Predicting Valorant Pro Play Standings: A Machine Learning Approach

Peilin Wu
Department of Computer Science
Emory University

Harry Chen
Department of Computer Science
Emory University

James Song
Department of Computer Science
Emory University

Chris Feng
Department of Computer Science
Emory University

Abstract

In this paper, we analyze the various factors that go into a competitive Valorant game and attempt to predict the result of a game using machine learning algorithms based on these factors.

We used scraped data from vlr.gg as our data set and applied various preprocessing and models, such as neural networks, to predict the outcome of these games.

We found that we can predict games relatively well, but improvements can and should be made on future iterations.

1 Introduction

Over the past few decades, eSport, or competitive video gaming, has rapidly emerged as a thriving industry, fueled by the increasing number of people including teenagers and adults who play games as a form of entertainment during their free time. With market revenue reaching 1,384 million [10] U.S. dollars and an audience of 532.1 million viewers [11] in 2022, the potential for growth in the eSport market is considerable. One game, in particular, Valorant, has garnered significant attention from both gamers and the medias due to its innovative gameplay and professional tournaments in recent years. With the exception of games like League of Legends, which have a large existing fanbase, Valorant eSports has now secured the fifth [4] position in terms of viewership on the PC platform. With the expansion of the eSports industry, predicting the performance of both teams and outcome of the game has become a crucial area of research.

The primary challenge in predicting the outcome of a game is determining how to measure the win rate for each team at the beginning. In such tournament, players select different agents to fight with the enemy team in each game and their performance throughout the tournament determines their final ranking and the prizes they will receive. As a competitive game, there are numerous factors that can improve a team's chances of winning. Certain agents have a special affinity with particular maps, while the combination of certain agents can result in a more powerful effect. Therefore, the selection of agents based on different map is a crucial pre-game strategy that can determine a team's possibility of success in winning games.

To address the above challenge, we have put forward four machine learning models to investigate whether the individual's selection of agent based on different map can be used as a predictor of the team's final outcome in the tournament. Enabled by the Naive Bayes model, we were able to compute the conditional probabilities of a team's win given the controlling factor such as agents and map. Furthermore, Perceptron model was trained on the dataset to classify and predict the team's win or loss based on the agents selected by each team player for each map. A Logistic regression model was also used to build linear classifier and uncover the relationship between tournament outcome and the individual's selection of agents. Lastly, a more complex artificial Neural network model was also used to capture more intricate patterns and relationships. To evaluate the performance of each machine learning model, we adopted the measure of F1-score and Accuracy as our metrics and conducted extensive experiments, demonstrating that our study achieves decent or competitive performance compared to the state of the art, e.g., accuracy of 64.4% on Logistic regression and F1-score of 70.2 % on multi-layer perceptrons, etc.

The findings of this research have practical implications for eSport betting, as they provide insight into which team is more likely to win based on the individual selection of agents for specific maps. Additionally, team managers can improve their pre-game strategies by visualizing the advantages of certain agents in specific maps, leading to a potentially better chance of success in the tournament.

Specifically, the main contributions of our study are as follows:

1. We create the newest Valorant pro match dataset based on the past two years match statistics to predict the outcome of the future tournament

2. We propose a novel and formal model comparison to provide comprehensive evaluation of the performance of four machine learning models

3. We introduce an approach of using out-game data to find the effects of players, agents, and maps to the outcome of each game in the tournament

2 Background

2.1 Valorant: Game Play

Figure 1: Example of Valorant pro matches.

Valorant is a 5v5 character-based tactical first-person-shooter game developed and published by Riot Games [1]. In Valorant, each match is played by two opposing teams: attackers and defenders. A match contains 3 games, and the team that wins two games will win the match. In each game, the two teams will be placed on one of the 7 maps. The attackers need to plant the Spike in one of the two or three designated areas and let it explode, and the defenders need to stop attackers from planting the Spike or defuse it before it explodes. The team that completes the objective will earn one score (called one **round**), and the first team that reaches 13 scores in total would win the game. After 12 rounds, the attackers and defenders will switch sides. In case of a tie situation of 12-12, the game will enter overtime where a team needs to win two rounds in a row to win the whole game.

