Match outcome prediction in SMITE

Erik Rakuščček
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
Email: er8261@student.uni-lj.si

***Abstract*—In this paper, we present an outcome predictor for a popular MOBA game SMITE. We implement two different classification-based predictors using random forest and SVM, as well as a logistic regression-based predictor. The prediction models are based on the character selection phase from the beginning of a match in SMITE. We collect match data from the official API and filter out redundant information and incomplete matches. Filtering out inexperienced players and matches where at least one player was absent on either team allowed us to more accurately model character influence on the match outcome. We tested different combinations of features and managed to achieve accuracy of 76.8% prediction accuracy on train data and 76.1% on test data by selecting only important features from a huge dataset provided by Hi-Rez Studios. We are the first to create a working win prediction model for SMITE.**

***Index Terms*—Logistic regression, random forest, SVM, predictive model**

I. Introduction

Multiplayer computer games, where two teams fight between themselves for a win, have been a very popular type of computer games in the last decade. Especially famous are games of type MOBA - multiplayer online battle arena, which attract millions of new players every month. The most famous MOBA games are DotA 2 [1], League of Legends [2] and SMITE [3]. Most of the research into win prediction and character selection recommendation went into the former two games. That is why we will focus on predicting the winning team for matches played in SMITE.

SMITE is an online multiplayer game created by Hi-Rez Studios. The game has a mythological theme, meaning players play as different gods from several different mythologies. As reported by Hi-Rez Studios, the game had more than 40 million users in April 2020. The game is played on multiple platforms (PC, Xbox, Playstation and Switch) with support for crossplay, meaning users from different platforms can play together. Only on Steam, which is one of the biggest gaming platforms on PC, SMITE has around 14.000 concurrent active players [4].

A SMITE match can be played on multiple different maps, each with its own rules and strategies. We will focus on the most popular map Conquest, which is also played on SMITE esports championships. A match on this map consists of a fight between two teams of five players. Both sides attempt to destroy the enemy team's titan. The map consists of 3 lanes and the jungle between them. Each player plays a single character for the duration of the match. Characters have 5 unique abilities that scale and upgrade as the player gains experience and levels up during the match. Players also gather gold during the match, which enables them to buy items and upgrade their character. There are hundreds of different items that give different benefits to the player's character.

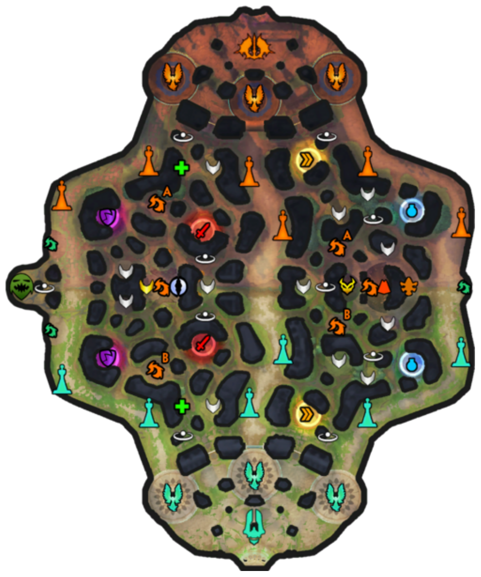


Fig. 1. SMITE Conquest map.

There are currently 118 different characters in SMITE to choose from. Each character can be picked only once during the selection phase of the match by either team. The correct choice of heroes plays an important role in the match's outcome. SMITE uses 10 ban draft pick system in Conquest, meaning each side gets to ban 5 characters that cannot be played during the match. The character selection is done alternating between the players of both teams. Every character has different abilities, strengths and weaknesses. That creates synergies between certain characters when playing on the same team, as one character complements the abilities of another. On the other hand, it also makes certain characters counter characters on the opposing team very well. There are 5 roles in SMITE:

- Jungler - tries to get ahead of the enemy jungler while ganking, defending and pushing lanes when needed
- Attack damage carry (ADC) - tries to get as much experience and gold as possible for late game, he is the damage of the team

- Support - protects the ADC at all costs and helps in the middle lane
- Solo laner - stays in his lane as long as possible, solely defends his lane
- Mid laner - fights in the middle lane

Different characters fit into different roles, where players must try to pick characters that counter the enemy characters while complementing their team's characters, to increase their winning chance. A team missing any of the above roles will struggle to keep up with the enemy team in some important aspects of the match [5].

