Using machine learning for predicting League of Legends match outcome

Jakob Maležič
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
Email: jm6421@student.uni-lj.si

***Abstract*—We use decision trees and random forest to predict the match outcomes of the popular online multiplayer game *League of Legends*. Features that are used for creation of decision trees are extracted from a Kaggle dataset that contains 108,000 ranked games from Korea. The data encapsulates champion picks, participants' usernames, players' roles and in-game players' statistics. We preprocess the data in such way, that each case contains data about participants of one team by their role. By doing so, our dataset contains 156344 cases. We find out, that we can very accurately predict outcome of a match with provided data.**

I. Introduction

League of Legends is a multiplayer online battle arena (MOBA) game developed by Riot Games. Players are matched with 4 teammates and play head-to-head versus enemy team, which consists of 5 players as well. The goal of the game is to destroy the enemy base and their main structure, called The Nexus. Each participant controls a unique champion chosen from a pool of more than a hundred different champions that differ by their characteristics and abilities. It is currently the world's most popular video game with over 100 million active monthly players and a vast and widely competitive e-sports scene. Because of a such huge community, predicting match outcomes for casual players as well as tournament games is very interesting and could be very valuable to players and fans.

A. Game Overview

In a classic *League of Legends* game, two teams of five players, called summoners, battle on a map called Summoner's rift. Contrary to MMORPG (Massively Multiplayer Online Role-Playing Game) games, there is no unit construction nor there is a massive number of players on a map at a time in a MOBA game. Much of the strategy revolves around cooperative team play and individual performance.

The map is split into three lanes: top lane, mid lane and bottom lane, as shown in picture 1. Between the lanes there is jungle with neutral monsters. Map is split by a river in the middle.

Each player chooses a unique champion from a pool of over one hundred different champions. Champions differ by their characteristics and abilities. Some champions are faster, some provide utility, some provide sustain, some are close range fighters some are long ranged etc. Each champion posses 4 different abilities unique for the champion.

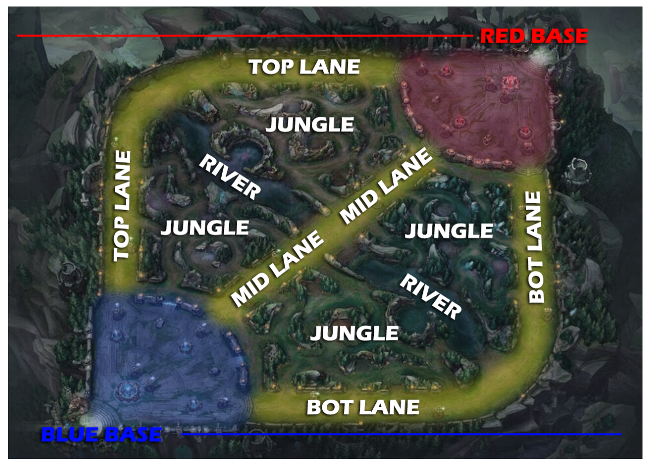

Fig. 1. Summoner's rift.

Each player is assigned a different role before the game starts. There are five different roles: top, jungle, mid, bottom and support (also called utility). This plays an important role on player's choosing of a champion, since champions are categorized by roles as well. A champion that does well on top lane does not necessarily do well in the bot lane. This is important, because players usually tend to specialize (main) one role and sometimes even only one champion (such player can be called one-tricks).

Champions are chosen in the draft phase, as shown in picture 2, before the start of the game. Each of the players first bans one champion from the game. Banned champions cannot be played by either of the teams. Players then alternately pick their champions. After everyone has picked, there is a 30 second countdown till the start of the game. In this time, any of the players can choose to leave the game. This is also called dodging, because the player has dodged the game. By doing this, he cancels the match and players need to find new match to play.

League of legends has a ranking system, matching the players of a similar skill level to play with and against each other. It comprises nine tiers which indicate the skill level of players. Players within each division are ranked using a system of points called League Points (LP).

