Research

Identifying player skill of dota 2 using machine learning pipeline

Methasit Pengmatchaya[1] · **Juggapong Natwichai**[1,2]

Received: 20 January 2024 / Accepted: 15 May 2024
Published online: 28 May 2024

Abstract
The esports industry is one of the prominent business sectors in the digital era, particularly, Multiplayer Online Battle Arena (MOBA) games which gain much attention from gamers and streaming audiences. Among such games, Defense of the Ancient 2 or Dota 2 is the record holder for the highest prize esports tournament. Therefore, various companies and investors start their esports teams to compete in the Dota 2 tournaments, the Internationals. To success in the competition, player recruitment is a crucial process as it usually takes considerable effort to find a skillful player. Watching the game replay to evaluate the player's skill is one of the approaches. However, it can be too exhaustive, also some player's ranking, which represent the player's skill, are often not available. In this paper, we propose an effective machine learning pipeline to evaluate the player's skill. Our designed pipeline includes data collection, feature engineering, and machine learning modeling. We show the data collection process using open-source API. An effective method for feature engineering is proposed. Features, e.g., end-game, or tactical decision related statistics, are incorporated along with the resource in the game distribution, harassment tactic, or spatiotemporal features, in order to provide effective models. Subsequently, we apply major machine learning models based on a single game data, i.e., logistic regression and random forest, to the processed data. The most effective model can achieve up to 0.7091 precision, 0.5850 recall, 0.6411 F1-score, and 0.8123 ROC AUC score.

Keywords Game Analytics · Machine Learning · Multiplayer online battle arena · Dota 2

1 Introduction

The esports industry is expanding continuously in the digital era. Not only the number of esports audiences is rising, but also the market revenue has dramatically increased and is likely to reach billions U.S. dollars in 2022 [25]. Moreover, the COVID-19 pandemic positively impacts the live streaming business, introducing several casual streaming viewers into games and esports [14]. These statistics point out the promising growth potential of the industry.

One of the most famous esports is Defense of the Ancients 2 or Dota 2, a multiplayer online battle arena game developed by Valve. Two teams of five players must select a character or "hero" for a competition, a match. Each team must destroy the opponent's base to win a game. Dota 2 is highly competitive as it demands both technical and soft skills from players, such as strategic thinking, analytical skills, and knowledge heroes. The game is very famous as the highest pool of the official esports tournament of Dota 2, the International, recorded the highest prize pool esports competition

Methasit Pengmatchaya and Juggapong Natwichai have contributed equally to this work.

✉ Juggapong Natwichai, juggapong.n@cmu.ac.th; Methasit Pengmatchaya, methasit_pe@cmu.ac.th | [1]Data Science Consortium, Chiang Mai University, Huay Kaew, Muang, Chiang Mai 50200, Thailand. [2]Department of Computer Engineering, Faculty of Engineering, Chiang Mai University, Huay Kaew, Muang, Chiang Mai 50200, Thailand.

at 34.33 million U.S. dollars in 2019 [11]. In the players' point of view, Dota 2 is also one of the top 10 highest-earning professional esports players [13]. Due to these causes, even big firms begin to invest in Dota 2. For example, Paris Saint-Germain, a French football club, and LGD Gaming, a Chinese esports organization, formed a Dota 2 esports team named PSG.LGD in 2018. The team was an immediate success as they won second place in the International 2018, they earned approximately 5.2 million U.S. dollars from competing in Dota 2 only in that year, 57.66% of the total prize earned from every esports that the team competed [12]. The case of PSG.LGD demonstrates the potential return on investment of the Dota 2 esports team.

In general, an effective recruitment system contributes to the success of any business, including esports, particularly, Dota 2 which has a considerable amount of players. To discover skillful players from the pool can be a challenging task. In order to evaluate players' skills, Dota 2 employs a rating system so called matchmaking rank (MMR). The higher MMR number means the higher player skills. However, Dota 2 does not show MMR publicly. The game uses "rank medals" to represent the interval of MMR instead [34]. Since there are numerous data of players available, players' skills estimation is a very complicated task.

To facilitate the analytics, Dota 2 records match replays, and the game developers provide an API to retrieve such data allowing the third parties to develop a service from such API, e.g., Opendota and Dotabuff. These services provide the data and also develop their own API for users, containing players' information and rank medals. However, one of the main issues is data completeness, i.e., some data are missing because of data collection process and data availability. The missing data may not be an obstacle for the competition regions with abundant data, but the regions with data scarcity. This can hugely affect the player recruitment process.

Machine learning is one of the important approaches to evaluate player skills. It is reported to be applied in many different sports, e.g., football [26], basketball [30], and golf [24]. Typically, machine learning approaches are great fits for problems with the higher amount of data. Since the esports, including Dota 2, can generate big amount of data, thus, there exist several attempts to apply machine learning to predict a player's future rank in the other MOBA games. However, there is no established technique to identify a player's present skill or rank in Dota 2.

In this paper, we propose an effective machine learning pipeline to evaluate the Dota 2 player skill. Firstly, we employ an existing open-source API and the automation tools for data collecting process. Subsequently, we propose a feature engineering process to represent tactics and skills in the game. The extracted features include a well known techniques from the literature, e.g. kills, dealths, or assists statistics, with our proposed work for further model improving. Features such as end-game, or tactical decision related statistics, are extracted to help building effective models. Further, the features are categorized into individual allies, enemies, team, and enemy's team performance, are also proposed. Finally, a few machine well known learning models are evaluated using five metrics: precision, recall, F1-score, ROC AUC, and the training time used. From the experiments, our proposed features can provide a very effective result comparing with the conventional features from the API. In addition, our proposed model can use only a single game match to predict a player's rank, not as high as ten matches as the previous work. This can help game analytic more effective when a large amount of player data is not available.

In summary, the following are our key contributions:

- We present a machine learning pipeline for Dota 2 player skill evaluation. Our work begins with data collection, feature engineering, until the classification task is performed.
- We propose a feature engineering which includes basic features, game-specific features, and features from tactical decision which can improve the analytical tasks.
- We evaluated the effectiveness of our proposed work based on game players' roles and winning condition on the selected six machine learning models, i.e., Logistic Regression, Ridge Regression, Support Vector Machine, Naive Bayes, Random Forest, and XGBoost.

This paper contains six main sections. The next section presents the summary of the related works including, data collection, data processing, machine learning models, and skill prediction in games. Section 3 shows the methodology of our works. Section 4 presents the result of the experiments, followed by the discussion in Sect. 5 Finally, Sect. 6 concludes our paper and provides the direction of future work.

2 Related work

In this section, the review of the related work of multi-player online battle arena games (MOBA) and their analytics are presented composed of the data collection, feature selection and processing, as well as the player skill prediction.

2.1 Data collection

Generally, most major MOBA games provide their official API for collecting data. For example, Riot Games, League of Legends (LoL) developer and publisher, provides match history, including in-game and players' basic statistics. They build an API for retrieving these data called Riot Games API. Dota 2 also has the same approach called WebAPI by Valve [36]. Such API allows developers to retrieve data, e.g. basic statistics. Also, Valve allows match replays available to be downloaded. Users, e.g., game analysts, are allowed to watch the replays via the game engine based on such data. Thus, the performance of players can be analyzed and the insights can be obtained from the plays. Earlier game analytic works mainly use WebAPI to retrieve data [9, 10, 16, 17, 28]

Alternatively, OpenDota API is an option to retrieve the game replay data for analytical purpose. Its advantage is that it can query data with more conditions, such as retrieving the matches with certain heroes [35]. The other benefit is the API yields replay's URL of a game in which WebAPI does not. Thus, the process to retrieve data is much less complicated. Therefore, later works started using OpenDota API [8, 35].

