DOTA2 Pre-game and Post-game Win Rate Predictor

Yuxiang Hu, Tiancheng Jiang

***Abstract* — DOTA2 (Defense of the Ancients 2), the esport game with the highest prize pool in the world is also lucrative for its in-game data. With the data of more than 50,000+ DOTA2 public matches, we build two predictors to predict the winning side pre-game and post-game.**

1. Introduction

DOTA 2 is a multiplayer online battle arena(MOBA) game developed by Valve. In this type of games, players form two random teams of 5 and play against the other team aiming to destroy the main turret of the opposing team. Each player can select a champion from a champion pool of over 100 champions, where each champion has unique skills and initial capabilities. In game, gold can be acquired by killing minions or killing champions on the opposing team. Gold acquired can be used to purchase items that improves the champions' capabilities substantially. Different champions are suited for different items in game, so the choosing of items purchased also has a significant impact on the result of the game.

In DOTA 2, the two teams—known as the Radiant and Dire—occupy fortified bases in opposite corners of the map, which is divided in half by a crossable river and connected by three paths, which are referred to as "lanes". The lanes are guarded by defensive towers that attack any opposing unit who gets within its firing range. Gold is only awarded to the player when the enemy target is killed by damage from the player, not the turrets or minions. This is referred to as 'Last hit' in games.

2. Data Collection And Cleaning:

The dataset we are using are recent DOTA 2 match statistics. We used public API at opendota.com (https://docs.opendota.com/#section/Introduction) to pull down the data of roughly 54000 recent DOTA 2 matches. The dataset included the in-game statistics of all ten players, the duration of the game, the meta of the game and so on. A list of all the keys are as Figure (a).

```
df.keys()
```

```
Index(['all_word_counts', 'barracks_status_dire', 'barracks_status_radiant',
       'chat', 'cluster', 'comeback', 'cosmetics', 'dire_score', 'dire_team',
       'dire_team_id', 'draft_timings', 'duration', 'engine', 'error',
       'first_blood_time', 'game_mode', 'human_players', 'league', 'leagueid',
       'lobby_type', 'loss', 'match_id', 'match_seq_num', 'my_word_counts',
       'negative_votes', 'objectives', 'patch', 'picks_bans', 'players',
       'positive_votes', 'radiant_gold_adv', 'radiant_score', 'radiant_team',
       'radiant_team_id', 'radiant_win', 'radiant_xp_adv', 'region',
       'replay_salt', 'replay_url', 'series_id', 'series_type', 'skill',
       'start_time', 'stomp', 'teamfights', 'throw', 'tower_status_dire',
       'tower_status_radiant', 'version'],
      dtype='object')
```

Figure (a)

