Determining optimal play-style to increase win-rate in DOTA 2: A machine learning approach

Rizky Ramdhani[1]

[1] University of Washington Bothell, Bothell, WA 98011, USA
rizkyr@uw.edu

Abstract. Various machine learning models and techniques were used to determine the optimal playstyle to increase win rates in DOTA 2, a popular team-based online multi-player game. Using a provided dataset from Kaggle and the SciKit-Learn machine learning library, models were trained and generated using 10-fold cross validation. Features were ranked based on weights/coefficients of the generated models. While the prevailing belief is for players to play farming core roles to win matches, the data from this research shows that the opposite is true. Players should instead play support roles that enable better team play in order to increase their win rate.

Keywords: Machine Learning, Feature Selection, DOTA 2.

1 Introduction

DOTA2 is a popular online multiplayer game in the genre of MOBAs[1] made by Valve Software. DOTA2 is also notoriously difficult and has one of the steepest learning curve of any game[1]. This project will attempt to identify important attributes that a player should focus on, in order to increase the player's win rate.

1.1 How to play DOTA 2

DOTA 2 is an online multiplayer video game that pits two five-player teams against one another. Each player controls a single avatar (hero) from a pool of 100+ possible heroes, each with its own special abilities, strengths, and weaknesses. No player on either team can play the same hero in the same match. The objective of the game is to destroy the opposing team's main building, called ancients, located on opposite sides of the map. In order to accomplish this objective, players must level up and gain gold, by killing neutral characters (creeps), scattered around the map, and opposing heroes. Levels increase the base hero attributes, while gold can be used to purchase items from a pool of 100+, that increases hero attributes as well give heroes additional abilities.

[1] Massive Online Battle Arenas. A genre of competitive team-based online multiplayer games with a top-down/isometric perspective where players control avatars to complete objectives. E.g. DOTA2, League of Legends, Heroes of Newerth.

This increase in hero power allows teams to complete sub-objectives and consistently defeat the opposing team's heroes in skirmishes (team fights), and eventually destroy the enemy ancient to win the game.

1.2 Quantifying DOTA 2

Economics of DOTA 2. While the gameplay of DOTA 2 can be flashy and action-packed, the game can be thought of mostly as a resource-management game. Creeps, towers, and opposing players can be thought of as resources that can be gathered to help a team gain an advantage over the opposing team. There is a strong correlation between a team's net worth advantage, which is determined by the difference between one team's net worth over the other, and winning a match.

$$Team\ net\ worth = \sum_{i=1}^{5}(Player_{i_{items}} + Player_{i_{gold}}) \quad (1)$$

Due to the limited resources on the map at any given time, team's must manage the available resources efficiently to gain the net worth advantage. One strategy is to split the resources evenly amongst team members, however, each individual hero may not accumulate items and levels fast enough to gain an advantage over the other team during team fights. Most teams tend to split resources by giving economic priorities amongst players, which lead to three main economic strategies:

- **Multi-core strategy:** Economic priority is given to 2 – 3 players.
- **4 Protect 1**: Economic priority is given to 1 player (carry) and the remaining players (supports) are focused on allowing that player to gain levels and gold at a rate where the opposing team finds it difficult to deal with that hero during team fights.
- **1.Carry ~ 5.Hard Support**: Economic priority depends on a player's role within the team. The 1 or carry player is focused on gaining levels and items to overwhelm the other team with individual net worth. The 5 (hard support) player is not focused on gaining resources, but instead uses the severely limited resource to purchase items and play in a manner that allows the rest of the team-members to do their roles safely. The 2 – 4 players adjust their items and playstyles according to the overall team strategy.

Because of these economic strategies, it is important to look at economic markers, such as gold and items to analyze playstyles.

Other data. The SteamAPI, provided by Valve Software, allows access to player and match data such as gold and items of every player in every match, which is useful to analyze the economics of every match. The SteamAPI also provides other statistics that can be useful as features to analyze such as: match duration, stuns, hero damage, tower damage, winning team, denies, assists, deaths, etc. These features may also be essential in determining optimal playstyles.

1.3 Current Beliefs.

The current popular belief for improving win rates is to play a core position with a high economic priority within a team. Essentially, if a player is able to gain an overwhelming economic advantage on his/her own, the player is less reliant on teammates' skill level to win a game. However, DOTA 2 is a team-oriented game, it's extremely difficult to win a match simply on skill level, team dynamics play an important part in winning games. There are also many players in higher skill brackets that exclusively play support or lower economic priority roles. Many players in higher brackets are able to play any role within a team and do not exclusively play a single role.

