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.

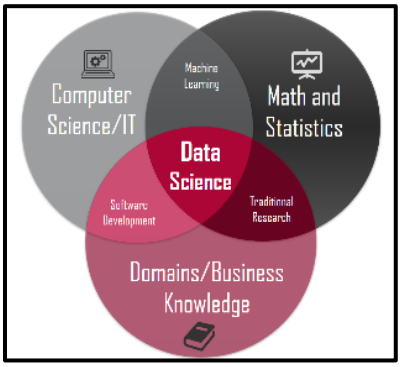


Figure 1 : Data Science's Venn Diagram

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.

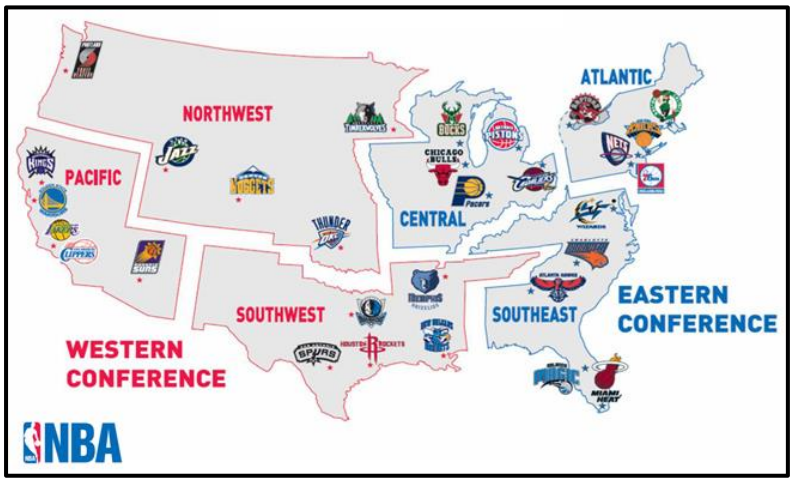


Figure 2 : NBA organisation into divisions and conferences [12]

