Data Science Workshop - NBA Free Throws Prediction

Guy Azov
guyazov@mail.tau.ac.il

Brian Gordon
briangordon@mail.tau.ac.il

Tomer Moinian
moinian@mail.tau.ac.il

Shachar Banayan
shacharb@mail.tau.ac.il

Abstract

In this paper we present a project as part of a Data Science Workshop at Tel-Aviv University [1]. *Our main purpose was to complete an entire end to end Data Science pipeline start from data collecting through data cleaning and exploring, until the modeling and predictions. As we will explain later, we found out that since a free throw its a statistical event, machine learning models having troubles to predict its outcome with high accuracy. In addition, since our data was collected from pro NBA players performances, it consisted of 76% made shots, caused our data set to be an imbalance one. As a result all of our primary models tend to label all shots as scores. Therefore, our point of focus in this work was dealing with that phenomenon and try increasing the recall parameter on the missed shots. Our source code and Jupyter notebook are available via* `https://github.com/guyazov/DS_Workshop`.

1. Introduction

We have based on a data set which contained 600K free-throws of NBA players and uploaded to Kaggle by Sebastian-Mantey [1]. The data was scraped from the ESPN.com website which belongs to the entertainment and sports programming network. In this work we will try to predict whether or not a free-throw will be made according to some parameters such as: position, score difference, shooting hand, height and more.

2. Related Work

There are 25 kernels on Kaggle related to this data set [2]. Most of the kernels summarize, analyze and visualize the data and do not try to predict anything. The kernel [3], engages with the questions: How does the shooting percent develop over time? Does it decrease as the game approaches its final minutes,when pressure is higher? Does it increase thanks to players being warmer, or alternatively - better shooters take the ball? They concluded that in the end of every quarter of a game, there is an increase in the number of free-throws and in the free-throws scoring percentage. They also found that most of the throws in the end of every quarter were performed by better shooters. Nevertheless the absolute time of the throw had more effect on the outcome than who threw the ball. Another interesting kernel is [4], in which they try to find the best players in the data set with statistical and probability methods such as complete pooling, and find the probability for every player to succeed in the free-throw. Outside Kaggle, we found the article [5] that investi-

[1] https://www.cs.tau.ac.il/ amitsome/datascience/201920/

gated the relationship between mindfulness, a pre-shot routine, and basketball free-throw scoring percentage. It suggested that the combination of mindfulness levels, skill level (practice free-throw percentage), and competitive experience (year in school), had contribution to the prediction of a competitive free throw percentage.

3. Data Collection

The original database consisted of 11 parameters and contained 618,019 free-throws - 575,893 were taken from regular games and 42,126 from the playoffs. There were 12,874 unique games, between 2006 and 2016, whom the data was collected from. In order to build more comprehensive model, we used an open source python library called 'PandasBasketball', and a web scrapper to get more players statistics from https://www.basketball-reference.com website. For each player we added it's position, FG%, 3P%, FT%, Height, Weight, Shooting hand and draft rank.

4. Data Cleaning

As we observed the collected data,we encountered some players that had missing values and weird behaviour(e.g. a player we know played in some position appeared in a different one etc). We found out that these phenomena were caused by:

- Several names to the same player. For instance - 'Jose Juan' and 'J.J Barea'. More radical examples are players with nicknames or that changed their name like 'Metta World' who was born as 'Ron Artest'.
- Several names to the same team. For example - 'PHO' and 'PHX' which represents Phoenix.
- Some players have not been drafted, such as 'Bobby Brown'.
- A player played in multiple teams in the same year.
- Several players have the same exact name.
- For some players the data on the website was missing.
- For some players the data on the website was missing.
- Some players have never thrown a 3-pointer.

In order to fix those issues we had to manually edit some of the players stats, using our knowledge in the problem domain. Another effort was made fixing some bugs in the 'PandasBasketball' library which caused some of the phenomena described.

