League of Legends
Strategy proposal

Liberati Pietro Andrea, 1814440
Battistini Tommaso, 1869913

Big data computing
A.A. 2020/2021

Contents

1 Project Proposal

1.1 Problem

League of Legends (LoL) is a massive online multiplayer game that has become very popular in the last decade. It has the largest competitive E-Sports scenes; the last world championship peaked 100M viewers, with prizes that reached about 6.45M dollars for the winning team. As expected from a videogame, LoL generates a great amount of data, that can be used to develop a winning strategy.
Since the videogame market is increasingly growing (https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/), so is the search for data science based approaches that aim to understand these games' mechanics.
LoL is a cooperative game in which players have to follow a common strategy to pursue particular goals inside the game. So, which are the main goals that a team has to follow to maximize the probability of winning a match?
We furthermore want to develop a model to predict the winning team of a game given the game's stats.

1.2 Dataset

We found a very large dataset featuring 200k games at the highest level of play. For each game dataset has about 50 columns, each one describing which team managed to reach a particular objective.
We therefore intend to understand which objectives usually leads to victory once obtained, in order to offer a coaching/data-analysis service to professional teams.
We got the dataset from Kaggle at this url:
`https://www.kaggle.com/gyejr95/league-of-legends-challenger-ranked-games2020`

1.3 Methodology

For each row, our dataset indicates which team won the game.
For what concerns the binary classification part of this project, we intend to develop and compare models using `logistic regression` and `decision tree` algorithms.
On the other hand we will use built-in function "gini" of Random Forests.

1.4 Evaluation framework

To evaluate our models we will use the most common metrics for classification: we will compare all our models by using auROC as a baseline metric, and then show some model-specific measures (such as thetas for logistic regression).

2 Data meaning, acquisition, exploration and pre-processing

In this section, we discuss the meaning of the data and it's structure, in order to explore how our data is managed to solve our problem.

2.1 Meaning

We believe that it's useful to briefly introduce the game's structure, in order to get the reader to properly understand the information given by values for every feature. Basically, in League of Legends there are two teams of 5 players, each controlling a character (Champion) in a closed, relatively small map. Each team has it's own base (Nexus), protected by towers and inhibitors. The map is also populated by AI - controlled groups of monsters (minions, jungle camps, Dragons, Barons) that give rewards (experience/gold) if slain. The purpose of the game is to level/gear up your character faster than the enemy team (using experience/gold), in order to ultimately destroy the enemy base.

2.2 Acquisition

As said in section 1, for this project we found a dataset containing 200k instances of ranked League of Legends games, divided in three sub-datasets, according to the "level" of play: (Master, Grandmaster, Challenger). Players from Master level and upwards are the elite of this game (top 0,05 percentile), and are mostly hired by teams who compete for high - prized tournaments.
We take the union of these three datasets to obtain the final one, that has about 200k instances.
The features of our dataset are:

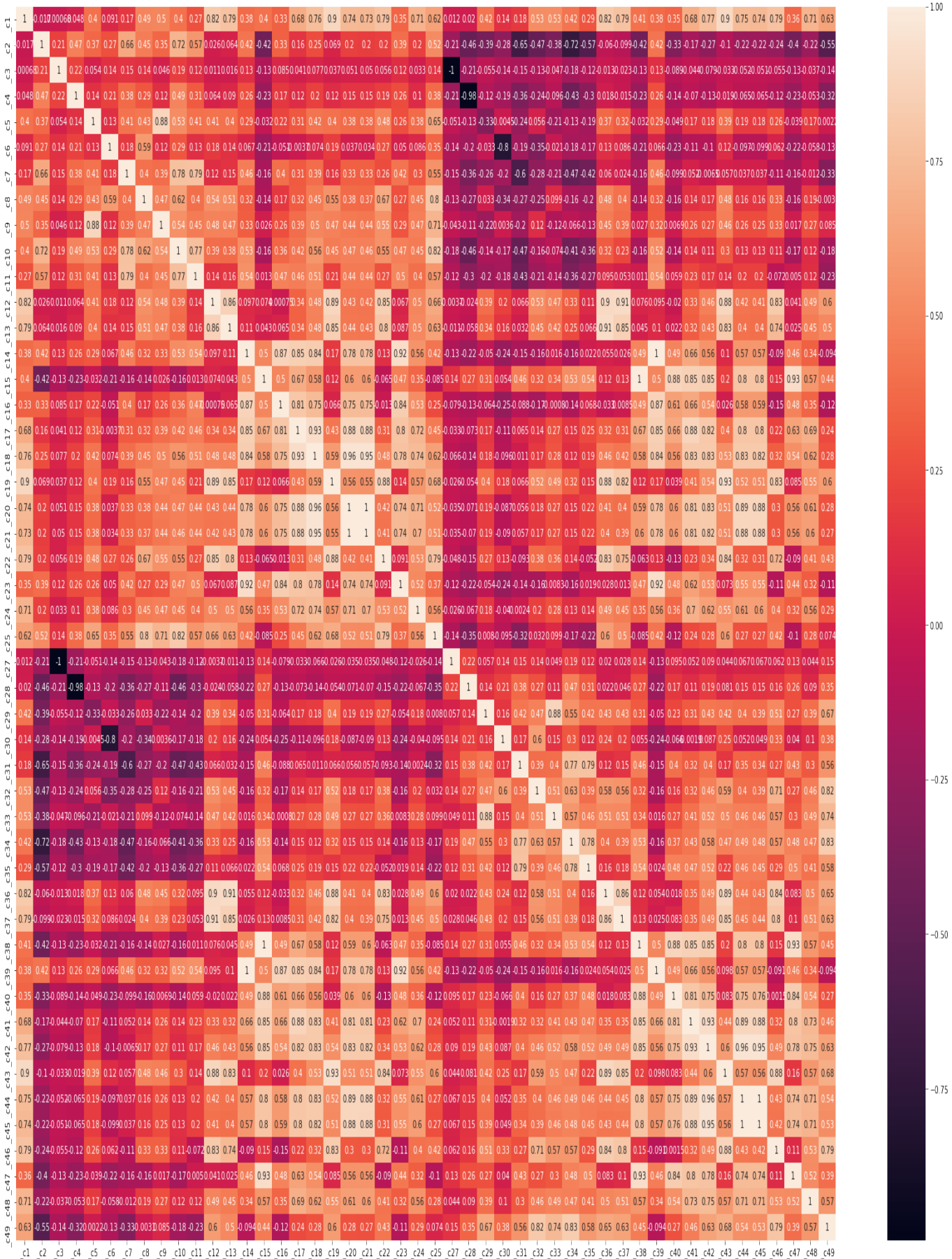
- gameId
- gameDuraton
- blueWins *
- blueFirstBlood *
- blueFirstTower *
- blueFirstBaron *
- blueFirstDragon *
- blueFirstInhibitor *
- blueDragonKills
- blueBaronKills
- blueTowerKills
- blueInhibitorKills
- blueWardPlaced
- blueWardkills
- blueKills
- blueDeath
- blueAssist
- blueChampionDamageDealt
- blueTotalGold
- blueTotalMinionKills
- blueTotalLevel
- blueAvgLevel
- blueJungleMinionKills
- blueKillingSpree
- blueTotalHeal
- blueObjectDamageDealt

