Champion Recommendation in League of Legends Using Machine Learning

Kinga Wiktoria Błaszczyk[1] and Dominik Szajerman[1][0000−0002−4316−5310]

Institute of Information Technology, Lodz University of Technology, Łódź, Poland
dominik.szajerman@p.lodz.pl

Abstract. League of Legends (LoL) is a Multiplayer Online Battle Arena (MOBA) game with over 160 champions and a competitive esports scene. The ban-and-pick system allows players to choose champions before a match, which can greatly impact the outcome, especially in professional leagues. This article presents an overview of the development of a champion recommendation system for League of Legends, with a focus on the evaluation and comparison of multiple machine learning models. Our system offers real-time recommendations during the pick and ban phase, utilizing data from professional and high-level games. The accuracy and performance of the various models are analyzed and presented, providing insights into the strengths and weaknesses of each approach in solving the task of champion recommendation. Results show that the player's statistics on a champion are the biggest determinants of a game's outcome.

Keywords: League of Legends · Champion recommendation · Machine Learning

1 Introduction

League of Legends (Lol) is a worldwide famous multiplayer online battle arena (MOBA) game highly known for its competitive esports scene. The gamereleased in 2009 by Riot Gameshas been consistently ranked as one of the most played games in the world, with an active player base of over 100 million people.

The game consists of two teams of five players competing against each other in a strategic battle to destroy the enemy's base. Before the match starts, the player must choose a unique champion to play, due to the ban and pick system used by the game code. With 162 champions to choose from, each with their own set of abilities and play style, the choice of the champions to play as, becomes a tough one. When choosing a champion, players must carefully weigh their strengths and weaknesses, as well as how they will complement their team's strategy and counter their opponents' chosen champions. This decision can have a significant impact on the game's outcome, especially in professional leagues.

To help players make a more informed decision, we developed a real-time recommendation system. The system uses machine learning algorithms to suggest champions that may be effective in the current game based on factors such as the player's past performance.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_12

1.1 Gameplay Overview

League of Legends (LoL) is a game where two teams of five players compete against each other. Players control a champion with unique abilities and battle it out on a map called the Rift. The goal is to destroy the enemy's nexus, a structure located in their base, while defending your own.

Each game starts with champions picking their loadout and heading to their lanes, where they will fight against the enemy team and minions (computer-controlled creatures). The objective is to earn gold and experience by killing minions and enemy champions, which will allow players to level up and purchase powerful items to enhance their champion's abilities.

Victory in LoL requires not only individual skill but also teamwork and strategy. Champions have unique abilities that can be combined with others to execute powerful combinations and take down the enemy. Teams must communicate and coordinate their movements and abilities to secure objectives and push towards the enemy nexus.

Overall, League of Legends is a fast-paced, action-packed game that rewards strategic thinking, teamwork, and quick reflexes. With over 160 champions to choose from, there is always a new challenge to tackle and a new way to play.

1.2 Pick and Ban Phase

In MOBA games, the drafting stage is of paramount importance. The selection of characters with strong synergy can be the deciding factor in a team's victory. Games like League of Legends have implemented the pick and ban system, which holds great power in shaping the outcome of the match. The pick and ban system is a way for teams to strategically select the champions they will play as, while also preventing the enemy team from using certain champions that may be particularly strong against them. Each team takes turns selecting and banning champions, with the first team selecting two champions, the second team selecting two champions and banning one champion, and the first team banning one champion. This process is repeated until all champions have been selected and banned.

2 Related Work

Recently, there have been limited publications on various strategies for developing effective Champion Recommendation systems in MOBA games. Costa et al. [5] investigated a method of creating team compositions in League of Legends using genetic algorithms. Their research found that the use of genetic algorithms guided by appropriate fitness functions significantly enhanced team composition, with generated teams showing a quality of between 76% and 95%. Hong et al. [9] found that a neural network classifier was more effective than a random forest classifier in their study of a champion recommendation system for League of Legends. In contrast, Porokhnenko et al. [11] found that linear regression was the fastest model when considering different machine learning methods for predicting match results in Dota 2. Harikumar et al. [2] found that their models achieved high accuracy (between 95% and 99%) for predicting match outcomes in League of Legends using feature selection and ensemble methods. Hanke et al. [7] also evaluated a recommender system for hero line-ups in MOBA games and found that the neural network they trained was capable of predicting the winning team with an accuracy of 88.63In a study of a predictor for the MOBA game Dota 2, Kinkade et al. [10] used Logistic Regression and Random Forest Classifier models to achieve an accuracy of 73% at the beginning of the match. Chen et al. [3] treated the drafting process as a combinatorial game and confirmed that Monte Carlo Tree Search-based recommendations could lead to stronger hero line-ups compared to other baselines. Yu et al. [6] found that a SVD-based collaborative filtering approach was suitable for recommending champions in League of Legends, while Conley et al. [4] achieved promising results for a recommendation engine for picking heroes in Dota 2 using logistic regression and K-nearest neighbor models. More recent work has focused on predicting the results of the games rather than building larger systems on this basis[8, 12]. Overall, these studies demonstrate that it is possible to make reliable predictions of match outcomes and recommend effective hero line-ups in MOBA games using various machine learning techniques.

