Data Analytics COMPG011

3rd Assignment

Due: March 29, 2018

Professor: Tomaso Aste

Name: Jenny Yang

Introduction

As emphasized before, teaming up with the right people is the key to winning and ranking up in competitive e-sports. Professional team member recruitment in CSGO has a very similar dynamic as the job market in our real world where more skilled players are generally more valuable and attractive to team owners. However, a rule on the DreamTeam platform that each play can only belongs to one team at any point of time and the norm that free players tend to join the strongest team available, have led to a very competitive market for team recruitment.

For some team owners who just started forming a team, it would be hard for them to recruit the very best experienced players with high ranks. Therefore, their next best option is to recruit the less experienced from lower ranks who have great potential to rank up quickly. Although the ranking system in CSGO would mainly reflect the players' performance in the game, there are still other non-performance related rules governing a player's value.

In this paper, I created an algorithm to identify high potential talents who are being undervalued by the games ranking system on early stage, so that professional team owners can target these players and have a better chance to get the matching talents before they are taken. This can also be used to check new players' potential ranking who have not collected 10 wins to get their first placement.

Keywords

Network correlation matrix; Global clustering coefficient; Tree based classification models; K-fold cross validation, High potential players identification.

Dataset

The data set used in this study is provided by the Dream Team containing data drawn from their platform, where strategic teams are formed, as well as Steam, the platform where the game is played. The database was shared in .sql format, which can be opened in PostgreSQL server. It consists in total of 67 tables categorized into five schemas covering different aspects of data over a 3-month period (09-10-2017 to 05-01-2018). Content of the 5 schemas can be summarized as below.

- **Core**: information from Dream Team's platform including platform users profiles, team vacancies, followers and reviews etc.
- **Csgo**: data about the players activities in the game, such as ranks, playing performance and weapons stats.
- **Data and Data Hardware**: all metadata with the former focusing on general data like locations, genders and languages and the later focusing on supplementary hardware data like chair, mouse and headset used etc.
- **Game**: has both metadata and basic data about the players for example, characteristics, roles and ratings etc.

Several main tables used in this analysis includes csgo.profiles, core.users, core.profiles, csgo.players_stats_log, games.profiles_servers, core.users_langs, csgo.maps_stats and csgo_weapons.

Idea and Methodology

In this analysis, we are trying to reclassify the players based on the features that we believe that would show their true capability in this game. It is important to note that our model is not a replacement of the current ranking system, but rather a assisting tool working under the current ranking system to recognize undervalued players in this game.

CS:GO follows a modified Glicko-2 ranking system that a beginner to the game must first accumulate 10 wins in competitive games to receive their ranking placement and then start building their rank from there and only playing in the competitive mode can one earn or lose rank. Many factors such as kills, deaths, MVPs, assists etc. determine one's rank but the points one can earn is proportional to their opponent's rank. The hierarchy is comprised of 18 ranks with more players in the higher ranks and the highest population in rank 11. European players are the best at this game with the highest average rank of 11.27, followed by Middle East as the second with average rank 10.90 and Asia as the third with average rank 10.16.

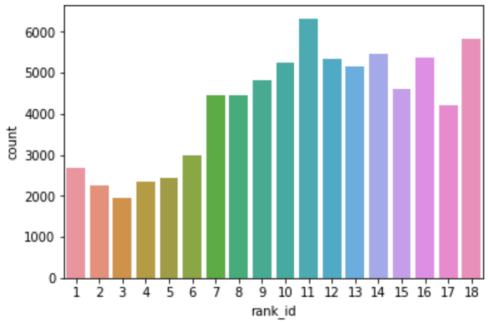


Figure 2. Global Rank Distribution

	region_id	region_name	rank
0	4	Europe	11.272064
1	5	Middle East	10.903522
2	2	Asia	10.157303
3	1	Africa	9.922145
4	8	South America	9.360148
5	6	North America	8.226732
6	9	The Caribbean	7.805556
7	7	Oceania	7.194499
8	3	Central America	5.747423

Table 2. Regional Rank Ranking

"A high potential player" is defined as someone who is currently being undervalued by the ranking system. It could be due to special rules other than their skills and performance in the game, for instance, new competitive players must accumulate 10 wins in competitive games before receiving their ranking placement. There have also been cases where players have been unexpectedly de-ranked due to being inactivity under the currently ranking system in CSGO. Although the company has not disclosed the official ranks calculation methodology, we know it is based on the modified Gilcko-2 system and there is a time factor in the term rating deviation in the Glicko-2 formula which explained the de-ranking. However, it may take longer to regain the rank even though the player's true ability has not worsened because now the system is matchmaking you back with lower rank opponents, consequently gaining them less point in each win. [1]

The first step is to broadly select all potential features from different tables. Secondly, we check the correlations by creating a network between the features. We will then perform detail statistical analysis on features that shows strong correlations to the rank. After we got the shortlisted features, we created several machine learning classification models for them to train. The players will be re-classified into rank groups by our model. The 18 ranks are divided into 3 groups and "early stage" is referring to group 0. (Group 0: Level 1-6; Group 1: Level 7-13, Group 2: Level 14-18) Further details will be discussed in the Model section below. Next, we evaluate the results and their accuracy to choose the best model for our final predictions. We will use our model to identify a list of the top potential upcoming players.

Features and Statistical Analysis

There are 3 main categories of features we looked into which are engagement, background and skills features because the first intuition of a high potential player is that he/she should be very engaged to the game, demonstrate high skills and likely to be coming from places where this game is more popular. All of the statistical analysis was done in python, utilizing the functions in the *scipy.stats* package *seaborn* library for graphs.

