NBA Season Prediction

By Jacob Kreinik
SI 330 Final Project

Introduction / Motivation:

I have always been a fan of athletics and sports. whether it is watching them on TV, or playing them competitively. At Michigan, I joined a club called Michigan Sports analytics society which has the goal of looking at different ways in which sports data can be leveraged to draw new conclusions about the game. My goal in this project was to pursue something similar. Specifically, I wanted to see what metrics were influential in terms of predicting a team's success (games won in the regular season) as well as what metrics determined how teams did against the Vegas point spread.

My project looked at data from the 2016-2017 NBA season (most recent completed season). I looked at metrics such as Margin of victory, win percentage, playoff vs not playoff teams, pre-season power rank, cover % (against the spread), wins against the spread, turnovers per game, shooting percentage, and rebounds per game. Many of these metrics did not come directly from the data set and had to be manipulated in some form to get the desired output.

While I did not run regression analysis on all of these variables, I have all of them mapped to a team in my final excel output. However, since these are aligned as rows of team records, I could conduct further analysis if I desired.

Data Sources:

Source 1:
Format: CSV
Size: 8kb (182 records – 30 NBA teams across 7 csv files (2 of the files only included playoff teams)
Process Method: DictReader
For my first data source, I used CSV's that I acquired from https://www.teamrankings.com/nba/. The information was in table format on the website, but the data was not able to be downloaded directly. To get around this problem I copied and pasted the tables into excel, cleaned them up slightly, in terms of formatting, and saved them as csv files. From this source, I used a total of 7 csv files that outlined various team metrics. All the files contained data that was based on teams' performance across the 2016-17 NBA season. The files were named: ATS-16-17-records (metrics: teams record against the spread, Cover% (% of time spread was covered), MOV (margin of victory), ATS +/- (average that a team covers the spread by), Playoff-ATS-16-17-

records (same as first file but only with playoff data), win-loss-16-17 (metrics: actual record for all games including playoffs, win%, and MOV), Playoff-winloss-16-17 (same thing but with playoff data), Rebounds-per-game-16-17, Shooting-percentage16-17, and Turnovers-per-game-16-17 (all of which contain rows of team with their respective metrics as an average for the whole season.

Data Source 2:
Format: Webpage
Size: N/A 30 NBA teams – 30 records
Process method: Beautiful Soup

For my second data source, I used beautiful soup to extract team name and power rankings from the following web page: https://www.si.com/nba/2016/10/24/nba-power-rankings-preseason-warriors-cavaliers. The page was formatted in a way that power ranking was next to team name within the same tag ex: 1. Golden State Warriors. These were the only two metrics I needed from this data source. After extracting I got this data into a list of strings and needed further manipulation to separate the two pieces of data.

Data Manipulation Methods:

Step 1 (writing functions to handle / organize the data in csv files):
After creating the Csv files by copying and pasting the tables, it was time to organize the data to be written into a master csv. To do this I created a function called map_team_to_record that took in a filename as a parameter. The function would read in the file using DictReader and create a dictionary with team name as they key and a three-element tuple containing win loss record, win% and MOV. A similar function, called map_team_to_ATS_record, was built to put the against the spread (ATS) data into a dictionary of similar format. A different function was needed due to the different headers in the file.

Step 2 (Further cleaning the ATS and regular season data)
I had dictionaries that mapped team name to different metrics, however this data was not sufficient for processing. I wanted to get wins losses and ties as a separate data point rather than just having overall record (83-16-0). I created a function called clean_team_record_dict that took in the previously created dictionary. This function split records by a "-" and extracted wins losses and ties from the list. Since the data set had data from the whole season (including playoffs) different teams played a different amount of games whether they were a playoff team or non-playoff team. This made the data set inconsistent. Note that there was no data set containing regular season data. I made a new metric called total games played that added up all wins, losses and ties to be used for further manipulation later. I put all my new metrics into a dictionary of similar format (team: (team metric 1, team metric 2….)).

Step 3 (Organizing data from the team statistics data sets ie rebounds per game):
In this step, I made a similar function, map_team_to_stats, for the three data sets containing team on court stats. (rebounds per game, turnovers per game, shooting %). The function was different due to different headers and mapped team name to the specific stat in a dictionary.

