Kristina Holcombe
Aaron Santilli
SI 206
Prof. Ericson
4/26/2022

Stocks and Streams: Using Billboard Hot 100 and Spotify Streaming Data to Analyze Stock Trends

https://github.com/holcombek/FinalProject206.git

1: Goals For Our Project

Our initial project plan was to see how the number of streams, on Spotify, for the Hot 100 songs on Billboard affected the Spotify Stock over a period of 4 weeks. Our goals were to analyze the total number of streams of the Hot 100 songs (from Billboard) on Spotify and use those numbers to calculate variance and volume of the Spotify stock. We planned to use those numbers to visualize the changes in stock and analyze the impacts of the streaming numbers.

2: Goals That Were Achieved

We ended up taking a different approach at the Spotify streams, as described in our problems section below, and were able to cross reference data by week between the Billboard Hot 100 songs and the Spotify Top 200 streamed songs. We were able to connect the data gathered from Spotify, including song title, artist, and number of streams, to the Billboard rank of the song.

In addition, we looked at the relationship between total streams for the cross-referenced songs on Spotify and the Spotify stock price. We determined that there are many larger external factors that can affect stock price and most investors will likely look at subscriber numbers or other indicators from the company in order to make investing decisions. The number of streams of the top songs did not seem to have a significant effect on the share price of Spotify as the stock price actually went down over the interval we looked at while total streams of top songs went up.

Finally, we calculated variance and volume of the stock, and created four visualizations related to these, as well as number of streams, which include a linear regression chart. The variance shows any day-to-day jumps in the stock price, which indicates if share price rose or fell that day. The trading volume shows how many shares were traded in any given day, which can be useful to determine if a stock had an unusually high or low day of trading, indicating there may be some external factors affecting the markets interest in the stock. The number of streams gives us insight into how well the very top songs on Spotify are doing, which are the most visible songs on the platform and may attract people to listen to new and/or trending songs on the platform. The linear regression chart shows the relationship between the two variables of stock price and total streams of top songs. As seen in the chart below, there is some evidence to

indicate a bit of an inverse linear relationship as total streams increased while stock price went down over the interval we looked at. Although we only had a few data points to analyze on this chart, the inverse linear relationship likely suggests that the total number of streams of top songs have little impact on investors' views on investing in the company. One would expect a positive linear relationship if this were the case due to the increasing number of streams.

3: Problems We Faced

In planning our data collection for the songs, we discovered that the Spotify API does not have the number of streams of a track available through any function. Our initial plan was to use the SpotiPi, however since there was no public access to that variable, we found a website, Spotify Charts, which has the streaming data for all the charts on Spotify. We then realized that we were unable to scrape data directly from this site, as it required a login before using the page, and we could not figure out how to bypass the sign-in through our code, despite us both having Spotify Accounts. Thankfully, the website had an option to download the site as an HTML file, and we ended up downloading the four weeks we were looking at as four separate HTML files, and used Beautiful Soup to collect the data. In addition to the problems with gathering the song data, we needed to clean the song data so that all the song titles and artists had the same name and we could cross-reference the data using this information. This proved to be tricky, as much of the data differed with even the smallest piece of punctuation.

Another confusion we ran into was when we cross referenced the data between the Billboard songs and the Spotify songs, we had expected to get information about 100 songs, when in reality we only gathered around 35 matching songs for each week. We believe that this may be due to the fact that Spotify top streamed songs were not all as recently released or radio popular like the Billboard songs are.

