Smurf Detection in Valorant: A Machine Learning Approach

University of California Davis, Iain Hennington, iehennington@ucdavis.edu

University of California Davis, Samarth Sridhara, sssridhara@ucdavis.edu

University of California Davis, Jovin Louie, jovlouie@ucdavis.edu

Department of Computer Science, University of California, Davis

1. Abstract

The purpose of this paper is to attempt to see if tabular data is sufficient in understanding player skill and detecting whether a user is playing at a rank they're not supposed to be in, or in other terms, "smurfing" in competitive multiplayer video games, specifically Valorant. We're scraping our data for our models from Tracker.gg, a popular video game stat tracking website. We also clean our data through winsorization, handling of NA values, hot-label encoding, removing outliers, and normalizing numeric columns, while plotting data with plots to see variable distributions. The methods we're using are machine learning models such as decision trees, logistic regression, naive bayes, etc. The results from our testing are that random forest is the best overall performing model, with our decision trees model coming in a close second.

ACM Reference Format: Ying-Jih Ding, Wun-She Yap, and Kok-Chin Khor. 2023. Profiling and Identifying Smurfs or Boosters on Dota 2 Using K-Means and IQR. In *IEEE Transactions on Games*. IEEE, USA, 1–9. https://doi.org/10.1109/TG.2023.3317053 Min L. Han, Byung I. Kwak, and Hyung K. Kim. 2022. Cheating and Detection Method in Massively Multiplayer Online Role-Playing Game: Systematic Literature Review. In *IEEE Access*. IEEE, USA, 1–1. https://doi.org/10.1109/access.2022.3172110 Hyeong Kim, Sang Lee, Jun Young Woo, and Hyung K. Kim. 2022. Justice League: Time-series Game Player Pattern Detection to Discover Rank-Skill Mismatch. In *16th International Conference on Automation (ICA 2022)*, 42–47. https://doi.org/10.1109/ica55837.2022.00014 Ruan Spijkerman and Elizabeth M. Ehlers. 2020. Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools. In *Proceedings of the 2020 3rd International Conference on Computational Intelligence and Intelligent Systems (CIIS 2020)*, November 13–15, 2020, Tokyo, Japan. ACM, New York, NY, USA, 6 pages. Ruan Spijkerman and Elizabeth Marie Ehlers. 2021. Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools. In *Proceedings of the 2020 3rd International Conference on Computational Intelligence and Intelligent Systems (CIIS '20)*. Association for Computing Machinery, New York, NY, USA, 87–92. https://doi.org/10.1145/3440840.3440857 Marcus Willman. 2020. Machine Learning to Identify Cheaters in Online Games. Master's thesis. Umeå University. Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-170973

1.1 Introduction

Smurfing is a severe issue in competitive multiplayer games. Smurfing is the concept of playing on an alternative account with a lower rank than their main account. Doing this can result in unfairness between teams, player dissatisfaction, and forfeiting of matches for those who are playing against a smurf. What is known of smurf detection in previous research is that it has been attempted to identify smurfs through the use of K-Means on features of players in Dota 2. There has also been research done on cheater detection. Cheater detection research has used keystroke logging as well as tabular data, although the research has been inconclusive, which raises our own question about smurf detection in first-person shooters through the use of tabular data: could smurfs be accurately identified based on player statistics for the recent Valorant competitive season?

2. Related Work

Related works have rarely touched on smurf detection within the past 3 years using traditional ML models on first-person shooters, validating our problem novelty. The closest that has been done was with smurf detection using K-Means in Dota 2 (Ding, Yap, Khor). The difference here is that they used unsupervised models and focused on dissimilarity between players to claim as a “smurf”, whereas we will be using supervised models. In addition, research on player mismatch using time-series and inputs to classify player mismatch in the popular video game Justice League through deep learning models has also been conducted (Lee, Kim, Young Woo, Kang Kim). However, this is a fighting game, and on top of this has an emphasis on time-series data instead of tabular data like our research. Besides this, there has been much interest from researchers in cheating detection, which is an adjacent subject but not quite the exact same thing as a smurf, and cheat detection hasn’t been focused on first-person shooters, but on adjacent genres. The closest research that we could find that was similar to our research was cheating detection in MMORPGs (Lan Han, Kwak, Kim). The issue with this, however, is that the research used in game services to gather data is a different type of video game, but does use similar ML models that we’ll be interested in using, like a decision tree and naive bayes.

3. Methodology

3.1 Data Collection

The general workflow of data collection was done with two scripts. The first script took in an initial player's username and user tag, and would go through that player's most recent game and log the username and user tag of those 9 other players. This process would be repeated until the recursive process stopped at the 800-player limit, which enabled no duplicate players. For the initial player, we had chosen a player from one of our recent matches (donkdonktime #donk). Our data collection only spanned the North American regions.

