PREDICTING VALORANT MATCH OUTCOME

Johnny Casares

Introduction

My project focuses on the development of a game match outcome prediction algorithm specifically designed for Valorant, a popular competitive video game. By evaluating over 330 Valorant matches obtained from Tracker Network (https://tracker.gg), my goal is to determine whether a team will emerge victorious or suffer defeat in a given instance. The application of this algorithm extends to various areas within the field of esports, including game development, game mechanics balancing, esports analytics, team management, and even esports betting.

The primary challenge I face in this project is the extraction of relevant data, as there is no readily available API to retrieve data specifically about Valorant matches. A similar predicament was encountered by researchers in the article titled "Hybrid Basketball Game Outcome Prediction Model by Integrating Data Mining Methods for the National Basketball Association." In their study, they were required to extract data from a website that lacks a formal API, which most likely led to the implementation of web scraping techniques.

Overcoming the data extraction challenge will involve employing web scraping methods to collect the necessary information from sources such as Tracker Network. By navigating these obstacles and utilizing appropriate methodologies, my project has the potential to contribute significantly to the fields of game development, game analytics, and esports management, providing valuable insights for both the gaming community and the esports industry.

Related Work

The sports market has experienced remarkable growth over the last decades, prompting an increased interest in sports analytics. A particularly appealing challenge within this domain is sports outcomes prediction, as it offers valuable insights for operations in the sports market.

The study titled "Integrating Data Mining Methods for Predicting NBA Game Outcomes: A Hybrid Model" gathered and employed data from all 1230 NBA games conducted throughout the 2018-2019 season. The games were divided into home and away team statistics, creating distinct datasets for each team. This approach resulted in a total of 2460 game scores being collected and utilized in the research. The data served as a valuable resource for analyzing and predicting NBA game outcomes in the study.

The study incorporated a set of 14 variables, including the final score of a team and 13 commonly used game statistics. To address potential issues arising from varying scales of these variables, min-max data normalization was applied. This normalization technique ensured that larger values of certain variables did not overshadow smaller values, allowing for the reduction of prediction errors.

Table 1. Variables description.

Variables	Definition	Description
$V_{1,t}$	2PA	2-Point Field Goal Attempts of a team in *t*-th game
$V_{2,t}$	2P%	2-Point Field Goal Percentage of a team in *t*-th game
$V_{3,t}$	3PA	3-Point Field Goal Attempts of a team in *t*-th game
$V_{4,t}$	3P%	3-Point Field Goal Percentage of a team in *t*-th game
$V_{5,t}$	FTA	Free Throw Attempts of a team in *t*-th game
$V_{6,t}$	FT%	Free Throw Percentage of a team in *t*-th game
$V_{7,t}$	ORB	Offensive Rebounds of a team in *t*-th game
$V_{8,t}$	DRB	Defensive Rebounds of a team in *t*-th game
$V_{9,t}$	AST	Assists of a team in *t*-th game
$V_{10,t}$	STL	Steals of a team in *t*-th game
$V_{11,t}$	BLK	Blocks of a team in *t*-th game
$V_{12,t}$	TOV	Turnovers of a team in *t*-th game
$V_{13,t}$	PF	Personal Fouls of a team in *t*-th game
Y_t	Score	Team Score of a team in *t*-th game

In this research they employed five data mining algorithms: MARS, ELM, XGBoost, KNN, and SGB.

In addition, the study evaluates the performance of the prediction models and determining the optimal game-lag by using the Mean Absolute Percentage Error (MAPE) as the indicator. According to the article, MAPE is widely used as a performance measure for forecasting and predicting methods. The study categorized models with MAPE < 10% as having "high accurate prediction ability," MAPE between 11% and 20% as "good prediction ability," MAPE between 21% and 50% as "reasonable prediction ability," and MAPE > 51% as "inaccurate prediction ability." In the article, they report the XGBoost algorithm to have the best performance. The numbers are shown in the tables below:

Table 3. Performance of the five single models under six game-lags.

Methods	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$
S-ELM	0.1020	0.0960	0.0915	0.0870	0.0931	0.0928
S-MARS	0.0910	0.0909	0.0897	0.0846	0.0917	0.0907
S-XGBoost	0.0919	0.0907	0.0911	**0.0842**	0.0927	0.0920
S-SGB	0.0910	0.0925	0.0913	0.0845	0.0923	0.0908
S-KNN	0.0992	0.1011	0.0947	0.0873	0.0934	0.0941

Table 4. Performance of the five two-stage models under six game-lags.

Methods	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$
T-ELM	0.1206	0.0924	0.0951	0.0863	0.0972	0.0902
T-MARS	0.0917	0.0911	0.0912	0.0845	0.0928	0.0900
T-XGBoost	0.0918	0.0930	0.0916	**0.0818**	0.0929	0.0920
T-SGB	0.0909	0.0918	0.0912	0.0829	0.0930	0.0908
T-KNN	0.0998	0.0984	0.0973	0.0872	0.0993	0.0970

Note: The bold indicates the best prediction performance.