Each tier from Iron to Diamond is divided into four divi-

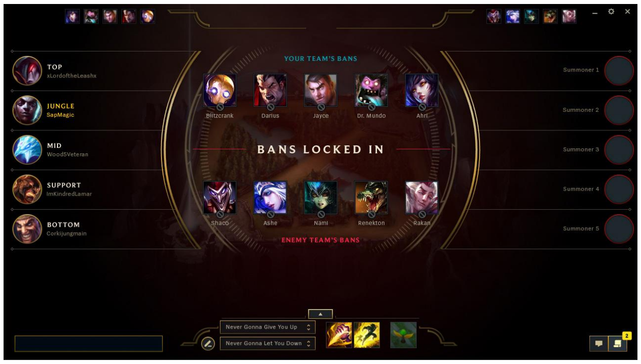

Fig. 2. Draft phase.

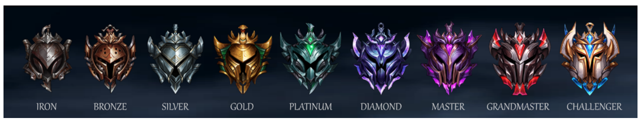

Fig. 3. Ranking system - tiers.

sions, depicted by a roman numeral starting from IV (4 being the lowest) to I (1 being the highest). Master tier and higher do not have divisions, they are instead exclusively reliant on LP and the population of players within these rank tiers. Tiers are shown in picture 3.

The player earns League Points when they win ranked games and lose them when they lose ranked games. The amount earned or lost depends on the player's hidden Match Making Rating (MMR). The higher the MMR, the more LP earned per win and the less LP lost per loss [1].

The penalty for leaving the game in the draft phase is losing some amount of League Points, however, the Match Making Rating (MMR) stays the same. This makes it very valuable to dodge games, if we can predict a loss before the game starts.

II. Related work

Since *League of Legends* is a popular game, there have already been a couple of researches that tried to predict the match outcome.

Lucas Lin used gradient boosted trees and gradient boosted trees with logistic regression. He used one-hot encoding to transform data to a feature vector. By using these methods, he tried to predict the outcome of a match based on in-game statistics. He described this in his article *League of Legends Match Outcome Prediction* [2].

Ani R, Vishnu Harikumar, Arjun K Devan, O.S. Deepa tried to predict match outcome by doing feature selection and ensemble models of classification algorithms. Their work is presented in the article *Victory prediction in League of Legends using Feature Selection and Ensemble methods* [3].

Yifan Yang, Tian Qin and Yu-Heng Lei tried to predict match outcome of a similar game, called Dota 2. They used pre-match features as well as real-time (during-match) features at each minute as the match progressed. They used logistic regression, Attribute Sequence Model and their combinations as the prediction models. They presented this in their article *Real-time eSports Match Result Prediction* [4].

Kenneth T. Hall used deep neural networks, based upon the historical game play statistics of match participants for predicting the winning team of the match. His most successful network achieves greater than 80 percent win prediction accuracy on testing data. His work is described in article *Deep Learning for League of Legends Match Prediction* [5].

III. Data collection

Riot Games provide a public API for accessing historical and live data of League of Legends matches. If we want to get statistic data for a summoner we need his summoner name. We then need to make a request to the API, to get summoner's id. With this id, we can then get list of summoner's match references, 100 per request. Each match reference contains match id which references one particular match. We then need to make another request to get all data about this particular match. The data contains participant identities and statistics for every participant in that match. This statistic encapsulates data like: kills, assists, deaths, damage per minute, gold per minute, vision score, wards bought, champion etc. This is exactly the data that we need for building a model for predicting match outcome.

However, Riot API only allows us to make 50 request per minute. That means, we can't get historical data, since that requires us to make a request for each match a player has played. Furthermore, we need to do that for each player of the team. That's why, we decided to build a model on dataset from Kaggle. This dataset consists of 108941 games from master, grandmaster and challenger players from Korea.