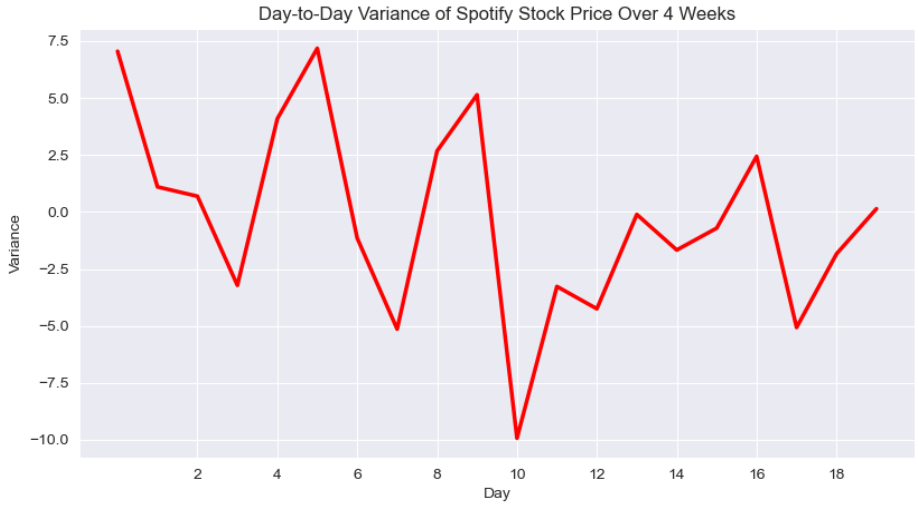
Other problems involved our use of the Yahoo Finance API, which the one we chose originally was behind a paywall, so we had to find a different version that was free to use. Once that was solved, we had to change a lot of code to reflect the new documentation of the API as well as to take advantage of the built-in functions the library had that accompanied the API. Our final problem occurred with using the database to retrieve the streaming data. When retrieving the data to use in one of our visualizations, the data was being returned in tuples, rendering it difficult to process. Eventually, using a Pandas dataframe and proper indexing, this problem was solved and the data was able to be graphed.

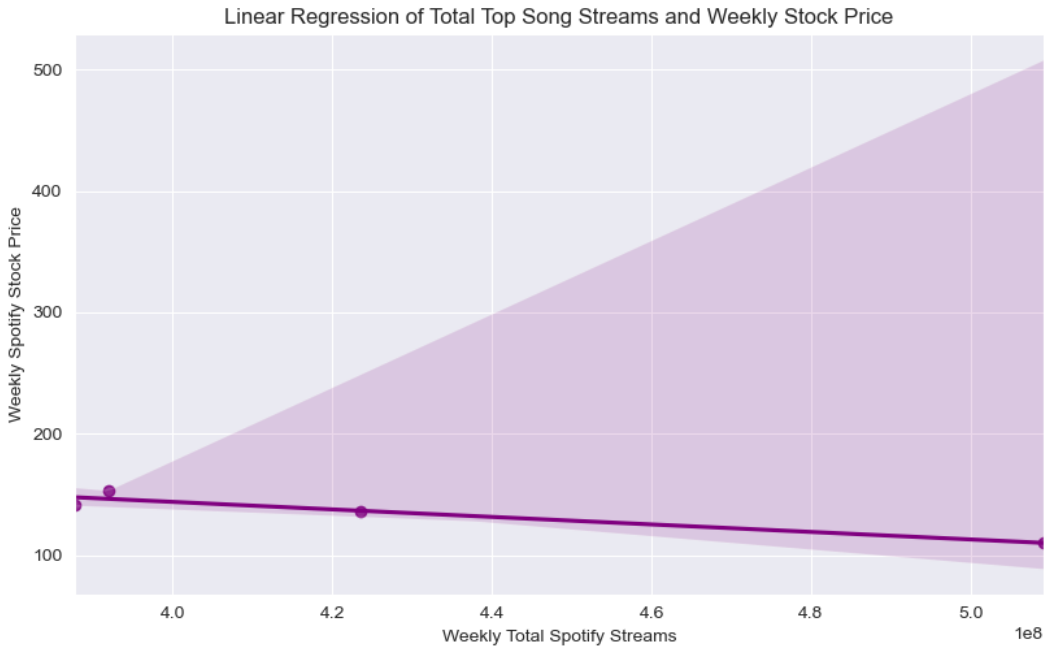
Another interesting issue was figuring out over which interval to analyze the stock price. Although it was fairly easy to retrieve day-to-day stock information, it was much more difficult to get daily streaming data (we were able to retrieve it on a week interval). This presented some accuracy problems when creating our linear regression plot comparing the two variables as we only had four overlapping data points to analyze (we adjusted the finance API to retrieve weekly data for this function in particular).