With a dataframe of our 800 players, it was time to gather data on these players' recent competitive season statistics. We used Tracker.gg, a website for tracking player data, for our data gathering. Now, to gather this data and manually search each player in our dataframe would be tiring, so we went with scraping our data, using a mixture of libraries and tools such as undetected-chromedriver, Tor, and Selenium. Our scraper was a script that sequentially went down our dataframe, and searched for the respective player on Tracker.gg. Once the player's tracker profile was accessed, if the player "did not exist" or was a private profile, we moved on to the next player. If we were able to successfully access the player's webpage, the scraper would search for if the player statistics div (HTML box for storing all the statistics widgets) existed and then grab the player's statistics. If there was any sort of error or issue due to rate limiting, the scraper would sit on cooldown for 30 minutes, picking back up again and cycling to a new IP address (thus avoiding hitting rate limiting or CloudFlare's bot protection).

3.2 Data Labeling

For data labeling, we decided to label the players as: normal player, suspicious, or likely a smurf. This was because the concept of a smurf is subjective, therefore, it is likely to say that our outcomes can be inconclusive and will serve as a mock concept for future endeavors in smurf detection with tabular data. We used a rule-based system to determine who is a normal player, suspicious, and likely a smurf. This rule-based system would compare each player to someone who is a known smurf and give them points based on similarity to the smurf's stats. The smurf we chose was streamer Sinatraa, a professional player who used the smurf account West Virginia #99999 on stream to avoid highly ranked players. We did not use his stats as one of one when comparing to other players because, naturally, not all players are of the same caliber as Sinatraa. Also, introducing this rule-based labeling is our proxy labeling, which automates labeling and keeps our concept of a smurf consistent at the cost of possible poor generalization and bias.

3.3 Data Preprocessing and Visualization

We used preprocessing techniques from Python and visualization techniques using R. NA values in the dataset were addressed by imputing numeric columns with their respective medians. Outlier treatment was applied using winsorization at the 5th and 95th percentiles, capping extreme values.. Standardization was performed on all the numeric columns using z-scores to ensure that all variables contributed equally to machine learning models. The categorical target variable smurf_label was one-hot encoded into binary columns to “normal player”, “suspicious”, and “most likely smurf”, which would be 0, 1, 2, respectively.

To visualize player behavior across the three smurf label categories, normal player, suspicious, and most likely smurf, we generated boxplots using ggplot2 and facet_wrap in R to present the distribution of different statistics using the smurf category as the category. The box plots showed that smurf-labeled players tended to have higher kill/death ratios and fewer deaths per game. In contrast, normal players tended to have the lowest values for many of the statistics represented.

4. Experiments and Results

4.1 Model Training and Evaluation

We trained and evaluated multiple supervised machine learning models on our labeled dataset to determine which approach would yield the most accurate smurf detection. The models were trained using an 80-20 train-test split with stratified sampling to maintain class distribution proportions. This resulted in approximately 354 training samples and 88 testing samples. We further employed 5-fold cross-validation during model training to ensure robustness and reduce overfitting.

4.2 Model Performance

We evaluated two primary supervised learning models, with the following results:

4.2.1 Logistic Regression

The logistic regression model achieved:

- Overall accuracy: 85.6%
- Macro average F1-score: 0.831
- Weighted average F1-score: 0.857

Actual / Predicted	Most Likely Smurf	Normal Player	Suspicious
Most Likely Smurf	15	0	1
Normal Player	1	60	7
Suspicious	3	4	20

This model demonstrated strong performance in identifying "most likely smurf" accounts with a recall of 93.8% (15 out of 16 correctly identified). However, it showed some weakness in classifying suspicious accounts, occasionally misclassifying them as normal players (7 instances) or as smurfs (4 instances).

4.2.2 Decision Tree

We also trained a decision tree classifier to evaluate how a single, interpretable model would perform compared to more complex ensemble methods. The decision tree achieved:

- Overall accuracy: 89.2%
- Macro average F1-score: 0.866
- Weighted average F1-score: 0.893

Actual / Predicted	Most Likely Smurf	Normal Player	Suspicious
Most Likely Smurf	13	0	3
Normal Player	2	62	4
Suspicious	0	3	24

This model demonstrated high precision and recall across all three classes. It correctly identified 81.2% of smurf accounts and achieved a recall of 91.2% for normal players. Notably, it performed strongly on the "suspicious" class with a recall of 88.9%, outperforming both the logistic regression and random forest models in that specific category.

4.2.3 Random Forest

After hyperparameter tuning via grid search across 2,400 parameter combinations, our optimized Random Forest classifier achieved:

- Best parameters: `{'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 7, 'min_samples_split': 2, 'n_estimators': 200}`
- Best cross-validated F1 score during training: 0.897
- Test set accuracy: 91.0%
- Macro average F1-score: 0.87
- Weighted average F1-score: 0.91

Actual / Predicted	Most Likely Smurf	Normal Player	Suspicious
Most Likely Smurf	16	0	0
Normal Player	1	67	0
Suspicious	5	4	18

The Random Forest model achieved perfect recall (100%) for the "most likely smurf" class, correctly identifying all 16 smurf accounts in the test set. It also demonstrated exceptional precision for the "suspicious" class (100%), meaning that when it classified an account as suspicious, it was always correct. However, the model showed some difficulty in correctly classifying players that were actually suspicious, with a recall of only 67% for this category.

