Dota2 Match Prediction Final Report

Curtis Smith, Shuangyu Hu, Deion Yu, David Bayly

I. Project Description

Dota2 is a competitive multiplayer online battle arena (MOBA) with a massive fan base which matches two teams of 5 players against each other. The team that destroys the enemy's Ancient building wins the game. An annual Dota2 championship tournament called The International (TI) is hosted by the games developer, Valve which prize pool has grown considerably over the past years.

- TI1 prize pool: $1.6 million
- TI2 prize pool: $1.6 million
- TI3 prize pool: $4 million
- TI4 prize pool: $10.9 million
- TI5 prize pool: $18 million

For a given Dota2 match, we will be attempting to use data on the history of the players and the heroes they are using to accurately predict which team will win the match. We will be collecting information from the Dota2 Match History WebAPI and applying our algorithm to determine the probability that each team has to win.

There have been a number of projects that look at player Match Making Rating (MMR) and make simple calculations as to which team has the advantage, but we intend to introduce a number of other factors combined with knowledge from data mining to increase the accuracy of these predictions.

Beyond predicting the winning team in Dota2 matches, our predictor can also be applied to other competitive MOBAs like League Of Legends and Heroes of the Storm because they have very similar game design. The only steps in our process that have to be tweaked in order to predict the winning teams of other MOBAs are the MMR predictors, and the team composition evaluator as there a minor differences in what makes a good team between MOBAs. The general framework is common between MOBA games.

II. Motivation

Applications for this project can be found within the game itself as Dota2 has a mechanic that allows players to predict which team will be victorious and receive rewards if their prediction is correct. This project would provide an easy way for a user to analyze their current match and receive accurate probabilities to the outcome of the game.

Other possible applications for this project would be in online esports betting. There are many websites like egb.com that offer services where a user can wager real money and face off against other users to correctly predict the winning team.

III. Related Work

Several attempts have been made to predict Dota2 matches.[1] For instance Kevin Conley, a Masters student who took the Stanford's machine learning class was able to attain a 69% win rate prediction. He used a K-nearest neighbors to classify his data set and had a very large training set (18 000) matches. As opposed to our algorithm, which focuses on winning team compositions and player history, this algorithm compares the current match to similar matches in the past.

[2]Kyung yul, Kevin Lim and Nicholas Kinkade of the University of California, San Diego used logistic regression and random forest classifiers to achieve a 69% and a 74% successful win prediction rate.These were trained on a data set of 62 000 entries.

[3]The website HackerRank.com has an open challenge to predict Dota2 based solely on hero choices.

[4]There exists several services that use win-rate prediction to recommend hero selection. Dotapicker.com is an example. Their algorithm is not published.

IV. Implementation Description

The ultimate goal of this project was to correctly predict which team wins a Dota2 match given 2 teams of 5 players, and the heroes they play. In order to do so, several features are going to be considered during the classification process for the match winner.

1) Player MMR
2) Team Composition
3) Player's recent win rate with hero currently being played

A. Datasets

Valve offers a free Dota2 Match History WebAPI that can be used to pull match results and player histories from their internal databases. The initial plan was to use this WebAPI to get the MMR of each player along with their Hero ID, and use that information to determine which team would win a given match. Unfortunately, we quickly discovered that the majority of player accounts do not have a publicly available MMR value, and that they are only available for the highest skill set of players. Instead of choosing to build our classifier around only players with publicly facing MMR values, we instead decided to create our own MMR classifier, to estimate player MMR values based on their recent statistics.

This new approach would require the creation of two separate data sets:

1) Player Dataset - filled with players and the stats from their recent games, used to train the MMR classifier

2) Matches Dataset - filled with recent matches, the player's heroes, MMR estimates and the outcome

The player data set would need to be generated first, to ensure that each player in the matches data set would have a reasonably accurate MMR prediction to allow the match results to be predicted with. This data set was generated using a script written in Python 2.7 which was built with the Requests library to allow for easy HTTP requests to the Dota2 WebAPI. While the majority of players have their MMR value hidden, the Dota2 WebAPI has a feature that allows you to request random matches from one of three skill categories that correspond to certain MMR ranges.