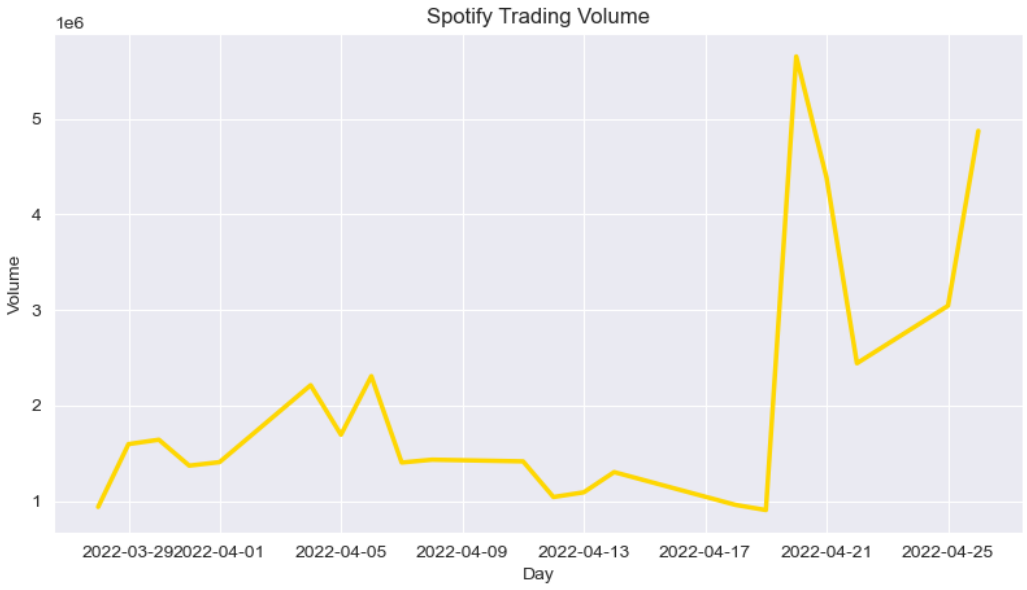
4: File That Contains Calculations from the Data in the Database

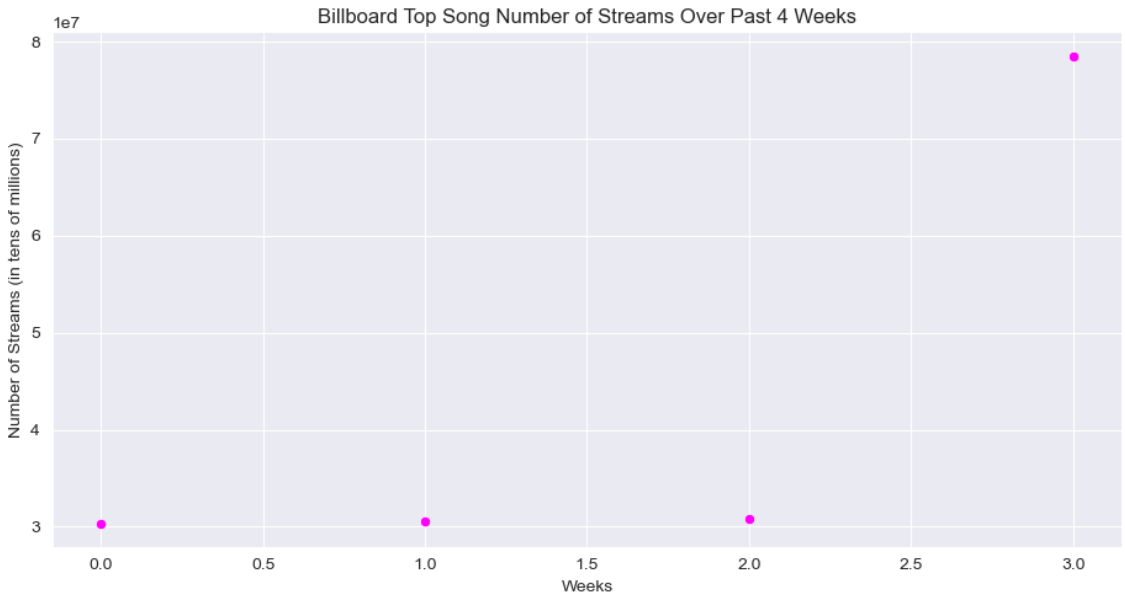
https://github.com/holcombek/FinalProject206/blob/main/calculations.txt