An example entry looks like Figure (b)

```
print(match_data[50000])
```

{'match_id': 5110113801, 'barracks_status_dire': 63, 'barracks_status_radiant': 3, 'chat': None, 'cluster': 131, 'cosmetics': None, 'dire_score': 34, 'dire_team_id': None, 'draft_timings': None, 'duration': 1748, 'engine': 1, 'first_blood_time': 10, 'game_mode': 22, 'human_players': 10, 'leagueid': 0, 'lobby_type': 7, 'match_seq_num': 4287734194, 'negative_votes': 0, 'objectives': None, 'picks_bans': [{'is_pick': False, 'hero_id': 105, 'team': 0, 'order': 0}, {'is_pick': False, 'hero_id': 54, 'team': 0, 'order': 1}, {'is_pick': False, 'hero_id': 47, 'team': 0, 'order': 2}], 'positive_votes': 0, 'radiant_gold_adv': None, 'radiant_score': 13, 'radiant_team_id': None, 'radiant_win': False, 'radiant_xp_adv': None, 'skill': 1, 'start_time': 1573566148, 'teamfights': None, 'tower_status_dire': 2047, 'tower_status_radiant': 4, 'version': None, 'replay_salt': 412682192, 'series_id': 0, 'series_type': 0, 'players': [{'match_id': 5110113801, 'player_slot': 0, 'ability_targets': None, 'ability_upgrades_arr': [5127, 5128, 5126, 5128, 5128, 5129, 5128, 5126, 5126, 5903, 5126, 5129, 5127], 'ability_uses': None, 'account_id': 156665192, 'actions': None, 'additional_units': None, 'assists': 5, 'backpack_0': 0, 'backpack_1': 0, 'backpack_2': 0, 'buyback_log': None, 'camps_stacked': None, 'connection_log': None, 'creeps_stacked': None, 'damage': None, 'damage_inflictor': None, 'damage_inflictor_received': None, 'damage_taken': None, 'damage_targets': None, 'deaths': 5, 'denies': 3, 'dn_t': None, 'firstblood_claimed': None, 'gold': 85, 'gold_per_min': 201, 'gold_reasons': None, 'gold_spent': 5850, 'gold_t': None, 'hero_damage': 9066, 'hero_healing': 0, 'hero_hits': None, 'hero_id': 5, 'item_0': 34, 'item_1': 1, 'item_2': 214, 'item_3': 40, 'item_4': 0, 'item_5': 21, 'item_uses': None, 'kill_streaks': None, 'killed': None, 'killed_by': None, 'kills': 1, 'kills_log': None, 'lane_pos': None, 'last_hits': 58, 'leaver_status': 0, 'level': 13, 'lh_t': None, 'life_state': None, 'max_hero_hit': None, 'multi_kills': None, 'obs': None, 'obs_left_log': None, 'obs_log': None, 'obs_placed': None, 'party_id': 0, 'party_size': 1, 'performance_others': None, 'permanent_buffs': [{'permanent_buff': 6, 'stack_count': 1}], 'pings': None, 'pred_vict': None, 'purchase': None, 'purchase_log': None, 'randomed': None, 'repicked': None, 'roshans_killed': None, 'rune_pickups': None, 'runes': None, 'runes_log': None, 'sen': None, 'sen_left_log': None, 'sen_log': None, 'sen_placed': None, 'stuns': None, 'teamfight_participation': None, 'times': None, 'tower_damage': 0, 'towers_killed': None, 'xp_per_min': 276, 'xp_reasons': None, 'xp_t': None, 'personaname': 'по хуй обнулят', 'name': None, 'last_login': None, 'radiant_win': False, 'start_time': 1573566148, 'duration': 1748, 'cluster': 131, 'lobby_type': 7, 'game_mode': 22, 'is_contributor': False, 'patch': 41, 'region': 3, 'isRadiant': True, 'win': 0, 'lose': 1, 'total_gold': 5855, 'total_xp': 8040, 'kills_per_min': 0.034324942791762014, 'kda': 1, 'abandons': 0, 'rank_tier': 45, 'cosmetics': [], 'benchmarks': {'gold_per_min': {'raw': 201, 'pct': 0.03774464119291705}, 'xp_per_min': {'raw': 276, 'pct': 0.057315936626281455}, 'kills_per_min': {'raw': 0.034324942791762014, 'pct': 0.14259086672879775}, 'last_hits_per_min': {'raw': 1.990846819221966, 'pct': 0.619291705498602}, 'hero_damage_per_min': {'raw': 311.1899313501144, 'pct': 0.25023209916129301955}, 'hero_healing_per_min': {'raw': 0, 'pct': 0.7232059645852575}, 'tower_damage': {'raw': 0, 'pct': 0.18359739049394222}, 'stuns_per_min': {'raw': 0, 'pct': 0.0009319664492078285}, 'lhten': {}}}, {'match_id': 5110113801, 'player_slot': 1, 'ability_targets': None, 'ability_upgrades_arr': [5723, 5716, 5723, 5716, 5721, 5725, 5723, 5723, 5716, 5906, 5716, 5725, 5721, 6421, 5721, 5721], 'ability_uses': None, 'account_id': 127045359, 'actions': None, 'additional_units': None, 'assists': 2, 'backpack_0': 11, 'backpack_1': 0, 'backpack_2': 0, 'buyback_log': None, 'camps_stacked': None, 'connection_log': None, 'creeps_stacked': None, 'damage': None, 'damage_inflictor': None, 'damage_inflictor_received': None, 'damage_taken': None, 'damage_targets': None, 'deaths': 8, 'denies': 9, 'dn_t': None, 'firstblood_claimed': None, 'gold': 129, 'gold_per_min': 329, 'gold_reasons': None, 'gold_spent': 9575, 'gold_t': None, 'hero_damage': 13864, 'hero_healing': 0, 'hero_hits': None, 'hero_id': 114, 'item_0': 240, 'item_1': 8, 'item_2': 21, 'item_3': 75, 'item_4': 252, 'item_5': 50, 'item_uses': None, 'kill_streaks': None, 'killed': None, 'killed_by': None, 'kills': 7, 'kills_log': None, 'lane_pos': None, 'last_hits': 112, 'leaver_status': 1, 'level': 16, 'lh_t': None, 'life_state': None, 'max_hero_hit': None, 'multi_kills': None, 'obs': None, 'obs_left_log': None, 'obs_log': None, 'obs_placed': None, 'party_id': 1, 'party_size': 2, 'performance_others': None, 'permanent_buffs': None, 'pings': None, 'pred_vict': None, 'purchase': None, 'purchase_log': None, 'randomed': None, 'repicked': None, 'roshans_killed': None, 'rune_pickups': None, 'runes': None, 'runes_log': None, 'sen': None, 'sen_left_log': None, 'sen_log': None, 'sen_placed': None, 'stuns': None, 'teamfight_participation': None, 'times': None, 'tower_damage': 161, 'towers_killed': None, 'xp_per_min': 41

