DOTA 2 Win Loss Prediction from Item and Hero Data with Machine Learning

Stanlly
Computer Science Department,
Faculty of Computing and Media,
Bina Nusantara University,
Jakarta, Indonesia 11480
stanlly@binus.ac.id

Fauzan Ardhana Putra
Computer Science Department,
Faculty of Computing and Media,
Bina Nusantara University,
Jakarta, Indonesia 11480
fauzan.putra003@binus.ac.id

Nunung Nurul Qomariyah
Computer Science Department,
Faculty of Computing and Media,
Bina Nusantara University,
Jakarta, Indonesia 11480
nunung.qomariyah@binus.ac.id

***Abstract*—Video gaming has become a titan in the overall market over the past decade, culminating in an estimated worth almost 180 billion US dollars by 2021. Aside from its growing influence in the overall market, video games have also created a new competitive format called eSports, a format where highly skilled players of certain video games play against each other in a tournament to see who the most skilled are and win a prize at the end. ESports are just one of many reasons why people have become interested in the idea of being able to predict the outcome of any given match between players. In this study, We conducted research on the importance of certain factors in determining the win or loss of any given Defense of the Ancients 2, better known as DOTA 2, match. After some trial and error, We found that Item and Hero choices play a large role in the winning chance of any given team. From this we concluded that we would be able to predict a match's outcome solely based off of these two factors and created an accurate model for predicting the outcome of any given match as long as we have the item and hero choices. In this study, we will be employing the use of Decision Tree, Random Tree and XGBoost classifiers in order to create our models. In the end, the XGBoost model ended up being our best model, with an accuracy of roughly 93% which can predict an outcome in roughly 1 minute.**

***Index Terms*—Game, win prediction, classification, machine learning, tree based model**

I. Introduction

Over the past decade, the video game industry has continuously surprised everyone with its rapid growth. Coming from its niche roots in the arcades back in the 1970s to becoming even bigger than the movie and music industry combined by 2021. With the rise of the video game industry, the creation of a new form of competition: "eSports" was created. ESports are essentially a collection of highly skilled players competing against each other in order to see who is the better player, a similar concept to real life sports. Spanning over hundreds of games and multiple unique genres, one of the most popular eSports game genres is the Multiplayer Online Battle Arena (MOBA).

One game in said genre is Defense of the Ancients 2, better known as DOTA 2. As the successor to one of the first major titles in the genre, DOTA 2 has garnered a sizeable fanbase that is still active to this day. Ranking as the 2nd most played game on the game platform Steam. The game is a competition between 2 teams of 5, radiant and dire. Players get a choice of a single hero each, with each hero being able to fill up to six slots with items throughout the match. The goal of each match is to tear down the opposing team's defenses in order to get to and destroy their ancient, a heavily guarded structure. DOTA 2 is considered one of the most lucrative eSports game in the current scene, because of this a lot of people are still interested in being able to predict which team will win or lose. The ability to predict the outcome of a given match could also prove useful in formulating new strategies and determining whether certain combinations of hero and item choices are better than others, a process that is constantly being done as the game evolves.

As avid fans of DOTA 2, we are also curious on whether such a model is possible. With personal knowledge on how the game is structured and a good idea on how each factor plays into the outcome of a match, our team believes that we will be able to create a simple but accurate model that is able to predict the outcome of any given match with ease. We also believe that the ability to predict match outcomes based on item and hero combinations can prove useful in creating "builds", combinations of items and heroes that are used to achieve certain play styles or goals, as well as many other things relevant to the DOTA 2 community.

II. Related Work

There have been multiple different studies conducted by a plethora of different researchers on how to predict the outcome of a DOTA 2 match using many different metrics and factors. Ranging from a study conducted by Grutzik, Higgins and Tran [1] and their 61% accurate team and draft feature model that aimed to take player history into account for their prediction to Akhmedov and Phan's [2], where they used linear regression and neural networks models with 82% and 88% accuracy. The authors of [3], also used four different machine learning models, one of which managed to give up to 85% accurate after 5 minutes of gameplay. Another study conducted by Yang, Qin and Lei [4] tried to create a machine learning model using real-time features which had an accuracy of 93.73% by the 40th minute of any given match.

