, dataset research and attainment and tools used. The evaluation metrics used to analyse the models' performances are defined and explored ethical aspects and threats within machine learning.

Chapters six, seven and eight discuss the steps of producing machine learning models for match outcome prediction through match history, bookmaker odds and fan sentiment, respectively. Each chapter starts with the exploration of the raw data, followed by its curation to obtain the final datasets. Then, pre-processing experiments take place revealing how the data affects the prediction, and finally, final data representation and machine learning technique is chosen as the final solution.

In the ninth chapter, a final solution is proposed. It explores stacking the previous models as well as creating a new, stronger, learner. A betting simulation is done to evaluate how each of the proposed solutions would behave in the real world.

At last, the tenth chapter, presents the final considerations and conclusions and is explored possible future developments for this project.

2 Domain Contextualization

In this chapter, the first three main tasks presented in Objectives are addressed. Within sports betting, first, bookmakers' odds decomposed and explained. Next, proven methods to beat the odds are exposed. Later, the game of Counter-Strike: Global Offensive is briefly explained, finally, parallels with traditional sports are traced.

2.1 Sports Betting

Sports and betting are seemingly inseparable. Many people thrive on the risk and the prospect of a big win. Even in the first century, circus would host chariot races and allow the fans to bet on the outcome, similarly to current day horse racing bets [2], [3]. With the digitalization of betting exchanges, or bookmakers, this prospect is closer than ever, just a click away, which is one of the main reasons this industry expected to grow to over 180 billion dollars by 2030 [6], even being comparable to the sports industry itself [4].

2.1.1 Odds

A bet is the act of putting a stake on an event with a given odd. The stake is the amount of money bet, whereas the odd represents the likelihood of that event happening. In fact, the odd is the multiplier by which the stake is returned in the case of winning [24]. Mathematically, an odd of an event **E** can be calculated from the probability [25]. As represented in equation (1), the odd of an event (Odd_E) is equal to the probability of E happening ($Probabity_E$) divided by the probability of E not happening.

$$Odd_E = \frac{Probabity_E}{1 - Probabity_E} \leftrightarrow Probabity_E = \frac{Odd_E}{1 + Odd_E} \quad (1)$$

Regardless, within the context of this document, the term odd refers to the multiplier of an event. In other words, an odd is the factor by which the money staked is returned. Therefore, the odd follows a different formula than (1). When calculating a betting odd, it is the odd of an event not happening, which ensures that the less likely the event is, the higher the multiplier. To that odd is then added 1, as to ensure that winning returns more than the initial stake [10]. Thus, as represented in equation (2), the betting odd of an event ($bOdd_E$) is equal to 1 plus the division between the probability of E not happening ($Probabity_{\sim E}$) and the probability of E

happening. This expression can be shortened to 1 divided by the probability of E happening ($Probabity_E$).

$$bOdd_E = 1 + \frac{Probabity_{\sim E}}{1 - Probabity_{\sim E}} \leftrightarrow \quad (2)$$
$$bOdd_E = \frac{1}{Probabity_E} \leftrightarrow Probabity_E = \frac{1}{bOdd_E}$$

Mathematically, if the odd is fair, there is no profit to be made. Consider a coinflip - an event that has a 50% chance of happening in which the fair odd is 2. If the coin were to be flipped 100 times, and 1€ was staked on heads each time, it is likely that it will land on head and tails 50 times each. This means that in half of the coin flips the stake would be doubled, whereas in the remaining, it would be completely lost, therefore nulling the previous profit. In this scenario, the long-term return of that bet can be represented by equation (3). The final balance ($Balance$) is the stake ($Stake$) multiplied by the probability of the event happening ($Probabity_E$) and the betting odd of that event ($bOdd_E$). Since the probability is the inverse of the odd, they cancel each other, meaning that the final balance will be the staked value that was initially invested, netting a profit of 0.

$$Balance = Stake \times Probabity_E \times bOdd_E \leftrightarrow \quad (3)$$
$$Balance = Stake \times Probabity_E \times \frac{1}{Probabity_E} \leftrightarrow$$
$$Balance = Stake$$

For this venture to be profitable, in a real-world scenario, the bookmaker also takes a cut, by offering lower odds. As displayed in equation (2), It is possible to calculate the implied probability of an event from its odd. Consider Table 2, if the sum of the implied probabilities was computed, the result would exceed 100%. This overestimation is the cut of the bookmaker. As reflected in equation (4), it is also possible to compute the percentual cut of the bookmaker ($Cut_{\%}$) by subtracting to 1 the division of 1 by the sum of the implied probabilities ($\sum Probability_{implied}$).

$$Cut_{\%} = 1 - \frac{1}{\sum Probability_{implied}} \quad (4)$$

Table 2 – Example of odds for an event

	Odd	Implied probability
Event takes place	1.20	83.3%
Event does not take place	5.5	18.1%
Cut	-	1.4%

2.1.2 Beating the Odds

As proven in the section above, unless the bookmaker choses to have a negative cut and, consequently, lose money, there is no guaranteed approach that consistently beats the bookmaker. Regardless, even though beating one bookmaker consistently is unlikely, there are means of exploiting competition between bookmakers to boost or even guarantee profit long term [26].

Line Shopping

Overall, in this online era, bookmakers provide similar odds to prevent being exposed to big losses on a misplaced odd. Regardless, the bookmaker odds can be outdated, such as in the case of a star player's injury before the match and unless manually altered, the odds could take some time to find the new balance. Furthermore, bookmakers often willingly take a worse odd to attract punters and balance out the stakes on each side of the bet or even as a marketing exercise [27]. Line shopping is, therefore, the act of being up to date with the odds offered by the different bookmakers and either choosing the best one, or simply waiting for an opportunity that is worthwhile. There are many sites built for this purpose, such as OddsShark [28].

Arbitrage Betting

As discussed in previous section, a combined implied probability above 100% means the bookmaker is taking a cut of the stake. In contrast, an implied probability under 100% means the opposite, that the bookmaker is losing money. Therefore, arbitrage betting builds upon line shopping and aims to find an opportunity where, by combining different bookmakers' odds, finding a combined implied probability below 100% is possible consequentially guaranteeing a profit, regardless of the outcome [26]. Websites like OddsPedia compile odds and expose those opportunities free of charge [29]. On web archive it was found the opportunity presented on Figure 2, with over 75% guaranteed profit. Even considering this might have been a fluke, it is possible to search the archive for multiple examples of arbitrage betting opportunities with returns over 20%.

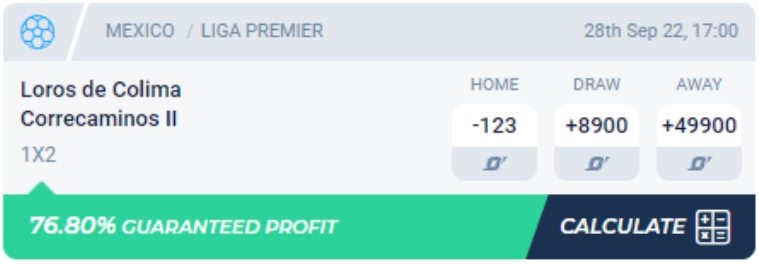


Figure 2 – OddsPedia Arbitrage betting example (from [29])

Positive Expected Value Betting

Positive Expected Value Betting consists in finding and taking advantages of inefficiencies within the market and capitalizing on them. Expected value defines the margin of profit of a given event long term. As represented in equation (5), the percentual expected value ($EV_{\%}$) can be obtained by computing the difference between the bookmaker odd ($bOdd_{bookmaker}$) and the actual, computed odd ($bOdd_{Actual}$) and then dividing that difference by the actual odd and multiplying the result by 100.

$$EV_{\%} = \frac{bOdd_{Implied} - bOdd_{Actual}}{bOdd_{Actual}} \times 100 \quad (5)$$

As an example [30], consider a weighted coin that lands heads 53% of the times and a bookmaker that offers odds of 2.0 on each side. Given equations (2) and (5), the expected value of this scenario can be computed as demonstrated in equation (6). It can be said that the coin flip has an expected value of 6% and would result in a profit long term. In other words, long term, betting on heads would yield a 106% return. Therefore, in practice, positive expected value betting relies on finding bets whose odds are higher than the actual fair odd and profit long-term [31].

$$bOdd_{Actual} = \frac{1}{0.53} = 1.887 \quad (6)$$

$$EV_{\%} = \frac{2.0 - 1.887}{1.887} \times 100 = 6\%$$

While arbitrage betting is risk free and bets on both sides, this approach offers better margins of profit by relying on a mathematical edge usually calculated from the bookmakers' odds. Platforms such as OddsJam are automatically updated with positive expected value bets on multiple bookmakers and provide that information for free, although, according to their website, the best value bets are saved for subscribers [14].

2.2 Counter-Strike: Global Offensive

The videogame Counter-Strike: Global Offensive (CS:GO) was developed by Valve and is a team based first person shooter. The teams are composed of 5 players and take turns attacking and defending. Attackers have as a goal planting a bomb in a designated place – the bombsite – and defending it until it explodes, after 40 seconds. In contrast, the defenders' objective is to stop the attackers by either preventing the bomb from being planted or defusing the bomb. If either side loses all players before the bomb is planted, they lose the round. Still, once the bomb has been deployed, the round only ends when it is defused or explodes [32].

At the start of the game, the teams play a dummy round to decide who attacks first. The game is divided into two halves of 15 rounds, and teams only swap sides once the half is over. Within the same half, surviving players can carry equipment to the next round, meaning the game does not fully reset until the half is over, which in turn leads to a lot of strategic decision making on when to buy new equipment, commit to a fight and even suicide. Since the game has two 15-round halves, the game ends once a team wins 16 rounds.

In the case of a 15-15 tie, extra rounds called overtime are played, where each team takes turns attacking and defending for 3 rounds. In the case of another tie, consecutive overtimes are played until a team wins 4 out of the 6 rounds.

2.2.1 Competitive Scene

Electronic Sports (eSports) is the term used for playing video games competitively. This is a fast-growing industry that is even surpassing some traditional sports among young men [12], [33]. CS:GO is one of the primary titles within eSports and in 2022, hosted 486 tournaments with over $15 million of combined prize pool and a peak viewership of over 2 million concurrent viewers [13].

Being such an sought after venture, CS:GO competitive scene is not taken lightly, there are support systems in place by teams and tournament organizers alike. One of the biggest tournament organizers, ESL list 47 different career paths on their website [34]. In the same vein, every professional team has a coach that help players improve in and out of the game [35]. This means that the competitive scene is more heavily regulated than the casual player experience.

Maps and Veto

A map is the game level on which the game takes place and is a big factor on the outcome of the game, as teams have different strengths and weaknesses that can be exploited [32], [36]. In contrast with casual play, the competitive scene is restricted to a 7-map pool at any given time. To select a map, a phase of picking and banning maps – the veto – takes place, meaning that preparation and anticipation are crucial and that the match starts way before the keyboards and mouses are plugged.

Online versus Local Area Network

Events, or parts of events can be run either online or on Local Area Network (LAN). As the names suggest, the former' matches are played from anywhere in the world and have latency whereas the latter are played in a single location and often have crowds cheering the players on.

Match format

Within competitive play, a match is a set of games where the team who wins the most games is declared the winner. As reflected in the match page of HLTV [1], the most common format is a best-of-three (bo3) series which is a set of 3 different maps where the first team to claim 2 maps wins. Though, best-of-one matches are common on the earlier stages of the tournaments and best-of-five (first to 3 maps wins) is commonly used in the final stages, as to better assess teams map pool depth. Within this document, the terms format and boX will be used interchangeably.

Events

CS:GO's competitive scene composed of several tournaments circuits. Different tournament organizers have their own set of events (circuit) [37], [38], that serve as qualifiers for their final event, usually a Major. A Major is a special kind of LAN event that is sponsored by Valve itself with a prize pool of at least $1 million, and crowded stadiums of fans cheering the teams [39]. Majors are considered the peak of the sport and is the most remarkable individual achievement a team can accomplish.

HLTV

HLTV[1] is a website that is the main hub for competitive CS:GO. It has detailed information about every competitive match since the introduction of the game. Likewise, it has a section where the upcoming matches are presented and is also possible to explore each matchup, providing information about every player, team maps stats, recent team history, odds and, most importantly, a comment section where the community actively discusses their thoughts about the matchup. This page is also available for previous matches, although some of the features, namely, the odds and map stats, are not available.

2.2.2 Similarities with Traditional Spots

CS:GO is a team based eSport which inherently makes it dependent on interactions between teammates and opponents. This kind of interaction is highly dependent on current form, but there is evidence that suggest that results between teams tend to be consistent throughout the years, giving birth to rivalries and mental edges [40], [41]. Therefore, it is comparable to traditional sports on the premise that CS:GO also relies on player interaction and matchup history.

Additionally, CS:GO's competitive scene has parallels to traditional sports, the overall structure is the most similar to Formula1's Grand Prix Series, but mainstream sports such as basketball and football offer tournaments like the NBA Playoffs and UEFA Champions League, that are akin to Major championships for the teams involved [42], [43].

2.3 Chapter Conclusions

In this chapter, a comprehensive overview of key concepts related to sports betting is done, focusing on bookmakers' odds and methods to potentially gain the edge. More specifically, it is demonstrated how the odds can be translated into the likelihood of an event happening and how they play a critical role in betting strategies.

Three betting strategies that build on each other are explored, namely, line shopping, arbitrage betting and positive expected value betting. While no method ensures success, they provide avenues for capitalizing in potential market inefficiencies.

Counter-Strike: Global Offensive's competitive environment is well structured and regulated, with numerous tournaments and significant following. Thus, it shares similarities with traditional sports not only in team and player dynamics and significance of historical matchups, but also in its competitive scene's formats.

3 Machine Learning

This chapter addresses the state of the art of Machine Learning (ML), exploring different techniques and key concepts to building a reliable ML model. Artificial Intelligence (AI) is a term generally used to describe the training of a machine to do a specific task [44]. Within the field of AI, ML is one of the subsets that has risen the most in popularity in the past decade [45] and can be defined as the ability of a computer to learn from a set of provided data to perform a task, without explicitly being told how to do it. If learning can be defined as the process of turning experience into knowledge, then a machine can learn by performing tasks (experience) and generating its own set of rules by recognizing patterns in the data (knowledge) [46]–[48].

There are often multiple variations of a given ML technique, nonetheless, the aim of this section is to provide a comprehensive overview on how these techniques operate at a high level.

ML techniques can be divided into four categories, supervised, semi-supervised, unsupervised and reinforcement learning [46], [49]. These categories describe how each of the techniques train the models and are usually dictated by the type of data available. Figure 3 is a visualization of the first three types of learning and will be further explained within each subsection by using as an example the goal of classifying which continent a pair of coordinates belong to.

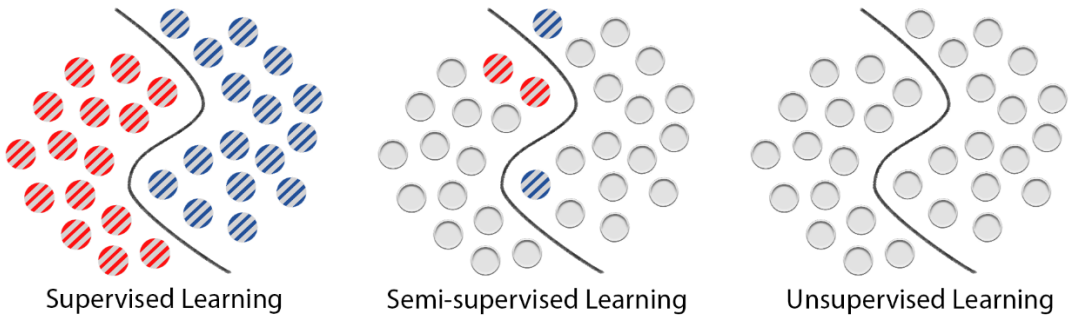


Figure 3 – Visualisation of supervised, semi-supervised and unsupervised labelling on a training dataset (adapted from [49])

3.1 Supervised Learning

Supervised Learning is a subset of ML where the expected output of a given input is known during training, i.e., the dataset is labelled. Considering the continent coordinates example mentioned previously, each pair of coordinates is correctly mapped to a continent. In this analogy, the model takes those labelled pairs of coordinates and defines the borders of each continent for future classification, as represented in the figure.

Supervised learning algorithms could be further divided into two categories, depending on whether the output is continuous or categorical, respectively, regression and classification [50]. The most common supervised learning ML techniques [48], [50] will be enumerated in the following subsections.

Linear and Polynomial Regression

Linear Regression and Polynomial Regression are the simplest supervised learning technique and consist in computing a polynomial expression that best matches a list of continuous points, as visualized in Figure 4. This technique is readily available in nearly any scientific calculator, such as the Texas Instrument's 84 [51].

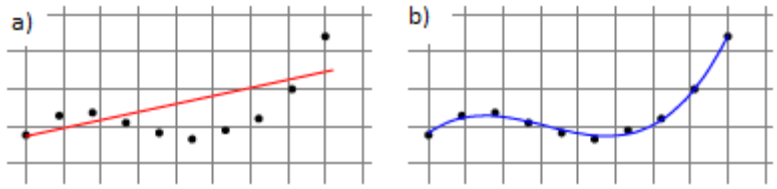


Figure 4 – Visualization of Linear (a) and Polynomial (b) regression (adapted from [48])

Logistic Regression

Although this technique is called Logistic (or Logit) Regression (LR), it is usually used for classification tasks. Through a sigmoid function, it computes the output of a linear function to the interval of [0,1]. This final output can be seen as the probability of a given class being true, as depicted by the example in Figure 5, the probability of passing an exam depending on the number of hours studied.

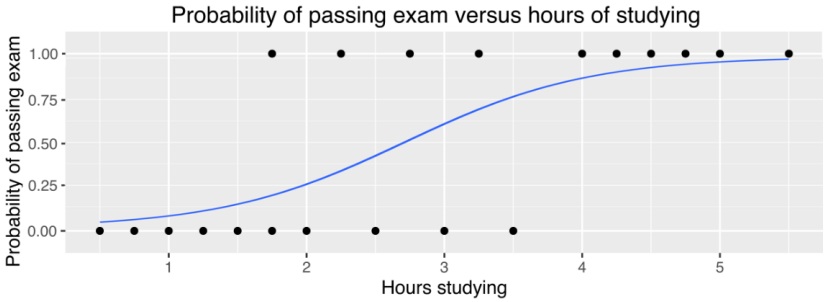


Figure 5 – Example of Logistic Regression (adapted from [52])

Support Vector Machine

Considering n as the number of features of a dataset and support vector as a list of features of a given point, Support Vector Machine (SVM) is a technique that uses the support vectors as coordinates to map each point into a n-dimensional space and segregates them, using (n-1)-dimensional hyperplanes. This concept can be easily visualised using a 2-dimensional space, such as the one in Figure 6. In this figure, it is intuitive that the dashed line is better than the green line at separating the two classes (a). To represent this intuition, SVM uses the concept of margins, depicted in (b) and (c) that penalise the solution (line) for having points in between their margins. This technique can be used for both regression, Support Vector Regression (SVR), and classification, SVM, although it is better suited for the latter.

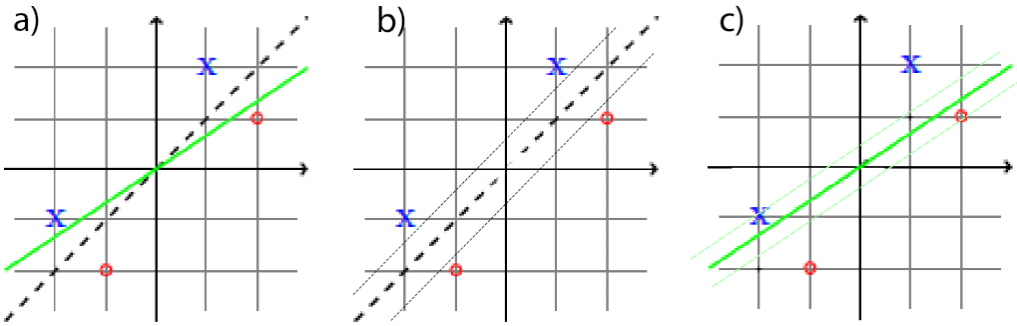


Figure 6 – 2-dimensional SVM (adapted from [48])

Decision Trees and Random Forest

Decision Tree is a simple, versatile, and comprehensive technique that consecutively divides the dataset according to its features, as represented in Figure 7. The training consists in finding and tuning the most relevant questions for the given dataset. Using the example in the figure, the training concluded the feature weather had the most impact on the decision of walking or taking a bus and is therefore the root node. Similarly, on a sunny day, the number of minutes to use as a threshold for walking versus taking the bus was continuously tuned and found that 30 was the best choice.

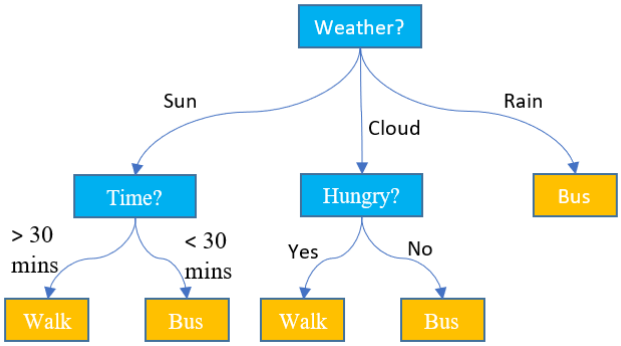


Figure 7 – Decision Tree for choosing a mean of transportation [53]

Classification Tree and Regression Tree are subsets of the Decision Tree that are used for classification and regression, respectively. Classification And Regression Trees (CART) are yet another subset that, as the name suggests, is capable of both categorical and continuous predictions.

Random Forests, on the other hand, is an ensemble of multiple Decision Trees that also can be used for classification and regression. It has the advantage of being more resilient to overfitting due to each Decision Tree being imbued with randomness. The final decision is made through processes like majority vote or mean. These can be Random Forrest Classifiers (RFC) or Random Forest Regressions (RFR).

Gradient Boosting Machine

The Gradient Boosting Machine (GBM) are also an ensemble of weak learners such as decision trees, but differ from the random forest technique because in a GBM, the learners are trained in sequence, having each new learner attempt to correct the prediction error of the previous learners, as demonstrated in Figure 8 [54].

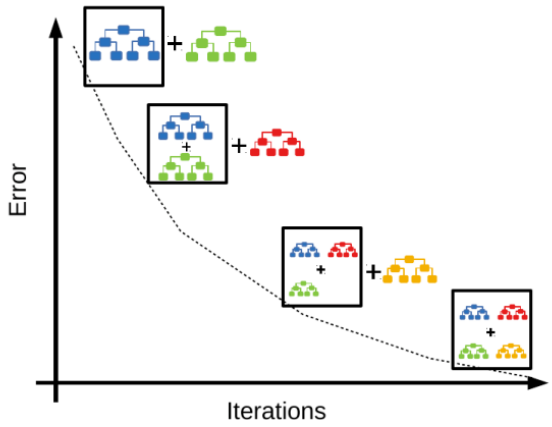


Figure 8 – Sequential addition of new learners [54]

K Nearest Neighbours

K Nearest Neighbours (KNN) consists in labelling a point according to its K nearest neighbours. Consider the 2-dimensional plane in Figure 9, where the features are the x and y coordinates of a given point. When K is set to 3, the model would classify the new instance as Orange whereas when K is set to 7 the instance would be classified as Black. In this technique the training lies in finding the optimal K.

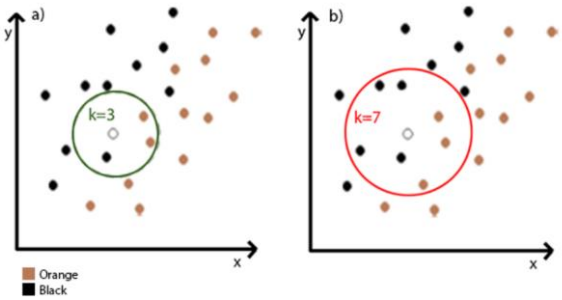


Figure 9 – Visualisation of KNN for (a) K=3 and (b) K=7 (adapted from [50])

Naïve Bayes

Naïve Bayes is a family of statistical classification techniques based on the Bayes Theorem, such as the Naïve Bayes Classifier (NBC). These are simplistic yet often yield surprising results. Table 3 represents a dataset for a typical application of this technique – classifying whether an email is spam based on its words. Due to the naïve assumption that the features are independent, word order is irrelevant and the chance of the sequence "Hi Lottery" and "Lottery Hi" being spam is the same.

As reflected in equation (4), the sequence "Hi Lottery" in considered spam because product between the probability of a message being spam (P_{Spam}), the probability of "Hi" appearing in a spam message ($P_{(Hi|Spam)}$) and the probability of "Lottery" appearing in a spam message ($P_{(Lottery|Spam)}$) is higher than the product between the probabilities of a message not being spam ($P_{Not\ Spam}$), "Hi" appearing in a non-spam message ($P_{(Hi|Not\ Spam)}$) and "Lottery" appearing in a non-spam message ($P_{(Lottery|Not\ Spam)}$).

Table 3 – Dataset for classification of spam emails

Words\Spam	Spam	Not Spam	Total
Hi	2	5	7
Lottery	7	1	8
Total	9	6	15

$$P_{Spam} \times P_{(Hi|Spam)} \times P_{(Lottery|Spam)} > P_{Not\ Spam} \times P_{(Hi|Not\ Spam)} \times P_{(Lottery|Not\ Spam)} \quad (4)$$

$$\leftrightarrow \frac{9}{15} \times \frac{2}{9} \times \frac{7}{9} > \frac{6}{15} \times \frac{1}{6} \times \frac{5}{6} \leftrightarrow 0.10 > 0.05 = Spam$$

Artificial Neural Network

Artificial Neural Networks aim to emulate the way the human brain works and can be described as a directed graph with each node being a neuron and the edges their links. Each neuron receives as input the weighted sum of the previous connected neuron's outputs. That sum is then fed to an activation function that will compute the output of that neuron. In a feed forward network, such as the one described, information flows unidirectionally from layer to layer. Consider the goal of knowing if a number is bigger than another. Figure 3 depicts one possible solution for this problem using a neural network, by feeding the second number to a neuron with a -1 weight, transforming the sum in a subtraction. Afterwards, the activation function will evaluate if that result is bigger than 0 and give its classification accordingly. This original concept presented in the 20th century has continuously evolved into several variations, some of which will be further expanded upon on the subsection 0.

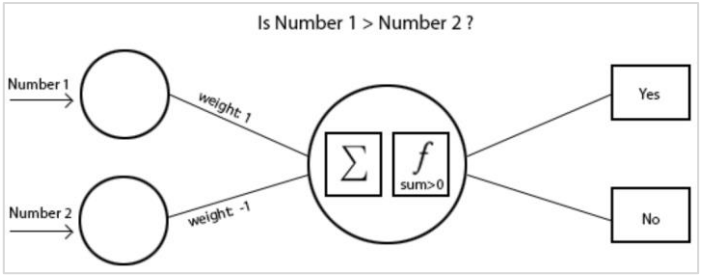


Figure 10 – Artificial Neural Network for classifying if a number is bigger than the other

3.2 Unsupervised Learning

In contrast to supervised learning, consider now that the dataset only contained the coordinates without labelling them, it would be highly unlikely to correctly label these pairs of coordinates. Nonetheless, it would be possible to group the coordinates according to patterns and define borders through clustering techniques, hopefully having each group represent a continent as represented in Figure 3.

This type of learning can be divided into 3 major categories: clustering, dimensionality reduction and association rules [48]. Clustering comes in many forms, as shown in Table 4, each with its advantages and disadvantages. Regardless, K-means, Hierarchical and Density-based Clustering cover most use cases of this category and therefore their algorithms' fluxes are further detailed below.

Table 4 – Notable clustering techniques [48], [55]

Technique	Key points
Fuzzy Clustering	Allows each datapoint to belong to two or more clusters at the same time.
Spectral Clustering	Uses eigenvalues of the similarity matrix to reduce dimensionality before clustering.
Graph Cut	Used in image recognition to separate foreground from background.
Information Bottleneck	Optimizes the trade-off between group accuracy and compression based on information theory.

K-means Clustering

The technique of centroid based clustering was firstly proposed in 1957 by Steinhaus [56] but only a decade later would it be baptized as "K-means" in [57]. This technique proposes an iterative aggregation process for a set number of clusters (K), represented in Figure 11. The process begins by selecting K random points as centroids (in the figure, step 1.1). In each iteration, each point is associated to its closest centroid (in the figure, step 1.2), and the centroids becomes de average of its associated points (in the figure, step 1.2). This process is repeated until the groups remain unaltered between iterations (in the figure, step 3) or a condition such as maximum iterations is reached. There are a few variations of this technique, such as X-means, where the K is set automatically and K-medoid, which always uses a point as a centroid, rather than averages [55].

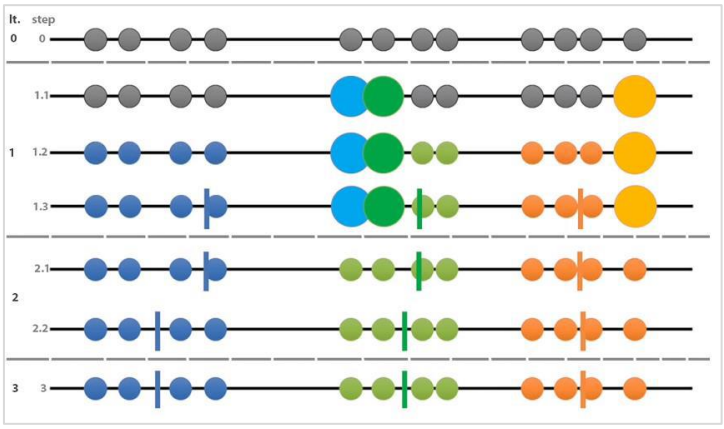


Figure 11 – Visualization of K-means clustering for K=3 [58]

Hierarchical Clustering

Hierarchical Clustering can be divisive (top down), iteratively splitting groups in half, such as DIANA [59] or agglomerative (bottom up), iteratively aggregating the 2 most similar groups, such as Linkage-Based [48]. The latter can be visualized in Figure 12 with a dataset that starts with four groups and iteratively merges the two closest groups until there is only one group. The biggest advantage of this technique is that it is deterministic and is not set to a given number of groups, computing them all at once.

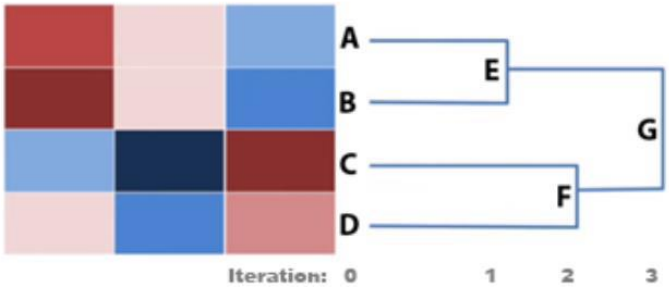


Figure 12 – Visualization of hierarchical clustering [58]

Density-Based Clustering

Density-Based Clustering can recognize non-spherical clusters such as the ones presented in Figure 13. Instead of prompting the amount of groups to form, this technique has the search radius as parameter. Firstly, a random point is selected, and a search is done within the defined radius for points that have no yet been grouped. The closest non-grouped point is then grouped with the previous and a new search is made using the new point as anchor. When all the points in the radius have been grouped, a new random point is randomly selected. Similarly, when a point does not detect any neighbouring point, it is considered an outlier and a new point is randomly selected. This process is repeated until every non-outlier point has been grouped [48], [60].

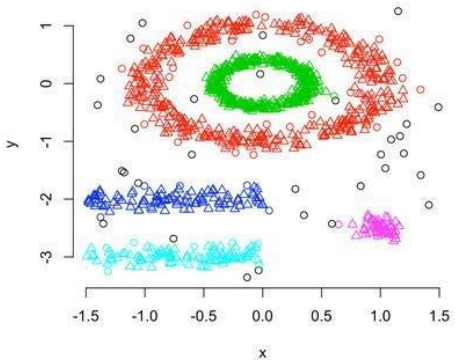


Figure 13 – Example of groups formed by a density-based clustering [61]

Dimensionality Reduction

As the name suggests, Dimensionality Reduction is the process of compressing a higher dimension data into fewer, more relevant dimensions. Reducing the dimensionality of data reduces the computing power needed to process it, makes it more generalizable and more interpretable, easing the process of finding relevant data structures [48]. This can be achieved by reducing the number of features present in the dataset by either feature selection or feature extraction. The former removes irrelevant and redundant features whereas the latter combines features, resulting in less features while maintaining their relevant information [62].

Association Rule Mining

Association Rule Mining is the process of extracting relations and reliance between features of a given dataset. One of the most iconic examples of these unexpected relations, according to Forbes [63], was reflected in the fact that in supermarkets there were often sales with both diapers and beer. While at first glance this might seem like two completely unrelated items, young fathers would, allegedly, make late-night runs to buy diapers and would also grab a pack of beer. Association Rule Mining is not a novel concept, in 2000 it had already emerged as an important problem in knowledge discovery and data mining [64]. Nonetheless, it is still relevant. The Apriori algorithm, originally proposed in 1994 by Agrawal and Srikant [65] is one of the simplest implementations of this technique and is still being explored in retail [66] and even in fields such as healthcare [67].

3.3 Semi-Supervised Learning

As the name suggests, Semi Supervised Learning lies in between Supervised Learning and Unsupervised Learning, shining when the dataset is partially labelled, as represented in Figure 3. Although it might seem like one is sabotaging themselves by intentionally using a partially labelled dataset, real world the data might not only regularly be incomplete, but also have diminishing returns for fully labelling it. As it is often the case, some is better than none, and having a partially labelled dataset tends to increase the clustering capabilities and outperforms fully unsupervised algorithms [68], [69]. Alternatively, approaches such as label propagation can be used to generate the missing labels and employ a supervised technique instead [70].

When dealing with this kind of data, some assumptions must be made. Firstly, it is assumed that the labelled data contains useful information about the remainder of the data [71]. Figure 14 is an example of isolated labelled data that will most likely not improve the accuracy of predictions.

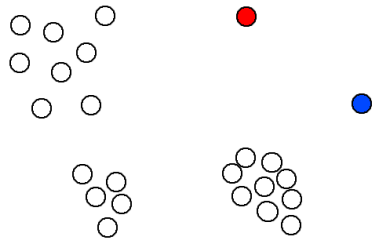


Figure 14 – Example of isolated labelled data

Assumptions

The smoothness assumption states that two points close in the n-dimensional space should belong to the same class. In the same vein, according to the low-density assumption, a boundary should not divide high-density regions, as high-density areas are likely to belong to the same class [71].

In the other hand, a manifold is a substructure of lower dimension than the space it is inserted in, in other words, it is a topological space that is locally Euclidean [71]. Consider a 2-dimensional space, in this scenario a circumference is a manifold. Although it might seem counterintuitive, as circumference is still represented in 2 dimensions, it is possible to unwrap it back into 1-dimensional line. Therefore, the manifold assumption states that (I) for a given space it is possible to contain every datapoint within multiple manifolds and (II) the class of unlabelled data points can be inferred from the labelled data points in its manifold. Similarly, when clustering datapoints, it is assumed that within each cluster, every point belongs to the same class.

In many scenarios, the labelled data does not properly represent the whole dataset and using supervised techniques without the assumptions mentioned above would result in scenarios such as the ones depicted in Figure 15.

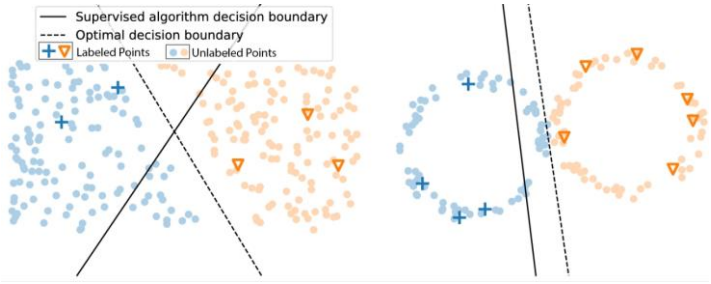


Figure 15 – Optimal versus Actual classification boundary when considering only labelled data (adapted from [71])

3.4 Reinforcement Learning

Reinforcement Learning consists in having an agent observe or interact with the environment and make decisions periodically. These decisions are evaluated and reinforced depending on their impact, which can be immediate or affect the outcome long term. This way, by trial and error, the agent optimizes its policy (its decision model) going forward [72]–[74]. Markov Decision Process (MDP) is often employed in Reinforcement Learning. Paraphrasing Puterman in [75], "the MDP is a class of stochastic sequential decision processes in which the cost and transition functions only depend on the current state and the action being taken". In other words, how a state got to be is irrelevant for the following states. Furthermore, Reinforcement learning can be distinguished between active and passive learning.

Passive learning

Consider the example of learning to play chess. A learner would be considered passive if it merely observed the states throughout the games, without directly affecting the boards [48]. Direct Utility Estimation defines utility as the expected total reward from a given state onwards. It is a simplistic approach that naïvely considers each state to be independent, missing out on learning opportunities and taking longer to converge [22]. In contrast, Adaptive Dynamic Programming (ADP) considers pairs of state-action and as such, it learns the transition model that connects states and solves the corresponding MDP.

Regardless, there is useful information within each transition, that is not being exploited in the previous approaches. Henceforth, the Temporal-Difference Learning approach differentiates itself by not waiting for end of the episode to determine the increment. Instead, each timestep is evaluated according to the expected value the following timestep [76], as represented in Figure 16. The figure maps the outcome at each state from the sequence. It is also considered that the timestep size is 1, so each timestep represents a single state. Although it is described in [22] as a crude approximation to the ADP, it is also recognized that it can be adapted to approximate ADP while saving computing power.

