Assignment 3

Data Analytics for DreamTeam Platform

Weisi Han -17028872- 29 March 2018

Introduction

Todays' gaming industry has been growing at a skyrocketing speed, in which eSports games have attracted more than 2.3 billion people coming to participate and engage in it. Out of the top-ten popular eSports games, Counter Strike: Global Offensive (CS:GO) is the most competitive one. It is a multiplayer first-person shooter video game offering five-on-five combat. Both sides in game have to complete separate tasks in order to eliminate the other. Due to the team-based feature, players need to form a team before starting the game. As the overall team performance depends on the skills of individual players and the cooperations among them, skills of team members have to be close.

However, allocating distinct roles to a specific person in a team or finding new teammates with similar overall performance but complementary skills is quite challenging for most users. To solve this problem, DreamTeam developed an eSport platform for gaming recruitment and management. With an aim of helping players find, manage their teams, and boosting their skills, it has collected mountains of game records and team data to analyse each individual player's and team's performance. Based on data provided by DreamTeam, our research draws attention to extract and capture information from DreamTeam dataset, finding meaningful facts and hidden correlations among raw data, and analysing the characteristics of different player roles, player performance and team needs. By doing so, we can recommend the platform users well-matched teammates with roles which best suited their own style of play. This will help maintain and boost team overall competence, and create a competitive gaming experience in the long run.

DreamTeam Dataset

The DreamTeam dataset aggregates basic statistics of CS:GO game play histories, platform activities and other game related information. The overall dataset can be categorised into five aspects: users' basic information, players' game profile, team data, vacancy and invite data, and other information. We will only introduce three of them which are used in our recommendation system.

- **Users' Basic Information**

Most of the game players' personal information is included in table core.user, core.profile and core.langs. These information involves the user's *id*, *gender*, *country*, etc.

- **Players' Game Profile**

Different tables can be put together to explore comprehensive game profile statistics. Under folder "game", we can find users' game roles by looking at *profiles_roles*. Tables under "csgo" also contains a detailed set of game play metrics. One of the most important table is csgo.profiles, which comprises most attributes relevant to users' performance. (eg. the total number of *kills*, *deaths*, *headshots*, *wins*, *mvps*, plus their *ranks*). These attributes will be further selected and analysed in our report. Other performance data involves the players' capability of using weapons. As different sets of weapons are correlated with distinct CS:GO roles, these statistics will be also of great value to mine particularly when we analyse players' role performance. This data can be found in csgo.weapons_stats.

- **Team Data**

As a team-based eSports game, DreamTeam dataset also contains basic information records for each team, reflecting their compositions and competencies. This information can be found from core.teams_data, and teams_users.

System Architecture

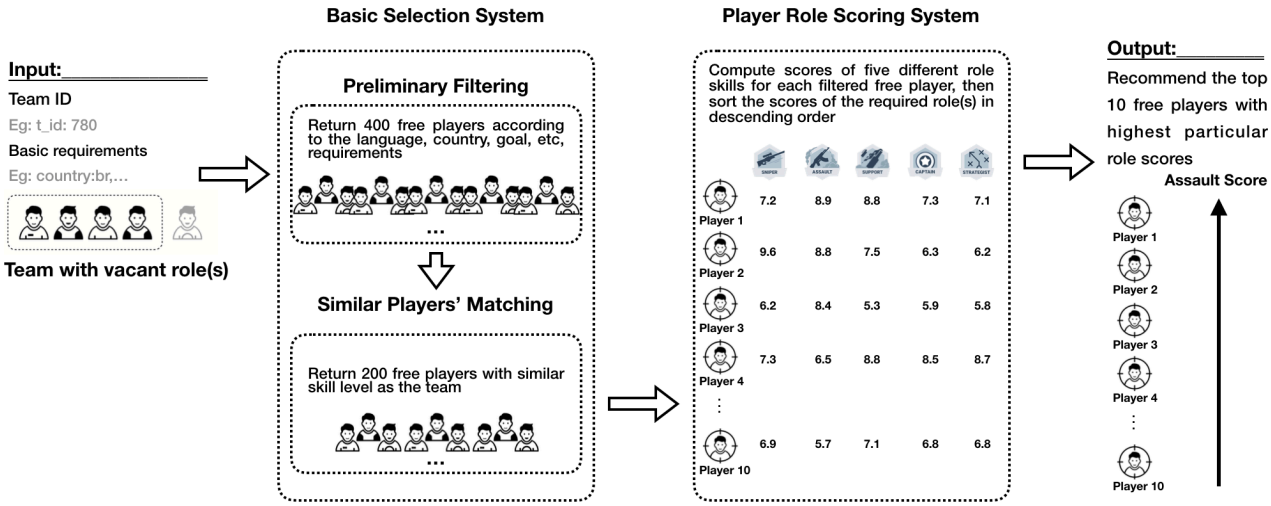


Figure1. System Architecture

The overall system diagram for player recommendation is presented in Figure1. The main components of the architecture contains four parts, team requirements input, basic player selection, role scoring and final recommendation. The input of the system include a specific team id and some hard requirements. As the team id is the unique identifier of a team, our system could identify the team composition and retrieve related information of each team member for further performance analysis. Apart from entering team id, users could also specify requirements for their new teammate(s). These requirements include language, country, region, server, player ranks and goals.

The basic selection process starts from creating a simple filter to select players who meet the query's requirements. Once the unwanted players are removed, the rest free players will be sent to player matching system. This involves comparing users' overall play style and and combat statistics. By identifying players that have similar combat performance, a shortlist of 200 players will be generated.

As players in a team have different roles and responsibilities that they should strive for in a fight, in the next procedure, every player will be assigned scores based on their role performance, i.e. how well the player could perform when they play on each role of the game. Similarly, each of the current team members will also be scored in this way. The team role scores could be calculated as the average of its team members' different role scores and a radar diagram of the team performance will be generated accordingly. By doing so, we could spot the team's role strengths and weaknesses and recommend a final list of ten free players who are better playing a particular role which the team currently lack of. As a result, the overall team performance could be balanced and improved.

Methodology

1. Team Members' Social Network Analysis

Before constructing the recommendation system, we decide to apply network analysis to investigate each individual player's team joining behaviour, i.e., the interactions between players. As each player can join different teams at different stages, they are connected with each other by playing together on the same team. Therefore, we can represent the player relationships as a social network and explore the social relations between players and their teammates.

We first extract the user id associated with the team id to find the team composition. By enumerating the pair combinations of each player and his different teammates, a social network could be constructed. From this step, we can obtain tens of thousands of links between different players who have once become teammates with each other over the previous months. Based on the edge list information, a social network's adjacency matrix could be formed.

In order to figure out whether players' social network degree distribution could be well approximated by a power law, we plot it in both linear scale as well as log-log scale, and use the least squares fitting to estimate its tail exponent alpha. Furthermore, we also calculate the clustering coefficient for each of the nodes in our network and investigate its frequency distribution since the clustering coefficient is a good measure of how complete the neighbourhood of a node is. The formula is shown as follow.

$$C_i = \frac{number\ of\ edges\ between\ the\ neighbours\ of\ i}{k_i(k_i - 1)/2}$$

2. Construction of Basic Selection System

As is depicted in Section III, aside from the user input and system output, the basic selection system contains two processes: preliminary filtering and similar players' matching. The preliminary filtering part is easy to implement. It will only select players who conform to the basic requirements set by the user.

