## PROJECT 4 - Justin Clifton, Julia Januchowski, Zac Laney.

### Introduction

The most prized individual accolade for a player in the NBA is the Most Valuable Player award. This award signifies that coaches, the media, and former players thought that the winner of this award was the most important player in the entire league. There have been 65 winners (with some overlap). These people played a variety of positions. They were Point Guards, Shooting Guards, Small Forwards, Power Forwards, or Centers. They all had different statistics, and come from different eras. What then, do they have in common? What does it mean for a player to be the MVP? Can we predict who will have an MVP season? What does a players statistics indicate about their chances of winning the award? Given the right dataset, these are all questions that could potentially be answered. The right dataset must contain both player statistics and a categorical variable describing whether the player won the MVP that year or not. Unfortunately, obtaining the data to answer these sorts of questions is not always as straight forward as downloading a dataset from Kaggle. In this case, there exists a dataset containing player statistics for a season, but it lacks which player won the MVP. Can we solve this problem? Can we pull data from the internet, combine it with data on player statistics, and then use that data to make a simple model? These are the questions that we seek to answer. In the process, we will use libraries like BeautifulSoup to parse basketball reference, and various pandas operations to combine the datasets.

There are three key skills we would like to demonstrate. The first being scraping data from the internet. Scraping data from the internet is a useful skill because there is not always a dataset available for any problem that needs to be solved. We will accomplish this using the python library BeautifulSoup. Next, we will attempt to combined our scraped dataset with data that already exists. Finally, we will display the skill of fitting a simple model to try and predict who will be an MVP.

### The Parsing Problem

Basketball-reference is a website that contains basketball statistics. However, we must somehow access this data using code. We will begin by using the python library BeautifulSoup to parse the data contained on this website, and extract the information that we would like. Beautiful Soup is a Python package for parsing HTML and XML documents. The first step in this process is to create a BeautifulSoup object that will find all the data contained in tables.

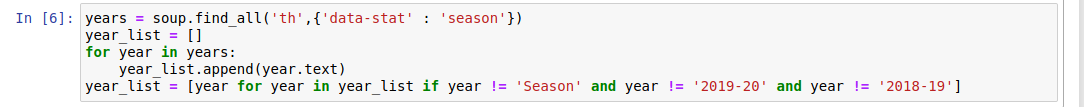
The code above produces a list of players contained in the MVP table. However, the format is not exactly how we want it. The cell block below shows the format that is returned.

 We will have to extract the players name using the .text method. Next, the names of the players who are contained in this table will be stored in a list. Later, this list will be used to create a new dataset of MVP winners. It is worth noting that Giannis Antetokounmpo (who won in both 2018 and 2019) will be removed from this list, as the dataset containing season statistics only goes up to 2017. Once we use the .text method, it can be seen that the names are in the format we desired.

'James Harden'

An example of the names after they have been parsed.

After the players names have been collected, we will also extract the year that they won the award. This process is similar to the one used above.

 Since the format of the years were extracted as a string (e.g. '2019-20') and the years in the season statistics dataset are formatted as floats (e.g. 2019.0), we had to extract the start year from the string. This was done using apply and a lambda function that slices the string to get the first 4 digits then casts that to a float.



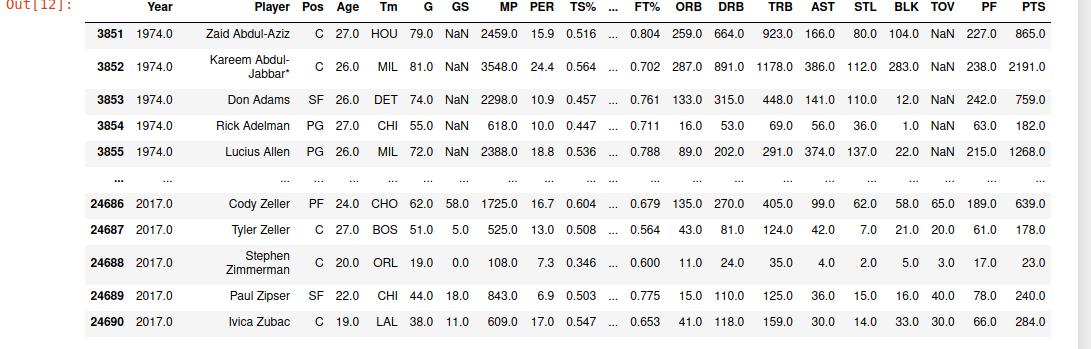
Our MVP dataset in its current form is shown below. It can be seen that we have successfully extracted the names and years corresponding to the MVP.

