Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in League of Legends

Tiffany D. Do
University of Central Florida
Orlando, Florida, United States
tiffanydo@knights.ucf.edu

Seong Ioi Wang
University of Texas at Dallas
Richardson, Texas, United States
seongiwang@gmail.com

Dylan S. Yu
University of Texas at Dallas
Richardson, Texas, United States
dylanyu461@gmail.com

Matthew G. McMillian
University of Texas at Dallas
Richardson, Texas, United States
matthewgmcmillian@gmail.com

Ryan P. McMahan
University of Central Florida
Orlando, Florida, United States
rpm@ucf.edu

ABSTRACT

League of Legends (LoL) is the most widely played multiplayer online battle arena (MOBA) game in the world. An important aspect of LoL is competitive ranked play, which utilizes a skill-based matchmaking system to form fair teams. However, players' skill levels vary widely depending on which champion, or hero, that they choose to play as. In this paper, we propose a method for predicting game outcomes in ranked LoL games based on players' experience with their selected champion. Using a deep neural network, we found that game outcomes can be predicted with 75.1% accuracy after all players have selected champions, which occurs before gameplay begins. Our results have important implications for playing LoL and matchmaking. Firstly, individual champion skill plays a significant role in the outcome of a match, regardless of team composition. Secondly, even after the skill-based matchmaking, there is still a wide variance in team skill before gameplay begins. Finally, players should only play champions that they have mastered, if they want to win games.

CCS CONCEPTS

• **Information systems** → **Massively multiplayer online games**;
• **Applied computing** → **Computer games**.

KEYWORDS

game outcome prediction, League of Legends, multiplayer online battle arenas, player experience, machine learning, matchmaking

ACM Reference Format:
Tiffany D. Do, Seong Ioi Wang, Dylan S. Yu, Matthew G. McMillian, and Ryan P. McMahan. 2021. Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in League of Legends. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021 (FDG'21), August 3–6, 2021, Montreal, QC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3472538.3472579

FDG'21, August 3–6, 2021, Montreal, QC, Canada
© 2021 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *The 16th International Conference on the Foundations of Digital Games (FDG) 2021 (FDG'21), August 3–6, 2021, Montreal, QC, Canada*, https://doi.org/10.1145/3472538.3472579.

1 INTRODUCTION

League of Legends (LoL), a popular computer game developed by Riot Games, is currently the most widely played Multiplayer Online Battle Arena (MOBA) [3] game in the world. In 2019, there were eight million concurrent players daily [14], and the player base has continued to grow since its release in 2009. A core aspect of LoL is competitive ranked gameplay. In typical ranked gameplay, ten human players are matched together to form two teams of approximately equal skill. These two teams, consisting of five players each, battle against each other to destroy the opposing team's base.

Fair matchmaking is crucial for player experience [31]. In 2019, Riot Games stated that ranked matchmaking should be as fair as possible [11]. This goal has persisted throughout the history of the game. In 2020, Riot Games stated that some of their main goals for the year were to preserve competitive integrity [12] and improve matchmaking quality [13] for ranked games. In order to create fair matches between players of approximately equivalent skill level, matchmaking is determined using an Elo rating system, similar to the one originally used by chess players [26]. Although this matchmaking system has improved in recent years (c.f., [10, 11, 13]), it does not consider players' champion selections when forming matches. LoL has over 150 playable characters, known as champions, that have their own unique playstyles and abilities [7]. Players select a champion at the start of every match after the matchmaking algorithm has formed teams. However, players will often perform better on some champions than on others due to their differing levels of mechanical expertise, which is defined as a player's knowledge of their champion's abilities and interactions [8]. Higher levels of mechanical expertise on particular champions allow players to make quicker and better judgments, which are essential in the game's fast paced environment. Since mechanical expertise plays such a large impact on a player's own performance, it can therefore cause a similar impact on the match's outcome.

