Machine Learning Methods for Predicting *League of Legends* Game Outcome

Juan Agustín Hitar-García, Laura Morán-Fernández, and Verónica Bolón-Canedo

***Abstract*—The video game *League of Legends* has several professional leagues and tournaments that offer prizes reaching several million dollars, making it one of the most followed games in the Esports scene. This article addresses the prediction of the winning team in professional matches of the game, using only pregame data. We propose to improve the accuracy of the models trained with the features offered by the game application programming interface (API). To this end, new features are built to collect interesting information, such as the skills of a player handling a certain champion, the synergies between players of the same team or the ability of a player to beat another player. Then, we perform feature selection and train different classification algorithms aiming at obtaining the best model. Experimental results show classification accuracy above 0.70, which is comparable to the results of other proposals presented in the literature, but with the added benefit of using few samples and not requiring the use of external sources to collect additional statistics.**

***Index Terms*—Esports, feature creation, *League of Legends* (*LoL*), model ensemble, prediction, video games.**

I. Introduction

In THE context of electronic entertainment, the broadcasting of professional video games, known as Esports, is gaining more and more relevance. In these games, the opponents face each other to achieve one of the important prizes at stake and the recognition of the public, reaching an audience of 495 million people and exceeding one billion dollars in revenues by 2020 [1]. In this emerging field, *League of Legends* (*LoL*) [2] from Riot Games Inc. is at the top of the most watched games with 348.8 million hours of viewing in 2019 [1], giving an idea of the popularity of this game. The creators of the game have made available to developers an application programming interface (API) that allows access to a large amount of detailed data for every game played. This makes research in this area very interesting, for example, to develop predictive models to assist teams about the strategy to follow. However, there are few studies focused on *LoL*, and there is an almost complete lack of research on the professional levels of the game.

This study focuses on using pregame data (teams, players, champions, etc.) from different *LoL* leagues and tournaments to predict the winning team of professional games. The main motivation of our pregame approach is their applicability. Predicting the winning team before match starts is very valuable data, for example, to make decisions regarding team composition in terms of players, champions, and roles (recommendation system), or as part of betting systems. Choosing the professional level of the game reduces considerably the number of available observations (compared to the other levels, thousands of games versus hundreds of thousands or even millions). In order to obtain robust models, it is necessary to exploit the particularities of this dataset, such as the fact that the matches are played by a limited number of teams and players. This allows the extraction of features that provide much more relevant information to the predictive model than just the fact of participating or not in the encounter, as would be the case when using the original set of features provided by the game API.

We propose an approach that, based on the analysis of the original data, detects possible new features that are important for predicting the outcome of a match. These features reflect, for example, the dexterity of the player in handling a certain champion, what synergies are produced when two players are in the same team, or the ability of a player to beat another player when they face each other. Subsequently, a preprocessing step is carried out, which includes the creation of these new features and feature selection. Next, several machine learning algorithms are trained, and those that show the best performance are selected to be used in a meta-model to further increase accuracy. The main contribution of this study is the development of a robust model for winner prediction in professional *LoL* games. Despite the fact of having a dataset with limited information, our approach solves this problem by creating new features which will help selecting the best predictive algorithms that will be combined into a meta-model.

The remainder of this article is organized as follows. In Section II, a general description of the *LoL* game is given. Section III reviews the related work. Section IV describes the methods and materials we work with, including the dataset (Section IV-A), preprocessing tasks (Sections IV-B and IV-C), as well as the classification algorithms (Section IV-D). Section V

Manuscript received 19 May 2021; revised 3 November 2021 and 18 January 2022; accepted 11 February 2022. Date of publication 23 February 2022; date of current version 16 June 2023. This work was supported in part by the National Plan for Scientific and Technical Research and Innovation of the Spanish Government under Grant PID2019-109238GB-C2, in part by the Xunta de Galicia under Grant ED431C 2018/34 with the European Union ERDF funds, in part by the CITIC, as a Research Center accredited by Galician University System, is funded by "Consellería de Cultura, Educación e Universidades from Xunta de Galicia," supported in an 80% through ERDF Funds, ERDF Operational Programme Galicia 2014–2020, and the remaining 20% by "Secretaría Xeral de Universidades' under Grant ED431G 2019/01. *(Corresponding author: Verónica Bolón-Canedo.)*

The authors are with CITIC, Universidade da Coruña, 15071 A Coruña, Spain (e-mail: j.a.hitarg@udc.es; laura.moranf@udc.es; veronica.bolon@udc.es).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TG.2022.3153086.

Digital Object Identifier 10.1109/TG.2022.3153086

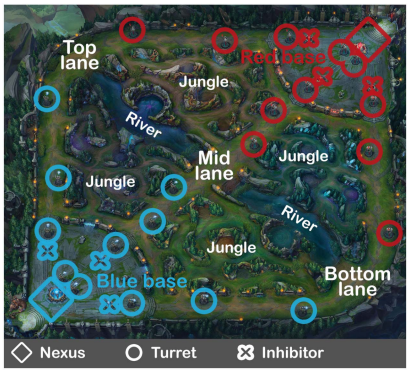

Fig. 1. *LoL* game map "Summoner's Rift."

shows the proposed approach, including the exploratory analysis performed (Section V-A), the data partitioning and preprocessing (Section V-B), and the training for model prediction (Section V-C). Then, in Section VI, the specific parameters of the implementation are presented (Section VI-A), and the solution is evaluated, presenting the results and comparing the proposed models with each other and with related works (Section VI-B). Finally, Section VII concludes this article.

II. Game Description

In professional *LoL* matches, two teams of five players, *blue team* and *red team*, face each other on a map called "Summoner's Rift." This map (see Fig. 1) is divided into two parts by a river, and each of these parts is assigned to a team, with the base of the blue team in the lower-left corner and the base of the red team in the upper-right corner. Three lanes (top, middle, and bottom) connect the bases, and between each of these lanes there are neutral zones, called jungles. Before the battle begins, the players of each team must decide how to distribute the five existing roles or positions: Top, jungle, middle, bottom (bottom role is also known as ADC or Attack Damage Carry), and support. These roles are named after the zone of the map they occupy, except for the support role, which helps and gives protection to the rest of the teammates [3]. Likewise, players must also choose a champion from the 139 available. These champions have unique abilities and characteristics, making the choice champion/role composition of the team an important strategic factor that can determine the final outcome of the match. During the course of the game, players on the same team cooperate to destroy the defensive structures of the enemy in a lane (called turrets and inhibitors) and reach the opposing base in order to destroy their main structure, the Nexus, the final goal of the game. To advance toward that goal, it is of vital importance that champions gain experience levels, which allows them to unlock and enhance their special abilities, as well as earning gold to be able to buy items that increase the characteristics of the champion. This is achieved by defeating nonplayer characters or enemy champions, and destroying opponent structures.