1.4 Existing work

There are DOTA2 websites that aggregate player and match data to provide statistics and analysis. However, these sites do not focus on the actions of an individual player statistically. Datdota.com uses data to analyze trends in the competitive scene[2]. Dotabuff.com contains player and match information but does not provide statistical analysis. The site mainly focuses on current meta-game trends (heros, items, strategies)[3], rather than individual players. OpenDota.com provides data and statistics on its site, along with an API to access the data, without providing any analysis[4].

There are machine-learning projects that have been performed using a dataset provided by Kaggle.com, but most are incomplete. Some projects of interest try to determine a match outcome given player history[5]. Another project attempts to predict the wining team during a match[6]. Most try to determine a match outcome based on a particular match attribute[7]. This means that determination of increased win rates based on individual player actions has not been previously explored or the research has not been found so far.

2 Approach

A dataset that contains a large number of DOTA 2 match and player data will be used to determine which attributes will be useful for playstyle analysis. Once the data has been preprocessed, various machine learning models will be trained using the data set. Model selection is dependent on its ability to rank the importance of certain features over others. This approach was used over other feature selections methods because the relative weights and order of the value provides additional information that is lost when only feature selection is done. Different models may provide different rankings of the feature and it does not invalidate their rankings, instead some rankings may be more frequently ranked high compared to others. The following models were used in the research:

Linear Regression. Linear regression models are dependent on weights on each attribute to determine the best fit to the data (Ordinary Least Squares, Lasso, Ridge).

$$y(w, x) = w_0 + w_1 x_1 + \cdots + w_p x_p \quad (2)$$

Selected player attributes can be used as the x_i of the model with individual player win rates as y in the linear regression model. The relative values of each weight can be used to rank attribute importance to determine win rate.

Regularization. Linear Regression models can be sensitive to noise and may overfit data. In order to minimize the effects of noisy data, regularization terms are used to minimize weight coefficients. Two regularization methods were used in this project L1 or lasso and L2 or Ridge. Lasso uses absolute deviation to minimize weight coefficients, which has the added benefit of feature reduction, since coefficient values can be 0. Ridge regression uses squared deviations which constrains the weights further leading to smaller coefficients.

Classification. Players can be labeled into two labels.: players that have high win rates and those that do not. Valve is able to successfully determine a player's skill using an ELO[2] based matchmake ranking. Because of this, the average player win rate is about 50%. Some players may have higher than average win rates, labeling these players will help with classification techniques.

Logistic Regression. Logistic Regression is similar to Linear Regression techniques but probability outcomes are determined using a *logistic* function. The sigmoid shape of the logistic function is appropriate to do binary classification. Similarly, L1 and L2 regularization can be applied to minimize coefficient values and reduce the effects of noisy data.

Decision Tree and Random Forest. Decision Trees classify data by splitting the data on certain attributes to maximize information gain on each split. Therefore, the earlier the split, the more important the corresponding feature.

Random forest classifier uses a number of decision trees with randomized parameters on smaller sub-samples of the training dataset to improve accuracy and control overfitting. Because Random Forest classifiers are just a collection of random decision trees, feature ranking can still be determined based on tree splits.

Support Vector Machine. Support Vector Machines (SVM) determines a decision boundary (hyperplane), which has a similar form to linear regression equation but optimizes the model by maximizing the margins between the data points to the hyperplane. The coefficients of the hyperplane created by the SVM optimization allows corresponding features to be ranked.

---

[2] A method of calculating relative skill levels of player in competitor-versus-competitor games. Elo is named after its created Arpad Elo and was originally used in professional chess[8].

Random Feature Subset. Random combinations of feature subsets of various size are chosen from all of the features in the dataset. A classifier is then trained on the reduced training dataset and a score is determined based on the classifier's performance on the reduced testing dataset. The best features are those that are contained in the combination of feature subsets that generated the best classifier model score. The downside to this approach is that features are simply selected and not ranked.

Results Aggregation. Due to the high variance of the data, the feature rankings between each model used may differ. But certain attributes may always be consistently at the top of the rankings. A voting system is used on each model's feature ranking to determine which features are most important in determining a higher win rates. The final ranking is then used to determine the optimal playstyle or features a player can focus on to increase win rate.

