EXPLOITING BOOKMAKERS BIASES TO CREATE HIGHLY PROFITABLE AI-BASED BETTING STRATEGIES

Side Personal Project

Sébastien CARARO

2020

" In God we trust. All other things must bring data"

William Edwards Deming (1900 – 1993) – American Statistician

TABLE OF CONTENTS

I - ABSTRACT

I created a pipeline for forecasting NBA games outcome and I developed a couple of tools to exploit this pipeline in order to place bets according to the market odds. The most precise model would have been capable of predicting the winner 69,34 % of the time over the last season, while the bookmaker's favorite won 67,64% of the time (2018/2019 – 1230 games). I also developed similar algorithms towards various sports such as football, tennis, baseball, basketball, rugby, hockey and eSports : the software is able to make predictions on a daily basis and yielded a highly competitive return on investment (ROI) of about 7,42 % on the first 694 bets - over a mean duration of 2h (average duration of a sport match) – between January 1st and March 13rd. Given the fact that the most recent models - presented in this report - are outperforming the previous ones, we can even expect this method to give even higher profit in the next months.

II - INTRODUCTION

This report contains information and techniques concerning the creation of a full pipeline from scratch to help predicting the outcomes of multiple sports and, upon everything, create a highly profitable betting strategy. The following report is furnished along with multiple result metrics, all contained in a directory called *Results.* For any information or question concerning this work, please contact me at sebcararo@hotmail.fr. The code is kept private for the moment, however I published the webscraping python code on my GitHub [1] at www.github.com/Seb943/scrapeOP.

The project contains multiple steps which are developed in this report, I tried as much as possible to make it intelligible towards a non-scientific audience, but also to display a few technical points which were crucial in the development of this project. As I created the NBA program first, I will focus on the NBA algorithm for the main explanations, then I will eventually describe how I adapted these techniques to create the other sports' algorithms afterwards.

III - WHY THIS PROJECT ?

As a 22-years old sports enthusiast, I had always been fascinated by sports analytics and have been focusing on studying the many various statistics in the NBA for years. As I started my gap year before my final Master's year of education, I wanted to involve myself towards a personal project such like this one because then I wouldn't be able to spend so much time when working full-time. I was largely inspired by some of the publications that I present in

the next section, and I was willing to mix my passion for sport with my growing interest for Data Science. Furthermore, such a project was a huge challenge for me, as I had to plan everything from scratch, from the conception of the pipeline to the code implementation, along with the many others related tasks such as data scraping, interface construction, or even computation-time optimization. The difficulty of the project was reinforced by the fact that it is quite a *niche* problem, because even if there are already a number of related works published online, only few of them publish their code, plus many of these works don't yield profitable betting strategies. Taking all these elements into considerations, I was fully enthusiastic to work on this project for a few months.

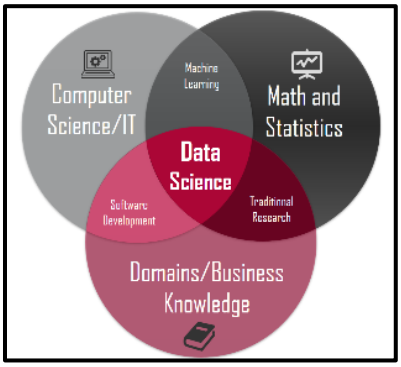


Figure 1 : Data Science's Venn Diagram

IV - STATE-OF-THE-ART CONCERNING PREDICTIVE MACHINE LEARNING ALGORITHMS IN SPORTS

I started this project by spending a few weeks looking for previous works related to Machine Learning applications for sports events predictions. As a modern field with a lot of interest, this subject is already well documented and there are plenty of papers written each year. The main papers which I was influenced by were :

- Beating bookmakers with their own odds – and how the online sports betting market is rigged (Kaunitz et al., 2017, [3])
- Non-Linear classification as a tool for predicting tennis matches (Jakub Hostacny, 2018, [4])
- A statistical approach to sports betting (Anton Altmann, 2004, [5])
- Predicting football results using Machine Learning techniques (Corentin Herbinet, 2018, [6])
- Prediction Markets : Theory and Applications (Michael Edward Ruberry, 2013, [7])
- Favorite-longshot bias in European Football betting market : Differences between popular and non-popular football competitions (Daniël van Raaij, 2019, [8])

I would say that the first two papers in this list inspired me the most, the paper from Kaunitz et al. clearly explains how the use of the odds as an input parameter is of great help for creating profitable betting strategies. Their pipeline - and especially their way of testing the bets in real-life conditions - plus the part concerning Monte-Carlo simulations were very inspiring. The paper from Jakub Hostacny was really impressive in the sense that he worked deeply into the tennis forecasting problem, as he implemented a lot of data as input parameters (in-game data, final result, age and characteristics of players, etc…). Furthermore, the data collection method that he develops in his thesis was very inspiring to me because it involved web scraping, an idea which I then used for creating my own data

collector program. The other papers were also very important in the construction of this project, because they clearly define the challenges concerning sports betting and they describe the many points one should pay attention to in order to build forecasting algorithms. Finally, the paper from Daniël van Raaij helped me a lot to differentiate between leagues, and to build league-specific models : actually, he found out that the dynamics across various football leagues were different and that it was more relevant to build league-specific models instead of creating one single model per sport.

I also documented myself towards GitHub repositories made available by their creators. These projects were really helpful to me, because even though none of them was coded in R, the structure of their pipeline and the explanations represented a huge time gain to me :

- *Bet on Sibyl* application code ([9])
- Beating the bookmakers on tennis matches ([10])
- Soccer betting ([11])

V - OBJECTIVES OF THE PROJECT AND PIPELINE PRESENTATION

(A) PROBLEM OVERVIEW AND REFORMULATION

The National Basketball Association (NBA) is the American basketball league, composed of 30 teams – including the only canadian team the Toronto Raptors – who compete all season long to win the championship. The regular season is composed of 82 games per team, which add up to 1230 games in total. The regular season starts around late October and finishes around the 15th of april, therefore there are multiple games each night during this period (around 50 games per week). The 30 teams are divided into 2 conferences, each of them being composed of of three divisions that each contains 5 teams. Finally, the best 8 teams from each conference participate in the playoffs, which first designates the conferences winner, before the Finals series which designates the NBA champion. Each round of the playoffs is designed as a best-of-7 game series.

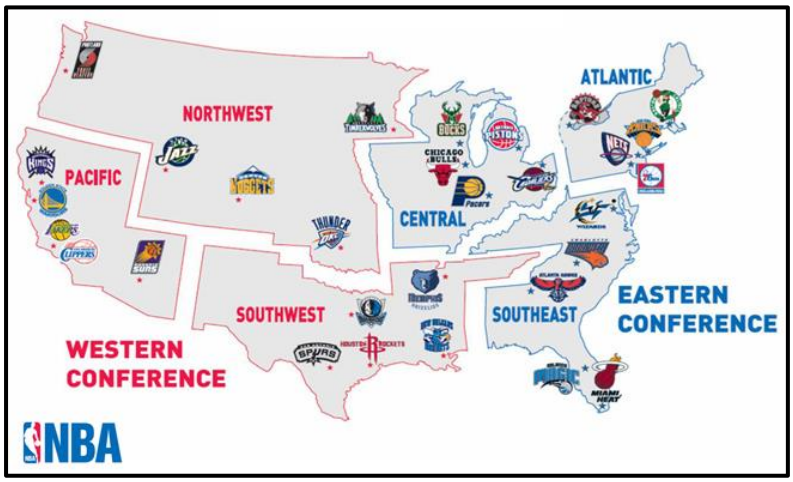


Figure 2 : NBA organisation into divisions and conferences [12]

The main idea was, from the raw historical data, to create a set of predictive variables $\{Vi\}_{1\leq i\leq n}$and a linear sequence of mathematical functions $\{Ti\}_{1\leq i\leq m}$ such that applying the convolution of the mathematical functions to the initial raw variables would result in an accurate estimation of victories for a given NBA game :

$$\left(\prod_{i=1}^{m} T_i\right)(V_1, \dots, V_n) = \left(\widehat{P_{model}}(winner = hometeam), \widehat{P_{model}}(winner = awayteam)\right)$$

$$\text{With } \widehat{P_{model}}(winner = hometeam) + \widehat{P_{model}}(winner = awayteam) = 1$$

A *bookmaker*, such as the one I studied during this work, is a company whose function is to take bets from gamblers. Bookmakers propose various betting markets which depend on the bookmaker itself. For the NBA the most usual markets are the *moneyline* (predicting the winning team), the *handicap* (predicting margin of victory, spread) and the *total* (predicting if the number of points is below or above a fixed threshold). This said, when a gambler bets on an event, he stakes an amount S of real currency onto the realization of the event, at a fixed odd C : if the event happens, the gambler takes back a total earnings of $C * S$, which represents a net profit of $(C - 1) * S$, and if the event doesn't happen then he just lost his stake S.

In sports betting, the *margin* designates the percentage of the stakes that the bookmaker keeps for himself. Actually, the odds are theoretically set to create a game with a negative expectancy for players, such that the bookmaker can make profit on the long-run. For example, if the match Federer vs Nadal is a 50-50 game, odds should be 2 and 2 in a null expectancy game, however bookmakers would then not be able to make profit, hence they set the odds to something around 1.9 and 1.9 so that for each euro staked, players receive on average 1.9 / 2 = 0.95 € back : the bookmaker wins 0.05 € on average (5% margin).

(B) STARTING FRAMEWORK

The initial objectives were to create an ultra precise model, which should have been able to predict accurately 80% of the games. However, I quickly realized my expectations were way too high and out of my range – if even possible. After having documented myself regarding to the subject, I realized that many of the similar projects only settled for creating the most accurate model but most of the time they couldn't exploit their model to make huge profit in sports betting because the bookmakers margin still compensated for the good accuracy of their models. I then tried to figure out how to counter this, and started creating my own models, then I thought about a way to identify biases in the bookmakers estimation.

In order to fullfill the many objectives, I created a whole pipeline from scratch, which can be represented in the following way :

Figure 3 : Descriptive pipeline of the project

VI - DATA SOURCES AVAILABLE

The only raw data that I needed to start from was an historical dataset which contained the result of each game, along with the teams names and the closing odds proposed by the bookmaker Bet365 [13]. Closing odds are the odds available just before the beginning of the match and it is very important to have this particular data because odds can fluctuate a lot between the opening and the closing time, mainly to cope with players injuries and with gamblers staking one or another team.

In order to create this dataset, I chose to create a web scraping algorithm in Python [14] to collect historical closing odds displayed on the website www.oddsportal.com , since season 2009-2010. The program was coded in order to collect the basic information for each game, *i.e.* the following variables :

- Date of the match
- Teams id
- Final score
- Money line odds proposed by the bookmaker bet365.com

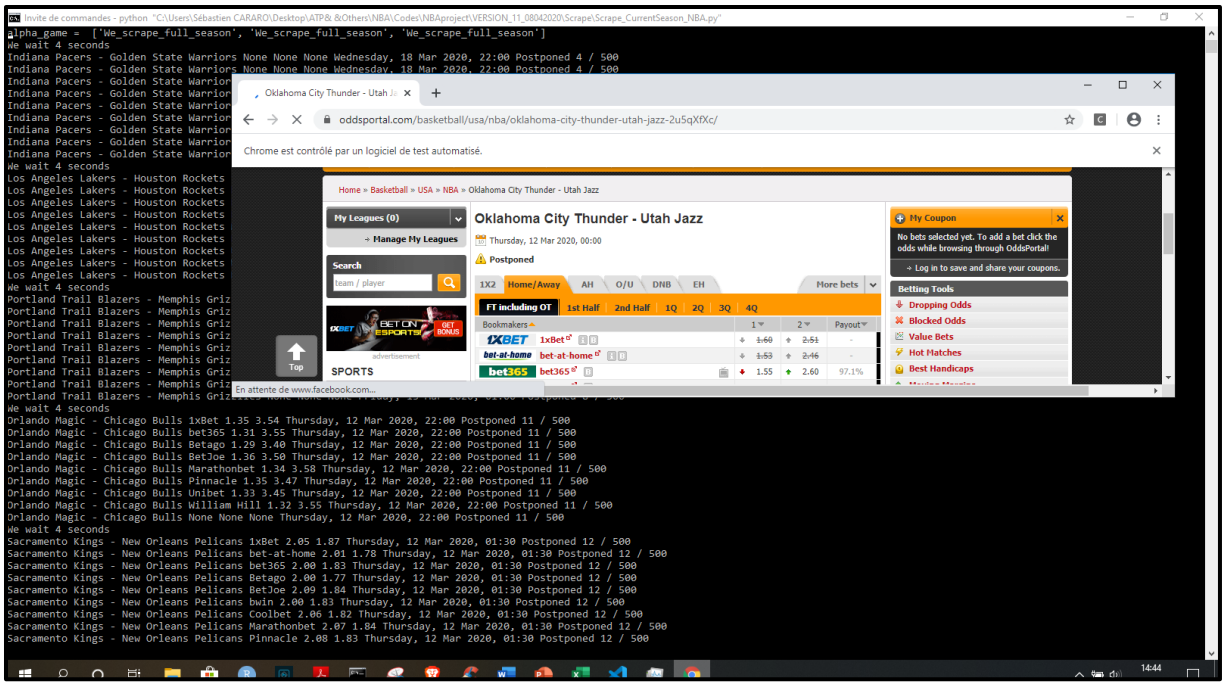


Figure 4 : Screenshot of the web scraping algorithm running

Home_id	Away_id	B365_OT_H	B365_OT_A	Score_home	Score_away	Seasons	Date	x	MatchId
Cleveland Cavaliers	Boston Celtics	1.5	2.7	89	95	2009/2010	27/10/2009	1	1
Dallas Mavericks	Washington Wizards	1.25	4.2	91	102	2009/2010	27/10/2009	1	2
Portland Trail Blazers	Houston Rockets	1.18	5.25	96	87	2009/2010	28/10/2009	1	3
Los Angeles Lakers	Los Angeles Clippers	1.15	5.5	99	92	2009/2010	28/10/2009	1	4
Toronto Raptors	Cleveland Cavaliers	3	1.41	101	91	2009/2010	28/10/2009	1	5
Orlando Magic	Philadelphia 76ers	1.22	4.5	120	106	2009/2010	28/10/2009	1	6

Figure 5 : Scraped table overview

I decided not to use in-game statistics neither player injuries information, mostly because I didn't find any reliable source to access this data on a daily basis. Furthermore, creating a player-based predictive model might have been too much of a charge for my computer and I, given the fact that the study would then have been tremendously heavier. The final dataset contains 12 054 games, distributed in the following way :

```
> a <- read.csv("C:\\Users\\Sébastien CARARO\\Desktop\\ATP& &Others\\NBA\\Codes\\NBAproject\\VERSION_11_08042020\\NBA_clean.csv")
> summary(a$seasons)
2009/2010 2010/2011 2011/2012 2012/2013 2013/2014 2014/2015 2015/2016 2016/2017 2017/2018 2018/2019
     1230      1225       990      1229      1230      1230      1230      1230      1230      1230
```

Figure 6 : Final dataset repartition by seasons

VII - SOFTWARES USED

I coded everything from scratch in R 3.6.0, a sample of the librairies that I used is :

- *ggplot2* for plotting results
- *MASS* for Machine Learning models and tools
- *e1071* for the Support Vector Machines models
- *nnet* for the averaged Neural Networks
- *dplyr* for better code clarity
- *taskscheduleR* to schedule R tasks from R interface (automation process – *Section XVII (b)*)

Considering the lack of publicly-available tools - especially in R - I had to create the vast majority of the functions by myself. The web scraping program was developed in Python 3.6.2 using the Pyzo environment, and a few usual and web scraping librairies were used such as *selenium*, *pandas*, *numpy* and *urllib*. The webscraping process is later used to automatically scrape odds right before the game, in order not to have to do it by hand.

VIII - PREPROCESSING STEPS

(A) EXPLORATORY ANALYSIS AND DATA CLEANING

Once I had scraped the raw source dataset, I first proceeded to an exploratory analysis in order to be familiar with the dataset and anticipate the next steps. I found out that there were a couple variations in the data, and that I had to proceed to data cleaning before any further step. First I excluded the playoffs games from the study, given the fact that for the moment I solely focused on the regular season, plus the dynamics during playoffs is certainly different from the regular season dynamics. Apart from this, a few games had odds missing : I decided to replace the missing value by the median value of the home/away odd in the dataset (9 matchs over the whole dataset).

After inspecting the amount of games per season, I noticed that the 2011-2012 season contained only 990 games : this is due to the fact this season was a *lockout* season, *i.e.* there was a players strike during this season which resulted in a number of cancelled games. This season was then only composed of 66 games per team.

(B) COMPUTE PREDICTIVE VARIABLES

The next step was to create predictive variables in order to fit the differents Machine Learning models on. I coded a total number of 133 features for this program, which are noted down in the appendix n°1 at the end of this report.

I chose to encode common variables used in sport analytics such as the offensive & defensive performance (number of points scored/against), the season record, the number of rest days since last game, along with multiple other measures. Furthermore, I also used odds-derived metrics as predictive variables : I computed the Return-On-Investment (*ROI*)

for each team, when we bet them winning or losing, when they're playing home or away, the average odd when winning/losing, the biggest odd passed this season, etc… These metrics give precious information about teams strength evaluation by bookmakers, as we can then deduce whether a team is overperforming this season, or in the other case if the team is underperforming according to the bookmakers' estimation.

Advancing towards my study, I discovered the existence of a rating system for sports teams called Elo [15] which is known for being a good estimation of a team power. At first I decided to directly use the historical Elo rating dataset proposed by the website fivethirtyeight [16], which is one of the most broadly followed website concerning sports predictions and analytics-based analysis. The open-source ELO table can be found on their github page [17] and is updated on a daily basis, which fitted my daily utilization of the NBA algorithm.

However, I later decided to compute the variable by myself, in order to being able to generalize the calculation on the future leagues and sports (which don't have similar publicly available datasets).