Figure (b)

After converting the json objects into a python dataframe, there are some null values. In order to smooth the dataset for classification, null values are dropped. This is intended to avoid destroying the effectiveness of linear models. Note that this does not affect in-game data, as in-game data are enclosed in a dict within the "players" column.

```
df[10000:10500]
```

10027	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
10028	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
10029	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
...	...	...	...	...	...	...	...	...	...	...	...
10470	NaN	0.0	63.0	None	132.0	NaN	None	29.0	NaN	NaN	...
10471	NaN	63.0	3.0	None	183.0	NaN	None	57.0	NaN	NaN	...
10472	NaN	3.0	63.0	None	187.0	NaN	None	14.0	NaN	NaN	...
10473	NaN	3.0	63.0	None	153.0	NaN	None	25.0	NaN	NaN	...
10474	NaN	51.0	63.0	None	154.0	NaN	None	38.0	NaN	NaN	...
10475	{'sick': 1, 'play': 1, 'boyz': 1}	51.0	51.0	[{'time': -261, 'type': 'chatwheel', 'key': '8...	191.0	2465.0	{'4501': 2, '4504': 128, '4768': 130, '5220': ...	49.0	NaN	NaN	...

Figure (c)

Match results are represented as 1 when the radiant team wins, 0 when the dire team wins.

3. Exploratory analysis

We first calculated the correlation matrix of the columns in the dataframe. The results are shown in Figure (d).:

```
df.corr()
```

