LolPredict

Martín Dans

May 25, 2019

```
library(reticulate)
use_python("~/anaconda3/bin/python", required = T)
setwd("~/Projects/Uni/LolPredict")
py_config()
```

```
## python:         /home/omega/anaconda3/bin/python
## libpython:      /home/omega/anaconda3/lib/libpython3.7m.so
## pythonhome:     /home/omega/anaconda3:/home/omega/anaconda3
## version:        3.7.3 (default, Mar 27 2019, 22:11:17)  [GCC 7.3.0]
## numpy:          /home/omega/anaconda3/lib/python3.7/site-packages/numpy
## numpy_version:  1.16.2
##
## NOTE: Python version was forced by use_python function
```

Introduction

In this project I tried to predict the outcome of a League of Legends match just after the champion select phase. This means, we know what champion is going to play which player but we don't anything more since the game time is 0. For doing so, we will use a simple idea: the team with previous higher winrates with the choosen champion will have more chances of winning this match.

Architecture build

For this project I have build 3 modules: api, transforms and model. The api module handles, takes care and caches the data from the official lol api. The transforms module transforms so it has an apropiate format for feeding the training of the models. The models are built with the sklearn python library.

Data manipulation

The game is a team game and the data is provided by each player, but for training the models we need rows of matches. The interesting part here is how we do that, since if we want to keep all the information available there will be too mucht features. The features I choosed are the following ones.

The api only allows for 100 req/minute, to get the winrate of a player you need to query each one of the players so I only could get 550 matches.

Data of a single team in a match:

```
## [
##     {
##         "num_matches": 4,
##         "num_wins": 2,
##         "win_ratio": 0.5,
##         "lane": "TOP",
##         "role": "SOLO"
##     },
##     {
##         "num_matches": 18,
##         "num_wins": 9,
##         "win_ratio": 0.5,
##         "lane": "BOTTOM",
##         "role": "DUO_SUPPORT"
##     },
##     {
##         "num_matches": 19,
##         "num_wins": 11,
##         "win_ratio": 0.5789473684210527,
##         "lane": "BOTTOM",
##         "role": "DUO_CARRY"
##     },
##     {
##         "num_matches": 18,
##         "num_wins": 9,
##         "win_ratio": 0.5,
##         "lane": "MIDDLE",
##         "role": "SOLO"
##     },
##     {
##         "num_matches": 19,
##         "num_wins": 8,
##         "win_ratio": 0.42105263157894735,
##         "lane": "JUNGLE",
##         "role": "NONE"
##     }
## ]
```

Output data for the match:

```
## {
##     "min_winrate0": 0.42105263157894735,
##     "max_winrate0": 0.5789473684210527,
##     "avg_winrate0": 0.5,
##     "num_matches0": 78,
##     "num_wins0": 2,
##     "TOP0": 0.5,
##     "JUNGLE0": 0.42105263157894735,
##     "MIDDLE0": 0.5,
##     "BOTTOM0": 0.5394736842105263,
##     "min_winrate1": 0.4444444444444444,
##     "max_winrate1": 0.625,
##     "avg_winrate1": 0.508625730994152,
##     "num_matches1": 59,
##     "num_wins1": 8,
##     "TOP1": 0.5,
##     "JUNGLE1": 0.4444444444444444,
##     "MIDDLE1": 0.5,
##     "BOTTOM1": 0.5493421052631579,
##     "TOP_DIFF": 0.0,
##     "JUNGLE_DIFF": -0.023391812865497075,
##     "MIDDLE_DIFF": 0.0,
##     "BOTTOM_DIFF": -0.009868421052631526,
##     "wins": false
## }
```

Almost every field explains by itself, notice that the field wins represetns if the “team 0” wins the match. Also, the role diff value is the substraction of players winrates for that role.

The dataset was not complete and I had to deal with some fields.

The first problem is what happens if it’s the first time a player plays that specific champion? I decided to assign a 0.5 chance of winning, but I know that it’s not accurate, it’s known that a player has a lower winrate when starting to play a champion. Also not incompletness but there is another problem when using the percentatges, when there are few matches they aren’t accurate and they introduce a lot of error, so maybe a standarization process could be done.

The second problem is that not all the roles are correctly set. To solve it I checked if I could fill it and in cases where it wasn’t possible I set them to the average of the wrong placed players in the missing roles.

The first model

The simplest model I built has been a random forest with 100 matches using the features mininum winrate, maximum winrate and average winrate for each team. The accuracy get is 45%. As a side note, in the first run of this model I got a 88% accuracity. That value is too high so I double checked the process that I did and I realised that I was cheating with the training. In the steps for getting the data, the history of a player is extracted and it was including the match that I wanted to predict, so the win percetages where there.

```
## (random_forest, basic, 550) - 0.44
```

We have something to work on!

Feature selection

From all the features above we want to see what are the one that are actually relevant. A simple test we can do it’s to calculate the correlation between each feature and what we can to predict. With this we are getting the linear relation between them, of course, we may lose non linear relations and even linear relations of more than one feature but it will give us an idea of what features we can discard.

The values obtained are the following:

```
## Feature: min_winrate0 - 0.09
## Feature: max_winrate0 - 0.03
## Feature: avg_winrate0 - 0.09
## Feature: min_winrate1 - -0.10
## Feature: max_winrate1 - -0.07
## Feature: avg_winrate1 - -0.15
## Feature: num_matches0 - -0.16
## Feature: num_wins0 - 0.00
## Feature: num_matches1 - 0.05
## Feature: num_wins1 - 0.07
## Feature: TOP0 - 0.07
## Feature: JUNGLE0 - 0.11
## Feature: MIDDLE0 - 0.04
## Feature: BOTTOM0 - -0.02
## Feature: TOP1 - -0.12
## Feature: JUNGLE1 - -0.13
## Feature: MIDDLE1 - -0.03
## Feature: BOTTOM1 - -0.06
## Feature: TOP_DIFF - 0.14
## Feature: JUNGLE_DIFF - 0.17
## Feature: MIDDLE_DIFF - 0.05
## Feature: BOTTOM_DIFF - 0.03
```

With that I took as relevant features:

min_winrate, avg_winrate, TOP_DIFF, JUNGLE_DIFF

so we have 3 feature sets, the mentioned above, the basic one with only win percentatges and one with all the calculated features.

Models

Linear regression model

Support vector machine

Support vector machine

Conclusion