Analysis and Prediction of PUBG Player Behavior

Yuheng Zhu*
yzhu63@ncsu.edu
NC State University
Raleigh, North Carolina, USA

Mengzhe Wang*
mwang39@ncsu.edu
NC State University
Raleigh, North Carolina, USA

Jiayuan Huang*
jhuang52@ncsu.edu
NC State University
Raleigh, North Carolina, USA

Shuai Chen*
schen76@ncsu.edu
NC State University
Raleigh, North Carolina, USA

Keywords

Datasets, Data Visualization, K Nearest Neighbors, Neural Networks, Random Forest, Linear Regression

1 Introduction & Background

1.1 Problem Statement

PUBG stands for PlayerUnknown's Battlegrounds. In this game, up to 100 players parachute onto an island and scavenge for weapons and equipment to eliminate other players while avoiding being killed themselves. The play area gradually shrinks over time, forcing players into closer proximity and increasing the likelihood of encounters.

In PUBG, a player's main objective is to be the last surviving player or team on the island. To achieve this, players must scavenge for weapons, ammunition, and equipment scattered throughout the island, while also managing their health and resources.

Players can also choose to engage in combat with other players or avoid confrontation by hiding and staying out of sight. The game's play area gradually shrinks, which forces players to move towards a central location, increasing the likelihood of encounters and making the game more intense as it progresses.

Other gameplay features include looting buildings, driving vehicles, and using various weapons and equipment such as guns, grenades, and medical supplies. PUBG is a game that requires strategic planning, quick reflexes, and good teamwork to succeed.

Similar to other FPS games, cheaters in PUBG are players who use hacks or exploits to gain an unfair advantage over other players. These cheats can range from aimbots, which automatically aim and shoot at opponents, to wallhacks, which allow players to see through walls and other objects to locate enemies.

We introduce our problem as a compound task, which covers data visualization, classification and prediction, taken over "PUBG Match Deaths and Statistics" dataset on Kaggle. The dataset includes 2 major types of data: general information of each match and detailed kill information for some matches. With proven success of KNN, RT and SVM, we may apply these models on this PUBG dataset in order to discover the patterns among attributes and records. In addition to that, we also visualize some statistics data and map it on the map. In such a case, players may directly view the abstract game data on the proportional game map.

1.2 Related Work

In close relation to the work we have done, there are also several published articles that has used some learning methods. An article of PUBG features [1] has tried to use several approaches to analyse and distinguish features with most influence factor. Regression Tree, Linear Regression, and Support Vector Machine are trained and applied as key models for analysis. As a result, 'walkDistance', which stands for the distance of a player in one match, is the most important attributes. For this article, we realize that we cannot directly train the model using raw data, since the 'weights' of different attributes have different influential degree on the final output. For instance, it is not persuasive enough to give a conclusion that the more a player move, the more probability he/she can win the match: in common sense, only if a player survives longer, the player is able to move longer, not the other way around.

There is also an article [3] that predicts the placement ranking of players by analyzing their position and stats data. Network techniques such as Deep Learning, light GBM model, and MLP regression model are used for model training. Consequently, light GBM shows the best result, whereas DL and MLP does not return satisfied answer. From our perspective, this paper has considerable credibility. And we will use position and stats data in the later part of this article.

Researcher Madhurya has published a paper to analyze the general information of players, help them stand above all by giving suggestions. This approach use MAE (Mean Absolute Error) to estimate the performance of Linear Regression, Random Tree and Deep Neural Network. The final MAE shows that Deep Learning has the best performance.

*Both authors contributed equally to this research.

Conference'23, April 2023, Washington, DC, USA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/1234567.1234567

Inspired by this article, we will use Random Forest and Linear Regression to build model, in order to analyze and explain a player's behavior at a specific time in a match, and later give suggestion for their further decision.

