# *Predicting the winner of a League of Legends match at the 10-Minute Mark*

Capstone Project

Machine Learning Engineer Nanodegree

Matías Sánchez Cabrera

## *Definition*

### *Project Overview*

League of Legends is a Multiplayer Online Battle Arena (MOBA) game released in 2009 by Riot Games. It has become one of the most popular videos games to date with over 64 million monthly active player base and 7.5 million concurrent players. The game consists of two teams of five players, from which each player chooses one unique champion out of a pool of 130 choices, with the objective to destroy the enemy nexus.

In order to destroy the enemy's nexus, teams must battle out each other in a symmetrical map that consists of three lanes (top, middle and bottom) and a jungle as shown in Figure Nº1. The team that starts at the bottom left is called the Blue Team and the one located at the top right is called the Red Team. The map is surrounded by a fog of war which brings uncertainty of where the enemy team is located at all times. Teams get vision of a particular zone when team members, turrets or minions are around. Players may also place wards around the map to grant vision for a limited amount of time when they are not around.

Teams gather gold in order to buy items to make their champions stronger. Gold is obtained by killing minions (which regularly spawn in each lane), killing neutral monsters (that spawn in jungle camps), killing enemy champions (other players) and securing global objectives. The game requires individual technical skill from each player in order to kill as much minions as possible (generally called creep score or "cs") and in order to kill (and assist in kills) enemy champions (the player with the finishing blow to an enemy champion gets the kill and all those players that helped in the kill get an assists).

There are also global objectives that award gold or buffs to each team member when they are accomplished. The most important global objectives are:

- ***Destroying enemy turrets***: Each team has three turrets per lane and two turrets protecting the nexus. Each turret destroyed awards gold to each team member and creates the opportunity to control bigger portions of the map.
- ***Destroying enemy inhibitors***: Inhibitors are located after the last turret of each lane (often call inhibitor turret). The team that destroys an inhibitor starts spawning supper minions (a much stronger version of minions) in that lane and get a small amount of gold.
- ***Killing the Dragon***: The dragon is a neutral monster that regularly spawns in the dragon pit. When a team gets a kill on a dragon, they receive a permanent buff (which makes team members stronger). As teams get more kills on the dragon, they get different and more powerful buffs.
- ***Killing the Rift Herald***: The Rift Herald is neutral monster that regularly spawns in the baron pit until the game has reached 20 minutes (at this time the Rift Herald disappears and doesn't spawn again in the game). This neutral monster awards global gold a gives one team member the power to make minions stronger for a limited amount of time.

- ***Killing Baron Nashor***: Baron Nashor is the most powerful neutral monster that spawns regularly in the baron pit after the game has passed the 20-minute mark. Baron is much harder to kill than the other neutral monsters and awards a big amount of gold to each team member and the whole teams is awarded with the ability to make minions stronger for a limited amount of time.

In general, League of Legends is a strategic team-oriented game that awards team play much higher than individual skill. Team coordinated plays include placing and protecting wards to deal with vision and reduced uncertainty, protecting and helping champions who deal lots of damage but are easily killed to win team fights and grouping to siege towers allow teams to control parts of the map and help them secure global objectives.

Matches on average last 35 minutes, ranging from 15 to 70 minutes, and teams are allowed to forfeit the match at any particular time after a team vote after the 20-minute mark. Matches can be played on Ranked Mode where players play seriously in order to climb a ladder and Normal Mode where players tend to play in a more relax basis and to try new champions.

### *Problem Statement and Metrics*

This project is aimed to predict the winner of a match with the information available at the 10-minute mark and determine which behavior is the most relevant to kill the enemy nexus.

This problem is going to be solved through supervised learning by collecting match data and modeling it as a classification problem where the label variable is going to be defined as ‘Blue win’ and ‘Red win’. Riot Games has a public API where users can get match data like kills, assists, creep score, towers killed and wards placed, among other, which is going to be used to create the dataset.

The metric chosen to asses the performance of the model is going to be the total accuracy. There is no label that is particularly important to win and win rates by side are pretty balanced, so accuracy makes sense. The model is also going to be compared to a naïve estimator to assess the improvement with the machine learning approach.

## *Analysis*

### *Data Exploration*

In order to construct the dataset, a python script was developed to query the Riot API in search for match data. The basic key of the Riot API allows 50 requests per minute (one request each 1.2 seconds). The scripts consider two types of request to the Riot API, get the last 10 matches played by any particular player and get the detailed match data for a particular match.

