How can we correctly value NBA players? (Aiden Liu)

In the past, majority of the NBA general managers (GMs) have been paying players according to their production on the court in terms of their counting numbers (Total/average points, rebounds, assists etc.) Often times, players that NBA GMs signed/traded for either did not exceed/surpassed expectations because the GMs misjudged the true value of the players. As the tracking technology for NBA improves, as well as more and more people coming out with complex formulas, a handful of high level, secondary statistics that attempted to measure a player's impact surfaced. They are known as "Advanced Analytics". "Advanced Analytics" is used to break the game down and to help players/GMs/viewers to better understand the game. However, even with all these complex statistics, a lot of people are still questioning the legitimacy of each individual stat. It is almost in agreement in the NBA circle that there is no one single stat that can accurately show the impact a player has on his team in terms of **winning,** as there are too many variables and factors that are involved. There are, though, two statistics, that are frequently used in sports debates, that emerged as the "all in one" statistics in measuring the total impact that a player has.

1. **Player Efficiency Rating (PER)**
   It is a per-minute rating developed by ESPN.com columnist John Hollinger. In John's words, "The PER sums up all a player's positive accomplishments, subtracts the negative accomplishments, and returns a per-minute rating of a player's performance."

2. **Win Shares (WS)**
   It is an estimate number of wins contributed by a player. Win shares is the name of the metric developed by James in his book. It considers statistics for baseball players, in the context of their team and in a sabermetric way, and assigns a single number to each player for his contributions for the year. Justin Kubatko applied the win shares concept to basketball players. It is different from PER in terms of the formula they use to calculate the impact of a player, however, win shares is also an "all in one" stat that factors in the overall statistics of each individual player.

In this study, I will attempt to find out what is the best way to measure the impact of a player by weighing which statistics is more accurate in measuring each player's impact on **winning.** In turn, a new statistics **"True Impact" (TI)** can then be associated with each player. With TI and the information on each player's salary, NBA GMs would be able to find out who are some of the underpaid/overpaid players in the league. In the future, they will know better about a true value a player has on the team's success, and make smarter decisions in the amount that they decide to pay him.

Research Questions

1. How to properly rank the two “all in one” statistics, Win Shares, and PER. Which is better in measuring a player’s contribution to the team’s success?

   Result:
   Winning percentage of players whose PER ranks are much lower than their WS ranks (better PER stat) = 0.341463414634

   Winning percentage of players whose WS ranks are much lower than their PER ranks (better WS stat) = 0.585365853659

   From this result, we can conclude that Win Shares(WS) is the more accurate statistic in measuring a player's impact on winning as for players with much lower rank number (ranks great) of Win Shares(WS), their winning percentage is better than that of the players who have lower rank number of PER.

2. How to come up with a more accurate “all in one” stat, correctly measuring a player’s true impact by making use of both Win Shares and PER?

   Result:
   True Impact(TI) is a new “all in one” stat can be used to measure a player’s true impact on winning.
   *TI’s formula: weightedPER * player's PER + weightedWS * factor * player's WS.*

3. Who are some of the most underpaid/overpaid players in the NBA?

   Result:
   Underpaid players (> $10000000 difference): ['John Collins', 'Will Barton', 'Spencer Dinwiddie', 'Jerami Grant', 'Jamal Murray', 'Joel Embiid', 'Montrezl Harrell', 'Gary Harris', 'Kyle Anderson', 'James Harden', 'Nikola Jokic', 'Donovan Mitchell', 'Clint Capela', 'Anthony Davis']

   Overpaid players (> $10000000 difference): ['Joakim Noah', 'Solomon Hill', 'Harrison Barnes', 'George Hill', 'Mike Conley', 'Carmelo Anthony', 'Paul Millsap', 'Marc Gasol', 'Tony Parker', 'Brook Lopez', 'Timofey Mozgov', 'Kent Bazemore', 'Wesley Matthews', 'Nicolas Batum', 'Stephen Curry', 'Kawhi Leonard', 'Reggie Jackson', 'Jeremy Lin', 'Luol Deng', 'Omer Asik', 'Gordon Hayward', 'Chandler Parsons', 'Danilo Gallinari', 'Ian Mahinmi', 'Allen Crabbe', 'Blake Griffin'].

