

# Rating Algorithms and the effect machine learning could have on their accuracy

A look into how machine learning implementation could potentially make future and current rating/matchmaking algorithms work more accurately and efficiently.

# Table of Contents

<b>Abstract</b> .....	3
<b>1 - Introduction</b> .....	4
1.1 – Overall Objectives .....	4
1.1.1 – Matchmaking Algorithms in Games Applications .....	4
1.1.2 – Matchmaking Algorithms in Dating Applications.....	4
1.1.3 – Elo Rating Systems .....	5
1.1.4 – Machine Learning Implementations .....	5
1.2 – Project Framework.....	5
<b>2 - Background Research</b> .....	7
2.1 - Programming Language Analysis .....	7
2.1.1 – Python .....	7
2.1.2 – Java.....	7
2.1.3 – C++.....	8
2.1.4 – Overall Decision and Conclusion .....	8
2.2 – The ELO System.....	11
2.2.1 - ELO System Background .....	11
2.2.2– Math behind the ELO system .....	12
2.2.3 – Gaps in the ELO System.....	14
2.2.4 – Modern Adaptations of the ELO system .....	14
2.3 – TrueSkill.....	15
2.3.1 – TrueMatch Matchmaking.....	15
2.3.2 – The TrueSkill Algorithm .....	15
2.3.3 – Advantages of TrueSkill .....	17
2.3.4 – Disadvantages of TrueSkill .....	17
2.3.5 – TrueSkill2.....	18
2.4 – Uses in Dating Applications.....	18
2.4.1 – Tinder and ELO .....	18
2.4.2 – Hinge and Gale-Shipley Algorithm .....	18
2.5 – The State of Online Matchmaking .....	19
<b>3 - Methodology</b> .....	20
<b>4 - Product Development</b> .....	22
4.1 – Choosing a dataset .....	22
4.2 – Analysing the dataset.....	22

<b>5 - Evaluation of Research/Product</b>	26
<b>6 - Conclusions</b>	27
<b>References and Bibliography</b>	28
<b>Appendices</b>	32
Appendix A	32
A.1 – URL of sourced Dataset	32
A.2	32
A.3 – Output from A.2	32
A.4	32
A.5	33

## Abstract

# 1 - Introduction

Since online applications have developed over time, there has been a rise in demand of accurate yet optimised matchmaking and sorting algorithms. These algorithms have been incorporated into many different modern-day mediums however, more commonly seen and most importantly utilised in online video games and dating applications. The nature of these two kinds of mediums is the reason why matchmaking algorithms need to be not only be accurate but also efficient in the way it runs, as the sole purpose is to find other users who fit a rubric that is currently set. Online dating has seen a rise in popularity in the last 5 years and will only continue to grow, so too will the algorithms used in its system, this is where artificial intelligence and machine learning could pave the way for their future.

Artificial intelligence has also been on the rise in recent years, where machine learning has been an integral part of many different systems so it is imperative, we learn new ways in which we can operate such a powerful tool in systems that could benefit from it. While there have been studies in the last two decades about machine learning being implemented into matchmaking algorithms, mainly in the Gaming industry, there hasn't been any indication that it is the way forward for said algorithms. However, the TrueSkill matchmaking algorithm currently being used and developed by Microsoft and its subsidiary games companies has been one of the more notable uses of machine learning, could be the forefront of machine learnt matching algorithms.

The main focus of this dissertation arose when trying to understand how modern-day gaming and dating applications would find the best matches for their user, this has been a main concern in the gaming community on the validity and accuracy of modern-day online games' matchmaking, faulty matchmaking can create an awful user experience so understanding the algorithms is a necessary step into the future of matchmaking. The aim of this project is to investigate current and past matchmaking algorithms, along with the math behind it, being used in both dating applications and online games, investigating the effect machine learning has on said algorithms and analyse data produced through an implementation of the TrueSkill algorithm where machine learning techniques are implemented as a way to demonstrate how the TrueSkill algorithm can be applied to real-world applications.

## 1.1 – Overall Objectives

Here are the main aims and objectives that I have identified for this project:

### 1.1.1 – Matchmaking Algorithms in Games Applications

Various algorithms and tried and tested based on the application it will be applied to, in the case we will be looking at in this project, the TrueSkill algorithm has been chosen by Microsoft as their flag runner approach to online matchmaking, where user experience is heavily reliant on the fairness of skill distribution in their lobby. With developments being made in recent years to upgrade to Trueskill2, there is still a question mark over whether the use of machine learning could increase accuracy and efficiency. A simple implementation of TrueSkill in python along with analysis of its matching distribution will be a main part of this project.

### 1.1.2 – Matchmaking Algorithms in Dating Applications

While gaming applications may use matchmaking algorithms with a large set of users in mind, dating applications are the opposite, where only a pair of two users are matched at any given time, while this is the case, most dating applications will have extra features where their algorithm is given room

to work. Tests will be conducted to validate whether or not the sample size affects the optimisation of the algorithms and also to test the TrueSkill library in a different environment.

### 1.1.3 – Elo Rating Systems

Matchmaking systems must have a way of identifying parameters values for their users in order to find others who are as close a match to those parameters as possible. With this in mind, the first ever 'rating system' called the Elo system was created specifically for matching pairs of chess players based on their 'ELO rating'. This would then go on to be the basis of how most if not all matchmaking systems would manage their users. While ELO ratings are still in use to this day, I will look at the math behind the original ELO system, how modern-day games have adapted it for their own specific use and if the ELO system's accuracy fairs against other, more modern techniques such as the TrueStrike algorithm used by Microsoft. This will hopefully be demonstrated using mathematical distribution graphs using the Matplotlib python library.

### 1.1.4 – Machine Learning Implementations

Artificial Intelligence has come a long way since its early iterations and is now seen deployed a wide variety of systems. When it comes to matchmaking and sorting algorithms, its use case on paper should provide a lot of utility, a look into past applications and the evaluation of those implementations will be investigated and conclude the validity when combined with said algorithms. This will help gain insight on past testing and implementation techniques which can then be utilised in this project.

## 1.2 – Project Framework

To achieve the results outlined for this project, there will be x key stages that should help develop a product that will be able to fulfil the needs required:

1. The past literature research stage will help develop understanding of previously used algorithms and techniques focusing on the development of the ELO rating systems in League of Legends, the evolution and influence it had on the TrueSkill algorithm and the way single pair matchmaking is performed on dating applications. A focus will be put on the math and any past test results of the ELO and TrueSkill algorithms along with any related information that could prove useful in comparison to test results in this project.
2. The product design stage involves consideration of which programming language to use as part of the product along with any additional software and libraries. Their positives and negatives will be a large part of the decision-making progress as well as the relevancy of its use in this project context. The way in which they will be utilized will also be outlined and a basic rubric of success will be developed as part of this stage.
3. The development stage will be the product production using the aforementioned techniques chosen in the design stage, there will be details about how parts of the project were implemented and noting down any relevant changes to the initial project design due to certain circumstances, if any.
4. The testing stage will complement the development stage as it will record results from testing methods used on the data and mathematical comparisons that will be made on pre-

existing data and data that will be produced in this project and will provide some insight on the outcome of these tests.

5. The final stage will be the critical evaluation and postproduction reflection, this will discuss the project as a whole, whether it was successful, analysis of future additions and improvements alongside overarching comments from the start to the end of this project.

## 2 - Background Research

This section on the report will focus on the key various facets that relate to the overall aim of this project, this will heavily impact the development of the project.