- redWins *
- redFirstBlood *
- redFirstTower *
- redFirstBaron *
- redFirstDragon *
- redFirstInhibitor *
- redDragonKills
- redBaronKills
- redTowerKills
- redInhibitorKills
- redWardPlaced
- redWardkills
- redKills
- redDeath
- redAssist
- redChampionDamageDealt
- redTotalGold
- redTotalMinionKills
- redTotalLevel
- redAvgLevel
- redJungleMinionKills
- redKillingSpree
- redTotalHeal
- redObjectDamageDealt

The star features (*) have binary values, the rest ones are pure Integers.

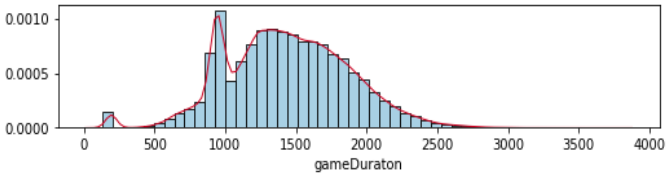
2.3 Exploration and preprocessing

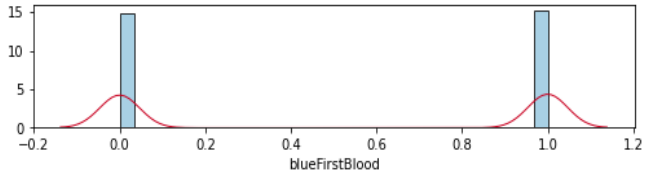
Once the data is obtained, we remove columns that directly correlate with the target feature, or that are useless.
To make these choices, we made a heat map of correlations between features:

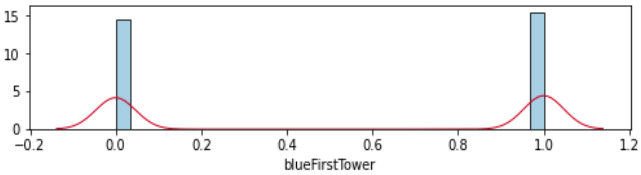

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20 c21 c22 c23 c24 c25 c27 c28 c29 c30 c31 c32 c33 c34 c35 c36 c37 c38 c39 c40 c41 c42 c43 c44 c45 c46 c47 c48 c49
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

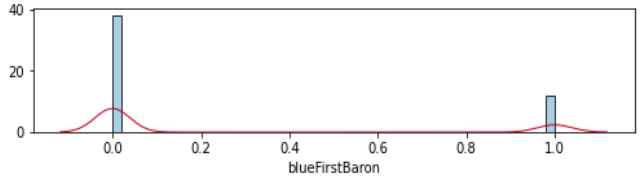
We only removed features that directly correlated with the target, or that had a direct correlation with some other feature whose importance is relatively low. To complete this process, we had to extract features importance repeatedly using the Random Forest classifier. To our surprise, some of the features that correlate most to the target column weren't selected my the random forest to be the most important ones.

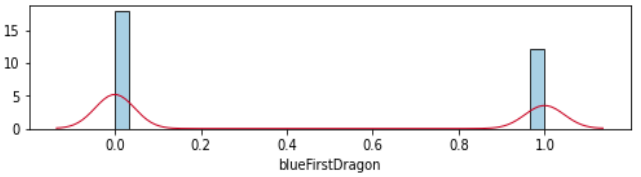
The graphs relating to the distribution of the values of each field are shown below.