The script starts after providing a set of seed players (player Ids or names inserted by hand to start the algorithm) from which it request the recent matches. The algorithm verifies the recent matches in order to check if it worth saving/requesting the detailed match information. In particular, match detail is requested if any particular match was not saved before and if it is a ranked game (to insure quality in match data). For each match saved, the script also stores the player id of each fellow player in the match (10 players per match). Once all matches have been verified, the algorithm starts again with all fellow players. If all players played distinct recent matches and all were ranked matches, with 5 seed players the first iteration could get to 450 fellow players (9 new players for each 10 matches for each 5 seed players).

The script was used to get 86.540 matches spread over four regions; North America (NA), Europe West (EUW), Korea (KR) and Latin American South (LAS). The detail of each match was saved to a JSON file (which is named after the Match Id). This file consist on summary data after the game has ended (like total amount of gold, kills, objectives, winner, etc.) and specific game data per minute (how many wards placed, how many kills, how much gold). Another script was developed to transform almost 20 GB of JSON files into the dataset used for Machine Learning in csv format. This new script captured the relevant information for the dataset and then passed it through to an excel file. The columns captured by this scripts are:

| | |
|---|---|
| Match Id | : The Identifier for the specific match |
| Match Version | : The patch version in which the game was played. Patches may significantly change the game, so this variable is used to control the patches considered in this analysis. |
| Region | : Region in which the match was played. The possible values are NA, EUW, KR and LAS. |
| Queue Type | : Queue from which the match was created. This involves Ranked and Normal games. This variable was used to ensure all games are ranked. |
| Match Creation | : Date in which the match was created. |
| Match Duration | : Length of the match in minutes. |
| Winner | : Winner of the match. ‘100’ represents that the blue team won, and ‘200’ represents that the red team won. |
| Gold* | : The difference of Gold between both teams |
| Dragons* | : The difference of Dragons killed between both teams |
| Rift Heralds* | : The difference of Rift Heralds killed between both teams |
| Wards* | : The difference of Wards placed between both teams |
| Wards Destroyed* | : The difference of Wards destroyed between both teams |
| Top Turrets* | : The difference of Top Turrets destroyed between both teams |
| Mid Turrets* | : The difference of Middle Turrets destroyed between both teams |
| Bot Turrets* | : The difference of Bottom Turrets destroyed between both teams |
| Inhibitors* | : The difference of Inhibitors destroyed between both teams |
| Kills* | : The difference of Kills between both teams |
| Assists* | : The difference of Assists between both teams |

| | |
|---|---|
| Creep Score* | : The difference of Minions killed between both teams |
| Jungle Minions* | : The difference of Jungle monsters killed between both teams |
| Experience* | : The difference of Experience between both teams |

* All differences are considered at 5, 10 or 15 minutes. Positive values represent an advantage for blue team and negative values represent an advantage for red team.

The general metric related to win rates by side seams pretty balanced, with Red side having the highest win rate in all regions (except for LAS) but with less than 1% of difference. The overall Blue side win rate is 49.68% for 86.540 analyzed in this project.

| ***Region*** | ***# Matches*** | ***Blue win rate*** |
|---|---|---|
| NA | 21824 | 49.57% |
| EUW | 22960 | 49.24% |
| LAS | 22775 | 50.22% |
| KR | 18981 | 49.70% |
| *Overall* | *86540* | *49.68%* |

Additional information rewarding the data set construction:

- In order to select Seed Players, the best players of each region were identified (through online websites that show the ranked ladder). Although only Challenger players were included as seed players (the best of the best) matches from all divisions end up been captured.
- Matches were collected from this year's patchs before the mid season changes to Dragons and Mages (From patch 6.2 to 6.6).
- Although relevant information could be found rewarding specific champions picks (Are they tanks? Do they just deal damage? Do they have crowd control abilities?) this analysis is based only team objectives in order to simplify the analysis and conclusions.

