|  |  |
| --- | --- |
| A picture containing object, table, sitting, stop  Description automatically generated  THE DOWNFALL OF THE BULLs!  IS IT THE RIGHT TIME TO INVEST IN STOCKS? LET’S FIND OUT! | Mourya Karan Reddy Baddam - 5564234  CSCI 5751 – BIG DATA ENGINEERING AND ARCHITECHTURE – SPARK PROJECT |

Table of Contents

[Disclaimer 2](#_Toc39704038)

[1. Objective 3](#_Toc39704039)

[2. Datasets 3](#_Toc39704040)

[2.1 Background 3](#_Toc39704041)

[3. Project Execution 4](#_Toc39704042)

[3.1 Fetching Data 4](#_Toc39704043)

[3.2 Data Cleaning and Processing 4](#_Toc39704044)

[3.3 Category Indexes and Weights 4](#_Toc39704045)

[3.4 Time Series Modelling, Prediction and Analysis 4](#_Toc39704046)

[4. Summary 9](#_Toc39704047)

[5. Learnings and Challenges faced 9](#_Toc39704048)

[Appendix 10](#_Toc39704049)

# Disclaimer

Stock investments are subject to market risks. This project is only an analysis of trends in the current scenario. Any suggestion made in it is the sole opinion of the author and is **not a financial advice for investment**.

# Objective

The project aims at analyzing the trends in US stock market and the impact of COVID-19 on the stocks of various business categories. The goal of the project is to quantify the impact of COVID-19 on various categories of stock market and to suggest the categories that are most likely to be profitable if invested in the current scenario, using Apache Spark on Databricks cloud platform.

The project is intended for people who are interested in investing in stocks and for technical audience interested in Time Series modelling and data analysis or any general audience interested in the trends of the US Economy.

# Datasets

I used three public datasets for the project, which are described in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Yahoo! Finance Stock Prices | Yahoo! Finance Stock Info | US COVID-19 confirmed cases (COVID-19) |
| Format | CSV | CSV | CSV |
| Volume | Rows – 587  Columns – 3025 | Rows – 124  Columns – 505 | Rows – 350000 approx.  Columns – 13 |
| Description | The dataset contains the Adjusted Close, Close, High, Low, Open Prices and Volume traded for 504 S&P 500 component stocks date-wise in the time frame below. The quantity of interest for us is Close price. | The dataset contains various information regarding each of the 504 S&P 500 component companies. The quantity of interest for the project is the Market-Cap. | The Dataset contains the information regarding the location like region, latitude, longitude, State and the number of confirmed COVID-19 cases in the region, for various regions in the US for each day in the time frame. |
| Source | <https://finance.yahoo.com/> | <https://finance.yahoo.com/> | <https://github.com/datasets/covid-19/blob/master/data/us_confirmed.csv> |
| Time Frame | 01/02/2018 – 04/30/2020 | 01/02/2018 – 04/30/2020 | 01/22/2020 – till date |
| Method | Web Scraping | Web Scraping | Direct Download |

## Background

Capitalization-weighted Index [1]: Cap-weighted index is a stock market index whose components are weighted according to the total market value of their outstanding shares (held by investors).

A cap-weighted index is calculated as follows:

Weight of a component stock =

Cap-weighted index = Sum of (weight\*stock price) of all components in the index.

S&P 500 Index [2]: S&P 500 is a capitalization-weighted(cap-weighted) stock market index which measures the performance of 500 large companies listed on stock exchanges in the United States. It is comprised of 505 component stocks, which are divided into 11 broad categories[3] – Industrials, Health Care, Information Technology, Communication Services, Consumer Discretionary, Utilities, Financials, Materials, Real Estate, Consumer Staples and Energy.

# Project Execution

## Fetching Data

I used the opensource yfinance[4] Python web scraping API to download the Yahoo! Finance Stock Prices data and Yahoo! Finance Stock Info data. The API is not completely robust, and I had to edit the source code([base.py](https://github.com/ranaroussi/yfinance/blob/master/yfinance/base.py)) to incorporate error handling to download the stock info for a few stocks. I also incorporated multiprocessing to make the downloads faster using 100 processes. I uploaded the code and the datasets to a public GitHub[5] repository to make it easy to download into Databricks. The COVID-19 dataset is updated daily and is readily available to download from GitHub. I downloaded all the 3 datasets to Databricks platform from GitHub using wget.

## Data Cleaning and Processing

The datasets had a lot of columns which were not of importance to the project. Hence, I extracted only the required information from the datasets as described below:

Stock Prices dataset – I extracted the Date and close price column of all the 504 components and replaced null values with zeros since the number of nulls were very less compared to total number of cells.

Stock Info dataset – I extracted the market-cap row for all the 504 component stocks.

COVID-19 – I extracted the Date and Total Cases by date from the dataset. I then calculated the number of new cases for each date by subtracting the total cases of previous date.

## Category Indexes and Weights

Since the goal of the project is to compare the performance of different business categories, I divided the 504 components into 11 categories. For each category, I calculated the weights for the components belonging to that category from the market-cap datasets. Using those weights and the close price from the stock price dataset, I calculated the cap-weighted index for each category.

## Time Series Modelling, Prediction and Analysis

I divided the calculated indexes and the S&P 500 index into training and test datasets for each category, based on the start date from the COVID-19 dataset.