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

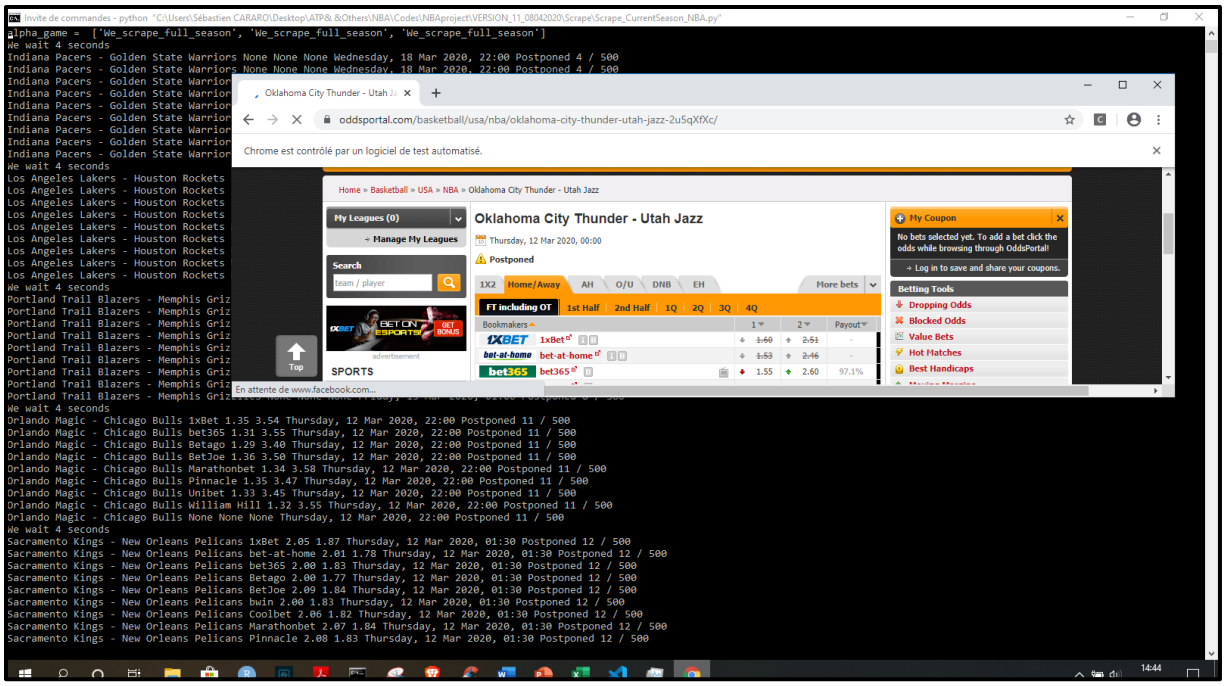


Figure 4 : Screenshot of the web scraping algorithm running

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>

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

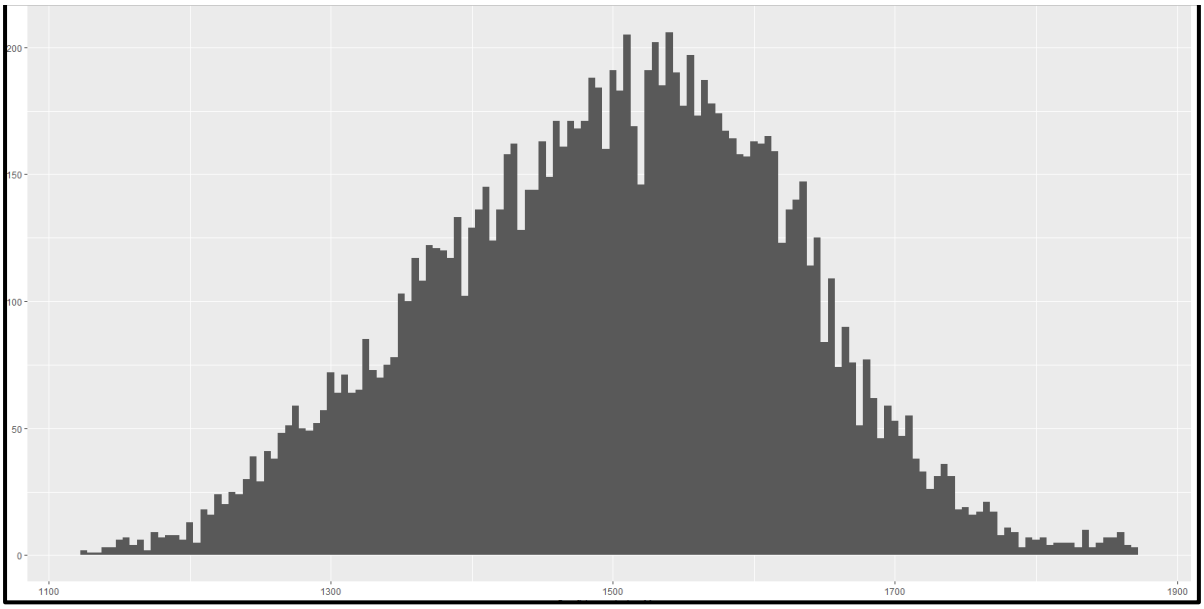
With K being a dynamics constant which I set to 32, and

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

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 :

