Stock Market Prediction

G1 - StockSight

Data Science Capstone Project
Data Acquisition and Pre-Processing Report

Date:

05/14/2020

Team Members:

Chengyi Wang

Richard Hong

Joan Kibaara

Manoj Venkatachalaiah

Introduction

Stock markets play a key role in the economy. Investors and corporations are interested in how the market moves. Stock market can be highly volatile. Some factors that cause the volatility include regional and national economic crises, major news announcements and even pandemics like COVID-19. A company's success can be measured by the volatility of its stock prices. Successful businesses tend to have stable or increasing prices over a period, despite any external disruptions.

Identifying Data

We identified 3 major data sources to use for our project.

Stock Data

There are a variety of stock market APIs that provide both real-time and historical prices across different markets. After comparing a few different APIs, we decided to get our data from Yahoo!Finance. Although Yahoo!Finance decommissioned their historical data API, we were still able to utilize a python library "yfinance" to download historical market data from Yahoo!Finance.

Although there are several thousand companies that are traded publicly in the US market, not all of them have significant market value. Companies listed in the S&P 500 makeup about 80% of the total market value. We decided to select a subset of companies from the S&P 500 stock index.

Google Trends Data

At the beginning of this project, we had tried to acquire daily based time-series text data from web scripting of Google news headlines. However, we could discover it unattainable to obtain the data over the past decade through Google News Web-scraping.

Google allows all users to access data on relative search volume of keywords through Google trends service. The data doesn't represent daily searching counts for specific keywords but search trends and relative search volume for the selected time frame. The data can be filtered by various criteria(such as geographical location, subject areas, different time frames, etc.) In this project, the data of 39 keywords were filtered by geographic location(US) and timeframe(1/1/2011 to 4/30/2020).

News Data

We had to look at a data source that could give us daily news headlines for the time frame we had settled on(a little over 10 years starting from 2011). We ran into a number of sources such as Reuters, Wall Street Journal etc., but all of them could give us news data for only the last 45-60 days and not beyond that. The only way we could get the daily headlines for the past 10 years was to scrape it (or make use of the API) from a subreddit **'r/news'.**

r/news is a subreddit where news articles from many different journals related to events happening all over the world are posted. It contains company specific news for Google, Microsoft etc., and also news related to an event such as the Presidential Elections, Covid-19 etc,. which is suitable for our project.

Acquisition Process:

Stock Data

Sure Dividend is a website that provides a lot of free information to individual investors. We were able to get the list of companies that are listed in the S&P 500 in 2020. Using this list, we identified a subset of 30 companies to use in our data set. The criteria used to select the 30 companies was based on their market capital. After reviewing our list, we decided to exclude one record of Alphabet, Inc since it was duplicated. Also although Berkshire Hathaway, Inc, MasterCard, Inc and Visa, Inc were part of the top 30 companies, we decided to exclude them from our data set because we had insufficient data. Using excel, we filtered the 30 companies, and stored the data in a csv file to be used to extract the historical stock information.

To get the stock data, we used the yfiance library in python .This library contains a ticker module which allowed us to access ticker data. The data we downloaded included over 10 years worth of stocks daily prices. For each company, we extracted the open , high, low, close, adj close and volume prices. We stored each company's results in a csv file for pre-processing.

Google Trends Data

We used **Pytrends** python module to obtain Google Trends data. Google trends have some limitations: the time resolution limitation based on the requested time frame and quota limits. For example, Google Trends provides hourly search trends for the time range of the last seven days. Daily data is presented for the time range shorter than nine months. Weekly data is provided for the time range between nine months and five years. For any time range longer than five years, it returns monthly data. Therefore, if we request data for the past ten years, we can only get monthly data, which is not ideal for our project. So We collected monthly data of the last ten years and also concatenated daily data of multiple one-month. Another limitation is quota limits(requesting limitations). If the user has sent too many requests in a given amount of time, the user would be blocked. To achieve daily data, we had to make 112 requests per keyword, so it was necessary to handle the quota limits properly.

