# Finding Best NBA Players & Teams

CSE 163 Final Project

Spencer James Knapp & Isaac Xu

## II. Introduction

Our research questions have been modified slightly to reflect how our project evolved over the course of this project. Each of the below summaries includes an *"evolution"* section which describes how the question changed between Part 1 and Part 2 of our project. The main reason for these changes is that we want to better reflect the work we actually performed in this report.

1. **What stats are the most important Key Performance Indicators (KPIs) to ranking players from best to worst? Spencer**

   *Summary:*

   We began with exploring what stats are available from calls made to the NBA-API. Our intention was to use the *.get_active_players()* call to first find all active players, and then to construct a dataframe with each player's associated stats. Initially we felt that team winning %, player =/- (plus-minus), and eFG% (effective field goal percentage) would be our most valuable stats to use as KPIs. Our logic applies the formula for each player to find out their score and fill out a new column "Player Score" with these scores. The score column will help us find out the best player based on the highest score and is ultimately used to assign ranks for players. These ranks will later be used as the label for the player ML model which is explored further in our second research question.

   *Evolution:*

   The main evolution between this question and our Part 1 Proposal is that we realized that determining the correct stats to use as our KPIs is a critical first step for our accurately determining player ranks and building team ranks upon that. After discussing with Alex, our TA mentor for this project, it became clear that our first question was centered around first trying out various combinations of KPIs in order to build the most robust and useful player scores to later use in rankings.

   *Results:*

   Here is a table showing the KPIs we established after iterating on which KPIs to include/exclude as well as what weights to assign for each KPI. This is meant as a summary, further details of our process are explained in our methodology section. We arrived at these weight values after testing 5 different combinations of weights (beginning with equivalent weights). Based on our findings, Total_Points, Player+/-, and eFG% are the most relevant KPIs (and therefore have higher respective weights).

*Final KPI Breakdown with Weights*

| Name | Type | Weight |
|---|---|---|
| Player + / - | Float | .25 |
| eFG % | Float | .25 |
| PlayerWinning% | Float | .05 |
| Total_Games* | Int | .05 |
| Total_Points* | Int | .4 |

**Key Changes Made**

- **Seasons Included:** Based on our iterations we elected to include only player stats from the 2018-2019, 2019-2020, and 2020-2021 seasons. We assign higher weights to the more recent seasons under the notion that more recent performances are better indicators of how a player will perform in the future.

*Season Weights Breakdown*

| Season | Weight |
|---|---|
| 2018-2019 | .05 |
| 2019-2020 | .2 |
| 2020-2021 | .75 |

2. **How can we use a player's regular season stats to assign ranks for previous seasons and predict ranks for the current season? Spencer**

*Summary:*

After refining our KPIs and weights to compute the player score metric, we next rank players based on these scores. Then we build a machine learning regression model to put our player scoring algorithm to use. Using the player scores determined from our first research question (described above and expanded on in the methodology section) we rank players based on their total_score, which is the sum of all scores combined. We opted to use sum across seasons instead of average or median because we found this to provide the most accurate output. Next we built a regression model using scikitlearn's DecisionTreeRegressor to predict player ranks for the current season. Our methodology section further explains how we built this model using train/test sets/adjusting the max depth of the decision tree.

*Evolution:*

The key changes to this research question section compared to our Part 1 Proposal is how we realized the need to output not only our rankings, but also the need to output an image of our decision tree in order to show how our model works. Furthermore, we realized that our decision tree needed integer ranks, as well as that setting a max_depth of 3 allows for a more readable output (compared to no max_depth) which shows how each ranking is decided, but becomes difficult to read as there are hundreds of players.

*Results:*

Here we see the results after determining player rankings (this summary output is only showing for the top 20 players).

| player_name | Rank |
|---|---|
| Giannis Antetokounmpo | 1 |
| James Harden | 2 |
| Damian Lillard | 3 |
| Kawhi Leonard | 4 |
| LeBron James | 5 |
| Luka Doncic | 6 |
| Anthony Davis | 7 |
| Devin Booker | 8 |
| Bradley Beal | 9 |
| Jayson Tatum | 10 |
| Khris Middleton | 11 |
| Nikola Jokic | 12 |
| Joel Embiid | 13 |
| Paul George | 14 |
| Donovan Mitchell | 15 |
| Kyrie Irving | 16 |
| Trae Young | 17 |
| Pascal Siakam | 18 |
| Rudy Gobert | 19 |
| Russell Westbrook | 20 |

3. **How can we determine team rankings using an aggregate of player rankings and predict ranks for the current season? Isaac**

*Summary:*

We start with combining the data frame with player rankings with the current nba team roster by using merge functions. After merging player and team dataframes, we apply groupby with team to aggregate players' scores in each team and use "mean()", "median", and "sum()" functions to get total, average, and median scores of each team. For predicting rankings of the current season, we create ML regression models in our player_team_model file and use "Rank_Overall" as labels and rest columns as features.

*Evolution:*

The key change in this question is how we get the data frame for training and testing our ML model. We firstly decided to use team stats, for example, team winning percentage, team pts per game, and etc, to calculate team scores. After we read the feedback from our mentor, we decided to aggregate player scores to calculate our team score.

*Results:*
Here we see the results after determining team rankings (this summary output is only showing for the top 20 players).

| TEAM | Rank_Overall |
|---|---|
| Uta | 1 |
| Mil | 2 |
| Bro | 3 |
| Lac | 3 |
| Lal | 5 |
| Phi | 6 |
| Bos | 7 |
| Por | 8 |
| Mia | 10 |
| Atl | 10 |
| Den | 10 |
| Pho | 12 |
| Dal | 13 |
| Ind | 14 |
| Nor | 14 |
| Tor | 16 |
| San | 17 |
| Sac | 18 |
| Was | 19 |
| Cle | 20 |

The mean squared error of player model is within range 1-4 and for team model, the range is within 5-10. The visualization of this model is presented in the **Challenge Goals** section.

4. **How do our player and team rankings compare to other models (538) Isaac**

*Summary:*
We compare our results with results published in 538 as they offer similar rankings for NBA players and teams. Based on our research into their documentation, 538's ranking algorithms provide some of the most accurate publicly available ranks and believe that comparing their ranks to ours is a strong indication of how accurate our own ranks are. We further describe the implications of this comparison in our methodology section.

*Results:*
This section showcases the side-by-side comparison between our rankings for players and teams compared to 538's respective rankings. The images are on the next page for readability.