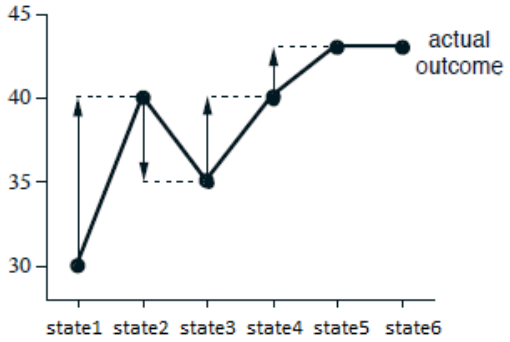


Figure 16 – Suggested changes for n=1 Temporal-Difference Learning (adapted from [76])

Active Learning

In contrast with Passive Learning, where the agent cannot fully explore the environment, Active Learning allows for decisions to be made as the agent is training. Nevertheless, the agent is tempted to exploit the current best policy, that maximizes immediate rewards in detriment of possible better policies, that offer long-term rewards. In another words, there must be a fine balance between exploiting the current information to collect the guaranteed rewards (exploitation) and exploring for more information to improve future decisions (exploration). The conflict between exploration and exploitation is present in ML as a whole but is exacerbated by the nature of Active Reinforcement Learning [73], [76]. There are multiple approaches ranging from exploring each action in each state an unbound number of times [22] to giving bonus rewards for uncharted states [73] to using heuristics that manage this dichotomy [76]. Regardless, at a high level, it is crucial that the agent firstly explores as much of the environment as it can and then polishes its knowledge to achieve better results [77].

As it is often the case with ML, the Passive Learning Techniques previously discussed can be adapted to work in an Active Learning setting. Most notably, Q-Learning is an off-policy Time-Difference control algorithm and one of the most important breakthroughs in Reinforcement Learning due to not having to model the environment. Instead, this approach defines Q values for each action-value pair while still tracking which state-action pairs have been visited and updated, in order to reward exploration[76].

One of the most iconic results of this type of learning was Google's Deepmind AlphaGo, which was the first machine to defeat a professional Go player. Similarly, AlphaGo has since dethroned by AlphaGo Zero with a final score of 100 to 0 [78].

In general, Reinforcement Learning is best suited for scenarios with a finite number of possible states and decision options at a given time. It also thrives on environments where there are both short and long-term rewards [79].

3.5 Deep Learning

The example provided in Figure 10 is a simple feed forward Neural Network (NN), consisting only of an input layer and an output layer. Nevertheless, this concept can be expanded, as seen in Figure 17, by adding (hidden) layers in between and establishing a Deep Neural Network that is able to better learn complex patterns found in real world problems such as image and speech recognition [79], [80].

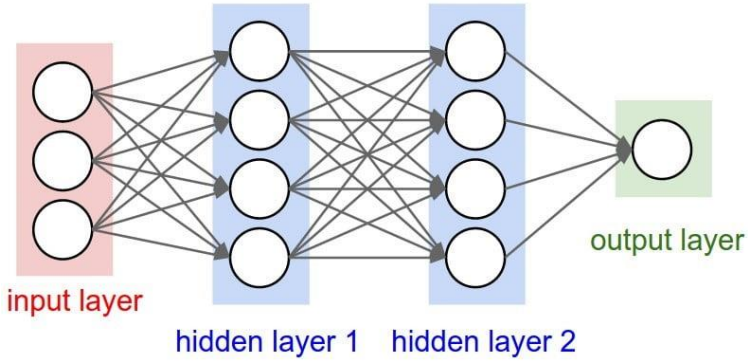


Figure 17 - Deep Neural Network [81]

The rapid ascension of Deep Learning is attributed to the increase of data availability as well as computing power over the last decades. In one hand, the sheer amount of data produced by smart devices is the perfect use case for Deep Learning to shine, as these models tend to get better the more data they are fed. On the other hand, the exponential growth of computational capabilities, namely the usage of Graphic Processing Units allowed for this type of learning to be further parallelized and vastly reduce the training time [82].

Convolutional Neural Network

To understand Convolutional Neural Networks (CNN), comprehending the meaning of a convolution is essential. A convolution is a mathematical operation on two functions which generates a third function that represents the overlap between them, often used in tasks such as reduction of noise in a signal. As represented in Figure 18, in ML, the first function is referred to as input, the second as kernel and the third is the output [83].

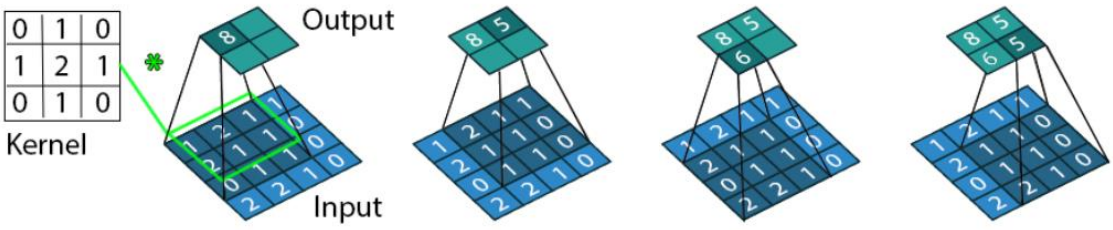


Figure 18 – Visualization of a convolution (adapted from [84])

A CNN is composed of one input layer, followed by one or more convolutional layers and finally a classification layer that compiles the outputs. More specifically, the convolutional layer is an umbrella term for the layers that compose a convolution, as shown in Figure 19. The first layer, performs multiple convolutions, segregating the input in several patches; the second layer, the detector layer, runs each item in the patch through a nonlinear activation function; the third

layer feeds the previous output to a pooling function that compiles that patch into a single value. The most common pooling function is max pooling, i.e. computing the maximum value of each patch, but there are other functions such as the average of the patch, which can be calculated through a kernel like in Figure 18 [83]–[85]. By design, each convolution can be seen as new feature that is automatically extracted, and the deeper the network, the more refined those features become making this type of networks the primary choice for grid-like data such as image processing [86].

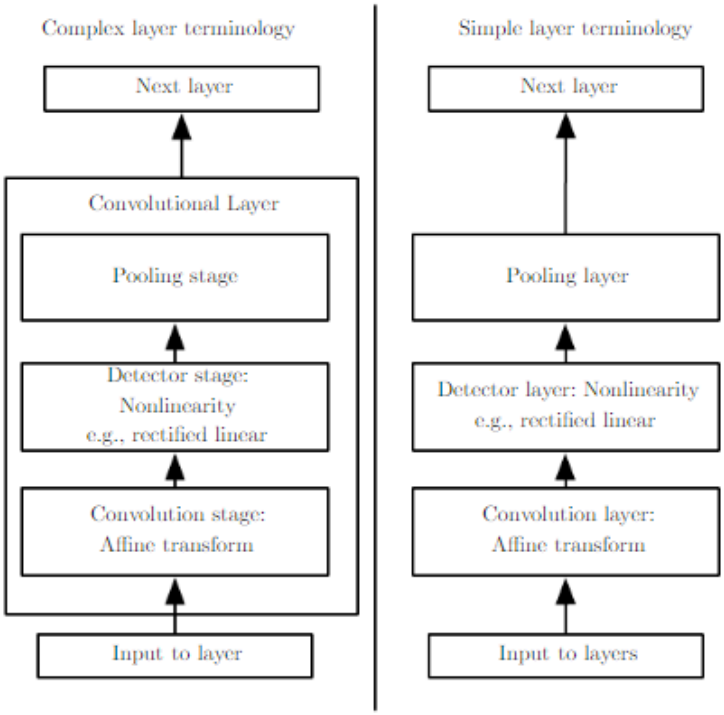


Figure 19 – Layer structure of a CNN [83]

Recurrent Neural Network

When employing a traditional feed forward neural network for the processing of sequential data such as text, the model would have to be trained with a fixed length, usually by padding shorter sequences to match the largest one. Nevertheless, the model would still struggle if each word was shifted by one position or appeared in a different order. As an example, consider the sequences "*I went to Nepal in 2009*" and "*In 2009, I went to Nepal*" [83]. In a traditional feed forward network, there are two evident approaches to tackle a sequential input:

1. Each feature is a representation of a word at a given position in the input [87];

2. Each feature is the frequency of a given word in the input (bag of words) [88];

The first approach would need to learn all the rules of the language separately at each position in the sentence just to handle having the year in different ends of the sequence [83]. In the other hand, the second would have no concept of order and would not be able to tell those examples apart from the sequence "In Nepal, I went to 2009", mistaking a time travel revelation for a fun holiday. However, there are approaches, such as the usage of n-grams (n-word combinations) [89], that can better tackle this issue, but this still showcases the limitations of the feed forward architecture.

In contrast, Recurrent Neural Networks (RNN) allow for loops to exist within its architecture. As represented in Figure 20, the loop in a RNN (a) can be unfolded into a deep neural network where each layer **n** is fed the **n**th item in the input as well as the output from the previous layer (b). In practice, the same weights and biases are applied to every item in the sequence. Henceforth, this type of networks does not need to learn weights for each position within a sequence allowing the model to process different input lengths and order [83].

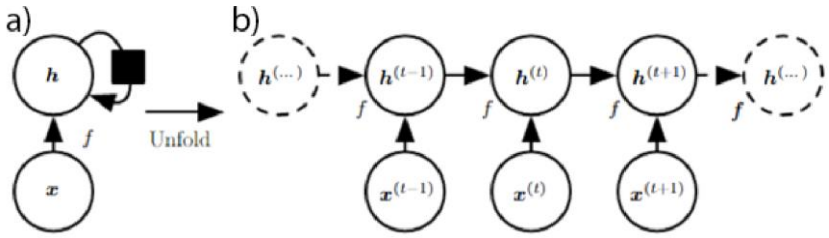


Figure 20 – Unfolding of a RNN (adapted from [83])

Therefore, the differentiating factor of RNNs is that information about previous states is considered by using the weighted output of the previous layer in the calculation. Nevertheless, due to that weight being the same for every input, traditional RNNs can suffer from short term memory, meaning that for a large enough sequence, when processing its later items, most information about the initial items has already vanished (vanishing gradient [90]), as represented in Figure 21. The opposite, gradient explosion, can occur when the weight of the previous output is bigger than 1, causing the previous information to increasingly overwhelm later information [91]. Long Short-Term Memory and Gated Recurrent Units are evolutions of the traditional RNN that can better preserve the most relevant information [92].

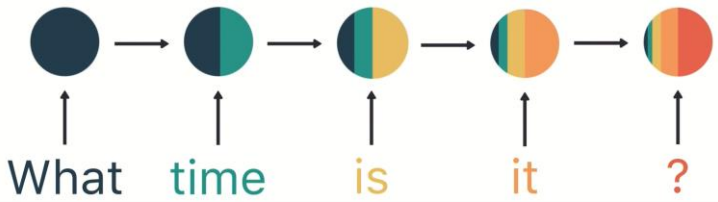


Figure 21 – RNN short term memory visualization from (adapted from [93])

Long Short-Term Memory

As previously stated, the root cause of vanishing and exploding gradient is the shared weight on the previous output. Long Short-Term Memory (LSTM) tackles this issue by having both a short-term memory, that is the previous output and a long-term memory (cell state) that varies depending on the input and the short-term memory. To compute the new short and long-term memory values, both the short-term memory and the input value are considered in 3 different stages, called gates [92], as represented in Figure 22, namely:

- **Forget gate (FG)** – computes the percentage of the long-term memory (LT0) that is to be maintained (LT1);
- **Input gate (IG)** – computes a new potential long-term memory (LT2), that is added to LT1 to obtain the new long-term memory (LT3);
- **Output gate (OG)** – computes the new short-term memory, that is to be used in the next step.

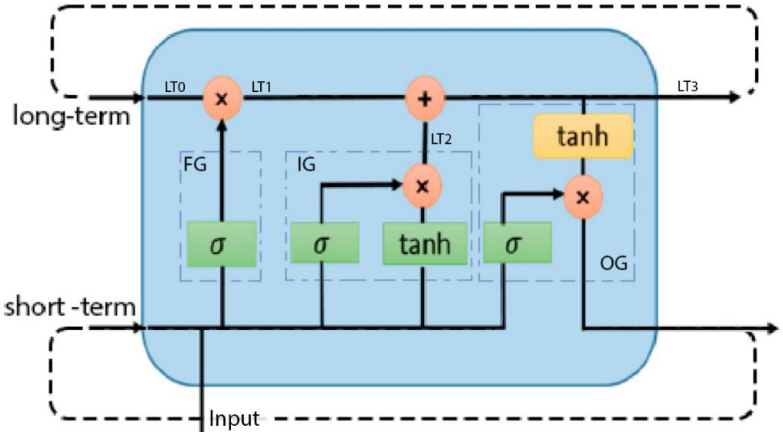


Figure 22 – Visualization of a LSTM cell (adapted form [92])

In this mechanism, the most relevant information is stored within the long-term memory and every weight in the cell is embedded within the gates' functions. This means that the long-term memory is not directly affected by fixed weights, but rather by weighted computations using the input value and the previous output.

Gated Recurrent Units

Gated Recurrent Units (GRU), as represented in Figure 23, are a variation of LSTMs that merge the forget and input gates into a single update gate and also have a reset gate that adds an extra non-linear relation between states [83], [92], [94]. More specifically, the reset layer controls how much of the old state is to be retained whereas the update gate (UG) computes a separate percentage of the old state that will be ignored. The value computed in the UG is subtracted to 1 and multiplied by the old hidden state (HS1). Similarly, a new potential hidden state (HS2) is computed using the input value and the outputs from RG and UG. The output is the sum of HS1 and HS2. Therefore, this design does not contain a separate cell state and has one less gate, decreasing the computational power needed to train this type of models. Nevertheless, while GRU is praised by being faster than LSTM, the latter still tends to perform better [92].

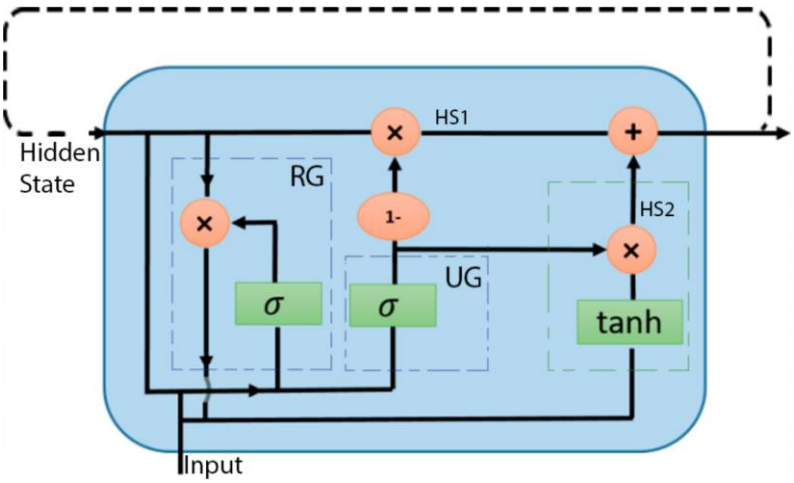


Figure 23 – Visualization of a GRU cell (adapted from [92])

Transformers

While LSTMs were a huge improvement over traditional RNNs for machine language processing, they are not designed to be parallelizable and are therefore computationally expensive to train. Therefore, [95] proposes a encoder-decoder neural network architecture that is solely based on attention mechanisms, dispensing recurrence and convolutions – the Transformer. There are 3 main components that allow parallelization to occur while still maintaining state-of-the-art performance, namely:

- **Positional Encoding** – position encodings for each token in the sequence are added to the embeddings, which means the model can infer the order of a token directly from the data rather than network structure or previous outputs;
- **Attention** – learned behaviour that tells the model which tokens of the input sequence to focus at when predicting the output sequence (Figure 24);
- **Self-Attention** – context of a token within sequence, as an example, the word **tank** has vastly different meanings in the sentences "A *tank* appeared." and "The stock price will *tank*."

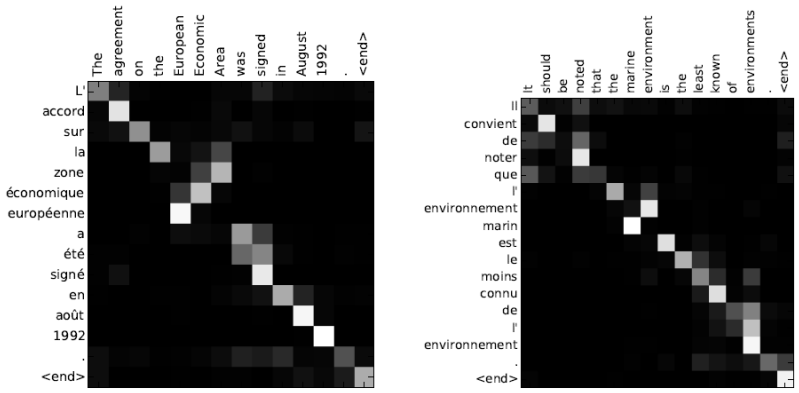


Figure 24 – Visualization of attention in English to French machine translation [96]

Generative Adversarial Networks

Generative Adversarial Networks are unique in their approach. As the name suggests, this concept consists of two separate neural networks that train against each (Figure 25) other in three stages. Firstly, there is a discriminator network, that is firstly trained to detect if an input is real or not [79], [92]. As an example [97], it can be fed multiple images of flowers to learn what a flower looks like and then multiple images of random object that are not flowers to ensure it knows what a flower does not look like.

On the second stage, the generator model will create noise and the discriminator will evaluate its quality until the generated input is close to the actual samples. On the third stage, a zero sum game begins, the generator continues to feed fake inputs to the generator and the latter guesses weather it is fake. The winner gets to maintain its current model, whereas the loser evolves further.

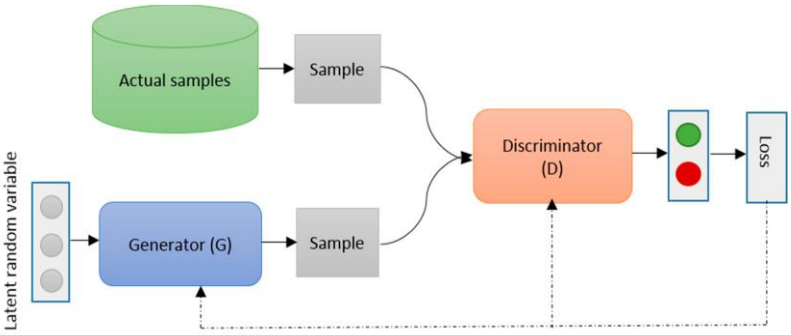


Figure 25 – Visualization of Generative Adversarial Networks [92]

Auto-Encoder

Auto-Encoder (AE) is an unsupervised neural network that learns to copy its input to its output. This process can be divided into two stages: firstly, the encoding, creates a representation of the data useful for dimensionality reduction, compression, fusion, etc. whereas the second stage, decoding tries to reconstruct the original data from that encoding, as represented in Figure 24 [79], [83], [92].

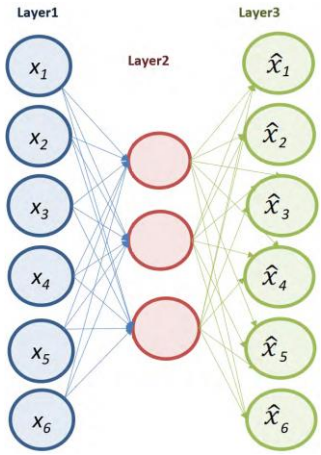


Figure 26 – Auto-Encoder encoding and decoding

Graph Neural Network

Machine learning is increasingly being applied to data from non-Euclidean spaces, that are usually represented through graphs with complex relationships and interdependency between nodes. Graph (or Geometric) Neural Networks (GNNs) are a family of deep learning techniques specialized in processing graph data. Although often overshadowed by its deep learning counterparts [98], GNNs are steadily rising in popularity.

Since words have complex relations between them that could be expressed through a graph, natural language suits the GNN paradigm, in fact, there have been multiple case studies with promising results [99]. One domain that might not seem as obvious is image processing and generation, but GNN have the advantage of being able to map complex concepts such as “Man behind tree.” Into a graph yielding surprising results [100]–[102]. Furthermore, GNNs have been successfully applied to network analysis, such as a citation network, particle physics and chemistry, molecule design, medical imaging and recommendation systems [103], [104].

Naturally, different use cases rely on different variations of the GNN with vastly different implementations, nonetheless, [105] proposes 4 categories for GNN classification, recurrent GNNs, Convolutional GNNs, Graph Autoencoders and Spatia-Temporal GNNs as described in Table 5.

Table 5 – GNN categories proposed in [105]

Category	Key points
Recurrent GNNs	Pioneer of GNNs. Assumes each node in a graph constantly exchanges messages with its neighbours until an equilibrium is reached.
Convolutional GNNs	Similar to CNNs but applied to graph data, grouping the information of a node and its neighbours.
Graph Autoencoders	Unsupervised learning frameworks that encode nodes/graphs into a latent vector space as well as reconstruct data from the encoded data. Can learn network embeddings and graph generative distributions.
Spatial-Temporal GNNs	Goal of learning hidden patterns from spatial-temporal graphs, such as traffic speed forecasting and human action recognition.

3.6 Ensemble Learning

Ensemble Learning can be described as leveraging multiple weaker ML algorithms and combining them to obtain a stronger prediction. The purest form of ensemble is the Bayes optimal classifier, which is a collection of all the hypotheses in the space, in other words, a perfect classifier. However, this approach is unfeasible for any the simplest problem [106], hence the existence of the Naïve Bayes Classifier, which vastly reduces the hypothesis space by assuming conditional independence. Regardless, when ensemble learning is mentioned, it is likely referring the techniques of Bagging, Boosting or Stacking [54], [107].

Bagging

Bagging consists in having several weak learners train independently on different subsets of the same data and combining their outputs, as exemplified by Figure 27. The goal is generalizing the model, making it more robust to randomness. One form of bagging is the Random Forest technique described in the Supervised Learning section.

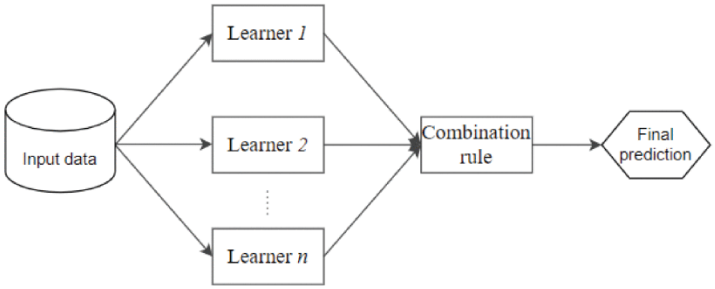


Figure 27 – Parallel ensemble learning [54]

Boosting

In Boosting, on the other hand, the estimators train sequentially, as represented in Figure 28. In this scenario, each estimator tries to offset the prediction errors of the previous estimators, as mentioned in the Gradient Boosting Machine technique, on the Supervised Learning.

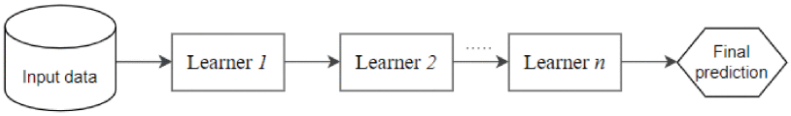


Figure 28 – Sequential ensemble learning [54]

Stacking

Stacking is very similar to Bagging in the sense that it uses multiple parallel learners. However, in Stacking, the models do not necessarily need to be trained on the same dataset nor provide the same kind of information. Instead, this technique is the most useful when the ML models excel in different tasks.

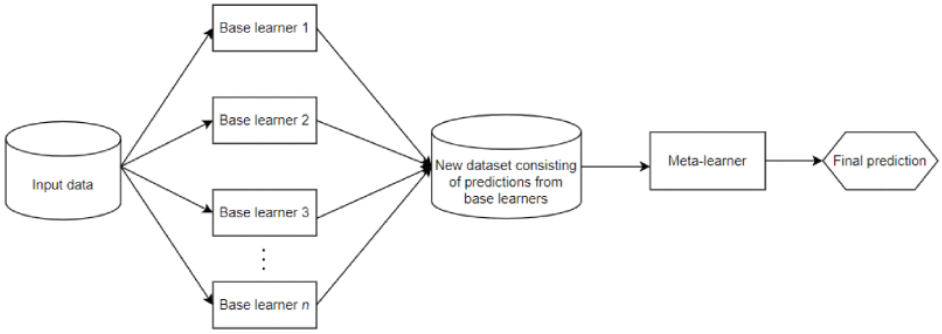


Figure 29 – Stacking ensemble learning [54]

3.7 Concept drift

Often in machine learning, the data used to train a model might not be an accurate representation of the testing or production data, in a phenom known as concept drift [108]. There are multiple anomalies that can cause this, from datasets with different class distributions to the data itself evolving. Techniques such as bagging or cross validation can mitigate the former, at least between the training and testing data, however, when the issue stems from the data evolving, the concepts and interactions the model learned while training are no longer valid, turning the model obsolete.

3.8 Chapter Conclusions

Machine Learning is an ever evolving and complex field in which the right technique can drastically change the system's prediction capabilities. The type of ML technique to be applied is often dictated by the data at hand. Supervised learning is great when the data is fully labelled, as it is in the use case of predicting the outcome of a CS:GO match. On the other hand, unsupervised learning could prove fruitful by grouping bookmakers according to their usual odds, leading to a smaller dataset. Unsupervised learning, in contrast, could serve the sentiment analysis module, as the dataset is unlabelled by nature and labelling it fully is unrealistic. Reinforcement learning, however, is the most useful when the model need to interact with the environment and react accordingly, which does not fit the use case.

Additionally, machine learning models can be combined to improve performance, the simplest way of doing it is by merging all the information into a single dataset and feeding it into a single strong learner, such as deep learning models and hoping the model is able to learn. However, it is possible to run into issues such as limited computing resources making the former approach unviable. Ensemble learning tackles this issue by instead having fewer weaker learners train on subsets of the data and then combining their outputs into a final prediction. The main advantage of this technique is that through stacking, each model is able to train independently and specialise on different components of the prediction. Nevertheless, ML models often fail to perform as expected in a production environment, the data distribution and assumptions learned during training are no longer an accurate representation of the world (Concept Drift).

4 Related Work

This chapter will go over the work that has been done whose approach is akin to any of the three machine learning models proposed in this dissertation. In other words, work that uses the same kind of input data to obtain the same kind of desired output, such as using multiple bookmakers' odds to uncover the implied odd of a match. Nevertheless, the work presented in this section does not necessarily rely on machine learning. The goal is to compile and discuss previous work that either presented unique takes or compared multiple solutions, all of which inspired the solutions proposed in this dissertation.

4.1 Outcome Prediction

Predicting the outcome of a match using past results is not a novel idea, in fact, every bettor does it, even if subconsciously. This is also reflected in the amount of literature found on this approach.

In 2017, Open Science Framework, hosted a competition where the goal was predicting the outcome of football games [109]. The dataset was composed of roughly 200 000 football games from 52 leagues around the world. In this competition, one group [110], while still unexperienced, had an interesting approach – instead of relying on a single model for all the predictions, they realised that historic results of a given matchup had more impact in the outcome of a match than the team's current form. As such, in a pre-processing phase, the teams are ranked according to their performance. Furthermore, the performance of a team in a season only accounts for 80% of their final ranking, with the remaining 20% coming from the previous season's performance, as to account for prior form. When evaluating a matchup, first, whether the teams have met over 10 times is checked. If this proves positive, a mathematical formula is followed to predict the match based on previous outcomes. Else, the rankings are fed to a statistical Bayesian-based model that computes the probability of a team winning.

While the approach did not yield any more promising results than a coin toss, that is likely because the authors created a model based mostly on national leagues where clubs play weekly and tested it a world cup where the national teams play every two years, meaning there was a big disparity between the datasets of training and testing. Regardless, this paper serves as a baseline introduction to outcome prediction.

In contrast, [111] presents an extensive analysis on tennis match prediction, by testing the techniques LR, NN, RFC, Gradient Boosting Machine (GBM), and SVM, all of which were able to reach an accuracy hovering 70%. Furthermore, the paper tested a baseline predictor from the bookmakers' implied odds, which also hovered 70% accuracy, meaning that even bookmakers can't overcome the randomness of tennis games outcomes.

Also, it was tested multiple betting strategies, most of them yielding negative returns over a decade, regardless of the model used. From the models mentioned previously, RFC had slightly better training accuracy, but SVM, NN and LR consistently performed better in the betting simulations. On the other hand, the paper proposed ensembles of the models that proved better on every betting strategy. Finally, it was also concluded that bookmaker odds were one of the most important features of the prediction models.

Despite the previous paper's findings, research on using bookmakers' odds to predict the outcome of a match is lacking. It is highly likely that bookmakers use machine learning models to compute such odds and it is no easy feat to compete directly against a multi-billion-dollar industry while having less resources. This mindset is what led to the solution proposed in [112], which consists of gathering odds from multiple bookmakers and then using math to find and bet on odds that were considered fair, in comparison to the remainder of bookmakers, this approach is alike the positive expected value bets provided by services like OddsJam [14], mentioned in section Beating the Odds.

In contrast, to this naïve statistical approach, [113] tested multiple models that used the variance in a bookmaker's odds over time to make a prediction of the outcome. The most promising techniques were Bagging and BayesNet boasting over 70% accuracy and NaïveBayes, hovering 65%.

From the research, it was found multiple works that analyse multiple models for predicting football matches [114]–[118] that are further detailed in Table 6. Within these works, RFC was consistently one of the best performing techniques, with other techniques such as SVM and NN presenting various degrees of success. Overall, it was concluded that feature selection is a very important step when building a prediction model and that team and player statistics tend to improve the predictions [114], [116], [118]. Similarly, combining different models tends to yield better results [116], [117]. A few works proposed using players to represent the teams [118], [119] and there are also experiments with unique techniques, with varying results, namely Polynomial classifier (POL) [115], that outperforming the other tested approaches and LSTM [119], which was barely better than a coinflip.

From the works previously mentioned, the most recent paper is from George Peters and Diogo Pacheco [118] and is a deep analysis ranging from a simple heuristic that always considers the home team the winner, to complex machine learning models with 52 features. The paper emphasizes that there are two main paths to outcome prediction: classification (whether a team wins, loses or ties) and regression (how much goals each team scores). The consensus is that classification models typically perform better, but the authors took the challenge and tested different regression approaches, using the following heuristics as baseline:

- **Home win –** home team always wins 1:0;
- **Tradition –** highest ranked team wins 1:0;
- **Recency –** each team scores the same number of goals as in their last match.

Secondarily, for each of the machine learning approaches, they proposed two independent models: one for home team goals and another for the away team's. Additionally, the analysed machine learning techniques were Linear Regression, K-Nearest Neighbours, Decision Tree Regression, Random Forrest Regression and Support Vector Regression, applied to each of the following feature combinations, resulting in a total of 15 pairs of models:

- **Players –** features are the embedding of players being 1, -1 or 0 depending on whether the player played for the home team, opposing team or neither;
- **Team Stats –** in game stats for each position in the game, for the current and previous seasons, such as 'shots on target';
- **Lineup Stats –** similar to Team Stats, but filtered by the players on the field that game

Finally, for the evaluation of the models, 3 different stages stand out. The first was empirical evaluation and consisted of comparing the Mean Absolute Error, Root Mean Squared Error and R2 scores of the different techniques. In this stage, the machine learning models clearly outperformed the heuristics, with SVR being the most promising overall. The second stage tested the ability of prediction a whole season and compared the accuracy of predicting top 4 and bottom 3 teams in the league. The machine learning models once again outperformed the heuristics with both the Lineup and Team stats models managing to predict 50% of the top 4 and 100% of the bottom 3 teams, whereas the best heuristic – Tradition – had an accuracy of 50% and 33%, respectively. For the final test, the techniques were put against a bookmaker, betting 1£ on the predicted outcome for 100 matches. Surprisingly, the only profitable techniques were Team and Lineup Stats using both KNN and DTR, with the Team Stats KNN profiting 42.5£, over twice as much as the second most profitable. Furthermore, both Home win and Tradition heuristics outperformed any of the other machine learning models.

This work's only fault is not taking full advantage of its prediction capabilities on the last evaluation stage by adapting their bet depending on the predicted result and the bookmaker's odd. As an example, if the model were to predict a 6-1 stomp in favour of a team while the bookmaker had an odd of 2 on that very team winning, it would be a waste to bet a mere 1£. Not all odds are created equal and treating them as such is a flawed metric of evaluation. I contacted one of the authors, Diogo Pacheco, via email about this topic, to which he replied with the following:

> *"I don't think we made any more laboured test for the betting scenario, but your suggestion is very good! The idea was to simply have one more scenario that was a bit more real beyond the traditional classification metrics."*

Freely translated from the original reply, in Brazillian Portuguese:

> *"Acredito que não fizemos nenhum teste mais rebuscado diante dos cenários de aposta não, mas sua sugestão é muito boa! A ideia foi ter apenas mais um cenário um pouco mais real além das tradicionais métricas de classificação."*

In summary this work provides great insight into match outcome prediction while also highlighting the importance of having simple baseline predictors to compare the more complex models to.

Table 6 – Research on sport match outcome prediction

Description	Sport	Approaches	Best results	Conclusions	Year	Ref.
Tested three different classification techniques for the prediction. Original dataset had 70 features, and feature selection was made depending on whether the team was playing away.	Football	SVM, BREP, RFC	RFC	Reduced dataset through feature selection proved to have better predictions.	2016	[114]
Proposed the usage of Polynomial classifier (POL) and compares it to other techniques.	Football	NB, DTC, RBF, POL, MLP, SVM	POL, MLP, DT, SVM	The proposed solution was slightly more accurate.	2017	[115]
Tested multiple classification techniques for the prediction of the outcome. Once detected the best model, goals number were predicted depending on the guessed outcome.	Football	NB, BN, LR, KNN, RFC, NN	LR, RFC, NN	Using historic results between teams improved every model's accuracy. LR was the best predictor.	2018	[116]
Extracted player stats, rating them from 0-100 in different areas as well as basic match history data. The data about the last 5 matches was fed to the LSTM	Football	LSTM	LSTM	The accuracy was 52.5%, but with the model having player stats, it is more resilient to roster changes.	2018	[119]
Analysed 6000 matches from the top 1000 players within a non-professional league on a play-by-play basis, traced their playstyle and grouped them based on it, and fed the playstyles of each player in a game to a NN to predict it.	CS:GO	K-means, NN	NN	Clustering players depending on their playstyle proved useful for generalized prediction.	2018	[32]
Trained different classification techniques individually and created an ensemble model for the prediction of football matches.	Football	SVM, RFC, NBC	Ensemble Model	Using multiple predictions model to create an ensemble can lead to better results.	2019	[117]
Tests the multiple ensemble models including bookmaker odds to predict the outcome of tennis matches as well as analysing different betting strategies.	Tennis	LR, NN, RFC, GBM, SVM	LR,NN,SVM	Regardless of the model and approach, the maximum accuracy rounds 70%, even for bookmakers.	2021	[111]
Predicted the number of goals of each team to obtain a winner. Player embeddings to represent teams, as well as team stats	Football	KNN, DTR, SVR, LR, RFR	KNN, DTR	Team stats worked better. Heuristics performed better than LR, SVR and RFR.	2022	[118]
Academic thesis. Tested multiple techniques for the prediction of professional CS:GO matches. Tested each model using player representations, team and both.	CS:GO	LR, KNN, RFC, NN, CNN,	RFC	The classic Elo rating model tended to perform the best followed by player averages.	2022	[36]

4.1.1 Applied to eSports

Esports is not yet a popular topic amongst scholars and although there is research within eSports outcome prediction, it is often regarding the current game state [120] rather than a prediction before the game starts.

In [32] the usage of a NN to predict the outcome of a match depending on their playstyle is proposed. For this purpose 6000 games from the top 1000 players were analysed on a round by round basis and the players were clustered depending on their playstyle, using K-means. It should be mentioned that these games are not professional games and the teams are assigned through a match making system. Thus, this method does not directly translate into a professional, structured setting. Regardless, grouping the players and capturing the interactions between playstyles rather than directly between players yielded promising results.

On the other hand, [36] is an academic work that tested compared the techniques LR, RFC, KNN linear NN and CNN. For each technique, it was also considered three approaches: team averages, player averages and both.

It was also created 2 baselines: the first always considered as the winner the 'home' team, that is, the team that appears on the left side of the matchup; the second is a chess-like ELO system where players would increase and decrease their ranking (ELO) depending on the outcome [121].

Overall, every single model was outperformed by the ELO baseline, with the linear NN player average model being the most promising and managing to perform close to the ELO system, albeit slightly worse, as reflected in Figure 30 – Comparison between linear NN and the baselines Figure 30. Regardless, from this work it was clear that the player average models were superior to the team averages and that the combined model was usually similar to the player averages.

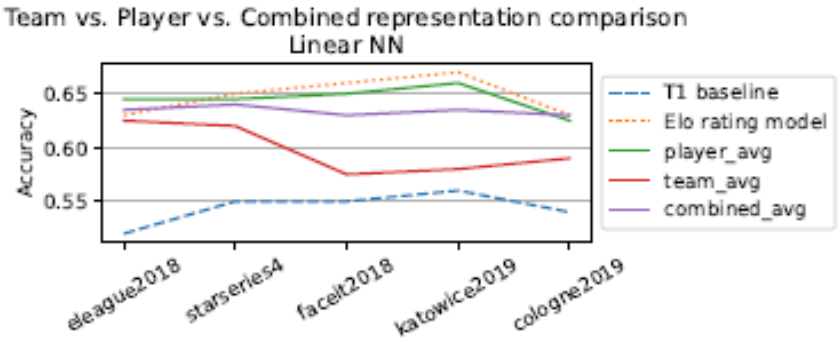


Figure 30 – Comparison between linear NN and the baselines [36]

4.2 Sentiment Analysis

Human communication is a cooperative exercise that can range from gestures to speech or writing. Effective communication relies on context and motives. Context allows the recipient to better understand what is being communicated – pointing at the sky in a religious context has a vastly different meaning than pointing at the sky while birdwatching. Similarly, motives reflect why it is being communicated – in the first example, pointing at the sky expresses a belief in a higher being whereas in the latter scenario the intent is for the recipient to find something they might be looking for [122].