Implementation

Libraries used

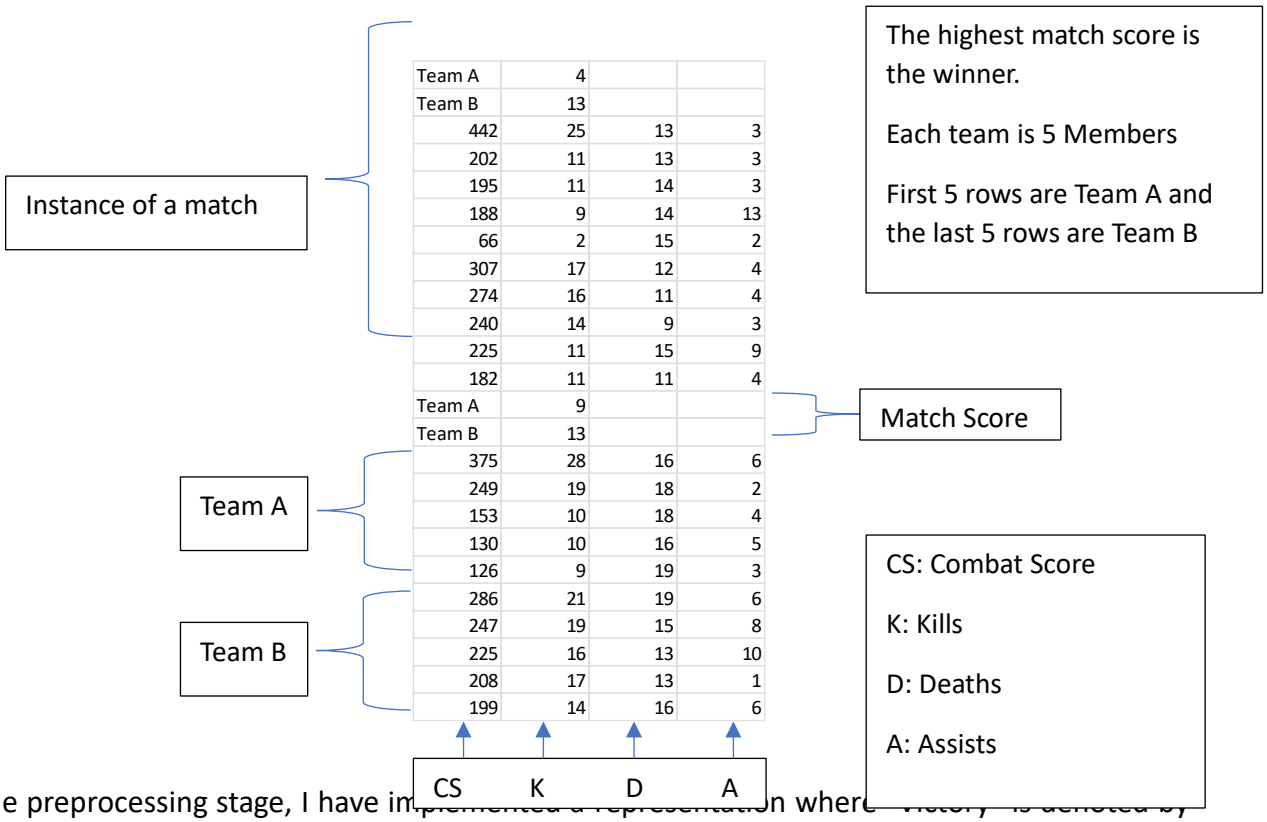
To run this project successfully, several Python libraries are required, including numpy, pandas, matplotlib.pyplot, warnings, sklearn.model_selection, sklearn.preprocessing, sklearn.neighbors, math, sklearn.metrics, IPython.display, and IPython.core.display. Each library plays a crucial role in different aspects of the project. Numpy is used for efficient numerical operations and array manipulations. Pandas is essential for data processing and handling data in tabular form. Matplotlib.pyplot is employed for data visualization purposes. The warnings library helps manage warning messages during the execution of the code. Sklearn.model_selection is utilized for splitting the data into training and testing sets. Sklearn.preprocessing.StandardScaler is used for data normalization. Sklearn.neighbors.KNeighborsClassifier is the core library for implementing the K-Nearest Neighbors algorithm. Math is used to find the number k-neighbors for the KNN algorithm. Sklearn.metrics provides necessary metrics such as confusion_matrix, f1_score, and accuracy_score for evaluating the performance of the model. Lastly, IPython.display and IPython.core.display are utilized to display images and HTML content in the Jupyter Notebook

environment. The proper installation and usage of these libraries are documented within the project code.

Data Preprocessing

For data extraction and preprocessing I used data from Tracker Network (https://tracker.gg). I made a Power Automate script that allows me to web scrape the site and store the information I want in a csv file.