With respect to select players with similar overall performance, we need to fulfil it step by step, from identifying the important features to construct a performance matrix, to computing the performance similarities between the team players and free players.

i) User Overall Performance Feature Selection and Creation

In order to efficiently analyse the combat performance of CS:GO players, a subset of parameters are selected from "csgo.profiles", which are further processed to generate new features for better skill measurement. The descriptions of the processed key performance features are shown in Table1.

Feature	
Kill/Death Ratio (KDR)	The number of total kills the player has scored divided by the number of total deaths. KDR is a standard measure of how effective a player is being. Typically, if the KDR is greater than 1, which means the player has more kills than deaths, he/she may perform effectively in combat.
Hit/Shot Rate	The number of target hit divided by the number of total shot fired. It can be seen as a measurement of hit accuracy.
Win Rate	The number of total wins divided by the number of total rounds played
Headshot Rate	The number of headshots divided by the number of total target hits

Table1. Feature Selection and Description

ii) Data Pre-processing and Outlier Removing

The first step of processing raw data should be eliminating meaningless record entries. Therefore, we first drop users who have never played the game (total play time is null or zero). Then, replace the missing values of all other attributes by zero. For example, if the total kill number is null, it means there is a zero-kill record for this player. We also find that duplicated steam id exists; however, their nickname, game records, ranking and goal varies. Therefore, we keep both of them for the following analysis.

In further pre-processing, we set a boundary to filter out the unrealistic data. If the KDR is higher than 10, we would remove it, as such phenomenon is extremely abnormal in CS:GO. For Hit/Shot Rate, Win Rate, and Headshot Rate, if any of them is larger than 1, it might be caused by faulty tracking errors, thus should be treated as extreme outliers and removed as well.

iii) Calculate the Best Fit and Plot the Distribution

After excluding the outliers, we could analyse the four moments of our features. We initially use Bootstrap method to derive an estimation of the confidence interval around the mean, variance, skewness, and kurtosis (See Appendix). The result is quite similar to what we compute by using python's *numpy* packet.

In order to find the best way of fitting for our features, we use the maximum likelihood method (MLE) to estimate the overall probability distribution. The idea is to apply a set of candidate probability distribution functions, including exponential, logistic, Beta, Gamma, T, Normal, Lognormal, and Skew-normal distributions to find parameters which best fit our observed data and maximise the loglikelihood. We also use another estimator, Akaike Information Criterion (AIC) to examine the relative quality of this set of distribution models. The relationship between AIC and MLE is: $AIC = 2k - 2ln(L)$, where k is the number of parameters and the preferred distribution should be the one with the minimum value of AIC. By comparing the values of loglikelihood and AIC, we could find the optimum distribution which best represents our empirical observations.

As the overall distributions for our four features tend to have a heavy-tailed characteristic, we need to visualise the Complementary Cumulate Distribution (CCD) from the data on

log-log axes for tails' property analysis. To achieve that, we sort the original feature data in ascending order and the CCD function can be estimated by $y_k = 1 - Rank(x_k)/(N+1)$. After generating the rank-frequency plot, we can better explore the tail region of large fluctuations.

The next step is to find a point to separate the feature's main body and tail. The CCD for heavy-tailed distribution declines as a power law in their tails. Thus, we would use *powerlaw* packet in python to create a Fit object and trigger a search for the optimal x_{min}, which is a value generates the minimum Kolmogorov-Smirnov distance between the data and fit. In order to test the goodness of fit, we computed the p-value from samples of the main body to assess whether a null hypothesis of "the feature's main body fitted a specific distribution" can be retained at the 5% significance level. Because of the page limit and presentation concern, the part of the tail analysis is placed in the appendix.

iv) Compute the Overall Performance Similarity between Team Members and Free Players

After having a general understanding of the players' overall performance distribution, we will continue to discover similar players by aggregating and comparing the above four performance features. We consider these features as dimensions in a 4-dimensional Euclidean Space. Then, each player's performance will be transformed as a point in this space, where each coordinate of one point denotes a certain type of performance (either KDR, Accuracy, Win Rate, or Headshot Rate). After constructing the current team players' and free players' performance matrix, we can then calculate the Euclidean distance between each pair of them, which is defined as:

$$Euc_Dis(P_A, P_B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + (a_4 - b_4)^2}$$

$$Euc_Dis\left(\begin{bmatrix} TP1_{kdr} & TP1_{acc} & TP1_{winr} & TP1_{hsr} \\ TP2_{kdr} & TP2_{acc} & TP2_{winr} & TP2_{hsr} \\ TP3_{kdr} & TP3_{acc} & TP3_{winr} & TP3_{hsr} \\ TP4_{kdr} & TP4_{acc} & TP4_{winr} & TP4_{hsr} \end{bmatrix} \cdot \begin{bmatrix} FP1_{kdr} & FP1_{acc} & FP1_{winr} & FP1_{hsr} \\ FP2_{kdr} & FP2_{acc} & FP2_{winr} & FP2_{hsr} \\ FP3_{kdr} & FP3_{acc} & FP3_{winr} & FP3_{hsr} \\ \vdots & \vdots & \vdots & \vdots \\ FP400_{kdr} & FP400_{acc} & FP400_{winr} & FP400_{hsr} \end{bmatrix}\right)$$

$$\Downarrow$$

$$\begin{bmatrix} Dis_{TP1\cdot FP1} & Dis_{TP1\cdot FP2} & Dis_{TP1\cdot FP3} & \cdots & Dis_{TP1\cdot FP400} \\ Dis_{TP2\cdot FP1} & Dis_{TP2\cdot FP2} & Dis_{TP2\cdot FP3} & \cdots & Dis_{TP2\cdot FP400} \\ Dis_{TP3\cdot FP1} & Dis_{TP3\cdot FP2} & Dis_{TP3\cdot FP3} & \cdots & Dis_{TP3\cdot FP400} \\ Dis_{TP4\cdot FP1} & Dis_{TP4\cdot FP2} & Dis_{TP4\cdot FP3} & \cdots & Dis_{TP4\cdot FP400} \end{bmatrix} \xRightarrow{Sum\ over\ column} min\ [Dis_{Team\cdot FP1} \quad Dis_{Team\cdot FP2} \quad \cdots \quad Dis_{Team\cdot FP400}]$$

TP: team player, FP: free player, $TP1_{kdr}$: KDR of team player 1
$Dis_{TP1\cdot FP1}$: distance between team player1 and free player1

The above flow chart shows the process of finding free players with similar overall performance with team players. By summing over columns of the matrix, the distance between each free player and the whole team players could be obtained. We only select the top 200 free players with the smallest Euclidean distance as those players can be treated as the ones with comparable overall combat performance as the team players.

3. Construction of the Player Role Scoring System

As is mentioned above, we will continue to evaluate how good the gamers are at different jobs by highlighting statistics that are important for the roles they play. Inspired by the TrueSkill ranking system (a skill-based ranking system developed for Microsoft Xbox Live gaming assessments), we work out our own scoring calculators to assess gamers' different role play performance. Each equation takes three parameters into account: Accuracy, KDR and Headshot rate. The scores are computed as follow.

$$Score_{sniper} = \alpha_{sniper} \times Accuarcy_{sniper_weaps} + \beta_{sniper} \times KDR + \gamma_{sniper} \times HeadshotRate$$
$$Score_{assault} = \alpha_{assault} \times Accuarcy_{assault_weaps} + \beta_{assault} \times KDR + \gamma_{assault} \times HeadshotRate$$
$$Score_{support} = \alpha_{support} \times Accuarcy_{support_weaps} + \beta_{support} \times KDR + \gamma_{support} \times HeadshotRate$$
$$Score_{captain} = \alpha_{captain} \times Accuarcy_{captain_overall} + \beta_{captain} \times KDR + \gamma_{captain} \times HeadshotRate$$
$$Score_{strategist} = \alpha_{strategist} \times Accuarcy_{strategist_overall} + \beta_{strategist} \times KDR + \gamma_{strategist} \times HeadshotRate$$

Although the KDR and headshot rate are same parameters for different roles, the Accuracy parameter is not the same as we defined in previous sections. In addition, we will also assign distinct roles unequal weights to the three parameters since each role demands a slightly different balance of skills. By doing so, we can distinguish roles from each other.