III. Related Work

Despite the potential that *LoL* can have among machine learning researchers to develop algorithms that make good predictions about the game, a review of the specialized literature reveals that this is not a widely explored topic. In the case of predicting the winning team, as discussed in this article, two groups of approaches can be distinguished depending on what information is considered. The first group takes into account the in-game information, that is, the information related to the game state generated during the course of the game. This information may include, for example, gold earned, experience gained, or turrets destroyed at a certain point in the game. Within this group, Silva *et al.* [4] employed recurrent neural networks to predict the winner of the game based on information related to the game state at a given minute. The dataset used by the authors, obtained from Kaggle repository, is composed of 7621 instances of professional games. Thus, they achieve an accuracy of 63.91% using data from the first 5 min of the game, and 83.54% for the interval between the 20th and 25th min. Kim *et al.* [5] proposed a method which considers uncertainty to better calibrate the confidence level in neural networks. In this way, they manage to predict the winning team in real time with an accuracy of 73.81%. Kho *et al.* [6] presented a logic mining technique, kSATRA, to obtain the logical relationship between team tactics during the game and the game objectives. The obtained rule is used to classify the outcome of future matches. The authors extract, from the game website, 400 instances with information about ten teams from each of three leagues. A training set accuracy of 73% is obtained with this dataset. Also with in-game information, Novak *et al.* [7] carried out a team performance analysis. For this purpose, they use the assessment of expert coaches to determine the features during the game most related to the fact of winning or losing the match. With the selected features, they create a generalized linear mixed model with binomial logit link function to find those contributing the most to the final result of the match. In this way they achieve an accuracy of 95.8% using a dataset of 119 match instances extracted from the official video and historical match repositories. To conclude with this group, Kang *et al.* [8] compared the Poisson model and the Bradley Terry model, evaluating their performance in terms of the predictive capacity for the match outcome. For this purpose, authors consider in-game data after the end of the match from 333 games, reaching an accuracy of 69.96% with the Poisson model of Dixon and Coles.

The second group of approaches only takes into account pregame information, in other words, information that is available before the match starts, such as the players involved, the champions they will play with, or the role that each player will assume. Within this group, Gu *et al.* [9] presented a novel neural network, NeuralAC, to predict the outcome of a match. This neural network captures the importance of interactions between members of the same team and between members of rival teams. To evaluate their approach, the authors use a dataset with 754 700 match instances, extracted from the game API, to achieve an accuracy of 62.09%. White *et al.* [10] exploited the temporal data from the games to determine the psychological momentum, positive or negative, of the players. Then, they combine this information with player experience to predict the winning team. These authors used a dataset of 87 743 game instances extracted from the game API. In addition, they also obtained a summary of the profiles of the participants in

those games from OP.GG Website [11], as well as information regarding the champions from CHAMPION.GG Website [12]. With all this data they trained their model, based on recursive neural network and logistic regression, to achieve an accuracy of 72.10%. Ong *et al.* [13] employed the k-means algorithm to cluster the behavior of players based on their statistics in played games. In this way, they obtain a series of groups that describe different playing styles of those players. Using these clusters as features to define each team, they train various classification algorithms to predict the winner, achieving an accuracy of 70.4% with support vector machines. The authors used 113 000 game instances, randomly extracted from the game API, in their study. They also used the game API to obtain the player statistics from which they perform the clustering. Finally, with the information available before the start of the match, Chen *et al.* [14] aimed to detect player skills that are determinant for the outcome of a match. To do so, they used several skill-based predictive models to decompose player skills into interpretive parts. The impact on game outcome of these parts is evaluated in statistical terms. This approach achieves an accuracy of 60.24% with a dataset of 231 212 game instances extracted from the game API. Note that [13] and [14] are not peer reviewed.

Other relevant articles use as a basis for study the game *Defense of the Ancients 2 (DotA2)*, which is very similar to *LoL* in its concept. This game is more studied in literature, and, therefore, has a larger number of articles on winning team prediction. For this game, Conley *et al.* [15] proposed a hero recommendation engine. For this purpose, they train a kNN (k-nearest neighbors) algorithm using as features the heroes that compose each team. In this way, they achieve an accuracy of 70% with a dataset of 18 000 instances. Agarwala *et al.* [16] performed a principal component analysis to extract the interaction between the heroes. With the obtained result, they train a logistic regression algorithm with a dataset composed of 40 000 instances. Applying their approach, the authors achieve an accuracy of 62%. Hodge *et al.* [17] presented the study of real-time prediction of the outcome in professional *DotA2* matches using in-game information. The authors used standard machine learning, feature creation, and optimization algorithms on a mixed professional and nonprofessional games dataset to create their model. They tested the obtained model in real time during a championship match, reaching an accuracy of 85% after 5 min of gameplay. Lan *et al.* [18] proposed a model that allows predicting the winning team using data on player behavior during the match (in-game). They first extract the features that define player behavior with a convolutional neural network. These features are modeled as a temporal sequence that is processed by a recurrent neural network. Finally, the output of these two networks for each team are combined to forecast the outcome. Thus, the authors use a training set of 20 000 instances to obtain, after the first 20 combats occurred during the match, an accuracy of 87.85%.

Outside the context of multiplayer online battle arena (MOBA) there are also articles on other Esports that attempt to predict the match outcome. Thus, Sánchez-Ruiz *et al.* [19] used a set of numerical matrices, which represent the units influence on the game map, to predict the outcome of the game *StarCraft*. The paper by Mamulpet [20] focused on the videogame players unknown battlegrounds, and used various artificial neural network techniques to predict the winning player of the game by knowing their starting position. For the game *Counter Strike*, Xenopoulos *et al.* [21] created a probabilistic model to predict the winning team at the beginning of each round. With this model as a basis, they presented a recommender system to guide teams in purchasing equipment. Meanwhile, Ravari *et al.* [22] studied the prediction of match outcome in the videogame Destiny, which combines the genres first person shooter and massively multiplayer online role playing game. The authors create two sets of predictive models, one set predicts the match outcome for each game mode, while the other set predicts the overall match outcome, without considering the game modes. In addition, they also analyze how game performance metrics influence each of the proposed models.

As seen in the related work, in-game approaches generally perform better than pregame approaches. This is because these approaches have much more information than pregame ones (such as experience gained, turrets destroyed, or enemies defeated). Thus, as the game advances, the features allow more accurate prediction of the winning team. Pregame approaches, however, have very limited information. This makes prediction much more complex and it is challenging to obtain models with good accuracy, but it provides useful information before game starts, which can be exploited in recommendation or betting systems. The pregame approaches that achieve the best results, Ong *et al.* [13] and White *et al.* [10], need to obtain additional information, besides the match information, to train their models. Our proposal, that belongs to the pregame approach, does not need additional data about the players or the champions.

IV. Materials and Methods

A. Dataset

The dataset used in this article has been obtained from Kaggle [23]. It consists of observations corresponding to professional games from various leagues and championships played between 2014 and 2018. It includes details regarding both match setup and match progress. The match progress data are discarded as this work is based on prediction before the match starts (pregame). Once cleaned and prepared, the dataset includes 241 teams, 1470 players, and 139 champions. It has 7583 instances, with a binary class variable, indicating the team that wins the game, and 26 features, corresponding to the year, season, league, and type of match, as well as the name of the team playing on each side (2 features, 1 per side), its composition of champions in each of the five roles (10 features, 5 per side), and its composition of players in each of the five roles (10 features, 5 per side). Table I shows the name and content of the class variable and each of these features.

B. Preprocessing

In order to obtain data of quality, adapted to the needs of the models to be trained, thus improving their predictive capacity, a series of preprocessing tasks are applied to the dataset.