I then implemented this metrics as a predictive variable, following the downwritten recursive simplified formula :

$$Elo_{teamA_{match1}} = 1500$$

- ✓ <u>If teamA won match n°i against team B:</u>

$$Elo_{teamA_{matchi+1}} = Elo_{teamA_{matchi}} + \frac{K}{1 - 10^{\frac{Elo_{teamB_{matchi}} - Elo_{teamA_{matchi}}}{400}}}$$

- ✓ <u>Else if teamA lost the match n°I gainst team B:</u>

$$Elo_{teamA_{matchi+1}} = Elo_{teamA_{matchi}} - \frac{K}{1 - 10^{\frac{Elo_{teamA_{matchi}} - Elo_{teamB_{matchi}}}{400}}}$$

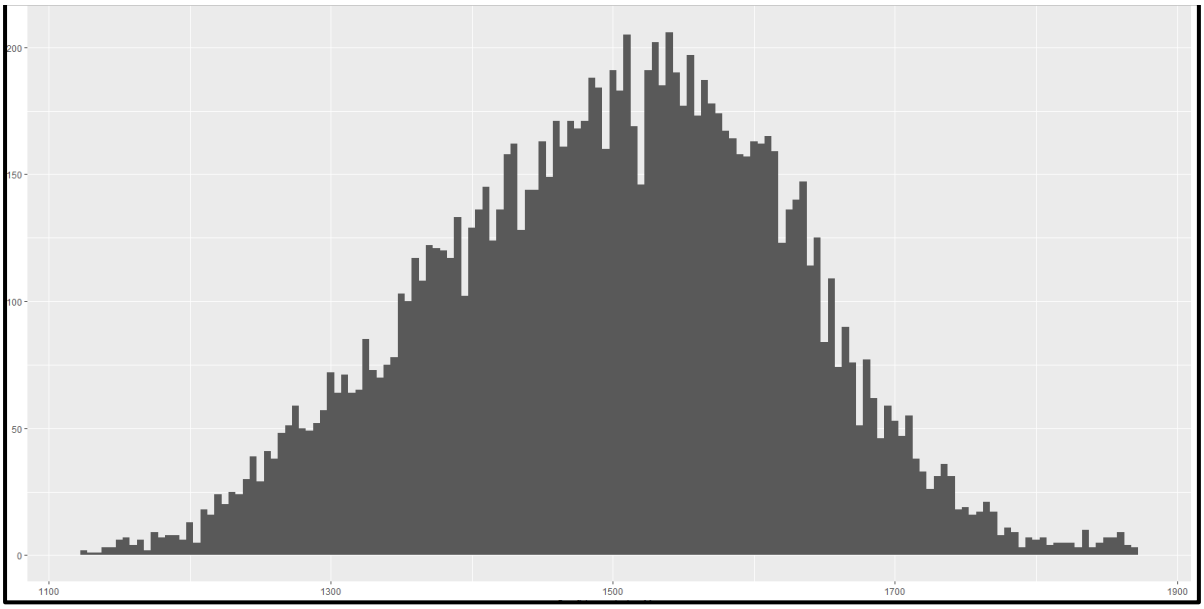
With K being a dynamics constant which I set to 32, and

$$\frac{1}{1 - 10^{\frac{Elo_{teamA_{matchi}} - Elo_{teamB_{matchi}}}{400}}}$$

being the probability that teamB wins the game, according to this Elo ranking system.

The Elo rating is a zero-sum system, *i.e.* the total amount of ELO points stays constant over the season as a consequence of the mathematical formulation. As one can expect from the formula, this rating puts the emphasis on strength difference for points earnings : an unexpected win from a weaker team against a supposedly stronger one will make the team win more points than a win against a supposedly weaker opponent. ([18]).

In the end, the ELO repartion histogram has the following bell curve shape – centered in 1500 - over the course of the 10 years period :



<u>Figure 7 :</u> ELO repartition in the NBA dataset

For each match, I computed the statistics concerning all matches this season, but also concerning home games for the home team, away games for the away team, the last 5 and last 10 games, the head-to-head record, along with some other composite variables, considering multiple combinations of the listed attributes. I also computed some percentages associated with the stats, *e.g.* once I had computed the number of wins and the number of loss this season, I could easily compute the percentage of victory this season.

I also computed statistics who took into account the previous seasons, so that the machine learning models would also learn from long-term dynamics to make predictions aswell.

IX - PCA

(A) VARIABLES OVERVIEW

After having correctly computed and checked the 133 predictive variables, we can first have a look at the correlation matrices, which we can compute according to different metrics. I chose to represent the correlation matrices with Spearman and Pearson methods, which I represent there down – the matrix are very similar, as expected :

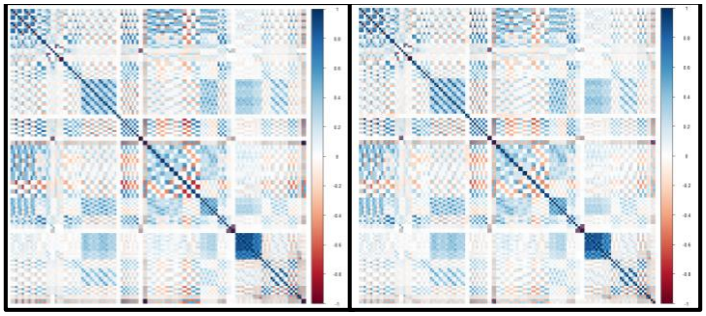
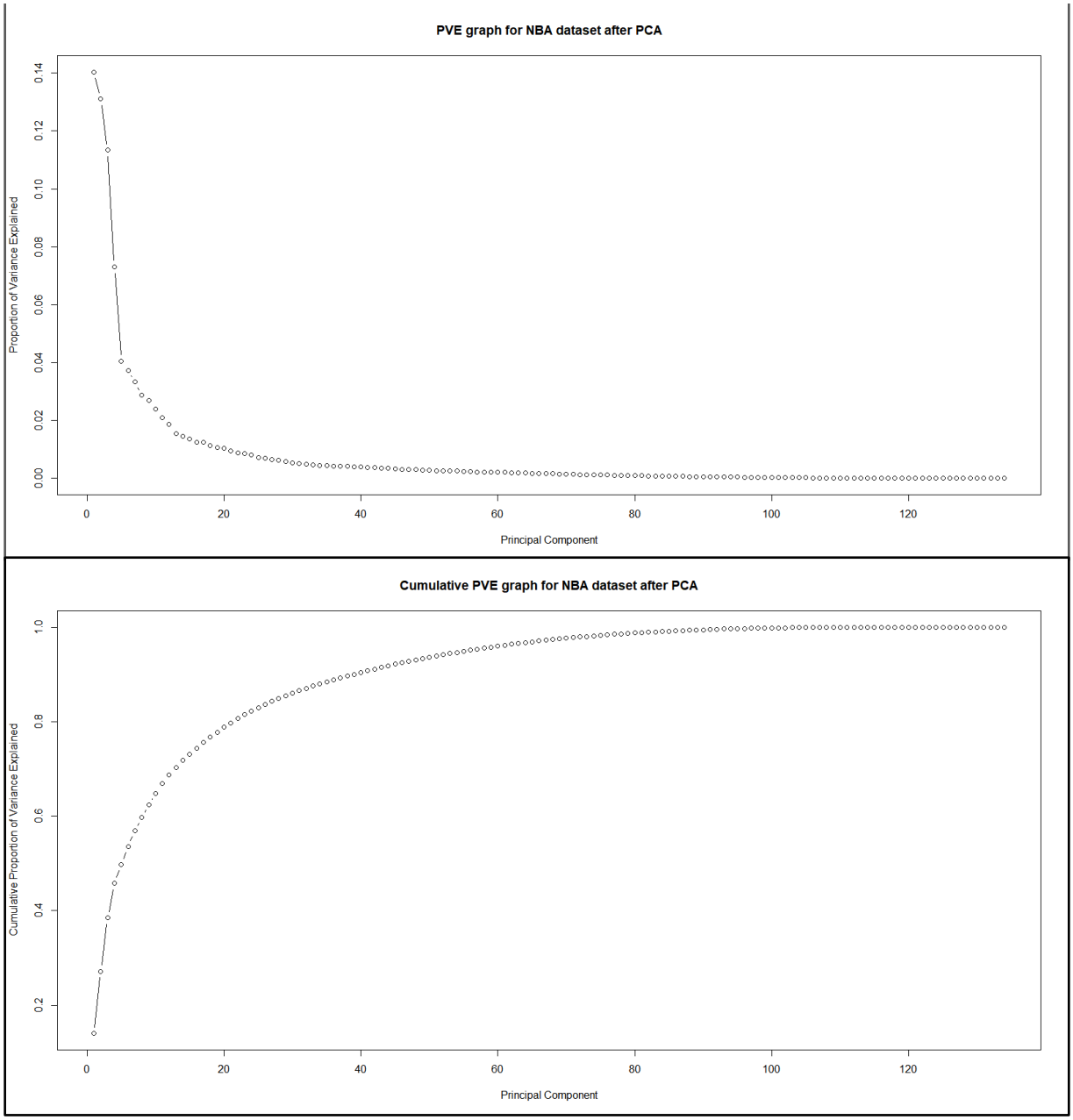


Figure 8 : Correlation matrices between predictive variables

- ✓ Comments : As a general statement, we can notice that a lot of the variables are totally uncorrelated, which might be a good sign as this would mean that we can capture different statistical patterns in the dataset. This could also mean that we have noise in the dataset, however because of their definition I would think that most of the variables are related to the outcome. The next section, Principal Component Analysis, is a great tool for filtering out the possible noisy information in the dataset.

(B) APPLYING PCA

After having computed all variables and before fitting my models, I decided to apply Principal Component Analysis through the Eigen Vectors Decomposition (EVD PCA - [19]). This widely used technique creates a new set of components from the initial 133 variables, in order to create a set of orthogonal components who concentrate the variance in the first components. This is a very powerful technique to enhance the Signal-To-Noise ratio (SNR – [20]) because we can then select the first components in order to capture most of the information. It can typically be a huge benefit to our study because many variables are nevertheless correlated, hence we will remove the repeated correlation with this step – correlated variables can worsen the accuracy of Machine Learning models by augmenting the variance. I chose to scale and normalize the original variables before applying PCA.



<u>Figures 9 & 10 :</u> PVE graph and Cumulative PVE graph

- ✓ <u>Comments :</u> We can identify a clear "elbow" pattern in the Cumulative Proportion of Variance Explained (Cumulative PVE) graph, which might indicate that the majority of the information would be contained in the first 15 Principal (*i.e.* that there is not so much variance in the last PCs). However we will perform cross validation when building the models (with *nPCs = 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 80*), in order to compare the results we obtain with differents sets of PCs. The final goal will be to determine the optimal number of PCs such that the bias/variance tradeoff is as good as possible in terms of accuracy.

(C) SCALING THE PRINCIPAL COMPONENTS

In order to gain time during the Machine Learning models training, I decided to scale the output of the PCA, such that the final variables are all centered to 0 and scaled to have *standard deviation* = 1. Such transformation is usually recognized as being a gain of training time, plus it may also put the emphasis on using not only the very first PCs to train the models. The models would then adjust the relative PCs importance during the training process.

X - FITTING MODELS : THE DIFFERENT TYPES OF MODELS, PARAMETERS

I chose to fit multiple types of models over the newly created Principal Components, and to fit them with different hyperparameters settings. The four types of models that I used were Random Forests [21], Support Vector Machines [22], averaged Neural Networks [23], and Discriminant Analysis (Linear/Quadratic) [24]. These models were trained to predict the outcome of the game in a binary way : either predict the home team ('H') or the away team ('A') as winner. I also fitted the models to output estimated probabilities of victory. All of the aforementioned models are supervised models, *i.e.* we train the models to predict Y given x, as opposed to unsupervised methods which are only trained to cluster the observations, using the predictive variables x. All models were trained using "*accuracy*" as a metric, even though various criterions such as *gini*, *entropy* or *Area-Under-the-Curve (AUC)* could have been used in this case of binary classification. Such new implementations could be the subject of future work (*hyperparameters tuning*).

I also performed two types of study : the first one considering the downsampled training set, *i.e.* removing training samples from the majority class in order to train the models with a balanced dataset, and a second study where I used the whole training set to train the models. The class repartition on each study is presented there down :

Training set :	**Downsampled**	**Normal Sampled**
Home Victories	4445	6379
Away Victories	4445	4445

Figure 11 : Training sets class repartition

Downsampling is a widely used method to cope with imbalanced training sets, which can cause some models to overpredict the majority class in the case of future predictions. Some models such as Neural Networks and SVM are known to cope well with such cases, but for instance Random Forests can diverge fastly when the offset is too wide. In our case the raw training set is composed of 58.93 % of home victories, which justifies the coherence of such study. On the other hand, downsampling forces us to let aside 1934 training observations, reducing the size of the training set and hence causing us to lose information. We will see in the *section XI* which set of models performed better.

Other methods can be used to deal with imbalanced datasets, such as upsampling and upweighting ([25]).

(A) RANDOM FORESTS

Random Forests is a powerful tree-based algorithm, that enhances the original classification tree thanks to the use of heuristics, which allow the training process to be more robust to new observations : deeply grown classification trees are known for not being very robust when confronting to new observations, this phenomenon is known as overfitting [26] and is a common problem in the Machine Learning field. Instead of simply building a lot of trees and averaging their predictions, Random Forests use the following heuristic : at each node, we select a number *p* of variables among the *n* original variables (usually $p = \sqrt{n}$), and we seek to find the best criterion among these criteria. By doing this, the classification trees that we build are way less correlated, hence when we average their predictions (*i.e.* majority voting in the case of classification) the system is more robust to new observations [27]. This even allows us to build deep trees without risking any type of overfitting.

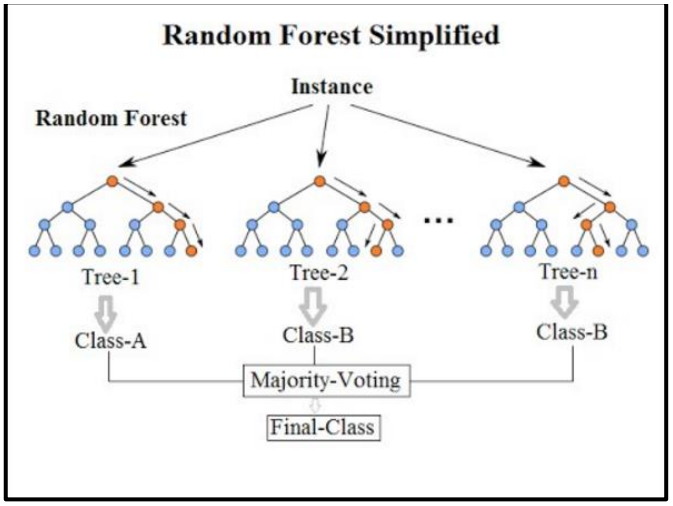


Figure 12 : Random Forest construction illustrated ([28])

The models were fitted using 10-fold cross validation, and I used a total of 500 trees to construct the model.

```
111 fitControl <- trainControl(method = "repeatedcv",
112                            number = 10,
113                            repeats = 10,
114                            ## Estimate class probabilities
115                            classProbs = TRUE)
116
117 rf_fit <- randomForest(Winner ~ .,
118                        family = "binomial",
119                        data = train.data,
120                        trControl = fitControl,
121                        ntree = 500,
122                        metric = "Accuracy")
123
```

Figure 13 : Hyperparameters for Random Forest training in R

(B) SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) is a supervised machine learning algorithm that, given a set of observations described by a total of m predictive variables in the training set , seeks for

the best hyperplane H in the space R^m that maximizes the separation between the two classes. Alternatively, the R^mdecision space can also be transformed such that the decision hyperplane can have another shape (radial, polynomial kernels, personalized functions).

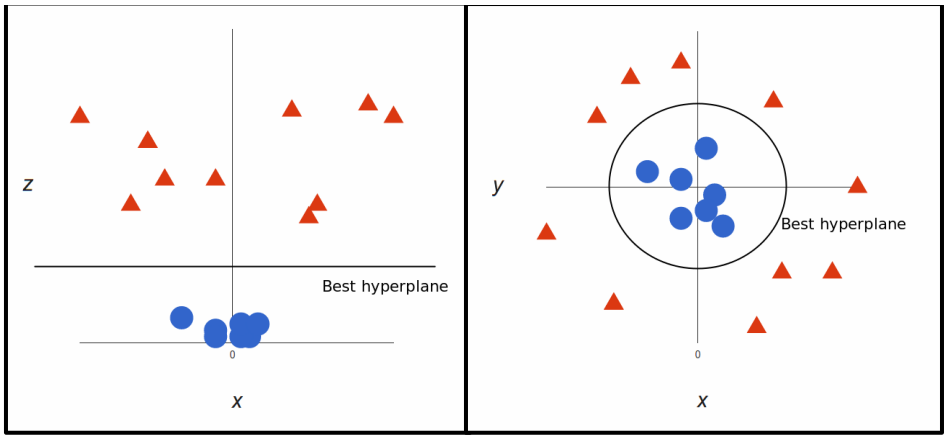


Figure 14 : SVM decision process illustrated ([29])

```
svm_fit <- svm(Winner ~ .,
               type = 'C-classification',
               kernel = 'linear',
               data = train.data,
               cross = 10, # 10-fold CV
               probability = TRUE,
               metric = "Accuracy")
```

Figure 15 : Hyperparameters for linear SVM training in R

(C) AVERAGED NEURAL NETWORKS

Neural Networks are considered a state-of-the-art technique for building powerful models that can cope with a huge number of variables and observations. Averaged Neural Networks is a construction process that, as during the Random Forest building process, builds and trains a large number of Neural Networks with different initial random number seeds and then proceeds to majority voting in order to reduce the variance related to building a single architecture.

Thanks to the *nnet* [30] package implemented in R, I was able to train feedforward neural networks without having to get too deep into the architecture. This packages offers a very simple way to build and call such models, hence I decided to fit multiple sizes of them and integrates the various models into my set of models.

The models were built using 10-fold cross validation :

```
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10,
                           ## Estimate class probabilities
                           classProbs = TRUE)

avNNET_fit <- avNNet(Winner ~ .,
                     repeats = 20,
                     data = train.data,
                     bag = TRUE,
                     trControl = fitControl,
                     size = 1,
                     maxit = 10000)
```

Figure 16 : Hyperparameters for avNNet size 1 training in R

(D) LDA AND QDA

Linear Discriminant Analysis (LDA – [31]) is a Machine Learning model that is often compared to Principal Component Analysis, in the sense that the model seeks for the best linear combination of the predictive variables in order to differenciate between the classes. However, LDA is a supervised technique, which allows it to be way more precise in the case of binary classification.

