Creating a Predictive Model for NBA 2k Player Ratings

By: Andrew Rabines

Introduction:

Basketball is one of the largest sports markets in the world. The NBA is the most difficult professional sports league to enter, seeing as only 15 players maximum can be on each team. For those without the adequate talent to make it to this level, there's alternatives to quench the thirst for basketball. Activities such as watching the games on TV, playing a pick-up game with friends, or playing the NBA 2k video game. The later of which retains a near monopoly on the basketball video game market, accomplishing this success due to the quality of the game and certain aspects such as player ratings. The exact algorithms used to determine these ratings are unknown but the controversies surrounding their results are easily observed with each new rendition of the video game. Being a massive basketball and NBA2k fan, I frequently found myself wondering how these player ratings were determined and if they accurately displayed player performance. Once my knowledge of pandas and its various techniques developed, I decided I would tackle this exact question and create my own model to predict NBA 2k19 player ratings.

Data Sources:

In order to create the model I would need adequate data sets of previous 2k ratings and previous season's player statistics. Unfortunately, it was quickly realized that due to the NBA season ongoing during the project timeline, and there not being any available dataset for the latest video game, 2k19, that I would run the model on last season's statistics and video game edition, 2k18. The first source, a dropbox file[1] of all the players in the NBA 2k18 video game, was found via a comment on a Kaggle post. The dataset was downloaded as an excel file and was 1.2 mb in size, containing many duplicate players as it contains historical and all-time versions of NBA teams with multiple NBA legends across these teams. The only variable of true importance to the model in this dataset is the player rating column as it gives us the baseline for improvement or digression that the model will predict.

The second dataset was taken from one of the most reliable basketball websites, basketball-reference[2]. The site keeps updated statistics of every team and player across a variety of criteria. I found the table of every player's statistics from the 2017-18 NBA season and scrapped it into an HTML file which I could then easily upload into pandas. The size of this dataset was slightly larger than the 2k18 dataset, containing 1.97 mb of player statistics, including key metrics such as PPG (points per game), APG (assists per game), and RPG

[1]*DropBox*, 14 Oct. 2017,
https://www.dropbox.com/s/js2hqcd152mq4kh/NBA%202k18%20All%20Players%20Master.xls?dl=0.
[2]*Basketball-Reference*, 14 Apr. 2018,
https://www.basketball-reference.com/leagues/NBA_2018_per_game.html.

(rebounds per game). These metrics, along with turnovers, steals, field goals attempted, and field goals made, would be crucial to defining the main function to determine player rating later on.

Data Manipulations:

Once the data sets were imported via the read_excel and read_html functions, manipulation could begin. Upon inspecting the data and parsing to find null values, I noticed the only location of said values was in two columns in the NBA 2k18 player ratings data set -loaded in as data_2k in the pandas file. These two columns, 'All Time Team?' and 'Duplicate?' were columns that returned an asterisk when the player was on one of the 2k18 all time teams or was a duplicate respectively. Seeing as how a player on the all time team would not be accounted for in my calculations, and my plan to merge did not involve the duplicate column, I did not feel a need to fill these null values. The second data set of nba player statistics from 2017-18 -now named players- contained some null values near the bottom of the set for players that barely played in the NBA last season and therefore had no statistics to produce. These nulls values did not make much of an impact either as upon merge, many of these players were dropped as they were not in the data_2k data frame.

As far as I could tell, the statistical data in the players data frame was accurate but did contain multiple entries of players, due to the fact that basketball-reference recorded multiple entries for players who were traded during the season. Additionally, the data_2k data frame contained duplicates of several big name players who were on an all time or historical team in addition to their regular team in the game. To combat the first duplicate issue, I called drop_duplicates on the players dataframe, calling a subset on column: 'Players', and keeping the first entry as this is the one that had each duplicate players' total statistics for the year. In the case of data_2k, I filtered the data frame by the 'Year' column set equal to 2017-18, to ensure I was only grabbing base-game players, then saved this filter into a new data frame.

The next big step was to merge the two dataframes now that individually they were relatively clean. This was accomplished by merging modern_2k -the new data frame of 2k players- with players on the column 'Players' and with an inner merge to ensure only players with a 2k rating were in the new data frame. The new dataframe, combined_data, contained a couple duplicate columns caused during the merge, I deleted and renamed the columns accordingly as well as removed a few more unnecessary columns to make the data frame a little smaller. The merge took several attempts as I tried both inner and outer merges before realizing the inner, in the order I merged on, gave me less null and more relevant data.

To make the predictive model I desired, I required a main metric, or combination of metrics to base a function around. Upon reading an article on Bleacher Report[3], I realized I could

[3] Fein, Zach. "Cracking the Code: How to Calculate Hollinger's PER Without All the Mess." *Bleacher Report*, Bleacher Report, 3 Oct. 2017, bleacherreport.com/articles/113144-cracking-the-code-how-to-calculate-hollingers-per-without-all-the-mess.