Another approach to regression problems is the use of Gaussian processes (GPs), which offer an alternative method for putting a prior over functions instead of specifying a prior over model parameters, as in the Bayesian approach to neural networks (Paper's authors, Year). The main aim of the discussed paper is to provide a tutorial on regression with Gaussian processes, starting with Bayesian linear regression and showing how it can be viewed as a Gaussian process predictor based on priors over functions rather than performing computations in parameter-space [5] . This approach has been applied to higher-dimensional problems traditionally tackled with other techniques, such as neural networks and decision trees, and has demonstrated promising results . The paper also covers hierarchical modeling, setting the parameters that control the Gaussian process, covariance functions for neural network models, and the use of Gaussian processes in classification problems.

Random forest classification is a widely used machine learning method for developing prediction models, particularly in the context of expert and intelligent systems [4]. One of the main challenges in using random forest models is reducing the number of variables needed to obtain a prediction in order to improve efficiency and reduce the burden of data collection. Numerous variable selection methods exist for random forest classification, and many R packages are available to implement these methods. However, there is limited guidance on which method may be preferable for different types of datasets. The paper investigates the performance of various random forest variable selection methods using 311 classification datasets, evaluating their prediction error rates, number of variables, computation times, and area under the receiver operating curve. The study provides valuable insights into the best variable selection methods based on different types of datasets, offering recommendations for their use in expert and intelligent systems.

2 Method

2.1 Novel Aspects

We state our novelty by following aspects:

1. We have implemented a in-time visualization tool that uses dataset to generate a time-varying kill distribution on a map that during the entire game. This feature can provide effective game strategy guidance for players, and has significant benefits for improving gaming skills for both amateur players and professional e-sports teams.
2. We have implemented a feature that detects outlier players using a large amount of player game data to identify abnormal behaviors. This feature can effectively help various cheat detection tools obtain datasets of abnormal behavior to be used for training models that detect cheating players, providing technical support for combating game cheating and other behaviors that affect the game environment.
3. We have implemented the feature that predicts a player's final ranking based on their in-game performance. This feature can utilize a player's past performance data up to the current time point to forecast the impact of different actions on the final ranking. It provides players with suggestions and guidance on the next steps in the game, which is of significant importance in improving their gaming skills.

2.2 Approaches

During our pre-investigation, we studied several relevant works which are widely used in machine learning tasks, for classification and prediction problems. Based on the investigation, we decided suitable algorithms and approaches for our project as:

1. K-Nearest Neighbors (KNN)
   K-Nearest Neighbors is a supervised machine learning algorithm used for classification and regression. In KNN, a new instance is classified based on the class of its nearest neighbors in the feature space. The "K" in KNN refers to the number of neighbors used to classify the new instance. KNN is an ideal model to deal with high-dimensional, low noise and sparse dataset.
2. Linear Regression
   Linear regression is a statistical approach used to establish the linear relationship between two variables by fitting a linear equation to the observed data. The goal of linear regression is to find the best fitting line through the data that explains the relationship between the independent variable (or predictor variable) and the dependent variable (or response variable).
3. Random Forest
   Random Forest is a machine learning algorithm that belongs to the family of ensemble learning methods. The algorithm builds multiple decision trees on different subsets of the input data, and combines their predictions to obtain a final output. The term "random" in Random Forest refers to the fact that the algorithm randomly selects a subset of features for each tree and samples a subset of the data points for each feature, thus introducing randomness into the model and reducing overfitting.
4. Artificial neural networks (ANNs)
   ANNs are a computational model inspired by the structure and function of the biological neural networks in the human brain. ANNs consist of interconnected nodes, or artificial neurons, that process and transmit

information. The neurons are organized in layers, with each layer processing and transforming the information received from the previous layer.

2.3 Rationale

After briefly introduce the methods we selected for further experiment, we may discuss the reason of choosing these methods. As a notice, we apply KNN to find cheaters, and random forest to predict the game:

1. KNN: Our PUBG dataset contains more than 8 aspects of gaming data and those values are sparse, as well as there are only a few of abnormal behavior records. These characteristics make KNN very suitable to find those few outliers from each game.
2. Linear Regression: Finishing Placement Prediction can be understood as a fitting problem. In this case, Linear Regression Model makes the estimation procedure simple and, most importantly, these linear equations have an easy to understand interpretation on a modular level.
3. Random Forest: The Random Forest Model can directly process high-dimensional data without dimensionality reduction, and its training speed is fast. Considering that the number and dimension of the data set used in this experiment are very large, the random forest model is an appropriate choice. In addition, it can judge the importance of different features and their mutual influence, which helps players analyze game data through this model.
4. ANNs: One of the downsides of the Linear Regression Model is that it assumes that there's always a linear relationship between the dependent as well as independent variables. Considering the complexity of the experimental data set, the relationship between prediction results and features may not be limited to a linear relationship. In this case, ANNs may give us a satisfactory result.

