Final Project: Predicting NBA Steals and Blocks

By: Ribhav Bose

Executive Summary:

In this project we aim to predict the steals and blocks performances of players in the NBA. Fantasy basketball, no matter what form it is played in, ultimately relies on the box score stats recorded by NBA players in game. However, box score stats are not equal, and steals and blocks end up being the most scarce and most volatile, with prediction of these statistics seeming inaccurate a lot of the time. Due to their volatility, successful prediction of steals and blocks in the NBA can create a significant edge in fantasy basketball and also have carry-over benefits into sports betting, making it a lucrative problem. Using publicly available data I test a few different model methods for the prediction of steals and blocks to see which performs best. Ultimately, gradient boosting looks to be the most effective prediction model for both steals and blocks in the NBA, although it is slightly outperformed by two benchmarks. Further analysis is done into the features that drive performance, alongside reflection on how results can be improved and next steps.

Introduction:

The box score statistics recorded for players during an NBA game are key in the world of fantasy basketball, with all of the following statistics carrying importance: points, rebounds, assists, threes, steals, blocks, and turnovers. However, not all of these statistics occur at the same frequency. While there may be a handful of players scoring in the double-digits in a given NBA game, a majority of the players in that game will also have recorded zero steals and zero blocks, as these two statistics are some of the hardest to come by in the league. The relative scarcity of steals and blocks means that they are of particular value in fantasy basketball. In points formats, where each statistic has a point value associated with it and the highest cumulative point value wins, steals and blocks are often multiple times more valuable than more common stats like points and rebounds. In other formats, each statistic category is kept track of separately, and the person who wins the majority of categories is awarded the win, steals and blocks remain supremely important because their low occurrence means that just one steal or one block can swing a matchup. This makes the ability to predict steals and blocks accurately supremely important. Gaining an information advantage with respect to the sparse and seemingly random categories of steals and blocks could lead to edges in all types of fantasy basketball competitions, and also create advantages in sports betting (on bets related to steals and blocks for players).

Previous Work:

At the moment there does not seem to be anything in the public domain that has to directly do with the prediction of steals and blocks in the NBA. I suspect that models and methods that predict NBA statistics exist, but may be the intellectual property of companies and individuals. There are some projects I was able to find that concern predicting the results of NBA games, and a project that predicts fantasy basketball scores. The projects presented below use similar data to what I ended up during the course of the project, making them valuable reference points.

A group of students at Cornell (Connor Young, Andrew Koo, Saloni Gandhi) did a project in Fall 2020 to predict NBA Fantasy Basketball Scores on the site DraftKings. They collected player box score stat-lines scraped from the website stathead, removing playoff games from their dataset, and used data from 2014-2020. In the preprocessing stage they also kept track the position the player played (PG, SG, SF, PF, C), whether the player was part of the home or away team, rolling averages of the players' box score stats over 3, 5, and 10 game windows, and an 'odds' feature related to the Vegas betting odds that the player's team would win the game. As part of preprocessing they also removed entries of players who had played no minutes. They had six total seasons of NBA data, so they split the first four years to be the train set, the next year to be the dev set, and the last to be the test set. For models, the students decided to try gradient boosting and different random forest models because they prioritized models which could perform on structured data and give information related to feature importance. For random forests, they use two approaches, a singular random forest and seven random forests for each of the counting stats (points, rebounds, assists, threes made, blocks, steals, turnovers). The project looks mostly at MSE for model evaluation, with the MSE being calculated from the player's actual fantasy basketball points earned during a given game compared to the model projection. They then compare the MSE of their models to the MSE of the projected fantasy points listed on the DraftKings website. The model that performed the best for their project was the gradient boosted model. Within the boosted model the students find that the most important feature is the 10-game rolling average (the largest rolling average window they constructed). In the end the project found that their gradient boost model was able to improve on the published projections by 8.6%. Interestingly, they present the R^2 values of the regressors in their random forest model, and somewhat in line with my thinking, the R^2 values in the dev and test sets for steals and blocks are significantly smaller than the R^2 for the other counting statistics. The project provided me with the baseline to try a gradient boosting model for my predictions. The use of the draftkings fantasy projections as a benchmark also made a lot of sense, and inspired my decision to benchmark my model with projections from similar websites. The filtering of playoff games from the dataset along with filtering out players that did not play also seemed like logical steps to apply to my own data cleaning process.