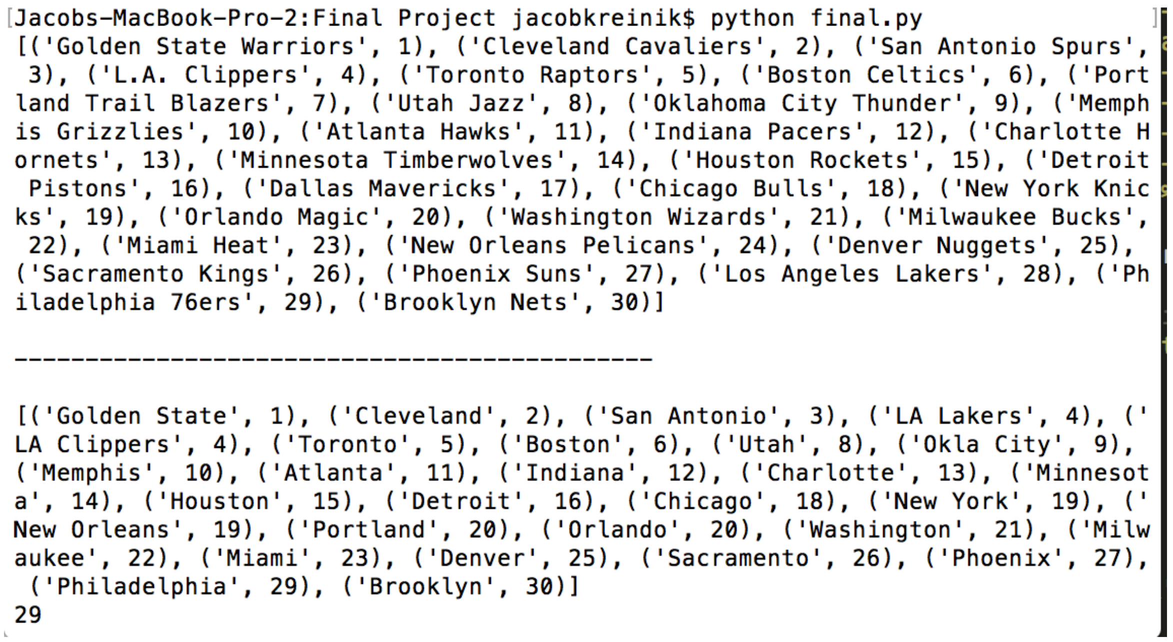
Note: all the dictionaries created in steps 1-3 have same format in the sense that team names are mapped to different metrics / stats.

Step 4 (Getting playoff teams from non-playoff teams):
I created two functions, get_playoff teams, and get_non_playoff_teams. The previous metric, total games played that I manipulated from the other data set was the determining factor. If a team played over 82 game (# of games in the regular season) they were a playoff team. The two functions retuned lists of the teams that fit the parameters of either playoff team or not.

Step 5 (Creating data structures with data from source 1):
In my main function, using all the previous stated functions, I created dictionaries from all the different csv's. After doing this, I knew that I had a multitude of dictionaries with team mapped to many different metrics. I knew at this point I could write this data to a csv. I waited to do this until I structured my data from my second source.

Step 6 (Gathering data from webpage using beautiful soup):
Using requests and the .content method, I created a beautiful soup object for the webpage: https://www.si.com/nba/2016/10/24/nba-power-rankings-preseason-warriors-cavaliers. After inspecting the page, I relieved that all the information I needed was within one specific tag. After iterating through the list of tags and putting the content into a list, I realized that there were a few pieces of random information that fell into the list. To get rid of this I used a regular expression in an if statement so only proper content was added. I now needed to separate my two pieces of desired information, Team and power rank. I iterated through the content, and split each element using a regular expression. I could not split by a "." Because of the edge case "L.A. Clippers." After doing this I now had a dictionary mapping team name to power rank.