3 Methodology

3.1 Formulation of machine learning problem

The objective of this research is to develop an efficient system for champion recommendation in League of Legends. To achieve this goal, the first step is to construct an algorithm for predicting match outcomes by using the League of Legends champions chosen by players or other relevant factors. This problem is a binary classification problem with two output classes, representing a victory for the "blue" or "red" team.

3.2 Datasets

Pre-made datasets. The datasets were sourced from Kaggle[1], each featuring a unique set of characteristics for describing professional League of Legends matches. The datasets include statistics from 2,840 matches that took place between January 1st, 2021, and March 23rd, 2021. Data for these datasets was specifically gathered from Oracle's Elixir[2], a reputable source for professional gaming information since 2015. A general explanation of the datasets is presented below:

[1] https://www.kaggle.com/datasets/tekpixo/leagueoflegendsprematch2021
[2] oracleselixir.com/about

- Players Statistics (**PS**) dataset: This dataset contains pre-game statistics for each player with their picked champion, including win rate percentage (WR), games played (GP), and the ratio of kills plus assists over deaths (KDA) for that champion in previous matches. The dataset includes a total of 31 features;
- Picked Champions and Players Statistics (**PC+PS**) dataset: This dataset merges data from the Picked Champions and Players Statistics datasets, resulting in a total of 41 features;
- Complete (**C**) dataset: This dataset includes all the information from the previous datasets, resulting in a total of 51 features.
- Picked Champions (**PC**) dataset: This dataset includes 11 features that provide detailed information about the champions picked by both teams in each match and the outcome of the game;
- Banned Champions (**BC**) dataset: The dataset includes 11 features that provide detailed information about the champions banned by both teams in each match, as well as the outcome of the match;

Table 1 provides the structure for PS dataset and a preview of data. The columns contain: btgp – blue top games played, btwr – blue top win ratio, btkda – blue top kill/death/assists ratio etc. for the remaining positions and side. The [. . .] columns include analogous statistics for red. Result: "1" – blue team won, "0" – it lost.

Table 1. Pre-made Player Statistics dataset preview.

game	btgp	btwr	btkda	bjgp	bjwr	bjkda	bmgp	bmwr	bmkda	. . .	result
6911-9185	31	0.32	2.4	17	0.47	4.3	6	0.67	2.7	. . .	0
6911-9186	10	0.6	3.4	21	0.33	3.6	6	0.17	1.9	. . .	1
6912-9187	35	0.77	2.8	11	0.82	8.2	84	0.64	5.3	. . .	0

Tables 2and 3 provide a similar preview of the pre-made dataset's categorization PC and BC.

Table 2. Pre-made – Picked Champions dataset preview. Names: btop, bjg, bmid etc. mean champion id picked from the blue side. Names: rtop, rjg, rmid etc. mean champion id banned from the red side. Result: "1" – blue team won, "0" – it lost.

game	btop	bjg	bmid	badc	bsupp	rtop	rjg	rmid	radc	rsupp	result
6911-9185	58	113	134	135	12	106	80	142	145	57	0
6911-9186	79	104	134	523	412	39	2	61	145	875	1
6912-9187	150	76	61	21	3	58	104	112	202	89	0

As previously mentioned, data were obtained from Oracle's Elixir portal. The platform provides a daily updated CSV file containing all professional matches of

Table 3. Pre-made – Banned Champions dataset preview. Names: bban1, bban2, bban3 etc. mean champion id banned from the blue side. Names: rban1, rban2 etc. mean champion id banned from the red side. Result: "1" – blue team won, "0" – it lost.

game	bban1	bban2	bban3	bban4	bban5	rban1	rban2	rban3	rban4	rban5	result
6911-9185	84	777	236	516	54	2	104	3	266	164	0
6911-9186	3	58	266	164	142	84	135	236	777	516	1
6912-9187	163	4	135	516	106	84	134	12	429	777	0

the year. The dataset includes the performance history of the 10 players involved in each match with the champions they selected for the game.