A paper from 2009 published in the Journal of Quantitative Analysis in Sports attempted to use neural networks to predict the outcomes of NBA games, where they use box score data from the

2007-2008 regular season. The data was separated into a 620 game train set and a 30 game val set. When separating the train and val sets, the author populated the features in the val set with the current season averages of the statistics of the relevant team. This is because the games in the val set are 'unplayed', and must be populated by features we can only know before the game actually happened. The author tried inserting rolling averages but found that the general season averages worked just as well for prediction. In terms of models tested, the author tries out four different types of neural networks: feed forward networks (with 1 hidden layer), radial basis functions, probabilistic neural networks, and generalized neural networks. The author also tests a fusion model that combines these four network types. The overall accuracy of predictions is used for evaluation, and these are then benchmarked against the accuracy exhibited by expert opinions collected by the author that attempted to predict the winners of NBA games. The results of the paper show that the feed-forward network performed the best of the four neural network types, and that the fusion network performed just as well as the feed-forward network. The feed-forward network was also able to slightly outperform the accuracy exhibited by the collected expert opinions, with the FFN exhibiting a top accuracy of 74.33% compared to the expert accuracy of 68.67%. This paper gave credence to my idea of using feed-forward neural networks as one way of trying to predict stocks and blocks, given the success it was able to show using similar features to those in my datasets.

A paper by Thabtah et al. published in 2019 also tries to tackle the question of trying to predict the results of NBA games with different models. For data the group uses a kaggle dataset consisting of box scores of NBA Finals games from 1980-2017, discarding missing values and turning some continuous variables into categorical ones. The group used three main models: neural networks, logistic model trees, and naive bayes regression, and evaluated their models based on accuracy, precision, recall, and F1-score. They attempt to use feature selection to narrow down which features will be inputted into the all three models, but ultimately they find that in the case of both the neural network and the logistic model tree that the raw data containing all the features performs the best. In terms of model performances, the neural network and logistic model tree exhibit similar performance, and outperform the regression model. The regression model exhibits an accuracy and F1-score of 0.76, while both the neural network and logistic tree model exhibit their best accuracy/F1-scores as 0.83/0.83. The work does not use any real-world benchmarks to compare their performance metrics to, and does not go into great detail regarding the structures of their neural network or logistic tree model. The paper suggested to me that in the models I intend on using, inputting my entire raw dataset might be the way to go, as the raw data tended to perform better in the more complex models in the paper. And additionally, the paper suggests to me that naive regression would not be the optimal model of choice, but it may serve as a good baseline.

There also exists some work related to k-nearest-neighbor predictions for NBA player related outcomes. A blogpost on 'towardsdatascience' by Christophe Brown uses data consisting of a player's career points, assists, rebounds, blocks, and steals with knn to predict the position that player played. The model was evaluated for its accuracy in matching players to their position, and this relatively simple implementation reached an accuracy of 70%. This is one of a few projects I found that used a knn-classification model with NBA player data showing the proof of concept, and shows that it could be a useful option during the project.

The website fivethirtyeight.com, known for both its politics and sports content, does its own player projections, and while the nitty-gritty of their methodology remains secret, in a blogpost they say, 'our player projections forecast a player's future by looking to the past, finding the most similar historical comparables and using their careers as a template for how a current player might fare over the rest of his playing days.' This idea of finding historical comparables being used in their algorithm as well leads me to believe a knn strategy would also be worth looking into for this project.

Solution Description:

Data used for this project comes from a range of sources. I came across a kaggle dataset that contained individual player box score data going back to 2003, which contains all the key counting statistics including points, assists, rebounds, steals, blocks, and turnovers. Other data used includes nba.com data containing player personal information, such as age, heights, and weight. The NBA also began tracking 'hustle' statistics in 2015 on nba.com, which I thought could be particularly relevant to steals and blocks (as they are also often thought of as 'hustle' categories), so an alternative dataframe starting from only 2015 was created using this hustle data alongside the game data from before. In order to get some of the data from nba.com, I used the nba_api package to scrape the relevant data needed for my project.