3 Methods

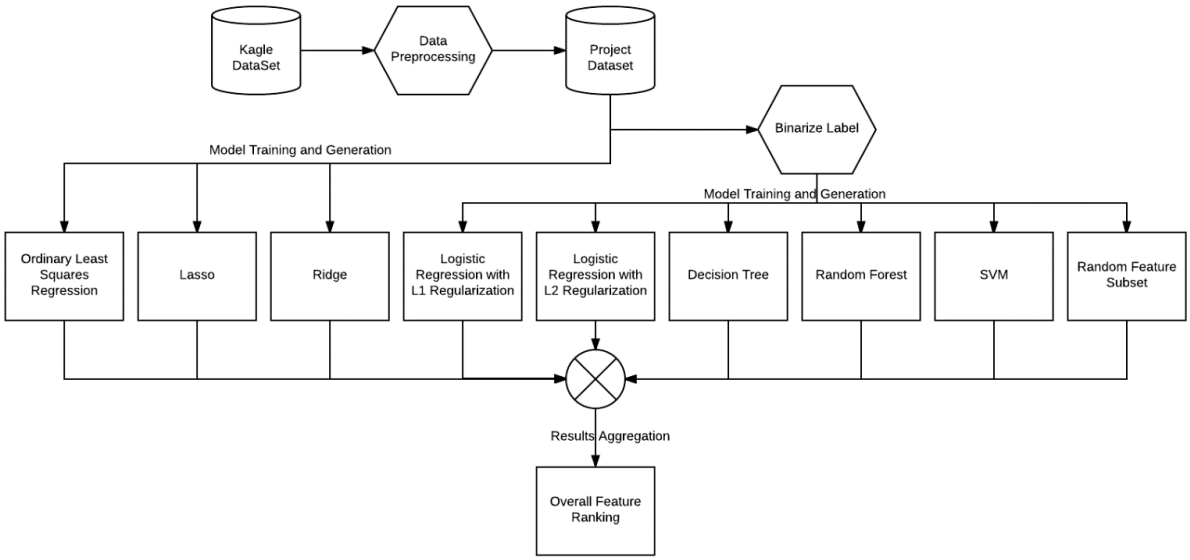

Fig. 1. Overall process flow

The dataset for this project was provided by a user named Devin through Kaggle[3][6]. The dataset contains player and match data from 50,000 ranked matches in Dota2. It is

[3] A website that provides resources for competitive data science and machine learning. Kaggle hosts user-provided datasets, kernels, and hosts competitions.

a smaller subset of a data-dump, given by Opendota, that contains 3.5 million matches from January 2015 to December 2015 inside of relational records organized into 19 different tables. Appendix A contains all the tables and their description. While the data is over 2 years old, the attributes of interest is patch[4] independent, and the game has not changed too significantly since the original data dump. SciKit-Learn[5] was used in a Python Anaconda environment. Code was written using Python IDE Spyder3. Figure 1 shows the overall process flow for determining best features in the dataset.

3.1 Data Preprocessing

Out of the 19 provided tables from the Kaggle Dataset, only match data and player data within those matches is of interest; this is given by the tables match and players. Unfortunately, the match_outcome and player_ratings tables could not be used in this project because they use separate match data. The match table is a 50,000 x 13 table, each row representing a single match. Players is a 500,000 x 73 table, where each row represents player data in a single match. Due to the sheer size of these tables, a significant effort for preprocessing is required. The following sections describe the preprocessing efforts, the steps do not necessarily flow from one to the next, the sections merely contain the efforts made to transform the original dataset to a more useful one. Appendix B contains sample dataset used to train and generate the models.

Feature Reduction. Both the players table and the match tables contain data that is much too detailed for the purposes of this project. Many of the attributes were deleted from these tables using heuristic means.

Some of the attributes were aggregated into a single feature, such as combining gold and gold spent into a single attribute called gold. This attribute describes the total economy gathered by the player in a single match.

Data Aggregation and Selection. In addition to reducing the number of features of the dataset, some useful features were added to generate better models.

Table Merger. The players table and the match table contain a match_id attribute. In the players table, this id describes the match that the individual player data was recorded from; there should be 10 player data points for every one of the 50,000 match data points. The data points from the match table was joined into the player table using match_id.

Unusable player data. About 1/3 of the player data was not usable for this project. This is due to the SteamAPI assigning an account_id of 0 to any account that is private

[4] Valve releases software patches mainly to balance advantages exploited in both the professional and casual scene. The dataset covers matches from patch 6.83 – 6.87. As of 10/29/2017, the game is using patch 7.06.

[5] A machine learning library built on top of NumPy, SciPy, and Matplotlib used in Python environment. http://scikit-learn.org/stable/#

(account-holder not willing to share account information). While their player and match data are given, it is not possible to identify that player with a specific account_id. For this reason, any player data point that has an account_id of 0 was discarded.