Different character synergies and counter can cause a new player to struggle when selecting the character to play. That is why we can see 2 possible machine learning problems related to SMITE match. The first is a recommendation system for character selection, where the model would recommend which character to pick based on the win probabilities of different team compositions. The other is win prediction, where the model would predict the winning chance and the winning team. Since the former is dependant on the latter, we will focus on predicting the win chance in this paper to establish the foundation for future work.

II. Related work

There has already been done a lot of research in this field. Many different approaches have been tested including logistic regression, deep neural networks, SVM... However, most of the related works have been done for League of Legends and DotA 2, while there are no viable SMITE-related models for win rate prediction yet.

Authors of [6] were the first to use neural networks to predict the winning team in DotA 2. They used data about character picks and the duration of the match. Their prediction accuracy did not beat that of similar models that use logistic regression to predict the outcome of the match. Additionally, they found out that adding a feature of match duration only slightly improved the accuracy of the model.

Authors of [7] implemented a recommendation system, which can be used to advise the player on which character to pick during the selection phase. The model optimizes its recommendations for the greatest win probability. Their research was also focused on DotA 2. They used a deep neural network and strategically chosen data. They filtered out matches of players with lower account level and matches where at least 1 player disconnected during the match. Their model achieved up to 89% accuracy. Out of 1000 matches they managed to correctly predict the winning team in 74.5% of cases. However, their testing method does not reflect a real-life scenario. They were calculating win prediction where one team is selecting characters using their recommendation system while the other is selecting characters randomly. The latter does not take into account the synergies and counter relations between characters, therefore giving the first team a head start.

Authors of [8] decided to approach the problem in a different way. Instead of focusing on character selection for win prediction, they focused on data about the state of the match for every minute during the match. They analyzed different recurrent neural networks and found out that a simple RNN returns the best results. They achieved 63.91% accuracy using the data from the first 5 minutes of the match, while they managed to predict the outcome of the match with 83.54% accuracy when predicting using the data taken from 20^{th} to 25^{th} minute of the match.

III. Dataset

We have collected 53,366 matches from the official Hi-Rez API over the period of 1/12/2021 to 15/12/2021. There is a lot of different data available for each match. However, we will focus only on a few that are important for our model. We collected the following data:

- Character played by each player
- Number of past wins for each player in this game mode
- Number of past losses for each player in this game mode
- Each player's Matchmaking Rating - MMR
- Winning team

We have also filtered out the matches that do not satisfy the following requirements:

- No disconnected or AFK (away from keyboard) players. We take into account only fair matches where each team has 5 players throughout the game.
- No inexperienced players. We only take into account matches where all players are above experience level 30. Level 30 is a good way to characterize a player as experienced since it proves they have at least 50 hours of play time behind them.
- The duration of the match is no less than 10 minutes, which filters out matches where one team would purposely lose quickly.

We randomly shuffled the data to hopefully avoid overfitting since data was ordered by the time of the match. Then we split the data into two groups in the following manner: 70% of matches for training and 30% for testing.