3 Plan & Experiment

Here we explain our dataset, the hypotheses we will evaluate, and design of our experiment.

3.1 Datasets

In this project, we use "PUBG Match Deaths and Statistics" dataset [2] on Kaggle. This dataset provides two zips: aggregate and deaths:

- In deaths, the files record every death that occurred within the 720k matches. That is, each row documents an event where a player has died in the match.
- In aggregate, each match's meta information and player statistics are summarized (as provided by pubg). It includes various aggregate statistics such as player kills, damage, distance walked, etc as well as metadata on the match itself such as queue size, fpp/tpp, date, etc.

The uncompressed data is divided into 5 chunks of approximately 2 GB each. The total size is about 20 GB.

3.2 Hypothesis

1. Where is the area where real-time battles occur during the game?
2. In the match, which players are AFK and which players are likely to be cheating?
3. Prior to conducting experiments, we hypothesized that although players' various in-game data exhibit randomness, a portion of this data may likely be associated with players' final rankings. Consequently, we aimed to ascertain the existence and accuracy of this relationship.

3.3 Experiment Design

In order to answer our hypothesis, we need to perform following steps in the experiment to obtain explainable results. See Figure 1.

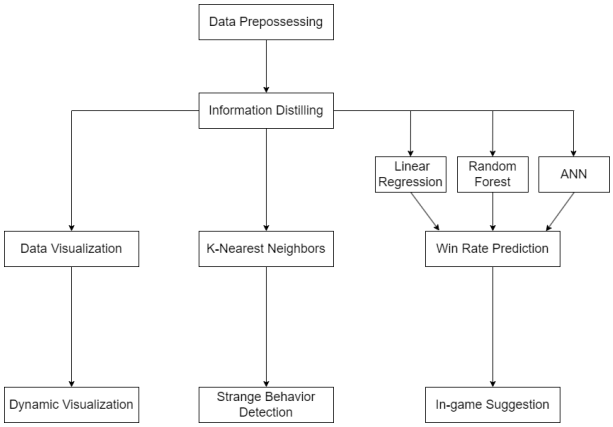


Figure 1. Experiment Design

3.3.1 Data Prepossessing. Data Prepossessing contains multiple steps including data extraction and data cleaning.

3.3.2 Information Distilling. As the dataset contains two different types of data, we need to distill different feature to supply the different needs of future missions.

3.3.3 Dynamic Visualization. Since the frequency and position of fights in game is abstract to players, we may apply data visualization on prepossessed data to achieve meaningful map images, such as maps with kill data and maps with resources data, just like the marauders' map in Harry Potter.

3.3.4 Strange Behavior Detection. Strange behavior players (aka "cheaters" or "seal clubbers") tend to outplay regular players in game which will be evident in their kill, damage

and other statistics. By applying outlier detection models we can distinguish them out of other players. In our experiment, we applied K-Nearest Neighbors model to unsupervisedly detect outliers from the dataset.

3.3.5 Finishing Placement Prediction.

1. To verify whether any data among the players' various in-game data has a strong correlation with their final rankings, we decided to train an initial random forest model using a small amount of data. The most significant challenge encountered during the design of this test experiment was determining the importance of features and the mutual influence between different features. Additionally, due to the high dimensionality and large volume of data, some models may require excessively long training time. To address this issue, we opted for the random forest model, which can handle high-dimensional data without the need for dimensionality reduction or feature selection. It can assess the importance of features and the mutual influence between different features while being relatively simple to implement and offering faster training speed.
2. To investigate the specific relationship between the player data retained in Experiment 1 and final rankings, as well as its accuracy, we employed random forest, linear regression, and ANN models to fit this relationship and compared their respective results. The most significant challenge encountered during the design of this test experiment was the ANN model design. Since neural networks lack a standardized format, the number of hidden layers and neurons can be freely determined. The more hidden layers and neurons, the higher the model's upper limit; however, the model's training time will also increase. To strike a balance, we designed an ANN model with three hidden layers, each containing 256 neurons. Considering the moderate dimensionality of the experimental data, our ANN model was capable of achieving satisfactory performance while limiting each training session to under thirty minutes.

