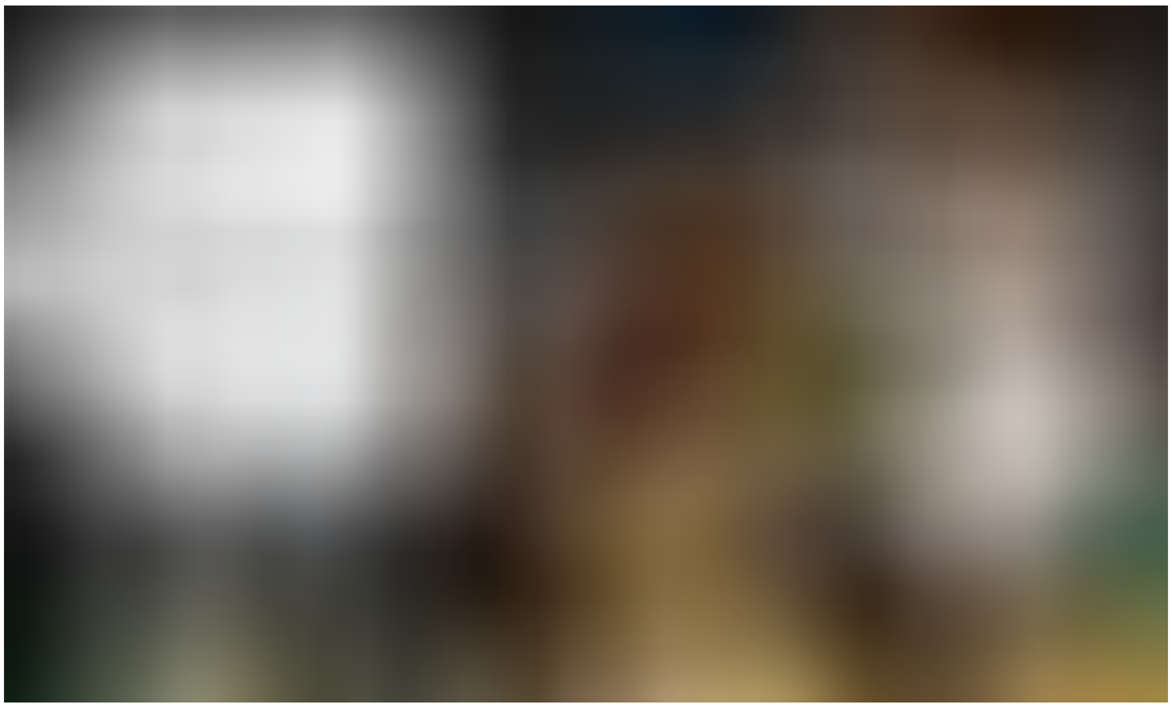
learning technics in Python.
Nov 2 · 5 min read

Sentiment Analysis on NBA top players’ Twitter Account — Part1 Data Collection



source: ftw.usatoday.com and nbcsports.com

Abstract

It is known that the leading players in a team play a vital role in the game-winning. **Hence, we are curious if the players’ emotion before the game will have correlation on their performance or even on the game result.** To measure this effect, we choose Twitter as our tool to measure the players’ sentiment because a majority of NBA players tweet their opinions or mind on twitter.

However, it is time-consuming to evaluate each player’s tweet in the league; hence we will be just focusing on the top 10 players according to ESPN.com. Now someone may start to debate why those are the best players on each team but here we just use one of the examples website instead of any personal preferences.

In the analysis, I will be using Python as the programming language and this topic will involve technics such as **web scraping, clustering, data visualization, and natural language processing**.

Metrics

Since the ultimate goal is to measure the correlation between sentiment and game performance. The sentiment here is by calculating the positive or negative score based on **polarity** and **valence**. We will discuss this later.

As for the performance, we use **Efficient Field Goals(eFG)** in this article due to its good balance between simplicity and accuracy.

eFG formula: ***(FG + 0.5 * 3P) / FGA***

Prerequisites for Part 1

Before we start, these packages are required for this part:

Don't panic! The details of the packages will be explained throughout the article.

```python
import datetime
import json
import numpy as np
import nltk
import pandas as pd
import requests
import re
```

But before things start, remember you should already have a Twitter developer account or you cannot proceed with the process.

