Project report for CSE 3025

Stock Market Sentiment Analysis using various Big Data tools and their Comparative Analysis.

Vedang Sawarkar	**19BCE1303**
Divyanshi Thapa	**19BCE1367**
Prakriti Sharma	**19BCE1655**

Abstract

The main goal of this project is to analyze the stock market with the help of sentiment analysis done on stock market data collected from twitter using the Hadoop ecosystem. In a financially volatile market, as the stock market, it is important to have a very precise prediction of a future trend. Because of the financial crisis and scoring profits, it is mandatory to have a secure prediction of the values of the stocks.The successful prediction of a stock's future price could yield significant profit. Investors are constantly aware of the behaviour of stock markets. This affects their emotions and motivates them to buy or sell shares. Financial sentiment analysis allows us to understand the effect of social media reactions and emotions on the stock market and vice versa. Twitter, which has large audience potential, currently attracts an estimated average of 271 million users every month. So twitter-based models can then be built to aggregate the opinions of the collective population. They can be used to predict future trends while gaining useful insights into individual behavior. Keeping this in mind, we will be collecting the twitter data using NLP techniques and processing the data. Now there are a number of big data tools available so we will be using different tools in the Hadoop ecosystem to analyze the data and help in predicting the movement of the stock. Then the ability and efficiency will also be used to compare the tools.

Motivation

Stock Market is the financial ground on which a high amount of data is released at every single point of time, which is very complex and nonlinear in nature. These data sets are used to predict the high profits and risks that a user can have in investing in a company. Manual predictive ways like studying the stock market with graphs, news, important dates, and many more practices failed many times which led to loss of profits because trading on the stock market required accurate and timely inputs. The magnitude of data that is generated within the stock market on a daily basis is impossible to be managed, analysed and made sense of by human beings due to the sheer volume of data generated and speed at which this financial data is being generated from various sources. Moreover, analyzing this much amount of data takes a lot of time. So, to overcome this challenge, we need some big data tools to analyze and give us filtered results. In the current day and age, financial analysis alone is no longer adequate for examining share prices and share price behaviour. These financial analyses need to be integrated with external factors such as social and economic trends within the economy, political environment, consumer behaviour and preferences etc. Social media platforms like Twitter are one of the biggest resources of this data. So we came up with an idea of generating common reviews of people regarding different stocks using sentiment analysis and then to predict stock from there. As the data is of a very huge amount, big data tools are required but there are so many tools available with their own speciality, it becomes difficult to choose. So we decided to use various tools and perform a comparative analysis on efficiency and accuracy of the tools.

Introduction

Forecasting of stock market returns is a challenging research activity that is now expanding with the availability of new data sources, markets, financial instruments, and algorithms. At its core, the predictability of prices still raises important questions. Stock markets are driven by volatile factors such as microblogs and news that make it hard to predict stock market index based on merely the historical data. The enormous stock market volatility emphasizes the need to effectively assess the role of external factors in stock prediction.

Big data is pushing the financial industry, and also has an impact on investing. Huge amounts of data are generated each day since online trading has simplified the job and it's easier to view the market from your mobile by using an online trading platform or various stock trading applications. Social networks offering microblogging services enable the rapid spread of user generated content (UGC) from a handful of individuals to millions of people around the world in the form of short text, images or videos. Microblogging platforms have grown so exponentially, that they are now perceived as indispensable sources of information and are fast gaining popularity amongst users, organizations and researchers in various disciplines. The rich and continuous mass of data made available by these platforms is being harnessed with the purpose of studying individual and group behavior as well as global patterns especially in regards to sentiment towards the stock market trend. Twitter is currently the 10th most popular website globally with over 300 million active monthly users and much data related to stock market trends can be harnessed from this platform. There is a predictive relationship between twitter mood and stock movements but only up to a certain point. It was found in many research activities that positive correlation exists between volume of positive tweets and close prices, but in a time lag of 48 h, negative tweets are better indicators.

So by making the use of twitter data and performing sentiment analysis on it, we can find the trend of the stock market. But as twitter is a huge platform, the data generated will be of large amount and here comes the big data tools in picture. There are many research works going on for understanding the trends via social media platforms but the first and foremost question for these researches is which tool is to be used? There are many big data tools present right now and choosing the correct tool is necessary. So here in our project we apply sentiment analysis for predicting the stock market and also we will be comparing the efficiency of some famous big data tools.

Objectives

1. Extraction of tweets related to stock market from twitter.
2. Processing and cleansing of data.
3. Hashtags extraction for future works.
4. Sentiment analysis on the clean dataset.
5. Visualization of the analysis result and prediction of trend.
6. Comparing the efficiency of various big data tools like Hive, Apache Spark and Cassandra by doing analysis by each one of them on the twitter data.

Software and Hardware requirements

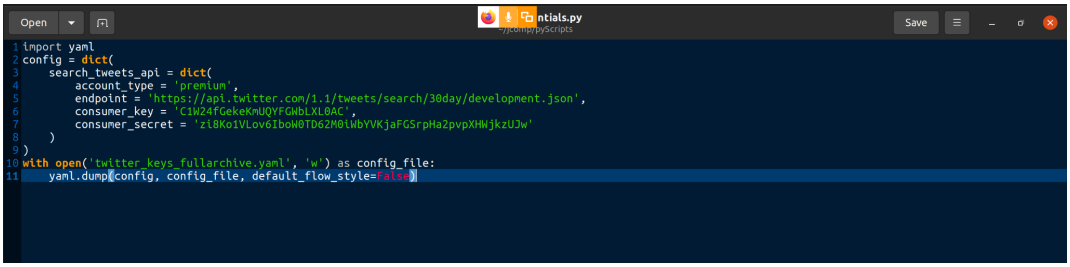
- The software and the hardware requirements of our project are listed below

1) Twitter developer account.
    - For generating secret keys for using the API to extract tweets.
2) System with linux environment.
3) Hadoop version 3 and above.
    - For storing the big data and processing it.
4) Hive version 2 and above
    - For sentiment analysis and querying the data.
5) Apache Spark
    - For querying the data.
6) Cassandra
    - For querying and comparison.
7) Public account on Tableau
    - For data visualization.
8) Jupyter Notebook.
    - For executing pySpark scripts.