Player data aggregation. Since many players played more than a single match through data collection period, many of the player rows can be grouped by its account_id. Using this grouping, the player data can be aggregated so that each player data point is associated to a unique account_id. Most of the attributes were aggregated by calculating the mean of that attribute.

Player selection. Many of the players have played less than 5 games during the data collection period. This is not a statistically significant amount of games to determine an accurate win rate. Only players who have played a significant amount of games during the period, are chosen. The minimum number of games was heuristically chosen as 20 games.

Normalization. After the previous steps were completed. The dataset was normalized where appropriate by both value and by time.

It is important to normalize the data by time because the duration of matches for DOTA 2 games vary greatly. Players in matches that last 60+ minutes will accrue more gold, items, experience, etc. than players that complete their match under 20 min. Many attributes were divided by the duration of the match to express their value as per minute to eliminate the match duration factor.

Because feature ranking is determined by the weights of the model generated, the attribute values are normalized based on the highest value of the feature in the dataset. If this is not done, values such as duration and gold per min, which have higher values on average, will have larger weights on average compared to other features no matter their importance in determining win rate.

The following table contains all features used in the final dataset.

Table 1. Feature names and their descriptions

Feature	Description	Feature	Description
gpm	*Gold per minute*	spm	*Stuns per minute*
xpm	*Experience per minute*	hdpm	*Hero damage per minute*
kpm	*Kills per minute*	tdpm	*Tower damage per minute*
dpm	*Deaths per minute*	lpm	*Levels per minute*
apm	*Assists per minute*	average team pos	*Average net worth ranking within team.*
denpm	*Denies per minute*	duration	*Match duration (min)*
lhpm	*Last hits per minute*		

3.2 Model Training and Generation

K-Fold Cross Validation[6]. In order to avoid overfitting, a 10-fold cross validation strategy was used to train and generate the models.

Model Scoring. In most cases, the model with the best score prediction score (R^2 for regression and F-Score for Classification models), was used to determine feature rankings. Both R^2 and F-Score were chosen because the values obtained by these metrics fall between the range of 0 and 1. Metrics such as Mean Squared Error, can vary greatly from model to model depending on the size of the error values. F-Score was used over accuracy because it considers precision and recall and is not easily swayed by imbalanced datasets.

Linear Regression. Linear regression models were trained and generated using SciKit-Learn's linear_model library. The three classes that were used were LinearRegression[7], Ridge[8], and LassoCV[9] to generate Ordinary Least Squares, Ridge, and Lasso regression models respectively. Default parameters were used in the training and generation of the regression models.

Classification. The win-rate attribute was binarized by labeling players with win-rates greater than or equal to 55% with a value of 1, with players that have win-rates below 50% win-rates with a value of 0. 55% was heuristically chosen to determine which players have higher than average win rates and could be used to determine the best playstyle to achieve those win rates.

Logistic Regression. Two Logistic Regression models were trained to classify the dataset, one with L1 regularization and the other with L2 regularization. The SciKit-Learn class used to generate the models is LogisticRegression[10] with mostly default parameters except the penalty parameter which is set to 'l1' or 'l2' for the different modes of regularization.

---

[6] Data is partitioned into multiple folds, one partition is selected as the testing set, while the rest is used as training set. https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[7] SciKit-Learn Ordinary Least Squares Regression, http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

[8] SciKit-Learn Least Squares Regression with L2 regularization (Ridge). http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge

[9] SciKit-Learn Least Squares Regression with L1 regularization (Lasso). http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html

[10] SciKit-Learn Logistic Regression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Decision Tree and Random Forest. The SciKit-Learn DecisionTreeClassifier[11] class was used to train and generate the model. Default parameters were used to split on attributes. The DecisionTreeClassifier class uses Gini as its default split criteria.

The SciKit-Learn RandomForestClassifier[12] class with default parameters was used to generate the model. This classifier uses random decision trees on sub-samples of the training data. Samples are drawn with replacement.

1000 trials of model training and generation were performed on both classifiers, since model generation is not consistent for these types of classifiers.

Random Feature Subsets: Different size (k) subsets of total dataset features (N) were examined to determine the best score. The values examined for k ranged from 1 to N-1. Random combinations of size k were taken out of N. The dataset features were then reduced by selecting the features selected in the random combination. This reduced dataset was then used to train and generate a logistic regression model with L2 regularization. The feature combination that generated the best F-Score was saved for the particular k-size. 5000 trials were performed on each k-value to ensure that all possible combination of feature subsets were tried.

Once the best possible combinations of feature subsets and F-Score at each k-value was obtained, the best combination of features was determined by the highest score out of all k-values.