1) Normal Skill - under 3100 MMR 72.5% of players
2) High Skill - 3100-3700 MMR 12.5% of players
3) Very High Skill - over 3700 MMR 15% of players

These categories gives us a target value with which we can train our MMR classifier. So instead of trying to predict the actual MMR value of each player, we would instead try to place them into one of these 3 skill categories. Players were pulled from these categories using code like the following:

```
params = {
    "key": API_KEY,
    "skill": NORMAL_SKILL,
    "game_mode": RANKED_5v5
}
response = requests.get(
MATCH_HISTORY_URL,
params=params
).json()
```

The skill parameter is where the MMR category is specified, the game_mode parameter is used to return only ranked 5v5 matches, and the API_KEY is required to actually access the API. The response is JSON formatted and uses the following structure:

```
"result": {
    "status": 1,
    "num_results": 100,
    "total_results": 500,
    "results_remaining": 400,
    "matches": [
        {
            "match_id": 101,
            ... # match details
            "players": [
                {
                    "account_id": 100,
                    "player_slot": 0,
                    "hero_id": 3
                }
                ... # each player
            ]
        }
    ]
}
```

Each match is then processed, and every unique account_id is added to a list that represents a pool of players for the given skill category. Once each skill category has a roughly even number of players, the recent match history of each individual player is processed and the averages are saved into the Players data set. This step uses code like the following:

```
params = {
        "key": API_KEY,
        "account_id": ACCOUNT_ID,
        "game_mode": RANKED_5v5
    }
response = requests.get(
    MATCH_HISTORY_URL,
    params=params
).json()

for match in response["result"]["matches"]:
    details_params = {
        "key": API_KEY,
        "match_id": match["match_id"]
    }
    details_response = requests.get(
        MATCH_DETAILS_URL,
        params=details_params
    ).json()

    # Process the match details
    # Save the player into Players
    # Dataset with their average
    # statistics over the last 50 games
```

The response that is processed looks like so:

```
{
    "result": {
        "players": [
            {
                "kills": 0,
                "deaths": 0,
                "assists": 0,
                ... # player statistics
            }
            ... # list all players
        ]
        "radiant_win": true,
        "duration": 0,
        "start_time": 1453585613,
        ... # match information
    }
}
```

Much of this information is not relevant for our MMR classifier, but the kills, deaths, assists, gold_per_minute, xp_per_minute, hero_damage, tower_damage, last_hits, and denies were recorded for current player, and averaged out over the number of games processed. These averages, along with the players win-rate on the current hero were saved into the Players data set to be used for training the MMR classifier. Some example entries from the Players Data set are shown below, each comma separated value corresponds to the follow-

ing data: KDA, GPM, XPM, Hero_Damage, Tower_Damage, Last_Hits, Denies, Matches_Processed, Skill_Category:

```
2.09,436,441,11222,767,133,3,50,0
2.56,344,280,7025,460,61,4,50,0
5.79,421,276,8142,364,31,1,50,0
```

Once the Player Data set was completed, and the MMR classifiers had been created (see following section), the Matches Data set was required to produce our final classifier and result set. This Matches Data set would consist of random matches pulled from any skill category, and the heroes, win-rates, and MMR predictions for each player in that match. The same process for pulling the matches used for both data sets, but the data processing step was different. The Matches Data set required that each player be fully processed by our MMR classifiers before saving their entry into the data set. The next section will explain that two separate MMR classifiers were used, so both results were saved with each entry of the Matches Dataset. The Matches Dataset was stored in the following format:

```
10x Hero Ids,
10x Recent Hero Winrate,
10x Player MMR Category 1,
10x Player MMR Category 2,
Winner
42, 53,...0.67,0.50,...1.0, 2.0,...,1
```

While this format allowed each entry to include only 41 values, it would have been beneficial so simply include each players statistics instead of pre-processing their predicted MMR and saving that value. If all the statistics were saved instead, we would have been able to tune our MMR classifiers with the same Matches Data set, instead of being forced to create a new Matches Data set every time we changed the MMR classifier tuning.