*Player Rankings:*

Our Rankings:

| player_name | Rank |
|---|---|
| Giannis Antetokounmpo | 1 |
| James Harden | 2 |
| Damian Lillard | 3 |
| Kawhi Leonard | 4 |
| LeBron James | 5 |
| Luka Doncic | 6 |
| Anthony Davis | 7 |
| Devin Booker | 8 |
| Bradley Beal | 9 |
| Jayson Tatum | 10 |
| Khris Middleton | 11 |
| Nikola Jokic | 12 |
| Joel Embiid | 13 |
| Paul George | 14 |
| Donovan Mitchell | 15 |
| Kyrie Irving | 16 |
| Trae Young | 17 |
| Pascal Siakam | 18 |
| Rudy Gobert | 19 |
| Russell Westbrook | 20 |

538 Rankings:

| | PLAYER |
|---|---|
| 1 | Nikola Jokic '20-'21 |
| 2 | Joel Embiid '20-'21 |
| 3 | Giannis Antetokounmpo '20-'21 |
| 4 | Stephen Curry '20-'21 |
| 5 | LeBron James '20-'21 |
| 5 | Kawhi Leonard '20-'21 |
| 7 | Fred VanVleet '20-'21 |
| 8 | Julius Randle '20-'21 |
| 9 | Luka Doncic '20-'21 |
| 9 | Paul George '20-'21 |
| 9 | Rudy Gobert '20-'21 |
| 12 | Mike Conley '20-'21 |
| 13 | Jimmy Butler '20-'21 |
| 13 | Clint Capela '20-'21 |
| 15 | James Harden '20-'21 |
| 16 | Jamal Murray '20-'21 |
| 17 | Damian Lillard '20-'21 |
| 18 | Trae Young '20-'21 |
| 19 | Bradley Beal '20-'21 |
| 19 | Myles Turner '20-'21 |
| 21 | Jordan Clarkson '20-'21 |

*Team Rankings:*

Our Rankings:

| TEAM | Rank_Overall |
|---|---|
| Uta | 1 |
| Mil | 2 |
| Bro | 3 |
| Lac | 3 |
| Lal | 5 |
| Phi | 6 |
| Bos | 7 |
| Por | 8 |
| Mia | 10 |
| Atl | 10 |
| Den | 10 |
| Pho | 12 |
| Dal | 13 |
| Ind | 14 |
| Nor | 14 |
| Tor | 16 |
| San | 17 |
| Sac | 18 |
| Was | 19 |
| Cle | 20 |

538 Rankings:

Nets
Clippers
Bucks
Lakers
Nuggets
Jazz
76ers
Suns
Celtics
Heat
Raptors
Hawks
Mavericks
Trail Blazers
Pacers
Hornets
Warriors
Grizzlies
Pelicans
Spurs
Knicks

## II. Motivation & Background

It is common to hear NBA fans or commentators debate which is the strongest team or who is the best player in the ongoing NBA season but there are still not solid answers for these questions. This project analyzes data from each active player to conclude a solid answer to these debates. Media, fans, or commentators also always like to predict the champions of the season and they give their opinion highly based on their impression of the team. This project would also use stats of players of teams of this season to predict the championship of 2020-21 using a new approach involving building two independent models (player performance and team rankings), iterating on these models.

## III. Datasets

| *Name* | *Source* | *Used For* |
|---|---|---|
| **NBA-API** | https://pypi.org/project/nba-api/ | Acquiring player rosters, gamelogs, stats (see below) |
| **NBA Player Roster** | https://www.nbastuffer.com/2020-2021-nba-player-stats/ | Assigning players to teams with data merge (see below) |

**NBA-API**

Most datasets used in this project come from querying nba-api (version 1.19). Before scraping NBA data, nba_api should be installed. One way to install is to input "pip install nba_api" into Anaconda Prompt and Anaconda Prompt would download the package. The dataset would be acquired by using modules from nba_api. For example, to acquire the dataset about teams stats of the current season:

```
from nba_api.stats.endpoints import leaguedashteamstats

league_data = leaguedashteamstats.LeagueDashTeamStats()
league_df = league_data.get_data_frames()
```

This would acquire the specified dataset and turn this dataset into a readable data frame.
Datasets would be used for this project:
Player Career Stats:

```
from nba_api.stats.endpoints import playercareerstats

bron_career_stats =
playercareerstats.PlayerCareerStats(player_id)
bron_career_stats_df = bron_career_stats.get_data_frames()
```

Player Awards:

```
from nba_api.stats.endpoints import playerawards

awards = playerawards.PlayerAwards(player_id)
awards_df = awards.get_data_frames()
```

Specific Team of a Specific Year:

```
from nba_api.stats.endpoints import teamdashboardbyyearoveryear

team_details =
teamdashboardbyyearoveryear.TeamDashboardByYearOverYear(team_id)
team_details_df = team_details.get_data_frames()
```

How to get player and team id:

```
from nba_api.stats.static import players
from nba_api.stats.static import teams

player_dict = players.get_players()
team_dict = teams.get_teams()

player = [player for player in player_dict if
player['full_name'] == 'Player_name'][0]
player_id = player['id']

team = [team for team in team_dict if team['full_name'] ==
'team_name'][0]
team_id = team['id']
```

## IV. Methodology

In this section we explain our methodologies for determining KPIs, assigning ranks, building a machine learning model for player ranks, and building a machine learning model for team ranks.

*Determining KPIs*

As described in our research question #1 section, we have made modifications to our methodology for deciding which KPIs should be included in our player score algorithm. This involved iterating on our algorithm as we adjusted weights, added in new KPIs, and checked our results.

| Iteration # | KPIs | Weights (respective) |
|---|---|---|
| 1 | +/-, eFG%, win% | All .333 (equal) |
| 2 | +/-, eFG%, win% | .7, .2, .1 |
| 3 | +/-, eFG%, win% | .75, .25, .05 |
| 4 | +/-, eFG%, win%, Total_Points | All .25 (equal) |
| 5 | +/-, eFG%, win%, Total_Points | .6, .15, .05, .2 |
| 6 | +/-, eFG%, win%, Total_Points | .5, .1, .05, .35 |
| 7 | +/-, eFG%, win%, Total_Points, Total_Games | All .2 (equal) |
| 8 | +/-, eFG%, win%, Total_Points, Total_Games | .4, .3, .05, .05, .2 |
| 9 | +/-, eFG%, win%, Total_Points, Total_Games | .25, .25, .05, .05, .4 |

*Determining Scope of Seasons and Weights:*
Also described in our research question #1 section is our modification of scope to only take in stats from the 2018-2019, 2019-2020, and 2020-2021 regular seasons. We realized the need for this after including the Total_Points and Total_Games KPIs. Without controlling for seasons, our ranks would become skewed by active players who may have had exceptional careers over many years, but are not top-performing current players. In order to protect against this, we opted to only include the most recent three seasons (including the current season). We also realized the need to add weights for each season to account for the most recent season providing the most accurate indication of how a player can be expected to perform in the future.