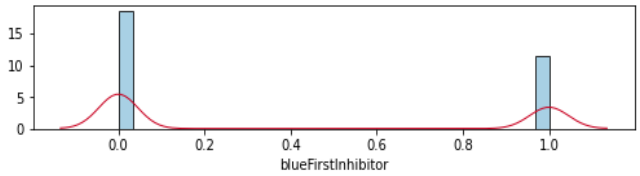

0.0010
0.0005
0.0000
0
500
1000
1500
2000
2500
3000
3500
4000
gameDuraton

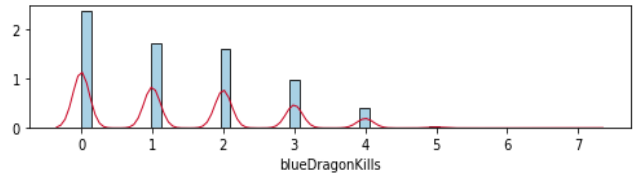

15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
blueFirstBlood

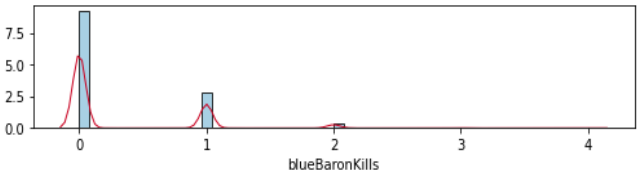

15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
blueFirstTower

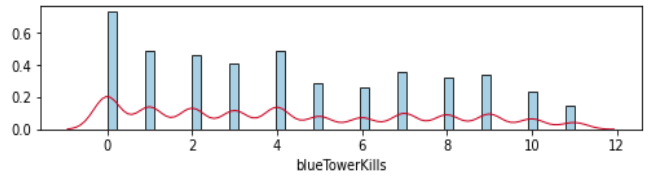

40
20
0
0.0
0.2
0.4
0.6
0.8
1.0
blueFirstBaron

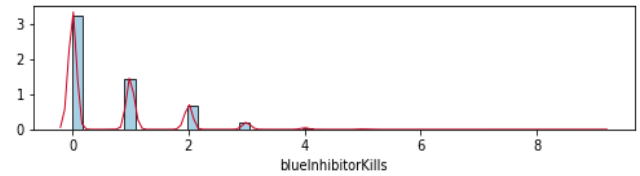

15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
blueFirstDragon

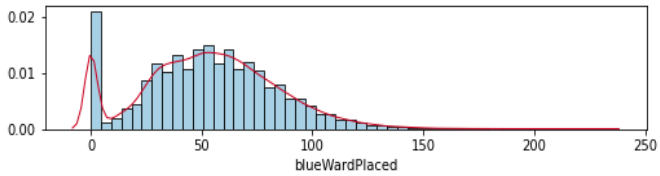

15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
blueFirstInhibitor

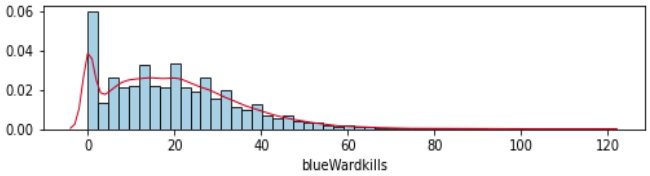

2
1
0
0
1
2
3
4
5
6
7
blueDragonKills

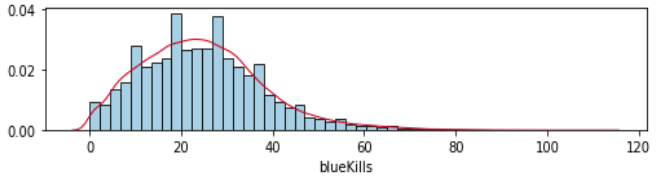

7.5
5.0
2.5
0.0
0
1
2
3
4
blueBaronKills

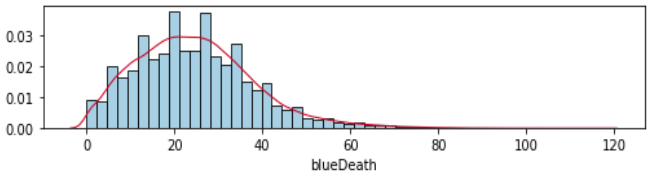

0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
blueTowerKills

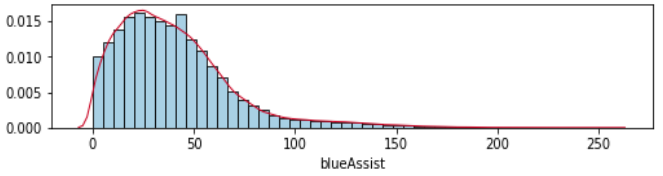

3
2
1
0
0
2
4
6
8
blueInhibitorKills

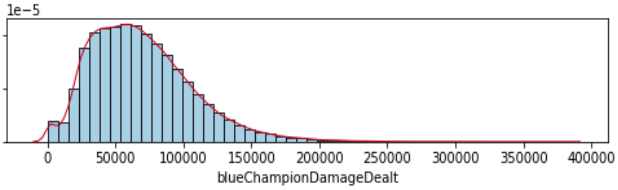

0.02
0.01
0.00
0
50
100
150
200
250
blueWardPlaced

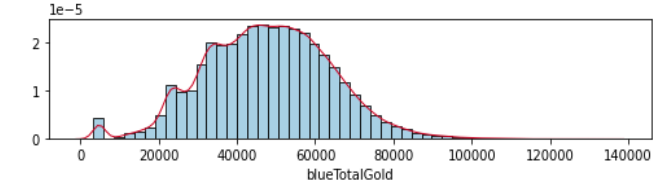

0.06
0.04
0.02
0.00
0
20
40
60
80
100
120
blueWardkills

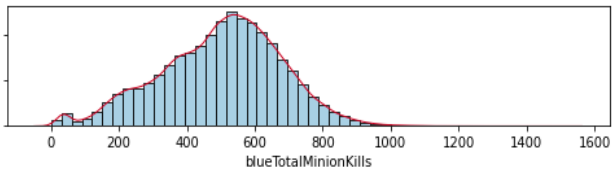

0.04
0.02
0.00
0
20
40
60
80
100
120
blueKills

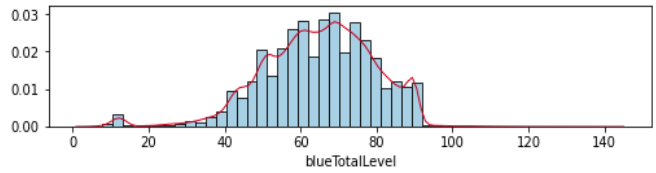

0.03
0.02
0.01
0.00
0
20
40
60
80
100
120
blueDeath

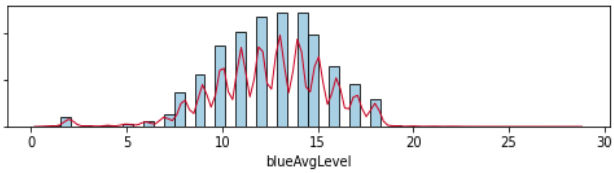

0.015
0.010
0.005
0.000
0
50
100
150
200
250
blueAssist

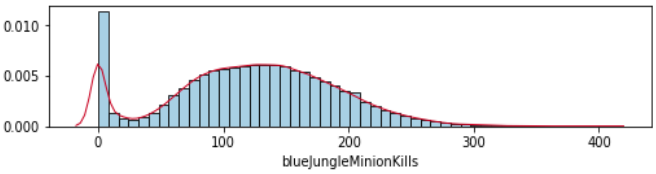

1e−5
1.0
0.5
0.0
0
50000
100000
150000
200000
250000
300000
350000
400000
blueChampionDamageDealt