B. Player MMR

Of the 3 feature types listed above, a player's MMR plays the biggest role in deciding the outcome of a match. So it was a major problem when information about a players MMR was not reliably available from the API. The reason for this was that unless explicitly specified by the player, each player's MMR is hidden by default. A workaround to this problem was to create a separate predictor for players MMR using the 3 skill categories given in the Players Data set. So the following features are going to be considered during the classification process for players MMR.

1) Average Kill Death Assist(KDA) ratio = (K+A) / (D+1)
2) Average Gold per minute(GPM)
3) Average Experience per minute(XPM)
4) Average Hero damage
5) Average Tower damage
6) Average denies
7) Average last hits

To attempt to classify a player into one of the 3 skill categories, the first approach was to use the classifiers that come with the scikit-learn library. The 3 classifiers selected for this process were:

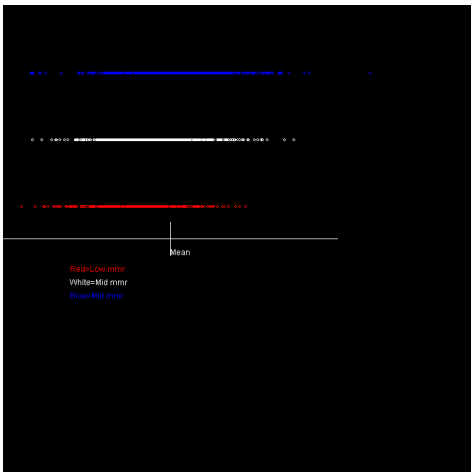
1) Random Forest
2) Support Vector
3) Multinomial Naive Bayes

Cross-validation was used, and the test-train split was 9-1. The results of these classifiers will be discussed in the results section of this report.

1) Custom MMR Classifier: MMR is one of the most important factors that have to considered in order to make an accurate prediction. MMR is an official score that accurately reflects the skill level of a player. It is the most comprehensive and straightforward score that players can use to infer how strong their teammates and enemies are. However, information about a player's MMR can be private. There is a large number of players who chooses to hide their MMR which introduces a lot of complications to the project. Therefore we had to develop our own classifier to predict the MMR players that have chosen to hide their MMR.

The original data set contained a classifier from 0-2 representing a players rough MMR. Initially our custom classifier took the form of simply splitting the players based on the mean for that MMR group. This process could be done recursively, creating as many MMR divisions as needed. We chose this method as a simple proof of concept. This was completed and produced satisfactory results, however, this was not a true classifier generated from each players attributes. To do this we used another data set without the rough MMR value.

Afterwards, we examined the data. Unfortunately there was a great amount of spread. Upon examining the data set, the attributes with the highest correlation with win-rate were GPM, XPM and last hits average. We put higher weight on these and used them to create an approximation of MMR. We called this value Score.



This shows the score attribute compared to the mean assuming all the attributes are given equal weight.
We used the original data set as a training data set.After tuning, we were able to predict rough MMR correctly only 50% of the time. We were hampered a lot by the high spread in values.

Our approach to MMR prediction was flawed in the end. There is a very weak correlation between in game stats and

MMR. MMR is based solely on wins. The in game stats reflect only how well a player is doing relative to their peers for the most part. An average player put in a game with novices will do exceptionally well, where a pro player put in with other pros will only do moderately well. This explains the large overlap. Since players are matched based on MMR, the MMR classifier we came up with is not very useful. For better MMR prediction, in future it would be best to have access to the players win rates with a given champion and to group players together based on who they play with.

C. Team Composition Evaluator

We came up with three rules that properly captures the qualities of a good team composition. In order to leverage these rules, we input the heroes IDs of the 10 players into our team composition evaluator. Based on these rules, the evaluator will make a prediction on which side has an advantage to win the match. Here are the three rules we used.

1) Team Balance : A carry is one of the most important roles in a Dota2 match. Heroes that fall under this role are the core of every team composition which everyone wants to play. So sometimes in a real game, a team can have a several carries but lack heroes from other roles like mid lane and support. Therefore, a team with too many carries has a lower chance to win the match.
2) Team Synergy : Heroes that fall under the category of disabler tend to work well with other heroes that can inflict a lot of damage on the enemy in a short time (nuker). If a team only have disablers and no nukers, the enemy will not be killed while if a team only have nukers and no disablers, the nukers will not have any opportunities to deal their damage to the enemy. Therefore, if one category like disabler or nuker is missing, the win possibility of that team decreases.
3) High winrate heroes : Some heroes have high winrates regardless of the skill of a player. This phenomena is due to the fact that game developers sometimes fail recognize these imbalances between heroes. Valves tries their best to balance the game but with 112 heroes, this goal is really hard to realize. So a team has a better chance to win a match by picking these unbalanced heroes.

