Random Forest Based Approach of Finding Factors that Determine the Result of a Competitive Match in Overwatch

1st Zhehao Xu
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
z352xu@uwaterloo.ca

***Abstract*—Overwatch is a famous video game developed and published in 2016 by Blizzard Entertainment company. In this paper, a data analysis was applied to overwatch to seek answers for overwatch players about problems that they encountered while playing competitive matches.In specific, the problem that many players found themselves a hard time to climb up in the overwatch ranking system and they did not know what caused it. This paper targeted at finding important factors that players should pay attention to in order to win more competitive matches to successfully climb up in the ranking system. Random forest classifier was proposed to achieve this goal as it had robust performance on classification problems. Then, few most important factors could be deduced by ranking those features based on their importance. After the training, the random forest classifier achieved an accuracy of 83.87% on the test set. Features like 'Objective Kills','Elimination','Death' were shown on the top of the ranked list.**

***Index Terms*—Random Forest, Overwatch Video Game, Classification, Feature Importance**

I. Introduction

Overwatch is a multi-player team-based first person shooter(FPS) game that was revealed in 2016 by Blizzard. It then soon became one of the greatest game of all time and earned a large number of game design rewards. Just in one year, overwatch had $1 billion dollar in revenue, and 30 million registered players according to publisher's fiscal Q1 2017 financial statement. [1] The rising of overwatch has many factors, some of the most important elements that lead to its success are attractive character designs where each character has their own storylines and unique abilities in the game, the second one is the plentiful playstyles and various team compositions player can choose from, and the third one is it is also an e-sport game and overwatch league is launched to have top tier players against each other.

However, the good time does not last for long. today after three years, overwatch revenue continuously decreasing by 12 percentage every year. Players found there were only few types of game modes and Blizzard did not introduce new content frequently, they quickly got bored at all the available heroes and one new hero were introduced every two to three months which was at a low frequency. The most important factor that let players quit was the imbalance ranking system. Similar to Dota,League of Legends,CSGO, Overwatch also has its ranking system that has value called 'Season Rank(SR)' to indicate how good the player is by completing competitive matches. In the first season which last for four months, SR had a range of 0 to 100. The higher SR the player got, the better he was at the game. Every competitive match players participate would effect the current SR score. After completing the first ten matches called placement matches, the player can receive a SR score. Starting from the second season, the SR range was adjusted and it was from 0 to 5000 where it also split into several small tiers according to the following table:

TABLE I: **Overwatch seven rank tiers**

Rank Tier	Season Rank Score
Bronze	0-1500
Silver	1500-1999
Gold	2000-2499
Platinum	2500-2999
Diamond	3000-3499
Master	3500-3999
Grandmaster	4000-5000

In an addition to the above table, the best 500 players in the ranking system are rewarded with top 500 logos every season. After player is placed in certain tier with the starting SR score, every competitive match he played will effect the player's SR. he will gain SR by wining matches or lose SR from losing matches. The above is a brief description of overwatch ranking system. The problem appears to be lots of players reported that they were stuck at certain rank tier and could not climb up by kept having a close number of matches that were won comparing to lost. Therefore, the SR score would remain at that level. This situation appears in the tier of gold and platinum mostly which are the two that around 50% overwatch players position at.

This concern by players is possible to be answered going from data aspects using machine learning techniques. In this paper, a players' competitive matches history records along with detailed performance statistics were used in order to feed into the model for training purpose to let the model learn. In this way, features could be ranked according to their importance towards the final predictions of the match results. The reason for doing that is that there is almost none of the research or data science related experiments applied to Overwatch video game. It is a new field where data-mining can contribute in order to answer questions and help players to rank up by showing them what are factors that they need to pay attention with in the future.

II. Literature Review

Applying machine learning techniques to analyze data in fields of sports competitions including e-sports was not a innovative idea while there were many papers already published. During the research, some of articles that explained the random forest algorithm and others that had similar ideas to perform analysis on video games or real athletes' competitions were selected and briefly discussed.