### 2.1 - Programming Language Analysis

Here I will analyse the different available programming languages that could be used as part of the product of this project, a comparison will be made on their available supplementary machine learning libraries and how well their strengths fit the purpose of this project.

#### 2.1.1 – Python

Python, first created in 1991 by Guido van Rossum, has been a long-standing giant in the world of programming, with its beginner friendly approach and multi-paradigm structure, there is no surprise that in the October of 2021, Python topped the TIOBE index, an industry renowned company that tracks the quality of software (About | TIOBE - the Software Quality Company, 2022). While initial uses of Python included web development and mathematics, It has since been the one of the main programming languages that house machine learning integration with libraries that can cater to a broad set of use cases, such as TensorFlow, Keras and PyTorch (Costa, 2020).

The language contains a variety of built-in libraries that could be useful in this project, specifically the NumPy library, which is used for scientific calculations, Panda who hosts high-performance structures and data analysis tools which can improve project performance speed, and finally Scikit-learn which provides tools for data mining and analysis, optimizing Python's already top of the line machine learning usability (*Pros and Cons of Python in Machine Learning* | Zarantech, 2018). In terms of our specific use-case, being matchmaking algorithms with Machine Learning implementation, it is stated that Python has been the lead programming language for machine learning where 60% of developers are prioritizing it for their development due to it being easy to learn (*Top 5 Programming Languages and Their Libraries for Machine Learning in 2020* - GeeksforGeeks, 2020). With this in mind, it is clear as to why it is the language I am leaning to heavily for in this project, however this does not mean that the other languages do not have their merits too.

#### 2.1.2 – Java

Java has been side by side with other top programming languages including the aforementioned Python, as being one of the best and most popular. Unlike Python, it does not have a multi-faceted paradigm structure but instead object-oriented, however is not considered a purely object-oriented language as it does support primitive data types. Java is used in all kinds of mediums, including but not limited to mobile applications, desktop applications, web applications and client-server applications (*Java Programming Language* - GeeksforGeeks, 2014).

Java also hosts a number of machine learning libraries such as Java-ML, an open-source framework that provides a variety of machine learning algorithms, RapidMiner, a platform designed for data science providing the use of machine learning algorithms through GUI and the Java API along with extensive documentation and Weka, a program which holds a collection of machine learning algorithms for data mining tasks and data visualisation (baeldung, 2017b).



When comparing Java and Python there seems to be a common understanding in the AI industry that Java runs faster than Python due to it being a compiled language whereas the latter is an interpreted language, therefore if speed of execution is vital when deciding which language to choose, Java holds the current advantage (*Which Is Better for AI: Java or Python? A Breakdown of Our Top Pick*, 2021).

### 2.1.3 – C++

C++ is high level programming language that is object-oriented with fast execution times in comparison to most other top-level languages due it being very close to machine language (*Introduction to Machine Learning Using C++*, 2021). Mainly used in the industry to create high performance applications due to its cross-platform capability, it being the main programming language used in Unreal Engine, a games development engine. Originally created as an extension of the C language, it has since then grown to become one of the most used programming languages across a variety of industries (*C++ Introduction*, 2014).

While not being the most popular machine learning programming language, it still hosts a number of useful libraries to help developers, including but not limited to C++ Boost Library, a powerful library used for big maths operations, ML pack C++ Library, a small but scalable machine learning library, TensorFlow from Google AI, a popular deep learning library developed by Google with an assortment of tools and resources to help deploy machine learning powered applications, Microsoft Cognitive Toolkit (CNTK), a unified deep learning toolkit which is designed to help translate neural networks through directed graphs, and many more helpful libraries (Chatterjee, 2020). Even with such powerful libraries to support developers looking for machine learning use-cases, a study by (Nation, 2017) shows that still only 20% of machine learning developers and data scientists prioritising it and holds only 44% of usage in comparison to Python with a usage rate of 58%. Personally, for this project, C++ will be the least likely programming language that I will use due to its moderate level of machine learning programming knowledge required when compared to Python's ease of use.

### 2.1.4 – Overall Decision and Conclusion

To clearly state each programming language's strengths and weaknesses in terms of machine learning, I have constructed a table to make comparison between them digestible, the information used in this table has been taken from various academic sources and online articles which will be referenced. *Table [1]*

Table [1]: A table summarising the advantages and disadvantages of three potential programming languages to be used in this project. Sources: (Which Is Better for AI: Java or Python? A Breakdown of Our Top Pick, 2021), (Joy, 2019), (Advantages and Disadvantages of C++ | Make Your next Move! - DataFlair, 2019)

Programming Language	Advantages	Disadvantages
Python	<ul style="list-style-type: none"> <li>- Easy to pick up and learn</li> <li>- Python has a TrueSkill library package built into its installation</li> <li>- Able to use Jupyter Lab for ease of ML implementation</li> <li>- Less coding necessary to implement solutions</li> <li>- A strong ecosystem of supplementary libraries</li> <li>- Readability of code is clearer</li> <li>- Good Visualisation of data with the use of libraries like Matplotlib</li> </ul>	<ul style="list-style-type: none"> <li>- Dynamically typed language so can come across design errors.</li> <li>- Slower runtime than Java</li> <li>- High Memory Consumption</li> </ul>
Java	<ul style="list-style-type: none"> <li>- Rich machine learning library to utilise</li> <li>- Use of rapid development tools are an option</li> <li>- Last In First Out (LIFO) stack allocation which help the management of data</li> <li>- External programs such as Weka for Machine Learning implementation</li> <li>- Fast runtime in comparison to other languages</li> </ul>	<ul style="list-style-type: none"> <li>- High performance cost, Java uses up more memory and is slower than compiled languages like C++</li> <li>- Complex code: Java code is notoriously long and difficult to navigate coherently</li> </ul>
C++	<ul style="list-style-type: none"> <li>- Portability, user can run on multiple platforms</li> <li>- An OO (Object oriented) language so code reusability is present</li> </ul>	<ul style="list-style-type: none"> <li>- Use of pointers can be relatively complicated to get to grips with</li> <li>- No built-in garbage collector so it is up to the user to manage their memory</li> </ul>

	<ul style="list-style-type: none"> <li>- Multi-paradigm, can also be used procedurally</li> <li>- Large amount of community driven documentation</li> <li>- Proven library of Machine Learning assets to use</li> </ul>	<ul style="list-style-type: none"> <li>- Harder code readability when compared to Python</li> </ul>
--	---	---

Python provides us with a lot of useful features in terms of this project's context, for example, the inclusion of a pre-built matchmaking library, TrueSkill, the use of libraries like matplotlib that helps display data and its ease of implementation. While the slow runtime speed and heavy memory consumption is a cost I may have to deal with, the upsides heavily outweigh its cons, while the product could do with better optimization, it is not a necessity and is not part of what will make the project a success.

On the other hand, Java brings the fast runtime execution alongside a consortium of machine learning libraries, although not pre-built like Python's TrueSkill, the use of Weka and other machine learning applications could be of great use when analysing our data. Similar to Python, Java's high memory consumption could prove costly depending on the size of our data and it's at times, messy code could prove difficult later down the line.

C++ offers the most portability out of the three, being able to be used across a wider range of platforms. Its vast community driven documentation with its library of Machine Learning assets sound appealing, however its heavy reliance on its use of pointers coupled with the user having to handle garbage collection sounds much more of a hindrance than what positives the language can offer.