There are also a few studies that corroborate the idea that item and hero choices play an important role in the outcome of any given match. Such as a study conducted by Wang [5] which uses a neural network trained model that ended with roughly 61% accuracy on the importance of hero drafts on the outcome of a match. Another study by Agarwala and Pearce [6] also tried to use linear regression to create models based on whether hero selection has an effect on win rate, culminating in a model that is 62% accurate. An interesting study conducted by Kinkade, Jolla and Lim [7] tries to incorporate hero matchup, synergy and counters for their model with a final prediction accuracy of 73%.

With these studies in mind, we decided to conduct our research as we noticed that a lot of them did not involve some of the currently popular methods of machine learning classifiers, mainly the ones we used in our paper: Decision Tree, Random Forest and XGBoost. While some of the studies do try to use the hero data and real-time match data to create their models, we also noticed that a majority of them do not combine the features of item and hero data in their predictive models.

III. Methodology

A. Data Collection

In this study, we use a dataset posted by user Devin Anzelmo found on Kaggle[1]. This dataset is an update to a previously posted dataset of the same name. The data is a subset of a data dump created using Opendota, otherwise known as yasp.co [8], which parsed ongoing matches and compiled them into CSV files based on specific parameters[2].

With this dataset, we explore the possibility of creating a predictive model to determine the outcome of any given match through the use of different factors. After a bit of research and experimentation, we determined that the simplest model that we could use to create an accurate model are item and hero data. This dataset is extensive and contains many different CSV files within the overall archive. For our research, we decided to use the CSVs that contain the hero and item names and ids, as well as the match and players CSVs. We chose to focus on using item and hero data alongside the information from the match data as these factors are chosen by the players in a match and are not solely determined through random chance.

In the CSV files that we used, the hero_names.csv and item_ids.csv contain the ID of the hero or item and the names associated with those IDs. Our features from these files will be item_id and hero_id while our label is only radiant_win.

Figure 1 depicts the counts for both false and true booleans for the column radiant_win which depicts a roughly equal amount of wins and losses across the entire dataset.

[1]https://www.kaggle.com/devinanzelmo/dota-2-matches
[2]https://github.com/odota/core/wiki/JSON-Data-Dump

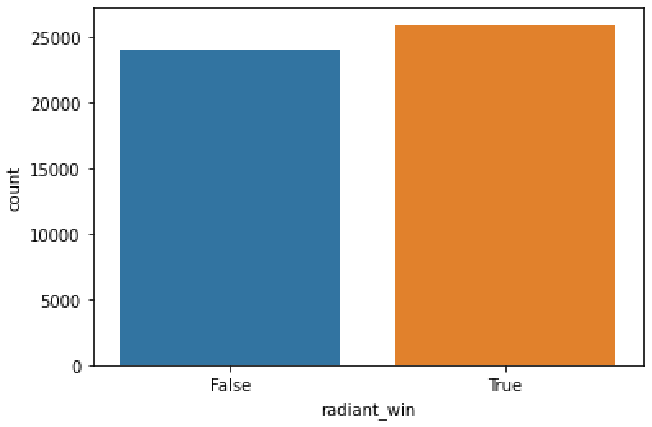


Fig. 1. Counting Radiant Wins and Losses

Figure 2 depicts the top 25 items owned by the end of all matches, with the top item being unknown which represents any empty slot found within the 6 slots that each player owns during any given match.