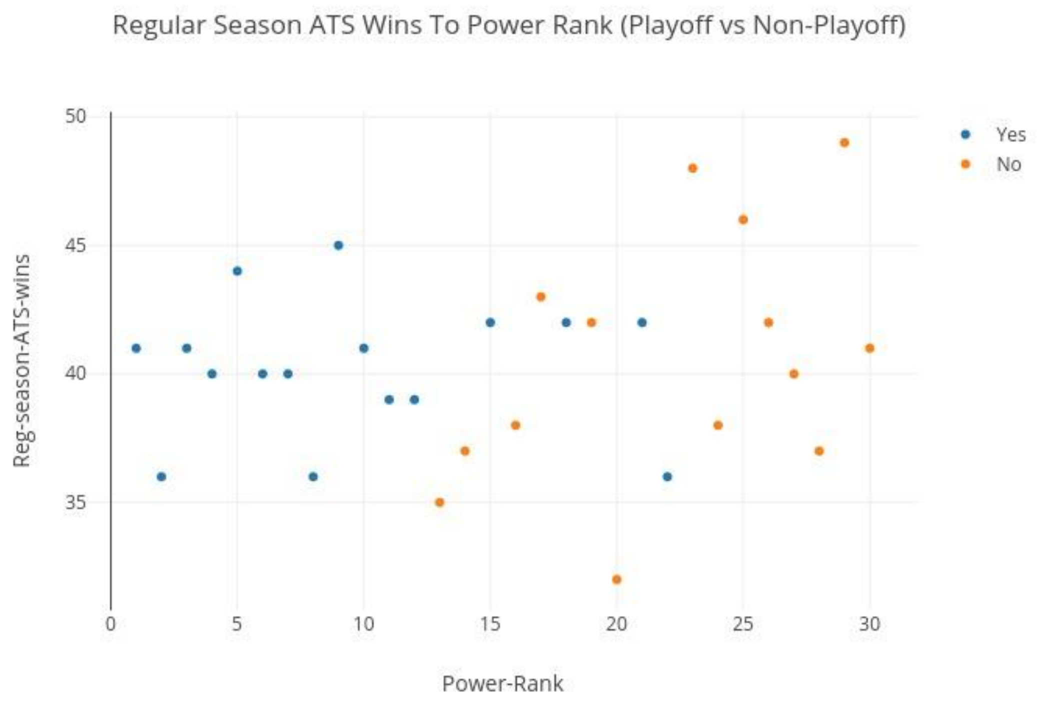
Step 7 (Mapping power rank team name to the team names in the Csv's)
Since I got the power rankings from a different source, the team names were slightly different. Ex: Los Angeles Lakers vs LA Lakers, New York Knicks vs New York etc. I needed these to have the same name to combine them in a csv. I build a function called string_difference which took two dictionaries as input. One was the team to power rank dictionary, and the other was one of my earlier created dictionaries (did not matter because all have the same team name.) I imported a module called difflib which has methods that compare the similarity between strings. I looped through the team name of both dictionaries and essentially compared every team name within the two. If the two strings met a certain threshold, the team name would be changed from the power rank data set and the mapped power rank would transfer over. After much testing and changing the threshold a decent amount, I was always off by 3-4 edge cases, yet all the other teams mapped correctly. If the threshold was too high, certain teams would never get mapped and the resulting dictionary would contain less than 30 teams. If the threshold was too low, teams would get mapped to a different team's value. My results are depicted in the picture below (old dictionary on top, resulting on bottom, notice the small inconsistencies in edge cases). Since this did not work, I had to resort to manually changing the data in lines 146-209.