Choosing a programming language for this project is an important step in its completion. Each of the aforementioned languages will bring certain strengths forward, for example, C++ being multi-paradigm allows me to choose how to code the product without being forced to go down the object-oriented route, with Java, the faster runtimes could be significant enough to be chosen over the other two along with programs like Weka – in which I have past experience with, make Java a serious contender. However, Python and its pre-integrated matchmaking package, TrueSkill, ends up making this an easy choice for me as TrueSkill is a vastly used algorithm in the industry, primarily utilised by Microsoft games so implementation will be more straightforward, and installation will be less of a hassle.

## 2.2 – The ELO System

In this section I will go over research on past matchmaking paradigms that have been used in previous mediums, most commonly in the games industry, primarily the ELO system which is the foundation that the TrueSkill algorithm was built on.

### 2.2.1 - ELO System Background

The ELO rating system, founded by American physics professor Arpad Elo, was initially created as a way to improve the U.S Chess Federation's way of measuring a players' skill, it is one of, if not the most well-known method of calculating relative skill levels in zero-sum-two-player games. This method was then adopted by the U.S Chess Federation in 1960 and then by the FIDE in 1970 (*Elo Rating System - Chess Terms*, 2014b). While the initial intent was for Chess, it has since been adopted over time to be used in other mediums, such as basketball, football, board games, online dating applications and more prominently, video games (Mittal, 2020).

Before the ELO rating system was adopted, the Harkness rating system would be the standard, in which this method would calculate the average rating of a player's competition in a given tournament, ratings after the tournament would be calculated by using the competition average, for example, if the average was 50%, and the player scored 30%, their rating would be the competition average, plus or minus (depending on if the player scored above or below the average) 10 points per percentage point difference (Jørgen Veisdal, 2019). While a simple and efficient way of calculating a player's rating, this was seen as somewhat inaccurate after feedback from observers and so the ELO system was created to calculate ratings on a deeper statistical scale that covers what the Harkness system lacked, depth.

The formulae below will bring this into perspective, where the new rating,  $R_N$ , is equal to the average rating of their competition,  $R_A$ , subtracted by the compounding of 10 and their score difference,  $SD$ .

$$R_N = R_A - (10 * SD)$$

*Equation 1: The formula in which the Harkness rating is calculated after every match.*

A trait that both the Harkness system and the ELO system share is that they both do not calculate absolutes, the ratings use inferred mathematics from player statistics such as their wins, losses, draws etc. A player's rating is dependent on their competitors' ratings and the results against them (Mittal, 2020). Here is a simple scenario between a higher-rated player and a lower-rated player to explain how the system would work. Table [2]

*Table 2: A quick summary of how the ELO system would calculate ratings after a match between a higher rated player and a lower rated player. NOTE: The numerical values provided is a simplification and are arbitrary.*

Match Outcome	Player Ratings Post Match
Higher Rated Player: Win Lower Rated Player: Loss	Higher Rated Player: +5 Lower Rated Player: -5
Higher Rated Player: Loss Lower Rated Player: Win	Higher Rated Player: -50 Lower Rated Player: +50
Higher Rated Player: Draw Lower Rated Player: Draw	Higher Rated Player: -5 Lower Rated Player: +5

As you can see from the table above, the ratings are zero-sum, meaning any player's gain, is the other players' loss and in turn would favour the lower rated player due to having "less to lose" against a higher rated player who statistically should have a higher win rate against them

## 2.2.2– Math behind the ELO system

Elo's main assumption was that the performance of each player in each game would be a randomly assigned variable that would follow a normal distribution curve, therefore would give some leeway to the player if they were to perform better/worse game to game, the average value of their true skill would remain the same. This means that this system better reflects a player's change in skill over a longer period of time (Mittal, 2020). This means there will be less frequent and abrupt jumps in a players' skill and would therefore be a more accurate reflection of one's skill.

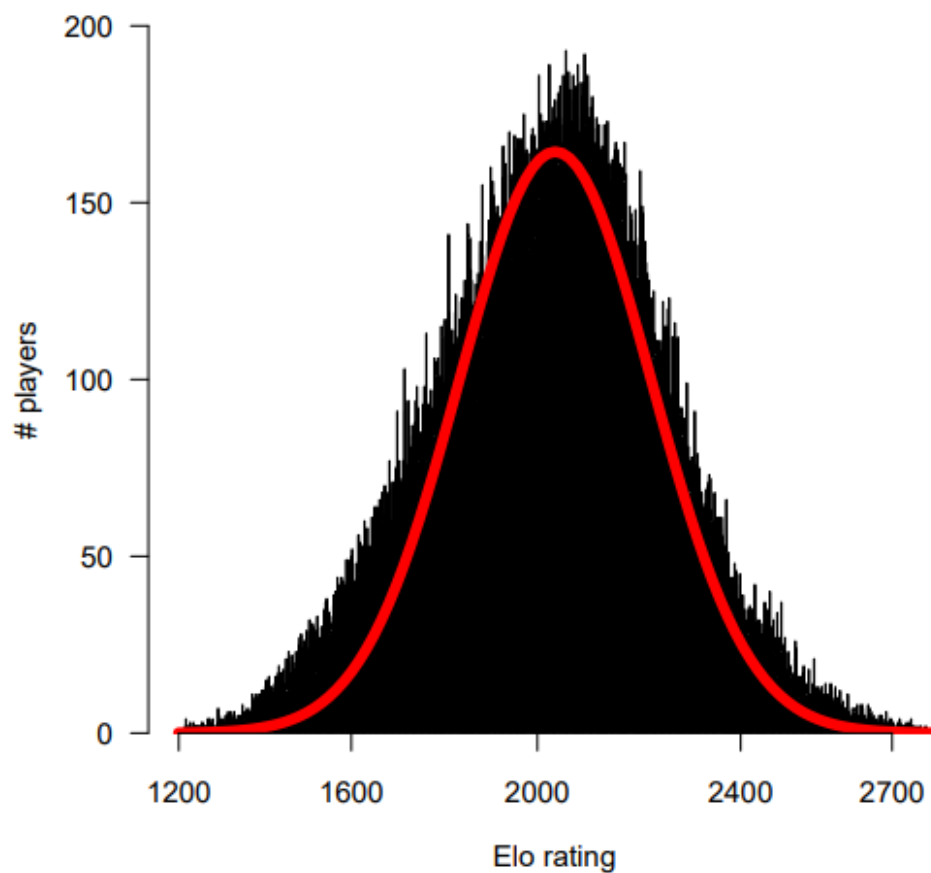


Figure 1: The distribution of player skill as recorded by the FIDE (Breznik & Vladimir Batagelj, 2011).

As an example, if this was put into practice, a match will be taking place between Player 1 and Player 2, both of which have undetermined strengths,  $R_1$  for player 1 and  $R_2$  for player 2. The expected score,  $E_1$ , for player 1 can be assumed using this formula:

$$E_1 = \frac{1}{1 + 10^{(R_1 - R_2)/400}}$$

Equation 2: Formulae for calculating predicted score for player 1 (Mittal, 2020).

And the same can be assumed for player 2:

$$E_2 = \frac{1}{1 + 10^{(R_2 - R_1)/400}}$$

*Equation 3: Formulae for calculating predicted score for player 2 (Mittal, 2020).*

This value is determined by that players' probability of winning + half of their probability of drawing, however if both players end up having equal ratings ( $R_1 = R_2$ ), then the scores would be given half to each player 50/50 meaning both would realistically expect to gain the same number of wins, again showing the zero-sum nature of this Arvad Elo's methodology.