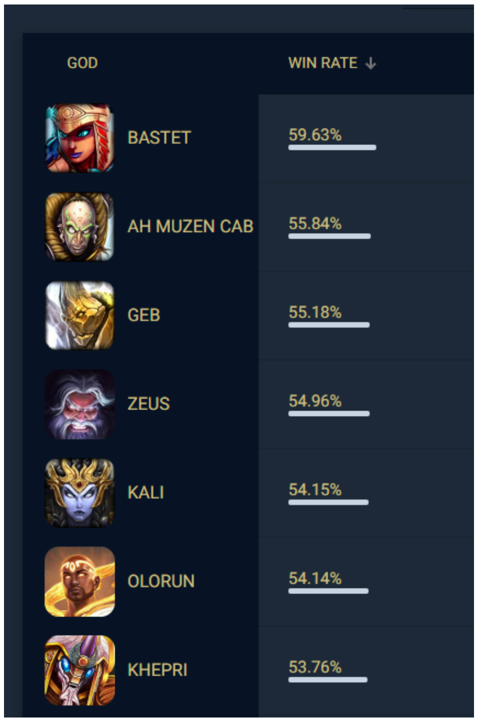
IV. Exploratory analysis

A. Character roles

At the time of writing, there are 118 different characters in SMITE. Each of the characters has different strengths and weaknesses. Characters are distributed into 5 different roles within the game: guardian, hunter, assassin, mage and warrior. The characters that belong to the same role have similar characteristics. For example, guardians generally have more health and damage protection while hunters deal more damage but have less defensive traits. It is important to note that these roles are different from the roles specific to Conquest game mode, which are: jungler, attack damage carry (ADC), support, solo laner and mid laner and are described in section I. While the in-game roles can be gathered from the Hi-Rez API, the Conquest roles are not so easily determined. There is a lot of strategy behind each conquest role when played correctly. Trying to fill the Conquest roles when choosing characters is the optimal way to increase your winning chance. Therefore we do not use the roles data in our model as they are not bullet-proof indicators of match outcome. We try to determine the success of certain combinations of characters by looking at their historical performance together.

B. Character statistics

We have access to many different data points provided by the Hi-Rez API. We are able to calculate past character-specific statistics for each of the characters. For instance, we can calculate characters' average kills per match, deaths per match, gold earned per match, towers destroyed per match, etc. These statistics have already been calculated on Casual Smite [9] however we found out that they are not key factors in predicting the winning team. The reason for this is that every character has a different playing strategy. While a character has on average scored fewer kills in a match does not necessarily mean they have a worse chance of winning. Especially so, if they play the role of support whose main objective is not to get kills but support others on their team.



GOD	WIN RATE ↓
BASTET	59.63%
AH MUZEN CAB	55.84%
GEB	55.18%
ZEUS	54.96%
KALI	54.15%
OLORUN	54.14%
KHEPRI	53.76%

Fig. 2. Best characters in SMITE by win rate [9]

C. Matchmaking Rating - MMR

Matchmaking is a term used to describe how to determine which players are going to be paired up to fight in a match. The matchmaking system tries to ensure each team has an even chance of winning any given match. It does this by pairing similarly skilled players with and against each other. The system used in SMITE is a modified version of the TrueSkill System [10]. The system tracks how players perform and assigns a Matchmaking Rating and a Variance score to the player. The Matchmaking Rating (also called MMR) is how skilled the system thinks a player is. The Variance value is how confident the system is that the assigned MMR is accurate.

Given that MMR shows us the technical skill of the player, we make a hypothesis that a better MMR means a better win chance. Therefore we try to use MMR as a feature to train our model.

V. Feature selection

A. Character selection

We represent selected characters via binary features corresponding to which characters are on team 1 and team 2 as follows:

$$X_i = \begin{cases} 1 & \text{if character } i \text{ is on team 1} \\ 0 & \text{otherwise} \end{cases}$$

$$X_{118+i} = \begin{cases} 1 & \text{if character } i \text{ is on team 2} \\ 0 & \text{otherwise} \end{cases}$$

These features should enable the model to consider the individual impact of selected characters on the match outcome.

B. Characters synergy

We implement a single feature to represent the combined synergies of both teams.

For calculating the synergies between the characters we need to know their past performance when playing together. Therefore we create a *good with* matrix, which consists of win rates when characters are played together. There are noticeable horizontal and vertical lines colored dark blue in Figure 3, meaning the win rate is 0%. Since this is almost impossible with the amount of data we are working with, we checked in the game and found out that the character with ID 20 was banned from ranked matches during the time interval we were collecting data from.

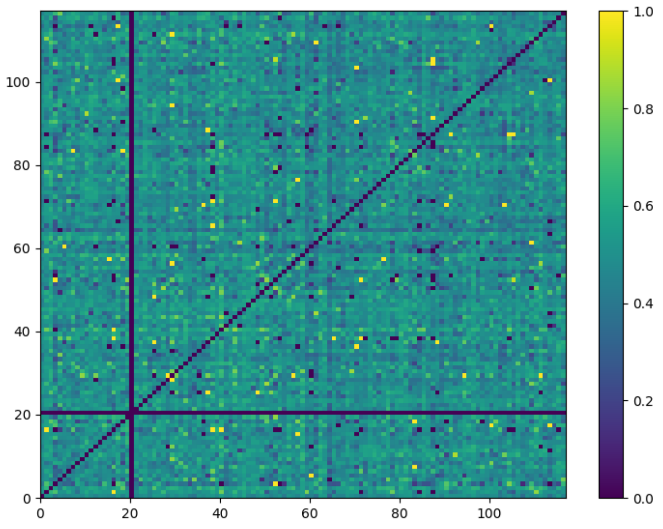


Fig. 3. Good with heat map.