Given the excessive number of sample sets and the limitations of our group's hardware devices, we have only selected data from the first file in the aggregate for model training. Before conducting the aforementioned experiments, we first partitioned the experimental data. We divided over 13 million data entries into training and testing sets, with the testing set comprising 33% of the data. Prior to Experiment 1, we randomly sampled 500,000 entries from the training set, designating 12% of the data as a validation set, which we then used to train an initial random forest model. In Experiment 2, we designated 12% of the training set as a validation set and used the remaining data to train the three models. Finally, we compared the performance of each model on the testing set. We tuned hyperparameters with CV. Both experiments employed mean absolute error (MAE) as the evaluation metric. In Experiment 2, we trained random forest, linear regression, and ANN models using the same training set and compared their performance as a control experiment.

3.3.6 In-game Suggestion. Suppose we have successfully finished all works above, we may have in-game suggestions, which stands for given visualized suggestion to players based on some statistics data. In our expectation, we should have the capability to display a 'dynamic map' with heatmap and arrows to advice the player's following behaviors, such as fight or escape.

4 Result

4.1 Dynamic Visualization

For Dynamic Visualization, we scheduled to visualize and map abstract in-game information on the map. For current project, we have generate heatmaps of kill-data, grouped by several killing types such as time of matches, weapon used and map.

The first step is data pre-processing, we firstly drop those meaningless data, such as accidental suicide or unstable network connection. Secondly, we filter out death caused by poison circle by estimating the last position of the playe. After processing data, we write a function to group and visualize death records by start time and end time in the game, weapon and maps. To be specific, the death data will be mapped on the proportional position via heatmap, on the corresponding map. In this process, Numpy, Pandas and MatplotLib are the key libraries we used. Consequently, we are able to have data visualization as Figure 2 and Figure 3:

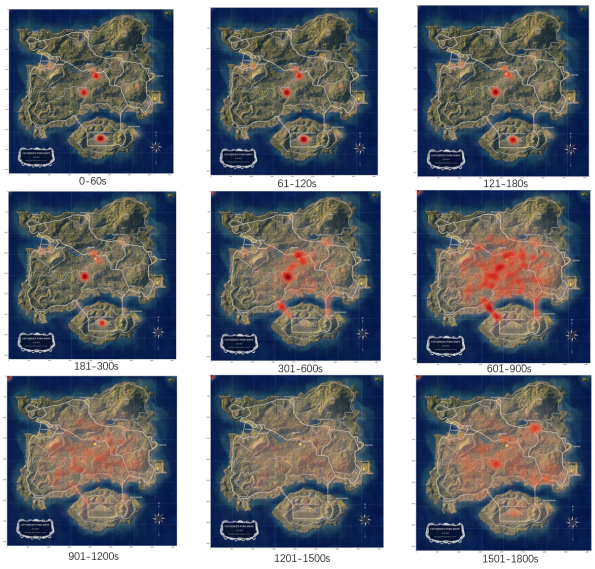


Figure 2. Heatmap of Kills In Map Erangel

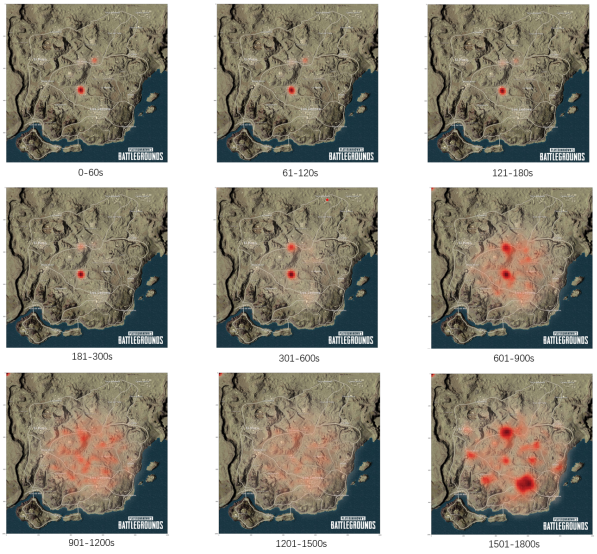


Figure 3. Heatmap of Kills In Map Miramar

According to the figure of map Erangel, we may notice that the major groups of players have stronger willing to land School, Pochinki, and Sosnovka Military Base. From the game start to the first 5 minutes, these 3 areas are with the most frequent kills. After that, survived players are likely to move to other places, such as Ruins, Shelter, and Prison. In fact, in the time period 5-30 minutes, the conflicts emerge on almost every corner in the map. In the last image, we may notice that the final circle is not 100% random: it is more likely to be around at a resource location in the middle of the map. In such a case, it is better for a player to arrive one of possible final center earlier in the match.

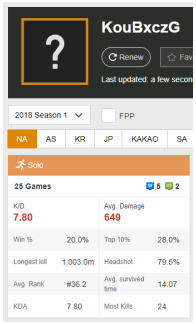
As for map Miramar, the major in-game method is similar with one of map Erangel, the most difference between maps is the position of resource location and density of death. Obviously, the kills records are more dense on this map. The possible reason may be the less equipment on the map, thus causing more frequent conflicts among players. Specifically analyzing, two most popular land-on places are El Pozo and San Martin. And different from Erangel, the final cycle of Miramar is almost with unlimited position: almost all the resource locations can be the center of the final cycle. In such a case, it is not helpful to arrive a center and defend. It is more helpful to simply killing other players to win the match than hide on bush in this map.

4.2 Strange Behavior Detection

For Strange Behavior Detection, we first considered making outlier detection using K-Nearest Neighbors Outlier Detection on solo game mode data, as it is cheaters' favorite game mode. By applying KNN on a partial training set contains 18,962 player records, we distinguished 17 outlier records.Their partial records are shown in Figure 4(a). Taking the first player in the list, "KouBxzczG", who also has the highest kill number, into account, we have a search on his historical game records. As shown in Figure 4(b), this player has 80% of head-shot rate and can take down other players from 1,000 meters away. This obviously is not from a manual operation. As this player has no further gaming record after 2018, we suppose this account has already been banned shortly after its cheating behavior. Apparently, our approach can easily distinguish such cheaters from regular players.

player_dmg	player_kills	player_name	player_survive_time	team_id	team_placement
2343	24	KouBxczG	1414.033	100048	1
1332	7	zhihuidehouge	1753.127	100077	4
1565	17	Wcccccc1	1879.379	100030	1
1212	5	NA-655-607-504	1656.966	100086	1
1783	18	SytherXII	1506.521	100042	6
1197	9	RNGxiyi	1490.115	100005	13
1657	17	BBBBBB888888	1876.975	100022	1
1405	16	xiuquantaqindie	1758.218	100093	1
1249	11	QQWSAD	1949.991	100098	1

(a) Cheaters' partial records



(b) Game history of @KouBxczG

Figure 4. Investigation On One Cheater

The 2D visualization of outlier detection is shown in Figure 5. X-axis is player kill numbers and Y-axis is player damage values. As most of the regular players represented by blue points have reasonable game records, red points represent outliers which have both significant higher damage or kill.

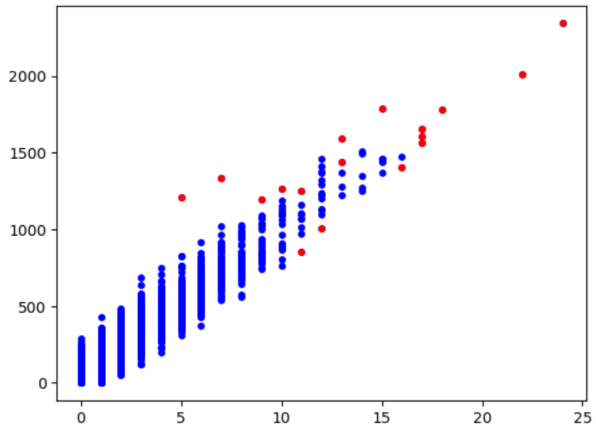


Figure 5. Outlier Points

4.3 Finishing Placement Prediction

4.3.1 Feature Engineering. Players Joined is a very valuable feature for our model. To get stronger predictions on individual players, we calculate how many people are in a match firstly. See Figure 6.