2.2 Feature engineering for dota 2

The features which can obtain from both both WebAPI and OpenData API are in JSON format. The data are numeric, e.g., end-game kills, deaths, or assists, are already in a well-defined format. So, these fundamental data required minimum effort to process. Most work further processed these features into per minute form as duration affects most numerical data [3, 9, 35]. For qualitative features such as item bought, or hero ability upgrades, some works applied one-hot encoding [35].

End-game and pre-game statistics are suitable for some tasks such as play-style classification [15], predicting the game outcome by hero lineups [16]. On the other hand, specific tasks such as role identification and outcome prediction need to analyze players' behavior. Such works required more effort in the data processing. Some of these statistics can be applied to our work as effectively, in which the explanation will be presented in Sect. 3.

Replay files, in.dem binary format, are the most relevant data source for the jobs as they contain the players' logs. The files need a parser to make it interpret-able. Recent works used Clarity [31] as a parser [4, 10, 17, 18, 35]. The parser typically extracts the match logs, such as player's position, damage from players to the others, or player's obtained gold and experience. These information is considered with regard to the time in the game. Eggert et al. showed an effective usage of replay mining [10]. In such work, role classification using a combination of end-game statistics and features mined from replays, for instance, team fight participation and early movement, is computed. In [9], it is found that the attributes are related to players' skills through replay analysis by analyzed players' spatiotemporal behavior partitioned by skills. It is also found that experienced teams usually change their play-zone in the early game more often and position themselves more compact than the amateurs.

2.3 Skill prediction

There is no known attempt to predict the skill of Dota 2 players using machine learning. However, there were several works concerning MMR prediction as follows. Min et al. applied a support vector machine and adaptive boosting technique to predict the change in MMR for each match [1]. From the report, adaptive boosting outperformed the support vector machine, producing a 3.6% error on average. However, they were not able to successfully identify players' skills. The best model generated a 28.8% error rate predicting the end-season ranking. On the other hand, Aung et al. presented a tier classification which was highly effective [1]. The players' first ten matches data were computed using logistic regression and random forest to identify the end-season ranking. Both models could score at the high result of ROC AUC at 0.923.

Even though both works are a good guideline for our work, they may not be a reasonable comparison. First, the latter included the mean MMR for the first ten matches as a feature, which could hugely influence the model. We intend to predict the unknown rank, not the future rank, thus, using MMR cannot be our option. Another reason is that, both studies were based on League of Legends, not Dota 2. The game mechanic is quite different from each other which leads to different tactics. For example, denying creeps is an essential play in Dota 2 as it cuts the enemy's gold and experience point significantly. League of Legends does not have this feature, nor other major similar features. Also, there are quite a number of differences in term of acquired abilities, skills to aim at the enemies, style of play required or etc, between the two games. Nevertheless, we applied their works for our features selection related to the skills.

Last, some orthogonal literature to our work, which is mainly focused onto predict the outcome of the MOBA games, is as follows. In [21], Johansson et al. applied Naive Bayes, Random forest, and Support vector machine to predict a winning team. They found that random forest was suitable classifier for the task. In [33], complex interaction between heroes are modeled and applied to improve the accuracy of the outcome prediction. Or, deep learning is proposed to predict the outcome, in this case, the death prediction of heroes, in [23]. The proposed method focuses on real-time prediction to enable in-game focusing by the audience or viewers. In [32], the systematic review and performance comparison of most frequently applied machine learning algorithms for prediction of the match winner from the teams' drafts in DotA 2 is presented. The most effective machine learning models for the important tasks of game analytics are identified. It is well noted that the factorization machine worked quite effective with a 0.71 ROC AUC score. In [17], the accuracy of using logistic regression and random forest as the outcome classifier was investigated. The logistic regression yielded a better result than random forest for professional matches, but the opposite went for the amateur datasets. The accuracy was 75% approximately.

3 Methods

This section presents the proposed method to build a machine learning pipeline for classifying Dota 2 player's skills. Our work are composed of three parts: data collection, feature engineering, and machine learning model.

3.1 Data collection

First, an approach to collect the data for our work is presented. Our method to collect the data is straightforward as only a single match is required to predict a player's skill as gathering multiple games of a player could be complicated. Also, using less data could be an advantage when developing the model in real-life. Nevertheless, the collected matches must satisfy the following conditions.

- The matches should contain ten human players with labeled rank medals.
- The matches should be within August 2020 on the patch 7.27c.
- The matches should be longer than fifteen minutes.
- The matches should be ranked-matches played in All drafts mode.

We use OpenDota API to query only "match_id" for the matches with such conditions from the OpenDota server. Then, the information for each match is acquired via the same API. The data are in JSON format, including replay URL, match statistics, and players' medals. After that, the replays are downloaded in full details. The replays are formatted in BZ2 files, which have to be decompressed into the DEM files which are the standard replay file for Valve's games, such as Counter Strikes - Global Offensive, Team Fortress 2, as well as Dota 2. This process can be done using Python's "requests" library to send HTTP requests to the API server.

After decompressing all the replay files, the OpenDota parser [6] is applied to parse the files Even though many works used Clarity for log extraction, the OpenDota parser can compute some relevant features which Clarity does not, such as lane efficiency. However, the parser is designated to run on Java Web Server. Therefore, we modify the parser so that it does not need to start an HTTP server, to lean the pipeline. In our work, the "Jpype" package, a Python package is used.

Once the parser outputs the match logs in JSON format, the data including the logs and other match statistics are transformed from the semi-structured to the structured and loaded into a relational database server, Microsoft SQL Server. Note that the corrupted replay are filtered out. Figure 1 shows the schema of the output database.

As a result, we are able to collect 13,416 matches from the data collection process, which means there are 134,160 observations or players as the dataset. From the observations, most players had a Legend rank medal, i.e., 30.08% of the

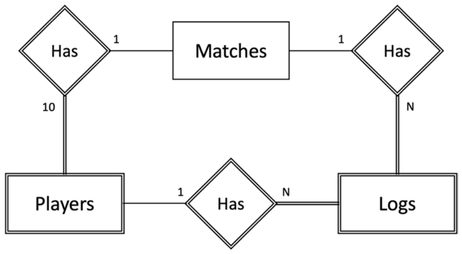


Fig. 1 Database schema

dataset (Fig. 2a). We consider that players who have a rank medal below or equal to Legend as unskillful players as in [22]. From our criteria, 38.66% of the observations are skillful, and the rest are the opposite (Fig. 2b).

3.2 Feature engineering

One of the main contribution of our work is feature engineering. Not only the feature engineering method is presented, but we also demonstrate how the Dota 2 tactical decisions is quantified into the numbers. Generally, the features described in the literature [8, 9] is applied, additionally the new features are introduced in this work,e.g., the harassment or survival efficiency.