The raw data looks like the following:



Team A	4		
Team B	13		
442	25	13	3
202	11	13	3
195	11	14	3
188	9	14	13
66	2	15	2
307	17	12	4
274	16	11	4
240	14	9	3
225	11	15	9
182	11	11	4
Team A	9		
Team B	13		
375	28	16	6
249	19	18	2
153	10	18	4
130	10	16	5
126	9	19	3
286	21	19	6
247	19	15	8
225	16	13	10
208	17	13	1
199	14	16	6

In the preprocessing stage, I have implemented a representation where "Victory" is denoted by 1, "Defeat" by 0, and a Draw is not part of the data, since in the Esports environments high stake matches, like the final, do not end in a draw. Furthermore, considering that each team consists of 5 members, I have chosen to calculate the average of their individual scores to create a

consolidated instance that is more convenient for handling and analysis. Additionally, I have assigned appropriate labels to each column to ensure clarity and coherence in the dataset.

The raw data presented before would look like the following:

CS	K	D	A	Result
218.6	11.6	13.8	4.8	0
245.6	13.8	11.6	4.8	1
206.6	15.2	17.4	4	0
233	17.4	15.2	6.2	1

Standard Scaler was used to normalize the data; this ensures that larger values in the dataset do not overshadow smaller values, ensuring fairness in the analysis. Since I have chosen to utilize the K-Nearest Neighbor (KNN) algorithm as my machine learning model, data normalization becomes crucial due to KNN's vulnerability to outliers and larger data values.

Pipeline

First, the data was loaded and labeled for easier manipulation. Next, I reviewed the data, making sure that no values were missing and that they all matched the numerical types. Since the data was carefully preprocessed, I moved on to evaluating the data on the KNN algorithm. According to the documentation by scikit learn, the principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point and predict the label from these. For the purpose of my model, the number of n-neighbors is 7, which was chosen by taking the squared root of the test data.

For the data splitting process, 20% of the dataset is allocated for testing purposes, while the remaining 80% will be used for training the model, ensuring a suitable balance between evaluation and learning.

To assess the performance of the model, I employed several metrics. These include the F1 score, accuracy score, and a confusion matrix, which visually represents the predicted results in comparison to the actual results. These metrics collectively provide a comprehensive evaluation of the model's effectiveness and can offer valuable insights into its performance.

The pipeline allowed me to achieve an accuracy score of around 93%, which means that the model was able to accurately predict most of the match outcomes based on the performance of the team.

Results

I am glad to share that the model yielded convincing and reliable results. The chart below, shows the results of all the performance measures I used to evaluate the model.

Confusion Matrix	[[36 0] [5 27]]
F1-Score	0.9152542372881356
Accuracy Score	0.9264705882352942
Run Time (Average)	< 4 seconds

In this case, the confusion matrix is as follows:

- True Positive (TP): 36 instances were correctly predicted as the positive class.
- False Positive (FP): 0 instances were incorrectly predicted as the positive class.

- False Negative (FN): 5 instances were incorrectly predicted as the negative class.
- True Negative (TN): 27 instances were correctly predicted as the negative class.

In summary, the confusion matrix indicates that the model correctly classified 36 instances as positive (correctly identified as "1" in the positive class) and 27 instances as negative (correctly identified as "0" in the negative class). However, the model misclassified 5 instances as negative when they actually belong to the positive class.

Overall, the model seems to perform well based on the confusion matrix, as it has a high number of true positives and true negatives and a low number of false positives and false negatives.

In addition to that, the F1-Score of 0.915 suggests that my model has a good balance between precision and recall, resulting in effective classification performance. And the accuracy score of 0.926 means that my model correctly predicts the outcome for approximately 92.6% of the data, which is a high accuracy rate.

Final Discussion

For future work I plan on using a larger data set and include characters from the game along with the player tag. These new features in the data would be for the purpose of providing specialized data for a particular player.

In addition to that, I would like to test the data in similar algorithms used in the study used as reference, such as XGBoost. I think testing the data in different algorithms could provide new insights and different performance measures that could help towards improving the model.

The future feature I am looking forward the most would be to get the Riot Games API approved to get real time data from the game and be able to provide insights on the match as it occurs.

References

Chen W-J, Jhou M-J, Lee T-S, Lu C-J. Hybrid Basketball Game Outcome Prediction Model by Integrating Data Mining Methods for the National Basketball Association. *Entropy*. 2021; 23(4):477. https://doi.org/10.3390/e23040477

KNN Algorithm In Machine Learning | KNN Algorithm Using Python | K Nearest Neighbor | Simplilearn. Simplilearn; 2018. https://www.youtube.com/watch?v=4HKqjENq9OU

1.6. nearest neighbors. scikit. (n.d.). https://scikit-learn.org/stable/modules/neighbors.html#neighbors