Once we have the good with matrix defined, we calculate the feature. Let *T1* represent characters on team 1, *T2* represent characters on team 2 and *S* represent synergy. We define S_{ij} as win rate when characters i and j are playing on the same team. Synergy for team 1 can be calculated as follows:

$$S_{T1} = \sum_{i \in T1} \sum_{j \in T1, i \neq j} S_{ij}$$

We can calculate synergy for team 2 in the same manner:

$$S_{T2} = \sum_{i \in T2} \sum_{j \in T2, i \neq j} S_{ij}$$

We then construct a feature that represents the difference between the synergy of team 1 versus the synergy of team 2. We rescale it to represent a number between 0 and 100. The closer it is to 100 the more synergy advantage there is in team 2 and vice versa.

$$X_{236} = \frac{S_{T1}}{S_{T1} + S_{T2}}$$

We expect that this way of calculating the synergy feature will also capture role distribution within the teams. Generally, support and ADC roles will have much better synergy than 2 support characters, which should be represented with the synergy feature. By using this feature, well-thought-out picks should be inherently rewarded, while poorly chosen characters should be punished.

C. *Characters counters*

On the opposite side of synergies, we have counters. Some characters are built to counter others very well, which we also want to include in our features. Firstly, we need a *good against* matrix to be able to calculate the counter feature. We iterate over the matches and calculate the win rate when a certain character played against another character in a similar way as we calculated the *good with* matrix.

Let C represent countering and C_{ij} represent the win rate for character i when character j was playing on the opposite team and n represent the number of possible counters. We define character countering of team 1 over team 2 as

$$C_{T1} = \frac{\sum_{i \in T1} \sum_{j \in T2} C_{ij}}{n}$$

Counter for team 2 C_{T2} is unnecessary to calculate since this information would be redundant because $C_{ji} = 1 - C_{ij}$. We use C_{T1} as our feature.

$$X_{237} = C_{T1}$$

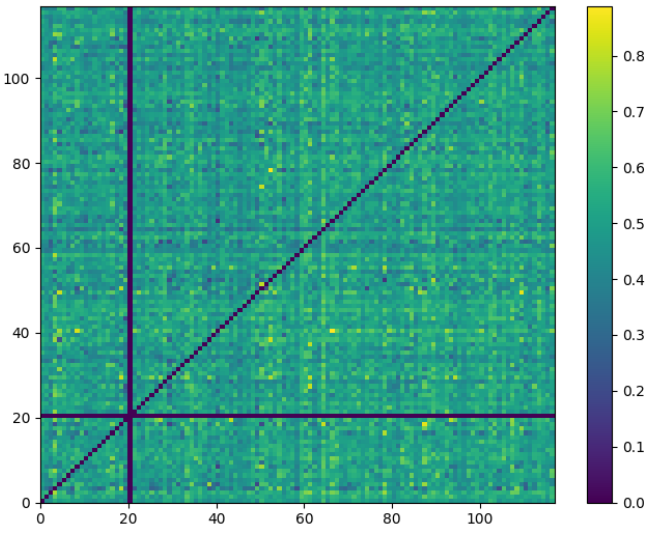


Fig. 4. Good against heat map.

D. *Player's win rate history*

Additionally, we added a feature that considers players' past matches. We gathered data about the number of wins and losses in Conquest for each player on each team and calculated the win rate. We sum up the win rates for each team individually and compare which team has a greater sum. Let W_i be the historical win rate for player i where player i is on team 1 and W_j be the historical win rate for player j where player j is on team 2. We calculate the feature as follows:

$$W_{T1} = \sum_{i \in T1} W_i$$

$$W_{T2} = \sum_{j \in T2} W_j$$

$$W = \begin{cases} 1 & \text{if } W_{T1} > W_{T2} \\ 0 & \text{otherwise} \end{cases}$$

We add this feature as X_{238} and hope it will help push prediction in favor of the more experienced team that has a proven track record of wins.