Using Fbprophet[6], an open source Python Time Series modelling API from Facebook, I trained 12 time series models, 1 for each category index and 1 for the S&P 500 index, on the training data(before COVID-19). I used the trained models to predict the indexes for 100 days beyond the training data, since there was an overlap of 100 days between stock prices and covid-19 datasets. The plots for predictions of S&P 500 index and Consumer Staples index are as shown below. The plots for other indexes can be found in the Appendix [7].

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

The black dots in the plots represent the actual data from the training set. The solid blue line is the model prediction. The light blue region is the 95% confidence interval for the prediction.

Below are the plots of the actual prices (training and test together) for some indexes. Other plots can be found in the Appendix [8].

A close up of a map

Description automatically generated

As we can clearly see, though the predictions showed an upward trend for almost all the indexes, the indexes crashed during the test (covid-19 time frame) period, indicating dependence on covid-19 cases.

Since the behavior of the Indexes is not predefined, it makes more sense to analyze the deviation from expected behavior with respect to new cases for a day. Hence, I plotted the percentage difference between the prediction and actual stock indexes (test data) and the new cases vs dates. Some of them are given below and the rest can be found in Appendix [9].

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

# Summary

The first plot above is the trend for new cases. The plots below are the trends for percentage difference in the indexes. We can see that the percentage difference increased, meaning the indexes crashed, as long as the new cases increased. Once the new cases curve started flattening (a bit before that, which is when the Shelter In Place was enforced in many states in the US resulting in the curve flattening), we see the percentage difference started to fall indicating that the indexes were increasing and the economy was reviving.

So, we can see that the Consumer Discretionary Index, Health Care Index, Real Estate Index and Consumer Staples Index recovered most. But looking at the potential predictions and the recovery rate, Industrials, Information Technology, Communication Services and Financials are the fields that have the highest immediate potential for growth. Hence, I think investing in those fields is probably profitable given the above facts.

Though the Energy Index looks like it has not revived yet, if we look at the series plot from above (the one in grey), it has a downward trend. Hence, I wouldn't suggest investing in the field of Energy.

# Learnings and Challenges faced

Overall, the whole project has been a great learning experience for me. But there are certain aspects that stand out.

1. To choose a project with undefined requirements is always challenging. Given the short time frame, the biggest issue is to determine if the execution is feasible with available resources (datasets, technologies). Hence, a backup project always comes to the rescue in such scenarios.
2. Open Source APIs: Open source APIs are great resources when it comes to data collection and project execution. But they are not always robust, efficient and fault resistant. So, we might have to tweak them to get the desired output.
3. Spark Optimizations: The Catalyst optimizer does very well with lazy execution. But when iterative processes are involved in the code, for example loops, in which the dataframes are being called, the dataframes are re-computed in every iteration which takes very long. The same happened while calculating weights in this project. The best way to handle this issue is to collect the dataframe into memory before the loop starts. This reduced the execution time for ‘Weights’ block from 3:30 mins to about 1 sec.
4. Data granularity: When dealing with time series data, the granularity of the data affects the performance of the model greatly. Based on the granularity, the model captures various seasonalities in the data. For example, the models cannot capture daily seasonality if the data is daily data. It requires sub daily data like hourly or per minute data.
5. KPIs: The outcomes and perspectives of the project greatly depends on the KPIs defined for the project. For example, the value of the indexes was a weak indicator of the dependence on COVID-19 cases. But the percentage difference of actual index from predicted index shows a clear dependence on number of new cases.

# Appendix

1. Cap-Weighted Index: <https://en.wikipedia.org/wiki/Capitalization-weighted_index>

<https://xplaind.com/317165/capitalization-weighted-index>

1. S&P 500 Index: <https://en.wikipedia.org/wiki/S%26P_500_Index>
2. List of S&P 500 stocks: <https://en.wikipedia.org/wiki/List_of_S%26P_500_companies>
3. Yfinance: <https://pypi.org/project/yfinance/>

<https://github.com/ranaroussi/yfinance>

1. GitHub repository: <https://github.com/aiBoss/Spark-Data>
2. Fbprophet: <https://facebook.github.io/prophet/>
3. Spark Loops Optimization: Refer to the first reply to the question at <https://stackoverflow.com/questions/40892459/spark-transpose-dataframe-without-aggregating>
4. Prediction Plots:

A picture containing indoor, text, map, black

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

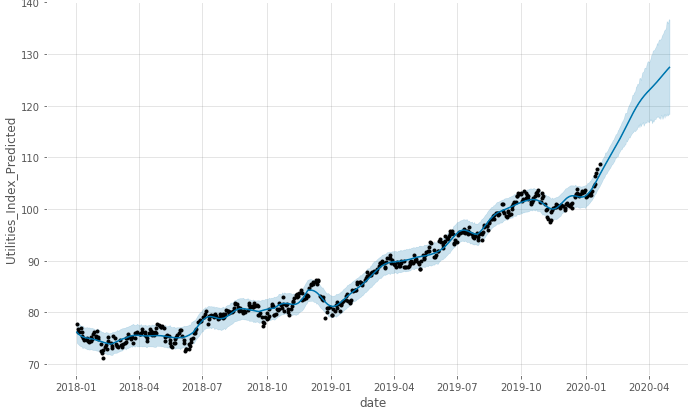
Description automatically generated

A close up of a map

Description automatically generated

A picture containing text, map

Description automatically generated



A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

1. Series Plots:

A close up of text on a white background

Description automatically generated

A close up of a map

Description automatically generated

1. Percentage Difference Plots:

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of text on a white background

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated

A close up of a map

Description automatically generated