Engagement features

For engagement features, we looked at playing frequency, players' goal and number of registered servers. These are expected to have a causality relationship with their ranks, because generally more practice causes better performance. Playing frequency was calculated by dividing the number of days played with total observed days (88 days). The distribution is skewed to the right with a fat tail declining in a rate of a chi2 distribution. Goal is referring to player's attitude in playing this game (1- play for fun; 2 - play in amateur leagues; 3 - become professional player/team). Number of registered servers is considered a potential relevant feature because the more servers they registered, the more chances they get to play matches against players with different style which translates to the more adaptive and eager to rank up they are. From the bar chat we can observe that players in higher rank groups tend to have >1 server registered.

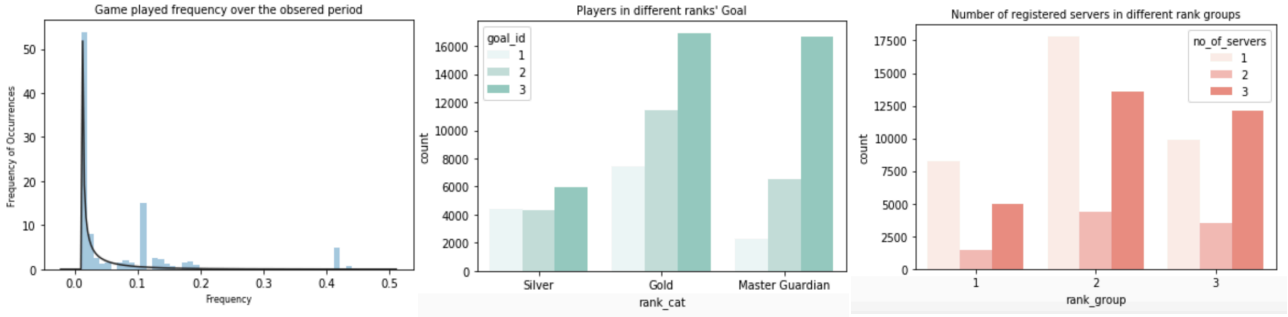


Figure 1: Histograms of played frequency, goals and registered servers

Background features

For background features, we looked at age, languages and countries. The game has received huge popularity around the world since its release back in 2012 with it being most popular among European and having second and third most players in South America and North America respectively. The majority of its players are teenagers with the most population in the 15 year-olds consisting 13.3% of all players. Although the range for players' age is large which goes from minimum 11 to maximum 69 in our dataset, the distribution is a fat-tailed distribution which actually declines as a power law in the tails. The right skewness of the distribution shows that players are more concentrated in the younger age group.

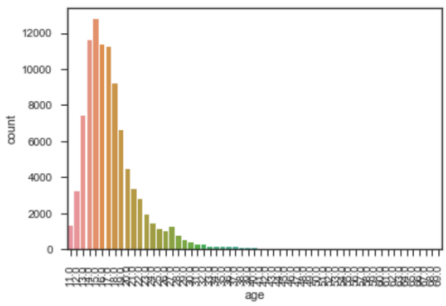


Figure 2. Age Distribution

	region_id	count	region_name
0	4	60313	Europe
1	8	9485	South America
2	6	7392	North America
3	2	4895	Asia
4	7	1018	Oceania
5	5	653	Middle East
6	1	578	Africa
7	3	194	Central America
8	9	72	The Caribbean

Table 1. Regional Count Ranking.

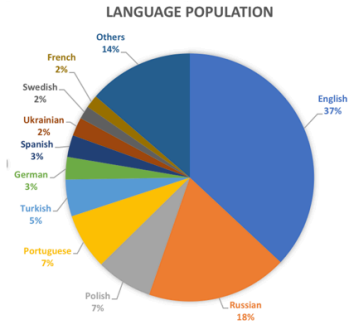


Figure3: Language pie chart (plotted with excel)

We observed that these three criteria: age, language, and country, are quite important features for team owners as selection criteria. For language, the more one can speak, the more vacancies they qualify for. Similarly, there are more opportunities in the form of team vacancies available from countries where the game is more popular. In processing the 2 category features (language and country), we have to quantify them by calculating their frequency value. For each language we calculate their frequency amongst the whole population of players, then for each player, we sum up the frequency value of each language they speak as their language score. The same calculation is done for country except each player belongs to one country, that the frequency value will be the same as their country score.

We expected these three features to be related with the ranks. They do not necessarily have a direct causality effect but we check if they are independent with the ranks or not by their covariance. Covariance measures how 2 variables vary together by calculating the difference between the joint probability and the product of the marginal probabilities. Our results show 2 features have positive non-zero covariance with the rank: Language score (0.1643) and Country score (0.0475) while Age has 0 covariance meaning it is independent from rank thus will not be considered further as our feature. However, covariance can only detect dependency, but not how strong 2 variables are related. We will further examine the strength of their relationship by checking the correlation in the next step.

Covariance:

$$Cov(x,y) = \iint (P(x,y) - p(x)p(y))dxdy$$

$$Cov(x,y) = E[(x - E[X])(y - E[Y])]$$

Skills features

For the skills features, we considered 6 game performance metrics in total and performed statistical analysis and studied their distributions. Due to the limited space in this report, we included results for 4 of them which are mvps/win, headshot/kills, shot-hit/shot-fire and kill/death. The other two are wins/round, and headshot/round. Mvps/win refers to how many times a player leads his team to win a game as the most valued player (mvp). It reveals an individual player's true quality without the luck factor of being carried. Headshot/kills refers to how many times a player kills an enemy with headshot which reflects one's killing ability. Shot-hit/shot-fire is one player's shooting accuracy which is correlated with what weapons one uses. Kill/death rate is a general skill metric commonly used by the gamers community.