Quadratic Discriminant Analysis (QDA – [32]) is based on the same principle as LDA, nonetheless the discriminative criterion is different because QDA models assume a quadratic function as boundary between classes :

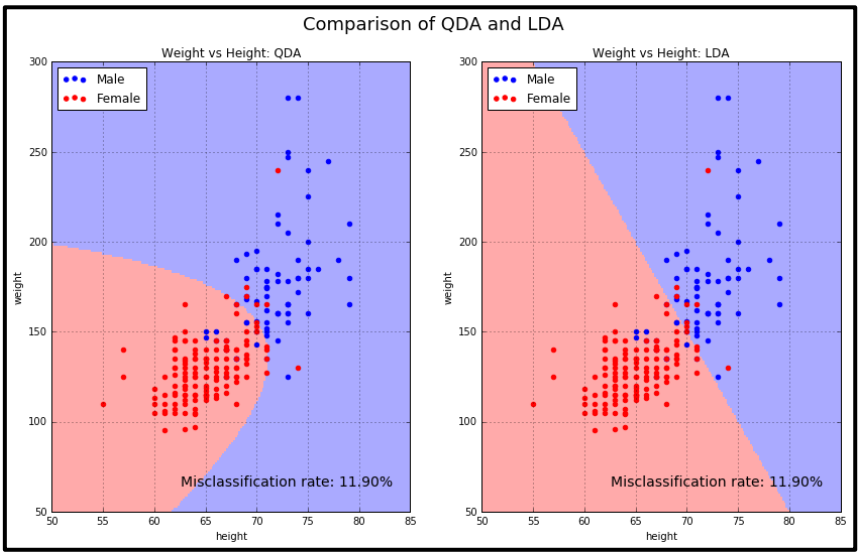


Figure 17 : QDA and LDA illustration ([33])

I fitted the two models using 10-fold cross validation :

```
# (d) Fourth model = LDA
lda_fit <- train (Winner ~ .,
                  data = train.data ,
                  method = 'lda' ,
                  trControl = fitControl ,
                  metric =  'Accuracy' )
```

Figure 18 : Hyperparameters for LDA in R

XI - COMPARISON BETWEEN MODELS

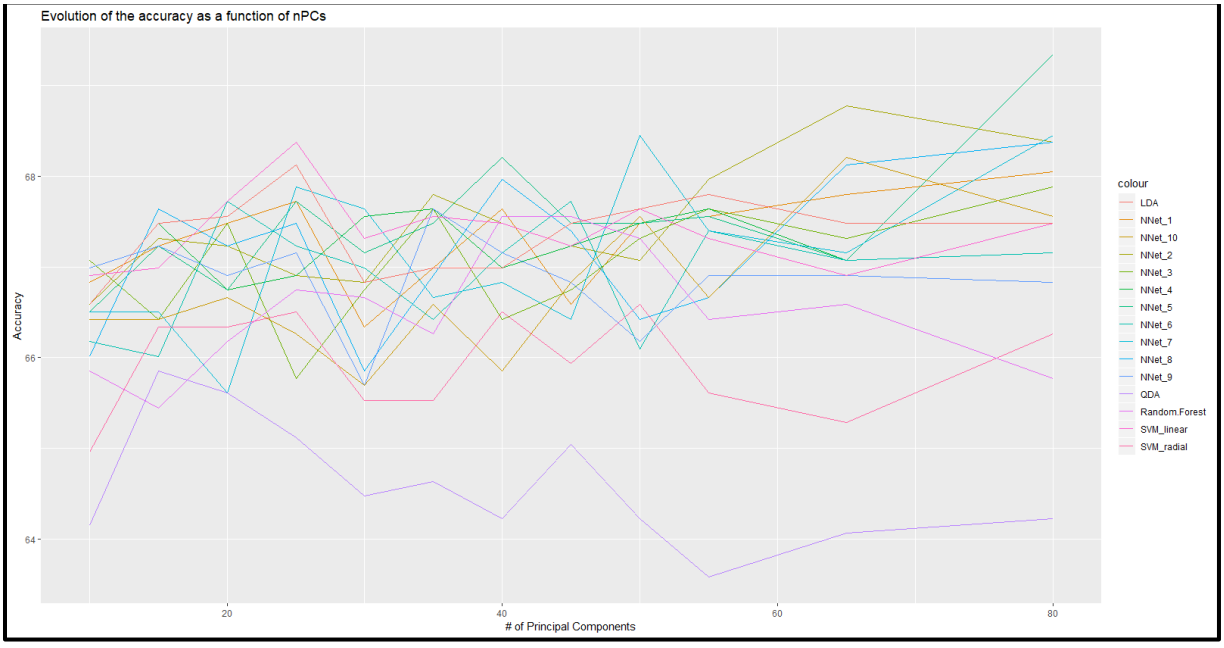
The test set for all models was composed of the very last season (2018/2019 season), hence I kept 9 seasons for training the models. The results, presented hereby, show a slight domination of the avNNet models concerning forecasting accuracy. The best models by study are highlighted in dark green and the best models by type of models are highlighted in light green (*i.e.* in the first study for the model QDA the best model was with 20 PCs and the best models overall were nnet sizes 4 & 5 with 80 PCs).

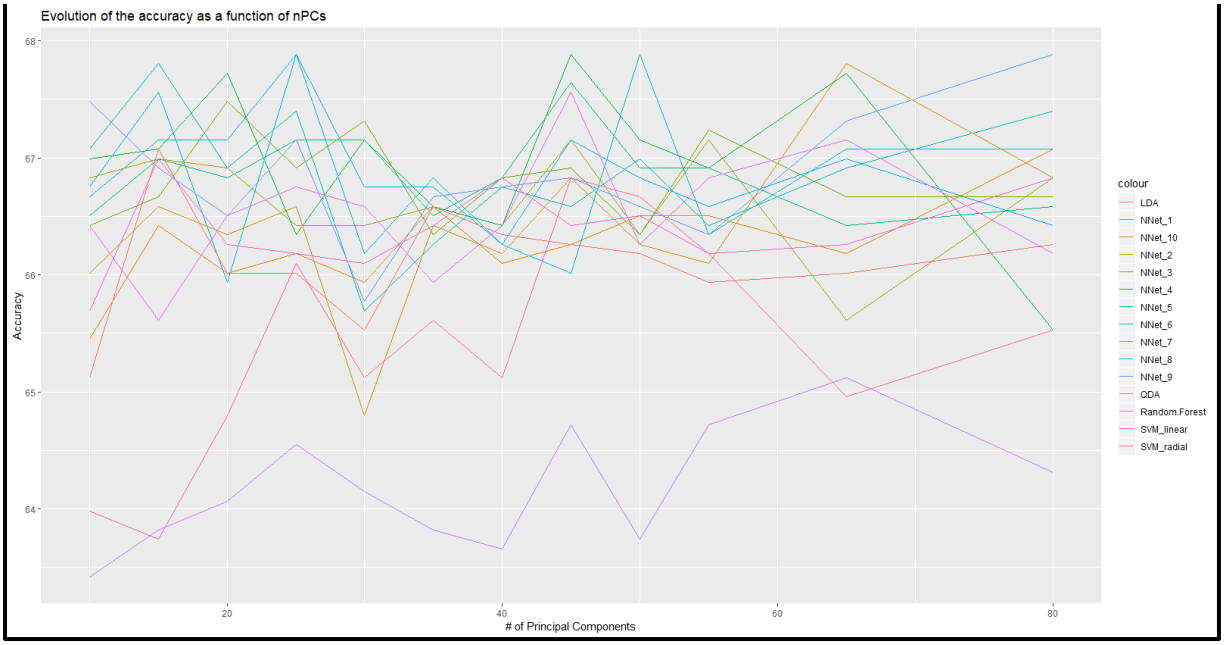
NNet_10	NNet_9	NNet_8	NNet_7	NNet_6	NNet_5	NNet_4	NNet_3	NNet_2	NNet_1	SVM_linear	SVM_radial	andom.Fores	LDA	QDA	nPCs
66.42276422	66.99186991	66.01626016	66.50406504	66.17886178	66.50406504	66.58536585	67.07317073	66.58536585	66.82926829	66.91056910	64.95934959	65.85365853	66.58536585	64.14634146	10
66.42276422	67.23577235	67.64227642	66.50406504	66.01626016	67.23577235	67.47967479	66.42276422	67.31707317	67.23577235	66.99186991	66.34146341	65.44715447	67.47967479	65.85365853	15
66.66666666	66.91056910	67.23577235	65.60975609	67.72357723	66.74796747	66.74796747	67.47967479	67.23577235	67.47967479	67.72357723	66.34146341	66.17886178	67.56097560	65.60975609	20
66.26016260	67.15447154	67.47967479	67.88617886	67.23577235	67.72357723	66.91056910	65.77235772	66.91056910	67.72357723	68.37398373	66.50406504	66.74796747	68.13008130	65.12195121	25
65.69105691	65.69105691	65.85365853	67.64227642	66.99186991	67.15447154	67.56097560	66.74796747	66.82926829	66.34146341	67.31707317	65.52845528	66.66666666	66.82926829	64.47154471	30
66.58536585	67.64227642	66.91056910	66.66666666	66.42276422	67.47967479	67.64227642	67.64227642	67.80487804	66.99186991	67.56097560	65.52845528	66.26016260	66.99186991	64.63414634	35
65.85365853	67.15447154	67.96747967	66.82926829	67.15447154	68.21138211	66.99186991	66.42276422	67.47967479	67.64227642	67.47967479	66.50406504	67.56097560	66.99186991	64.22764227	40
66.82926829	66.82926829	67.39837398	66.42276422	67.72357723	67.47967479	67.23577235	66.74796747	67.23577235	66.58536585	67.23577235	65.93495934	67.56097560	67.47967479	65.04065040	45
67.56097560	66.17886178	66.42276422	68.45528455	66.09756097	67.47967479	67.47967479	67.31707317	67.07317073	67.47967479	67.64227642	66.58536585	67.31707317	67.64227642	64.22764227	50
66.66666666	66.91056910	66.66666666	67.39837398	67.39837398	67.56097560	67.64227642	67.64227642	67.96747967	67.56097560	67.31707317	65.60975609	66.42276422	67.80487804	63.57723577	55
68.21138211	66.91056910	68.13008130	67.15447154	67.07317073	67.07317073	67.07317073	67.31707317	68.78048780	67.80487804	66.91056910	65.28455284	66.58536585	67.47967479	64.06504065	65
67.56097560	66.82926829	68.37398373	68.45528455	67.15447154	69.34959349	69.34959349	67.88617886	68.37398373	68.04878048	67.47967479	66.26016260	65.77235772	67.47967479	64.22764227	80

NNet_10	NNet_9	NNet_8	NNet_7	NNet_6	NNet_5	NNet_4	NNet_3	NNet_2	NNet_1	SVM_linear	SVM_radial	andom.Fores	LDA	QDA	nPCs
66.01626016	67.47967479	66.66666666	66.74796747	67.07317073	66.50406504	66.99186991	66.42276422	66.82926829	65.44715447	65.69105691	63.98373983	66.42276422	65.12195121	63.41463414	10
66.58536585	66.91056910	67.15447154	67.56097560	67.80487804	66.99186991	67.07317073	66.66666666	66.99186991	66.42276422	66.99186991	63.73983739	65.60975609	67.07317073	63.82113821	15
66.34146341	66.50406504	67.15447154	65.93495934	66.91056910	66.82926829	67.72357723	67.47967479	66.91056910	66.01626016	66.26016260	64.79674796	66.50406504	66.01626016	64.06504065	20
66.58536585	67.15447154	67.88617886	67.88617886	67.39837398	67.15447154	66.34146341	66.91056910	66.42276422	66.17886178	66.17886178	66.09756097	66.74796747	66.01626016	64.55284552	25
64.79674796	65.77235772	66.74796747	66.17886178	65.69105691	67.15447154	67.15447154	67.31707317	66.42276422	65.93495934	66.09756097	65.12195121	66.58536585	65.52845528	64.14634146	30
66.42276422	66.66666666	66.74796747	66.82926829	66.26016260	66.50406504	66.58536585	66.34146341	66.58536585	66.58536585	66.42276422	65.60975609	65.93495934	66.58536585	63.82113821	35
66.17886178	66.74796747	66.26016260	66.26016260	66.74796747	66.82926829	66.42276422	66.82926829	66.42276422	66.09756097	66.82926829	65.12195121	66.42276422	66.34146341	63.65835685	40
66.82926829	66.82926829	67.15447154	66.01626016	66.58536585	67.64227642	67.88617886	66.91056910	67.15447154	66.26016260	66.42276422	66.82926829	67.56097560	66.26016260	64.71547154	45
66.26016260	66.58536585	66.82926829	67.88617886	66.99186991	66.91056910	67.15447154	66.34146341	66.34146341	66.50406504	66.50406504	66.66666666	66.26016260	66.17886178	63.73983739	50
66.09756097	66.34146341	66.58536585	66.34146341	66.42276422	66.91056910	66.91056910	67.23577235	67.15447154	66.50406504	66.17886178	66.17886178	66.82926829	65.93495934	64.71547154	55
67.80487804	67.31707317	66.99186991	67.07317073	66.91056910	66.42276422	67.72357723	66.66666666	65.60975609	66.17886178	66.26016260	64.95934959	67.15447154	66.01626016	65.12195121	65
66.82926829	67.88617886	66.42276422	67.07317073	67.39837398	66.58536585	65.52845528	66.66666666	66.82926829	67.07317073	66.82926829	65.52845528	66.17886178	66.26016260	64.30894308	80

<u>Figures 19 and 20 :</u> Comparison between models accuracies over 2018-2019 season (up : raw dataset, down : downsampled dataset)

We can also plot these tables for visualization purposes :





<u>Figure 21 and 22 :</u> Models comparative accuracies display (up : raw dataset, down : downsampled)

- ✓ <u>Comments :</u> We can observe that the normal sampled training set (*i.e.* not downsampling the training set) yields slightly better results than the downsampled study, mainly because we remove observation samples during downsampling. The overall best model was able to correctly guess the winner of the game in **69,34 %** of the time over the 1230 games of the 2018/2019 NBA season (*Nnet 4 & 5 – 80 PC – NormalSampled*). The Neural Networks models are clearly dominating the accuracy tables, hence we could imagine their typical structure is able to perceive statistical patterns that the other classical models can not identify. Furthermore, we can also notice that the SVM linear models are overperforming the SVM radial models, and that the LDA models are also outperforming the QDA models : from this clear observation we can assume that the decision boundary between classes is likely to have a linear shape, and is surely not a quadratic boundary. This can be due to the fact that we already transformed the data in an optimal form through the PCA step. Finally, we can also notice that the number of PCs looks positively correlated with accuracy, meaning that the models cope pretty well with the information contained in the last PCs. However, I decided not to train the models for a larger amount of PCs as the accuracy gain was not so huge and that the training times increased exponentially.

XII - CREATING PROFITABLE BETTING STRATEGIES

(A) STEP-BY-STEP PROCESS

As described earlier, the litterature shows a lot of very powerful models which take many parameters as input and employ a lot of computation power to compute the variables and the models. However, few of them are capable of yielding a great betting strategy, because even if the estimator accurately estimates chances of victory, one also has to cope with the bookmakers margin. Actually, this margin is settled around 5%, which means one should outperform the bookmakers models by more than 5%, which is practicaly impossible most of the time because the bookmakers models are state-of-the-art forecasting models, and they have way more information than I have.

The key point for understanding my strategy is to conceive we don't aim to create the perfect model, which is too difficult given the fact I don't have in-game data for recent matches, but instead I try to identify clear biases in the bookmaker's estimated chances of victory. To do this, I first created a metric which represents the relative confidence offset between my estimated probabilities and the bookmaker's estimated probabilities.

The main innovation is the creation of a metric called confidence index - often referred to as *conf_index* in my scripts - which is the result, for a given game and a predictive model, of the odd multiplied by the estimated winning probability :

$$\begin{cases} ConfidenceIndex_{hometeam} = Odd_{home} * \hat{P}_{model}(winner = hometeam) \\ ConfidenceIndex_{awayteam} = Odd_{away} * \hat{P}_{model}(winner = awayteam) \end{cases}$$

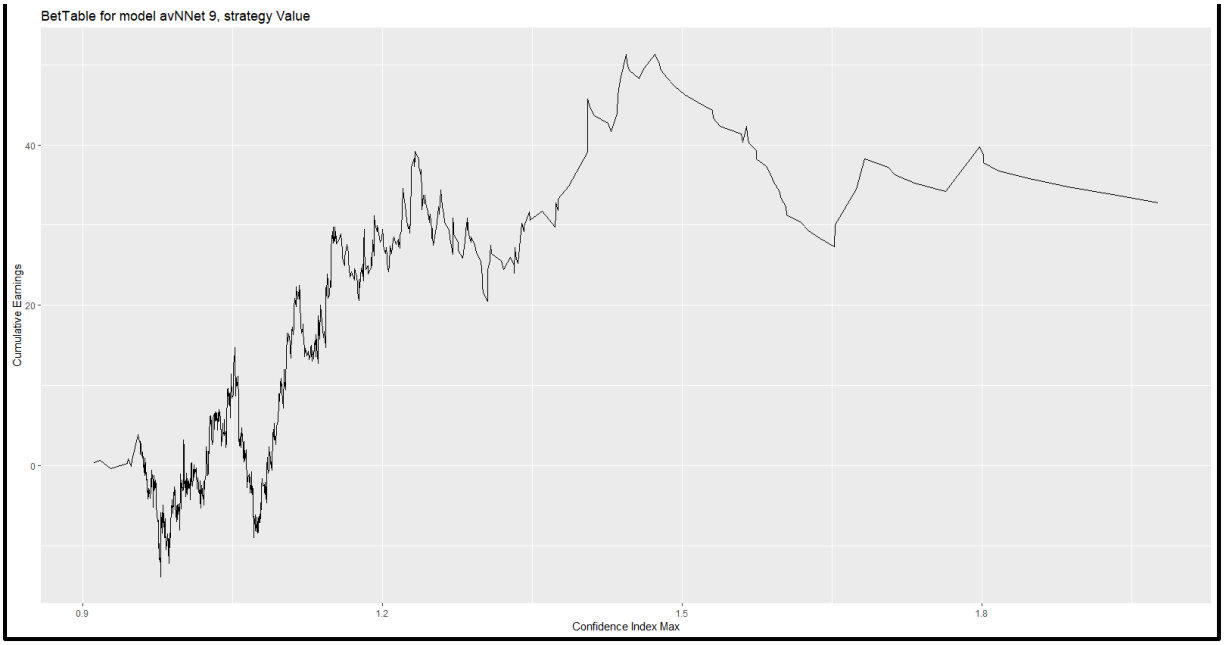
Eventually we compute the maximum of these two quantities to create a unique metric called *conf_index_max* (also referred to as $conf_{index}$ later) :

$$ConfidenceIndexMax = max\left(ConfidenceIndex_{hometeam}, ConfidenceIndex_{awayteam}\right)$$

This metric is supposed to represent the offset between my estimated probabilities of a team winning the game, and the bookmakers' estimated probability (we assume there that the odd should equals the inverse of the bookmakers' estimated probability). As theorically the odd should be equal to $\frac{1}{p}$, then the higher the confidence index, the higher the odd is considered a *value bet*. On the other hand, the lower the confidence index, the lower the bet is a *value bet*.

For each model, we can then sort the current season's games by order of $conf\_index\_max$, and display the cumulative earnings depending of the strategy of betting.