Riot API generated datasets. The datasets were created utilizing the API provided by Riot Games, the developer of League of Legends. The datasets include data from 2277 games, collected between July 1st, 2022, and October 30th, 2022. These datasets were modeled after the pre-made datasets to expand the available data for the study. The data satisfies the following requirements:

- The game mode is RANKED SOLO/DUO, which features a ban and pick system. This system allows for the collection of valuable data on banned and picked champions for the study.
- The skill level of the players is "very high", which corresponds to roughly the top 0.011% of players. It is believed that the performance of available champions will be optimal in the hands of these highly skilled players.
- The game duration is set at over 20 minutes, this criteria is established to ensure that the games are as fair as possible (LoL has a surrender system, which allows players to end the game from the 15th minute, it requires 100% agreement from the whole team of 5 players. After the 20th minute of the game, the surrender requires 80% agreement).

Similar to the pre-made datasets, a crawler was developed to collect performance history data on the 10 players involved in each match with the champions they selected for the game. The crawler gathered data from the 20 previously played games for each individual player from the 10 in question. This data were used to create Players Statistics datasets.

Both of these dataset groups pertain to binary classification, with the target feature in the final column indicating the game result for the blue team, represented by "1" for a win or "0" for a loss. The target feature is consistent across all datasets, with a class distribution of:

- Pre-made datasets: 46.37% wins for the blue team and 53.63% wins for the red team.
- Riot API generated datasets: 51.12% wins for the blue team and 48.88% wins for the red team.

It is noteworthy that the pre-made datasets included data for 155 champions available in League of Legends, whereas the datasets generated through the Riot API included data for 161 champions.

3.3 Machine learning models for solving the problem

This research employed various models to determine the likelihood of victory for the blue team. These models were grouped into seven categories:

1. Ensemble Models, which combine multiple base models to create a stronger overall model. These models are often used when the base models are weak learners, but when combined, they can increase the overall performance.
   - Gradient Boosting (GB)[1]: a method that iteratively trains base models and adjusts the weights of the training instances to focus on the misclassified instances in the previous iteration.
   - Extreme Gradient Boosting (XGB)[1]: an optimized version of Gradient Boosting that uses histogram-based algorithms for faster and more accurate training.
   - Bagging Classifier (BC)[1]: a method that trains multiple instances of a base model on different subsets of data and combines their predictions.
   - AdaBoost (AdaB)[1]: an algorithm that combines weak learners to improve classification accuracy. It adjusts instance weights in each iteration to focus on misclassifications.
   - Random Forest Classifier (RFC)[1]: a method that trains multiple decision trees and combines their predictions.
   - Random Forest Regressor (RFR)[1]: a method that trains multiple decision trees for regression tasks and combines their predictions.
2. Neural Networks (NN), which are inspired by the structure and function of the human brain. They are widely used in tasks such as image recognition, speech recognition, and natural language processing, and are characterized by multiple layers and high adaptability.
   - Neural Network MLP (MLP)[1]: a simple feedforward NN that has multiple layers of neurons, it s used for supervised learning tasks.
   - Recurrent Neural Network (RNN)[1]: a type of neural network that is particularly well-suited for sequential data, it uses feedback connections to process sequences of variable length.
   - Fully Connected Neural Network (FCNN)[1]: a type of neural network that is commonly used in image recognition tasks, it is a neural network where all the neurons in a layer are fully connected to the neurons of the previous and next layers.
3. Bayesian Models, which are based on Bayes' theorem. They are characterized by their ability to handle uncertainty and complexity in data, and are widely used in tasks such as classification and regression.
   - Gaussian Process Classifier (GPC)[1]: a probabilistic model that can be used for classification tasks, it makes predictions based on the probability distribution of the input.
   - Gaussian Process Regressor (GPR)[1]: a probabilistic model that can be used for regression tasks, it makes predictions based on the probability distribution of the input.

   - Quadratic Discriminant Analysis (QDA)[1]: a type of Bayesian model that is used for classification, it is characterized by its ability to model quadratic decision boundaries.
   - Gaussian Naive Bayes (GNB)[1]: a probabilistic model that uses Bayes' theorem to classify data based on the probability of a given class given a set of features.
4. Linear Models, which are based on linear equations to make predictions. They are simple yet powerful models and are used for prediction and forecasting purposes.
   - Logistic Regression (LogR)[1]: a linear model that is used for binary classification, it is characterized by the logistic function that maps the input to a probability value between 0 and 1.
   - Linear Support Vector Classification (LSVC)[1]: a type of SVM model that separates the data with a linear boundary.
   - Ridge Classifier (RC)[1]: a linear model that is used for classification, it uses L2 regularization method to prevent overfitting.
   - Linear Regression (LR)[1]: a technique used to model the relationship between a scalar dependent variable and one or more independent variables represented by a vector, it is a simple yet powerful model that is widely used for prediction and forecasting purposes.
5. Decision Tree Models, which create a tree-like structure to make predictions. These models are characterized by their ability to handle non-linear relationships and categorical variables.
   - Decision Tree Classifier (DTC)[1]: a model that creates a tree-like structure to make predictions, it is characterized by its ability to handle non-linear relationships and categorical variables.
   - Cat Boost Classifier (CBC)[1]: an optimization of the decision tree algorithm, it is particularly useful for dealing with categorical variables.
6. Support Vector Machines (SVMs), which are based on the concept of maximizing the margin between different classes. These models are widely used in tasks such as classification and regression.
   - SVM linear (SVM-L)[1]: a type of SVM model that separates the data with a linear boundary.
   - SVM radial (SVM-R)[1]: as above with a non-linear boundary.
7. k-Nearest Neighbors (k-NN), which is a non-parametric, instance-based learning algorithm. It does not have a model equation like linear or logistic regression models. Instead, it makes predictions based on the similarity between the input instance and the instances in the training set.
   - k-Nearest Neighbors (kNN)[1]: a non-parametric, instance-based learning algorithm, it makes predictions based on the similarity between the input instance and the instances in the training set.