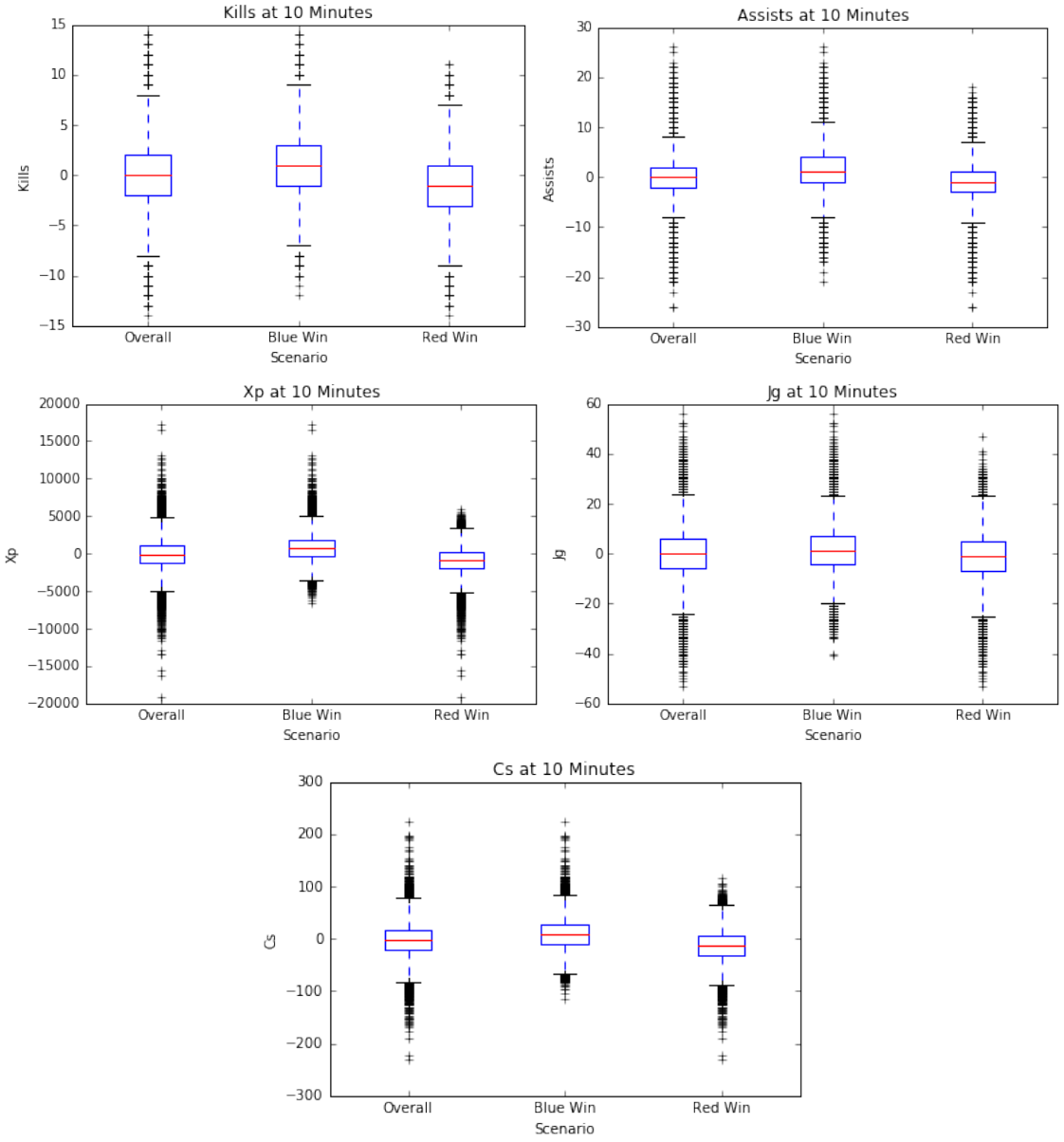
## *Exploratory Visualization*

In order to visually analyze if any particular objective is more relevant towards predicting the winner of a match, all objective wise metrics (like Kills and Assists) where analyzed in three scenarios:

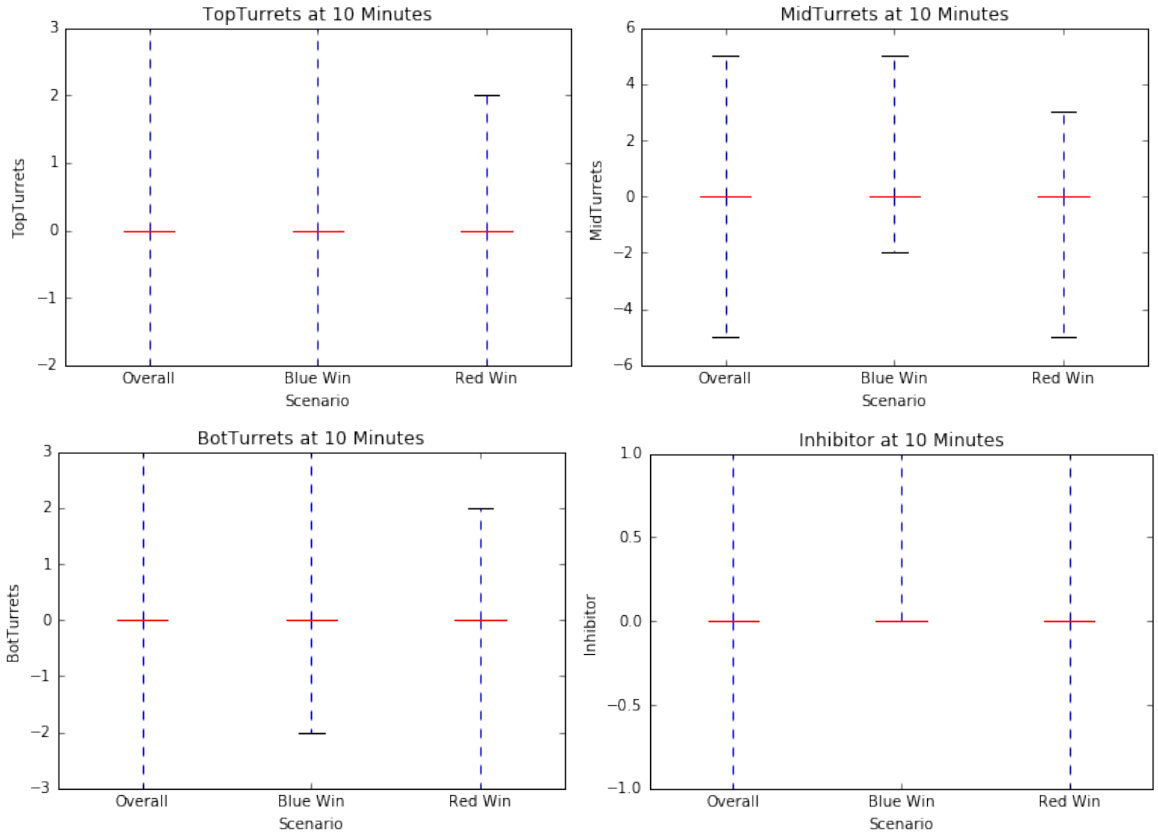
| ***Scenario*** | ***Description*** |
|---|---|
| Overall | All matches, independent of the winner |
| Blue Win | Metrics in matches where Blue side won |
| Red Win | Metrics in matches where Red side won |