	barracks_status_dire	barracks_status_radiant	cluster	comeback	dire_score	dire_team_id	duration	engine	first_blood_time
barracks_status_dire	1.000000	-0.636706	-0.011994	-0.441564	0.443159	0.072403	-0.000367	NaN	0.003373
barracks_status_radiant	-0.636706	1.000000	0.011620	0.177568	-0.555536	-0.220348	-0.172649	NaN	-0.016376
cluster	-0.011994	0.011620	1.000000	0.011082	-0.031515	0.023539	-0.015299	NaN	0.036655
comeback	-0.441564	0.177568	0.011082	1.000000	0.091833	1.000000	0.477660	NaN	-0.011113
dire_score	0.443159	-0.555536	-0.031515	0.091833	1.000000	0.452221	0.603744	NaN	-0.074155
dire_team_id	0.072403	-0.220348	0.023539	1.000000	0.452221	1.000000	-0.179806	NaN	-0.120449
duration	-0.000367	-0.172649	-0.015299	0.477660	0.603744	-0.179806	1.000000	NaN	0.043456
engine	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first_blood_time	0.003373	-0.016376	0.036655	-0.011113	-0.074155	-0.120449	0.043456	NaN	1.000000
game_mode	-0.006233	-0.003366	-0.325128	0.041801	0.041272	-0.101132	0.034943	NaN	-0.041023
human_players	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
leagueid	0.006976	-0.002167	-0.010069	-0.019188	-0.010446	-0.478368	-0.012345	NaN	0.007782
lobby_type	-0.006519	-0.003377	-0.067756	-0.027593	0.020315	NaN	-0.019121	NaN	-0.051894
loss	-0.333574	0.297765	0.001769	NaN	-0.468108	-1.000000	-0.270161	NaN	0.020322
match_id	0.011999	-0.014828	-0.052035	0.004991	0.021114	0.617974	-0.010903	NaN	-0.005605
match_seq_num	0.011989	-0.015161	-0.051908	0.006007	0.022265	0.618140	-0.008942	NaN	-0.005482
negative_votes	0.002364	0.001657	-0.008175	0.012381	-0.007713	-0.889957	-0.004673	NaN	-0.002265
patch	0.012017	-0.014740	-0.040636	0.015204	0.022445	0.620509	-0.006174	NaN	-0.003551
positive_votes	0.001019	0.000831	-0.010726	0.016287	-0.007750	-0.882581	-0.004554	NaN	-0.000981
radiant_score	-0.478358	0.397324	-0.019142	0.565068	0.018463	-0.283833	0.507391	NaN	-0.102380
radiant_team_id	0.999663	-0.566302	0.341267	1.000000	0.895763	0.915454	0.223846	NaN	-0.070646
region	-0.020756	0.007100	0.816672	0.000878	-0.031061	-0.050573	0.003815	NaN	0.042830
replay_salt	0.003765	-0.006239	-0.010494	0.036456	-0.005118	0.338215	-0.005228	NaN	0.002032
series_id	0.011091	-0.003792	-0.005395	-0.019184	-0.010061	-0.060349	-0.015475	NaN	0.007543
series_type	0.005313	-0.001532	-0.006951	-0.014581	-0.009872	-0.535485	-0.009952	NaN	0.005790
skill	0.012788	-0.027852	-0.074662	0.091657	0.030059	NaN	0.069011	NaN	-0.069143
start_time	0.011908	-0.014781	-0.050921	0.004909	0.020974	0.618553	-0.010787	NaN	-0.005344

Figure (d)

Since most correlation values are close to 0, we can safely assume that these features are relatively independent from one another, thus we shouldn't worry too much about the problem of double counting while constructing feature vectors or doing naïve bayes.

We then plotted histograms of "radiant_score", "dire_score", "tower_status_radiant" and "tower_status_dire". It seems that the distribution of radiant and dire team statistics is very similar. This means that the game is well balanced for different factions in the game at this point. Whether the player is on the dire team or the radiant team shouldn't be a good feature for the model. Figures are shown below.