News Data

We acquired the news data using the reddit API through **PRAW**. **PRAW**, an acronym for "Python Reddit API Wrapper", is a python package that allows for simple access to reddit's API. Since we were looking to acquire the headlines, which is basically the text for a given post (submission) we made use of a pushshift api wrapper called **PSAW,** a minimalist wrapper for searching public reddit comments/submissions via the pushshift.io API. In our case, it was just the submissions that we needed to acquire.

In order to use the reddit API, we had to create a new application in the reddit API portal, after which we got a Client_id and Client_secret both of which are needed to use the reddit API. We used these parameters to create a reddit API object, which is passed through PSAW to create an object that would then retrieve the submissions (headlines) for a given set of parameters. Some of these parameters include keyword, time frame, limit (the number of headlines we want per day), sort (based on what the api would get the number of headlines we set as limit), the subreddit we wanted to get the data from. The value for the subreddit was 'r/news', the limit was set to 25 because we figured 25 headlines would be more than enough to get the general sense of the sentiment of news on a given day, the keywords were the names of all 30 companies that we had chosen, and other event based keywords such as 'pandemic', 'virus', 'elections' etc,. since we're interested in looking at how the stock market reacts to everything happening in the world. So for every day starting from January 1st 2011, we acquired the top 25 headlines (sorted based on score: upvotes), for each company and each keyword (pandemic,disease, elections,virus, economy) and merged all the headlines to create one single string for a given day and stored them based on dates for preprocessing.

Data Integration

We will discuss this part of the report after the '**Data-Processing'** section, because we integrated all the acquired data after preprocessing.

Issues:

We couldn't manage the google trends quota limit properly. We can only request six keywords every 12 hours. To obtain the data of 39 keywords, each team member had to acquire data of 10 keywords for about two days.

Data-Processing

Stock Data

We first acquired the stock data for top 30 companies. Then we found that there are two companies that are the same:Alphabet, Google as well as three credit other companies that did not have any meaningful data. We replaced them with four other companies and acquired the data for them. The data acquired has 8 columns: Company ticker,Date, Open, High, Low, Close, Adj Close, Volumn. We found that there is data missing for certain dates, which are holidays and weekends(the stock market is closed on these days). Before we analyze the data we will need to fill in those missing values. We decided to fill the stock data for missing dates with the average of the previous and next available stock data. For example, we fill 2011-01-08 and 2011-01-09(Saturday and Sunday) with the average value of date 2011-01-07 and 2011-01-10(Friday and Monday) for all 30 companies. After that we will catch the closed price for each company during 10 years and combine them for further analysis.

Google Trends Data

The relative search volume for a keyword is indexed between zero and 100. Zero indicates the lowest relative search interest, and 100 means the maximum relative search interest within the selected time range. The data of every request has a result between 0 and 100 of search volumes in the requested time frames. If we just collected the daily data for the last ten years, the data would not adequately present the trends of the past decade. Because the daily data of each month with values between 0 and 100 would be repeated every month. To obtain the proper daily data for the last ten years, we needed to concatenate the daily data of multiple one-month and normalized it by corresponding monthly trends data. We used the monthly data as a weight for adjusting daily data. Here is the calculation: Daily data * (Monthly data/100) = Adjusted data.

The final google trends dataset has 8 columns, ‘date’, ‘company’, ‘company_trends’, ‘virus_trends’, ‘pandemic_trends’, ‘disease_trends’, ‘elections_trends’ and ‘economy_trends’. The company column contains the company ticker and the last 6 columns are the google trends score related to a specific topic(keyword).

News Data

After getting the top 25 headlines for each day using the api for different keywords, we now had a single string of text for each day. In order to do sentiment analysis on the text data, we made use of the built-in python sentiment analyzer from nltk.sentiment.vader,SentimentIntensityAnalyzer. But we updated this analyzer with a lexicon that was more related to the stock market, so that words and phrases related to the stock market would be given extra weight.