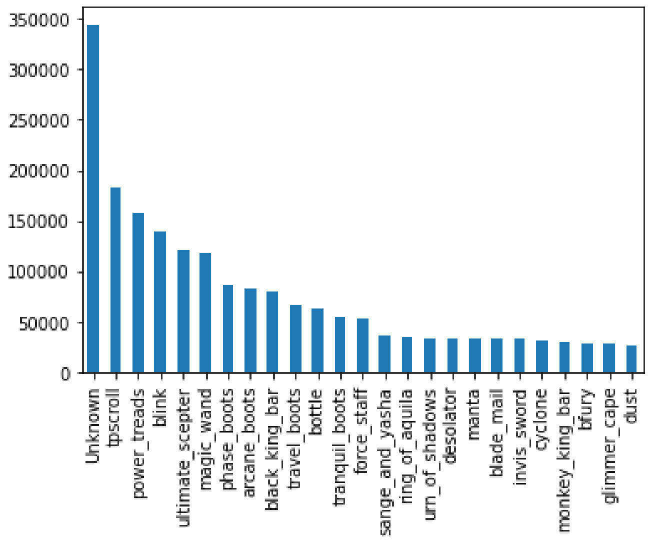


Fig. 2. Top 25 Most Bought Items

Figure 3 depicts the top 25 heroes chosen by players and their respective win rates. As depicted by the graph, there's no real correlation between the picked hero alone and the outcome of a given match. The highest picked hero: Windranger, does not necessarily mean that the team with said hero is more likely to win that match through that pick alone as the highest win rate we see is from teams that picked the hero Legion Commander.

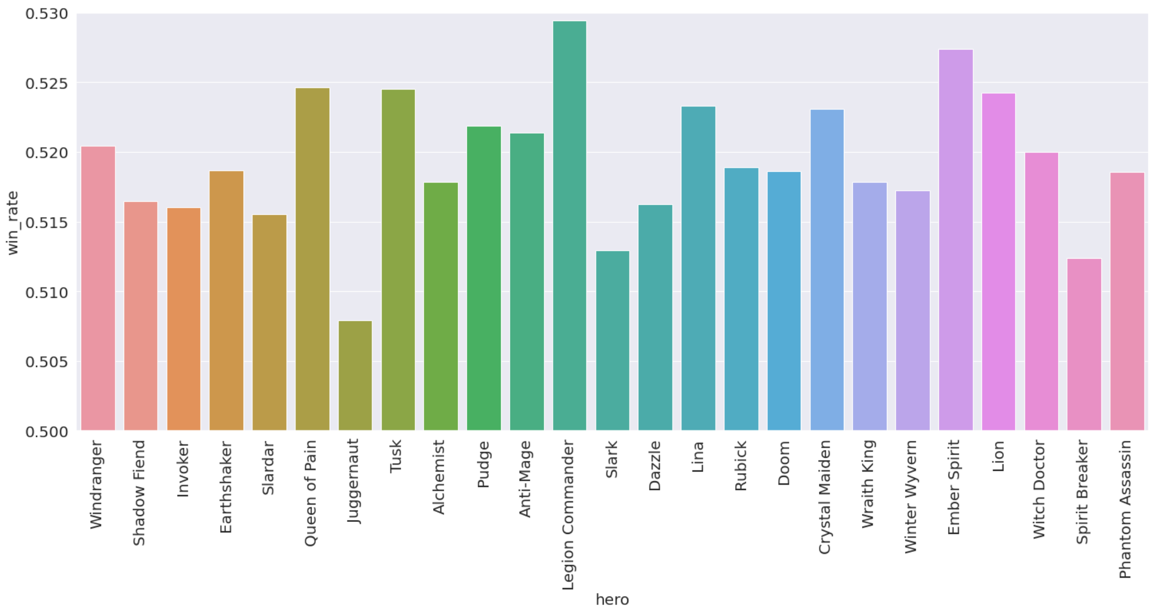
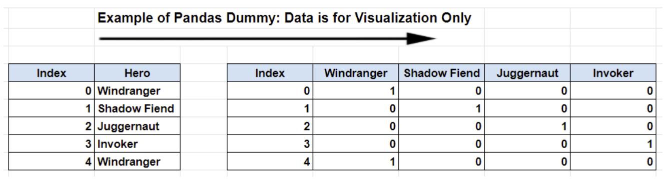


Fig. 3. Top 25 Heroes vs Win Rate

B. Data Preprocessing

After making sure that all item and hero data are correct and that there are no null values, we started to one-hot encode our data through the use of binary encoding. We do this as we are trying to train our predictive model, which learns better from binary data. The following figure depicts the code that we used to do this.