We first plotted histograms with the raw data with bin-size = 5000 and calculated their 4 moments (mean, variance, skewness and kurtosis) to observed the original distributions and realized they are all heavily skewed to the right with positive skewness values. All have positive >0 excess kurtosis (fisher method), which means they all have fatter tails than Normal Gaussian distribution demonstrating leptokurtic properties. Because of such characteristic, we took logarithm of these variables to normalized the distribution for easier visualization of the analysis results. Comparing the 4 moments before and after the transformation, we observe some of the means have gone from positive to negative and the magnitude of skewness and kurtosis have been reduced.

	Shot_Hit /Fired	Shot_Hit/ Fired (log)	Kill_Death	Kill Death (log)	MVP_ Wins	MVP Wins (log)	Headshot _Kill	Headshot_ Kill (log)
Mean:	0.19	-1.69	2.38	0.04	0.25	-1.41	0.37	-1.03
Variance:	0.0001	0.02	57757.90	0.08	0.01	0.10	0.08	0.09
Skewness:	2.94	-0.32	230.71	3.96	1.38	-0.27	225.91	-5.52
(Excess) Kurtosis:	50.16	11.60	56139.01	111.36	5.78	11.53	57837.83	129.78

Table.2: Key Features Statistics

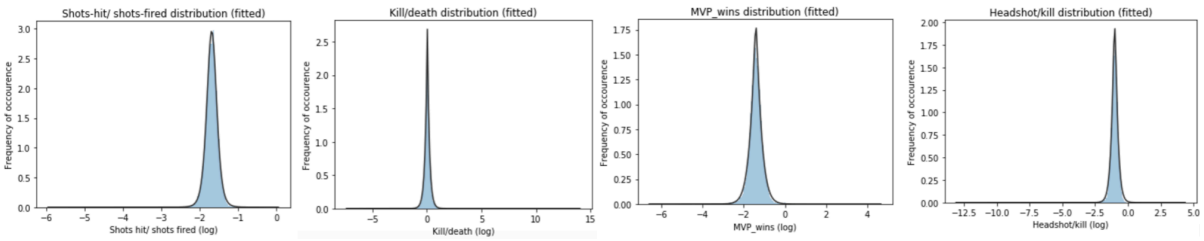


Figure 4: Fitted distributions of the log of the 4 key performance features

Mean (1st Moment):

$$\hat{\mu}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Variance (2nd Moment):

$$var(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{\mu_x})^2$$

$$where\ it's\ square\ root\ is\ the\ standard\ deviation\ \hat{\sigma}(x) = \sqrt{var(x)}$$

Skewness (3rd Moment):

$$\hat{\xi}(x) = \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i - \widehat{\mu_x})^3$$

Kurtosis (4th Moment):

$$\hat{\kappa}(x) = \frac{1}{\sigma^4}\sum_{i=1}^{N}(x_i - \widehat{\mu_x})^4$$

As aforementioned, all 4 distributions are heavily skewed with fat tails, so we study the tails and the body separately. For each variable, we compute the parameters and log likelihood against all selected potential fitting distributions including Normal, Student-t, Gamma, Laplace, Pearson, Pareto and Weibull. We chose the one with the largest log-likelihood value (MLE method) and double check with running a goodness of fit test. However, since the Kolmogorov-Smirnov (KS) test is too sensitive to any subtle difference in the shape between 2 distributions, due to the fat tails, all p-values presents to be 0 or close to 0. Instead we ran an alternative test, the Wilcoxon-Mann-Whitney test, which is also a nonparametric test but based on ranks whose null hypothesis is that the 2 samples have the same distribution. [3] The main difference between the Wilcoxon test and KS test are that the Wilcoxon test focuses on detecting the shift in the median, which would serve the purpose in our case when we just want to fit the body.

W test statistic:

$$W = \sum_{i=1}^{N}\left[sign\left(x_{2,i} - x_{1,i}\right) * R_i\right] \quad where\ R\ is\ the\ rank$$

The resultant p-values are greater than the critical level at 0.05 for 95% confidence allowing us to accept the null hypothesis of no difference between 2 tested distributions which implies our selected distributions are good fits to these variables. From the QQ plots and rank frequency plots below, we can see the distributions fit very well in the body parts with disparities in the tails. For the tails, we computed the tail index (α) for each side and observed the left tails for kill/death(log), mvp/wins(log) and head-shot/kill(log) are actually thin tails with $\alpha = \infty$. The rest presents to be power law tails. [9]

	Shot_Hit/Fired (log)	Kill_Death(log)	MVP_Wins(log)	Headshot_Kill(log)
Alpha_Right:	1.0342	0.4528	0.3035	0.1772
Alpha_Left:	0.8349	inf	inf	inf
Fitted distribution (body):	Normal	Laplace	Student-t	Weibull
Parameters:	Mu: -1.69 Sigma: 0.16	Location: -0.04 Scale: 0.18	Location: -1.41 Scale: 0.24	Location: -1.01 Scale: 0.21
Test Statistics (W):	1386993836	1392758322	1394267594	1394258850
P-value:	0.19126543	0.42832003	0.59173735	0.5907139