## 

This dataset will be used in combination with the data on player statistics for a season. To accomplish this, we must find a way to merge these two dataframes.

### The **Season Stat** Dataset

As mentioned before, there is some data that already exists on this topic that can be obtained from Kaggle. We will now import a dataset containing player stats from 1950-2017. However, since the NBA did not keep track of blocks and steals until 1973, datapoints prior to 1973 will be removed.



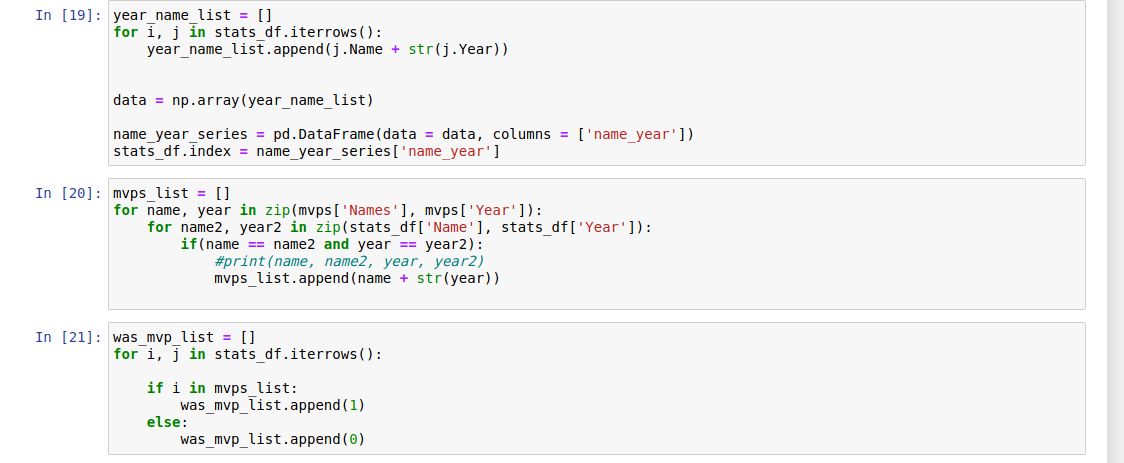
The dataset after omitting data prior to 1973.

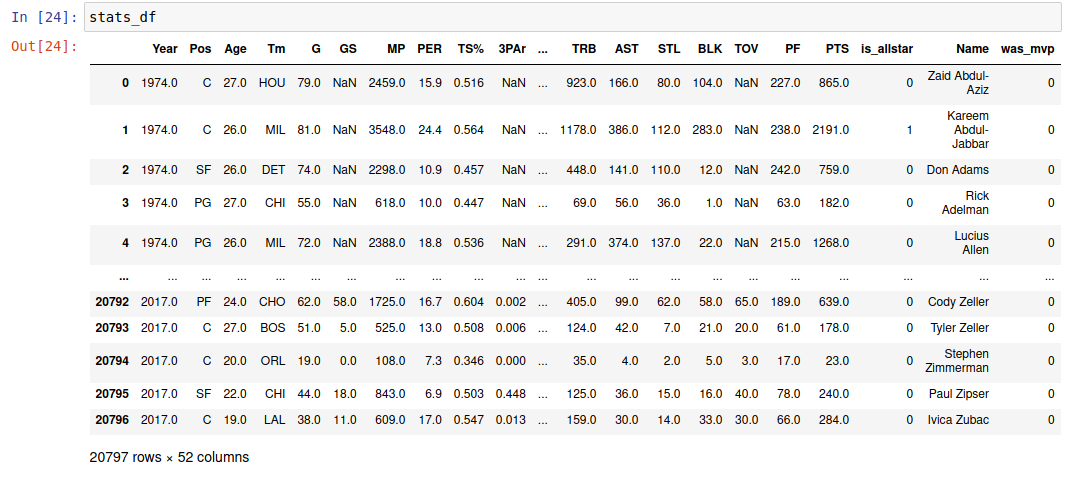
During the process of exploring this data, our team noticed that some names had a '\*' contained in their string. After some research, we learned that this star indicates that a player was an all-star that season. The all-star award indicates that a player was voted by fans or coaches to play in the all-star game. Only 12 different players are selected for the all-star game. On account of the exclusivity, the players who make this team are typically the best in the league. We will utilize this information to engineer a new feature called 'is\_allstar'. This feature will be 1 if the player was an all-star that season, 0 otherwise. This could be quite a useful feature, since a MVP is essentially guaranteed to have also been an all-star player. However, this feature also presents the potential for a new question. One might consider predicting whether a player will be an all-star or not in a given season. After creating the new column, we will remove the \* from the names to clean the data.



