# Capstone Project

**Machine Learning Engineer Nanodegree**

**Arthur Elwing Torres**

**July 18th, 2020**

## Definition

### Project Overview

NBA basketball is one of the most famous and watched events on the planet. It is also frequently an object of statistical studies: lots of data can be found in the internet, as long as lots of related works. One common problem researchers try to solve is predicting NBA games outcomes. This topic is relevant to teams, for planning games ahead and resting players; to fans, for entertainment or betting; to broadcasting, for spreading information; and even for statistic researchers, due to the complex environment of the game and great potential for practical studies. It is also relevant for me, as a huge basketball fan and passionate for machine learning.

In this project, I developed an XGBoost model that can predict if a home team will win an NBA game regular season game, based on variables that describe each team's momentum, with around 66% accuracy. This model was developed in AWS SageMaker.

### Problem Statement

The goal of this project is to create, train and test a model which is able to predict the outcome of an NBA game. The steps for doing so are:

a) Download NBA games data
b) Select relevant features for the game outcome
c) Clean and preprocess data

d) Split data into train, validation and test datasets
e) Upload datasets to AWS S3
f) Select algorithm, create and train model
g) Test model

The expectation of this project is that the model is able to achieve a prediction accuracy that follows or even beat the standards of other models found in the literature, which are discussed in the benchmark session.

## Metrics

The metric that was used in this project is accuracy, due to its simple comprehension and potential to describe if the model is performing well, compared to other models. The accuracy equation is as follows:

$$accuracy = \frac{TruePositives + TrueNegatives}{TotalNumberOfPoints}$$

# Analysis

## Data Exploration and Visualization

The dataset used in this project was “NBA Enhanced Box Score and Standings (2012 - 2018)”, available in Kaggle[1]. This set contains several data from NBA season 2012-2013 to season 2017-2018. Following the goal of this project, the Team Box Score dataset was chosen as the main source of data, as it contains several basic and advanced statistics from each team in each game. Below there is an example of one

[1]Rossotti, P. (2017, November). NBA Enhanced Box Score and Standings (2012 - 2018), Version 27. Retrieved June 29, 2020 from https://www.kaggle.com/pablote/nba-enhanced-stats/.

game data. Note that there are two rows for each game, where both have the same data, but they are presented in different order – in the first row, the ‘team" prefix represents the “away" team, and the “oppt" prefix represents the “home” team. In the second row, ‘team" is the “home" team, and “oppt" is the “away" team. The meaning of each abbreviation and a brief description of each statistic can be found in the file *datasets_4389_168008_metadata_teamBoxScore.pdf*, which comes with the dataset.