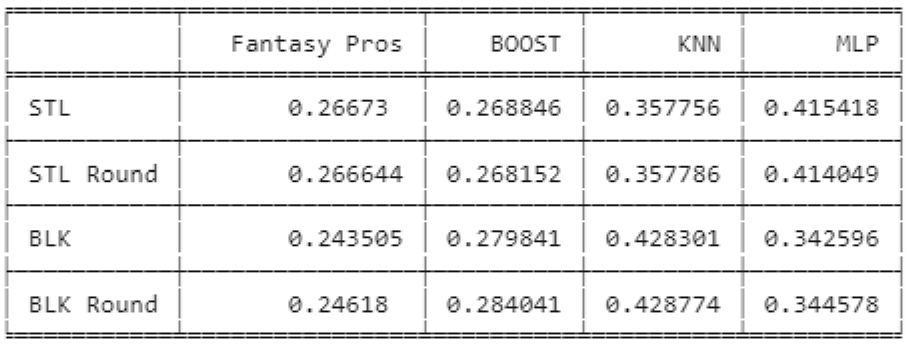
Lots of preprocessing was done on the data using numpy and pandas.I made the decision that it would be most feasible to predict the average amount of steals and blocks a given player records in a season, rather than training models to predict the amount of steals and blocks a player records in a given game, due to the sheer amount of individual player games in the dataset (which from preliminary testing was too computationally expensive for certain model types), alongside some idiosyncrasies in the game by game dataset. In order to create features most beneficial for predicting player steals and blocks in a given season, I had to compute season average features for each of the key statistics, and also took time to create teamwide season averages (such as team blocks per game, team steals per game), and opponent season averages (such as opponent blocks per game, opponent steals per game). Similar to the paper discussed in the earlier section, I made the decision to drop all games that did not occur during the regular season of the NBA, and dropped players who did not meet a certain threshold for average time played per game. Since I was using team-wide metrics, I also decided to drop players who played on multiple teams over the course of a given season. This left a dataset with around 8000 total observations ranging from 2003 to 2022.

Additionally, as alluded to above, I created a second dataset starting from 2015 that incorporates the hustle statistics tracked on nba.com, with similar filtering as the previous dataframe. This came out to around 3000 observations from 2015 to 2022.

I split each of the datasets into train, val, and test splits. For the larger dataset, the training split consists of games from NBA seasons starting in 2003-04 and ending in 2017-18. The val split consists of the next two NBA seasons: 2018-19, 2019-20. And the test split is the last two NBA seasons: 2020-21, 2021-22. For the smaller dataset that is combined with the hustle statistics, we create our train split from NBA seasons 2015-16 to 2019-20. Our val split is NBA season 2020-21, and our test split is NBA season 2021-22. It is important to note at this point that the features we use to predict a player's steal and block average in their current season cannot include statistics from that season, because from the perspective of the model those statistics would not exist at the time of prediction. We use the previous season's averages as the features for predicting the current season's blocks and steals, which involves some finagling with the data.

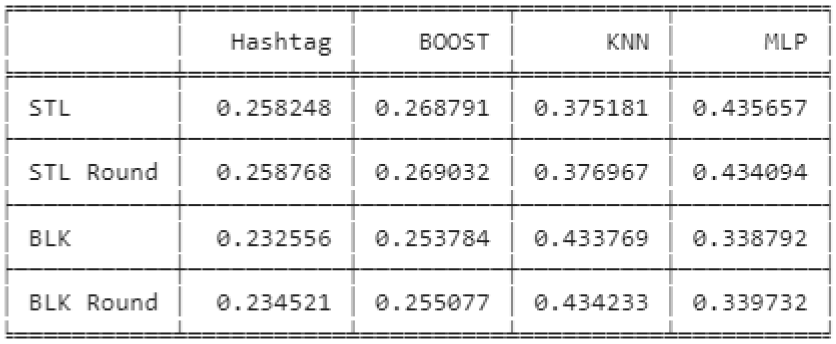
I mainly tested three types of models: KNN regression, gradient boosting, and multi-layer perceptron. This is because these were the three main approaches I identified through the research described in the previous section. For all three, I used the built-in relevant functionalities in scikit-learn to train and test models. For each model type, I did some hyperparameter tuning to find the best model for each dataset and for each predicted statistic (i.e. tuning separately for steals and for blocks), selecting the best models by virtue of having the lowest RMSE on their respective val set. This meant after the first stage I was left with 12 tuned models (3 model types * 2 datasets * 2 output features). I then took the best hyperparameter set and trained each model on the combined train and val sets, and evaluated the RMSE on the test set. Based on these RMSEs, we find that in general the MLP performs the worst, and that the boosted regression performs the best. RMSE for MLP models were very poor, with the best being an RMSE of 0.33 for the blocks predictor using the large dataset, but the other RMSEs all being >= 0.39. In the case of the KNN regressor, the RMSEs of the best model of each type were in the 0.30-0.33 range, and in the case of the gradient boosting regression the RMSEs were in the range of 0.245-0.27. At this point I selected the lowest RMSE model for each model type between the two datasets to compare against benchmarks. Interestingly in the case of the boosted regression, the smaller dataset containing the hustle statistics slightly outperformed the larger dataset without the hustle statistics. In the KNN case, the large dataset performed better for predicting steals, while the hustle dataset was better for predicting blocks. And for the MLP case, the larger dataset performed the best across both predictors.

