# Introduction

## PlayerUnknown’s Battlegrounds

PlayerUnknown’s Battlegrounds (PUBG) is one of the top online multiplayer games in widely popular nowadays genre of battle royale shooters. This genre usually implies that players are dropped on the map with no weapons and have to find them and survive, the last man standing is the winner. Variations of battle royale in PUBG include team competition where the last surviving team is the winner, and this particular game mode will be analyzed in our research.

PUBG is currently the 5th best sold game of all time reaching more than 50 million downloads since its release back in 2017. The game is present on most of relevant gaming platforms, such as PC, PS4 and Xbox One. Later, the PUBG Mobile was launched bringing PUBG experience to mobile platforms such as iOS and Android. Unlike its main competitor Fortnite where players from all platforms (including mobile ones) play together in one version of the game, PUBG and PUBG Mobile are separate games with different userbases. That’s the reason why our research only covers PUBG and not PUBG Mobile.

# The problem

We need to build a system to collect the PUBG API data, clean the data, analyze behavioral data of team players and identify the different strategies to out survive the other teams in addition to find out the most attractive places to collect resources and weapons.

Our idea was to discover how collaboration affects team performance. We chose eSports as our field of study, but we want to transfer our findings to how teams function in the business environment, as we tend to think that human behaviors follow the same general outline in any situations.

Features of battle royale as a game mode leads to great variation in terms of strategies selected by teams. Players can stick together to multiply their chances to survive or spread out to collect more resources, they can be careful and silent through most of the game or they can engage in warfare from the very beginning, they can use vehicles to quickly travel to the new safe zone or go by foot slowly but unnoticed. As you can see, the possibilities are endless. Thus, besides our main goal of assessing role of collaboration in teams’ performance, we can also consider and analyze role of risk-taking behavior and player experience in their relation to winning. We want to find out which strategies are more efficient and lead to victory more often.

# Our goals

As mentioned previously we are interested in three aspects of the game: how teams collaborate, how they risk and how experience is related to winning. Therefore, we have three goals, the first one being the most important:

* Understanding the role of collaboration in performance (from strong to loose collaboration)
* Understanding the role of risk-taking behaviour in performance (exploration strategies vs exploitation strategies)
* Understanding the role of experience in performance (amateurs vs experienced players)

## Added value to society

When it comes to thinking of value that our work will bring to society, we focus on two main things. The first one is gaining understanding on human behavior related to collaboration and teamwork. There is not a lot of research on how collaboration and different strategies affect team performance and if we manage to find something interesting in PUBG data, we think it can be valuable knowledge that can be generalized to any field. The second thing we want to do with our research is to prove the potential of eSports data as experimental data that can help analyze and understand behavior of teams in different contexts. We believe that data generated by gamers have a huge potential in understanding human behavior that is yet to be realized. Millions of people around the world are put in various challenges by game developers and find different ways to overcome them. If all this data was thoroughly analyzed, the humanity can get a better understanding on how people approach challenges and react to them.

# Our work

In order to reach all of the goals we set, we needed to build a system to collect the PUBG API data, clean the data, analyze behavioral data of team players and identify the different strategies to out survive the other teams. This section will describe in detail what challenges we faced on each stage of our project and how we overcame them.

## API data collection

The first stage of our project is API data collection. The goal of this stage is to understand and to collect the necessary raw data from the different API endpoints in order to build our datasets.

PUBG API can be used to download various types of game data. Two types that we decided to collect were match telemetry data and player data. The first one, match telemetry contains events that happened inside the match. From this we collect the following events:

* + Location of players at different points in time. Location is the main variable we use to assess collaboration within the team. This variable is defined as the location of every player every 10 seconds. This means that, for each player in the match, we will have their location starting at second n (between 1 and 10) and, from then on, we have their location every 10 seconds (n + 10, n + 20...).
  + Parachute landing location. Landing location of each player is another important piece of information. As most players tend to land in locations where they can find initial weapons and other resources, we can infer that they consider these locations valuable and, therefore, these areas are risky. We need this data to create heatmaps that are used to assess riskiness.
  + Match information (ranking, type of match, map, players in the match). This part of data is needed for technical purposes, as we need to identify how player ranked throughout the match, whether they play in duo or squad mode (we leave out solo matches), which map they are playing on, which players were in that match.