For example, for the LDA model with 65 PCs, the $ConfidenceIndexMax$ associated cumulative earnings when we bet on the value bet look like this :



<u>Figure 23 :</u> Cumulative earnings when betting on the value bet as a function of the confidence_index_max for the LDA model with 65 PCs

- ✓ <u>Interpretation :</u> We can clearly distinguish at least two zones, according to the value of the $ConfidenceIndexMax$: when the metric is lower than 1.1 we would be losing money betting on the value bet, however the trend is straight opposed when the metric is between 1.1 and 1.47. By selecting games where the metric is between 1.1 and 1.47, we would then select profitable games, therefore earning money. In a sense it is quite understable, given the definition of the metric : the higher the ratio, the higher we estimate the offset between our estimated probability and the bookmakers' probability is higher, hence this looks like an opportunity to make money. We can still notice than when the metric is above 1.47, we are losing money, therefore we might want to select only games between 1.1 and 1.47, as mentioned above.

Furthermore, to complete the study, I identified 6 types of strategies, which are respectively betting of our favorite (*i.e.* the team which has more than 50% estimated winning chance), on our value (*i.e.* the team which has the highest *confidence_index*), on the hometeam when the value is on betting home, or on the awayteam when the value is on betting away, on the favorite when the value is on the favorite and finally on the outsider when the falue is on the outsider.

However, the identified region might aswell yield an increasing trend by pure chance, without being a revelator of any bias. But we will see in the results part at the end of the report that this technique works in the long term, hence we can be confident about this process for finding biases.

Finally, I repeated this technique of identifying potential biases over the six strategies, and over the different models. I then obtained a set of "Methods", which I define as the tuple $[Strategy,\ Model,\ region\_of\_potential\_bias]$.

In our example it is then :

$$[Value, LDA, (1.34,1.56)]$$

Once I builded this set of methods, I then had to construct a conflict handling strategy, as sometimes the methods were yielding contradictory bets for the same match – for exampled method A is saying to bet *home* and method B is saying to bet *away*. In such cases, I simply decided to choose the method yielding the best ROI over the season's matches at first, but I later evolved toward another metric to deal with conflicts (developed in next section *Automatic Implementation*)

Once this step done, I also filtered the predictions by having a look at badly predicted areas, depending on the odd :

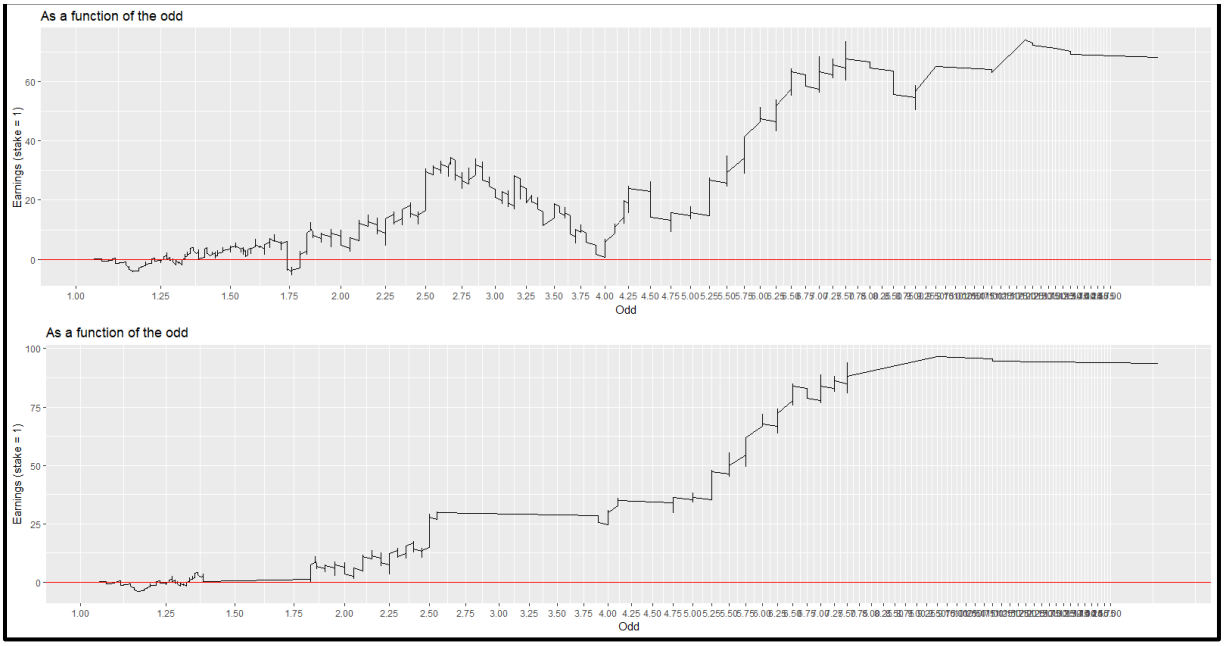


Figure 24 : Post-filtering badly predicted odds

```
> test2 <- post_filtering_table(na.omit(FINAL_TABLE_FILTERED))
1.4 1.8 Filtered!
2.6 3 Filtered!
3 3.4 Filtered!
3.4 3.8 Filtered!
4.2 4.6 Filtered!
8 9 Filtered!
12 13 Filtered!
13 14 Filtered!
```

Figure 25 : Output for the post-filtering step

After having done the selection steps described in the previous paragraph, there we are, the final process is finally over ! I will now describe how I automated the whole process, such that the program is now capable is updating itself and choosing the best strategies all season long, the latter evolving depending on the season's dynamics.

(B) AUTOMATIC IMPLEMENTATION

After having done by hand all the previously described steps, *i.e.* noting down the profitable segments and implementing them by hand on the code every two weeks, I decided to start coding an implementation of these steps so that the program would update itself automatically. The only way I could achieve that was by coding the following process :

1. Given one model, compute all probabilities for the m current season's games, then compute $ConfidenceIndexMax$ for each game, hence for the m first games we obtain a set $\left\{conf_{index_i}, \forall i \in [\![1, m]\!]\right\}$ which we can order such that $i \leq j$ implies $conf_{index_i} \leq conf_{index_j}$
2. Compute all possible segments $\left\{\left[\!\left[conf_{index_i}, conf_{index_j}\right]\!\right], \forall (i,j) \in [\![1, m]\!]^2, i < j\right\} = \{S_l\}_{1 \leq l \leq \frac{m(m-1)}{2}}$
3. For every possible segment S_l, create a table T_l containing all games where $conf_{index} \in S_l$. These tables are our starting set of possible betting methods.
4. Select only the tables T_l which contain at least 30 games, whose ROI is above 15 % and whose ROI over last 10 matches is above 15% aswell, creating a subset of profitable methods $\left\{T_{profitable_j}\right\}$
5. Select only the highest-ROI method from this set of selected methods (the metric can be changed). This step allows a great computation gain, however we might miss the other methods for this model but it is not a huge loss of information compared to the time gain.
6. Repeat steps 1 to 5 for every model in our set of models, and regroup all profitable methods.
7. These methods might eventually yield contradictory predictions for future games, hence to cope with these conflicts, for each predicted game we select only the highest-ROI method (metric can be changed) among our profitable set of methods $\left\{T_{profitable_j}\right\}$

As the process computes all possible segments, we compute a total of $\frac{m(m-1)}{2}$ objects, hence this method has a complexity of $O(m^2)$. However, this exponential complexity is not a limitation to running the algorithm on a daily basis, as the running time is around 30 min and needs to be done at least twice in a week to be efficient. This step is the core of my method as it produces automatically evolving strategies all season long !

The methods are then saved onto a csv file which is later used for predictions. The file looks like this :

Min	Max	ROI	ROIlast10	nMatches	Roi_metric	Model	Strategy	Type
1.2	1.36	0.165151515151515	0.123	33	0.144075757575758	BetTable_s2	Favorite	GAMMA
1.16	1.26	0.1925	0.181	28	0.18675	BetTable_s3	Favorite	GAMMA
1.34	1.54	0.365	0.289	26	0.327	BetTable_s4	Favorite	GAMMA
1.32	1.7	0.167352941176471	0.15	34	0.158676470588235	BetTable_s5	Favorite	GAMMA
1.22	1.44	0.202352941176471	0.155	51	0.178676470588235	BetTable_s6	Favorite	GAMMA
0.48	0.98	0.274444444444444	0.216	36	0.245222222222222	BetTable_s8	Favorite	GAMMA
1.2	1.34	0.396538461538462	0.274	26	0.335269230769231	BetTable_svm_linear	Favorite	GAMMA
0.48	1.04	0.125925925925926	0.125	27	0.125462962962963	BetTable_svm_radial	Favorite	GAMMA
1.12	1.2	0.207380952380952	0.198	42	0.202690476190476	BetTable_rf	Favorite	GAMMA
1.16	1.78	0.195102040816327	0.176	49	0.185551020408163	BetTable_s9	Favorite	GAMMA
1.38	1.7	0.551481481481481	0.415	27	0.483240740740741	BetTable_s10	Favorite	GAMMA
1.34	1.5	0.327777777777778	0.31	27	0.318888888888889	BetTable_lda	Favorite	GAMMA
1.46	1.68	0.410769230769231	0.331	26	0.370884615384615	BetTable_qda	Favorite	GAMMA
0.96	1.02	0.431142857142857	0.16	35	0.295571428571429	BetTable_s1	Value	GAMMA
1.1	1.32	0.382602739726027	0.315	73	0.348801369863014	BetTable_s3	Value	GAMMA
1.36	1.56	1.91846153846154	1.53	26	1.72423076923077	BetTable_s4	Value	GAMMA
1.26	1.4	0.425641025641026	0.315	39	0.370320512820513	BetTable_s5	Value	GAMMA

Figure 26 : Output from the methods selection algorithm

I chose to select the methods according to another metric, instead of the ROI : I created the metric

$$ROI_{metric} = \frac{Roi + Roi_{last10}}{2}$$

The choice of this metric is justified by the fact that then we can then select methods that are both working on the long-run and on the last matches, so that we select methods that are supposedly in great form aswell. The research of new metrics could be the subject of future work and implementations.

XIII - CREATE A USER INTERFACE SUITABLE FOR EVERYBODY

Most of the tasks have now been automatically. As a result, the process to make predictions over the script is :

- 1 – Run NEXT_PRED.R : the code will automatically call the python webscraping script in order to collect the very last odds, then it will compute winning probabilities and predicted bets.
- 2 – The output is a table which displays in Rstudio – and is also sent by mail via an SMTP server directly from R - where we can read the predictions since the beginning of the season and the future games
- 3 – Bet on the games.

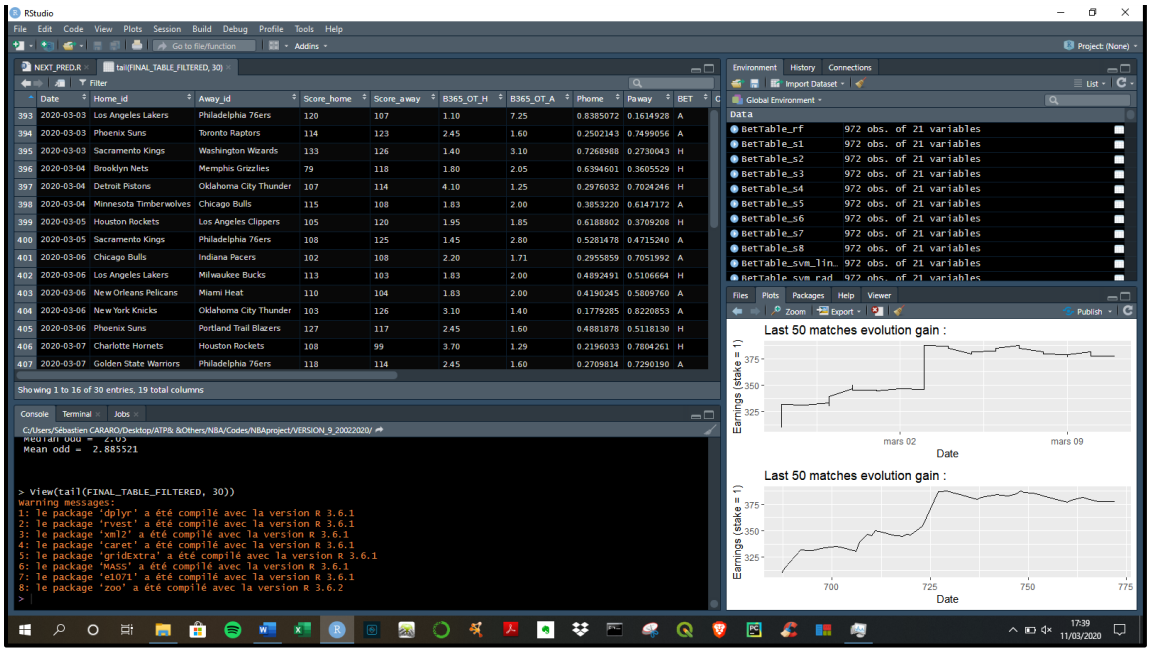


Figure 27 : Final Rstudio rendering

XIV - DESCRIBE THE PRINCIPLE WITH ATP, FOOTBALL, BASKET, HOCKEY, ESPORTS, RUGBY AND BASEBALL

As I was satisfied with the NBA program and the profits earned thanks to it, I then decided to apply the same pipeline process to various sports forecasting. As I was more at ease with the principle, I could then quickly code the algorithms and proceed to betting on the proposed games.

Concerning football and tennis, the creation of the two programs was facilitated by the two source websites which contain an impressive amount of already scraped data ([34] & [35]). Concerning the other sports (hockey, basketball, eSports, rugby), the data was directly scraped onto *oddsportal.com.* For the hockey program I decided to focus on the *moneyline* market, *i.e.* forecasting the winner – as opposed to 1X2 forecasting.

The latter programs are more evolved, for example I was able to compute a lot more variables (respectively 191 and 330 variables for football and tennis – [appendix n° 1]). I just had to adapt the computed variables based on the sport and proceed to a few modifications. Also, the betting strategies were different, according to the sport format : for football I studied 7 different betting strategies (Value, Favorite, Home, Draw, Away, Value&Outsider, Value&Favorite) and for tennis I only had 4 strategies available (Favorite, Value, Value&Outsider, Value&Favorite).

Once I had finished the first football program for the British Premier League and as my source datasets were in the same format for each league I simply had to run the pipeline onto the other leagues to create a program. In total I created 35 football scripts ([appendix n° 2]), 6 rugby leagues, 1 hockey league, 20 basketball leagues, 1 tennis league, 3 baseball leagues and 10 eSports leagues, which add up to 76 leagues in total (!).

XV - RESULTS IN TERMS OF ACCURACY

As a first approach to assess my models' accuracy relevancy, I compared the bookmaker's performance in predicting the winner with my models' predictions. For example, if the odds from bet365 from a given match are :

Odd Home	**Odd Draw**	**Odd Away**
2	3	3,5

Then we can infer that the bookmakers' estimated favorite is the home team (lowest odd).

This being said, I observed in each league the winning percentage of the bookmaker's favorite over the test set and I compared it to my personal models. The differential can give an idea of the accuracy of my models, even though a good performance could also be result of luck over the test set forecast. In the following results that are displayed hereafter, we can notice that football and basketball models are performing extremely well. These recap results are very interesting in the sense that we can notice the pipeline is working better on certain leagues than others.

Figure 28 : Football results

Country	League	TestSet	Bookmaker	# Games in test set	Bookmakers' favorite winning %	Best model accuracy	Difference
Argentina	SuperLiga	test sur 2018 et 2019	Pinnacle	702	48,1481	49,0028	0,8547
Australia	A-League	test sur 2018 et 2019	Average	280	55,3571	54,2857	-1,0714
Austria	Tipico Bundesliga	test sur 2018 et 2019	Pinnacle	377	55,4376	53,84	-1,5976
Belgium	Jupiler Pro League	test sur 2018 et 2019	bet365	480	51,25	51,0416	-0,2084
Bielorussia	vysshaya-liga	test sur 2018 et 2019	bet-at-home	224	54,4642	54,4642	0
Brasil	Serie A	test sur 2018 et 2019	Pinnacle	760	55,5263	55,5263	0
China	Super League	test sur 2018 et 2019	Pinnacle	480	56,6666	57,5	0,8334
Denmark	Superliga	test sur 2018 et 2019	Pinnacle	506	51,1857	51,581	0,3953
Finland	Veikkausliiga	test sur 2018 et 2019	Pinnacle	369	50,9485	48,2384	-2,7101
France	Ligue 1	test sur 2018 et 2019	bet365	760	51,9736	52,1052	0,1316
France	Ligue 2	test sur 2018 et 2019	bet365	760	48,5526	47,8947	-0,6579
Germany	Bundesliga	test sur 2018 et 2019	bet365	612	52,6143	54,0849	1,4706
Germany	Bundesliga 2	test sur 2018 et 2019	bet365	612	41,1764	44,4444	3,268
Iles Féroés	Premier League	test sur 2018 et 2019	bwin	172	62,7906	66,279	3,4884
Ireland	Premier Division	test sur 2018 et 2019	Pinnacle	371	58,7601	61,725	2,9649
Italy	Serie A	test sur 2018 et 2019	bet365	760	57,6315	57,5	-0,1315
Italy	Serie B	test sur 2018 et 2019	bet365	804	45,1492	46,0199	0,8707
Japan	J-League	test sur 2018 et 2019	Pinnacle	619	46,3651	46,8497	0,4846
South Korea	K-League	test sur 2018 et 2019	bet365	462	47,619	48,0519	0,4329
Mexico	Liga MX	test sur 2018 et 2019	Pinnacle	668	49,8502	49,4011	-0,4491
Netherlands	Eredivisie	test sur 2018 et 2019	bet365	612	56,8627	58,0065	1,1438
Norway	Eliteserien	test sur 2018 et 2019	Pinnacle	483	49,4824	51,3457	1,8633
Poland	Ekstraklasa	test sur 2018 et 2019	Pinnacle	592	46,7905	46,9594	0,1689
Portugal	Liga NOS	test sur 2018 et 2019	bet365	612	59,3137	60,4575	1,1438
Romania	Liga 1	test sur 2018 et 2019	Pinnacle	540	51,2962	54,6296	3,3334
Russia	Premier League	test sur 2018 et 2019	Pinnacle	488	53,8934	53,8934	0
Scotland	Scottish Premiership	test sur 2018 et 2019	bet365	456	56,5789	56,5789	0
Spain	Liga Primera	test sur 2018 et 2019	bet365	760	51,8421	52,3684	0,5263
Spain	Liga Secunda	test sur 2018 et 2019	bet365	902	48,5587	49,0022	0,4435
Sweden	Allsvenskan	test sur 2018 et 2019	Pinnacle	484	57,6446	60,5371	2,8925
Switzerland	Super League	test sur 2018 et 2019	Pinnacle	362	50,8287	53,3149	2,4862
Turkey	Süper Lig	test sur 2018 et 2019	bet365	609	53,0377	54,1871	1,1494
United Kingdom	Premier League	test sur 2018 et 2019	bet365	760	56,9736	57,3684	0,3948
United Kingdom	Championship	test sur 2018 et 2019	bet365	1104	47,192	47,826	0,634
USA	MLS	test sur 2018 et 2019	Pinnacle	829	54,7647	54,7647	0
							=23/35