In each match, every player needs to choose one character with unique abilities, called "agents", from 21 characters. Though the two teams can choose the same agent, the agents picked in one team has to be unique. Each agent is categorized into one of the four types (Controller, Sentinel, Initiator, and Duelist) based on his/her ability set. The type of an agent represents his/her position and duty in the team. In the beginning of each round, the players need to purchase their abilities, guns, and armors for the round. In the first round of the game and the first round after switching sides, each player would have 800 initial gold to purchase these items. For the following rounds, the gold is earned based on each player's performances in the previous rounds.

2.2 Related Work

As mentioned above, eSports is currently one of the major international and popular sports. A large number of matches have been held every year, generating a huge amount of data and opportunities for machine learning (ML) studies.

Extensive researches have been done on applying machine learning on eSports related predictions. These tasks include but are not limited to predicting match outcomes, recommending items, predicting the ranking of players, identifying roles, etc. [3]. Among these tasks, the outcome prediction has promising commercial value, and thus is one of the most important problems in the field [3]. However, most of the researches focused on Multiplayer Online Battle Arena (MOBA) games, such as League of Legends (LoL) and Defense of the Ancients 2 (DotA2), because of their consistent high popularity [5]. These researches used ML models to predict game results based on different features: pre-game features [13], in-game features [5], and post-game features [3]. Pre-game features are the character ban/picks and item selections before the match starts. In-game features are mainly used for live predictions, such as the player or the team's performances within the next few minutes [6]. Post-game features are summaries generated after a game is finished. These features include kill/death ratios, damages, gold earned, etc. [3].

Since Valorant is released in 2020, the game and its professional eSport matches are still relatively young compared to other popular eSports. To our best knowledge, there has not been any researches conducted on applying ML on Valorant eSprots predictions. Therefore, we plan to explore the feasibility of using ML models on Valorant professional match result predictions using pre-game features similar to the related works done on MOBA games.

3 Methods

Firstly we preprocessed the data. The first thing we did was to group the entire dataset by each game played. After grouping, we horizontally concatenated every player in the dataframe such that it became a 1×N array. After that, we one-hot encoded all the categorical variables such that they became 1s and 0s.

Using that array, we applied several different decoding techniques on the array, which represent a single game, to predict the game outcome.

3.1 Logistic Regression

First, we ran logistic regression. It is an algorithm that applies the Sigmund function to the data and then gives out the probability of the event happening. To use it as a linear classifier, if $Pr(Event = 1|x) \geq 0.5$, then we say it belongs to the group True, otherwise, it belongs to the group false.

The model is shown below:

$$\begin{cases} Pr(Team1Win = 1|x) = \Lambda(\beta_0 + \beta_1 + ... + \beta_N + \mu) \geq 0.5 : 1 \\ Otherwise : 0 \end{cases}$$

We choose logistic regression because it has a very simple implementation and training process yet an excellent compared with some other more complex models according to the previous work on other esport games. However, as a linear classifier, linear regression may suffer from its simplicity and produce a relatively bad result in terms of classification.

3.2 Naive Bayes

The second method we ran was Naive Bayes. This algorithm uses the Bayes equation

$$\begin{cases} P(A|B) = \frac{P(B|A)*P(A)}{P(B)} : 1 \\ Otherwise : 0 \end{cases}$$

Here, specifically, we use Bernoulli NB, which is the best type of Naive Bayes for binary features. And since we one-hot encoded the dataset, we have a lot of binary features. In Bernoulli NB, the prior becomes the Bernoulli distribution, and the P(A|B) is calculated based on that.

We choose Naive Bayes because we would like to add a generative model for supervised learning into comparison. Besides, as Valorant being a First Person Shooting (FPS) game, the personal performance is usually considered as more important than the teamwork effects. This matches with the assumption that different factors in a sample are independent to each other. Also, Naive Bayes provides not only class predictions but also probabilities for each class. This can help quantify the uncertainty of the prediction and make more informed decisions.