| **gmDate** | **gmTime** | **seasTyp** | **offLNm1** | **offFNm1** | **offLNm2** | **offFNm2** | **offLNm3** | **offFNm3** |
|---|---|---|---|---|---|---|---|---|
| 10/30/2012 | 19:00 | Regular | Brothers | Tony | Smith | Michael | Workman | Haywoode |
| 10/30/2012 | 19:00 | Regular | Brothers | Tony | Smith | Michael | Workman | Haywoode |
| | | | | | | | | |
| **teamAbbr** | **teamConf** | **teamDiv** | **teamLoc** | **teamRslt** | **teamMin** | **teamDayOff** | **teamPTS** | **teamAST** |
| WAS | East | Southeast | Away | Loss | 240 | 0 | 84 | 26 |
| CLE | East | Central | Home | Win | 240 | 0 | 94 | 22 |
| | | | | | | | | |
| **teamTO** | **teamSTL** | **teamBLK** | **teamPF** | **teamFGA** | **teamFGM** | **teamFG%** | **team2PA** | **team2PM** |
| 13 | 11 | 10 | 19 | 90 | 32 | 0.3556 | 58 | 24 |
| 21 | 7 | 5 | 21 | 79 | 36 | 0.4557 | 59 | 29 |
| | | | | | | | | |
| **team2P%** | **team3PA** | **team3PM** | **team3P%** | **teamFTA** | **teamFTM** | **teamFT%** | **teamORB** | **teamDRB** |
| 0.4138 | 32 | 8 | 0.25 | 20 | 12 | 0.6 | 18 | 21 |
| 0.4915 | 20 | 7 | 0.35 | 22 | 15 | 0.6818 | 18 | 36 |
| | | | | | | | | |
| **teamTRB** | **teamPTS1** | **teamPTS2** | **teamPTS3** | **teamPTS4** | **teamPTS5** | **teamPTS6** | **teamPTS7** | **teamPTS8** |
| 39 | 24 | 15 | 23 | 22 | 0 | 0 | 0 | 0 |
| 54 | 31 | 19 | 24 | 20 | 0 | 0 | 0 | 0 |
| | | | | | | | | |
| **teamTREB%** | **teamASST%** | **teamTS%** | **teamEFG%** | **teamOREB%** | **teamDREB%** | **teamTO%** | **teamSTL%** | **teamBLK%** |
| 41.9355 | 81.25 | 0.4251 | 0.4 | 33.3333 | 53.8462 | 11.6279 | 12.3678 | 11.2434 |
| 58.0645 | 61.1111 | 0.53 | 0.5 | 46.1538 | 66.6667 | 19.1466 | 7.8704 | 5.6217 |
| | | | | | | | | |
| **teamBLKR** | **teamPPS** | **teamFIC** | **teamFIC40** | **teamOrtg** | **teamDrtg** | **teamEDiff** | **teamPlay%** | **teamAR** |
| 17.2414 | 0.9333 | 67.25 | 56.0417 | 94.4447 | 105.6882 | -11.2435 | 0.3765 | 18.8679 |
| 8.4746 | 1.1899 | 74 | 61.6667 | 105.6882 | 94.4447 | 11.2435 | 0.439 | 16.7072 |
| | | | | | | | | |
| **teamAST/TO** | **teamSTL/TO** | **opptAbbr** | **opptConf** | **opptDiv** | **opptLoc** | **opptRslt** | **opptMin** | **opptDayOff** |
| 2 | 84.6154 | CLE | East | Central | Home | Win | 240 | 0 |
| 1.0476 | 33.3333 | WAS | East | Southeast | Away | Loss | 240 | 0 |
| | | | | | | | | |

| opptPTS | opptAST | opptTO | opptSTL | opptBLK | opptPF | opptFGA | opptFGM | opptFG% |
|---|---|---|---|---|---|---|---|---|
| 94 | 22 | 21 | 7 | 5 | 21 | 79 | 36 | 0.4557 |
| 84 | 26 | 13 | 11 | 10 | 19 | 90 | 32 | 0.3556 |

| oppt2PA | oppt2PM | oppt2P% | oppt3PA | oppt3PM | oppt3P% | opptFTA | opptFTM | opptFT% |
|---|---|---|---|---|---|---|---|---|
| 59 | 29 | 0.4915 | 20 | 7 | 0.35 | 22 | 15 | 0.6818 |
| 58 | 24 | 0.4138 | 32 | 8 | 0.25 | 20 | 12 | 0.6 |

| opptORB | opptDRB | opptTRB | opptPTS1 | opptPTS2 | opptPTS3 | opptPTS4 | opptPTS5 | opptPTS6 |
|---|---|---|---|---|---|---|---|---|
| 18 | 36 | 54 | 31 | 19 | 24 | 20 | 0 | 0 |
| 18 | 21 | 39 | 24 | 15 | 23 | 22 | 0 | 0 |

| opptPTS7 | opptPTS8 | opptTREB% | opptASST% | opptTS% | opptEFG% | opptOREB% | opptDREB% | opptTO% |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 58.0645 | 61.1111 | 0.53 | 0.5 | 33.3333 | 66.6667 | 19.1466 |
| 0 | 0 | 41.9355 | 81.25 | 0.4251 | 0.4 | 46.1538 | 53.8462 | 11.6279 |

