Predicting NBA Minutes Played

Jarvis Nederlof, Jack Tam, and Roc Zhang

January 25, 2020

1 Summary

We have built a regression model using a light gradient boosting model to predict the number of expected minutes an NBA basketball player will play in an upcoming game. Our final model performed well on an unseen test data set, achieving mean squared error of 38.24 with a coefficient of determination of 0.65. Both metrics showed better performance compared to a players 5-game average minutes played (our evaluation metric) of 50.24 and 0.55, MSE and R^2 respectively. The results represent significant value in the context of Daily Fantasy Sports, and the prediction model could be used as is. However, we note possible areas of further improvement that, if explored, could provide improved predictions, and more value.

2 Introduction

Fantasy sports represents a large and growing market in North America. According to a 2019 survey by the Fantasy Sports & Gaming Association (FSGA), 19% of Americans aged 18+ participate in fantasy sports [1]. Overall, the fantasy sports market represents millions of active participants spending billions of dollars yearly. DFS involves the process of creating lineup(s) for a given sport on a given day. As the games are played for a particular sport on that day, individual players will rack up fantasy points based on their real-world in-game performance. For example, in NBA basketball a player will accumulate fantasy points based on his total number of registered points, rebounds, assists, blocks, steals, and turnovers in the day's game. A lineups' total value is representative of the sum of the total fantasy points for each player in the lineup. Therefore, accurately predicting a players' fantasy points for an upcoming game holds a lot of value in what is a very competitive DFS market.

For this project we test various machine learning models to predict the total number of minutes a NBA basketball player will play in an upcoming game. As in most sports, but with NBA basketball in-particular, more playing time translates directly to increased fantasy point production [2]. Therefore, having a more accurate prediction of a player's expected minutes in an upcoming game would give users an edge when constructing their lineups for DFS contests, providing significant value to the end user.

3 Methods

3.1 Data

The data set used in this project is of the NBA Enhanced Box Score and Standings (2012 - 2018) created by Paul Rossotti, hosted on Kaggle.com. It was sourced using APIs from xmlstats. A copy

of this dataset is hosted on a separate remote repository located here to allow easy download with authenticating a Kaggle account. The particular data file used can be accessed here. Each row in the data set represents a player's box score statistics for a particular game. The box score statistics are determined by statisticians working for the NBA. There were 151,493 data examples (rows).

3.2 Analysis

We made predictions with 3 separate regression models: a simple linear regression model, an extreme gradient boosting model (XG Boost), and a light gradient boosting model (LGBM). The `R`[3] and `Python`[4] programming languages and the following R and Python packages were used to perform the analysis: `pandas`[5], `numpy`[6], `docopt`[7], `requests`[8], `tqdm`[9], `selenium`[10], `altair`[11], `scikit-learn`[12], `matplotlib`[13], `plotly`[14], `tidyverse`[15], `jupyterlab`[16]. The code used to perform the analysis and create this report can be found here.

4 Results & Discussion

To reduce the complexity of the models, and to remove noisy features, we did a substantial amount of feature engineering. The linear regression model represents the simplest approach, while the tree models were chosen as they work well for continuous count data like in our dataset. The hyperparameters for each model, mainly the number of estimators (n_estimators=60), were chosen through an iterative approach in order to reduce overfitting.

After all of our feature analysis we were left with two general categories of stats for each example: historical minutes played per game, and a player game rating. The minutes stats were calculated based on various rolling and exponential moving averages of different decay rates. The player game rating was a combination of different stats developed into a single number which was also calculated based on various rolling and exponential moving averages. Finally, we added in a flag for whether the player was a starter or on the bench for the upcoming game, and whether or not they were on the home team or the visiting team. All of the features were developed through our own knowledge of the game, and by examining correlations with the target, as seen in Figure 1:

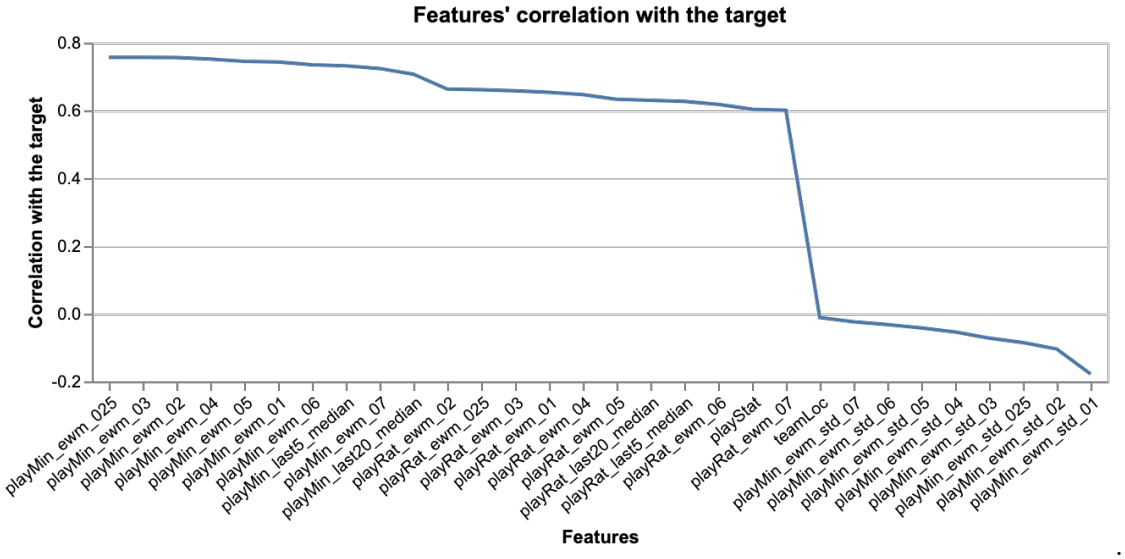

Figure 1. Feature Correlations with Target

From Figure 1 we can see that the most correlated stats are based on the players weighted average minutes played with various different decay rates ("playMin_ewm_decay"). The player ratings ("playRat_ewm_decay") appear to also be highly correlated with the target, but to a lesser degree. We can see that the players' exponentially weighted minutes standard deviation (as noted by "std") is negatively correlated with the target, which represents an interesting but sensible insight. Including both highly correlated and negatively correlated features into our training data represented a good mix of features that were reasonably expected to interact well resulting in good predictions.

We chose to evaluate our model by comparing our predictions to a players' five-game historical average minutes played (the base model). This represents a good test as it is reasonable to assume that a players five-game average would be a good indication of their expected minutes played in an upcoming game. We compared the prediction errors (mean squared errors and the R^2) of each model and the base model, as follows:

[6]:

	lgbm	xgboost	linaer regression	base model
MSE	38.24	38.30	39.59	50.24
Coefficient of Determination	0.65	0.65	0.64	0.55

Figure 2. Comparison of model fitness

From Figure 2. we can see that all of the models beat the base model in terms of both the mean squared error and the coefficient of determination. The LGBM model slightly outperformed the other regression models and was also significantly faster to train. In addition to the above analysis, we also examined the prediction residuals for each model to get a visual idea of how the predictions performed throughout the range of minutes played in a game. The residual results are displayed below in Figure 3:

Figure 3. Model Residuals

We can see from the residuals that the tree-based models performed significantly better across the range of minutes played. All models struggle a little bit at the extremes but that is expected, as predicting injuries (i.e very few minutes played) and predicting games that go into overtime (i.e. extra minutes played), are uniquely challenging problems and cannot be fully modeled with the data present in the data set.

Based on the above analysis and results, we chose to look further into the LGBM model to identify

what were the most important features by plotting the feature importance as determined by the model splits:

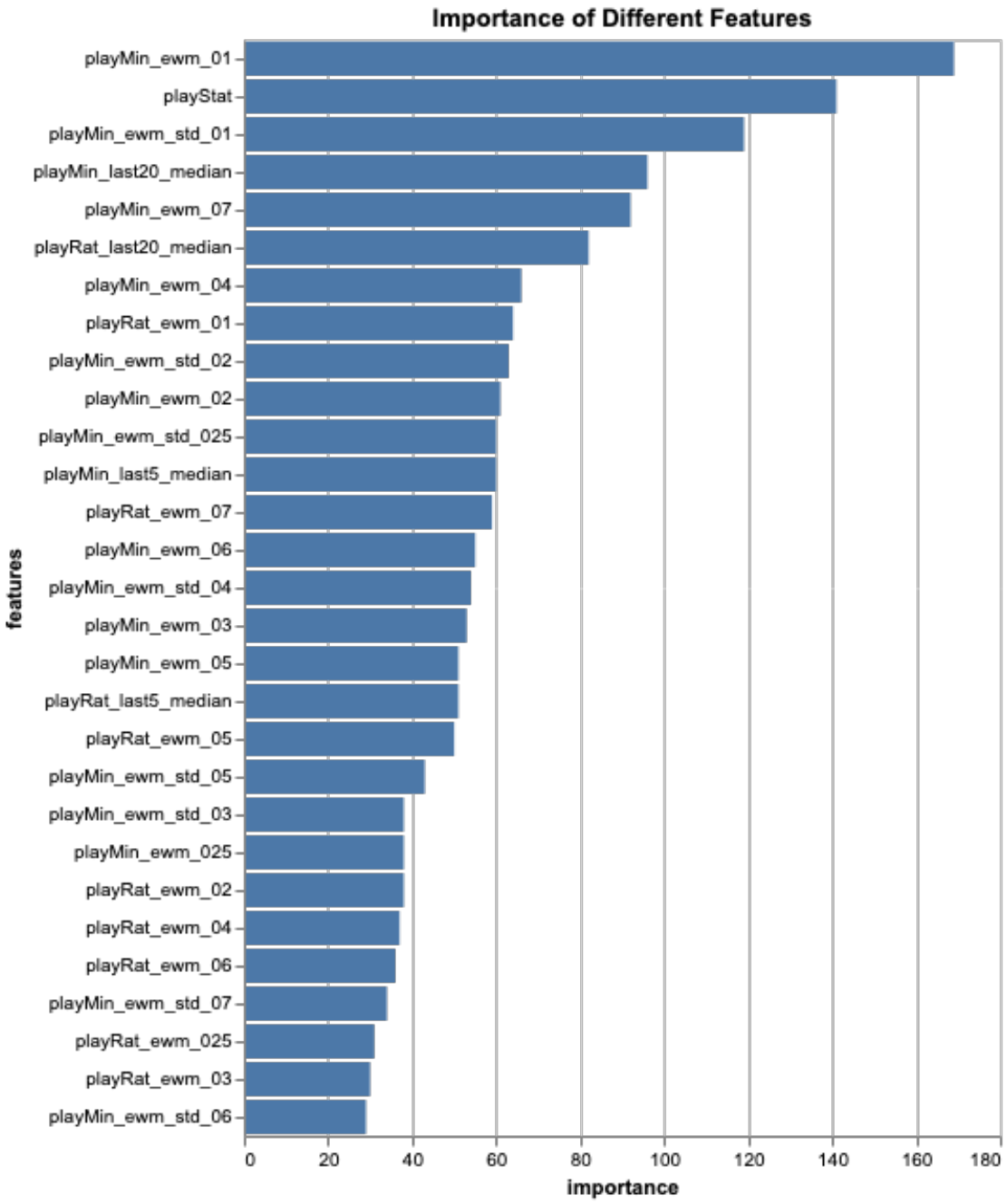

Figure 4. LGBM Feature Importance

We can see from Figure 4 that the players previous minutes played as measured with an exponential moving average and standard deviation ("ewm" - decay rate of 0.1) is the most important feature followed by "playStat" which represents whether a player was a starter or a bench player in the game. We can see that all of the features had fairly good representation in the model. Additionally, the historical minutes stats ("Min") are generally more important than the player rating stats

("playRat").

Overall, we demonstrate that using a players' historical average stats to predict minutes in an upcoming game can and does beat a baseline model of only their previous 5-game minute average. The tree-based models, particularly LGBM, provide better correlation and overall lower error rates and represent a small, but noticeably present advantage to a DFS user.

To further improve this model there are several areas that should be explored. Particularly pertaining to predicting the extremes of player minutes (i.e. the high end and the low end). Player injuries are tough to predict, but knowing if a teammate of a player will be in or out of the lineup on a current day would add significant insight. For example, if a player who normally plays 30 minutes is out of the lineup for the upcoming game, then the missing minutes will be distributed among the remaining players in the lineup, or possibly his replacement player (by position). Additionally, adding a feature indicating the tightness of games could help predict players who play more minutes. The thinking goes that in a blow-out game, the team will rest their starters, but if the score remains close until the end then the starters might get more minutes. This metric could be measured via the Vegas odds for the game, and how evenly matched up the teams are expected to be. Finally, more sophisticated models could be explored including model stacking and ensembles, and running rigorous cross-validation on the hyper-parameter set. We didn't fully explore this in our report but it could offer additional accuracy gains.

References

[1] Fantasy Sports and Gaming Association. Industry demographics, 2019.

[2] Draftkings. Nba all-star - lesson 01 - minutes and usage, 2019.

[3] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[4] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[5] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[6] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. [Online; accessed <today>].

[7] Edwin de Jonge. *docopt: Command-Line Interface Specification Language*, 2018. Python package version 0.6.2.

[8] Rakesh Vidya Chandra and Bala Subrahmanyam Varanasi. *Python requests essentials*. Packt Publishing Ltd, 2015.

[9] da Costa-Luis. tqdm: A fast, extensible progress meter for python and cli tqdm, 2019.

[10] Sagar Shivaji Salunke. *Selenium Webdriver in Python: Learn with Examples*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 1st edition, 2014.

[11] Jacob VanderPlas, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. Altair: Interactive statistical visualizations for python. *Journal of Open Source Software*, dec 2018.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[13] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[14] Plotly Technologies Inc. Collaborative data science, 2015.

[15] Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. R package version 1.2.1.

[16] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, may 2007.