3.3 Perceptron

The third method we ran was a single-layer perceptron. This algorithm is just a one-layer perceptron where the perceptron is a linear equation that predicts whether or not Team 1 would win.

$$\begin{cases} Layer \geq 0 : 1 \\ Otherwise : 0 \end{cases}$$

We choose to consider perceptron because it serves as the foundation for more complex neural network models, such as multi-layer perceptrons (MLPs) and other deep learning architectures. Starting with a simple perceptron model can us understand the underlying mechanisms and data representation before moving on to more advanced models.

The perceptron acts as a linear equation such that:

$$Team1Win = \Lambda(\beta_0 + \beta_1 + ... + \beta_N + \mu)$$

and each β is updated after each case, where an incorrect prediction will yield a +1 or -1 on the β used.

3.4 Neural Network

Lastly, we ran a neural network. The neural network used in this experiment has the same idea as a multi-layer perceptron (MLP), where the middle layers are called hidden layers. It will give us an N->1 prediction and give us whether team 1 wins or not.

We choose to use MLP because it can learn non-linear decision boundaries, allowing them to capture complex relationships between features that might be present in eSports data. This can lead to more accurate predictions compared to linear models. The hidden layers in an MLP enables the model to learn higher level representations of the input features, which allows us to effectively extract more meaningful information that can contribute to better predictions.

4 Experiments

In this section, we will talk about the details of the dataset we used in Section 4.1, the detailed implementation or hyperparameters used for each model in Section 4.2, and the results for different models in Section 4.3 with brief interpolation and comparison.