Dataset + Reproducing Result

Data:

(2017-2018-nba-players-stats-advanced)
https://drive.google.com/file/d/1n_2OgwhB28WexV-pbPKYqkUQKKfnuMXa/view?usp=sharing

(2017-2018-nba-team-record)
https://drive.google.com/file/d/1JOKg_NNxpQmhyQqUs4YKXxoNyBmq9RQ7/view?usp=sharing

(nba-players-salary)
https://drive.google.com/file/d/1ZDgr2iSd8A5Uz6OT6Mwobuhe90xPp24s/view?usp=sharing

To reproduce the results and analysis:

- Click into each link above and download each individual files.
- Save it in the same directory as final_project.py.
- Open up canopy terminal
- Change the directory to where all the files are at
- Type “python final_project.py”

Methodology

1st Question: How to properly rank the two "all in one" statistics, Win Shares, and PER. Which is better in measuring a player's contribution to the team's success?

- Read the data file using function read_csv
- Filter out rows that contains the same players using function filter_data (each player will only have one row)
- Extract players' PER numbers using function extract_data, e.g. {"Kevin Durant": 28.7…}
- Extract players' WS numbers using function extract_data
- Sort the ranks of the two stats and return with a dictionary mapping player to his rank difference using the functions sorting_rank and find_differential
- Focus on players whose absolute number of the rank difference is > 80 (out of 400 players in the league, 20% difference) using the function find_high_deviation
- Read the team data file and map each player to his team using the functions process_team_record_data and extract_data
- Calculate the winning percentages for players who have much higher PER rank number and players who have much higher WS rank number (difference > 80) using the function calculate_winning_percentage
- Compare the two different winning percentages and conclude which statistic is a better measurement of a player's impact on the court using the function compare_winning_percentage

2nd Question: How to come up with a more accurate "all in one" stat, correctly measuring a player's true impact by making use of both Win Shares and PER?

- Calculate how much weightage should we give to PER and WS in the computation of a player's True Impact(TI) using calculate_weighted_stat

- Find out what is the factor to multiply to each player's Win Shares in calculating TI by dividing the league average PER to league average WS. This is so that an average player's PER and WS value will be of roughly equal magnitude. This can be done by using the function calculate_stat_factor

- Compute each player's TI value and map each player to his TI value This can be done by using the function compute_player_ti

3rd Question: Who are some of the most underpaid/overpaid players in the NBA?

- Read player's 2017-2018 salary data using the functions read_csv, extract_data and convert_string_to_float

- Focus on players that appear on both the stat dataset as well as the salary dataset using the functions find_datasets_intersection and filter_dictionary

- Compute how much each player's TI value deviates from the league average TI using compute_stat_index

- Compute the deserved salary each player should get based on his TI index using the function compute_players_salary_deserved

- Compute the salary difference (deserved – actual) for each player using the function find_differential

- Find the list of overpaid/underpaid players using the function find_high_deviation_players

Work Plan Evaluation

1st Question:

Initial plan (Estimated time to complete: 8 hours) –

- Create two lists of data. One with all the strings of NBA players, the other with their respective WS/PER
- Sort the list of WS/PER from greatest to smallest.
- Find out each player's ranks for WS/PER and their rank differential.
- Store them in a tuple e.g. (5, 10, -5)
- Map each player to their respective tuples.
- Create two empty dictionaries, one for players with positive differential number, one for that of negative numbers.
- If the absolute number of the player's differential is significant, store the players into the empty dictionaries according to their sign of differential number.
- Find out each player's team's winning percentage using the data set 2017-18 NBA Standings. Compute the average the winning percentage out for the players in both the dictionaries.
- The corresponding stat (WS or PER) in which the players in the dictionary with an overall better winning percentage is the better stat.

Evaluation:

Instead of creating two lists of data, it was much easier for me to create a function called extract_data to extract the relevant data that I needed for each player and store them into a dictionary. (Key: player, Value: player's stat)

I also managed to use this function multiple times and it just makes my code much more flexible.

Instead of storing the player's ranks and differential in a single tuple, I created a function called find_differential, which was also used later on, in order to map each player to the difference in the ranks of his two stats.