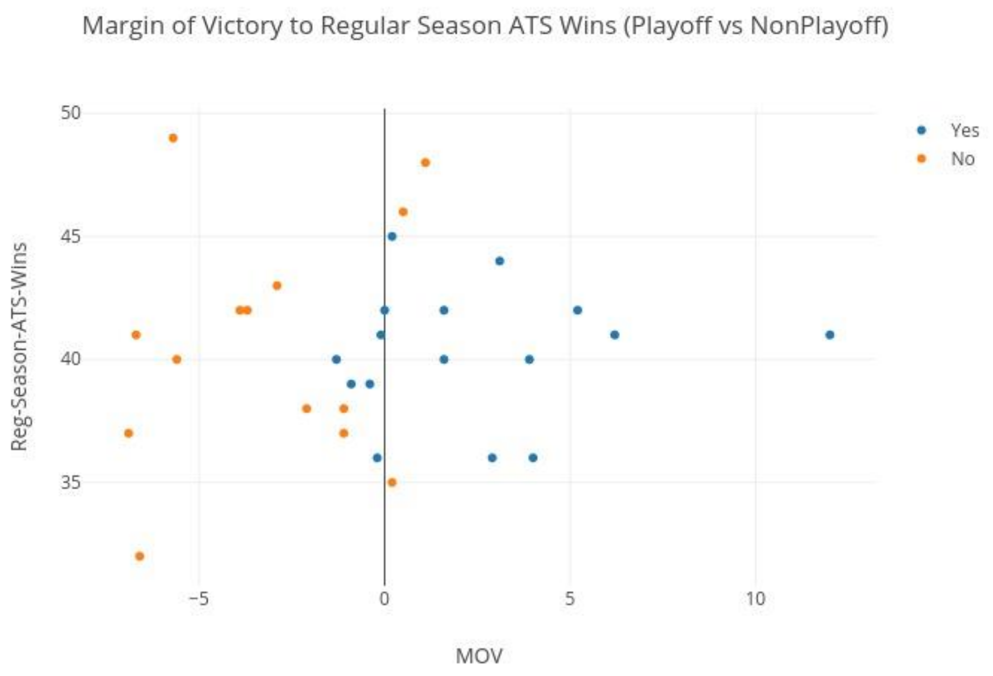


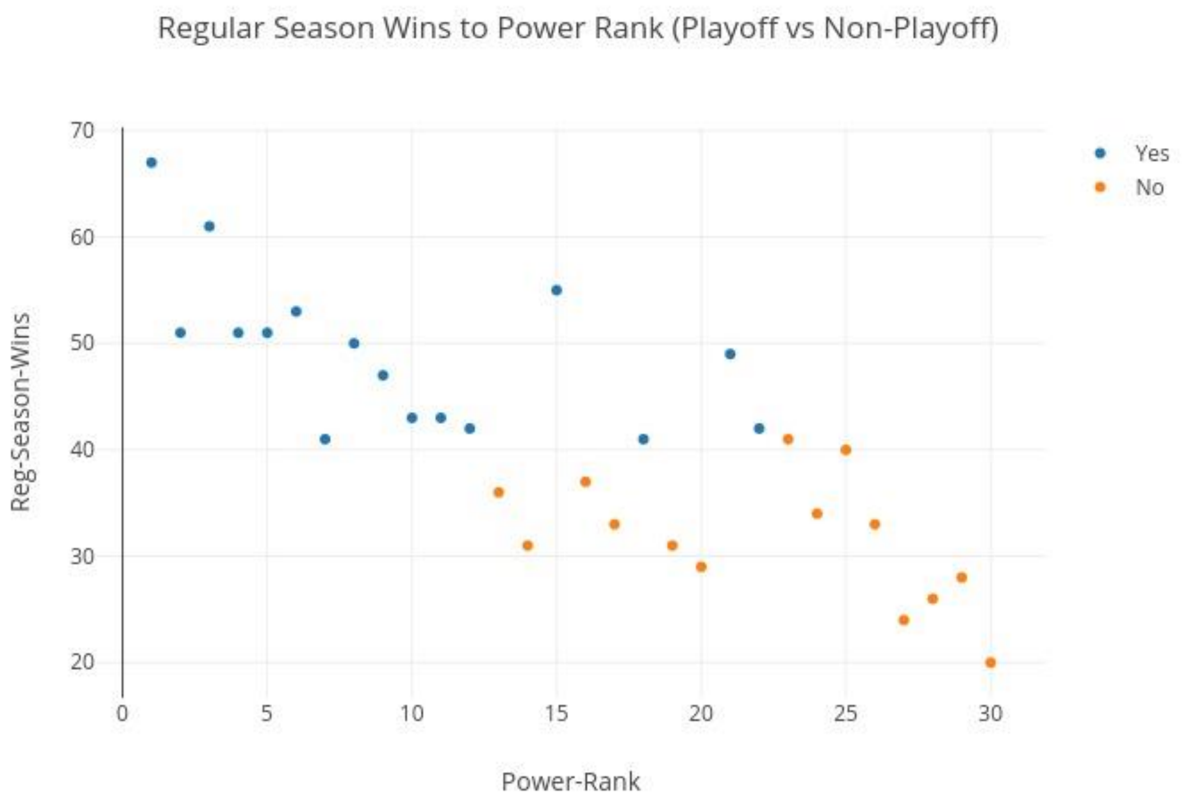
```
Jacobs-MacBook-Pro-2:Final Project jacobkreinik$ python final.py
[('Golden State Warriors', 1), ('Cleveland Cavaliers', 2), ('San Antonio Spurs',
 3), ('L.A. Clippers', 4), ('Toronto Raptors', 5), ('Boston Celtics', 6), ('Port
land Trail Blazers', 7), ('Utah Jazz', 8), ('Oklahoma City Thunder', 9), ('Memph
is Grizzlies', 10), ('Atlanta Hawks', 11), ('Indiana Pacers', 12), ('Charlotte H
ornets', 13), ('Minnesota Timberwolves', 14), ('Houston Rockets', 15), ('Detroit
 Pistons', 16), ('Dallas Mavericks', 17), ('Chicago Bulls', 18), ('New York Knic
ks', 19), ('Orlando Magic', 20), ('Washington Wizards', 21), ('Milwaukee Bucks',
 22), ('Miami Heat', 23), ('New Orleans Pelicans', 24), ('Denver Nuggets', 25),
('Sacramento Kings', 26), ('Phoenix Suns', 27), ('Los Angeles Lakers', 28), ('Ph
iladelphia 76ers', 29), ('Brooklyn Nets', 30)]

---------------------------------------------

[('Golden State', 1), ('Cleveland', 2), ('San Antonio', 3), ('LA Lakers', 4), ('
LA Clippers', 4), ('Toronto', 5), ('Boston', 6), ('Utah', 8), ('Okla City', 9),
('Memphis', 10), ('Atlanta', 11), ('Indiana', 12), ('Charlotte', 13), ('Minnesot
a', 14), ('Houston', 15), ('Detroit', 16), ('Chicago', 18), ('New York', 19), ('
New Orleans', 19), ('Portland', 20), ('Orlando', 20), ('Washington', 21), ('Milw
aukee', 22), ('Miami', 23), ('Denver', 25), ('Sacramento', 26), ('Phoenix', 27),
 ('Philadelphia', 29), ('Brooklyn', 30)]
29
```