All data were divided into training and test sets. The data were split into 70% for training, and 30% for testing. After that, they were submitted to the k-fold cross-validation method with 10 folds and 3 repetitions. Followed by parameter optimization with the GridSearchCV algorithm. Table 4 lists the models that contain hyperparameters.

Table 4: Enumerated values of hyperparameters of machine learning models.

Model	**Hyperparameter and Values**
RFC	n-estimators: [100, 300, 500, 800, 1200]
	max-depth: [5, 10, 15, 25, 30]
	min-samples-split: [2, 5, 10, 15, 100]
	min-samples-leaf: [1, 2, 5, 10]
SVM-R	C: np.arange(0.1, 1, 0.1)
	kernel: ['rbf']
SVM-L	C: np.arange(0.1, 1, 0.1)
	kernel: ['linear']
DTC	max-depth: range(1, 16, 1)
	min-samples-split: range(2, 10, 1)
	min-samples-leaf: range(1, 21, 1)
	criterion: ['gini', 'entropy', 'log-loss']
GNB	var-smoothing: np.logspace(0, -10, num=100)
kNN	leaf-size: np.arange(1, 50, 1)
	n-neighbors: np.arange(10, 30, 1), 'p: [1, 2]
AdaB	n-estimators: range(10, 200, 10)
GB	n-estimators: range(10, 200, 10)
	max-depth: range(1, 16, 1)
	min-samples-split: range(200, 1001, 200)
	min-samples-leaf: range(30, 71, 10)
	max-features: range(7, 20, 2)
	subsample: [0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 1]
XGB	n-estimators: range(10, 200, 10)
	max-depth: range(1, 16, 1)
MLP	hidden-layer-sizes: [(sp-randint.rvs(100, 600, 1), sp-randint.rvs(100, 600, 1),), (sp-randint.rvs(100, 600, 1),)]
	activation: ['identity', 'logistic', 'tanh', 'relu']
	solver: ['sgd', 'adam', 'lbfgs']
	alpha: np.arange(0.0001, 0.01, 0.001)
	learning-rate: ['constant', 'adaptive', 'invscaling']
LogR	penalty: ['l1', 'l2', 'elasticnet', 'none']
	C: np.logspace(-4, 4, 20)
	solver: ['lbfgs', 'newton-cg', 'liblinear', 'sag', 'saga']
	max-iter: [100, 1000, 2500, 5000, 10000]
GPC	kernel: [1*RBF(), 1*DotProduct(), 1*Matern(), 1*RationalQuadratic(), 1*WhiteKernel()]
GPR	'kernel: [1*RBF(), 1*DotProduct(), 1*Matern(), 1*RationalQuadratic(), 1*WhiteKernel()]
	alpha: [1e-2, 1e-3]
QDA	reg-param: np.arange(1e-04, 1e-01, 1e-04)
LSVC	penalty: ['l1', 'l2']

	C: np.logspace(1, 4, 20)
	tol: np.arange(1e-6, 1e-5, 1e-6)
	loss: ['hinge', 'squared-hinge']
	multi-class: ['ovr', 'crammer-singer']
	class-weight: ['None', 'balanced']
CBC	learning-rate: [0.001, 0.01, 0.1]
	eval-metric: ['RMSE', 'AUC', 'MultiClass']
	loss-function: ['Logloss', 'CrossEntropy']
	iterations: np.arange(100, 200, 10)
	depth: np.arange(4, 10, 1)
	l2-leaf-reg: np.arange(2, 10, 1)
	random-strength: np.arange(0, 10, 1)
BC	n-estimators: np.arange(500, 1000, 100)
	max-samples: np.arange(0.1, 1, 0.1)
	max-features: np.arange(1, 20, 1)
RC	alpha: np.logspace(0.1, 1, 20)