For the treatment of missing values an imputation with kNN [24] is performed. This algorithm searches the dataset for

TABLE I
CLASS AND FEATURES OF THE ORIGINAL DATASET

Class	Content
win	Winning team (*BLUE*, *RED*)
Feature	**Content**
League	Matching league
Year	Year played
Season	Season played
Type	Matching type
blueTeamTag	Blue team name
blueTop	Name of the blue player in role top
blueJungle	Name of the blue player in role jungle
blueMiddle	Name of the blue player in role middle
blueADC	Name of the blue player in role ADC
blueSupport	Name of the blue player in role support
blueTopChamp	Name of the blue champion in role top
blueJungleChamp	Name of the blue champion in role jungle
blueMiddleChamp	Name of the blue champion in role middle
blueADCChamp	Name of the blue champion in role ADC
blueSupportChamp	Name of the blue champion in role support
redTeamTag	Red team name
redTop	Name of the red player in role top
redJungle	Name of the red player in role jungle
redMiddle	Name of the red player in role middle
redADC	Name of the red player in role ADC
redSupport	Name of the red player in role support
redTopChamp	Name of the red champion in role top
redJungleChamp	Name of the red champion in role jungle
redMiddleChamp	Name of the red champion in role middle
redADCChamp	Name of the red champion in role ADC
redSupportChamp	Name of the red champion in role support

the "k" nearest neighbors of the observation with the missing value. Once these neighbors are found, their feature values are used to impute the missing value, using, for example, the mean or the mode.

Feature generation [25] is the process of creating new features from one or multiple existing features. These new representations of the dataset make it easier for predictive models to extract useful information, improving their performance. However, the creation of new features can lead to having some of them highly correlated with each other, achieving the opposite effect to the initially pursued objective. For this reason it is appropriate, after performing this task, to do a feature selection that avoids this problem.

The categorical features contained in the dataset must be transformed into numerical features so that some of the algorithms used can handle them correctly. Thus, we choose to one-hot encode them (replacing a categorical feature with n levels by $n-1$ binary features). In this way the binary feature corresponding to the level of the categorical feature in an observation takes the value 1, leaving the others at 0.

To avoid the influence of the scale and variance of the continuous features on their weight in the predictive models, their transformation is necessary. We used a normalization Z-score, i.e., subtracting the mean of each feature value and dividing the result by the standard deviation. Therefore, these features have a mean of 0 and a standard deviation of 1.

C. Feature Selection

In order to create a robust model, it is necessary to identify those features that are highly correlated with the output, since they will be the ones that really provide valuable information to the classification algorithm. Using too many features can sometimes lead to creating a model that overfits the training data and, therefore, reduces its predictive capacity. Furthermore, since the dataset contains a limited number of instances, reducing the number of features simplifies the models, thus requiring fewer observations to achieve good results. In this study, feature selection is performed to obtain a reduced representation of the original dataset that preserves the relevant information contained in it, using a combination of two techniques as follows.

1) *Feature Selection Using Univariate Filtering [26]:* In which the relevance of each feature with respect to the class is evaluated with independence of the predictive models. In this case, a test is performed, using ANOVA [27] as model, to see if the mean of each feature is different between classes. Only the set of features that shows statistically significant differences between classes is finally selected. The main advantage of this technique is its computational efficiency. However, it does not consider possible correlations between features, and the selection it performs is not directly related to the accuracy of the predictive models.
2) *Feature Selection Using Wrapper Methods [26]:* In this kind of selection, a model is trained, for example, using decision trees, to evaluate a subset of features using the value obtained from the chosen metric. To find the finally selected set of features, the sequential backward search [28] will be used. This algorithm first trains the model with the complete dataset to establish a ranking of features according to their importance. It then performs an iterative procedure in which the training is repeated at each step with a subset of features in which the least important ones have been eliminated. This technique usually obtains better results than the filtering technique, since it generates compact subsets of features, optimized for the used metric. However, there is a risk of overfitting, which can be reduced by using cross-validation, and it requires training the models numerous times, making them computationally expensive. An added disadvantage is the need to adjust the hyperparameters of the algorithms used.

D. Classification

For the modeling step, different algorithms have been evaluated that, *a priori*, could give good results for this binary classification problem. Finally, among all the trained models, we will select those that obtain the best results (either individually or as part of meta-models). These algorithms are described as follows.

1) *Extreme Gradient Boosting [29]:* It is an efficient and scalable implementation of decision trees with gradient boosting [30]. It consists in building trees sequentially, so that, in each iteration the tree that minimizes the errors of the previously generated trees is added.
2) *Support Vector Machines [31]:* It seek to define a set of hyperplanes in a space of a certain dimensionality, that separate as best as possible the classes of the output variable. Two variants of this algorithm are applied, the

first with a linear kernel and the second with a radial basis function Gaussian kernel.

3) *Logistic Regression [32]:* It is a statistical model that uses a logistic function to predict the outcome of a binary nominal variable. In this case, two variants are used that provide good results, the generalized linear model with boosting [33] and the generalized linear model with elastic-net regularization [34].
4) *Naive Bayes Classifier [35]:* It is based on the Bayes theorem and on assuming independence between features to calculate the probability that an observation belongs to one class or another.
5) *kNN Algorithm [36]:* It is based on the idea of searching for "k" observations in the training data that have the smallest Euclidean distance to a new observation (nearest neighbors), assigning it the most repeated class of those training observations.
6) *Neural Networks [37]:* They are based on generating an interconnected group of nodes that attempts to imitate the biological behavior of the axons of neurons. Each of these nodes receives as input the output of the nodes of the previous layers multiplied by a weight. These values are aggregated at each receiving node and, optionally, modified or limited by an activation function before propagating the new value to the next neurons. Within the broad field of neural networks, in our work we use a multilayer perceptron (MLP) [38], a kind of fully connected feed-forward neural network composed of multiple layers of perceptrons.

In this work, we also include two meta-models that are interesting for the results obtained. The first one is a model based on label fusion with majority voting [39], while the second one performs stacking [40], where the outputs of a set of classifiers are used as new features to train a new model.

V. Methodology

As mentioned before, in this work we are considering only the state of the game before starting the battle (pregame) to perform the prediction of the winning team in professional games. With this limited information, both in terms of available features and number of instances, it is recommended, during the exploratory analysis, to guide a search process for possible new features that can provide additional relevant information about the class. In the preprocessing of the data, the new candidate features, found in the previous analysis, are created and feature selection is performed to determine which ones are really important, discarding the rest. The final dataset is trained on a set of classifiers. Finally, voting- and stacking-based meta-models are created from the trained models to improve the final accuracy obtained. A diagram of the methodology followed is shown in Fig. 2.

A. Exploratory Analysis

The analysis of the original dataset revealed the need to perform some preprocessing tasks to prepare it to the classification phase, such as the handling of missing values and the one-hot encoding of categorical features.