For each day, the text data was passed through the updated sentiment analyzer to retrieve a sentiment score. After processing all the text data, we now had a sentiment score ranging from -1 to 1 for each date for each keyword. We observed that the sentiment score for many of the dates was zero. This is because for a given date, no news could be generated for a particular company or keyword, and naturally the sentiment analyzer would return zero if no text data (an empty string) is passed through it. We decided to fill in these missing values in accordance with the google trends data. That is, if for two dates the sentiment score is 0 for one of them, it would get replaced with the sentiment score of the other date multiplied by a weight. This weight is the percentage in change of the google trends data from the missing value date to another. For example, if the sentiment score for two dates in 0.5 and 0 and the google trends data for the same two dates is 0.8 and 0.96 there is a 20% increase in the google trends data from date1 to date2. So the zero value for sentiment would be replaced with the value (0.5+20% of 0.5) which totals to 0.6.

The final news dataset has 8 columns, 'date', 'company', 'company_news', 'world_news', 'virus_news', 'pandemic_news', 'disease_news', 'elections_news' and 'economy_news'. The company column contains the company name and the last 7 columns are the sentiment score of the news related to a specific topic(keyword).

Data Integration

After preprocessing all the acquired data, we had 3 different datasets: Stock data, News data and Google Trends Data. The common column in all these 3 datasets is the company ticker, so all the 3 different datasets are merged to one based on the company ticker. The final dataset contains 30 different class(30 companies) and features for a particular class(company) would be the stock data for that particular company, the sentiment score data(for the news related to that company, world news and the news for other keywords we have chosen) and the google trends data(company as keyword) and other keywords we have chosen. The number of rows in the dataset is approximately 150,000 and the number of columns is 23. Each of the classes and features are defined in the data definition part of the report.

COVID-19

With everything that's been happening over the past few months we wanted a part of our project to focus on the pandemic factor alone to see how it has affected the stock market. We used our overall dataset and derived a subset of it to form the **COVID-19** dataset. This dataset will contain about 3 years worth of data from three different time frames: 2009-2010(H1N1), 2015-2016(Zika virus), 2019-2020(Covid-19). We have also included two months before and after the different pandemic time frames so that our model can pick up on the skewness in the different variables(stock prices, news, google trends) from when the pandemic started and ended.

With the covid-19 dataset, we wouldn't have 30 different classes because we are going to be looking at how the overall stock market has reacted to the pandemic. So instead of using the company specific data like we did in the previous part, we will be using the S&P 500 data for the different pandemic timeframes which is the average data for the top 500 companies. The dataset will have approximately 2000

rows and 19 columns, with the features being the same as the overall dataset excluding some of the company specific data (news and google trends.)

Appendix

Figure 1: Code

```
import requests
import pandas as pd
import numpy as np
import io, os
import yfinance as yf
```

```
#Read CSV file containing the subset of the data from SureDividend
sp_500 = pd.read_csv('SP_500_REVISED.csv')

#Create a dictionary with Ticker and Company name for the 30 selected companies
sp = pd.DataFrame(sp_500, columns = ["Ticker","Name"])
sp_dict = sp.to_dict()
sp.head(10)
i = 0

#Save Ticker Information in a list l
l = []
while i < len(sp_dict['Ticker']):
    a = sp_dict['Ticker'][i]
    i += 1
    l.append(a)

```

```
#Function to download each ticker data
def download_ticker(tick, start_date, end_date):

    filepath = 'data/'+tick+'.csv'

    #Check if file exists

    check_file = os.path.exists(filepath)
    if check_file == True:
        print(check_file)
    else:
        tickA = yf.download(tick,start_date, end_date)
        pd.DataFrame(tickA).to_csv(filepath)
```

```
for item in l:
    download_ticker(item,"2011-01-01", "2020-01-01")
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```

Figure 2: Sample of Microsoft Data prior to pre-processing