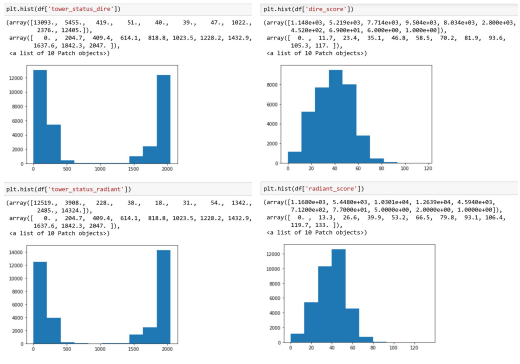
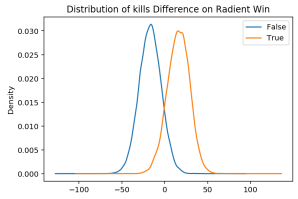


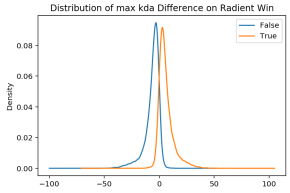
Figure (e)

We plotted the gold acquired by each player with respect to the result of the game. It seems that there is a clear difference in the average of gold obtained per minute between games won and games lost. This means that gold per minute can be a good feature to be incorporated into the design of our model.

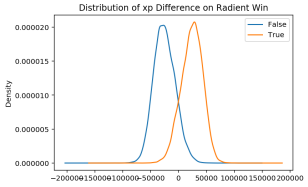
The following Figure(f) visualizes the effect of gold difference on the result of the match. The winning side tends to have a positive gold difference at the end of the match. Therefore, gold difference is also a good feature to consider while building our model.



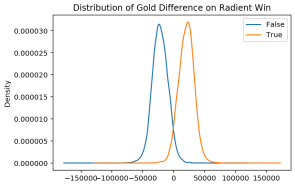
Figure(f)



Figure(g)



Figure(h)



Figure(i)

Figure(g) visualizes the difference of max kill death ratio (kda) between two teams. Kill death ratio is a good measurement of the in-game performance of a player. A high kda means that a player kills a lot of enemy heroes and dies only a few times. The player with max kda in each team is the best performing player. Intuitively, we want to know whether the best performing player can carry his team to win. From the graph, though the team with higher max kda players tend to win, the difference is not as significant as we expected. This might not be a good feature to build model.

Figure(h) and Figure(i) are the graphs of total experience difference and total kill difference conditional on Radiant win. As the distributions are seperate. Those two features might be good to predict the win/lose

We also plotted a scatterplot of total gold of the radiant team and the dire team. The winning team tends to have higher total gold, so the data points right of the diagonal is more likely to be a radiant victory, the data points left of the diagonal is likely to be a dire victory. A lot of data points are located near the diagonal, making the classification task difficult.

```
plt.scatter(ra, di)
<matplotlib.collections.PathCollection at 0x18dd7c80390>
```

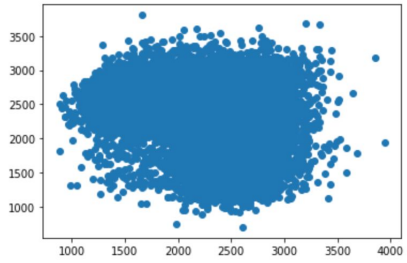


Figure (j)

4. Predictive Tasks

We designed two predictive tasks for our model. The first one gives a pre-game prediction of the result of the match with data from the ban/pick stage, where selection of champions are done. The second one is to train a real-time predictor of the probability of winning for both sides in game. An ongoing match should have data to all the features in our dataframe accessible in real time. Therefore, we can build a model that predicts the result of the game with it. We also want to predict the probability of winning with the data at hand.

5. Pre-game Prediction

The first predictive task we want to solve is pre-game win/lose prediction. More specifically, we want to build a model to predict the win/lose only with the statistics we can obtain before the match starts (ie. hero draft, skill level, game mode, etc), without using ingame statistics of match (ie. gold, xp, kill death ratio, etc).

5.1 Pre-game Prediction Baseline

Since this is a binary classification problem, we set the baseline accuracy to be 0.5. And we expect the prediction accuracy to be close to 0.5, because the players on both sides should start at the same point before the game starts. The hero draft should not affect the win rate if all the heroes are balanced. Also, the matching system is expected to balance the skill level of Radiant and Dire. As we assume dota2 is a well-balanced game, we expect the win/lose should only based on the ingame performance if dota2 is well-balanced.