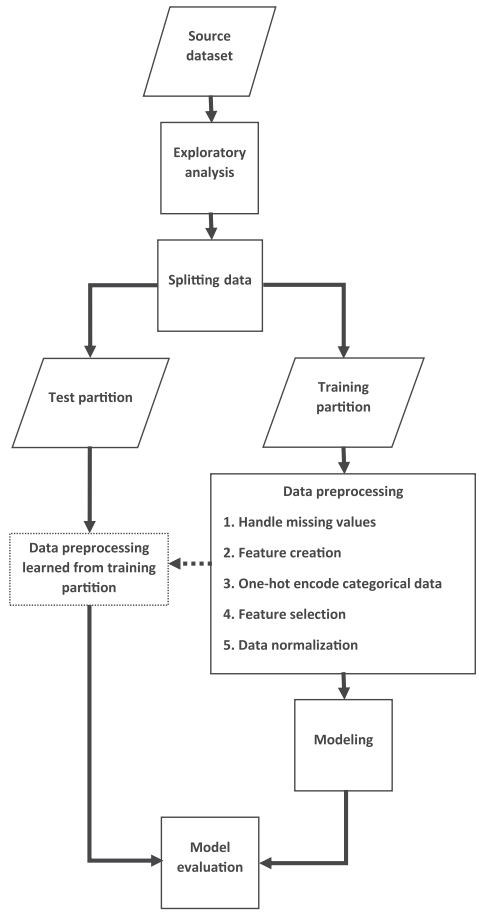

Fig. 2. Schematic diagram of the steps followed in the methodology.

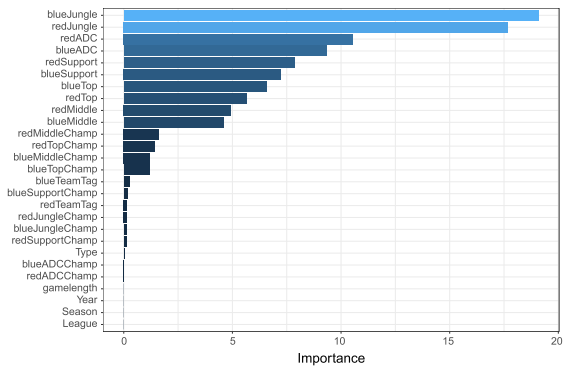

Fig. 3. Features importance of the original dataset.

On the other hand, to guide the search for new features that capture useful information, a supervised learning algorithm based on decision trees is trained with the aim of determining the importance of the features. The obtained ranking (see Fig. 3) shows that the most relevant features with respect to the class correspond to the roles of the players of both teams, followed by those reflecting the selected champion, and finally those referring to the name of the team playing the match. The rest of the features are detected as not important for the outcome.

From these features detected as most relevant, a search for possible new features is carried out. First, the win ratios for each value of these features are calculated (as matches won for that

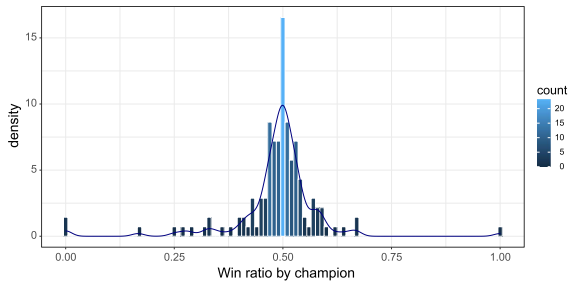

Fig. 4. Distribution of win ratios regarding champions.

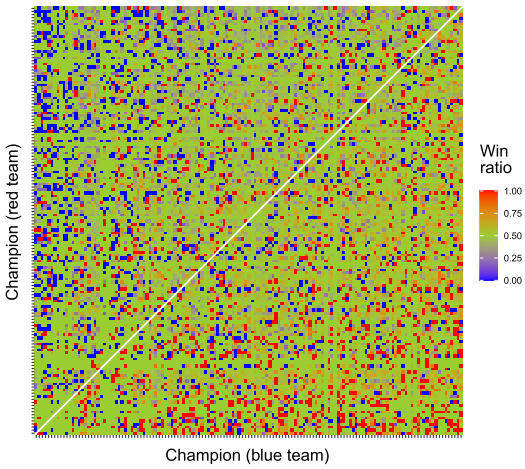

Fig. 5. Win ratio for combination of champions in opposing teams heat map.

value divided by matches played for that value). The distribution of these win ratios is analyzed to determine their possible predictive ability. For example, the distribution of win ratios regarding champions (see Fig. 4) follows a normal distribution, with mean 0.5 and small standard deviation. This indicates that selecting one or other champion, analyzed individually, does not seem to lead to winning the game.

For this reason, and continuing with the search for new features, we opted to explore win ratios, but this time for the combination of those features detected as most relevant. In this case, to generate these win ratios, the number of matches won by a combination of two values from two features is divided by the number of matches played by that combination of two values from those features. For example, for a new possible feature that captures the champion versus champion combination, the number of matches won by a given champion when facing another champion is divided by the number of matches played between both champions. The possible relevance of these combinations with respect to the class can be observed graphically in a heat map. In this case, each axis is one of the features selected from the original dataset, and the value of the win ratio is represented by a color gradient, with blue for values close to 0 and red for values close to 1. For example, the heat map of the champion versus champion combination win ratio (see Fig. 5), which presents the ability of a champion to defeat another champion on the opposing team during the game, shows a significant number of pairs near the extremes, suggesting that it may be interesting to include this information in the dataset due to its ability to discriminate class.

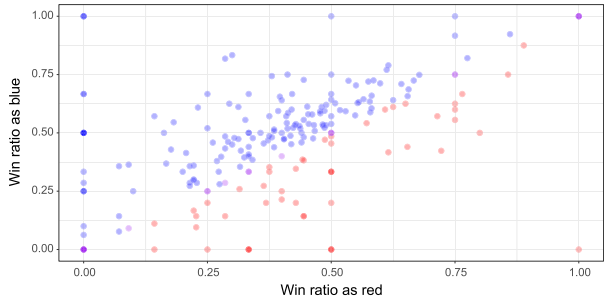

Fig. 6. Win ratio for each team according to the side in which they play (blue/red).

For the sake of brevity, it is not possible to plot all the heat maps, but we concluded that the evaluation focused on different combinations of player-related and champion-related features, produces apparently interesting features that capture the following aspects: The performance of a player leading a certain champion, the performance of a player in a given role, the synergies between players of the same team, the ability of a player to win or lose when facing another player, the performance of a champion in a given role, the synergies between champions on a team, and the ability of a champion to defeat another champion on the opposing team.

Finally, in the case of teams, it is observed that most of them win more games when they are in the blue side (see Fig. 6). For this reason, it is additionally proposed to create a feature containing information about the performance of a team when playing in each of the sides.

B. *Partition and Preprocess*

Before performing any data preprocessing, it is important to split the data into training and test sets. In this way, the independence between the two sets is maintained, which is necessary to perform an honest evaluation of the models.

After data splitting with a 90%/10% ratio, all the preprocessing is carried out using only the data in the training set, and then it is applied to the test set. The first preprocessing task is to handle the missing values, which occur in the feature corresponding to the name of the team in the blue side. Since this feature is categorical, we impute them using the mode of the kNN, with $k = 5$, from the features containing the name of the players of that side, since it is possible that they have played in more occasions together in that team.

