

Dota 2 Draft Based Win Prediction

CSC 503: Data Mining Final Project

Sunil Kumar

December 2, 2019

1 Abstract

The goal of this project is to train a machine learning model to predict the winning team for the popular online game Dota 2 as accurately as possible based on the selected heroes by both teams at the start of their game. We collected data using the Stream API for recently played matches and applied different machine learning models in our experiments to get the best win predictor model. We have included all details including data collection, exploratory analysis, feature selection, modeling, and results.

2 Introduction

Dota 2 or Defense of the Ancients 2 is a 5v5 Multiplayer Online Battle Arena (MOBA) video game developed and published by Valve. Dota 2 holds the largest prize pools tournaments in e-sports every year. In 2018, the estimated prize money for Dota 2 tournaments reached 41.4 million U.S. dollars [?]. A Dota 2 match consists of a match between two teams of five players with each of them controlling a single character, known as a hero. Both sides, Radiant and Dire, attempt to destroy one another's fortress, the Ancient. Team who destroy the opponent's Ancient first win the game. Currently, there are 117 different heroes to choose from. One hero can be only selected by one player i.e. no two player can play with same hero. The choice of heroes and the combination of heroes in the team plays a major role in determining the match result. Each hero's strengths and weaknesses can serve to complement other heroes on the friendly team or counter heroes on the opposing team. The concept of accumulating in-game gold is called farming. In general, a proper team select a hero for each of the following 5 different positions:-

1. Position 1 (Carry) - Hero have highest farm priority and becomes strongest when game reaches long duration.
2. Position 2 (Mid) - Hero strength is less dependent on items more on experience/levels. Controls the tempo of the game.
3. Position 3 (Offlaner) - Hero who starts in a hard lane and has an ability to be tank or starts a good team fight. strong team fights.
4. Position 4 & Position 5 (Supports) - Lowest farm priority, help others.

So, hero picked for each position impacts the game differently, in other words same hero can be picked for any of the position but will have different impact on the game outcome based on its position in the draft.

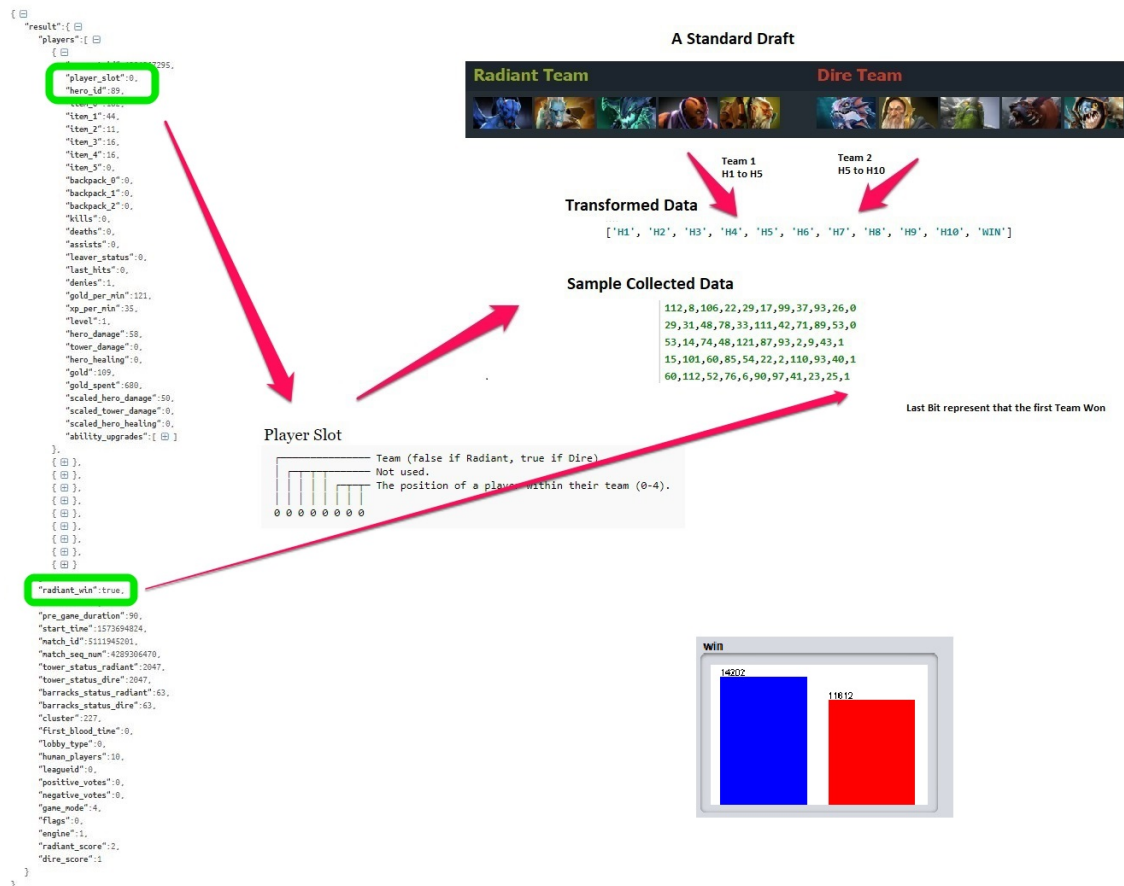
There is clear need of machine learning model to either recommends hero which would increase the win probability or predict the win from the select draft of the two teams. We have selected win prediction as our primary interest in this project.