3.3 Result Aggregation

After weights have been determined, points are awarded to each feature based on the following formula.

$$P = \left((N + 1 - R) + \frac{\sum_{i=1}^{F} i}{N - F} \right) \cdot z \tag{3}$$

Where P is the amount of points awarded. N is the total amount of features. R is the ranking of the feature. z is the score of the model (R^2 for regression and F-Score for classification). F is the number of features that have been selected out by the model, or in other words, have non-zero feature weights.

While it would be easy to award points directly based on the reverse of the rankings, i.e. rank 1 gets 13 points and rank 13 gets 1 point, some techniques, such as random feature subsets and models that use L1 regularization can perform feature selection by setting coefficients to 0. Features that have been selected out will be awarded 0 points, with the remaining unused points redistributed to the remaining features. This points system is given by Equation 3.

Since the Random Feature Subset technique only selects the best features without ranking, the points formula is slightly altered to reflect the lack of ranking.

---

[11] SciKit-Learn Decision Tree Classifier, http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[12] SciKit-Learn Random Forest Clasifier, http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

$$P_{Unranked} = \left(\frac{\sum_{i=1}^{N} i}{N} + \frac{\sum_{i=1}^{N} i}{N} \cdot \frac{F}{N-F}\right) \cdot z \quad (4)$$

After points are rewarded for each feature in each model generation, the points were tallied and compared to determine the overall feature ranking.

4 Results

The following tables show the results of each model's feature points based on the feature ranking given by the model coefficients. The actual coefficients are given in Appendix C.

Table 2. Feature ranking points for all Linear Regression models used.

	Ordinary Least Squares		Ridge		Lasso	
Features	**R2**	0.622949351	**R2**	0.581992625	**R2**	0.623395618
gpm	8.098341569		6.983911504		8.376870312	
xpm	3.114746757		3.491955752		0	
kpm	6.229493515		5.237933628		6.506683457	
dpm	3.737696109		4.655941002		4.636496601	
apm	5.606544163		6.401918878		5.883287838	
denpm	0.622949351		2.327970501		2.142914128	
lhpm	1.868848054		1.745977876		3.389705365	
spm	1.245898703		1.163985251		0	
hdpm	7.475392218		4.073948377		7.753474693	
tdpm	4.983594812		5.819926253		5.25989222	
lpm	4.36064546		2.909963126		4.013100983	
average team pos	6.852442866		7.565904129		7.130079075	
duration	2.491797406		0.581992625		2.766309746	

Table 1 shows that *gpm* is consistently in the top of feature rankings for all regression models. However, in Ridge regression the top feature is given to average team position. While that feature is also considered important in both ordinary least squares and lasso, they were not the number one feature. The bar graphs for each model shows that Ordinary Least Squares regression and Lasso both have similar rankings of features with small differences in points for each feature. The top features for Ridge is mostly the same as the top features of the other two models, but the ranking of the features is much more different. The F-Scores for each regression is around the 0.6 mark, which suggests that the dataset has relatively high variance.

Table 3. Feature ranking points for Decision Tree and Random Forest classifiers

	Decision Tree		Random Forest	
Features	**F-Score**	0.705882353	**F-Score**	0.766666667
gpm	4.941176471		7.666666667	
xpm	2.823529412		6.133333333	
kpm	3.529411765		3.066666667	
dpm	7.058823529		6.9	
apm	9.176470588		9.966666667	
denpm	5.647058824		5.366666667	
lhpm	2.117647059		4.6	
spm	1.411764706		0.766666667	
hdpm	0.705882353		2.3	
tdpm	8.470588235		8.433333333	
lpm	4.235294118		3.833333333	
average team pos	7.764705882		9.2	
duration	6.352941176		1.533333333	

Both decision tree and random forest selected *apm* as the top feature. While both *tdpm* and *average team pos* were selected in the top three features, their ordering is different. The bar graphs for these two models appear to be mostly similar for the top features, but there are more differences in value for the middle and lower-ranked features.

Table 4. Feature points for Logistic Regression models, both L1 and L2 regularization

	Logistic Regression (L1)		Logistic Regression (L2)	
Features	**F-Score**	0.8	**F-Score**	0.666666667
gpm	10.48888889		6.666666667	
xpm	0		4	
kpm	8.088888889		5.333333333	
dpm	7.288888889		7.333333333	
apm	8.888888889		8.666666667	
denpm	5.688888889		4.666666667	
lhpm	0		2	
spm	0		1.333333333	
hdpm	6.488888889		2.666666667	
tdpm	9.688888889		8	
lpm	0		0.666666667	
average team pos	11.28888889		6	
duration	4.888888889		3.333333333	

Table 4 shows that the bar graphs for feature points vary greatly by using different regularization parameters. The top features have overlaps, but are not the same. Lasso

is able to do feature selection by optimizing weights so that some coefficients can be reduced to 0. Three out of the 4 lowest ranked features in the L2 regularized Logistic Regression model is selected out the L1 regularized model. The L1 regularized logistic regression model has a much higher F-Score *(0.8)* than the L2 regularized model *(0.67)*.