On the next page is a table documenting our iterative process for determining the weights for each season:

| Iteration # | Seasons | Weights (respective) |
|---|---|---|
| 1 | All | n/a |
| 2 | Last 3 | All .333 (equal) |
| 3 | Last 3 | .5, . 3, .2 (recent-less recent) |
| 4 | Last 3 | .75, .2, .05 (recent-less recent) |

*Assigning Ranks*
We found using sum of scores to be the best for ranking players. This is after comparing our rankings when using sum of score across all seasons, vs average of score, vs median of scores. We arrived at a similar conclusion for ranking teams. For teams we only take the highest ten player scores in a team (sum of all player scores, not average or median scores of players). This is optimal because we implemented controls to only include top 10 players instead of all active players on a roster. This is further described in our testing section.

*Building Player Model(Machine Learning):*
This is an example of how we will approach research question #1.

*Scope:* We are analyzing player performance in the regular seasons of 2018-2019, 2019-2020, and 2020-2021 to predict player rankings for the current season.

*KPIs (Key performance indicators):*

| Name | Type | Feature | Label |
|---|---|---|---|
| Player + / - | Float | True | False |
| eFG % | Float | True | False |
| Team Winning% | Float | True | False |
| Total_Games | Int | True | False |
| Total_Points | Int | True | False |
| Rank | Int | False | True |

*Our Process:*

1. Query NBA API to get KPI data for players
   Use the NPA API to obtain necessary player information. This will include getting all active players and all respective KPI information. These KPIs (three are identified in the above table) will be columns of data we want to pull from the NPA API.
2. Format/clean data as needed
   Depending on how the data is structured from the NBA API, we may need to reformat data or make joins between different datasets to capture all relevant player stats for our model. This step will also involve cutting out any unnecessary columns to improve the efficiency of our program. We will also handle missing data, NaNs, other issues during this step. As we are not working with any geospatial data we believe the Pandas library will be sufficient for these activities.
3. Create ML regression models for all KPIs
   Once we have our consolidated active player dataset we will split our data into train and test sets. We are aiming to rank players based on their performance of the most recent three years regular season. The trained model would be used to predict rankings of active players.
4. Create weighted Player Score ML model
   Next we will add weights to all of our KPIs, beginning with equal weights, and adjusting based on our intuition for each iteration. By adjusting weights we are attempting to gauge which KPIs have the highest impact on a player's overall ranking. Here is an example of this approach:
   - Test out x sets of weights
   - Team Score = ((plus_minus * weight1) + (eFG_% * weight2) + (Team Winning% * weight3))

   - Where sum of all weights must = 100%
5. Repeat 1-4 with different KPIs (or add in) and compare outcomes of each model
   We will trest out our model using different combinations of KPIs and assess mean squared error and accuracy scores in order to determine which KPIs and weights produce the best results.

*Using Player Models to find Team Ranks:*

This is an example of how we will approach question #2. We are including this section to showcase our approach to building a team ranking model.

*Our process:*

1. Gather player scores of players in the same team:
   Get weighted player scores from the ML model created from question #1 and aggregate them together to get the total player score from that team. This requires a data merge

between our player scores dataframe and an external CSV with players and respective teams. We must make the merge in order to be able to group players on teams.

2. Further adjust player scores:
   In this example, we calculate total, average, and median team scores and rank teams accordingly. The step would be repeated because we are also going to rank the team by their total team scores (sum of players for that team), median team scores (median of players for that team), and average team scores (mean of players for that team). We will compare these results with our intuition and 538 to determine which team score provides the most accurate rankings. **In our project we found the sum of player scores to be the preferred score to rank teams on as it provided the most accurate rankings.**

3. Create ML regression model for teams
   Features are team scores teams and labels are team ranks from step 2.

## V. Results

We have two main sections for our results, rankings and models, that each have two respective sections for players and for teams.

*Rankings:*

This section will cover our results for player and team rankings respectively.

*Players:*

Top 20 Players Ranked Best-Worst

| | player_name | plus_minus | eFG_avg | ... | total_score | avg_score | Rank |
|---|---|---|---|---|---|---|---|
| 1 | Giannis Antetokounmpo | 9.097222 | 59.903769 | ... | 41.530826 | 13.843609 | 1 |
| 0 | James Harden | 4.743590 | 54.059717 | ... | 40.496034 | 13.498678 | 2 |
| 7 | Damian Lillard | 6.175000 | 52.152642 | ... | 37.824662 | 12.608221 | 3 |
| 5 | Kawhi Leonard | 5.950000 | 54.561559 | ... | 37.669108 | 12.556369 | 4 |
| 8 | LeBron James | 2.109091 | 56.027397 | ... | 37.374429 | 12.458143 | 5 |
| 55 | Luka Doncic | -1.597222 | 49.747049 | ... | 36.749155 | 12.249718 | 6 |
| 14 | Anthony Davis | 2.357143 | 53.996101 | ... | 36.092060 | 12.030687 | 7 |
| 37 | Devin Booker | -5.187500 | 52.071713 | ... | 35.584701 | 11.861567 | 8 |
| 15 | Bradley Beal | -1.426829 | 53.977626 | ... | 35.083454 | 11.694485 | 9 |
| 57 | Jayson Tatum | 4.493671 | 50.579151 | ... | 34.965551 | 11.655184 | 10 |
| 23 | Khris Middleton | 7.272727 | 51.872822 | ... | 34.963313 | 11.654438 | 11 |
| 19 | Nikola Jokic | 3.950000 | 54.519071 | ... | 34.561910 | 11.520637 | 12 |
| 6 | Joel Embiid | 5.812500 | 51.668057 | ... | 34.504028 | 11.501343 | 13 |
| 4 | Paul George | 6.363636 | 52.850062 | ... | 34.418038 | 11.472679 | 14 |
| 12 | Donovan Mitchell | 4.779221 | 49.346405 | ... | 34.225750 | 11.408583 | 15 |
| 9 | Kyrie Irving | 4.865672 | 55.680902 | ... | 34.061027 | 11.353676 | 16 |
| 86 | Trae Young | -4.123457 | 48.009554 | ... | 33.814643 | 11.271548 | 17 |
| 18 | Pascal Siakam | 7.425000 | 59.100529 | ... | 33.709954 | 11.236651 | 18 |
| 16 | Rudy Gobert | 4.666667 | 66.853933 | ... | 33.648379 | 11.216126 | 19 |
| 25 | Russell Westbrook | 3.972603 | 46.809233 | ... | 33.333332 | 11.111111 | 20 |

*Interpretation:*

This data frame of player rankings is ranked based on total_scores.