3 Data Collection

We collected 30,000 recently played matches data from 23-Nov-2019 to 24-Nov-2019 via Dota 2 Steam Web-API. We used **GetMatchHistory** to get all the match ids and then used **GetMatchDetails** to get details of the match. We have following criteria to filter out the valid matches for our model:-

1. We removed the games where one or more of the player left the game before the match finishes. This kind of games are unfair to the team which less player and hence unfit to use in our model.
2. Only 'very high' skill level games are considered. This will decrease the impact of player skills on the match outcomes.
3. Game mode is 5v5 only. Other modes such as 1v1 is not a standard dota match.

Image Below show the transformation of data from the API into the required draft data for our machine learning model. Finally we had our 28.5K rows with 10 features column data.



4 Experiment and Analysis

4.1 Initial Experimentation and Results

4.1.1 Decision Tree using ID3

We divided the data into training set (around 26000 rows) and test set (1000 rows) and then tried to apply ID3 decision tree algo. using weka on our initial data.

The following Tree was generated (we were not able to get the picture of the whole tree):-

```
=== Classifier model (full training set) ===
```

```
Id3
```

```
h4 = 1
| h10 = 1: null
| h10 = 2: 1
| h10 = 3: 0
| h10 = 4
| | h1 = 1: null
| | h1 = 2: null
| | h1 = 3: null
| | h1 = 4: null
| | h1 = 5: null
| | h1 = 6: null
| | h1 = 7: null
| | h1 = 8: null
| | h1 = 9: null
| | h1 = 10: null
| | h1 = 11: null
| | h1 = 12: null
| | h1 = 13: null
| | h1 = 14: null
| | h1 = 15: null
| | h1 = 16: null
| | h1 = 17: null
| | h1 = 18: null
```

```
=== Summary ===
```

