# Mini Project: NBA stats analysis

Jonathan Guerne

HES-SO – MSE - MLBD - 25.06.2019

## General context and objectives

The purpose of this project is to "play" with NBA players stats by using different ML technique by focusing mainly on their position in the court. There is 5 position in the (current) NBA (PG, SG, SF, PF and C) but, recently many NBA analytics and enthusiast have begun to talk about “position-less” basketball. In other word we are entering a time redefining the importance of those position and maybe even trading them for new ones. This project is divided in 3 parts the first 2 are related.

### Position prediction

We will attempt to predict a player position on the court based only on his stats to assess if it is possible and if so easy to determine a player position only by analyzing data. If not, we will have an argument going in the direction of this new “position-less” NBA.

### Clustering

We will try to create our own new position by using clustering technique on the dataset. This will allow us to group player in a new way maybe more representative of what’s going on in the league currently.

### Space jam

More of a fun side project. In this part we will use stats to create the best casting of the new Space Jam movie. Space Jam [1] is a movie starring Michael Jordan and other elite NBA player of the late 1990’s. What player should we select to get the closest match in today league? This idea is based on a YouTube video [2].

[1] *Space Jam (1996) - IMDb*.

[2] FiveThirtyEight, *We used math to help LeBron cast « Space Jam 2 » | FiveThirtyEight*.

## Data description

As said in the introduction we use NBA player stats as our data. The dataset was found on the website: *www.basketball-reference.com* [3].

There is two type of player stats that we are interested in:

- **stats per game** containing 30 features [4]
- **advanced stats** containing 29 features [5]

The first on, stats per game, is the more common one. Any NBA fan will know them. They mainly give information about the players points, rebound and assists per game. We can also find the player shooting percentage, their fouls per game and other common data.

The advance stats, as their name may tell, are more in-depth data. They will almost only speak to expert that are able to understand for example, a player defensive and offensive impact trough them. They are way more difficult to read but might add a completely new layer of information.

Both are stored in *csv* files. They each count around 700 entries but they will need a bit of pre-processing

For the first and second part of the project we will use only current players data, but we will obviously need stats for the players in the Space Jam movie to be able to complete the last part. We were still able to find that information on the same website.

## Data pre-processing and feature extraction

Our first task was to remove any unwanted values in the dataset. Firstly, we looked to get rid of every uncomplete entry in our dataset. Pandas [6] (the library we choose to use) allow us to easily remove any entry containing NaNs which made it easy for use. In a second time we has to look more closely to the features themselves and find the one that would either not work in a ML model (for example non-numerical like name or team) or give information not directly related to the player way of playing in the court (age, minutes per game, game played, …).

We implanted those preprocessing function in a way that they could stack one after another thus making it easy to rapidly add or remove a feature from the original dataset.

```
df = remove_age(remove_team(remove_game(remove_min(df))))
```

Rater later in the project we attempted to merge the advanced stats and stats per game files. It is possible because they both give information about the same players. So, our task was to merge those 2 datasets into 1 based on player name (while making sure that we did not include redundant features!). After merging both files we ended up with 52 features per player.

[3] « Basketball Statistics and History ».
[4] « 2018-19 NBA Player Stats ».
[5] « 2018-19 NBA Player Stats: Advanced | Basketball-Reference.com ».
[6] « Python Data Analysis Library — pandas: Python Data Analysis Library ».

# ML-Tasks

## Position prediction

The purpose of the position prediction part of this project is to use Machine Learning to attempt to predict the position of a player based on his stats. How accurate can we get?
To reach a better understanding of this subject the problem will be subdivided in 3 parts:

- Using a TreeClassifier and regular stats per game
- Using a TreeClassifier and advanced stats per game
- Trying to find the best classifier while using merge stats

The idea behind using a TreeClassifier in both part 1 and 2 is to be able to get a better understanding of the model "way of thinking". To find the best classifier we selected a few models (with a little bit of manual parameters tuning) based on their classification score. We used the selected models to create a voting model

## Clustering

As we were, in this part, not able to use the player real position as our label we had to switch to an unsupervised approach. Our goal was to find the best fitting unsupervised model to split our dataset into K clusters of players. We used different model approach (and parameters) and a metric called "silhouette_score" [7] to determine which one would be the best to split the players into the "cleanest" kernels possible. The silouhette score tend to rewards samples to be close to each other within a cluster while punishing cluster for being to close for one another.