A. Classification and Regression by randomForest [4]

Random Forest was an 'ensemble learning' method which it constructed many classifiers and produced the final prediction based on votes coming from all classifiers. For example, if there were 10 classifiers, 6 of them predicted that the new sample fell into class 1 and the rest 4 said it would be in class 2, then the final prediction for the new sample would be labelled with class 1. Ensemble learning improved accuracy as multiple classifiers participating in the final decision instead of just one. Second it was more consistent as it could effectively avoid overfitting problem. Lastly it had smaller bias and variance errors due to multiple classifiers working together. The biggest difference between random forest and classic decision trees was that it randomly chose a number of features as a subset and found the best split in the subset for the node instead of finding the best split while considering all the features together. [4]

B. A Neural Network Method for Prediction of 2006 World Cup Football Game [5]

In this paper, the same approach is brought in predicting the win rate of football teams in the 2006 world cup football game, the multi-layer perceptron with back propagation algorithm. Author Kou-Yuan Huang proposed a multi-layer perceprton neural network with an architecture of 8 inputs, 11 hidden nodes, and 1 output to predict the win rates of all the participated football teams in the 2006 world cup at different stages. the test found that the highest accuracy appeared in the prediction of stage 2 and final. [5] Similarly, the overwatch competitive match records could also be fed into multiple decision tree classifiers in order to predict the result of a match by aggregating their predictions and then found out high influential features.

III. Methodology

In order to achieve the goal of finding important factors that determine the result of a competitive match in overwatch, an organized dataset is required and by doing data-mining analysis, features that with high contribution towards deciding whether a match should win or lose are those considered as important factors. the following figure is the basic workflow that illustrates stages that are required to complete the task.

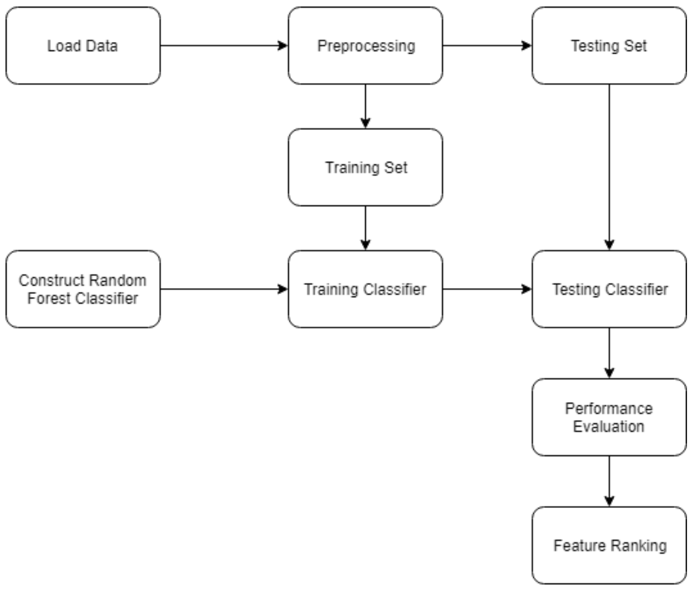


Fig. 1: Method workflow

The starting point is at the top left corner where first two stages are launched. "Load Data" is where the dataset is loaded. The selected dataset is a list of competitive match records with detailed performance statistics for one player through different seasons from Kaggle website. The player is the dataset creator himself who placed between gold and platinum. This player is a good representation of average player level as around 50% players are also in gold and platinum and they are those who complain about the stucking problem mostly. On the other hand, a Random Forest(RF) classifier is built in order to learn the pattern inside the dataset and successfully predict match results with an adequate accuracy. Following the branch of loading data, there is a preprocessing stage. It is necessary to do some cleaning for the dataset before feeding into the model because some of the fields are missing and also categorical features are needed to be converted to numerical in order to use them in the training stage otherwise it will result in error at the training stage.

The dataset will be split into two sets after the preprocessing stage is completed, one is for training which called "Training Set" and another is for testing purpose which called "Testing Set". the training set is firstly fed into the built RF classifier and it is going to construct the number of decision trees based on the setting parameters to see which one best estimates the results of matches. Then, the trained classifier will be tested using the test set as it shows in the figure for the "Testing Classifier" stage, those are datapoints that have not participate in the training stage which can show how good the performance of the trained classifier is against unseen samples. At the final stage, if the performance evaluation indicates an acceptable result, features will be ranked by its importance according to how much they influence the match results. Therefore, factors that contributes to the final result of a match can be found at this point.

IV. Implementation

A. Software Environment

The implementation was purely on the software side whereas there was no hardware involved. A detailed environment specification table was provided as below:

TABLE II: Software Environment

Operating System	Windows 10 Home
Programming Language	Python 3.6
Integrated Development Environment(IDE)	Pycharm
Library used	Pandas Scikit-learn Category Encoder Scikit-plot Matplotlib

The workstation had an installed Windows 10 Home operating system along with a version 3.6 python environment installed too. There were several libraries used in the development which were pandas,scikit-learn,matplotlib,and category encoder. The pandas library was mainly used for manipulating the dataset including cleaning, filtering , and filling etc. to make the dataset suitable for the built model during the training and testing phases. Scikit-learn was used to build the random forest classifier as it was a built-in class and it was convenient for users to implement it and tune parameters of the classifier. Then, matplotlib was used to plot graphs for the performance evaluation stage which can give a better understanding and visualization as comparing to comparisons of values. The last one was the category encoder which was used to convert categorical features to numerical features so that those features could be fed into the classifier. Here ”One Hot Encoder” was used to make the conversions.