Table 3: Body and tails statistics

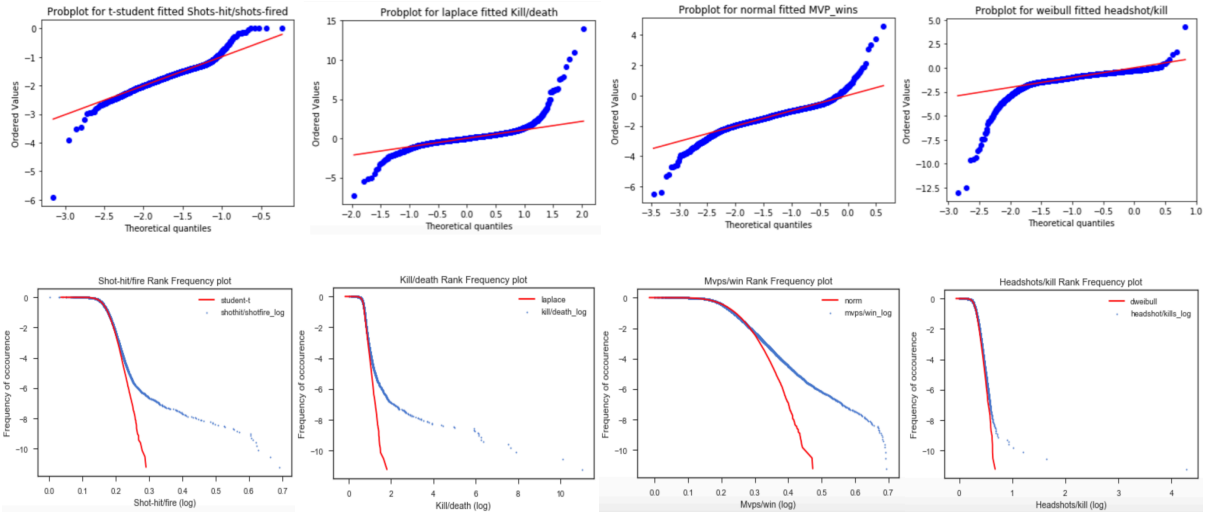


Figure 5: QQ plots and Rank Frequency plots

Tail Index:

$$\hat{\alpha} = \frac{N}{\sum_{t=1}^{N} \log\left(\frac{x_t}{x_{min}}\right)}$$

Weapons

Additionally, we looked at weapons data. For weapons, there are totally 33 different types. From the available data, we could find the accuracy of each weapons with M249 with 45.14% overall accuracy rate. Second most accurate weapon is G3SG1 with 44.96% overall accuracy followed by Scar20 with 38.75% overall accuracy. However, each weapon fits each player differently, therefore, we looked at individual shot/hit rate by using the pivot function from *pandas* library to create a pivot table. In the next step features filtering, each weapon will be considered as an individual feature.

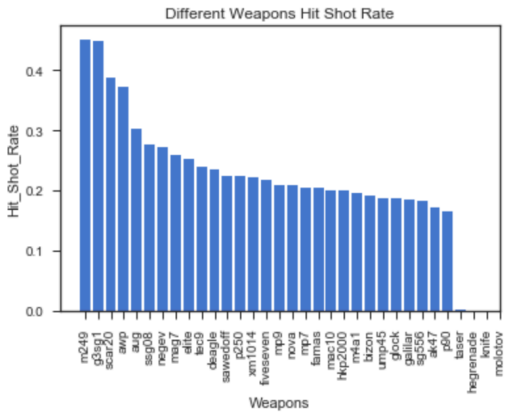


Figure 6. Historgam of Weapons Hit Shot Rate

Data cleaning and filtering

For data cleaning, we first clean up the noise data by dropping all entries with null in total_time_played and total_round_played because these are not valid data points containing no information about the players' performance. Then I filled the null with 0 in other columns including total_kills, total_kills_headshot, total_shots_hit, total_MVPs, total_wins, and total_shots_fired. Except for total_deaths, whose null were replaced by 1 instead. Before calculating the performance ratios, 5 more logic checks listed below were done to ensure the data quality. Eventually that leaves us with 75898 data points for the analysis.

1. total_mvps < total_round_played
2. total_wins < total_rounds_played
3. total_shots_hit < total_shots_fired
4. total_kills_headshot < total_shots_fired
5. total_mvps < total_wins

Full data was used for statistical analysis above but for model training data, reliability of the labels (system's ranks) is ensured by using only data from active players which is the body of the distribution of play_freq. We excluded the both mid outliers and extreme outliers using data within the 3-interquartile range. (Q1 - 3*IQR to Q3 + 3*IQR)

After reviewing all the potential features, we perform the final filtering to shortlist the features that we will actually use in our model training. To do that we calculated the correlation between our selected features and the ranks. Since there are too many features, we will visualize weapons features separately. First, we look at the 3 main types of features and create a correlation matrix with Spearman method. We chose Spearman over Pearson method because the feature functions are not liner but generally monotonic. Also, since we are fitting observation data points from a 3 months sample set, a non-parametric measure of correlation will be more suitable.

Spearman Correlation Coefficient (ρ_s)

$$\rho_s = \frac{6}{\pi}\arcsin\left(\frac{\rho}{2}\right) \quad where\ \rho\ is\ Pearson\ Correlation\ Coefficient$$

Pearson Correlation Coefficient (ρ)

$$\rho = \frac{\sum_1^n (x_i - \widehat{\mu_x})(y_i - \widehat{\mu_y})}{\sqrt{\sum_1^n (x_i - \widehat{\mu_x})^2 (y_i - \widehat{\mu_y})^2}}$$

To form an adjacency matrix, we set the threshold as $\tau = 0.1$ and then set all diagonal elements and matrix entries less than the threshold to 0. We use this data to form a directed weighted network graph to visualize the correlations of the features. The length of the edges are distances between nodes (features) calculated as $d = \sqrt{2(1 - \rho_s)}$ where ρ_s is the Spearman correlation coefficient. [6] The closer the nodes, the stronger correlations. 2 colors are to show the opposite directions of the edges with red representing negative correlation and green representing positive correlation.

In the first network, we can observe all of our selected features have positive correlations with rank_id except country ratio as it is not connected with the rest of the network due to its correlation coefficient being less than threshold. It serves as counterevidence to our initial perception that people from popular counties tend to have higher rank. Therefore, country ratio will not be included in our final feature list for training.