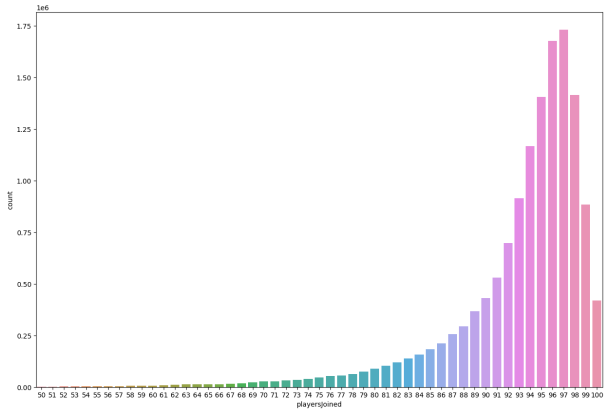


Figure 6. Players Joind

There are very few matches with fewer than 50 players that are not displayed here. As we can see most of the matches have nearly 100 players while others have less. Normally, the more players in the game, the more likely your team will encounter other teams, which means more opportunities for players to deal damage, kills, and rescue teammates, but also means more chances to be knocked down or be out the game. So it is reasonable to normalize some features like player_kills, player_dmg, player_assists, player_dbno and player_survive_time. Additionally, we add a new attribute named winPlacePerc as the target of prediction. This is a percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match. Finally, we drop some features including match_id, player_name, team_id, match_mode and team_placement, because they probably won't be useful for our Machine Learning algorithm. The changes in the data set before and after feature engineering are shown in the Table 1.

4.3.2 Experiment 1. Through our initial random forest model, we got the importance of each feature for predicting the target. See Figure 7.

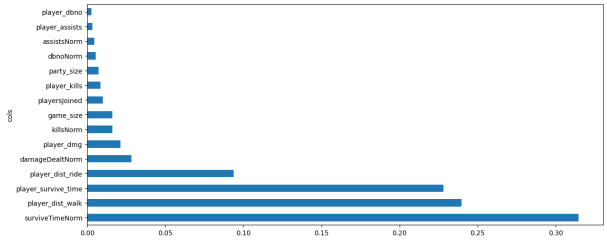


Figure 7. Feature Importance Graph

We removed those features whose importance is less than 0.005, and got 12 significant features and the interaction between them. See Figure 8 and Figure 9.

	Origin Dataset	New Dataset
Features	date game_size match_id match_mode party_size player_assists player_dbno player_dist_ride player_dist_walk player_dmg player_kills player_name player_survive_time team_id team_placement	game_size party_size player_assists player_dbno player_dist_ride player_dist_walk player_dmg player_kills player_survive_time playersJoined killsNorm damageDealtNorm assistsNorm dbnoNorm surviveTimeNorm winPlacePerc

Table 1. Changes in the dataset before and after feature engineering.