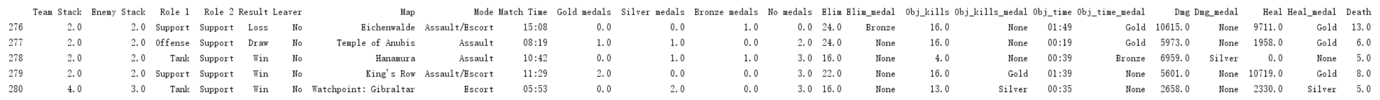
B. Dataset Walkthrough

The dataset that was used in this project was from kaggle database. the name of it was ”Overwatch Ranked Data”. The original dataset contained 582 match records through different seasons with 38 features which were match healing done, total elimination, objective time etc. However, a lot of fields were missing in the dataset for seasons from three and four. Therefore, those samples were removed as they did not provide any useful information to the model. Furthermore, features like which season the match was played, the SR scores before and after matches did not really relate to the performance of the player during matches and those would also be removed. Some of datapoints with few missing fields were filled with the most frequently appeared values or classes for each of the feature.

Figure 2 was the first 5 rows of the dataset after removing those unnecessary features and blank rows. there were 306 match records with 23 features remained in the dataset. Every feature had its own representation. Starting from the left most, each feature was given a short description as below:

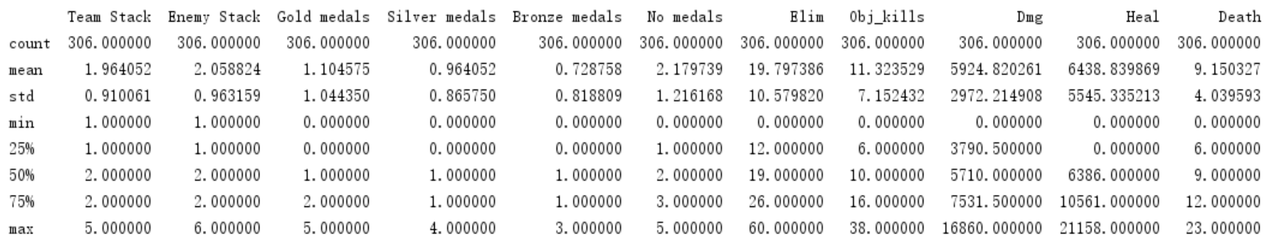
- Team and Enemy Stacks: the number of players that were played in groups before entering the match. Players could group up and participate all in one match as long as the game was 6 players against the other 6 players.
- Role 1 & 2: it indicated what positions the player selected during the match. There were totally 4 classes which were ”Tank”,”Offense”,”Defense”,”Support”. Each class contained different heroes that player could choose from. Tank was mainly shielding to protect teammates from taking damage, offense and defense heroes were good at outputting damage to enemies, and last support heroes were good at healing teammates to prevent them from death.
- Leaver: If there was any player left the match in the middle whether on the enemy’s team or the player’s team.
- Map & Mode: those two features were the names of the map and mode the match had
- Match Time: how much time did that match last
- Gold & Silver & Bronze Medals: players who had the best performance statistics would be rewarded gold medals on his team, silver medals for the second best and bronze medal for the third best. There were different fields of performance and they would be rewarded medals individually instead just one gold medal for the player which meant a player could have multiple gold or silver or bronze medals. Furthermore, the dataset also considered how many fields the player did not earn any medal.
- Elim & Elim_medal: the number of times that the player eliminate an enemy hero. This number could go large as enemy players could respawn after death continuously until the match finished and elim_medal was the type of medals he earned.
- Obj_kill & Obj_kills_medals: same term as elimination but only counted eliminations that were on the objective points.
- Obj_time & Obj_time_medal: the time that the player stayed on the objective point and the type of medals he earned comparing among other teammates
- Dmg & Dmg_medal: the total damage done to the enemy team for the match and the type of medals the player earned
- Heal & Heal_medal: the total healing done to the teammates for the match and the type of medals the player earned
- Death: the total number of death the player had for the whole match

The above description briefly explained what features the dataset contained and how those related to the match as factors that could influence the final result of the match whether win or lose. This was the final dataset that was processed in the training stage.