Having computers analyse and extract meaning from text is an ongoing research field – Sentiment Analysis – more specifically, it is the computational study of opinions, sentiments, and subjectivity of text [123]. As previously mentioned, to extract the correct meaning from a communication mean understanding the context and motive is crucial. For machine learning, motive is not particularly useful, as it tends to be inherent to the task at hand. Consider a dataset about sentiment of product reviews, the underlying motive of each review is that the customer wanted to express their opinion. Context, on the other hand, can vary within a dataset, and providing information about it often proves useful [124], [125].

Nevertheless, human communication is complex, humans are often not clear with their words. Detecting irony and sarcasm can be intelligible even for an adult human, who has a deep understanding of the language and cultural context, it shouldn't come as a surprise that is also challenging for a machine [126]. In the age of the internet and digital communication, emoticons like ":)" and, more recently, emojis, are a more effective mean of transmitting sentiment than words. Consequentially as it becomes more prevalent in real world data, machine language models that ignore them are missing out on precious information [127].

Large Language Models

Currently, the most effective publicly available machine learning technique for language processing is the Transformer architecture. Within it, there are multiple prolific pre-trained models, that can be fine-tuned for a specific task, such as sentiment analysis. GPT-3 [128] is a Large Language Model (LLM) designed to generate text form a prompt. The model is available through a paid Application Programming Interface (API) and can be fine-tuned on the fly by providing some examples within the prompt. It is the successor of the GPT-2 model proposed in [129] and is considered to write better sentences than some humans [130]. OpenAI also launched ChatGPT in late 2022 [17] – a version of the GPT-3 model, fine-tuned for chatting, that has captured the interest of internauts and mainstream alike, and is on the path to become more popular than artificial intelligence itself, according to Google Trends, as reflected in Figure 31 [131].

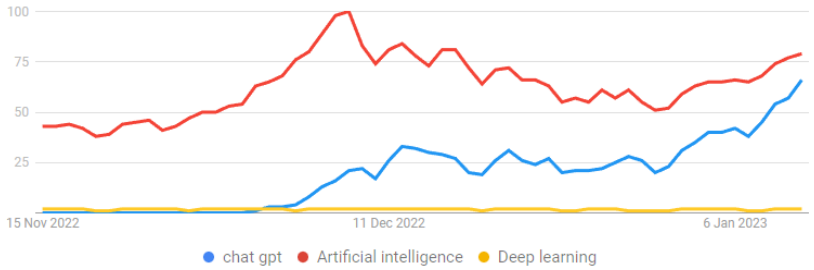


Figure 31 – Popularity of ChatGPT versus Artificial Intelligence and Deep Learning [131]

Nevertheless, one of the earliest adopters of the transformer architecture was the Bidirectional Encoder Representations from Transformers (BERT) that was presented by Google engineers in [132]. BERT is a Pre-trained Language Model (PLM), which means it has previous knowledge about language and can be fine-tuned for specific tasks without needing to be trained to understand a sentence from scratch [133]. However, spiritual successor of BERT and was developed by researchers at Meta (previously known as Facebook): Robustly Optimized BERT Approach (ROBERTa). This model was trained using more data than BERT, using bigger batch sizes and longer sequences. The architecture was also modified by removing next sentence prediction and introduced dynamic masking, achieving state-of-the-art results on GLUE language model benchmark [134], [135].

As LLMs became more powerful, the GLUE benchmark became obsolete and replaced by SuperGLUE [136]. Both the GLUE and SuperGLUE leaderboards can be consulted on the benchmarks' website [137]. Unsurprisingly, in this ever-evolving field, many other models, mostly transformer based, have emerged and claimed high ranks in the leaderboards, as presented by Table 7.

Table 7 – Super GLUE top scorers, as of January 2023

Rank	Model	SuperGLUE Score	Public	Ref.
1	Vega v2	91.3	No	[138]
2	ST-MoE-32B	91.2	No	[139]
3	Turing NLR v5	90.9	No	[140]
4	ERNIE 3.0 (Chinese)	90.6	Yes [141]	[142]
5	PaLM-540B	90.4	No	[143]
6	T5 + UDG, Single Model (Google Brain)	90.4	Only T5	[144]
7	DeBERTa / TuringNLRv4	90.3	Yes [145]	[146]
8	Human Baselines	89.8	N/A	[136]
9	T5	89.3	Yes [147]	[148]
	…	…	…	…
16	RoBERTa	84.6	Yes [149]	[134]
	…	…	…	…
24	GPT-3 few-shot	71.8	(paid) [128]	-
25	BERT	69.0	Yes [150]	[132]

Since this section was first drafted in late 2022, the SuperGLUE benchmark relevance has faded and the newer models have not even been listed. Furthermore, meanwhile, OpenAI launched their new model, GPT-4 in March 2023 and has since further improved it along with GPT-3 by making them cheaper, faster and capable of interpreting more context [151], [152]. This trend was also observed with other tech giants, such as Google and Meta. Google moved their focus to PaLM – a Language Model that uses their new architecture – Pathways [143] and launched a Chat-GPT competitor – BARD – in February 2023 which has since been upgraded with the release of PALM-2 in May 2023 [153], [154]. In a similar vein, Meta launched LLaMA in February 2023 and released an improved version, LLaMA2, in July of the same year [155], [156].

Nevertheless, most of these models are not open source, OpenAI's models are available through a paid API, Google's are only available through BARD and Meta's first model was also non-commercial. Although, LLaMA had its weights leaked, from which Stanford researchers

created Alpaca [157], [158]. LLaMA2 however, is publicly available and has a limited free commercial licence.

One leaderboard that rose in prominence is Chatbot Arena by Language Model System's Organization, hosted on Hugging Face [159]. This leaderboard considers 3 metrics:

- **Chatbot Arena –** a crowdsourced, randomized battle platform. Users ask questions and receive answers from 2 different anonymous models and rate which (if any) was the best. Currently over 50K user votes are used to compute Elo ratings;
- **MT-Bench –** a set of challenging multi-turn questions, using GPT-4 to grade the model responses;
- **MMLU (5shot) –** a test to measure a model's multitask accuracy on 57 tasks.

GPT-4 holds the first place in every metric, with the following 4 positions being split between GPT-3.5, Claude-1, Claude-instant-1, and Claude-2 [160], with the latter performing the best overall. Both are pay-to-use, with Claude models being significantly cheaper, but limited to the United States and United Kingdom. Overall, the best free-to-use models are LLAMA and its variants, with the caveat that the best variants require very expensive hardware to run. Therefore, the top 20 best rated models are either pay-to-use or expensive to run, with the only viable option being GPT-3.5-turbo. However, there are less powerful variants of the top models, which are less computationally expensive. Out of these viable options, the best performer of each metric is represented in Table 8. It is worth noting that these smaller models, while being able to run on some consumer-grade hardware, are still expensive to fine-tune.

Table 8 – Chatbot Arena rankings of viable models

Model	Arena Elo Rating	MT-Bench	MMLU	License	Ref.
Vicuna-7B	1006	6.17	53.6	Non-commercial	[161]
Llama-2-7B-chat	961	6.27	45.8	Commercial (limited)	[156]
FastChat-T5-3B	892	3.04	47.7	Non-commercial	[162]
GPT-3.5-turbo	1122	7.94	70	Proprietary	[152]

4.2.1 Pre-match sentiment analysis

Sentiment analysis is often the end goal of machine learning models. The proposed solution aims to rather use that analysis as a steppingstone of the final prediction. Sentiment analysis not being used as a middle step is reflected in the lack of literature on this topic. Searching for 'bet sentiment' on 3 different databases – b-on [21], IEEEXplorer [18] and Web of Science [163] – yielded few relevant results.

Nevertheless, this research proved to be reassuring as there is evidence that fan sentiment affects their betting behaviour in the National Basketball Association [164]. In a similar way, [165] backs up the claim that in football, bookmakers overreact when teams don't perform as expected. Nevertheless, it is possible the bookmakers do this purposefully to maximize their profits by baiting the casual pundit, mainly because it is not uncommon for bookmakers to limit or close accounts that take advantage of their shortcomings [112], [166].

4.3 Chapter conclusions

While eSports is a blooming industry with a lot of potential for machine learning exploration it is still in its infancy and is seldomly a topic explored by researchers [33]. Nevertheless, since CS:GO presents a lot of parallels with traditional sports, the research made on traditional sports predictions can be transferable and the findings, such as the type of techniques used also apply.

Historic data, both match history and previous odds, is a type of data that is naturally labelled, because the winner is known. For the prediction of matches using match history, SVM, RFC, LR as well as NN are the most promising techniques. On the other hand, there is few examples of using bookmaker odds to cast a prediction, but according to [111], every prediction model tested improved when adding the associated odd. Therefore, it might make sense to merge the odds with the match history dataset and have a single model for both.

On the field of sentiment analysis, transformer-based architectures are very prominent and as alluded above, pre-trained language models, LLMs, specifically, greatly facilitate the implementation of sentiment analysis by fine tuning or even prompt engineering. From the publicly available models, GPT-3.5-turbo seems to be a great baseline model, due to it being able to interpret prompts and thus able to do classification without prior training. In the other hand, the models Vicuna-7B and Llama-2-7B-chat are fine tuned versions of LLaMMA-2 whereas FastChat-T5-3B is based on the T5 model. This proves that decent results can be obtained by fine-tuning pre-trained models. Although, if fine-tuning these models proves to be too computational expensive, there are also alternatives such as DeBERTa, which are lighter but less powerful.

5 Methods and Materials

This chapter describes the methods employed for the different stages of the project, namely, how the datasets were obtained, and the tools used for data scraping, storage and processing of said data and machine learning. Additionally, the base structure of each dataset is explored and an evaluation on ethical aspects and data protection is provided.

5.1 Data scraping

It is undisputed that the internet provides vast amounts of information which can be accessed by anyone. Naturally, extracting relevant information automatically in a process known as data scraping [167] is very appealing. In this field, the term “crawler” is often used to describe an algorithm that can navigate the web, whereas “extractor” refers to the data processing and collection. Nevertheless, in the context of this document, a “crawler” refers to a mechanism that can navigate its environment and extract the information.

Information available on the internet might seem like it is free but that is rarely the case. The websites either run on donations, balance selling adds and providing content [168] or sell their users’ information. The latter, although ethically dubious, became so profitable that the European Union had to regulate it through the General Data Protection Regulation [169]. Thus, website owners are not too keen of having their information freely extracted by automated systems.

There are several design choices of a web platform that can affect the crawler’s performance, either by coincidence or with purpose. During the exploration and extraction of the datasets, some anti crawler tactics [170] were found and are listed below.

IP Restrictions

Websites preventing multiple consecutive requests is very common, to protect themselves against denial-of-service attacks [171] and general spam. This is usually done by blocking certain IP addresses from connecting to the website. However, the blocking is usually a timeout rather than a permanent blacklist. To overcome this, firstly, the crawler should wait between requests that are known to cause timeouts. Secondly, in the case of timeout, the crawler should be able to read the “retry-after” header, and sleep the specified duration, to minimize useless spam.

Access control via User-Agent

This is a very low level anti-crawler tactic that blocks every request that does not provide a valid browser version under the "User-Agent" header. This can be easily overcome by setting a valid value for that header.

Browser rendering

Modern single page applications, such as AngularJS [172], React [173] and Vue [174] work differently from the classic web design by rendering data from the browser instead of the server. This means that when requesting for a given page, the data will be often missing from the HTML and will then be obtained through requests to the server. There are several solutions for this approach, but the easiest and fastest is exploiting this architecture and crawling the API directly. Regardless, there are scenarios where the data returned by the API is encrypted and unintelligible, which requires either a deeper analysis of the page's Javascript [175] or a browser emulator such as PhantomJS [176].

Session and CAPTCHA

Session refers to mechanisms that require any kind of user token to be provided, which are usually obtained through a login. Although, bypassing it by login in beforehand and passing that token to the crawler is a possibility.

CAPTCHAs can range from having the user respond to an input, such as analysing an image, to having the browser perform a computation through Javascript. The first one is a mature approach that is hard to bypass, whereas the second requires either a deeper analysis of the page's Javascript or a browser emulator such as PhantomJS.

Some APIs are also protected by CAPTCHAS, such is the case of OddsPedia's [29]. This verification was bypassed by writing the crawler in Javascript and running it directly through the browser's console while on their website.

5.2 Datasets

This subsection is firstly an analysis of existing datasets as well as data providers. Secondly, the solution selected for each machine learning model is presented and explored the structure of those datasets. Finally, a brief overview of the datasets is made.

5.2.1 Existing Datasets

As mentioned in the Introduction, compared to mainstream sports like football, CS:GO is severely lacking in readily available datasets. Nevertheless, there are alternatives. There are numerous publicly available datasets listed on Kaggle, of which three stand out, namely Mateus Machado's [177], Gabriel Salles' [178] and Artem and Maxim Sinyaev's [179]. All three datasets were scrapped from HLTV and only contain data up to 2020. Since one of the goals of this project is the application of the proposed solution in the real world, the dataset needs to be as recent as possible. Furthermore, HLTV does not provide the historic odds of the matches.

In contrast, there are two main data providers with paid subscriptions, PandaScore [180] and GameScoreKeeper [181]. Since the match history data can be obtained through HLTV, it would

be relevant to know the pricing of the data related to previous odds, as well as knowing if they provided odds from multiple bookmakers. Detailed information about odds and pricing is not available on their websites. Upon contacting them, GameScoreKeeper clarified they do not provide any kind of historic odds and that the historic match information is accessible through a subscription of 800€ per month. PandaScore, on the other hand, did not reply initially, but eventually informed that they usually do not resell this information, but that they could provide a quota depending on the date range. Although, considering the price of their monthly subscription and the fact they did not reply in due time, it was discarded as an option. Regardless, none of these solutions provide any kind of fan interactions from which to infer the sentiment. A comparison between these 5 different alternatives can be found Table 9.

Table 9 – Comparison between different data sources

Source	Machado, M. Kaggle	Salles, G. Kaggle	Artem & Sinyaev Kaggle	Panda Score	GameScore Keeper
Date range	2015 – 2020	2016 – 2020	2019 – 2020	lifelong	lifelong
Game Results	Yes	Yes	Yes	Yes	Yes
Player Data	Yes	Yes	Yes	Yes	Yes
Detailed Player Data	Yes	Yes	Yes	Yes	Yes
Team rankings	Yes	Yes	Yes	Yes	Yes
Maps	Yes	No	Yes	Yes	Yes
Comments	No	No	No	No	No
Past Odds	No	No	No	Yes	No
Price of Stats	Free	Free	Free	150€	800€
Price of Odds	N/A	N/A	N/A	N/A	N/A

5.2.2 Match history

In line with the Kaggle datasets, HLTV scrappers previously created for the BetaCS platform were repurposed to retrieve the information. Considering what was discussed in the Data scraping section, and the fact that HLTV uses static HTML a crawler for navigation and HTML parsing was created.

This dataset contains all the matches where at least one of the teams was top 20 in the world at the time and ranges the past 6 years (2017-2023). For each match, it was extracted and saved in a database the following information:

- **Match Id –** HLTV identification number of the match
- **Date –** date and time the match started
- **BoX –** best-of format used in the match
- **Event –** league, championship or tournament the match belongs to
- **Maps –** In-game level each game was played on
- **Veto –** Maps picked and banned by each team
- **Type –** Whether the game was played online or on LAN
- **Teams' Information:**
    - **HLTV id –** HLTV identification number of the team
    - **Name –** Name of the team
    - **Rank –** Global HLTV rank of the time at the time of the match
    - **Players –** The players that played for the team each game of the match
- **Players' Information:**
    - **HLTV id –** HLTV identification number of the player
    - **Name –** In-game name of the player
    - **Performance metrics –** performance indicators of the player, per game
        - **Kills –** Number of eliminations the player finished
        - **Deaths –** Number of times the player died
        - **Assists –** Number of times the player helped a teammate get an elimination
        - **Average Damage per Round –** Average damage inflicted to enemies by the player each round
        - **Rating –** HLTV's complex rating system [182]

5.2.3 Sentiment Analysis

HLTV also provides a comment section where fans often express their predictions, and feelings coming into and out of a match. As such, it was also extracted and saved in the database those comments with the following structure:

- **commentId** – Order by which the comment was inserted in the match thread
- **user** – Username of the fan that commented
- **date** – Date and time of the comment
- **text –** Text content of the comment
- **team1** – Array of:
    - **name** – Name of team
    - **players** – List of player
        - **alias** – In-game name of the player
        - **Name** – First and last name of the player
        - **Nationality** – Nationality of the player
- **team2** – {same structure for team2}

5.2.4 Obtaining Odds

In contrast with the previous datasets, the data about previous odds is not obtainable through HLTV, despite listing multiple bookmakers on the match pages of future matches, as seen in Figure 32. I reached out for further insight of whether this information was stored and could be accessed but obtained no response.

Betting

	Apeks		HONORIS
GG.BET	1.18	-	4.49
bet365	1.22	-	4.00
LOOT.BET	1.22	-	3.80
THUNDERPICK	1.22	-	3.80

Figure 32 – Listings of odds from multiple bookmakers for a match in HLTV

Furthermore, as mentioned above, of the alternatives presented only PandaScore provided historic data about odds. After extensive research the most promising odd providers were esport.net [183], Odds Shark [28], Odds Portal [184], OddsPedia [29] (OP) and Strafe [185]. As exposed in Table 10, the first two also do not have past odds publicly available, but as big players are likely to store them. Regardless, none responded to my reach for that information. Odds Portal, on the other hand, has previous odds from multiple bookmakers, but does not use standard static HTML nor API requests and would need a browser emulator which exponentially increases the scraping efforts. Finally, OddsPedia and Strafe both use OP's API to obtain past odds, in a structured, easily comprehensive manner.

Table 10 – Comparison of odd providers

Source	Current Odds	Past Odds	Crawler needed
esports.net	Yes	No Response	HTML
Odds Shark	Yes	No Response	HTML
Odds Portal	Yes	Yes	Browser emulator
OddsPedia	Yes	Yes	API
PandaScore	Yes	Yes	API is provided
Strafe	Powerd by OP	Powerd by OP	N/A

This way, a dataset was created by extracting information about every match in the past 6 years, and was saved the same database with the following structure:

- **id –** OP identification number of the match
- **date** – date and time the match started
- **t1_id** – OP identification number for team 1
- **t1_name** – name of team 1
- **t2_id** – OP identification number for team 2
- **t2_name** – name of team 2
- **winner** – OP identification number of winning team
- **odds –** array of**:**
    - **house** – name of the bookmaker
    - **t1_odd** – closing odd for team 1
    - **t2_odd** – closing odd for team 2

5.2.5 Overview

The raw datasets vary widely in size, as exposed in Table 11, both the match history dataset and the comments (sentiment analysis) dataset were extracted by scrapping HLTV's HTML using a BeautifulSoup. The first contains 7913 unique matches, each containing multiple games and over 2 400 000 unique comments. On the other hand, the odds history dataset was extracted via the OddsPedia API and consists of 51 302 matches, each with multiple bookmakers' odds. Although, it should be noted that this dataset contains duplicates, includes games from lower tier teams and is not fully labelled. Thus, will be further refined in 7.1 – Data cleanup.

Table 11 – Overview of the datasets

Dataset	Size	Source	Crawler
Match history	7913 matches	HLTV	HTML
Comments	>2 400 000 comments	HLTV	HTML
Players	2095	HLTV	HTML
Odds history	51302 matches /	OddsPedia	API

5.3 Technologies

This section describes the tools and processes of extracting, storing, and curating data and generating the final datasets from it. Additionally the tools used for pre-processing the data and creating the machine learning models are listed.

5.3.1 Data Scraping and storage

There are several technologies for data scraping, from desktop-based environments to frameworks and libraries for multiple programming languages [167]. The best choice is highly dependent on the task at hand as well as previous experience. As reflected in Table 11, scraping HLTV required an HTML crawler. Due to already having developed similar crawlers using python [186] and Beautiful Soup (a python library for pulling data from HTML and XML files) [187] as I was well aware of this tools' capabilities and fitness for this specific task. As for OP, the most efficient way of extracting data was through their API using the browser's embedded console. As such, pure Javascript was used for this task.

Considering exploration of data will be a big part of this project, the way the data is consumed will vary vastly. To name a few examples, a team can be represented by the average of their players, their ***id*** via one hot encoding or via player embeddings, each being reflecting widely different dataset structures. Furthermore, the merging of the datasets is planned, in order to properly combine the three different models. The best course of action would be to store the raw data, then curate and merge the information and then generating the final datasets from the curated data, as represented in Figure 33. For this purpose, a local mongoDB [188] instance was setup, as it allows for object-based data to be stored seamlessly.

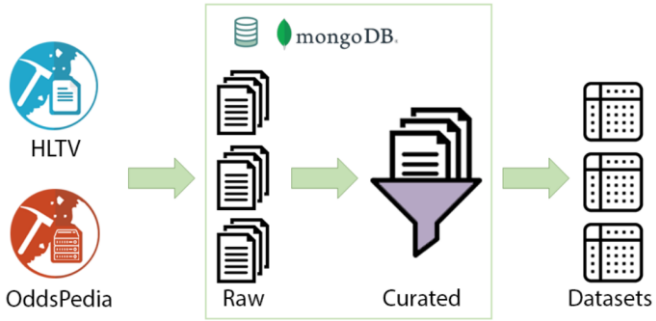


Figure 33 – Information storage

5.3.2 Data pre-processing

Python has become a staple for machine learning its extensive library repository can answer pretty much any necessity within this field, from pre-processing and unsupervised learning to reinforcement learning and parallel training [189]. One of the most popular tools within python machine learning is Jupyter Notebook [190], that provides a simple and intuitive interface that seamlessly blends text, code and results visualization, which is very useful when pre-processing data.

As for the actual pre-processing of data, four libraries stand out. The first is Numpy [191], that provides an array object that can handle multiple dimensions, objects derived from arrays such as masked arrays and matrices, and a wide range of functions for performing quick operations on arrays including mathematical, logical, and sorting operations. Secondly, Pandas [192] integrates effortlessly with Numpy and Jupyter and provides a flexible interface for working with labelled data. Then, Sickit-Learn [193] provides a plenitude of machine learning models, both unsupervised and supervised as well as crucial data-processing techniques such as one hot encoding. Finally, matplotlib[194] is also a staple of machine learning as it is a simple library for creating graphs that allow for better visualization of the data.

5.3.3 Machine learning models

In the Machine Learning chapter, it was extensively covered various unsupervised and supervised learning techniques, many of which will be used within this project. Sickit-Learn offers nearly all of those techniques apart from deep learning, and as such, is the clear choice for this type of models. For deep learning, Tensorflow, PyTorch and Keras are all valid options [195], with the main difference being that the latter is more high level, whereas the other two are faster and offer more flexibility. Having worked with both, PyTorch feels more natural and is the primary choice for deep learning, despite having a smaller community and overall online guidance.

To compliment Sickit-Learn's algorithms, it was also Intel's extension [196], which greatly increases the training and testing speed, allowing for faster and reliable iteration. Aditionally, Amazon Web Services were also used to accelerate the training times. More specifically, it was used "*ml.c5.9xlarge"* instances [197]. Finally, for the sentiment analysis task, OpenAI's GPT-3.5 model was used through their API.

5.4 Evaluation methods

When evaluating the performance of a predictor, accuracy (equation 1) is the most intuitive and commonly used metric. The accuracy is a measurement of how often the predictor is correct. Therefore, it is the division between the number of correct classifications and all classifications (correct and incorrect). Naturally, this value can be converted to a percentage, by multiplying by 100.

$$Accuracy = \frac{correct_{Classifications}}{correct_{Classifications} + incorrect_{Classifications}} \quad (1)$$

Nevertheless, according to Harrel [198], classification is a premature decision. If a classifier concluded that an event **A** has a 51% chance of happening, the prediction would be that **A** will happen. In other words, forcing a decision is likely to discard precious information that is imbued within its likelihood. Similarly, when evaluating a model, if it was only considered its accuracy, close calls would not be valued [199].

In the same vein, this document tackles eSports betting, whose odds are directly generated from the expected likelihood of an event happening, thus merely judging its accuracy would be

a waste of potential and an unfair evaluation. In statistics, the assessment between the predicted probabilities and the actual probabilities is made through calibration [200].

5.4.1 Calibration Metrics

The goal of calibration is ensuring that the predicted probabilities of a model align with the real probabilities. Ideally, these probabilities could be compared directly, but real probabilities are very seldomly attainable. In other words, real world events do not have an explicit likelihood of happening. Thus, statisticians have been studying the field for decades.

One of the simplest approaches to tackle this issue is known as absolute error, which is the absolute difference between the predicted value and the actual result (0 or 1). However, this metric is blind to calibration. Consider the example of a coinflip, a predictor that always predicted heads and a predictor that predicted a 50% chance of either would have the same mean absolute error.

Brier Score

One improvement to the absolute error metric is proposed by Glenn W. Brier in [201], where instead of computing the absolute error, the error is squared, meaning that predictions further from the real result (such as wrongly predicting a 100% chance of heads) have more weight. This metric is known as Brier Score (BS) and is defined in equation (2), where N is the number of instaces, K is the number of possible outcomes, $pred_{ij}$ is the predicted probability of class j for event i and $actual_{ij}$ is the binary result of the outcome of class j for event i.

$$BS = \frac{1}{N} \times \sum_{i=1}^{N} \sum_{j=1}^{K} (pred_{ij} - actual_{ij})^2 \tag{2}$$

Logarithmic Loss

Within machine learning and mathematical optimization, the cost of prediction inaccuracies is computed through loss functions. In fact, for binary classification BS is equivalent to a square loss function. One of the most commonly used loss functions is the Logarithmic Loss (LL). For a predicted probability p and actual probability y, equation (3) exposes how the LL is computed [200].

$$L(y, p) = -(y \times log(p) + (1 - y) \times log(1 - p)) \tag{3}$$

Expected Calibration Error

Although the real probability of an event is often unknown, it can be approximated by averaging the occurrence of similar events. More specifically, the predictions can be divided into bins, according to their predicted probability (confidence). The accuracy of each bin can them be compared to the bin's mean confidence to obtain the bin's calibration error. Next, the error of each bin can be averaged to obtain the Expected Prediction Error (ECE) [202]. This way, equation (4) is the formula of ECE, where n is the number of samples, M the number of bins, B_m is the m^{th} bin of which $accuracy(B_m)$ and $confidence(B_m)$ are the accuracy and confidence of, respectively. When using this approach, it is important to also account for the

number of empty bins along the ECE itself. As an example, in a betting context, a predictor that considers every event to have a 50% chance of happening and predicts both sides of a matchup, will have a single bin with 50% confidence and accuracy, resulting in a deceiving ECE of 0.00%.

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} \times |accuracy(B_m) - confidence(B_m)| \quad (4)$$

Within the context of this document, ECE will always be calculated using 20 bins. Additionally, considering the ECE is based on the accuracy of each bin, it will be treated a percentage, meaning its value is multiplied by 100.

5.4.2 Positive Expected Value betting

To analyse how the final model would perform in the real world, a betting simulation is made. More specifically, bets are made according to the Expected Value (EV) of the odd offered by the bookmakers. As described in 2.1.2, the EV represents the expected return on investment of a given bet. In order words, if it was bet on odds with a positive 5% EV, long term, it should translate to a 5% profit.

To compute the EV a bookmaker odd is needed. Using the highest odd would result in extremely inflated results and expose the experiment to unrealistic odds, that are the result of promotions or reading errors. Similarly, using the bookmakers' average would be very biased towards higher odds because the minimum possible odd is 1, while there is no upper limit. On the other hand, using the lowest odd would result in significantly fewer bets, and thus increase the effect of randomness. This way, the bookmakers' odds median was chosen to compute the EV a bet.

While calibration metrics can be applied to both sides of the matchups, positive EV betting, aside for rara arbitrage opportunities, can only be made on one side. Furthermore, not every match will have a positive EV betting opportunity. Both of these factors vastly reduce the number of matches used to compute the EV betting simulation, and thus, this metric is only discussed in chapter 9.3.

5.5 Ethical Aspects and Threats

As machine learning increasingly becomes more prominent in decision-making such as in employment, credit rating and social justice, the themes ethics and security also rise in notoriety. Firstly, ML algorithms learn from data and while the algorithms themselves are not biased, the data might be, causing the ML model to be unfair on its decisions [203]. Furthermore, if the training data is not properly protected, it is possible to alter it and make the model deliberately biased or unstable [204].

5.5.1 Ethical and legal concerns

Traditionally, the accountability of an advisor, be it a individual or a collective, scales with the impact of said advice. A meteorological institute wrongly predicting a rainless day is understandably of less importance than a non-licenced financial coach promoting investing savings in a failing business. Similarly, the latter is not as grave as a cancer misdiagnosis. However, when it comes to artificial intelligence, even if we can measure the impact of a wrong prediction, it is often unclear who should be held accountable [205].

Considering OddAssist is a betting recommendation system, there could be considerable legal concerns regarding financial advice. A wrong prediction could gravely impact one's financial health. However, such is not the purpose of this research. OddAssist is not planned to be publicly available nor commercialized.

If it were to be commercialized, further ethical and legal issues might arise. Using as reference the Foundation for Science and Technology's ethics self-assessment guide [206], only issue 2 – Humans and 4 – Protection of personal data might apply. In both issues, the data collected for the sentiment analysis is at risk. It contains information about the players real names and nationality along with the comments made by thousands of users.

Within human issues (2), this data does involves participants that were unable to give informed consent and is very likely to include minors. Regardless, all information was consciously made public and consists mostly of discussion about the game, teams and players, which should not trigger any privacy concerns. Similarly, the protection of personal data (4) is also not applicable, the players are public figures, and their personal information is readily available online. In regards to the users that comment, the only parameter that could be considered personal data, apart from information they willingly share in the comment itself, is the username of each user on the HLTV forum.

Finally, there could be tangible ethical and legal concerns about scraping information from other sources. Although, this research does not breach any of the enumerated precedents described in [207], namely:

- **Terms of Service –** the data scraping was made without ever login into the websites and thus, no user agreement was signed;
- **Copyrighted material –** the data obtained will not be distributed nor used as-is, thus is will not constitute copyright infringement;
- **Damage to de website –** precautions to prevent overloading the website were taken, such as not constantly spamming it when requests are getting denied and waiting a fixed interval between requests;
- **Individual Privacy –** all the data scrapped was data that is publicly available without requiring a log in, and thus does not constitute any more of a privacy concern than the original website;
- **Diminishing Value for the Organization –** the data collected is not being redistributed nor commercialized, therefore, is not driving away traffic from the website.

5.5.2 Threats to Machine Learning

Machine learning models are created from data and make predictions based on new data. In both instances, the data can be corrupted, resulting in wrong predictions that can be exploited by bad actors. According to [204], there are 5 types of threats:

Training data poisoning

Training data poisoning is the process of adding maliciously produced data to a training dataset with the goal of making a ML model make bad predictions or act in other unfavourable ways. By poisoning the training data, it is hoped that the model would be trained to make mistakes that the attacker can predict and exploit.

Adversarial Examples

Adversarial examples, on the other hand, refer to inputs that have been specifically designed to cause a ML model to make an incorrect prediction. These inputs are typically created by adding small, carefully chosen perturbations to legitimate inputs, with the goal of fooling the model into misclassifying the input. Unlike data poisoning, adversarial examples are not added to the training data, but rather used to test the robustness of a trained model.

Backdoors in the training data

A backdoor is a method like training data poisoning, where a bad actor can trigger a behaviour in a ML model by introducing a specific input or feature. These inputs or features are added to the training set to create a hidden method of controlling the model's behaviour without being detected by users or even the developers.

Model stealing

A model stealing attack, also known as model extraction, is a type of attack in which a bad actor attempts to reverse-engineer a ML model to recreate or steal it. The goal of a model stealing attack is to gain access to the underlying architecture and parameters of a machine learning model, which can then be used to impersonate the model, make predictions on new data, or perform other malicious actions.

Recovery of sensitive training data

Similarly, the privacy of machine learning systems can be jeopardized through attacks that aim to learn about the training data used. In a membership inference attack the bad actor tries to assess whether a certain piece of data was used in the model's training. In the same way, a model inversion attack, the perpetrator seeks to deduce details about the training set from the model.

Considering the proposed solution will constantly retrain the models as new information arrives, poisoning the data and provide adversarial examples is possible. Firstly, the websites from which the data is scrapped could blacklist the machine’s IP and deliberately feed false information about the game results and odds. Although, this is very unlikely due to the sheer volume of requests they receive every day. However, the most likely scenario and vulnerability of the proposed solution lies within the sentiment analysis. Even without any knowledge about the architecture of the system, anyone can comment unlikely scenarios under a match discussion which will be taken at face value by the sentiment analysis module. Regardless, this could be minimized by developing a trust score system that rates the commenters’ reliability.

Finally, since the model is being developed by a single person and will not be distributed, the threats of backdoors, model stealing and recovery of sensitive training data are not applicable.

5.6 Chapter Conclusions

In this chapter, first, the methods, materials and tools used to develop the proposed solution were described. For web scrapping, python was chosen, specifically, the library BeautifulSoup, due to previous experience and success on scrapping the same platform using this tool. For storing the data, a local MongoDB database was chosen, due to the data being complex objects that are easier to navigate if kept as-is. Lastly, for machine learning, python was chosen due to the variety of libraries available, from Scikit Learn to PyTorch, which allied with Jupyter notebook, matplotlib and Amazon Web Services, offer an environment that allows for quick and effective iterative development.

Furthermore, the evaluation metrics to be used to access the models' performance were detailed. More specifically, the accuracy will be used as a baseline because it is intuitive and a great indicator of the models' general performance. On the other hand, Brier score, log loss and expected calibration error will ensure the model is well calibrated, with the log loss being more punishing of wrong high confidence predictions.

Furthermore, an assessment of ethical and legal aspects that might apply to the proposed solution was made along with an exploration of the threats to machine learning models and concluded that, as long as OddAssist is not commercialized nor distributed, there were no ethical concerns and that the biggest vulnerability of this solution lies within the sentiment analysis module. Regardless, the latter can be fixed by rating the users' reliability.

6 Prediction through match history

In this chapter, first, the data obtained through scrapping is explored in order to find patterns that might hint what approach to follow is explored. Afterwards, multiple approaches for representing the data and player performance are analysed to select the best features. Additionally, experimental findings using a multi-input-label neural networks are reported. Finally, the best representations found through feature selection are compared and the best is chosen.

6.1 Data exploration

As mentioned previously, data exploration can reveal patterns within the data that can inspire different data representations. Thus, the distribution of games throughout the years is explored. Additionally, patterns around the lifespan of teams as well as the impact an individual player can have on certain maps are also highlighted.

6.1.1 Games played

The first thing analysed was the distribution of games played by year, as seen in Figure 34. Overall, the first two months of the year tend to have less games, as there are fewer events running. Additionally, older games are more likely to not be labelled on the environment they took place in (LAN or Online). Also, the effect of the pandemic over the years is clearly reflected by the Online to LAN ratio from 2020 to 2023.

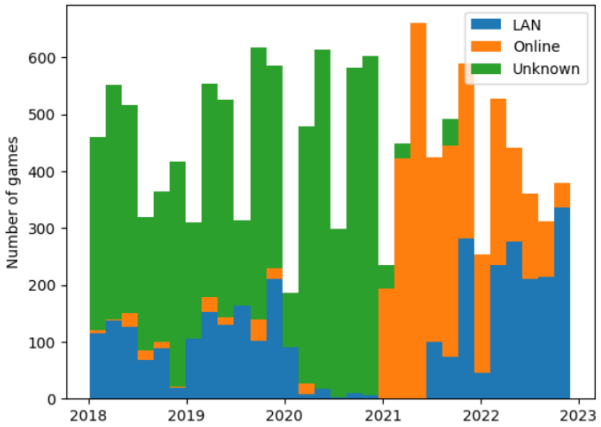


Figure 34 – Distribution of games per year

CS:GO has a rotating map pool, which means that at any given time, there is a limited amount of maps that are available for competitive play. Consequently, maps such has Inferno and Mirage that have been staples for the duration of the dataset are the most popular while maps like Cobblestone and Cache have become obsolete since. There are also other alternatives such as Ancient and Vertigo that were added more recently to the map pool and are rising in popularity, as reflected in Figure 35.

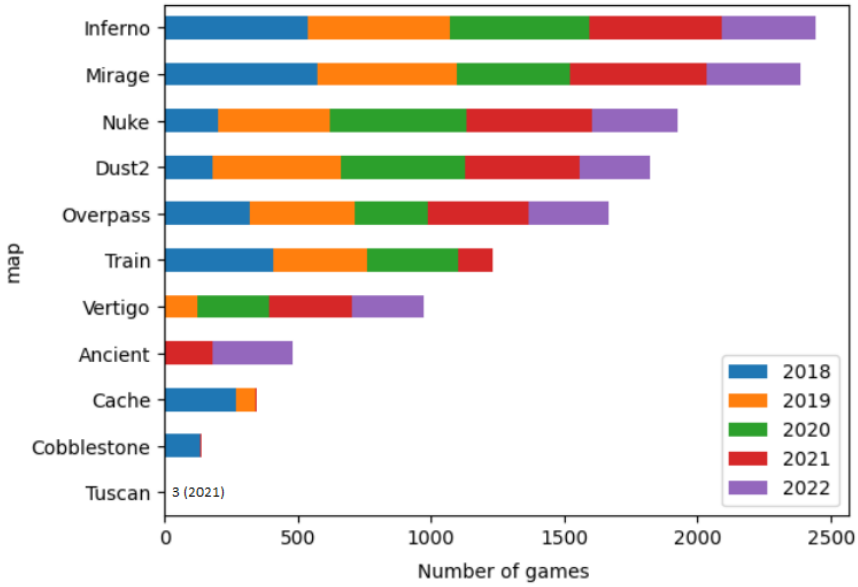


Figure 35 – Distribution of maps played per year

6.1.2 Team and player dynamics

Firstly, due to the dataset consisting of games that contained at least one top 20 team, it is naturally biased towards top ranking teams. Thus, as reflected in Figure 36, the higher the team's ranking, the higher the number of games played.