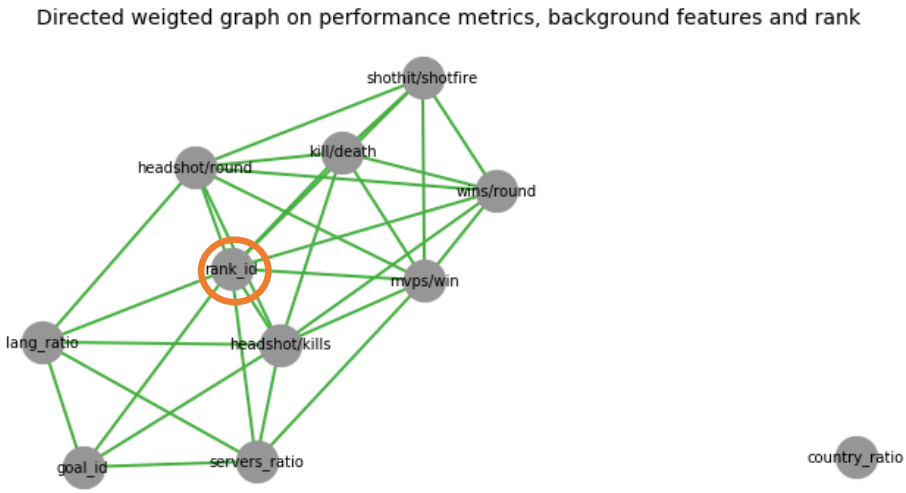


Figure 7: Features Network – Performance Metrics and Background Features

Next, we move on to the weapons network which is the second network we draw with some non-related weapons removed from the observation in the first attempt. In this network there are more negative direction edges which means they are anti correlated to rank, in other words, the more they use those weapons, such as p250, ak47 and aug, the less likely they are with high rank. Similar to the country ratio, as m249 is completely disconnected with the rest of the network, it shows no significant correlation with the rank, thus not strong enough a feature to be used in our final training.

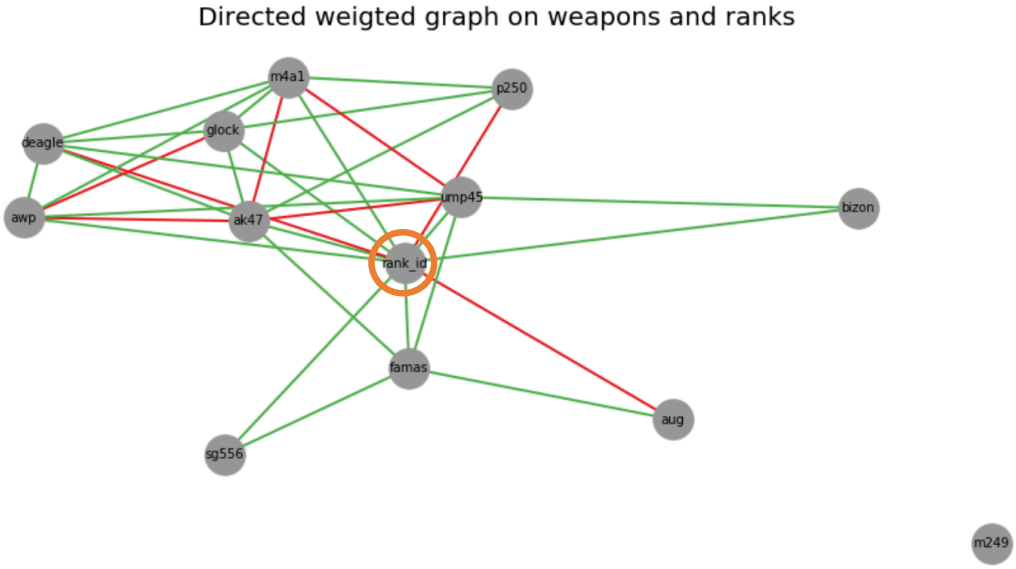


Figure 8: Top Weapons Features Network

Additionally, we calculated the clustering coefficients and eigenvector centrality for each selected feature to study the properties of the network graphs.

There are two types of clustering coefficient which are the local and the global. Local clustering coefficient measures on node levels (features in our case) how well their neighbors connect with each other, while global clustering coefficient measures the overall degree of clustering in the network which can be calculated in 2 ways: either to take the average of all the local clustering coefficients or to calculate the ratio of close triplets among all connected triplets. Both of our networks have clustering coefficient around 0.8 which is considering high within the range 0 and 1 which means our features are very well connected together, with significant correlations.

Eigenvalue centrality measures the influence a node has in the network. In the first network, language ratio and mvps/wins have the highest eigenvalues centrality meaning they have more influence over other nodes. It makes sense in our context as both are more fundamental qualities that players who have high language talents and mvp ratios are generally the top players and are more likely to have high measures in other qualities as well.

Local Clustering Coefficient:

$$c_{local} = \frac{\#\ E\ between\ neighbours}{k(k-1)/2}$$

Global Clustering Coefficient:

$$C_{avg} = \frac{1}{v}\sum_i C_i \qquad C_{Global} = 6\frac{\#\ of\ closed\ triplets}{\#\ of\ paths\ of\ length\ 2}$$

Eigenvalue Centrality:

$$X_i = \lambda^{-1}\sum_j A_{i,j}X_j$$

Properties results (Performance and Background Features):

	0	1	2	3	4	5
Features	average	rank_id	goal_id	servers_ratio	country_ratio	lang_ratio
Sum of Adjacency Matrix		9.00	4.00	5.00	0.00	5.00
Clustering Coeff	0.8244	0.6111	1.0000	0.8000	NaN	0.8000
Eigenvalues Centrality		0.4126	0.0917	0.3384	0.1817	0.5244

	6	7	8	9	10	11
Features	mvps/win	wins/round	headshot/kills	shothit/shotfire	kill/death	headshot/round
Sum of Adjacency Matrix	7.00	6.00	8.00	5.00	6.00	7.00
Clustering Coeff	0.7619	0.9333	0.6429	1.0000	0.9333	0.7619
Eigenvalues Centrality	0.5467	0.3162	0.0240	0	0	0

Properties results (Weapons Features):

