Final Project – Spotify AP

Kyle Morris

For my final project, I chose to work with a site I use daily – Spotify. They do have a public API I was able to use. I figured that I could tap into their massive song database in order to scrape data. I created an account and got authentication keys – a fairly straightforward process, even easier than getting on to the Twitter API.

Reading their documentation, it turns out there is a library called “Spotipy” that is a wrapper for the Spotify API we could use. I settled on implementing the search function, which returns the top 10 results matching the string regardless of it being album, song, artist, etc. Though this search could in theory be done without authentication, I received “Not authorized” messages every time I attempted to access the API without setting it up.

Once authenticating, we fetched the search data. Printing it shows that the information is provided in the JSON format. My first step was to print out the different lines. I printed out the header names and the first entry so I would have a better idea of what the search gave us. The album field was another nested field. I printed that one out in order to strategize how best to extract it.

Now on to the transformation. My first task was to extract the four fields I found most useful in the albums field – album type, album name, release year, and number of tracks. I cycled through each item and appended each individual record to a series of lists I created and then added them to our dataframe.

My next challenge was to pull the artists. Some songs were by more than one artist – the first song on my sample search, Champions, has about 6 or 7 performers listed. I wanted to make sure our scripts could handle multiple artists in a clean way, so I wrote a function to pull all of the artists and convert them to a single string, Artist A & Artist B & etc. I made sure that it would handle both single artists and multiple artists.

After pulling the artists, we had one last transformation to do. Our original data provided song length in milliseconds, which I’m sure is great for their algorithms but isn’t very human readable. We converted into seconds and renamed the column by dividing by 1000. We ended up with 20 columns, and each search provides the top ten results. While current state does not do anything with the data, future state I would probably implement a SQL database to store the data to look for trends over time.

This process is immensely important as a data scientist. Often times, a model is created and refined but over time new data needs to be inputted. In my current role as a junior analyst, it’s my job to pull data from our database, import it into the model and verify the output. In order to get the data to that point, we’ve had to set up complex SQL commands in order to pull the data from our production databases into a format that our model is expecting. If this step hadn’t been done, as the expression goes it is “garbage in, garbage out”.

In Spotify’s case, I expect there wouldn’t be too much of a shift daily. Music has reported on a weekly basis for decades, so I would expect that would be the minimum length of time between queries. You could track a song’s popularity over time, or how high in the search rankings a song was over time. Search results from Spotify seem fairly standardized, so if we ran other searches we should be able to reuse a lot of the code as our needs change as we finalize our questions.