Table 5. Feature Points for Support Vector Machine

	SVM	
Features	**F-Score**	0.714285714
gpm	5.714285714	
xpm	5	
kpm	6.428571429	
dpm	7.142857143	
apm	7.857142857	
denpm	2.857142857	
lhpm	2.142857143	
spm	1.428571429	
hdpm	4.285714286	
tdpm	8.571428571	
lpm	3.571428571	
average team p	9.285714286	
duration	0.714285714	

The feature ranking from the Support Vector Machine places *average team pos* and *tdpm* as the most important features with *denpm*, *spm*, and *duration* as the lowest ranked features.

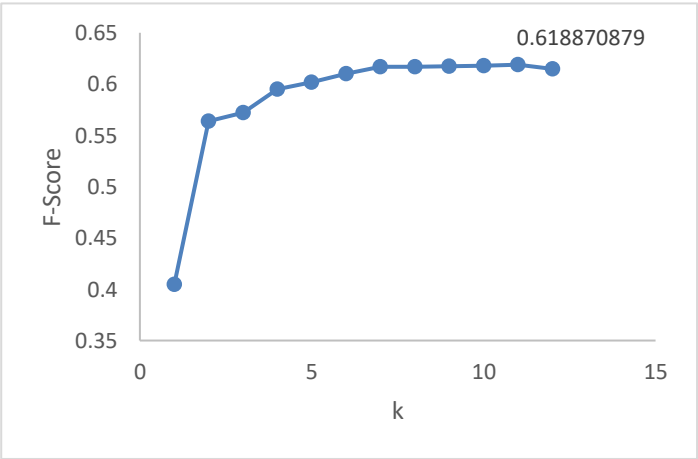

Fig. 2. Best F-Score of Logistic Regression Model with L2 Regularization at various feature subset size.

Table 6. Feature points for Random Feature Subset

	Random Feature Subset	
Features	**F-Score**	0.618870879
gpm	5.119749997	
xpm	5.119749997	
kpm	5.119749997	
dpm	5.119749997	
apm	5.119749997	
denpm	0	
lhpm	0	
spm	5.119749997	
hdpm	5.119749997	
tdpm	5.119749997	
lpm	5.119749997	
average team po	5.119749997	
duration	5.119749997	

Figure 2 shows that the F-score peaks when k is 11. The two features that were not selected were *denpm* and *lhpm* as shown in **Table 6**. However, the results from L1 regularized Logistic Regression, which is the other model that is able to remove**,** removes 4 features, of which *denpm* is removed for both Random Feature Subset and L1 Logistic Regression. The F-Score of the Logistic Regression model with L1 regularization (0.8) is significantly higher than the F-Score (0.62) of the Random Feature Subset technique.

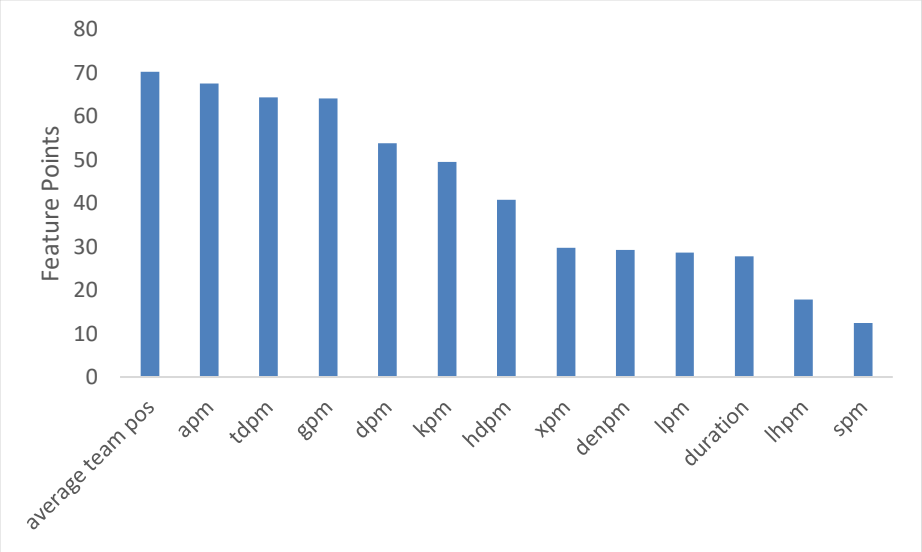

Fig. 3. Total attribute points awarded.