Let's consider their weapon use preference first. In terms of the three main roles, sniper, assault and support, they have a various preference towards weapon combinations. Referring to the official document provided by CS:GO, the popular weapon set for snipers include AWP, SSG08, SCAR20 and G3SG1, whereas the popular weapon set for assaults include AK-47, M4A1, FAMAS and GALIL AR. For support players, the main job for them is to practice all sorts of grenades and throw them at exact time and place. Focusing on our dataset, we only have data for High-explosive (HE) Grenade and Flame Grenade (Molotov). Therefore, the support weapon set could be composed of those two grenades.

Because of the various weapon set preference, analysing different roles' overall accuracy would be rendered useless and the calculation of the accuracy parameter should vary from role to role. Therefore, when computing the accuracy of snipers and assaults, we only consider the average accuracy for their own weapon lists. In terms of support players, their accuracy is treated as their overall grenade damage dealt. For other two tactic roles (captain and strategist), they do not have a specific weapon list, so we have to use their overall accuracy.

Now we need to use a scientific way to assign unequal weights to the three parameters under the five situations. Our approach first selects players who only played a single role in previous months so that all one-role players can be retrieved from the dataset. They will form 5 groups showing distinct characteristics of different roles. Then, for this composed training set, we treat the players' win rate as the criterion for their role performance and calculate the Pearson correlation between each of the factors in our formula and win rate. As correlation is a technique for investigating the strength of the association between pairs of quantitative variables, we are able to figure out how different roles carry different emphases the three factors. Then, the coefficients α, β, and γ could be tuned according to

factors' correlation to their role performance. The higher the correlation coefficients, the higher weight will be allocated to each parameter. When all of the values have been settled, we could calculate role scores of each player and recommend the team a list of candidates with higher particular role scores.

Results and Analysis

1. Team Members' Social Network Analysis

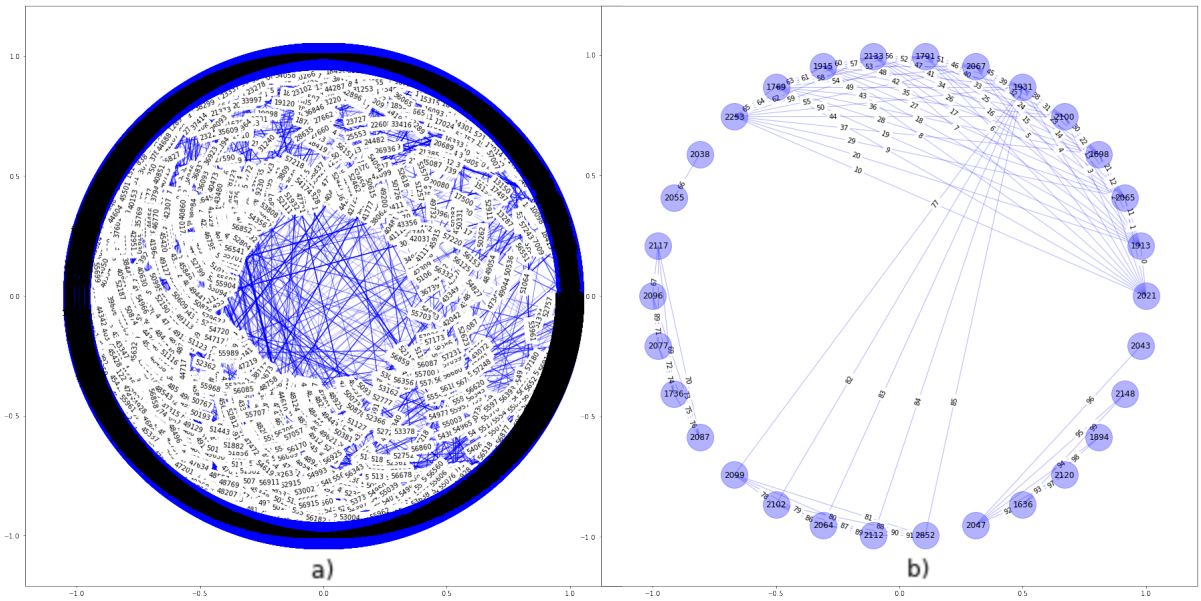


Figure2. Teammate Social Network Plot
(a) Network Graph with All Players
(b) Simplified Version of Network Graph, with only 30 players

Plot a) in Figure2 shows the teammate social network of all CS:GO players who have ever joined in at least one team and played with at least one teammate, whereas plot b) is a simplified version which only visualises the relations among 30 players. Each node in the undirected graph represents an individual player and an edge between two players signifies that they used to play on the same team or they are currently teammates. The basic information about the network has been summarised in Table2. Out of the total 84393 users who ever registered a steam id, less than thirty percent, 25262 players are in our network (only 25262 players have teammates). Each of them has an average teammates number of 4.53. The maximum degree is 77 which means this player has formed several teams and built relationships with 77 players. Exemplified by user 1931, his teammates include players from two teams (2099, 2102, 2964, 2112, 2852 in team 2073, and 2021, 1913, 2065, etc. in team 2039). We did not assign any weight on edges, so we can just ignore the numbers on links since they are meaningless in this plot.

	Nodes	Edges	Degree					
			Avg Degree	Min Degree	Q1	Q2	Q3	Max Degree
Number	25262	57253	4.526	1	2	3	5	77

Table2. Degree of Team Player Social Network

Table3 denotes the frequency count of the occurrence of each clustering coefficient. It is obvious that over 70% of nodes have a clustering coefficient equal to 1, which means the vast majority of players only join one group and as a result, the neighbourhood of a player is just the rest people in this team and they are mutually connected. 20% of nodes have a NAN clustering coefficient, which means these kind of people are on a team with only 1 teammate. The average clustering coefficient is 0.7492.

Clustering Coefficient C	Frequency
NAN	20.68%
0	0.17%
0 - 0.33	1.26%
0.33 - 0.66	6.10%
0.67 - 1	1.18%
1	70.60%
Average Clustering Coeffcient	0.7492

Table3. Clustering Coefficient Frequency Count

Figure3 plots the players' degree distribution in both linear scale as well as in log-log scale. The degree of each player in the graph refers to the number of links to other players. In plot a), it is obvious that the majority of the CS:GO players have a very low degree (perhaps they only join one team and most of the teams have less than 5 players). However, a few players have an extremely high degree, and we can infer that such kind of player is more likely to swap teams and cooperate with new teammates. Therefore, the linear plot displays us a "long right tail" characteristic.

Focusing on plot b), as the Log-log plot shows roughly a linear decay, we can infer the degree distribution of our CS:GO player social network is quite similar with many other real-world social networks, which exhibits a typical power-law of the form $P(d) \propto d^{-\alpha}$. Since $P(d)$ decreases polynomially quickly, it results in a heavy tail characteristic for the extreme values on the probability distribution plot. We then fit a line (red line in the picture) by using the least square method and obtain the power law exponent $\alpha = 3.728$, which is just a bit higher than the typical range [2, 3].