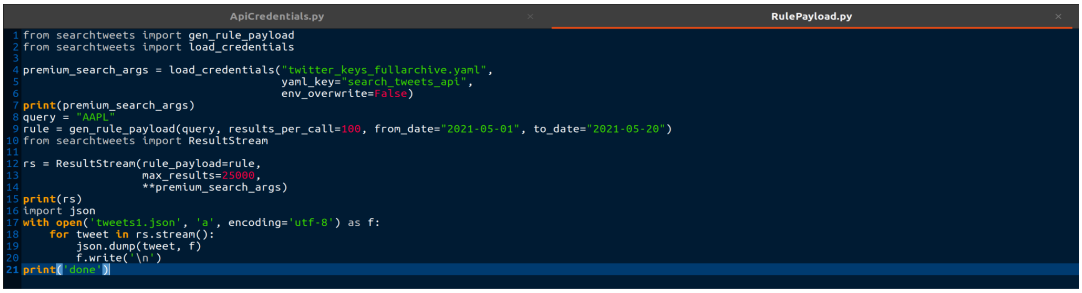
Project Explanation

Tweets extraction

To extract the tweets, we used different API's or library's in python. So we used to retrieve the tweets using searchtweets library. This code is importing the credentials from the other file and sending a query(here we are querying AAPL) between the dates 1st May 2021 to 21st May 2021.



```
import yaml
config = dict(
    search_tweets_api = dict(
        account_type = 'premium',
        endpoint = 'https://api.twitter.com/1.1/tweets/search/30day/development.json',
        consumer_key = 'C1W24fGekeKmUQYFGWbLXL0AC',
        consumer_secret = 'zi8KolVLov6IboW0TD62M0lWbYVKjaFGSrpHa2pvpXHWjkzUJw'
    )
)
with open('twitter_keys_fullarchive.yaml', 'w') as config_file:
    yaml.dump(config, config_file, default_flow_style=False)
```

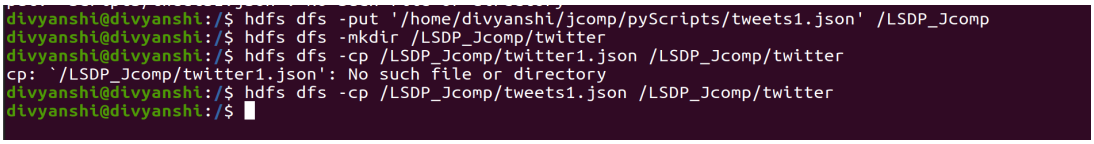


ApiCredentials.py

```
from searchtweets import gen_rule_payload
from searchtweets import load_credentials

premium_search_args = load_credentials("twitter_keys_fullarchive.yaml",
                                       yaml_key="search_tweets_api",
                                       env_overwrite=False)
print(premium_search_args)
query = "AAPL"
rule = gen_rule_payload(query, results_per_call=100, from_date="2021-05-01", to_date="2021-05-20")
from searchtweets import ResultStream

rs = ResultStream(rule_payload=rule,
                  max_results=25000,
                  **premium_search_args)
print(rs)
import json
with open('tweets1.json', 'a', encoding='utf-8') as f:
    for tweet in rs.stream():
        json.dump(tweet, f)
        f.write('\n')
print('done')
```

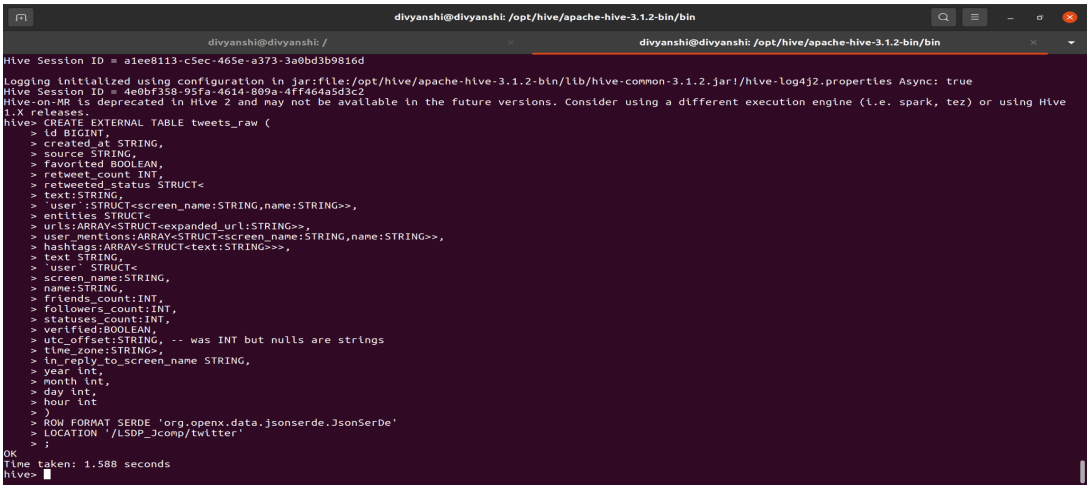
RulePayload.py

Storing in HDFS



```
divyanshi@divyanshi:/$ hdfs dfs -put '/home/divyanshi/jcomp/pyScripts/tweets1.json' /LSDP_Jcomp
divyanshi@divyanshi:/$ hdfs dfs -mkdir /LSDP_Jcomp/twitter
divyanshi@divyanshi:/$ hdfs dfs -cp /LSDP_Jcomp/twitter1.json /LSDP_Jcomp/twitter
cp: `/LSDP_Jcomp/twitter1.json': No such file or directory
divyanshi@divyanshi:/$ hdfs dfs -cp /LSDP_Jcomp/tweets1.json /LSDP_Jcomp/twitter
divyanshi@divyanshi:/$
```

We created the table in hive tweets_raw with the few required attributes from the tweet object which we extracted from twitter api.