We do this through putting the hero column in the players CSV file through pandas's get_dummies function which turns a categorical column into indicator columns of 0s and 1s. In Figure 4, we can see an example visualization on how get_dummies turns our data into binary.



Example of Pandas Dummy: Data is for Visualization Only

Index	Hero
0	Windranger
1	Shadow Fiend
2	Juggernaut
3	Invoker
4	Windranger

Index	Windranger	Shadow Fiend	Juggernaut	Invoker
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

Fig. 4. Pandas Dummy Binary Encoding

We also did this process to our item data, with a default of 0 in each item slot. The values of these slots are then added in an array that can then be separated during the learning process. Player and hero data are then separated into two teams, this data will also include the previous arrays.

C. Models and Techniques

We decided to use three different classifiers in order to create three different predictive models and find the most accurate model. We used XGBoost, Decision Tree and Random Forest classifiers for creating our predictive models.

The decision tree algorithm works by making decisions based on a condition, which then branches out into multiple endpoints that have different conclusions. Random forest uses the same base concept as decision tree but adds a bit of randomness while calculating a user-defined cluster of different trees, making each individual tree uncorrelated to each other but able to predict more accurately than an individual calculated tree.

XGBoost is an open-source implementation of the gradient boosted trees algorithm, which can be used for regression or classification tasks. It works through the use of boosting, an ensemble method that combines predictions from several models into one, taking each predictor sequentially and improving itself based on its predecessor's errors by giving more weight to predictors that perform better. It also uses a specific type of boosting known as gradient boosting that tries to minimises the loss function using a gradient descent algorithm. It also uses the base concept of decision trees to determine its ”weak” predictors.

We trained our model through the use of the following libraries:

- Xgboost to train our XGBoost classifier model.
- Sklearn to train our Decision Tree and Random Forest classifier models.
- Seaborn and matplotlib for visualizations
- Pandas and numpy for number and array processing

Using the data that we previously processed in section II.B, we will create a training dataset X by concatenating the hero and item data. We will then create data set y and split the data through a boolean to determine whether team radiant had won the match or not. If the data is false, the team is dire. Dataset X is then simplified through the use of Principal Component Analysis (PCA). The data is currently too complex as it has roughly 612 estimated number of features, PCA is used to shrink this number to only 70 features which will improve the training time as well as the efficiency of our models.

In order to test how accurate our models are, we will be employing two methods of testing: cross validation and holdout testing. For our holdout testing, we will be splitting training and test sets as 80% training and 20% testing datasets, while we use sklearn's built-in cross validation method to test the cross validation score.

We decided that our random_state hyperparameter for all of our models are the same in order to gain some form of consistency between each model, a random_state of 42. Each of our models are also using the following hyperparameters:

- For our Decision Tree Classifier, we decided to set the hyperparameters max_depth as 10.
- Our Random Forest Classifier uses a max_depth of "None" which means that all of our nodes are expanded until all leaves are pure or all leaves contain less than the default min_sample_split of 2. We also set the number of n_estimators, the number of trees in the forest, to 100
- Our XGBoost Classifier uses a max_depth of 6 in order to prevent the model from becoming too complex and overfitting, with a gamma of 0. While setting the eta, learning rate, to 0.3 in order to get the best chances of obtaining an optimal model.

IV. Result and Discussion

After training our models, we used the sklearn's built-in cross validation method to determine how accurate our predictive models are. The function is built to evaluate the accuracy of a dataset through the use of different parts of the datasets that our models have not yet seen in order to give a fair assessment on the accuracy of our predictions.

The Decision Tree classifier returned a cross validation score of roughly 82% accuracy with a training time of roughly 19 seconds. Random Forest, on the other hand, returned a cross validation score of roughly 91% accuracy while being trained for 3 minutes. XGBoost is our final and best model of the three we created. Returning an impressive 94% accuracy on the cross validation test and taking only one minute to train. Aside from the cross validation score from sklearn, we also used a few different metrics, including but not limited to: precision, recall and F1 scores as well as how much time was taken to train each model, to measure accuracy in Table I.