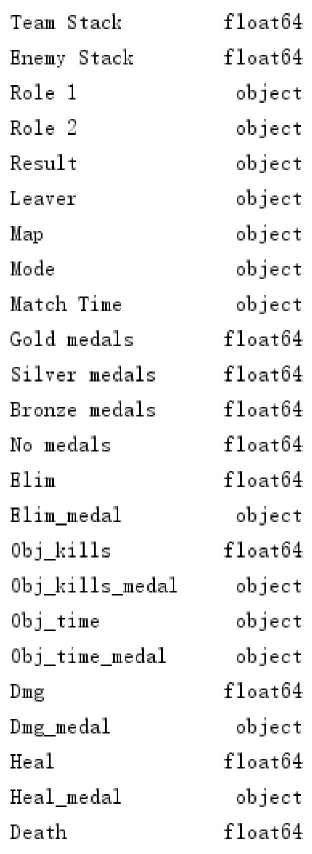
```
     Team Stack  Enemy Stack  Role 1   Role 2 Result Leaver                   Map           Mode Match Time  Gold medals  Silver medals  Bronze medals  No medals  Elim Elim_medal  Obj_kills Obj_kills_medal Obj_time Obj_time_medal      Dmg Dmg_medal     Heal Heal_medal  Death
276         2.0          2.0  Support  Support   Loss     No            Eichenwalde  Assault/Escort     15:08          0.0            0.0            1.0        0.0  24.0     Bronze       16.0            None    01:49           Gold  10615.0      None   9711.0       Gold   13.0
277         2.0          2.0  Offense  Support   Draw     No      Temple of Anubis         Assault     08:19          1.0            1.0            0.0        2.0  24.0       None       16.0            None    00:19           Gold   5973.0      None   1958.0       Gold    6.0
278         2.0          2.0     Tank  Support    Win     No              Hanamura         Assault     10:42          0.0            1.0            1.0        3.0  16.0       None        4.0            None    00:39         Bronze   6959.0    Silver      0.0       None    5.0
279         2.0          2.0  Support  Support    Win     No            King's Row  Assault/Escort     11:29          2.0            0.0            0.0        3.0  22.0       None       16.0            Gold    01:39           None   5601.0      None  10719.0       Gold    8.0
280         4.0          3.0     Tank  Support    Win     No  Watchpoint: Gibraltar          Escort     05:53          0.0            2.0            0.0        3.0  16.0       None       13.0          Silver    00:35           None   2658.0      None   2330.0     Silver    5.0
```

Fig. 2: Samples of the dataset after preprocessing



```
       Team Stack  Enemy Stack  Gold medals  Silver medals  Bronze medals   No medals        Elim   Obj_kills           Dmg          Heal       Death
count  306.000000   306.000000   306.000000     306.000000     306.000000  306.000000  306.000000  306.000000    306.000000    306.000000  306.000000
mean     1.964052     2.058824     1.104575       0.964052       0.728758    2.179739   19.797386   11.323529   5924.820261   6438.839869    9.150327
std      0.910061     0.963159     1.044350       0.865750       0.818809    1.216168   10.579820    7.152432   2972.214908   5545.335213    4.039593
min      1.000000     1.000000     0.000000       0.000000       0.000000    0.000000    0.000000    0.000000      0.000000      0.000000    0.000000
25%      1.000000     1.000000     0.000000       0.000000       0.000000    1.000000   12.000000    6.000000   3790.500000      0.000000    6.000000
50%      2.000000     2.000000     1.000000       1.000000       1.000000    2.000000   19.000000   10.000000   5710.000000   6386.000000    9.000000
75%      2.000000     2.000000     2.000000       1.000000       1.000000    3.000000   26.000000   16.000000   7531.500000  10561.000000   12.000000
max      5.000000     6.000000     5.000000       4.000000       3.000000    5.000000   60.000000   38.000000  16860.000000  21158.000000   23.000000
```

Fig. 3: Dataset description



```
Team Stack         float64
Enemy Stack        float64
Role 1              object
Role 2              object
Result              object
Leaver              object
Map                 object
Mode                object
Match Time          object
Gold medals        float64
Silver medals      float64
Bronze medals      float64
No medals          float64
Elim               float64
Elim_medal          object
Obj_kills          float64
Obj_kills_medal     object
Obj_time            object
Obj_time_medal      object
Dmg                float64
Dmg_medal           object
Heal               float64
Heal_medal          object
Death              float64
```

Fig. 4: Data types of features

In an addition, the description of the dataset along with data types of features were given as figures 3, and 4. From figure 3, there were 306 datapoints in total and also the skill level of the player could be told. For example, the mean value of gold medals was 1.1 which meant that he performed the best in one field in his team during matches. he only had an average value of 0.49 on the streak feature so he was not the player who won a lot. His SR score remained in a certain range and would not go up to diamond tier which was exactly the stucking problem.