For benchmarks, I used the wayback machine to find two websites that published their preseason projections for player statistics, so I was able to find two sources for 2022 NBA preseason projections. I was able to find data from two sources: HashtagBasketball and FantasyPros. To get the data I was able to just copy and paste the data into a spreadsheet. I calculated the RMSEs of their predictions vs the actual steal and block averages of each player, and compared it to my own. Results can be seen in the tables below (the STL Round and BLK Round rows refer to the fact that the projections from the two sources only had season averages to the nearest tenth, so I rounded my data off similarly for a fair comparison):



	Fantasy Pros	BOOST	KNN	MLP
STL	0.26673	0.268846	0.357756	0.415418
STL Round	0.266644	0.268152	0.357786	0.414049
BLK	0.243505	0.279841	0.428301	0.342596
BLK Round	0.24618	0.284041	0.428774	0.344578

Table 1: FantasyPros Benchmark Performance (RMSE)



	Hashtag	BOOST	KNN	MLP
STL	0.258248	0.268791	0.375181	0.435657
STL Round	0.258768	0.269032	0.376967	0.434094
BLK	0.232556	0.253784	0.433769	0.338792
BLK Round	0.234521	0.255077	0.434233	0.339732

Table 2: HashtagBasketball Benchmark Performance (RMSE)

We see that in general, the RMSEs of both benchmarks outperform all models, with only the boosted regression model being competitive. Also in both cases it looks like our models get closer to the benchmark for STL models, while they perform relatively worse on BLK models in terms of distance from the benchmark. As a note, the reason for the differing RMSEs across the optimized models is because the two sources only have predictions for differing subsets of players. I think that the closeness of the RMSEs between the boosted model and the benchmark in general does suggest that I am on the right track in using the boosted model, but need to find some better features (as will be discussed in the following section), and that possibly both these websites use a similar model to predict player performances.

Directly below in figures 1 and 2 are the two importance plots of the best steal and best block boosting model.