4.1 Dataset and Preprocessing

4.1.1 Data Collection

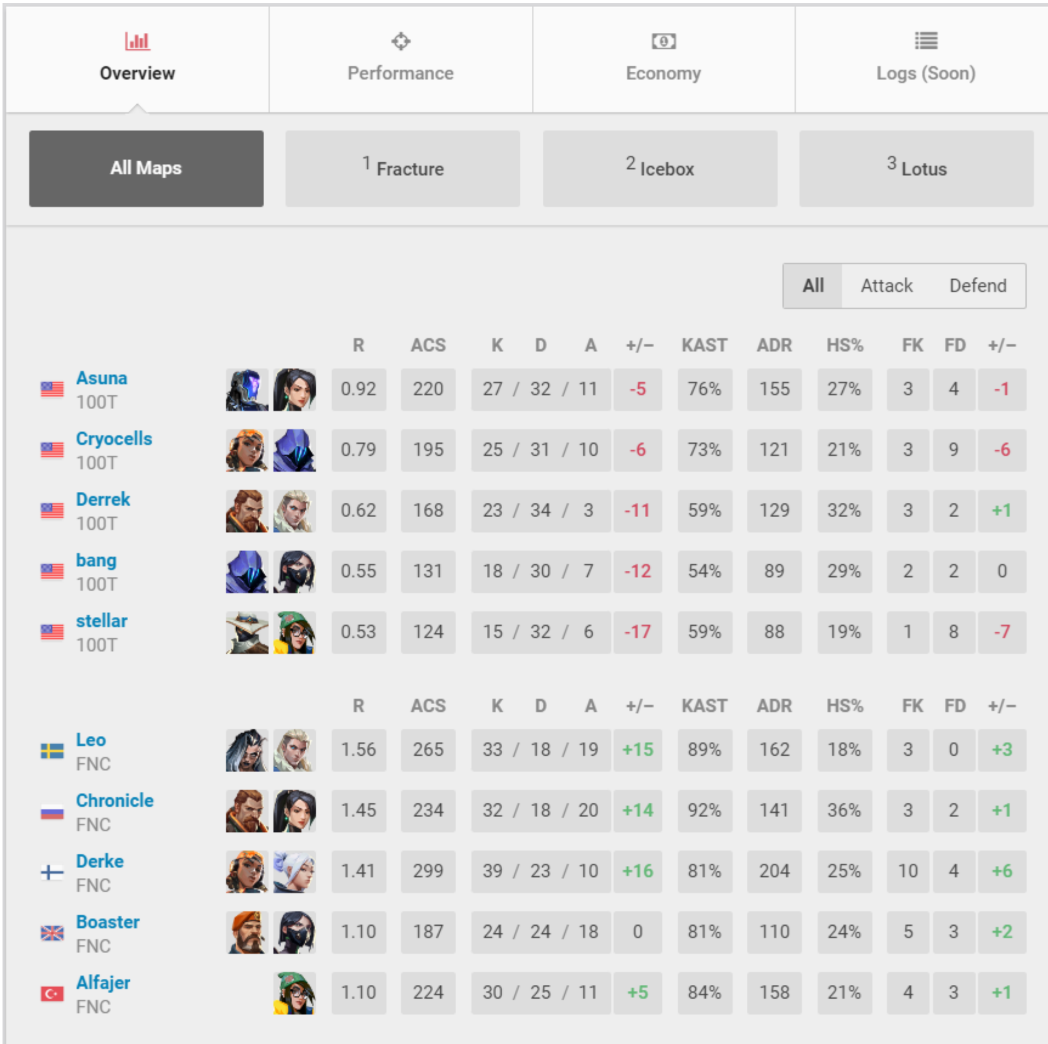

Figure 2: Interface of vlr.gg.

In this experiment, we utilize the data from the past Valorant pro matches. Generally there are three primary data source: vlr.gg[14], rib.gg[8], and thespike.gg[12]. All of them provide detailed and thorough statistics about pro players' personal performance and contributions to the team up to each game, which has one map exactly. Examples of these personal performance and contributions are kills, deaths, assists and so on. All three websites also gives comprehensive scores like average combat score (ACS), kill, assist, survive, trade percentage (kast), adr (average damage per round). However for map that has significant difference in score like 13:1 or 13:2, the data for one side may be skewed. rib.gg does offer a paid services for more detailed map statistics and in-game real time data, which is out of our consideration in this experiment because the price is high for large amount of data.

These three data sources have their own advantages and disadvantages. vlr.gg and thespike.gg provide their own comprehensive rating scores, which has very similar algorithms that take other data such as kills and deaths into account. rib.gg is the only website that gives the number and success rate of clutches (the ability to win the round at the end of each round). thespike.gg is the only website that gives the success rate of first blood (first kill in each round). vlr.gg has the most active community and thus provides help in improving the speed and accuracy of data update.

We end up choosing the vlr.gg as our primary source of data because of its updating speed and easy to use interface. As a open esports community, vlr.gg allows scraping content from the webiste. We choose to use beautifulsoup4[9] for html analysis and requests library for acquiring html file for each game in Python because all the data are included in the html file of the website while no API is provided for directly requesting data from the backend server.

In terms of the size of the primary data, we collect all the matches between April 2021 and April 2023, which is about the past two years' data counting from the date of the beginning of this work. The match statistics earlier than April 2021 on vlr.gg has significant amount of missing values in the data which we decided not to include. We also eliminate all the matches that has significant amount of in-game player changing since that may cause the inconsistency of the input data size to the machine learning model. The aforementioned elimination provides us about 250000 data samples each with 47 features. Each data sample represents a player for a team in a game. 5 consecutive samples form a team and 10 consecutive samples form a game. The raw data includes the following attributes displayed in Figure 2.

```
name: str
team: str  # might be different for the same team
agent: str
rating: float
acs: int  # average combat score
k: int  # kills
d: int  # deaths
a: int  # assists
tkmd: int  # total kills minus deaths
kast: float  # kill, assist, survive, trade %
adr: int  # average damage per round
hs: float  # headshot %
fk: int  # first kills
fd: int  # first deaths
fkmd: int  # first kills minus first deaths
t # attack side
ct # defend side
```

Figure 3: Attribute name, type, and brief explanation for raw data.

All the raw data and code for scraping data from vlr.gg are open sourced and have been published to Kaggle `https://www.kaggle.com/datasets/qualidea1217/valorant-pro-matches-since-april-2021`.