We had to select only models allowing to specify the number of cluster as it would be useful in the following treatment of our results.

## Space jam

We will use a neighbors-based learning model to determine witch current player is the most similar to a certain player from the original casting of the movie. KNN (or k nearest neighbors) is used to determine the K closest value in a dataset to an input entry.

Sklearn offer an implementation of the KNN algorithm with many parameters optimizable. One of these parameters is called the metrics end it describe the way the compute the distant between two neighbors. It is important to have use an appropriate metric. To find witch metric was the best in our case we did the following operations:

First, we loaded 2 datasets, one with every current player stat and one with the stats of every player of the 95-96 season (the year of the movie). Then we fitted our model on the 95-96 dataset and, for each eligible metric, selected the closest current player for each 95-96 player. This allow used the take the distance, **normalize it,** and then saved the average of the normalized distance. We were then able to select the best performing metric.

[7] « sklearn.metrics.silhouette_score — scikit-learn 0.21.2 documentation ».

# Experiments and Results

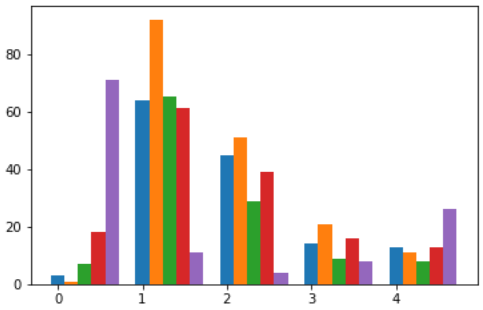
## Position prediction

The first part (TreeClassifier with stats per game) was very promising and not especially because of its classification performance. We use the ability of the TreeClassifier [8] to easily "explain" its classification decisions and we were able to generate a graph representing the classification process of the model. This was considered so interesting and worth sharing that we wrote and article about it and publish it on reddit [9]. We used the k-fold approach to compute the classification performance of our models and take advantage of sklearn cross_val_predict function [10] to generate a CM. We didn't tough it would be the case but the stat per game gave better results than the advanced stats (51% vs 49%).

In a second time we implemented our vote-based model with and without the merge dataset to see how much it would help. There was a significant increase in performance, but it seems that the major difference was the model rather than the dataset as the "merge-dataset voting model" ended up with 65% and the non-merge version with 64%.

## Clustering

Once a model and its parameters were selected (including K the number of clusters) we would then plot these new positions (1 to K) with the current position repartition (see the plot were each colored line represent the previous potions repartition at each new position). We could see that there was no (strong) correlation between our proposed position and the already existing ones.




We would attempt to look at each new position's strengths and weakness. Our conclusion was that the clustering did in a way make a cluster with only great players (the 4) having a lot of impact throughout the statsheet. We were also now able to look at the representation of each "new position player" trough each team of the league and paid mostly attention to the worst and best teams (trying to find patterns).

We used those patterns to create our very own good and bad teams and look at their roster to see, as an NBA fan, how we would consider them. We learnt that to predict a good team we had to force the minimum number of clusters to be high. Because to higher number of clusters the fewer players per cluster meaning reducing the "random-factor" when we select N players from a specific cluster to form our team.

## Space Jam

Having found the best available metric, we were able to analyze our model prediction for each star of the Space Jam movie. The results are obviously the most subjective of this whole project, because it is purely based on our capacity to compare two players. But, speaking as an NBA fan, it did work!

So, we went, further and implemented a way to recreate a whole team from a certain era (period) into another one. This ended up being on of the most fun functionality of this whole project.