E. *Player's MMR*

We also add a feature for players' Matchmaking Rating - MMR. MMR is often used by the game engine as a way to classify players as experienced and skilled. Players with high MMR will have a track record of very good performances in past matches. Therefore we hope to improve our model's accuracy by adding this feature. This feature is calculated similarly to the past win rate feature. Let M_i be the MMR of player i where player i is on team 1 and M_j be the MMR of player j where player j is on team 2. We calculate the feature as follows:

$$M_{T1} = \sum_{i \in T1} M_i$$

$$M_{T2} = \sum_{j \in T2} M_j$$

$$M = \begin{cases} 1 & \text{if } M_{T1} > M_{T2} \\ 0 & \text{otherwise} \end{cases}$$

This is our 239^{th} and final feature to train our model on.

VI. Model

We develop 3 models for win prediction, two of which are classificators that determine if a match is a win or lose and one is a regressor, which determines what is the win chance of a certain team. We have built all models using scikit-learn library [11] in Python.

A. *Classification*

We developed 2 different models for classification: SVM (support-vector machine) [12] and Random forest [13]. SVM is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new test data. Random forest is a supervised learning algorithm that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

The implementation of classifiers was trivial since we carefully preprocessed data in advance and are using the scikit-learn library which enables us to generate the model in a few lines of code. We used *linear* kernel in SVM and 1000 estimators in random forest classification.

B. *Regression*

We used logistic regression [14] to get the winning chance for a certain match. Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. The logistic model is used to model the probability of a certain team winning.

We used LogisticRegression class from scikit-learn with *lbfgs* [15] as the solver. We had some trouble trying since the algorithm did not converge at first. We fixed the problem by setting the maximum iterations to 2000.

VII. Results

Training accuracy and validation accuracy for logistic regression were almost identical: 76.8% and 76.1% respectively. This is likely due to the fact that SMITE matches are largely random in nature and the outcome depends on many seemingly unrelated factors, especially related to the players that are prone to human errors in decision-making. Logistic regression had no problems with overfitting while it was a huge issue with random forest algorithm. Initially, we were getting 90.4% accuracy on the training set, while getting only 58.9% on the test set using random forest. Tweaking several parameters we managed to get the accuracy to 62.6% which is still much lower than logistic regression. SVM algorithm proved more accurate than random forest with 70.1% accuracy on test set.

Selection	Synergy	Counter	Player	MMR	Accuracy
✓					58.9%
		✓			63.3%
	✓				63.8%
✓	✓				71.4%
	✓	✓			73.5%
✓	✓	✓			75.4%
✓	✓	✓	✓		75.8%
✓	✓	✓	✓	✓	76.1%

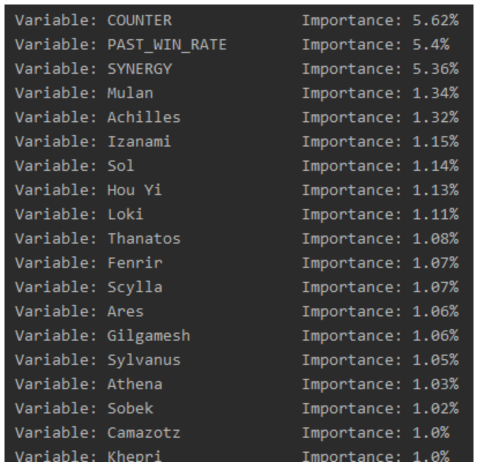
TABLE I
LOGISTIC REGRESSION ACCURACY BASED ON FEATURE SELECTION ON TEST SET

After significant experimentation, we determined that logistic regression was the most appropriate way to model our problem.

We tested different combinations of features to find out which features provide the best accuracy. The results are displayed in Table I. From this table, we can see that each feature improves the accuracy of the model. We can see that the character selection feature, synergy feature and counter feature all have a significant impact on the accuracy of the model. On the other hand, players' past win rate feature and MMR feature only slightly improve the accuracy and are not the best tellers of the match's outcome.