5: Visualizations We Created









6: Instructions For Running The Code

billboard_data.py file:

1. Go to billboard_data.py file
2. Run file four times, waiting 5 seconds in between each run
3. Check database to ensure there are 100 items for each week of Billboard data (a total of 400 items)

spotify_charts.py file:

1. Go to spotify_charts.py file
2. Set start = 0 in main
3. Run file once, wait 5 seconds
4. Change start = 25
5. Run file again, wait 5 seconds
6. Change start = 50
7. Run file again, wait 5 seconds
8. Change start = 75
9. Run file one last time
10. Check database to ensure that song_ids for reach week range from 1-100, even if there is not 100 items in each table, for both spotify_streams_week tables and BillboardxSpotify tables

For Yahoo Finance file:

1. Go to yahoo finance.py file
2. Run file once and see visualizations, click close in order to stop viewing each one
3. Writes calculations file, which can be viewed locally

7: Documentation for Each Function We Wrote (Including Input and Output)

All files:

All files have the same setUpDatabase function.

```
def setUpDatabase(db_name):
    '''
    Takes in database name; returns cursor and connection objects.
    Used in main to set up database.
    '''
```

Billboard_data.py file:

```
def get_top_100_billboard():
    '''
    Takes in nothing; returns list of dictionaries with song rank
     as key and song title, artist as values
     for each week of Billboard Hot 100 songs.
    Scrapes Billboard Hot 100 websites of last four weeks
     for song data using Beautiful Soup objects.
    '''
```

```
def top_100_into_database(billboard_lst, table_name, cur, conn):
    '''
    Takes in list returned from get_top_100_billboard(), table name,
     cursor, and connection; returns nothing.
    Creates table in database for given week of Billboard data and
     adds song information 25 items at a time.
    '''
```

```
def main():
    '''
    Main function of file.
    Sets up database and sets up list of dictionaries with song info.
    Adds song information for each week to corresponding table
     25 items at a time (for each of the four weeks).
    '''
```

Spotify_charts.py file:

```
def get_charts_data():
    '''
    Takes in nothing; returns list of dictionaries with song rank
     as key and song title, artist, and number of streams as values
     for each week of Spotify top 200 streamed songs.
    Scrapes SpotifyCharts HTML files of given weeks below for
     song data using BeautifulSoup objects.
    '''
```

```
def database(dictr, week, name, start, cur, conn):
    '''
    Takes in given weeks dictionary of Spotify Charts data, week variable,
     name of table to be created, start variable, cursor, connection; returns nothing.
    Creates Spotify Streams table for given week
     with Billboard rank (song_id) and number of streams
     by looping through Spotify Charts dict 25 items at a time.
    '''
```

```
def joining_billboard_spotify_tables(week, cur, conn):
    '''
    Takes in week variable, cursor, connection; returns nothing.
    Uses SQL query with JOIN on song_ids to create a new, combined table of
     song info from Billboard table and Spotify table for given week
     (with billboard rank, song title, artist, and spotify streams).
    '''
```

```
def streams_visualisation(cur, conn):
    '''
    Takes in cursor, connection; returns nothing.
    Finds MAX numbers of streams using SQL query from
     joined BillboardxSpotify table for each week
     and creates a scatter graph using Seaborn showing the
     change over last four weeks.
    '''
```

```
def main():
    '''
    Main function of file.
    Sets up database and sets up list of dictionaries with song info.
    Adds song information for each week to corresponding table
     25 items at a time (for each of the four weeks) with changing start value.
    Joins tables in database to create new tables with
     billboard rank and number of spotify streams.
    Shows visualization.
    '''
```