TABLE I
Accuracy of Models Created and Time Taken to Train

Classifier	Precision	Recall	F1	Training Time
Decision Tree	0.82	0.82	0.82	19 seconds
Random Forest	0.91	0.91	0.91	3 minutes
XGBoost	0.94	0.94	0.94	~1 minute

As XGBoost returned the best results overall, we will be expanding on the overall accuracy and specifics on our best model. The following depicts the confusion matrix of our XGBoost model. As seen from the matrix 5, we have a total of 677 false metrics from a total of 10,000 samples for a total of 93.23% true accuracy on our predicted labels.

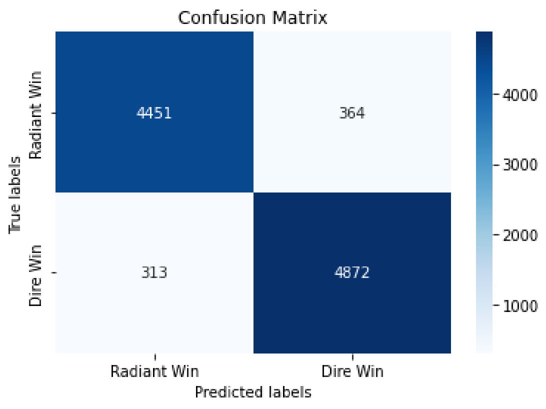


Fig. 5. Confusion Matrix

We also created a Receiver Operating Characteristic (ROC) curve to visualize our model's accuracy. As seen by the visualization in Figure 6, the curve is very close to 1.0 true positive rate, also known as the ideal clinical discriminator, the closer the curve is to this point the more accurate it is. Our curve is very close to this point and away from the diagonal which shows that our model is very accurate.

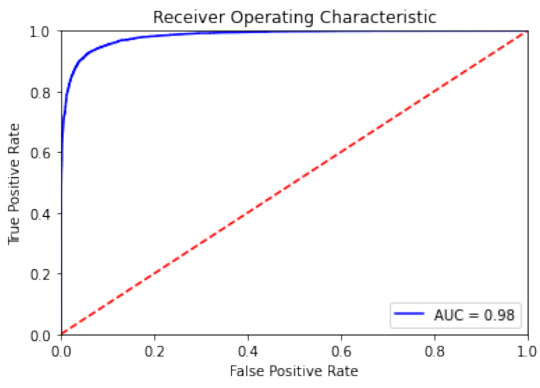


Fig. 6. ROC Curve

Figure 7 is a visualization of the tree created by our XGBoost algorithm model. F* represents the features, as we passed our features through PCA the result is a number from -1 to 1. For example, if f23 is less than -0.2 we then check for f47, etc. The leaf is an estimate of the gradient, which can be predicted manually through the use of the following formula:

$$1/(1+e^{-leafvalue})$$

Now that we know that XGBoost is our final and best model, we decide that we should conduct a test to determine if our model is overfitted. As our model was able to accurately predict the outcome of a match, we will assume that the model is not underfitted. We then used the classification report method that comes with sklearn in order to check for the presence of overfitting. After our test, we concluded that our model has been overfitted.

After analyzing our workflow and our implementation, we boiled down the possible reasons of why our data became