Ratings change from match to match for both players, so a method of calculating new ratings was formed by Elo in which is still commonly in the industry today. His thought process was a simple linear adjustment based on the amount that player over or under performed thus making it more sophisticated than the then current Harkness system (Jørgen Veisdal, 2019). The maximum possible difference a rating could undergo is called the K-factor. This K-factor is dependent on the body that is conducting this matchmaking, this can be official sports bodies such as the U.S. Chess Federation for chess tournaments or games developers for their respective games. The K-factor can be different based on the bracket of player skill, as it was in the U.S. Chess Federation where for masters  $K = 16$  and for lower skilled players,  $K = 16$ , this was to allow for newer players to climb the ranks as a steady rate and keeps the distribution of player skill on the normal distributed curve. Jørgen Veisdal. (2019) brings up the point that lowering the value of the K-factor the higher the skill bracket, the more the evolution of the players' skill and performance was shown over time, however a paper from (Sonas, 2002) would suggest that a slightly higher K-factor of 24 for players in the Chess skill bracket of 2100 – 2400 ELO provides more accurate results. This enforces the idea that depending on the context, evolution in the ELO system would give freedom to the governing body of said rating system on their choice of rules and regulations while still providing a degree of accuracy as long as the trend of the K-factor and skill rating sharing a negatively correlated relationship.

Using the K-factor to determine the resulted change of players' ratings would be based on the difference between their actual score,  $S_A$ , and the score they were predicted before the match,  $E_A$ . The rating for each player is updated post-match using this formula:

$$R'_A = R_A + K(S_A - E_A)$$

*Equation 3: Formula for calculating new ratings after matches. (Mittal, 2020).*

What makes this system fascinating and almost 'future proof' is that the mathematics behind how this whole system works makes it so that it is self-correcting. What this means is that if a player were to rank up to a bracket that their true skill level did not properly reflect, they will most likely perform worse against opposition that truly belonged in that bracket, therefore dropping points, and returning to a bracket in which they will find opponents that are of their actual skill level. This does not hinder skill improvement as the player could very well manage in that bracket and in turn would show player evolution.

### 2.2.3 – Gaps in the ELO System

While the ELO system was breaking ground for its statistical approach to rating skill levels, there were some notable drawbacks and potential issues when deployed in different contexts. In terms of Chess player pools, each governing body/region would have to define their own set of rules and in turn the K-factor, this means that if two players from different regions were to play against one another, the calculation will be unreliable as an ELO rating is only valid within the rating pool it was established (Mittal, 2020). This would make sense as different regions could very well have player skill distributions that skew higher or lower than another regions' player pool average.

Articles and publications such as Sonas. (2020) and Mittal. (2020) both suggest that while the innovation of the ELO system worked fairly in its early time of creation, as years go by the rating of players would hit a point called rating deflation due to the system's zero-sum nature. As newer players entered rating pools, they would start off with a relatively low rating and a higher-than-average K-factor. As they play more and more games, they would take rating off of their opponents which in turn factors into the eventual rating deflation as the once novice player could retire with a high rating. Mittal. (2020), Sonas. (2020) and Jørgen Veisdal. (2019) all concur that increasing the K-factor as the skill brackets increase would somewhat circumvent this issue but as only a temporary measure and as of now, requires a tricky workaround.

Achieving a high rating and climbing the ranks is what the main objective is for competitive players, regardless of skill shown to get there. Players are known to be able to 'protect' their rating by avoiding any game activity. This is prevalent when players can pick and choose when they play their next match and the calibre of opposition they would face where their risk of losing rating is minimal. Mittal. (2020) states this is why some organisations opt out of adopting the ELO system.

### 2.2.4 – Modern Adaptations of the ELO system

The ELO system has been adapted to and changed to suit many different modern-day mediums. In sports, Tennis uses an ELO-based system called the Universal Tennis Rating (UTR) to rank professional tennis players, and FIFA, the global governing body for football, also use an ELO based system to rank national teams. While tennis continues with the one versus one scenario that Arpad Elo intended its use for, FIFA ranks the whole team based on their performance, a player on said team could play their worst game but still end up winning the match and gain rating.

Dating applications are all about finding the perfect match and so the ELO system would ideally be a strong starting point for developers to work off of, this was seen as Tinder confirmed their original matchmaking system used an adaptation of the ELO system (*Powering Tinder® — the Method behind Our Matching*, 2019). It would work in principle, the same as it would in rating chess players, likes on your profile would add value to your rating and would increase the likelihood of getting shown to 'more attractive' profiles. While an adaptation of the ELO system is what Tinder had used initially, they have since then developed their own technology to supersede it. Hinge on the other hand has stated their application uses a variation of the Gale-Shapely algorithm, also known as the stable marriage problem.

Where the ELO system thrives in modern applications would be online gaming. Generally, players of any game that is somewhat competitive will want to be matched with players of similar skill level, considering the ELO system was designed to predict a player's skill level more accurately, it is of no surprise that many modern-day matchmaking systems have a customized version of what the ELO system was before the rise of machine powered algorithms. The simplicity of the ELO system was due to the lack of computer programming power during their time but now that the simplification is not needed, development on more powerful and versatile matchmaking packages have been on the rise, most notably Microsoft and their work on the TrueSkill algorithm.

## 2.3 – TrueSkill

Microsoft took the basis of what made the ELO system work and developed their own matchmaking and rating system called the TrueSkill algorithm. This uses the TrueMatch matchmaking system that Microsoft also built to facilitate their rating system.

### 2.3.1 – TrueMatch Matchmaking

TrueMatch is a reinforcement learning based, machine learning algorithm which utilises artificial intelligence to dynamically adjust matchmaking rubrics in order to find the best quality of match in a timely manner (*TrueMatch Matchmaking System - Microsoft Research, 2020*). The core functionality of this algorithm increases the boundaries of a suitable match if few viable players are within your location, the importance of securing a game with low latency shows the impact that higher ping can have on the quality of a match. In comparison to older systems, Microsoft claim that TrueMatch cuts waiting time in half when used in lower populated regions and will give the player an estimate on the duration of their wait.

### 2.3.2 – The TrueSkill Algorithm

The basis of what makes TrueSkill the definitive successor to the ELO system is that while player score/performance is still an important factor in its calculations, the player's skill is also taken into consideration. For example, a player with a higher skill rating than another player will more likely than not have a better performance that match and will win. In this model, the players' estimated skill is seen as a Gaussian distribution, similar to the ELO system, again making the mean ( $\mu$ ),  $\mu$ , the player's estimated skill value and the variance (sigma),  $\sigma$ , the uncertainty of said skill value (Ibstedt et al., n.d.).

The players' visible rating can be seen as the following:

$$VR = \mu - 3\sigma$$



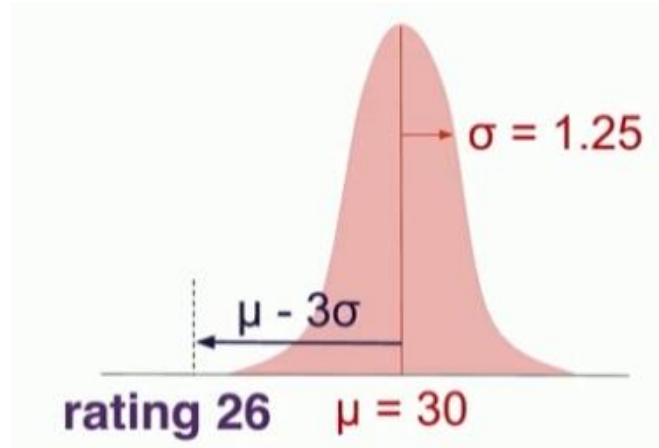


Figure 2: A visual representation of a players' skill for TrueSkill (GDC, 2019).