Once the missing values have been dealt with, the new features, detected as potentially relevant for the class in the exploratory analysis, are created. To do this, a set of matrices are generated, always from the training set, which collect, for each combination of original features considered, the corresponding winning ratio. For example, the player–champion matrix has as many rows as players and as many columns as champions, and contains the winning ratio for each possible pair of player and champion. Since all matrices are built based on win ratios, those combinations that do not occur in the training partition are assigned the average value between the maximum and the minimum (ratio of 0.5). In this way, it is intended that they do not have a relevant influence on the output. It should be noted that,

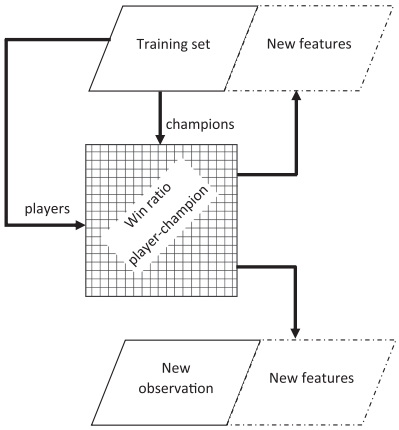

Fig. 7. Schematic diagram of the feature creation process for champion-player.

for the calculation of the ratios of these matrices, the formula "$wins/matches$" has been applied. Using these described matrices it is easy to create new features for an observation, since it is only necessary to extract the value of the ratio by accessing the row and column corresponding to the combined original features. For example, to create the new feature that collects the combination of a given player with the champion he controls, the value contained in the intersection of the row corresponding to that player with the column corresponding to that champion of the player–champion matrix is extracted. A diagram of the full feature creation process for this champion–player combination is shown in the Fig. 7.

In this manner, the values are extracted to create the 38 new features described below. The notation used in these formulas is as follows: *i={top, jungle, middle, ADC, support}*, *j={blue, red}*, *Bp={players in blue team}*, *Rp={players in red team}*, *Bc={champions in blue team}*, and *Rc={champions in red team}*.

1) Ten features, one for each player present in the match, with the winning ratio of each combination of player and role played. Each of these features reflect how effective a player is in a given role

$$\text{playerRole}_{ij} = \frac{\text{wins player }_i}{\text{matches player }_i}.$$

2) Ten features, one for each player present in the game, with the winning ratio of the combination of each player with the champion he controls. It indicates how well a player controls a given champion

$$\text{playerChampion}_{ij} = \frac{\text{wins player champion}}{\text{matches player champion}}.$$

3) Two features, one for each team present in the game, calculated as the sum of the winning ratios of the ten possible combinations of each pair of players when they are on the same team. They capture the synergies between the members of each team

$$\text{coopPlayer}_{blue} = \sum_{m\in Bp}\sum_{n\in Bp,\ m\neq n}\frac{\text{wins}_{mn}}{\text{matches}_{mn}}$$

$$\text{coopPlayer}_{red} = \sum_{m\in Rp}\sum_{n\in Rp,\ m\neq n}\frac{\text{wins}_{mn}}{\text{matches}_{mn}}.$$

4) One feature calculated as the sum of the winning ratios of the 25 possible combinations of each player of the blue team, when facing each of the players of the red team. It determines the superiority of the blue team players against the red team players

$$\text{vsPlayer} = \sum_{m\in Bp}\sum_{n\in Rp,\ m\neq n}\frac{\text{wins}_{mn}}{\text{matches}_{mn}}.$$

5) Ten features, one for each champion present in the game, with the winning ratio of the combination of each champion with the role played. It establishes how effective a champion is in a certain role

$$\text{championRole}_{ij} = \frac{\text{wins champion }_i}{\text{matches champion }_i}.$$

6) Two features, one for each team present in the game, calculated as the sum of the winning ratios of the ten possible combinations of each pair of champions when they are on the same team. They capture the synergies between the champions of the same team

$$\text{coopChampion}_{blue} = \sum_{m\in Bc}\sum_{n\in Bc,\ m\neq n}\frac{\text{wins}_{mn}}{\text{matches}_{mn}}$$

$$\text{coopChampion}_{red} = \sum_{m\in Rc}\sum_{n\in Rc,\ m\neq n}\frac{\text{wins}_{mn}}{\text{matches}_{mn}}.$$

7) One feature calculated as the sum of the winning ratios of the 25 possible combinations of each champion of the blue team, when facing each of the champions of the red team. It shows the superiority of the champions of the blue team against the champions of the red team

$$\text{vsChampion} = \sum_{m\in Bc}\sum_{n\in Rc,\ m\neq n}\frac{\text{wins}_{mn}}{\text{matches}_{mn}}.$$

8) Two features, one for each team present in the game, with the winning ratio of the combination of each team with the side where it plays. It shows how well or poorly a team plays on each side

$$\text{teamColor}_j = \frac{\text{wins team }_j}{\text{matches team }_j}.$$

Once the task of creating new features has been completed, the categorical features are one-hot encoded to convert them into numerical ones.

The next preprocessing performed is feature selection. As a result of the previous operations, the number of features increased to 4257. For this reason, a filter type selection, since it requires less computational resources, is made with ANOVA statistical test and $p-\text{value} = 0.05$ as threshold. In this way, a first subset of 705 features is obtained. Then, and over this subset, a wrapper selection is applied using random forest [41] as training algorithm and recursive backward selection as search algorithm. Thus, the best metric in the cross-validation with repetition is obtained for a set composed of 28 features (see

TABLE II
FINAL SET OF SELECTED FEATURES

Feature	Description
btPlayerRole	w.r.* player in role top blue
bjPlayerRole	w.r. player in role jungle blue
bmPlayerRole	w.r. player in role middle blue
baPlayerRole	w.r. player in role ADC blue
bsPlayerRole	w.r. player in role support blue
rtPlayerRole	w.r. player in role top red
rjPlayerRole	w.r. player in role jungle red
rmPlayerRole	w.r. player in role middle red
raPlayerRole	w.r. player in role ADC red
rsPlayerRole	w.r. player in role support red
btPlayerChampion	w.r. player with champion in role top blue
bjPlayerChampion	w.r. player with champion in role jungle blue
bmPlayerChampion	w.r. player with champion in role middle blue
baPlayerChampion	w.r. player with champion in role ADC blue
bsPlayerChampion	w.r. player with champion in role support blue
rtPlayerChampion	w.r. player with champion in role top red
rjPlayerChampion	w.r. player with champion in role jungle red
rmPlayerChampion	w.r. player with champion in role middle red
raPlayerChampion	w.r. player with champion in role ADC red
rsPlayerChampion	w.r. player with champion in role support red
bCoopPlayer	$\sum$ w.r. pairs blue player with blue player
rCoopPlayer	$\sum$ w.r. pairs red player with red player
bCoopChampion	$\sum$ w.r. pairs blue champion with blue champion
rCoopChampion	$\sum$ w.r. pairs red champion with red champion
vsPlayer	$\sum$ w.r. pairs blue player vs red player
vsChampion	$\sum$ w.r. pairs blue champion vs red champion
bTeamColor	w.r. team in blue side
rTeamColor	w.r. team in red side

*w.r. = win ratio

Table II). All the selected features are from the group of new ones. Among these new features, only the 10 corresponding to how effective a champion is in a certain role have been discarded. Finally, the 28 numerical features are Z-score normalized.