4.1. Handle Missing Values

The only missing values left at this point were some of the 3P%, FT% and draft ranks. Since the missing data was only less than 8% we filled the 3P% and the FT% with their median value. For the draft ranks missing values we used the rank '61' since there were only 60 players which been drafted every year.

4.2. Sanitize Some Columns

We converted the shooting hand and the playoffs to binary variables while converting the position, height and weight variables to numeric ones. [1]

[1] We used our knowledge on the position distribution and appropriately determined the numeric values.

5. Feature Engineering

Construction of the following features has been made:

- The second, minute, absolute minute and absolute time of the shot.
- The team the player played in.
- The score difference (not an absolute value) in the moment of the shot.

5.1. Prevent Data Leakage

The original FT% feature was obtained from the **entire career** of a player. So, using this feature means using future information which means a leak. Hence, the FT% column has been fixed so that the FT% feature represents player's past seasons only.

5.2. Advanced Feature engineering

During our work and as we will describe later we came to an understanding that our data is not separable. In order to overcome this obstacle we added more features - which shot (first/second/third) was taken at the time and what was the outcome of previous shots(in case there were such shots). Since we see it as unnecessary to repeat the process with and without these parameters we refer them as part of our database from here on.

6. Data Analysis

Our data consist of:

- 76% of made shots.
- 91% of right handed players.
- Average weight of 101.23 Kg
- Average height is 200.3 cm.
- 7% playoffs game.
- 21% of the players are Small Forwards, 19% are Centers, Power Forwards, Point guards and Shooting guards holds 20% of data-set each.

Deeper review of some throws information appears in figures 1, 2

Figure 1. Throws Information

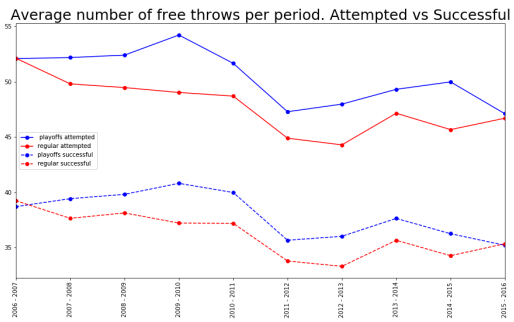

Figure 2. Shots Per Season Information

An analysis of the success per position can be found in 3

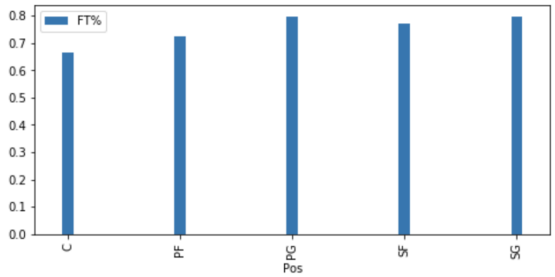

Figure 3. Average Free Throw % By Position

We have also examined the distribution of some of the parameters as in 4, 5

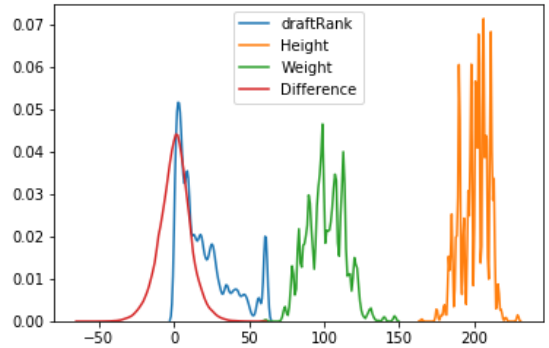

Figure 4. Distributions part A

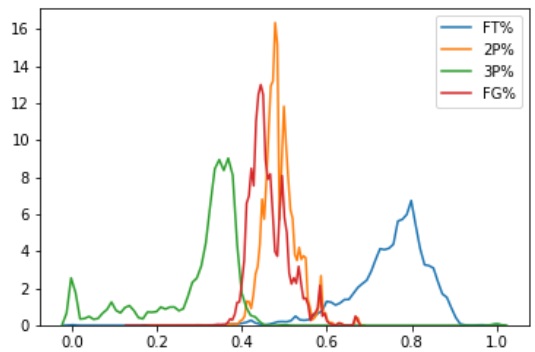

Figure 5. Distributions part B

Finally, finding the correlations between all parameters 6 had a significant influence on the models behaviour understanding.