Note that the consumer_key, consumer_secret, access_token, and access_secret should be using your own token. Do not share with anyone!!

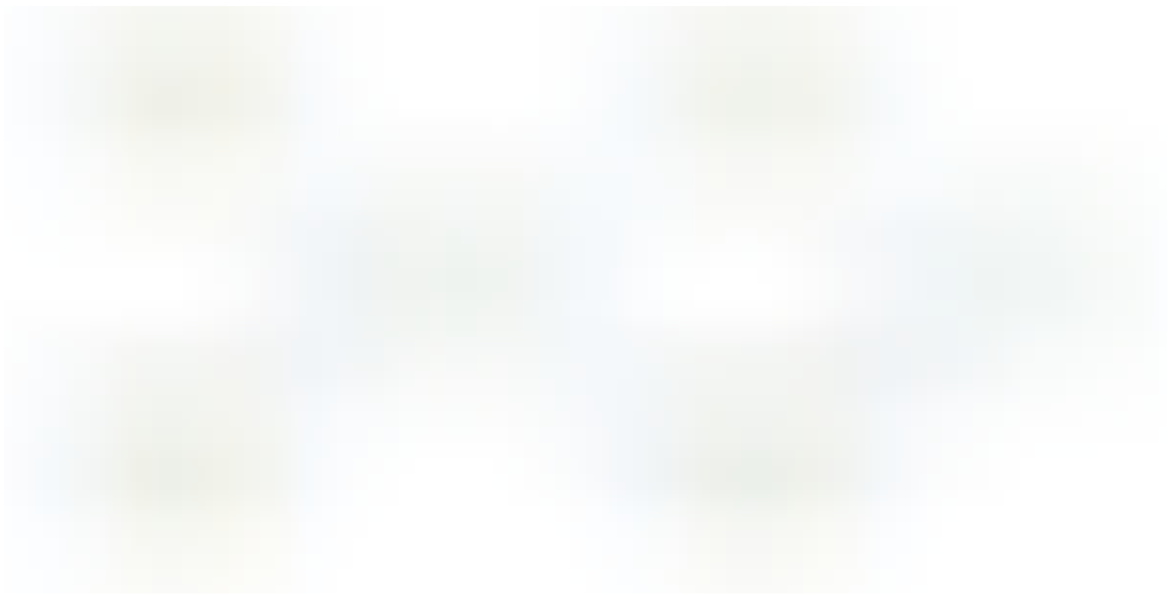
```python
consumer_key = “”
consumer_secret = “”
access_token = “”
access_secret = “”
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)
```

Steps for Data Collection

Knowing that we are looking for the relationships between the players' tweet and performance. There will be four resources in order to complete the data collection before we dive into the analysis.

1. Get the top 10 players' name from ESPN.com
2. Collect a list of NBA players' Twitter account and their name from reference.com, and match the top 10 players name and their Twitter
3. Scrap the matched players' stats on reference.com
4. Grab the players' tweets and their creation time of each tweet.

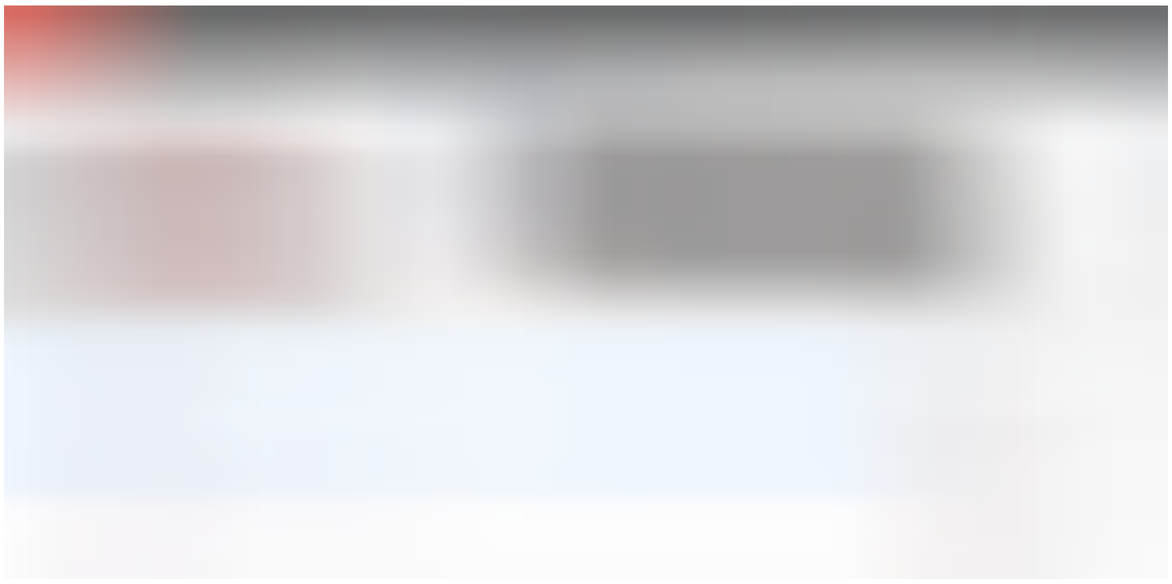
Here is the figure of how these data are connected and match:



The first three parts of the data will be collected by web scraping technics. Here we use Python's ***requests*** and ***BeautifulSoup*** package for the extraction. Requests is a very simple HTTP library for Python which half of the scraping tasks can be done in this function: **requests.get(yout_url_link)**. This function simply just download the HTML content into your computer without any modification.

Then, what we need is to apply BeautifulSoup to parse the data. The function here is simple either. We just need to right click the content you want and click "inspect". Then, the source code of the web page comes out instantly and finds the exact content through the ***tag***. Note the tag is a kind of marked up words that can be detected by the search engine. Usually, the ***tags*** will attach with different kind of words called ***atrribute***. For example, the word "***a***" is the tag, ***href*** is the attribute name, the URL behind is the value of the attribute href. And, of course, ***Joel Embiid*** is the content what we are looking for.

Now, that's get started!



1. Get the top 10 players' name from ESPN.com

```python
# Find the best player on each team according to ESPN.
url_player = "http://www.espn.com/nba/story/_/id/24668
players = requests.get(url_player)
player = BeautifulSoup(players.text, "lxml")
player_list = []
for h2 in player.findAll("h2"):
    for name in h2.findAll({"a":"href"}):
```

2. Collect a list of NBA players' Twitter account and their name

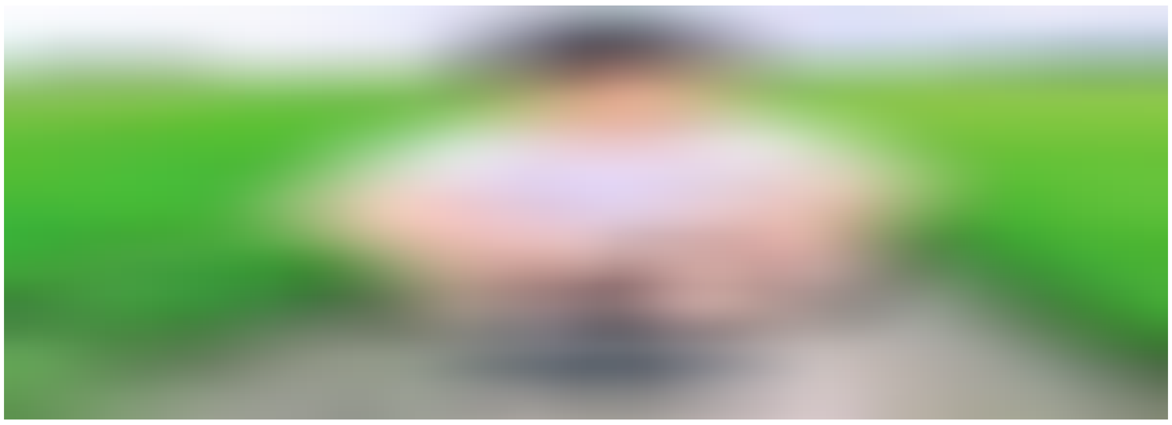
```python
# Extract nba player list and their twitter account f
url_twitterlink = "https://www.basketball-reference.c
tweets = requests.get(url_twitterlink)
tweet = BeautifulSoup(tweets.text, "lxml")
player_list2 = []
tw_list = []
for td in tweet.findAll("td",{"data-stat":"player"}):
```

The above result is stored in a dictionary format like this:

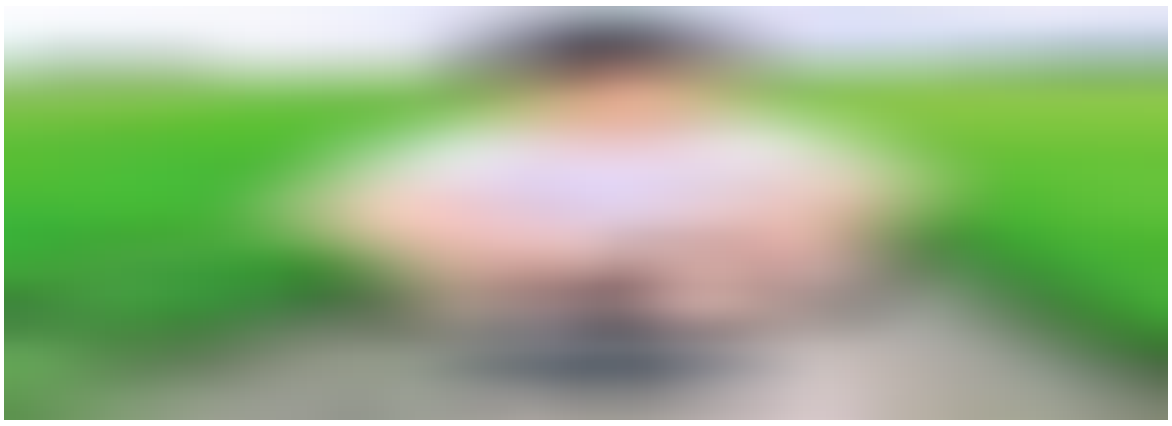
The names are the key of the dictionary and their Twitter accounts are the values:

3. Scrap the matched players' stats on reference.com

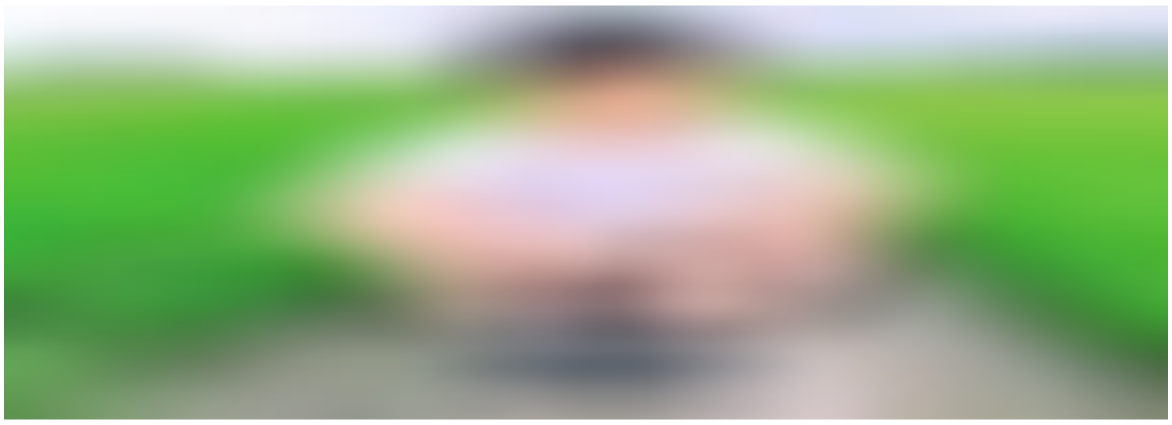
Now, this part is about extracting the stats that players performed during the court. We are taking stats between 2016–2019 season but here we run into a very serious issue. Each URL will be different according to each player and there is no any fixed pattern for the URL. Thus, the only way I come up with right now is to lists all the URL of each player. After that, we time the year according to each player's URL and get the whole URLs we need.