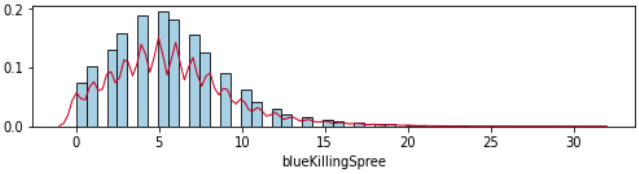

1e−5
blueTotalGold

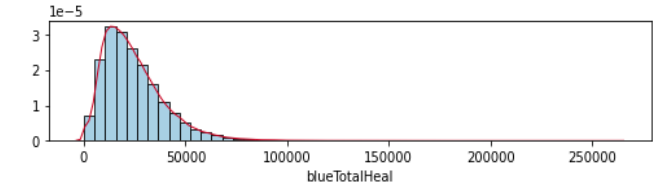

blueTotalMinionKills

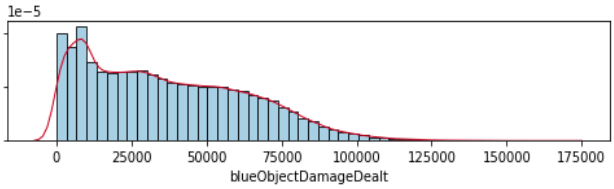

blueTotalLevel

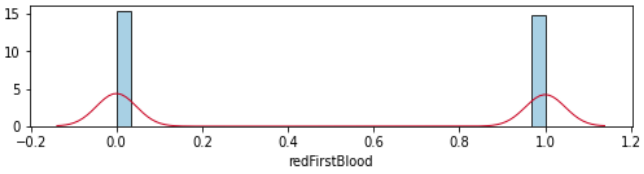

blueAvgLevel

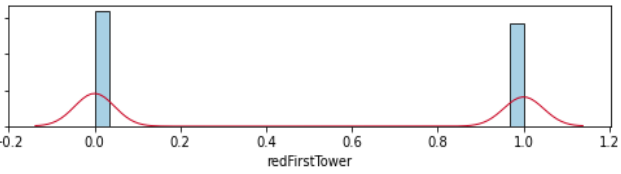

blueJungleMinionKills

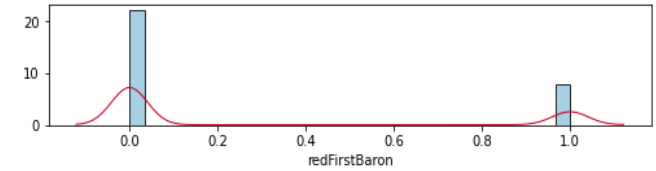

blueKillingSpree

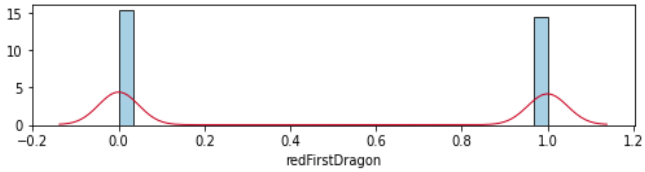

1e−5
blueTotalHeal

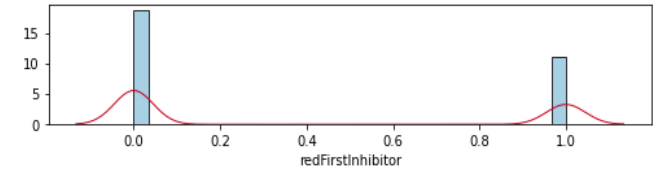

1e−5
blueObjectDamageDealt

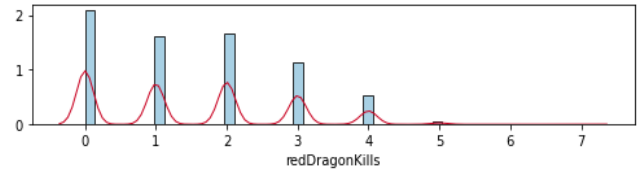

redFirstBlood

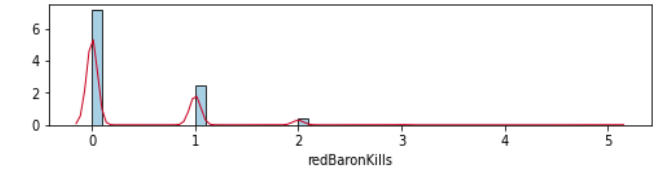

redFirstTower

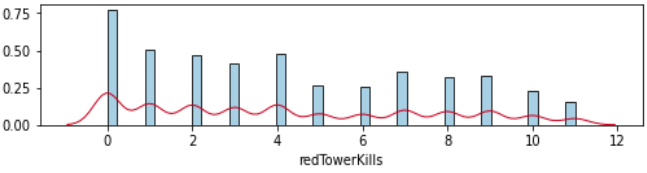

redFirstBaron

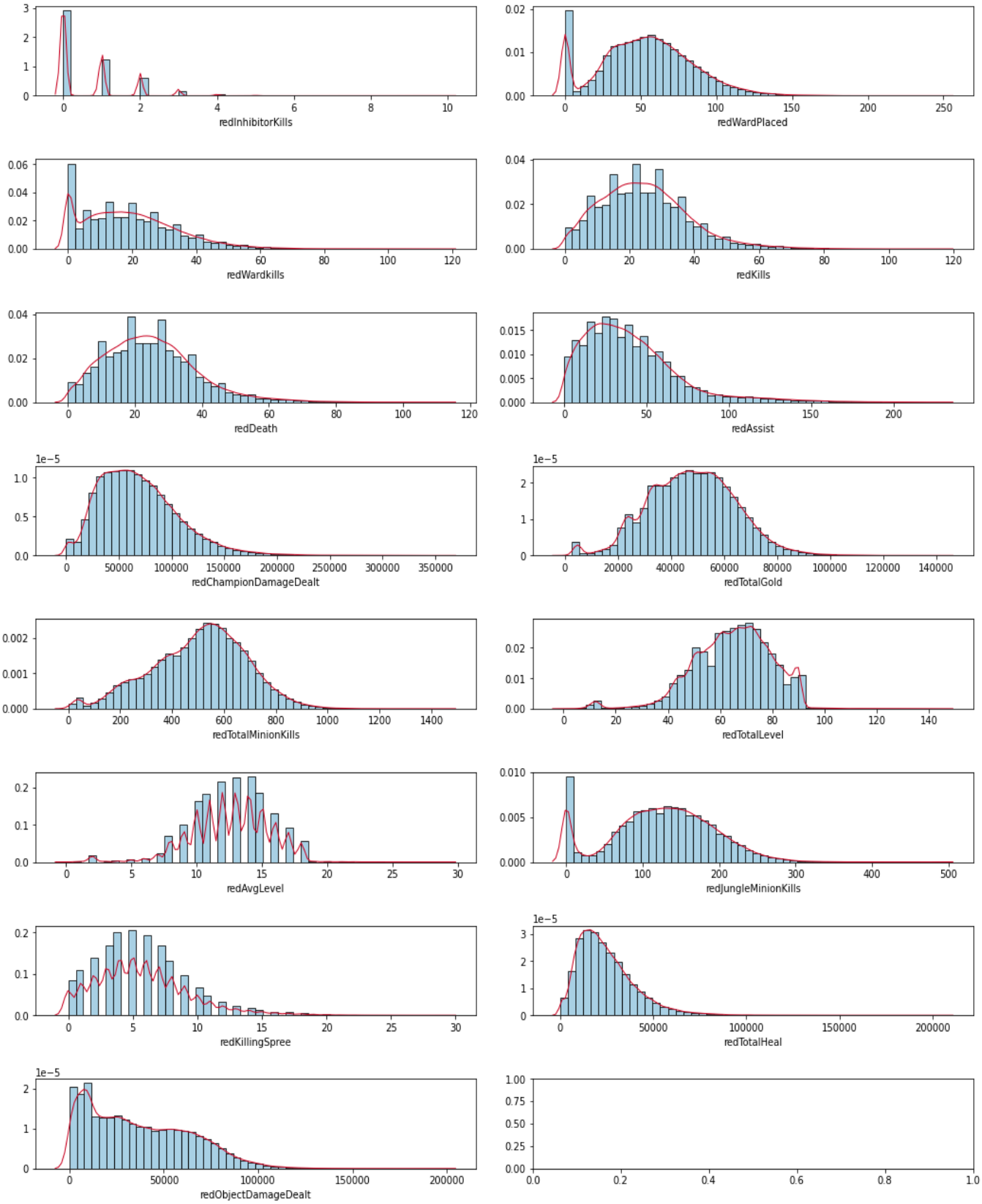

redFirstDragon


redFirstInhibitor


redDragonKills


redBaronKills


redTowerKills



As we can see, the non-binary features all approximately follow a Gaussian distribution.

3 Classification models

In this section we present the classification models we implemented to try an predict the winner of each game/sample.