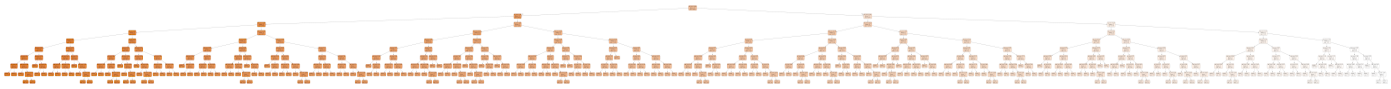
*Implications:*
This dataframe mostly makes sense but there still some players are overrated here, for example, Devin Booker. The reason is since we put weights for player scores of different years and the weight for 2020 is highest. Suns plays pretty well in this season and Devin Booker also has very good stats so this is the reason why he looks overrated here.

*Teams:*
Ranks of all teams, 'Rank_Overall' column is on far right

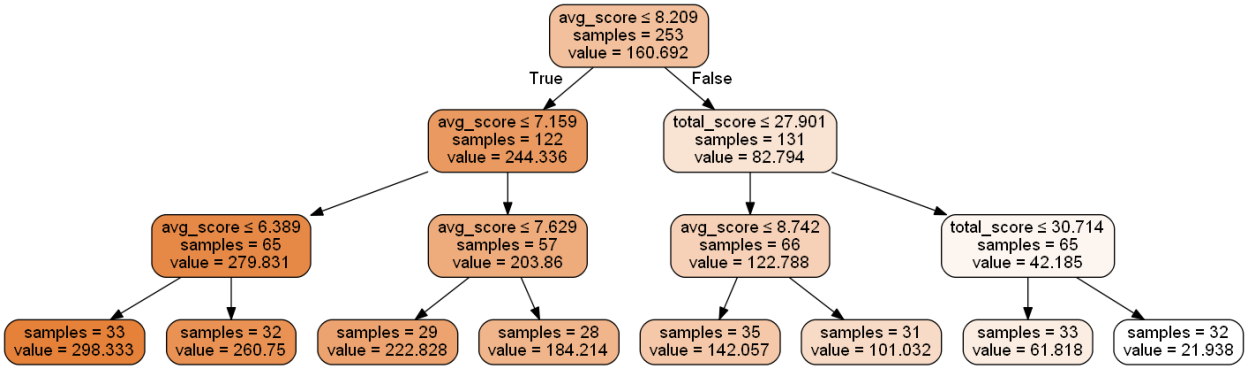
Top 20 Teams Ranked Best-Worst

| TEAM | Sum | Average | Median | ... | Rank_Med | Rank_Avg_All | Rank_Overall |
|---|---|---|---|---|---|---|---|
| Uta | 254.931052 | 25.493105 | 25.262659 | ... | 1.0 | 1.000000 | 1 |
| Mil | 253.874802 | 25.387480 | 24.246399 | ... | 6.0 | 3.333333 | 2 |
| Bro | 246.530679 | 24.653068 | 25.140660 | ... | 2.0 | 4.333333 | 3 |
| Lac | 247.814742 | 24.781474 | 24.633438 | ... | 5.0 | 4.333333 | 3 |
| Lal | 248.005000 | 24.800500 | 23.868563 | ... | 9.0 | 5.000000 | 5 |
| Phi | 245.372255 | 24.537226 | 24.150088 | ... | 7.0 | 6.666667 | 6 |
| Bos | 222.394812 | 24.710535 | 25.015613 | ... | 4.0 | 7.666667 | 7 |
| Por | 238.694792 | 23.869479 | 23.235990 | ... | 11.0 | 9.666667 | 8 |
| Mia | 218.021715 | 24.224635 | 24.081768 | ... | 8.0 | 11.333333 | 10 |
| Atl | 231.002872 | 23.100287 | 22.860395 | ... | 12.0 | 11.333333 | 10 |
| Den | 219.233785 | 24.359309 | 23.648617 | ... | 10.0 | 11.333333 | 10 |
| Pho | 231.715515 | 23.171551 | 22.093558 | ... | 18.0 | 12.666667 | 12 |
| Dal | 230.838624 | 23.083862 | 22.369660 | ... | 16.0 | 13.333333 | 13 |
| Ind | 229.763492 | 22.976349 | 22.382763 | ... | 15.0 | 13.666667 | 14 |
| Nor | 168.380833 | 24.054405 | 25.062777 | ... | 3.0 | 13.666667 | 14 |
| Tor | 228.442546 | 22.844255 | 22.642581 | ... | 14.0 | 14.000000 | 16 |
| San | 222.766613 | 22.276661 | 22.852285 | ... | 13.0 | 14.666667 | 17 |
| Sac | 221.279857 | 22.127986 | 22.125937 | ... | 17.0 | 17.000000 | 18 |
| Was | 214.751734 | 21.475173 | 21.068305 | ... | 22.0 | 20.333333 | 19 |
| Cle | 212.922509 | 21.292251 | 21.292479 | ... | 20.0 | 21.000000 | 20 |

*Interpretation:*
Team rankings data frame is ranked based on the average score of 'Sum', 'Average', and 'Median' columns.
*Implications:*
Denver Nuggets looks underrated here. Although it has Jokic, a super star center, and Jamal Murry, a very excellent guard, other players in this team are not that good so their player scores would not be good. That's why they are underrated here.
***Decision Tree Classification***
*Players:*
*Not setting max steps:*