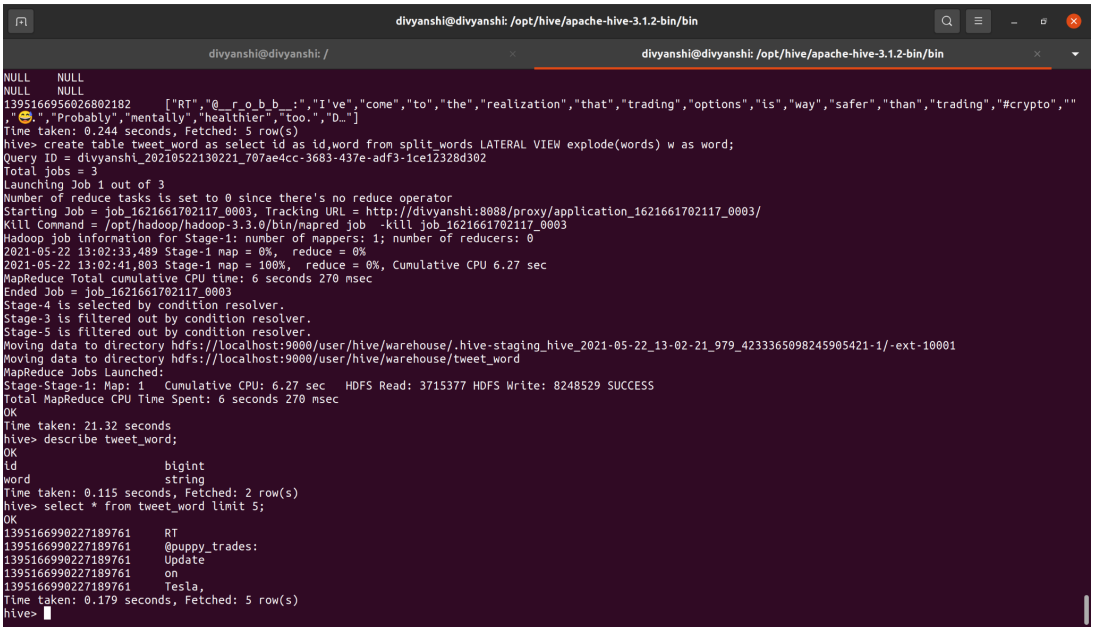


```
divyanshi@divyanshi: /opt/hive/apache-hive-3.1.2-bin/bin
divyanshi@divyanshi: /
divyanshi@divyanshi: /opt/hive/apache-hive-3.1.2-bin/bin
Hive Session ID = a1ee8113-c5ec-465e-a373-3a0bd3b9816d

Logging initialized using configuration in jar:file:/opt/hive/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.properties Async: true
Hive Session ID = 4e0bf358-95fa-4614-809a-4ff464a5d3c2
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive> CREATE EXTERNAL TABLE tweets_raw (
    > id BIGINT,
    > created_at STRING,
    > source STRING,
    > favorited BOOLEAN,
    > retweet_count INT,
    > retweeted_status STRUCT<
    > text:STRING,
    > `user`:STRUCT<screen_name:STRING,name:STRING>>,
    > entities STRUCT<
    > urls:ARRAY<STRUCT<expanded_url:STRING>>,
    > user_mentions:ARRAY<STRUCT<screen_name:STRING,name:STRING>>,
    > hashtags:ARRAY<STRUCT<text:STRING>>>,
    > text STRING,
    > `user` STRUCT<
    > screen_name:STRING,
    > name:STRING,
    > friends_count:INT,
    > followers_count:INT,
    > statuses_count:INT,
    > verified:BOOLEAN,
    > utc_offset:STRING, -- was INT but nulls are strings
    > time_zone:STRING>,
    > in_reply_to_screen_name STRING,
    > year int,
    > month int,
    > day int,
    > hour int
    > )
    > ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
    > LOCATION '/LSDP_Jcomp/twitter'
    > ;
OK
Time taken: 1.588 seconds
hive>
```

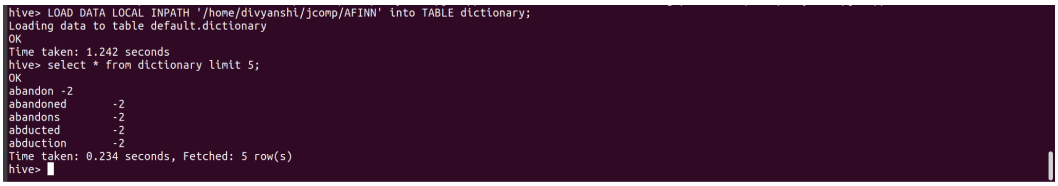
Sentiment Analysis:

We separated the word using split function on text data in tweets _raw table and saved into split_words table using below command. To perform sentiment analysis, we are using the AFINN dictionary which contains more than 2.5K words and values of polarity which range from -5.0 to +5.0 (Negative to Positive).

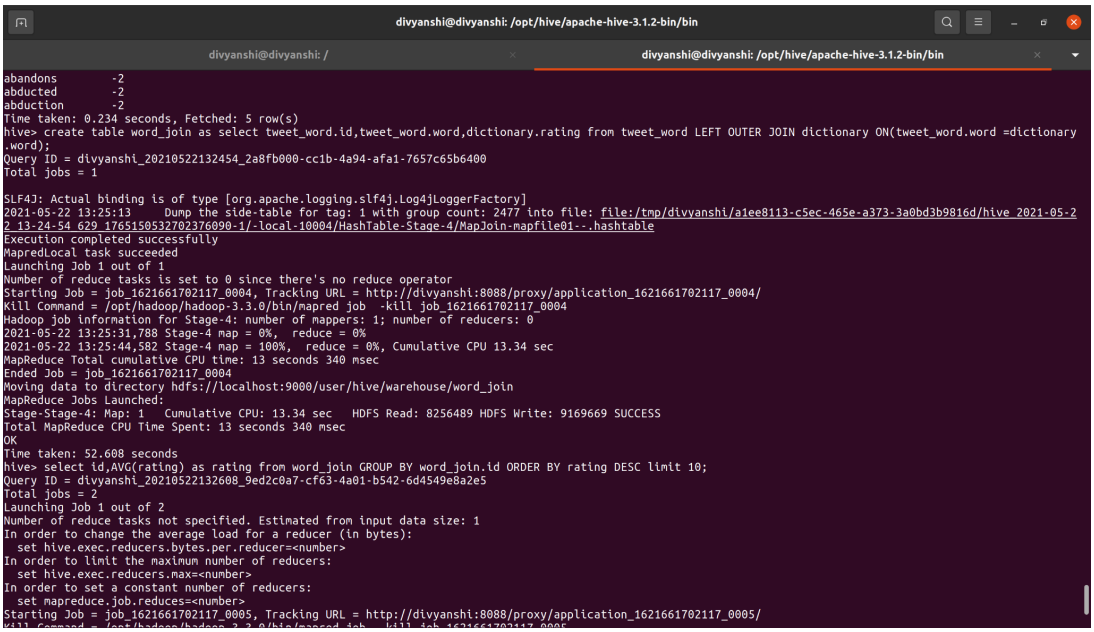
Now, we are splitting each word in the array as the new row. For this, we are using the explode function which extracts the element from array into new row using LATERAL VIEW. A table **tweet_word** is created.



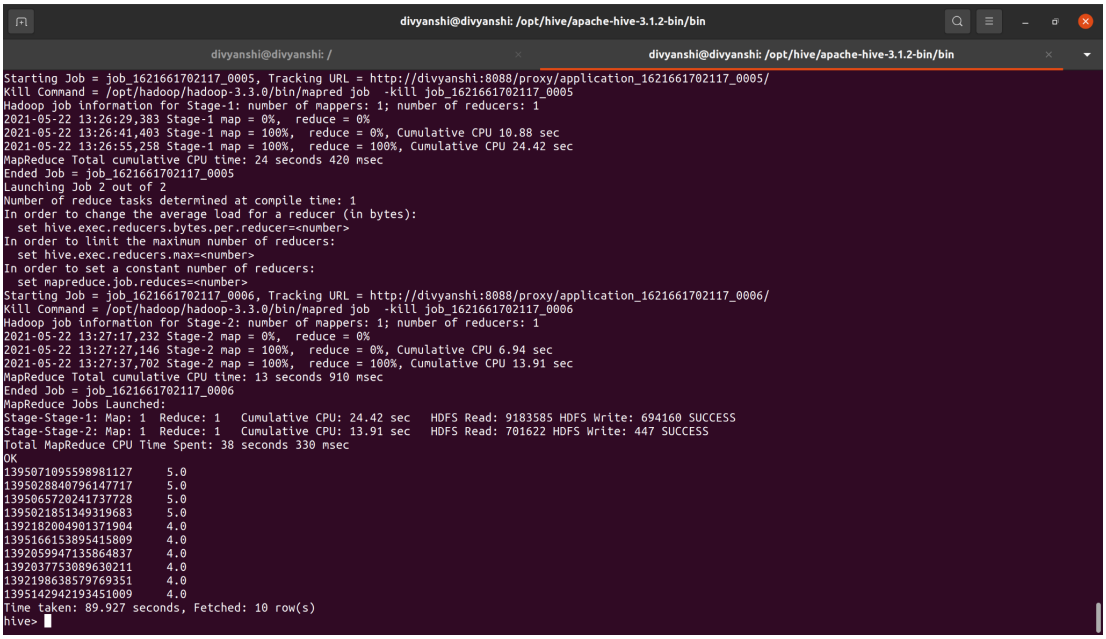
Now, we will create a new table named dictionary and will load the AFINN dictionary data into that table **dictionary.**