C. Random Forest Classifier

When it came to the construction of the classifier, scikit-learn made it easy as it had built-in classes. The dataset was split into training and testing sets with a percentage of 20%. That meant there were 20% randomly picked samples that would not participate in the training phase and grouped as the testing set saved for later. The built-in 'RandomForestClassifier' in scikit-learn library was used, a list of parameters for the classifier was provided:

TABLE III: Random Forest Classifier Parameters

parameters	**specification**
n_estimators	to be determined in the training phase
criterion	gini
bootstrap	True
max_features	auto
min_samples_leaf	1
min_samples_split	2
max_depth	None

The table 3 provided some of the main tuning parameters for the classifier and an explanation for each of those items was given below:

- n_estimators: this estimator parameter sets the number of decision trees to have in the forest. if the value is 10, then 10 decision trees will be constructed in the random forest.
- criterion: the function to measure the quality of a split.
- bootstrap: if true, bootstrap method is used when developing each tree. It randomly selected features to form subsets as training sets for each decision trees to avoid overfitting and improve accuracy.
- max_features: the maximum number of features considered when making a split. If it is a value of 10, then it randomly select 10 features and look for the best one to use. This parameter can prevent the classifier from overfitting effectively.
- min_samples_leaf: the minimum number of samples required to be a leaf node.
- min_samples_split: the minimum number of samples required to split a leaf node.
- max_depth: it states the maximum depth for each tree. If the value sets to None, it means that trees will be expanded until all the data come from the same class for every leaf which is also called 'pure leaf'.

1) n_estimators: The optimal number of decision trees to grow for the random forest classifier was determined by trying a wide range of values. In this project, multiple classifiers were trained using various number of decisions trees starting from 10 to 50 trees. Then, the optimal number was the one which had the best performance on the test set.

2) gini criterion: In the setting, gini criterion was used for the classifier as the gini impurity was calculated at each level in those decision tree classifiers to show the probability of a randomly chosen sample in a node been labelled incorrectly if it was labelled according to the distribution of the samples in the same node. It followed the following formula

$$I_G(p) = 1 - \sum_{i=1}^{J} (p_i)^2 \tag{1}$$

Where it was one minus the total summation of the fractions of samples in each class. The best split followed the rule by maximizing the 'gini gain' which was that if a node got split, the new weighted gini impurity had the greatest difference comparing to the original gini impurity.

D. Testing Metrics

For the performance evaluation and testing, few metrics were examined to quantify the goodness of the trained classifier. Firstly, cross validation test was executed to provide the training score and cross validation score. those scores were expected to increase as more samples were fed into the classifier during the training process.If it had a high training score but low cross-validation score, it meant that overfitting problem occurred for the classifier as it predicted trained data perfectly, but poor on unseen data. Then, a performance test was ran by feeding the test set data into the trained classifier to see if it still performed acceptable to handle unseen samples. Lastly, the classification report was generated as a conclusion of the performance on the test set.

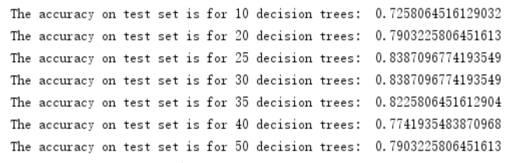
E. Feature Ranking

Once the random forest classifier was successfully trained with an acceptable performance on the test set. The final part was to rank features from the most to least important towards the prediction whether the match was going to win or lose. It used the gini values calculated when constructing those decision trees and this method was called 'Mean Decrease Impurity'(MDI). MDI summed the total decrease in gini impurity for each feature among all decision trees constructed in the random forest. The higher the decrease in gini impurity, the more important the feature was. Another method was 'Mean Decrease Accuracy' (MDA) which was not used in this project because the gini impurity was already calculated when constructing those decision trees as the extra computation required was minimized.