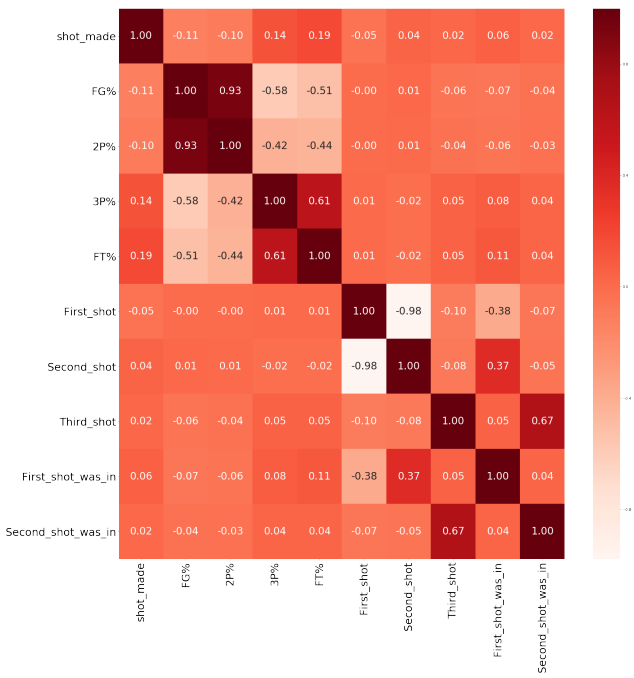

Figure 6. Correlations

7. Problem Formulation

Our main target is to predict whether a player will score or not a certain free throw (an online prediction) according to some given parameters. We believe that this information could benefit with a basketball coach while considering which player he wants or needs on the court in a certain moment.

8. Initial Models And Results

8.1. Baseline Model

A basic reference model which labeled every throw taken by player with FT% higher than 0.5 as a **score** was initiated.

- Model's Performance:

	Precision	Recall	f1-score
0	0.52	0.03	0.06
1	0.77	0.99	0.86
accuracy			0.76
macro avg	0.64	0.51	0.46
weighted avg	0.71	0.76	0.67

8.2. Logistic Regression Model

One basic ML model for a classification task is the logistic regression. It's results, as we can see are very similar to the baseline model.

- Model's Performance:

	Precision	Recall	f1-score
0	0.00	0.00	0.00
1	0.76	1.0	0.86
accuracy			0.76
macro avg	0.38	0.50	0.43
weighted avg	0.58	0.76	0.66

8.3. Random Forest

A second 'of the shelf' ML algorithm was tried - random forest which is an ensemble learning method for classification.

- Model's Performance:

	Precision	Recall	f1-score
0	0.38	0.09	0.15
1	0.77	0.95	0.85
accuracy			0.75
macro avg	0.58	0.52	0.50
weighted avg	0.68	0.75	0.68

8.4. XGBoost Models

In order to try a wide range of methods, we also used a gradient boosting by the XGBoost algorithm.