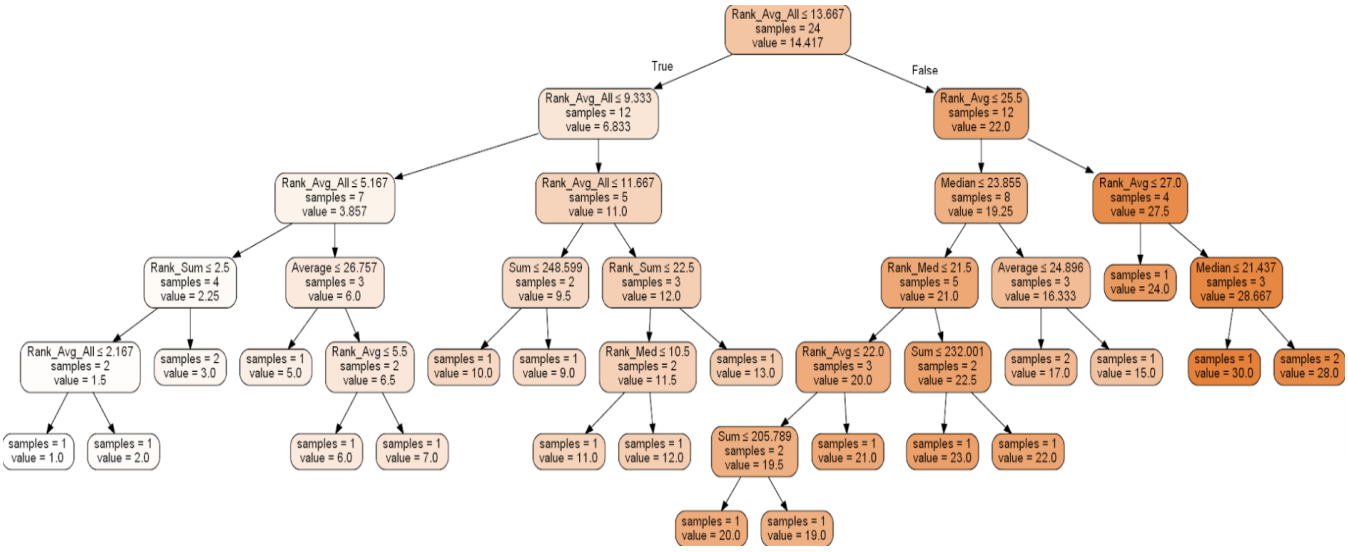
*Max steps = 3*




*Interpretation:*
This is the decision tree of the player model which only shows the result till depth 3 which reflects how player model predicts player rankings.

*Implications:*
The player model picks a player whose stats and scores are on average. For those whose avg scores are lower than that going left and greater than that going right. The similar processor would iterate over and over again until leaf nodes come out.

***Teams:***




*Interpretation:*
This is the decision tree of team models which reflects how team models predict team rankings.

*Implications:*
The team model picks a team whose scores are on average. For those whose Rank_Avg are lower than that going left and greater than that going right. The similar processor would iterate over and over again until leaf nodes come out.

## VI. Challenge Goals

In this final project, we completed two challenge goals, **Multiple Datasets** and **Machine Learning**.

**Multiple Datasets:**
There are four datasets we are using in this final project, "player_2018"(active player stats of 2018), "player_2019"(active player stats of 2019), "player_2020"(active player stats of 2020), and "teams_players"(team roster). To find out total and average player scores in the most recent three years. We merge three datasets together and calculate the total and average score of each player and put these data in new columns, "total_score" and "avg_score". We also apply the "sort_value" and "rank" formula for this merged data frame in order to find out players' ranking. We also merged the data frame included players' total and average scores in three years with the team roster data frame and we group by each team with player's total score and apply different group by functions, "sum()", "mean()", and "median", for this to get each teams total, average, and median scores. We also applied "sort_values" and "rank" functions for ranking each team in this dataframe.

**Machine Learning:**
In our final project, we complete ML challenges by creating new regression models and use data from merged player and team csv files we get from **Multiple Datasets** challenge to train our models and we also use data from these data frames to test our trained model. The test size is 0.2. We also use graphviz to illustrate our ML models(visualizations are above) to reflect who decisions are made in our ML model.

## VII. Work Plan Evaluation

We have updated our work plan (below) to reflect new tasks that were worked on during this project, actual time worked on tasks, and real status of tasks. Next we show summary statistics comparing our expected vs. actual time worked, concluded with a short reflection on why our project ended up taking about twice the time we anticipated.

*Please see next page for these updated tables (we moved them onto a separate page for readability)*

| Task | Description | Expected Time (hours) | Actual Time (hours) | Status (backlog, in-progress, needs review, complete) | Owner(s) |
|---|---|---|---|---|---|
| Create API query scripts for necessary data | - Player Data (current)<br>- Team Data<br>- Player Data (historical playoffs) | 4 | **3** | Complete | **Issac/Spencer** |
| Wrangle/Clean Data for use | (All datasets cleaned/joined as needed) | 2 | **7** | Complete | **Issac/Spencer** |
| Determine Player Scores | Aggregate stats, assign weights, compute scores | n/a | **3** | Complete | **Issac** |
| Rank Players | Compute ranks based on scores | n/a | **1** | Complete | **Spencer** |
| Determine Team Scores | Aggregate stats, assign weights, compute scores | n/a | **3** | Complete | **Spencer** |
| Rank Teams | Compute ranks based on scores (determine to use team sum, average, or median) | n/a | **1** | Complete | **Spencer** |
| Build, Train, Test Model 1 | Model for individual players to assign playoff ranks for current season | 2 | **1.5** | Complete | **Isaac** |
| Build, Train, Test Model 2 | Aggregate of player ranks on respective teams to assign team playoff ranks for current season | 2 | **1.5** | Complete | **Spencer** |
| Create README/ restructure code | Create .md file to explain running code and edit code to ensure solid documentation/pass flake8/follow project code guidelines | 2 | **5** | Complete | **Isaac** |
| Create Final Report | PDF Slide deck | 4 | **6** | Complete | **Isaac/Spencer** |