Step 8 (Writing all data to master csv) (lines 230-277):
Using DictReader and writer, I could map all the team metrics to a csv. This is the point where having the same team name as the key to every dictionary was important. Reading through one of the csv files, I used the team header to insert into the various dictionary to produce an output. The first part was relatively self-explanatory, going into the different dictionaries and extracting the metric and using the proper position in the tuple. Then to normalize the data set for all teams, I had to subtract out the values from the playoff data set to get the set from the regular season data. I used an if statement to see if a team was in the playoff list. If they were, I would subtract the playoff data set metric from the all games data set. A list of all my metrics include: Team, Power-Rank, Total-Record, Total-Win%, MOV, Total-Wins, Total-losses, Total-ATS-record, Total-Cover%, Total-ATS-wins, Total-ATS-losses, Total-ATS-ties, Reg-season-record, Reg-season-wins, Reg-season-losses, Reg-season-win%, Reg-season-ATS-wins, Reg-season-ATS-losses, Reg-season-ATS-ties, Turnovers-per-game, Rebounds-per-game, Shooting-percentage, playoff-team. You can see these metrics in the csv file final-team-output.csv.

Analysis & Visualizations:







I used Plotly to create the three linear regression models. There is no code representing this in my python script, however I used the final output csv and imported it to plotly to create the visualizations. The questions I aimed to answer were to see what metrics were good predictors of season success ie (total wins) as well as predicting success against the spread. I correlated power rank with both regular season wins and wins ATS wins. Power rank does a decent job of predicting actual wins. Generally, as power rank goes up, number of wins goes down. This means that the creators of preseason power rank did a good job at predicting a team's overall success before the season started. It is also clearly indicated that playoff teams have more wins than non-playoff teams. An interesting caveat are the teams that were power ranked slightly higher than others, but did not make the playoffs.

Using simple linear regression to predict wins against the spread was not very effective. Vegas bookmakers have the goal of creating 50-50 odds for each game, so the expected value of a team's ATS record is 41-41 over an 82-game season. Vegas bookmakers use a variety of different metrics when making spreads, ie home team, amount of rest, previous games, recent player performance, past meeting between teams etc. It makes sense why using simple linear regression did not yield significant results. MOV and power rank were not good predictors of ATS success. Additionally, there did not seem to be a difference between the success of playoff teams vs. non-playoff teams. To see more effective results, a multiple linear regression would have to be run. It would be possible to do this with some of the different metrics in my data set ie turnovers and or shooting percentage. These are not directly correlated to wins so it would be interesting to see how these metrics predicted ATS. If I continued testing predictors, I would gather more data from teamrankings.

Sources:

Data set 1:
https://www.teamrankings.com/nba/

Data set 2:
https://www.si.com/nba/2016/10/24/nba-power-rankings-preseason-warriors-cavaliers

Difflibmodule:
https://stackoverflow.com/questions/1471153/string-similarity-metrics-in-python (line 76)