Figure 29 : Basketball results

Country	League	TestSet	Bookmaker	# Games in test set	Bookmakers' favorite winning %	Best model accuracy	Difference
Italy	Lega A	test sur 2018/2019 + 2019/2020	Average	447	69,5749	71,1409	1,566
Spain	ACB	test sur 2018/2019 + 2019/2020	Average	530	70,1886	70,566	0,3774
USA	NBA	test sur 2018/2019	bet365	1230	67,6422	69,3495	1,7073
France	Pro A (lnb)	test sur 2018/2019 + 2019/2020	Average	550	71,2727	71,0909	-0,1818
Russia	vtb-united-league	test sur 2018/2019 + 2019/2020	Average	326	72,0858	73,0061	0,9203
Argentina	liga A (Argentine)	test sur 2018/2019 + 2019/2020	Average	696	67,9597	69,3965	1,4368
Australia	NBL (australie)	test sur 2018/2019 + 2019/2020	Average	255	64,7058	65,8823	1,1765
Brazil	NBB (Brésil)	test sur 2018/2019 + 2019/2020	Average	425	70,8235	72,2352	1,4117
China	CBA (Chine)	test sur 2018/2019	bet365	493	76,8762	78,0933	1,2171
Europe	Euroleague	test sur 2018/2019 + 2019/2020	Average	512	69,5312	70,1171	0,5859
Europe	Eurocup	test sur 2018/2019 + 2019/2020	Average	353	70,5382	70,5382	0
Balkans	ABA league	test sur 2018/2019 + 2019/2020	Average	271	74,9077	76,0147	1,107
Germany	BBL (Allemagne)	test sur 2018/2019 + 2019/2020	Average	502	71,3147	72,5099	1,1952
Lithuania	LKL (Lithuanie)	test sur 2019/2020	Average	120	75,8333	77,5	1,6667
South Korea	KBL (Corée du Sud)	test sur 2019/2020	Average	213	54,46	60,5633	6,1033
Turkey	Tbl (Turquie)	test sur 2018/2019 + 2019/2020	Average	407	76,4127	78,3783	1,9656
France	Pro B (France)	test sur 2018/2019 + 2019/2020	Average	531	66,1016	64,7834	-1,3182
Europe	Champions League	test sur 2019/2020	Average	271	71,2177	69,3726	-1,8451
Europe	Europe Cup	test sur 2019/2020	Average	161	75,1552	77,0186	1,8634
USA	WNBA	test sur 2019	Average	233	66,5236	71,6738	5,1502
							= 16/ 20

Figure 30 : Hockey result

Country	League	TestSet	Bookmaker	# Games in test set	Bookmakers' favorite winning %	Best model accuracy	Difference
USA	NHL	test sur 2018/2019	Average	1435	56,2369	58,1184	1,8815
							= 1/ 1

Figure 31 : Tennis result

Country	League	TestSet	Bookmaker	# Games in test set	Bookmakers' favorite winning %	Best model accuracy	Difference
World	ATP	2018 & 2019	bet365	5058	68,8216	67,0225	-1,7991

Figure 32 : eSports results

Country	League	TestSet	Bookmaker	# Games in test set	Bookmakers' favorite winning %	Best model accuracy	Difference
Chine	LoL Pro League	test sur 2020 (first 110)	1xBet	110	61,8181	69,0909	7,2728
South Korea	Lol Champions Korea	test sur 2020 (first 85)	Unibet	85	67,0588	77,647	10,5882
Usa	OverWatch League	test sur 2020 (first 61)	1xBet	61	63,9344	77,0491	13,1147
Brasil	LoL Brasileiro	test sur 2020	1xBet	72	56,9444	59,7222	2,7778
Australia	LoL Oceanic Pro League	test sur 2020	1xBet	88	71,5909	76,1363	4,5454
Russia	LoL Continental League	test sur 2020 (first 50)	1xBet	50	72	80	8
South Korea	Starcraft 2 Korea	test sur 2020	bet365	13	84,6153	84,6153	0
Europe	LoL European Championship	test sur 2020 (first 98)	1xBet	98	75,5102	76,5306	1,0204
Europe	LoL European Masters	train sur 90 premiers, test sur 15 derniers	Coolbet	15	80	93,3333	13,3333
World	Counter Strike ALL TOURNAMENTS	465 last games (as of 27/04/2020)	bet365	465	71,3978	71,3978	0
							= 8/ 10

Figure 33 : Rugby results

Country	League	TestSet	Bookmaker	# Games in test set	Bookmaker s' favorite winning %	Best model accuracy	Difference
Australia	NRL	test sur 2018 & 2019	Average	401	64,0897	63,0922	-0,9975
Australia	AFL	test sur 2018 & 2019	Average	413	68,7651	71,4285	2,6634
World	Super Rugby	test sur 2018 & 2019	Average	246	72,3577	71,1382	-1,2195
France	Top 14	2018-2019 & 2017-2018	Average	365	73,4246	73,6986	0,274
France	Pro D2	2019-2020	Average	178	76,9662	76,9662	0
England	Aviva Premiership	2018-2019 & 2019-2020	Average	208	70,673	70,673	0
							= 2/ 6

Figure 34 : Baseball results

Country	League	TestSet	Bookmaker	# Games in test set	Bookmaker s' favorite winning %	Best model accuracy	Difference
South Korea	KBO	test sur 2019	Pinnacle	724	60,0828	61,1878	1,105
Taiwan	CPBL	test sur 2018	1xBet	208	57,2115	57,2115	0
USA	MLB	test sur 2018 & 2019	Average	5653	58,8713	58,3407	-0,5306
							= 1/ 3

XV - RESULTS IN TERMS OF SPORT BETTING

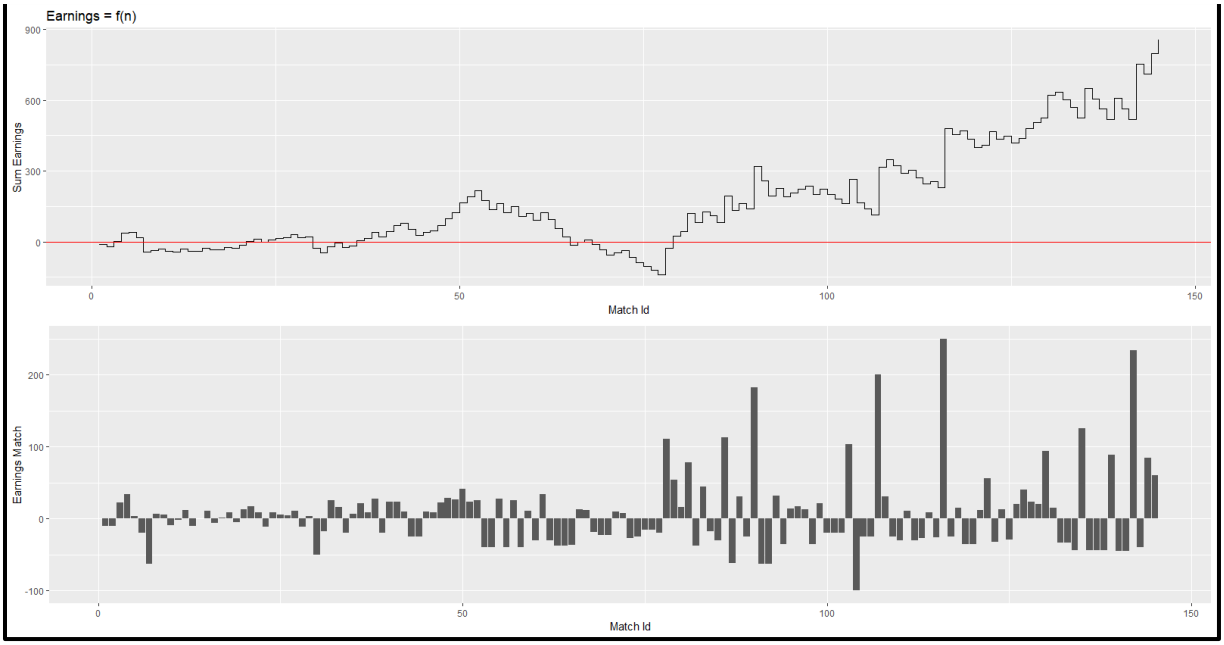
NB : The models and techniques developed during this project have been evolving for the best all way through, hence at the day I am writing this paper I consider having my best pipeline ever, but when I started to bet with the first algorithms I used models and techniques that I now consider way less efficient than the more recent ones. In particular, the automatic implementation of strategies was a huge step forward, as before this I had to do it by hand (~1h/week/league), hence I could only do it once in a couple weeks.

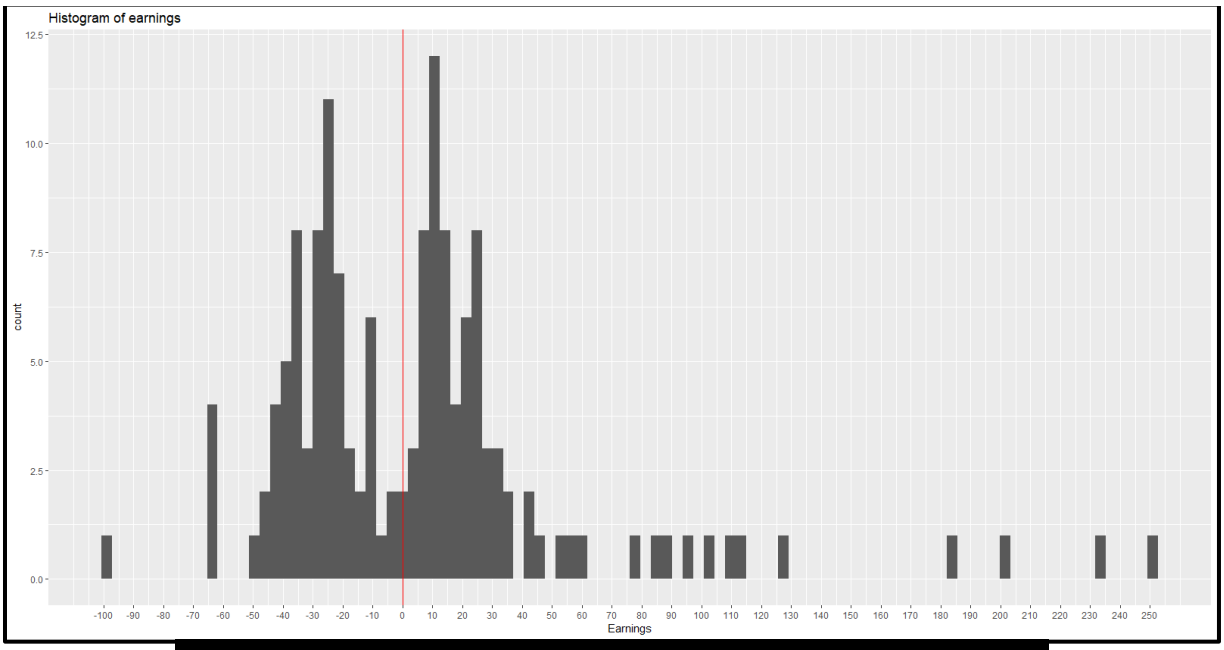
As a consequence, the newly constructed models are likely to give way better results in the future than the ones presented in the next section. Nonetheless, one might appreciate the already high-performing results obtained from January 15th to March 13rd, presented there down :

(A) NBA BETTING

Thanks to this pipeline, I've been able to apply betting instructions derived from this algorithm, and was able to earn 856€ thanks to the NBA games. The NBA league is by far my best program, which is delightful considering the fact that there are a lot a games over the 6-months season. Here are a few plots that display the path to earning this amount :

Figure 35 & figure 36 : Cumulative earnings and marginal earnings (for each game)
Figure 37 : Histogram of earnings
Figure 38 : Detailed NBA betting statistics






```
ROI since 31/12/2020 is  19.83605 % with  145 matches
Earnings =  856.737 €
Sum stakes =  4319.09 €

Accuracy =  53.10345 %
Median odd =  1.91
Mean odd =  2.602579
Maximum Earnings in one match =  250 €
Maximum lost in one match =  -100 €
Maximum level reached =  856.737 €
Minimum level reached =  -138.7 €
```

- ✓ <u>Commentary :</u> We can notice that the NBA has been a very profitable league, with a double-digit ROI percentage. Even though there had been a dark series between game n°50 and game n°80 – which might be due to the fact that this period was around the All-Star break, hence the dynamics could have been different (star player

can rest to avoid injuries before the All-Star, plus some teams feel exhausted after 4 straight months) – the general trend is clearly profitable, and during the last dozens of games it was very profitable. We can also notice that the program advices to bet on outsider (mean odd = 2.6) and that we could eventually make huge winning bets : Charlotte Hornets @ Toronto Raptors on February, 29th (Odd 9) and Golden State Warriors @ Denver Nuggets on March, the 4th (Odd 12).

(B) ALL RESULTS

As of the 13rd of march, 2020, the application of the described strategy, coupled with a staking strategy along which I staked more on well-performing leagues such as the NBA and the italian football league *Serie A,* I realised a net profit of 1284,33 € with an intial bankroll of 250€ between the 31st of December and the 13rd of March, with a return of investment of 7,42%, which perfectly fits my expectations :

Figure 39 : Profit evolution since January 2020 (€) as a function of the total stakes placed

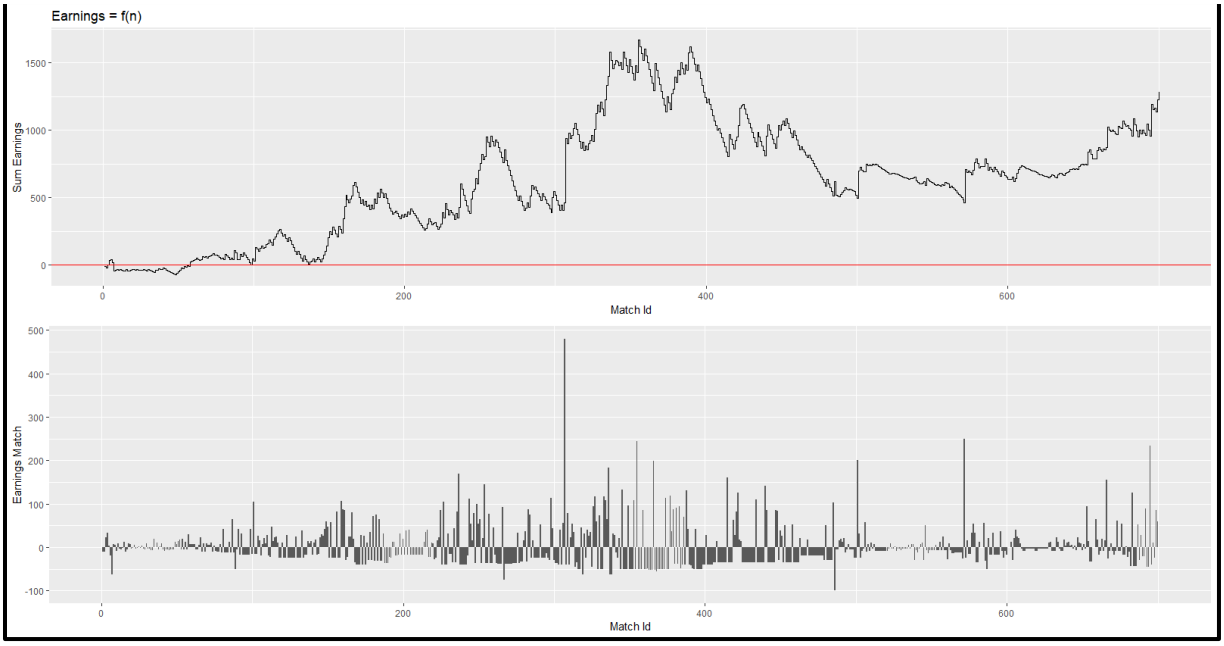


Figure 40 : Profit evolution as a function of the number of bets

```
ROI since 31/12/2020 is  7.425827 % with  694 matches
Earnings =  1284.33 €
Sum stakes =  17295.45 €

Accuracy =  37.75216 %
Median odd =  2.955
Mean odd =  3.368375
Maximum Earnings in one match =  480 €
Maximum lost in one match =  -100 €
Maximum level reached =  1672.226 €
Minimum level reached =  -71.05 €
```

Figure 41 : Betting statistics since january 2020

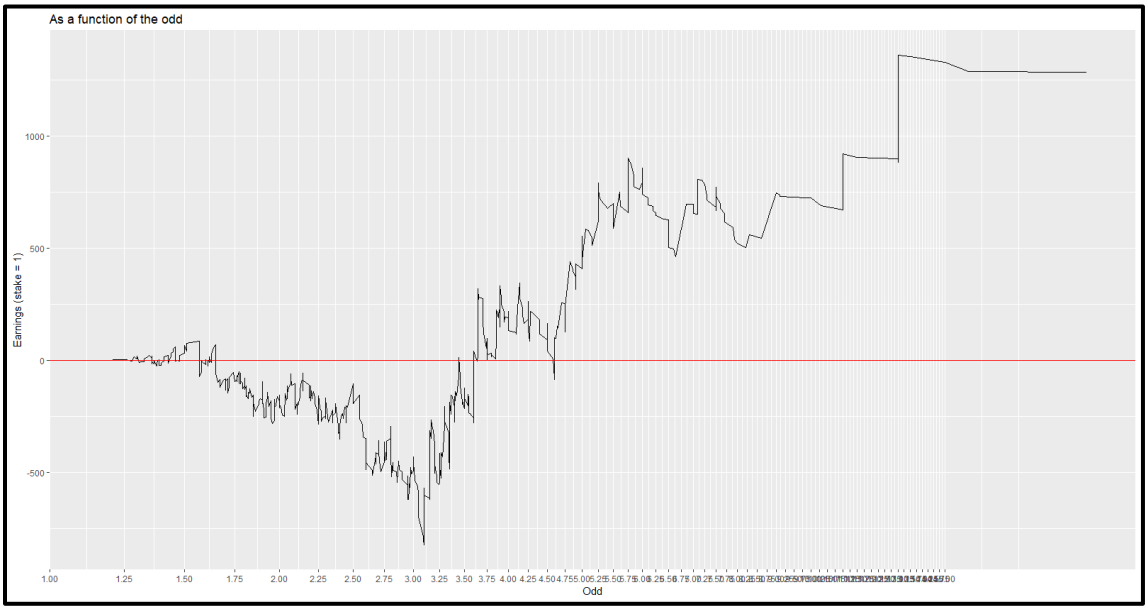


Figure 42 : Profit evolution as a function of the odd

- ✓ <u>Commentary :</u> The shape of the profit curve was at first very promising, as the ROI was oscillating between the 10% and the 20% projection, however at some point there was a *really* dark series which made me lose half of my profit. I was a bit disapointed at this time, hence I chose to drastically review my staking plan. We can see that during the last two weeks of exploitation the algorithm was once again yielding great results, which were brutally stopped by the raise of the Covid-19 pandemic, forcing all championships to stop the season from then on.
  We can also notice, from the last plot, that we are losing money on odds <3.15 and earning our money on great odds (odd > 3.15).