D. Player Win-rate

The final metric that was to be considered was the recent win rate for each player on the hero that they would be playing. The idea being that a player could be on a hot or cold streak in their recent games, which would impact their performance in the game to be predicted. This data was gathered during the MMR prediction stage, as the game info was already loaded into the system, and minimal effort would be required to tally up the wins and losses on the chosen hero for that player. While the data gathering process seemed to be going smoothly, the results from using the player win rate numbers would end up negatively impacting out classifier.

Now that we had all the components necessary for predicting the winner of a Dota2 match, the match winner classifier could be created.The details of these results will be discussed in the following section.

V. Results

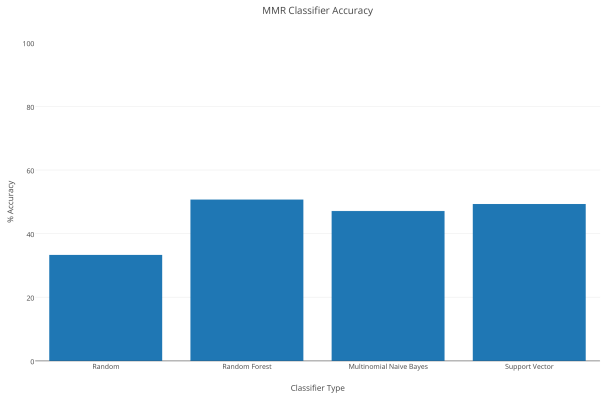
Each step of this project produced a set of results that is worth discussing, so the results section will be broken in to two parts:

1) MMR Classifier Results
2) Match Predictor Results

A. MMR Classifier Results

Two separate MMR classifiers were created from the Players Data set, the scikitlearn classifier and the custom MMR classifier. The results of each will be discussed in this section.

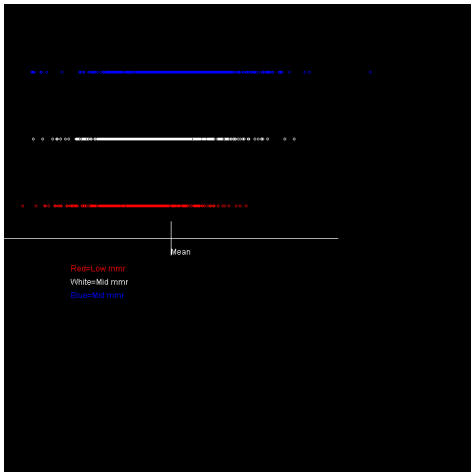
The scikitlearn classifier used the random forest, multinomial naive bayes, and support vector classifier included in the scikitlearn library. Cross validation was used with a 9 to 1 split, and each classifier was tested and received the following accuracy scores:



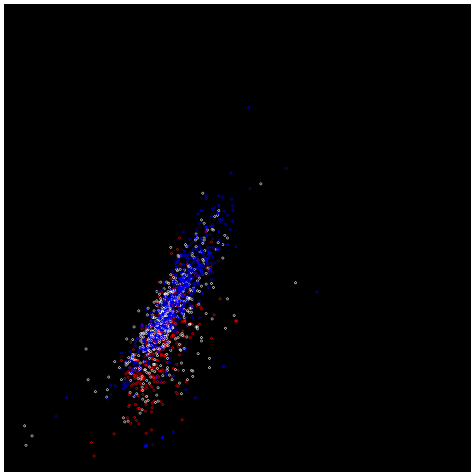
Each of the classifiers obtained an accuracy of 50%, with Random Forest being the most accurate.

An accuracy of 50% is a positive result, and the scikitlearn classifier should improve the results of the Match Predictor.