[8] « sklearn.tree.DecisionTreeClassifier — scikit-learn 0.21.2 documentation ».
[9] Guerne, « Using Machine learning to predict NBA Position ».
[10] « sklearn.model_selection.cross_val_predict — scikit-learn 0.21.2 documentation ».

# Analysis and conclusions

Trough this whole project and each one of its aspect we discovered the amazing capability of rather simple ML models. These models once trained were able to produce classification completely valid (that make sense) even for an NBA enthusiast like myself.

## Position prediction

We decided that to be able to make a real conclusion we lacked some more information. The idea of position less basketball is becoming popular those past few years so to be able to determine if the NBA is evolving or not, we should compare out results on current data with result with data from past basketball eras.

So, we took the stats of the 95-96 season and compare their results with the current ones.

| YEAR | Results |
|---|---|
| 1995-1996 | 63% |
| 2018-2019 | 65% |

As we can see there is not a huge difference. But a more in-depth analysis would be needed to be able to get a conclusion. What we can say is that our model can make pretty good predictions on a 5 classes environment (random = 20%) thus heling us to understand that this idea of position-less basketball is either not today's ground truth or is not as different as we think from our classic 5 positions basketball.

## Clustering

To test the clustering of our program we tried to create our generated good team in a video game. This game would allow us to simulate a full season and see how our good team stands again the other NBA teams. This ended up being a **very** time-consuming idea… but it gave us interesting results as our team ended up with the best record (win-loss) in the league (69-13) and carried on to win the championship.

The strength of our team was that it was composed of 2 of the (arguably) best 5 players in the league, we also even had a couple of good players to help in both offense and defense. So, we understand that this may be called cheating but, in a way, it is where the NBA is going. The best way to win a championship is to put a/ a couple of superstar(s) around a core of good player on both side (offense/defense) of the court. You only have to look at the Toronto Raptors (the 2019 champions) and their recent acquisition of Kawhi Leonard to understand that. This means that we could in a way understand that there is a new "position" representing those superstars players capable of carrying a team to the championship.

## Space Jam

As suggested in the previous chapter it is difficult to make a conclusion on this part as it is subjective. The better metric to evaluate our performance would be the players stats themselves (the very same use to train the model so it's not necessary a good solution). One solution could be to get the opinion of an expert or of a group of people, if they think that our selected players make sense or not.

## Personal conclusion

It was amazing to work with technology that I like in a subject that passionate me. It has made my job clearly easier.

# References

« 2018-19 NBA Player Stats: Advanced | Basketball-Reference.com ». Consulté le 21 juin 2019. https://www.basketball-reference.com/leagues/NBA_2019_advanced.html.

« 2018-19 NBA Player Stats: Per Game ». Basketball-Reference.com. Consulté le 21 juin 2019. https://www.basketball-reference.com/leagues/NBA_2019_per_game.html.

« Basketball Statistics and History ». Basketball-Reference.com. Consulté le 21 juin 2019. https://www.basketball-reference.com.

FiveThirtyEight. *We used math to help LeBron cast « Space Jam 2 » | FiveThirtyEight*. Consulté le 21 juin 2019. https://www.youtube.com/watch?v=GzujfiDF4vU.

Guerne, Jonathan. « Using Machine learning to predict NBA Position », s. d. https://www.reddit.com/r/nba/comments/bjtazv/using_machine_learning_to_predict_nba_position/?utm_source=share&utm_medium=web2x.

« Python Data Analysis Library — pandas: Python Data Analysis Library ». Consulté le 21 juin 2019. https://pandas.pydata.org/.

« sklearn.metrics.silhouette_score — scikit-learn 0.21.2 documentation ». Consulté le 21 juin 2019. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

« sklearn.model_selection.cross_val_predict — scikit-learn 0.21.2 documentation ». Consulté le 21 juin 2019. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html.

« sklearn.tree.DecisionTreeClassifier — scikit-learn 0.21.2 documentation ». Consulté le 21 juin 2019. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

*Space Jam (1996) - IMDb*. Consulté le 21 juin 2019. http://www.imdb.com/title/tt0117705/fullcredits.