V. Results

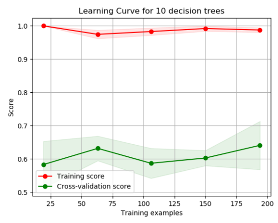
A. Training Result

The training was successfully completed using different numbers of decision trees constructed in the random forest classifier to see the optimal value of trees that the classifier should grow in order to achieve the best accuracy. The

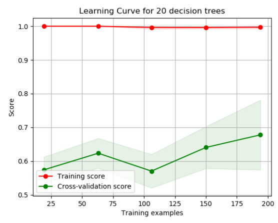


```
The accuracy on test set is for 10 decision trees:  0.7258064516129032
The accuracy on test set is for 20 decision trees:  0.7903225806451613
The accuracy on test set is for 25 decision trees:  0.8387096774193549
The accuracy on test set is for 30 decision trees:  0.8387096774193549
The accuracy on test set is for 35 decision trees:  0.8225806451612904
The accuracy on test set is for 40 decision trees:  0.7741935483870968
The accuracy on test set is for 50 decision trees:  0.7903225806451613
```

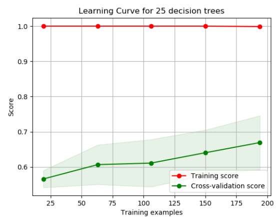
Fig. 5: Accuracy comparison between random forest classifiers with different numbers of decision trees grown



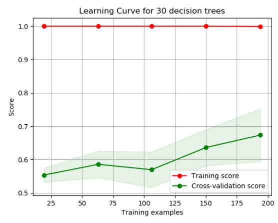
(a) Learning curve using 10 decision trees



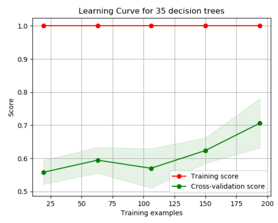
(b) Learning curve using 20 decision trees



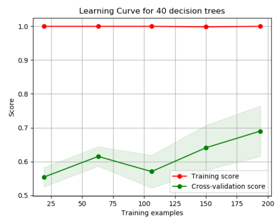
(c) Learning curve using 25 decision trees



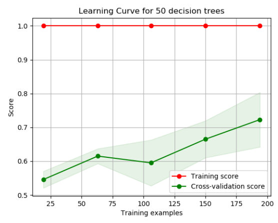
(d) Learning curve using 30 decision trees



(e) Learning curve using 35 decision trees



(f) Learning curve using 40 decision trees



(g) Learning curve using 50 decision trees

Fig. 6: Learning curves comparison using different numbers of decision trees

The red line represented training score and green line represented the cross-validation score the trained classifier earned. The training score was very high in all classifiers as it remained at 1.0 while only a slight decrease when the random forest classifier only had 10 decision trees. On the other hand, a trending pattern found for the cross validation score in all situations as more training examples involved. However, a common decrease in scores happened when 100 samples were trained. This pattern was very obvious for classifiers that had decision trees that were below 20 and above 30. Combining with the accuracy readings on the test set, the optimal number of decision trees that the random classifier should have fell into the range between 25 to 30 as their accuracy readings were the same and the best comparing to other classifiers. Therefore, a random forest classifier that had 25 decision trees were used for the final outcome as 25 trees would had less computation cost as 30 trees while maintaining the same accuracy.

Furthermore, the confusion matrix and prediction accuracy on the test set for the trained classifier was also provided in figures 6 and 7. The accuracy on the test set achieved a value of 83.87% which was acceptable. As the below confusion matrix showed, there were 18 matches that were lost correctly predicted with 7 wrong predictions. On the other hand, 34 matches that the player won were all predicted correctly and 3 were predicted wrong. The reason for using accuracy as one of those metrics to evaluate the classifier was that the number of matches that were lost and win were balanced which nearly 60% test samples belonged to win class while the rest 40% belonged to loss class. [9]

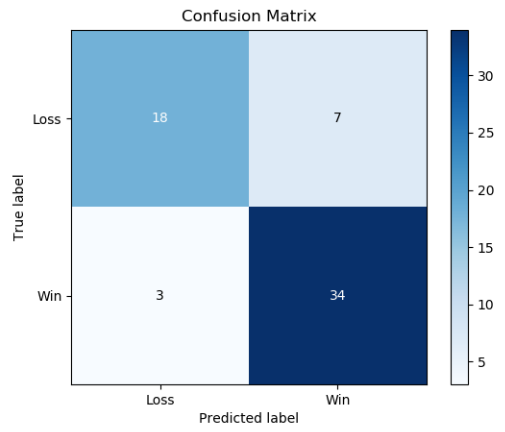
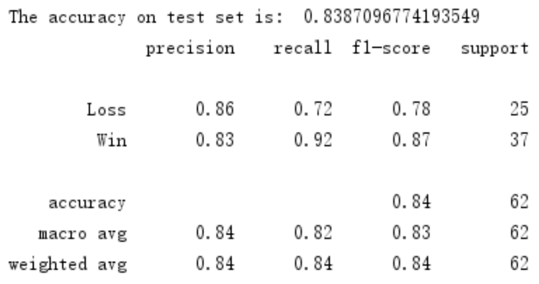