	0	1	2	3	4	5	6
Features	average	rank_id	ak47	aug	awp	bizon	deagle
Sum of Adjacency Matrix		7.00	8.00	0.00	6.00	0.00	6.00
Clustering Coeff	0.8371	0.6667	0.6071	NaN	0.8667	NaN	0.8667
Eigenvalues Centrality		0.3810	0.3348	0.0966	0.2724	0.4876	0.3684

	7	8	9	10	11	12	13
Features	famas	glock	m249	sg556	ump45	m4a1	p250
Sum of Adjacency Matrix	3.00	4.00	0.00	0.00	6.00	7.00	3.00
Clustering Coeff	1.0000	1.0000	NaN	NaN	0.8000	0.7143	1.0000
Eigenvalues Centrality	0.5345	0	0.1673	0	0	0	0

Table 4: Networks Properties

Models

5 machine learning algorithms including Adaboost, Random Forest, Extra Tree, Support Vector Machine (SVM) and Logistic Regression were tested in classifying the players' rank based on the selected features. K-fold cross validation was also performed to maximize trainings and boost accuracy.

Out of all 5 models, we used 3 tree-based models: Random Forest, Extra Tree and Adaboost whose based model is a Decision Tree. Random Forest is basically a bagging version of Decision Tree (building trees on bootstrapped training data) while Extra Tree (Extremely Randomized Tree) is essentially a variation of Random Forest. Tree-based algorithms are popular approaches for classification problem due to its simplicity in training and boosting. One advantage of using trees in our case is that they are capable in handling qualitative features, like Goals and Ranks, thus we would not need to convert them into dummy variables for training. At each split, a decision is made base on the criterion function, for which there are 2 options, Gini method or entropy method. Gini method calculates the Gini impurity index which measures the total variance across the classes and Entropy method calculates the information gain which is the also the Kullback-Leibler divergence. In most cases, two approaches give similar results, thus we have used Gini by default in our training which is comparatively more efficient. [4]

Gini Index:

$$G = \sum_{k=1}^{K} \widehat{P_{mk}}(1 - \widehat{P_{mk}})$$

Cross-Entropy:

$$D = -\sum_{k=1}^{K} \widehat{P_{mk}} log(\widehat{P_{mk}})$$

In model training, it is very important to avoid overfitting. In this exercise, since there are many features available in the data set, it is easy to keep on adding parameters to train a 'perfect' model. However, such model will not be the optimal choice, as it will not perform as well in datasets other than its training set. Overfitting was also the reason why the feature selection steps explained above are so important. Instead of adding new parameters, we improved our model by finding the optimal weight and depth for the parameters. To find the parameters, we ran multiple iterations of a model with different values to find the optimal ones that minimize the cost functions. Figure 9 shows the optimization results for our 2 best models, Random Forest and Adaboost. For Random Forest parameters, we used 35 samples leaf and 15 tree depth while for Adaboost, we used 0.9 learning rate and 50 estimators.

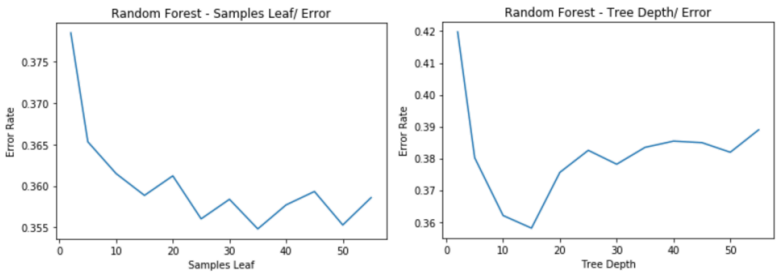


Figure 9: Optimization results for Random Forest

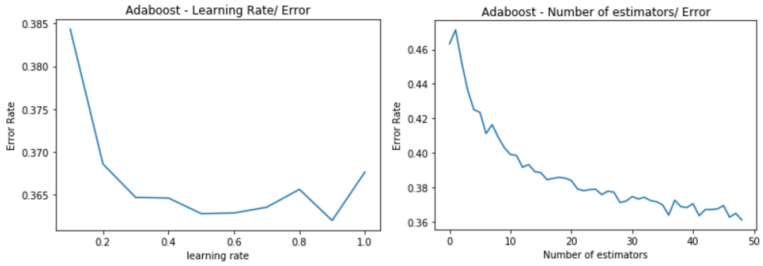


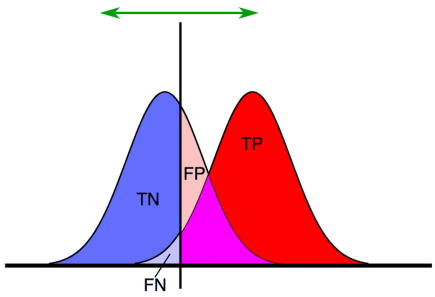
Figure 10: Optimization results for Adaboost

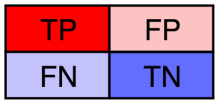
To increase training and boost accuracy, we utilized the K-fold function in *sklearn* to perform cross validation. The methodology of K-fold is to separate the total dataset into K groups with equal size and repeat the training K times. In each iteration, a different group of data is used as testing data with the rest as training data. The final result is the average of each iteration output. The method of cross validation is better than random shuffling as each data point will be used for the same amount of times, avoiding biases. We used 6 folds in our training consistently achieving 1% accuracy improvement in all models.

To validate the fitness of the models, we checked the area under the curve (AUC) of the receiver operating characteristic curve (ROC). The ROC is plotting the (TRP) true positive rate against the (FPR) false positive rate. The AUC is the measurement used to compare model fitness. Although typically ROC is used for binary classifier system, twisting the model's outputs into multiple binary classes allowing us to extend this test to multi-class classification. [2] In the below graphs, 4 ROC curves were drawn for each model, one for each class (solid color lines) and one for the overall average (dotted line). For simplicity we use macro-averaging here.