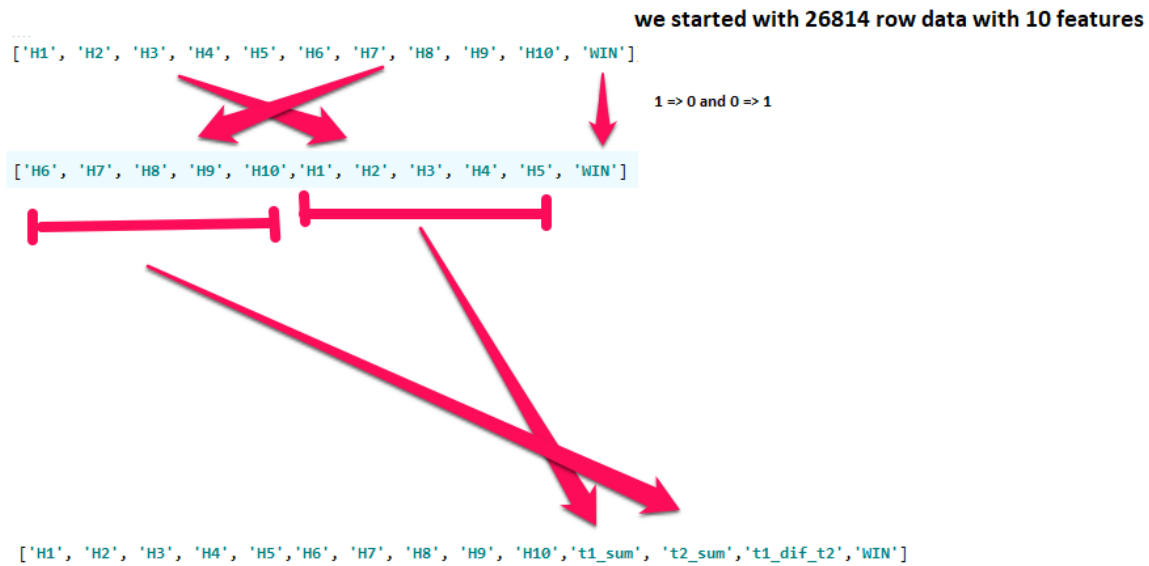
Correctly Classified Instances	190	19	%
Incorrectly Classified Instances	196	19.6	%
Kappa statistic	-0.0393		
Mean absolute error	0.5078		
Root mean squared error	0.7126		
Relative absolute error	266.2779	%	
Root relative squared error	230.9928	%	
Unclassified Instances	614	61.4	%
Total Number of Instances	1000		

Looking at the tree we can see it says if 'AntiMage' i.e. hero id 1 is selected by team one at pos 4 and team two have 'Axe' i.e. hero id 2 then team one will win.

But when we see results of the test they say 61% of the test instances were unclassified and the one which were classified had more then 50% error (Incorrectly Classified). This demonstrate that Win prediction is now a simple task for a simple decision tree. we moved on to apply more complex models.

4.1.2 Machine Learning models using Sklearn

We applied different models to our data but the results were not good , so we doubled our data by changing the team 2 as team 1 and reversing the results. this gave us a total of about 55000 rows to train on. we also tried to add 3 extra columns 'team 1 total sum', 'team 2 total sum' and 'difference of team 1 and team 2' thinking this might help us improve. The initial experimental feature engineering is showing in the below image:-



Finally we had 53628 row data for training with 3 custom features

Below are our Results (Average of 5 Cross Validation 80% train : 20% test):-

Model	Default Data Set with 26.5K rows		Modified Data Set with 54K rows	
	Accuracy on Training Set	Accuracy on Test Set	Accuracy on Training Set	Accuracy on Test Set
Logistic Regression	54.95314904	55.15569644	50.9032213	51.41245572
Random Forest Classifier	100	53.39051526	100	51.23997762
Ada Boost Classifier	56.84428076	55.07489589	54.44967601	52.29815402
Extra Trees Classifier	100	53.0735285	100	50.82975946
Linear Discriminant Analysis	54.95314904	55.16191187	50.865927	51.47305613
K-Neighbors Classifier	75.05166814	48.27521909	74.82751387	50.42420287
Decision Tree Classifier	100	50.93542172	100	50.72720492
Gaussian Naive Bayes	54.97645797	55.16812729	51.16311594	51.6175648
Perceptron	53.16613056	53.07974392	50.19113328	50.14450867



4.1.3 Analysis of Initial Experimentation

Our data manipulation to create more data and adding more column seems to not help. We realise that our initial data of 26.5K had 55% win and 45% loss and after modifying the data we had 50% win and 50% loss and hence the accuracy dropped.

Another reason for bad performance was that our data had all 10 columns as categorical

attributes (with Label-Encoded) because they represent hero ID and hence hero with ID 1 is not less/weak then hero with ID 100. Hero id does not represent strength of the hero in any way. Hence our data was miss-leading for the machine learning models and needed another round of Features Engineering where we need to


4.2 Evaluation and Analysis

4.2.1 Final Features Engineering and training


As all of our 10 columns are categorical attributes with 117 possible values, we used One-Hot Encoding and got 1170 features.