3.2.1 Role identification

First, let us explain the concept of roles as it is one of the most important basic in the game. First, there are five positions in the game: Carry (Safelaner), Midlaner, Offlaner, Soft Support (Roamer), and Hard in Support. The carry player is the one who carries the team in the late game but is vulnerable in the early game. This position needs a lot of items. So, they must be positioned to the safest lane to earn more gold. Midlaner is the player who goes to the mid-lane, which is the lane that gives the player the most resource as the midlaner usually goes alone. This position is also known as semi-carry as the player must carry the game in the early game. They must help both other lanes, unlike safelaner who concentrates on farming until they get enough items. Offlaner is the one who takes the most challenging route as the environment of the off-lane makes the player hard to survive and farm the resources. This position will meet the opponent's safelaner. Their task is to make the lane hard for the enemy's safelaner farming as well as not feed the enemy. Therefore, offlaner should not need a lot of gold with high survivability. The two supports are the position that must not be greedy at all, especially the hard support. Their duties are to supply health and mana (magical power) regeneration to the core players, the three earlier mentioned positions. They must provide a vision to the team by warding and disrupt an enemy's sight. The soft support is usually roaming to help in every lane but tends to stay the most on the off-lane as such lane is highly critical. The hard support's most crucial task is to babysit the carry. They must do the creep stacking in the jungle, harassing the enemy to make space for the safelaner, and protect the safelaner as much as they can. Note that this position requires a minimal number of golds.

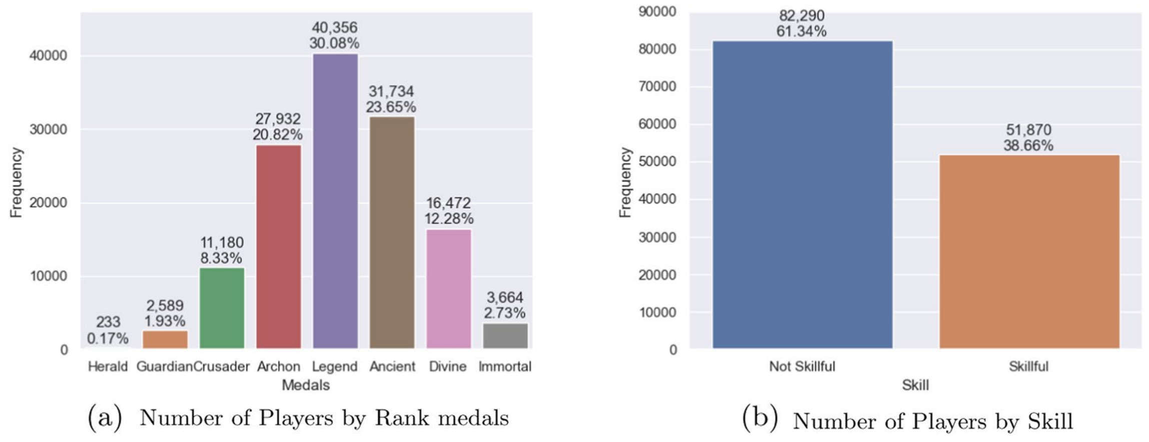


Fig. 2 Rank medal and skill of players. a) Number of players by rank medals, and b) Number of players by skill

Each role directs the way the players play, which affects their statistics. For instance, carry players should earn money and kill the enemies as much as possible, while the offlaner should harass the enemy's carry and make the space for the team. We classify the roles by a rule-based method:

1. We identify the lane to which the players belong and whether they roam. The proposed calculation from OpenDota is applied here.
2. We identify the core players. The core players are such player with the highest gold received for each lane who do not roam. We consider the players who do not qualify the conditions as the support players.
3. We determine soft and hard support. The player who is in the safelane and is not the core as hard support. The rest are soft support.

After the players are classify, we select only a match with unique roles as our dataset. For instance, the match which contains the team with two soft supports would not qualify for our dataset.

3.2.2 OpenDota statistics

Simple statistics provided by OpenDota could be a good expression of player's skill. Last-hitting and denying are the good examples. The team relies on the resource in Dota 2. The gold mostly comes from the enemy's creep and neutral creeps, which needs a player to blow a final hit to them or "last hit." The last-hitting is an important challenge for amateurs as the players need to practice their timing to execute properly. Also, cutting the enemy's resource by last-hitting the allies or "denying" is also essential.

OpenDota provides many statistics ranged from the most simple number such as kills, deaths, and assists to the very complex attributes. Some complex features may need further derive from the logs in which OpenDota provides the method to calculate such features.

We separate the features from OpenDota into four categories as shown in Table 1. The "basic" category is the retrievable statistics via API without calculation. The second category is "end-game statistics," the features found from the last interval (or second) of logs. The next category is death related features, including life duration and buyback count. The last one contains action count, town portal score purchased, teamfight participation and lane efficiency, which reflects the tactical decision of player.

3.2.3 Team synergy and counter-pick

Selecting an appropriate hero for each match indicates how much the player understands the game. In general, players consider two factors when choosing heroes. The first factor is synergy, or how well the selected hero works with a team. The other factor is counter-pick, which means selecting a hero that has an advantage over the enemy hero. We acquire the data from http://dotapicker.com. The website calculates the heroes' synergy and counter-pick in two ways: advantage score and wins rate score. The advantage score is the score calculated by the synergy of hero skills, and the wins rate score uses the hero combination's actual win rate calculated from the played matches. In our work, wins rate score is applied as the data indicated the meta at the time. A web scraping tool based on Selenium is developed to collect the data since the Dotapicker does not provide an API.

Table 1 Selected features from OpenDota

Category	Features
Basic (No calculation needed)	Kills, Deaths, Assists, Hero Healing
	Hero Damage, Gold, Experience, Last Hits, Denies
End-game Statistics	Camps stacked, Firstblood claimed, Rune pickups
	Observer ward placed, Sentry ward placed, Stuns
Killing Statistics	Neutral creeps, Enemy courier, Enemy creeps
	Ancient creeps, Observer wards, Sentry wards
Deaths Related Statistics	Life duration, Buyback Count
Tactical Decision Related	Actions, Town portal score purchased
	Teamfight Participation, Lane efficiency

3.2.4 Resource distribution

Generally, many amateur players do not have an awareness of resource distribution. The position order, one to five, is the order of their gold dependencies. Position one, which is the carry, depends on the item to carry the team. Therefore, the team should let position-one player farms the most, and should attempt to help them farm. In contrast, the position-five players, or hard support, should focus less on the gold than the carry. So, they should help position-one player farms. These are the reason position-one and position-five players frequently go to safelane, the most accessible lane to farm, together. The same goes for position-three and position-four in the offlane. Here, the amateur players can make a simple mistake. For example, hard support constantly last-hit the enemy's creeps while the carry is in the lane. We exposed this lack of understanding of the game by calculating resource distribution.

$$r_i = \frac{g_i}{\sum_{j=1}^{5} g_j} \tag{1}$$

Equation 1 shows the calculation method for the resource distribution. The resource distributed to player position-i, r_i is equal to their gold g_i divided by their team gold, $\sum_{j=1}^{5} g_j$.

We also calculate the safelane and offlane gold ratio. This directly exposes the understanding of the support position. We divide the gold earned by core position with support position of each lane, excluding midlane. The lower number indicates that support players takes the resources from the core players more, which can cost the team a match.

3.2.5 Harassment

Many factors contribute to the team's chance of winning. Resource advantage is one of the principles for winning the game. Not only the team should gather as much gold as they can, but they must also prevent the enemy from harvesting, especially in the early game. Harassing the enemy helps the team achieving these two factors simultaneously. The enemy cannot farm, and the team has space to farm if the enemy heroes are far from the creeps.

We use the damage to enemy heroes in the first fifteen minutes to measure the harassment. The harassment is classified into two features, which are total harassment and in-lane harassment ratio. In-lane harassment ratio is the damage done to the enemy in the same lane respected to their position out of their harassment. The low value of ratio indicates that the player focused less on harassing in-lane.

3.2.6 Survival efficiency

This feature indicates how much the players can obtain resources while getting harassed. The gained gold is multiplied by the damage received in the first fifteen minutes. Therefore, a small number means the players are not getting much harassed, but they still cannot farm. A very high number indicates that the players are getting harassed; however, they still manage to gather a lot of golds. An average number can be interpreted a few ways, e.g. the players obtain resources because they are not getting harassed, however, they cannot farm due to harassment, or they can reasonably harvest while getting attacked mildly. Our assumption is that a higher number indicates the higher skill of the player.