The second type of data we collect is player data. We use it to get information about the experience of the players to later analyze how it affects performance. A player’s experience is measured with rank points for the season. Every season, each player in the game increase their rank points by getting good ranks, killing other players, and other actions that characterize the experience of the player. For that reason, we decided to take this piece of data as a proxy for the experience of the player.

At this stage of our project we experienced two major challenges. The first one was unfamiliarity with API processes and communication. PUBG API is the first one that we have ever worked with, so we haven’t known where to start. Thankfully, PUBG corporation provide very thorough API documentation on their development portal that along with help of our mentor made it possible to understand everything we needed in the matter of weeks.

The other challenge that we faced was rate limits that slow player data collection. The default rate for players data requests (that we need to get the rank points) presents a limit of 10 requests per minute which made the process of collection really slow and lead to possibility of not including rank points into the analysis as we would not have enough data. In order to overcome this, we filed a request to increase this limit for us and provided brief description of our project. After few weeks of silence, we received an email that stated that the limit will be increased up to 50rmpand wished us good luck with our project.

The result of this stage is the creation of fully automated API processes that collect required data from both the Match endpoint (telemetry data) and the Players endpoint (player data). Raw data is stored in individual JSON files. In the case of Telemetry data, we stored it in individual files for each match that would be later cleaned. For the players, we stored the rank points in a single file that would be later transformed in order to be suitable to merge with the main telemetry dataframe. Totally we have collected data on 37,919 squad-mode matches in which participated around 3 million players. The total volume of collected data is around 130GB.

# Data preparation

After collection of data we needed to transform it in order to reach all the goals we set - understand role of collaboration, risk taking and experience in relation to team performance. We split the work into three big parts, each corresponding to one goal: location data for collaboration, landing data for riskiness and rank points for experience. Finally, we needed to prepare ranking data as it will be our target variable.

### Collaboration variables

The goal of this set of variables is understanding the role of collaboration in performance, something we defined as one of our primary goals for the project. Due to the nature of the game and the data that we had collected, we decided to define collaboration as the dynamics of the team in terms of distances among them and their tendency to stay clustered. For that reason, we defined 3 types of variables:

1. Pair distances - these are the distance between any two players inside a same team at every point in time defined in spans of 10 seconds. Therefore, for a team of 4 people we had 6 data points at every point in time. Whenever a player died or if a player was not playing (e.g. teams of 3 or 2 people), we imputed a Null value. This is done in order to be able avoid data leakage by ignoring the death of players for later calculations when aggregating data to a team level.
2. Distances to team centroid - these are the distances of every player to the Euclidean centroid of the team at the point in time. In order to get this, we first found the centroid by getting the mean of the X and Y coordinates of each player at each point in time and then calculated the Euclidean distance between every player and that centroid. This means that, at every point in time in a team of 4 players, we would have 4 data points.
3. Clusters - these variables are defined as the number of people in the team staying in a same cluster at each point in time. In order to make these calculations, we used DBScan algorithm, where we inputted a minimum density of 1, so that a single player playing far from the rest could be considered a cluster, and 100m epsilon. This threshold distance was inferred by looking at the distribution of the distances among the players for several matches, where we observed that the mean was around 100m. Moreover, we estimated that, in the real world, this would be a considerable number in order to say when two people are playing together or not, considering that players can chat and know through a mini map where their teammates are located.

After defining these 3 variables, we decided to go one step further and define a fourth, categorical variable. This one would be used to label the collaboration of the players. For that, we used pair distances, and tagged the strategy with a set of 6 letters, each one corresponding to one pair distance. The possible letters were *c (close)* and *f (far)*, and we used the same 100m threshold to set this. Therefore, a team would have a defined strategy for every point in time. For example, one possible strategy would be *cccffc,* meaning that Player 1 is close to players 2, 3 and 4, Player 2 is far from players 3 and 4, and Player 3 is close to Player 4. We later used this variable to assess the agility of the teams, something we will explain later.