| opptSTL% | opptBLK% | opptBLKR | opptPPS | opptFIC | opptFIC40 | opptOrtg | opptDrtg | opptEDiff |
|---|---|---|---|---|---|---|---|---|
| 7.8704 | 5.6217 | 8.4746 | 1.1899 | 74 | 61.6667 | 105.6882 | 94.4447 | 11.2435 |
| 12.3678 | 11.2434 | 17.2414 | 0.9333 | 67.25 | 56.0417 | 94.4447 | 105.6882 | -11.2435 |

| opptPlay% | opptAR | opptAST/TO | opptSTL/TO | poss | pace |
|---|---|---|---|---|---|
| 0.439 | 16.7072 | 1.0476 | 33.3333 | 88.9409 | 88.9409 |
| 0.3765 | 18.8679 | 2 | 84.6154 | 88.9409 | 88.9409 |

Advanced statistics generally describe how each team performed in a specific field of the game. A high *assists over turnovers ratio* (AST/TO), for example, says that a team has committed few turnovers and/or has passed the ball well and often created score opportunities. In addition, advanced statistics basically derive from the basic ones, but they can also take into account both teams statistics in their calculations (e.g.: *total rebounding percentage*, or TREB%, takes into account both teams total number of rebounds in its calculation). So, logically speaking, high advanced statistics should translate into a team playing well, whereas poor team play probably produces low advanced statistics.

Due to this fact, advanced statistics were used in this project as the main input of the model. However, the statistics of a game can only be known after it has finished, therefore previous data must be considered in order to make predictions about a game yet to happen. But, another question comes up: how many games prior to the current one may describe accurately the team’s momentum, generating information that could effectively help predict a game outcome?

“Last 10”, a common statistic found in almost all NBA standings, is usually used to show a team’s current momentum. However, it only says how many wins and losses each team had in their last 10 games. In other words, it only shows the outcome of playing well or not, which is the win or loss itself, not diving into deeper details of a team’s performance. Based on this common aspect, a 10-game span average was used to determine a team’s momentum carrying into the current game. In addition, weights were assigned to each of the 10 games, where the games that happened closer to the current game have greater weights, meaning that they have higher impact in describing the team’s momentum.

This design creates a limitation: only games where both teams have already played 10 games can be used in the model. Games that don’t follow this condition won’t have a prior 10-game span to calculate advanced statistics averages. This also creates another requisite – the Standings dataset needs to be used in order to check how many games each team has played at a specific date. A sample of this dataset can be found below. The meaning of each abbreviation and a brief description of each statistic can be found in the file *datasets_4389_168008_metadata_standing.pdf*, which comes with the dataset.

| stDate | teamAbbr | rank | rankOrd | gameWon | gameLost | stk | stkType | stkTot | gameBack | ptsFor | ptsAgnst | homeWin | homeLoss | awayWin | awayLoss | confWin | confLoss | lastFive | lastTen | gamePlay | ptsScore | ptsAllow | ptsDiff | opptGmPlay | opptGmWon | opptOpptGmPlay |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10/30/2012 | ATL | 3 | 3rd | 0 | 0 | - | - | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10/30/2012 | BKN | 3 | 3rd | 0 | 0 | - | - | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| opptOpptGmWon | sos | rel%Indx | mov | srs | pw% | pyth%13.91 | wpyth13.91 | lpyth13.91 | pyth%16.5 | wpyth16.5 | lpyth16.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 82 | 0 | 0 | 82 |
| 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 82 | 0 | 0 | 82 |

Two other variables that may prove to be useful in this project are *days off* (daysOff), which represents the number of days a team has rested between the previous game and the current one, and *winning percentage*. It is expected that both proportionally influence the chances of a team winning the game (teams well rested usually have more energy in the game, and teams with high winning percentage usually have been winning more often and probably being playing well). The second one also required the Standings dataset in order to be calculated.

The summary of the features that were used in this project is as follows (for both the home and away teams):

[“Winning Percentage”, “daysOff”, ‘tREB%”, “ASST%”, ‘tS%”, “EFG%”, “OREB%”, “DREB%”, ‘tO%”, ‘sTL%”, “BLK%”, “BLKR”, “PPS”, “FIC”, “FIC40”, “Ortg”, “Drtg”, “EDiff”, “Play%”, “AR”, “AST/TO”, ‘sTL/TO”]

## Algorithms and Techniques

As the main information in this project can be categorized as tabular data, several algorithms may be used in order to create a model. In this specific case, XGBoost was selected due to its recent popularity and good performance with this kind of data.