This calculation covers around 98% of the area of this distribution curve therefore the algorithm will be 98% confident that the actual skill is more than what the visible skill shows, this is due to the conservative nature of this calculation, this works based off of how new players will get an initial rating of 0, therefore their actual skill will reflect more accurately the more they gain rating and play more games. This method is seen as an improvement from the ELO system due to their starting rating of 1500 which can be a misleading rating value and compared to a fresh rating of 0 which better encapsulates the starting perceived skill level of a new player.

With newer players, their  $\sigma$  value will be quite high as the algorithm has no past data to base their estimated skill on so their uncertainty is larger compared to a more experienced player with a lower  $\sigma$  value due to having more data available. The more games one plays, the lower their uncertainty value gets. This allows the algorithm to quickly gauge a players' actual skill level. Calculations for updating players' rating follows these formulae dependent on the result:

$$\begin{aligned}
 \mu_{winner} &\leftarrow \mu_{winner} + \frac{\sigma_{winner}^2}{c} \cdot v\left(\frac{(\mu_{winner} - \mu_{loser})}{c}, \frac{\epsilon}{c}\right) \\
 \mu_{loser} &\leftarrow \mu_{loser} - \frac{\sigma_{loser}^2}{c} \cdot v\left(\frac{(\mu_{winner} - \mu_{loser})}{c}, \frac{\epsilon}{c}\right) \\
 \sigma_{winner}^2 &\leftarrow \sigma_{winner}^2 \cdot \left[1 - \frac{\sigma_{winner}^2}{c^2} \cdot w\left(\frac{(\mu_{winner} - \mu_{loser})}{c}, \frac{\epsilon}{c}\right)\right] \\
 \sigma_{loser}^2 &\leftarrow \sigma_{loser}^2 \cdot \left[1 - \frac{\sigma_{loser}^2}{c^2} \cdot w\left(\frac{(\mu_{winner} - \mu_{loser})}{c}, \frac{\epsilon}{c}\right)\right] \\
 c^2 &= 2\beta^2 + \sigma_{winner}^2 + \sigma_{loser}^2
 \end{aligned}$$

Figure 3: Formulae the TrueSkill algorithm uses to update player ratings (TrueSkillTM Ranking System - Microsoft Research, 2021).

A key difference between the ELO system and TrueSkill's methods of updating player skill is that TrueSkill will not completely guarantee a zero-sum exchange. This is due to the weighting factors being roughly proportional to the total sum of skill uncertainties, therefore a zero-sum exchange could only happen if both player 1 and player 2 have the same  $\sigma^2$  value (*TrueSkill™ Ranking System - Microsoft Research*, 2021).

### 2.3.3 – Advantages of TrueSkill

Comments from Izquierdo. (n.d.) and forum posts (*R/Gamedev - What Are the Pros and Cons of TrueSkill vs Glicko vs Glicko2?*, 2021) suggest the following as the main advantages of using the TrueSkill package:

- Flexible, the algorithm can model a variety of competitive game types as seen with it being implemented for both 1v1 and team-based scenarios
- Fast and Efficient, saves both computation time and the players' time with its use of the TrueMatch system, the rubric for a potential match will be changed on the fly
- Easier to model newer players with their initial rating set to 0
- Versatility, allowing for auto-balancing of teams based on late entries and early leavers

The advantages are understandable due to its basis on the ELO system, all of them are significant upgrades to the shortcomings of the older system. The use of high-powered machines is able to make the calculations more complex and thus more robust than what was formerly deemed to be sufficient. This increase in complexity can also be seen as one of TrueSkill's downfalls.

### 2.3.4 – Disadvantages of TrueSkill

While TrueSkill has been an overall huge improvement on the ELO system and addresses its predecessors' shortcomings, there are still some disadvantages that can come from its use:

- Complex calculations can be confusing for players, especially when lobbies are not fairly matched due to errors, this can be viewed as unfair and non-transparent from the developers
- Proprietary, to use the TrueSkill package, one must obtain a license from Microsoft to use commercially
- Does not consider statistical values of player performance in given matches, therefore their contribution to a team's performance is a sample of their estimated skill instead of their true performance in the match

The disadvantages come to no surprise, considering that the ELO system was based off of a 1v1 scenario, team-based calculations can still be further improved however is still able to function somewhat effectively on updated player skill. Many standalone developers seem to be swaying toward using the Glicko rating system instead due to its open-source capabilities or platforms such as Codeforces and TopCoder opted for an in-house rating system (Vancouver & Liu, n.d.), however in larger, more commercial use, basic ELO implementations and TrueSkill seem to be more popular. Inaccuracy in team-based calculations was then worked on by Microsoft and a newer iteration of the TrueSkill package has since been developed called TrueSkill2.

### 2.3.5 – TrueSkill2

TrueSkill2 is the successor to Microsoft's original rating system, having worked on its flexibility of calculations on team-based competitions, TrueSkill2 has filled in the gaps that were previously on show in its predecessor. With the original TrueSkill being its foundations, TrueSkill2 is backwards compatible with a database that uses its' original rating values and systems. The only real change in this iteration is how the algorithm infers from historical data (Minka et al., 2018).

Minka et al. (2018) interviewed the developers of Gears of War 5 and Halo to describe qualities that make a functional and coherent matchmaker, these properties include:

1. Support for team games, a system should be able to account for more than a 1v1 scenario
2. Changeable skill ratings. Skill values should be able to fluctuate higher or lower regardless of that players' historical data to provide a sense of growth and meaningful feedback
3. Compatibility with existing matchmakers in which skill is described as a single number for simplicity
4. Aligned incentives. The skill rating system should create incentives that coincide with the spirit and nature of the game, i.e., rewarding teamplay regardless of the match outcome for team-based competitions
5. Minimal training data requirements. Matchmaking and rating systems should function well when deployed early in a game's life cycle where this is little to no data to train a model
6. Low computational cost. Skill representations and calculations should be cheap due to data being processed and hosted by the game studio
7. Minimal tuning. Manpower should not have to be allocated to fine tune the matching algorithm so a self-tuning or minimally tuned algorithm should be available

With these in mind, the original TrueSkill package already ticked 6 out of the 7 properties and the seventh property is what came with TrueSkill2, where it is achieved from automatic parameter estimation over a chunk of historical data. Online research has indicated that there is no version of TrueSkill2 available for use in a Python library as the TrueSkill package is already readily available.

## 2.4 – Uses in Dating Applications

### 2.4.1 – Tinder and ELO

Dating application have taken a meteoric rise in recent times where users would let an application find their most suitable partners. This of course would need a working matchmaking system and popular applications such as Tinder have revealed that their early system was a variation of the ELO system. While they have moved onto their own in-house system, the use of ELO shows its versatility in modern mediums. While Tinder did not disclose how they implemented a version of the ELO system, it is assumed to have the player skill translated as profile 'attractiveness', the higher this value is, the more equally attractive profiles are shown to the user, after each like, profile view or match, the user's new rating will be calculated (Mittal, 2020).