C. Classification

With the preprocessed data, binary classification algorithms are trained to find the patterns present in the training set, allowing generalization to new observations to predict their class. For this training we have selected, from the whole set of algorithms tested in the experimentation, those with the best results obtained, by themselves or in one of the proposed meta-models.

In order to find the hyperparameters for each algorithm, a grid search has been performed. Then, a set of candidate values for the hyperparameters is created. For each possible combination of these values, the model is fitted and the accuracy is estimated using a tenfold cross-validation with five repetitions. Finally, the hyperparameter values with the best metric are chosen. The hyperparameter values chosen for each algorithm can be found in the Appendix.

Among all the trained algorithms based on decision trees, extreme gradient boosting has obtained the best results in this study. This algorithm tends to overfit, so, to avoid this issue, it is especially critical to correctly determine the value of its hyperparameters. Support vector machines performed well in other related work [13]. For this reason, two variants are included in this study. The first one uses a linear kernel and the second variant uses a Gaussian radial basis function kernel with class weights.

As in the previous case, logistic regression is also found in the literature as one of the algorithms with the best predictive capacity for the problem addressed. In this case, the two variants with the best results for this dataset have been selected. One of them is the generalized linear model with boosting. The other variant is the generalized linear model with elastic-net regularization.

The Naive Bayes classifier, despite its simplicity, achieves very good results in this problem. Similarly, a simple classifier such as kNN obtains good accuracy, better than some much more complex algorithms.

Using a neural network, in this case, does not obtain results as good as other models presented above. However, its combination in a meta-model with other classifiers adds enough diversity in the prediction to improve the results obtained. A feedforward network is implemented with one single hidden layer and with dropout to avoid overfitting. The number of hyperparameters to be adjusted is high, which makes the process of finding their optimal values difficult.

Finally, two meta-models (also known as ensembles) are created combining the previous models and improving their results. The first one uses the models obtained with extreme gradient boosting, the Naive Bayes classifier, and the neural network to create a label fusion with majority voting. This simple meta-model has no hyperparameters. In the second meta-model, a stacking is performed, where the output of the Naive Bayes classifier and the neural network is used as new features to train a model based on the extreme gradient boosting algorithm.

VI. RESULTS

A. Experiments Framework

In the experimentation, a balanced random splitting with a 90%/10% ratio is used on the dataset. Thus, the training of the algorithms in the modeling phase is performed using 6826 instances, leaving 757 instances for testing. In this case, a tenfold cross-validation with five repetitions was chosen for training the models. These partitions were created using seeds to ensure reproducibility. The metric used for evaluating all models, including those used in the wrapper feature selection, is accuracy. It should also be noted that the experimentation of this work is done in R programming language [42].

B. Comparative Study

As mentioned before, the test partition was not used for the preprocessing and modeling. This allows an honest estimation of the error to compare the results between classifiers. For this purpose, the accuracy obtained on the test partition for each of the algorithms is computed. Comparing the models (see first column of Table III), it is easy to note how the meta-models improve the results of the other ones. In particular, the meta-model based on stacking has the highest predictive capacity of the models evaluated.

However, model performance in terms of computation times could be a critical factor for practical application of classifiers (e.g., for real-time use during the champion and role selection phase). In that case, if the computation times employed by the algorithms for this dataset with new features and feature selection are compared (see columns fourth and fifth in Table III), a good choice would be the Naive Bayes classifier. This model

TABLE III
MODEL ACCURACY ON TEST PARTITION ACCORDING TO THE DATASET USED FOR TRAINING

Model	NF + FS	NF	Original dataset	Training time (s)	Prediction time (s)*
nb	**0.6896**†	0.6830†	0.5020	19.45	**0.05**
xgboost	**0.6882**†	0.6711	0.6275	830.85	0.17
svmLinear	**0.6777**†	0.6764†	0.6341	78.60	0.35
knn	**0.6658**†	0.6579	0.6129	**18.65**	0.22
glmnet	**0.6605**†	**0.6605**†	0.6565	30.85	0.08
glmboost	**0.6579**†	**0.6579**†	0.6037	344.55	0.25
svmRadial	**0.6407**†	0.6380†	0.5839	401.70	0.47
nnet	0.6407	**0.6777**†	0.6486	152.50	0.17
Meta-model					
xgbStack	**0.7041**†	0.6935	0.6116	994.30	0.41
voteStack	**0.6975**	0.6922	0.6486	1002.80	0.45

NF = New features, FS = Feature selection.
*For the full test partition (757 instances).
†For the same algorithm, the accuracy test results are not significantly worse than the best. The bold values in first three columns represents the best accuracy for each row (method). The bold values in last two columns (training time and prediction time) is the lowest time among all the methods.

TABLE IV
COMPARISON OF THE PROPOSED MODELS RESULTS WITH THOSE EXISTING IN THE LITERATURE

Algorithm used	Training instances	Test accuracy
RNN + LR [White et al.]	70,194	0.7210
xgbStack	6826	0.7041
k-means + SVM [Ong et al.]	117,000	0.7040
voteStack	6826	0.6975
nb	6826	0.6896
xgboost	6826	0.6882
svmLinear (SVM)	6826	0.6777
glmnet (LR)	6826	0.6605
SimpleRNN (0-5 minutes) [Silva et al.]	5107	0.6391
NeuralAC [Gu et al.]	603,760	0.6209
LR [Chen et al.]	208,090	0.6024

has a slightly lower accuracy than the meta-models, but requires less than 1/8 of time to compute a prediction.

It is also interesting to evaluate the impact on the models accuracy, both of the new features created and of the feature selection. For this purpose, the algorithms are retrained, setting again the best hyperparameters, with two new datasets. The first one is obtained by applying, on the original dataset, all the preprocessing tasks indicated in this methodology, except the one related to feature selection. The second is the original dataset, with only the preprocessing essential tasks for the correct working of the classification algorithms (imputation of missing values and one-hot encoding for the qualitative features). Comparing the results obtained on the test partition of each of the three datasets (see Table III), the best accuracy is obtained with those datasets containing the new features created. Among them, the one with a feature selection has better metrics in almost all the models (9 of the 10 finally chosen), while the one that does not include feature selection only achieves the same results in the algorithms based on logistic regression, and only outperforms it in the neural network. This drop in accuracy can be explained because sometimes a neural network trained with irrelevant features is more flexible than the network after feature selection, which can lead to better results [43]. To verify if there are significant differences between the accuracy values obtained on each dataset, it is necessary to carry out a statistical analysis of the results.[1] A Wilcoxon signed-rank test [44] is performed by comparing these results across all cross-validation folds and repetitions of the same algorithm trained on each of the three datasets. The statistical tests (see Table III) reveal how the datasets with the new features (with and without feature selection), besides showing the best metrics in all models, are significantly better than those obtained with the original dataset. It should be noted that the majority voting meta-model has not been tested, because it is not really trained (the majority outcome is chosen from a set of classifiers), and, therefore, cross-validation folds are not available. Nevertheless, these results highlight the importance of these two preprocessing tasks and the stacking-based meta-model, the combination of which improves the predictive capacity of this proposal, increasing it by around 5% when it is compared with the results for the best model trained on the original dataset.

