Alvaro Hu

7/30/2018

Capstone 1: Data Wrangling Steps Part 1

To start my project, I went online to ‘Basketball-reference.com’ and imported the 2017-2018 NBA statistics and the 2018-2019 salaries for each qualifying player. The reason why the years are not the same is because the player is given the contract based on his previous season’s performance, not usually midseason. And even if he was given a contract midseason, we would still see this reflected in the following year’s salary report. After I downloaded the two, I imported them both as pandas dataframes. For some reason, there was a string attached to the player’s name so that the ‘Name’ contained something like “First Last/firstlast”. I dealt with this by splitting the ‘Name’ column at the ‘/’ character and dropping the extra columns. I then concatenated the two data frames, so that if a row from the salaries data frame had the same index as a row from the stats dataframe, it would automatically append the columns together horizontally. I used the merge() method using the ‘Name’ column as the column the two merged on.

Before merging the data frames, I had to get rid of some repeat values in the statistics table, for whenever a player is traded or switches teams for whatever reason, they split his stats up between the two teams and add a ‘TOT’ row that comes first, but he still only gets one row in the salaries table. In order to do this, I used drop_duplicates(), and used the name column as its key, and chose that it keep the first of the duplicate values, which is the ‘TOT’ row.

After concatenating the data frames, there were many NaN values that I needed to get rid of. Some of these NaN numbers make sense to keep, like NaN for ‘3P%’ if a player never took a 3-pointer, but others did not make sense to keep in the data frame. The two columns that I decided make the row unusable were ‘2018-19 salary’ and ‘G’, meaning “games played”. The

former I dropped because, if the player does not have a salary this upcoming year, then he must be either retired, or not have a contract, which means I cannot relate how his play reflects his contract. The latter I dropped because of the same logic, but vice versa; I cannot relate a person's play/statistics to their salary if they did not play any games last season. One thing that I have thought about here though is that I do not allow someone who has been injured for the entire year to make it into the data frame. A potential way I can get around this is to import the 2016-2017 statistics as well, and use an if-else statement saying that if the 'G' column for a row is NaN for 2018 but not in 2017, then use the 2017 table's value instead.

I ran into a problem, sorting by the '2018-19' column, where python was automatically only sorting by the first digit of the number, because it was initially an object type. Meaning, since 9 is greater than 1, then 9,000,000 would also be greater than 10,000,000, which we all know is not true in the slightest. In order to work around this, I needed to cast the entire '2018-19' column as an integer type and then call the sort_values method. I originally tried converting all of the salary columns to float types, since floats can deal with NaN, but python sorted floats the same way it sorted objects. So I will either convert the rest of the columns as floats in a later call or just drop the other salary columns altogether, as I currently can't think of a reason to use them.

As of right now, I currently only have the stats for last year and the salary for the upcoming year, but I am planning on adding previous years to the data and see how it has changed. So as of right now, this is only part 1 of my data wrangling steps.