We want to do a comparison between three different classification models, in order to derive which of these is the best in our application.

3.1 Logistic regression

First of all we applied the `logistic regression` model.
`Logistic regression` is used to predict the value of a variable, if the output space is a closed values space. In the simplest case that space contains only two values: 0 and 1. Our case is this one; in fact the value of BlueWins is 1 if blue player win the match, 0 otherwise.

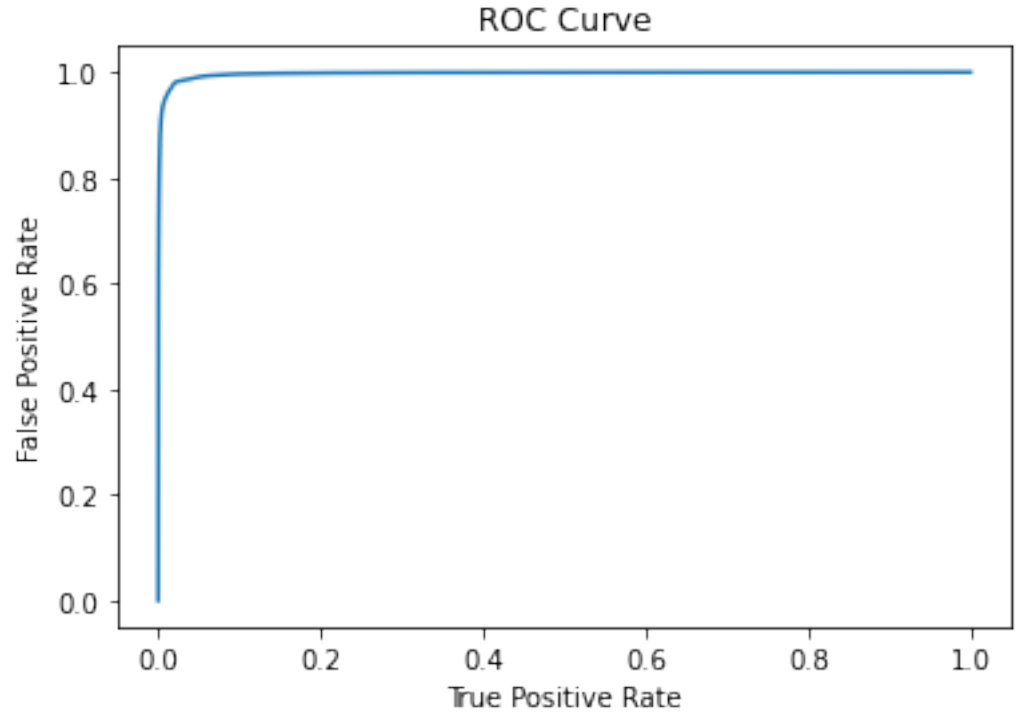
3.1.1 Application

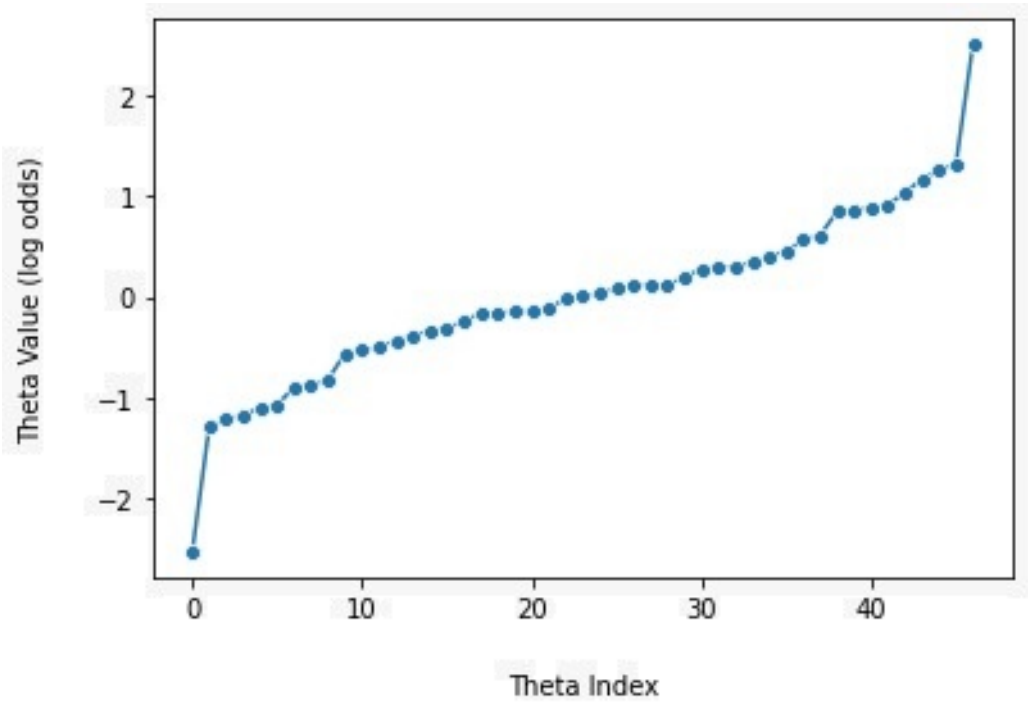
After various trials, we realized the model is able to generalize well without needing any form of regularization: the area under the roc curve during testing phase is just 0.025 less than the one in the training phase, and adding regularization makes this difference only slightly less evident, but significantly decreases performance. We chose the L-BFGS optimizer (by leaving the relative parameter as set by default), as it converges (way) faster than the mini-batch GD.