Among all the related literature, only Silva *et al.* [4] used the same dataset as the one we use. Nevertheless, their work takes into account in-game information. For this reason, and to make a comparison as honest as possible with this approach, their result for the interval [0–5] minutes of gameplay is considered. On the other hand, our work could be compared with the approaches available in the literature that address the problem knowing only the pregame information. However, in these approaches there are no standard datasets due to the notable variations the game suffers with each update, so this comparison is not fair. Thus, each article uses its own dataset, extracted from the source the authors have considered, and with its particular features. In addition, the works that obtain the best accuracy, need to extract extra information, such as statistics of players and/or champions involved, from additional sources. For example, White *et al.* [10] use a dataset containing 87 743 matches from various levels of play, not only from the professional, obtained from the game API but also extract a set of statistics of all players and champions disputing those matches from OP.GG Website [11] and CHAMPION.GG Website [12]. Comparing the results of these works with the ones obtained by our proposal (see Table IV), we can see how our models outperform the in-game model that uses the same dataset, even though it uses information from the first 5 min of the match. In addition, for pregame approaches our meta-models obtain an accuracy similar to those of the state-of-the-art, despite having a much smaller number of observations for training, and without the need to use additional sources to obtain statistics of the participants.

The most important features for determining the class in the meta-model with the best accuracy in this article can be seen in Fig. 8. It seems that, in this case, the relevant features are those that determine the superiority of players and champions of the blue team over those of the red team. These are followed by the features that indicate how well a player controls a given champion. Next are the features that capture synergies, both between champions on the same team and between players on the same team. Finally, and with similar importance to each other, are the features that indicate how effective a player is in

[1]Standard deviations for each model and dataset. [Online]. Available: https://github.com/JuAGarHi/ML-methods-predicting-LoL-outcome/blob/main/models-datasets-standard-deviations.md

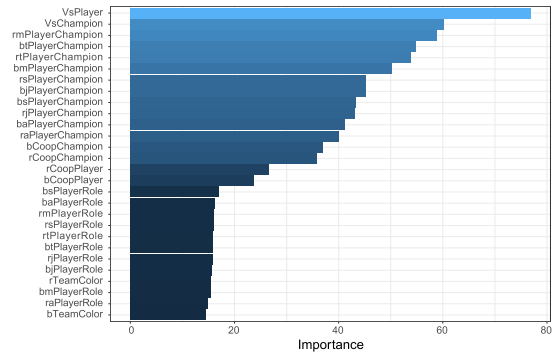

Fig. 8. Ranking of feature importance in the stacking-based meta-model.

a certain role and those that capture how well or poorly a team plays on each side.

VII. Conclusion

In this article we have seen how it is possible to create a classifier for determining the winning team in professional *LoL* games that, with a limited number of observations and features, obtains good results, in line with those offered by other approaches that have datasets with tens of thousands of instances and with additional information on player performance. To this end, this proposal has been based on the search for and creation of new features that provide additional relevant information to the classifier. To the best of our knowledge, this is the first attempt in the literature to create new features (without relying on external sources) to improve the prediction of the winner in *LoL*. These new features have been subjected, together with the original ones, to a feature selection process. The result has been used as a final dataset to train different models and assemble a selection of them into a meta-model. The best results were obtained by the meta-model based on stacking, achieving accuracy over 70%. This result is comparable to other approaches from the state-of-the-art, but with the added benefit of using few samples and not requiring the use of external sources to collect additional statistics.

The approach presented in this article can be considered as a proof of concept for its application in other videogames (not exclusively in MOBA) or even sports. If the particularities of the dataset allow combining features to create new ones based on win ratios, our methodology can be applied to predict the match outcome.

This proposal opens the door to different ways for future development. The first one is to increase the number of observations in the dataset. Getting more observations would allow studying how it impacts the predictive capacity of models. It is also interesting to get information about when each match was played. Having temporal information offers a whole range of possibilities, both for the creation of new features that exploit this information (e.g., trends in players to detect periods in which the player performs more or less), and for using classification algorithms that consider the time factor, such as recursive neural networks. Another development option is the implementation of a recommendation system for professional teams. This system would advise about which team members should participate, what role they should play, or which champion they should select, according to the configuration of the opposing team.

Appendix
Classifier Hyperparameter Values

Extreme Gradient Boosting
1) Number of trees to be adjusted, nrounds $= 14\,000$;
2) maximum depth of each tree, maxdepth $= 2$;
3) reduction of the step size in each update, eta $= 1e-6$;
4) reduction of minimum loss required to perform a new partitioning of a node, gamma $= 0$;
5) ratio of subsample of columns for tree construction, colsamplebytree $= 1e-9$;
6) minimum of the sum of the instance weight required on a child node, minchildweight $= 1$; and
7) ratio of data to be used to generate the trees, subsample $= 1e-1$.

SVM Linear Kernel:
1) Cost for hyperplane margin infringement, $C = 5e-5$.

SVM RBF Kernel With Class Weights:
1) Radial kernel coefficient, sigma $= 2.8657e-4$;
2) cost for hyperplane margin infringement, $C = 0.25$; and
3) class weight, Weight $= 3$.

Generalized Linear Model With Boosting:
1) Initial number of boosting iterations, mstop $= 200$.

Generalized Linear Model With Elastic-Net Regularization:
1) Penalty mix ratio for elastic-net, alpha $= 0.55$; and
2) penalty value, lambda $= 0.09$.

Naive Bayes Classifier:
1) Laplace smoothing value, laplace $= 0$;
2) use of a function to estimate conditional densities for the class of each feature, usekernel $=$ TRUE; and
3) adjustment value for the function of the previous hyperparameter, adjust $= 8$.

k-Nearest Neighbors:
1) Number of neighbors, $k = 16$.

Neural Network (MLP):
1) Number of neurons in the hidden layer, size $= 256$;
2) ratio of information to be discarded after each layer, dropout $= 0.4$;
3) batch size used in each iteration, batch size $= 2275$;
4) learning rate, $lr = 3e-4$;
5) gradient decay value, rho $= 0.9$;
6) learning rate decay value, decay $= 0.2$; and
7) activation function used in neurons, activation $=$ tanh.

Meta-Model With Extreme Gradient Boosting Algorithm:
1) Number of trees to be adjusted, nrounds $= 14\,000$;
2) maximum depth of each tree, maxdepth $= 2$;
3) reduction of the step size in each update, eta $= 1e-6$;
4) reduction of minimum loss required to perform a new partitioning of a node, gamma $= 0$;
5) ratio of subsample of columns for tree construction, colsamplebytree $= 1e-9$;
6) minimum of the sum of the instance weight required on a child node, minchildweight $= 1$; and
7) ratio of data to be used to generate the trees, subsample $= 0.15$.

REFERENCES

[1] "Global Esports market report," NEWZOO, Amsterdam, The Netherlands, 2020.

[2] Riot Games, League of Legends, Accessed: Jan. 2022. [Online]. Available: https://www.leagueoflegends.com

[3] Riot Games, "League of Legends: How to play—Lane positions." Accessed: Jan. 2022. [Online]. Available: https://www.leagueoflegends.com/en-us/how-to-play/

[4] A. L. S. Cardoso, G. P. Lobo, and L. Chaimowicz, "Continuous outcome prediction of League of Legends competitive matches using recurrent neural networks," in *Proc. SBCGames*, 2018, pp. 2179–2259.