Yahoo Finance Data.py file:

```
#Get Yahoo Finance Spotify Data from APIs
def get_Spotify_Price():
    '''
    Takes in nothing and returns price_list
    Uses Yahoo Finance API to get day to day closing price for the Spotify
Stock
    Appends each day's price into a list
    '''
def get_Spotify_Volume():
    '''
    Takes in nothing and returns volume_list
    Uses Yahoo Finance API to get day to day trading volume for the
Spotify Stock
    Appends each day's volume into a list
    '''
#Calculate Day-to-day Variance
def get_Variance():
    '''
    Takes in nothing and returns variance_list
    Uses Yahoo Finance API to get day to day price for the Spotify Stock
    Then, calculates the variance by subtracting each day's price from the
day prior
    Beginning with the second day as the first day has no prior day to
compare to
    Appends each day's variance into a list
    '''
#Connect to Database for Data Retrieval
def setUpDatabase(db_name):
```

```
    '''
    Takes in db_name and returns cur, conn
    Connects to database (in this case final.db)
    Establishes cur and conn variables for later use with the database
    '''
#Retrieve Number of Streams from the Database
def retrieve_Streams(cur, conn):
    '''
    Takes in cur and conn and returns total_streams
    Uses database connection and SELECT SQL command to retrieve the
streams for each week
    Sums each weeks streams
    Adds the sum for each week into a list of total streams for each week
    '''
#Visualize Variance
def visualize_Variance():
    '''
    Takes in nothing and returns variance_Graph
    Uses get_Variance function from above to get list of day-to-day
variance of Spotify's stock price
    Uses a pandas dataframe in order to properly label and set the data
    Uses seaborn and matplotlib to visualize variance_graph
    '''
#Visualize Volume
def visualize_Volume():
    '''
    Takes in nothing and returns volume_graph
    Uses the code from get_Volume to retrieve data for volume
    Uses seaborn and matplotlib to visualize volume_graph
    '''
#Visualize a linear regression plot for the two variables of Streams and
Stock Price
def visualize_correlation(cur, conn):
    '''
    Takes in cur and conn and returns visualizeCorrelation
    Creates the linear regression plot for the variables of Spotify
Streams and Spotify Stock Price
    Uses retrieved streaming data from retrieve_Streams
    Uses pandas dataframes to combine the data for use
    matplotlib and seaborn are used to visualize visualizeCorrelation
```

```
    '''
#Write Calculations to Text File which can then be zipped
def write_to_text(calculation1, calculation2, calculation3, calculation4):
    '''
    Takes in the four calculations from above and returns None
    Writes calculations into a text file in directory
    '''
#Main function of the file to call the code
def main():
    '''
    Takes in nothing and returns None
    Used to set up and run all the code above
    Utilizes cur, conn, and calculations 1-4 variables as well as final.db
string
    To make sure each function has the necessary variables to run properly
    '''
```

8: Documentation of Resources Used

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
4/12	Scraping website that requires a login	https://eonofrey.medium.com/scraping-data-behind-site-logins-with-python-ee0676f523ee	No; this solution did not work, ended up downloading HTML files
4/14	Combining the JOIN and CREATE TABLE AS statements in SQL	https://stackoverflow.com/questions/29450301/sqlite-left-join-syntax-with-create-table-as https://www.techonthenet.com/sqlite/tables/create_table_as.php	Yes, the problem was solved.
4/15	Needing to clean strings of data and replace letters with blanks	https://www.w3schools.com/python/ref_string_replace.asp	Yes, the problem was solved.
4/15	Looking for specific	https://www.techonth	Yes, the problem was

	item in a database using SQL	enet.com/sql/in.php#:~:text=The%20SQL%20IN%20condition%20	solved.
4/18	Database was locked and unable to be edited/used	https://www.convertertools.org/blog/sqlite-database-is-locked-error-code-5/	Yes, the problem was solved.
4/18	How to order variables in creating visualizations	https://seaborn.pydata.org/index.html	Yes, the problem was solved.
4/18	How to use dataframes to combine two different sets of data	https://pandas.pydata.org/docs/user_guide/index.html#user-guide	Yes, the problem was solved.
4/18	How to select data from the database without generating a tuple when processing the data	https://stackoverflow.com/questions/59340771/how-to-fetch-data-from-mysql-as-int-instead-of-tuple-in-python	Yes, the problem was solved.
4/18	Graph visually unappealing due to difficult number of ticks, showed how to custom set ticks	https://www.statology.org/seaborn-ticks/	Yes, the problem was solved.
4/17	Yahoo Finance API originally selected had paywall and API Key issues, had to select new one in order to get data	https://pypi.org/project/yfinance/	Yes, the problem was solved.
4/17	How to plot a list of values against their index position	https://stackoverflow.com/questions/66727913/seaborn-plot-list-of-values-vs-their-indexes	Yes, the problem was solved