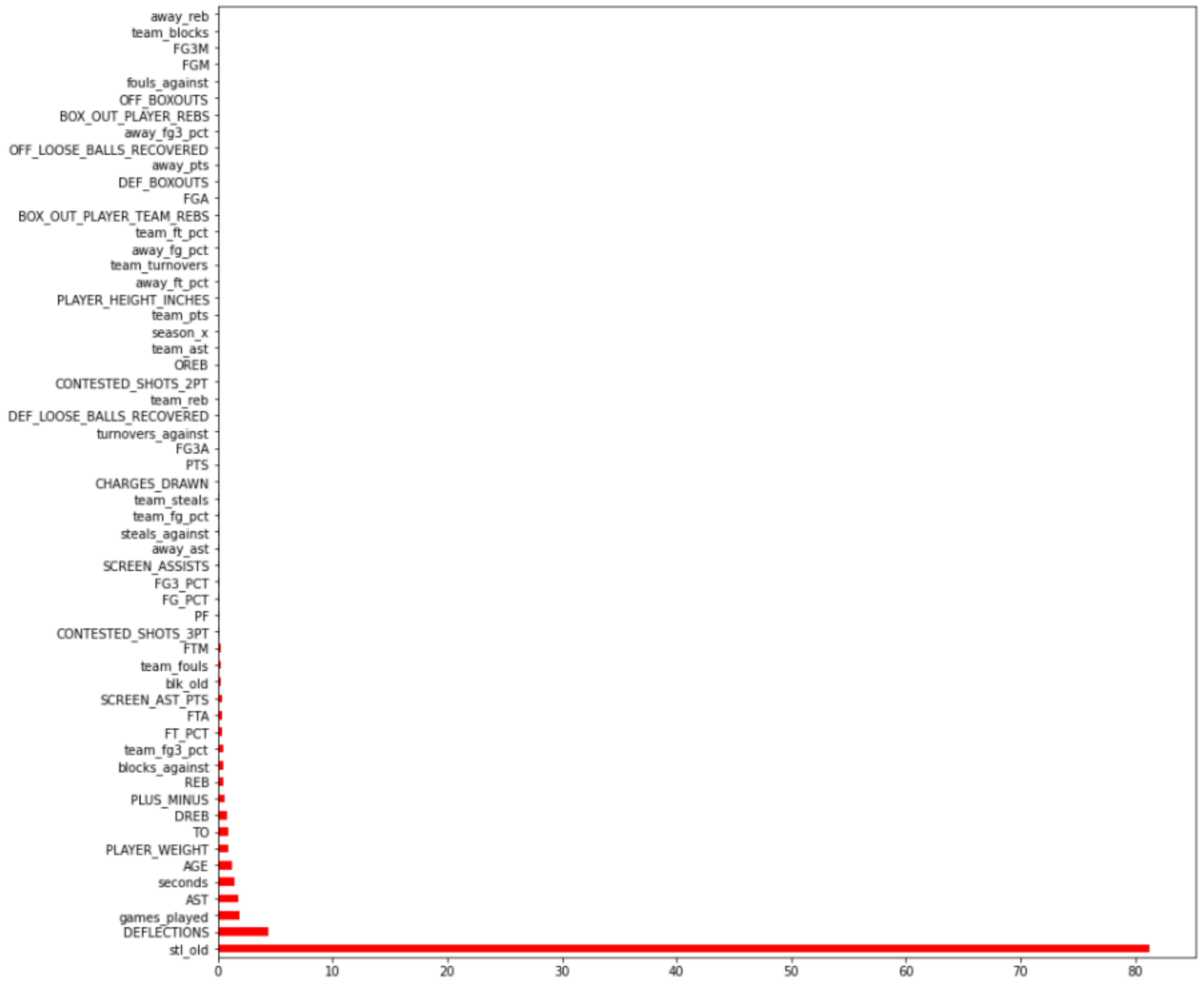


Figure 1: Importance Plot of Best Steal Prediction Boosting Model

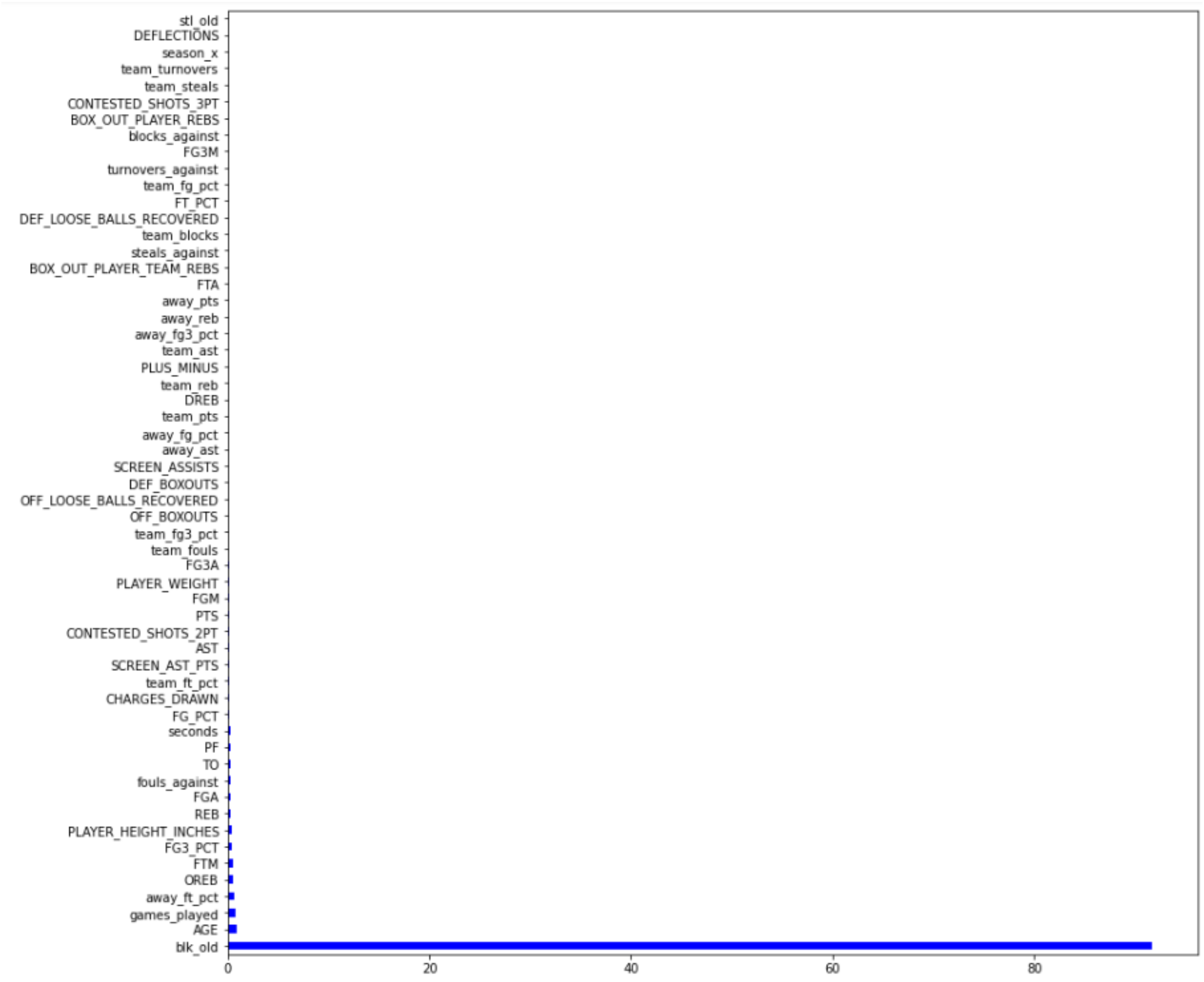


Figure 2: Importance Plot of Best Block Prediction Boosting Model

Here we see that overwhelmingly the most important feature used by the model to predict steals and blocks in the next season are steals and blocks from the previous season. I had expected this to be the case, but not quite to the degree suggested by these diagrams. In the steal diagram, we also see that deflections per game (where the player deflects an opponent's pass) are the second most predictive feature, although at a much smaller scale. This is promising because deflections were one of the hustle stats, meaning that the addition of these statistics was able to add some more predictive power to the boosting model. Interestingly, in the importance plot of the block model (Figure 2), the significance of the 'seconds' metric, which represents average time played in a game, is much less important than the same metric in the steals importance plot. Although both importances are relatively small, this could suggest that blocks are in fact more random than steals, seeing that time played in a game is not predictive of blocks while it is slightly for steals.

Creation of the plots and tables above was done with the tabulate and matplotlib packages in python.

Challenges and Next Steps:

Data that could have been useful for this project could be the incorporation of more advanced 'hustle' like statistics into the dataframe, but unfortunately this data is often behind paywalls. Some other data that could have been helpful would be incorporating some information about the player's form going into a season. At the moment the dataset has no feature that distinguishes whether a player started the previous season hot and cooled down over time, or whether things started to click for them at the end of the previous season, because the only feature we use were entire season averages. It might be the case that players who ended the previous season on hot streaks involving steals and blocks could have higher steal and block outputs