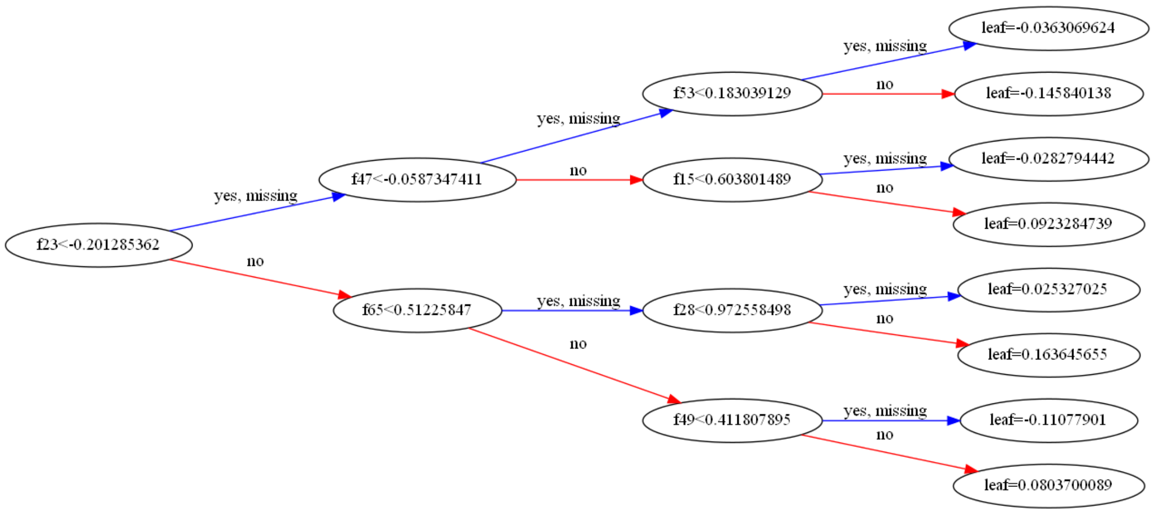


Fig. 7. XGBoost Tree Visualization

overfitted to the following reasons. The first reason is that we failed to standardize the data before starting to train our model. We also failed to use any form of data augmentation to regularize the training data in order to avoid the overfit. We also did not include a stopping mechanism to limit the amount of time the model had spent while training, thus causing an overfit due to training for too long. The final possible reason is that our model is simply too complex.

V. Conclusion and Future Work

We began our research with the goal of creating an accurate predictive model. After seeing the results, we can conclude that we were successful in creating accurate models for predicting the outcome of any given match

As seen from the results, XGBoost is the overall best model that we had created, its speed also means that with a bit of work, we could use XGBoost Model to predict a match's outcome while it is being played as long as we have access to the item and hero data.

We can also safely conclude that item and hero data are definitely integral to creating an accurate prediction of the outcome of a given match.

There are many things that we could have done to improve our models. Normalizing and regularizing the data before training would have definitely created a more accurate model that was not overfitted. We also could have benefited more from planning out which data we wanted to use and for what reasons in order to create a more accurate model that can incorporate more factors into its prediction as we were only able to incorporate item and hero data into the main prediction process.

In hindsight, the ability to predict if a finished match is a win or loss is a bit redundant, but with a bit of modification we could make a graphical display to figure out a match's outcome through the item and hero data at any given point of the match instead which would be more practical for predicting a match's outcome before the match has been completed.

Supplementary Source Code

To see the source code please head to the following link: https://github.com/fauzanardh/DotaMatchPredictor

References

[1] P. Grutzik, J. Higgins, and L. Tran, "Predicting outcomes of professional dota 2 matches," Technical Report. Stanford University, Tech. Rep., 2017.
[2] K. Akhmedov and A. H. Phan, "Machine learning models for dota 2 outcomes prediction," *arXiv preprint arXiv:2106.01782*, 2021.
[3] V. J. Hodge, S. M. Devlin, N. J. Sephton, F. O. Block, P. I. Cowling, and A. Drachen, "Win prediction in multi-player esports: Live professional match prediction," *IEEE Transactions on Games*, 2019.
[4] Y. Yang, T. Qin, and Y.-H. Lei, "Real-time esports match result prediction," *arXiv preprint arXiv:1701.03162*, 2016.
[5] W. Wang, "Predicting multiplayer online battle arena (moba) game outcome based on hero draft data," Ph.D. dissertation, Dublin, National College of Ireland, 2016.
[6] A. Agarwala and M. Pearce, "Learning dota 2 team compositions," *Sl: sn*, 2014.
[7] N. Kinkade, L. Jolla, and K. Lim, "Dota 2 win prediction," *Univ Calif*, vol. 1, pp. 1–13, 2015.
[8] A. Cui, H. Chung, and N. Hanson-Holtry, "Yasp 3.5 million data dump."