Also, a dimensionality reduction was performed, in order to give greater importance (weights) to more relevant features of the data. This was done using a Principal Component Analysis (PCA). Inherently, the data was also needed to be scaled, and, in this particular case, with a minimum-maximum scaler.

The platform used in this project was Amazon AWS SageMaker, which contains all the tools necessary to preprocess the data, create and train a model, and then deploy and evaluate it.

## Benchmark

As there are lots of NBA data available to be used as input for machile learning models, different solutions and designs can be found. Some of them fall short on accuracy, but other good models can be said to predict the correct game winner with around 60-70% accuracy. Fivethirtyeight, for example, a statistics company, wrote an article in 2015 where it says that it correctly predicts the winner of NBA games 63.7% of the times[2].

# Methodology

## Data Pre-Processing

As discussed in the previous sections, lots of calculations and data manipulation is needed for this problem, turning this part into arguably the longest in this project (Pandas was heavily used). In the following lines, a summary of each step is presented (if the dataset is not specified, please assume that it is the Team Box Score Dataset).

a) Remove duplicate lines for each game – keep only the second row for each game (where ‘team" prefix represents the home team, and “oppt" prefix represents the away team);
b) Create column “season”, which represents the season where the game is part of. This was only used to check the number of games per season;

[2] FiveThirtyEight. How Our 2015-16 NBA Predictions Work. Retrieved June 29, 2020 from https://fivethirtyeight.com/features/how-our-2015-16-nba-predictions-work/

c) Create columns “teamWins”, “teamLosses”, “opptWins” and “opptLosses”. These represent the number of wins and losses by each team at the game day. This information is found in the Standings Dataset, by looking at the standings date one day before the game date (standings are calculated at the end of the day, after all games have finished);

d) Calculate 10-game weighted averages for each target statistic for each team – in order to accomplish this, the code creates a column for each of these statistics, plus the prefix “AVG”. Then, for each game, it looks at the previous 10 games of each team and calculates the weighted averages, then stores the result;

e) Drop games which happened when either team hadn’t played 10 games yet;

f) Calculate winning percentages at the date of each game, for each team (wins over sum of wins and losses);

g) Create target variable – if “teamRslt” is “Win”, it means that the home team has won the game (1 – home team won; 0 – away team won);

h) Drop unnecessary columns;

i) Split dataset into train and test sets, using randomness and stratified split;

j) Apply data scaling – in this project, SKLearn’s MinMaxScaler was used to keep train data values between 0-1;

k) Apply PCA dimensionality reduction – in this project, a 95% target variance was selected, reducing the number of features in the dataset from 44 to 18;

l) Finally, split the reduced and scaled train dataset into 2 – train and validation sets.

## Implementation

With data ready, the process of creating and training a model in SageMaker is pretty straightforward. First, train and validation datasets were uploaded to Amazon AWS S3. Then, the XGBoost model was constructed with the following Hyperparameters:

```
xgb.set_hyperparameters(max_depth=10,
    eta=0.5,
    gamma=3,
    min_child_weight=2,
    subsample=0.5,
    objective="binary:logistic",
    early_stopping_rounds=20,
    num_round=500)
```

Then, the model can be trained using the uploaded datasets. However, there is a problem with this approach: how can the user know if the chosen Hyperparameters are adequate, if the model has room for improvement, and how to adjust them with efficiency? This will be discussed in the next section.

## Refinement

Instead of guessing hyperparameters, SageMaker offers the feature called *Hyperparameter Tuner*, which can train multiple jobs in parallel, with different values for hyperparameters, based on a ranging input. This allows the user to get the best model without the need of training different models by hand.