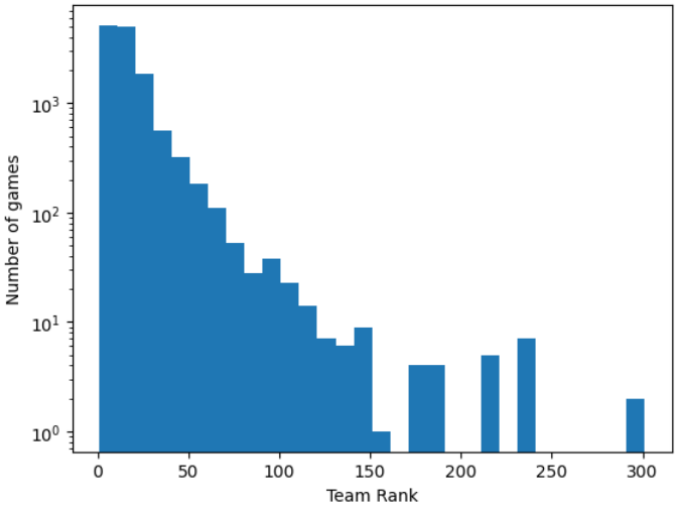


Figure 36 – Distribution of games played per team rank

Team cores and lifespan

To understand team dynamics, it was analysed the lifespan of 5-players lineups. In other words, how many games a team of 5 players plays together. There are a total of 1687 different 5-player teams that have played at least 1 game together within the last 5 years. Nevertheless, as reflected in Figure 37, most of them played under 10 games together. It should also be noted that the scale on the y-axis is exponential, else the disparity between 1 and 10 games and the remainder of the histogram would turn the latter unintelligible.

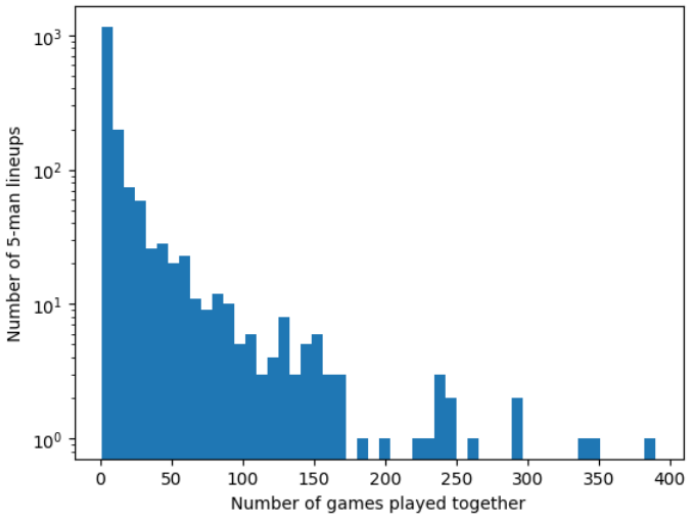


Figure 37 – Distribution of unique 5-man lineups per games played

In CS:GO, building a team around a set of players, called the **core** is not unusual as well as swapping the remaining players when the team underperforms. Figure 38 represents the distribution of games played by 4, 3 and 2-player cores and reflects this trend. As the cores become smaller, the distribution gets more even. Furthermore, if we consider the exponential nature of the y-axis, the difference from a 4-man to a 3-man core is much more accentuated than from a 3-man to a 2-man core.

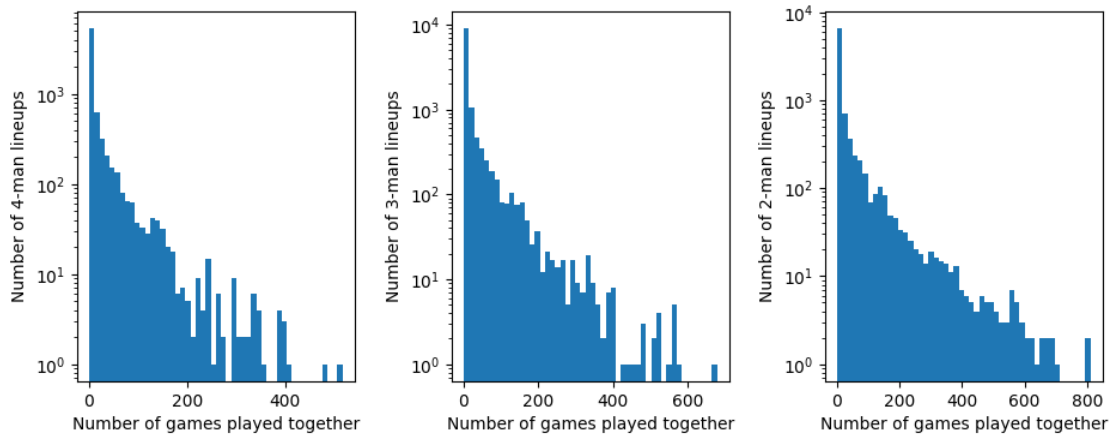


Figure 38 – Distribution of games player per 4, 3 and 2-man cores, respectively

When analysing the commitment between players and teams, it was found that for teams that peaked between rank 1 and 30, the lower the ranking, the more the players moved around. As represented in Figure 39, this trend does not continue from rank 30 onwards, but that is highly likely to be due to the diminishing amount of data regarding lower tier teams within the dataset as well as the fact that lower tier teams are often not ranked. A similar trend was found when considering the team's average rank instead of peak. In fact, consistently higher rated teams were even less likely to swap players than teams that peaked higher. One possible explanation is that teams that peak in a higher rating tend to make changes when they can't reclaim their ranks for an extended period.

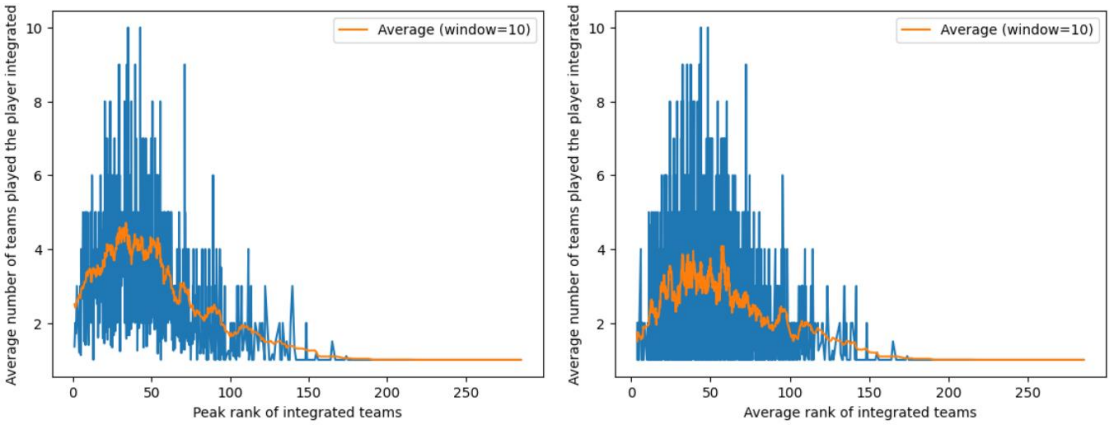


Figure 39 – Average number of teams the player integrated by rank the teams

Team cores and map influence

When studying the win rate per map of the top 20 5-man lineups over the last 5 years (Figure 40), it is clear not even top teams excel at every single map. Nearly every team has at least a map that they tend to avoid, and hence have less wins on. In contrast, they also have a map where they boast win rate over 70%. Therefore, the map played is a huge factor for determining the outcome of a game.

The impact of a player in a team can also be identified by looking at the teams highlighted in Figure 40. The 4-man core of “s1mple”, “electronic” “Booml4” and “Perfecto” had two major lineups, the first, where the fifth player was “flamie” and another where it was “b1t”. Just by swapping one player, the team improved its performance on all of their most played maps, with Nuke standing out with an increase of 28 percentile points, going from 64% to 92% win rate.

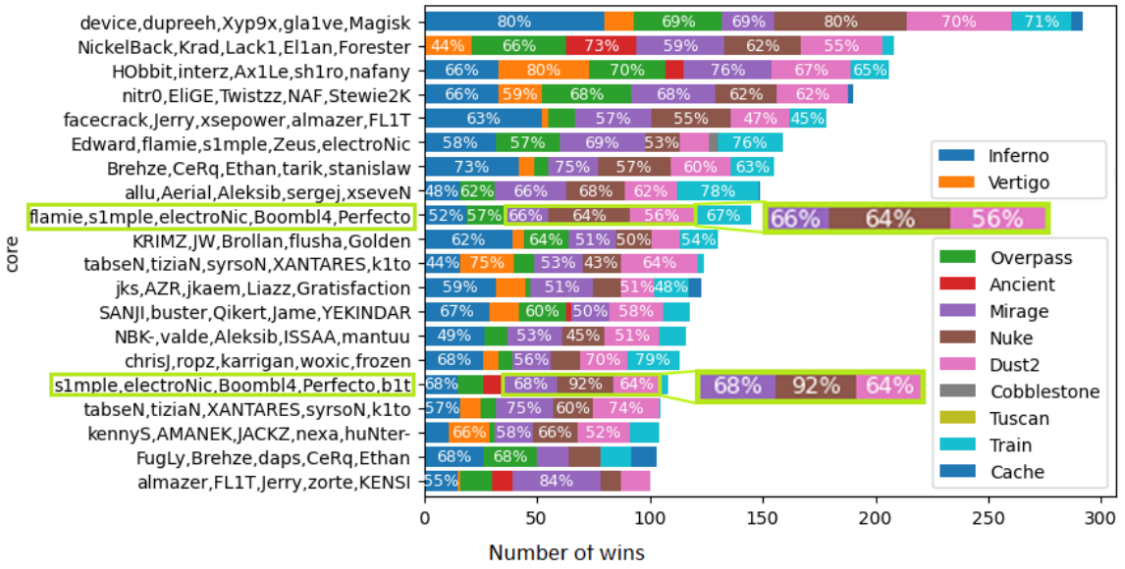


Figure 40 – Top 20 3-man cores with the most overall wins and their win rate per map

6.2 Curating the dataset

Since the objective of this model is to predict the outcome of games between teams, the balance of the dataset is crucial. CS:GO does not possess a home team advantage like football. This is important because in CS:GO the order of the teams is interchangeable. In this scenario, balancing means ensuring that both sides of a matchup are taken into consideration when training, otherwise the matchup of “Team 1 vs Team2” would be considered different from “Team 2 vs Team 1”. This is addressed by duplicating the data and mirroring the matchup’s information.

Furthermore, for this model, four different representations for the match history were tested, while they don’t necessarily require different datasets, it simplifies the Pre-Processing steps. As an example, a team in a match could be represented by its name (T) or by its players (P) while a match could be represented as the whole best-of series (M) or by its individual games (G), thus resulting in four total combinations of representations. Regardless, the data is always represented in the same manner, so that the datasets are interchangeable. This is achieved by (I) always representing teams and maps as an array that will later be encoded to features; and

(II) representing the outcome as the game difference (games won - games lost) instead of a Boolean value, as exemplified Table 12 (omitting columns).

Table 12 – Example line of each combination dataset – (T)eam, (P)layers, (M)atch and (G)ames

Fields	t1Rep	t2Rep	t1Rank	t2Rank	maps	gameDiff
TM	[team1]	[team2]	2	13	[map1, m2, m3]	2
TG	[team2]	[team1]	13	2	[map1]	-1
PM	[player8, p9 …]	[player1, p2 …]	13	2	[map1, m2, m3]	-2
PG	[player1, p2 …]	[player8, p9 …]	2	13	[map1]	1

6.3 Pre-Processing

After curating the dataset, the first step is handling missing values. Throughout the dataset, it was found that the maps, team rank and the type of match (LAN or online) had missing data. The first occurred only one time, in a show match, and thus was discarded. Team ranking, on the other hand, was missing 117 times, and from a partial analysis of these cases, it was concluded that these are makeshift teams that only played for a few games together. Regardless, these games might provide valuable insight between player interaction and rather than removing these samples from the dataset, their rank was defaulted to 300 as to account for poor team synergy. On the other hand, the game type is null for nearly half of the dataset and will not be considered.

Next, the team's representation and maps played were initially encoded into features using Scikit-Learn's multi-label binarizer [208] which transforms the labels into a sparse feature representation, as exemplified in Table 13.

Table 13 – Multi-label binary encoding example

Original	Encoded		
maps	map1	map2	map3
[map1, map2]	1	1	0
[map3]	0	0	1

Nevertheless, player performance was also tested, requiring a method that could attribute continuous values in a multi label format. Thus, a custom non-binary multi-label encoding function was created that, as seen in Table 14, uses two columns, one contains the labels (in the example, players) and the other contains the values to encode (playerScore).

Table 14 – Custom Multi-label encoding example

Original		Encoded			
players	playerScore	p1	p2	p3	p4
[p1, p2]	[10.0,5.3]	10.0	5.3	0	0
[p3, p4]	[1.0,0.3]	0	0	1.0	0.3

6.3.1 Concept drift

Considering that CS:GO is ever-evolving and a team's performance can vary drastically within a few weeks, before any major testing and feature selection, a case study was made to understand how long it takes before a model drifts and becomes obsolete. Over a testing period of November 2022 to March 2023, there was a total of 90 days with games played. Each month, a new model was created trained on data up to that date. Those static models are then compared with a predictor that is retrained daily. As shown in Figure 41, the accuracy (a) of the static models peaks within a week and then quickly decreases, stabilizing at 50%. Similarly, in the positive EV betting simulation (b), the profit of the static models is vastly different than the daily retrained one, with the latter having a profit over 25€ while the model from November 2022 barely reaches a 5€ profit.

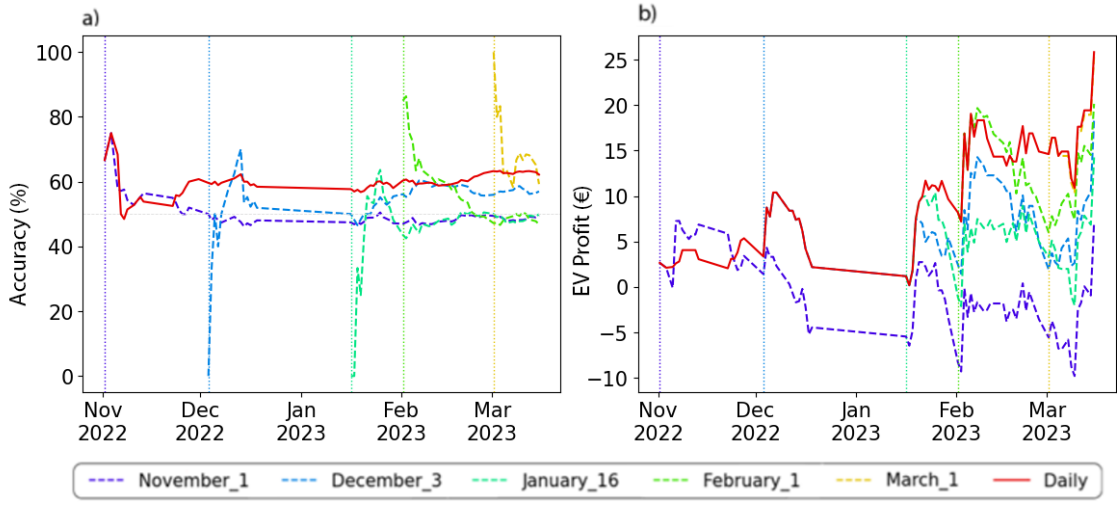


Figure 41 – Comparison of Accuracy (a) and EV Profit (b) from November 2022 to March 2023

In the same vein, the calibration metrics were also tested, with similar conclusions. As exposed in Figure 42, the BS (a) and LL (b) had identical patterns, hinting that they might be redundant. Regardless, it is clear that the static models become less calibrated over time. It is also noticeable that as the testing data increases, the calibration ECE (c) decreases, likely due to the bins having more samples.

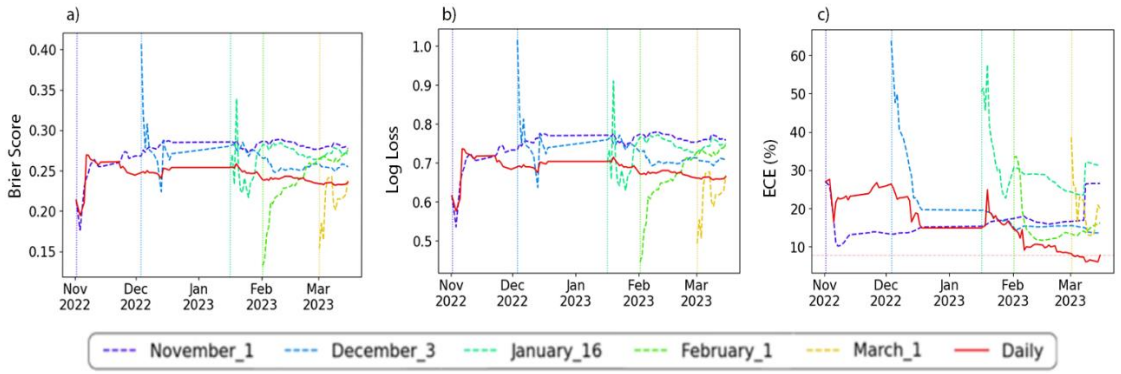


Figure 42 – Comparison of BS (a) and LL (b) and ECE (c) from November 2022 to March 2023

In this instance, the drift could have been addressed by using cross validation, mixing the training and the testing data to provide better coverage. Nonetheless, that would have been a

data leak, as we would be using data from the future to predict an event that occurred in the past. To minimize luck being a factor, the models to be tested over a large period, but at the same time, the models quickly become obsolete, which does not allow for the accurate testing. Therefore, the tests should be made retraining the models as often as it is viable.

6.3.2 Feature Selection

To select the best possible feature combination, it was compared multiple representations. The training and test data was selected sequentially instead of randomly, to prevent data leaks and better represent the model's actual use case. Once the final model is trained, it will be constantly fed new information, sequentially.

Therefore, the dataset is initially split using a 60-40 holdout, with 3725 matches for training and the remaining 2500 matches used for testing. The testing set consists of matches that took place between June 2020 and March 2023. Within each test, the models were retrained at least once a month, to prevent concept drift. As an example, the model testing the matches of June 2020 was trained on data up to May 2020, while the model used to test the matches of March 2023 was trained on data up to February 2023. Depending on which feature was being tested, different machine learning algorithms were used. Features that were expected to independently contribute to a prediction, such as player encodings, the tests were conducted using Logistic Regression (LR). As LR are extremely fast to train, the models were retrained daily. On the other hand, features that, on their own, do not have predicting prowess, such as maps played, the Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel was chosen. Since SVMs are more computationally expensive, these models were retrained once a month.

Both maps and dates of match are useless on their own. As an example, the maps by themselves have no impact on the outcome, because their encodings are the same among both sides of the matchup. However, when coupled with the teams that are playing the map, it might become a factor. Linear models, such as LR, could theoretically handle this feature interaction, but would require explicitly mapping of those features, i.e. creating one feature for each team-map pair, for each side of the match up. Although possible, this would exponentially increase the number of features of the model, and thus it is more efficient to use techniques than can capture non-linear interactions, such as SVM with the RBF kernel. For these features, in this section, SVM is used, but Attachment A displays the results of using LR instead, as expected, adding these features doesn't have a sizeable impact on the models' predictions.

Data representation

As previously mentioned, there are 4 total high-level representations for the data: (I) game-wise team encoding; (II) game-wise player encoding; (III) match-wise team encoding; and (IV) match-wise player encoding. Hence, the first step is to understand which representation has the most potential. For each of the four major representations, it was also tested the addition of team rank and maps played. However, by adding the team rank to the encodings, it would be unclear if the performance changed due to feature interaction, or due to the team rank itself holding predicting prowess. Therefore, models with just the Team Rank per Game and Team Rank per Match were also tested as a baseline comparison, displayed in Table 15.

A surprising revelation was that the base representations Team Rank by games and matches outperformed every other games and matches representation, respectively, in every metric but

the calibration bins, due to the fact that they have significantly more empty bins than the other representations.

Overall, predicting games instead of a whole match resulted in worse accuracies, Brier scores and log loss while being significantly more computationally intensive, thus they will be discarded. Regardless, they could prove useful for making prediction mid-match, for example, predicting the second game of match.

Table 15 – Comparison between team versus player binary encoding, game versus match-wise prediction and the effect of using team rank and/or maps played

Data representation		Accuracy (%)	Brier Score	Log Loss	Calibration bins	
					ECE (%)	Empty
Team Rank Games		61.35	0.234	0.660	2.30	10
Game Team Binary Encoding (I)	None	57.23	0.243	0.679	5.21	10
	Rank	57.84	0.242	0.677	5.53	10
	Maps	58.84	0.240	0.674	4.68	8
	Both	60.16	0.238	0.669	13.46	6
Game Players Binary Encoding (II)	None	59.87	0.237	0.667	3.72	7
	Rank	59.77	0.238	0.669	4.24	7
	Maps	60.36	0.235	0.663	2.54	5
	Both	60.43	0.235	0.662	2.07	5
Team Rank Match		**65.85**	**0.218**	**0.628**	2.42	8
Match Team Binary Encoding (III)	None	61.42	0.233	0.659	3.23	8
	Rank	62.52	0.230	0.653	2.51	6
	Maps	61.86	0.230	0.652	**1.27**	4
	Both	63.02	0.226	0.643	1.50	4
Match Players Binary Encoding (IV)	None	64.64	0.222	0.635	2.50	**2**
	Rank	64.86	0.221	0.633	2.03	**2**
	Maps	64.39	0.221	0.633	2.06	**2**
	Both	65.11	0.220	0.630	1.93	**2**

On the other hand, using player encodings instead of team encodings did yield noticeable improvements in the accuracy, Brier score and log loss. The number of empty calibration bins also decreased drastically, while maintaining relatively low ECE.

Furthermore, adding either the team rank or maps played improved the calibration metrics across the board being and were more significative in the team representations. However, in a real-world application, the maps will seldom be considered, because the maps to be played are only revealed minutes before the match starts, leaving a very tight time window for a prediction-based bet to be made.

Date of match

Next, adding the normalized epoch representation of the date of the match was tested. To get a better understanding of how the date impacted the prediction, it was tested on every match-wise representation, excluding maps. The date was included using 2 different approaches: in the first the date is used as an extra feature while in the second the date translates to the weight of each training sample. The former is expected to help identifying the periods where the teams were peaking, whereas the latter overvalues later results. The results are exposed in Table 16.

Table 16 – Comparison between not using dates and using them as a feature and weight

Data representation		Accuracy (%)	Brier	Log Loss	Calibration Bins	
					ECE (%)	Empty
Team Binary Encoding	No date	61.42	0.233	0.659	3.23	8
	Feature	62.60	0.229	0.650	1.75	6
	Weight	62.62	0.230	0.652	**1.34**	6
Team Binary Encoding + Team Rank	No date	62.52	0.230	0.653	2.51	6
	Feature	64.02	0.226	0.643	1.73	6
	Weight	64.02	0.226	0.643	1.58	6
Players Binary Encoding	No date	64.64	0.222	0.635	2.50	**2**
	Feature	64.62	0.222	0.635	2.17	2
	Weight	64.14	0.222	0.635	2.29	**2**
Players Binary Encoding + Team Rank	No date	**64.86**	**0.221**	0.633	2.03	**2**
	Feature	64.84	**0.221**	0.633	2.11	**2**
	Weight	64.54	**0.221**	**0.632**	2.04	**2**

Observing Table 16, team encodings generally improved in every metric. On the other hand, for player encodings, using date either as a feature or as a weight had very little impact on the predictions. However, they did improve the calibration bins significantly, mainly when used as weight.

The gap between the improvements of each encoding was expected due to the Team and player dynamics discussed previously: it is known that in CS:GO, the players within a team have more impact in the outcome of a match than the team's name value. Thus, by linking the team's name with the match date, the model can infer the different peaks and valleys of the team across its rosters. On the other hand, player encodings already allow for this information to be inferred, as it is rare for the same exact group of players to have massive variance in performance.

Previous player performance

To provide more context about the players current performance, it was tested replacing player binary encoding with stats encoding. In other words, instead of using 0 or 1 to tell if a player is playing in a match, it is used 0 or a positive continuous value normalized, providing not only if a player will play, but also information about their recent performance. To represent the recent performance, the following metrics were considered:

- **Kills –** Number of times the player finished an opponent in a match;
- **Deaths –** Number of times the player died in a match;
- **Assists –** Number of times the player helped finishing off an opponent in a match;
- **KDA –** Feature extracted by adding the kills and assists and dividing by the deaths of the player in a match, it is metric regularly used in videogames;
- **Rating –** A complex rating system developed by HTLV [182];
- **Average damage per round (ADR) –** The average amount of damage inflicted by the player, per round, in the match.
- **RADR –** Feature extracted by multiplying thr Rating and ADR.

Furthermore, it of utmost importance to settle on how many games to use for each metric. An interval of 3 to 19 games was analysed. Additionally, the metrics over those games needed to be aggregated into a single value. For this purpose, 4 approaches were studied:

- **Average –** The mean value of the performance metric over the selected games;
- **Median –** The median value of the performance metric over the selected games;
- **Recency-weighted average –** Mean where more recent games were accounted for more heavily, eg: for 3 games, the last, penultimate and antepenultimate had a weight of 3, 2 and 1, respectively;
- **Opponent-ranking-weighted average –** Mean where higher rated opponents were weighted more heavily, eg: an opponent with rank 1 would have a weight of 500, whereas a team of rank 100 would have a weight of 400.

This resulted in 420 unique combinations, that could be further combined to test for feature interaction, exponentially increasing the number of scenarios and computing power needed to test them. Therefore, other feature selection techniques must be employed.

Naturally, most of these 420 features are redundant, having information relative to the last 14 or 15 games is unlikely to impact the prediction power of the model. To detect these redundant features it was used a correlation matrix [209]. Within the python ecosystem, the most popular methods are Pearson, Spearman and Kendall, of which the latter is the most robust to both outliers and non-linear relationships [210], [211]. Therefore, Kendall's method was used to identify redundant features to remove as well as correlation patterns. Nevertheless, the dataset has 2095 players, each with 420 possible features resulting in a matrix of 774 billion correlations. To reduce this, the players were ordered by number of games and only the games between the top 150 most popular players were considered. This resulted in a near tenfold reduction in the number of games to 893 and a correlation matrix with under 4 billion entries. Furthermore, it was only considered players on one side of the matchup, as considering them on both sides would duplicate the number of features per player, to 840 and square the matrix size to nearly 16 billion entries.

The generated matrix has 63000 features on each axis, but it can be decomposed into multiple 420x420 combination chunks each representing the correlation of the features between players. The absolute values of the chunks were then summed and formed Figure 43. In matrix (a), Deaths correlation with itself overshadowed every other correlation, therefore, it was created a matrix without it (b).

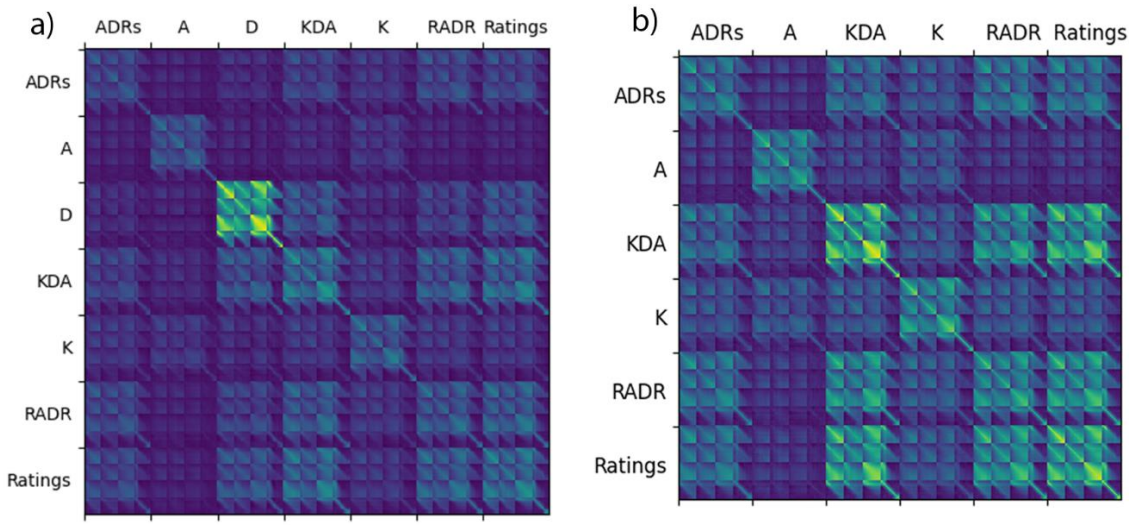


Figure 43 – Correlation Matrix between player performance metrics

Apart from Deaths, the Kills and Assists also show very weak correlation with the other metrics. The ADRs also had overall very little correlation with the other metrics. On the other side, KDA, RADR and Rating all seem to convey similar information, as they have comparable patterns of correlation with the other features. KDA is slightly more sensible to Deaths and out of RADR and Ratings, RADR had a slightly lower correlation to the KDA than the Ratings. Therefore, the following player performance indicators will be tested:

- **Kills, Deaths and Assists –** these metrics had the weakest correlations with the remainder;
- **ADR –** this metric had a unique pattern;
- **RADR –** out of redundant features KDA, Rating and RADR, RADR has the least correlation with the remainder.

Figure 44 is a close up of the first line of the matrix – the correlation between ADRs and the remaining features – and also labels the different aggregation methods. Observing the figure below, it is apparent that using either average, median or recency-weighted average as aggregation method show similar patterns across every metric, whereas the one generated by opponent ranking is vastly different.

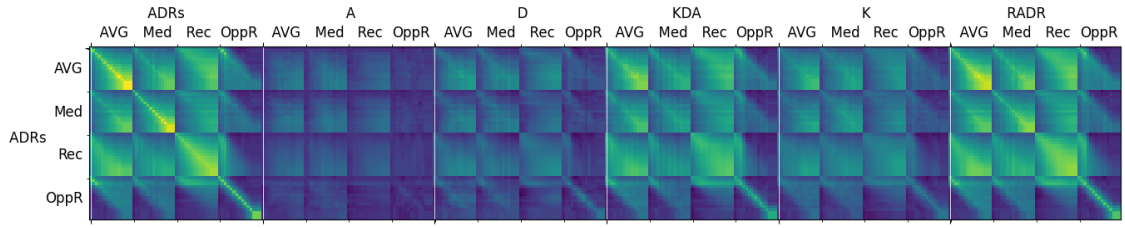


Figure 44 – Partial Correlation Matrix between player performance metrics, focusing on aggregation methods

However, to better understand those patterns, zooming in on the matrix is necessary. Figure 45 focuses just on the ADR metric and the similarity between different aggregation methods and number of games. In this example, the average and the median have very similar patterns of correlation with the remaining aggregation types. Also, recency-weighted average has a

stronger correlation with the other metrics when the former has more games than the latter and the opposite effect is observed for opponent-ranking-weighted average. This way, the tests will be made using only the averages: recency-weighted, opponent-ranking-weighted and unweighted (mean). The median will not be tested because it had similar patterns to the latter.

Furthermore, the higher the number of games, the stronger the correlation between the metrics, regardless of the aggregation method. Also, The number of games plays a major factor in the correlation strength, with further apart numbers having the lower values. Therefore, it will only be tested aggregation of the last 3, 11 and 19 games, representing 3 evenly spaced-out points of the spectrum.

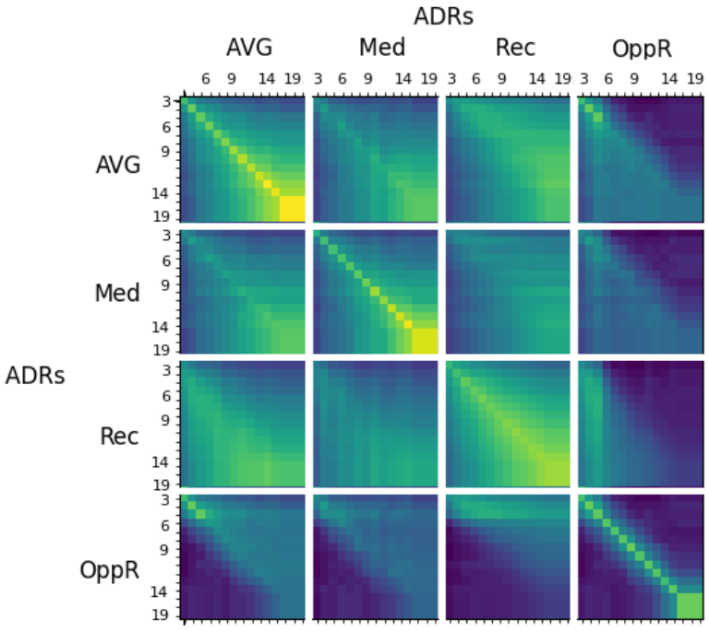


Figure 45 - Partial Correlation Matrix between player performance metrics, focusing on game number

The goal is to test how the player performance indicators used impact the prediction, therefore, the representations had no other features apart from the player encodings, which also allowed for the tests to be made on the same dataset of 2500 matches using Logistic Regression retrained daily.

The results of the experiment are further detailed in Attachment B, overall, every metric was an improvement over binary player encodings, but the best performer, by every evaluated metric, was the RADR encodings, and therefore their results are displayed in Table 17. RADR encodings reached record high accuracies of 66.36%, and record lows for the BS, LL and ECE with 0.214, 0.618 and 1.31%, respectively.

Within the RADR encodings, the simple averages performed the best overall, but the recency-weighted average had similar results and a slightly better ECE, and thus was selected. Similarly, it was chosen 19 over 11 games, because it had better calibration metrics ECE while only losing 0.30% accuracy.

Table 17 – Comparison between different binary and RADR encodings

Metric	Aggregation Type	Number of Games	Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
						ECE (%)	Empty
RADR	Average	3	65.80	0.217	0.625	2.55	2
		11	66.04	0.214	0.618	1.37	2
		19	66.14	0.215	0.618	1.63	2
	Recency	3	65.46	0.219	0.628	1.95	2
		11	66.36	0.215	0.618	1.6	2
		19	66.06	0.214	0.618	**1.31**	2
	Opp Ranking	3	65.48	0.218	0.626	2.47	2
		11	65.32	0.217	0.624	1.57	2
		19	64.96	0.218	0.625	1.31	2
RADR 19-Game Recency		Rank	**66.66**	**0.213**	**0.614**	2.10	**0**
		Date	66.14	0.214	0.618	1.34	2
		Both	**66.66**	**0.213**	**0.614**	2.10	**0**
Player Binary Encoding			64.45	0.223	0.636	4.41	2

Considering the previous results of the Table 16 – Comparison between not using dates and using them as a feature and weight, it was also tested the impact of adding team rank as well as using date as weight on the RADR 19-game recency-weighted average. It was observed that the date improved the accuracy by 0.08% whereas rank increased the accuracy by 0.60% as well as reduced the Brier score in 0.01 and log loss in 0.004. The ECE increased by 0.66%, but also filled 2 previously empty bins. As seen in Table 18, it significantly improved the bins on the extremes, while maintaining good accuracies for the middle bins. Even though using the date as weights did not have a lot of impact, it is more future proof and therefore, both team rank and date as weight will also be considered.

Table 18 – Effect of rank on the calibration bins of RADR 19-game recency-weighted average

Bin	[0-5[	[5,10[	[10,15[	[15,20[	…	[80,85	[85,90[	[90,95[	[95,100]	ECE
Rank	10.00	20.00	10.28	21.11		78.87	91.75	77.78	100.00	2.10
No Rank	-	35.29	12.56	15.15		86.57	86.17	64.71	-	1.31

6.4 Experimental Findings

When mapping how players interact in CS:GO, one could say that each player brings a certain value to a map and that the winner of a map would be the team with highest cumulative player value. This interaction could be mapped to a neural network as represented in Figure 46, where the initial layer takes the player embeddings as inputs while the second layer captures the map embeddings. In the example, it is a match between team B and team C, thus the players of those teams have 1 as input, while other teams will not affect the outcome. Similarly, Map 1 will be played, and therefore will have 1 as input. Each player has different impact on a map, which is represented by their weight on that map. Therefore, the outcome of a map is the sum of the weights of all participating players multiplied by the embedding of the map. In the same vein, the match outcome is the sum of the outcomes of each map.

The concept could even be further expanded by adding another layer of players that would represent how each player affects the performance of other players in the game, both as a teammate and as a foe. If the network is deep enough, these relationships can also be found through training, without the need of having multiple input layers. Nevertheless, decomposing relationships by layers would allow for better explainability and a lighter model, that would have a fraction of the neurons of a deeper solution, while achieving the same results.

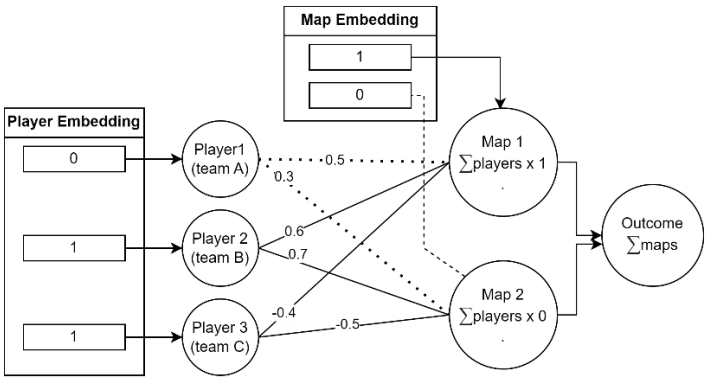


Figure 46 – Representation of a multi-input-layer neural network

Pytorch provides great flexibility and allows the user to define, amongst other things, the feed forward function, in other words, how the NN consumes the inputs and how each layer interacts with the following layer. Therefore, it was made a prototype testing the architecture described above as well as the extended version, none of which yielded any promising results, not even being able to achieve high accuracy within the training set. NN are usually not good for sparse data, and player embeddings have a dozen of 1's within thousands of 0's. This approach could still have potential, but further research should be made using attention mechanisms.