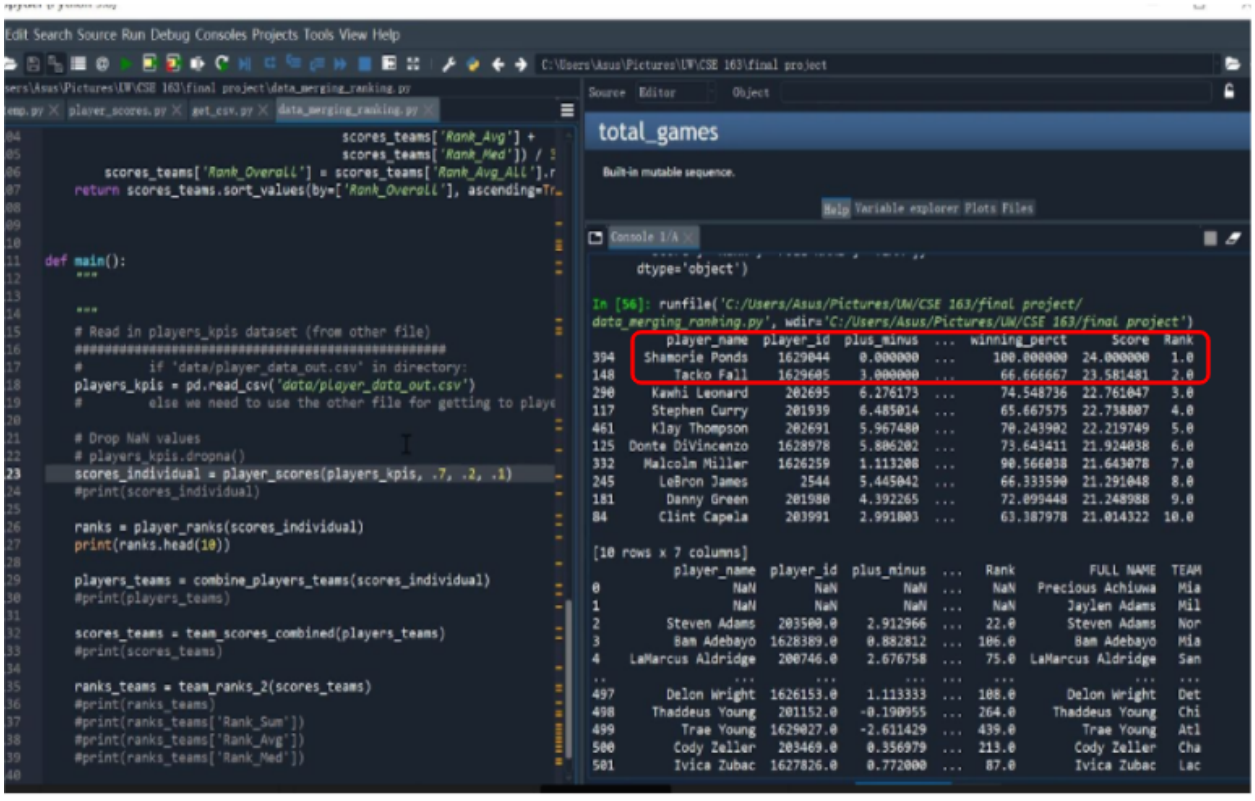
*Actual vs. Expected Summary:*

| Total Hours (Expected) | Total Hours (Actual) | Actual:Expected Ratio | % Actual of Expected |
|---|---|---|---|
| **16** | **32** | **2:1** | **200%** |

*Reflection on Actual vs Expected Hours:*
After finalizing our work plan post-project we found it interesting how our actual hours of work were twice as much as our expected hours. We feel that this can be explained by the additional work we had not expected in combining datasets to merge player data with team data, as well as the time we spent improving our KPIs to build a better score algorithm. Overall we feel this was worth the effort, as our rankings for players and teams are now far more robust. We also recognize that this is a lesson for future projects and work; be prepared to work longer than anticipated because sometimes "we don't know what we don't know."

## VIII. Testing

Testing for ranked player dataset:
At first, we only have three kpis, +/-, eFG, and winning percentage. Since the winning percentage is on scale 0-100 and there are players who don't play much game and he wins every game he plays so his winning percentage would be extremely high. This also applies to eFG. Even adjusting +/- to a larger weight. Players like Shamorie Ponds or Tacko Fall would still appear at the top of our ranking.



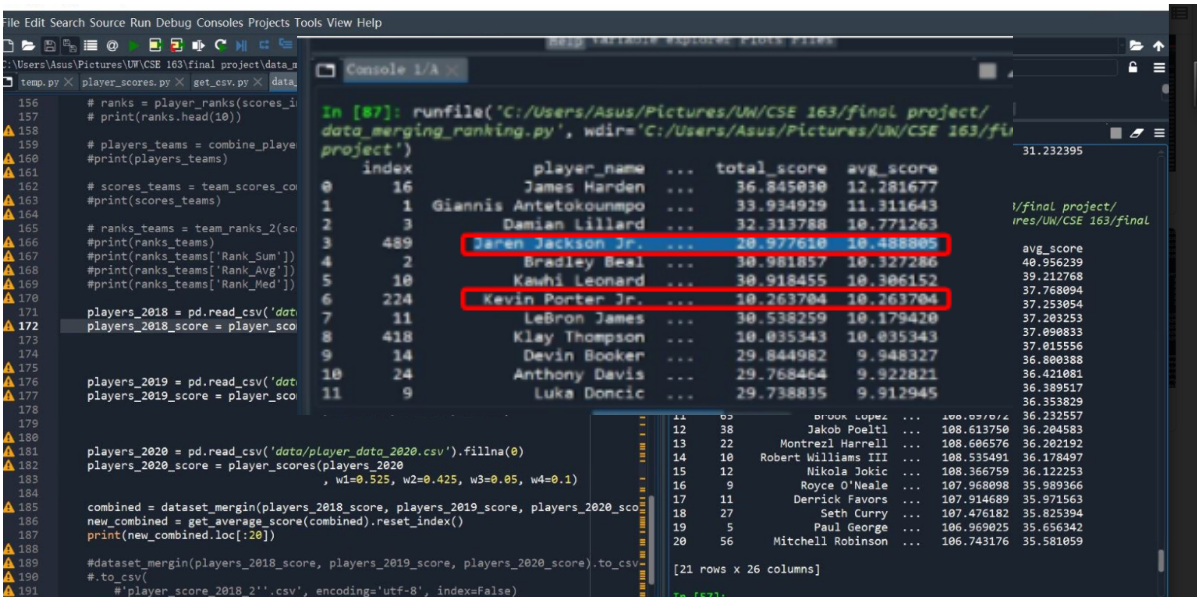

Therefore we add total_game as a new kpi to make sure these players would not appear at the top of our dataframe again.

```
In [11]: runfile('C:/Users/sknap/Documents/Academics/Winter 2021/CS 163/CSE163_Proje
2021/CS 163/CSE163_Project')
      player_name  player_id  ...  total_games        Score
87     Vince Carter       1713  ...         1541  2334.084482
245    LeBron James       2544  ...         1301  1978.687016
275     Kyle Korver       2594  ...         1232  1874.826312
232  Andre Iguodala       2738  ...         1163  1769.006005
92   Tyson Chandler       2199  ...         1160  1766.462317
..              ...        ...  ...          ...          ...
340     Matt Mooney    1629760  ...            4    16.700000
195    Jared Harper    1629607  ...            5    16.380000
379     Tariq Owens    1629745  ...            3    10.366667
266    Stanton Kidd    1629742  ...            4     9.650000
227  William Howard    1629739  ...            2     2.300000

[518 rows x 7 columns]
      player_name  player_id  plus_minus  ...  total_games        Score   Rank
87     Vince Carter       1713    1.178456  ...         1541  2334.084482    1.0
245    LeBron James       2544    5.445042  ...         1301  1978.687016    2.0
275     Kyle Korver       2594    2.314123  ...         1232  1874.826312    3.0
232  Andre Iguodala       2738    2.479794  ...         1163  1769.006005    4.0
92   Tyson Chandler       2199    0.280172  ...         1160  1766.462317    5.0
..              ...        ...         ...  ...          ...          ...    ...
340     Matt Mooney    1629760    1.750000  ...            4    16.700000  514.0
195    Jared Harper    1629607   -2.800000  ...            5    16.380000  515.0
379     Tariq Owens    1629745   -5.333333  ...            3    10.366667  516.0
266    Stanton Kidd    1629742   -0.250000  ...            4     9.650000  517.0
227  William Howard    1629739   -8.000000  ...            2     2.300000  518.0
```