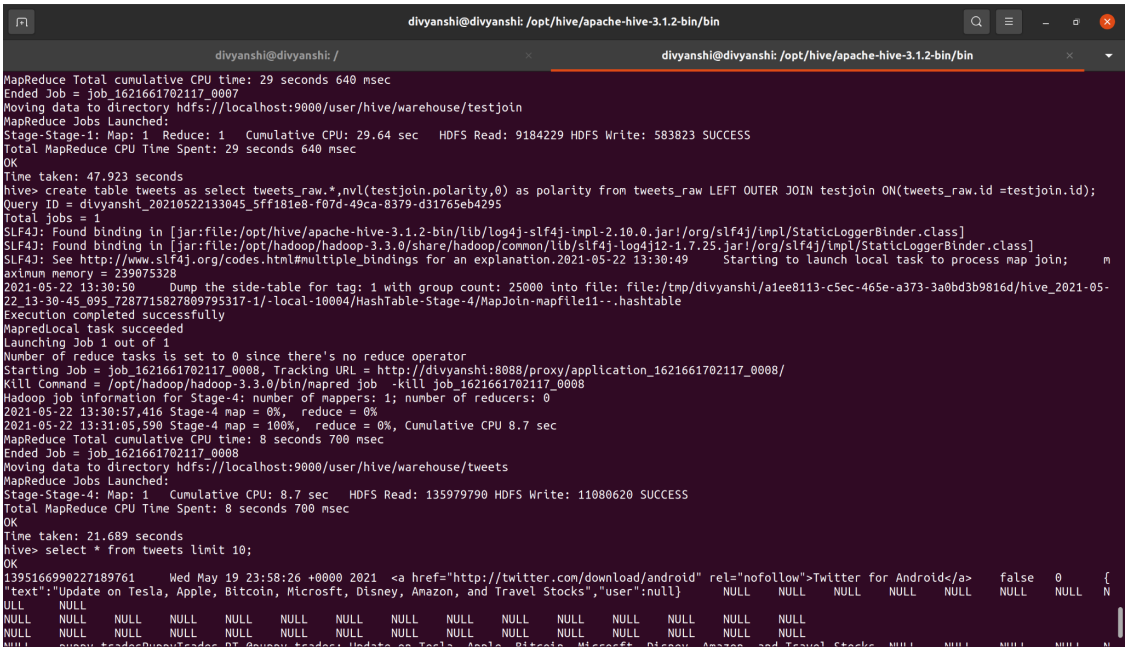
Create a new table word_join by joining the dictionary table with tweet_word table.

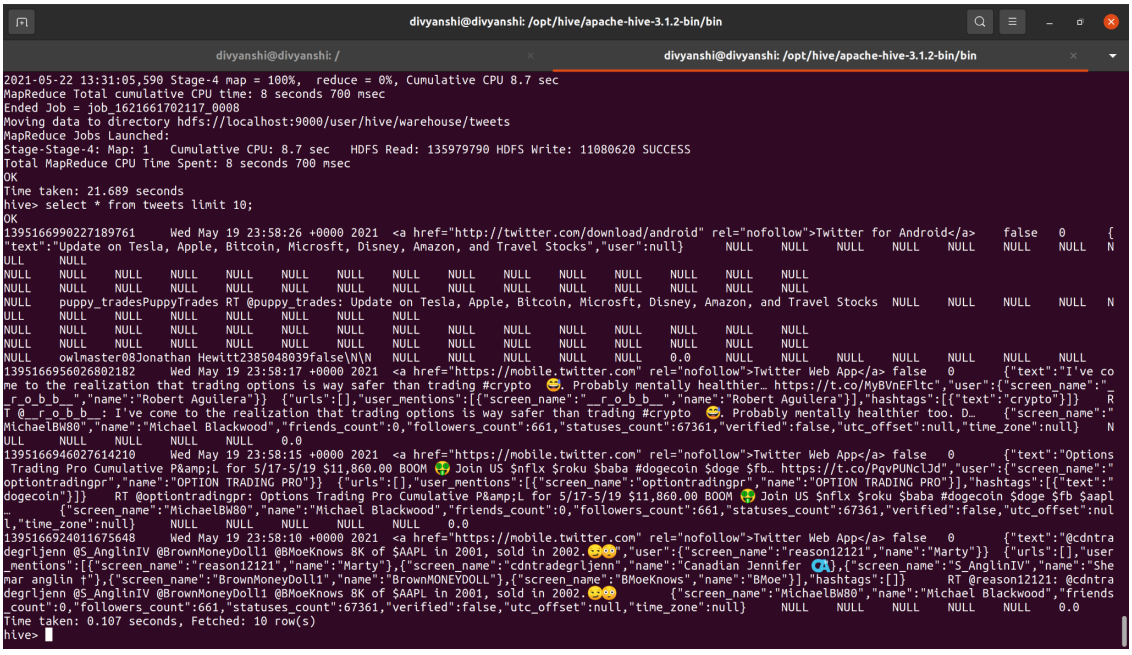


After joining now, we got the polarity value for each tweet in the word_join table.



Now, we are saving the same id based grouped data into the **testjoin** table and we are joining the tweets_raw table and test join table based on id and creating new table **tweets**.