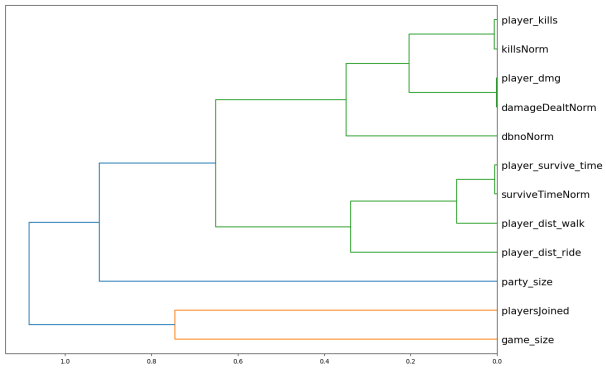


Figure 8. Correlated Features Dendrogram

Experiment 1 successfully judged the importance of each feature and the correlation between them. Among the 12 features, the four features player_survive_time, player_dist_walk, surviveTimeNorm and player_dist_ride have a greater impact on the prediction results. As for the correlation of different features, we can see that those normalized features have a strong correlation with their original features, which is in line with common sense. In addition, player_survive_time has a strong correlation with player_dist_walk and player_dist_ride, which means that the longer the player survives, the farther they walk and drive. This is in line with the rules of the game. Players need to constantly shift positions as the safe zone changes. dbnoNorm has a strong correlation with player_kills, killsNorm, player_dmg, damageDealtNorm, which means that the more times a player gets knocked down, the more chances he has to deal more

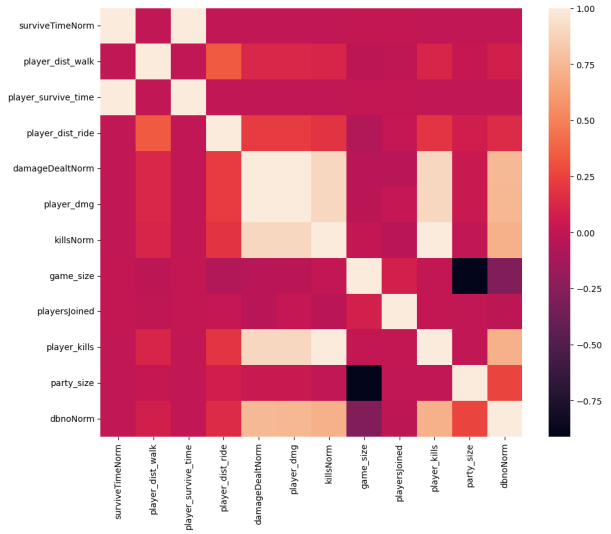


Figure 9. Correlation Heatmap

kills and damage before the game ends. This is something we didn't expect, and we speculate that it's because the player gets knocked down multiple times, meaning he's fighting more often and has more chances to deal damage and kills.

4.3.3 Experiment 2. We successfully trained three prediction models and compared their results. See Table 2.

	MAE-Train	MAE-Val	MAE-Test
Linear Regression	0.1902	0.1903	0.1902
Random Forest	0.0539	0.0883	0.0880
ANNs	0.0848	0.0864	0.0861

Table 2. MAE of Models

Random forests achieve better results on the training set, while ANNs perform better on the validation and test sets. We tend to choose the ANNs model, but that doesn't mean it's a definitive answer. For random forest models, it still has many hyperparameters need to be tuned. If we keep increasing the number of estimators, it may get better results than ANNs. Similarly, for ANN, we only designed a network with three hidden layers in this experiment. It can be seen from the change of the loss function that our current ANNs model fits the training data well after the third epoch. See Appendix.

This means we may have reached the upper line of the current model. But if we continue to increase the number of hidden layers and neurons, we may be able to get a more accurate ANNs model. Due to our hardware limitations, we were unable to explore the above questions. But if conditions permit, we will conduct more detailed experiments on the parameter settings of these two models.

5 Conclusion

In conclusion, we have gone though a complete steps of a data analysis project. We learnt how to do data-preprocessing, data visualization, related work searching and reading, as well as some algorithms in practical such as KNN, linear algebra, and random forest. In fact, we also tried other approaches such as SVM and PCA. Unfortunately, they performed badly on this project and therefore we do not mention them in above pages.

As for finished work, this project has successfully demonstrated the application of data visualization, simple outlier detection, and simple prediction techniques on a PUBG game dataset. Through data visualization, we were able to gain valuable insights into the relationships and patterns within the data, which facilitated a better understanding of the underlying game dynamics. These visualizations also aided in identifying trends and areas of interest that warranted further investigation.

The implementation of cheater-like outlier detection techniques enabled us to identify players with anomaly behaviors. be. By applying clustering approaches such as KNN, we ensured that our analyses is capable to group potential cheater in the match.