4.2.4 Performance Comparison

Model	Accuracy	Macro Avg. F1	Weighted Avg. F1
Logistic Regression	85.6%	0.831	0.857
Decision Tree	89.2%	0.866	0.893
Random Forest	<u>91.0%</u>	<u>0.870</u>	<u>0.910</u>

The Random Forest classifier consistently outperformed the logistic regression and decision tree models across overall accuracy and weighted metrics. However, the decision tree model achieved a competitive balance of performance and interpretability, with particularly strong results on the “suspicious” class, suggesting its utility in scenarios where model transparency is a priority.

5. Discussion and Analysis

5.1 Model Interpretation

The performance comparison between our three models (91.0% accuracy for Random Forest, 89.2% for Decision Tree, and 85.6% for Logistic Regression) demonstrates that the relationship between player statistics and smurfing behavior is inherently complex and non-linear. This aligns with the multifaceted nature of player skill in tactical first-person shooters like Valorant, where performance is determined by a combination of mechanical skill, game knowledge, decision-making, and teamwork.

- <u>Most likely smurf:</u> The Random Forest achieved perfect recall (100%) for smurf detection, outperforming both the Decision Tree (81.2%) and Logistic Regression (93.8%). This suggests that high-skill smurf accounts exhibit distinctive statistical patterns that are best captured through ensemble methods that can identify complex combinations of features.
- <u>Normal player:</u> All three models performed well in identifying normal players, with the Random Forest achieving 99% recall compared to 91.2% for Decision Tree and 88% for Logistic Regression. This demonstrates that typical player behavior within a given rank exhibits consistent statistical patterns that can be reliably identified.

- Suspicious: Interestingly, the Decision Tree outperformed the other models in identifying suspicious accounts with 88.9% recall, compared to only 67% for Random Forest and 74.1% for Logistic Regression. This suggests that the decision boundaries for the "suspicious" category might be more effectively captured through the simpler, more interpretable splits of a decision tree than through the more complex ensemble approach of Random Forest.

6. Conclusion

In conclusion, we can see that the different models that were used to train the model were successful with a high recall rate for every model, achieving on average over a 70 percent recall rate with the 3 models being used. Unfortunately, the dataset was based on only one region, the control group was one player, and we also subjectively made labels for the different players, which can be problematic in the long run. In addition, our proxy labeling results in poor generalization and bias, meaning the results are more mock than conclusive. If we're able to find a way to get labels that are more accurate than our current method, this could make our data much stronger. Another simpler idea is to avoid machine learning for smurf detection in general, and reward smurf players (instead of detecting and banning) with more MMR to rank up faster, resulting in more evenly matched games for them. This could simply be done with a rule-based system or a threshold for a vital statistic from the player instead of an entire ML model. We also have to consider ethical considerations, which include watching behavioral patterns of players, which can lead to data privacy issues, and false positives can be a concern as well. In the future for this project, we were thinking of incorporating a semi-supervised machine learning model that may be able to enhance the accuracy of the labeled and unlabeled data in the future.

7. Contribution Statement

- Iain Hennington: Data gathering, labeling, “Abstract”, “Introduction”, and “Related Works” sections.
- Samarth Sridhara: “Data Preprocessing”, “Conclusion”, and “Abstract” sections.
- Jovin Louie: Model creation, “Experiments and Results”, and “Discussion and Analysis” sections.

References

Ding, Ying-Jih & Yap, Wun-She & Khor, Kok-Chin. (2023). Profiling and Identifying Smurfs or Boosters on Dota 2 Using K-Means and IQR. IEEE Transactions on Games. PP. 1-9. 10.1109/TG.2023.3317053.

Han, M. L., Kwak, B. I., & Kim, H. K. (2022). Cheating and Detection Method in Massively Multiplayer Online Role-Playing Game: Systematic Literature Review. *IEEE Access*, 1–1. https://doi.org/10.1109/access.2022.3172110

Kim, H., Lee, S., Woo, J. Y., & Kim, H. K. (2022). *Justice League: Time-series Game Player Pattern Detection to Discover Rank-Skill Mismatch*. *16*, 42–47. https://doi.org/10.1109/ica55837.2022.00014

Ruan Spijkerman and Elizabeth M. Ehlers. 2020. Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools. In *2020 The 3rd International Conference on Computational Intelligence and Intelligent Systems (CIIS 2020), November 13-15, 2020, Tokyo, Japan*. ACM, New York, NY, USA, 6 pages.

Ruan Spijkerman and Elizabeth Marie Ehlers. 2021. Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools. In Proceedings of the 2020 3rd International Conference on Computational Intelligence and Intelligent Systems (CIIS '20). Association for Computing Machinery, New York, NY, USA, 87–92. https://doi.org/10.1145/3440840.3440857

Willman, M. (2020). Machine Learning to identify cheaters in online games (Dissertation). Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-170973