3.2.7 Hero mastery

We compare the basic statistics of players to those statistics of professional players who competed in the tournaments. We average the professional players' statistics group by heroes and winning status. The reason to categorize by heroes is to measure how well the players play such character. Raw statistics are valuable features to indicate how well the player "plays the game," not controlling the heroes. Some heroes are already excellent at something. For instance, Zeus is the nuker; hence the heroes deal much more damage to the opposite heroes than the others. Or, heroes with blink or teleport skills are less likely to die. Therefore, we compare a player's statistics to professionals by minus the number to the professionals' average number by heroes and winning status. A positive number implies that a player can do better than the standard professional players at selected heroes. We compare only the features retrieve from OpenDota. We use the professional's data from July to August 2020 to calculate the mean statistics.

3.2.8 Team statistics

From our point of view, team statistics reveal more than team performance. It also shows the effectiveness of individual plays. For instance, the team may not die too often and frequently kill the enemy if the midlaner plays well. The team may have more gold if the offlaner is excellent as they can draw the enemy's attention to themselves instead of the others. We compute the team sum of OpenDota basic statistics as the features for the model.

3.2.9 Spatiotemporal features

In [9], the authors found that the spatial behavior of players could be different according to their skill level. Thus, we employ three spatiotemporal features, including the early-game area covered, mean, and standard deviation of team spread. The early-game area covered is the player's area traveled in the first fifteen minutes proportioned to the total accessible space. Team spread is the distance between each hero to the allies at the time, and is computed as in [9]. After the spread at a time is computed, the mean and standard deviation of it is computed for a period of time. The mean reveals the closeness of the team's position on average, and the standard deviation presents how much the team maintain their compactness. A higher number means the less distance the team can maintain, or the members are not close to each other.

3.2.10 Allies and enemies data

The allies and enemies calculated features are also selected as part of the features in our work. Allies can influence the player's statistics. For example, the offlaner can have many kills, and the midlaner can also have plenty of assists. The scenario might hint that offlaner's performances are high because the midlaner is often tanking. Therefore, the offlaner might be skillful, but not that much. We consider enemies' data in the same approach but in the opposite direction as those of allies. Note that these statistics might be exaggerated at times if the opposite team is mediocre.

Thus, we categorize our features into five groups.

- Individual Performance
- Allies Performance (Individually)
- Enemies Performance (Individually)
- Team Statistics
- Enemy's team statistics

Finally, we separate the dataset into ten parts. It is split the by the role of players and their winning status, since each role has different objective. Therefore, they need a different model to evaluate their skill. The player's winning status also affects the player's statistics. Losing skillful players might have identical statistics to those of the winning non-skillful players, which the model may consider expert players as amateurs, or the opposite. So, we assume that the winning and losing players need different treatments.

In our work, the feature engineering process is carried out on SQL Server as the data from the pre-processed phase is stored there. Also, processing the data using other tools, e.g. Python, might be slower as the size of the data, approximately half of a terabyte, exceeds the memory limit.

3.3 Machine learning modelling

Before we apply machine learning models to the dataset, the data are normalized using the min-max scaling technique as the range of each feature is highly different. Subsequently, we apply six models, which have been applied in game analytics as mentioned in Sect. 2, to each dataset. The selected models are:

1. Logistic Regression
2. Ridge Regression [19]

3. Support Vector Machine [7]
4. Naive Bayes
5. Random Forest [2]
6. XGBoost [5]

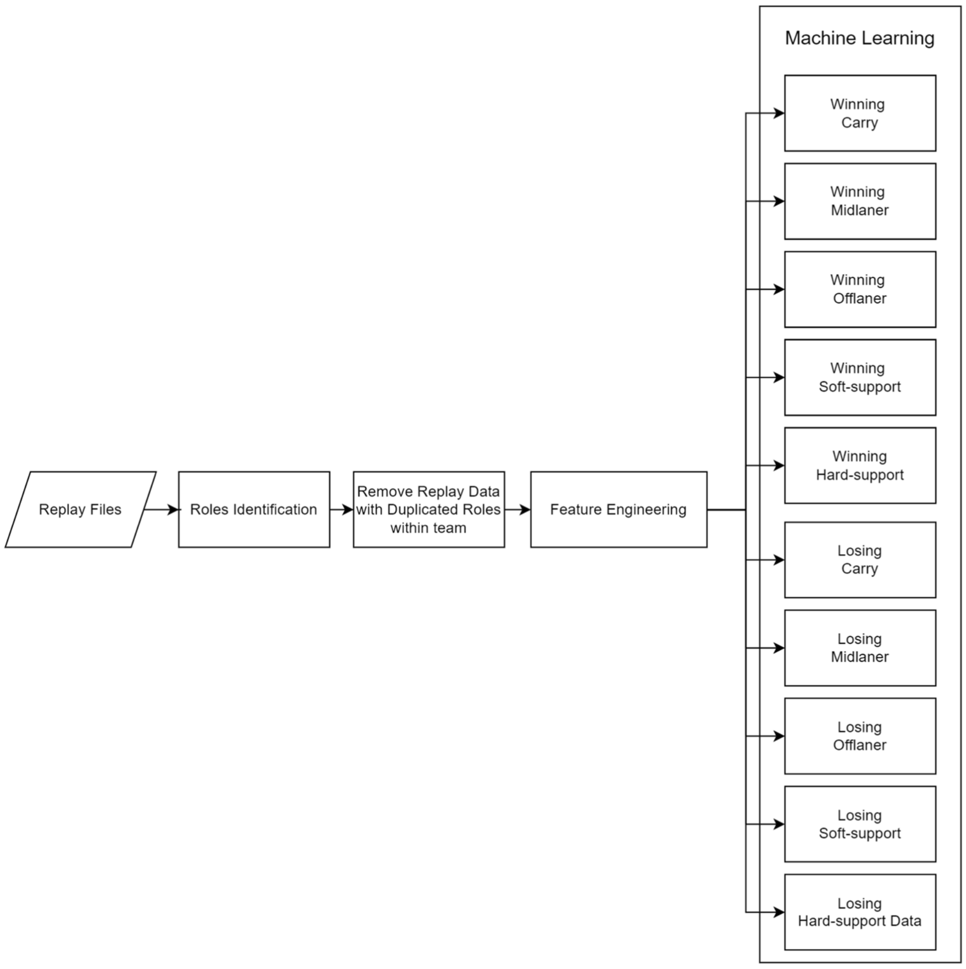
We saved a tenth of the dataset as our testing set, and the rest data are used as our training set. We used five-fold cross-validation to validate each model. The selected metrics were precision, recall, F1-score, and time used to validate the model in seconds.

The models were developed by Python based on Scikit-learn [27] which includes major many machine learning algorithms, including most of the presented algorithms, except the XGBoost. Standard feature pre-processing algorithms and metrics were implemented based on the Scikit-learn library.

4 Results

We applied the role identification to replay files and filtered out the replays which contained duplication roles within teams, i.e., the replay data with two winning carries. Hence, we had 12,412 replay data for this experiment. The features are the basic features which are obtained without computation, OpenDota feature only features, and extracted features as shown in Fig. 3. Then, we applied feature engineering, as mentioned in the earlier section. We then split the data into ten chunks by positions and teams (winning and losing). Finally, we applied six machine learning models to each piece of data. This experiment is a two-classes classification, in which the players with Ancient rank or higher are classified as skillful players, and the others are the opposite.