One hot encoding per column

Player 1 had hero id 5



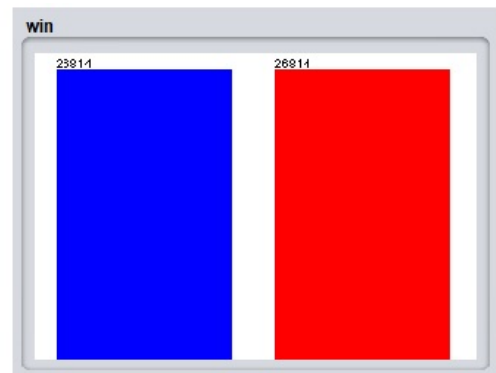
Winner is Team 1



	0	1	2	3	4	5	6	7	8	9	...	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170
0	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

Now our data size have become large due to so many columns per row. Hence we applied all model again we could and got following results:-

Model	One Hot encoded Data with 54K row x 1170 features	
	Accuracy on Training Set	Accuracy on Test Set
Logistic Regression	59.57764207	55.4726832
Decision Tree Classifier	100	50.16781652
Gaussian Naive Bayes	58.7944618	54.73615514
Perceptron	53.00452193	51.82733545
Random Forest Classifier	100	54.25135186
Ada Boost Classifier	59.47042096	55.43539064
Extra Trees Classifier	100	52.89017341
K-Neighbors Classifier	75.65381567	50.73652806
Linear Discriminant Analysis	59.53568598	55.49132948



We have gained up to 5% accuracy but not good enough. Lets try one more transformation of our data as following:-

Combined One Hot Encoding

Hero Selected by Team 1

	0	1	2	3	4	5	6	7	8	9	...	120	121	122	123	124	125	126	127	128	129
0	0	0	0	0	1	0	0	-1	0	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	-1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	-1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

Hero Selected by Team 2

Match Result

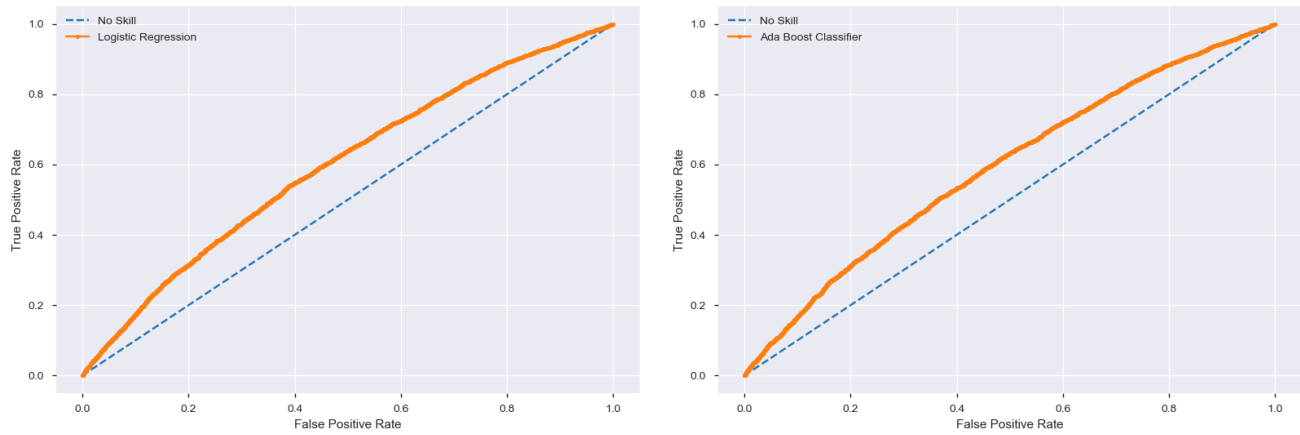
Here we are losing the information that first player have selected which hero but rest of the information is still there i.e. all 5 selected heroes by team 1 as 1 and team 2 as -1. Now we again applied all models to this data-set of 130 columns. Following are the results:-

	Combined One Hot Data with 54K row x 129 features	
Model	Accuracy on Training Set	Accuracy on Test Set
Logistic Regression	57.68029462	57.16949469
Decision Tree Classifier	100	50.96028342
Gaussian Naive Bayes	56.19784625	53.36565355
Perceptron	53.27257471	52.20958419
Random Forest Classifier	100	54.89464852
Ada Boost Classifier	57.64999301	56.61010628
Extra Trees Classifier	100	54.57766176
K-Neighbors Classifier	75.65847746	51.26794704
Linear Discriminant Analysis	57.66630926	57.09490957

Here we notice another 2-3% improvement but not much. Lets move on to Analysis section now. We will be working with Logistic Regression as it had the highest test accuracy.