Figure 1 shows the overall features ranking after tallying all of the feature points. The chart shows clearly which 4 features were most important in determining win rates, or which features best classify players with higher than average win rates. The 4 features are *gpm, apm, tdpm,* and *average team pos*. The two lowest ranked features are *lhmp* and *spm*.

5 Analysis

The scores of the models are generally between .6 and .7, this is expected since DOTA 2 is an extremely complex game, and boiling down playstyle into 13 features is optimistic. There are of course different factors at play such as: player skill-level, motor-cortex response, player environment, network conditions, etc. that can impact win rates that are not included in the dataset. Additional actionable features should yield better scores.

One feature ranked amongst the highest is *gpm*. This suggests that players should focus on acquiring gold to translate into a net worth advantage over the other team. However, most teams that win matches tend to acquire more gold than the losing team because winning teams tend to complete more objectives such as destroying towers and buildings as well as killing more players. In other words, in the process of winning games, more gold is naturally accumulated over the losing team.

Average team position is a high ranked feature. The weights generated by relevant models are all positive values, which means that an increase in win rate correlates with an increase in average team position. This goes against the belief that high economic priority players are associated with higher win rates. Instead this suggests that support players have higher win rates. This again is also supported by the fact that *lhpm, xpm,* and *lpm*, features that are normally associated with high economic priority core players, are low in the feature rankings, and have mostly negative coefficients.

A*pm* is ranked more highly than *dpm*, this also does not support the belief that core players get higher win rates. Core players tend to get more kills in game, while support and offlane players usually use their skills and abilities to help core players get these kills during teamfights. Support and offlane players tend to have higher assists than kills.

What this data implies is that players that have higher win rates are objective players (high *tdpm* or tower damage per min), and are more focused towards enabling the team to play well. These features are characteristic to support or offlane players rather than core players.

6 Future Work

Using a larger dataset could strengthen or weaken the results of this research. The current dataset provided by Kaggle only contained 50,000 matches from a limited period of time. The SteamAPI is able to provide data on all matches that have been played since the start of DOTA 2. While this would require significantly more time and computing resources. The increase in data may change the results drastically or not at all.

The current assignment of points could be slightly tweaked to represent the strengths of each feature within a model. The current system assigns points solely based on ordering and does not take into account the relative values of the coefficients. Instead of ranking the features, the weights could be normalized to determine relative strengths. It would be interesting to see how the results will change based on this change.

In addition, adjusting model parameters and searching best parameters with grid search as well as adding more models to participate in feature ranking could be beneficial in strengthening the overall feature rankings.

Lastly, this feature ranking process could be used on other competitive team games or any competitive team sports where team rosters may vary from game to game to determine how to increase win rates by focusing on individual playstyle within a team.

7 Conclusion

DOTA 2 is one of the most complex, difficult games currently in the market. The complexity and difficulty, compounded by team dynamics can make the playing experience miserable if not outright hostile between players, all just to gain victory in a match. These conditions often lead many players to forego relying on their teammates and play farming core roles to carry their teammates to victory. Using various models trained on player and match data obtained from Kaggle, features were ranked based on its importance on obtaining higher than average win rates. The results disprove the prevailing belief that playing cores allow a player to win more matches. The top-ranked features suggest that the lowly support player is best role a player should play to increase his/her win rate. DOTA 2 is a team-oriented game, a support player focuses on allowing the team to succeed from minute-to-minute in every match. Therefore, to win more games a player should focus on being a team-focused player for a team-game.

References

1. The Hardest Game We've Ever Played, http://www.ign.com/articles/2014/03/05/the-hardest-games-weve-ever-played, 2014/03/05
2. Datdota, https://www.datdota.com, last accessed 2017/10/28
3. Dotabuff, https://dotabuff.com, last accessed 2017/10/28
4. Opendota, https://www.opendota.com, last accessed 2017/10/28
5. Devin, Setting up a prediction problem, https://www.kaggle.com/devinanzelmo/setting-up-a-prediction-problem-dota-2, last accessed 2017/10/29
6. Fleite, Predicting win probability during match, https://www.kaggle.com/felipempleite/predicting-win-probability-during-match, last accessed 2017/10/29
7. Dota 2 Matches Kernels, https://www.kaggle.com/devinanzelmo/dota-2-matches/kernels, last accessed 2017/10/29
8. Glickman, A Comprehensive Guide To Chess Ratings, http://www.glicko.net/research/acjpaper.pdf, last accessed 2017/10/29
9. Dota 2 Matches: Explore player behavior and predict match outcomes, https://www.kaggle.com/devinanzelmo/dota-2-matches, last accessed 2017/10/29

Appendix A: Kaggle Dataset

Table A. Dataset tables and their description given by Devin in Kaggle[8]. Not all of the tables use the same match data.