How to use

1) Run the RulePayload.py file to extract all the tweets from a particular start date to end date and store them in tweets1.json file
2) Store this tweets1.json file in hdfs
3) Start hive
4) Add json serde dependency jar to hive. JSON SerDe is used to map the json object to the table schema.
5) Create tweets_raw table in hive.
6) Now we perform sentiment analysis using the AFINN dictionary.
7) We created a split_words table. We separated the words using split function on text data in tweets _raw table and saved into split_words table.
8) Now, we are splitting each word in the array as the new row. For this, we are using the explode function which extracts the element from the array into a new row. Since it has some limitations we are using a LATERAL VIEW. We are storing these words in a new table called tweet_words
9) Now, we will create a new table named dictionary and will load the AFINN dictionary data into that table.
10) Create a new table word_join by joining the ‘dictionary’ table and ‘tweet_word’ table.
11) Find average rating or polarity value of each tweet using ID attribute.
12) Save the ID based grouped data in a new table called ‘testjoin’.
13) Now join the tweets_raw table and testjoin table based on id and create a new table called ‘tweets’.
14) Now we can run sql queries in hive to analyse different sections of data.
15) Create another table know as ‘outtable’ by joining ‘tweets_raw’ table and ‘testjoin’ table
16) Export the data stored in ‘outdata’ table in a ‘out_data.csv’ file
17) Now we extract all the hashtags from the tweets by running th parse.py python script.
18) All hashtags will be stored in the hashtags.txt file.
19) Store the hashtags.txt file in hdfs.
20) Run the word count program to count each hashtag.
21) Store the result in a local json file called as ‘tweets1.json’.
22) Now we create data frames of out_data.csv and tweets1.json in spark.
23) Then we run the pyspark script to perform data analysis on the two data frames in spark and compare the execution time of same queries running in hive.
24) Visualize the polarities in the ‘out_data.csv’ file which tells us the percentage of positive, negative and neutral tweets.
25) Plot the graph of time vs avg. polarity.

Dataset Description

Dataset:

tweets1.json: it contains the data related to stock market extracted from the Twitter API

Tables and HDFS files:

- tweets_raw : with the few required attributes from the tweet object.

- split_word : separated the word using split function

- tweet_word : splitting each word in the array as the new row.

- dictionary : load the AFINN dictionary data.

- word_join : by joining the dictionary table with tweet_word table.

- testjoin : the same id based grouped data.

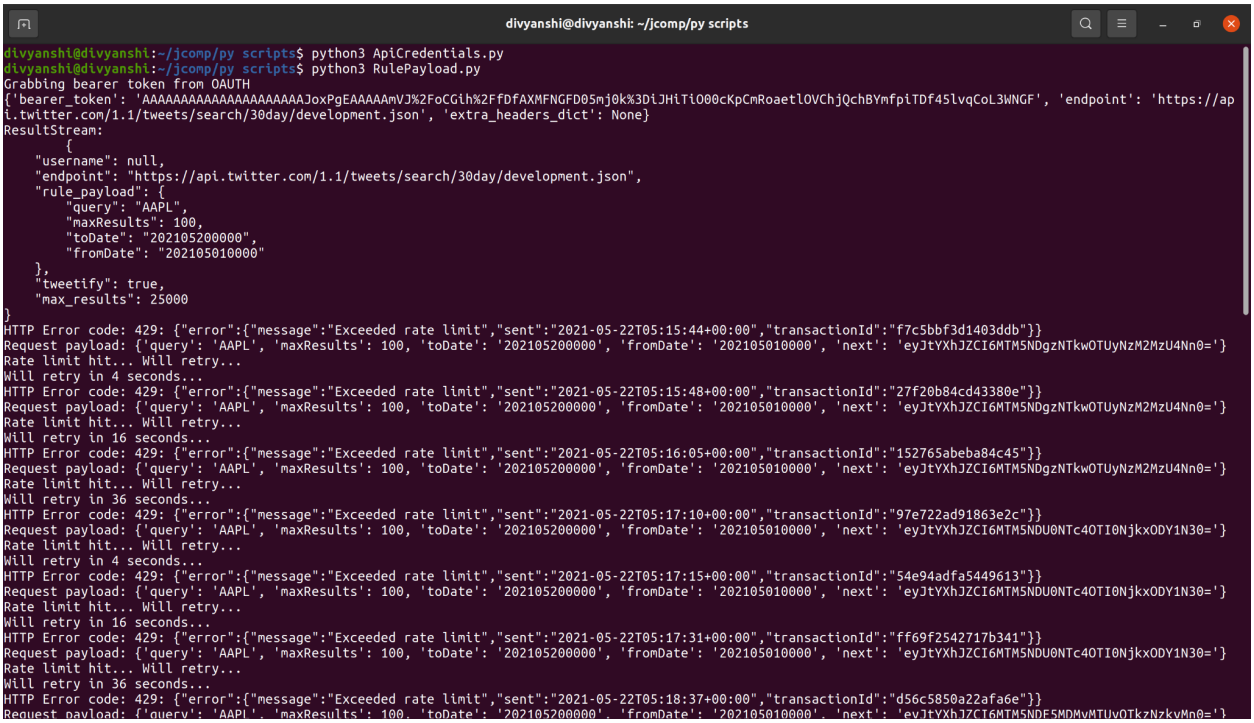
- tweets : joining the tweets_raw table and test join table based on id.

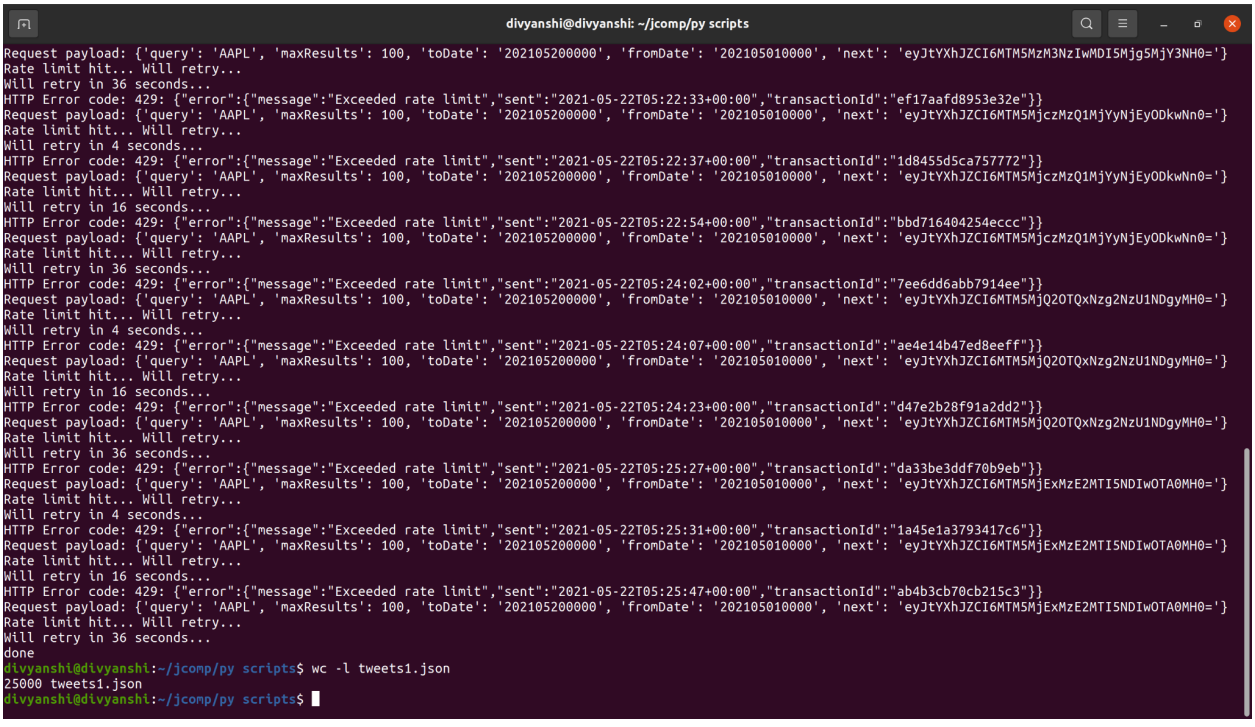
hashtags.txt : file containing the extracted hashtags.