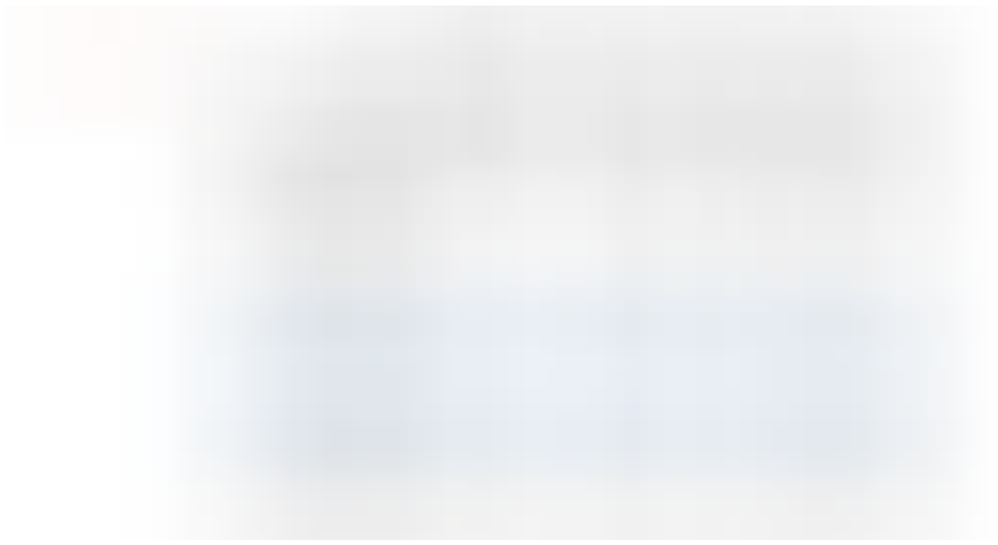
After generates the URLs, the send them into the following function. Here we collect the game time, game result, field goals(FG), field goals attempted(FGA), and 3-point field goals(FG3). Also, the reason why we have a lot of "if…else" statements is that reference.com marks the record with the specific words if the player did not play the game such as "Inactive" or "Did Not Play".



Then, we reshape the list type data into data frames. Here we also convert data types into the correct format and minimize the data size through assigning the “int” type to “int16”. Recall that we are measuring the performance of the players and their tweets. The performance we define is the **Effective Flied Goal (eFG)** and the formula is **(FG + 0.5 * 3P) / FGA**.



The figure shows one of the player’s stats we collected:



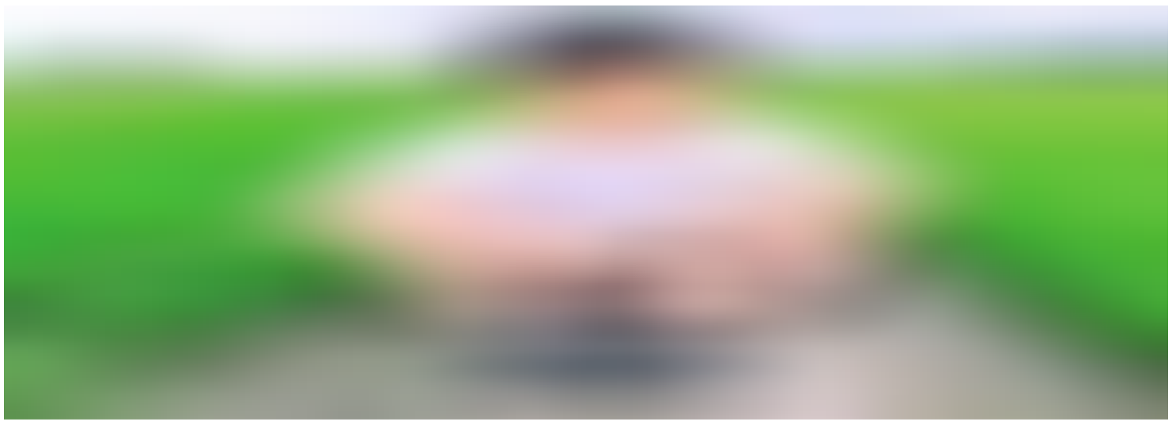
Now, here Twitter comes, we start to grab the tweet data according to the accounts we just got.

4. Grab the players’ tweets and their creation time of each tweet

The function here looks complex, but actually, it is just about getting the tweets, number of followers, number of friends, number of favorites, and combining the stats of their perfamance and the players’ name into a giant data frame.

Note:

Remeber to add **tweet_mode = ‘extended’** for the complete amount of characters or you only get the first **140 characters** on each tweet. Also, only the **tweepy.Cursor( ).items(1000)** api can have access to more than 200 tweets. The **api.user_timeline(id = ID, count = 200)** api has this limitation.



That's all for the data collection and next chapter we are going to have some data manipulation on the tweets.