True Positive Rate and False Positive Rate:

$$TPR(Recall) = \frac{TP}{P} = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \qquad Precision = TP\,TP + FP$$





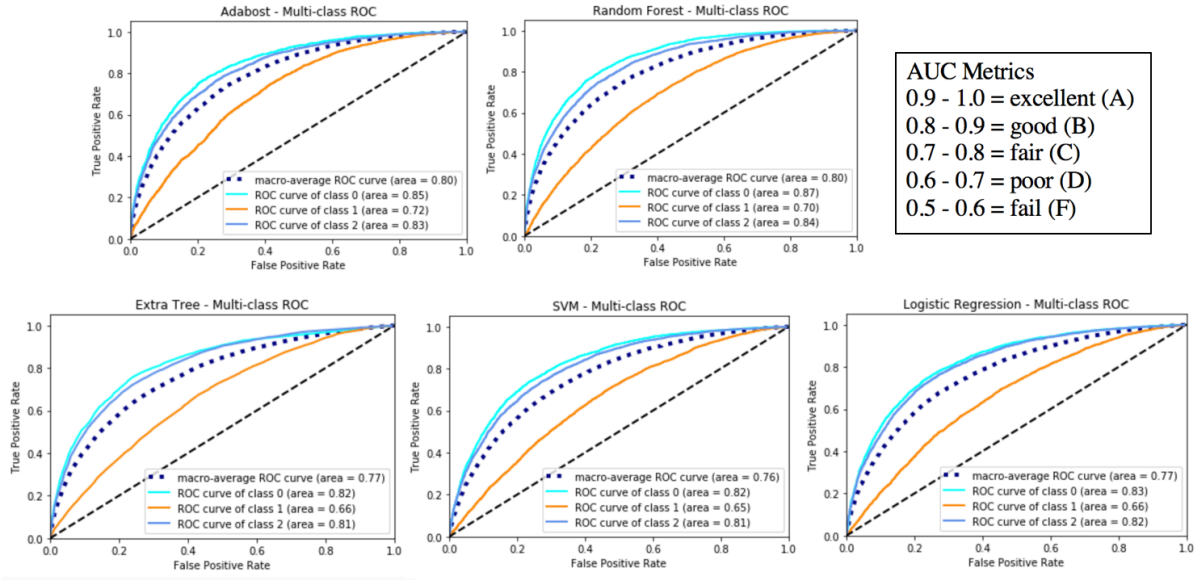
[5]

Area under the receiver operating characteristic curve:

$$A = \int_{\infty}^{-\infty} TPR(T)\left(-FPR'(T)\right)dT = \frac{G + 1}{2}$$

where T is the threshold and G is the normalized Gini Index

With the below metrics, we have achieved fair to good accuracy of 0.7- 0.8 for all of our models. All ROC curves are significantly above the 0.5 random prediction baselines which are the black diagonal dashed line in the graph. Adaboost and Random Forest have the best fit among all.



Figures 11: ROC graphs of all models and AUC Metrics [8]

Results and Conclusions

To evaluate the actual classification results, we used the accuracy score function from *sklearn.metrics* library. Accuracy = (TP + TN) / (TP + TN + FP + FN) while Error rate = (1- accuracy). Additionally, we looked at the F1-score to check how well our models predict. F1 score, otherwise known as the Dice similarity coefficient, measures the accuracy of a model by taking the weighted average of the recall and precision. Recall is the sensitivity of the model which refers to the ratio of true values selected by the model among all true values in ground truth. While precision is the ratio of real true value among all labeled true value by the model. In our case, an unweighted averaging (macro averaging) is used to calculate F1 score and since we have multi-class models, the final results are the average of the F1 score of each class.

F1 Score:

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{Precision}} = 2 * \frac{precision * recall}{precision + recall}$$

$$F_\beta = \frac{(1+\beta^2)*true\ positive}{(1+\beta^2)*true\ positive + (\beta^2)*false\ negative + false\ positive} \qquad (\beta = \text{Type II error } **)$$

i.e. Type I error refers to incorrectly rejecting a true null hypothesis and type II error refers to failing to reject a false hypothesis.

	Model	Accuracy Score (Single Run)	Accuracy Score (N-fold Average)	Error Rate	Run Time (s) (Single Run)	Run Time (s) (With N-fold)	F1-Score (Unweighted)	# of High Potential Player	Ratio of Players Identified
1	Adaboost	0.63	0.64	0.35	142.51	722.53	0.60	103	0.0041
2	Random Forest	0.63	0.64	0.36	4.38	31.70	0.61	110	0.0063
3	Extra Tree	0.60	0.61	0.39	2.94	17.13	0.57	188	0.0126
4	SVM	0.55	0.56	0.38	290.34	1814.08	0.56	248	0.0167
5	Logistic Regression	0.60	0.61	0.40	1.98	10.69	0.54	129	0.0095

Table 5: Models Results

Among the 5 models, 2 of them (Adaboost and Random Forest) provided similar accuracy above 0.6 in single run, significantly better than random assignment at 0.33 considering having 3 classes. Other than accuracy our next criteria to compare is the run time because the more efficient the model the less cost in computing. In this case Random Forest runs a lot faster than Adaboost and is thus selected as our best model. On the other hand, SVM performs the worst with the lowest accuracy while being the slowest at the same time.

Players who are currently ranked by the game as level 1-6 (our group 0) but are classified as group 2 (level 14-18) by our model are identified as the high potential talents. These are most likely the next top-notch players; thus, our definition would be more rigorous to make sure they do demonstrate significant outstanding qualities. For this reason, we want our results to be as precise as possible, so we would actually prefer a model with the lower identifying ratio. Random Forest with the second lowest of identifying ratio again meets our criteria. Lastly, in the prediction part of our model, it would provide a list of user id of the identified upcoming players. The full list of the 110 user-ids can be found in Appendix.