Since the model will be build only on games from high ranked games from Korea, we need to keep in mind, that the model might not work as well in different region and rank. However, we decided it would suffice for this article.

IV. Data processing

The data we got from Kaggle is split into three files:

- *match_data* - contains all statistical data about matches
- *match_winner_data* - contains match references of winning matches
- *match_loser_data* - contains match references of losing matches

The statistical data about a match contains data for the winning and the losing team. This means, that the *match_winner_data* references the same games as the *match_loser_data*. This is the reason why we only need to use one table of match references and not both.

First, we read *match_data* and *match_winner_data* in Python[1] using Pandas[2] library and merge match references with match data. By doing this, we created a table of 108941 rows, where each row contains all statistical data about specific

[1]https://www.python.org/
[2]https://pandas.pydata.org/pandas-docs/stable/index.html

match. We then sorted this table, by match's timestamp. This is important, to preserve the actual sequence of the matches.

We then loop through the matches and extract the features we are interested in. Since we want each our case to presents one team of the match, we need features for each player of the team in the same case. Because the columns of the row always need to be in the same order, we sorted players by their role: top, jungle, mid, bottom and support. We did this, by using a python library Cassiopeia[3], which uses champion's statistical data to decide player's role. The features, that we extracted for each participant are:

- **Total games** - the amount of matches played.
- **Wins** - the amount of matches won.

And the features we extracted for each participant's roles are:

- **Kills** - the amount of enemy champions killed in a match. This feature is important, because killing enemy champions grant the player extra gold, which is used for buying items and therefore getting and edge over the enemy. This feature together with deaths and assists forms the kills, deaths and assists ratio (KDA) which is the most often used statistic in statistical analysis of League of Legends matches. It's the quickest way of knowing whether or not the player played well.
- **Deaths** - the number of times the player got killed by the enemy. This feature is the inverse of the kills feature, since getting killed by the enemy, gives gold to the enemy player.
- **Assists** - the number of times the player assisted at killing an enemy champion. Similarly to kills feature, this feature is important, because assisting at killing an enemy champion grants gold to the player.
- **Gold earned** - the total amount of gold earned in a game by the player.
- **Total damage dealt to champions** - the amount of damage dealt to enemy champions.
- **Total minions killed** - the number of minions killed in the match. Minions are units that comprise the main force sent by the Nexus. They spawn periodically from their Nexus and advance along a lane towards the enemy Nexus, automatically engaging any enemy unit or structure they encounter. They are controlled by artificial intelligence and only use basic attacks [6]. The feature is important because, similarly to kills, killing a minion gives gold to the player. Twelve minion kills equals to one champion kill.
- **Vision score** - stat that indicates how much vision a player has influenced in the game, this includes the vision they provided and denied. League of Legends uses a fog of war which only reveals terrain features and enemy units through a player's reconnaissance [7]. Without a unit actively observing, previously revealed areas of the map are subject to a shroud through which only terrain is visible, but changes in enemy units are not.

[3]https://github.com/meraki-analytics/cassiopeia

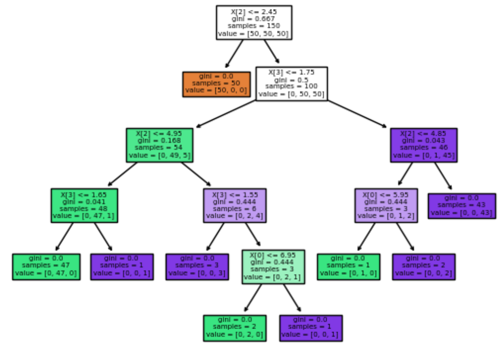

Fig. 4. Decision tree example.