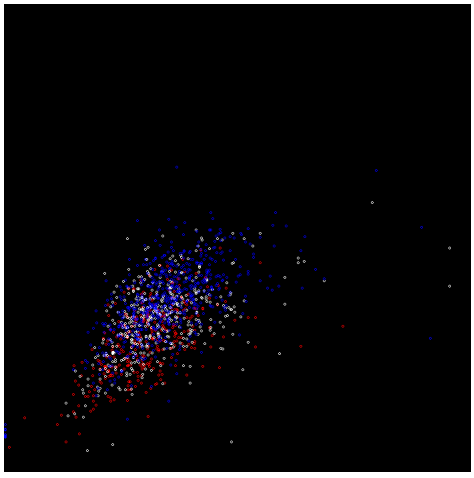
The Custom MMR classifier had some more in depth results:



This shows the score attribute compared to the mean assuming all the attributes are given equal weight.



This graph shows the performance of gold per minute alone as a classifier vs an aggregate classifier. Gpm is a good classifier.

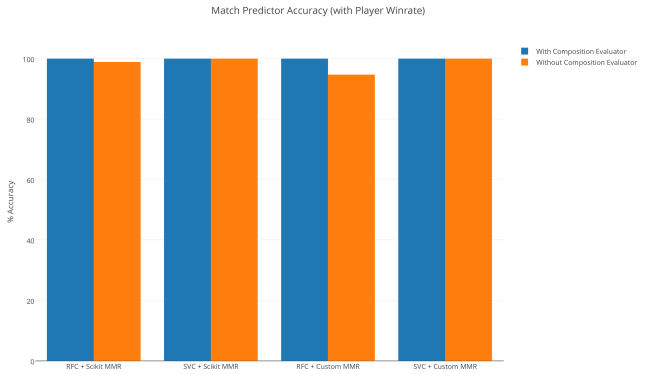


This graph shows the performance of KDA alone as a classifier vs an aggregate classifier. KDA is a Bad classifier.

B. Match Predictor Results

After creating and training the match predictor, we tested it with 10 fold cross validation and obtained the results below.

Results including Player win rate.

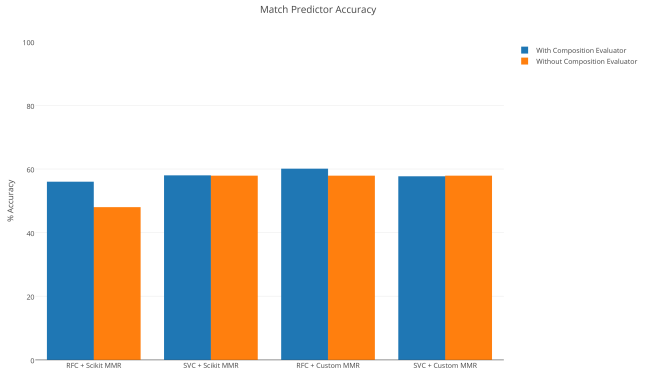


Method	Accuracy with team composition	Accuracy without team composition
RFC with custom MMR	100	94.7
SVC with Custom MMR	100	100
RFC with SciKit MMR	100	98.9
SVC with SciKit MMR	100	100

The results we obtained was frankly way too good which lead us to conclude that there was an aspect to our process that was faulty or overlooked. After scrutinizing our process, we came to find that the problem lied with the player win rate feature. Further analysis showed that during the data set creation phase, we included the test game when calculating the players win rate. So if the test match had a player who played a hero for a first time and won, the player would have a 100% win rate. This essentially meant that we had encoded the outcome of a game into the players win rate feature which the classifiers we used easily picked up on.

We worked around this problem by simply excluding players winrate from the list of features that our classifiers trained on. As such, we received the results shown below.

Results excluding Player win rate.



Method	Accuracy with team composition	Accuracy without team composition
RFC with Custom MMR	60.1	57.9
SVC with Custom MMR	57.7	57.9
RFC with SciKit MMR	56.0	48.0
SVC with SciKit MMR	58	57.9

The results we received matched our expectation more closely this time. Averaging the results of each Predictor combination, we obtained an accuracy of 57.95% which approaches the accuracy levels obtained by previous works but there is still room for improvement.

We were also interested in how effective the team composition evaluator was on its own, so we ran some tests using only that component. The results were as follows:

Method	Accuracy with Composition Evaluator
SVC	58.0
RFC	56.7

It is clear to see that our team composition evaluator performs favorably as it outperforms a random predictor.