[5] D.-H. Kim, C. Lee, and K.-S. Chung, "A confidence-calibrated MOBA game winner predictor," in *Proc. IEEE Conf. Games*, 2020, pp. 622–625.

[6] L. C. Kho, "Logic mining in league of legends," *Pertanika J. Sci. Technol.*, vol. 28, no. 1, pp. 211–225, 2020.

[7] A. R. Novak, K. J. Bennett, M. A. Pluss, and J. Fransen, " Performance analysis in Esports: Modelling performance at the 2018 League of Legends world championship," *Int. J. Sports Sci. Coaching*, vol. 15, no. 5/6, pp. 809–817, 2020.

[8] D.-K. Kang and M.-J. Kim, "Poisson model and Bradley terry model for predicting multiplayer online battle games," in *Proc. 17th Int. Conf. Ubiquitous Future Netw.*, 2015, pp. 882–887.

[9] Z. Chen, Y. Sun, M. S. El Nasr, and T. -H. D. Nguyen, "NeuralAC: Learning cooperation and competition effects for match outcome prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4072–4080.

[10] A. White and D. M. Romano, "Scalable psychological momentum forecasting in Esports," in *Proc. Workshop: State-based User Model., 13th ACM Int. Conf. Web Search Data Mining*, 2020. [Online]. Available: https://www.k4all.org/event/wsdmsum20/

[11] *LoL Stats*. Accessed: Jan. 2022. [Online]. Available: https://www.op.gg

[12] *LoL Champions Stats*. Accessed: Jan. 2022. [Online]. Available: https://www.champion.gg

[13] H. Y. Ong, S. Deolalikar, and M. Peng, "Player behavior and optimal team composition for online multiplayer games," 2015, *arXiv:1503.02230*.

[14] Z. Chen, Y. Sun, M. S. El Nasr, and T. -H. D. Nguyen, "Player skill decomposition in multiplayer online battle arenas," 2017, *arXiv:1702.06253*.

[15] K. Conley and D. Perry, "How does he saw me? A recommendation engine for picking heroes in Dota 2," Stanford Univ., Stanford, CA, USA, 2013.

[16] A. Agarwala and M. Pearce, "Learning Dota 2 team compositions," Stanford Univ., Stanford, CA, USA, 2014.

[17] V. J. Hodge, S. D. Michael, N. S. John, F. B. Oliver, P. C. Ivan, and A. Drachen, "Win prediction in multi-player Esports: Live professional match prediction," *IEEE Trans. Games*, vol. 13, no. 4, pp. 368–379, Dec. 2021.

[18] X. Lan, L. Duan, W. Chen, R. Qin, T. Nummenmaa, and J. Nummenmaa, "A Player behavior model for predicting win-loss outcome in MOBA games," in *Proc. Int. Conf. Adv. Data Mining Appl.*, 2018. pp. 474–488.

[19] A. A. Sánchez-ruiz and M. Miranda, "A machine learning approach to predict the winner in StarCraft based on influence maps," *Entertainment Comput.*, vol. 19, pp. 29–41, 2017.

[20] M. M. Mamulpet, "Pubg winner placement prediction using artificial neural network," *Int. J. Eng. Appl. Sci. Technol.*, vol. 3, no. 12, pp. 107–118, 2019.

[21] P. Xenopoulos, B. Coelho, and C. Silva, "Optimal team economic decisions in counter-strike," 2021, *arXiv:2109.12990*.

[22] Y. N. Ravari, P. Spronck, R. Sifa, and A. Drachen, "Predicting victory in a hybrid online competitive game: The case of destiny," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2017, pp. 207–213.

[23] Kaggle, "League of Legends - competitive matches 2014–2018." [Online]. Available: https://www.kaggle.com/chuckephron/leagueoflegends

[24] O. Troyanskaya *et al.*, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[25] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[26] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proc. 11th Int. Mach. Learn. Conf.*, 1994, pp. 121–129.

[27] P. Jafari and F. Azuaje, "An assessment of recently published gene expression data analyses: Reporting experimental design and statistical factors," *BMC Med. Inform. Decis. Making*, vol. 6, no. 1, 2006, Art. no. 27.

[28] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, no. 1–3, pp. 389–422, 2002.

[29] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[30] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, 2001, pp. 1189–1232.

[31] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[32] D. G. Kleinbaum, K. Dietz, M. Gail, and M. Klein, *Logistic Regression*. New York, NY, USA: Springer-Verlag, 2002.

[33] P. Bühlmann and B. Yu Bin, "Boosting with the L. 2 loss: Regression and classification," *J. Amer. Statist. Assoc.*, vol. 98, no. 462, pp. 324–339, 2003.

[34] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. Stat. Softw.*, vol. 33, no. 1, pp. 1–22, 2010.

[35] I. Rish, "An empirical study of the naive Bayes classifier," in *Proc. Workshop Empirical Methods Artif. Intell.*, 2001, pp. 41–46.

[36] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[37] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1995.

[38] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numerica*, vol. 8, pp. 143–195, 1999.

[39] L. Lam and S. Y. Suen, "Application of majority voting to pattern recognition: An analysis of its behavior and performance," *IEEE Trans. Syst., Man, Cybern.-Part A: Syst. Humans*, vol. 27, no. 5, pp. 553–568, Sep. 1997.

[40] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[41] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[42] "The R project for statistical computing," The R foundation, Vienna, Austria. Accessed: Jan. 2022. [Online]. Available: https://www.r-project.org

[43] E. Romero and J. S. María, "Performing feature selection with multilayer perceptrons," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 431–441, Mar. 2008.

[44] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, no. 6, pp. 80–83, 1945.

Juan Agustín Hitar-García received the B.S. degree in industrial engineering from the Polytechnic University of Valencia, Valencia, Spain, 2001, the M.S. degree in artificial intelligence research from the Menéndez Pelayo International University, Madrid, Spain, 2020. He is currently working toward the Ph.D. degree with the University of A Coruña, A Coruña, Spain, and his thesis focuses on explainable artificial intelligence.

He has worked in the private sector, in recent years as chief operating officer.

Laura Morán-Fernández received the B.S. and Ph.D. degrees in computer science from the University of A Coruña, A Coruña, Spain, 2015 and 2020, respectively.

She is currently an Assistant Lecturer with the Department of Computer Science and Information Technologies, University of A Coruña. She has coauthored three book chapters, and more than 15 research papers in international journals and conferences. Her research interests include machine learning, feature selection, and big data.

Verónica Bolón-Canedo received the B.S. and Ph.D. degrees in computer science from the University of A Coruña, A Coruña, Spain, in 2009 and 2014, respectively.

After a Postdoctoral Fellowship with the University of Manchester, Manchester, UK, in 2015, she is currently an Associate Professor with the Department of Computer Science and Information Technologies, University of A Coruña. She has authored or coauthored extensively in the area of machine learning and feature selection. She has coauthored two books, seven book chapters, and more than 80 research papers in international conferences and journals, on these topics.