Comparing match objectives with these scenarios, it is made clear that the most relevant objectives that determine which team is going to win at 10 minutes are Kills, Assists, Experience, Jungle Minions and Creep Score. In most of this scenarios the winning team has

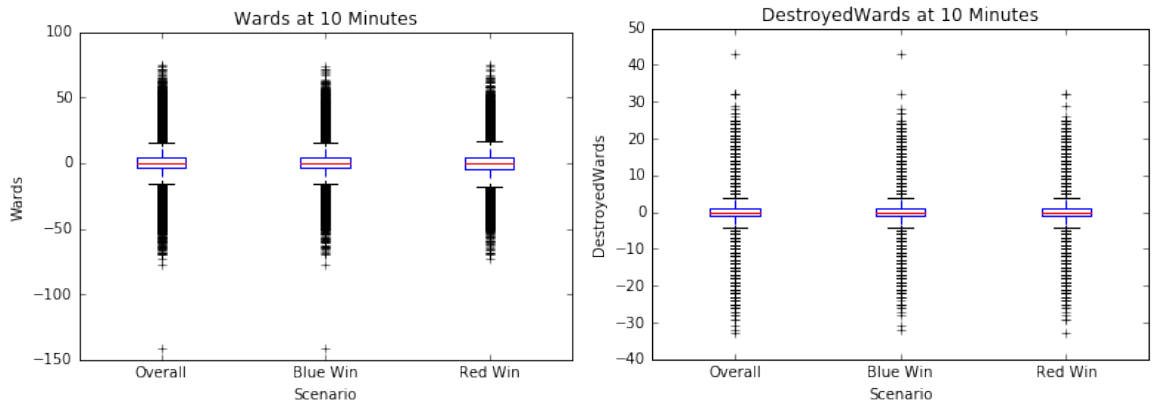
an advantage with their opponents. In the case of Experience, Jungle Minions and Creep Score, this advantage seams very small, but the distribution shows a clear tendency towards the winning team. This shows that although the difference on average may seam small, in matches were a team has an advantage in this objectives the likelihood of winning is higher.