In order to properly define these variables, we had to do some previous preparation of the location data. The problem that we encountered was that the locations of the players were not recorded at the same points in time. For example, you could have a player’s location starting at second 1, while for another one in the same team it would start at second 4. Therefore, if we used these location data directly to calculate distances, we would be getting flawed conclusions. In order to fix this, we decided to interpolate the location in order to harmonize time. Therefore, we decided to infer the location of the players starting from second 1, and to get it for every 10 seconds (e.g. location at time 1, time 11, time 21, etc. for all players). The way we did this interpolation was finding the previous and following location of the player compared to the time we were interested in, calculate the speed of movement for that 10 seconds span in meters per second, and then infer where the player was at the time, we wanted by summing the speed times the time difference to the previous location. To illustrate: when interpolating location at time 21 for Player X, we see that we have their location at time 19 and 29. Then, we calculate the speed of movement for that time span. Imagine that we get that he moved at (3,4) meters speed (3 meters coordinate X speed and 4 meters coordinate Y). Therefore, if Player X was at location (i,j) at time 19, their location at time 21 would be (i+3\*2, j+4\*2). This way, we interpolated location for all players at all points in time in 10-second spans from second 1 till the end of the match, and these are the locations we use to calculate the variables.

In the end, what we had was a set of functions defined to do all these calculations automatically for each match telemetry data file. This function was computationally intensive, mainly due to the interpolation of distances and DBScan, and it took around 4 minutes to transform one match.

### Risk taking

The next stage is to prepare risk taking data. Our goal at this stage is to create two function one of which needs to scrub through match telemetry data, get player landing location and store it at previously created heatmap dataframes, and the other needs to assess risk the players is taking at each point in time by being in specific area (our initial idea was to measure risk only on the landing but as we realized that areas where a lot of players land have the most weapons and therefore are the most risky, we decided to measure risk players are taking throughout all match).

The first challenge during this stage was to create the heatmaps. PUBG offers a variety of differently sized maps for their players. Nowadays this list consists of four maps: Miramar (8x8 km in size and 100 players on the map), Erangel (8x8 km size and 100 players on the map), Sanhok (4x4 km in size and 100 players on the map) and Karakin (2x2 km in size and 64 players). The distance in PUBG is measured in centimeters, so coordinates for maps can be in range of either 0 - 816,000 cms, 0 - 408,000 cms, or 0 - 204,000 cms on both longitude and latitude axis. We have decided that the optimal size of chunk we use for measuring riskiness would be 10 by 10 meters.

First thing that we did was the creation of empty data frames (with zeros in every cell) representing every 10 by 10 ms chunk for each map. For Miramar and Erangel its 816 by 816 cells, for Sanhok - 408 by 408 cells and for Karakin – 204 by 204 cells.

Then we needed to create a function that will input raw match telemetry data and output landing location for each player by adding one to corresponding cell on the heatmap. In order to do it the function took only two events from match telemetry. The first one, LogMatchStart, is needed to extract the mapname, so the function will work with correct heamap. The second one, LogParachuteLanding, is needed to extract player’s landing location.

When the function was ready, we ran it on the data and created visualizations by putting heatmap on top of the actual data in order to make sure that it works correctly. Here are the results for Karakin map.

As you can see, the areas that marked red on the heatmap correspond to location of buildings on the map, thus, proving that the function works properly. Then we checked whether risky areas tend to differ if we add more data, but the results stayed the same. We decided that 500 matches for each map would be reasonable number to create the heatmap. The heatmap data frames for each map were stored as independent CSV files for further use as a reference.

The second challenge was to create a function that will use cleaned location data as an input and will output risk level for each row (that represents a point of time). To do it a part of code that was used to clean location data was repurposed. It was made to make sure that rows in the output (containing risk level) are in the same order like in the cleaned location data (clean data did not have X and Y variables by this time).

Finally, we ran this code on all data that we had and merged clean location data with risk levels that player takes at each point of time.