than the previous season average would predict, and the opposite for those who ended the season cold. This feature could have taken the form of something like 'steals averages during last 10 games' to be an indicator of a player's form. A feature I did not have that I suspect the benchmarks take into account is the position of a player. For positions, generally guards (point guards, shooting guards) record more steals, and forwards (small forward, power forward) and centers record more blocks. In general guards tend to be shorter, and forwards and centers tend to be taller, so this is somewhat already captured in my current dataset, but each player in the NBA is unique, meaning extra information to feed the model regarding position played could be useful, but unfortunately this data was not in any of the data I found from nba.com.

In terms of next steps that I would take with the project, I would try to find features like those discussed above and see if model performance improves. I would also try different strategies, including more random forest based models, and finding ways to improve performance on neural network based models. In particular, the neural network performed especially poorly compared to the other models, which leads me to believe that there is lots of room for improvement on that front, whether it be using a different type of neural network entirely, or having to take more time to understand how best to structure the MLP. Additionally, while steal and block predictions are important for fantasy basketball, the most important thing in fantasy basketball is finding value in players, so I would probably extend the reach of the project to include more relevant fantasy basketball statistics, and also try to ask questions about how to best find undervalued picks. While it is a given that players like Rudy Gobert (3x Defensive Player of the Year) will likely record strong steal and block statistics, more valuable in fantasy is the ability to find players who are likely to take a leap in the upcoming season, or players that I've never heard of who can contribute positively to my team while costing me very little.