Fig. 7: Confusion matrix on the test set

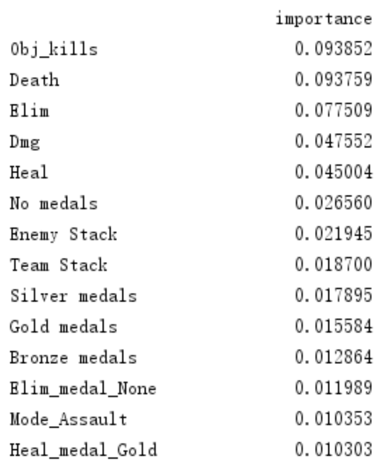


```
The accuracy on test set is:  0.8387096774193549
              precision    recall  f1-score   support

        Loss       0.86      0.72      0.78        25
         Win       0.83      0.92      0.87        37

    accuracy                           0.84        62
   macro avg       0.84      0.82      0.83        62
weighted avg       0.84      0.84      0.84        62
```

Fig. 8: Training score and report for random forest classifier

B. *Feature Ranking List*

Once the trained classifier had an acceptable performance. Features based on importance towards the final match result prediction were quantified and ranked from the highest to lowest. A high value indicated that the predicted match result would have a high possibility to be influenced by that feature. the following list was the first 14 features with highest scores of importance. Features after those 14 had values that were below 0.01 which were not as significant as those selected features.



```
                         importance
Obj_kills                  0.093852
Death                      0.093759
Elim                       0.077509
Dmg                        0.047552
Heal                       0.045004
No medals                  0.026560
Enemy Stack                0.021945
Team Stack                 0.018700
Silver medals              0.017895
Gold medals                0.015584
Bronze medals              0.012864
Elim_medal_None            0.011989
Mode_Assault               0.010353
Heal_medal_Gold            0.010303
```

Fig. 9: Feature importance ranking

VI. Discussion & Future Improvements

A. *Result Discussion*

Observing the ranking list in figure 7, it could be told that the most important feature to consider was the 'Obj_kills' feature which was the number of kills happened on the objective point. The winning conditions for the enemy team were capturing objective points while enemy players presented on the objective point for enough time, killing enemy players that were trying to capture the point could slow and prevent them from winning the match.

Surprisingly, the second most important feature was the 'Death' which was the number of death the player had during the match.If the player died too much time, there were 6 seconds penalty for death as he could not respawn to join back the game. Therefore, enemy team would have number advantage to have a high chance of defeat the player's team. In this way, the player should consider to make the number of death as low as possible. 'Elimination' was behind the death feature and there was a value gap 0.02 between while the objective kills and death both had a similar value of 0.093. This showed that player should be more careful to their death than the number of enemy players they eliminated.

Features 'Dmg' and 'Heal' could be examined together as the first one was the total damage done to the enemy team and second one was the healing done to teammate. Characters had their own heal bar and they would be consider as dead if the heal bar went empty. On the other hand, abilities that could raise teammates' health contributed to the total healing value. More damage dealt to enemies from the player, higher possibility they would die and spent 6 seconds waiting to rejoin the match to waste the total available time they had for the match. The more the player healed teammates, the less chance those teammates would die so that they could continue to fight in the front line to prevent enemy team from winning as enemy team could not make progress on the objective point while any of player's team member presented on the point too.

Finally, features like 'Gold medals','Silver medals' and 'No medals' were overlapped in terms of meanings. It basically showed that having more gold medals, the player would had a higher chance to win because he was performing his best as it was related to those previous features like elimination, damage dealt, healing done etc. Another interesting feature was the 'Enemy Stack' which was number of enemy players grouped before entering the match. This meant enemy players knew each other before the match and they may played together previously which indicated that they would have a better team work,communication and collaboration skills. This interpretation could be applied to 'Team Stack' feature too as the player's team could have similar strengths as mentioned above.

B. *Future Improvements*

There were many changes that could be made in order to further extend the project to be more accurate and comprehensive. The first thing that could improve was the dataset used in the project. As mentioned above, there were many fields missing in season 3 and 4 match records which were forced to remove from the dataset. As a consequence, the total samples used for training and testing were shrank from 582 to 306. If all samples could participate in the training and test phases, the trained classifier would be more accurate as those new samples may contained different situations such as winning the match but with poor performance than the player's average or lose but the performance from the player was better than usual. Although the player would only match with people that had similar SR scores, the skill level of each player may vary as some players may good at aiming while others may good at making strategies.