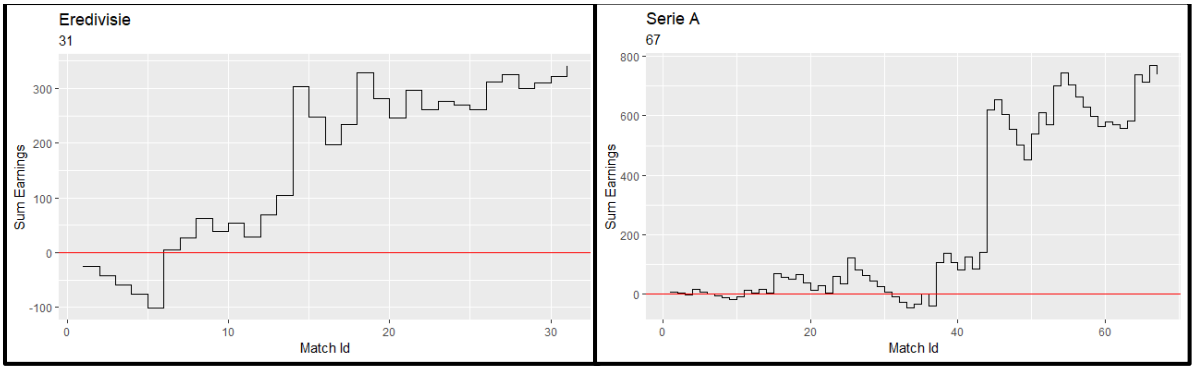
There down is displayed the results according to the league, with some leagues performing way better than others. In the future one could think about testing the leagues with paper betting before betting real money on them, which would have had allowed me to exclude some leagues such as *Bundesliga*, *Bundesliga 2* and *Liga NOS*. However, I did not have time to do such task and preffered starting betting because the season was already coming to an end.

	N_matches	Earnings€	ROI%	Accuracy%	Mean odd	Median Odd
NBA	145	856.7370	19.836053	53.10345	2.602579	1.910
Serie A	67	737.2165	44.147873	38.80597	4.306552	3.400
Championship	91	418.9990	19.051384	34.06593	3.308264	3.200
Ligue 2	34	352.2920	38.542717	41.17647	3.507647	2.910
Eredivisie	31	339.4502	38.709811	51.61290	2.636839	2.150
SC0	24	85.9500	12.233656	37.50000	4.234167	3.420
Turquie	1	51.5620	254.000000	100.00000	3.540000	3.540
Jupiler League	28	-15.2560	-2.034405	42.85714	3.349643	2.815
Liga Secunda	14	-24.2020	-15.766775	35.71429	3.278571	3.005
Bundesliga	32	-128.0510	-15.194243	28.12500	3.584062	3.175
Premier League	51	-156.6650	-21.984368	33.33333	4.513137	3.800
Ligue 1	57	-191.3640	-15.522964	36.84211	2.919211	2.250
Liga Primera	42	-216.1400	-21.410175	19.04762	3.644286	3.600
Serie B	20	-228.7000	-48.328473	20.00000	3.117500	2.825
Bundesliga 2	31	-273.8984	-37.297565	22.58065	3.299355	3.400
Liga NOS	26	-323.6000	-47.261574	19.23077	3.688077	3.250

<u>Figure 43 :</u> Earnings by league

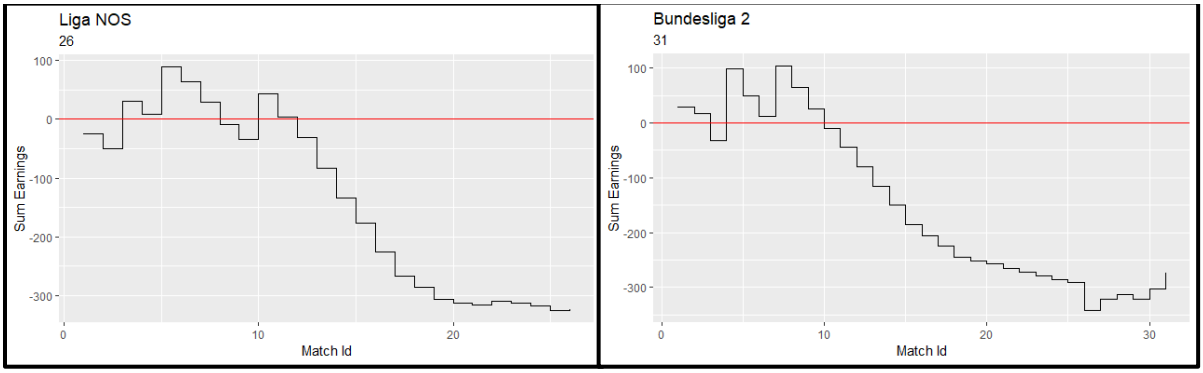
- ✓ Commentary: We can notice that the results are very polarized, in the sense that for each league, the related algorithm is either yielding tremendous profits (double-digit ROI's) or yielding dramatic losses. The programs were created with the same pipeline though, therefore we could explain these differences by the fact that there are relatively few matches per league, hence it could be a matter of variance, or that our method is simply working with some leagues, and isn't working for some other leagues. However, those results are very satisfying because overall we made profits, and that if, for the losing leagues, the losing trend continues then we would just need to stop betting on these leagues and simply continue with the profitable leagues.

As examples, we can have a look at well performing leagues such as *Eredivisie* and *Serie A* profiles, which display monotonous high-profitable figures and very enjoyable to look at. For these leagues the process is clearly validated :



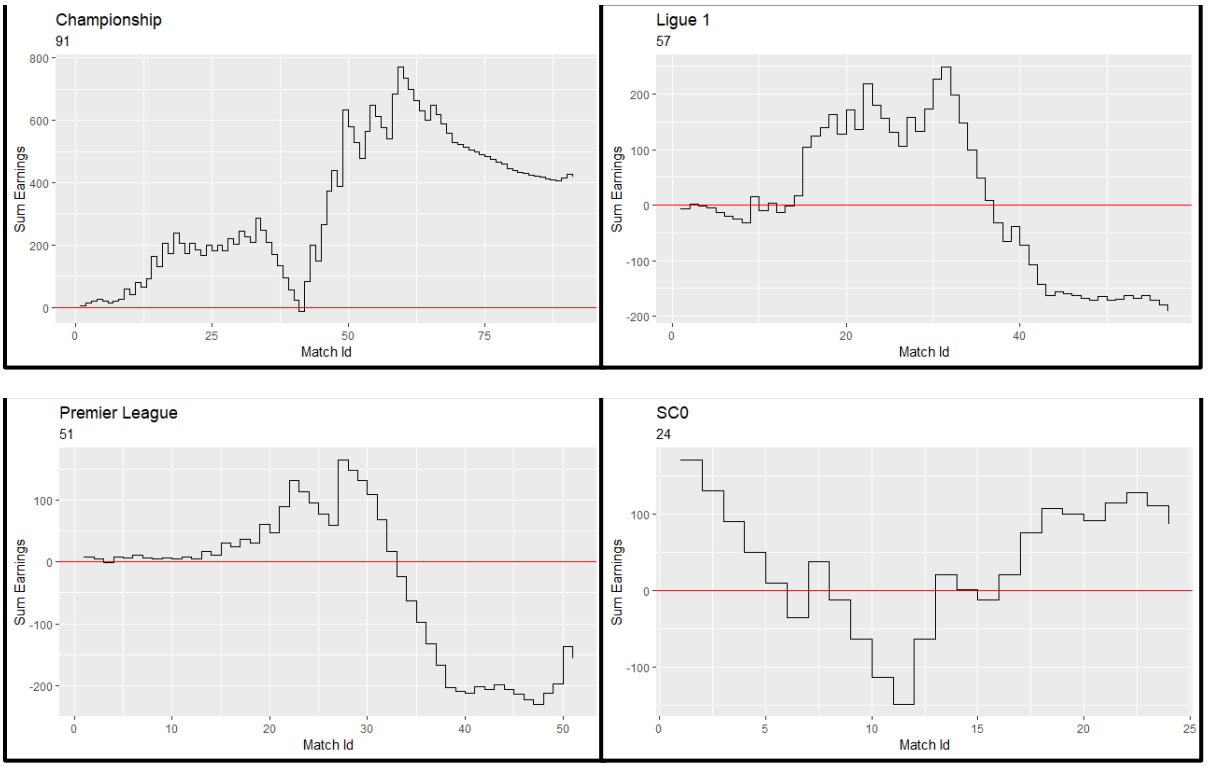
Figures 44 & 45 : Eredivisie and Serie A cumulative earnings

In the other hand, some leagues display clear opposite trends, which - as I explained – may be attributed either to bad luck or to an invalidation of the process for these leagues :



Figures 46 & 47 : Liga NOS and Bundesliga 2 cumulative earnings

Finally, there are multiple leagues for which the trend is changing over time, with great periods alternating with dark series :



<u>Figures 48, 49, 50, 51 :</u> Championship, Ligue1, Premier League and Scottish Premiership cumulative earnings

(C) MONTE-CARLO APPROACH FOR NBA

Having all these results, we may now want to evaluate the degree at which these results are satisfying. To do this, we will proceed to a Monte-Carlo [36] set of simulations in order to benchmark our results with multiple basic strategies, which are respectively :

- Random Betting
- Home Betting
- Away Betting
- Favorite Betting (*lowest odd*)
- Outsider Betting (*biggest odd*)

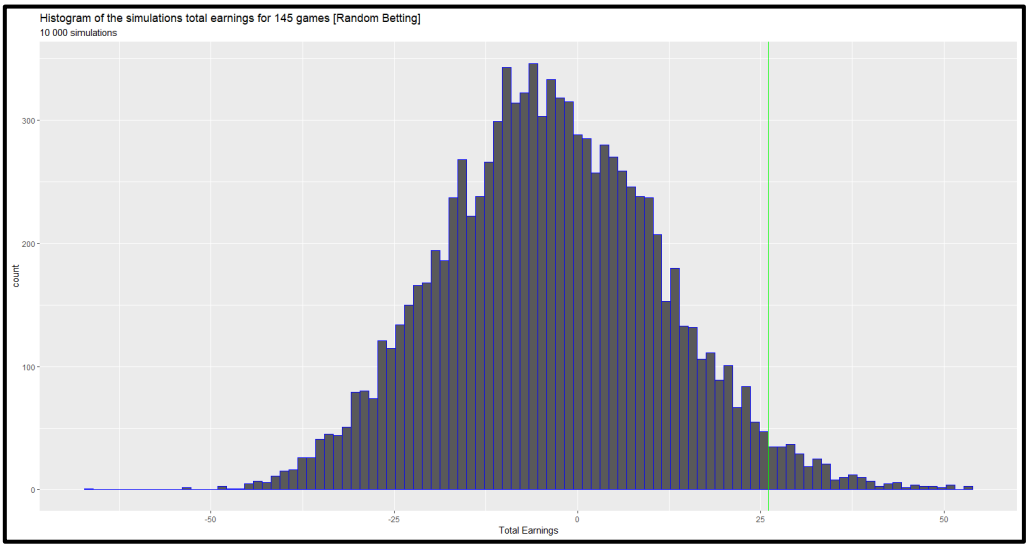
For each strategy, we will draw 145 games *(I did bet on 145 NBA games this year*) out of the 970 finished games this season, and look at the theoretical profit accordingly to each strategy. I decided to draw a total of $n = 10000$ simulations. I then had a look at the cumulative earnings in order to compare them with my selection of bets.

In order to account for the fact that, during my real-life bets, I did not stake the same amount of money for each game, I had to normalize my record of betting first (*i.e.* simply set the stake of each bet to 1€, and update the theoretical earnings for each match). After having done this step, my set of 145 matches yields the following results :

```
After normalization, the 145 matchs yield the following results :
 Earnings =  26.012 €
 ROI% =  17.93931 %
```

From the simulations – whose results are presented here after – we can draw the conclusion that our real-life bets are very unlikely to have happened by pure luck , and that therefore the profits reflect a real profitable strategy. The best dummy strategy over this season was the away betting strategy, even though – of course – in the long-run such strategy would yield losses.

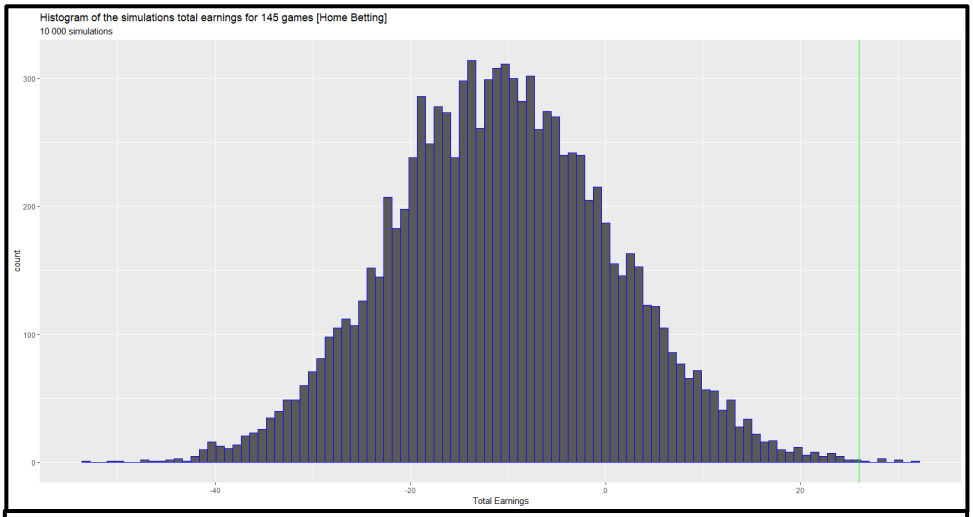
[I] – RANDOM BETTING




```
Statistics of the 10 000 simulations for 145 games [Random Betting]:
 Mean earnings =  -3.568851 €
 Standard Deviation = 15.11048 €

 which is equivalent to :
 Mean ROI% =  -2.461277 %
 Standard Deviation ROI% = 10.42102 %
 # of simulations that yield better results than mine: 285 ( 2.85 %)
```

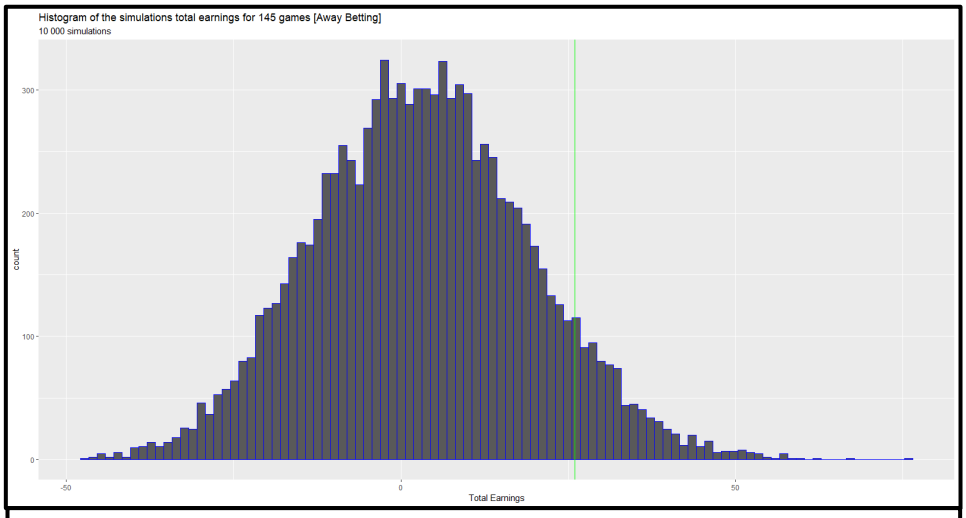
[II] – HOME BETTING




```
Statistics of the 10 000 simulations for 145 games [Home Betting]:
 Mean earnings =  -10.60622 €
 Standard Deviation = 11.39654 €

 which is equivalent to :
 Mean ROI% =  -7.314637 %
 Standard Deviation ROI% = 7.859684 %
 # of simulations that yield better results than mine: 8 ( 0.08 %)
```

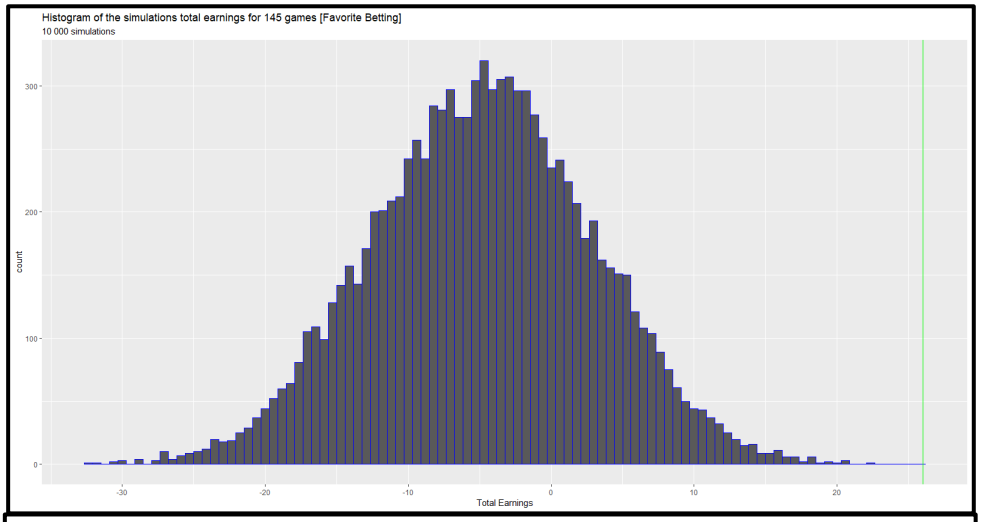
[III] – AWAY BETTING




```
Statistics of the 10 000 simulations for 145 games [Away Betting]:
 Mean earnings =  3.169425 €
 Standard Deviation = 16.19288 €

 which is equivalent to :
 Mean ROI% =  2.18581 %
 Standard Deviation ROI% = 11.1675 %
 # of simulations that yield better results than mine: 824 ( 8.24 %)
```