Fig. 3 Experiment Pipeline



This section presents the training result for each dataset and the testing result of ensemble models.

4.1 Training results

We trained ten models, including a model for each player-position, i.e., position one to position five of the winning and the losing team. We analyzed how much our proposed feature engineering can improve the model. We compared the performance of three feature sets. We first selected only the simple per-minute statistics, which required no computation. The second option was to select the statistics calculated by OpenDota only, and the final dataset used all suggested features. We ranked the model by summing the rank of each metric classified by the feature sets. Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 show the average results of each model. We ordered the models by their ranking. Note that the top two ROC AUCs, the representative results are highlighted as bold letters.

From the results, it can be seen that when all extracted features were applied, the highest performance can be achieved. That is, the ROC AUC could be achieved as high as 0.7852–0.8013 from all the results.

Figure 4 shows that the performance of most models can be improved by our suggested features, except Naive Bayes. Random Forest perform the best in term of precision. By using all extracted features, linear models such as Logistic Regression, Ridge Regression, as well as Support Vector Machine outperform the other models.

4.2 Testing results

Finally, we evaluated the combined model's performance with the testing set from the ten-fold cross-validation as the result is shown in Table 12.

5 Discussion

From the experiment results presented in Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and Fig. 4 in the previous section, it shows that, the proposed feature engineering approach was very effective. Particularly, the linear models, including Logistic regression (0.7932 ROC AUC) and Ridge regression (0.7927 ROC AUC), performed better than the ensemble models and naive Bayes. Either logistic regression and Ridge regression, and the support vector machine were the top two highest

Table 2 Classification results of machine learning models for losing safelaner dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Seconds)
Basic Features (No Computation)	Ridge Regression	0.5914	0.4024	0.4789	**0.6699**	0.0406
	Logistic Regression	0.5917	0.3925	0.4719	0.6691	0.4252
	Support Vector Machine	0.6105	0.3589	0.4520	**0.6725**	12.3669
	Naive Bayes	0.4949	0.5579	0.5242	0.6061	0.0286
	Random Forest	0.5986	0.2525	0.3551	0.6345	6.7368
	XGBoost	0.5297	0.4245	0.4713	0.6267	10.0118
OpenDota Features Only	Ridge Regression	0.6430	0.5378	0.5857	**0.7417**	0.1241
	Logistic Regression	0.6425	0.5277	0.5795	0.7394	1.3289
	Support Vector Machine	0.6511	0.4974	0.5640	**0.7396**	29.4817
	Naive Bayes	0.5309	0.6168	0.5702	0.6532	0.0936
	Random Forest	0.6784	0.3196	0.4345	0.6938	11.8697
	XGBoost	0.6021	0.4970	0.5444	0.6959	26.5099
All Extracted Features	Ridge Regression	0.6804	0.6080	0.6418	0.7861	0.2882
	Support Vector Machine	0.6988	0.5679	0.6265	**0.7897**	49.6481
	Logistic Regression	0.6806	0.6073	0.6415	**0.7876**	3.4504
	Random Forest	0.7255	0.3448	0.4668	0.7380	23.2077
	Naive Bayes	0.6861	0.2974	0.3880	0.7037	0.2159
	XGBoost	0.6535	0.5584	0.6018	0.7536	60.8718

Table 3 Classification results of machine learning models for winning safelaner dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5957	0.4578	0.5177	**0.6676**	0.0450
	Support Vector Machine	0.6045	0.4498	0.5158	**0.6703**	12.6711
	Logistic Regression	0.5993	0.4515	0.5149	0.6663	0.4620
	Naive Bayes	0.5178	0.5856	0.5495	0.6017	0.0272
	XGBoost	0.5400	0.4586	0.4960	0.6174	9.6967
	Random Forest	0.5636	0.3152	0.4042	0.6229	7.1046
OpenDota Features Only	Ridge Regression	0.6597	0.5761	0.6151	**0.7429**	0.1440
	Logistic Regression	0.6560	0.5667	0.6080	0.7399	1.4421
	Support Vector Machine	0.6611	0.5513	0.6012	**0.7442**	29.2866
	Naive Bayes	0.5416	0.6168	0.5765	0.6431	0.0898
	XGBoost	0.6190	0.5423	0.5781	0.6992	26.3092
	Random Forest	0.6595	0.3963	0.4951	0.6923	11.5869
All Extracted Features	Logistic Regression	0.6953	0.6430	0.6681	**0.7903**	3.4862
	Ridge Regression	0.6940	0.6383	0.6649	0.7902	0.2868
	Support Vector Machine	0.7048	0.6053	0.6512	**0.7907**	50.1778
	Naive Bayes	0.7300	0.1256	0.2102	0.6944	0.2438
	Random Forest	0.7076	0.4244	0.5306	0.7404	18.3874
	XGBoost	0.6680	0.5940	0.6288	0.7603	64.8065

Table 4 Classification results of machine learning models for losing midlaner dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5824	0.3855	0.4637	**0.6636**	0.0486
	Logistic Regression	0.5861	0.3779	0.4594	0.6629	0.5370
	Support Vector Machine	0.6030	0.3482	0.4414	**0.6669**	12.5436
	Naive Bayes	0.5118	0.4863	0.4979	0.6034	0.0264
	XGBoost	0.5236	0.4154	0.4632	0.6150	9.0018
	Random Forest	0.5482	0.2246	0.3179	0.6170	7.7800
OpenDota Features Only	Ridge Regression	0.6476	0.5389	0.5882	**0.7386**	0.1554
	Support Vector Machine	0.6630	0.5034	0.5721	**0.7366**	29.3752
	Logistic Regression	0.6470	0.5281	0.5815	0.7363	1.5012
	Naive Bayes	0.5442	0.6021	0.5712	0.6492	0.0876
	Random Forest	0.6629	0.3107	0.4231	0.6902	11.8495
	XGBoost	0.6092	0.4976	0.5476	0.7023	26.2581
All Extracted Features	Ridge Regression	0.6764	0.6087	0.6407	0.7847	0.3248
	Logistic Regression	0.6779	0.6057	0.6397	**0.7855**	3.5487
	Support Vector Machine	0.6951	0.5758	0.6298	**0.7875**	51.8835
	Random Forest	0.7128	0.3513	0.4706	0.7368	25.2266
	Naive Bayes	0.7121	0.2080	0.2975	0.7008	0.2216
	XGBoost	0.6559	0.5584	0.6032	0.7539	57.3334

Table 5 Classification results of machine learning models for winning midlaner dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5948	0.4678	0.5237	0.6574	0.0398
	Logistic Regression	0.5961	0.4628	0.5210	**0.6567**	0.3970
	Support Vector Machine	0.6000	0.4504	0.5145	**0.6607**	13.4493
	Naive Bayes	0.5151	0.5837	0.5470	0.5968	0.0292
	XGBoost	0.5390	0.4777	0.5065	0.6076	6.8853
	Random Forest	0.5783	0.3520	0.4376	0.6230	8.8460
OpenDota Features Only	Ridge Regression	0.6530	0.5827	0.6158	**0.7367**	0.1300
	Logistic Regression	0.6537	0.5777	0.6133	0.7343	1.3576
	Support Vector Machine	0.6584	0.5661	0.6087	**0.7371**	29.5859
	Naive Bayes	0.5681	0.6126	0.5892	0.6483	0.0828
	XGBoost	0.6330	0.5657	0.5973	0.7073	25.9846
	Random Forest	0.6536	0.4279	0.5171	0.6903	11.9204
All Extracted Features	Logistic Regression	0.6955	0.6464	0.6699	**0.7852**	4.1870
	Ridge Regression	0.6916	0.6480	0.6690	0.7847	0.3296
	Support Vector Machine	0.6985	0.6369	0.6662	**0.7884**	51.4482
	Naive Bayes	0.7265	0.2695	0.3488	0.6976	0.1928
	Random Forest	0.6936	0.4494	0.5453	0.7369	21.8859
	XGBoost	0.6709	0.6179	0.6432	0.7581	63.7045