4.2.2 Evaluation

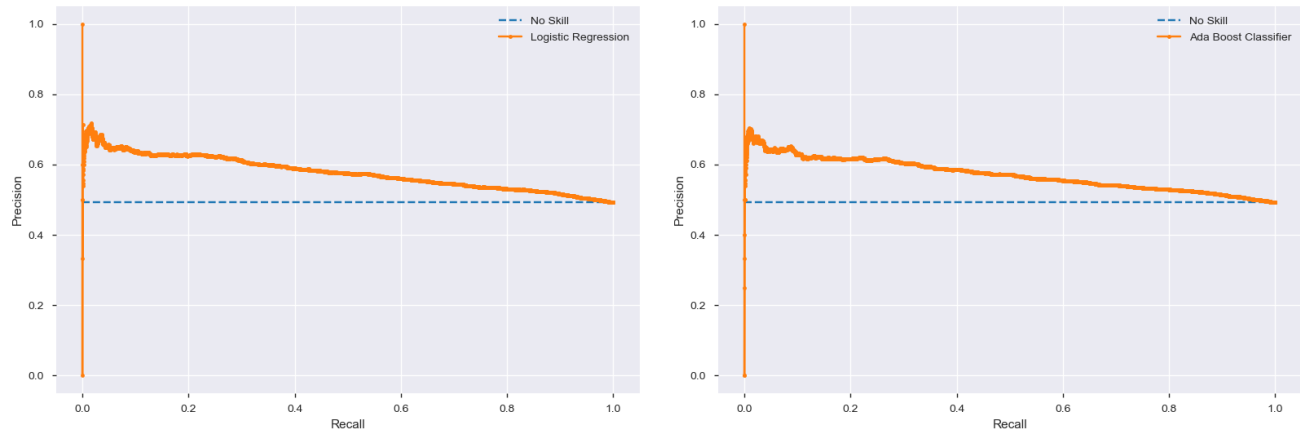
Below are the 2 ROC curve we obtained from Logistic Regression and Ada Boost Classifier:



From the ROC curve we also got ROC accuracy results on test set (random 20%) as following:-

Models	ROC Accuracy (test set)
Logistic Regression	59.9
Ada Boost Classifier	59.3
Linear Discriminant Analysis	59.9

And we also obtained Precision-Recall plot on the same models :



We also plotted confusion matrix but the output image had issue so not included here.

5 Conclusion

We tried to come-up with a machine learning model to predict the winner of a game based on just the draft. Our expectation was that we could get up-to 75% accuracy but we only get a maximum of 60% among all the experiments we did. Results shows that it is not easy to predict the outcome of a match based upon only the draft and that other factors like player skill level must have a role in the outcome as well.

For future work we will try to fetch player performance stats from there recent games and add to our data-set to predict the winner of a game based on draft and player performance in recent history.

6 APPENDIX

1. Hero IDs to hero Names can be found here : <https://liquipedia.net/dota2/MediaWiki:Dota2webapi-heroes.json>
2. API details can be found at: <https://dev.dota2.com/showthread.php?t=58317>
3. ROC plot and PR plot code was motivated from : <https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python>
4. dataCollection.py - data collection from the API and saving to the database
5. Solution.py - apply models on final data, plot ROC and PR curves
6. InitialData.csv - data we started with 26.5k row 10 features
7. finalData.csv - data with 54k rows and 10 features
8. finalExtraColumnsData.csv - with 3 custom extra features i.e. 54k rows with 13 features. used updateData.ipynb file for the process
9. oneHotData.csv - finalData with onehotencoding.csv 54k rows with 1170 features
10. CombinedOneHotData.csv - combined one-hot encoding 54k rows with 129 features . used combinedOneHotEncoding.ipynb file for the process.
11. dbutils.py - save API data into to postgresql database
12. .arff files for the weka are also