- **Vision wards bought** - the number of control wards bought in a match. Control wards are consumable items in League of Legends. Consumable items are one-time use items that disappear or lose effect after the first use, and can be used again only by buying them again [8]. Control wards grant sight over the surrounding area and reveal enemy wards and hidden traps.
- **Total games** - the amount of matches played with a given role.
- **Wins** - the amount of matches won with a given role.

We picked these features by our domain knowledge and statistical data, that is usually shown in professional e-sports broadcasts. In conclusion, each our case now has 260 features.

V. Methodology

We tested three different machine learning classifiers... **TODO**

A. *Naive Bayes*

B. *Decision trees*

A decision tree is a classifier and it consists of nodes that form a rooted tree, meaning it is a directed tree with three types of nodes:

- **root** - a node that has no incoming edges, there is exactly one such node in a tree
- **internal or test node** - a node with exactly one incoming edge and outgoing edges
- **leaf or terminal node** - a node with exactly one incoming edge and no outgoing edges

In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attributes values. Each leaf is assigned to one class representing the most appropriate target value [9]. Picture 4 shows an example of a decision tree.

Induction of an optimal decision tree from a given data is considered to be a hard task. It has been shown that finding a minimal decision tree consistent with the training set is a

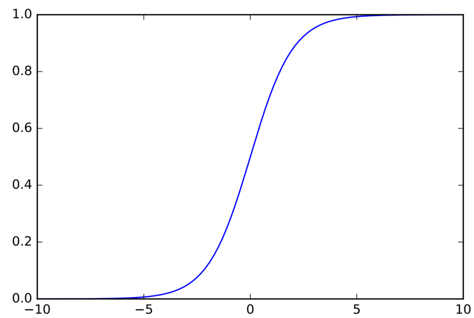

Fig. 5. Logistic function.

NP-hard. Because of this, heuristic methods have to be used for solving the problem.

In most of the cases, the functions in internal nodes are univariate, meaning that an node is split according to the value of a single attribute. Consequently, the tree building algorithm searches for the best attribute upon which to split. There exist many univariate criteria, these are some of them:

- **Information Gain** - an impurity-based criterion that uses the entropy measure as the impurity measure (Quinlan, 1987).
- **Gini Index** - an impurity-based criterion that measures the drivergences between the probability distributions of the target attribute's values.
- **Gain Ratio** - the gain ratio "normalizes" the information gain as follows:

$$\text{GainRatio}(a_i, S) = \frac{\text{InformationGain}(a_i, S)}{\text{Entropy}(a_i, S)} \tag{1}$$

C. Logistic regression

Logistic regression is a statistical model that in its basic form uses a logistic function, picture 5, to model a binary dependent variable. However, unlike linear regression the response variables can be categorical or continuous, as the model does not strictly require continuous data, as seen on picture 6. To predict group membership, logistic regression uses the log odds ratio rather than probabilities and an iterative maximum likelihood method rather than a least squares to fit the final model. Because of this, the method is more appropriate for non-normally distributed data.

D. Random forest

Random forests are a scheme for building a classification ensemble with a set of decision trees that grow in randomly selected subspaces of data [10]. Experimental results have shown that random forest classifiers can achieve high accuracy in classifying data in domains of high dimensions with many cases [10], [11].

Since our data consists of huge number of features and cases, we tried to use random forest for predicting the match outcome.

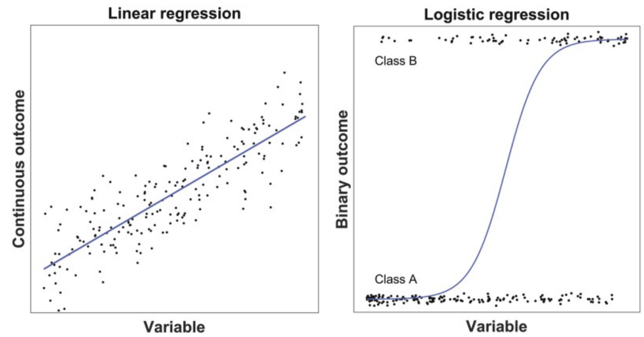

Fig. 6. Linear regression compared to logistic regression.