### Player experience

In order to see the clear effect of collaboration and risk taking on the team performance, we included rank points of the season for each player to control for the variations in experience among the teams. To collect the data on the rank points for player we had two metrics stored in the telemetry and season stats endpoint. One metric showed lifetime rank points of the player, and the other showed the rank points of the most recent season. However, players might quit playing the game for months or years suggesting that once they return their rank points would not be illustrative of their skill level. For our research purposes, it is crucial to account for how skillful the player is in this specific match, not how good he/she was two years ago. Henceforth, we have decided to collect the season stats rank points, which showcase the most recent level of skills for a player.

One of the obstacles in this stage was that the access to season rankpoints for players was located in the Season Stats API endpoint. Due to the vast amount of data we had to collect, more than 5 weeks would be required, considering the request limit of API calls to the aforementioned endpoint. Through contacting PUBG API Support team and filling in an application, our team obtained an increase of a request limit from 10 players in a minute to 50. Moreover, as our data collection coinsided with the start of a new season,hence, some part of the rankpoints data was impossible to obtain as it was unavailable at the moment. To differentiate between console and PC matches we had to build a function to attribute for separate endpoints. The final output was rank points for each player in the matches collected, which were extracted and summed, and later merged with the main dataframe.

### Ranking information

The target variable as well as the indicator of success in our analysis is the ranking of a team. Ranking will be the dependent variable in both our regression and classification models. Data on ranking was collected through a match telemetry API endpoint. One of the complications at this stage of data collection was that ranking data varied across different endpoints. The source of such disproportion stemmed from the fact that players receive individual rankings based on their performance along with a team ranking. In the teams of four, players who are more successful than their fellow teammates receive higher rankings. Our team filtered those rankings to arrive to homogenous ranking data, by extracting the highest ranking for a team. Considering that the unit of analysis for our research was the team, we took team ranking as the target variable.

# Data Aggregation

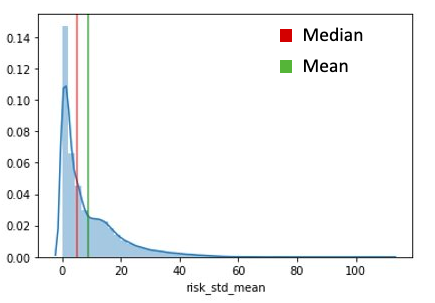
Our initial data was more suitable for a time series analysis, consisting of 10 second timeframes for each team and player. Moreover, the data included rows referring to dead teams. To transform and aggregate all the data we had, our team built an aggregation function, which consisted of seven steps.

1. To extract information from of distance data, we ordered the values, allowing us to illustrate the range of distances for each team.
2. Through analyzing distances between players, we could identify strategies they predominantly followed during the match. The strategies were coded “c” for close and “f” for far. Noting that, “close” collaboration is defined by 100-meter distance between players. To arrive to a variable that illustrates the team dynamics, our team feature engineered the “agility”. Agility shows the percentage of time the team is dynamic, meaning not following one strategy but pivoting from it. However, it is important to note that pivoting does not necessarily imply the ability to adapt to external changes, but simply change of an intra-team strategy.
3. Naturally, our initial data had information about players in the team dying at some point of the game. To avoid data leakage, we transformed the clusters information. Initially, it was referring to the absolute number of players that were playing in that team. Therefore, if only 2 players were playing, the maximum number that this variable would show would be 2. For that reason, we transformed cluster variables to represent the proportion of alive players in that cluster. With this, we avoided the model learning that some team had less players playing.
4. For the purposes of having a clean dataset, we dropped the rows referring to dead teams.
5. During our research we experimented by following an intuitive approach in transforming some variables. Logarithmic form for the distance variable, follows the logic that variation of distance on closer distances matters more than variation in further distances. For example, if the players are 100 meters away from each other or 400 matters more than 1000 meters and 1300 meters between players. We also evaluated models both with logarithmic distances and without the transformation, and the former proved to result in better models.
6. The rationale behind normalizing distances lies in the nature of the game. The playground gets smaller as the game goes on, restricting the movement of the players and incentivizing them to confront other teams to laying low and hiding forever is not an option. As this happens, intra-team distances get smaller as well, posing a risk of data leakage as it directly signals that some teams are further in the game than others. Hence, we normalized the distances to avoid the leakage. As the nature of our research is experimental and we adopted iterative approach, we checked for both normalized and standard distances, arriving to the conclusion that normalization improved our models significantly.
7. Finally, the last step of the aggregation function included extracting the means and standard deviations of the variables to attribute for loss of some data due to aggregation.