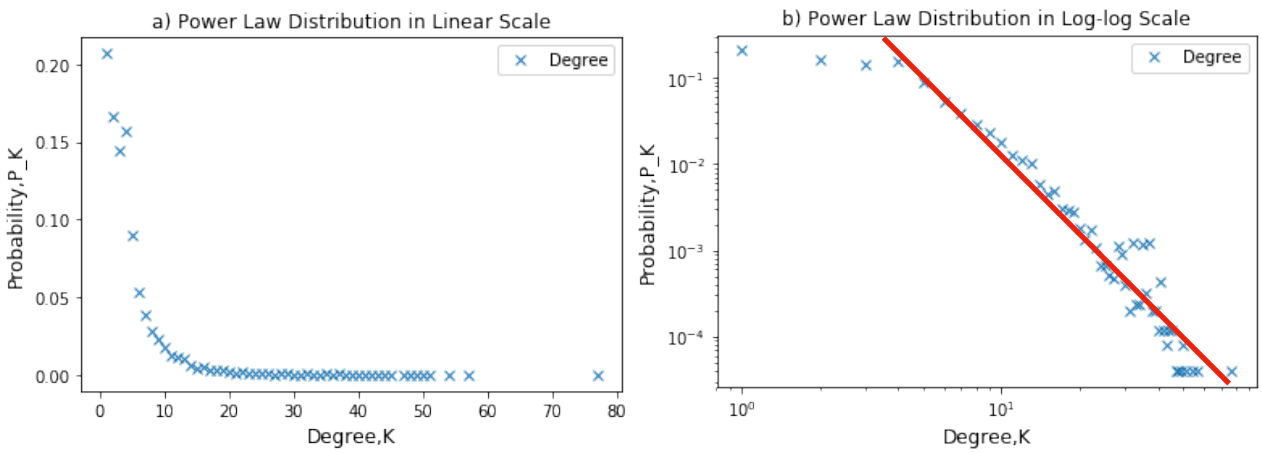


Figure3. Team Members' Degree Plot

2. Performance Feature Distribution and Analysis

i) Distribution Plot for the Four Features

Figure4 shows the overall distribution of the four features after removing the outliers. We can find that the average KDR for CS:GO players is slightly over 1, the win/loss rate is approximately 1:1, the average shooting accuracy and headshot rate is 0.186 and 0.12 separately. The distribution for users' KDR has a slightly large variance than other variables, which indicates that the players' KDR values are spread out from its mean and the difference between each player's KDR varies a lot. The KDR and the shot accuracy has a larger positive skewness as the tail on the right side is much heavier than the left side. From this, we can infer that the left tail index is larger than the right one. In terms of kurtosis, it measures the probability amount in the tails. All of the kurtosis values are substantially larger than 3, the expected value for a Gaussian distribution, which shows the dataset displays leptokurtic behaviour.

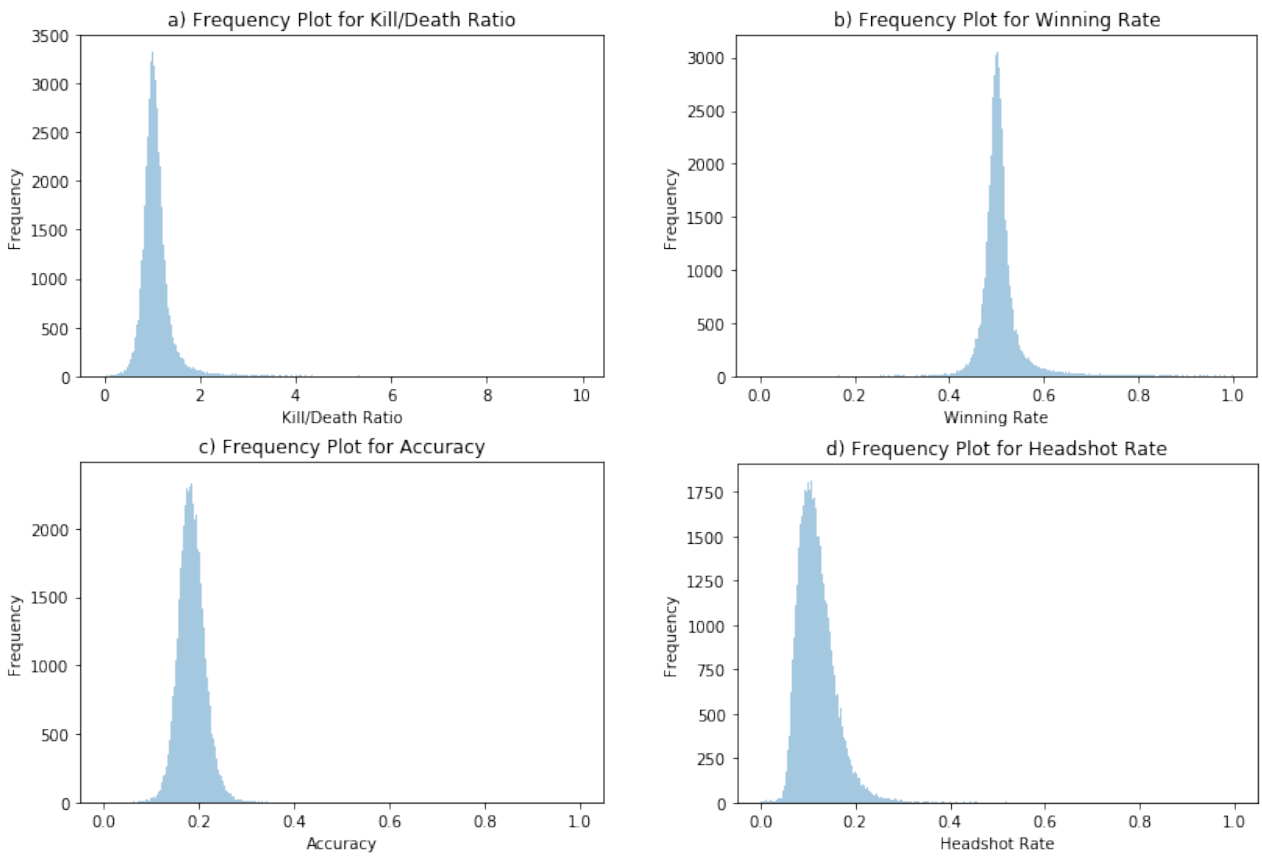


Figure4. Distribution Plot of the Four Features

Feature	Four Moments				Quartile				
	Mean	Variance	Skewness	Kurtosis	Min	Q1	Q2	Q3	Max
Kill/Death Ratio (KDR)	1.103	0.230	7.753	97.241	0.0	0.9143	1.0383	1.1811	9.9466
Wining Rate	0.509	0.003	2.068	22.381	0.0	0.4885	0.5022	0.5176	0.9999
Hit/Shot Rate	0.186	0.001	4.412	82.846	0.0	0.1675	0.1846	0. 2027	0.9999
Headshot Rate	0.120	0.002	3.200	35.721	0.0	0.0915	0.1130	0.1398	0.9999

Table4. Four Moments and Quartiles of Four Features

ii) Fit Probability Distributions into Feature Data

In this step, we fit our four features into possible probability distributions, and the top three probability distributions as well as its corresponding negative loglikelihood and AIC values are shown in Table5. The objective is to select a minimum negative likelihood and AIC, and fit to one specific distribution.

We can find that the best fit distribution for the headshot is Lognorm distribution and the blue line represents how the dataset fitted into this kind of distribution (See Figure5 d)). T distribution is the best fit for the rest three features. From the pictures below, we can also find the tails of the distributions still contain a great amount of probability, this heavy-tailed characteristic induces us to fit a power law distribution to our features.