In this paper, we introduce a machine learning model based on a deep neural network (DNN) that can predict ranked match outcomes based on players' experience on their selected champion (i.e., player-champion experience). We show that the outcome of a ranked match can be predicted with over 75% accuracy after all players have selected their champions. This occurs before the match actually begins. Our results indicate that individual champion skill plays a significant role in the outcome of a match, regardless of

team composition. Additionally, current matchmaking may not form teams of approximately equal skill level in game because matchmaking is done before players select their champions. Our results also imply that players should only play champions that they have mastered in order to win ranked games. This can decrease gameplay variety as players typically master only a few champions out of the 150+ champions available in the roster [5, 7].

2 BACKGROUND AND RELATED WORK

2.1 Ranked Gameplay in League of Legends

Ranked gameplay in LoL is based on seasonal gameplay, similar to traditional sports. During each season, LoL separates players into six main divisions known as Leagues based on skill level. These Leagues are known as Iron, Bronze, Silver, Gold, Platinum, and Diamond [23]. There are a few distinctions past Diamond. However, we do not consider these players for the purposes of this player as they make up a minuscule portion of the playerbase. In order to rank up and enter higher Leagues, players must win ranked matches. In ranked matches, two teams of five players each are formed using a matchmaking algorithm.

For ranked matches, LoL utilizes a matchmaking algorithm based on a hidden Elo rating system, similar to the one originally used by chess players [26]. This matchmaking algorithm has been continuously improved upon since the release of the game and is used to create matches consisting of players of approximately equal skill level [10, 11, 13]. In 2020, Riot released a new matchmaking algorithm that better identifies player skill level [13].

When a player decides to play a ranked match, they will be matched with nine other players (via the matchmaking algorithm) who are then separated into two teams. Ranked matches are played with a minimum of eight strangers. Players may choose to enter matchmaking alone or with a similarly ranked partner, in which case they will be placed on the same team as their partner [26].

Since mechanical expertise is important to a player's performance, it seems logical that players will select champions that they have high expertise on. However, this may not always be the case. An important factor of champion selection is team composition. An optimal team has a combination of different roles in the game [21]. Certain champions have different roles, with some examples being "Tank" (champions with high defense), "Mage" (long ranged champions with burst damage and crowd control), and "Support"(champions with support ability) [15]. However, since players are playing with strangers, they cannot predetermine optimal team compositions. For instance, if no players on the team typically play "Tank" champions, one player may be forced to play a "Tank" champion, even though they may not have high mechanical expertise on it.

2.2 LoL Game Outcome Predictors

In this section, we describe previous work on LoL game outcome predictors. Most MOBA game outcome predictors are focused on two popular MOBAs: LoL and Dota 2. For the purposes of this paper, we focus only on LoL predictors due to the differences in gameplay between LoL and Dota 2. For instance, Chen et al. [5] indicated that LoL has more diverse skill compositions than Dota 2.

There are several predictors that use pre-match data (i.e., before the match begins) like our predictor. However, the majority of these pre-match predictors have an accuracy around 55-60% (e.g., [5, 20, 24]). Similar to our work, Chen et al. [5] used player-champion experience to predict game outcomes. Using a combination of player-champion experience, champion experience, and player experience, they predicted game outcomes with an accuracy of 60.24% using logistic regression. They found that player-champion experience was more influential than both champion and player experience on game outcomes. We further expand on this work by investigating multiple features of player-champion experience, since Chen et al. [5] used only one feature to describe player-champion experience.

Within-match predictors predict the winner while the match is ongoing. Typically, as the game time elapses, the accuracy increases. Data such as kill difference, gold difference, and towers destroyed can help determine which team has the lead and will ultimately win the game. Lee et al. [22] found that within-match data can predict the winner with 62.26% accuracy at 5 minutes elapsed and 73% accuracy at 15 minutes elapsed using Random Forest (RF). Similarly, Silva et al. [25] predicted the winner with an accuracy of 63.91% at 5 minutes elapsed and 83.54% at 25 minutes elapsed using a recursive neural network. Unlike these predictors, our model does not rely on within-match data and instead predicts the match outcome before the match has begun, deviating entirely in functionality and application.

Ani et al. [1] found that pre-match data can predict the game outcome of professional LoL games with very high accuracy (>90%) using RF trees and ban data (each team is allowed to ban up to five champions). However, we focus on ranked gameplay rather than organized professional gameplay. Unlike professional games, ranked games are designed to have fair teams consisting of strangers and support all varieties of skill levels. Additionally, professional games contain a much different playerbase from ranked games, as they are only played among a couple dozen players.

