Valorant Pro Match Analysis: Predictive Modeling Based on Team Performance and Player Statistics

Students

Ahmed Zarir Siddique

Dept. of Computer Science and Engineering

BRAC University

Dhaka, Bangladesh

ahmed.zarir.siddique@g.ac.bracu.bd

Mohammad Ryan Ur Rafi

Dept. of Computer Science

BRAC University

Dhaka, Bangladesh

mohammad.ryan.rafi@g.ac.bracu.bd

Hasnat Md. Imtiaz

Dept. of Computer Science

BRAC University

Dhaka, Bangladesh

hasnat.md.imtiaz@g.ac.bracu.bd

Faculties

Mr. Rafeed Rahman

Dept. of Computer Science and Engineering

BRAC University

Dhaka, Bangladesh

rahman.rafeed@bracu.ac.bd

Md. Tanvir Zahid

Dept. of Computer Science and Engineering

BRAC University

Dhaka, Bangladesh

ext.tanvir.zahid@bracu.ac.bd

Abstract—In the world of professional esports, predicting the outcome of a Valorant match is a crucial problem that has gathered significant interest. Being able to accurately forecast match outcomes has numerous practical uses, such as aiding teams in their strategic decisions, supporting bettors in their wagers, and providing game developers with crucial insights on the game's meta. The goal of this project is to create a machine learning model that can predict the outcome of Valorant matches and player stats based on data scraped from esports matches over the past two years. To construct and train a categorization model, we use Python and its sophisticated machine learning tools, such as Scikit-learn, Pandas, and NumPy. To accomplish this, we first clean and prepare the data before picking and refining the important attributes. Then we use models like the Random Forest Classifier and Decision Tree Regressor to train and predict the target. To evaluate the model's usefulness, we examine its performance using metrics such as accuracy and precision. Overall, this project is an excellent opportunity to obtain hands-on experience with machine learning techniques and their practical applications in professional esports.

I. INTRODUCTION

Predicting the outcomes of professional esports matches has become a critical problem, with substantial practical consequences such as aiding teams in strategic decisions, helping investors place wagers, and providing game developers with critical insights into the game's meta. Machine learning has emerged as a viable method for forecasting esports match outcomes because it can use massive amounts of data to identify patterns and relationships that people find difficult to detect. In this project, we will use Python and its machine learning libraries to create a classification model that can accurately predict the winning team from a list of teams in a tournament, predict which team will win given a balance of 10 different players with 5 in each team, and judge players based on their average ACS, ADR, Econ, and other stats. By the end of this project, learners will have a good understanding of how to approach a machine learning problem and apply these techniques to real-world scenarios in the field of professional esports.

II. RELATED WORK

There are only a handful of works available that focus on valorant match related predictions. According to [1], the author has provided their work on Github. Even though we don’t have much information about the author, after analyzing their work, we see that it does an excellent job predicting different outcomes from valorant matches. They have used both Decision Tree Classifier and Decision Tree Regressor to predict different targets, such as player ACS and other stats.

On the other hand, [2] has shared their repository on Github, which also focuses on predicting valorant match outcomes, especially in the Berlin Tournament. The author has scraped relevant data from all over the web and applied it to their ML algorithms. K-Means Clustering Algorithm was used in this case which is an unsupervised learning algorithm. The work shows promising accuracy in the predictions.

III. MODEL APPROACHES

In our experiment, we have used two of the most commonly used machine learning models, the Decision Tree Regressor and the Random Forest Classifier. The Decision Tree Regressor takes input data and produces a regression output. The architecture of this model is similar to that of a tree. The algorithm breaks down the dataset into smaller subsets, where each feature is reviewed and examined in order to make an informed decision based on the pattern made by the dataset. The Decision Tree Regressor is a powerful model for predicting numerical values. It can handle both categorical and continuous variables and is robust to outliers. It is also easy to interpret, making it a popular choice for data analysis.