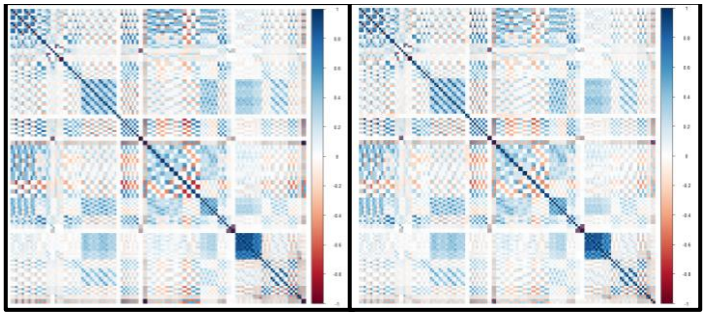
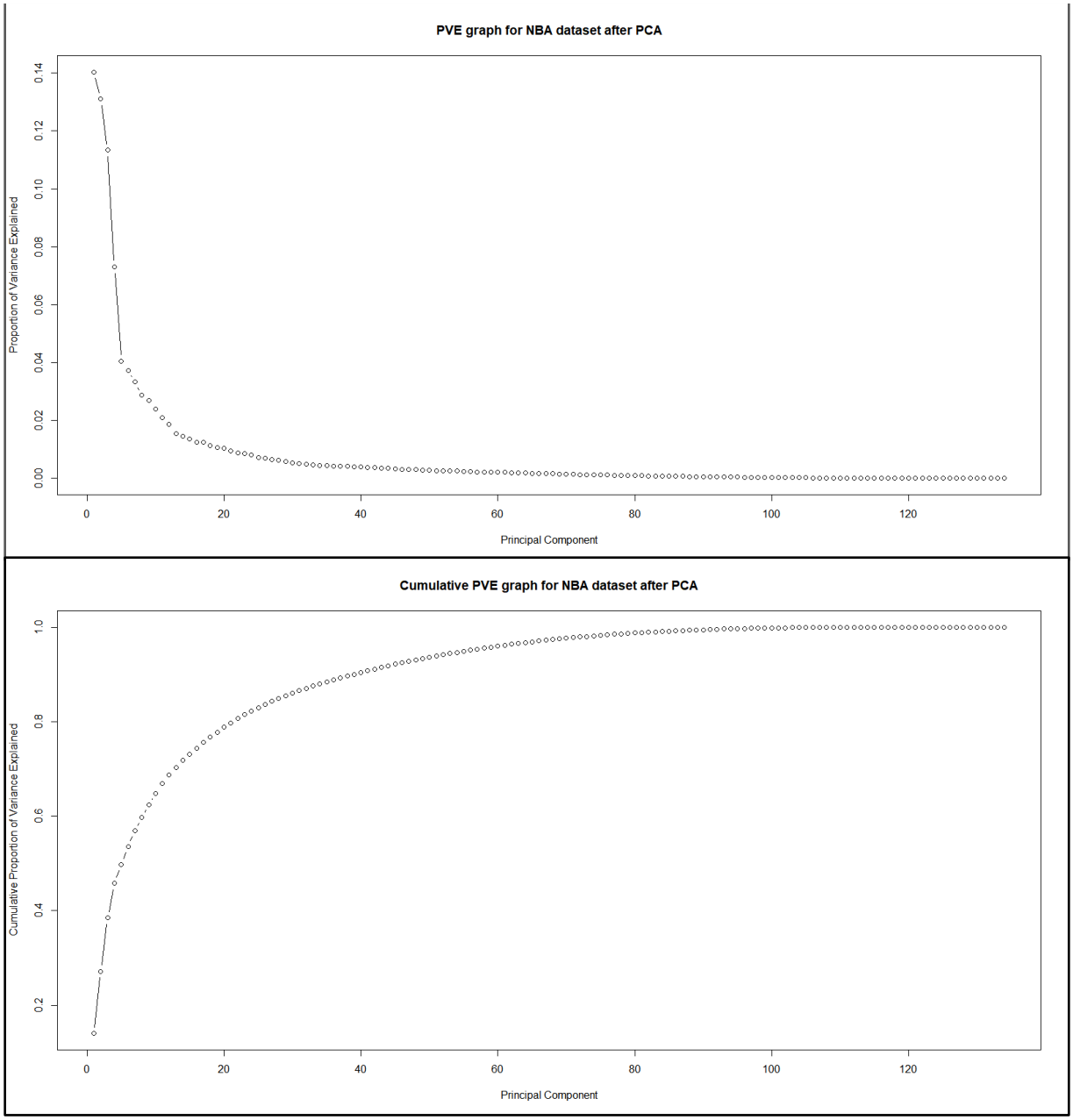


Figure 8 : Correlation matrices between predictive variables

- ✓ 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]).