Our final output from the Aggregate Function is a clean, ready-to-use dataframe, which is safeguarded from data leakage, has new variable: agility, as well as ordered and normalized distance variables.

### Risk Categories

To take our research further, our team feature engineered a risk category variable. Risk categories were divided into three groups: exploration, ambidexterity, exploitation. The first step of the feature engineering was exploring the distribution of risk variable. Orienting on the distribution of the data, most of the teams follow the exploration strategy, which implies low risk-tolerance. As a starting point the median and mean were located. The median was closer to the apex of the distribution, capturing a more intuitive number of teams, whereas the mean captured more than 75% of the teams, which was not a relevant segregation for our analysis.



Henceforth, we took the median as a threshold for separating exploration strategy from ambidexterity. The ambidexterity, defined as a strategy that combines elements from both exploration and exploitation, took up the proportion of teams located in between the median and one standard deviation away from it. Finally, the exploitation strategy was defined as all the remaining teams that had high values, respectively high-risk tolerance.

# Exploratory Data Analysis

After transforming our data and having it ready for the analysis, we conducted an initial Exploratory Analysis to understand our variables, look for high correlations and be able to have our data ready to input into Machine Learning models.

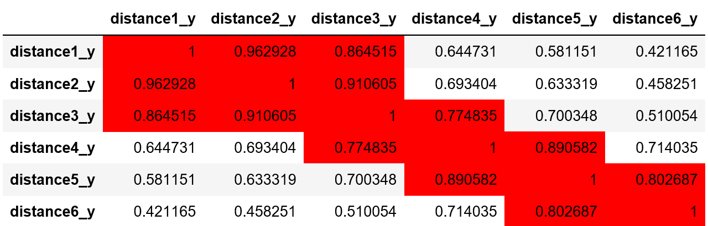
At this point, the amount of data that we had was 436,324 rows of data, each one corresponding to a team in a match. This meant having data on around 1.5 million players, 20k matches and for a time span of 30 days. The reason why we did not have the data for all the 37k matches we had initially collected is that we were not able to transform all the data to the final format due to computational limitations.

Moreover, we did not have all this data for the experience of the players, thus we were forced to shrink the size of our data even more. This way, we ended up having a data frame of 214,871 rows, around 650k players, 9k matches and for a time span of 18 days. This is the data we conducted Exploratory Analysis to.

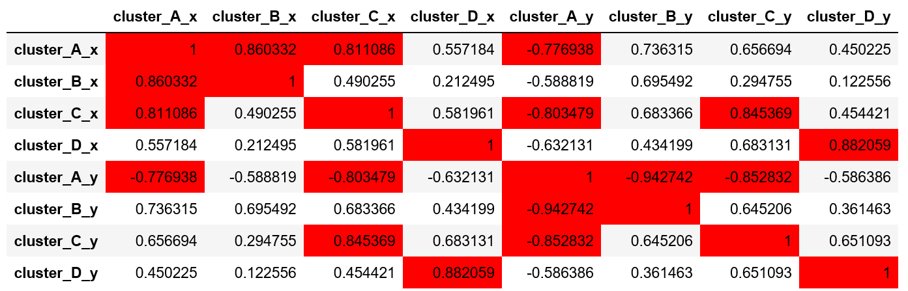
The first step was looking at missing values. At this point, we realized that 35% of the rows of our dataset had some null values. This was because 35% of the teams we had in our dataset was composed by 3 or less players. Therefore, and due to the vast amount of data that we were dealing with, we decided to drop all the rows with at least one null value. This way, we stayed with 4-player teams only. Moreover, since we care about team dynamics, this step was like controlling for team size, thus we concluded that it was a wise step. However, we also tried some models without doing this and replacing the values instead, and we saw how the resulting models were performing worse. This was the final confirmation that our approach was right. After this step, we were left with 129,977 rows and around 520k players from the same number of matches and time span.