### 2.4.2 – Hinge and Gale-Shapley Algorithm

Hinge another popular online dating application has does not use a built-in rating system and it does not use a variation of the ELO system like Tinder used to. According to the director of hinge, the application uses the Gale-Shapely algorithm, also commonly known as the Stable Marriage problem

(Iovine, 2021). The Stable-Marriage problem states that given  $n$  men and  $n$  women where each member of the two groups have ranked the opposite in terms of preference, match the partners with one another in which neither would rather have someone else over their current partner. Once all pairs have satisfied this requirement, then the set of marriages are deemed stable (Scanlon, 2020). While this is the basis of what Hinge uses when finding potential matches, other factors such as profile interactions and historical data of past matched individuals will also be considered as stated by Logan Ury, Hinge's director. Therefore, the more you use the app, the larger pool of factors their algorithm can use to find your 'perfect match'.

In contrast to Tinder, Hinge is designed to be deleted, in which how well the match fits the criteria is more important than speed of finding potential matches, whereas Tinder uses some sort of variation of the ELO system in which simplicity and speed is their main concern. The argument on which is more effective in finding a suitable match is up for debate depending on the context of the user.

## 2.5 – The State of Online Matchmaking

The state of online matchmaking has been stable in the recent years, there has been no cries for broken algorithms or immediate faults with current systems. Work has been made on TrueSkill and the original ELO system to forge the algorithms that are currently available for use today. Machine learning has been an integral part of upgrading the matching system for TrueSkill using a heuristic approach to both finding suitable matches and updating player ratings and in comparison to where rating systems were before the starts of the 21<sup>st</sup> century. With a wide variety of models to base new rating systems on and the ever-increasing power of machine computation, the future of matchmaking looks very bright.

### 3 - Methodology

This section will focus on the methods and techniques I will be employing for this project along with brief explanations on why they were chosen.

#### 3.1 – Choice of IDE

The product will be developed inside of the Visual Studio Code IDE (Integrated Development Environment), this choice has been made due to the ease of use and integration of external applications such as the Anaconda Python Package and JupyterLab extensions. The use of anaconda over the typical Python package was due to Anaconda being more suited toward Data Science and Machine Learning. This gives me a larger range of libraries to help develop the product along with the inclusion of the TrueSkill library, which will be doing most of the skill calculations.

Visual Studio Code also provides an interactive window where I can freely interact with the data outputs along with a useful console that allows for the JupyterLab variables extensions to provide a list of all my variables which will help immensely with debugging any errors.

#### 3.2 – Statistical Analysis

I will be going for a statistical approach when it comes to analysing the data, the use of libraries such as Matplotlib will help to break down the data that is produced. The plotted data can be then compared to other data and will help bring conclusions to the hypotheses presented.

#### 3.3 – Procedural Programming

For this specific product, a procedural approach will be taken due to the nature of Python, I think this will be more beneficial. This methodology will be useful for the fact that the data frame in which the analysis will be used on will be incremental and having other methodologies such as object-oriented will not give me more benefits than procedural.

However, object-oriented programming can be considered for future improvements of this product in which entities that are competing can be held in its own object type, the same can be said for teams of said entities. This is a consideration for the future however and will be explored later in the report.

#### 3.4 – Rating Algorithms

As mentioned previously, I will be implementing the use of the TrueSkill algorithm, in which the data frame will be manipulated with. As such, values such as the  $\mu$  mu and  $\sigma$  sigma will be assigned using functions from the TrueSkill package. An implementation of an ELO system could also be favourable due to its simplistic design, the data produced from this implementation could also be used as a comparison to the current TrueSkill design and could serve as a basis in the comparison.

I briefly covered the Gale-Shapely algorithm when implemented into a dating application however I though with the data sets that I am considering using, this algorithm did not provide as much utility as the TrueSkill algorithm due to its design, instead of trying to find perfect matches, which would suit a more robust and complex matchmaking and rating system combination.

### 3.5 – Machine Learning Techniques

## 4 - Product Development

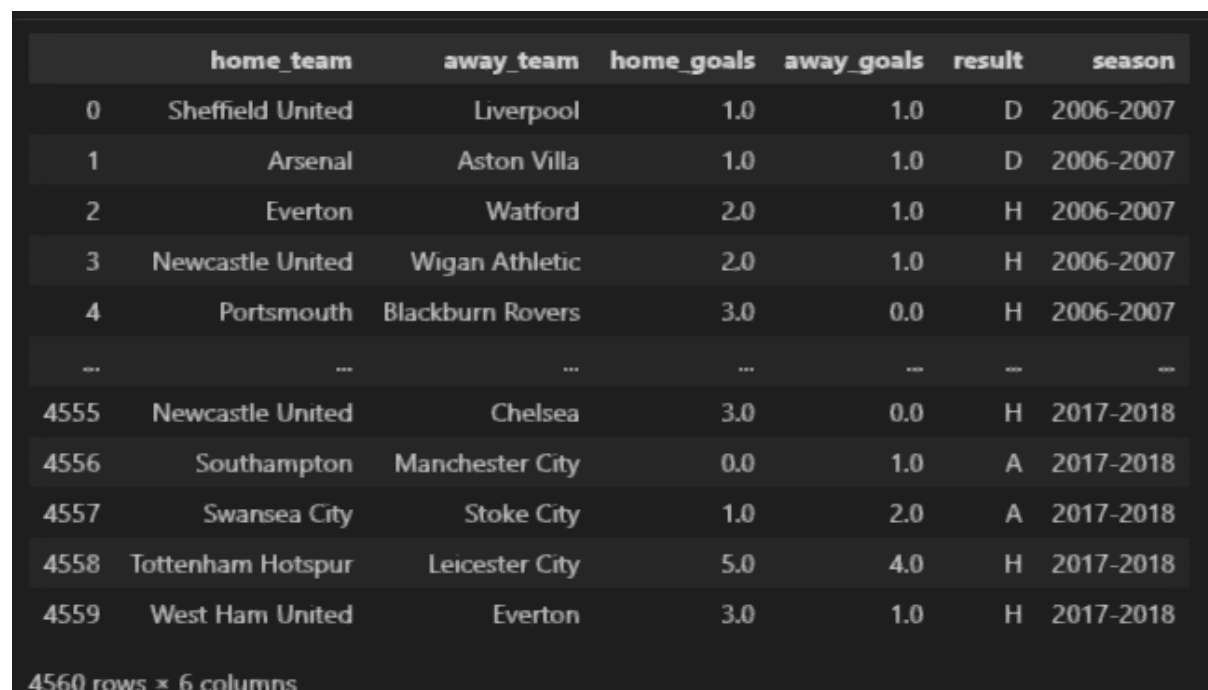
This section will go through the development timeline of the product including any changes to the original plan due to unforeseen errors or otherwise along with code explanations.