Feature	Best Fit Distribution (negative loglik, AIC)	Second (negative loglik, AIC)	Third (negative loglik, AIC)
Kill/Death Ratio (KDR)	T (6923.02, 13852.03)	Lognorm (19000.5, 380007.08)	Logistic (19868.8, 39741.51)
Wining Rate	T (-147907.51, -295787.04)	Logistic (-129735.36, -323316.35)	Skewnorm (-111330, -222384.89)
Hit/Shot Rate	T (-162784.40, -325562.79)	Logistic (-161160.17,-323316.3)	Lognorm (-159078, -318150.6)
Headshot Rate	Lognorm(-141380,-282754.91)	Gamma(-139692, -279377.24)	Beta(-139599, -279194)

Table5. Top 3 Probability Distribution Fits

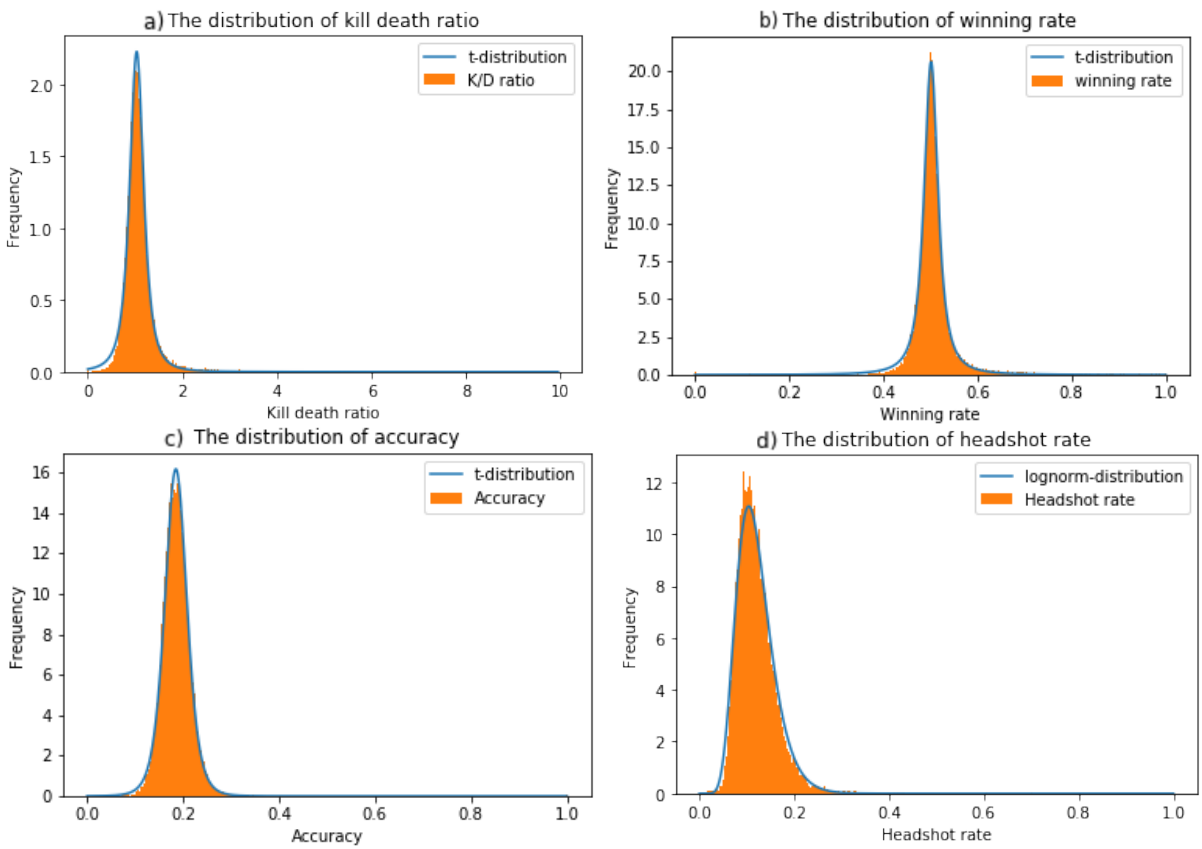


Figure5. Best Fit Distribution for Four Feature Data

iii) Rank-Frequency Plot and Test

The following rank-frequency diagrams accurately represent the CCDs for user performance. The orange line is the normal distribution with mean and variance value calculated from our filtered feature data and the blue cross reflect their real distribution on log-log axes. The probability of extreme value occurrence is larger than what predicted by a normal distribution with the same mean and variance, which will lead to heavy-tailed distributions.

The minimum values to capture the tails are shown in Table6. Any KDR value larger than 2.0, winning rate, accuracy and headshot rate higher than 0.57, 0.2 and 0.15 will be categorised as tails for each individual distribution. We then calculate the p-values for the main body distribution of the four features. The result shows only the winning rate's p-value is higher than the chosen 5% significance level, so only for the winning rate the null assumption can be accepted.

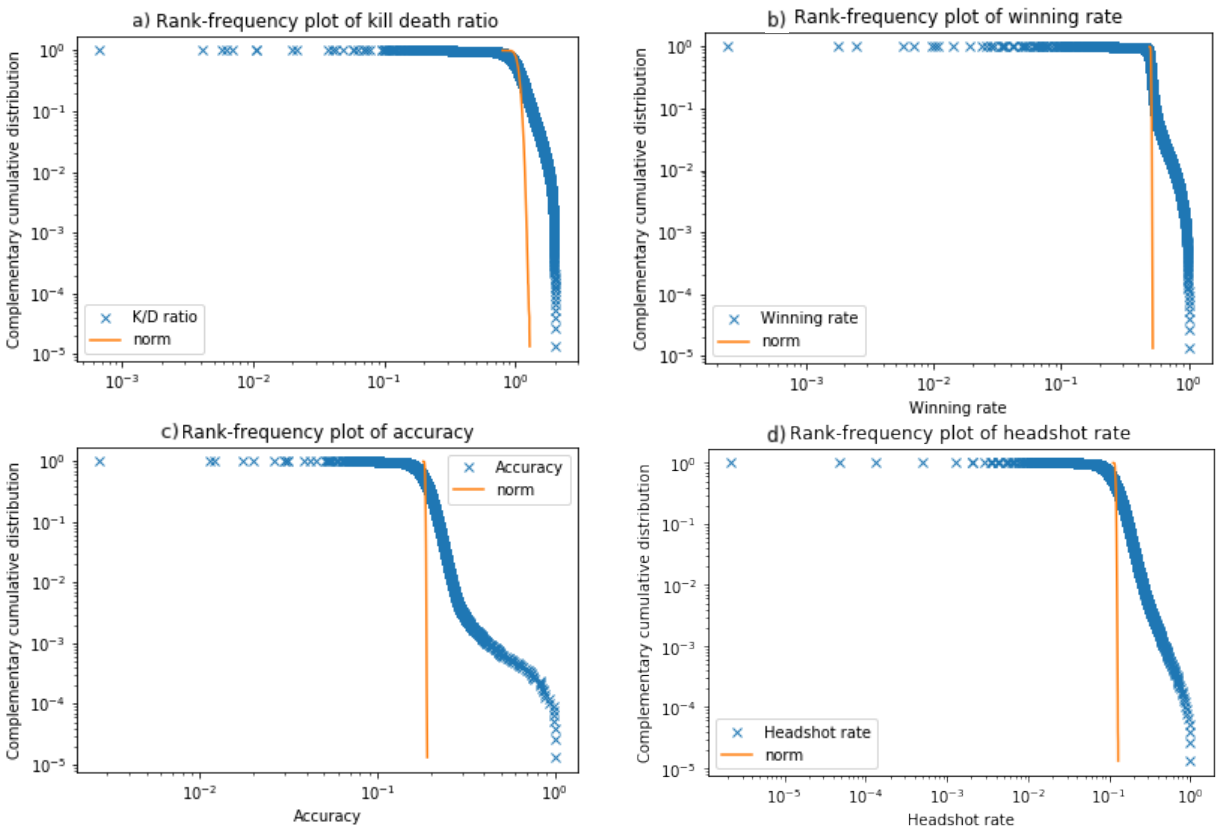


Figure6. Rank-Frequency Plot of Four Features

Feature	Optimal Value to Split the Tail	Fitted parameter α	P-value for the main body
Kill/Death Ratio (KDR)	2.0150	3.4834	Fit T: 0.027
Wining Rate	0.5683	7.8031	Fit T: 0.073
Hit/Shot Rate	0.2036	11.9658	Fit T: 3.299*10-19
Headshot Rate	0.1526	6.2518	Fit Lognorm: 9.996*10-6