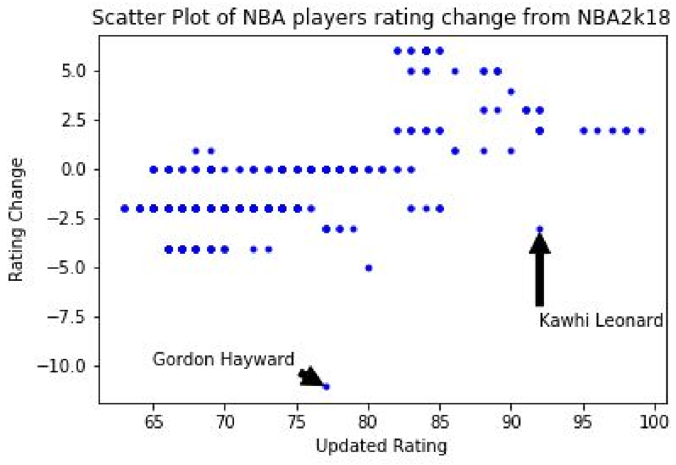
use PER (player efficiency rating) to achieve the baseline for my rating calculator. Typically, PER takes into account more complex metrics from both individual players and their teams but seeing as most of the metrics used were not available or able to be calculated given my data sets, this easier version would have to do. In order to calculate even this simpler PER, I had to create a couple new columns to display field goals missed and free throws missed per game, this was achieved by subtracting the FG and FT columns from the FGA and FTA columns respectively (FGA and FTA standing for field goals attempted and free throws attempted). With these columns created, I had all the variables needed for my PER function and so I ran the one provided in the Bleacher Report article, having to divide the final calculation by 82 (the number of games in the NBA season). The results were very similar to actual player PERs from the given NBA season. The leader in the PER column was James Harden (last year's MVP) with a score of 32.16, which is slightly higher than his actual PER of 29.8 but both calculations are the highest from this season, so the slight inflation will work for the prediction.

The actual function to predict improvement or digression -named improvement- took the longest amount of time due to the necessity for the function to be comprised of roughly 35 different if and elif statements. There likely are simpler ways to write this function but this was the easiest way in my head and once I started writing, I couldn't stop. It took awhile to tweek the function in order to get proper results as I had to determine what PER and previous rating ranges to run the conditional statements over. I broke the PER categories into 7 different levels, the highest being 25 + and the lowest being 0-5. While each PER conditional came with player ratings in 5 different ranges from 90+ to 65-70 (as 65 was the lowest player ratings in the data frame). Once complete, I applied the improvement function on the data frame in it's own column and used it to define a new column, New_Rating, which simply took the Rating column and added the improvement column. With the functions defined and applied, I was now able to view my predicted ratings for NBA 2k19.

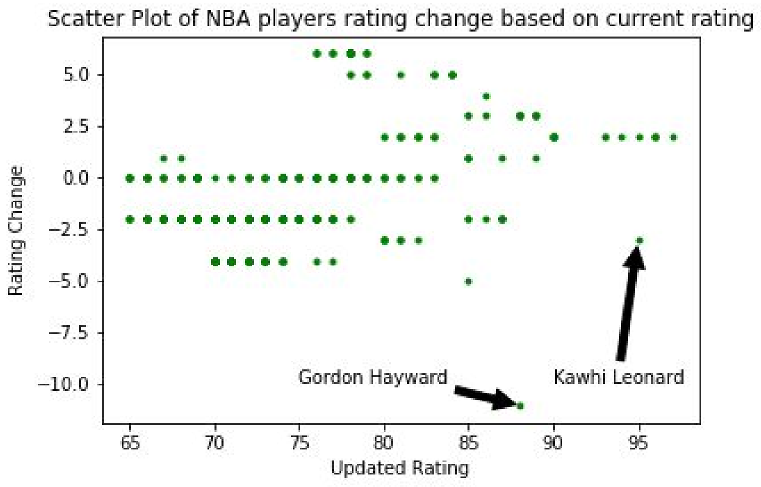
Visualizations and Analysis:

The first, and simplest way to view my data was via calling sort_values(ascending=False) on the New_Rating column. This gave me the results I more or less expected, LeBron James at the top spot with a rating of 99. The next three players, James Harden, Stephen Curry, and Kevin Durant were all rated 98, kind of unusual to have three players rated that high in 2k but based on PER and improvement, they seem to have deserved it. I then sorted the values based on improvement to find several players with upgrades of 6 ratings, but shockingly none higher. The players with these upgrades were all players rated between 70-80 with a PER 0f 15.0-17.5.0, indicating a quality season for these level players. Players such as Kyle Kuzma, Ricky Rubio, and Elfrid Payton were included in this group. Finally, I decided to see who dropped the most and it was Gordon Hayward, who dropped 11 ratings to a 77, due to the fact that he was injured 4 minutes into the first game of the season and missed the next 81.9 games. I thought about adjusting the improvement function to neglect players who were injured but missing time from an NBA court will hurt a player's game, so I felt the results to be sad but fair.