4 Results and discussion

4.1 Pre-made datasets

Table 5 represents the accuracy of the models for different datasets from Pre-made set. Observing the data behavior, it is noticeable that models have the highest accuracy for the Players Statistics (**PS**) dataset, whereas that accuracy drops slightly for the Picked Champions and Players Statistics (**PC+PS**) and Complete (**C**) datasets and then drastically for the Picked Champions (**PC**) and Banned Champions (**BC**) datasets. The difference that causes the model precision to drop between the Players Statistics and Picked Champions and Players Statistics is that the Picked Champions and Players Statistics dataset has an additional feature (Picked Champion ID), which is correlated with other features in the Players Statistics dataset so including it in Picked Champions and Players Statistics dataset increases the multicollinearity and decrease the model's performance. A similar thing is happening with the Complete dataset, which also has this extra feature (Picked Champion ID) as well as more additional features which are Banned Champions ID. Not only does the correlated feature lower the accuracy but so do the other features, because they are deemed not relevant to the task at hand. Information about the banned champions does not impact the outcome of the game. It leads us to the two last datasets, PC and BC. Models performance was the worst for those datasets simply because the data about champions included and excluded in the game does not provide enough information to greatly affect the outcome of the game. The best-performing models are Gradient Boosting, Linear Support Vector Classification, Random Forest Regressor, and Cat Boost Classifier. It's noticeable that some of the models struggle to perform as well as others on all of the available datasets. Them being: Support Vector Machines radial, k-Nearest Neighbors, Neural Network

MLP, Recurrent Neural Networks, and Fully Connected Neural Networks. Support Vector Machines radial and Neural Network MLP are both complex models that have many parameters to adjust, which can make them more prone to overfitting if not properly regularized. They also require more data to train properly. k-Nearest Neighbors is a non-parametric method that is sensitive to the scale and distribution of the features. It is also sensitive to the presence of outliers. Recurrent Neural Networks and Fully Connected Neural Networks are both deep learning models that require large amounts of data to train and can be sensitive to the quality and quantity of data. They are also sensitive to the architecture of the network, which can be difficult to optimize.

Table 5: Experimental Results for Machine Learning Models on Pre-made Datasets: Model Name, Accuracy. The bold text in the table represents the best-scoring model for a specific dataset, while the grey background indicates the worst-scoring model overall.

Model	**PS**	**PC+PS**	**C**	**PC**	**BC**
Linear Regression	91.67%	86.97%	86.97%	53.05%	51.76%
Random Forest Regressor	**92.02%**	88.26%	87.,21%	55.75%	51.29%
Random Forest Classifier	87.44%	80.28%	81.10%	55.87%	53.76%
Support Vector Machines radial	78.99%	52.81%	52.93%	51.06%	51.53%
Support Vector Machines linear	91.31%	87.56%	87.44%	53.52%	53.87%
Decision Tree Classifier	83.92%	80.05%	78.52%	50.82%	53.40%
Gaussian Naive Bayes radial	88.50%	83.10%	82.75%	50.94%	50.59%
k-Nearest Neighbors	69.95%	52.46%	51.41%	51.41%	51.88%
AdaBoost	91.78%	86.74%	86.97%	56.81%	51.76%
Gradient Boosting	**92.37%**	86.62%	86.74%	55.99%	51.41%
Extreme Gradient Boosting	92.14%	88.26%	87.09%	55.40%	51.88%
Neural Network MLP	85.56%	53.87%	53.87%	53.87%	53.87%
Logistic Regression	90.85%	81.46%	81.92%	52.46%	51.99%
Recurrent Neural Network	91.31%	52.82%	52.58%	52.11%	52.58%
Fully Connected Neural Network	78.64%	46.13%	46.24%	46.13%	46.13%
Gaussian Process Classifier	90.96%	87.44%	87.32%	53.52%	53.52%
Gaussian Process Regressor	91.67%	86.97%	86.97%	52.93%	51.88%
Quadratic Discriminant Analysis	88.26%	82.04%	82.75%	51.06%	49.53%
Linear Support Vector Classification	**92.25%**	87.68%	87.56%	53.05%	51.76%
Cat Boost Classifier	**92.02%**	87.44%	87.91%	55.16%	51.53%
Bagging Classifier	89.79%	85.92%	84.51%	52.82%	51.88%
Ridge Classifier	91.78%	87.44%	87.21%	53.05%	51.76%

Table 6 shows the improved accuracy of each model when evaluated on the same datasets as in Table 5. These models were optimized using optimal parameters obtained from the model tuning process. However, some models did not improve at all, with accuracy scores remaining the same as in Table 5. These include

Linear Regression, Random Forest Regressor, Support Vector Machines (linear), Recurrent Neural Network, Fully Connected Neural Network, Gaussian Process Regressor, and Linear Support Vector Classification. This lack of improvement suggests that either the models have reached their maximum potential or that the chosen parameters were not suitable for the task at hand. The greatest improvement in accuracy was seen in the Neural Network MLP model, with an increase of 30.75% for the "Picked Champions and Players Statistics" dataset, 24.06% for the "Complete" dataset, and 5.99% for the "Player Statistics" dataset. In the "Player Statistics" dataset, the models with the best improvement were k-Nearest Neighbors (9.51%), Neural Network MLP (5.99%), and Support Vector Machines radial (4.70%). Two models, Extreme Gradient Boosting (92.61%) and AdaBoost (92.49%), even outperformed the highest-performing model from the previous experiment (92.37%).