I definitely underestimated how hard it was to calculate the winning percentage of the players with significant rank difference. I had to "fit in" the two different files in order to calculate the winning percentages.

Nonetheless, the result turned out to be just as what I was expecting.

2nd Question:

Initial plan(Estimated time to complete: 8 hours) –

- Calculate how much (in %) the better overall winning percentage is higher than that of the other percentage. (winning % - losing %) / losing % * 100. This will be stored as better_stat_percent.

- weightedPER/weightedWS = 0.5 * better_stat_percent + 0.5
  weightedWS/weightedPER = 1 – weightedPER/weightedWS

- Sum up the total win shares of all the players and find out the average by dividing by the number of players. (sum_ws / num_players)

- WSindex calculation: (player's WS – league average WS) / league average WS * 100
  PERindex calculation: (player's PER – 15) / 15 * 100

- Map each player to their TI value.
  To calculate TI value of a player: WSindex * weightedWS + PERindex * weightedPER

Evaluation:

I was able to almost closely follow this work plan except when I was calculating the players' TI values. Initially, I wanted to multiply weightedWS with WSindex and weightedPER with PERindex. PERindex and WSindex are the percentage of deviation a player's stat is from the league average. However, I found out that this equation of calculating TI did not get me anywhere as I am taking both the positive and negative deviation from the league average. Therefore, summing up, the league average TI will just be zero. Hence, I changed the way in calculating a player's TI.

Firstly, I still found out the league average's WS and PER stat using the function calculate_league_average. Then, I calculated what is the multiple that the league average PER is higher than that of WS. This is called the "factor". Instead of using indexes to calculate TI, I used the actual values of player's PER and WS, with the exception that players' WS stat will be multiplied by "factor". This is to ensure that the numerical values of PER and WS are on roughly equal level. With the weightage tuple I created using calculate_weighted_tuple, I was able to find out how much weight I should put into PER/WS in calculating the player's TI value. I then followed the plan to map each player to his TI value.

This question took me exceptionally long as I was trying to figure out what is wrong with my formula.

3rd Question:

Initial plan (Estimated time to complete: 8 hours) –

- Compute the average salary by looping through the salary column in 2017-2018 NBA Player Contracts (avg_salary = total_salary / num_players)

- Compute the average TI (avg_TI = total_TI / num_players)

- Map each player to their salary_index (salary_index = (player's salary – avg_salary) / avg_salary * 100)

- Map each player to their TI_index (TI_index = (player's TI – avg_TI / avg_TI * 100)

- Map each player to how much he should be earning based on his salary_index and TI_index

- Compare it with the dictionary which maps each player to what he is actually earning.

- Create a list of players with the greatest discrepancies.

Evaluation:
Before doing this project, I made the assumption that there will be the same number of players in the advanced stat and player salary datasets. However, I found out that some of the players in the advanced stat did not appear in player salary dataset, and vice-versa. As such, I had to create functions find_datasets_intersection as well as filter_dictionary in order to just focus on players that appear in both the datasets.
I computed the average salary and the average TI using the function calculate_league_average. I did not include (* 100) in calculating my index as the indexes should be in percentage (40% = 0.4).
I created a function called compute_players_salary_deserved which takes in the league average salary and the TI index dictionary to map each player to how much he should be paid according to his TI value. I then followed the plan to find out the difference in how much a player deserves against how much he is actually earning.

Overall:
Overall, to execute this work plan is much harder than what I have expected, and I definitely spent much more time than what I thought I would. There are a lot of problems in a dataset that may seem minor to solve and continue executing my plan, however, some of them are most certainly not and it requires extra effort and time to get rid of the problems.

Testing

The testing that I have done for this project is to create a testing file (final_project_testing.py).

I used assert statements for most of the functions that I am testing. This is because my program is mostly made up of functions that will give me a calculation or computed number. To ensure that it works, I tested the individual function to see if it works as described on my function DocString. I plugged in some numbers and manually calculated the result and made sure that my function gave me the same result.

Live Presentation or Video

I will be doing a live presentation.

Collaboration

This project is entirely done by myself with little help from the TAs as they are not familiar with my project.