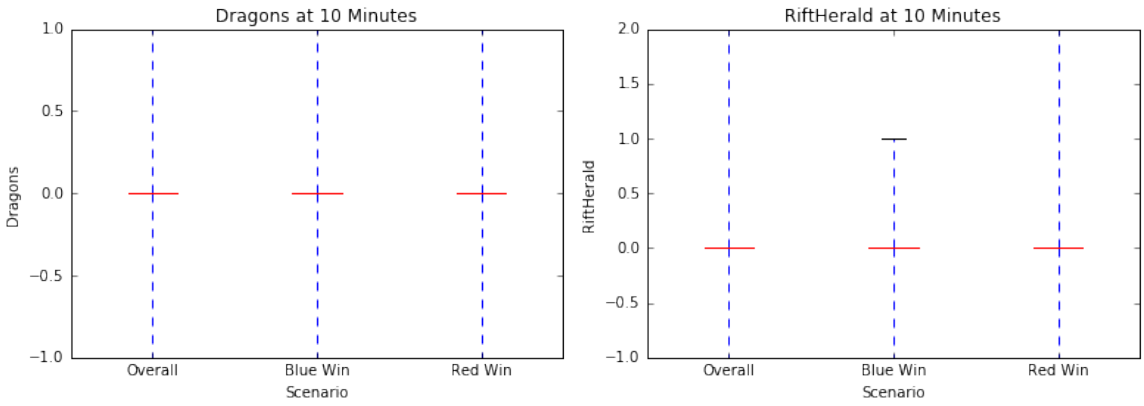



In theory, the objectives of Inhibitors, Top, Middle and Bottom turrets destroyed should a tendency towards predicting the winner because they show lane advantages and control of the map. Data shows that only Middle Turrets seam relevant towards identifying the winner because the winner shows a larger distribution towards taking that objective. The rest of the objectives do not show this tendency. My feeling this is due to teams not been able to these objectives before the 10 minute mark as often, so there are only extreme cases were this happens.




Finally, the objectives of Wards placed, Wards Destroyed, Dragons and Rift Heralds show practically no differences between the winners. This is due to the difficulty of destroying wards in the early game because in general teams don't have the required items to identify wards in the map. In the case of Dragons and Rift Heralds, both objectives are quite hard to get before the 10-minute mark, so there is no clear tendency towards predicting the winner.






## *Algorithms and Techniques*

Determining the winner of a particular match is a classification problem because the goal is to predict the winner label which has values for 'blue side win' or 'red side win'. In this sense, classification algorithms like Logistic Regression, Decision Trees, Support Vector Machines and Ensemble Trees are appropriate. The reason behind choosing this algorithms are:

- ***Logistic Regression***: A simple algorithm which is fast and easy to understand the results. Although the predictive power is smaller compared to more complex algorithms, I have found good results using this algorithm.
- ***Decision Trees***: They are also fast and easy to understand the results graphically. In the past I have found that they tend to over fit.
- ***Support Vector Machines***: Although this algorithm has shown very good results in the past, this dataset consists of 86k observations with 13 dimensions and so it is likely to be slow to train.
- ***Ensemble Algorithms***: This are complex algorithms were several models are trained. This projected used Random Forest and Ada Boost were many decision trees are trained. The key difference is that each model focuses more on classification errors from the past models. Although their predictive power may be strong than the other algorithms, they are quite likely to end up overfishing.

In order to asses the quality of the predictive functions found by these algorithms we are going to use cross validation, splitting the data into training and testing samples, in order to find the algorithm with the best accuracy on the test set without incurring into over fitting in the train sample. Because no class is more important than the other, the general metric used to evaluate the performance of this model is going to be accuracy.

### *Benchmark*

The Gold Difference metric is the easiest metric to look at to determine which team is winning a match at any particular moment because it is a good summary of most objectives that reward gold (kills, assists, cs, turrets, etc.). Competitive matches that are streamed over the internet use this metric to asses which team has the advantage.

A naïve approach is determined in order to compare the machine learning approach. The naïve approach will predict the winner of a match by looking at which team has the largest amount of gold (independent of difference amount) at the 10-minute mark.

## ***Methodology***

### *Data Preprocessing*

The data pulled from the Riot Games API is in a very good state with no missing values and/or errors in the data. Most preprocessing tasks were completed in the script in charge of reading the match detail data from the JSON files and create the Dataset for machine learning in csv format.

The main task regarding preprocessing were related to calculate objectives (for example each kill is reported separately so the script was in charge of adding all kills for each team) and then calculating the differences between each team. There are no categorical features, so the use of one hot encoding was not needed for this particular data set. Some algorithms required the data to be normalized (for example SVM) so the data was normalized through standard normalization (0-1 normalization) before the model training phase. Some objectives are quite rare to be achieved before the 10-minute mark but the 0-1 normalization was still used because this objective denote a clear advantage.

The distribution of each objective was checked in order to identify outliers. There are matches were teams have a huge advantage (14 Kills of difference and 14k Gold difference) but they do not seam extremely rare and evenly distributed between teams. In this sense I consider them to be valuable information rewarding games were one team is dominating.

### *Implementation*

All scripts regarding with the acquisition of data from the Riot Games API, the preprocessing of match data from the JSON files, the visualization of data and the machine learning scripts were developed over Jupyter Notebooks using python 2.7 and the relevant Python libraries like Pandas, Numpy and Scikit Learn. Scikit Learn is able to provide all the relevant algorithms and metrics used to face this problem. All scripts and example files are available to test the results.