The Random Forest Classifier works by creating a large number of decision trees and aggregating their predictions. Each decision tree is trained on a random subset of the input data, and a random subset of the features is selected at each split. This randomness helps reduce overfitting and make the model more robust. The Random Forest Classifier has several advantages. It is less prone to overfitting and can handle larger datasets with many features. It is also more accurate than a single Decision Tree, as the aggregation of multiple models helps reduce variance and improve prediction accuracy.

To conclude, the Random Forest Classifier has shown significant improvement in the performance of the first two experiments. By modifying the work of [1] and using Random Forest Classifier instead of Decision Tree Classifier, we have achieved better accuracy in our

predictions. Cleaning the data properly before feeding it to our model has also helped us improve the results significantly. However, for our third experiment, where we predict the outcome of matches played by defined teams, the Decision Tree Regressor shows better performance than the Random Forest Classifier. Because the data we provide here is smaller and more clustered than the original dataset. As Decision Tree Regressor is more prone to overfitting, it performs better on the small modified dataset.

IV. PERFORMANCE METRICS

We have used the accuracy metric to evaluate how well our models are performing.

Experiment	Model	Training Accuracy	Testing Accuracy
Predicting Winning Team	Random Forest Classifier	100%	93%
Predicting Top Players	Random Forest Classifier	100%	60%
Predicting Player Stat (Kills)	Decision Tree Regressor	100%	98%

Table I

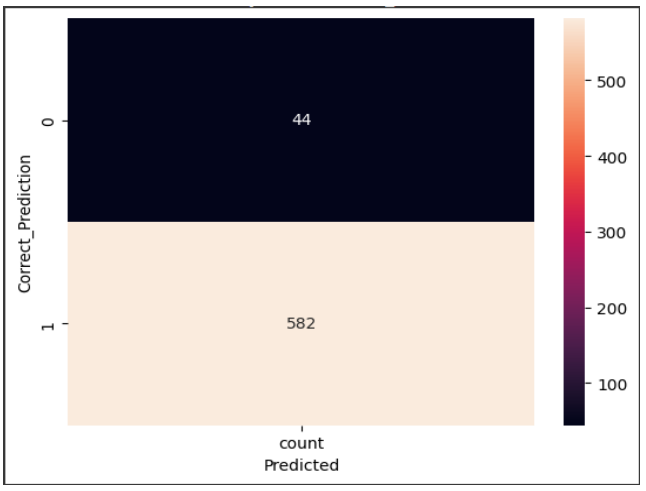

Fig-1: Confusion Matrix (Predicting Winning Team)

The confusion matrix shows a representation of the count of correct and wrong predictions.

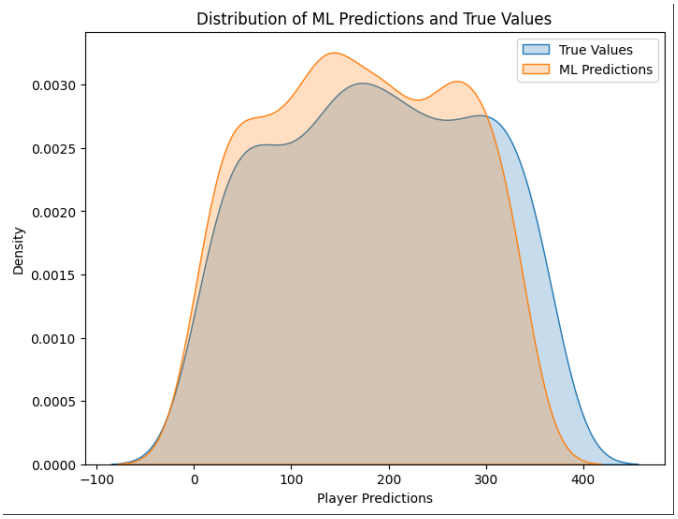

Fig-2: Density Plot (Predicting Top Players)

The density plot graph shows the distribution of True ACS and Predicted kills of players.