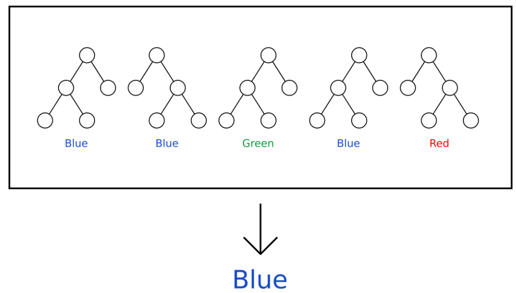

Fig. 7. Random forest example.

E. K-Fold Cross Validation

Cross-validation is a resampling procedure used for evaluating machine learning models. The procedure takes one parameter k that refers to the number of groups that a given data sample is to be split into. That is why, it's often called k-fold cross validation.

It is mainly used to evaluate the machine learning model on unseen data. That is, to build a model on a limited sample of data in order to estimate how well does it perform in general, when the model is built on data that is not used during the training phase. The general procedure is as follows:

- Shuffle the dataset randomly.
- Split the dataset into k groups, also called folds.
- For each group:
  - Take the group as test data set
  - Take the remaining groups as training data sets
  - Fit the model on the training data sets and evaluate it on the test set
  - Retain evaulation score and discard the model
- Summarize the score of the model using the sample of model evaluation scores

F. Confusion matrix

Confusion matrix is a matrix that helps us evaluate the accuracy of classification. It enables us to visualize the performance of an algorithm, usually a supervised learning one. Each row in the confusion matrix represents an instance of predicted class while each column represents the instance of the actual class [12].

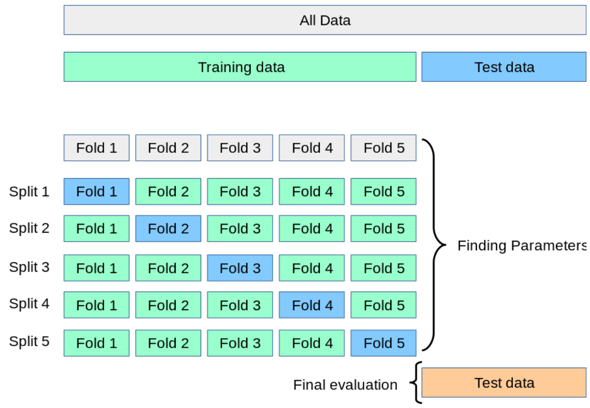

Fig. 8. K-Fold Cross Validation.

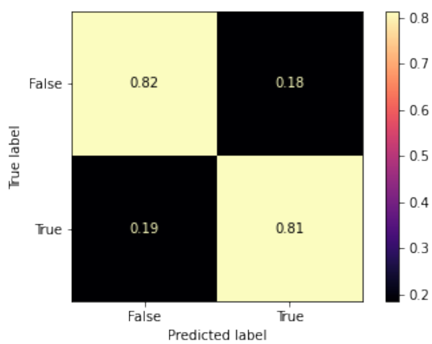

Fig. 9. Random forest confusion matrix.

In our case, the target class is a binary value: true, if the match outcome is win and false if the match outcome is loss. Therefore, all our confusion matrices will be size 2 by 2:

- **true positive** - When prediction was a win and the actual result was also a win.
- **true negative** - When prediction was a loss and the actual result was also a loss.
- **false positive** - When prediction was a win, but the actual result was a loss.
- **false negative** - When prediction was a loss, but the actual result was a win.

In our case, the value written in each square shows a normalized number of cases of that instance.