VI. Conclusion and Future Work

In the end, our predictor is far from being 100 percent accurate which can be explained by the fact that our dataset was relatively small and our results were only based on MMR and team composition. The recent winrate couldnt be included because of mistake we made during the dataset creation phase. However, we only expect an increase in accuracy of about 3 to 5 percent with the recent player winrate. Acheiving a perfect prediction is really hard because a game of Dota2 is much more complex than what our current predictor takes into account. There are many other factors that could affect the result of a game like team strategies and a player's state of mind which are unpredictable and impossible to collect at the beginning of a game. However, our predictor's accuracy of 57.95% which is better than a random predictor by 7.95% does show that the features we decided to investigate does have an effect on the conclusion of a game.

We would like to improve the project by obtaining a larger dataset to train our classifiers on and fixing the problem with the player winrate feature. These changes are made difficult by the inefficient API format that requires 50+ requests to process a single player. The Matches Dataset was also especially difficult to form as it requires 10 players that have not set their account to private. So sometimes the script would be search for an eligible game for minutes at a time. Other than that, we would also like to include more features to capture the different complexities within Dota2 like team strategies or specific hero synergies.

VII. Glossary

Ancient: The core objective of Dota2 is the Ancient which is a large building with high health in the center of a teams base. The team that destroys the opposing team's Ancient is victorious and brings the match to an end.

Deny/Denies: Denying is the act of killing an allied unit or even oneself to prevent an opposing hero from getting the full reward from a kill.

Dota: Dota stands for Defense of the Ancients. It originated as a mod map for Warcraft 3. Dota was acquired by Valve and remade into Dota2, a separate game.

Hero: A hero is a playable character in Dota. Currently, players can choose from a roster of 111 heroes.

Item: Items are a game construct. Players can purchase them at a shop to further strengthen their heroes.

MMR: MMR stands for match making rating. It is a metric used to assess the skill of a play. 94 percent of players have an MMR between 1100 and 4000.

Disabler: Often a Support hero, a Disabler's job is to incapacitate enemy heroes.

Nuker: Nuker is a role taken by heroes who deal burst damage, usually magical damage, as opposed to sustained "right-click" damage.

Carry: They are the heroes that can obtain the greatest offensive power as the game progresses, the name so derived from the act being "carried" by a team into the late game; that is, to eventually bear the responsibility for ultimate victory.

Support: They are heroes whose purpose is to keep their allies alive and give them opportunities to earn more gold and experience.

Mid-lane: The second highest priority with respect to farm. The hero played mid is more often than not a semi-carry that can do well in a 1vs1 situation.

Jungler: It is when a player concentrates on killing Neutral Creeps during the game, usually for additional Gold or Experience. It refers to the Forests or Jungles between the lanes on either side where Neutral Creeps commonly spawn.

VIII. Bibliography

[1]K.Conley "Using machine learning to recommend heroes for Dota 2" http://kevintechnology.com/post/71621133663/using-machine-learning-to-recommend-heroes-for

[2]N. Kinkade, K, Lim "Dota 2 win prediction",http://cseweb.ucsd.edu/ jmcauley/cse255/reports/fa15/018.pdf

[3]HackerRank Match prediction challenge https://www.hackerrank.com/challenges/dota2prediction

[4]Dota Picker, http://dotapicker.com/

Dota2 online esports betting http://egb.com/esport/dota_2

IX. Task Breakdown

Curtis

1) Interfacing with match history API.
2) Generating and preparing usable data set.
3) Creating training and testing data sets.
4) Apply our algorithm on the dataset we already built.
5) Submit project.

Shuangyu Hu

1) Developing a new algorithm to apply on our data set to achieve our goal.
2) Create a Team Composition Evaluator.
3) Initial algorithm test.
4) Evaluate algorithms and findings.

David

1) Create a custom MMR Predictor.
2) Final tests.
3) Report editing

Deion

1) Collect 1000 random players' information.
2) Tune algorithm.
3) Midterm Report editing.
4) Final report editing.

Everybody

1) Proposal.
2) Midterm Report.
3) Final report.
4) Presentation.