8.4.1 XGBoost Regressor Model

- Model's Performance:

	Precision	Recall	f1-score
0	0.00	0.00	0.00
1	0.76	1.0	0.86
accuracy			0.76
macro avg	0.38	0.50	0.43
weighted avg	0.58	0.76	0.66

8.4.2 XGBoost Classifier Model

- Model's Performance:

	Precision	Recall	f1-score
0	0.59	0.00	0.01
1	0.76	1.00	0.86
accuracy			0.76
macro avg	0.68	0.50	0.44
weighted avg	0.72	0.76	0.66

We used SHAP [6] to get deeper insights on the model decision making 7

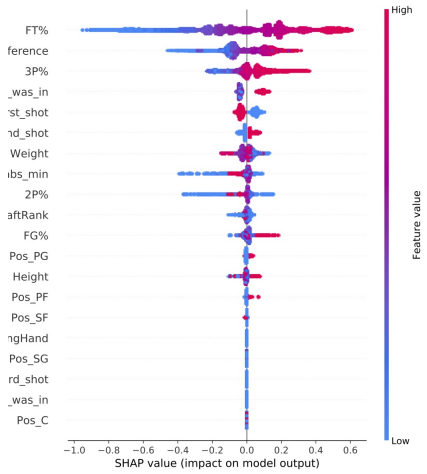

Figure 7. Shap Visualization

8.5. Neural Net - Binary CE LOSS

We contributed to our work variety by using a Neural Net Network with 4 hidden layers with sizes: 12-20-10-5 (the layers were dense connected and activated with leakyRelu) and one cell output with sigmoid activation. A batch size of 256 was chosen along side with 1000 epochs. Binary CrossEntropy loss has been the target function for the model.

	Precision	Recall	f1-score
0	0.62	0.00	0.00
1	0.76	1.0	0.86
accuracy			0.76
macro avg	0.69	0.50	0.43
weighted avg	0.73	0.76	0.66

9. Point Of Focus

As the initial results suggested most of our models yielded **score** for every predicted shot, caused the recall of the '0' class to be zero or close to it.Hence we decided to focus in increasing the recall metric of the **miss** class hoping to improve the model's performances.

10. Imbalanced Data-set

Since most of our data consists of made shots we believed that this is a major reason for the model's tendency to easily label shots as 'score'. Two techniques have been initiated through the attempt of coping with this issue.

10.1. Over-Sampling

Because of our data is imbalanced, a logical thing to do is to balance it artificially. Therefore we used an over sampling method called 'SMOTE' (Synthetic Minority Oversampling Technique) [**?**] which selects examples that are close in the feature space, draws a line between them and creates a new sample along that line. Visualization of SMOTE process can bee seen in Figure 8. After that process, an equal number of misses

and made free throws was achieved in both train and test.

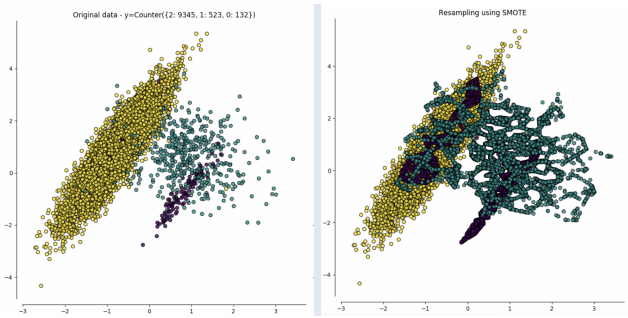

Figure 8. SMOTE Example. Left side unbalanced classes. Right side after running SMOTE the classes are quantity equals

10.1.1 Neural Network With Balanced Data Set

A neural net as described in 8.5 was trained with such balanced data set.

- Performance achieved:

	Precision	Recall	f1-score
0	0.32	0.57	0.41
1	0.82	0.61	0.70
accuracy			0.60
macro avg	0.57	0.59	0.55
weighted avg	0.70	0.60	0.63

10.1.2 Random Forest With Balanced Data Set

A random forest as described in 8.3 was also trained with such balanced data set.

- Performance achieved:

	Precision	Recall	f1-score
0	0.28	0.60	0.38
1	0.80	0.51	0.62
accuracy			0.53
macro avg	0.54	0.55	0.50
weighted avg	0.68	0.53	0.56

As we can see, this method helped us to achieve better recall on the under represented class, but decreased the accuracy around 10%.