On the grounds that this approach is easily explainable, it was built a graph that would allow for the same architecture. Firstly, as this was not a linear problem, optimization of the graph weights was made using particle swarm optimization, which did not yield any promising results. Then, it was added a functionality that created a flattened version of the graph, as represented in Figure 47, allowing for linear programming to be used. Unfortunately, after 17h of training, the machine ran out of memory and thus the experiment could not be concluded.

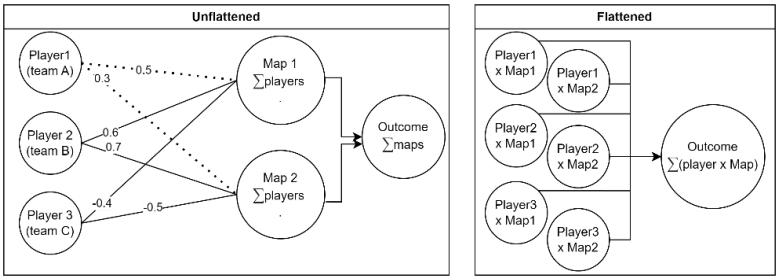


Figure 47 – Graph flattening example

6.5 Model selection

Aside from the experiments mentioned above, it was also tested deeper neural networks, also without attention mask, but due to taking a lot of time to train and still performing significantly worse than the models explored in the Feature Selection, it was discarded.

Furthermore, the best performing techniques reviewed in the Outcome Prediction state of the art, namely Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting Machine (GBM) and Random Forest Regressor (RFR). RFR was chosen over RFC because the former performed better in early testing. It was also tested Naïve Bayes Classifier (NBC) as a baseline. The algorithms were tested with the following settings:

- **Logistic Regression (LR) –** 100 000 maximum iterations;
- **Gradient Boosting (GB) –** a max depth of 4, a learning rate of 0.2 and 1446 estimators;
- **Support Vector Machine (SVM) –** RBF kernel with probabilities.
- **Naïve Bayes Classifier (NBC) –** using the Bernoulli variant;
- **Random Forest Regressor (RFR)** – using 1446 estimators with Poisson criterion.

The most promising representations found in the Feature Selection were tested, more specifically: (I) Team binary encoding with team rank using match date as weight; (II) Player binary encoding with team rank and match date as weight; (III) RADR 19-game recency-weighted average encoding with team rank and date as weight. It was also tested a heuristic that used the team rank to calculate the probability of a team winning, following the formula below in by equation (1). As an example, if the first team was currently rank 1 and the second team was rank 3, the former would have 75% chance of winning, whereas the latter 25%.

$$Pred_{Team1} = 1 - \frac{Rank_{Team1}}{Rank_{Team1} + Rank_{Team2}} \quad (1)$$

The results are shown in Table 19. The team rank heuristic performed surprisingly well. The undisputed best performant machine learning technique for team binary encoding was LR, with 63.66% accuracy, 0.221 BS, 0.632 LL and a ECE of 2.17%.

As for player encodings, the binary representation performed the best with SVM, having an accuracy of 64.86%, 0.223 BS, 0.636 LL and an ECE of 3.03%. On the other hand, RADR encodings performed the best when using LR, having the lowest overall BS (0.213), LL (0.614) and ECE (2.10%). And as seen in Table 20, it had by far the best distribution, with accuracies very close to the bins' expected values.

It is also noticeable that SVM did also have reasonable distributions in every single representation. All in all, RADR encodings with LR is the most promising choice.

Table 19 – Comparison between machine learning techniques the most promising representations

Feature set	Technique	Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
					ECE	Empty
Team Binary Encoding + Rank + Date as Weight (I)	NBC	61.06	0.234	0.661	3.07	2
	LR	63.66	0.221	0.632	2.17	**0**
	SVM	62.52	0.230	0.653	2.51	6
	GBM	62.30	0.244	0.709	10.05	**0**
	RFR	62.08	0.243	0.707	10.92	**0**
Player Binary Encoding + Rank + Date as Weight (II)	NBC	63.82	0.273	0.952	19.45	**0**
	LR	64.06	0.220	0.638	6.97	**0**
	SVM	64.86	0.223	0.636	3.03	2
	GBM	61.60	0.253	0.746	14.71	**0**
	RFR	64.58	0.230	0.678	8.51	**0**
RADR 19-game average (III)	NBC	63.84	0.273	0.952	19.44	**0**
	LR	**66.66**	**0.213**	**0.614**	**2.10**	**0**
	SVM	64.68	0.220	0.629	2.52	2
	GBM	60.36	0.260	0.756	14.70	**0**
	RFR	61.00	0.240	0.682	7.50	**0**
Team Rank Heuristic		65.80	0.224	0.650	6.90	**0**

Table 20 – Average accuracy of the most promising match history representations and ML techniques per predicted probability bin and ECE

Bins (%)	Team Encoding + Rank + Date					Player Encoding + Rank + Date					RADR + Rank + Date Encodings					Team Rank
	NBC	SVM	LR	GBM	RFR	NBC	SVM	LR	GBM	RFR	NBC	SVM	LR	GBM	RFR	
[0, 5[	-	-	14	20.81	20.92	27.23	-	15.94	23.97	24.41	27.30	-	10.00	24.71	11.11	14.74
[5, 10[	14.29	-	17.14	27.05	26.53	35.26	0.00	20.09	21.28	32.97	35.58	0.00	20.00	28.79	19.51	19.90
[10, 15[	23.17	-	9.30	34.56	35.61	41.44	0.00	17.33	30.00	33.02	41.18	26.67	10.28	35.66	33.11	25.59
[15, 20[	20.83	-	24.00	38.54	35.06	38.20	17.74	21.59	37.98	37.21	37.91	17.74	21.11	35.98	31.72	29.45
[20, 25[	30.94	18.00	26.80	28.51	36.19	42.49	16.57	37.17	39.41	44.21	42.41	22.50	21.14	38.83	32.81	28.46
[25, 30[	36.95	25.81	25.98	36.96	44.08	37.41	32.36	30.65	39.30	44.20	38.19	28.46	26.58	44.91	40.05	36.02
[30, 35[	35.13	35.56	31.43	43.94	41.56	39.57	33.41	38.46	45.14	44.18	39.29	37.50	33.33	39.29	38.15	37.85
[35, 40[	39.90	33.18	36.47	39.75	44.76	45.81	34.54	41.35	49.63	49.33	45.22	36.19	39.75	50.64	45.05	37.19
[40, 45[	40.77	44.78	44.60	44.60	50.00	41.33	42.79	52.32	44.65	50.17	41.06	43.75	40.26	51.39	45.91	44.69
[45, 50[	47.62	47.66	50.39	49.67	54.55	46.32	46.43	51.78	49.24	52.22	46.10	49.54	46.10	45.56	49.03	44.83
[50, 55[	53.26	51.73	50.58	53.87	55.38	53.14	52.98	47.68	51.09	58.87	53.25	49.54	55.21	50.66	54.11	55.17
[55, 60[	59.79	58.78	56.09	58.72	59.04	53.15	58.99	48.48	54.18	57.25	53.15	60.42	59.42	50.00	58.43	54.32
[60, 65[	59.42	66.12	63.17	58.24	63.56	56.46	64.98	56.37	50.00	59.01	57.62	64.76	60.64	56.90	56.10	61.68
[65, 70[	67.25	64.72	69.41	56.49	64.44	62.50	68.20	63.86	51.74	66.51	62.57	60.81	65.91	55.04	57.45	62.73
[70, 75[	63.95	77.04	73.23	56.14	63.93	54.04	67.26	66.67	61.48	64.98	53.80	74.69	74.31	58.24	64.44	64.64
[75, 80[	65.77	83.78	72.83	66.38	68.04	62.57	83.97	66.67	60.91	68.72	62.13	75.52	78.04	63.26	69.33	69.97
[80, 85[	79.59	-	80.65	71.03	67.07	56.91	80.33	76.82	62.91	69.31	57.14	81.67	78.87	63.91	69.23	70.03
[85, 90[	78.82	-	90.14	63.55	72.67	59.64	100	83.77	75.50	72.33	59.91	76.92	91.75	61.57	77.78	73.83
[90, 95[	81.82	-	82.35	73.33	83.87	65.14	100	79.70	74.79	76.32	64.83	100	77.78	69.87	72.48	80.00
[95, 100]	-	-	80.00	80.30	75.53	72.14	-	83.33	77.70	72.09	72.10	-	100	75.27	79.69	85.15
ECE	3.07	2.51	2.17	10.05	10.92	19.45	3.03	6.97	14.71	8.51	19.44	2.52	2.1	14.7	7.5	6.9

6.6 Chapter Conclusions

In this chapter patterns in the data were explored and found that most teams play less than 10 games with the same 5-player lineup. Also, teams with higher peaks tend to make less changes, and that fact is even more accentuated within teams that have consistently higher ratings. It was also concluded that a single player can affect a team's win rate on multiple maps, thus considering which players are in a team is of utmost importance.

Additionally, it was explored how to balance the dataset to contain both sides of a matchup, as well as creating a dataset structure that would allow for teams to be represented by its id or by its players, and represent whole matches and single games as well, interchangeably, without having to change the model's code.

For outcome prediction, simply using a heuristic with the team rank proved to be surprisingly accurate. Furthermore, whole-match predictions proved more fruitful than game-wise predictions, although predicting the games can prove useful for mid-match bets. In the same vein, team encodings were less fruitful than player binary encodings which were also worse than player's performance metrics encodings. Between player metrics, 19-game recency-weighted average proved to be the best fit. Additionally, every tested representation benefited from the addition of team rank and the usage of the date as sample weight.

One area that could prove worthwhile is exploring feature interaction between different player performance metrics and aggregation types through SVMs. Nevertheless, that option will not be explored within the context of this document.

Finally, it was explored a multi-input-layer NN architecture, although without any promising results, as well as tested multiple unsupervised learning models, namely, NBC, RFR, GBM, SVM and LR. From these SVM and LR were the clear standouts. While the former was consistently good, the latter was faster to train and had the best overall results with RADR 19-game recency-weighted average. This model was the best by every single metric, making it the clear best choice. It had an accuracy of 66.66%, BS of 0.213, LL of 0.614 and a ECE of 2.10% with 0 empty bins.

7 Prediction through bookmaker odds

The success of a bookmaker is closely related to the quality of the odds. Bookmakers must ride the fine line between profitability and quality of offer. In other words, their odds cannot be so high that they lose their margin of profit, nor so low that they become unattractive to punters.

In this chapter, first, the method for purging and merging the initial OddsPedia dataset with HTLV's dataset is detailed. Next, the new dataset is explored to find patterns that might hint what approach to follow. Afterwards multiple approaches for representing the data are analysed to select the best. Finally, multiple machine learning techniques are analysed and the one that better fits the data is chosen.

7.1 Data cleanup

The raw data extracted from OddsPedia required a lot of work to be useable. Some of the issues found were: duplicated matches with different odds; matches with the wrong team(s) associated; wrong match start dates; wrong/missing outcome of the match; and matches that were not recorded on the website. All of these meant that the information presented on the OddsPedia database could not be trusted at face value. There is no surefire way to validate that the odds data itself was trustable either.

This difficulted the task of crossing the odds information with HLTV's match data, which would be needed for the betting simulations. To overcome it, a semi-automatic approach was conducted. It was created a script that ran through each match of HLTV, and crossed information with OddPedia's matches that took place in a 12h time window. If the names of the teams coincided, it would automatically link the two matches, else, the user was prompted to manually review the matches. The user was also allowed to add the team name as an alias, which would automatically link the teams in the future. This process required going through both HLTV and OddsPedia websites to get the context of the match and cross the information. In the end, this process also allowed for some of the odds to be validated. As an example, if it was being analysed a match between two teams with similar HLTV ranking, it was very likely that the odds of both teams were between 1.7 and 2.0. This way, it was linked 6225 out of the 7913 matches.

7.2 Data exploration

On paper, the bookmaker odds data is very straightforward, it merely contains the odds of each bookmaker, per match. Regardless, from that data, bookmakers' accuracies, as well as their bias can be extracted. Some bookmakers consistently bet below the markets' median, to ensure profit, whereas others will risk providing higher odds to a underdog as a possible marketing strategy. Bookmakers' presence, i.e. how often they provide odds is another feature that can be extracted. In this chapter, all of these are explored to understand if there could be any patterns between the bookmaker's behaviour and the match outcome.

7.2.1 Bookmakers' statistics

The first thing analysed was the number of matches per bookmaker. There are a total of 73 unique bookmakers, but even though OddsPedia has a huge database of past odds, it is incomplete and is missing odds. Naturally, some bookmakers are missing more often. Nevertheless, as seen in Figure 48, established bookmakers such as Betway and GGBet have over 4000 matches, which is nearly 2/3 of the dataset. Additionally, even the less popular bookmakers are present in at least 50 matches in the dataset. The figure also discriminates the accuracy of each bookmaker. The vast majority of bookmakers have an accuracy above 60% with the top bookmakers reaching 66%. Although high, it is not as good of an accuracy as the theoretical maximum of 70%, proposed in [111].

Next, patterns between top 20 bookmaker were analysed, more specifically, how they deviated from the median bookmakers' odd for the match. As reflected in Figure 49, every bookmaker, on average, puts the odds of the favourite team below the median and most prefer to offer higher odds on the underdog. This is more evident on GGBet, which has the lowest average odds for the favourite and the highest odds for the underdogs. Nonetheless, there are bookmakers such as Betway and Vbet who consistently offer below market-median odds on the underdog, maximizing their overall return. From these behaviours, it wouldn't be surprising if using multi-label odd encoding of each bookmaker uncovered patterns between strategies of each bookmaker and the game outcome.

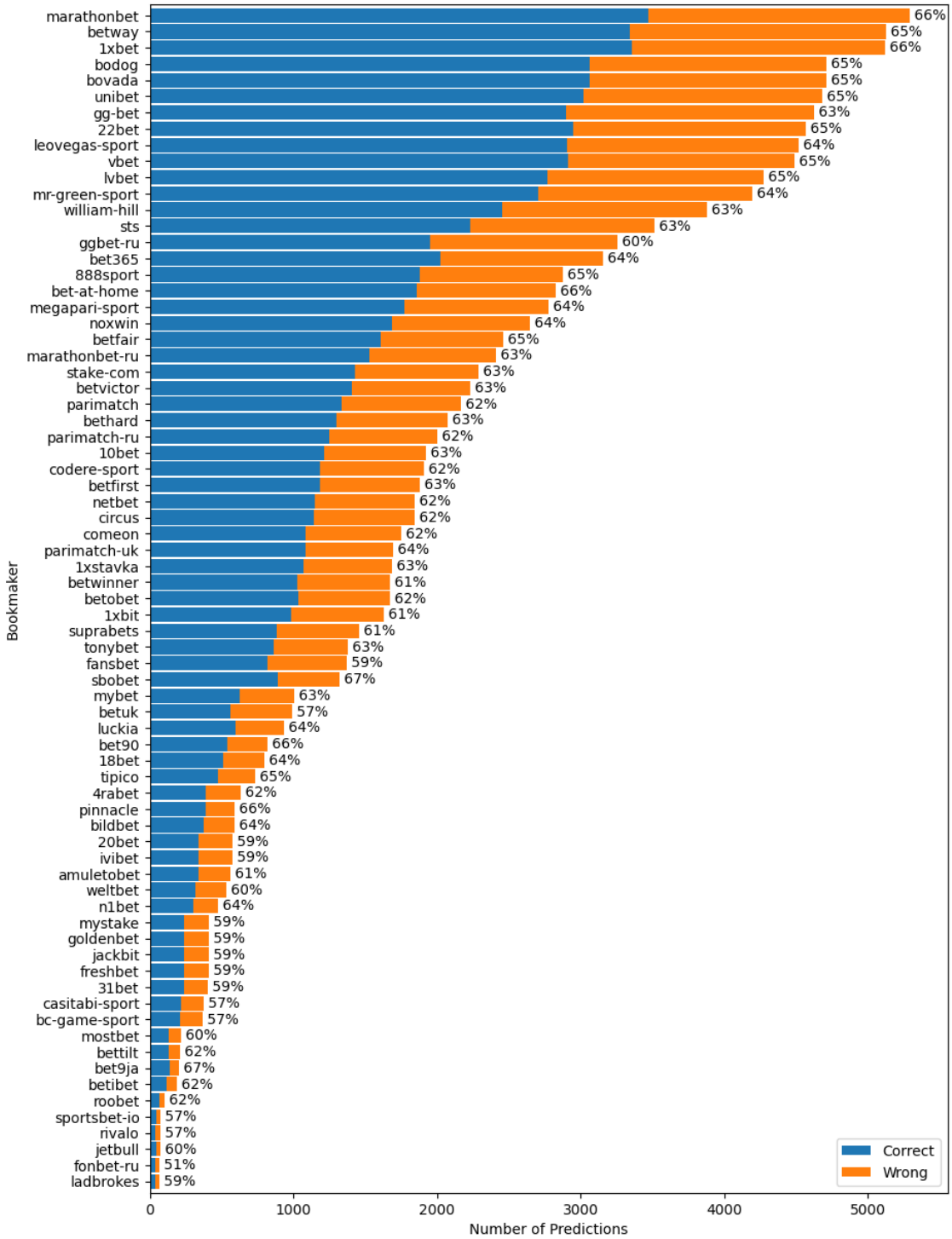


Figure 48 – Number of guesses and accuracy of each bookmaker

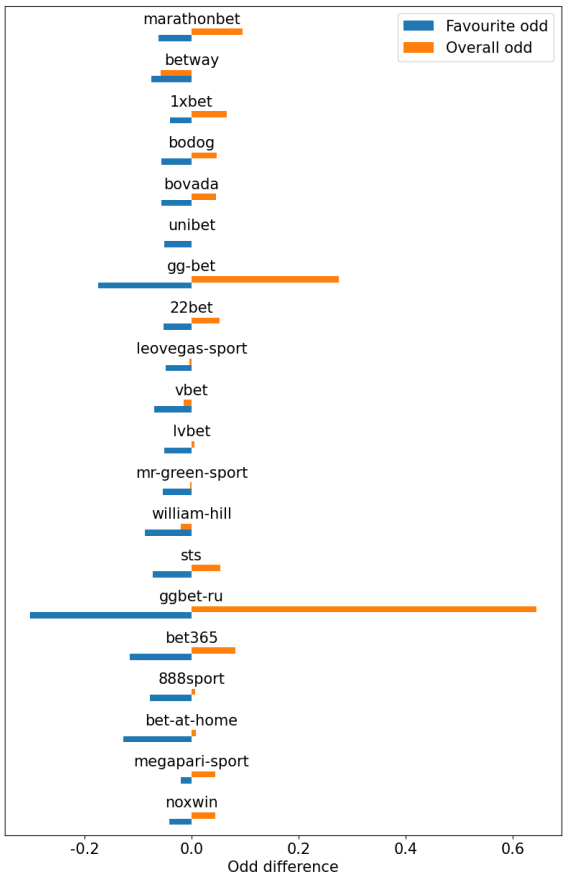


Figure 49 – Favourite and overall odd difference versus median odd per bookmaker

7.2.2 Bookmaker match distribution

At last, it was analysed how many matches were available depending on the number of bookmakers that provided odds for that match. As seen in Figure 50, the median number of bookmakers per match is 19 and represent nearly 3500 matches, or 56% of the dataset, but for a lower threshold, such as 10 bookmakers per match, the number of matches available increases to 5500, which is 88% of the whole dataset.

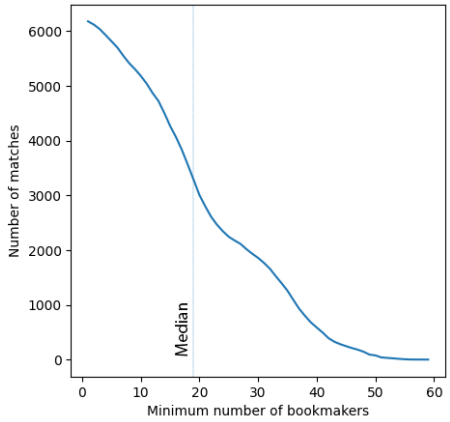


Figure 50 – Distribution of matches per minimum number of bookmakers with odds

7.3 Curating the dataset

The curation process is very similar to the one described in the section 6.2 of Prediction through match history. This time the dataset would be balanced by itself, the odds could be ordered in a way that ensured that the first team was always the favourite and second team the underdog. However, it might be necessary to merge this model with the previous one, and as such, for consistency, each line will be duplicated, ensuring the matchup is capture both ways – "Favourite vs Underdog" and "Underdog vs Favourite".

Furthermore, for this model, three different representations were tested: (I) Multi-label Odd Encoding; (II) Odds Average; and (III) Odds Median. The same format used previously was used to facilitate the multi-label encoding – one column containing the names of the bookmakers, and the others contains the odds for each team, as seen in Table 21.

Table 21 – Example lines of bookmaker

bookmakers	t1Odds	t1Odds	Won
[betway,1xbet, bet365]	[1.23, 1.24, 1.21]	[4.90, 4.95, 5.00]	True
[betway,1xbet, bet365]	[4.90, 4.95, 5.00]	[1.23, 1.24, 1.21]	False
[betway, ggbet]	[1.76, 1.90]	[1.85, 1.85]	False
[betway, ggbet]	[1.85, 1.85]	[1.76, 1.90]	True

7.4 Pre-Processing

After curating the dataset, the first step is handling missing values. The matches dataset goes up to July 2023, however, the odds were only retrieved up to March 2023, which means that information is missing in 2300 matches, which represents 13% of the dataset and will be discarded. Since there are only 120 unique bookmakers, and they all have at least 50 matches, no bookmakers will be discarded.

Next, the odds are normalized by applying the equation (2) described in section 2.1.1 - Odds, which transforms the odd into the implied probability, which is normalized by default. Finally, it is computed the median and average odd of each team per match as well as encoded the bookmakers' odds for each team, as show in Table 22.

Table 22 – Custom Multi-label encoding example

Original			Encoded			
bookmakers	t1Odds	T2Odds	Betway_t1	betway_t2	…	ggbet_t2
[betway, ggbet]	[0.56, 0.53]	[0.54, 0.54]	0.56	0.54	…	0.54
[betway, ggbet]	[0.54, 0.54]	[0.56, 0.53]	0.54	0.56	…	0.53

7.4.1 Concept drift

To understand if the models were resilient to concept drift, it was compared their EV betting profit using bookmaker encodings and bookmakers' odds median. As reflected in Figure 51, the former (a) is surprisingly very sensitive to new information, which might hint that bookmakers are constantly switching strategies. On the other hand, using the odds median (b) is very stable,

performing similarly months after creation. Therefore, the test will be run retraining the model daily, as to give the fairest evaluation of the

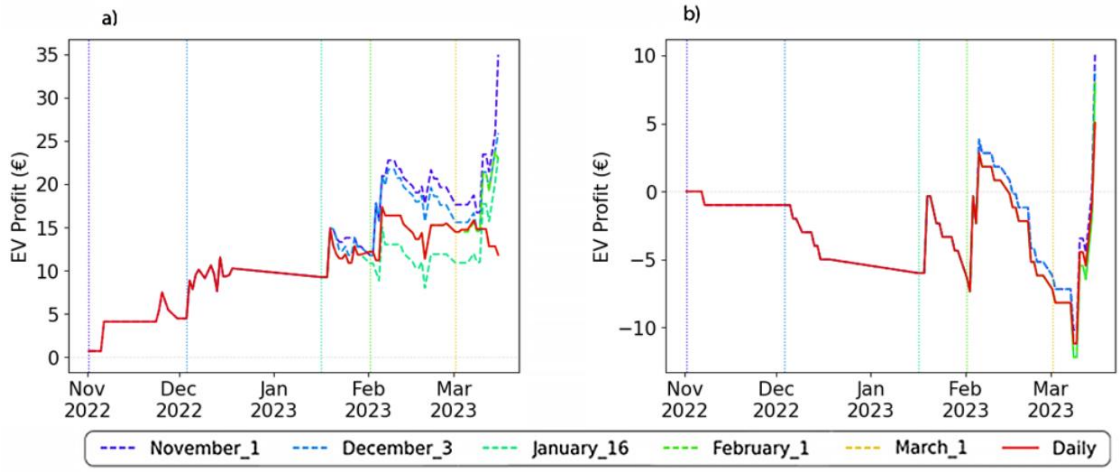


Figure 51 – Comparison of EV Betting simulation using bookmaker encodings (a) and bookmakers' median odd (b) from November 2022 to March 2023

7.5 Model Selection

Due to the nature of bookmakers' business model, using their odds as a baseline indicator of the outcome of a match is hugely beneficial. Therefore, the main goal of this chapter is to build a model that can aggregate the odds of the various bookmakers and hopefully extract patterns from the data, that can outperform simple heuristics. That is the reason there is no dedicated Feature Selection section in this chapter and instead, the main representations are directly compared against two heuristic: Odds Median and Odds Average.

The Odds Median and Odds Average heuristics calculate the bookmakers' median and average odd, respectively, and obtain the implicit probability of said odd using the equation (2) described in section 2.1.1 - Odds. However, due to bookmakers' profit margins, these raw probabilities will seldomly add to 1, thus they are normalized and used as actual predictions, using the formula presented in equation (1) below.

$$Pred_{Team1} = \frac{\frac{1}{Odd_{Team1}}}{\frac{1}{Odd_{Team1}} + \frac{1}{Odd_{Team2}}} \leftrightarrow \frac{Probabity_{Team1}}{Probabity_{Team1} + Probabity_{Team2}} \quad (1)$$

In Bookmakers' statistics, it was already observed that the accuracy of the bookmakers does not vary widely. Hence there is little merit to the extraction of features such as bookmaker accuracy. Therefore, it will be tested 3 data representations: (I) Bookmakers' Odds Encodings; (II) Bookmakers' Odds Median; and (III) Bookmakers' Odds Average. It should be noted that the latter 2 are a direct comparison to the heuristics and thus use very similar data: it is used as input the raw implied probabilities, without normalization.

This way, each representation will be tested using different machine learning techniques. For this chapter's final model, it was tested the same algorithms as the Prediction through match history, with the caveat that the tree-based models used 2 estimators per bookmaker for a total of 120 and that the NBC used was the Gaussian instead of Bernoulli's. Therefore, as displayed in Table 23, the following machine learning techniques and parameters were tested:

- **Logistic Regression (LR) –** 100 000 maximum iterations;
- **Gradient Boosting (GB) –** a max depth of 4, a learning rate of 0.1 and 120 estimators;
- **Support Vector Machine (SVM) –** RBF kernel with probabilities;
- **Naïve Bayes Classifier (NBC) –** using the Gaussian variant;
- **Random Forest Regression (RFR)** – using 120 estimators with Poisson criterion.

From Table 23, it is observed that overall, every single model managed an accuracy above 62%, and most managed 65%, which was nearly the maximum accuracy between bookmakers. LR consistently had the highest accuracy while maintaining very low Brier score and log loss. NBC also consistently had high accuracies, but performed substantially worse in all calibration metrics, mainly in the bin ECE.

For the bookmakers encoding, LR, GBM and RFR performed very similarly, even though, LR had the highest accuracy by over 1% and the best overall ECE, 2.12%. GBM on the other hand had slightly better Brier score and log loss.

Between Odds Average and Median, the Averages performed overall better, with LR and GBM being the most promising techniques. LR had higher accuracy (66&) as well as the lowest overall BS (0.210) and LL (0.607), while the GBM had a lower ECE. The also heuristics performed surprisingly well, with Odds Average sharing the best BS and LL and having an accuracy of 65.73%. The Odds Median Heuristic was slightly behind on the mentioned metrics but had a slightly better ECE (2.85%).

To have a better understanding the calibration of the models, the average accuracy of each probability bin is displayed in Table 24. It can be observed that for both Bookmakers' Odds Median and Average, the techniques SVM and LR had 12 and 4 empty bins, respectively, meaning that these models only predicted teams to have a chance of winning within the intervals [30, 70[and [10-90[. In contrast, GBM filled all the bins, and for the Bookmakers' Odds Average, the accuracies very close to the bin's expectations, having the lowest ECE.

Even though Bookmakers Odds Encodings with both LR, RFR and GBM also had very good bin accuracies, the former had 2 empty bins and the remainder were still worse than the Bookmakers' Odds Average by every metric. Using Odds Average also has the advantage of being bookmaker independent, which should make it more versatile. The heuristics could fulfil the same purpose with similar results, nevertheless, the Bookmakers' Odds Average using GBM was more accurate when filling the bins in the interval [10,85[.

Table 23 – Comparison between different machine learning techniques

Feature set	Technique	Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
					ECE (%)	Empty
Bookmakers Odds Encoding	NBC	65.68	0.325	6.574	31.88	0
	LR	65.88	0.215	0.619	**2.12**	2
	SVM	**66.06**	0.215	0.621	3.52	8
	GBM	64.64	0.214	0.616	3.04	0
	RFR	64.60	0.218	0.633	3.27	0
Bookmakers Odds Median	NBC	65.68	0.224	0.656	11.29	0
	LR	65.72	0.212	0.612	2.72	4
	SVM	65.76	0.220	0.632	3.85	12
	GBM	65.20	0.212	0.613	2.60	0
	RFR	62.80	0.238	0.713	8.70	0
Bookmakers Odds Average	NBC	66.00	0.226	0.663	12.15	0
	LR	66.00	**0.210**	**0.607**	2.90	4
	SVM	65.84	0.220	0.631	3.48	12
	GBM	65.06	0.212	0.612	2.53	0
	RFR	62.64	0.236	0.720	9.46	0
Odds Median Heuristic		65.49	0.212	0.613	2.43	0
Odds Average Heuristic		65.73	**0.210**	**0.607**	2.85	0

Table 24 – Average accuracy of each Odds representation and ML technique per predicted probability bin and ECE

Bin (%)	Bookmakers Odd Encoding					Bookmakers Odd Median					Bookmakers Odd Average					Heuristics	
	NBC	SVM	LR	GBM	RFR	NBC	SVM	LR	GBM	RFR	NBC	SVM	LR	GBM	RFR	Median	Average
[0, 5[	32.69	-	-	9.09	9.09	15.58	-	-	16.67	12.70	12.71	-	-	8.64	10.87	0.00	0.00
[5, 10[	37.74	-	15.79	13.64	8.70	18.21	-	-	13.33	21.67	20.79	-	-	11.54	24.26	13.33	4.17
[10, 15[	44.90	-	17.14	11.72	16.54	26.33	-	28.57	17.78	25.13	29.93	-	10.23	15.83	24.30	18.80	16.36
[15, 20[	46.88	-	17.95	17.83	23.16	33.00	-	14.29	13.71	28.71	31.54	-	13.41	18.53	35.27	12.63	12.64
[20, 25[	56.25	19.74	16.67	27.13	26.51	32.94	-	17.31	21.36	38.99	42.28	-	22.04	23.23	36.74	20.99	20.24
[25, 30[	69.57	20.94	26.09	26.30	31.25	43.51	-	25.81	28.02	37.97	40.63	-	28.87	27.90	39.73	27.44	29.12
[30, 35[	40.00	28.12	28.69	31.39	36.47	44.21	30.21	33.49	29.12	41.45	48.17	30.37	31.62	31.61	43.73	32.55	29.43
[35, 40[	42.86	36.47	37.22	43.96	43.61	47.51	45.48	40.52	35.25	41.94	43.65	43.24	42.96	41.64	42.58	40.45	42.16
[40, 45[	71.43	44.64	42.80	39.78	49.10	51.09	51.79	45.34	45.03	47.04	50.79	49.76	46.44	46.72	48.46	45.37	44.52
[45, 50[	69.57	49.89	46.76	50.85	45.30	50.86	49.32	50.55	50.22	49.44	48.70	48.19	50.00	47.99	50.49	50.27	48.77
[50, 55[	47.06	51.93	54.37	50.58	54.49	48.41	49.67	50.87	52.48	54.38	49.68	48.11	51.99	48.57	51.54	49.48	50.99
[55, 60[	50.00	56.29	58.73	56.04	59.43	50.48	51.58	53.94	52.33	58.84	52.53	56.85	54.39	57.11	57.19	54.75	55.26
[60, 65[	42.86	63.92	66.93	61.41	61.24	54.03	57.85	61.84	68.88	56.90	51.72	53.31	57.53	61.26	56.79	59.16	57.84
[65, 70[	44.44	73.94	69.94	75.44	68.16	55.20	70.84	67.85	70.45	63.70	55.41	71.24	68.29	73.52	58.39	67.63	70.57
[70, 75[	62.07	78.72	73.31	67.16	72.73	54.03	-	75.63	72.68	60.61	57.85	-	72.01	70.29	60.09	72.53	70.88
[75, 80[	57.14	81.61	81.73	71.74	73.21	67.53	-	84.59	78.67	66.67	60.25	-	81.00	78.08	60.70	78.95	79.76
[80, 85[	48.15	-	84.11	85.21	80.88	68.86	-	84.66	85.31	66.14	69.72	-	84.33	83.19	69.76	86.91	87.36
[85, 90[	51.28	-	81.94	85.35	84.76	74.50	-	50.00	86.67	72.32	71.57	-	95.52	82.57	75.42	81.20	83.64
[90, 95[	60.00	-	81.25	90.67	87.50	83.15	-	-	85.71	78.68	79.31	-	-	88.00	77.71	86.67	95.83
[95, 100]	66.90	-	-	90.48	89.47	83.82	-	-	75.00	73.95	87.55	-	-	91.89	82.65	100	100
ECE	31.88	3.52	2.12	3.04	3.27	11.29	3.85	2.72	2.6	8.7	12.15	3.48	2.9	2.53	9.46	2.43	2.85

7.6 Chapter Conclusions

In this chapter, the initial dataset from OddsPedia undergoes thorough cleaning to fix issues such as duplicated matches, incorrect team associations and missing information. The exploration of the new dataset reveals insights into the bookmakers' accuracies, strategies and presence. It is observed that most bookmakers have accuracies above 66% and that more established bookmakers are more present in the dataset. It was also noted that within the top 20 bookmakers, the majority, on average, offered odds below the median for the favourite and above the median for the underdog.

The dataset is then further curated by handling missing data and translating the odds into their implied probabilities and duplicating each match, to ensure both sides of the matchup are captured and consistency with other chapters' datasets.

Three data representations are explored, namely multi label odd encoding, odds average and odds median. Concept drift is assessed to understand whether the models are robust to changing trends. Odds median proves to be more stable when compared to bookmaker encodings.

Due to the low number of representations, the chapter does not include a dedicated feature selection section and instead the three representations are explored using various machine learning techniques and compared to two simplistic heuristics based on the odds' average and odds' median. Bookmaker odds encoding is promising with either LR or GBM. Nevertheless, the best approach is bookmakers' odds average with GBM with 65% accuracy, 0.212 BS, 0.612 LL and 2.43% ECE with 0 empty bins.

8 Sentiment Analysis

In this chapter it is firstly explored the data obtained through scrapping to find patterns that might hint what approach to follow. Afterwards, it is analysed multiple approaches for representing the data and selected the best features. Finally, experimental findings are reported, and the final model is selected.

8.1 Data exploration

Delving into data exploration is crucial because it highlights shortcomings and relationships within the dataset, that can be used to curate it. Furthermore, patterns within the data that are also revealed, which can inspire different data representations. It is first accessed how to filter the 2 400 000 comments to selecting only the most relevant. Additionally, it is explored the amount of data available considering certain thresholds, namely, minimum number of comments and rank of the best team in the match.

8.1.1 Identifying relevant comments

Even though the extracted dataset contains over 2.4 million comments, most of them are not fit for the task at hand. On HLTV, the match page (or thread) is created before knowing which teams will play it. As an example, when a tournament is created, a match thread is created for every match, including the finals, even though the finalists are only decided at the very end of the tournament. Therefore, it is possible to comment on a match without having any context on the participants. Likewise, the match thread does not close, meaning that it is possible to comment during and after the match. Both scenarios' comments are pointless for pre-game sentiment analysis, therefore, each comment was filtered according to 3 rules:

- **Each game is considered to take 1 hour –** matches that are best of 1, 3 and 5 are considered to last 1, 3 and 5 hours, respectively;
- **Minimum date –** the comment must have been made after the conclusion of the previous game of each team within the same event (start date + expected duration)
- **Maximum date –** the comment must have been made before the start time of the game;

This way, it is possible to filter out irrelevant comments. As shown in Figure 52, 117 500 comments (5%) were made before the teams were locked to play the match. The valid

comments represent 25% of the total comments (613 000) and were made between the moment the teams were locked and the start time of the match. The vast majority, over 1 700 000 comments (75%) were made after the match had started. It is possible that these rules eliminate comments that would be valid, but considering the sheer volume of comments, it is preferable to abdicate of some useful data than to use data where over 75% of the of it was either irrelevant or a leak from the outcome we are trying to predict.

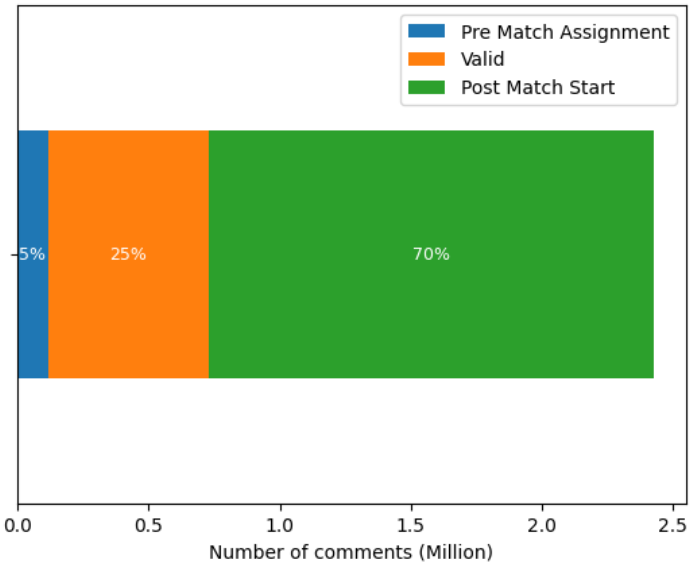


Figure 52 – Distribution of comments over the lifetime of a match thread

In the match page, it is possible to reply directly to the thread or to other comments. Figure 53 shows the distribution of these 2 types of comments within the valid comments and it is observed that over 350 000, representing 57% of the comments are made directly to the thread.