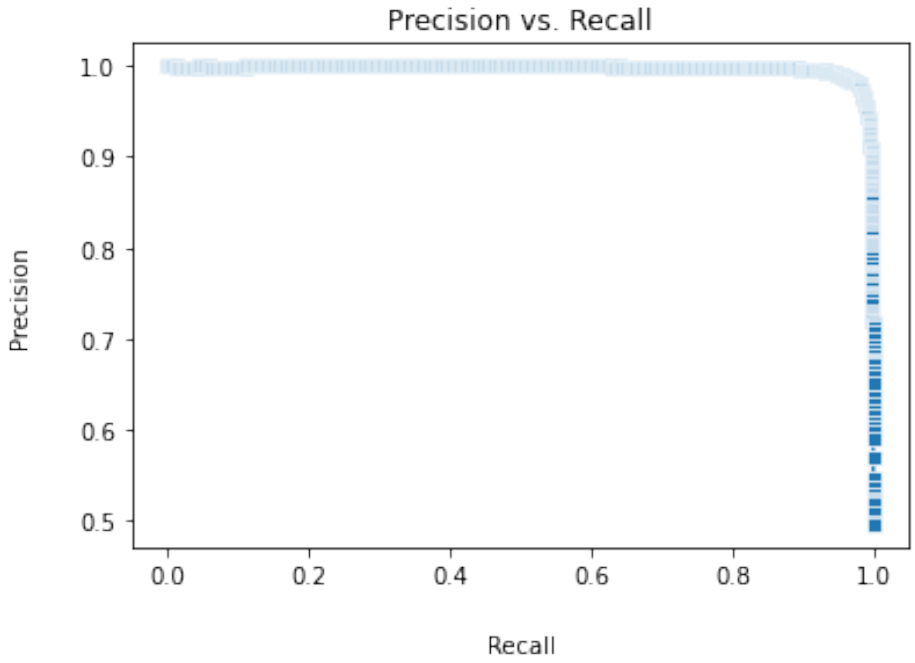
3.1.2 Evaluation

The following plots display the resulting classifier's scores and coefficients, obtained after properly scaling the feature's values: auROC, precision-recall and coefficients. Three graphs are shown below:

- the first one represents the auroc curve, which plots precision vs false positive rate. This is arguably the most used metric for binary classification.
- the second one relates each feature with its odds: we represent the log of the odds in order to have a value in range of $[-\infty, \infty]$
- the third one relates the precision with the recall.
  The precision is defined as $\frac{TP}{TP+FP}$.
  The recall is the ratio is defined as $\frac{TP}{TP+FN}$


ROC Curve
False Positive Rate
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Theta Value (log odds)
Theta Index
2
1
0
−1
−2
0
10
20
30
40



As we can see, we obtained an areaUnderROC in the training set of 0.99 and an anreaUnderROC in the test set of 0.98298.

3.2 Decision tree

The second model we applied for classification is the `decision tree` model.
`Decision tree` is the base model related to the tree-based-models-family. In this model we can do a multiple classification. The model build a decision tree where each node is a choice over a specific feature, in order to deduce in witch class the instance have to go.

3.2.1 Evaluation

To evaluate the model we us the auROC metric. For the decision tree model we obtained an accuracy of 0.96.

3.3 Random forest

Finally, we applied the `random forest` model.
Random forest is a specific usage of decision tree model and an improve of bagging "submodel". In this "submodel" we build multiple decision trees learned on bootstrapped samples of the original training set, but for each individual tree, every time it comes to splitting a node only a random sample of k ¡ n features is considered.

3.3.1 Evaluation

Also here we use the auROC metric. For the random forest model we obtained an accuracy of 0.974.

4 Feature importance analysis

One of the main goals in this project is to gain some insight regarding which features are associated to key objectives for victory in a league of legends game. The more important a feature is, the more convenient it is to pursue it, in order to increase the chances of winning a LoL match.