4.1.2 Data Preprocessing

The raw data collected from vlr.gg contains each pro-players' performances in each game and other features related to the match, including their teams, maps, champion selections, match time, in-game performances, etc (Figure 3). Every row represents a player's performances in a single game. Every five rows combined together are the players who played in the same game for the same team and their performances. And every ten rows combined together represent the players of both teams that played against each other in one game.

We firstly removed the irrelevant features, such as match date and time, from the dataset. Since we aim to conduct pre-game predictions, we also dropped the player's performance features of each game, such as the number of kills, number of deaths, headshot ratios, etc. These performance features are only available after a game is finished and thus conflict with our aim. Then, to train the model to predict the result of a single game, we need to append data from the same game into a single row. The appended and processed dataset contains the following features:

- **Map**: one hot encoded all map names representing which map the game was played on.
- **Team_1**: one hot encoded names of all teams, representing the first team participated in the game.
- **Team_1_Player**: one hot encoded names of all players, representing players in **Team_1**.
- **Team_1_Agents**: one hot encoded names of all agents, representing **Team_1**'s agent selections.
- **Team_2**: Same to **Team_1** but representing the second team participated in the game.
- **Team_2_Player**: Same to **Team_1_Player**, but representing players in **Team_2**.
- **Team_2_Agents**: Same to **Team_1_Agents** but representing **Team_2**'s agent selections.
- **Result**: binary label indicating which team won the game. "1" represents **Team_1** won and "0" represents **Team_2** won.

4.2 Model Implementation

For all the non-neural network models (Logistic Regression, Naive Bayes, Perceptron), we use the default implementations provided by scikit-learn[7]. For neural network, a simple multilayer perceptron (MLP) is implemented with Tensorflow 2.0[2]. Comparing to PyTorch, Tensorflow provides a easier implementation by using Keras API and predefined layers instead of writing our own class for building neural network. For the settings in TensorFlow, we used 5 hidden layers and 10 epochs to guarantee convergence while not overfitting. Specifically, we used 128*64*32*16*1 layout with all hidden layers to be full dense layer. Using number of nodes for each layer as the power of 2 is a convention in machine learning. We use ReLU as activation as it is by far the most widely used activation function for neural network. Because we are doing binary classification, we use binary cross entropy as loss function and sigmoid as output function. For training the network, we use Adam optimizer as it is the most effective optimizer comparing to sgd and other optimizers for gradient descent.

Despite the basic layout, we also use dropout technique to control the overfitting and improve the performance. Dropout is to randomly disable the output of any neuron's output with a certain chance. In this work we set the chance to 50%. In this way, the technique force the neural network to learn the more robust features with all neurons learning instead of rely on a few important neurons.

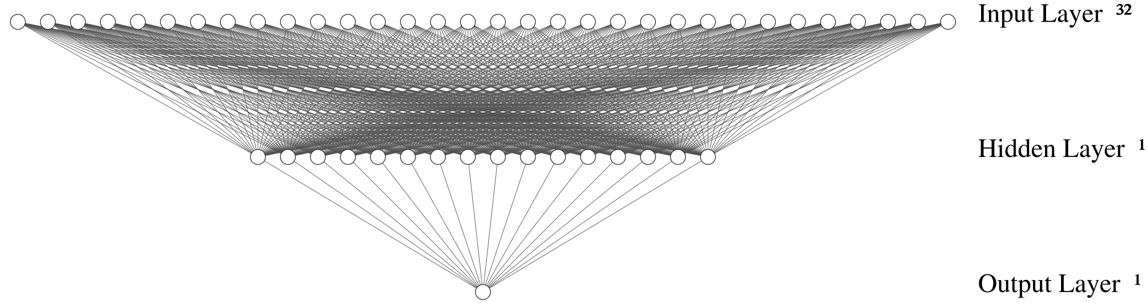

Figure 4: Part of the neural network (lower half) used in this work.