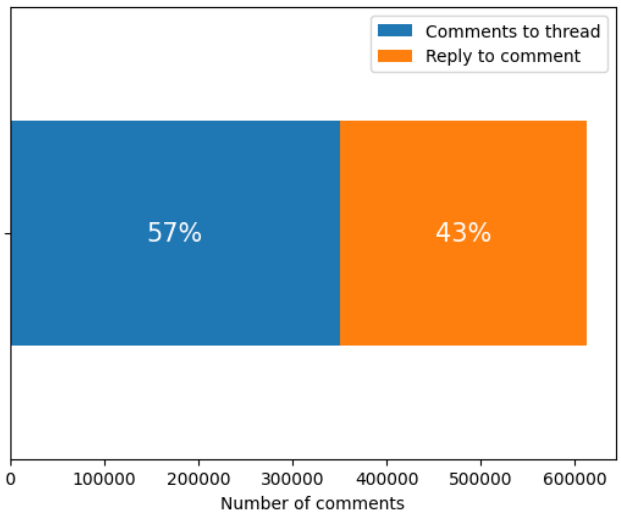


Figure 53 – Distribution of replies to the match thread and other comments

Furthermore, replies to comments are complex. Examples of replies are shown in Table 25. For context, it is a match between team ***Vitality*** and team ***NAVI***. ***ZywOo*** is the best player of team ***Vitality*** while ***Zonic*** is their coach. The replies are very varied, ranging from a simple "+1" as an agreement sign to irony displayed in the last comment that implies ***Zonic*** being out has no

impact o the team, because he is not a player ("*does not hold the mouse*s"). Extracting these meanings requires the context of the chain of comments it is replying to, which could confuse less powerful models and balloon the costs of pay-to-use ones.

Table 25 – Examples of replies to comments

Original Comment	Reply	Reply Meaning
ez 4 NAVI	+1	Clearly agrees with previous comment
i have never been so sure of the winner 100% ez win for Vitality, there's absolutely no chance for NaVi to beat them ABSOLUTELY 0 % CHANCE FOR NaVI ZywOo numba 1!	you wish	Clearly disagrees with previous comment
Vitality is Zonicless. Best opportunity for NAVI to win this.	Blud thinks zonic hold the mouses xD	Disagrees through irony

In natural language processing, providing context comes with its challenges. In this scenario, the context needed to interpret a reply is in the message being replied to, however, it is common for a reply to need the context of the replies above it, which would introduce a lot of variance when training the model.

Therefore, there are three possible approaches, of which the last is the most feasible given the scope and resources of this project:

- **Create a model capable of context –** obtain more comments with replies, label them and extensively fine tune a weaker model**;**
- **Create a model for reply classification –** classify if the reply agrees, disagrees or is neutral and combine the output with the original comment's classification. As an example, in the first line of the table above, the original comment's classification would be "NAVI", the reply's initial classification would be "Agree" which would then be mapped to "NAVI";
- **Use only comments that reply directly to the thread –** the remainder of the dataset still has over 350 000 unlabelled entries, of which a small portion will be manually classified, meaning the dataset is still good for the purpose of this study.

8.1.2 Frequency of comments

Once the dataset was reduced to include only the comments considered valid above, the patterns of the commenters were studied. First, it was explored how many matches are available for different thresholds of minimum number of comments. As seen in Figure 54, the number of matches (a) starts decreasing drastically as the minimum number of comments increases: while there are 7000 matches with over 10 comments, there are only 4500 over 25. The median number of comments per match is 36, which corresponds to nearly 4000 matches, filtering out nearly 45% of the total matches. The frequency of which the same commenter

contributes to discussions was also analysed (b). The median commenter posted 47 comments to the thread and 33 replies to other comments. A pattern found is that the people that comment less, are more prone to replying to other comments than to the thread. Furthermore, there are few dedicated contributors with over 100 comments to their name, and one with over 3000 comments.

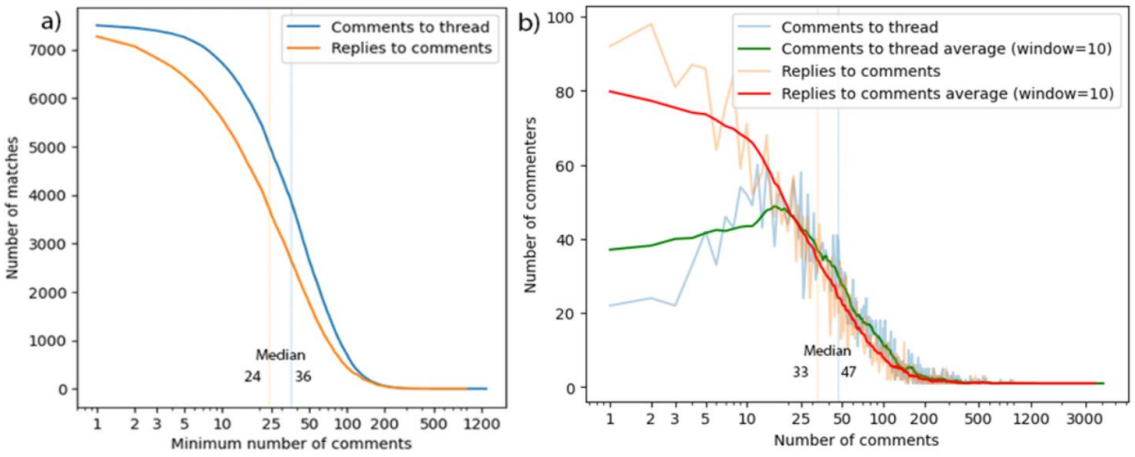


Figure 54 – Distribution of the number of matches (a) and commenters (b)

Whether the skill of the teams participating in a match impacted the number of comments in the thread was also studied. In Figure 55 a trend emerged where the higher the ranking of the best team in the match, the higher the mean number of comments, which means fans are more vocal about higher quality games. When the match involves a top 10 team, the median number of comments is 40, while the 25th percentile is 20. In other word 75% of matches involving a top 10 team contain at least 20 comments, which is substantial for prediction. In contrast, if the best team is below top 20, the number of comments per match is halved.

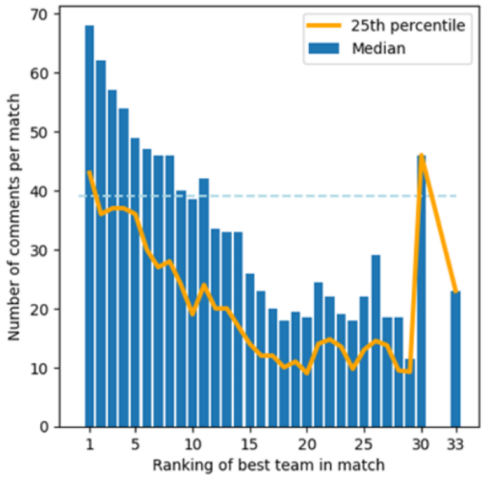


Figure 55 – Comments per match by ranking of the best team

8.2 Classification Using Large Language Models

Overall, working with LLMs revolves around 3 different approaches: fine-tuning, embeddings and prompt engineering. Fine-tuning is not a concept unique to LLMs, it involves providing the model new information: examples of how to answer in each instance. However, this is the very way the model was trained, on millions of examples, meaning fine-tuning is barely noticeable unless the amount of new information is also substantial, making it an expensive and time-consuming procedure.

In addition, embeddings are also not a new concept, with works such as Word2Vec and Doc2Vec being published in 2013 and 2014, respectively [212], [213]. Embeddings consist in mapping a word or a string of words into a n-dimensional space, where words with similar meanings would be closer. The multi-dimensional canvas allows for different relationships to be captured: the word “Man” should be closer to “Men” than “Woman” in a hypothetical “Gender” axis, but closer to “Woman” than “Men” on the “Plurality” axis.

Prompt Engineering, however, is a fairly new concept that emerged with generative AI and consists in communicating effectively with a generative AI model. In other words, it is the art of getting the model to output the right information [214], [215]. In OpenAI’s documentation, there’s a best practices section that tackles how to get the most out of their models [216], out of which the following should be highlighted:

- Write clear instructions;
- Test changes systematically;
- Include details in the prompt to get a more relevant answer;
- Ask the model to adopt a persona;
- Use delimiters to clearly indicate distinct parts of the input, such as quotes.

8.2.1 Prompt Engineering

Following OpenAI’s best practices tips, the prompt was progressively iterated and tested on a smaller labelled dataset with 1088 comments. The goal of this model is to guess which team the comment is cheering or is favouring to win. Nevertheless, besides the name of each team, there is a third class – “Inconclusive” – that should be used when the comment is either unclear or irrelevant to the task at hand.

The first version of the prompt is transcribed below. It was also tested how adding more context about the players, namely their full name and nationality impacted the prediction. This emerged because many commentors referred to the players by their name or nationality, which the model would not know without the context. It is also worth noting that the information about the second team is below the first one, but here is represented side by side to improve visibility.

> *"You are a sentiment analysis classifier for a comment about a game that is about to occur between two teams. If the comment mentions a scoreline, whoever has the highest score is better. Answer with just the name of the team or "Inconclusive", without explanation nor conversation. The game is between these two teams and players:*
>
> | | |
> |---|---|
> | *Natus Vincere:* | *Vitality:* |
> | *Aleksib* | *apEx* |
> | *B1t* | *flameZ* |
> | *iM* | *Magisk* |
> | *jL* | *Spinx* |
> | *S1mple* | *ZywOo"* |

Following this version, a second one was analysed where the team names were mentioned earlier in the prompt, it was divided into two paragraphs and had clearer language, as highlighted below:

> *"You are a sentiment analysis classifier for a comment about a game that is about to occur between 'Natus Vincere' and 'Vitality'. If the comment mentions a scoreline, whoever has the highest score* ***is the favourite****.*
>
> *Answer with just the name of the team or "Inconclusive", without explanation nor conversation.* ***Below is context with information about the players of each team****:*
>
> | | |
> |---|---|
> | *Natus Vincere:* | *Vitality:* |
> | *Aleksib* | *apEx* |
> | *B1t* | *flameZ* |
> | *iM* | *Magisk* |
> | *jL* | *Spinx* |
> | *S1mple* | *ZywOo"* |

The results of this experiment can be observed in Table 26. To transform these classifications into predictions, the ones that predict a team to win are the most relevant. Therefore, the table separates the overall accuracy from the accuracy on comments that were not labelled "Inconclusive". It is clear the second version of the prompt is an improvement and reflects the importance to tweaking and testing the prompt according to the OpenAI's directives.

Table 26 – Comparison between different prompts on GPT-3.5-turbo

Prompt version	Conclusive comments only		All comments	
	Accuracy (%)	Quantity	Accuracy (%)	Quantity
Base version 1	88.63	695	83.46	1088
Base version 2	88.11	757	85.85	1088
Names version 1	88.83	537	71.23	1088
Names version 2	90.50	705	84.65	1088
Nationality version 1	87.58	636	78.13	1088
Nationality version 2	88.17	**761**	**86.21**	1088
Name and Nationality version 1	87.42	493	67.00	1088
Name and Nationality version 2	**91.46**	691	84.28	1088

It is also noticeable that the second version of the prompt with name and nationality had slightly better accuracy when on conclusive comments. To better understand the prediction patterns of each iteration, their confusion matrices was studied. Figure 56 illustrates the confusion matrix of the second prompt version with player names and nationality, whereas the remainder can be found in Attachment A. As reflected in the matrices, the variants that managed to increase the overall accuracy, came at the cost of predicting the wrong team more often. As previously mentioned, it is preferable that the model wrongly predicts "Inconclusive" than for it to predict the wrong team. Therefore, the second prompt version with player names and nationality was deemed the most promising and used to classify the distilled dataset of nearly 350 000 comments.

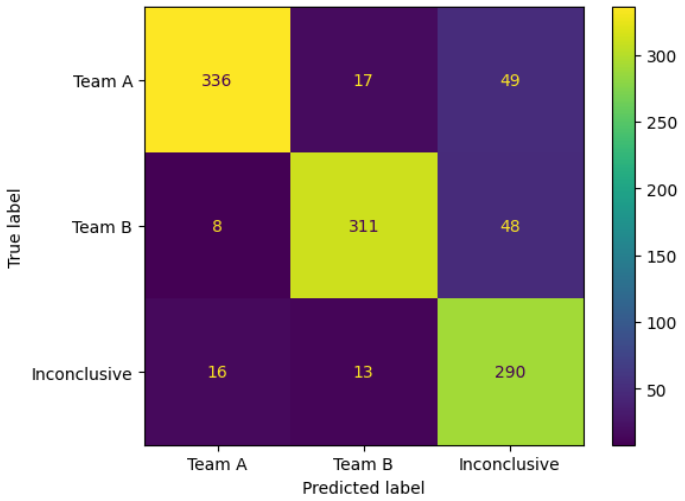


Figure 56 – Confusion matrix of predictor using version 2 prompt with names and nationality

8.3 Curating the dataset

As mentioned in section 0, extracting the sentiment within sentences is often the end goal of machine learning systems. Nevertheless, in this instance, it was used for labelling the dataset used to predict the outcome based on the fan sentiment. In this chapter, the curation of the dataset revolved around parsing the outputs of the dataset classification into a format that could be embedded within a match, as well as extracting how many times each commenter was right and wrong in former guesses. An illustration of the sentiment analysis dataset can be observed in Table 27, each match has two lines, representing each side of the matchup. The commenter guess is 1 if in favour of the first team, 0 if in favour of the second team, and null if inconclusive. Additionally, the dataset contains information about the commenters themselves, namely the number of previous right and wrong guesses. There is also information omitted from the table namely the bookmaker odds which are needed to run the EV betting simulation. This dataset follows the template used in the Prediction through match history, each field related to a commenter is a list of values that can then be easily transformed into multi-label encodings.

Table 27 – Example line of the sentiment analysis dataset

MatchId	commenterId	comenterGuess	rightGuesses	wrongGuesses	Won
000001	[com1, com2, …]	[1, null, …]	[10, 2, …]	[5, 1, …]	True
000001	[com1, com2, …]	[0, null, …]	[10, 2, …]	[5, 1, …]	False

8.4 Pre-Processing

The first step of pre-processing is removing rows that have do not have comments. Additionally, commenters with less than 10 comments were removed from the dataset, reducing the number of unique commenters from 40 000 to 6 000, significantly improving the training time. Information about each commenter is encoded using multi-label encoding. As shown in Table 28, the guesses of each commenter are mapped to 1 if the they think the first team will win, -1 if they think the team will lose and 0 if the commenter did not contribute to the discussion, either by being inconclusive or not commenting at all. On the other hand, the right and wrong number of previous guesses are normalized.

Table 28 – Example line of processed sentiment analysis dataset

MatchId	comm1_Guess	comm1_rightGuesses	…	comm2_Guess	…	Won
000001	1	0.82	…	0	…	True
000001	-1	0.82	….	0	…	False

8.4.1 Concept Drift

When compared to the experiments made on section 6.3.2, using multi-label encoding for commenters is extremely time consuming, taking 10 to 40 times as long, for 6 000 and 40 000 unique commenters, respectively. In this scenario, the previously employed strategy of creating 640 separate models is not viable, therefore testing for concept drift is of utmost importance. This way, the model's performance over time was analysed. A daily re-trained model was compared with monthly trained static models. As shown in Figure 57, the accuracy (a) of the

static models did not decrease massively over the months. Similarly, the betting simulation profit (b) of every model performed equally well. Similarly, the calibration metrics exposed in Figure 58, BS (a) and ECE (c) improved over time for every model. In other words, the models did not become less calibrated nor obsolete within the 5-month timeframe. Therefore, there is no significant gain in training the 640 daily models instead of monthly, or even trimester models.

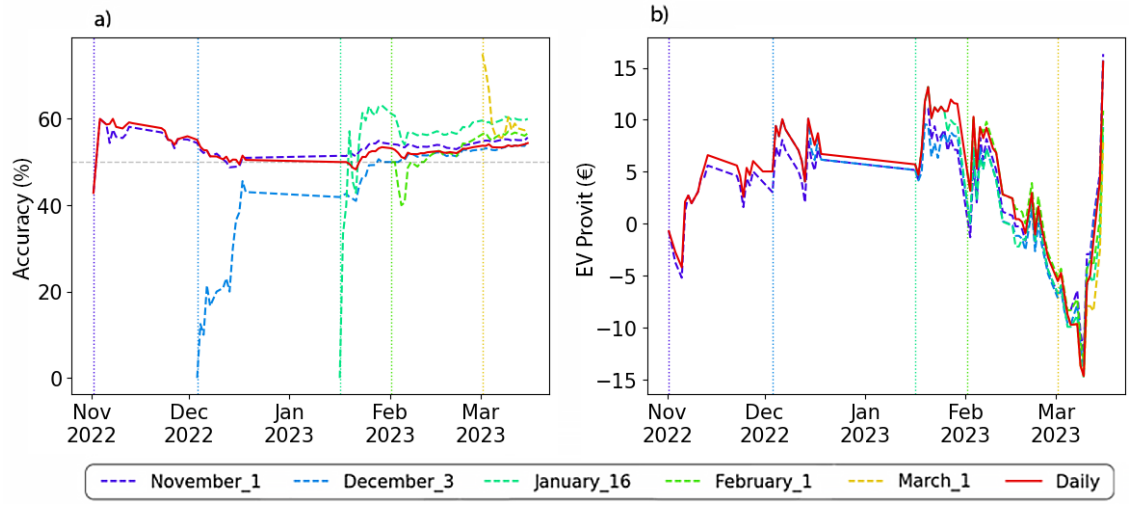


Figure 57 – Comparison of Accuracy(a) and EV Profit (b) from November 2022 to March 2023

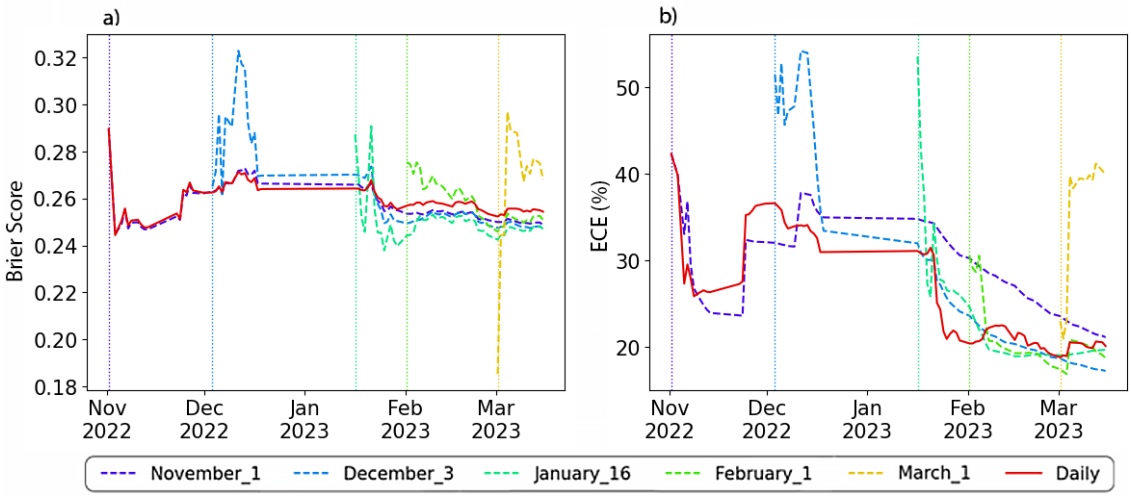


Figure 58 – Comparison of BS (a) and ECE (b) from November 2022 to March 2023

8.4.2 Feature Selection

For feature selection, it was compared multiple representations of the data using a SVM classifier with RBF kernel, which allowed for feature interaction. As previously mentioned, the dataset ignored commenters with less than 10 comments. The dataset used for training mirrors the one used in previous chapters., which is a substantial for evaluation. The selection was sequential rather than random, to prevent data leaks and better represent the model's use case. In this section it was evaluated a total of 4 representations for the commenters' guesses and proficiency:

- **Guess encoding –** the guess of each commenter is a feature through multi-label encoding. In other words, for each commenter, there is a column with -1, 0 or 1;
- **Guess encoding + Previous guesses –** along with the guess encoding, each commenter had 2 extra columns, one for the right number of guesses, and another one for the ones they got wrong;
- **Guess encoding + Accuracy –** in an effort to reduce the number of features, it was tested joining the previous guesses' columns into a single one, by calculating the accuracy of each commenter up to that point;
- **Guess x Accuracy encoding –** each guess is multiplied by the accuracy of its commenter.

The base representation, Guess Encoding, had an accuracy of 59.13%, a BS of 0.241, LL of 0.676 and an ECE of 3.84%, although with 4 empty calibration bins. Surprisingly, as displayed in Table 29, adding previous guesses or commenter accuracy did not greatly impact the model's performance, there is no metric that improved significantly, despite taking longer to compute. The model's accuracy did improve slightly to 59.73% and in the case of Guess Encoding + Commenter Accuracy, the calibration bins' ECE also improved to 3.75%, but on the other hand, it increased the number of empty bins, which is not desirable. Similarly, multiplying the guess by the accuracy barely changed the evaluation metrics apart from the calibration bins, although this time, the number of empty bins was halved while also slightly improving the ECE to 3.22%. Therefore, considering that this approach has the same number of features as the simple guess encoding while containing more information and better bin distribution it is the most promising and will be moving forward to the model selection, where it is compared against three heuristic-based approaches.

Table 29 – Comparison between different representations for comment predictions

Representation	Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
				ECE (%)	Empty
Guess encoding	59.13	**0.241**	0.676	3.84	4
Guess encoding + Previous guesses	**59.73**	**0.241**	0.676	4.57	4
Guess encoding + Accuracy	**59.73**	**0.241**	**0.675**	3.75	6
Guess x Accuracy encoding	59.26	**0.241**	0.676	**3.22**	2

8.5 Model Selection

To select the final model, multiple machine learning algorithms were tested for the most promising representation: Guess x Accuracy Encoding. The same dataset as the feature selection was used, which consisted of 2500 matches and discarded commenters with less than 10 previous comments, resulting in 6 000 unique commenters. This means there are 6000 unique features which makes RFR take exponentially longer to train as well as being less effective. Therefore, it was excluded from the final selection. Each model was retrained monthly, and the following techniques were evaluated:

- **Naïve Bayes Classifier (NBC) –** using the Bernoulli variant;
- **Gradient Boosting (GB) –** a max depth of 4, a learning rate of 1 and 1000 estimators;
- **Logistic Regression (LR) –** 100 000 max iterations;
- **Support Vector Machine (SVM) –** rbf kernel with probabilities.

Along the representation Guess x Accuracy Encoding, it was tested 3 heuristics:

- **Unbiased –** Every comment is taken at face value and has the same importance;
- **Positive Bias –** Only commenters that have a correct guess rate over 50% are considered and more correct guesses the more weight the commenter has;
- **Nuanced Bias –** Every commenter is considered, but if the commenter has a guess rate below 50%, it counts as a prediction to the opposing team. The comments also gain weight depending on the number of correct and wrong guesses of the commenter.

As reflected in Table 30, no model managed to reach an accuracy above 60% and the calibration metrics were also very poor overall. For Guess x Accuracy encodings, the GBM performed the worst. LR and NBC performed similarly, with LR being slightly better in every metric. On the other hand, SVM performed the best by every metric with 59.26% accuracy, 0.241 BS, 0.676 LL and a ECE of 3.22%. Besides having lower accuracy, the heuristics actually performed better than the GBM, NBC and LR algorithms, having slightly better calibration metrics. Out of the heiristics, the nuanced approach had the best calibration metrics, at 0.254 BS, 0.769 LL and 1.88 ECE. Although, this heuristic had 12 out of the 20 calibration bins empty, which means that the Guess x Accuracy encodings using SVM is still preferable.

The poor overall performance is also reflected on the bin accuracy distribution displayed in Table 31. Even the best performing model, Guess x Accuracy encoding using SVM, struggled to keep up with the bin expected accuracy, outside of the interval [40,65[.

Table 30 – Comparison between multiple machine learning techniques for each of the most promising representations

Feature set	Technique	Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
					ECE (%)	Empty
Guess x Accuracy Encoding	NBC	57.23	0.299	0.974	21.30	**0**
	GBM	56.00	0.339	1.321	28.71	**0**
	LR	57.36	0.282	0.834	18.22	**0**
	SVM	**59.26**	**0.241**	**0.676**	3.22	2
Unbiased		52.91	0.265	1.148	10.14	2
Positive Bias		56.08	0.255	0.821	5.28	10
Nuanced Bias		54.76	0.254	0.759	**1.88**	12

Table 31 – Average accuracy of each fan comments representation and ML technique per predicted probability bin and ECE

Bin (%)	Commenter Guess x Accuracy				Heuristics Bias		
	NBC	GBM	LR	SVM	Unbiased	Positive	Nuanced
[0, 5[	37.23	40.82	29.86	-	-	-	-
[5, 10[	35.61	40.66	40.59	0.00	0.00	-	-
[10, 15[	43.48	51.65	37.79	40.00	47.71	-	-
[15, 20[	46.19	45.50	35.29	25.93	53.53	-	-
[20, 25[	45.13	44.81	46.37	36.67	32.91	-	-
[25, 30[	44.75	50.00	51.14	39.46	44.39	0.00	-
[30, 35[	48.79	47.37	52.42	39.23	52.12	64.60	16.22
[35, 40[	45.45	42.57	46.71	42.56	43.23	42.59	54.67
[40, 45[	45.88	47.12	43.60	42.13	48.81	47.62	38.45
[45, 50[	53.38	51.17	46.19	44.39	41.81	42.90	46.61
[50, 55[	48.21	57.69	55.87	54.58	57.29	57.10	53.39
[55, 60[	46.52	41.04	56.71	58.66	50.16	52.38	61.55
[60, 65[	54.63	48.92	51.67	60.70	57.24	57.41	45.33
[65, 70[	52.55	54.55	45.28	59.49	45.61	35.40	83.78
[70, 75[	53.65	50.00	50.00	60.74	55.29	100.00	-
[75, 80[	57.81	56.58	56.60	65.45	68.65	-	-
[80, 85[	62.23	56.25	64.13	75.00	40.83	-	-
[85, 90[	62.56	56.54	63.21	62.50	53.62	-	-
[90, 95[	55.23	57.36	59.17	100.00	100.00	-	-
[95, 100]	60.05	57.02	70.37	-	-	-	-
ECE	21.3	28.71	18.22	3.22	10.14	5.28	**1.88**

8.6 Chapter Conclusions

In this chapter, first, the dataset was filtered, only 25% of the 2.4 million comments were truly useful for match prediction of which only direct replies to the thread were used, totalling 350 thousand comments. Patterns within the data were also found. Commenters with less than 20 comments are more likely to reply to other comments rather than the thread. Also, the higher the rank of the team, the more comments the thread has. Overall, there are over 7000 matches that have at least 10 replies to the thread, which is nearly the whole dataset.

Next, classification using LLMs is analysed. Out of fine-tuning, embeddings and prompt engineering, the latter best suits the use case. Therefore, it was analysed different sets of prompts, improving them iteratively and according to OpenAI's guidelines. The best prompt had the context of the game, players as well as the player's names and nationalities and achieved a 91.46% accuracy labelling comments that were conclusive.

Furthermore, the dataset was curated embedding the sentiment analysis results within the match data where each match has two lines representing each side of the matchup. From the curated dataset, it was removed rows without comments and commenters with less than 10 comments, as it would be hard to judge their significance. This way, the number of unique commenters was reduced from 40 thousand to 6 thousand, which were then encoded using multi-label encoding. Considering the massive number of features, studying the concept drift was once again crucial. Fortunately, it was not found any evidence of concept drift within a 5-month period and hence the models were trained monthly.

Four representations for commenters' guesses and proficiency were evaluated, of which using multiplying the commenter's accuracy with the commenters guess proved to be the most successful.

Finally, multiple machine learning techniques were evaluated, on a 3-year period with 2500 matches. The performances were not very promising, but SVM severely outperformed the remaining techniques, with an accuracy of 59.92%, 0.241 BS, 0.676 LL and an ECE of 3.22% and thus Accuracy x Guess encoding using SVM was chosen as the best solution.

9 Final Solution

In chapters 6, 0 and 1, it was explored the feature combinations as well as machine learning techniques that best fit outcome prediction through match history, bookmaker odds and fan sentiment. In this chapter, it is explored the usage of Ensemble Learning techniques to join the outputs of the three models.

9.1 Stacking

Considering the three models developed in previous chapters, match history, bookmaker odds and fan sentiment, are vastly different, the most suitable ensemble technique is stacking. As explained in section 3.6, it consists in training a fourth model – the stacking model – that learns from the predictions of the other three models. To prevent overfitting, it was settled on an approach that also uses bagging to better generalize the models. In other words, instead of training a single stacking model, it is trained multiple stacking models on different subsets of previous predictions, as represented in Figure 59.

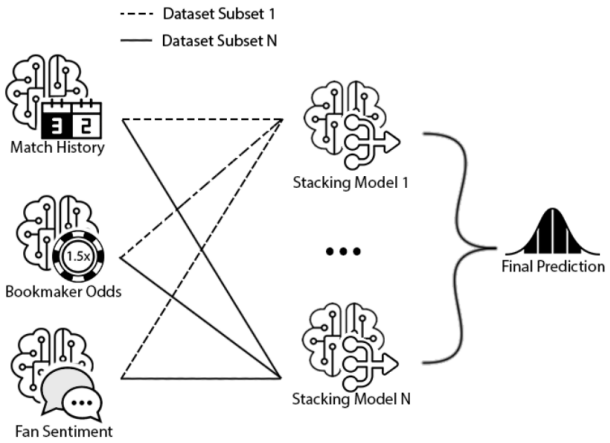


Figure 59 – Bagging of Stacking models

This was once again tested with multiple machine learning techniques and compared against an average prediction heuristic, that, as the name suggests averages the predictions of the three base models. The results are displayed in Table 32.

Table 32 - Comparison of stacking different machine learning algorithms

Approach		Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
					ECE (%)	Empty
Stacking	NBC	65.27	0.279	1.139	23.60	**0**
	NBC Bagging	65.31	0.278	1.134	23.41	**0**
	GBM	64.09	0.252	0.763	11.62	**0**
	GBM Bagging	64.35	0.252	0.762	11.55	**0**
	LR	64.15	0.238	0.696	10.18	**0**
	LR Bagging	64.28	0.235	0.686	9.60	**0**
	SVM	64.67	0.269	0.829	20.78	**0**
	SVM Bagging	64.71	0.268	0.823	20.43	**0**
	RFC (Bagging)	64.37	0.261	0.908	16.03	**0**
Best Match History Model		66.66	0.213	0.614	**2.10**	**0**
Best Bookmaker Odds Model		65.06	**0.212**	**0.612**	2.53	**0**
Best Fan Sentiment Model		59.26	0.241	0.676	3.22	2
Average Prediction Heuristic		**67.62**	0.213	0.616	5.19	4

It can be observed that the usage of bagging, in general, slightly improved the models' accuracy and calibration. Regardless, the stacking models still performed worse than the average prediction heuristic mainly when focusing on the accuracy bin distribution detailed in Table 33. In fact, the average heuristic managed to get the highest overall accuracy, while having one of the lowest Brier Score and Log Loss. Regardless, due to the calibration bins' distribution, the models based solely on match history and bookmaker odds still performed better.

Table 33 – Average accuracy of each Ensemble and ML technique and previous models per predicted probability bin and ECE

Bin (%)	SVM		LR		NBC		GBM		RFC	Average Heuristic	Match History	Bookmaker Odds	Fan Sentiment
	Simple	Bagging	Simple	Bagging	Simple	Bagging	Simple	Bagging					
[0, 5[	37.70	40.00	22.06	20.00	27.13	27.18	35.04	36.44	30.08	-	10.00	8.64	-
[5, 10[	29.55	30.00	28.57	27.12	37.84	37.84	27.27	24.87	33.83	-	20.00	11.54	0.00
[10, 15[	35.86	33.46	32.24	33.33	33.67	33.67	32.74	33.33	39.25	33.33	10.28	15.83	40.00
[15, 20[	33.02	36.84	30.43	27.69	43.75	43.08	35.04	33.88	26.17	16.67	21.11	18.53	25.93
[20, 25[	49.30	44.87	28.93	29.06	37.93	37.93	36.80	34.71	34.13	16.85	21.14	23.23	36.67
[25, 30[	42.67	42.86	39.73	40.54	49.28	49.28	28.35	35.09	39.78	23.08	26.58	27.90	39.46
[30, 35[	49.09	46.55	36.22	35.82	48.08	48.08	41.53	35.78	37.50	27.50	33.33	31.61	39.23
[35, 40[	39.13	40.38	44.80	43.41	45.10	45.10	51.04	47.50	39.78	31.22	39.75	41.64	42.56
[40, 45[	60.00	61.11	48.06	48.94	47.92	45.65	50.00	57.14	46.34	35.51	40.26	46.72	42.13
[45, 50[	42.22	42.31	49.19	48.39	53.85	55.17	50.91	50.00	40.91	47.04	46.10	47.99	44.39
[50, 55[	59.38	59.38	52.75	56.12	38.46	39.47	52.27	48.98	56.00	53.70	55.21	48.57	54.58
[55, 60[	47.83	48.28	56.19	51.30	60.61	63.89	45.10	49.18	56.63	64.73	59.42	57.11	58.66
[60, 65[	51.35	58.33	53.91	58.62	72.22	65.79	57.41	55.56	55.26	73.21	60.64	61.26	60.70
[65, 70[	47.62	45.45	64.52	62.60	43.48	46.51	53.33	61.18	61.86	70.72	65.91	73.52	59.49
[70, 75[	57.45	54.55	64.96	67.80	56.00	56.86	59.48	50.86	64.29	77.94	74.31	70.29	60.74
[75, 80[	59.52	66.00	66.17	67.44	52.83	50.00	63.08	68.50	63.55	83.78	78.04	78.08	65.45
[80, 85[	61.22	57.45	75.00	73.73	67.61	68.49	72.54	70.25	62.60	85.19	78.87	83.19	75.00
[85, 90[	65.66	69.06	69.60	68.60	51.69	50.56	71.74	71.79	70.63	60.00	91.75	82.57	62.50
[90, 95[	71.20	70.02	72.81	76.53	68.61	69.57	70.66	70.83	69.23	-	77.78	88.00	100.00
[95,100]	68.24	69.08	78.57	77.78	71.67	71.52	66.94	66.13	68.03	-	100.00	91.89	-
ECE	20.78	20.43	10.18	9.6	23.6	23.41	11.62	11.55	16.03	5.19	2.1	2.53	3.22

9.2 Strong Learner

Using smaller, weaker learners have advantages such as better performance and the ability to generalize, however, by separating the models, some important interactions might be lost. The data not being shared between models makes them unable to learn complex interactions such as:

- Are some fans biased towards certain players? Are fans of specific players able to properly access the outcome of a match?
- Are odds biased towards certain teams? Are certain teams consistently undervalued by bookmakers?
- Are there matchups where fans tend to have a better grasp on the outcomes than the bookmakers or vice versa?

To address this issue and hopefully have a model that can outperform the independent models based on match history, bookmaker odds and fan sentiment, tests were conducted where the models' datasets were merged and a "strong learner" was trained. Since the goal is to capture feature interaction, LR and NBC were discarded. On the other hand, due to the large number of features, the techniques RFR and GBM also become unviable, due to requiring a lot of tuning and being very computational expensive.

Strong learners are often associated with deep learning, however, deep neural networks would require careful training. Each of the 3 datasets has different levels of sparsity. More specifically, the match history's, out of the 3000+ different player encodings, only 20 are filled each game. In contrast, the odds' dataset only has 73 unique bookmakers and the sentiment analysis dataset sparsity can vary significantly depending on the number of comments. To ensure the neural network is learning interactions between these datasets and not merely focusing on the less sparse data, approaches such as attention mechanisms or pre-training each neural network on each of the individual datasets would be required. Additionally, as this is the final solution, the tests will be made by retraining the models daily, to minimize concept drift as much as possible, disqualifying deep neural networks as a viable option. This way, the best feasible option for this use case is the SVM.

Considering SVM will be used and that in section 7.5, the representation that performed the best with SVM (Bookmakers' Odds Encodings) was not the best overall representation (Bookmakers' Odds Average), the former will also be tested.

A total of 7 combinations were tested stemming from the following 4 feature sets: RADR Encodings with team rank and match date as weight (RADR), Bookmaker Odds Average (BOA), Bookmaker Odds Encodings (BOE) and comment Guess x Accuracy Encoding (GAE). Table 34 contains the results of these tests, as well as the baseline metrics of each of the feature sets on their own, using SVM.

Table 34 – Comparison between strong learner representations

RADR	BOA	BOE	GAE	Accuracy (%)	Brier Score	Log Loss	Calibration Bins	
							ECE (%)	Empty
X	**X**		**X**	65.33	0.224	0.639	5.18	0
X		**X**	**X**	**66.81**	0.216	0.622	2.67	2
X	**X**			66.43	0.216	0.621	3.36	4
X		**X**		66.05	**0.214**	**0.618**	3.94	6
X			**X**	63.24	0.228	0.650	4.49	**0**
	X		**X**	63.96	0.225	0.643	3.27	**0**
		X	**X**	66.43	0.217	0.625	3.32	2
X				64.68	0.220	0.629	2.52	2
	X			65.84	0.220	0.631	3.48	12
		X		66.06	0.215	0.621	3.52	8
			X	56.26	0.246	0.687	3.89	2

Immediately, including Odds representations seems like the key to an accuracy above 65%. The addition of either odd representation to the individual RADR and GAE models yielded improvements in accuracy, BS and LL. The GAE representation also benefited in the calibration bins, having lower ECE. For the RADRs, however, the calibration bins worsened, having more empty bins without significant improvements to the ECE. Overall, between models that contained odds, BOE's improvement of accuracy, Brier score and log loss was more noticeable than BOA's. Nevertheless, it also increased the number of empty bins.