3 METHODOLOGY

3.1 Dataset

Riot Games provides a public API [9] containing several types of data pertaining to players and their matches. This data contains features such as champion mastery, games played per season, and win rates. Since our methodology involves training machine learning models with data from ranked matches, we used this API to query random players and filter their most recent ranked match. Using this method, we pulled a total of 5000 unique ranked matches for our dataset. Any duplicate matches were removed. From these matches, we recorded general match information, as well as information about each player, specifically because they describe a player's experience on a given champion:

- **Champion mastery points** - An integer $\in [0, \infty]$ approximating the player's lifetime experience on the given champion [7]. Champion mastery points are accrued by playing matches on the champion. The mean champion mastery points of players in our dataset was 122,368.44, while the max value was 9,364,624.

- **Player-champion win rate** - A percentage (0.0 to 1.0) indicating the player's victory ratio for ranked games played on the given champion during the 2020 ranked season.
- **Season total number of games played on the champion** - An integer $\in [0, \infty]$ signifying the total number of ranked games the player has played on the given champion during the 2020 ranked season. The mean season total number of games of players in our dataset was 58.95, while the max value was 1,819.
- **Number of recent games played on the champion** - An integer $\in [0, 20]$ representing the number of ranked games that the player has played on the champion within their last 20 ranked games during the 2020 ranked season.

These features represented each player's experience levels on their chosen champion and helped us determine whether player-champion experience can predict match outcomes. Overall, our methodology allowed us a large pool of data to build upon and use for our model, while also stripping away personally identifiable data from each player.

3.2 Feature Selection

Using Pearson's correlation test, we discovered that player-champion win rate and champion mastery points were the only features that correlated with the outcome of a match, and that the total number of games played and number of recent games played on a particular champion had no effect on the outcome of the match. Furthermore, during preliminary analysis, our models performed worse when the other features were included in the training dataset.

For each match, we derived additional features to better reflect each team's overall player-champion experience. For both teams, we added the team's average, median, coefficient of excess kurtosis [32] (i.e., how normal the distribution is), coefficient of skewness [32], standard deviation, and variance of the players' champion win rate and of their champion mastery points. The final training dataset contained 44 features per match (2 features per player for 10 players and 12 features per team).

3.3 Models

We chose to explore several machine learning models based on the success of previous work that also investigated MOBA game outcomes [1, 2, 22, 24, 29]. We investigated Support Vector Classifiers (SVC) [19], k-Nearest Neighbors (kNN) [18], Random Forest (RF) trees [17], Gradient Boosting (GBOOST), and Deep Neural Networks (DNN) [17]. In this section, we briefly explain each model, as well as any parameters that we used for our implementations.

3.3.1 Support Vector Classifier. An SVC is a form of Maximal Margin Classifier that attempts to form a $(p-1)$ dimensional separable hyperplane on a p dimensional feature space [18]. SVCs intentionally misclassify training observations in an attempt to improve the classification of remaining observations [18]. We used scikit-learn's [28] implementation of SVC with the following modified parameters: `C=8`, `coef0=1`, `kernel='poly'`, and `tol=1e-2`. We opted to use a polynomial kernel over a radial-basis-function kernel [18] because the polynomial kernel showed better classification accuracy.

3.3.2 k-Nearest Neighbors. The kNN [27] algorithm takes a given record x and compares it with the k closest records from its corresponding dataset, with closeness defined using some distance measure. We used scikit-learn's [28] implementation of kNN with the following modified parameters: `leaf_size=5`, `n_neighbors=600`, `p=1` and `weights='distance'`. For this model in particular, we noticed that including more features caused our overall prediction accuracy to drop dramatically. In order to circumvent the curse of dimensionality [18], we reduced our feature space to only include the team's average win rate.

3.3.3 Random Forest Trees. RF trees [17] use bagging to create multiple base decision tree classifiers and aggregate them into a single model. This method can yield a model with less variance and overfitting in comparison to classic decision trees. We used scikit-learn's [28] implementation of RF trees [17] with the following modified parameters: `n_estimators=350`, `min_samples_leaf=3`, `min_samples_split=10`, and `max_features=7`.