Table 6: Experimental Results for Hyperparameterized Machine Learning Models on Pre-made Datasets: Model Name, Accuracy. The bold text in the table represents the best-scoring model for a specific dataset, while the grey background indicates the worst-scoring model overall.

Model	**PS**	**PC+PS**	**C**	**PC**	**BC**
Linear Regression	91.67%	86.97%	86.97%	53.05%	51.76%
Random Forest Regressor	92.02%	88.26%	87.21%	55.75%	51.29%
Random Forest Classifier	92.14%	87.68%	87.44%	56.92%	52.00%
Support Vector Machines radial	79.23%	53.40%	54.11%	53.05%	53.87%
Support Vector Machines linear	91.31%	87.56%	87.44%	53.52%	53.87%
Decision Tree Classifier	85.56%	77.11%	80.40%	53.87%	52.82%
Gaussian Naive Bayes radial	88.50%	83.10%	82.75%	53.52%	52.82%
k-Nearest Neighbors	79.46%	56.10%	50.94%	54.11%	52.35%
AdaBoost	**92.49%**	87.79%	87.79%	54.58%	54.58%
Gradient Boosting	92.02%	89.08%	88.62%	53.76%	52.93%
Extreme Gradient Boosting	**92.61%**	88.73%	88.50%	54.23%	52.70%
Neural Network MLP	91.55%	84.62%	77.93%	54.34%	53.87%
Logistic Regression	92.14%	86.03%	86.85%	53.87%	52.93%
Recurrent Neural Network	91.31%	52.82%	52.58%	52.11%	52.58%
Fully Connected Neural Network	78.64%	46.13%	46.24%	46.13%	46.13%
Gaussian Process Classifier	92.14%	87.68%	87.32%	53.52%	52.35%
Gaussian Process Regressor	91.67%	86.97%	86.97%	52.93%	51.88%
Quadratic Discriminant Analysis	88.26%	82.86%	82.04%	51.17%	49.77%
Linear Support Vector Classification	92.25%	87.68%	87.56%	53.05%	51.76%
Cat Boost Classifier	91.55%	88.62%	87.68%	55.05%	50.70%
Bagging Classifier	91.90%	88.50%	88.38%	58.22%	54.93%
Ridge Classifier	91.78%	87.56%	87.44%	53.05%	51.76%

4.2 Riot API datasets

Table 7 presents the accuracy scores of various models evaluated on datasets sourced from the Riot API. These models were optimized using optimal parameters determined through the model-tuning process. It's apparent that the results for this data set, especially for "Players Statistics", "Picked Champions and Players Statistics", and "Complete" datasets, surpass those from the previous experiments. The reason behind that first is the accumulation of the data gathered in order to calculate the "Player Statistics" on related champions. There's no information provided on the pre-made set of data, about how many matches were used to determine the player statistics whereas, in this experiment, a number of 20 last played games was used. It's assumed that a pre-made set of data used a smaller volume of matches to compute the player statistics. The best-performing models are Random Forest Classifier, Bagging Classifier, and Gradient Boosting. Random Forest Classifier and Bagging Classifier are both Ensemble methods that build multiple decision trees and combine their predictions. These methods are robust to overfitting and outliers and handle well with high dimensionality and correlated features. Gradient Boosting is also an ensemble method that combines multiple decision trees. Gradient Boosting is a powerful technique that builds new trees which complement the already-built trees. It is also robust to overfitting and outliers and is able to capture complex interactions among the features. This dataset has the right characteristics to train these models effectively, they are well-suited to the task at hand.

The 98% accuracy achieved is slightly higher than the scores reported by Harikumar et al. [2] and Costa et al. [5].

Overall, all models achieved greater accuracy for the Riot API dataset. Figure 1 shows the differences in accuracy for individual models processing data from both datasets: Pre-made and Riot API.

Table 7: Experimental Results for Hyperparameterized Machine Learning Models on Riot API Datasets: Model Name, Accuracy, Execution Time for PS. The bold text in the table represents the best-scoring model for a specific dataset. The best models for the task at hand, in terms of accuracy and execution time, are indicated by an underline.