10.2. Focal Loss

Another way to confront the imbalanced data set is the Focal loss [8]. This has the net effect of putting more training emphasis on the under represented class. It down-weights the well-classified examples,while ensuring that high accuracy on our minority class been achieved. We can see at Figure 9 the loss function behaviour per training sample.

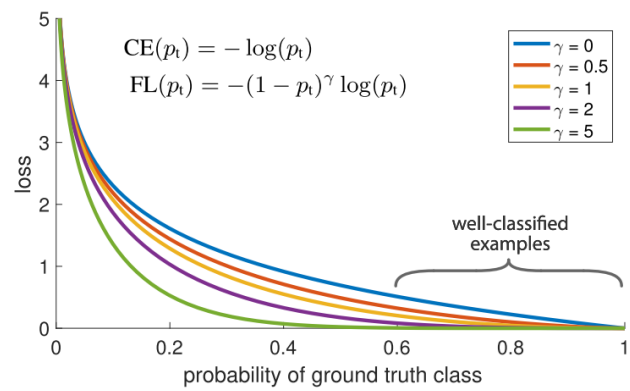

Figure 9. Focal Loss function

10.2.1 Neural Network Model - With Focal Loss

We used the same NN architecture as in 8.5 replacing the target function to focal loss.

- Performance achieved:

	Precision	Recall	f1-score
0	0.33	0.51	0.40
1	0.81	0.67	0.73
accuracy			0.63
macro avg	0.57	0.59	0.57
weighted avg	0.70	0.63	0.65

10.2.2 XGBoost Classifier Model - With Focal Loss

- Performance achieved:

	Precision	Recall	f1-score
0	0.59	0.03	0.01
1	0.76	1.00	0.86
accuracy			0.76
macro avg	0.68	0.50	0.44
weighted avg	0.72	0.76	0.66

As one can observe, an improvement was achieved on the 'miss' recall metric, but model's accuracy is still lower comparing to the baseline.

11. Threshold Fine-Tuning

Although a significant improvement has achieved in the recall, the sharp decreasing in the accuracy raised our concerns. A different approach was taken - forcing the model to label a shot as 'shot made' only if it has big confidence in this decision. This so called confidence was controlled by a threshold. Until now all of our models had such threshold of 0.5. A grid search was performed on several models in order to find the highest recall possible while maintaining a decent accuracy score (over 70%). We can see in Figures 10 and 11 what influence the threshold has on the discussed metrics over the Neural Network and Random Forest models. We prefer the behaviour of the random forest model (Figure 10), since we achieved a bigger improvement over the recall on label 0, without damaging the accuracy so much. We explored the entire threshold range from 0 to 1 in order to find such optimal threshold.

The performance of the Random Forest model with the optimal threshold with value 0.64:

	Precision	Recall	f1-score
0	0.34	0.24	0.28
1	0.78	0.81	0.81
accuracy			0.71
macro avg	0.56	0.55	0.55
weighted avg	0.68	0.71	0.69