5.2 modeling and methodologies

To represent the ban/pick of champions, we use one-hot encoding to create a vector for each game. which encodes the information as a vector of 0s and 1s, where 1s indicate the champions selected. As dota2 have 130 heros, we use an array of 260 entree to represent a game. We have 5 1s in the first 130 entree representing the hero combination of Radiant and 5 1s in the last 130 entree representing Dire.

Other categorical columns are also one-hot encoded, including game mode, skill level, region, league and start time.

5.3 Model Training and Evaluation

We separated data roughly by an 80/10/10 ratio, using the first 80% data as training set and the latter 10% as validation set and the last 10% as test sets.

We built the feature based on the sparse matrix built in 5.2. and ran a logistic regression on the training set. The accuracy of the model on the test set is around 62.33 percent. The performance of SVM classifier is 61.37.

This turned out to be a week predictor as we expected. However, the 62 percent accuracy predicting the win/lose before the game starts is way above the 50-50 for each team as we expected. This means that dota2 is not perfectly balanced. Some heroes are overpowered and some are too weak.

6. Post-game Prediction

The second predictive task we want to solve is post-game win/lose prediction. In this section, we want to build a model utilizing all the statistics and data we have to build a high performance model to predict the win/lose after the game ends.

6.1 Post-game Prediction Baseline

Since this is a binary classification problem, we set the baseline accuracy to be 0.5.

6.2 modeling methodologies

In the scatterplot we made on total gold, classification is hard around the diagonal. Support vector machines are useful for handling situations like this, as SVM focuses on the data points close to the line that separates the two categories.

In our exploratory data analysis, we identified gold per minute as a good feature to use. Therefore, we are interested in building a feature vector that consists of the gold per minute data of all 10 players in the match.

The first 5 players are on the radiant team and the latter 5 are on the dire team. We can then run a logistic regression with what we have.

In addition, as we plotted in Exploratory Analysis, the difference of total gold between two teams, the difference of total experience between two teams, the difference of kills are all good features to use.

Other features we considered are one-hot encodings of champion selection and items purchased. As shown in the introductory part, champions and items strongly influences the result of the match, so including these as features should be beneficial for our model.

6.3 Model Training and Evaluation

We separated data roughly by an 80/10/10 ratio, using the first 80% data as training set and the latter 10% as validation set and the last 10% as test sets.

For the Post-game predictor, we first trained a svm classifier on a feature vector that consists of the gold per minute data of all 10 players in game. The resulting accuracy is only 55 percent.We changed our feature vector to the sum of gold per minute data of players on both teams. The resulting accuracy improved to 0.62, still not a satisfactory result.

We then trained another model with logistic regression using the gold per minute data of all 10 players in game to build the feature vector.

A logistic function or logistic curve is a common "S" shape (sigmoid), with equation:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

Figure (k)

where e = the natural logarithm base (also known as Euler's Number),

x0= the value of the sigmoid's midpoint,

L= the curve's maximum value, and

k = the logistic growth rate or steepness of the curve

This classifier reached 95.7 percent accuracy on our test set. After changing the feature to the total gold acquired by all 10 players in game, the accuracy dropped to 93.7 percent. Nevertheless, we are satisfied with the results this model delivers.

However, we are concerned about overfitting in this model. The features don't contain any time data, which is an important factor in predicting match outcomes. In MOBA games, there are champions that are powerful in the early stage of the game, while some champions become powerful going into the game as they acquire more gold to purchase items. If the test set consists of data points with an early timestamp, the behavior of our model on the test set would be bad. Also, some champions rely more on the experience they gain in game to upgrade their skills compared to gold acquired. If the test set contains a large amount of matches that included these champions, our model will not perform well. Therefore, we added the total experience obtained by all 10 players in game and game duration into the feature vector. Running logistic regression with this feature vector gave us 95.8 accuracy on the test set. We changed our logistic regression regularization constant to 0.01 to further counter overfitting in our model.