After this, we realized that many columns had outliers, and this would make our models perform consistently worse. Therefore, we dropped all the rows with at least one outlier. To do this, we used the Z-score method, setting a threshold of 3 standard deviations. This way, any value that is more than 3 standard deviations away from the mean of the variable would be considered an outlier. After taking this step, we were left with our final amount of data. This was a total of 116,460 rows and around 470k players.

The next step was to look at correlation between variables. We decided to conduct this by groups of variables. First, we looked at distance-based variables (pair distances and centroid distances). Due to the sorting we had done before aggregating, we observed very high correlations between the variables, some bigger than 0.95. Since this would create problems of multicollinearity in our regression models, we decided to drop all the inner variables, and stay only with Maximum and Minimum distance. This way, our data would be reflecting the range of distances in which the team was playing.



Next, we looked at the correlations of the cluster variables. The first thing we noticed was that, due to the way we had defined this set of variables, the majority cluster (Cluster A) was very highly correlated to the other clusters. This is due to the fact the the 4 clusters always sum 1, thus they are a linear combination of all the others. For that reason, the first thing we did was dropping Cluster A. After that, we also noticed that there were high correlations between Clusters C and D means and standard deviations. After trying to input both into the model, we decided that the best choice would be to drop the means for Clusters C and D. This way, the final Cluster variables that we stayed were Clusters B, C and D standard deviations and Cluster B mean.



Finally, we looked for any other interesting behavior of the data, but we did not find anything else. Therefore, we stopped our exploratory analysis at this point, and jumped into Control Variable definition and Feature engineering in order to later feed our models.

## Models

As we have already mentioned throughout this document, we tried feeding the models with many combinations of variables. Some examples are:

1. Combining collaboration variables in different ways
2. Models using logarithmic distances and non-logarithmic distances
3. Controlling for the average of the team or for the individual variables
4. Creating different engineered of variables
5. Using different transformations and aggregations of risk variables

In the end, we stayed with the following set of variables because they were the ones that proved to give the best performance to the models:

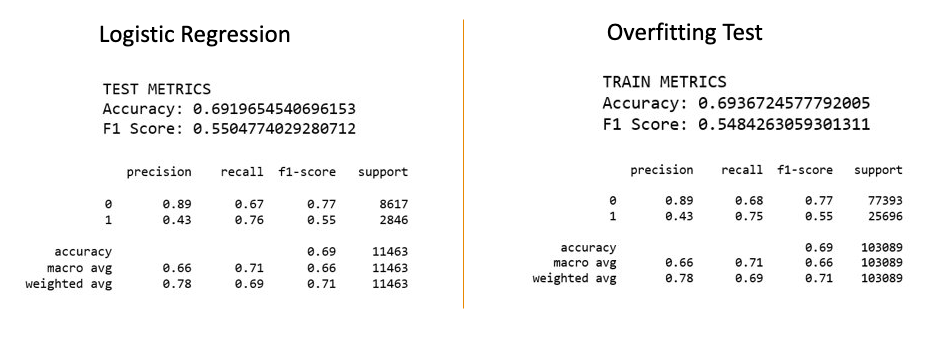
1. Maximum and minimum pair distances (mean and standard deviation for the team)
2. Mean of the second majority cluster
3. Mean and standard deviation for the two minority clusters
4. Agility, previously mentioned in Data Aggregation, is a variable our team feature engineered to showcase the frequency of strategy changes for the teams.
5. Landing risk of every players
6. Maximum and minimum mean risk
7. Control variables for mean distance, cluster proportion, agility, risk and landing risk for every match to attribute to inter-match differences.
8. Team experience is obtained by normalizing by the mean experience of the match and aggregating the experience of all players in the team.

With these variables, we created our final models, that we will explain now.

## Classification Models: Logistic Regression

In the light of identifying behavioral patterns and team dynamic in winning teams, we have created a classification variable: Top 5. It will serve as our dependent variable for the classification models.