Table6. P-value for main body and the Tail index

3. Overall Performance Similarity between Team Members and Free Players

Suppose we have already filtered out the players who do not satisfy the hard requirements set by a particular team (eg. team 780, team members' id are 334, 827, 844 and 849), it is better to visualise the performance of each player in the system. However, as the performance features are within a 4-dimensional space, we need to embed the the high dimensional performance data into a smaller dimensional space (two dimensions). Figure7 visualises each player's overall performance by conducting dimensionality reduction. The blue cross represent the pre-selected 400 free players and the orange dots represent the four players in the team. Intuitively, the closer the blue cross located to an orange point, the more similar the two players' performance are, and what we are more concerned is to identify the players with comparable abilities with the whole team.

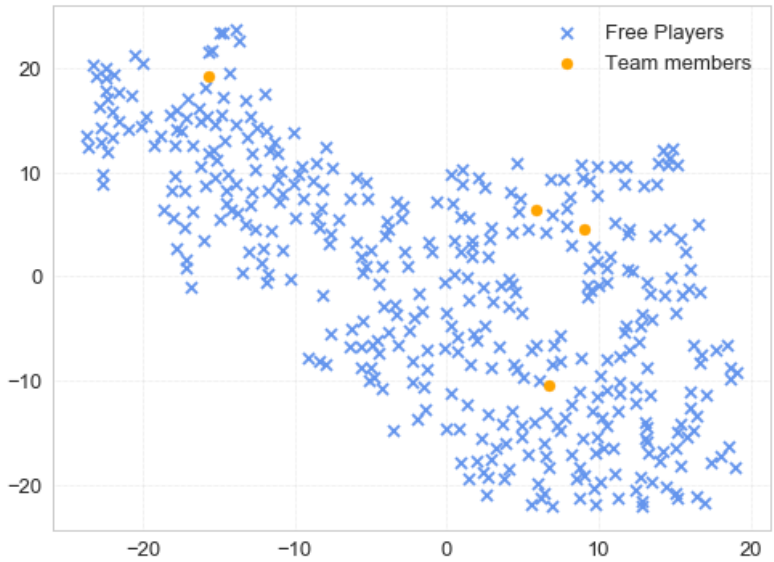


Figure7. Two-Dimensional Visualisation of Player Performance

Figure8 help us to deal with this issue. The performance Euclidian distance matrix of the team members and the 400 free players are computed as follow. Each numbers on the mosaic stands for the distance between a pair of players and the lighter colour represents a smaller distance, ie. more similar to each other. When we sum up over the columns, we can get the total distance from each player to the four team members. Therefore, player no.3, no.5, and no.396 in the above figure are the ones we want to selected. After this process, 200 players with relatively small summed distance will be selected.

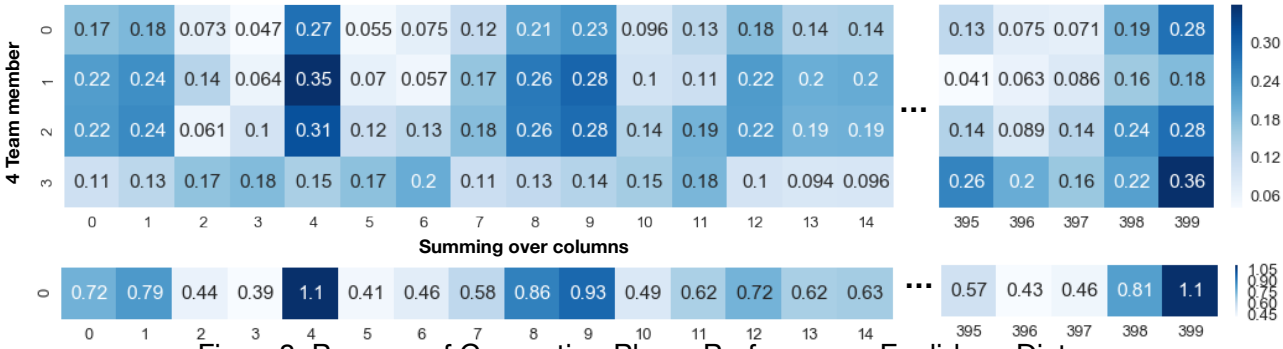


Figure8. Process of Computing Player Performance Euclidean Distance

4. Determine the coefficients of role score formula

In order to calculate each player's five role score, we need to determine the coefficients for score formulae. Based on the idea we proposed before, we first select players who only played on one role to compose our training set. The number of pure snipers is 3362, and we also have 7069 pure assaults and 794 support players. However, none of the players only played on captain or strategist. Therefore, we just select set of players who once played captain or strategist but may also acted as other roles.

The Pearson correlation coefficients between factors in the equation and role performance are shown on the left-hand columns of Table7. Accordingly, the α, β and γ can be computed by proportionally transforming the correlation coefficients into a suitable scale (sum is 1). The results are displayed on the right side of the table. Once the formula has been settled, the five role skill scores for each individual can be also obtained. Since the range of Accuracy, KDR and Headshot rate are not exactly the same, we will first normalise each of the value, putting them into the same scale, then multiply by their corresponding coefficients. The final score will be adjusted to fit in the range between 0-10.

To further explore the different Pearson coefficients, we can observe that all of the variables have positive correlations with roles' win rate. The weapon shot accuracy can be regarded as the main criteria to measure snipers' performance as the proportion of the accuracy coefficient is the highest among other factors. However, accuracy is not a promising factor for any other roles. For assaults, the higher KDR and a good headshot rate could promote a better performance. However, for captains and strategists, it is hard to distinguish one of them from another only by using the three parameters as the three coefficients are almost of equal importance on the last two tactical roles. In spite of this, the scores for sniper, assault and support can still effectively assist us in assessing players' role skill performance to some extent.

	Correlation Coefficients				Coefficients in the formula		
	Accuracy of Weapon set	KDR	Headshot		α	β	γ
Sniper	0.456832	0.605732	0.456736	—>	0.4	0.3	0.3
Assault	0.193911	0.721780	0.496960	—>	0.14	0.51	0.35
Support	0.229335	0.513527	0.367690	—>	0.21	0.46	0.33
Captain	0.235415	0.595073	0.363498	—>	0.2	0.5	0.3
Strategist	0.250470	0.601793	0.365731	—>	0.21	0.49	0.3

Table7. Correlation Coefficients between Each Parameter in Each Role Score Formula and Win Rate
Right: Tunned α, β, γ

5. Visualisation of the Current Team performance and Future Team performance

Based on our role performance scoring system, each of the current team members will be assigned five scores and the team's overall score of different roles will be also computed and displayed in a visual diagram (See Plot a) in Figure9). In this way, the team players can identify their strengths and weaknesses. In terms of our specific example, the lack of a proper assault greatly reduces the team's overall combat ability. Therefore, our system will immediately generate scores for the 200 free players selected by the previous steps and only return the top ten players with stronger assault skills. Plot b) presents us a situation when one of the recommended players joins this team. Even though the support score and sniper score have been affected slightly by the new player, the overall team performance is enhanced to a great extent.