Although players like Tack Fall are not showing here anymore, we are getting a new problem. Players like Vince Carter or Kyle Korver get extremely high scores. This is because they have played thousands of games in the league and 1000 is a really huge number comparing to winning%, eFG%, and +/- so we change the fourth kpi from total games throughout the whole career to the most recent three years because Vince Carter in 20 years ago is not the same Vince Carter as now. The first three kips also only counts the most three recent years based on the change of the fourth kpi.

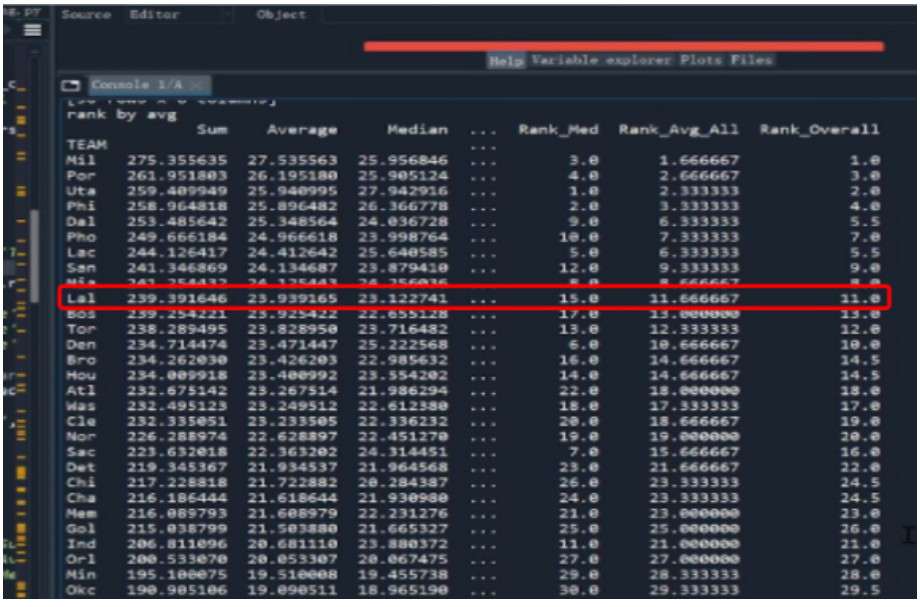
After we change the fourth KPI, the output starts to make sense. We can see a lot of all star players on this printed data frame but there are still many players that shouldn't be here so we introduce the fifth kpi, total points (pts) per game.



The output makes much more sense after introducing the fifth kpi but we always find out that Jaren Jackson Jr., Kenvin Porter Jr. are also showing in the out no matter how we change our weight. This is because we rank our data frame in avg_score instead of total_score so for players like Jaren Jackson Jr who played good before but not playing currently would not be shown here. Same reason for Kevin Porter Jr. who hasn't played much time in the league and we also adjust weights for player scores in each year, the more recent the year, the higher the weight. This is our final output of player rank:

| player_name | plus_minus | eFG_avg | ... | total_score | avg_score | Rank |
|---|---|---|---|---|---|---|
| Giannis Antetokounmpo | 9.097222 | 59.903769 | ... | 41.530826 | 13.843609 | 1 |
| James Harden | 4.743590 | 54.059717 | ... | 40.496034 | 13.498678 | 2 |
| Damian Lillard | 6.175000 | 52.152642 | ... | 37.824662 | 12.608221 | 3 |
| Kawhi Leonard | 5.950000 | 54.561559 | ... | 37.669108 | 12.556369 | 4 |
| LeBron James | 2.109091 | 56.027397 | ... | 37.374429 | 12.458143 | 5 |
| Luka Doncic | -1.597222 | 49.747049 | ... | 36.749155 | 12.249718 | 6 |
| Anthony Davis | 2.357143 | 53.996101 | ... | 36.092060 | 12.030687 | 7 |
| Devin Booker | -5.187500 | 52.071713 | ... | 35.584701 | 11.861567 | 8 |
| Bradley Beal | -1.426829 | 53.977626 | ... | 35.083454 | 11.694485 | 9 |
| Jayson Tatum | 4.493671 | 50.579151 | ... | 34.965551 | 11.655184 | 10 |
| Khris Middleton | 7.272727 | 51.872822 | ... | 34.963313 | 11.654438 | 11 |
| Nikola Jokic | 3.950000 | 54.519071 | ... | 34.561910 | 11.520637 | 12 |
| Joel Embiid | 5.812500 | 51.668057 | ... | 34.504028 | 11.501343 | 13 |
| Paul George | 6.363636 | 52.850062 | ... | 34.418038 | 11.472679 | 14 |
| Donovan Mitchell | 4.779221 | 49.346405 | ... | 34.225750 | 11.408583 | 15 |
| Kyrie Irving | 4.865672 | 55.680902 | ... | 34.061027 | 11.353676 | 16 |
| Trae Young | -4.123457 | 48.009554 | ... | 33.814643 | 11.271548 | 17 |
| Pascal Siakam | 7.425000 | 59.100529 | ... | 33.709954 | 11.236651 | 18 |
| Rudy Gobert | 4.666667 | 66.853933 | ... | 33.648379 | 11.216126 | 19 |
| Russell Westbrook | 3.972603 | 46.809233 | ... | 33.333332 | 11.111111 | 20 |