For effort, I would say 65% of my effort went into sourcing, cleaning, and creating features for my data, while the 25% was devoted to the evaluation and tuning of models, with 10% being left for research and debugging. I expected that the cleaning of data would likely take most of my

time, but some of the messiness and idiosyncrasies native to the dataset made this process take longer than I had initially budgeted, especially when I made the decision to pivot from individual game prediction to season average prediction. Overall I would say I put a pretty large amount of effort into the entirety of the project itself, but due to some sickness earlier in the quarter much of the work regarding the project was back loaded to the latter 2 weeks of the quarter.
In terms of things attempted that did not end up panning out, I spent a lot of my initial effort into getting a usable dataset to test on individual game prediction. I was able to get such a dataset, but found that when I tried to train any sort of model on my dataset, it would take far too long for the computations to finish, likely due the sheer size of the dataset (on the scale of hundreds of thousands of entries). My initial plan was to train both an individual game prediction model and a season average model, but it became evident that training an individual game prediction model would not be computationally feasible. There were also a number of features I tried to add in the dataset, such as number of years played in the league, but the data available to me often contained small issues, such as the fact you cannot find the first year an 'undrafted' player played in the NBA. In terms of arriving at the final solution, since doing game by game prediction was unfeasible, I decided to devote more of my time into creating valuable features for the season average dataset, and figured that since I was no longer doing the game by game predictions, that testing the models with two different datasets could also be interesting. In particular, since one dataset had more features while the other had more data, I was interested in whether there would be a significant difference in model performance, and whether some models performed better with more data, and others with more features.

The project as a whole was very enjoyable. The data cleaning process, while tedious at times, was pretty eye opening, and I found myself having a much better grasp of pandas and dataframes the more time I spent on cleaning. Something particularly challenging in the project for me was knowing when to make certain calls regarding the data. At some point I had to decide whether I wanted to keep playoff games in my dataset, and whether I wanted a certain quality of players in my dataset, and these were questions that I was not sure the right answer to. For some of these questions, I had to weigh whether the time invested in finding the best answer was worth not having that time for further downstream tasks, which was something new and challenging. Something I found interesting about the project was the ease at which I could make tiny tweaks once I had a sort-of pipeline between data to model creation, which opened my eyes to some more efficient ways I can go about doing similar projects in the future.

Bibliography:

1. Young, Connor, et al. "Final Project: NBA Fantasy Score Prediction." (2020).

2. Loeffelholz, Bernard, Earl Bednar, and Kenneth W. Bauer. "Predicting NBA games using neural networks." Journal of Quantitative Analysis in Sports 5.1 (2009).

3. Thabtah, Fadi, Li Zhang, and Neda Abdelhamid. "NBA game result prediction using feature analysis and machine learning." Annals of Data Science 6.1 (2019): 103-116.

4. Brown, Christophe. “Simple Modeling of NBA Positions Using the K-Nearest Neighbors Machine Learning Algorithm.” Medium, Towards Data Science, 6 Nov. 2021, https://towardsdatascience.com/simple-modeling-of-nba-positions-using-the-k-nearest-neighbors-machine-learning-algorithm-223b8addb08f.

5. Lauga, Nathan. “NBA Games Data.” *Kaggle*, 23 Dec. 2022, https://www.kaggle.com/datasets/nathanlauga/nba-games.

6. “NBA Stats.” The Official Site of the NBA for the Latest NBA Scores, Stats &amp; News. | NBA.com, https://www.nba.com/.

7. Paine, Neil. “How Our NBA Predictions Work.” FiveThirtyEight, FiveThirtyEight, 18 Dec. 2018, https://fivethirtyeight.com/methodology/how-our-nba-predictions-work/.

8. “Fantasy Basketball Projections.” Fantasy Basketball Projections and Rest of Season Rankings 2021-22 | Hashtag Basketball, https://web.archive.org/web/20211006155355/https://hashtagbasketball.com/fantasy-basketball-projections.

9. “Fantasy Basketball Projections.” NBA Fantasy Basketball Overall 2020-21 Average Projections | FantasyPros, https://web.archive.org/web/20210730013226/https://www.fantasypros.com/nba/projections/avg-overall.php.