Similar patterns were observed when considering the most complex models which include all 3 representations. The RADR + BOE + GAE representation had the highest overall accuracy, at 66.81% and maintained a very low BS (0.216) and LL (0.622). The ECE was very close to the RADR's, at 2.67% with 2 empty bins. It was also observed that the RADR + BOA + GAE representation's improvements were less pronounced, having a ECE of 5.18%, but managing 0 empty bins.

All in all, as reflected in the calibration bins displayed on Table 35, the most promising strong learner tested is the RADR + BOE + GAE variant, but the BOE + GAE is also very promising.

Table 35 – Average accuracy of each Strong Learner representation and ML technique per predicted probability bin and mean bin ECE

Bin (%)	RADR BOA GAE	RADR BOE GAE	RADR BOA	RADR BOE	RADR GAE	BOA GAE	BOE GAE	RADR	BOA	BOE	GAE
[0, 5[	0.00	-	-	-	13.33	-	0.00	-	-	-	-
[5, 10[	17.39	0.00	-	-	18.18	0.00	18.18	0.00	-	-	0.00
[10, 15[	13.24	17.95	0.00	-	21.95	4.35	16.42	26.67	-	-	30.00
[15, 20[	29.17	23.57	19.05	16.67	26.87	25.66	27.16	17.74	-	-	35.29
[20, 25[	30.47	27.32	19.12	17.21	27.52	23.44	29.46	22.50	-	19.74	41.67
[25, 30[	34.52	29.80	32.29	24.21	35.56	32.37	33.75	28.46	-	20.94	35.45
[30, 35[	36.71	32.45	36.16	29.69	35.98	36.16	35.90	37.50	30.37	28.12	43.37
[35, 40[	32.14	35.47	41.42	41.28	36.95	34.33	34.69	36.19	43.24	36.47	41.36
[40, 45[	37.39	39.69	39.33	47.66	34.72	41.63	41.43	43.75	49.76	44.64	43.82
[45, 50[	45.25	49.72	46.41	48.06	49.66	48.30	46.76	49.54	48.19	49.89	48.46
[50, 55[	54.36	56.67	56.18	57.64	51.11	56.99	53.87	49.54	48.11	51.93	49.42
[55, 60[	62.60	60.12	62.09	52.94	63.50	62.15	60.00	60.42	56.85	56.29	55.85
[60, 65[	68.84	62.03	58.79	56.89	64.68	59.12	64.82	64.76	53.31	63.92	60.83
[65, 70[	60.12	65.90	60.26	71.24	60.77	64.91	62.34	60.81	71.24	73.94	61.54
[70, 75[	69.38	69.76	72.62	75.99	70.40	70.24	72.08	74.69	-	78.72	62.11
[75, 80[	70.83	77.35	78.57	82.03	75.00	77.54	71.84	75.52	-	81.61	63.89
[80, 85[	74.47	76.06	82.72	90.00	75.00	73.57	72.37	81.67	-	-	70.59
[85, 90[	82.81	83.72	92.86	-	75.00	90.91	84.48	76.92	-	-	80.00
[90, 95[	86.67	100.00	-	-	84.21	100.00	85.71	100.00	-	-	100.00
[95, 100]	100.00	-	-	-	77.78	-	100	-	-	-	-
ECE	5.18	2.67	3.36	3.94	4.49	3.27	3.32	2.52	3.48	3.52	3.89

9.3 Betting Simulation

Considering OddAssist is a betting recommendation system, it would not be complete without extensively testing how it would perform in the real world. Therefore, betting simulations are conducted. To test whether the best possible solution was achieved, it is tested every representation that reached the Model Selection stage of chapters 6, 0 and 1. More specifically, the best machine learning for each representation presented in Table 19, Table 23 and Table 30 were tested, for brevity Player and Tam Binary Encodings are represented by PBE and TBE, respectively. Along the representations, the Team Rank and Match Average Odds heuristics from chapters 6 and 0, respectively, are also tested. Additionally the Average Prediction Ensemble heuristics from this chapter's Stacking section, as well as the Strong Learner models compared in Table 34 are also evaluated.

The odds considered for the betting simulations are the median odds of the bookmakers, for each side of the matchup. First, an evaluation of each model's performance over time was conducted. Figure 60 displays a betting simulation where it was always bet 1€. However, it was only bet when the bookmakers' odds had an expected return value of at least 5%. Considering not all models bet the same amount of times, instead of raw profit, it is used the percentual profit of each model. In other words, average return on investment, per bet, of each model. The simulation takes place from October 2020 to March 2023, however, in Figure 60, it is only shown results after March 2021 for visibility. Additionally, to provide some information about the number of bets of each representation, the thicker the line, the higher the number of bets.

Considering it was only bet on odds with an EV over 5%, it is expected that the model profits at least 5%. In other words, any model with a profit below 5% failed this experiment. At this EV threshold, the Odds Average and Team Rank heuristics performed the best by far, and it is noticeable that Team Rank heuristic bet significantly more that he Odds Average's. As for the machine learning techniques, the best performers were overwhelmingly Odds based. The only models without odds that managed to achieve the expected return were the RADR representation and PB), both using LR.

Surprisingly, the BOE representations that used only RADR or GAE slightly outperformed the representation that used all 3. Additionally, the PBE with Rank and Date as weight also outperformed the RADR 19-game opponent-weighted-average using LR, even though it was previously deemed worse by every metric. Regardless, at this threshold, the best machine learning model was RADR + BOE SVM followed by the best odds representation, BOA GBM. The best strong learners, RADR + BOE + GAE and BOE + GAE, also made the cut with 10.8% profit.

One of the best features of running simulations, is that it is possible to tweak the minimum EV threshold and observe how each model reacts. On paper, as the minimum EV increases, the profit of the models should increase accordingly, and always be superior to the minimum EV. This way, The SVM models RADR + BOE + GAE and BOE + GAE performed as expected, their profit increased at a higher pace than the minimum EV threshold and, as suggested in the Strong Learner section, these models performed very similarly within the different minimum EV thresholds. Similarly, the RADR + BOE representation, initially, was very close to the former two, although did not increase the same pace as the minimum EV thresholds, and barely reached a profit of 50% on the last minimum EV threshold (50%).

Table 36 displays how each model, performed at different minimum EV thresholds, ranging from 5 to 50.

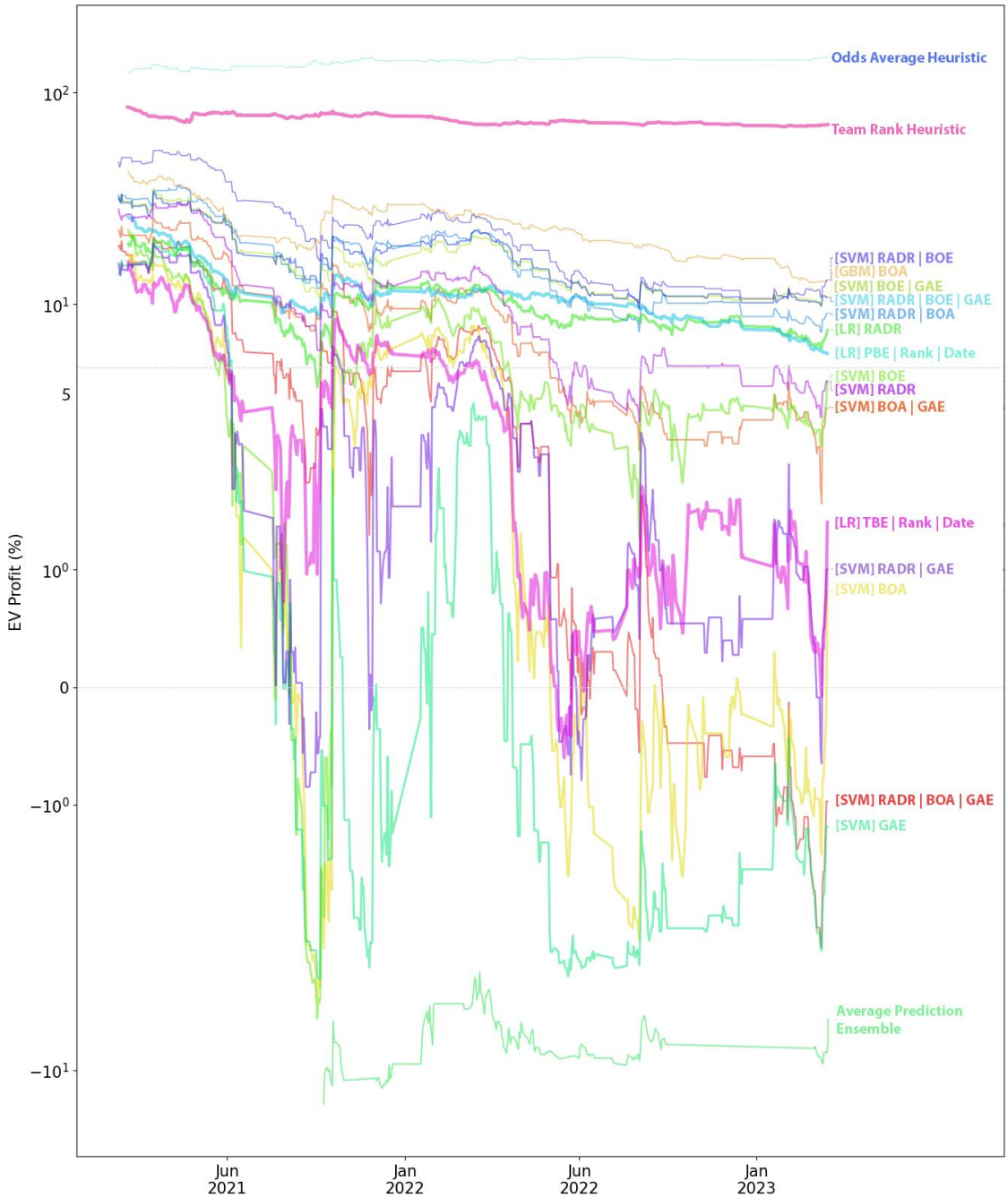


Figure 60 – Positive EV betting simulation results at 5% minimum EV over 30 months

At every minimum EV threshold, the Team Rank and Odds Average heuristics were still the most profitable approach, although with the increase of the minimum EV, the BOA GBM model was getting closer to the heuristics' profit and ended with a profit of 211% on the 50% EV threshold. Surprisingly, the individual LR RADR representation was the second best model, with over 90% profit at the 50% EV threshold.

The SVM models RADR + BOE + GAE and BOE + GAE performed as expected, their profit increased at a higher pace than the minimum EV threshold and, as suggested in the Strong Learner section, these models performed very similarly within the different minimum EV thresholds. Similarly, the RADR + BOE representation, initially, was very close to the former two, although did not increase the same pace as the minimum EV thresholds, and barely reached a profit of 50% on the last minimum EV threshold (50%).

Table 36 – Average profit per bet of each model per minimum EV

Model		Minimum EV (%)									
Representation	Algo	5	10	15	20	25	30	35	40	45	50
RADR | BOA |GAE	SVM	-1.0	4.8	6.6	11.2	10.3	17.5	25.4	26.1	27.3	27.9
RADR |BOE | GAE	SVM	10.8	19.2	26.2	34.5	42.5	47.5	52.7	61.9	68.3	76.8
RADR | BOA	SVM	9.0	11.6	20.2	24.0	36.4	41.9	52.6	55.6	62.5	65.6
RADR |BOE	SVM	13.0	18.6	21.9	31.4	31.6	38.8	40.5	42.8	43.4	50.1
RADR | GAE	SVM	1.0	2.2	1.6	4.0	2.8	3.7	6.9	6.7	7.8	12.8
BOA | GAE	SVM	3.2	5.0	7.7	10.8	16.1	18.0	17.1	20.4	24.9	32.7
BOE | GAE	SVM	10.8	16.5	23.0	35.1	46.9	47.8	50.7	60.6	64.3	69.6
RADR	SVM	4.3	7.2	10.9	15.5	25.6	32.9	35.5	42.7	60.6	72.8
BOA	SVM	0.8	1.5	0.3	3.0	5.8	5.7	8.0	7.5	10.6	13.3
BOE	SVM	4.3	8.6	11.3	18.3	18.9	23.1	24.0	26.5	26.6	30.6
RADR	LR	7.5	10.6	12.5	21.7	27.3	36.7	41.8	48.3	69.5	90.3
PBE	LR	5.8	10.0	12.6	19.3	27.8	33.0	39.2	49.9	54.1	69.5
TBE	LR	1.4	3.2	6.1	7.7	12.0	10.0	11.1	13.2	20.7	29.7
BOA	GBM	13.0	25.3	34.7	51.3	68.1	88.2	105	119	150	211
GAE	SVM	-1.2	0.2	0.4	0.8	1.3	1.6	1.3	1.2	-0.4	1.1
Ensemble_Average		-5.8	-5.6	-2.5	1.1	4.8	4.0	7.5	1.0	-1.6	2.5
Team Rank Heuristic		70.8	81.8	92.7	104	123	142	160	182	199	220
Odds Average Heuristic		146	165	185	181	193	206	222	223	236	243

Interestingly, except for RADR SVM. Every model that failed to meet the 5% expected profit in Figure 60 did not recover for higher minimum EV thresholds. In other words, these models' profit did not increase at the same pace as the minimum EV. The opposite was also observed, models that were above the 5% profit previously, generally kept increasing at a pace equal or higher than the minimum EV.

It is also evident that, in general, more conservative models, in other words, models that generate lower odds and thus bet less, had more success in the betting simulation, and while not model could match the Team Rank and Odds Average heuristics, the BOA GBM, RADR LR and the final strong learners were all very stable and did yield noticeable profits long terms, which was the main goal of this project.

9.4 Chapter Conclusions

In this chapter, first, methods of joining the different machine learning models created on Prediction through match history, Prediction through bookmaker odds and Sentiment Analysis.

were explored, namely Stacking and the usage of Strong Learners. The best individual to be merged were RADR LR, BOA GBM and GAE SVM.

For stacking, multiple machine learning algorithms were tested and bagging was also used to better generalise the models. However, every one of these algorithms performed worse than the individual models of RADR and BOA and were also outperformed by a simple heuristic that averaged the output of the previously created models. This heuristic had a record high accuracy of 67.62%, as well low BS and LL, 0.213 and 0.616, respectively. Nonetheless, it had a higher ECE than the individual models at 5.19% with 4 empty bins, which is not desirable.

Strong learners, however, did yield significant improvements, both in the accuracy of the predictions and the calibration metrics. Even though the best individual model for the odds was BOA GBM, when combining the representations, it was used SVM, resulting in the BOA underperforming when compared to the BOE. In fact, the best strong learners were RADRs + BOE + GAE and BOE + GAE. While the former had an accuracy of 66.81%, BS of 0.216, LL of 0.622 and ECE of 2.67 % with 2 empty bins, the latter was only slightly worse with an accuracy of 66.43%, BS of 0.217, LL of 0.625 and ECE of 3.32% the same number of empty bins.

Lastly, a betting simulation was made: each model makes a prediction, calculates the EV of the bookmakers' median odd and decides whether to bet 1€. The betting simulation was run for a total of 10 minimum EV thresholds, ranging from 5% to 50%. Surprisingly, the best performers were not the machine learning models, but rather the heuristics. More specifically, the heuristic that considered just the teams' ranks, and the bookmakers' average odds. The former had a profit ranging from 70% to 220% while the latter ranged from 146% to 243%. In a similar way, the best model of Prediction through bookmaker odds. BOA GBM was based purely on the bookmakers' average odd performed significantly better than the remaining machine learning approaches, with profits from 13% to 211%. The next best machine learning approaches were the LR RADR model with profits ranging from 7.5% to 90.3% and the SVM model RADR + BOE + GAE with profits ranging from 10.8% to 76.8%.

All in all, the closest model to the theoretical highest accuracy of 70% proposed in [111] was the average prediction ensemble, at 67.62%. However, this accuracy did not translate to the betting simulation. From the underperformance of this stacking approach, to the massive overperformance of the heuristics and the PBE representation, the results of this chapter were unexpected and illustrate the importance of thorough and unbiased testing, as well as the importance of comparing machine learning results to simplistic heuristics, as proposed in [118]. Without considering the betting simulation, the final proposed solution would probably be RADR + BOA + GAE using SVM and that model did achieve the goal of this thesis of being able to generate long-term profits.

10 Conclusions

This chapter is a summary of the developed work and the conclusions drawn from it. Additionally, the achieved objectives are described and future developments for further improvement are proposed.

10.1 Summary and conclusions

It is a fact that sports fans often like to prove their knowledge through putting money on the line. Esports, although still a blooming industry, counts with millions in prize pools and viewership numbers among young men which allied to the rise of the digitalization and spread of bookmakers is in a prime position to be prevalent topic of research in future generations. This thesis delves into the intersection of sports betting and machine learning within the dynamic context of CS:GO.

First, the bookmakers' odds are exposed in detail along with progressive betting strategies, namely line shopping, arbitrage betting and positive EV betting. While no method guarantees success, these offer avenues for capitalizing in potential market inefficiencies. Additionally the parallel between CS:GO and traditional sports is established, emphasizing the importance of historical matchups and player dynamics.

To ensure the best machine learning system is developed, an extensive review on machine learning techniques was conducted. In the same vein, robust evaluation metrics and methods were introduced, providing a rigorous framework for accessing the models' performance, while avoiding concept drift and data leaks.

Previous works that attempted similar feats were studied and criticized. For match outcome prediction, although most of the works were applied to a betting context, they often tunnel visioned on the model's accuracy in detriment of its calibration, and many lacked extensive betting simulations. Within the sentiment analysis context, it was not found relevant works.

The techniques and technologies used to retrieve, process and store information are also presented along with ethical aspects and possible threats of such endeavours. While there were no ethical concerns, the models are reliant on analysing fan sentiment which can be easily

tempered with. Although, precautions were taken to minimize these aspects, namely, encoding each commenter as an individual feature and considering the commenter's accuracy, to give the models further nuance.

For Prediction through match history, relationships between player and team dynamics were uncovered. It was also observed that history-based models were very susceptible to concept drift. Additionally, it was found that encoding player metrics instead of a simple binary encoding was promising. More specifically, using the recency-weighted-average of the players' RADR over the last 19 games had the best results. Furthermore, the addition of team rank and weighting recent games more highly proved to improve the model's prediction capabilities. This resulted in a model with 66.66% accuracy, 0.213 BS, 0.614 LL and 2.10% ECE.

Exploring the bookmakers' odds, it was found that the every bookmaker, on average, set the favourite's odds below the market's median, while the underdog was often overvalued. This time, when testing for concept drift, it was observed that bookmakers' odds encodings was very sensitive to it while bookmakers' odds median was more stable. The best resulting model was a GBM trained on bookmakers' odds averages, having 65.07% accuracy, 0.212 BS, 0.612 LL and 2.53% ECE.

For sentiment analysis, it was firstly used GPT 3.5 to classify the sentiment of each comment with 84.28% accuracy. However, from the comments that were deemed conclusive, in other words, comments that the model believed were favouring a team, it was observed a 91.46% accuracy instead. These labelled comments were then compiled and the accuracy of each commenter was tracked. The final solution was encoding the sentiment of each commenter, multiplied by their accuracy, which resulted in an SVM model with 59.26% accuracy, 0.241 BS, 0.676 LL and 2.85% ECE.

Compiling the result of each model, however, did not go as expected. Stacking the models only yielded promising results when considering their average prediction. It had a promising accuracy of 67.62%, 0.213 BS and 0.616 LL, however, it had a poor ECE, 5.19% with 4 empty calibration bins. Instead, merging the datasets and building a strong learner yielded better results. The best model was a SVM trained on RADR encodings with team rank, bookmakers' odds encodings and the encodings of each commenter's guess multiplied by their accuracy. This solution had 66.81% accuracy, 0.216 BS and 0.618 LL. Although it had a slightly higher ECE (2.67%) than some individual models, it is far more robust as it considers more variables.

From the betting simulation it was concluded that the team rank and average odd heuristics performed massively better than the machine learning approaches. Nevertheless, every model that was deemed good within the model selection, namely bookmakers' odds average using GBM, player RADR encodings using LR and the previously mentioned strong learner all yielded profits long term, ensuring the long-term profit objective.

Finally, the goal of profiting long term from CS:GO betting and every task proposed in Objectives was successfully completed, namely:

- Study how odds are calculated by the bookmakers - **Odds**;
- Study approaches that have successfully beaten bookmakers - **Beating the Odds**;
- Study similarities between CS:GO and traditional sports - **Similarities with Traditional Spots**;

- Study of data retrieval techniques - **Data scraping**;
- Study of the literature of machine learning techniques **- Machine Learning;**
- Study of the literature of sentiment analysis - **Sentiment Analysis**;,
- Study of the literature of sport match outcome prediction models based on historic data and/or bookmakers' predictions - **Outcome Prediction**;
- Experimentation with multiple pre-processing and processing techniques for each model - **Prediction through match history**, **Prediction through bookmaker odds** and **Sentiment Analysis;**
- Evaluation of the performance of the tested models and techniques - **Betting Simulation**.

10.2 Future work

Even though this solution proved to be able to have long-term profits, there are still avenues to improve it further. Firstly, due to concept drift, it was not possible to reliably test the impact of hyperparameter tuning. It was either tested on a substantially smaller dataset or without retraining the models, both of which do not compose a strong case for their efficacy. The focus of this thesis was creating a machine learning system from scratch, extracting and processing the data, extracting the best features and creating a model that could source its prediction on different kinds of data.

Additionally, within the match odds prediction, the most obvious opportunity stems from further exploring the player performance metrics and how they would interact, either by choosing different metrics, different number of games or different aggregation methods. Further experimentation with neural networks with attention masks, mainly when allied with more features ought to bring promising results.

From the perspective of bookmaker odds, the main improvement would be in automation and quality control. From a machine learning standpoint, there could also be further hyperparameter tuning, but it is unlikely to bring massive returns.

The sentiment analysis module could be improved in two main ways. The first one is further refining the GPT prompt, it currently is 0-shot and giving it some examples is very likely to improve its classification quality. Nonetheless, the picked examples ought to be carefully picked, as not to cause overfitting. Additionally, it could be improved by moving it off the cloud. In other words, by training a smaller LLM that could even be finetuned to be an expert of CS:GO and its teams.

Finally, the final solution's module could have further experimentation, namely, using team encodings along the player encodings, and testing combination of strong learners and stacking. For example, stacking the match history and sentiment analysis outputs, and then passing it as a feature for the odds average model.

References

[1] HLTV, “CS:GO Matches & livescore | HLTV.org.” Accessed: Jan. 21, 2023. [Online]. Available: https://www.hltv.org/matches

[2] J. H. Humphrey, *Roman circuses: arenas for chariot racing. Univ of California Press*. Univ of California Press, 1986. Accessed: Nov. 30, 2022. [Online]. Available: https://books.google.co.uk/books?hl=en&lr=&id=couetXBQO9AC&oi=fnd&pg=PA1&dq=chariot+racing+betting&ots=jYeMMSI5MM&sig=-fgjW81IZDY0CfwPm3cByldTkkk&redir_esc=y#v=onepage&q=chariot%20racing%20betting&f=false

[3] S. Bell, “Roman Chariot Racing,” *A Companion to Sport and Spectacle in Greek and Roman Antiquity. Chichester: Wiley Blackwell*, 2014.

[4] I. TURCU, G. B. BURCEA, and D. L. DIACONESCU, “THE IMPACT OF THE BETTING INDUSTRY ON SPORTS,” *Series IX Sciences of Human Kinetics*, vol. 13(62), no. 2, pp. 251–258, Dec. 2020, doi: 10.31926/but.shk.2020.13.62.2.32.

[5] statista, “Top gambling companies by brand value 2021 | Statista.” Accessed: Jan. 21, 2023. [Online]. Available: https://www.statista.com/statistics/883685/top-online-gambling-companies-by-total-assets/

[6] Data Bridge Market Research, “Sports Betting Market Size Is Likely to Experience a.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20221025100135/https://www.globenewswire.com/en/news-release/2022/07/14/2479929/0/en/Sports-Betting-Market-Size-Is-Likely-to-Experience-a-Tremendous-Growth-of-USD-167-66-billion-by-2029-registering-a-CAGR-of-10-26-by-Size-and-Share-Industry-Growth-Regional-Outlook-.html

[7] D. Lenton, “How Do Bookmakers Set Their Odds?” Accessed: Dec. 01, 2022. [Online]. Available: https://www.cheekypunter.com/faq/how-do-bookmakers-set-their-odds/

[8] Google Scholar, “sports betting machine learning - Google Scholar.” Accessed: Jan. 21, 2023. [Online]. Available: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=sports+betting+machine+learning&btnG=

[9] uktech.news, “How Artificial Intelligence Continues to Shape the Sports Betting Industry - UKTN | UK Tech News.” Accessed: Dec. 01, 2022. [Online]. Available: https://www.uktech.news/other_news/how-artificial-intelligence-continues-to-shape-the-sports-betting-industry

[10] onlinebetting.org.uk, “How Do Bookmakers Calculate Odds, Price Markets And Make Money | Betting Margins Explained | Online Betting UK.” Accessed: Dec. 01, 2022. [Online]. Available: https://www.onlinebetting.org.uk/betting-guides/how-do-bookmakers-set-odds-and-make-money.html

[11] M. Busby, “Revealed: how bookies use AI to keep gamblers hooked | Technology | The Guardian,” 2018. Accessed: Dec. 01, 2022. [Online]. Available: https://www.theguardian.com/technology/2018/apr/30/bookies-using-ai-to-keep-gamblers-hooked-insiders-say

[12] Business Insider, “Study: Young Men Like ESports More Than Traditional Sports.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20220909060933/https://www.businessinsider.com/nfl-ratings-drop-study-young-men-watch-esports-more-than-traditional-sports-2017-9

[13] Esport Charts, “CS:GO - Esports Viewership and Statistics | Esports Charts.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20230121023842/https://escharts.com/games/csgo

[14] OddsJam, “Positive Expected Value Betting Tool - Positive EV Bets | OddsJam.” Accessed: Jan. 21, 2023. [Online]. Available: https://oddsjam.com/betting-tools/positive-ev

[15] Twitter, “Twitter.” Accessed: Jan. 22, 2023. [Online]. Available: https://twitter.com/home

[16] Reddit, “Reddit.” Accessed: Jan. 22, 2023. [Online]. Available: https://new.reddit.com/

[17] OpenAI, “ChatGPT: Optimizing Language Models for Dialogue.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20230121133613/https://openai.com/blog/chatgpt/

[18] IEEE, “IEEE Xplore.” Accessed: Jan. 21, 2023. [Online]. Available: https://ieeexplore.ieee.org/Xplore/home.jsp

[19] ScienceDirect, “ScienceDirect.com | Science, health and medical journals, full text articles and books.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.sciencedirect.com/

[20] Web of Science, “Web of Science Master Journal List - Search.” Accessed: Jan. 22, 2023. [Online]. Available: https://mjl.clarivate.com/search-results

[21] b-on, “b-on.” Accessed: Jan. 21, 2023. [Online]. Available: https://www.b-on.pt/

[22] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[23] arXiv, "arXiv.org e-Print archive." Accessed: Jan. 22, 2023. [Online]. Available: https://arxiv.org/

[24] M. Montone, "Optimal pricing in the online betting market," *J Econ Behav Organ*, vol. 186, pp. 344–363, 2021.

[25] J. K. Blitzstein and J. Hwang, *Introduction to probability*. Crc Press Boca Raton, FL, 2015.

[26] E. Franck, E. Verbeek, and S. Nüesch, "Inter-market Arbitrage in Betting," *Economica*, vol. 80, no. 318, pp. 300–325, Apr. 2013, doi: 10.1111/ECCA.12009.

[27] Matheu Berry's Fantasy Life, "Why shopping for lines is important. Tips from a sports bettor." Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20220815064726/https://www.fantasylife.com/articles/betting/what-is-line-shopping

[28] Odds Shark, "Line Shopping | Odds Shark." Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20220216102552/https://www.oddsshark.com/sports-betting/line-shopping

[29] OddsPedia, "Sure Bets Today » Arbitrage Betting Finder & Calculator 【Free】." Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20220928234126/https://oddspedia.com/surebets

[30] OddsJam, "Positive EV Sports Betting: Form Your Own 'Hedge Fund' | Best Sports Betting Strategies, Advice - YouTube." Accessed: Jan. 21, 2023. [Online]. Available: https://youtu.be/MtpdM5VjPfU?t=18

[31] W. EDWARDS, "Probability-preferences in gambling.," *Am J Psychol*, vol. 66, no. 3, pp. 349–364, 1953, doi: 10.2307/1418231.

[32] A. Björklund, W. Johansson Visuri, F. Lindevall, and P. Svensson, "Predicting the outcome of CS:GO games using machine learning." 2018. Accessed: Jan. 15, 2023. [Online]. Available: https://hdl.handle.net/20.500.12380/256129

[33] J. G. Reitman, M. J. Anderson-Coto, M. Wu, J. S. Lee, and C. Steinkuehler, "Esports Research: A Literature Review," *Games Cult*, vol. 15, no. 1, pp. 32–50, Jan. 2020, doi: 10.1177/1555412019840892/ASSET/IMAGES/LARGE/10.1177_1555412019840892-FIG2.JPEG.

[34] ESL Gaming, “Careers - ESL Gaming GmbH.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20221220152332/https://careers.esl.com/

[35] Ghazz, “The Sixth Player: The Roles of a Coach - Articles - Dignitas.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20181215020502/http://www.dignitas.gg/articles/blogs/CSGO/11771/the-sixth-player-the-roles-of-a-coach/

[36] O. Švec, “Predicting Counter-Strike Game Outcomes with Machine Learning,” 2022.

[37] liquidpedia, “BLAST Premier - Liquipedia Counter-Strike Wiki.” Accessed: Jan. 21, 2023. [Online]. Available: https://liquipedia.net/counterstrike/BLAST_Premier

[38] ESL Gaming, “ESL Pro Tour CSGO - One Tour. One Story.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20230107185553/https://pro.eslgaming.com/tour/csgo/

[39] liquidpedia, “CS:GO Major Championships - Liquipedia Counter-Strike Wiki.” Accessed: Jan. 21, 2023. [Online]. Available: https://liquipedia.net/counterstrike/Majors

[40] J. Rutting, “The greatest CS:GO rivalries.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20210125064050/https://thegamehaus.com/esports/greatest-csgo-rivalries/2017/07/11/

[41] S. Klim, “5 Most Iconic Rivalries In The History Of CS:GO.” Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20220128124910/https://gurugamer.com/esports/5-most-iconic-rivalries-in-the-history-of-csgo-5019

[42] National Basketball Association, “2022 Playoffs | Bracket | NBA.com.” Accessed: Jan. 21, 2023. [Online]. Available: https://www.nba.com/playoffs/2022

[43] UEFA, “UEFA Champions League | UEFA.com.” Accessed: Jan. 21, 2023. [Online]. Available: https://pt.uefa.com/uefachampionsleague/

[44] J. Harika, P. Baleeshwar, K. Navya, and H. Shanmugasundaram, “A Review on Artificial Intelligence with Deep Human Reasoning,” *Proceedings - International Conference on Applied Artificial Intelligence and Computing, ICAAIC 2022*, pp. 81–84, 2022, doi: 10.1109/ICAAIC53929.2022.9793310.

[45] G. Georgiev, “Has Interest in Data Science Peaked Already? | by Georgi Georgiev | Towards Data Science.” Accessed: Jan. 21, 2023. [Online]. Available:

https://web.archive.org/web/20230121153826/https://towardsdatascience.com/has-interest-in-data-science-peaked-already-437648d7f408?gi=7255bbbf1521

[46] S. v. Mahadevkar *et al.*, "A Review on Machine Learning Styles in Computer Vision - Techniques and Future Directions," *IEEE Access*, vol. 10, pp. 107293–107329, 2022, doi: 10.1109/ACCESS.2022.3209825.

[47] S. Pouyanfar *et al.*, "A Survey on Deep Learning," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 92, Sep. 2018, doi: 10.1145/3234150.

[48] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057135. Cambridge University Press, 2014. doi: 10.1017/CBO9781107298019.

[49] L. Schmarje, M. Santarossa, S. M. Schroder, and R. Koch, "A Survey on Semi-, Self- and Unsupervised Learning for Image Classification," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3084358.

[50] P. C. Sen, M. Hajra, and M. Ghosh, "Supervised Classification Algorithms in Machine Learning: A Survey and Review," *Advances in Intelligent Systems and Computing*, vol. 937, pp. 99–111, 2020, doi: 10.1007/978-981-13-7403-6_11/COVER.

[51] Texas Instruments, "Solution 34506: Calculating and Graphing a Linear Regression on the TI-84 Plus family of graphing calculators." Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20221126181425/https://education.ti.com/en/customer-support/knowledge-base/ti-83-84-plus-family/product-usage/34506

[52] Wikipedia, "Logistic regression." Accessed: Jan. 21, 2023. [Online]. Available: https://web.archive.org/web/20230118185342/https://en.wikipedia.org/wiki/Logistic_regression

[53] J. Hoare, "What is a Decision Tree? - Displayr." Accessed: Jan. 21, 2023. [Online]. Available: https://www.displayr.com/what-is-a-decision-tree/

[54] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022, doi: 10.1109/ACCESS.2022.3207287.

[55] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," *Inf Softw Technol*, vol. 122, p. 106287, Jun. 2020, doi: 10.1016/J.INFSOF.2020.106287.

[56] H. Steinhaus, "Sur la division des corps matériels en parties," *Bulletin L'Académie Polonaise des Science IV*, pp. 801–804, 1957. Accessed: Jan. 07, 2023. [Online].

Available:
https://scirp.org/reference/referencespapers.aspx?referenceid=2408792

[57] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” *https://doi.org/*, vol. 5.1, pp. 281–298, Jan. 1967, Accessed: Jan. 07, 2023. [Online]. Available: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fifth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992

[58] C. Coelho, “Plataforma para o Agregador de Energia,” 2020.

[59] Leonard. Kaufman and P. J. Rousseeuw, “Finding groups in data : an introduction to cluster analysis,” p. 342, 1990, Accessed: Jan. 07, 2023. [Online]. Available: https://www.wiley.com/en-us/Finding+Groups+in+Data%3A+An+Introduction+to+Cluster+Analysis-p-9780470317488

[60] H. P. Kriegel, P. Kröger, J. Sander, and A. Zimek, “Density-based clustering,” *Wiley Interdiscip Rev Data Min Knowl Discov*, vol. 1, no. 3, pp. 231–240, May 2011, doi: 10.1002/WIDM.30.

[61] Sudha and A. GirijammaH, “Appraising Research Direction & Effectiveness of Existing Clustering Algorithm for Medical Data,” *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 3, 2017, doi: 10.14569/IJACSA.2017.080348.

[62] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph, “A Review of Dimensionality Reduction Techniques for Efficient Computation,” *Procedia Comput Sci*, vol. 165, pp. 104–111, Jan. 2019, doi: 10.1016/J.PROCS.2020.01.079.

[63] Forbes, “Diaper-beer syndrome.” Accessed: Jan. 21, 2023. [Online]. Available: https://www.forbes.com/forbes/1998/0406/6107128a.html?sh=48bb111f6260

[64] M. J. Zaki, “Scalable algorithms for association mining,” *IEEE Trans Knowl Data Eng*, vol. 12, no. 3, pp. 372–390, 2000, doi: 10.1109/69.846291.

[65] R. Agrawal, R. S.-Proc. 20th int. conf. very large data bases, and undefined 1994, “Fast algorithms for mining association rules,” *cs.duke.edu*, Accessed: Jan. 08, 2023. [Online]. Available: http://www.cs.duke.edu/courses/compsci516/spring16/Papers/AssociationRuleMining.pdf

[66] M. Qisman, R. Rosadi, and A. S. Abdullah, “Market basket analysis using apriori algorithm to find consumer patterns in buying goods through transaction data

(case study of Mizan computer retail stores)," *J Phys Conf Ser*, vol. 1722, no. 1, p. 012020, Jan. 2021, doi: 10.1088/1742-6596/1722/1/012020.

[67] Y. Ali, A. Farooq, T. M. Alam, M. S. Farooq, M. J. Awan, and T. I. Baig, "Detection of schistosomiasis factors using association rule mining," *IEEE Access*, vol. 7, pp. 186108–186114, 2019, doi: 10.1109/ACCESS.2019.2956020.

[68] G. Zhong and K. Huang, Eds., *Semi-supervised learning : background, applications and future directions*. 2018.

[69] X. Yang, Z. Song, I. King, and Z. Xu, "A Survey on Deep Semi-supervised Learning," *IEEE Trans Knowl Data Eng*, pp. 1–20, Feb. 2021, doi: 10.48550/arxiv.2103.00550.

[70] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*. Springer Cham, 2009. doi: 10.1007/978-3-031-01548-9.

[71] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach Learn*, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/S10994-019-05855-6/FIGURES/5.

[72] N. C. Luong *et al.*, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3133–3174, Oct. 2019, doi: 10.1109/COMST.2019.2916583.

[73] Y. Li, "Deep Reinforcement Learning," *Smart Innovation, Systems and Technologies*, vol. 273, pp. 136–142, Oct. 2018, doi: 10.48550/arxiv.1810.06339.

[74] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," May 2020, doi: 10.48550/arxiv.2005.01643.

[75] M. L. Puterman, "Chapter 8 Markov decision processes," *Handbooks in Operations Research and Management Science*, vol. 2, no. C, pp. 331–434, Jan. 1990, doi: 10.1016/S0927-0507(05)80172-0.

[76] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[77] S. Padakandla, "A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, Jul. 2021, doi: 10.1145/3459991.

[78] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature 2017 550:7676*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, doi: 10.1038/nature24270.