4.3 Results

	Logistic	**Naive Bayes**	**Perceptron**	**MLP**
Accuracy	0.6442	0.6033	0.6064	0.6369
F1	0.6926	0.6502	0.6504	0.7024

Table 1: Experiment Results

Table 1 contains the performances of all models. Both logistic regression and multi-layer perceptron seem to be the most suit models for Valorant professional match pre-game predictions, since they yielded the best accuracies and F1-scores among the models tested.

5 Discussion

Our research focuses on predicting the outcome for the game Valorant's eSports tournaments by utilizing the individual selection of agents based on different maps as a predictor. Through the implementation of four machine learning models: Naive Bayes, Perceptron, Logistic Regression, and multilayer perceptron (MLP), we show the relationships between agent selection, map, and a team's chances of success.

The key findings from our experiments demonstrated that our models achieved competitive performance, with Logistic Regression achieving an accuracy of 64.4% and MLP an F1-score of 70.2%. This indicates that the individual selection of agents for specific maps can be a viable predictor for a team's outcome in a game. The results have practical implications for pre-game strategy development for team managers, as they provide valuable insights into team performance based on agent and map choices.

Through the creation of the newest Valorant pro match dataset, a model comparison, and an approach to using pre-game data to analyze the effects of players, agents, and maps on tournament outcomes, our study has contributed to the field of eSports winning team prediction. Ultimately, the lessons learned from this experiment highlight the potential for machine learning models in improving eSports predictions, thereby informing decision-making in strategy planning and game success.

6 Contribution

Mike: Scrapped the data off of vlr.gg, wrote machine learning and preprocessing code, wrote the paper.

James: Wrote machine learning and preprocessing code, wrote the paper

Chris: Wrote machine learning and preprocessing code, wrote the paper

Harry: Wrote machine learning and preprocessing code, wrote the paper

Github

https://github.com/FF15enc/CS334_Final_Project

Kaggle

https://www.kaggle.com/datasets/qualidea1217/valorant-pro-matches-since-april-2021

References

[1] Riot games' competitive 5v5 character-based tactical shooter. URL https://playvalorant.com/en-us/.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[3] Kokten Ulas Birant and Derya Birant. Multi-objective multi-instance learning: A new approach to machine learning for esports. *Entropy*, 25(1), 2023. ISSN 1099-4300. doi: 10.3390/e25010028. URL https://www.mdpi.com/1099-4300/25/1/28.

[4] Esports Charts. All-time popular esports games by viewership. URL https://escharts.com/top-games?order=peak&year=all-time/.

[5] Victoria J. Hodge, Sam Devlin, Nick Sephton, Florian Block, Peter I. Cowling, and Anders Drachen. Win prediction in multiplayer esports: Live professional match prediction. *IEEE Transactions on Games*, 13(4):368–379, 2021. doi: 10.1109/TG.2019.2948469.

[6] Adam Katona, Ryan Spick, Victoria J. Hodge, Simon Demediuk, Florian Block, Anders Drachen, and James Alfred Walker. Time to die: Death prediction in dota 2 using deep learning. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, 2019. doi: 10.1109/CIG.2019.8847997.

[7] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] RIB.GG. Valorant analytics: Schedules: Matches: News. URL https://www.rib.gg/.

[9] Leonard Richardson. Beautiful soup documentation. *April*, 2007.

[10] Statista. esports market revenue worldwide from 2020 to 2025, . URL https://www.statista.com/statistics/490522/global-esports-market-revenue/.

[11] Statista. esports audience size worldwide from 2020 to 2025, by type of viewers, . URL `https://www.statista.com/statistics/490480/global-esports-audience-size-viewer-type/`.

[12] THESPIKE.GG. Valorant news: Latest events amp; competitive news. URL `https://www.thespike.gg/`.

[13] Markos Viggiato and Cor-Paul Bezemer. Trouncing in dota 2: An investigation of blowout matches. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):294–300, 2020. doi: 10.1609/aiide.v16i1.7444.

[14] VLR.gg. Valorant esports coverage. URL `https://www.vlr.gg/`.