Link to dataset and hdfs files:
https://drive.google.com/drive/folders/1bGtk24KlJQzVMigj7qEFAj4mGK0_rIom?usp=sharing

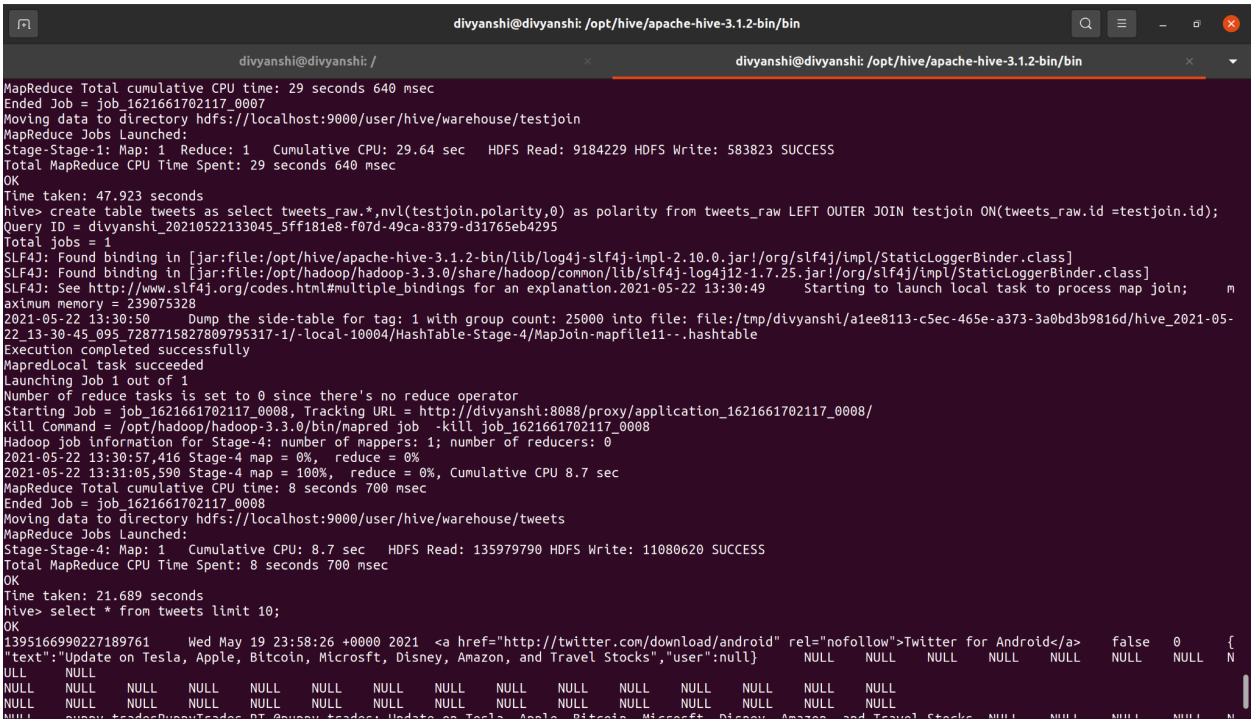
Results

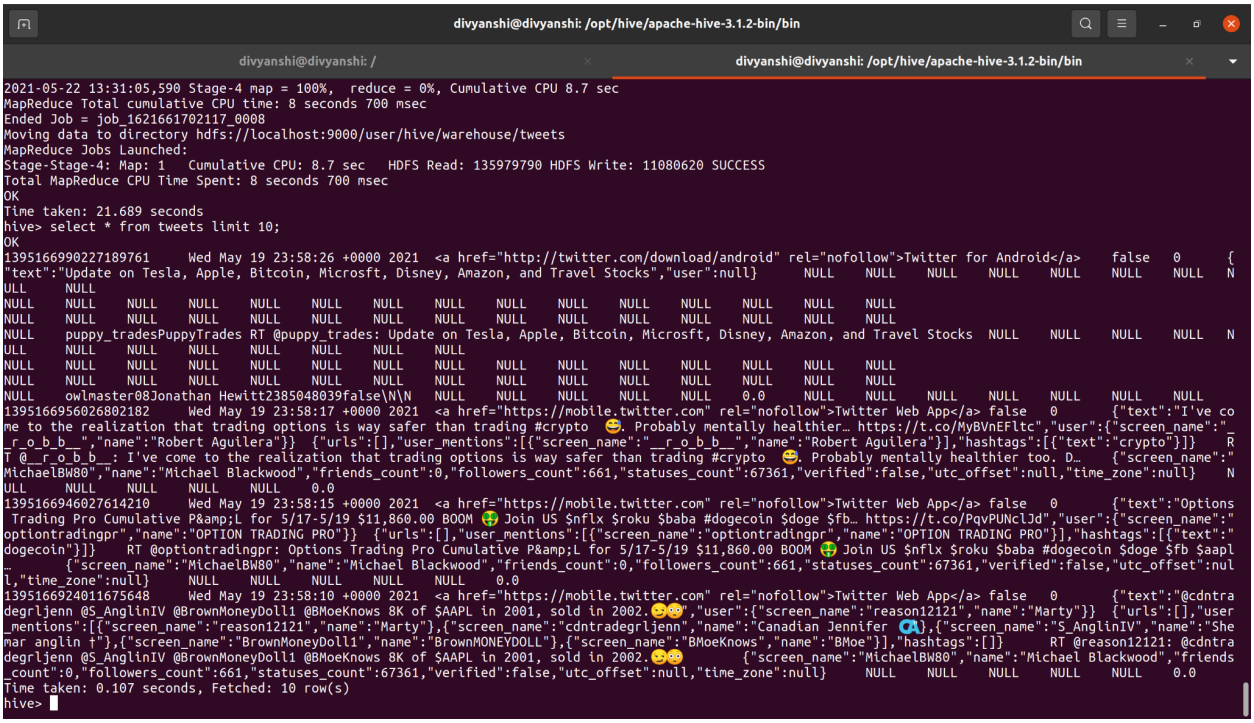
Tweets extraction



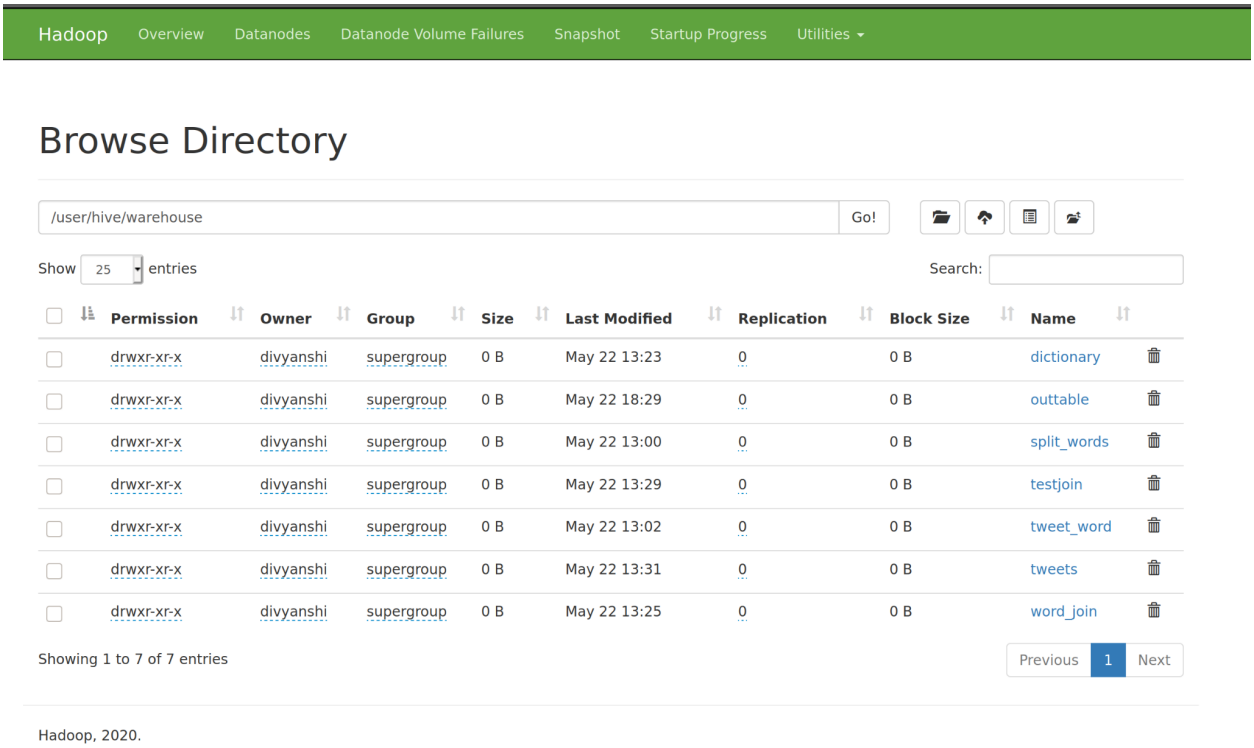