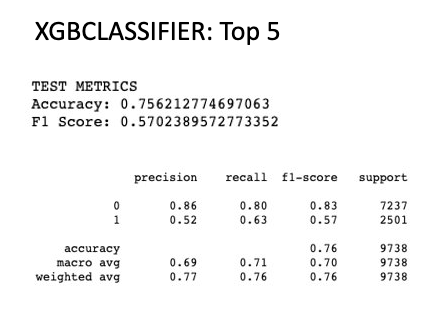
All the variables have proven to be significant and with the logistic regression achieved accuracy was 69.19%. Other traditional metrics to evaluate a classification model such as recall and F1 score are presented in the report below. Moreover, to ensure the models are not overfitted, those reports were generated both for train and test sets. The metrics are nearly identical, illustrating that there is no overfitting.

As classification models do not provide interpretability in terms of coefficients, we have

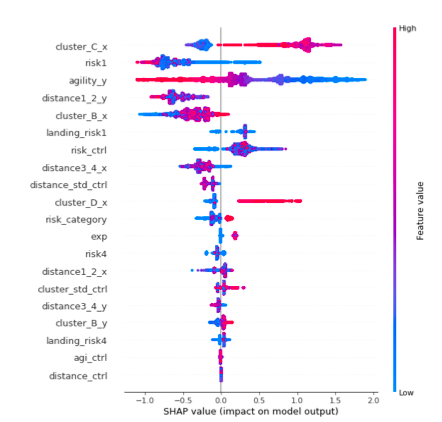
generated a feature importance report.

## Classification Models: XGBOOST

Along with running logistic regressions with variations of variables, we have run more advanced classification models such as XGBoost. As XGBoost combines few hundred trees to make the predictions, the accuracy improved to significant 75%, along with the other metrics for classification models such as recall and F1 score.



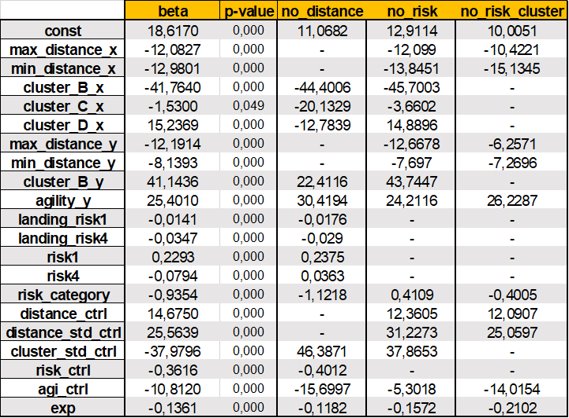
The common problem among advanced classification models is the interpretability. Our team has researched numerous libraries to extract information from the XGBoost model. Finally, we used the SHapley Additive exPlanations(shap) package, which uses Shapley values, originating from coalitional game theory by Lloyd Shapley. These values allow us to assess the impact of variations in the values of each independent variable on the dependent variable. To interpret results, it is necessary to evaluate both the axis with coefficients and indicator of values: high and low. As an example, we can evaluate the agility variable: high values result in decrease of probability of a team winning. Noting that for classification models a result of 1 means a probability of a team being in Top 5, and 0 not being in Top 5.



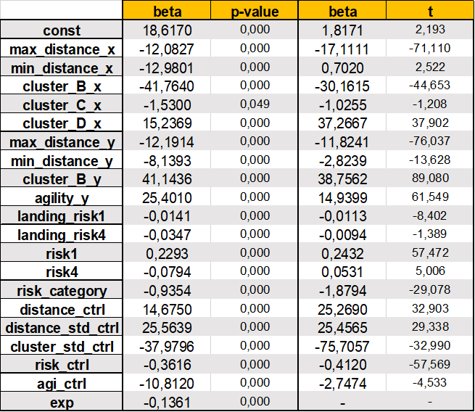
## Regression Models

Since the goal of our project is to be able to identify behavioral patterns and not to have the most accurate models, for Regression we decided to focus our work in trying to get the most accurate Linear Regression that we could. This was due to the ease of explaining these types of models, that would allow us to more easily extract results.

For our main model, we achieved an R2 score of 0.398 and a Mean Squared Error of 35.12 in the Training set. For the Test set, the metrics were 0.389 R2 and 34.98 MSE, showing that our model was not overfitted. In order to understand our variables better and to see if the effect of the variables was real, we created models with different combinations of variables and compared their coefficients. These are shown in the table below, and our conclusion will be explained later in this document.