```
com1 = pd.read_csv('data/MSFT.csv')
sp1 = pd.DataFrame(com1)
sp1.head(30)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2011-01-03	28.049999	28.180000	27.920000	27.980000	22.420481	53443800
1	2011-01-04	27.940001	28.170000	27.850000	28.090000	22.508627	54405600
2	2011-01-05	27.900000	28.010000	27.770000	28.000000	22.436506	58998700
3	2011-01-06	28.040001	28.850000	27.860001	28.820000	23.093576	88026300
4	2011-01-07	28.639999	28.740000	28.250000	28.600000	22.917286	73762000
5	2011-01-10	28.260000	28.400000	28.040001	28.219999	22.612789	57573600
6	2011-01-11	28.200001	28.250000	28.049999	28.110001	22.524651	50298900
7	2011-01-12	28.120001	28.590000	28.070000	28.549999	22.877232	52631100

Figure 3: Sample of Microsoft Data afterpPre-processing

```
# Create additional rows for missing dates(Weekend and holidays)
sp1['Date'] =  pd.to_datetime(sp1['Date'], format='%Y-%m-%d')
sp1.set_index('Date', inplace=True)
sp1 = sp1.resample('D').ffill().reset_index()

#Create day_of_week_column
sp1['day_of_week'] = sp1['Date'].dt.day_name()

#Set weekend values = average of Friday and Monday data
for index, row in sp1.iterrows():
    if row['day_of_week'] == 'Saturday':
        mon = sp1.iloc[index-1,2:7]
        fri = sp1.iloc[index+2,2:7]
        avg = (mon + fri) / 2
        sp1.iloc[index,2:7] = avg
        sp1.iloc[index+1,2:7] = avg
sp1.head(30)
```

	Date	Open	High	Low	Close	Adj Close	Volume	day_of_week
0	2011-01-03	28.049999	28.180000	27.920000	27.980000	22.420481	53443800.0	Monday
1	2011-01-04	27.940001	28.170000	27.850000	28.090000	22.508627	54405600.0	Tuesday
2	2011-01-05	27.900000	28.010000	27.770000	28.000000	22.436506	58998700.0	Wednesday
3	2011-01-06	28.040001	28.850000	27.860001	28.820000	23.093576	88026300.0	Thursday
4	2011-01-07	28.639999	28.740000	28.250000	28.600000	22.917286	73762000.0	Friday
5	2011-01-08	28.639999	28.570000	28.145000	28.410000	22.765038	65667800.0	Saturday
6	2011-01-09	28.639999	28.570000	28.145000	28.410000	22.765038	65667800.0	Sunday
7	2011-01-10	28.260000	28.400000	28.040001	28.219999	22.612789	57573600.0	Monday
8	2011-01-11	28.200001	28.250000	28.049999	28.110001	22.524651	50298900.0	Tuesday

Figure 4: Code for Google trends

```
from pytrends.request import TrendReq
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
import csv
import calendar
```

```
time_ranges = []
for y in range(2011,2021):
    for m in range(1,13):
        monthrange = calendar.monthrange(y, m)
        start_day = "%d-%d-%d" %(y,m,1)
        end_day = "%d-%d-%d" %(y,m,monthrange[1])
        s_e_day = start_day +" " + end_day
        time_ranges.append(s_e_day)


print(time_ranges)
```

```
# 1/1/2011 to 4/30/2020
time_ranges = time_ranges[0:-8]
```

The First Run

```
keywords = [['Procter & Gamble','Procter_and_Gamble'],
['JPMorgan Chase','JPMorgan_Chase'],
['UnitedHealth Group','UnitedHealth_Group']]

# monthly daya for past 100 years
for key,df_name in keywords:
    pytrends = TrendReq(hl='en-US', tz=360)
    pytrends.build_payload([key], timeframe='2011-1-1 2020-4-30', geo= 'US', gprop ='')
    data_key_all_days= pytrends.interest_over_time()


    globals()[df_name+'_all'] = pd.DataFrame()

    for day in time_ranges:
        pytrends = TrendReq(hl='en-US', tz=360)
        pytrends.build_payload([key], timeframe=day, geo= 'US', gprop ='')
        df_day = pytrends.interest_over_time()

        #Calculation final value = daily value*(monthly value/100)
        Monthly_rate_percent = data_key_all_days.loc[day[0:9],key]/100
        df_day = df_day*Monthly_rate_percent
        globals()[df_name+'_all'] = globals()[df_name+'_all'].append(df_day)
    del globals()[df_name+'_all']['isPartial']

    globals()[df_name+'_all'].to_csv(df_name+".csv", encoding = "utf-8")
```