Model	**PS**	**PC+PS**	**C**	**PC**	**BC**	**Ex Time [s] (for PS)**
LR	98.68%	98.83%	98.83%	49.93%	48.17%	0.0180
RFR	97.95%	98.10%	97.80%	52.12%	49.19%	1.1571
RFC	**98.98%**	98.98%	98.83%	52.56%	51.83%	2.5695
SVM-R	89.46%	51.24%	53.00%	51.24%	49.05%	0.3335
SMV-L	98.54%	98.68%	98.24%	52.56%	49.19%	0.0922
DTC	95.90%	95.02%	94.58%	48.61%	51.98%	0.0336
GNB	97.51%	97.36%	97.51%	50.51%	49.78%	0.0200
kNN	89.31%	51.83%	53.59%	49.78%	52.12%	0.1225
AdaB	98.68%	98.68%	98.68%	49.93%	52.42%	1.1007

GB	**98.83%**	98.54%	98.54%	49.93%	48.02%	0.2319
XGB	98.39%	98.68%	98.98%	51.39%	51.24%	0.3107
MLP	97.36%	95.75%	94.29%	51.39%	49.78%	0.1389
LogR	98.68%	98.54%	98.24%	47.88%	49.78%	0.1075
RNN	97.22%	96.34%	56.52%	48.17%	47.73%	16.5964
FCNN	87.99%	51.98%	52.12%	52.27%	49.78%	5.4875
GPC	98.68%	98.54%	97.95%	50.95%	53.73%	21.3553
GPR	98.68%	98.83%	98.83%	49.93%	48.17%	6.3128
QDA	97.07%	97.07%	97.07%	49.63%	48.76%	0.0580
LSVC	97.95%	88.29%	83.75%	52.27%	51.54%	0.0470
CBC	98.54%	98.39%	98.68%	50.80%	51.10%	3.6650
BC	**98.96%**	98.83%	98.83%	49.49%	50.37%	5.0829
RC	98.68%	98.83%	98.68%	49.93%	48.17%	0.0520

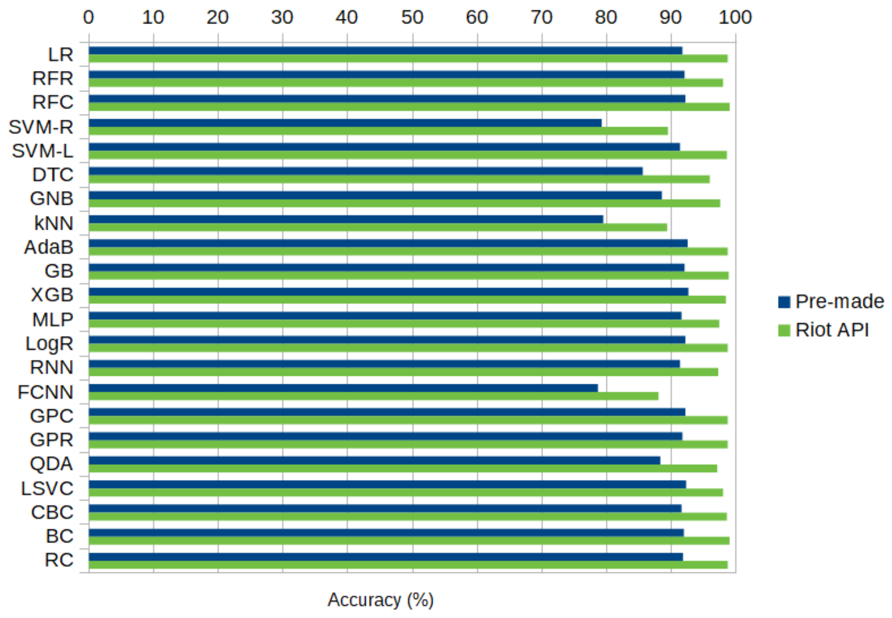

Fig. 1. Performance of tested models for PS and both datasets.

4.3 Execution time

Table 7 including the column "Ex Time" presents juxtaposition of models accuracy and their execution time for the Riot API – "Player Statistics" dataset.

It is important to note that the execution time of a model is essential in a recommendation system because it is performing in real time. Therefore a rapid and accurate model is more valued than just an accurate but slow one. It is observable that between the three highest-scoring models and the three fastest models, there is no correlation. That is because more accurate models need more time to perform above others. In such a situation, it is better to forsake the best scoring models and focus on faster-executing ones. This being the case, the best models for the given task are Linear Regression, Ridge Classifier, and Support Vector Machines linear (fig. 2).

Machine configuration: Processor Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 8.00 GB RAM, Windows 10 Home 64-bit.