3.3.4 Gradient Boosting. GBOOST is another ensemble method that sequentially adds predictors to its ensemble and tries to fit each new predictor based on the residual errors of the previous predictor [16]. We used scikit-learn's [28] implementation of GBOOST, whose default estimator is a Decison Tree [16], with the following modified parameters: `learning_rate=0.14` and `n_estimators=55`.

3.3.5 Deep Neural Networks. We utilized Keras [6], a Python deep learning API, to build our model. We decided to go with a pyramid network architecture based on examples described in [16], as we noticed that it performed better than networks with hidden layers with the same number of neurons in our preliminary experiments. We used the following network topology to define our model:

- A flattened input layer resulting in a 1×44 output.
- Alternating dropout, normalization, and dense layers for a total of 15 layers (5 dropout, 5 normalization, and 5 dense layers). Each group of alternating layers had 160, 128, 64, 32, and 16 neurons, in that order.
  - Each dropout layer had a dropout rate of 0.69%.
  - Each normalization layer utilized batch normalization.
  - Each dense layer used Exponential Linear Unit (ELU) activation [16], He initialization [16].
- A 1×1 dense layer with Sigmoid activation

Notably, we used ELU activation, He initialization and batch normalization to deal with unstable gradients [16], which led to more accurate model predictions. We also added dropout layers to prevent overfitting [30].

4 EXPERIMENTS AND RESULTS

4.1 Experimental Settings

For our experiments, we split our dataset into a training and testing dataset. For the training dataset, 80% of the matches (4000 matches) were randomly chosen, while the other 20% (1000 matches) were chosen for the testing dataset. We utilized stratified k-fold validation [4] with $k = 10$ to evaluate our models. Stratified k-fold validation tries to keep the class distribution of the testing dataset as similar to the actual class distribution as possible, which can lead to better

Table 1: The performance of each machine learning model, including the accuracy, 95% confidence interval (CI), standard deviation, and standard error.

Model	Mean Accuracy ± CI	Std. Dev.	Std. Error
SVC	74.3% ± 1.21%	1.7%	0.54%
KNN	72.7% ± 1.23%	1.2%	0.38%
RF	74.7% ± 1.2%	2.0%	0.63%
GBOOST	75.4% ± 1.19%	5.25%	1.66%
DNN	75.1% ± 1.2%	1.9%	0.60%

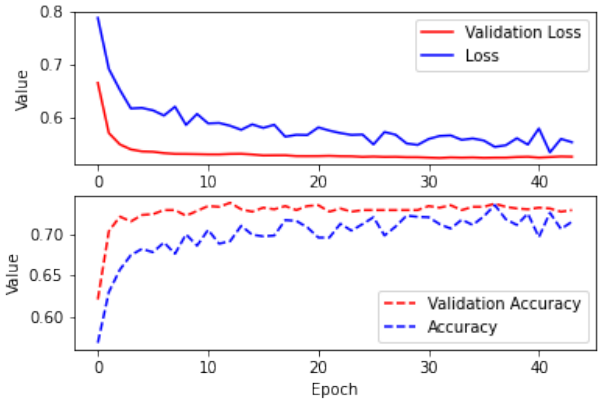

Figure 1: The loss and accuracy of our deep neural network while training and validating.

accuracy. Each model's average accuracy, its standard deviation, and its standard error are shown in Table 1.

4.2 Results

From our experiment results (see Table 1), we can see that all models predicted the winner with relatively high accuracy (>70%). GBOOST had the highest accuracy with 75.4%, which was marginally higher than our DNN model's of 75.1%. However, the standard error of GBOOST was much greater than all of our other models. Therefore, we consider our DNN model to be most suitable for game outcome prediction since its standard error was low. Interestingly, from Figure 1, we can see that the validation accuracy is higher than the training accuracy of our DNN model. This is a byproduct of using dropout layers [16]. Using dropout layers artificially increases the difficulty of learning for the network during training, with the intent to increase test/validation accuracy.