Figure 5 and 6: Code for Reddit news.

```
import requests
import datetime
import time
import csv
import json
from random import randint
import praw


r = praw.Reddit(client_id='J3XLJbhE9LB9mQ', client_secret='7NrPOUcTqRtzB9iBY1HsczZr8Ok', user_agent='NewsScraping')

api = PushshiftAPI(r)

#generating the dates for the time frame
date1 = '2011-01-01'
date2 = '2020-04-30'
mydates = pd.date_range(date1, date2)
companies=['Coca-Cola','Bank of America','Netflix','Pepsi','Disney']
```

```
for k in companies:


    date=[]
    news=[]

    for i in range(len(mydates)):
        if i!=len(mydates)-1:
            start=mydates[i].timestamp()
            end=mydates[i+1].timestamp()
            #extracting the top 25 headlines for each company in the list
            c=list(api.search_submissions(after=int(start),before=int(end), q=k,
                                          subreddit='news',sort_type='score',
                                          filter=['url','author', 'title', 'subreddit'],
                                          limit=25))

        else:
            start=mydates[i].timestamp()
            c=list(api.search_submissions(after=int(start), q=k,
                                          subreddit='news',sort_type='score',
                                          filter=['url','author', 'title', 'subreddit'],
                                          limit=25))

        headlines=''
        # merging all the headlines for a given day to create one single string
        for j in c:

            headlines=headlines+j.title
        star=mydates[i]
        date.append(star)
        news.append(headlines)
    #creating csv file for each company
    filename=k+'.csv'
    df=pd.DataFrame({'date':date,'news':news})
    df.to_csv(filename,header= True, encoding='utf-8')
```

Data Definition

The target variable in both the datasets is ‘Close’, i.e., the close price of a stock for a given date. We will be trying to predict the close price of a stock given the features consisting of different news sentiment score variables and different google trends score variables.

Company ticker	Each company has its own ticker, therefore we will have 30 different classes in our dataset.
Open, High, Low, Close, Adj Close, Volume	These are the variables consisting of the stock data for respective companies on a given date.
company_news, worlds_news	These are the variables consisting of the stock data for respective companies on a given date.
pandemic_news,disease_news, elections_news,virus_news, economy_news:	These are the variables consisting of the sentiment score for company specific news and world news.
company_trends:	Variable for google trends score for the company name as a keyword.
pandemic_trends,disease_trends, elections_trends,virus_trends, economy_trends:.	These are the variables consisting of google trends score for each of the keywords chosen

Sample Data

Figure7: Final dataset

	Company ticker	date	Open	High	Low	Close	Adj Close	Volume	company_news	world_news	company_trends
0	MSFT	2011-01-03	28.049999	28.180000	27.920000	27.980000	22.420481	53443800.0	0.679823	0.7987	88.00
1	MSFT	2011-01-04	27.940001	28.170000	27.850000	28.090000	22.508627	54405600.0	0.733900	-0.8556	95.00
2	MSFT	2011-01-05	27.900000	28.010000	27.770000	28.000000	22.436506	58998700.0	0.757076	-0.9796	98.00
3	MSFT	2011-01-06	28.040001	28.850000	27.860001	28.820000	23.093576	88026300.0	0.726175	0.0990	94.00
4	MSFT	2011-01-07	28.639999	28.740000	28.250000	28.600000	22.917286	73762000.0	0.687548	-0.5754	89.00
...	...	...	...	...	...	...	...	...	...	...	...
3280	MSFT	2019-12-27	159.449997	159.550003	158.220001	158.960007	158.527008	18412800.0	0.906600	-0.9656	38.22
3281	MSFT	2019-12-28	159.449997	159.285004	157.474998	158.275002	157.843872	17380600.0	0.674138	0.8124	28.42
3282	MSFT	2019-12-29	159.449997	159.285004	157.474998	158.275002	157.843872	17380600.0	0.581154	0.5770	24.50
3283	MSFT	2019-12-30	158.990005	159.020004	156.729996	157.589996	157.160736	16348400.0	0.739700	-0.9489	35.28
3284	MSFT	2019-12-31	156.770004	157.770004	156.449997	157.699997	157.270432	18369400.0	-0.580500	0.2243	32.34