Research Retrospect

- **Limitations and challenges:**

2 general challenges appear though out the project. First, there are 2 sets of ids being used by the game and DreamTeam platform, special cautions were required in merging tables and matching the data for each player. Secondly, the data collection methodology of different tables was not clear which has led to the limitations of our research. Several features could have potentially been included in algorithms to improve the accuracy, however, were eventually excluded due to too many missing data. For example, in the maps.stats data, the total wins and rounds played do not match with the data in csgo.profiles table, meaning not all the games data were being captured. This could be due to customized map being excluded or only standard top played maps played in competitive modes were included. With the data collection logics remain unknown, it is hard to collaborated with the main profile data as that would create biases in the analysis. Otherwise, our models could have potentially been improved by added weights on those features, rather than excluding them completely. Also, due to the same reason, we were not able to remove some of the extreme value even though we suspect them to be outliers or flaw data, otherwise we could have prepared a better training data for the models and achieve higher accuracy.

- **Further applications:**

It is expected that the new labels our application provides to the players can help them in better team matching and filling up vacancy. To evaluate the results, we can run an A/B test as the next phase of the implementation that we randomly split the top players that our model identify in to 2 groups (one test group and one control group) and only show the top potentials label (treatment) for the test group. After a fixed period of time, we measure how many team joining requests they have received as the overall

evaluation criterion of the experiment. We can then compute a T-test on the difference of the mean of joining requests to see if we can reject null hypothesis (H_0: our top potential player labels have no effects on helping the players to be recruited by the right team), proving our implementation a success.

Adding this feature of signaling if a player is selected as upcoming players on the platform would not only help team owners to identify high potential team candidate but also serve as a check to the players if their strategies for ranking have been working effectively. Possibly by increasing the accuracy of the models with better features input and more training data points, we will achieve higher confidence in telling more precisely the difference between player rankings and their true abilities.

Appendix:

Prediction Results (by Random Forest Model)
110 high potential players:

	user_id		user_id		user_id
1	61225	44	78778	87	34710
2	67133	45	46694	88	17342
3	87435	46	2449	89	89056
4	41050	47	54252	90	52986
5	93066	48	67793	91	100055
6	64823	49	74353	92	94361
7	73455	50	63531	93	62628
8	11674	51	24167	94	10340
9	19829	52	83498	95	30943
10	7127	53	93357	96	46567
11	85511	54	57063	97	37066
12	38856	55	99740	98	7996
13	92563	56	57121	99	42886
14	73283	57	89839	100	87421
15	30425	58	42118	101	94275
16	68034	59	70372	102	185
17	92894	60	34078	103	43457
18	69705	61	75393	104	100918
19	54586	62	25299	105	36346
20	66635	63	26901	106	81953
21	54331	64	8272	107	73167
22	43573	65	51784	108	45364
23	97897	66	71722	109	78066
24	46417	67	32753	110	29796
25	38842	68	79782		
26	85931	69	96236		
27	88108	70	20253		
28	90291	71	81251		
29	73078	72	32098		
30	30833	73	98598		
31	50952	74	55188		
32	92248	75	34218		
33	20248	76	37147		
34	50190	77	69289		
35	48350	78	9204		
36	74976	79	22892		
37	60177	80	43780		
38	91924	81	16447		
39	86551	82	46099		
40	71226	83	9607		
41	95844	84	24297		
42	73099	85	71078		
43	63635	86	7157		

References:

[1] Mark E. Glickman, Example of the Glicko-2 system, November 30, 2013
Available at: http://www.glicko.net/glicko/glicko2.pdf (Accessed 20 Mar. 2018)

[2] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011. http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (Accessed 26 Mar. 2018)

[3] Ramesh Sridharan, 2017 MIT 6.S085 Statistics for Research Projects Week 2, [Lecture Notes: Nonparametric statistics and model selection] http://www.mit.edu/~6.s085/notes/lecture5.pdf (Accessed 28 Mar. 2018)

[4] Jonathan Taylor, Stanford Univeristy, Statistics 202: Data Mining Classification & Decision Trees Based in part on slides from textbook, slides of Susan Holmes, October 19, 2012 http://statweb.stanford.edu/~jtaylo/courses/stats202/restricted/notes/trees.pdf (Accessed 28 Mar. 2018)

[5] Wikipedia contributors, "Receiver operating characteristic." Wikipedia, The Free Encyclopedia, 26 Feb. 2018. https://en.wikipedia.org/wiki/Receiver_operating_characteristic (Accessed 28 Mar. 2018)

[6] Tomaso Aste, UCL 2018 Data Analytics COMPG011 week 5, [Dependencies & Causalities] https://moodle.ucl.ac.uk/pluginfile.php/3264831/mod_resource/content/2/COMPG011%20Part_2_Dependency_2018_v2.1.pdf (Accessed 27 Mar. 2018)

[7] Tomaso Aste, UCL 2018 Data Analytics COMPG011 week7, [Networks] https://moodle.ucl.ac.uk/pluginfile.php/3307093/mod_resource/content/2/COMPG011%20Part_3_Networks_2018_v2.1.pdf (Accessed 27 Mar. 2018)

[8] Thomas G. Tape, MD, University of Nebraska Medical Center 2018, Interpreting Diagnostic Tests, [The Area Under an ROC Curve], http://gim.unmc.edu/dxtests/roc3.htm (Accessed 27 Mar. 2018)

[9] Tomaso Aste, UCL 2018 Data Analytics COMPG011 week2, [Probability & Statistics] https://moodle.ucl.ac.uk/pluginfile.php/3230565/mod_resource/content/9/COMPG011_Part_1b_Probability_Further_2018_v1.4.pdf (Accessed 27 Mar. 2018)