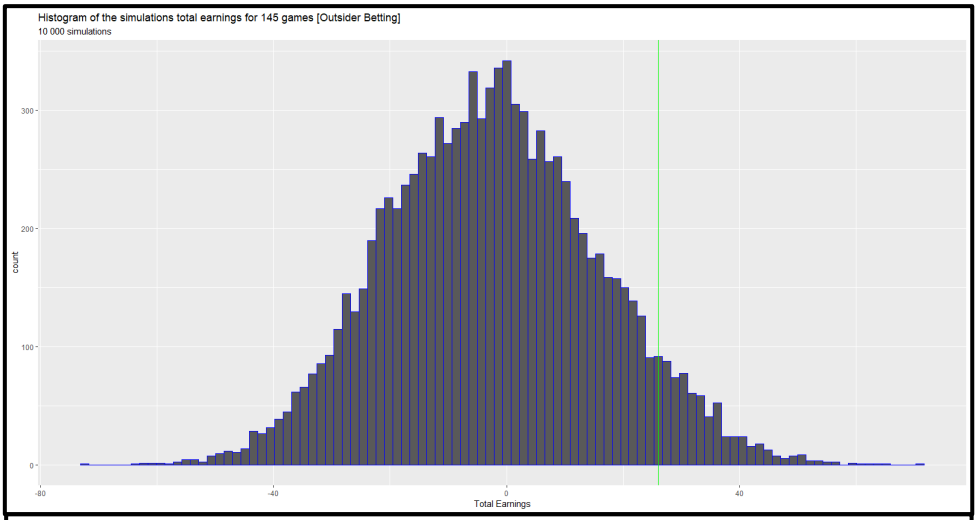
[IV] – FAVORITE BETTING




```
Statistics of the 10 000 simulations for 145 games [Favorite Betting]:
 Mean earnings =  -4.800317 €
 Standard Deviation = 7.723374 €

 which is equivalent to :
 Mean ROI% =  -3.310563 %
 Standard Deviation ROI% = 5.326465 %
 # of simulations that yield better results than mine: 0 ( 0 %)
```

[V] – OUTSIDER BETTING




```
Statistics of the 10 000 simulations for 145 games [Outsider Betting]:
 Mean earnings =  -2.430745 €
 Standard Deviation = 18.38893 €

 which is equivalent to :
 Mean ROI% =  -1.676376 %
 Standard Deviation ROI% = 12.68202 %
 # of simulations that yield better results than mine: 674 ( 6.74 %)
```

(D) MONTE-CARLO APPROACH FOR FOOTBALL

With the exact same approach, we can estimate the magnitude of the results by applying a Monte-Carlo set of simulations for each football league that I bet on. The results are displayed on the following table, with columns 1 to 3 representing my personal betting record, columns 4 & 5 representing my normalized betting record, and columns 6 & 7 the simulations results :

	N_matches	Earnings€	ROI%	NormalizedEarnings€	NormalizedRoi%	SimulationsMeanRoi%	SimulationsStdRoi%	Mean difference%	Mean difference σ	#Simulations that were better than me
ALL_LEAGUES	549,00	427,59	3,30	3,01	0,55	-4,74	6,83	5,29	0,77	2157 (21,57%)
Bundesliga	32,00	-128,05	-15,19	-7,81	-24,41	-2,59	31,65	-21,81	-0,69	7503 (75,03%)
Bundesliga2	31,00	-273,90	-37,30	-6,95	-22,42	-3,81	25,85	-18,61	-0,72	7554 (75,54%)
Championship	91,00	419,00	19,05	2,47	2,71	-3,04	15,77	5,76	0,37	3521 (35,21%)
Eredivisie	31,00	339,45	38,71	10,14	32,71	-6,41	31,87	39,12	1,23	1092 (10,92%)
Jupiler League	28,00	-15,26	-2,03	-0,14	-0,50	-10,86	28,62	10,36	0,36	3418 (34,18%)
Liga NOS	26,00	-323,60	-47,26	-11,44	-44,00	-5,06	31,69	-38,94	-1,23	8991 (89,91%)
Liga Primera	42,00	-216,14	-21,41	-16,66	-39,67	-6,46	23,39	-33,21	-1,42	9269 (92,69%)
Liga Secunda	14,00	-24,20	-15,77	1,62	11,57	-3,02	39,35	14,59	0,37	3462 (34,62%)
Ligue 1	57,00	-191,36	-15,52	-8,67	-15,21	-5,70	21,32	-9,51	-0,45	6513 (65,13%)
Ligue 2	34,00	352,29	38,54	13,89	40,85	-4,11	25,44	44,96	1,77	451 (4,51%)
Premier League	51,00	-156,67	-21,98	6,58	12,90	0,04	26,70	12,86	0,48	2929 (29,29%)
SC0	24,00	85,95	12,23	8,01	33,38	-7,94	35,08	41,32	1,18	1149 (11,49%)
Serie A	67,00	737,22	44,15	21,38	31,91	-2,62	20,71	34,53	1,67	560 (5,60%)
Serie B	20,00	-228,70	-48,33	-11,95	-59,75	-6,97	31,63	-52,78	-1,67	9603 (96,03%)
Turquie	*1,00*	*51,56*	*254,00*	*2,54*	*254,00*	*7,39*	*148,80*	*246,61*	*1,66*	*739 (7,39%)*

Figure 52 : Results of Monte-Carlo simulations concerning the football record

- ✓ Commentary : We can see that the results are very polarized, with some leagues working very well (*Eredivisie*, *Ligue 2*, *Scottish premiership*, *Serie A*) and some others which are far from being profitable. We could explain this by the fact that there are relatively few games that I've bet on certain leagues, or by a bad luck or even because the pipeline wouldn't work with certain leagues because of their specific dynamics that somehow the algorithm cannot perceive. However, on average the ROI is positive and we still have the choice to ignore non-profitable leagues in the future, as written earlier.

XVI - LIMITATIONS AND FUTURE WORK

(A) LIMITATIONS

In order to complexify the models, we could think about encoding supplementary variables to get more precise models. We could take the players performances into account, along with in-game statistics such as the field goal %, the 3-point shots %, free throws %, rebounds statistics, etc… . We could also look forward to taking injuries into account.

Concerning the organization side, as at the beginning of the project I had to manually update odds before each game, at some point – especially during week-ends where there was about 60 games on Saturday/Sunday – I would make a few staking errors. Especially at some point I staked 100 euros and 67 euros instead of 20 euros (and lost both times!). There are few ways to cope with this issue, as most online bookmakers don't allow bots to place bets, hence it is probable that I will need to place bets manually for a long time.

(B) DEPLOYMENT AND CURRENT WORK

Thanks to windows task scheduler and various personnally created webscraping programs, at this moment the program is practicaly autonomous, in the sense that I only need to schedule the times at which I want each program to run, and then the algorithm scrapes the next games on www.oddsportal.com , then computes predictions and it finally sends the predictions by mail directly onto my email address. Thanks to this organization I now only need to schedule running times and updates of the programs – which I simply need to launch for night computing. Apart from computation times optimization, the program is fully completed and is fully deployed right now, I am waiting for the start of next season in order to – hopefully – make great profits !

For information, this is the current deployment process, the user only needs to schedule the time of running with Windows Task Scheduler and then the whole process is done autonomously, as the python webscraper is directly called at the beginning of the R script :

```
system("python 'C:/Users/Sébastien CARARO/Desktop/ATP& &Others/NBA/Codes/NBAproject/VERSION_11_08042020/Scrape/Scrape_NextPred_NBA.py'")
```

Figure 53 : Calling a Python script from R

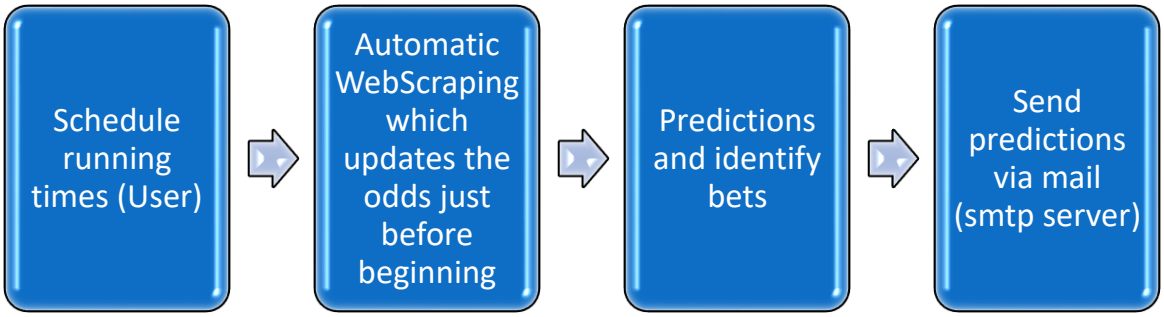


Figure 54 : Scheme of the current deployed pipeline. The user simply needs to schedule the running time and the script does everything – even sending predictions via mail.

XVII - CONCLUSION

I consider this project a great success, as I fulfilled many objectives that I had at the beginning of the project. This made me learn about various technical fields, from learning to conceive a whole pipeline, to building it : collecting the data, merging the different data sources, imagining the predictive variables, comparing different Machine Learning models along with hyper-parameters tuning, creating from scratch a metric to handle the many different strategies, create profitable stratgies, automate the whole process, and finally building a user-friendly interface. Along with my different researches, I also learned how to scrape data on various websites, in order to access historical results and closing odds for certain leagues.

Finally, this project is currently the biggest one that I have single-handled, and the fact that I had nobody to help me on a daily basis forced me to move towards various forms of learning, such as online MOOCs, forums and researches on the web. It was also the occasion to enforce my capabilities in coding, web scraping and finally in redaction through this document. Although this is not my first english report, it is still difficult to express myself concerning complicated notions, hence I hope this report was still understable.

XVIII - SOURCES

[1] GitHub description
https://github.com/
[2] Creative Commons licenses
https://creativecommons.org/licenses/
[3] Beating bookmakers with their own odds – and how the online sports betting market is rigged, Lisandro Kaunitz,Shenjun Zhong and Javier Kreiner
https://www.researchgate.net/publication/320296375_Beating_the_bookies_with_their_own_numbers_-_and_how_the_online_sports_betting_market_is_rigged
[4] Non-Linear classification as a tool for predicting tennis matches (Jakub Hostacny, 2018)
https://www.semanticscholar.org/paper/Non-Linear-Classification-as-a-Tool-for-Predicting-Hosta%C4%8Dn%C3%BD/f6e5c9cffa57e29f77f4d4c2a5a073b47943cdd6
[5] A statistical approach to sports betting (Anton Altmann, 2004)
https://1library.net/document/6qm6pe4y-a-statistical-approach-to-sports-betting.html
[6] Predicting football results using Machine Learning techniques. Corentin Herbinet, 2018.
https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1718-ug-projects/Corentin-Herbinet-Using-Machine-Learning-techniques-to-predict-the-outcome-of-profressional-football-matches.pdf
[7] Ruberry, Michael Edward. 2013. Prediction Markets: Theory and Applications. Doctoral dissertation, Harvard University.
http://nrs.harvard.edu/urn-3:HUL.InstRepos:11181140
[8] Favorite-longshot bias in European Football betting market : Differences between popular and non-popular football competitions (Daniël van Raaij, 2019, [xx])
https://www.semanticscholar.org/paper/%E2%80%98%E2%80%99Favorite-longshot-bias-in-European-Football-and-van/d782c15add5ee1a60a55e38776f481da74cf765b
[9] Bet on Sibyl application code
https://github.com/jrbadiabo/Bet-on-Sibyl
[10] Beating the bookmakers on tennis matches
https://github.com/edouardthom/ATPBetting
[11] Soccer betting
https://github.com/jiawei90/Soccer-Betting
[12] NBA organisation into divisions and conferences
https://buzzlesdotorg.files.wordpress.com/2015/11/carte-nba.jpg?w=650&h=388
[13] Bet365 website
www.bet365.com
[14] Python website
www.python.org
[15] Elo calculation process
https://en.wikipedia.org/wiki/Elo_rating_system#Mathematical_details
[16] FiveThirtyEight website
www.fivethirtyeight.com

[17] FiveThirtyEight's GitHub page
https://github.com/fivethirtyeight/data/tree/master/nba-forecasts
[18] FiveThirtyEigth ELO description
https://fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings/
[19] EigenVector decomposition for PCA
https://blog.clairvoyantsoft.com/eigen-decomposition-and-pca-c50f4ca15501
[20] Signal To Noise ratio explanation
https://www.sciencedirect.com/topics/computer-science/noise-to-signal-ratio
[21] Random Forests
https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[22] Support Vector Machines
https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
[23] & [30] nnet R package
https://cran.r-project.org/web/packages/nnet/nnet.pdf
[24] & [31] & [32] Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). Introduction to Statistical Learning, chapter 4.6. James et al.
http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf
[25] Upsampling and upweighting a dataset
https://towardsdatascience.com/handling-imbalanced-datasets-in-deep-learning-f48407a0e758
[26] Description of overfitting
http://cv.znu.ac.ir/afsharchim/AI/lectures/Decision%20Trees%203.pdf
[27] Explanation of the Random Forests heuristics : Introduction to Statistical Learning, chapter 8.2.2. James et al.
http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf
[28] Random Forest construction illustrated
https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d
[29] SVM decision process illustrated
https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
[23] & [30] nnet R package
https://cran.r-project.org/web/packages/nnet/nnet.pdf
[24] & [31] & [32] Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). Introduction to Statistical Learning, chapter 4.6. James et al.
http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf
[33] QDA and LDA illustration
http://www.phillypham.com/projects/Machine%20Learning%3A%20a%20Probabilistic%20Perspective%20in%20Python
[34] Football data source
http://www.football-data.co.uk/
[35] Tennis data source
http://www.tennis-data.co.uk/

[36] Monte-Carlo Simulations
https://www.palisade.com/risk/monte_carlo_simulation.asp

XIX - APPENDICES

- Appendix n°1 : NBA, football and tennis variables
- NBA variables (133)

List of predictive variables used in the NBA program :	Meaning (Statistics only take the current season into account)
B365_OT_H ,B365_OT_A	Hometeam and awayteam closing odds (bet365.com)
TotWins_home ,TotLose_home ,Pct_home	Hometeam overall record this season : Wins, Loss, % of wins
TotWins_away ,TotLose_away ,Pct_away	Awayteam overall record this season : Wins, Loss, % of wins
TotWinsHome_home ,TotLoseHome_home ,PctHome_home	Hometeam overall record when playing home this season : Wins, Loss, % of wins
TotWinsAway_away ,TotLoseAway_away ,PctAway_away	Awayteam overall record when playing away this season : Wins, Loss, % of wins
Last5_home, Last10_home	Number of wins for hometeam during last 5 and 10 games
Last5_away ,Last10_away	Number of wins for awayteam during last 5 and 10 games
H2Hv_home ,H2Hv_away , H2H_ratio	Heah-to-head record : games won by hometeam, games won by awayteam, ratio
Elo_home ,Elo_away	Elo rating before the game for hometeam and for awayteam
H2Hv_home_home ,H2Hv_away_away , H2H_home_away_ratio	Heah-to-head record when hometeam plays home : games won by hometeam, games won by awayteam, ratio
Serie_home ,Serie_home_home	Current streak for hometeam and when playing home (>0 means streak of victories, <0 means streak of defeats)
Serie_away ,Serie_away_away	Current streak for awayteam and when playing away (>0 means streak of victories, <0 means streak of defeats)
ROI_home ,ROI_away ,ROI_home_home ,ROI_away_away	Return on Investment if we always bet respectively on hometeam or on awayteam
Moy_scored_season_home_home ,Moy_against_season_home_home	Mean points scored/against for hometeam when playing home
Moy_scored_season_away_away ,Moy_against_season_away_away	Mean points scored/against for awayteam when playing away
Moy_scored_last5_home_home ,Moy_against_last5_home_home	Mean points scored/against for hometeam over last 5 games when playing home
Moy_scored_last5_away_away ,Moy_against_last5_away_away	Mean points scored/against for awayteam over last 5 games when playing away
Moy_scored_season_home ,Moy_against_season_home	Mean points scored/against for hometeam
Moy_scored_season_away ,Moy_against_season_away	Mean points scored/against for awayteam
Moy_scored_last5_home ,Moy_against_last5_home	Mean points scored/against for hometeam over last 5 games
Moy_scored_last5_away ,Moy_against_last5_away	Mean points scored/against for awayteam over last 5 games
rest_days_home ,rest_days_away	Number of rest days for each team
mean_spread_season_home ,mean_spread_season_away	Mean spread (=pts scored - pts against)
mean_spread_season_home_home ,mean_spread_season_away_away	Mean spread (=pts scored - pts against) when playing home/away respectively
mean_spread_last5_home ,mean_spread_last5_away	Mean spread (=pts scored - pts against) over last 5 games
mean_spread_last5_home_home ,mean_spread_last5_away_away	Mean spread (=pts scored - pts against) over last 5 games when playing home/away respectively
Elo_prob_home ,Elo_prob_away	Probabilities of victory implied by the ELO rating (see documentation for formula)
B365_prob_home ,B365_prob_away	Probabilities of victory implied by the bet365 odds

- Football variables (191)