4.1 Random Forests' Gini

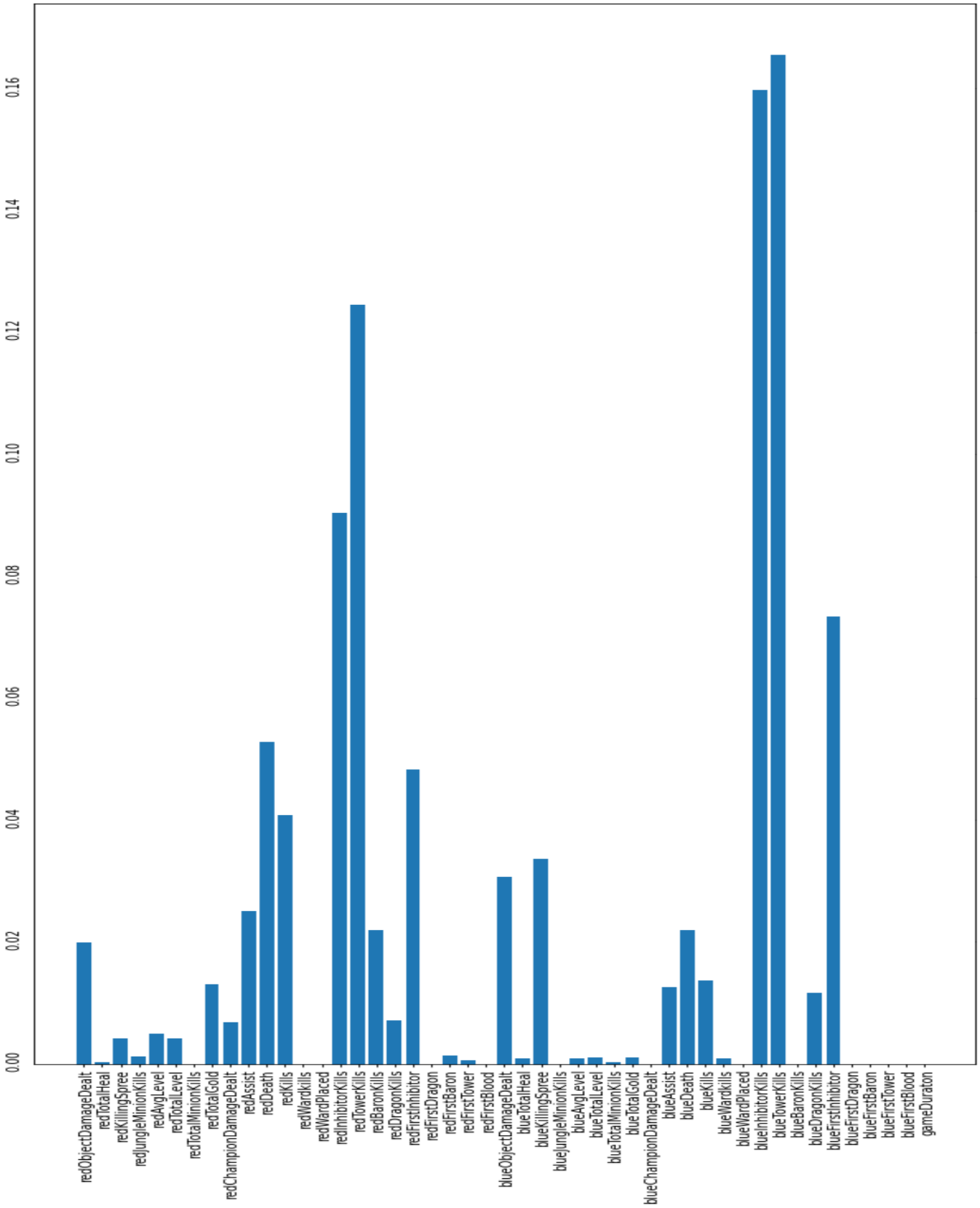
For this reason, in this section we extract and display the feature importance computed by our most performing classifiers: the Logistic Regression Model and the Random Forest model. This function is a generalization of Sci-Kit's learn "Gini", which computes the importance of each feature by:

- Summing the information gain for splitting the samples over that feature
- Normalizing these sums (importance), in order to get them to sum to 1

We could have considered the feature importance of the Decision Tree classifier instead, but chose Random Forest for two reasons:

- It's our best performing model
- Importance for single decision trees can have high variances between different decision trees, while those computed by random forests are generally more accurate.

To calculate it in the context of random forests, let's see how the `RandomForestClassifier` class provides the `SparseVector featureImportances`.
Below we report the outcome of the feature importance applied to our model.


0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
redObjectDamageDealt
redTotalHeal
redKillingSpree
redJungleMinionKills
redAvgLevel
redTotalLevel
redTotalMinionKills
redTotalGold
redChampionDamageDealt
redAssist
redDeath
redKills
redWardkills
redWardPlaced
redInhibitorKills
redTowerKills
redBaronKills
redDragonKills
redFirstInhibitor
redFirstDragon
redFirstBaron
redFirstTower
redFirstBlood
blueObjectDamageDealt
blueTotalHeal
blueKillingSpree
bluejungleMinionKills
blueAvgLevel
blueTotalLevel
blueTotalMinionKills
blueTotalGold
blueChampionDamageDealt
blueAssist
blueDeath
blueKills
blueWardkills
blueWardPlaced
blueInhibitorKills
blueTowerKills
blueBaronKills
blueDragonKills
blueFirstInhibitor
blueFirstDragon
blueFirstBaron
blueFirstTower
blueFirstBlood
gameDuraton

4.1.1 Where to improve?

We can see that the blueTowerKills and blueInhibitorKills columns are the most important to lead the blue team to victory and all this, contextualized to the game and its mechanics, makes sense.
It is also worth pointing out that the reason why these two indicators are considered so important is that, to win the game, it is a necessary condition for a team to demolish at least 3 towers and an inhibitor: however, it's not at all true that killing more towers than your opponent means that you are guaranteed a win; it must be mentioned that there are many towers in the map, with different purposes, as they are spread across the map in positions that have different strategical roles. Other than that, there are some surprising results as well:

- the kill / death / assists features are valued (much) more than those associated to baron / dragon kills. Considering that in the world championships players are often praised by match commentators for sacrificing their character's life in order to steal a baron kill, this is worth mentioning. Moreover,

- One of the apparently least important features that this study takes into account is the "WardsPlaced" one. In the game, wards are small objects (similar to lamps) that allow players to have vision on the map (that is, otherwise, covered in "fog"). In low levels of play, wards are completely ignored. Even though in the last couple of years there has been an increasing in the awareness of their importance, it's still surprising too see how their impact on games is greater than the one given by barons/dragons.

- Last but not least, one of the metrics that media's like to show after every world championship game is how much damage each character dealt to other characters, using this as a measure to grade each player's performance. However, our model points out that the damage dealt to buildings is actually much more important.