Final table (tweets) after sentiment analysis (with polarity of every tweet)

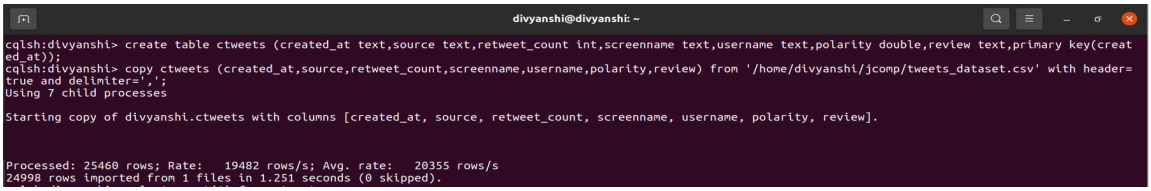


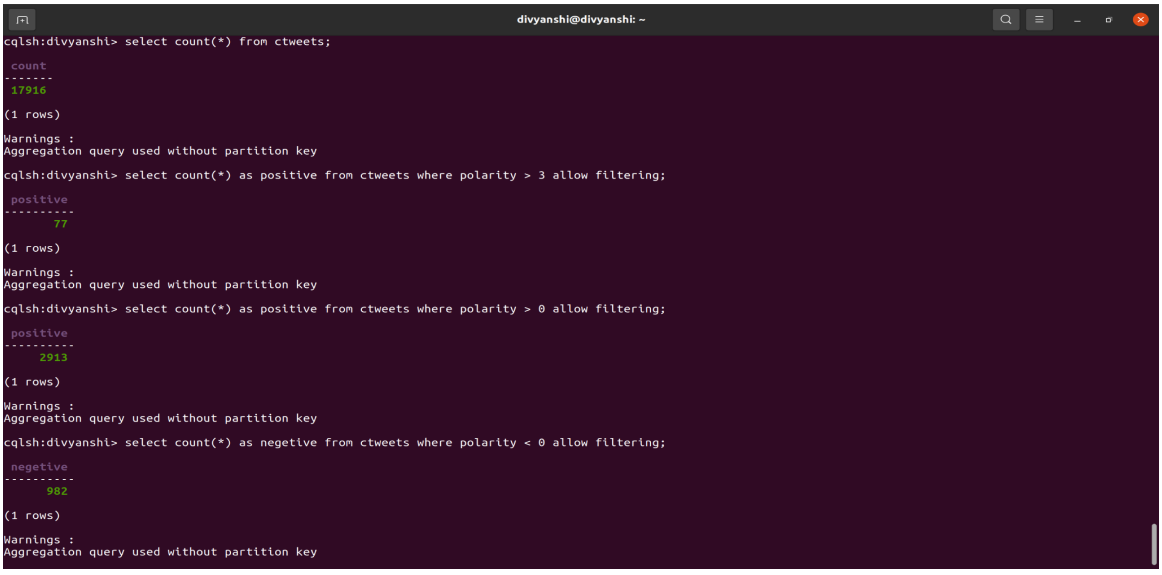


Hive tables



Cassandra Analysis

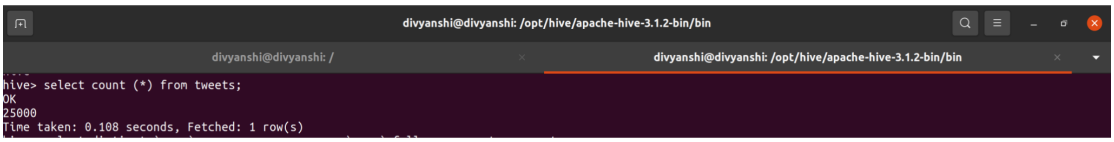




Tools Comparison (Hive vs SparkSQL)