VI. Results

VII. Discussion

VIII. Future work

IX. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

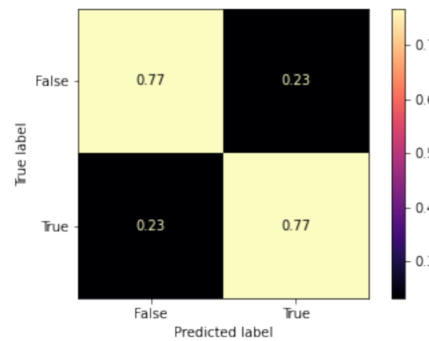

Fig. 10. Decision tree confusion matrix.

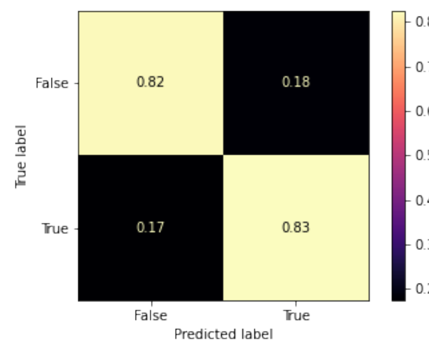

Fig. 11. Logistic regression confusion matrix.

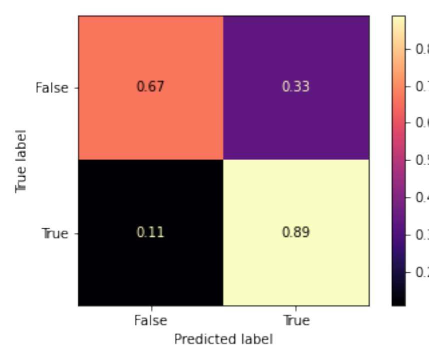

Fig. 12. Naive Bayes confusion matrix.

References

[1] RiotGames, *Rank*, 2020 (accessed December 30, 2020), https://leagueoflegends.fandom.com/wiki/Rank.
[2] L. Lin, "League of Legends Match Outcome Prediction," 2016.
[3] R. Ani, V. Harikumar, A. K. Devan, and O. Deepa, "Victory prediction in League of Legends using Feature Selection and Ensemble methods," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, no. Iciccs. IEEE, may 2019, pp. 74–77.
[4] Y. Yang, T. Qin, and Y.-H. Lei, "Real-time eSports Match Result Prediction," *arXiv*, no. Nips, pp. 1–9, dec 2016.
[5] K. T. Hall, *Deep Learning for League of Legends Match Prediction*, 2020 (accessed December 30, 2020), https://github.com/minihat/LoL-Match-Prediction/blob/master/FINAL%20REPORT.pdf.

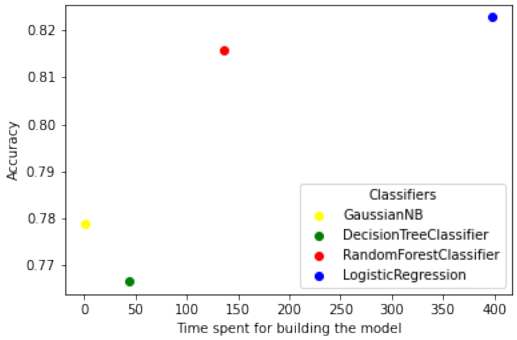

Fig. 13. Methods compared by time and accuracy.

[6] RiotGames, *Minions*, 2020 (accessed December 30, 2020), https://leagueoflegends.fandom.com/wiki/Minion.

[7] Wikipedia, *Fog of war*, 2020 (accessed December 30, 2020), https://en.wikipedia.org/wiki/Fog%20of%20war.

[8] RiotGames, *Consumable item*, 2020 (accessed December 30, 2020), https://leagueoflegends.fandom.com/wiki/Consumable%20item.

[9] L. Rokach and O. Maimon, *Decision Trees*. Boston, MA: Springer US, 2005, pp. 165–192.

[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, 2001.

[11] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. Kegelmeyer, "A Comparison of Decision Tree Ensemble Creation Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 173–180, jan 2007.

[12] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, jun 2006.