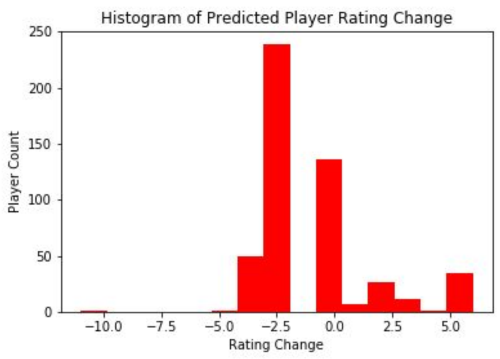
Next came the actual visualizations provided via matplotlib. The first visualization I created was a scatterplot of the player's updated ratings(x-axis) by improvement(y-axis). Due to the pythonic nature of my improvement function, I found that the scatterplot looked odd as many players were bunched together in terms of improvement, I was a bit disappointed by this but you were able to clearly visualize a few outliers such as Gordon Hayward, in this and many of the plots to come. We are also able to observe Kawhi Leonard, who refused to play much of last year, dropping 3 overalls to a 92.

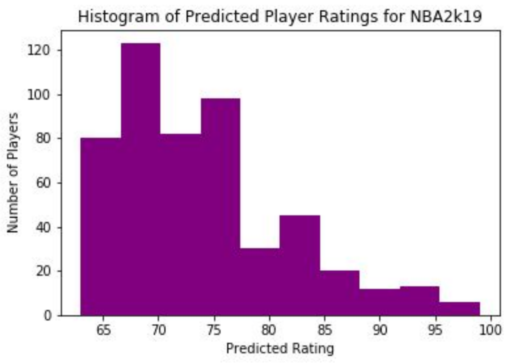


I created another scatter plot to show the effect of improvement on the previous rating(2k18). It showed a similar trend with more of the data spread out. When compared to the scatterplot of my new ratings, one can observe that my predictive model may favor to upgrade players more than downgrade, just slightly. Most players digressed by 2 overalls or remained the same but it seems that higher rated players tended to improve more often than digress -as we observed with Kawhi and Gordon.

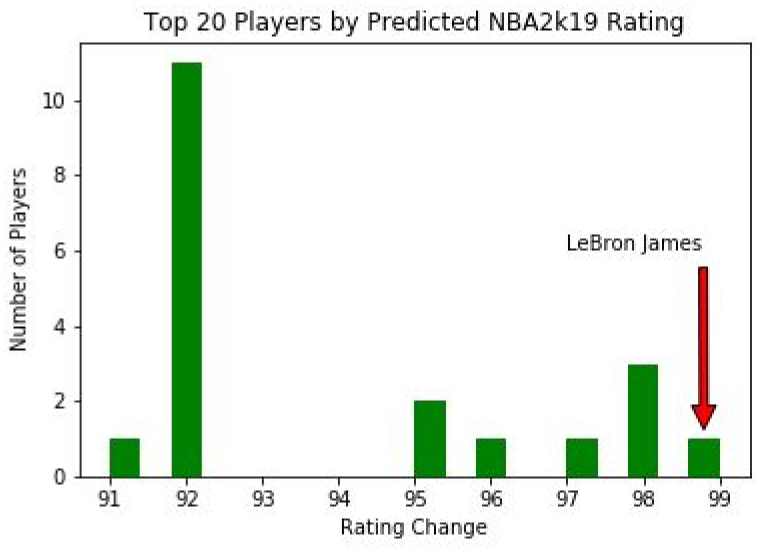


I then decided to create a couple histograms to more neatly view improvement and player ratings. The first histogram, below on the left, shows the number of players with various rating changes. As we can observe, many players digressed by 2-4 overalls, slightly less than 250, while many players remained the same rating. We are able to notice a slight bump of roughly 30 players who improved by 6 overalls but around an additional 30 improved at any level besides this group. In the second histogram, we can see the predicted ratings by number of players in each rating range. The general trend is that most players are rated at 77 or lower, while roughly 100 make up the 77-99 range. Within the highest-ranked range, 95-99, less than 10 players are included which seems properly representative of the number of elite NBA players.





The final visualization, another histogram was used just to show the top-20 rated-players and determine if the spread of ratings seemed correct for a NBA 2k game. The results has 1 at 99 overall, 3 at 98, 1 at 97, 1 at 96, 1 at 95, 11 at 92, and 1at 91. The general spread seemed accurate to me with the only concern being the 11 players rated 92. However, 92 rated-players generally are players who are not quite elite superstars but are still players every team wants and can lead a team if need be, having 11-12 of these calibre players seems correct.


Top 20 Players by Predicted NBA2k19 Rating
Number of Players
Rating Change
LeBron James