Name	*Description*
Ability_ids	Hero ability ids and names
Ability_upgrades	Ability upgrades and upgrade times for the original 50K matches
Chat	Chat log for 50,000 matches, includes player names, and slot variable.
Cluster_regions	Links cluster to geographical regions
Hero_names	Hero_ids and their names
Item_ids	Item ids and item names
Match	Top level information about each match
Match_outcomes	900K matches with outcomes
Objectives	Objectives and completion times
Patch_dates	The update dates for different patches
Player_ratings	Wins, matches, TrueSkill rating computed from 900K matches
Player_time	Summary of player actions at 60 second intervals of a match
Players	Aggregate statistics for player performance during each match
Purchase_log	Item purchase times for a player during each match
Teamfights	Start and stop time for teamfights
Teamfights_players	Specific information for each player in each team_fight
Test_labels	100K match_ids with labels and outcomes.
Test_player	Player information for test matches
Yasp_sample.json	Contains 100 matches in original JSON format

Appendix B: Sample Data

Table B. First 3 rows of processed dataset. Table is split to accommodate page, use id value as index.

Id	gpm	xpm	kpm	dpm	Apm	denpm	lhpm
68	0.1719	0.249411	0.157161	0.430647	0.471468	0.109645	0.289
69	0.297502	0.362806	0.266357	0.332509	0.527692	0.272818	0.458308
70	0.237319	0.226633	0.030178	0.313566	0.241845	0.210796	0.248866

Id	spm	hdpm	tdpm	lpm	average team pos	duration
68	0.289	0.209245	0.141807	0.241402	0.733432	0.588516
69	0.458308	0.288025	0.123803	0.332521	0.493293	0.694079
70	0.248866	0.135633	0.081181	0.273028	0.723472	0.452722

Appendix C: Model Weights/Coefficients

Table C. Weight values of each model used. The Random Feature Subset technique only selects important features and does not provide weight coefficients. A value of 1 indicates that the feature is included, 0 indicates that the feature has been removed.

	Ordinary Least Squares	Ridge	Lasso
gpm	0.649697614	0.332313494	0.635212
xpm	-0.088893552	0.116706346	0
kpm	0.474178747	0.195071224	0.451762
dpm	-0.168310413	-0.16361596	-0.16837
apm	0.282595869	0.258069565	0.280892
denpm	-0.024097906	-0.03313924	-0.02343
lhpm	-0.030178398	-0.022500003	-0.0577
spm	-0.030178398	-0.022500003	0
hdpm	-0.551697654	-0.119173925	-0.52489
tdpm	0.23686259	0.23494729	0.237935
lpm	0.227028969	0.063117008	0.135238
average team pos	0.514944583	0.377928407	0.507617
duration	0.081302089	0.011869383	0.051372

	Logistic Regression (l1)	Logistic Regression (l2)	Decision Tree	Random Forest	SVM	Random Feature Subset
gpm	5.994283934	2.132655394	0.070995	0.087252359	2.355281109	1
xpm	0	0.882816025	0.043667	0.082132316	1.463680285	1
kpm	3.68807639	1.558720597	0.054629	0.060810568	2.569635913	1
dpm	-3.349258587	-2.365780663	0.104162	0.08244098	-2.596113396	1
apm	4.704606768	3.658758327	0.166365	0.138264237	3.725334372	1
denpm	-0.876255932	-0.983430851	0.075123	0.074988114	-0.331898358	0
lhpm	0	-0.225843487	0.037725	0.069690808	-0.296062963	0
spm	0	-0.225843487	0.031523	0.046317509	-0.296062963	1
hdpm	-1.661988296	-0.54750995	0.023113	0.054589788	-1.348420173	1
tdpm	5.699766563	2.588307409	0.134313	0.092622474	3.80474693	1
lpm	0	-0.105882559	0.069481	0.061997718	0.677807277	1
average team pos	6.381111564	1.859850117	0.113373	0.096579563	4.084970268	1
duration	0.33090586	-0.578433664	0.075531	0.052313566	0.130492296	1