### Combining The Data

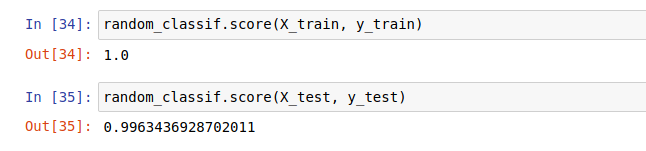
At this point, we are now going to begin the process of combining our two data-sets. This proved to be no small task. Initially, we attempted to just find any player who was in both dataframes. If the player was contained in both dataframes, they would get a 1 in a new list to indicate they were the MVP. However, this idea was highly flawed. The new datframe that was produced implied that Steve Nash was the MVP in 1996 when he averaged 3 points per game. After much pondering, we determined that the best solution would be to compare the name of the player and the year of the season as contained in the MVP and season\_stat dataframes. If these two features are the same then the corresponding player in the season\_stat dataframe must have won the MVP award in that season. Unfortunately, this idea also proved to have flaws. When a list was created using this method, the list ended up having too many entries due to the used of a nest for-loop. As a work around, manipulation of the dataset was required. First, the index column of the dataset was changed. Instead of being indexed by numbers, we chose to temporarily index the dataset by the players name concatenated with the year. For example, James Harden's 2017 season would temporarily be indexed by ‘James Harden2017.0’. Manipulating the dataset in this way enabled us to put a 1 in locations where the name and year matched and 0 in locations where they did not. Finally, created the list needed to create a new column. We then used that list to create a new column to the dataset denoting if the player was an MVP or not. Below is the code that

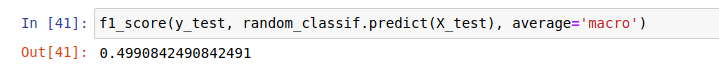
After all that work, let us bask in the glory of our newly formed dataset.



At last, we have the answer to our 'big question'. It is possible for us to scrape data from the internet, and combine it with data that already exists. We will now recap what unfolded. First, data was parsed from basketball-reference to obtain information on which players won the MVP and in what year. Then, a dataset containing season stats was imported from Kaggle. Next, a feature indicating if a player was an allstar that season was engineered based on a '\*' contained in the players name. Lastly, a new dataset was created by combining our MVP dataset and our season stats dataset. This new dataset contains season stats and information about whether a player won the MVP or was an allstar that season.

We will now encode the position variable. This will ensure that we can use this as a feature in our model. We will also be dropping the names and teams features. This serves as the preprocessing for our dataset. Since was\_mvp is either a 0 or a 1, this is a categorical variable. Next we will fit a simple Random Forest Classifier to our data. This model has a fantastic R score on the training data of 1.0. Furthermore, it scored well on the testing data with an R score of .96.

These scores are surprisingly high. The R score is not always the best indicator of how well a model fit the data. For this reason, we will also score it using a f1\_score. An f1\_score tells us how precise our classifier is (how many instances it classifies correctly), as well as how robust it is (if it does not miss a significant number of instances).

This score is much lower and might indicate that this model does not perform as well as the R score might suggest.

### **Conclusion**

In this project, we sought to create our own dataset out of information parsed from basketball-reference. During this process, we used a library called BeautifulSoup to extract what NBA players won the MVP. Once we obtained this, we then manipulated the dataframes in such a way that we could obtain one dataframe with all the information that we wanted. In the end, we fit a random forest classifier which had a strong success rate but a low f\_score.

### **Citations**

<https://stackoverflow.com/questions/45746426/data-scraping-basketball-data-using-beautiful-soup> <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> <https://medium.com/hardwood-convergence/intro-to-virtual-environments-and-scraping-nba-data-with-beautifulsoup-6ce745f8c26e> <https://stackoverflow.com/questions/3559559/how-to-delete-a-character-from-a-string-using-python> <https://www.kite.com/python/answers/how-to-create-pandas-dataframe-from-a-numpy-array-in-python> <https://stackoverflow.com/questions/3559559/how-to-delete-a-character-from-a-string-using-python> <https://thispointer.com/python-how-to-use-if-else-elif-in-lambda-functions/> <https://thispointer.com/pandas-check-if-a-value-exists-in-a-dataframe-using-in-not-in-operator-isin/> <https://www.geeksforgeeks.org/iterating-over-rows-and-columns-in-pandas-dataframe/> <https://www.basketball-reference.com/awards/mvp.html> <https://stats.stackexchange.com/questions/49226/how-to-interpret-f-measure-values>