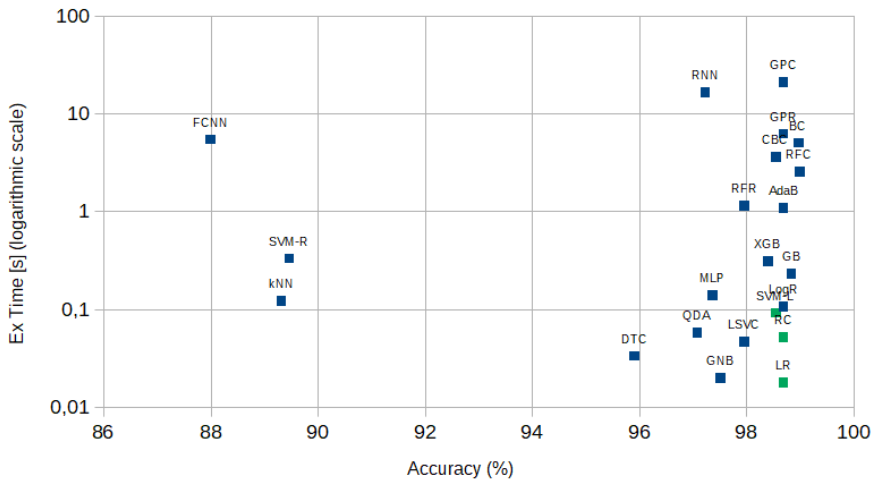

Fig. 2. Scatter plot showing Ex Time vs. Accuracy. Top-performing models are marked green.

5 Conclusions

In conclusion, the research results show that the models have the highest accuracy for the "Players Statistics" dataset, but accuracy drops slightly for datasets with additional correlated features. The highest performing models are Extreme Gradient Boosting and Random Forest Classifier. The 98% accuracy achieved in this study represents a potentially improved outcome compared to previous findings in the literature. However, when considering accuracy as well as execution time, the optimal models for the task at hand are Linear Regression, Support Vector Machines linear, and Ridge Classifier. Some models, such as Support Vector Machines radial, k-Nearest Neighbors, Neural Network MLP, Recurrent

Neural Networks, and Fully Connected Neural Networks, struggle to perform as well on all datasets due to factors such as overfitting, sensitivity to scale and distribution of features, and the need for large amounts of data. Additionally, the results from using datasets sourced from the Riot API show improved performance, likely due to the accumulation of data used to calculate player statistics. It is worth noting that some of the models may have longer execution time compared to others, it is important to consider the trade-off between accuracy and execution time when selecting a model for recommendation system.

References

1. Supervised learning, https://scikit-learn.org/stable/supervised_learning.html
2. Ani, R., Harikumar, V., Devan, A.K., Deepa, O.: Victory prediction in league of legends using feature selection and ensemble methods (may 2019). https://doi.org/DOI: 10.1109/ICCS45141.2019.9065758
3. Chen, Z., Nguyen, T.H.D., Xu, Y., Amato, C., Cooper, S., Sun, Y., El-Nasr, M.S.: The art of drafting: A team-oriented hero recommendation system for multiplayer online battle arena games (2018). https://doi.org/https://doi.org/10.48550/arXiv.1806.10130
4. Conley, K., Perry, D.: How does he saw me ? a recommendation engine for picking heroes in dota 2 (2013)
5. Costa, L.M., Souza, A.C.C., Souza, F.C.M.: An approach for team composition in league of legends using genetic algorithm (oct 2019). https://doi.org/10.1109/sbgames.2019.00018
6. Do, T.D., Yu, D.S., Anwer, S., Wang, S.I.: Using collaborative filtering to recommend champions in league of legends (aug 2020). https://doi.org/DOI:10.1109/CoG47356.2020.9231735
7. Hanke, L., Chaimowicz, L.: A recommender system for hero line-ups in MOBA games. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **13**(1), 43–49 (jun 2021). https://doi.org/https://doi.org/10.1609/aiide.v13i1.12938
8. Hitar-Garcia, J.A., Moran-Fernandez, L., Bolon-Canedo, V.: Machine learning methods for predicting league of legends game outcome. IEEE Transactions on Games pp. 1–1 (2022). https://doi.org/10.1109/tg.2022.3153086
9. Hong, S.J., Lee, S.K., Yang, S.I.: Champion recommendation system of league of legends (oct 2020). https://doi.org/10.1109/ICTC49870.2020.9289546
10. Kinkade, N.: Dota 2 win prediction (2015)
11. Porokhnenko, I., Polezhaev, P., Shukhman, A.: Machine learning approaches to choose heroes in dota 2 (apr 2019). https://doi.org/10.23919/FRUCT.2019.8711985
12. Shen, Q.: A machine learning approach to predict the result of league of legends. In: 2022 International Conference on Machine Learning and Knowledge Engineering (MLKE). IEEE (feb 2022). https://doi.org/10.1109/mlke55170.2022.00013