About 70% of the time spent developing this scripts were related to data acquisition and data preprocessing. The first challenges were related to getting used to work around the

Riot Games API. The RiotWatcher library for Python was used due to its simplicity in calling requests methods to the API. At first the data acquisition script was quite slow getting matches because it called for match information of matches that were not ranked and searched in players that had very few ranked games. This script data acquisition script was exposed to several optimizations so that it would capture match data faster by requesting only relevant players (that tend to play ranked games) and ignore non ranked matches. It has to be considered that the basic key for the API is able to request data each 1.2 seconds, so the process lasted for a couple of days.

The next challenge was related to understanding the JSON structure to capture relevant data. A script was developed to read through 16 GB of json files searching for the match information that is used in the dataset. Once the JSON files were understood and the script developed, only a couple of minutes are needed to construct the csv files with the relevant information.

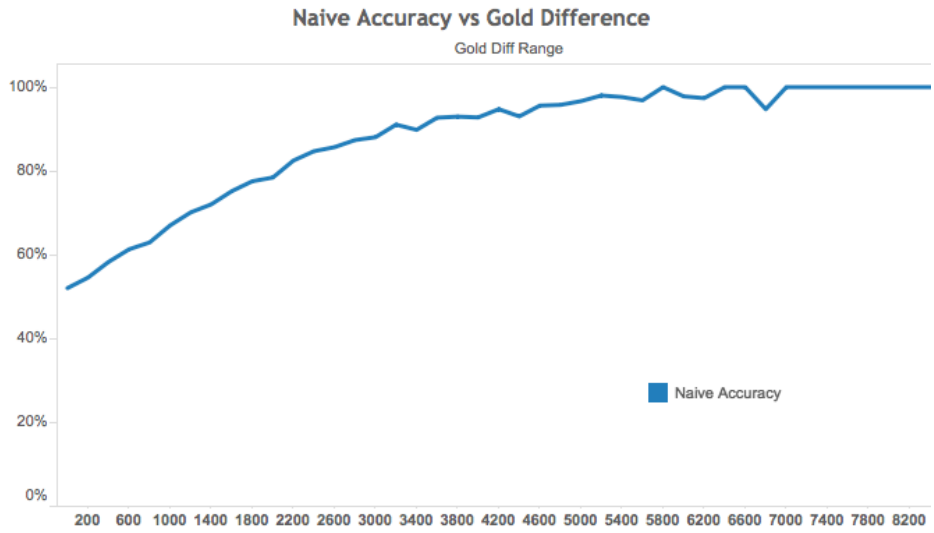
The machine learning script was not a challenge to implement (due to past exercises in the Nanodegree) but it was a challenge to get to relevant results. At first the dataset consisted on number of objectives achieved by each team (for example kills had two variables, blue_kills and red_kills) but slightly better results were achieved by using the difference in only one variable. Most algorithms were fast at the training stage with the exception of SVMs that take 4-5 minutes to get a results (although still reasonable times).

Most algorithms lead to over fitting or bad results compared to the benchmark. Different algorithm configurations were used in order to get to better results, but big improvements were not found.

### *Refinement*

The initial solution for this problem is related to the naïve approach, were the team with the highest amount of gold at the 10-minute mark is predicted to win. This approach has a 70.14% accuracy on the whole sample. As expected, the number of classification errors is higher when there is a small difference in gold between teams and the number of classification errors gets lower as the gold difference increases. The Naïve approach achieves a 100% accuracy in matches were the gold difference between teams is equal or higher to 8000 gold, and the worst accuracy is achieve were the gold difference is between 0 and 200 were only 52% (almost random) matches are correctly predicted.

After an extensive evaluation of different machine learning algorithms, a 70.4% accuracy was achieved using all data except the feature ‘Gold’ and a 70,8% accuracy was achieved using all the data including ‘Gold’. The reason behind hiding this feature was because most objectives award Gold, so the importance of this variable may explain the importance of others. Solving this problem without this feature can show us which objective is more relevant towards winning the game rather than being explain by the Gold.




After an extensive evaluation of different machine learning algorithms, a 70.4% accuracy was achieved using all data except the feature 'Gold' and a 70,8% accuracy was achieved using all the data including 'Gold'. The reason behind hiding this feature was because most objectives award Gold, so the importance of this variable may explain the importance of others. Solving this problem without this feature can show us which objective is more relevant towards winning the game rather than being explain by the Gold.