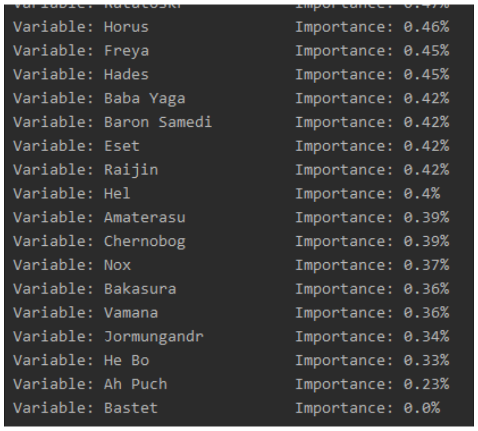
We use two baseline predictors for comparison. A random baseline predictor that chooses the winning team with 50% accuracy and a predictor which classifies the match outcome in favor of the team with a greater combined character win rate. The second predictor correctly predicts the winning team in 63% of cases.

Our final predictor which uses character selection, team synergy, character counters, past player performance and player MMR as features gives an accuracy of 76.1%. This is a significant improvement on both baseline predictors.



```
Variable: COUNTER            Importance: 5.62%
Variable: PAST_WIN_RATE      Importance: 5.4%
Variable: SYNERGY            Importance: 5.36%
Variable: Mulan              Importance: 1.34%
Variable: Achilles           Importance: 1.32%
Variable: Izanami            Importance: 1.15%
Variable: Sol                Importance: 1.14%
Variable: Hou Yi             Importance: 1.13%
Variable: Loki               Importance: 1.11%
Variable: Thanatos           Importance: 1.08%
Variable: Fenrir             Importance: 1.07%
Variable: Scylla             Importance: 1.07%
Variable: Ares               Importance: 1.06%
Variable: Gilgamesh          Importance: 1.06%
Variable: Sylvanus           Importance: 1.05%
Variable: Athena             Importance: 1.03%
Variable: Sobek              Importance: 1.02%
Variable: Camazotz           Importance: 1.0%
Variable: Khepri             Importance: 1.0%
```

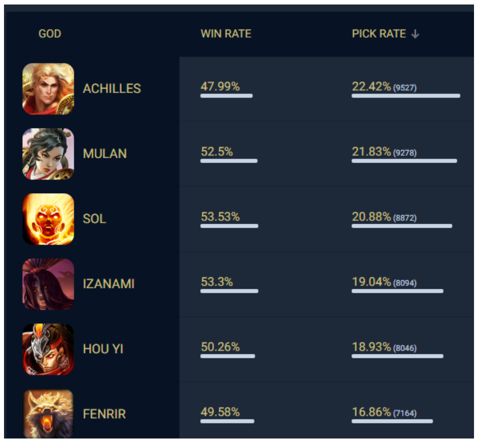
Fig. 5. Most important features in logistic regression.



```
Variable: Horus                     Importance: 0.46%
Variable: Freya                     Importance: 0.45%
Variable: Hades                     Importance: 0.45%
Variable: Baba Yaga                 Importance: 0.42%
Variable: Baron Samedi              Importance: 0.42%
Variable: Eset                      Importance: 0.42%
Variable: Raijin                    Importance: 0.42%
Variable: Hel                       Importance: 0.4%
Variable: Amaterasu                 Importance: 0.39%
Variable: Chernobog                 Importance: 0.39%
Variable: Nox                       Importance: 0.37%
Variable: Bakasura                  Importance: 0.36%
Variable: Vamana                    Importance: 0.36%
Variable: Jormungandr               Importance: 0.34%
Variable: He Bo                     Importance: 0.33%
Variable: Ah Puch                   Importance: 0.23%
Variable: Bastet                    Importance: 0.0%
```

Fig. 6. Least important features in logistic regression.

Looking at Figure 5 gives us an insight into which features are the most important when deciding the outcome of the match. The character selection is split between 236 features and is the most important technique for determining the outcome of the match. We can see that counter is the most important feature, closely followed by past player win rate and character synergy. Out of 118 characters in the game, Mulan, Achilles, Izanami and Sol are the most important characters when it comes to determining the outcome of the match. Meanwhile, Bastet, Ah Puch, He Bo and Jormungandr are the least important characters. Importance is closely related to the character's popularity (how many matches they were in) as all of the most important characters are also most popular choice in SMITE as can be seen in Figure 7.