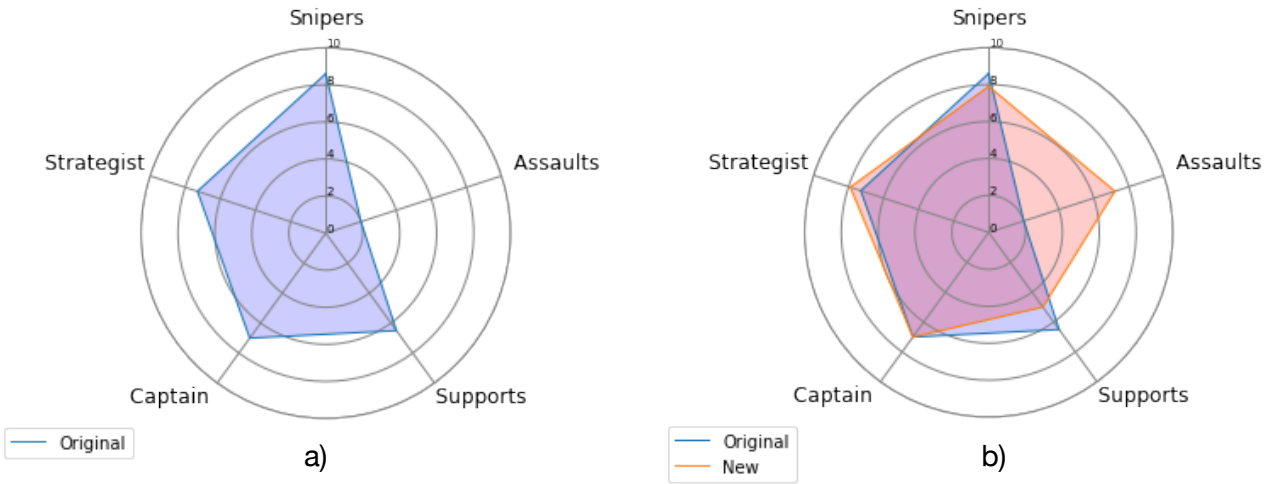


Figure9. Team Performance Visualisation (Before and After)

Conclusion

In this coursework, we constructed a player recommendation system to effectively suggest needed players for teams. The whole system workflow involves a simple player filtering process, team members and free players' overall performance comparison, players different role performance rating and assigning players with suitable role skills to the team.

This report recorded the whole process of how we implement the system step by step. We first analysed the existing CS:GO team players' social network to get an intuition of how team are formed. Our finding is that the degree distribution of the players' social network could be well approximated by a power law and players' cluster coefficient distribution suggests us most players only joins one team over the previous few months. Then, we move to analyse CS:GO players' overall performance and role performance. By conducting the basic exploration of the database, we generated four features (kill death ratio for combat capability, win rate for overall estimation, hit/shot rate and headshot rate for shooting accuracy) to assess players' combat ability. Each of the them has been detailed

analysed in the above sections. The overall performance similarities are measured by calculating the Euclidean distance between the team players and free players. Players with shorter distance to the team are the ones we seek after. To evaluate the players' role performance, we worked out five formulae which could measure how well the gamer can perform when playing specific roles. Each formula is composed of same variables but associated with different weights. The weights are determined by the correlation coefficients between each variable and role performance. We discovered that the sniper performance is more determined by shot accuracy whereas assault performance is more correlated to KDR. Till now, the team's role strength and weakness can be identified by calculating the role scores and the final recommendation can be achieved by suggesting list of candidates with strong complementary role skills with the team. This is the whole idea of our research.

Retrospective

Although we have developed our player recommendation system, which could simplify user experience of finding suitable teammates, there are still some insufficiencies in the role scoring process.

For one thing, the coefficients for each factor in our role score formulae are determined by the Pearson correlation, and the data applied to compute the Pearson correlation coefficients belongs to all one-role players. Take the sniper score formula as an example, we selected people who only acted on snipers before, and calculate the Pearson correlation coefficients between their weapon use accuracy, KDR, headshot rate and their winning rate to determine α, β, and γ. However, even if they are all snipers, their rank levels are different. Snipers with higher ranks are more skilled than the lower's and thus, they may place different emphasis on the three variable. This is what we need to discover in further steps.

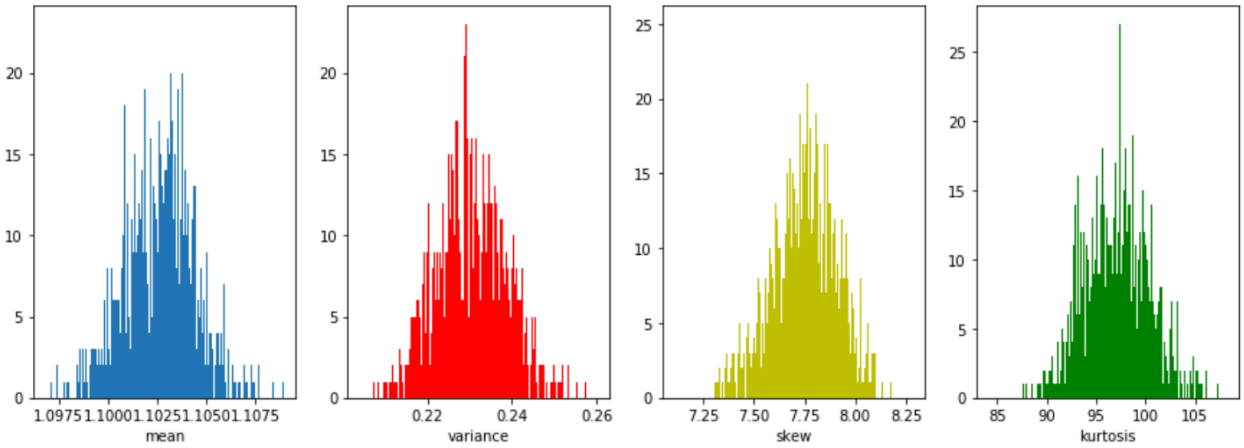
For another, it is quite difficult to measure the performance of a strategist or a captain as our current score formulae for the two roles are proved to be very similar. In fact, if a player hopes to take up these roles, focusing on improving shot accuracy or KDR is far from enough. Both of them are tactical roles which require comprehensive ability to effectively coordinate the whole team and work out tactics to achieve their ultimate goal. In order to assess their skill performance, we might need to consider other parameters like their ability to make calls and devise flexible strategies.

Last but not least, we now use the average role' scores to represent the overall team skill and our current goal is to improve the average team performance. However, in reality, as each player has their own preference and advantage in playing specific roles, and a team is formed to enable each team member to play to their own strengths, we would better to use the highest score of team players to represent the team's overall performance on each role. If we could solve these issues in the future, our system could be improved and generates more benefits for eSports players.