Figure8: Covid-19

	date	Open	High	Low	Close	Adj Close	Volume	virus_news	pandemic_news	virus_trends
1459	2015-01-01	46.730000	47.439999	46.450001	46.450001	41.587284	21552500.0	0.383700	0.1432	33.84
1460	2015-01-02	46.660000	47.419998	46.540001	46.759998	41.864841	27913900.0	-0.207100	0.7539	41.04
1461	2015-01-03	46.660000	47.074999	46.395000	46.545000	41.672354	33793900.0	-0.476500	-0.9880	38.16
1462	2015-01-04	46.660000	47.074999	46.395000	46.545000	41.672354	33793900.0	-0.521453	0.9035	41.76
1463	2015-01-05	46.369999	46.730000	46.250000	46.330002	41.479866	39673900.0	0.645200	0.6441	51.84
1464	2015-01-06	46.380001	46.750000	45.540001	45.650002	40.871037	36447900.0	0.810700	0.7543	52.56
1465	2015-01-07	45.980000	46.459999	45.490002	46.230000	41.390320	29114100.0	0.788489	-0.9909	51.12
1466	2015-01-08	46.750000	47.750000	46.720001	47.590000	42.607944	29645200.0	0.821805	-0.9865	53.28
1467	2015-01-09	47.610001	47.820000	46.900002	47.189999	42.249817	23944200.0	0.766278	-0.9540	49.68
1468	2015-01-10	47.610001	47.680000	46.630001	46.894999	41.985703	23798050.0	0.010600	-0.9620	39.60
1469	2015-01-11	47.610001	47.680000	46.630001	46.894999	41.985703	23798050.0	0.010215	-0.9571	38.16
1470	2015-01-12	47.419998	47.540001	46.360001	46.599998	41.721588	23651900.0	0.519900	-0.6915	54.00
1471	2015-01-13	46.970001	47.910000	46.060001	46.360001	41.506714	35270600.0	0.526832	-0.4174	54.72
1472	2015-01-14	45.959999	46.240002	45.619999	45.959999	41.148579	29719600.0	0.554560	0.1971	57.60
1473	2015-01-15	46.220001	46.380001	45.410000	45.480000	40.718842	32750800.0	0.636400	-0.8781	55.44
1474	2015-01-16	45.310001	46.279999	45.169998	46.240002	41.399284	35695300.0	0.302900	0.9451	52.56
1475	2015-01-17	45.310001	46.279999	45.169998	46.240002	41.399284	35695300.0	0.491000	0.2716	36.00
1476	2015-01-18	45.310001	46.279999	45.169998	46.240002	41.399284	35695300.0	0.520460	-0.9510	38.16
1477	2015-01-19	45.310001	46.279999	45.169998	46.240002	41.399284	35695300.0	-0.229100	-0.0263	47.52
1478	2015-01-20	46.299999	46.650002	45.570000	46.389999	41.533566	36161900.0	0.037600	-0.9296	56.16
1479	2015-01-21	45.939999	46.139999	45.480000	45.919998	41.112770	39081100.0	0.882200	0.7196	63.36
1480	2015-01-22	46.380001	47.139999	46.080002	47.130001	42.196106	35898000.0	0.180700	-0.9665	72.00
1481	2015-01-23	47.360001	47.389999	46.799999	47.180000	42.240868	26211600.0	-0.270100	-0.7299	59.76