### 4.1 – Choosing a dataset

While collecting data manually is a possibility, Kaggle has a wide variety of datasets to choose from which saves time and provides data from a wide range of games and real-world scenarios. For this project I have chosen to use a dataset consisting of Premier League matches from the years 2006-2018. This dataset contains over 4500 matches and their respective information. This has been sourced from this URL, refer to **Appendix A.1**

### 4.2 – Analysing and cleaning the dataset

After loading the dataset into the python environment, I will load the first 10 and last 10 cells to give myself an understanding of the data inside, see **Appendix A.2** for the code, here is the output given:



	home_team	away_team	home_goals	away_goals	result	season
0	Sheffield United	Liverpool	1.0	1.0	D	2006-2007
1	Arsenal	Aston Villa	1.0	1.0	D	2006-2007
2	Everton	Watford	2.0	1.0	H	2006-2007
3	Newcastle United	Wigan Athletic	2.0	1.0	H	2006-2007
4	Portsmouth	Blackburn Rovers	3.0	0.0	H	2006-2007
...	...	...	...	...	...	...
4555	Newcastle United	Chelsea	3.0	0.0	H	2017-2018
4556	Southampton	Manchester City	0.0	1.0	A	2017-2018
4557	Swansea City	Stoke City	1.0	2.0	A	2017-2018
4558	Tottenham Hotspur	Leicester City	5.0	4.0	H	2017-2018
4559	West Ham United	Everton	3.0	1.0	H	2017-2018

4560 rows x 6 columns

Figure 4: First 10 and last 10 indices of the available data. Available to view in **Appendix A.3**

Looking at the data available, there are a number of statistics available for calculation with the TrueSkill algorithm, however for to keep background data noise to a minimum, filtering out unnecessary data will be helpful. First I will, find all matches that are from the latest season, refer to **Appendix A.4**.

	home_team	away_team	result
4180	Arsenal	Leicester City	H
4181	Watford	Liverpool	D
4182	Chelsea	Burnley	A
4183	Crystal Palace	Huddersfield Town	A
4184	Everton	Stoke City	H
4185	Southampton	Swansea City	D
4186	West Bromwich Albion	AFC Bournemouth	H
4187	Brighton and Hove Albion	Manchester City	A
4188	Newcastle United	Tottenham Hotspur	A
4189	Manchester United	West Ham United	H
4550	Burnley	AFC Bournemouth	A
4551	Crystal Palace	West Bromwich Albion	H
4552	Huddersfield Town	Arsenal	A
4553	Liverpool	Brighton and Hove Albion	H
4554	Manchester United	Watford	H
4555	Newcastle United	Chelsea	H
4556	Southampton	Manchester City	A
4557	Swansea City	Stoke City	A
4558	Tottenham Hotspur	Leicester City	H
4559	West Ham United	Everton	H

Figure 5: First 10 and last 10 matches played in the 2017/2018 season.

Now that the latest season has been extracted from the entire data frame, I can move onto calculating the overall ratings for each team after each match and in turn, after the entire season.

#### 4.3 – Calculating the Overall Rating

Finding the unique number of teams that are participating in this season is important, we don't want overlapping teams that could overwrite any pre-existing skill values during rating calculations, refer to **Appendix A.5**. Here is the number of unique teams:

```
✓ import sys ...

Number of participating teams: 20
```

Figure 6: The number of unique participating teams in the season.

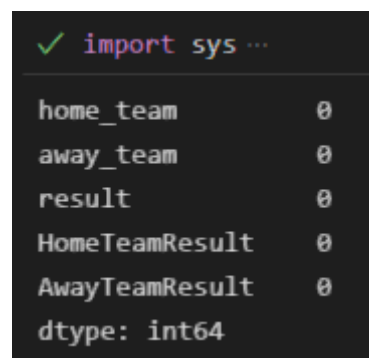


The latest season data frame will have two new columns, 'HomeTeamResult' and 'AwayTeamResult' which will be filled with a 0 or 1 depending on the overall result. See **Appendix A.6**. The new data frame will look like this:



	home_team	away_team	result	HomeTeamResult	AwayTeamResult
4180	Arsenal	Leicester City	H	0	1
4181	Watford	Liverpool	D	0	0
4182	Chelsea	Burnley	A	1	0
4183	Crystal Palace	Huddersfield Town	A	1	0
4184	Everton	Stoke City	H	0	1

The code in **Appendix A.6** states that a loss is shown as a 1 and a win is shown as a 0, this will be helpful in our rating calculations later. To make sure there are no erroneous values, the following code, **Appendix A.7** will show us how many erroneous values, NaN, for each column:



```

import sys ...

home_team      0
away_team      0
result          0
HomeTeamResult 0
AwayTeamResult 0
dtype: int64

```

Figure 7: Number of erroneous data.

In particular there is no erroneous data in our current data frame, therefore it is safe to start looking at our calculation definition. Transferring our current data into a dictionary will make data manipulation easier, this code will create keys, the team names, along with their respective trueskill values, both of which are the default 25 for  $\mu$  and 8.33 for  $\sigma$ . Arrays for both home and away results are made from the values of the respective columns in the truncated latest season data frame, see **Appendix A.8**

The arrays for home teams and away teams for any given match are also created using the same method as the result arrays, both are stacked into one array where all the required information for calculating ratings is stored in a single array. See **Appendix A.9**



## 5 - Evaluation of Research/Product

## 6 - Conclusions

## References and Bibliography

About | TIOBE - The Software Quality Company. (2022, April). Tiobe.com.

<https://www.tiobe.com/company/about/>

Advantages and Disadvantages of C++ | Make your Next Move! - DataFlair. (2019, July). DataFlair.

<https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/>

baeldung. (2017a, January 15). *Overview of AI Libraries in Java | Baeldung*. Baeldung.

<https://www.baeldung.com/java-ai>

baeldung. (2017b, January 15). *Overview of AI Libraries in Java | Baeldung*. Baeldung.

<https://www.baeldung.com/java-ai>

Breznik, K., & Vladimir Batagelj. (2011). *FIDE Chess Network*. ResearchGate; unknown.

[https://www.researchgate.net/publication/316552507\\_FIDE\\_Chess\\_Network](https://www.researchgate.net/publication/316552507_FIDE_Chess_Network)

C++ Introduction. (2014). W3schools.com. [https://www.w3schools.com/cpp/cpp\\_intro.asp](https://www.w3schools.com/cpp/cpp_intro.asp)

Chatterjee, S. (2020, October 18). *Top C/C++ Machine Learning Libraries For Data Science*.

Hackernoon.com. <https://hackernoon.com/top-cc-machine-learning-libraries-for-data-science-nl183wo1>

Costa, C. D. (2020, March 24). *Best Python Libraries for Machine Learning and Deep Learning*.

Medium; Towards Data Science. <https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c>

Elo Rating System - Chess Terms. (2014). Chess.com. <https://www.chess.com/terms/elo-rating-chess>

GDC. (2019). Ranking Systems: Elo, TrueSkill and Your Own [YouTube Video]. In *YouTube*.

<https://www.youtube.com/watch?v=VnOVLBbYIU0>

Ibstedt, J., Rådahl, E., Turesson, E., & Vande Voorde, M. (n.d.). *Application and Further Development*

*of TrueSkill™ Ranking in Sports*. Retrieved April 28, 2022, from <http://www.diva-portal.org/smash/get/diva2:1322103/FULLTEXT01.pdf>

*Introduction to Machine Learning using C++*. (2021). Engineering Education (EngEd) Program |

Section; Introduction to Machine Learning using C++ | Engineering Education (EngEd)

Program | Section. <https://www.section.io/engineering-education/an-introduction-to-machine-learning-using-c++/>

*Introduction to Python*. (2022). W3schools.com.

[https://www.w3schools.com/python/python\\_intro.asp](https://www.w3schools.com/python/python_intro.asp)

Iovine, A. (2021, October 9). *How do all the best dating app algorithms work?* Mashable; Mashable.

<https://mashable.com/article/tinder-bumble-hinge-okcupid-grindr-dating-app-algorithms>

Izquierdo, M. (n.d.). *Math for Game Programmers: Ranking Systems; Elo, TrueSkill and Your Own*.