Appendix

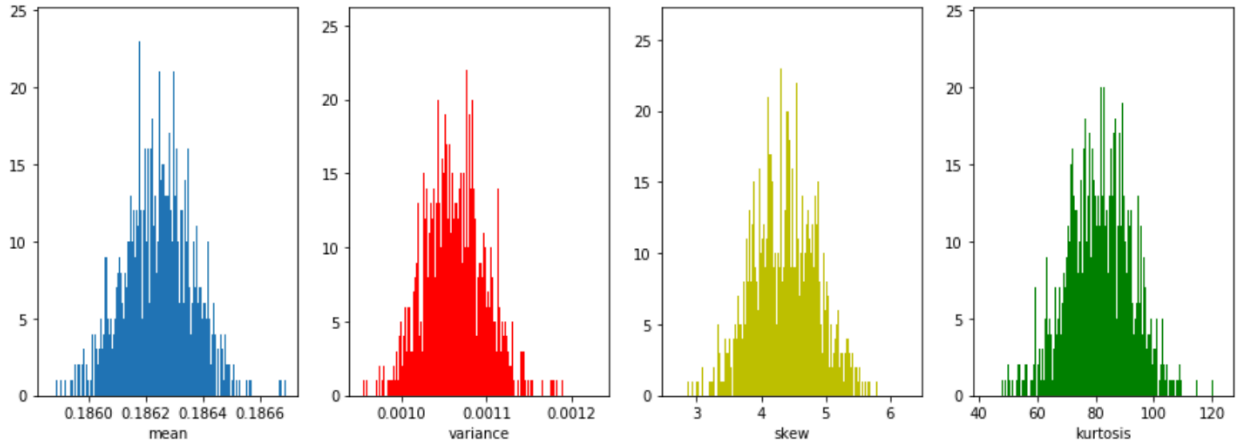
Confidence Level for Three Features

Kill/Death Ratio



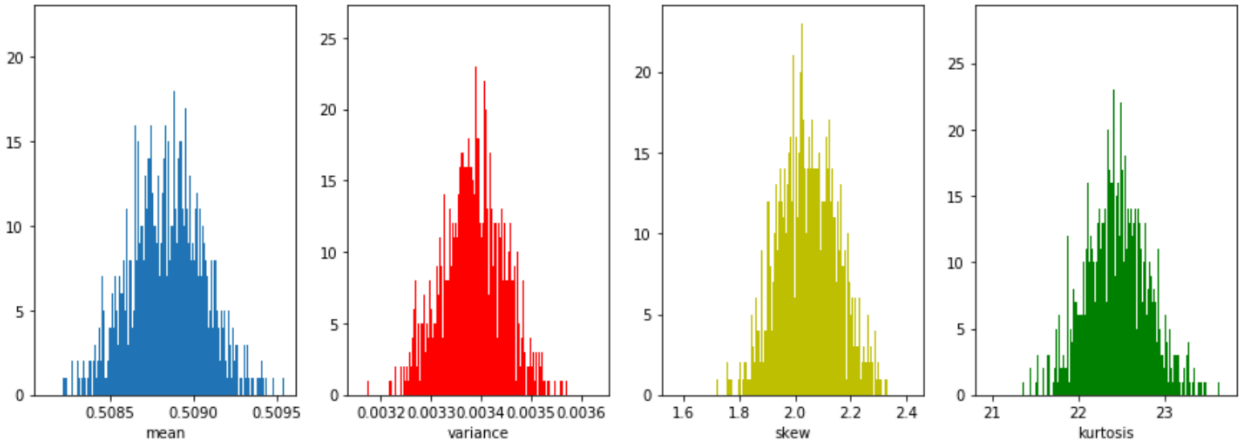

```
confidence interval for the mean:[  1.0993 - 1.1061]
confidence interval for the skew:[  7.4288 - 8.0468]
confidence interval for the variance:[    0.2139 - 0.2461]
confidence interval for the kurtosis:[ 90.6183 - 103.5524]
```

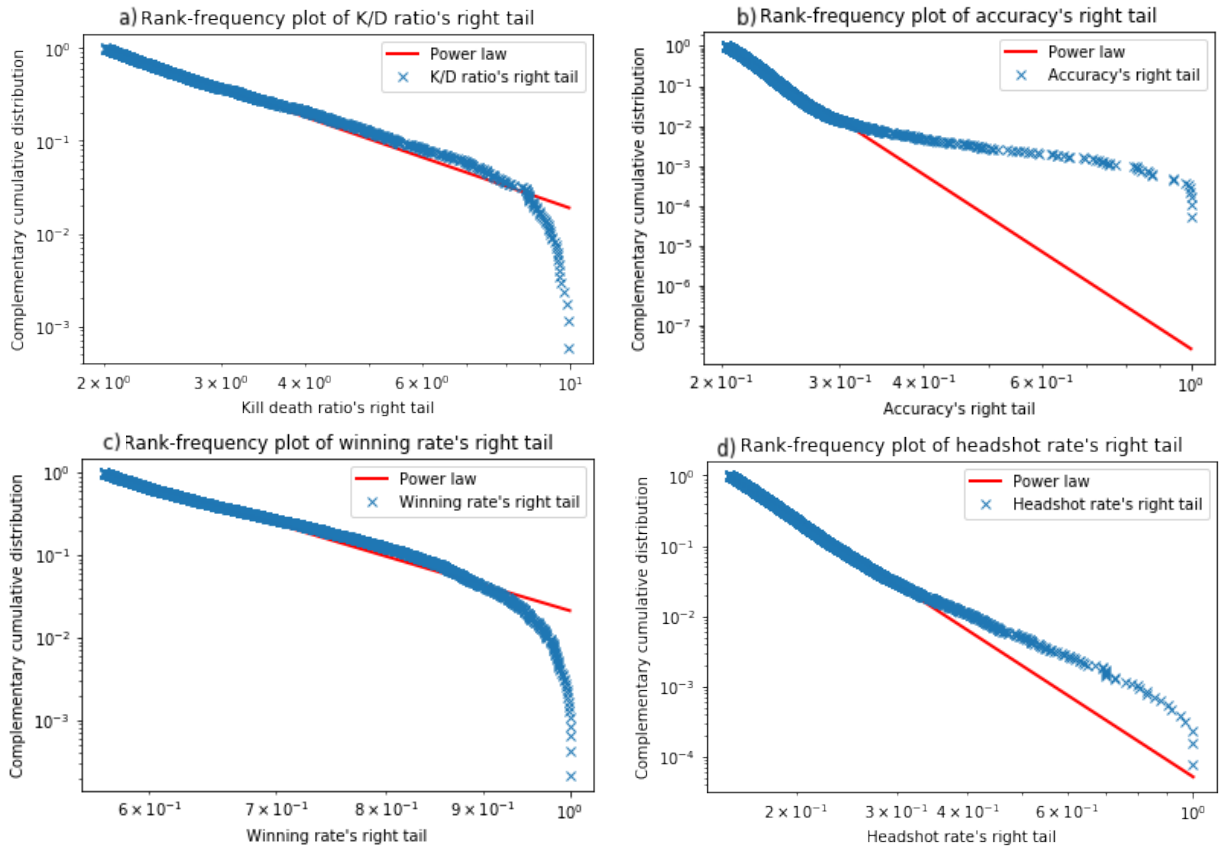
Kill/Shot Rate




```
confidence interval for the mean:[  0.1860 - 0.1865]
confidence interval for the skew:[  3.4013 - 5.3371]
confidence interval for the variance:[    0.0010 - 0.0011]
confidence interval for the kurtosis:[ 59.7971 - 103.0568]
```

Winning Rate




```
confidence interval for the mean:[  0.5084 - 0.5093]
confidence interval for the skew:[  1.8390 - 2.2567]
confidence interval for the variance:[   0.0033 - 0.0035]
confidence interval for the kurtosis:[ 21.7193 - 23.1797]
```

Tails for the Four Features