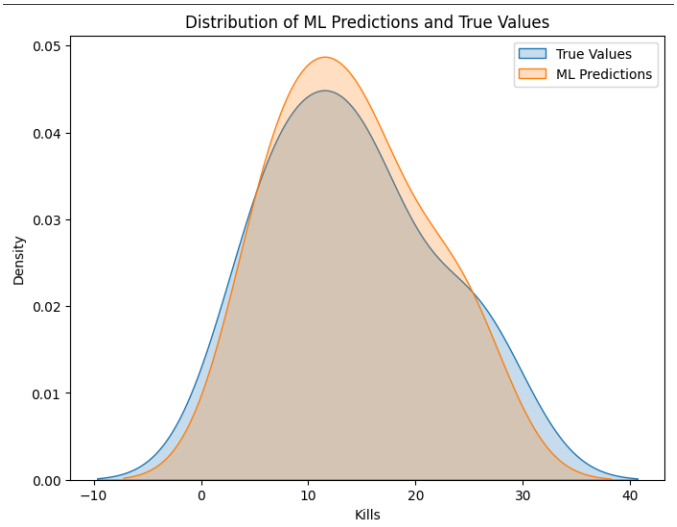

FIG-3: DENSITY PLOT (PREDICTING PLAYER STATS)

The density plot graph shows the distribution of True kills and Predicted kills of players.

Here we notice 100% accuracy on our training set. For predicting the winning team, we have achieved 93% accuracy, but for predicting the future top players based on their ACS, our model did not perform well, showing an accuracy of only 60%. For predicting individual stats of players, however, the accuracies can vary. When we predicted the kills of 5 random players, we achieved an outstanding accuracy of 98% on the training set. It might not be the case for other statistics such as ACS, ADR, Econ, and so on. The accuracy can vary depending on the data we have on individual players' stats.

V. EXPERIMENTAL ANALYSIS

A. DATASET

The dataset [3] for our experiment was obtained from Kaggle. The dataset is in sqlite format and contains 4 tables consisting of different data about the esports matches. The dataset contains a detailed history of matches played in the past two years. We have converted the sqlite tables into CSV files and merged all four CSVs into a single merged CSV dataset for usability.

B. DATASET PRE PROCESSING

In order to feed the dataset to our models, we first preprocessed the dataset. We have merged the necessary features that will be required to make proper predictions and dropped all the irrelevant features. Then we dropped all the NULL rows. Finally, we have scaled all the numerical values and one-hot-encoded the categorical values. After splitting the data into training and test sets, we trained the models to make predictions on the testing sets.

VI. RESULTS

For the part where the model is able to predict the winning team in a tournament, On the test dataset, the machine learning model constructed in this research performed well, with an accuracy of 93%. A Random Forest Classifier with 100 estimators was utilized as the model. The training accuracy was determined to be 1.0, and the testing accuracy was determined to be 0.93. The model was used to forecast the winning team in a tournament given a balance of ten distinct players, with five on each team, as well as to rate players based on their predicted ACS. The results demonstrated that the model could correctly estimate the winning team from a list of teams in the competition. However, the accuracy of the model was lower when predicting a sample of top players in the future based on their ACS. Further improvements could be made by tuning the hyperparameters of the model or by using similar machine learning models that adopt the Tree machine learning model architecture. Overall, this model showed significant promise and could be a good alternative for predicting the winning team in a tournament.

REFERENCES

[1] Analysis-of-T1-T3-VALORANT-tournaments https://github.com/hemmys/Analysis-of-T1-T3-VALORANT-tournaments

[2] Valorant-Berlin https://github.com/jasonren12/Valorant-Berlin

[3] Valorant Pro Matches Full Data [Data set]. Retrieved from https://www.kaggle.com/datasets/visualize25/valorant-pro-matches-full-data