Team rank is derived based on player score dataframe. After we get decent output from player ranks we test our team rank and the output looks like this:




| TEAM | Sum | Average | Median | ... | Rank_Med | Rank_Avg_All | Rank_Overall |
|---|---|---|---|---|---|---|---|
| Mil | 275.355635 | 27.535563 | 25.956846 | ... | 3.0 | 1.666667 | 1.0 |
| Por | 261.951803 | 26.195180 | 25.905124 | ... | 4.0 | 2.666667 | 3.0 |
| Uta | 259.409949 | 25.940995 | 27.942916 | ... | 1.0 | 2.333333 | 2.0 |
| Phi | 258.964818 | 25.896482 | 26.366778 | ... | 2.0 | 3.333333 | 4.0 |
| Dal | 253.485642 | 25.348564 | 24.036728 | ... | 9.0 | 6.333333 | 5.5 |
| Pho | 249.666184 | 24.966618 | 23.998764 | ... | 10.0 | 7.333333 | 7.0 |
| Lac | 244.126417 | 24.412642 | 25.640585 | ... | 5.0 | 6.333333 | 5.5 |
| San | 241.346869 | 24.134687 | 23.879410 | ... | 12.0 | 9.333333 | 9.0 |
| Lal | 239.391646 | 23.939165 | 23.122741 | ... | 15.0 | 11.666667 | 11.0 |
| Tor | 238.289495 | 23.828950 | 23.716482 | ... | 13.0 | 12.333333 | 12.0 |
| Den | 234.714474 | 23.471447 | 25.222568 | ... | 6.0 | 10.666667 | 10.0 |
| Bro | 234.262030 | 23.426203 | 22.985632 | ... | 16.0 | 14.666667 | 14.5 |
| Hou | 234.009918 | 23.400992 | 23.554202 | ... | 14.0 | 14.666667 | 14.5 |
| Atl | 232.675142 | 23.267514 | 21.986294 | ... | 22.0 | 18.000000 | 18.0 |
| Was | 232.495123 | 23.249512 | 22.612380 | ... | 18.0 | 17.333333 | 17.0 |
| Cle | 232.335051 | 23.233505 | 22.336232 | ... | 20.0 | 18.666667 | 19.0 |
| Nor | 226.288974 | 22.628897 | 22.451270 | ... | 19.0 | 19.000000 | 20.0 |
| Sac | 223.632018 | 22.363202 | 24.314451 | ... | 7.0 | 15.666667 | 16.0 |
| Det | 219.345367 | 21.934537 | 21.964568 | ... | 23.0 | 21.666667 | 22.0 |
| Chi | 217.228818 | 21.722882 | 20.284387 | ... | 26.0 | 23.333333 | 24.5 |
| Cha | 216.186444 | 21.618644 | 21.930980 | ... | 24.0 | 23.333333 | 24.5 |
| Mem | 216.089793 | 21.608979 | 22.231276 | ... | 21.0 | 23.000000 | 23.0 |
| Gol | 215.038799 | 21.503880 | 21.665327 | ... | 25.0 | 25.000000 | 26.0 |
| Ind | 206.811096 | 20.681110 | 23.880372 | ... | 11.0 | 21.000000 | 21.0 |
| Orl | 200.533070 | 20.053307 | 20.067475 | ... | 27.0 | 27.000000 | 27.0 |
| Min | 195.100075 | 19.510008 | 19.455738 | ... | 29.0 | 28.333333 | 28.0 |
| Okc | 190.905106 | 19.090511 | 18.965190 | ... | 30.0 | 29.333333 | 29.5 |

From this output, we can see some competitive teams are not in a very decent standing, such as the Los Angeles Lakers (Lal) being ranked 10th. This is because we sort teams in total score and some teams have more players than others so they have higher team scores. To avoid this issue, we only count highest ten player scores when we rank our team and new output looks like this:

| TEAM | Sum | Average | Median | ... | Rank_Med | Rank_Avg_All | Rank_Overall |
|---|---|---|---|---|---|---|---|
| | | | | ... | | | |
| Uta | 254.931052 | 25.493105 | 25.262659 | ... | 1.0 | 1.000000 | 1 |
| Mil | 253.874802 | 25.387480 | 24.246399 | ... | 6.0 | 3.333333 | 2 |
| Bro | 246.530679 | 24.653068 | 25.140660 | ... | 2.0 | 4.333333 | 3 |
| Lac | 247.814742 | 24.781474 | 24.633438 | ... | 5.0 | 4.333333 | 3 |
| Lal | 248.005000 | 24.800500 | 23.868563 | ... | 9.0 | 5.000000 | 5 |
| Phi | 245.372255 | 24.537226 | 24.150088 | ... | 7.0 | 6.666667 | 6 |
| Bos | 222.394812 | 24.710535 | 25.015613 | ... | 4.0 | 7.666667 | 7 |
| Por | 238.694792 | 23.869479 | 23.235990 | ... | 11.0 | 9.666667 | 8 |
| Mia | 218.021715 | 24.224635 | 24.081768 | ... | 8.0 | 11.333333 | 10 |
| Atl | 231.002872 | 23.100287 | 22.860395 | ... | 12.0 | 11.333333 | 10 |
| Den | 219.233785 | 24.359309 | 23.648617 | ... | 10.0 | 11.333333 | 10 |
| Pho | 231.715515 | 23.171551 | 22.093558 | ... | 18.0 | 12.666667 | 12 |
| Dal | 230.838624 | 23.083862 | 22.369660 | ... | 16.0 | 13.333333 | 13 |
| Ind | 229.763492 | 22.976349 | 22.382763 | ... | 15.0 | 13.666667 | 14 |
| Nor | 168.380833 | 24.054405 | 25.062777 | ... | 3.0 | 13.666667 | 14 |
| Tor | 228.442546 | 22.844255 | 22.642581 | ... | 14.0 | 14.000000 | 16 |
| San | 222.766613 | 22.276661 | 22.852285 | ... | 13.0 | 14.666667 | 17 |
| Sac | 221.279857 | 22.127986 | 22.125937 | ... | 17.0 | 17.000000 | 18 |
| Was | 214.751734 | 21.475173 | 21.068305 | ... | 22.0 | 20.333333 | 19 |
| Cle | 212.922509 | 21.292251 | 21.292479 | ... | 20.0 | 21.000000 | 20 |

**X. Collaboration:**

Online Sources:

A Youtube tutorial about how to scrape data from nba_api:
https://www.youtube.com/watch?v=NCyPY-jfb3I&t=685s

Table of contents of nba_api:
https://github.com/swar/nba_api/blob/master/docs/table_of_contents.md

Player Rankings from 538:
https://projects.fivethirtyeight.com/nba-player-ratings/

Team Rankings from 538:
https://projects.fivethirtyeight.com/2021-nba-predictions/

Debugging strategies from stackoverflow

Syntaxes about pandas function on GeeksforGeeks

Online markdown editor:
https://jbt.github.io/markdown-editor/

Lesson 11 and 12 for ML

Lesson 14 for os module

People:

No one besides course staff and group mates helped us.