7. PREVIOUS WORK

There are some previous work on predicting the outcome of MOBA matches. [1] Methods used includes gradient boosted trees, which uses a decision tree structure that does not rely on linear features to perform well. At the ith iteration, a decision tree h(i) with j leaves separate the input field into j regions, denoted by R(j). Then do gradient descent on each interval of the input field to find the global minimum of the loss function. The mathematical rules for this algorithm are listed below:

loss function

$$\arg\min_F E_{x,y}[L(y, F(x))] \qquad F(x) = \sum_{i=1}^{m} \gamma^{(i)} h^{(i)}(x) + c$$

$$F^{(i)}(x) = F^{(i-1)}(x) + \sum_{j=1}^{J} \gamma_j^{(i)} h^{(i)}(x)[x \in R_j^{(i)}]$$

$$\gamma_j^{(i)} = \arg\min_{\gamma} \sum_{x_k \in R_j^{(i)}} L(y_k, F^{(i-1)}(x_k) + \gamma h^{(i)}(x_k))$$

Figure (l)

This study concluded that pre-game data is a weak predictor of the result of the game, while in-game data is a strong predictor. This is consistent with the results we get with our model.

There is also work on using champion selection data to create a recommender system on drafting champions in game. [2] The article defined an association rule I => j, where j is a single item in a set I. If I is a subset of another set K, then j is likely to be in K. After that, the author used the Apriori Algorithm to extract association rules that tells us which champions have a higher chance of winning when they are on the same side.

The algorithm workflow is further illustrated by the chart below:

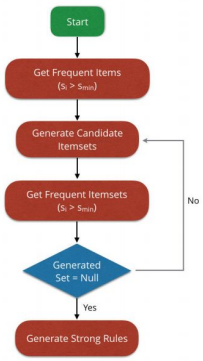


Figure (m)

The author then built a neural network using selected champions as input data to predict the result of the match using sigmoid function as the activation function. Using the predictor and the association rules obtained earlier, the author built a recommender system that can draft a team with a statistically significantly higher rate of winning when the opposing team is drafting champions randomly.

8. Conclusion/Future improvements

In all, we built two predictors to predict the winning side of a single DOTA2 game. The first predictor, the Pre-game predictor, featuring the hero draft, game mode, skill levels and other pre-game statistics, is a weak predictor of 62% accuracy. However, this is reasonable, as the matching system and heroes should be relatively balanced in this game, avoiding any factors before the game even starts advantaging one side too much. The second predictor, Post-game predictor is a much stronger predictor, reaching accuracy over 95%, utilizing all the statistics and data in the whole game. We used the logistic regression for both predictors not only because it has the best performance, but also for its convenience of showcasing probability. The probability of 1 (Radiant win) can be easily used as a win rate estimator, which is widely used in real life.

In fact, we intended to build the three predictors at first, Pre-game, In-game, and Post-game predictor. However, we are not able to build the In-game predictor, which should be a predictor that can predict the win rate based on the statistics in real time when the game is playing because of data deficiency. Building this predictor required us to use several arrays in the data set which track some key features of both sides each minute (ie. the array tracking the gold a player hold every minute in game). Unfortunately, as the data we used in this project are real world data scraped from public api, only 3,000 points out of 50,000 + data points contains such arrays, and the rest of them are all NANs. Thus, we can only give up on our In-game predictor. We will try to scrape more data and build it up later.

References

[1] League of Legends Match Outcome Prediction, Lucas Lin, 2016

[2] A Recommender System for Hero Line-Ups in MOBA Games, Lucas Hanke, Luiz Chaimowicz, Computer Science Department Universidade Federal de Minas Gerais, Proceedings, The Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)