## ***Results***

### *Model Evaluation and Validation*

Several machine learning algorithms were tested with this dataset. In particular, ensemble algorithms like the Random Forest Classifier and Ada Boost Classifier (with Random Forest as base estimator) showed good performance on the test set but over fitted on the training set.

| ***Random Forest*** | *Pred Blue win* | *Pred Red win* | | |
|---|---|---|---|---|
| *Blue win* | 32082 | 211 | ***Train Accu*** | 98.40% |
| *Red win* | 827 | 31785 | ***Test Accu*** | 67.22% |

| ***Random Forest estimators = 10*** | *Pred Blue win* | *Pred Red win* | | |
|---|---|---|---|---|
| *Blue win* | 32104 | 189 | ***Train Accu*** | 98.43% |
| *Red win* | 827 | 31785 | ***Test Accu*** | 66.73% |

| ***Random Forest estimators = 100*** | *Pred Blue win* | *Pred Red win* | | |
|---|---|---|---|---|
| *Blue win* | 32293 | 0 | ***Train Accu*** | 100% |
| *Red win* | 0 | 32612 | ***Test Accu*** | 69.49% |

| ***Adaboost (with Random Forest)*** | *Pred Blue win* | *Pred Red win* | | |
|---|---|---|---|---|
| *Blue win* | 32293 | 0 | ***Train Accu*** | 100% |
| *Red win* | 0 | 32612 | ***Test Accu*** | 68.51% |

Other kind of algorithms like SVM, Decision Trees, K-nearest Neighbors and Gradient Descent showed poor performance on the test sample (almost always behind the naïve approach).

The Logistic Regression Classifier and the base Ada Boost Classifier were the best models found, showing the best performance on the test set, beating the naïve approach by approximately 0.65% and showing good generalization by not showing over fitting in the training set. On the other hand, both algorithms had the 'feature importance' attribute available so we could identify the most relevant behavior in order to predict the winner.

In the case of the Ada Boost Classifier, the best configuration was found using gridsearch over the n_estimators and learning_rate attributes and the base base estimator (Decision Tree Classifier). The parameter n_estimators was search over the values of 1, 10, 50 and 100 while the learning rate was searched over the values 0.1, 0.3, 0.5, 0.7 and 1. The best parameter configuration was n_estimators = 100 and learning_rate = 0.1. This makes sense because it keeps the algorithm out of overfitting by using many estimators but with each iteration the algorithms are less relevant. In the case of the Logistics Regression, the base model was found to achieve the best results.

| ***Logistic Regression*** | *Pred Blue win* | *Pred Red win* | | |
|---|---|---|---|---|
| *Blue win* | 22712 | 9581 | ***Train Accu*** | 70.59% |
| *Red win* | 9510 | 23102 | ***Test Accu*** | 70.61% |

| ***Adaboost base*** | *Pred Blue win* | *Pred Red win* | | |
|---|---|---|---|---|
| *Blue win* | 22678 | 9823 | ***Train Accu*** | 70.50% |
| *Red win* | 9321 | 23483 | ***Test Accu*** | 70.52% |

## *Justification*

The evaluation of different machine learning algorithms and working around with some feature engineering showed little to no improvements over the naïve approach

accuracy. This benchmark is in fact quite powerful due to the fact that Gold is a very good signal for how well a team has achieved objectives in game.

The machine learning approach show a very little improvement over the naïve approach, getting a performance improvement of over 0.65%. The big difference between both approach is that the machine learning approach has the ability to identify which behavior in the first 10 minutes is more important towards winning match (this will be discussed in next section).

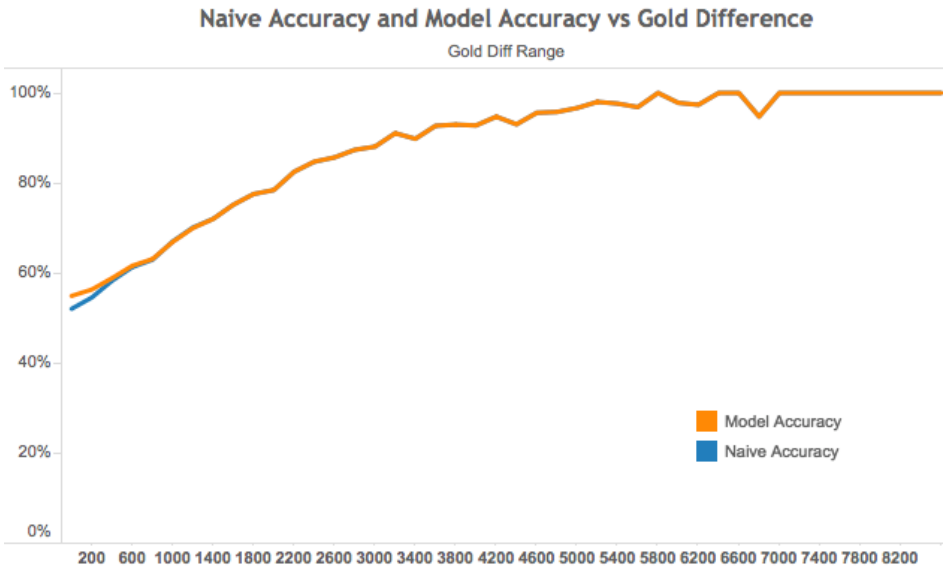
The main objective of this study was to predict the winner of a game at the 10-minute mark and to identify the most relevant behavior to win the game. With the naïve and machine learning approach, over 70% of the games are predicted correctly and the machine learning approach clearly identifies the most important behavior to win a game. In this sense, the objective of this study is achieved.