This brought the second improvements to the current dataset which was adding more features. There were currently 23 features excluding the match result that were considered in predicting the final outcome. More new features could be introduced into the dataset such as physical and mental conditions of the player while playing those matches. Was the player feeling angry or happy at that time? Was he experiencing any illness but still playing those matches? These were players related features outside the game because they were the people who played the game. They may have a better aiming level while they were excited or feeling happy. On the other hand, they could make mistakes that they rarely made before while they felt angry or depressed. These were things that was not considered in the dataset yet as the current one only had in-game performance statistics.This improvement could further extend the coverage of possible elements that may had impacts on the match result.

VII. Conclusion

In an conclusion, this project aimed to answer a popular confusion that overwatch players had, going from the data perspective using random forest machine learning techniques. Overwatch players often found themselves could not increase their SR score as they could not keep wining competitive matches. A dataset of one specific player who represented the average players' skill level was used in the project to find out what factors that actually players should pay attention to in order to win more matches in the future. The final dataset contained 306 match records with 24 features and it was then split into training set and testing set. After the training, The random forest classifier achieved an accuracy of 83.87% in predicting the match results based on in-game performance statistics.

Those features were then ranked in a descending order according to its importance in the decision of the final prediction of the match. The most important one was the objective kills feature which meant players should prioritize their target to enemy players that were on the objective point as eliminating them did not only just remove them temporarily from the fight but also preventing enemy team from capturing the point to win the game. The winning condition was to make enough progress on capturing the point not eliminating enough enemy players. The second suggestion would be making less death in matches because players could not do anything for 6 seconds while waiting to respawn. The third one was grouping up with friends or players that were familiar with and played competitive matches together because players that knew each other were easy to collaborate and communicate with according to the stack features appeared on the feature importance list. Finally during the game, players should tried their best to eliminate enemy players as this could remove them temporarily from the game and they could not help their teammates which could help the player's team to gain more advantages in the game.

References

[1] GRUBB, J. (2017, MAY 4). With $1 billion in revenue, Overwatch is Blizzards fastest-growing franchise. Retrieved from venturebeat: https://venturebeat.com/2017/05/04/with-1-billion-in-revenue-overwatch-is-blizzards-fastest-growing-franchise/

[2] Maher, C. (2019, March). Overwatch ranks explained: how to get ranked and what each rank . Retrieved from pcgamesn: https://www.pcgamesn.com/overwatch/ranks-explained-how-to-get-ranked

[3] Haykin, S. S. (2009). Neural networks and learning machines/Simon Haykin. New York: Prentice Hall,.

[4] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

[5] K. Huang and W. Chang, "A neural network method for prediction of 2006 World Cup Football Game," The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010, pp. 1-8. doi: 10.1109/IJCNN.2010.5596458 keywords: backpropagation;learning (artificial intelligence);multilayer perceptrons;neural nets;sport;statistical analysis;neural network method;2006 world cup football game prediction;official statistical data;multilayer perceptron;back propagation learning rule;training samples;Neurons;Training;Games;Australia, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5596458&isnumber=5595732

[6] M. Anshori, F. Mari, M. W. Alauddin and F. Abdurrahman Bachtiar, "Prediction Result of Dota 2 Games Using Improved SVM Classifier Based on Particle Swarm Optimization," 2018 International Conference on Sustainable Information Engineering and Technology (SIET), Malang, Indonesia, 2018, pp. 121-126. doi: 10.1109/SIET.2018.8693204 keywords: computer games;particle swarm optimisation;pattern classification;support vector machines;prediction result;svm classifier;particle swarm optimization;victory prediction;DotA 2 game;popular online games;winning opportunity;popular classification method;SVM method;good accuracy results;optimization methods;SVM parameter;SVM accuracy;C parameter;PSO;classification;DotA2;SVM;PSO, URL: http://ieeexplore.ieee.org.proxy.lib.uwaterloo.ca/stamp/stamp.jsp?tp=&arnumber=8693204&isnumber=8693112

[7] Reiichiro Nakano, reiinakano/scikit-plot: 0.3.7. Zenodo, 19-Feb-2017

[8] Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and douard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12 (November 2011), 2825-2830.

[9] Sunasra, M. (2017, Nov 11). Performance Metrics for Classification problems in Machine Learning. Retrieved from medium: https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b