This is how the Hyperparameter Tuner was configured:

```
HyperparameterTuner(estimator = xgb,
objective_metric_name = "validation:rmse",
    objective_type = "Minimize",
    max_jobs = 50,
    max_parallel_jobs = 3,
    hyperparameter_ranges = {
        "max_depth": IntegerParameter(2, 20),
        "eta"      : ContinuousParameter(0.05, 0.95),
        "min_child_weight": IntegerParameter(1, 10),
        'subsample": ContinuousParameter(0.05, 0.95),
        "gamma": ContinuousParameter(0, 20),
})
```

# Results

## Model Evaluation and Validation

The XGBoost model which achieved the best training job according to Sagemaker Hyperparameter tuning job has the following characteristics:

| | | |
|---|---|---|
| **_tuning_objective_metric** | FreeText | validation:rmse |
| **_tuning_objective_metric value achieved** | | 0.460509 |
| **early_stopping_rounds** | FreeText | 20 |
| **eta** | Continuous | 0.05666330730415221 |
| **gamma** | Continuous | 2.0281117607396926 |
| **max_depth** | Integer | 3 |
| **min_child_weight** | Integer | 8 |
| **num_round** | FreeText | 500 |
| **objective** | FreeText | binary:logistic |
| **subsample** | Continuous | 0.7131571431265465 |

Using SageMaker's deployment feature, the model above was deployed. Then, it became possible to make the predictions for the test dataset. However, before that, this dataset was scaled and reduced using, respectively, the same MinMaxScaler and PCA created in the training step (in this case, both did not fit the test data, but instead just transformed it).

The predictions are received as a comma-separated string, so it can be transformed in an array with Numpy. The values are float numbers ranging from 0 to 1. To determine the winner, these values are rounded to 1 if greater than 0.5, and to 0 if lower. Then, Using SKLearn's accuracy_score, this array is compared with the array of test labels, and the output is the accuracy of the model applied in this dataset. The result obtained was 0.6574, which means the model predicted if the home team will win the NBA game with approximately 66% accuracy.

The model also seems to be fairly consistent, as the aforementioned result was also approximately obtained after other four times the notebook was executed from scratch, ranging between 65%-67% (this means that, even though data for train, validation and test were different each time, due to the fact that the train_test_split function was called with randomness, the result was still consistently close every time).

### Justification

Comparing the result obtained by this model with the accuracies described in the Benchmark section, it is possible to state that the model performed really well for the problem, falling in the threshold of 60-70% accuracy. It is also possible to infer that the combination of advanced statistics throughout a 10-game span with the amount of days off before the game have a decent impact in the outcome of an NBA regular season game. Furthermore, even though other models may consider different variables and approaches to determine the winner of a game, they can incorporate team's momentum, days off and winning percentage to their data structure and check if they are able to get an increase in accuracy.

## Conclusion

### Reflection

In this project, as it probably happens with the majority of machine learning problems, the most challenging part was dealing with the data prior to the model development itself. Lots of preprocessing and cleaning steps had to be performed in order to create a dataset with consistent values. Due to this, I cannot stress enough how important it is to a) regularly visualize and check the data across the steps, and b) test the input and the output with expected results. This will avoid problems during further steps and even save lots of time if an issue happens. For example, I only found out that the 2012-2013 NBA season had one game cancelled after testing the number of games of each season.

After preprocessing the data, the next steps were pretty straightforward due to the practicality of SageMaker. Uploading the datasets to S3, creating, training and testing the model became easy tasks, and I could take more time analyzing the data and the model than actually developing the model.

## Improvement

This model still has huge potential for improvements. Here are a few:

a) Test different weights for the last 10 games, generating new different 10-game averages;
b) Include data from other seasons, to check if model is valid across other seasons in the NBA;
c) Include other relevant data that may affect the outcome of a game. E.g. player injuries or suspensions, team strength, among others;
d) Develop a strategy to accommodate predictions for games where both teams haven’t played 10 games yet;
e) Test different XGBoost hyperparameters;
f) Test other algorithms that are suitable for this data, such as LinearLearner. Even though XGBoost is a well-known good performer for tabular data, nothing in this project attaches the problem to this algorithm.