GOD	WIN RATE	PICK RATE ↓
ACHILLES	47.99%	22.42% (9527)
MULAN	52.5%	21.83% (9278)
SOL	53.53%	20.88% (8872)
IZANAMI	53.3%	19.04% (8094)
HOU YI	50.26%	18.93% (8046)
FENRIR	49.58%	16.86% (7164)

Fig. 7. Most popular characters in SMITE [9]

VIII. Future work

Although we are satisfied with the results of our model there is still room for improvement. There is a lot of different data available about matches and characters in SMITE in Hi-Rez API. Therefore, more data could be included as features when building the model and try to further increase its accuracy.

Our prediction model should also be made available to the SMITE community for testing in real situations and to collect feedback from users. Another good use of our predictor would be to use it for a recommendation system for character selection. That would be especially useful for beginner players who often already struggle in this phase of the match.

Our predictor is also a good foundation on which to build predictors for other game modes in SMITE, which have different rules but are just as well dependant on character synergies and counters.

IX. Conclusion

Our initial task was to create an accurate prediction model to predict the winning team in a SMITE match. We have created 3 models and compared their accuracies. Two models were classificators, while one was a regressor. The random forest classificator proved similarly accurate to our baseline and also suffered from overfitting. The SVM classificator proved much more accurate and a better choice. However, we managed to get better accuracy using logistic regression than with both classification models and had no issues with overfitting. We conclude that logistic regression is the better option. We achieved 76.1% accuracy using logistic regression on the test set. We were using data from the beginning of the match, initially including only data about the character selection, but then extended the data by adding the player's past win rate and player's MMR.

Even though the accuracy may seem low, it is very comparable to the accuracies of models created in the related works. We must take into account the random nature of matches to see the full picture. We have explored the data provided by the Hi-Rez API and found out that SMITE matches are inherently tricky to predict. There are a lot of nuances that cause a match in SMITE to have a very unpredictable and in some cases almost random outcome.

Our predictor was not perfect since the outcome can not be bluntly decided at the beginning of the match (in the character selection phase). The match's outcome is heavily affected by the events that happen during the match. Therefore adding more data by tracking events during the match would greatly improve the accuracy of the predictor at a later stage of the match.

References

[1] V. Corporation, DOTA 2, https://www.dota2.com/home, [Online; accessed 28-November-2021] (2021).

[2] R. Games, League of Legends, https://www.leagueoflegends.com/, [Online; accessed 28-November-2021] (2021).

[3] H. R. Studios, SMITE, https://www.smitegame.com/, [Online; accessed 28-November-2021] (2021).

[4] activeplayer.io, League of Legends Live Player Count and Statistics, https://activeplayer.io/league-of-legends/, [Online; accessed 28-November-2021] (2021).

[5] SmiteFire, A beginners guide to the roles and etiquette of SMITE (conquest), https://www.smitefire.com/smite/guide/a-beginners-guide-to-the-roles-and-etiquette-of-smite-conquest-4401, [Online; accessed 8-December-2021] (2021).

[6] W. Wang, Predicting multiplayer online battle arena (moba) game outcome based on hero draft data, Ph.D. thesis, Dublin, National College of Ireland (2016).

[7] L. Hanke, L. Chaimowicz, A recommender system for hero line-ups in moba games, in: Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2017.

[8] A. L. C. Silva, G. L. Pappa, L. Chaimowicz, Continuous outcome prediction of league of legends competitive matches using recurrent neural networks, in: SBC-Proceedings of SBCGames, 2018, pp. 2179–2259.

[9] C. Smite, Casual Smite, https://ranked.casualsmite.com/, [Online; accessed 6-December-2021] (2021).

[10] R. Herbrich, T. Minka, T. Graepel, Trueskill™: A bayesian skill rating system, in: Proceedings of the 19th international conference on neural information processing systems, 2006, pp. 569–576.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[12] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[13] T. K. Ho, Random decision forests, in: Proceedings of 3rd international conference on document analysis and recognition, Vol. 1, IEEE, 1995, pp. 278–282.

[14] D. R. Cox, The regression analysis of binary sequences, Journal of the Royal Statistical Society: Series B (Methodological) 20 (2) (1958) 215–232.

[15] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming 45 (1) (1989) 503–528.