Table 6 Classification results of machine learning models for losing offlaner dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5826	0.2887	0.3860	**0.6674**	0.0470
	Logistic Regression	0.5907	0.2802	0.3799	0.6661	0.4256
	Naive Bayes	0.4512	0.4943	0.4713	0.6028	0.0320
	Support Vector Machine	0.6389	0.1730	0.2717	**0.6699**	12.1327
	XGBoost	0.4939	0.3339	0.3983	0.6216	9.6920
	Random Forest	0.5635	0.1011	0.1714	0.6183	7.2276
OpenDota Features Only	Ridge Regression	0.6367	0.4640	0.5366	**0.7486**	0.1518
	Logistic Regression	0.6369	0.4513	0.5282	0.7448	1.3722
	Naive Bayes	0.4962	0.5679	0.5292	0.6495	0.0959
	Support Vector Machine	0.6611	0.4016	0.4994	**0.7471**	28.3910
	XGBoost	0.5918	0.4341	0.5008	0.7095	25.3413
	Random Forest	0.6785	0.1729	0.2754	0.6980	13.5730
All Extracted Features	Logistic Regression	0.6649	0.5546	0.6045	**0.7953**	3.7624
	Ridge Regression	0.6649	0.5485	0.6010	0.7940	0.3339
	Support Vector Machine	0.6983	0.4892	0.5750	**0.7951**	48.1505
	Random Forest	0.7448	0.2039	0.3198	0.7390	18.7031
	XGBoost	0.6398	0.4904	0.5551	0.7595	64.5788
	Naive Bayes	0.6528	0.3213	0.3692	0.7032	0.2264

Table 7 Classification results of machine learning models for winning offlaner dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5996	0.3843	0.4683	**0.6730**	0.0400
	Logistic Regression	0.6002	0.3740	0.4608	0.6721	0.4824
	Naive Bayes	0.4841	0.5129	0.4979	0.6019	0.0288
	Support Vector Machine	0.6106	0.3376	0.4347	**0.6797**	12.1713
	XGBoost	0.5249	0.3914	0.4484	0.6282	9.7475
	Random Forest	0.5837	0.1960	0.2934	0.6346	7.2492
OpenDota Features Only	Ridge Regression	0.6442	0.5238	0.5778	**0.7478**	0.1448
	Logistic Regression	0.6454	0.5129	0.5716	0.7447	1.4866
	Support Vector Machine	0.6618	0.4893	0.5626	**0.7474**	28.4866
	Naive Bayes	0.5200	0.5898	0.5525	0.6462	0.0879
	Random Forest	0.6601	0.2643	0.3773	0.6988	10.9849
	XGBoost	0.5964	0.4768	0.5298	0.7054	27.2785
All Extracted Features	Ridge Regression	0.6857	0.6053	0.6427	**0.7953**	0.3553
	Logistic Regression	0.6880	0.6033	0.6425	0.7952	3.7542
	Support Vector Machine	0.7076	0.5604	0.6252	**0.7965**	51.7482
	Random Forest	0.7341	0.3138	0.4393	0.7419	27.5172
	XGBoost	0.6662	0.5464	0.6002	0.7632	54.8290
	Naive Bayes	0.6860	0.2522	0.3450	0.7033	0.2088

Table 8 Classification results of machine learning models for losing soft support dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5637	0.2277	0.3244	**0.6636**	0.0438
	Logistic Regression	0.5696	0.2198	0.3171	**0.6621**	0.4084
	Naive Bayes	0.4362	0.4567	0.4458	0.5909	0.0326
	XGBoost	0.4893	0.3053	0.3758	0.6228	8.9141
	Support Vector Machine	0.6519	0.0407	0.0764	0.6633	11.8173
	Random Forest	0.5465	0.0626	0.1121	0.6021	6.7082
OpenDota Features Only	Ridge Regression	0.6247	0.4294	0.5088	**0.7445**	0.1462
	Logistic Regression	0.6297	0.4180	0.5022	0.7407	1.3044
	Naive Bayes	0.4713	0.5706	0.5157	0.6441	0.1062
	Support Vector Machine	0.6625	0.3494	0.4574	**0.7466**	26.8221
	XGBoost	0.5927	0.3944	0.4735	0.7066	24.5256
	Random Forest	0.6884	0.1209	0.2054	0.6879	10.6561
All Extracted Features	Ridge Regression	0.6750	0.5303	0.5939	**0.7953**	0.3362
	Logistic Regression	0.6653	0.5354	0.5932	0.7939	4.0807
	Support Vector Machine	0.6983	0.4634	0.5570	**0.7978**	47.4450
	Naive Bayes	0.6686	0.2486	0.3334	0.7058	0.2698
	Random Forest	0.7733	0.1644	0.2711	0.7402	18.8273
	XGBoost	0.6462	0.4751	0.5475	0.7693	66.8138

Table 9 Classification results of machine learning models for winning soft support dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5831	0.2875	0.3851	**0.6672**	0.0502
	Logistic Regression	0.5823	0.2742	0.3727	0.6663	0.4302
	Naive Bayes	0.4485	0.4679	0.4573	0.5877	0.0292
	Support Vector Machine	0.6526	0.1644	0.2625	**0.6720**	12.2928
	XGBoost	0.4991	0.3334	0.3997	0.6204	9.9054
	Random Forest	0.5770	0.0967	0.1654	0.6160	6.9868
OpenDota Features Only	Ridge Regression	0.6378	0.4713	0.5419	**0.7502**	0.1452
	Logistic Regression	0.6378	0.4569	0.5324	0.7466	1.4868
	Support Vector Machine	0.6551	0.4033	0.4991	**0.7477**	28.0414
	Naive Bayes	0.4919	0.5552	0.5208	0.6411	0.0910
	Random Forest	0.6682	0.1581	0.2556	0.6910	11.3261
	XGBoost	0.5846	0.4232	0.4909	0.7047	27.5768
All Extracted Features	Ridge Regression	0.6740	0.5644	0.6141	0.8009	0.3202
	Logistic Regression	0.6708	0.5622	0.6115	**0.8010**	3.6273
	Support Vector Machine	0.6976	0.5051	0.5859	**0.8010**	47.3140
	Naive Bayes	0.5374	0.5957	0.5238	0.7015	0.2072
	Random Forest	0.7557	0.2069	0.3248	0.7508	19.2310
	XGBoost	0.6563	0.5141	0.5765	0.7716	64.7814