5 LIMITATIONS AND FUTURE WORK

It is important to note that our work is limited to our specific domain. We only analyzed ranked matches from the North American (NA) server that occurred in 2020 within ranks Iron to Diamond. These results may not be similar for other servers such as the Oceania (OCE) server, due to differences between the playerbases, and may not reflect highly skilled play (top 1%). Additionally, the playerbase may change over time. In the future, it would be interesting to investigate other regional servers (e.g., EUNE, EUW, OCE) to determine if player demographics have an effect on prediction accuracy. Furthermore, it may be of value to determine player-role experience and determine if this could serve as a similar predictor of a player's skill.

6 CONCLUSION

In this paper, we introduced a DNN machine learning model that can predict the accuracy of ranked LoL matches with an accuracy of 75.1% using player-champion experience. To reflect each player's champion experience in a ranked match, we extracted multiple features regarding player statistics on their chosen champion. However, we found that only champion mastery points and ranked win rate on the selected champion were suitable for predicting game outcomes. We used this data to create summary features that represented the overall player-champion experience of the team. Using these summary features, we also found high accuracy (73%-75%) using other machine learning models, such as random forest trees and support vector classifiers, showing that player-champion experience can be a strong indicator of team performance in general.

Our results offer insight into the importance of champion selection and matchmaking design. Since we are able to determine the winner with reasonable accuracy after champion selection, this implies that even after the skill-based matchmaking, there is still a wide variance in team skill before gameplay begins. Fair matchmaking is important for player experience [31], and ranked matches in League of Legends are designed to be as fair as possible [11]. However, the current skill-based matchmaking system does not consider a player's champion selection, which can widely vary their skill level and thus cause the match to feel imbalanced. We show that the outcome of a ranked match can be determined with relatively high accuracy before gameplay begins based solely on player-champion experience. Although this problem cannot be resolved with the current matchmaking system, our results imply that players should limit their choices to champions that they have already mastered, regardless of team composition. However, most players can only master a handful of champions [5]. This can be an issue for gameplay variety as players may choose not to experience the large roster of champions, in favor of playing the same few champions in order to win games.

REFERENCES

[1] R. Ani, Vishnu Harikumar, Arjun K. Devan, and O. S. Deepa. 2019. Victory prediction in league of legends using feature selection and ensemble methods. *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019* Iciccs (2019), 74–77. https://doi.org/10.1109/ICCS45141.2019.9065758

[2] Mochammad Anshori, Farhanna Mar'i, Mukhammad Wildan Alauddin, and Fitra Abdurrahman Bachtiar. 2018. Prediction Result of Dota 2 Games Using Improved SVM Classifier Based on Particle Swarm Optimization. *3rd International Conference on Sustainable Information Engineering and Technology, SIET 2018* (2018), 121–126. https://doi.org/10.1109/SIET.2018.8693204

[3] M. Aung, V. Bonometti, A. Drachen, P. Cowling, A. V. Kokkinakis, C. Yoder, and A. Wade. 2018. Predicting Skill Learning in a Large, Longitudinal MOBA Dataset. *IEEE Conference on Computatonal Intelligence and Games, CIG* 2018-August (2018). https://doi.org/10.1109/CIG.2018.8490431

[4] José Ramón Cano, Francisco Herrera, and Manuel Lozano. 2007. Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability. *Data & Knowledge Engineering* 60, 1 (2007), 90–108. https://doi.org/10.1016/j.datak.2006.01.008 Intelligent Data Mining.

[5] Zhengxing Chen, Yizhou Sun, Magy Seif El-Nasr, and Truong Huy D. Nguyen. 2017. Player skill decomposition in multiplayer online battle arenas. *arXiv* (2017). arXiv:1702.06253

[6] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

[7] Tiffany D Do, Dylan S Yu, Salman Anwer, and Seong Ioi Wang. 2020. Using Collaborative Filtering to Recommend Champions in League of Legends. In *2020 IEEE Conference on Games (CoG)*. 650–653. https://doi.org/10.1109/CoG47356.2020.9231735

[8] Scott Donaldson. 2017. Mechanics and Metagame: Exploring Binary Expertise in League of Legends. *Games and Culture* 12, 5 (2017), 426–444. https://doi.org/10.1177/1555412015590063