Finally, we employed simple prediction methods to generate forecasts for various game-related variables, providing valuable insights into the possible outcomes of future matches. The prediction models developed in this project demonstrated satisfactory performance, offering a solid foundation for further improvement and refinement. These models have the potential to be used as a basis for decision-making support tools or as a feature in applications targeted at PUBG players and enthusiasts.

It is also worth mentioning the group work of this project is also challenging since each of the member is responsible for different part of this project. As a result, there are always conflicts and misunderstanding between us. After several meetings, we gradually modify our experiment processes normally. Thankfully, the finished work is acceptable.

There are still some work we want to pursue, however not yet able to do in this project. In-game text suggestion is one of them, we have tried to write some scripts to implement this idea, however, it is not even able to be tested since it is detected as a illegal plug-in application. After some attempts, we decided to give up this interesting idea and work on data analysis. If it is possible, we may make use of other free time to design an interesting game-suggestion application on other digital games.

6 Meeting attendance

	Yuheng Zhu	Mengzhe Wang	Jiayuan Huang	Shuai Chen
Meeting1	Y	Y	Y	Y
Meeting2	Y	Y	Y	Y
Meeting3	Y	Y	Y	Y
Meeting4	Y	Y	Y	Y

Table 3. Meeting Attendance

References

[1] NF Ghazali, N Sanat, and MA As' ari. 2021. Esports Analytics on PlayerUnknown's Battlegrounds Player Placement Prediction using Machine Learning. *International Journal of Human and Technology Interaction (IJHaTI)* 5, 1 (2021), 17–28.

[2] Kaggle. 2018. Pubg match deaths and Statistics. https://www.kaggle.com/datasets/skihikingkevin/pubg-match-deaths

[3] Brij Rokad, Tushar Karumudi, Omkar Acharya, and Akshay Jagtap. 2019. Survival of the Fittest in PlayerUnknown BattleGround. *arXiv preprint arXiv:1905.06052* (2019).

[4] Jaime Lynn Speiser, Michael E. Miller, Janet Tooze, and Edward Ip. 2019. A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications* 134 (2019), 93–101. https://doi.org/10.1016/j.eswa.2019.05.028

[5] C. K. I. Williams. 1998. *Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond.* Springer Netherlands, Dordrecht, 599–621. https://doi.org/10.1007/978-94-011-5014-9_23

Appendix

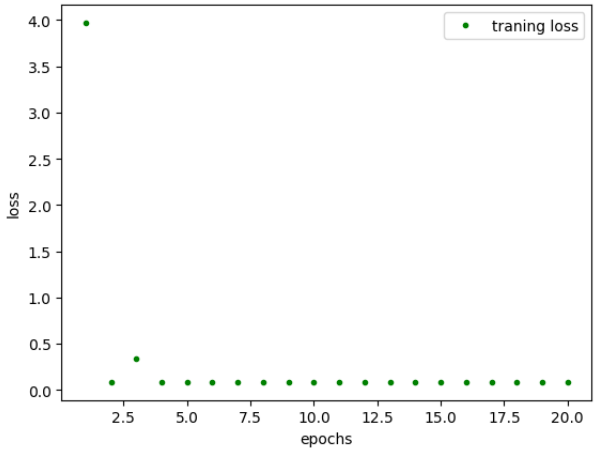


Figure 10. Training Loss

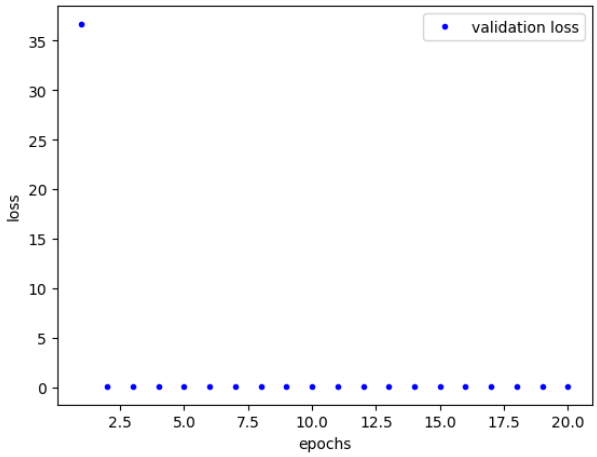


Figure 11. Validation Loss