Table 10 Classification results of machine learning models for losing hard support dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5724	0.1835	0.2779	**0.6571**	0.0454
	Logistic Regression	0.5824	0.1735	0.2673	0.6550	0.4276
	Naive Bayes	0.4108	0.3271	0.3629	0.5752	0.0362
	XGBoost	0.4589	0.2451	0.3194	0.6104	7.9283
	Support Vector Machine	0.9333	0.0038	0.0075	**0.6567**	11.3977
	Random Forest	0.4971	0.0240	0.0458	0.5979	6.8850
OpenDota Features Only	Ridge Regression	0.6213	0.3989	0.4857	**0.7432**	0.1512
	Logistic Regression	0.6202	0.3854	0.4752	0.7396	1.2866
	Naive Bayes	0.4533	0.5182	0.4832	0.6386	0.1180
	Support Vector Machine	0.6742	0.2909	0.4060	**0.7429**	25.1021
	XGBoost	0.5689	0.3420	0.4271	0.7005	22.6299
	Random Forest	0.6835	0.0776	0.1392	0.6809	10.6189
All Extracted Features	Ridge Regression	0.6694	0.5045	0.5750	**0.7906**	0.3308
	Logistic Regression	0.6605	0.5074	0.5735	**0.7906**	3.9012
	Support Vector Machine	0.7095	0.4289	0.5341	**0.7944**	44.9401
	XGBoost	0.6379	0.4305	0.5139	0.7600	63.7330
	Random Forest	0.7842	0.1142	0.1992	0.7270	18.6859
	Naive Bayes	0.5825	0.2807	0.3183	0.7000	0.3098

Table 11 Classification results of machine learning models for winning hard support dataset

Features	Model	Precision	Recall	F1 Score	ROC AUC	Time Used (Second)
Basic Features (No Computation)	Ridge Regression	0.5540	0.2216	0.3164	**0.6658**	0.0400
	Logistic Regression	0.5608	0.2074	0.3027	0.6640	0.3976
	Naive Bayes	0.4318	0.4238	0.4271	0.5899	0.0282
	XGBoost	0.4933	0.2951	0.3692	0.6237	9.6583
	Support Vector Machine	0.7011	0.0281	0.0540	**0.6676**	12.1728
	Random Forest	0.5380	0.0531	0.0967	0.6113	7.3864
OpenDota Features Only	Ridge Regression	0.6241	0.4232	0.5043	**0.7477**	0.1568
	Logistic Regression	0.6304	0.4172	0.5021	0.7445	1.3410
	Support Vector Machine	0.6687	0.3430	0.4533	**0.7485**	27.0325
	Naive Bayes	0.4774	0.5144	0.4944	0.6432	0.1022
	XGBoost	0.5784	0.3842	0.4616	0.7077	26.8033
	Random Forest	0.6897	0.1073	0.1856	0.6959	11.2901
All Extracted Features	Logistic Regression	0.6716	0.5336	0.5945	**0.7998**	4.2375
	Ridge Regression	0.6688	0.5337	0.5934	0.7995	0.3358
	Support Vector Machine	0.6967	0.4573	0.5521	**0.8013**	46.9519
	Naive Bayes	0.5389	0.5589	0.5072	0.7091	0.2562
	XGBoost	0.6523	0.4795	0.5525	0.7701	56.5664
	Random Forest	0.7592	0.1501	0.2506	0.7429	18.0127

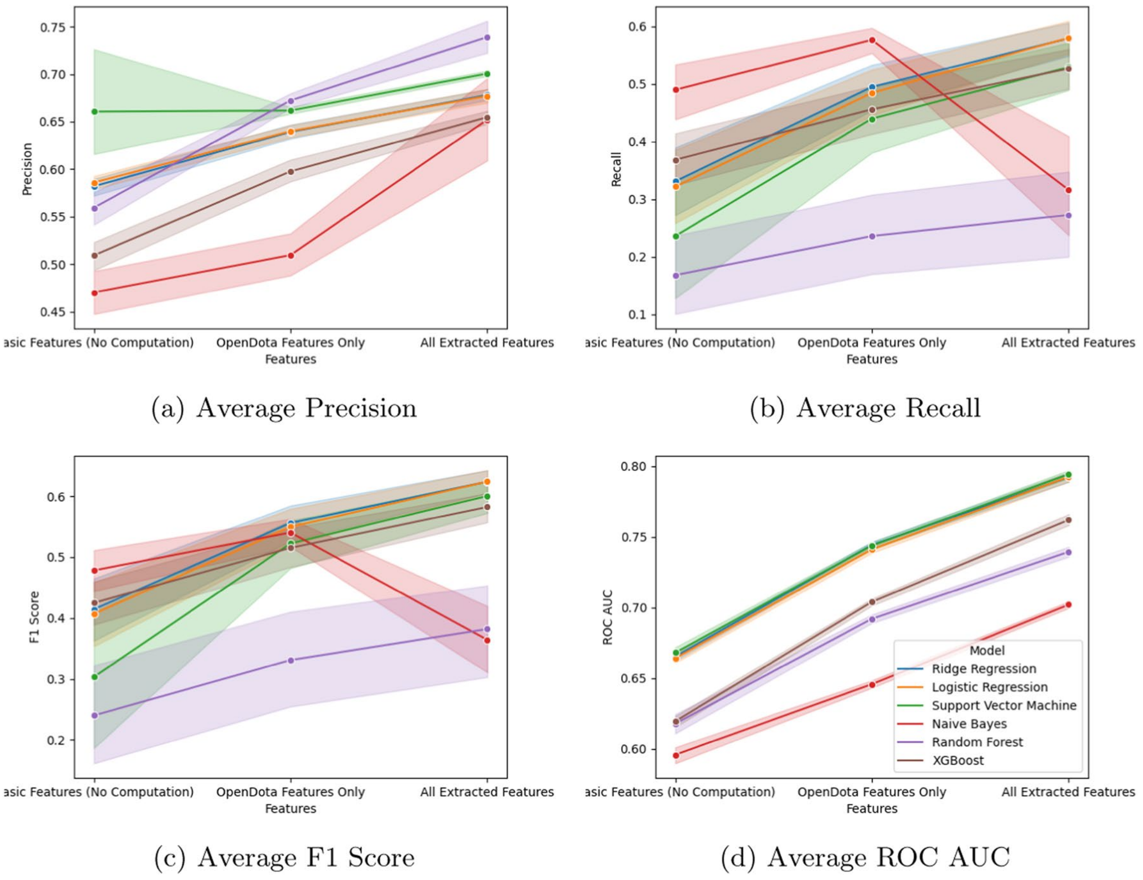


Fig. 4 Performance Group by Model

Table 12 Classification results of the testing dataset

Precision	Recall	F1 Score	ROC AUC
0.7091	0.5850	0.6411	0.8123

performance models regardless of the feature sets. Though the execution time of the support vector machine was rather high compared to the regression models, i.e., the support vector machine execution times (48.6955 s on average) were almost 50-times higher than the execution time of the regression models (2.1415s on average). These could give options for consideration when applying the work in real-life application since the constraints could be varied.

When the models were combined to evaluate the testing set as the result is shown in Table 12, the combined pipeline achieved 0.71 precision, 0.58 recall, 0.6411 F1, and 0.8123 ROC AUC score. Such results not only illustrate the prediction ability of the proposed work, but also, it can help stakeholder, i.e., game players, game designers, or etc, improving their work. For example, game players can apply the work to better prepare the matches in advance, both as the team and the individual role. Meanwhile, game designers can better estimate the impact of changes, such as to improve monetary mechanics [20], in game according to player skills or behavioral types [29].

6 Conclusion

This paper proposed a machine learning pipeline to evaluate Dota 2 player skill. The model consisted of three main parts, including data collection, feature engineering, and machine learning modeling. First, we employed OpenDota API to collect the data. Such data included players' rank, their statistics of each match, and their replay URL. Then, the data were downloaded and parsed. Subsequently, the feature engineering, including players' basic statistics, the computed statistics by OpenDota, and our suggestion, was carried out. Then, we calculated the team's statistics of players as the features and used every player's features in a match and both teams' statistics as features of each player. We separated the datasets into ten parts by their roles and winning conditions and cut a tenth of each as our testing set.