[9] Riot Games. [n.d.]. Riot Developer Portal. https://developer.riotgames.com/apis

[10] Riot Games. 2019. /dev: Making Matchmaking Better. https://nexus.leagueoflegends.com/en-us/2018/03/dev-making-matchmaking-better/

[11] Riot Games. 2019. /dev: Updates to Ranked for 2019. https://nexus.leagueoflegends.com/en-us/2018/04/dev-updates-to-ranked-for-2019/

[12] Riot Games. 2020. /dev: A look back at League in 2020. https://na.leagueoflegends.com/en-us/news/dev/dev-a-look-back-at-league-in-2020/

[13] Riot Games. 2020. /dev: Updates on 2020 Ranked & Matchmaking. https://na.leagueoflegends.com/en-us/news/dev/dev-updates-on-2020-ranked-matchmaking/

[14] Riot Games. 2020. Game Updates. https://na.leagueoflegends.com/en-us/news/game-updates/

[15] Gege Gao, Aehong Min, and Patrick C. Shih. 2017. Gendered design bias: Gender differences of in-game character choice and playing style in league of legends. *Proceedings of the 29th Australian Conference on Computer-Human Interactio* (2017), 307–317. https://doi.org/10.1145/3152771.3152804

[16] Aurélien Géronen. 2017. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Sebastopol, CA.

[17] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning.* Springer New York Inc., New York, NY, USA.

[18] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated.

[19] V. Kecman. 2005. *Support Vector Machines: Theory and Applications.* Vol. 177. 1–47 pages.

[20] Cheolgi Kim and Soowon Lee. 2019. Predicting Win-Loss of League of Legends at the Start of the Game Using Machine Learning Techniques. In *Proceedings of ISSAT International Conference on Data Science in Business, Finance and Industry.*

[21] Yubo Kou and Xinning Gui. 2014. Playing with strangers: Understanding temporary teams in league of legends. *CHI PLAY 2014 - Proceedings of the 2014 Annual Symposium on Computer-Human Interaction in Play* (2014), 161–169. https://doi.org/10.1145/2658537.2658538

[22] Sang Kwang Lee, Seung Jin Hong, and Seong Il Yang. 2020. Predicting Game Outcome in Multiplayer Online Battle Arena Games. *International Conference on ICT Convergence* 2020-October (2020), 1261–1263. https://doi.org/10.1109/ICTC49870.2020.9289254

[23] Xiangqian Li, Liang Huang, Bingxin Li, Haoran Wang, and Chengyang Han. 2020. Time for a True Display of Skill: Top Players in League of Legends Have Better Executive Control. *Acta Psychologica* 204 (2020). https://doi.org/10.1016/j.actpsy.2020.103007

[24] Lucas Lin. 2016. League of Legends Match Outcome Prediction. (2016).

[25] Antonio Luis, Cardoso Silva, Gisele Lobo Pappa, and Luiz Chaimowicz. 2018. Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks. *Proceedings of SBGames 2018* (2018), 639–642.

[26] Marçal Mora-Cantallops and Miguel Ángel Sicilia. 2018. Exploring player experience in ranked League of Legends. *Behaviour and Information Technology* 37, 12 (2018), 1224–1236. https://doi.org/10.1080/0144929X.2018.1492631

[27] Mahmoud Parsian. 2015. *Data Algorithms: Recipes for Scaling Up with Hadoop and Spark* (1st ed.). O'Reilly Media, Inc.

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

[29] Aleksandr Semenov, Peter Romov, Sergey Korolev, Daniil Yashkov, and Kirill Neklyudov. 2017. Performance of machine learning algorithms in predicting game outcome from drafts in Dota 2. *Communications in Computer and Information Science* 661 (2017), 26–37. https://doi.org/10.1007/978-3-319-52920-2_3

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[31] Maxime Véron, Olivier Marin, and Sébastien Monnet. 2014. Matchmaking in multi-player on-line games: Studying user traces to improve the user experience. *Proceedings of the 24th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2014* (2014), 7–12. https://doi.org/10.1145/2578260.2578265

[32] Daniel Zwillinger and Stephen Kokoska. 1999. *CRC standard probability and statistics tables and formulae.* Crc Press.