[79] A. Shrestha and A. Mahmood, “Review of deep learning algorithms and architectures,” *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[80] S. Dong, P. Wang, and K. Abbas, “A survey on deep learning and its applications,” *Comput Sci Rev*, vol. 40, p. 100379, May 2021, doi: 10.1016/J.COSREV.2021.100379.

[81] J. Johnson, “What’s a Deep Neural Network? Deep Nets Explained – BMC Software | Blogs.” Accessed: Jan. 21, 2023. [Online]. Available: https://www.bmc.com/blogs/deep-neural-network/

[82] Y. Lu, “Artificial intelligence: a survey on evolution, models, applications and future trends,” *https://doi.org/10.1080/23270012.2019.1570365*, vol. 6, no. 1, pp. 1–29, Jan. 2019, doi: 10.1080/23270012.2019.1570365.

[83] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[84] V. Dumoulin, F. Visin, and G. E. P. Box, “A guide to convolution arithmetic for deep learning,” Mar. 2016, doi: 10.48550/arxiv.1603.07285.

[85] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, doi: 10.48550/arxiv.1704.04861.

[86] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *IEEE Trans Neural Netw Learn Syst*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[87] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1746–1751, Aug. 2014, doi: 10.48550/arxiv.1408.5882.

[88] D. Yan, K. Li, S. Gu, and L. Yang, “Network-Based Bag-of-Words Model for Text Classification,” *IEEE Access*, vol. 8, pp. 82641–82652, 2020, doi: 10.1109/ACCESS.2020.2991074.

[89] J. A. Botha *et al.*, “Natural language processing with small feed-forward networks,” *arXiv preprint arXiv:1708.00214*, 2017.

[90] Y. Bengio, P. Simard, and P. Frasconi, “Learning Long-Term Dependencies with Gradient Descent is Difficult,” *IEEE Trans Neural Netw*, vol. 5, no. 2, pp. 157–166, 1994, doi: 10.1109/72.279181.

[91] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, “Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN,” *Proceedings of the IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition, pp. 5457–5466, Mar. 2018, doi: 10.48550/arxiv.1803.04831.

[92] M. Z. Alom *et al.*, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics 2019, Vol. 8, Page 292*, vol. 8, no. 3, p. 292, Mar. 2019, doi: 10.3390/ELECTRONICS8030292.

[93] M. Phi, "Illustrated Guide to Recurrent Neural Networks: Understanding the Intuition - YouTube." Accessed: Jan. 21, 2023. [Online]. Available: https://www.youtube.com/watch?v=LHXXI4-IEns&t=292s

[94] J. Chen, H. Jing, Y. Chang, and Q. Liu, "Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process," *Reliab Eng Syst Saf*, vol. 185, pp. 372–382, May 2019, doi: 10.1016/J.RESS.2019.01.006.

[95] A. Vaswani *et al.*, "Attention Is All You Need," *Adv Neural Inf Process Syst*, vol. 2017-December, pp. 5999–6009, Jun. 2017, doi: 10.48550/arxiv.1706.03762.

[96] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, doi: 10.48550/arxiv.1409.0473.

[97] IBM Technology, "What are GANs (Generative Adversarial Networks)? - YouTube." Accessed: Jan. 21, 2023. [Online]. Available: https://www.youtube.com/watch?v=TpMIssRdhco

[98] Google Trends, "Graph neural network, Convolutional neural network, Recurrent neural network, Autoencoder, Generative adversarial networks - Explore - Google Trends." Accessed: Jan. 21, 2023. [Online]. Available: https://trends.google.com/trends/explore?cat=1299&date=all&q=%2Fg%2F11nx0n3ymb,%2Fm%2F0x2dbhq,%2Fm%2F05pygp,%2Fm%2F0grsv6,%2Fg%2F11cmqkkd8d

[99] L. Wu *et al.*, "Graph Neural Networks for Natural Language Processing: A Survey," Jun. 2021, doi: 10.48550/arxiv.2106.06090.

[100] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph Convolutional Networks for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 5966–5978, Jul. 2021, doi: 10.1109/TGRS.2020.3015157.

[101] J. Johnson, A. Gupta, and L. Fei-Fei, "Image Generation from Scene Graphs," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1219–1228, Apr. 2018, doi: 10.48550/arxiv.1804.01622.

[102] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, “Scene Graph Generation by Iterative Message Passing,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 3097–3106, Jan. 2017, doi: 10.48550/arxiv.1701.02426.

[103] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst, “Geometric Deep Learning: Going beyond Euclidean data,” *IEEE Signal Process Mag*, vol. 34, no. 4, pp. 18–42, Jul. 2017, doi: 10.1109/MSP.2017.2693418.

[104] W. Fan *et al.*, “Graph neural networks for social recommendation,” *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, pp. 417–426, May 2019, doi: 10.1145/3308558.3313488.

[105] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 1, pp. 4–24, Jan. 2019, doi: 10.1109/tnnls.2020.2978386.

[106] Y. Yang, “Ensemble learning,” *temporal data mining via unsupervised ensemble learning*, pp. 35–56, 2017.

[107] N. Thomas Rincy and R. Gupta, “Ensemble learning techniques and its efficiency in machine learning: A survey,” *2nd International Conference on Data, Engineering and Applications, IDEA 2020*, Feb. 2020, doi: 10.1109/IDEA49133.2020.9170675.

[108] A. S. Iwashita and J. P. Papa, “An Overview on Concept Drift Learning,” *IEEE Access*, vol. 7, pp. 1532–1547, 2019, doi: 10.1109/ACCESS.2018.2886026.

[109] D. Berrar, P. LOPES, J. Davis, and W. Dubitzky, “The 2017 Soccer Prediction Challenge.” Accessed: Jan. 21, 2023. [Online]. Available: https://osf.io/ftuva/

[110] L. Hervert-Escobar, T. I. Matis, and N. Hernandez-Gress, “Prediction Learning Model for Soccer Matches Outcomes,” *Proceedings of the Special Session - 2018 17th Mexican International Conference on Artificial Intelligence, MICAI 2018*, pp. 63–69, Oct. 2018, doi: 10.1109/MICAI46078.2018.00018.

[111] S. Wilkens, “Sports prediction and betting models in the machine learning age: The case of tennis,” *Journal of Sports Analytics*, vol. 7, no. 2, pp. 99–117, Jan. 2021, doi: 10.3233/JSA-200463.

[112] L. Kaunitz, S. Zhong, and J. Kreiner, “Beating the bookies with their own numbers - and how the online sports betting market is rigged,” Oct. 2017, doi: 10.48550/arxiv.1710.02824.

[113] K. Odachowski and J. Grekow, “Using bookmaker odds to predict the final result of football matches,” *Lecture Notes in Computer Science (including subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7828 LNAI, pp. 196–205, 2013, doi: 10.1007/978-3-642-37343-5_20.

[114] P. Tüfekci, "Prediction of football match results in Turkish super league games," *Advances in Intelligent Systems and Computing*, vol. 427, pp. 515–526, 2016, doi: 10.1007/978-3-319-29504-6_48/COVER.

[115] R. G. Martins, A. S. Martins, L. A. Neves, L. v. Lima, E. L. Flores, and M. Z. do Nascimento, "Exploring polynomial classifier to predict match results in football championships," *Expert Syst Appl*, vol. 83, pp. 79–93, Oct. 2017, doi: 10.1016/J.ESWA.2017.04.040.

[116] N. Zaveri, U. Shah, S. Tiwari, P. Shinde, and L. K. Teli, "Prediction of football match score and decision making process," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 6, no. 2, pp. 162–165, 2018.

[117] Q. Zhang, H. Z. Xu, L. Q. Zhou, and L. Wei, "Prediction of football match results based on model fusion," *ACM International Conference Proceeding Series*, vol. Part F148152, pp. 198–202, 2019, doi: 10.1145/3319921.3319969.

[118] G. Peters and D. Pacheco, "Betting the system: Using lineups to predict football scores," Oct. 2022, doi: 10.48550/arxiv.2210.06327.

[119] N. Danisik, P. Lacko, and M. Farkas, "Football match prediction using players attributes," *DISA 2018 - IEEE World Symposium on Digital Intelligence for Systems and Machines, Proceedings*, pp. 201–206, Oct. 2018, doi: 10.1109/DISA.2018.8490613.

[120] G. Brewer, S. Demediuk, A. Drachen, F. Block, and T. Jackson, "Creating Well Calibrated and Refined Win Prediction Models," *SSRN Electronic Journal*, Mar. 2022, doi: 10.2139/SSRN.4054211.

[121] M. E. Glickman and A. C. Jones, "Rating the chess rating system," *CHANCE-BERLIN THEN NEW YORK-*, vol. 12, pp. 21–28, 1999.

[122] M. Tomasello, *Origins of human communication*. MIT press, 2010.

[123] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, Dec. 2014, doi: 10.1016/J.ASEJ.2014.04.011.

[124] G. Katz, N. Ofek, and B. Shapira, "ConSent: Context-based sentiment analysis," *Knowl Based Syst*, vol. 84, pp. 162–178, Aug. 2015, doi: 10.1016/J.KNOSYS.2015.04.009.

[125] A. Yadav and D. K. Vishwakarma, “Sentiment analysis using deep learning architectures: a review,” *Artif Intell Rev*, vol. 53, no. 6, pp. 4335–4385, Aug. 2020, doi: 10.1007/S10462-019-09794-5/TABLES/11.

[126] D. I. H. Farias and P. Rosso, “Irony, Sarcasm, and Sentiment Analysis,” *Sentiment Analysis in Social Networks*, pp. 113–128, Jan. 2017, doi: 10.1016/B978-0-12-804412-4.00007-3.

[127] T. Hu, H. Guo, H. Sun, T. V. T. Nguyen, and J. Luo, “Spice up Your Chat: The Intentions and Sentiment Effects of Using Emoji,” *Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017*, pp. 102–111, Mar. 2017, doi: 10.48550/arxiv.1703.02860.

[128] OpenAI, “Models - OpenAI API.” Accessed: Jan. 21, 2023. [Online]. Available: https://beta.openai.com/docs/models/gpt-3

[129] A. Radford *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[130] L. Floridi and M. Chiriatti, “GPT-3: Its Nature, Scope, Limits, and Consequences,” *Minds Mach (Dordr)*, vol. 30, no. 4, pp. 681–694, Dec. 2020, doi: 10.1007/S11023-020-09548-1/FIGURES/5.

[131] Google Trends, “chat gpt, Artificial intelligence, Deep learning - Explore - Google Trends.” Accessed: Jan. 21, 2023. [Online]. Available: https://trends.google.com/trends/explore?date=2022-11-15%202023-01-15&q=chat%20gpt,%2Fm%2F0mkz,%2Fm%2F0h1fn8h

[132] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, Oct. 2018, doi: 10.48550/arxiv.1810.04805.

[133] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained models for natural language processing: A survey,” *Sci China Technol Sci*, vol. 63, no. 10, pp. 1872–1897, 2020.

[134] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” Jul. 2019, doi: 10.48550/arxiv.1907.11692.

[135] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding,” *EMNLP 2018 - 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Proceedings of the 1st Workshop*, pp. 353–355, 2018, doi: 10.18653/V1/W18-5446.

[136] A. Wang *et al.*, “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems,” *Adv Neural Inf Process Syst*, vol. 32, May 2019, doi: 10.48550/arxiv.1905.00537.

[137] GlueBenchmark, “SuperGLUE Benchmark.” Accessed: Jan. 21, 2023. [Online]. Available: https://super.gluebenchmark.com/leaderboard

[138] Q. Zhong *et al.*, “Toward Efficient Language Model Pretraining and Downstream Adaptation via Self-Evolution: A Case Study on SuperGLUE,” Dec. 2022, doi: 10.48550/arxiv.2212.01853.

[139] B. Zoph *et al.*, “ST-MoE: Designing Stable and Transferable Sparse Expert Models,” Feb. 2022, doi: 10.48550/arxiv.2202.08906.

[140] P. Bajaj *et al.*, “METRO: Efficient Denoising Pretraining of Large Scale Autoencoding Language Models with Model Generated Signals,” Apr. 2022, doi: 10.48550/arxiv.2204.06644.

[141] PaddlePaddle, “GitHub - PaddlePaddle/ERNIE: Official implementations for various pre-training models of ERNIE-family, covering topics of Language Understanding & Generation, Multimodal Understanding & Generation, and beyond.” Accessed: Jan. 21, 2023. [Online]. Available: https://github.com/PaddlePaddle/ERNIE

[142] Y. Sun *et al.*, “ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation,” Jul. 2021, doi: 10.48550/arxiv.2107.02137.

[143] A. Chowdhery *et al.*, “PaLM: Scaling Language Modeling with Pathways,” Apr. 2022, doi: 10.48550/arxiv.2204.02311.

[144] Z. Wang, A. W. Yu, O. Firat, and Y. Cao, “Towards Zero-Label Language Learning,” Sep. 2021, doi: 10.48550/arxiv.2109.09193.

[145] Microsoft, “GitHub - microsoft/DeBERTa: The implementation of DeBERTa.” Accessed: Jan. 21, 2023. [Online]. Available: https://github.com/microsoft/DeBERTa

[146] J. H. Oh, R. Iida, J. Kloetzer, and K. Torisawa, “BERTAC: Enhancing Transformer-based Language Models with Adversarially Pretrained Convolutional Neural Networks,” *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 2103–2115, 2021, doi: 10.18653/V1/2021.ACL-LONG.164.

[147] Google Research, “GitHub - google-research/text-to-text-transfer-transformer: Code for the paper ‘Exploring the Limits of Transfer Learning with a Unified Text-

to-Text Transformer.’” Accessed: Jan. 21, 2023. [Online]. Available: https://github.com/google-research/text-to-text-transfer-transformer

[148] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer.,” *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[149] Hugging Face, “roberta-large · Hugging Face.” Accessed: Jan. 21, 2023. [Online]. Available: https://huggingface.co/roberta-large

[150] Hugging Face, “bert-large-uncased · Hugging Face.” Accessed: Jan. 21, 2023. [Online]. Available: https://huggingface.co/bert-large-uncased

[151] OpenAI, “GPT-4 Technical Report,” Mar. 2023, Accessed: Aug. 22, 2023. [Online]. Available: https://arxiv.org/abs/2303.08774v3

[152] P. Bellow, “Gpt-3.5-turbo-0613: Function calling, 16k context window, and lower prices - API - OpenAI Developer Forum.” Accessed: Aug. 22, 2023. [Online]. Available: https://community.openai.com/t/gpt-3-5-turbo-0613-function-calling-16k-context-window-and-lower-prices/263263

[153] “Google AI updates: Bard and new AI features in Search.” Accessed: Aug. 22, 2023. [Online]. Available: https://blog.google/technology/ai/bard-google-ai-search-updates/

[154] R. Anil *et al.*, “PaLM 2 Technical Report,” May 2023, Accessed: Aug. 22, 2023. [Online]. Available: http://arxiv.org/abs/2305.10403

[155] H. Touvron *et al.*, “LLaMA: Open and Efficient Foundation Language Models,” Feb. 2023, Accessed: Aug. 22, 2023. [Online]. Available: https://arxiv.org/abs/2302.13971v1

[156] “Llama 2 - Meta AI.” Accessed: Aug. 22, 2023. [Online]. Available: https://ai.meta.com/llama/

[157] “Meta’s powerful AI language model has leaked online — what happens now? - The Verge.” Accessed: Aug. 22, 2023. [Online]. Available: https://www.theverge.com/2023/3/8/23629362/meta-ai-language-model-llama-leak-online-misuse

[158] R. Taori *et al.*, “Stanford CRFM.” Accessed: Aug. 22, 2023. [Online]. Available: https://crfm.stanford.edu/2023/03/13/alpaca.html

[159] “Chatbot Arena Leaderboard - a Hugging Face Space by lmsys.” Accessed: Aug. 22, 2023. [Online]. Available: https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

[160] “Anthropic \ Introducing Claude.” Accessed: Aug. 22, 2023. [Online]. Available: https://www.anthropic.com/index/introducing-claude

[161] Vicuna Team, “Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality | LMSYS Org.” Accessed: Aug. 22, 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[162] LMSYS, “lmsys/fastchat-t5-3b-v1.0 · Hugging Face.” Accessed: Aug. 22, 2023. [Online]. Available: https://huggingface.co/lmsys/fastchat-t5-3b-v1.0

[163] Web Of Science, “Web of Science Master Journal List - Search.” Accessed: Jan. 21, 2023. [Online]. Available: https://mjl.clarivate.com/search-results

[164] A. Feddersen, B. R. Humphreys, and B. P. Soebbing, “Sentiment Bias in National Basketball Association Betting,” *J Sports Econom*, vol. 19, no. 4, pp. 455–472, May 2018, doi: 10.1177/1527002516656726.

[165] E. Wheatcroft, “Profiting from overreaction in soccer betting odds,” *J Quant Anal Sports*, vol. 16, no. 3, pp. 193–209, Sep. 2020, doi: 10.1515/JQAS-2019-0009/MACHINEREADABLECITATION/RIS.

[166] D. Purdum, “Gambling - Should bookmakers in a growing U.S. legal betting market be allowed to ban bettors?,” 2019. Accessed: Dec. 23, 2022. [Online]. Available: https://www.espn.com/chalk/story/_/id/24425026/gambling-bookmakers-growing-us-legal-betting-market-allowed-ban-bettors

[167] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, and F. Fdez-Riverola, “Web scraping technologies in an API world,” *Brief Bioinform*, vol. 15, no. 5, pp. 788–797, Sep. 2014, doi: 10.1093/BIB/BBT026.

[168] R. Dewan, M. Freimer, and J. Zhang, “Managing Web sites for profitability: Balancing content and advertising,” *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2002-January, pp. 2340–2347, 2002, doi: 10.1109/HICSS.2002.994170.

[169] The European Parliament and Council of the European Union, “Regulation (EU) 2016/679 - General Data Protection Regulation,” *Official Journal of the European Union*, 2016.

[170] F. Zhou and Y. Wang, “Exploring The Role of Web Crawler and Anti-Crawler Technology in Big Data Era,” *Proceedings - 2022 11th International Conference of Information and Communication Technology, ICTech 2022*, pp. 316–319, 2022, doi: 10.1109/ICTECH55460.2022.00070.

[171] A. Gaurav, B. B. Gupta, W. Alhalabi, A. Visvizi, and Y. Asiri, “A comprehensive survey on DDoS attacks on various intelligent systems and it’s defense techniques,” *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 11407–11431, Dec. 2022, doi: 10.1002/INT.23048.

[172] B. Green and S. Seshadri, *AngularJS*. “ O’Reilly Media, Inc.,” 2013.

[173] S. Aggarwal, “Modern web-development using reactjs,” *International Journal of Recent Research Aspects*, vol. 5, no. 1, pp. 133–137, 2018.

[174] VueJS, “Vue.js - The Progressive JavaScript Framework | Vue.js.” Accessed: Jan. 22, 2023. [Online]. Available: https://vuejs.org/

[175] Javascript, “JavaScript.com.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.javascript.com/

[176] PhantomJS, “PhantomJS - Scriptable Headless Browser.” Accessed: Jan. 22, 2023. [Online]. Available: https://phantomjs.org/

[177] M. Machado, “CS:GO Professional Matches | Kaggle.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.kaggle.com/datasets/mateusdmachado/csgo-professional-matches?select=results.csv

[178] G. Salles, “CS:GO Professional Matches | Kaggle.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.kaggle.com/datasets/gabrieltardochi/counter-strike-global-offensive-matches

[179] Artem, “CS:GO match history | Kaggle.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.kaggle.com/datasets/artem1337/csgo-matches

[180] PandaScore, “PandaScore - Esports Data & Odds API for LoL, CS:GO, Dota 2 and more.” Accessed: Jan. 22, 2023. [Online]. Available: https://pandascore.co/

[181] GameScoreKeeper, “GameScorekeeper – Esports B2B Solutions – GameScorekeeper Blog.” Accessed: Jan. 22, 2023. [Online]. Available: https://gamescorekeeper.com/

[182] HLTV, “Introducing Rating 2.0.” Accessed: May 04, 2023. [Online]. Available: https://www.hltv.org/news/20695/introducing-rating-20

[183] Esports.net, “CSGO Odds » Compare the best CS:GO odds for your bets.” Accessed: Jan. 22, 2023. [Online]. Available: https://web.archive.org/web/20221130005111/https://www.esports.net/betting/csgo/odds/

[184] Odds Portal, “Odds Portal: Odds Comparison, Sports Betting Odds.” Accessed: Jan. 22, 2023. [Online]. Available: https://web.archive.org/web/20230114232010/https://www.oddsportal.com/

[185] Strafe, “▷ Latest CSGO Odds | CS:GO Betting Odds for today [Live].” Accessed: Jan. 22, 2023. [Online]. Available: https://www.strafe.com/esports-betting/odds/csgo/

[186] Python, “Welcome to Python.org.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.python.org/

[187] Crummy, “Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[188] MongoDB, “MongoDB Atlas: Cloud Document Database | MongoDB.” Accessed: Jan. 22, 2023. [Online]. Available: https://www.mongodb.com/

[189] S. Raschka, *Python machine learning*. Packt publishing ltd, 2015.

[190] Jupyter, “Intro to Jupyter.” Accessed: Jan. 22, 2023. [Online]. Available: https://jupyter.org/try-jupyter/retro/notebooks/?path=notebooks/Intro.ipynb

[191] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature 2020 585:7825*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[192] J. Reback *et al.*, “pandas-dev/pandas: Pandas 1.4.3,” Jun. 2022, doi: 10.5281/ZENODO.6702671.

[193] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. “ O’Reilly Media, Inc.,” 2022.

[194] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Comput Sci Eng*, vol. 9, no. 03, pp. 90–95, 2007.

[195] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[196] “Intel® Extension for Scikit-learn* — Intel(R) Extension for Scikit-learn* 2023.2 documentation.” Accessed: Aug. 20, 2023. [Online]. Available: https://intel.github.io/scikit-learn-intelex/

[197] Amazon Web Services, “Amazon SageMaker Pricing - Machine Learning - Amazon Web Services.” Accessed: Sep. 28, 2023. [Online]. Available: https://aws.amazon.com/sagemaker/pricing/?nc1=h_ls

[198] F. Harrell, “Statistical Thinking - Classification vs. Prediction.” Accessed: May 03, 2023. [Online]. Available: https://hbiostat.org/blog/post/classification/index.html

[199] F. Harrell, “Statistical Thinking - Damage Caused by Classification Accuracy and Other Discontinuous Improper Accuracy Scoring Rules.” Accessed: May 03, 2023. [Online]. Available: https://hbiostat.org/blog/post/class-damage/index.html

[200] T. Hastie, J. Friedman, and R. Tibshirani, “The Elements of Statistical Learning,” 2001, doi: 10.1007/978-0-387-21606-5.

[201] G. W. Brier, “Verification of forecasts expressed in terms of probability,” *Mon Weather Rev*, vol. 78, no. 1, pp. 1–3, 1950.

[202] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, “Obtaining Well Calibrated Probabilities Using Bayesian Binning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, pp. 2901–2907, Feb. 2015, doi: 10.1609/AAAI.V29I1.9602.

[203] M. Loi and M. Christen, “How to include ethics in machine learning research,” *Ercim News*, vol. 116, no. 3, p. 5, 2019.

[204] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, “Machine Learning Security: Threats, Countermeasures, and Evaluations,” *IEEE Access*, vol. 8, pp. 74720–74742, 2020, doi: 10.1109/ACCESS.2020.2987435.

[205] C. Novelli, M. Taddeo, and L. Floridi, “Accountability in artificial intelligence: what it is and how it works,” *AI Soc*, vol. 1, pp. 1–12, Feb. 2023, doi: 10.1007/S00146-023-01635-Y/FIGURES/1.

[206] Foundation for Science and Technology, “ETHICS SELF-ASSESSMENT GUIDE STIMULUS OF SCIENTIFIC EMPLOYMENT, INDIVIDUAL SUPPORT (CEECIND) 5th EDITION.”

[207] V. Krotov and L. Silva, “Legality and ethics of web scraping,” 2018.

[208] Scikit Learn, “sklearn.preprocessing.MultiLabelBinarizer — scikit-learn 1.2.0 documentation.” Accessed: Jan. 22, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html

[209] J. H. Steiger, “Tests for comparing elements of a correlation matrix,” *Psychol Bull*, vol. 87, no. 2, pp. 245–251, Mar. 1980, doi: 10.1037/0033-2909.87.2.245.

[210] Rooney, “Correlation (Pearson, Spearman, and Kendall) | Kaggle.” Accessed: Aug. 14, 2023. [Online]. Available: https://www.kaggle.com/code/kiyoung1027/correlation-pearson-spearman-and-kendall

[211] Mordekhai, “Pearson, Spearman, and Kendall-Tau Correlations: What are the Differences? | Python in Plain English.” Accessed: Aug. 14, 2023. [Online]. Available: https://python.plainenglish.io/pearson-spearman-and-kendall-tau-correlations-what-are-the-differences-d0a3963b4c94

[212] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, Jan. 2013, Accessed: Sep. 03, 2023. [Online]. Available: https://arxiv.org/abs/1301.3781v3

[213] Q. Le and T. Mikolov, “Distributed Representations of Sentences and Documents,” *31st International Conference on Machine Learning, ICML 2014*, vol. 4, pp. 2931–2939, May 2014, Accessed: Sep. 03, 2023. [Online]. Available: https://arxiv.org/abs/1405.4053v2

[214] A. Ziegler and J. Berryman, “A developer’s guide to prompt engineering and LLMs - The GitHub Blog.” Accessed: Sep. 04, 2023. [Online]. Available: https://github.blog/2023-07-17-prompt-engineering-guide-generative-ai-llms/

[215] J. Cook, “AI Prompt Engineers Earn $300k Salaries: Here’s How To Learn The Skill For Free.” Accessed: Sep. 04, 2023. [Online]. Available: https://www.forbes.com/sites/jodiecook/2023/07/12/ai-prompt-engineers-earn-300k-salaries-heres-how-to-learn-the-skill-for-free/

[216] OpenAI, “GPT best practices - OpenAI API.” Accessed: Sep. 04, 2023. [Online]. Available: https://platform.openai.com/docs/guides/gpt-best-practices

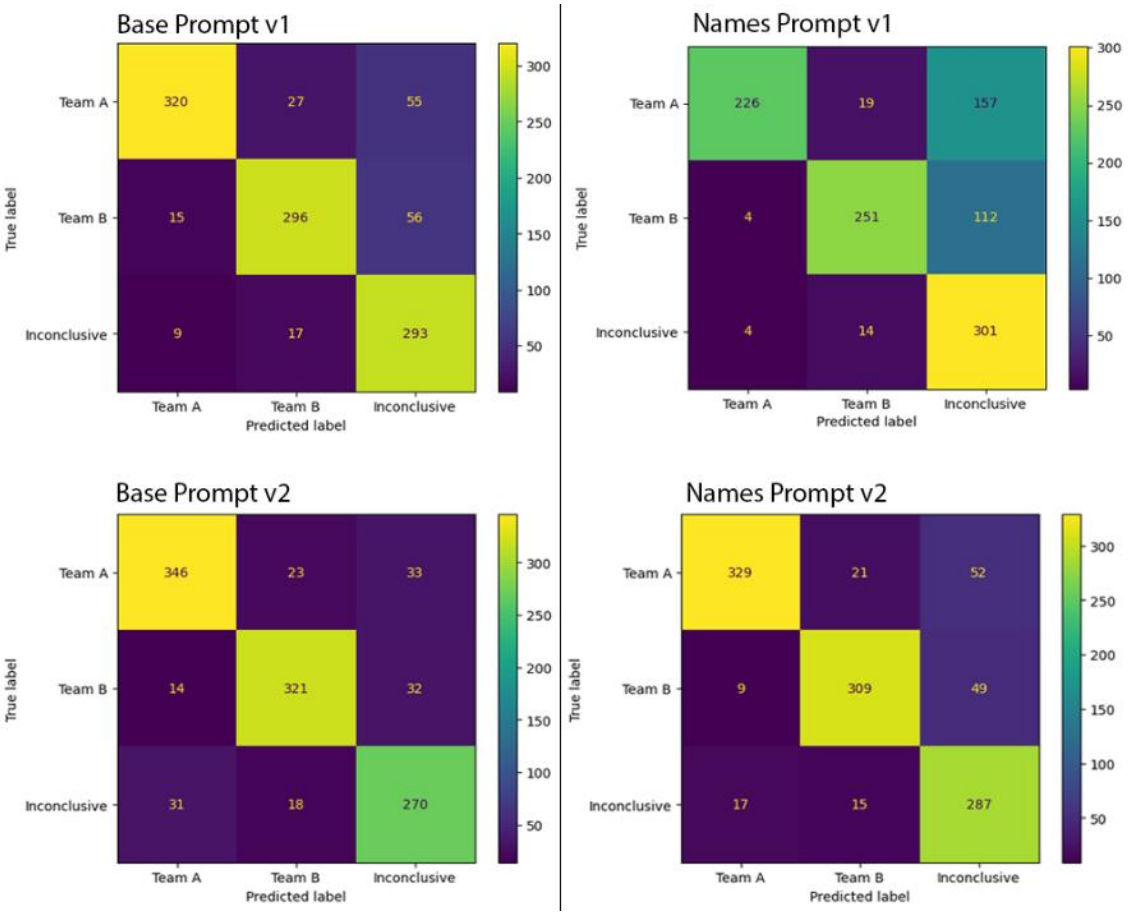
Attachment A Tests of team and player representations using Logistic Regression

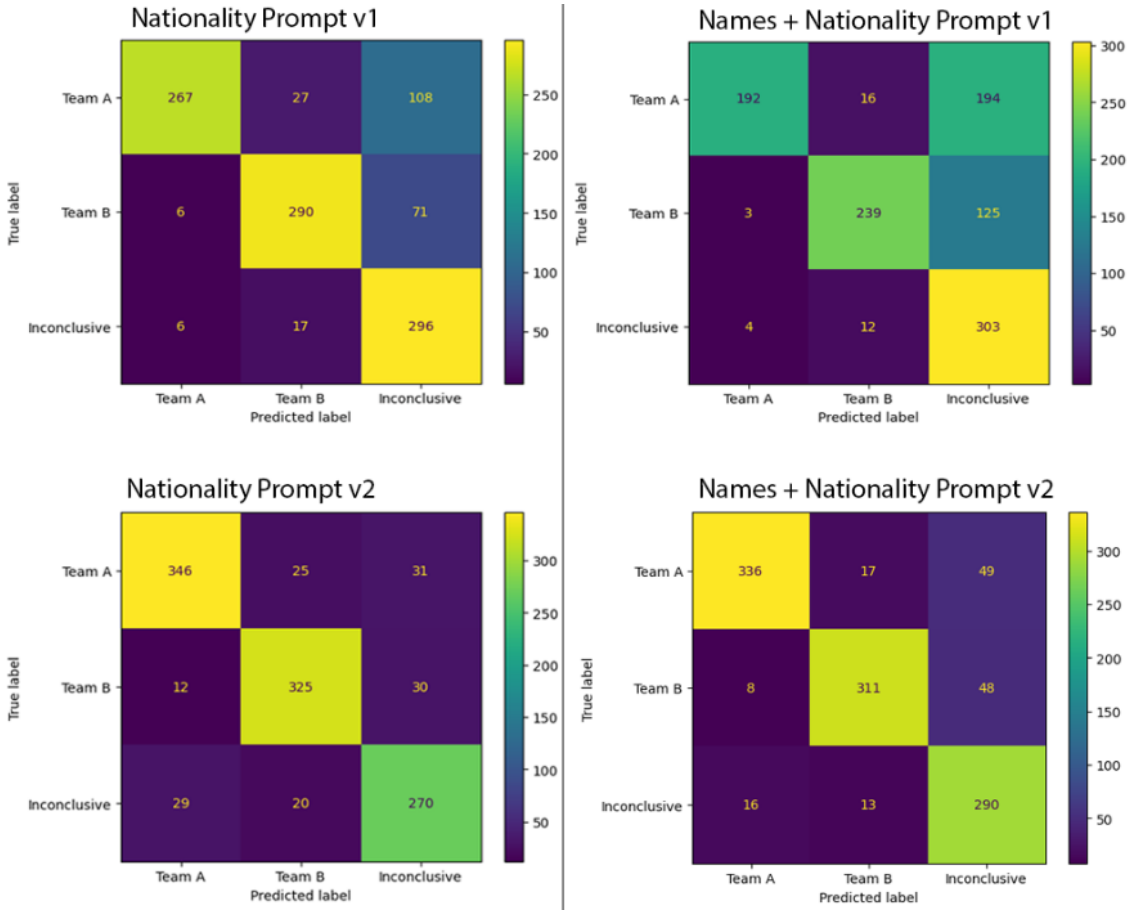
Data representation			Accuracy (%)	Brier	Log Loss	Calibration	
						ECE (%)	empty
Team Rank Match							
Encoding	Rank|Maps	Date					
Match Team Binary Encoding	None	None	62.10	0.229	0.651	3.45	2
		Feature	62.30	0.228	0.649	3.22	2
		Weight	63.26	0.224	0.639	2.39	2
	Rank	None	63.14	0.225	0.640	3.95	0
		Feature	63.26	0.225	0.640	3.84	0
		Weight	63.66	0.221	0.632	4.25	0
	Maps	None	62.32	0.228	0.649	3.37	2
		Feature	62.24	0.228	0.649	3.41	2
		Weight	63.24	0.224	0.639	2.48	2
	Both	None	63.14	0.225	0.640	3.95	0
		Feature	63.22	0.225	0.640	3.82	0
		Weight	63.80	0.221	0.632	3.93	0
Match Player Binary Encoding	None	None	64.48	0.224	0.652	8.50	0
		Feature	64.66	0.223	0.648	8.23	0
		Weight	64.20	0.220	0.637	7.33	0
	Rank	None	64.60	0.223	0.648	8.14	0
		Feature	64.60	0.223	0.648	8.18	0
		Weight	64.06	0.220	0.638	7.53	0
	Maps	None	64.48	0.223	0.648	8.13	0
		Feature	64.50	0.223	0.648	8.09	0
		Weight	64.14	0.220	0.637	7.45	0
	Both	None	64.56	0.223	0.648	8.12	0
		Feature	64.54	0.223	0.648	8.10	0
		Weight	64.22	0.220	0.638	7.36	0

Attachment B Tests of different player performance metrics and aggregation

Representation	Accuracy (%)	Brier	LogLoss	ECE (%)	Empty Bins
Team Binary Encoding	62.1	0.229	0.651	2.59	2
Player Binary Encoding	64.48	0.224	0.652	7.76	0
ADRs_AVG_03	65.42	0.220	0.635	5.45	0
ADRs_AVG_11	65.2	0.218	0.631	5.61	0
ADRs_AVG_19	65.3	0.218	0.631	5.42	0
ADRs_WAVG_03	65.58	0.220	0.635	5.55	0
ADRs_WAVG_11	64.84	0.218	0.631	5.71	0
ADRs_WAVG_19	64.8	0.218	0.631	5.80	0
ADRs_WAVG_OppS_03	65.36	0.220	0.636	5.51	0
ADRs_WAVG_OppS_11	64.32	0.219	0.633	6.46	0
ADRs_WAVG_OppS_19	65.4	0.220	0.636	5.83	0
K_D_A_AVG_03	64.45	0.221	0.636	6.03	0
K_D_A_AVG_11	65.09	0.219	0.631	5.61	0
K_D_A_AVG_19	64.95	0.219	0.631	5.81	0
K_D_A_WAVG_03	63.77	0.223	0.64	5.89	0
K_D_A_WAVG_11	64.55	0.221	0.635	5.92	0
K_D_A_WAVG_19	64.67	0.22	0.634	5.86	0
K_D_A_WAVG_OppS_03	64.67	0.221	0.637	5.97	0
K_D_A_WAVG_OppS_11	64.85	0.22	0.634	6.98	0
K_D_A_WAVG_OppS_19	64.29	0.22	0.635	5.86	0
RADR_AVG_03	65.8	0.217	0.625	2.55	2
RADR_AVG_11	66.04	0.214	0.618	1.37	2
RADR_AVG_19	66.14	0.215	0.618	1.63	2
RADR_WAVG_03	65.46	0.219	0.628	1.95	2
RADR_WAVG_11	66.36	0.215	0.618	1.60	2
RADR_WAVG_19	66.06	0.214	0.618	1.31	2
RADR_WAVG_OppS_03	65.48	0.218	0.626	2.47	2
RADR_WAVG_OppS_11	65.32	0.217	0.624	1.57	2
RADR_WAVG_OppS_19	64.96	0.218	0.625	1.31	2
Ratings_AVG_03	65.7	0.218	0.629	4.33	0
Ratings_AVG_11	65.36	0.217	0.625	4.07	0
Ratings_AVG_19	65.42	0.217	0.625	4.12	0
Ratings_WAVG_03	65.6	0.219	0.631	4.19	0
Ratings_WAVG_11	65.9	0.217	0.625	4.12	0
Ratings_WAVG_19	65.56	0.217	0.625	4.04	0
Ratings_WAVG_OppS_03	65.44	0.219	0.630	3.56	0
Ratings_WAVG_OppS_11	64.6	0.219	0.629	4.65	0
Ratings_WAVG_OppS_19	65.2	0.219	0.631	4.47	0

Attachment C Sentiment Analysis Confusion Matrices




Nationality Prompt v1
Team A
Team B
Inconclusive
True label
267
27
108
6
290
71
6
17
296
Team A
Team B
Inconclusive
Predicted label
250
200
150
100
50
Names + Nationality Prompt v1
Team A
Team B
Inconclusive
True label
192
16
194
3
239
125
4
12
303
Team A
Team B
Inconclusive
Predicted label
300
250
200
150
100
50
Nationality Prompt v2
Team A
Team B
Inconclusive
True label
346
25
31
12
325
30
29
20
270
Team A
Team B
Inconclusive
Predicted label
300
250
200
150
100
50
Names + Nationality Prompt v2
Team A
Team B
Inconclusive
True label
336
17
49
8
311
48
16
13
290
Team A
Team B
Inconclusive
Predicted label
300
250
200
150
100
50