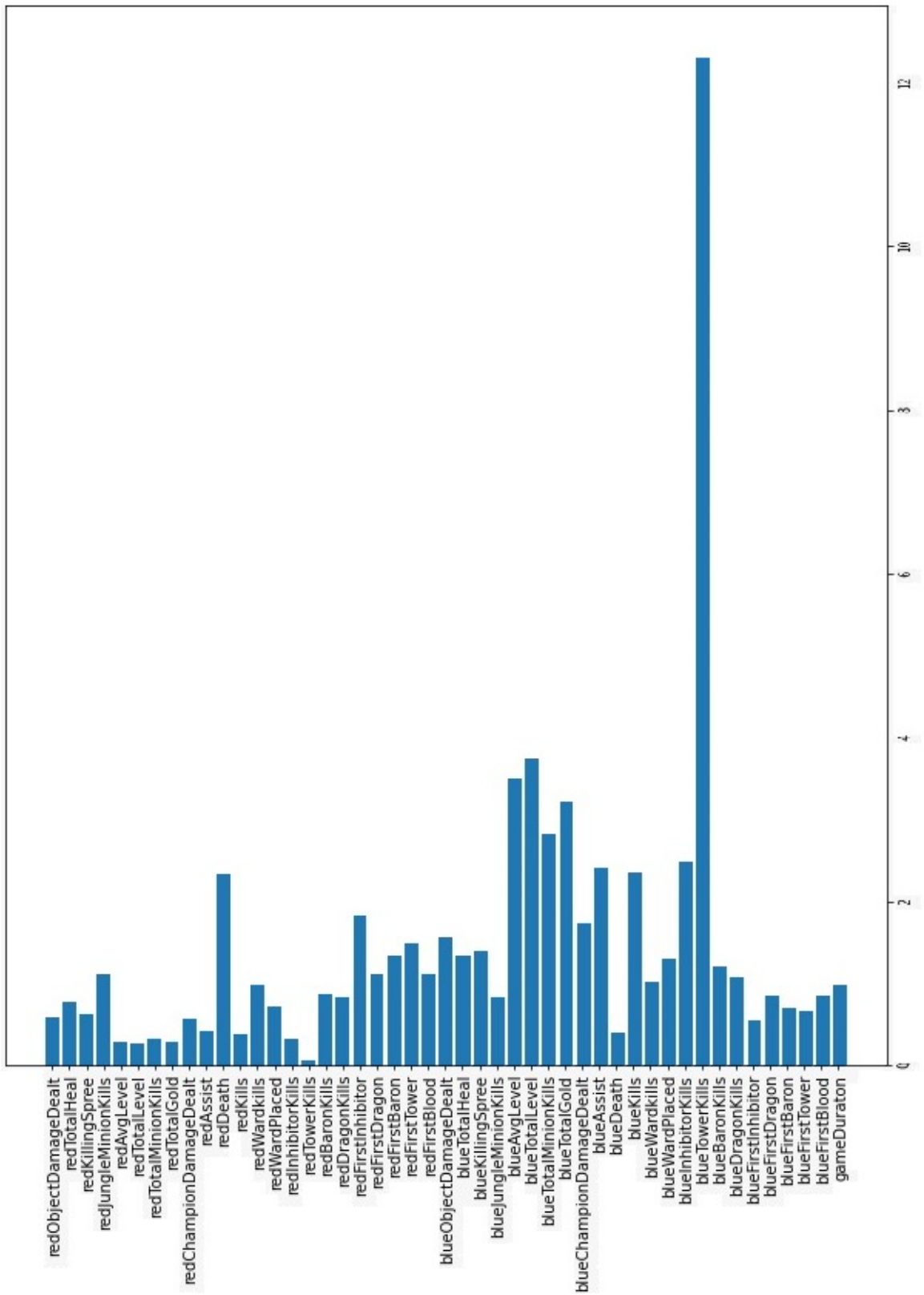
4.2 Odds Ratio

Another possible method to gain some insight on which features have a higher impact on the game involves some analysis on our first model.
To do it, we analyze the `odds ratio`, related to the `logistic regression` study, so on normalized data by `StandardScaler`. In fact, by seeing the odds ratio, we can deduce how important is to improve **by one unit** that feature.

$$\text{odds-ratio} = e^{\theta_i}$$

N.B. So, the odds-ratio does not depend by the value the feature has in that moment.

We need for the θ values.
By calculating all the odds-ratio of the features, we got these results:


redObjectDamageDealt
redTotalHeal
redKillingSpree
redJungleMinionKills
redAvgLevel
redTotalLevel
redTotalMinionKills
redTotalGold
redChampionDamageDealt
redAssist
redDeath
redKills
redWardkills
redWardPlaced
redInhibitorKills
redTowerKills
redBaronKills
redDragonKills
redFirstInhibitor
redFirstDragon
redFirstBaron
redFirstTower
redFirstBlood
blueObjectDamageDealt
blueTotalHeal
blueKillingSpree
bluejungleMinionKills
blueAvgLevel
blueTotalLevel
blueTotalMinionKills
blueTotalGold
blueChampionDamageDealt
blueAssist
blueDeath
blueKills
blueWardkills
blueWardPlaced
blueInhibitorKills
blueTowerKills
blueBaronKills
blueDragonKills
blueFirstInhibitor
blueFirstDragon
blueFirstBaron
blueFirstTower
blueFirstBlood
gameDuraton
0
2
4
6
8
10
12

4.2.1 Where to improve?

According to logistic regression, the feature that (if incremented by 1) has more impact on the game is BlueTowerKills, which is in line with what observed so far. However, this analysis also points out how the character's levels are key to victory as well. Since the target is "blueWins", all features indicating positive achievements for the red team have a low odds ratio.

4.3 Usage proposal

We have shown how the extracted features contribute to victory. Now, we try to suggest new, team-specific strategies to exploit this knowledge. Let's suppose the Sapienza e-sports team is looking for some insight in their playing style, and brings a bunch of games for us to analyze them. As in chess, where strategies are different for black and white, in Lol there are major changes from playing in the red/blue side: let's suppose the team wants to improve their play as Blue. We could compare their game's performances with all the games where blue teams win in our dataset, and see which opportunities the examined teams are missing on. In order to do so, for each objective (feature), we define the feature's priority as:

$$Priority_i = \frac{BlueWinAvg_i - TeamAvg_i^t}{team - avg_i} * importance_i$$

In order to weight each feature's displacement with the importance computed during the training of our Random Forest Model.

5 Conclusions

All professional teams have coaching teams behind them, led by both experts and data analysts. Our objective was to analyze the (high quality) dataset provided by Riot (the company that owns the game) in order to provide this kind of service.
Our classification models successfully separated the data right from the start; with some adjustments, the AUC metric reached nearly 0.99 on the test set. For this reason, we tried to get as much insight as possible on how Random forests and Logistic Regression were achieving such results by examining, respectively, their feature importance vector and the odds ratio, in order to hypothetically transfer this knowledge to teams willing to improve their playstyle. These importance reflected the feature's correlation with the target examined in the data exploration phase in most cases. Some of these correlations are fairly easy to infer knowing the game, but some of them aren't. We tried to propose a playstyle improvement direction by combining each feature's importance with how teams play as the Blue side wrt the average blue-victory games in a home made metric, "priority". Moreover, since the company that owns the game made all game data public, by examining a specific's team playstyle it is easy to obtain information regarding which objectives the team tends to neglect, and to exploit these weaknesses.