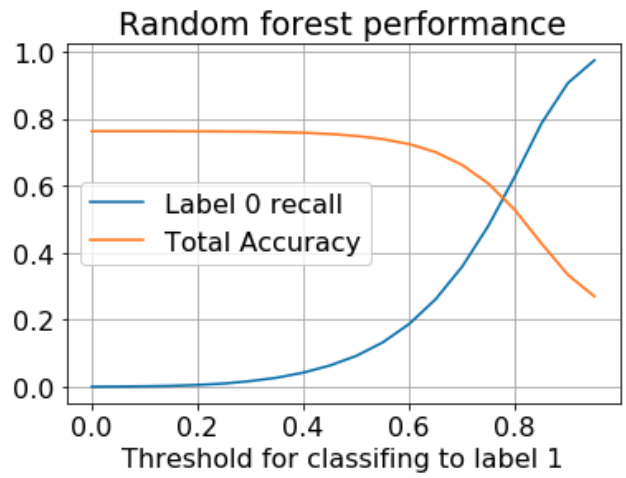

Figure 10. Label '0' recall and accuracy as function of dec. threshold

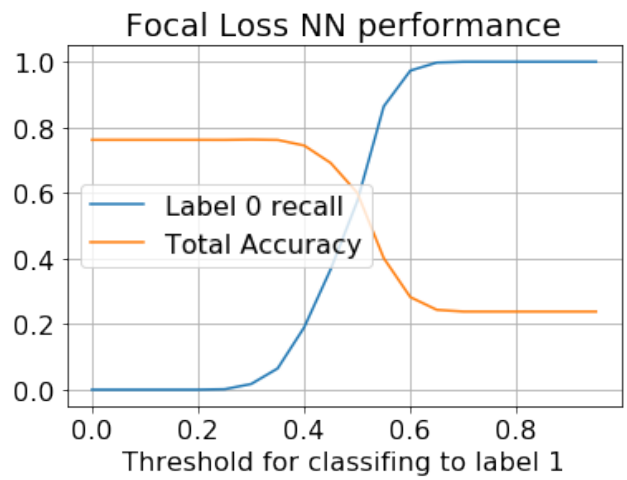

Figure 11. Label '0' recall and accuracy as function of dec. threshold

12. Conclusion

In this work we performed an entire data science process in order to predict a free throw result. As we reviewed, the ML methods we used got mediocre accuracy results. One explanation to that can be found in the fact that a free throw is a stochastic event. Moreover, the data is not separable at all - there is almost no difference between hits and misses. e.g is a human can decide which one of the samples is a hit and which one is a miss?

playoffs	Pos	ShootingHand	FG%	2P%	3P%	FT%	Height	Weight	draftRank	abs_min	Difference
0	SG	0	0.454	0.486	0.339	0.803670	201	104	30	36	2.0
0	SF	0	0.466	0.518	0.333	0.745030	198	97	9	23	3.0

Figure 12. A Miss Or A Hit?

We tried to make the data more separable by adding features related to the shots as described in 5.2 but, as said, it has not affected the results greatly. We had a success with increasing the recall on the misses but an unfortunate decreasing in the accuracy happened following the no free lunch rule.

13. Acknowledgements

We would like to thanks our mentors - Amit Somech[2] and Daniel Deutch [3] who assisted and guided us through this work.

References

[1] *FreeThrowsDataSet*, *https://www.kaggle.com/sebastianmantey/nba-free-throws* 1

[2] *FreeThrowsKaggleKernels*, *https://www.kaggle.com/sebastianmantey/nba-free-throws/kernels* 1

[3] *Kaggle :Shooting percent over time*, *https://www.kaggle.com/drgilermo/shooting-percent-over-time* 1

[4] Kaggle:free-throw-champs-via-hierarchica-partial-pooling, *https://www.kaggle.com/astrospv/free-throw-champs-via-hierarchica-partial-pooling* 1

[5] Gooding, Amy, and Frank L. Gardner. *"An investigation of the relationship between mindfulness, preshot routine, and basketball free throw percentage." Journal of Clinical Sport Psychology 3.4 (2009): 303-319.* 1

[6] Lundberg, Scott M., and Su-In Lee. *"A unified approach to interpreting model predictions." Advances in neural information processing systems. 2017.* 5

[7] Chawla, Nitesh V., et al. *"SMOTE: synthetic minority over-sampling technique"* In *Journal of artificial intelligence research 16 (2002): 321-357..*

[8] Lin, T., Goyal, P., Girshick, R. B., He, K., Dollár, P. *"Focal loss for dense object detection."* In *ICCV (pp. 2980–2988).* 6

[2] amitsome@mail.tau.ac.il
[3] danielde@mail.tau.ac.il