## *Conclusion*

### *Free-Form Visualization*

The Machine Learning approach had to advantages over the naïve approach. In first place, a small (but relevant) accuracy improvement was achieved and secondly, the model results shows us insights of which objectives are the most relevant to predict a winner at the 10-minute mark.

The machine learning approach achieved a low improvement over the naïve approach, but it was an improvement over matches were the gold difference was low. As the gold difference was bigger, both the naïve and machine learning approach had the same accuracy. This can be explained because some important objectives like Experience are not measured by gold.




On the other hand, the feature importance attribute of the Logistic Regression Classifier lets us see which variable is most important for the results in the machine learning approach. In this case, the most important variables in order of importance are 'Experience', 'Creep Score', 'Kills', 'Dragons' and 'Jungle Creeps'. 'Experience' and 'Dragons' are the only objectives that are not directly related to gold but they clearly show a sign of in game advantage. In general players have more experience when the opponent is pushed out of lane (due to kills or outplay) and the Dragon (at least in this set of patches) is taken when there is no threat of opponents (again, due to kills or pushed out of lane).

| ***Feature*** | ***Importance Score (Logistic Regression)*** | ***Importance Score (AdaBoost)*** |
|---|---|---|
| Dragons | 17.28 | 4 |
| Rift Herald | 4.66 | 2 |
| Wards | 2.76 | 0 |
| Destroyed Wards | 2.1 | 0 |
| Top Turrets | 1.58 | 0 |
| Mid Turrets | 4.95 | 0 |
| Bot Turrets | 3.33 | 0 |
| Inhibitors | 3.69 | 0 |
| Kills | 32.54 | 12 |
| Assists | 9.06 | 6 |
| Creep Score | 33.22 | 12 |
| Jungle Minions | 12.66 | 8 |
| Experience | 63.5 | 56 |

This results shows us that the first 10 minutes of the game should be focused around killing as much minions as possible ('Creep Score' and 'Jungle Minions') and trying to get Kills of opponents. Although this is the general behavior of most League matches, there

should be low emphasis in global objectives like Turrets (specially Top Turret which seams irrelevant), Rift Herald and Wards (placed and destroyed) due to them not being relevant in the machine learning approach. It must be noted that these objectives could be relevant in later stages of the game.

In this sense, the machine learning approach gives us a little accuracy improvement over the naïve approach, but gives a lot of insights rewarding which is the appropriate behavior in the first 10 minutes of the game in order to increase the likelihood of winning.

## *Reflection*

This study gave me the chance to get to work with an established API and creating scripts to acquired and process a huge set of data with Python. Apart from creating this data set, I got the chance to work with data from a field (videogames) which I'm passionate about. In particular:

- ***Interesting aspects of this project:***
  - The script in charge of acquiring game data is based on an interesting algorithm which was optimized along the way. In particular, finding which match to store is particularly difficult, and using seed players to start the process made that with as few as 3-4 players were inserted, the script could run for days searching for matches.
  - The match data has specific information of each match for every minute in game. Further exploration of this data could be use to conduct a series of different studies related to League Matches.

- ***Difficult aspects of this project:***
  - In general, this game seams pretty balanced, with no particular sign showing a distinctive difference to predict the winner of a match. At some point this was particularly frustrating due to the fact that the improvement of the machine learning approach was quite low compared to the naïve approach. The big difference was related to the interpretation of the results.

## *Improvement*

Further studies could be made to improve the performance of this predictive model by exploring additional information to this data set. As this study was related to identify which team behavior lead to a higher chance of winning a game, only general information regarding objectives was used. Further information could be used like:

- **Specific team composition (champions chosen)**: Each champion has its unique play style; some may be stronger late game while others excel at playing aggressive early game.
- **Which role got an advantage in gold early**: Some roles are much more gold-hungry than others. For example, AP and AD carries make a much better use of gold than Supports early game.
- **Analyze specific differences by the quality of the player**: The ranked ladder allows us to identify the level of play of each team member. The playstyle of Bronze players (bottom of the ladder) may be significantly different from Diamond/Master Players (top of the ladder).
- **Analyze the impact of wards in game**: In this study, wards were only used to quantify the amount of wards placed and wards destroyed. Although the machine learning approach showed that they were not relevant to predict the winner of a match, it is well known that vision control is a crucial aspect of the game. Further studies could be done by analyzing the location were wards are optimal to be placed and the impact of them in the later stages of the game.