Moreover, and since we did not have any real hypothesis from the beginning, we tried to replicate our results to check that they were not a result of pure chance. In order to do this, we used the half of the data for which we did not have the Experience variable. That caused a problem for interpretation, since we were missing one variable. However, we observed that the results were the same in general, therefore confirming the importance of our model. In this case, the R2 was 0.275, and the Mean Squared Error was 44.577. The coefficients are shown in the following table.



# Conclusion

## Results

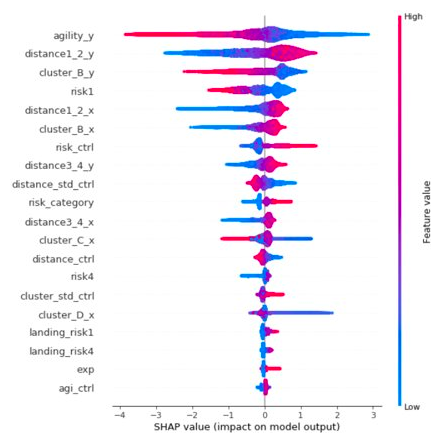
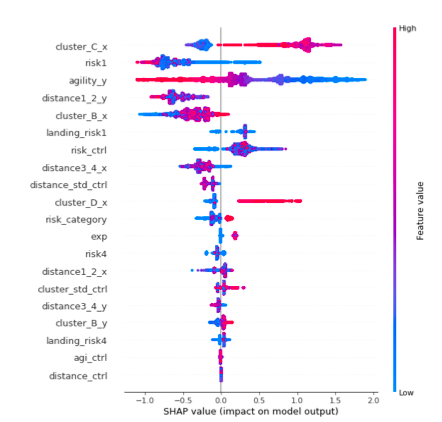
Our final step was to interpret the models we had created. For us, this was the most interesting step of the process, since it concluded our work and would give meaning to all the work done previously. We will analyze the results in three steps: collaboration results, risk results and model comparison results.

For the collaboration, the most captivating finding was that the more agile a team is, the less likely they are to win. This can have two explanations. First, it could mean that organized teams, which have a clear strategy have more chances to win. However, as the variable shows not adaptivity but frequency of pivoting from one strategy to another, it could also mean that some teams do not coordinate and play chaotically, reducing their chances to win. Hence, we can draw a conclusion that it is not about the agility of the team, but about coordination between players in the team.

Another interesting finding was that collaboration increases the chances of winning. The conclusion stems from analyzing the cluster variables: the bigger the non-majority clusters are, the worse a team is expected to perform. Meaning that the more time a team spends in the majority cluster, the more likely they are to win the match. On the other hand, looking at the distance variable, we see that collaborating too closely might be unproductive, observed in the positive coefficient of this variable. We assume that if players play physically close and the enemy attacks, the chances of killing the whole team increase.

For risk, we see that risky landings translate into better rank positions at the end of the game. We believe that when a player manages to survive the initial minutes of the match in a risky area, he/she becomes better equipped, consequently, better prepared for the match. However, when we look at the overall risk, we notice that riskier teams perform worse. Our assumption is that, as the match goes on, riskier players get exposed to better equipped enemies. Therefore, players should start risky, but play safer towards the end of the match. Finally, we see that teams that are exploiters, meaning that they stick to one risky level, get, on average, better rank positions than the other teams by two positions.

This conclusion links to our observations when comparing the outputs of classification models. Looking at Top 5 versus Top 10 models of the XGBoost classifier, we see that, in order to get in the Top 10, collaboration variables have more importance, whereas for the Top 5, risk variables gain importance. Therefore, we conclude that collaboration is more important at the beginning of the game, but towards the end, players should focus on avoiding high-risk more than on collaborating.



Finally, we see that our models are not performing perfectly. In the linear models, we are only able to explain roughly a 40% of the variance, and for the classification models, as we try to explain higher positions, the models get worse. For that reason, we believe that there are many other variables that are important for the match that we are not looking at (e.g. interaction among teams). Moreover, we also believe that luck is very important in this game, and that cannot be captured by any model. However, we believe that we got very good models for our general purpose, and we think that our findings are strong and trustworthy.