Retrieved April 28, 2022, from [https://ubm-](https://ubm-twvideo01.s3.amazonaws.com/o1/vault/gdc2017/Presentations/Izquierdo_Mario_Ranking_Systems_Elo.pdf)

[twvideo01.s3.amazonaws.com/o1/vault/gdc2017/Presentations/Izquierdo\\_Mario\\_Ranking\\_Systems\\_Elo.pdf](https://ubm-twvideo01.s3.amazonaws.com/o1/vault/gdc2017/Presentations/Izquierdo_Mario_Ranking_Systems_Elo.pdf)

*Java Programming Language - GeeksforGeeks*. (2014). GeeksforGeeks.

<https://www.geeksforgeeks.org/java/>

Jørgen Veisdal. (2019, September). *The Mathematics of Elo Ratings - Cantor's Paradise*. Medium;

Cantor's Paradise. <https://www.cantorsparadise.com/the-mathematics-of-elo-ratings-b6bfc9ca1dba>

Joy, A. (2019, September 14). *5 Main Disadvantages of Python Programming Language*. Pythonista

Planet; Pythonista Planet. [https://pythonistaplanet.com/disadvantages-of-](https://pythonistaplanet.com/disadvantages-of-python/#:~:text=The%20main%20disadvantages%20of%20Python%20are%20its%20slowness%20during%20execution,in%20the%20enterprise%20development%20sector.)

[python/#:~:text=The%20main%20disadvantages%20of%20Python%20are%20its%20slowness%20during%20execution,in%20the%20enterprise%20development%20sector.](https://pythonistaplanet.com/disadvantages-of-python/#:~:text=The%20main%20disadvantages%20of%20Python%20are%20its%20slowness%20during%20execution,in%20the%20enterprise%20development%20sector.)

Minka, T., Cleven, R., & Zaykov, Y. (2018). *TrueSkill 2: An improved Bayesian skill rating system*.

<https://www.microsoft.com/en-us/research/uploads/prod/2018/03/trueskill2.pdf>

Mittal, R. (2020, September 11). *What is an ELO Rating? The mathematics behind it, and its relation*

*to chess, Tinder... | by Raghav Mittal | Sep, 2020 | Medium | Purple Theory*. Medium; Purple Theory. <https://medium.com/purple-theory/what-is-elo-rating-c4eb7a9061e0>

Moser, J. (2015). *Computing Your Skill*. Moserware.com.

<http://www.moserware.com/2010/03/computing-your-skill.html>

Nation, D. (2017, May 5). *What is the best programming language for Machine Learning?* Medium; Towards Data Science. <https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7>

*Powering Tinder® — The Method Behind Our Matching.* (2019). Tinder Newsroom. <https://www.tinderpressroom.com/powering-tinder-r-the-method-behind-our-matching/>

*Pros and Cons of Python in Machine Learning | Zarantech.* (2018, November 28). Zarantech. <https://www.zarantech.com/blog/pros-and-cons-of-python-in-machine-learning/>

*Python Programming Language - GeeksforGeeks.* (2016). GeeksforGeeks. <https://www.geeksforgeeks.org/python-programming-language/>

*r/gamedev - What are the pros and cons of TrueSkill vs Glicko vs Glicko2?* (2021). Reddit. [https://www.reddit.com/r/gamedev/comments/m81po4/what\\_are\\_the\\_pros\\_and\\_cons\\_of\\_trueskill\\_vs\\_glicko/](https://www.reddit.com/r/gamedev/comments/m81po4/what_are_the_pros_and_cons_of_trueskill_vs_glicko/)

Scanlon, K. (2020, August 31). *Dating Data: An Overview of the Algorithm - The Startup - Medium.* Medium; The Startup. <https://medium.com/swlh/dating-data-an-overview-of-the-algorithm-afb9f0c08e2c>

Sonas, J. (2020, April 19). *What's Wrong With the Elo System?* Chess News; ChessBase. <https://en.chessbase.com/post/what-s-wrong-with-the-elo-system>

*Top 5 Programming Languages and their Libraries for Machine Learning in 2020 - GeeksforGeeks.* (2020, June 26). GeeksforGeeks. <https://www.geeksforgeeks.org/top-5-programming-languages-and-their-libraries-for-machine-learning-in-2020/>

*TrueMatch Matchmaking System - Microsoft Research.* (2020, May 14). Microsoft Research. <https://www.microsoft.com/en-us/research/project/truematch/>

*TrueSkill™ Ranking System - Microsoft Research.* (2021, November 15). Microsoft Research. <https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/>

Vancouver, A., & Liu, P. (n.d.). *An Elo-like System for Massive Multiplayer Competitions.* Retrieved April 29, 2022, from <https://arxiv.org/pdf/2101.00400.pdf>

*Which is Better for AI: Java or Python? A Breakdown of Our Top Pick.* (2021, October 8). Springboard Blog. <https://www.springboard.com/blog/data-science/which-is-better-for-ai-java-or-python/#:~:text=Java%20is%20faster%20than%20Python,which%20increases%20the%20execution%20time>.



## Appendices

### Appendix A

#### A.1 – URL of sourced Dataset

<https://www.kaggle.com/datasets/zaeemnalla/premier-league?select=stats.csv>

#### A.2

```
#read the results.csv file and display the data for for the first 10 and last
#10 indices of the database
match_results = pd.read_csv('C:/Users/44745/Documents/Final Year/FYP/input/re-
sults.csv')
match_results.head(10).append(match_results.tail(10))
```

#### A.3 – Output from A.2

	home_team	away_team	home_goals	away_goals	result	season
0	Sheffield United	Liverpool	1.0	1.0	D	2006-2007
1	Arsenal	Aston Villa	1.0	1.0	D	2006-2007
2	Everton	Watford	2.0	1.0	H	2006-2007
3	Newcastle United	Wigan Athletic	2.0	1.0	H	2006-2007
4	Portsmouth	Blackburn Rovers	3.0	0.0	H	2006-2007
...	...	...	...	...	...	...
4555	Newcastle United	Chelsea	3.0	0.0	H	2017-2018
4556	Southampton	Manchester City	0.0	1.0	A	2017-2018
4557	Swansea City	Stoke City	1.0	2.0	A	2017-2018
4558	Tottenham Hotspur	Leicester City	5.0	4.0	H	2017-2018
4559	West Ham United	Everton	3.0	1.0	H	2017-2018

4560 rows × 6 columns

#### A.4

```
#find all data that is from the 2017/2018 season using column value verifica-
tion. A
#truncated version of the df will be stored while dropping noisy columns
latest_season = pd.DataFrame(match_results.loc[match_results['season'] ==
'2017-2018'])
latest_season_tr = latest_season[['home_team', 'away_team', 'result']].copy()
latest_season_tr.head(10).append(latest_season_tr.tail(10))
```

A.5

```
teams = pd.DataFrame(latest_season.iloc[:,0].unique(), columns=['Team'])
```

A.6

```
#adding new columns into the dataframe for binary results
latest_season_tr['HomeTeamResult'] = latest_season_tr['result']
latest_season_tr['AwayTeamResult'] = latest_season_tr['result']

# #Add binary results for both home and away teams, where Win and Draw = 0 &
Loss = 1
latest_season_tr.replace({'HomeTeamResult' : {'H':0, 'D':0, 'A':1}},
inplace=True)
latest_season_tr.replace({'AwayTeamResult' : {'H':1, 'D':0, 'A':0}},
inplace=True)
```

A.7

```
print(latest_season_tr.isna().sum())
```