In our future work, there are a few open research questions are to be addressed. First, as the models in this paper are belonged to the individual, identifying the group of skillful players with the potential to be a good teamwork, given the team characters, is to be investigated. Such team-working with a player's style to align and support the other players could be a challenging task, but it could enormously benefit the team evaluation. This could be investigated by modeling the problem into a bipartite matching.

Secondly, a more-complex spatiotemporal feature, i.e. the trajectory of the players both allies or enemies in a time window, is to be explored. We believe that the additional information from the trajectory could provide a better prediction models. Though such kind of feature could require a high computational power to extract.

Lastly, a real-time live skill prediction problem will be addressed. The rationale is from the situation where the new players enter the game without much background knowledge, or some returned-players whose skill changes substantially. Thus, extracting the live-game features and predict the skill in a real-time manner will be essential. Similar to the previous problem, a better execution performance of the machine learning pipeline will be an important issue.

Acknowledgements This research work was partially supported by Chiang Mai University.

Data availability The datasets generated during and/or analysed during the current study are available in the OpenDota repository, https://api.opendota.com/api/explorer?sql=select%20match_id%20from%20 public_matches%0D%0Awhere%20start_time%20%3E%3D%20 extract(epoch%20from %20timestamp%20%272,020–09-14T17%3A00%3A00.000Z%27)%0D%0Aand%20 start_time%20%3C%3D%20 extract(epoch%20from%20timestamp%20%272,020–10- 14T17%3A00%3A00.000Z%27)%0D%0Aand%20game_mode%20IN%20(1%2C%202 %2C%2022)%0D%0Aand%20num_rank_tier%20%3D%2010%0D%0Aand%20lobby_type %20%3D%207%3B.

Declarations

Competing interests We wish to confirm that there are no known competing interests associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

1. Aung M, Bonometti V, Drachen A, et al. Predicting skill learning in a large, longitudinal moba dataset. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG), 2018;1–7, https://doi.org/10.1109/CIG.2018.8490431
2. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32. https://doi.org/10.1023/A:1010933404324.
3. Castaneda L, Sidhu M, Azose J, et al. Game play differences by expertise level in dota 2, a complex multiplayer video game. Int J Gaming Comput-Med Simul. 2016;8:1–24. https://doi.org/10.4018/IJGCMS.2016100101.
4. Cavadenti O, Codocedo V, Boulicaut J, et al. What did i do wrong in my moba game? mining patterns discriminating deviant behaviours. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016;662–671, https://doi.org/10.1109/DSAA.2016.75
5. Chen T, Guestrin C. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, KDD '16, 2016;785–794, https://doi.org/10.1145/2939672.2939785,
6. Chung H. parser. 2016; https://github.com/odota/parser
7. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273–97.
8. Demediuk S, York P, Drachen A, et al. Role identification for accurate analysis in dota 2. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2019;15(1):130–8. https://ojs.aaai.org/index.php/AIIDE/article/view/5235
9. Drachen A, Yancey M, Maguire J, et al. Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2). Piscataway: IEEE; 2014.
10. Eggert C, Herrlich M, Smeddinck J, et al. Classification of player roles in the team-based multi-player game dota 2. In: Chorianopoulos K, Divitini M, Baalsrud Hauge J, et al., editors. Entertainment computing - ICEC 2015. Cham: Springer International Publishing; 2015. p. 112–25.
11. Esports Earnings. Largest overall prize pools in esports. 2021a;https://www.esportsearnings.com/tournaments
12. Esports Earnings. Paris saint-germain esports results for dota 2 (psg.lgd).2021b; https://www.esportsearnings.com/teams/623-paris-saint-germain-esports/results/2439-dota-2-psg-lgd
13. Esports Earnings. Top 100 highest overall earnings. 2021c;https://www.esportsearnings.com/players
14. Fakazlı AE. The effect of covid-19 pandemic on digital games and esports. Int J Sport Cult Sci. 2020;8:335–44.
15. Gao L, Judd J, Wong D, et al. Classifying dota 2 hero characters based on play style and performance. Univ of Utah Course on ML 2013;
16. Hanke L, Chaimowicz L. A recommender system for hero line-ups in moba games. In: AIIDE 2017;
17. Hodge V, Devlin S, Sephton N, et al. Win prediction in esports: Mixed-rank match prediction in multi-player online battle arena games. 2017;
18. Hodge V, Devlin S, Sephton N, et al. Win prediction in multi-player esports: live professional match prediction. IEEE Trans Games. 2019. https://doi.org/10.1109/TG.2019.2948469.
19. Hoerl AE, Kennard RW. Ridge regression: biased estimation for nonorthogonal problems. Technometrics. 1970;12(1):55–67. https://doi.org/10.1080/00401706.1970.10488634.
20. Jensen KH, Fynbo L, Hansen NN. The interplay between game design and social practice. New Media Soc. 2023. https://doi.org/10.1177/14614448231187820.
21. Johansson F, Wikström J. Result prediction by mining replays in dota 2 2015;
22. Johns T. All you need to know about dota 2 ranking system. 2024; https://dmarket.com/blog/dota2-ranks/
23. Katona A, Spick R, Hodge VJ, et al. Time to die: Death prediction in dota 2 using deep learning. In: 2019 IEEE Conference on Games (CoG), 2019;1–8, https://doi.org/10.1109/CIG.2019.8847997
24. König R, Johansson U, Riveiro M, et al. Modeling golf player skill using machine learning. In: International Cross-Domain Conference for Machine Learning and Knowledge Extraction : CD-MAKE 2017: Machine Learning and Knowledge Extraction, no. 10410 in Lecture Notes in Computer Science, pp 275–294, 2017;https://doi.org/10.1007/978-3-319-66808-6_19
25. Newzoo. Global esports & live streaming market report. Newzoo: Tech. rep; 2021.
26. Pappalardo L, Cintia P, Ferragina P, et al. Playerank: data-driven performance evaluation and player ranking in soccer via a machine learning approach. ACM Trans Intell Syst Technol. 2019. https://doi.org/10.1145/3343172.
27. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
28. Pobiedina N, Neidhardt J, Calatrava Moreno MdC, et al. Ranking factors of team success. In: Proceedings of the 22nd International Conference on World Wide Web. Association for Computing Machinery, New York, NY, USA, WWW '13 Companion, 2013;1185-1194, https://doi.org/10.1145/2487788.2488147,
29. Santos ACG, Oliveira W, Hamari J, et al. The relationship between user types and gamification designs. User Model User-Adapt Interaction. 2021;31:907–40.
30. Sarlis V, Tjortjis C. Sports analytics - evaluation of basketball players and team performance. Inform Syst. 2020;93: 101562. https://doi.org/10.1016/j.is.2020.101562.
31. Schordt M. Clarity 2. 2021; https://github.com/skadistats/clarity
32. Semenov A, Romov P, Korolev S, et al. Performance of machine learning algorithms in predicting game outcome from drafts in dota 2. In: Ignatov DI, Khachay MY, Labunets VG, et al., editors. Analysis of images, social networks and texts. Cham: Springer International Publishing; 2017. p. 26–37.

33. Wang N, Li L, Xiao L, et al. Outcome prediction of dota2 using machine learning methods. In: ICMAI '18: Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence, 2018;61–67
34. Wells J. Dota 2 ranks: ranking system explained, mmr, and more. 2021; https://www.theloadout.com/dota-2/ranks
35. Yuen S, Thomson JD, Don O. Automatic player identification in dota 2. 2020;
36. Zoid. Dota 2 match history webapi. 2012; https://dev.dota2.com/forum/dota-2/spectating/replays/webapi/48733-dota-2-match-history-webapi