List of predictive variables used in the football programs :	Meaning
B365_OT_H ,B365_OT_D ,B365_OT_A	Bet365 closing odds (Home/Draw/Away)
TotWins_home ,TotDraw_home ,TotLose_home	Hometeam record over the season (Wins/Draws/Loss)
Pct_wins_home ,Pct_draw_home ,Pct_lose_home	Percentages of Wins/Draws/loss for hometeam
TotWins_away ,TotDraw_away ,TotLose_away	Awayteam record over the season (Wins/Draws/Loss)
Pct_wins_away ,Pct_draw_away ,Pct_lose_away	Percentages of Wins/Draws/loss for awayteam
TotWinsHome_home ,TotDrawHome_home ,TotLoseHome_home	Hometeam record over the season when playing home (Wins/Draws/Loss)
PctWinsHome_home ,PctDrawHome_home ,PctLoseHome_home	Percentages of Wins/Draws/loss for hometeam when playing home
TotWinsAway_away ,TotDrawAway_away ,TotLoseAway_away	Awayteam record over the season (Wins/Draws/Loss) when playing away
PctWinsAway_away ,PctDrawAway_away ,PctLoseAway_away	Percentages of Wins/Draws/loss for awayteam when playing away
Last3_home_wins ,Last3_home_draw ,Last3_home_lose	Hometeam record over the last 3 games (Wins/Draws/Loss)
Last6_home_wins ,Last6_home_draw ,Last6_home_lose	Hometeam record over the last 6 games (Wins/Draws/Loss)
Last3_away_wins ,Last3_away_draw ,Last3_away_lose	Awayteam record over the last 3 games (Wins/Draws/Loss)
Last6_away_wins ,Last6_away_draw ,Last6_away_lose	Awayteam record over the last 6 games (Wins/Draws/Loss)
Serie_home_wins ,Serie_home_home_wins	Victory/Defeats streak for home team, + when playing home
Serie_away_wins ,Serie_away_away_wins	Victory/Defeats streak for away team, + when playing away
Moy_scored_season_home ,Moy_against_season_home	Average scored goals and goals taken by hometeam
Moy_scored_season_away ,Moy_against_season_away	Average scored goals and goals taken by awayteam
Moy_scored_last3_home ,Moy_against_last3_home	Average scored goals and goals taken by hometeam over last 3 games
Moy_scored_last3_away ,Moy_against_last3_away	Average scored goals and goals taken by awayteam over last 3 games
Moy_scored_season_home_home ,Moy_against_season_home_home	Average scored goals and goals taken by hometeam when playing home
Moy_scored_season_away_away ,Moy_against_season_away_away	Average scored goals and goals taken by awayteam when playing away
Moy_scored_last3_home_home ,Moy_against_last3_home_home	Average scored goals and goals taken by hometeam over last 3 home games
Moy_scored_last3_away_away ,Moy_against_last3_away_away	Average scored goals and goals taken by awayteam over last 3 away games
mean_spread_season_home ,mean_spread_season_away	Mean spread (goals scored - goals taken) for hometeam and for awayteam
mean_spread_season_home_home ,mean_spread_season_away_away	Mean spread (goals scored - goals taken) for hometeam and for awayteam, when playing home/away
mean_spread_last3_home ,mean_spread_last3_away	Mean spread (goals scored - goals taken) for hometeam and for awayteam over their last 3 games
mean_spread_last3_home_home ,mean_spread_last3_away_away	Mean spread (goals scored - goals taken) for hometeam and for awayteam over their last 3, when playing home/away
B365_prob_home ,B365_prob_away ,B365_prob_draw	Probabilities implied by the closing odds
RestDaysHome ,RestDaysAway	Rest days for home and away team
ROI_home_Wins ,ROI_away_Wins	Return On Investment if we always bet hometeam, if we always bet awayteam
ROI_home_home_wins ,ROI_away_away_wins	Return On Investment if we always bet hometeam when playing home, if we always bet awayteam when playing away
H2H_wins_home ,H2H_draws ,H2H_wins_away	Head-To-Head record
H2H_wins_home_home ,H2H_draws_homeaway ,H2H_wins_away_away	Head-To-Head record when hometeam is playing home
Pct_H2H_wins_home ,Pct_H2H_draws ,Pct_H2H_wins_away	Percentages of home wins/Draws/away wins during Head-To-Head
Pct_H2H_wins_home_home ,Pct_H2H_draws_homeaway ,Pct_H2H_wins_away_away	Percentages of home wins/Draws/away wins during Head-To-Head when hometeam is playing home
PctPointsHome ,PctPointsAway	Percentage of points earned by each team throughout this season (Victory = 3 points, Draw = 1 point)
PctPointsHome_home ,PctPointsAway_away	Percentage of points earned by each team, respectively playing home/away throughout this season

- Tennis variables (330)

List of predictive variables used in the tennis program :	Meaning
P1WinsEver ,P1LostEver	P1 record ever (W/L)
P1GamesWinsEver ,P1GamesLostEver	P1 games record ever (Won/Lost)
P1TbWinsEver ,P1TbLostEver	P1 tb record ever (Won/Lost)
P1ROIEver	P1 ROI ever
P2WinsEver ,P2LostEver	P2 record ever (W/L)
P2GamesWinsEver ,P2GamesLostEver	P2 games record ever (Won/Lost)
P2TbWinsEver ,P2TbLostEver	P2 tb record ever (Won/Lost)
P2ROIEver	P2 ROI ever
Serie_p1_Ever ,Serie_p2_Ever	W/L streaks for P1 & P2
P1SetsWinsEver ,P1SetsLostEver	P1 sets record ever (Won/Lost)
P2SetsWinsEver ,P2SetsLostEver	P2 sets record ever (Won/Lost)
P1WinsSeason ,P1LostSeason	P1 record season (W/L)
P1GamesWinsSeason ,P1GamesLostSeason	P1 games record season (Won/Lost)
P1TbWinsSeason ,P1TbLostSeason	P1 tb record season (Won/Lost)
P1ROISeason	P1 ROI season
P2WinsSeason ,P2LostSeason	P2 record season (W/L)
P2GamesWinsSeason ,P2GamesLostSeason	P2 games record season (Won/Lost)
P2TbWinsSeason ,P2TbLostSeason	P2 tb record season (Won/Lost)
P2ROISeason	P2 ROI season
Serie_p1_Season ,Serie_p2_Season	W/L streaks season for P1 & P2
P1SetsWinsSeason ,P1SetsLostSeason	P1 sets record season (Won/Lost)
P2SetsWinsSeason ,P2SetsLostSeason	P2 sets record season (Won/Lost)
P1WinsSurface ,P1LostSurface	P1 record surface (W/L)
P1GamesWinsSurface ,P1GamesLostSurface	P1 games record surface (Won/Lost)
P1TbWinsSurface ,P1TbLostSurface	P1 tb record surface (Won/Lost)
P1ROISurface	P1 ROI surface
P2WinsSurface ,P2LostSurface	P2 record surface (W/L)
P2GamesWinsSurface ,P2GamesLostSurface	P2 games record surface (Won/Lost)
P2TbWinsSurface ,P2TbLostSurface	P2 tb record surface (Won/Lost)
P2ROISurface	P2 ROI surface
Serie_p1_Surface ,Serie_p2_Surface	W/L streaks for P1 & P2
P1SetsWinsSurface ,P1SetsLostSurface	P1 sets record surface (Won/Lost)
P2SetsWinsSurface ,P2SetsLostSurface	P2 sets record surface (Won/Lost)
P1WinsSurfaceSeason ,P1LostSurfaceSeason	P1 wins record surface season (Won/Lost)
P1GamesWinsSurfaceSeason ,P1GamesLostSurfaceSeason	P1 games record Surface Season (Won/Lost)
P1TbWinsSurfaceSeason ,P1TbLostSurfaceSeason	P1 Tb record surface season (Won/Lost)
P1ROISurfaceSeason	P1 ROI surfaceseason
P2WinsSurfaceSeason ,P2LostSurfaceSeason	P2 wins record surface season (Won/Lost)
P1GamesWinsSurfaceSeason ,P1GamesLostSurfaceSeason	P1 games record Surface Season (Won/Lost)
P1TbWinsSurfaceSeason ,P1TbLostSurfaceSeason	P1 Tb record surface season (Won/Lost)
P1ROISurfaceSeason	P1 ROI surfaceseason
P2WinsSurfaceSeason ,P2LostSurfaceSeason	P2 wins record surface season (Won/Lost)
P2GamesWinsSurfaceSeason ,P2GamesLostSurfaceSeason	P2 games record Surface Season (Won/Lost)
P2TbWinsSurfaceSeason ,P2TbLostSurfaceSeason	P2 Tb record surface season (Won/Lost)
P2ROISurfaceSeason	P2 ROI surfaceseason
Serie_p1_SurfaceSeason ,Serie_p2_SurfaceSeason	W/L streaks surface season
P1SetsWinsSurfaceSeason ,P1SetsLostSurfaceSeason	P1 set record surface season
P2SetsWinsSurfaceSeason ,P2SetsLostSurfaceSeason	P2 set record surface season
P1WinsH2H ,P1LostH2H	P1 record Head to Head(Won/Lost)
P1GamesWinsH2H ,P1GamesLostH2H	P1 games record h2h (Won/Lost)
P1TbWinsH2H ,P1TbLostH2H	P1 tb record h2h (Won/Lost)
P1ROIH2H	P1 ROI H2H
P2WinsH2H ,P2LostH2H	P2 wins record H2H (Won/Lost)
P2GamesWinsH2H ,P2GamesLostH2H	P2 games record h2h (Won/Lost)
P2TbWinsH2H ,P2TbLostH2H	P2 tb record h2h (Won/Lost)
P2ROIH2H	P2 ROI H2H
Serie_p1_H2H ,Serie_p2_H2H	W/L streaks H2H
P1SetsWinsH2H ,P1SetsLostH2H	P1 sets record h2h (Won/Lost)
P2SetsWinsH2H ,P2SetsLostH2H	P2 sets record h2h (Won/Lost)
P1WinsH2H_Surface ,P1LostH2H_Surface	P1 record H2H surface (W/L)
P1GamesWinsH2H_Surface ,P1GamesLostH2H_Surface	P1 games record h2h surface (Won/Lost)
P1TbWinsH2H_Surface ,P1TbLostH2H_Surface	P1 tb record h2h surface (Won/Lost)
P1ROIH2H_Surface	P1 ROI H2H surface
P2WinsH2H_Surface ,P2LostH2H_Surface	P2 record H2H surface (W/L)
P2GamesWinsH2H_Surface ,P2GamesLostH2H_Surface	P2 games record h2h surface (Won/Lost)
P2TbWinsH2H_Surface ,P2TbLostH2H_Surface	P2 tb record h2h surface (Won/Lost)
P2ROIH2H_Surface	P2 ROI H2H surface
Serie_p1_H2H_Surface ,Serie_p2_H2H_Surface	W/L streaks H2H surface
P1SetsWinsH2H_Surface ,P1SetsLostH2H_Surface	P1 set record H2H surface (Won/Lost)
P2SetsWinsH2H_Surface ,P2SetsLostH2H_Surface	P2 set record H2H surface (Won/Lost)
P1WinsLast5 ,P1LostLast5	P1 record last5 (W/L)
P1GamesWinsLast5 ,P1GamesLostLast5	P1 games record last5 (Won/Lost)
P1TbWinsLast5 ,P1TbLostLast5	P1 tb record last5 (Won/Lost)
P1ROILast5	P1 ROI Last5
P1WinsLast10 ,P1LostLast10	P1 record last10 (W/L)
P1GamesWinsLast10 ,P1GamesLostLast10	P1 games record last10 (Won/Lost)
P1TbWinsLast10 ,P1TbLostLast10	P1 tb record last10 (Won/Lost)

P1ROILast10	P1 ROI Last10
P2WinsLast5 ,P2LostLast5	P2 record last5 (W/L)
P2GamesWinsLast5 ,P2GamesLostLast5	P2 games record last5 (Won/Lost)
P2TbWinsLast5 ,P2TbLostLast5	P2 tb record last5 (Won/Lost)
P2ROILast5	P2 ROI Last5
P2WinsLast10 ,P2LostLast10	P2 record last10 (W/L)
P2GamesWinsLast10 ,P2GamesLostLast10	P2 games record last10 (Won/Lost)
P2TbWinsLast10 ,P2TbLostLast10	P2 tb record last10 (Won/Lost)
P2ROILast10	P2 ROI Last10
P1SetsWinsLast5 ,P1SetsLostLast5	P1 sets record last5 (Won/Lost)
P2SetsWinsLast5 ,P2SetsLostLast5	P2 sets record last5 (Won/Lost)
P1SetsWinsLast10 ,P1SetsLostLast10	P1 sets record last10 (Won/Lost)
P2SetsWinsLast10 ,P2SetsLostLast10	P2 sets record last10 (Won/Lost)
P1WinsTournament ,P1LostTournament	P1 record tournament (W/L)
P1GamesWinsTournament ,P1GamesLostTournament	P1 games record tournament (Won/Lost)
P1TbWinsTournament ,P1TbLostTournament	P1 tb record tournament (Won/Lost)
P1ROITournament	P1 ROI tournament
P2WinsTournament ,P2LostTournament	P2 record tournament (W/L)
P2GamesWinsTournament ,P2GamesLostTournament	P2 games record tournament (Won/Lost)
P2TbWinsTournament ,P2TbLostTournament	P2 tb record tournament (Won/Lost)
P2ROITournament	P2 ROI tournament
P1SetsWinsTournament ,P1SetsLostTournament	P1 sets record tournament (Won/Lost)
P2SetsWinsTournament ,P2SetsLostTournament	P2 sets record tournament (Won/Lost)
EloDiff ,RankDiff	ELO difference, Rank difference
B365_prob_1 ,B365_prob_2	Winning probabilities implied by bet365.com closing odds
P1RatioEver ,P1RatioSetEver ,P1RatioGamesEver ,P1RatioTbEver	Various ratios
P1RatioSeason ,P1RatioSetsSeason ,P1RatioGamesSeason ,P1RatioTbSeason	Various ratios
P1RatioSurface ,P1RatioSetsSurface ,P1RatioGamesSurface ,P1RatioTbSurface	Various ratios
P1RatioSurfaceSeason ,P1RatioSetsSurfaceSeason ,P1RatioGamesSurfaceSeason ,P1RatioTbSurfaceSeason	Various ratios
P1RatioH2H ,P1RatioSetsH2H ,P1RatioGamesH2H ,P1RatioTbH2H	Various ratios
P1RatioH2H_Surface ,P1RatioSetsH2H_Surface ,P1RatioGamesH2H_Surface ,P1RatioTbH2H_Surface	Various ratios
P1RatioLast5 ,P1RatioSetsLast5 ,P1RatioGamesLast5 ,P1RatioTbLast5	Various ratios
P1RatioLast10 ,P1RatioSetsLast10 ,P1RatioGamesLast10 ,P1RatioTbLast10	Various ratios
P1RatioTournament ,P1RatioSetsTournament ,P1RatioGamesTournament ,P1RatioTbTournament	Various ratios
P2RatioEver ,P2RatioSetEver ,P2RatioGamesEver ,P2RatioTbEver	Various ratios
P2RatioSeason ,P2RatioSetsSeason ,P2RatioGamesSeason ,P2RatioTbSeason	Various ratios
P2RatioSurface ,P2RatioSetsSurface ,P2RatioGamesSurface ,P2RatioTbSurface	Various ratios
P2RatioSurfaceSeason ,P2RatioSetsSurfaceSeason ,P2RatioGamesSurfaceSeason ,P2RatioTbSurfaceSeason	Various ratios
P2RatioH2H ,P2RatioSetsH2H ,P2RatioGamesH2H ,P2RatioTbH2H	Various ratios
P2RatioH2H_Surface ,P2RatioSetsH2H_Surface ,P2RatioGamesH2H_Surface ,P2RatioTbH2H_Surface	Various ratios
P2RatioLast5 ,P2RatioSetsLast5 ,P2RatioGamesLast5 ,P2RatioTbLast5	Various ratios
P2RatioLast10 ,P2RatioSetsLast10 ,P2RatioGamesLast10 ,P2RatioTbLast10	Various ratios
P2RatioTournament ,P2RatioSetsTournament ,P2RatioGamesTournament ,P2RatioTbTournament	Various ratios
Best.of	Match format
C1 ,C2	P1 and P2 odds
Elo1 ,Elo2	P1 and P2 ELO ratings
Rank1 ,Rank2	P1 and P2 ranks

- Appendix n°2 : Football leagues scripts

Country	League	Script	Script finished ?
Argentina	SuperLiga	NEXT_PRED_ARG.R	✓
Australia	A-League	NEXT_PRED_AUS.R	✓
Austria	Tipico Bundesliga	NEXT_PRED_AUT.R	✓
Belgium	Jupiler Pro League	NEXT_PRED_B1.R	✓
Bielorussia	Vysshaya Liga	NEXT_PRED_BLR.R	✓
Brasil	Serie A	NEXT_PRED_BRA.R	✓
China	Super League	NEXT_PRED_CHN.R	✓
Denmark	Superliga	NEXT_PRED_DNK.R	✓
Faroe Islands	Premier League	NEXT_PRED_ILF.R	✓
Finland	Veikkausliiga	NEXT_PRED_FIN.R	✓
France	Ligue 1	NEXT_PRED_F1.R	✓
France	Ligue 2	NEXT_PRED_F2.R	✓
Germany	Bundesliga	NEXT_PRED_D1.R	✓
Germany	Bundesliga 2	NEXT_PRED_D2.R	✓
Ireland	Premier Division	NEXT_PRED_IRL.R	✓
Italy	Serie A	NEXT_PRED_I1.R	✓

Italy	Serie B	NEXT_PRED_I2.R	✓
Japan	J-League	NEXT_PRED_JPN.R	✓
Mexico	Liga MX	NEXT_PRED_MEX.R	✓
Netherlands	Eredivisie	NEXT_PRED_N1.R	✓
Norway	Eliteserien	NEXT_PRED_NOR.R	✓
Poland	Ekstraklasa	NEXT_PRED_POL.R	✓
Portugal	Liga NOS	NEXT_PRED_P1.R	✓
Romania	Liga 1	NEXT_PRED_ROU.R	✓
Russia	Premier League	NEXT_PRED_RUS.R	✓
Scotland	Scottish Premiership	NEXT_PRED_SC0.R	✓
South Korea	K-League	NEXT_PRED_KLEAGUE.R	✓
Spain	Liga Primera	NEXT_PRED_SP1.R	✓
Spain	Liga Secunda	NEXT_PRED_SP2.R	✓
Sweden	Allsvenskan	NEXT_PRED_SWE.R	✓
Switzerland	Super League	NEXT_PRED_SWZ.R	✓
Turkey	Süper Lig	NEXT_PRED_T1.R	✓
United Kingdom	Premier League	NEXT_PRED_E0.R	✓
United Kingdom	Championship	NEXT_PRED_E1.R	✓
USA	MLS	NEXT_PRED_USA.R	✓

- Appendix n°3 : Table of leagues under development

Sport	**League**	**Data Collected [15/04/2020]**	**Full program working**
Rugby	Top 14	✓	✓
	English Premiership	✓	✓
	Pro D2	✓	✓
	NRL	✓	✓
	Super Rugby	✓	✓
Australian Football	AFL	✓	✓
Basketball	Italian Lega A	✓	✓
	Spanish ACB	✓	✓
	Amercian NBA	✓	✓
	French Pro A (lnb)	✓	✓
	Russain Vtb United League	✓	✓
	Liga A (Argentina)	✓	✓
	NBL (Australia)	✓	✓

	NBB (Brazil)	✓	✓
	CBA (China)	✓	✓
	Euroleague	✓	✓
	Eurocup	✓	✓
	ABA league	✓	✓
	BBL (Germany)	✓	✓
	LKL (Lithuania)	✓	✓
	KBL (South Korea)	✓	✓
	Tbl (Turkey)	✓	✓
	Pro B (France)	✓	✓
	Champions League	✓	✓
	Europe Cup	✓	✓
	WNBA	✓	✓
Hockey	NHL	✓	✓
American Football	NFL	✓	
Baseball	MLB	✓	✓
	KBO (South-Korea)	✓	✓
	CPBL (Taiwan)	✓	✓
eSports	LoL Pro League	✓	✓
	LoL Continental League	✓	✓
	LoL Oceanic League	✓	✓
	ESL Pro League	✓	✓
	LoL Brazilian League	✓	✓
	Starcraft 2 South-Korean League	✓	✓
	LoL European Championship	✓	✓
	CS-GO Championship Series	✓	✓
	Overwatch League	✓	✓
	LoL Korean League	✓	✓
Tennis	ATP	✓	✓
	WTA	✓	