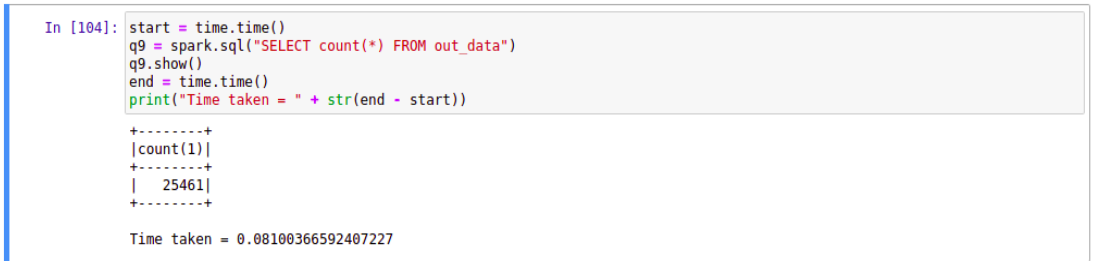
- **Query: select count(*) from tweets;**

Hive



Time Taken: 0.108 seconds

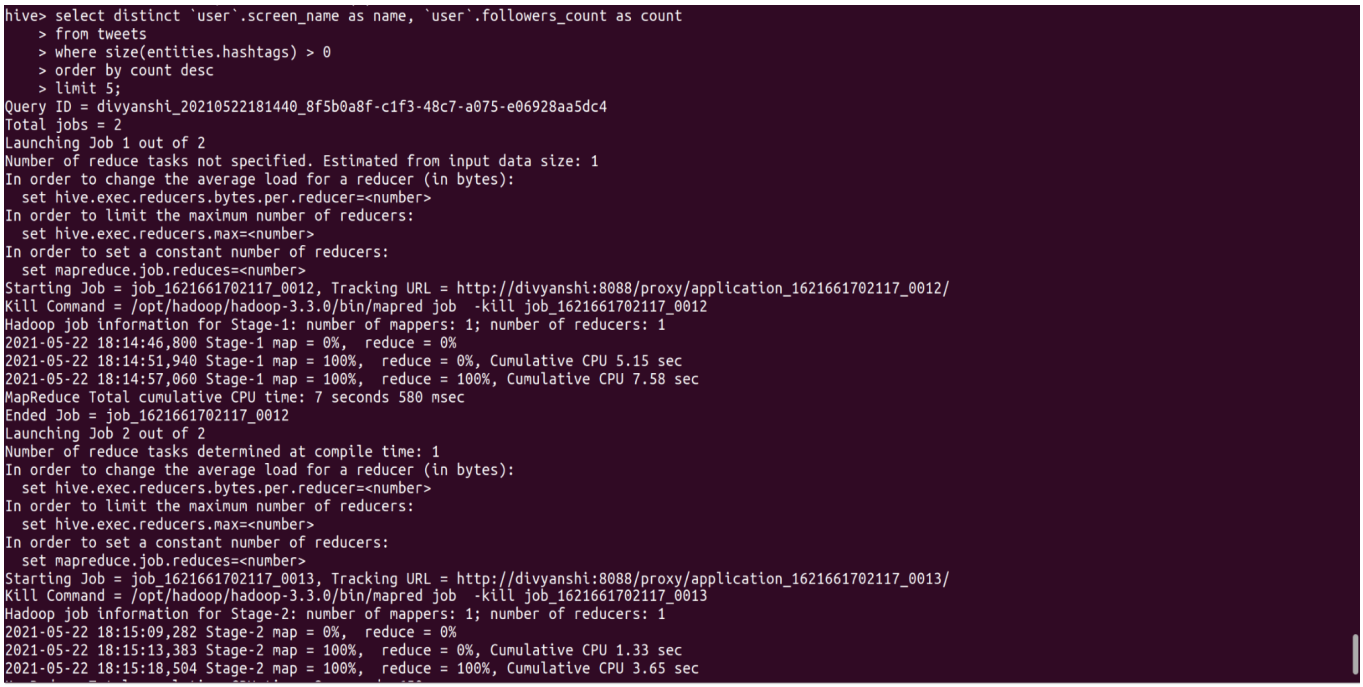
SparkSQL

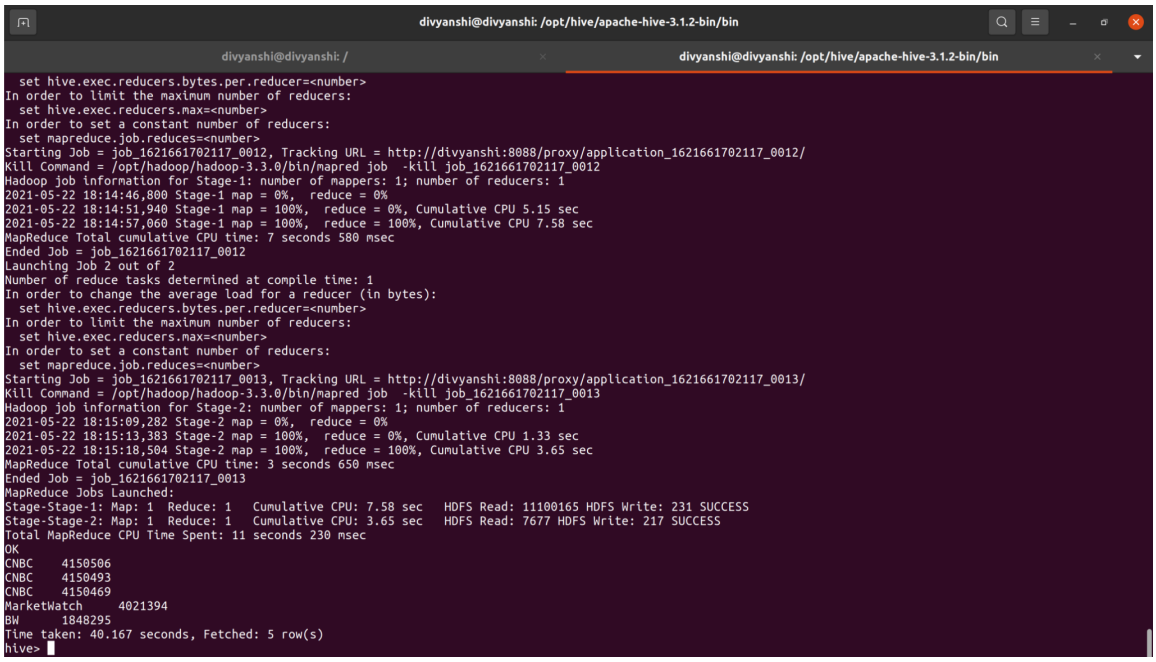


Time Taken: 0.08100366592407227 seconds

- **Query :select distinct `user`.screen_name as name, `user`.followers_count as count from tweets where size(entities.hashtags) > 0 order by count desc limit 5;**

Hive





Time taken: 40.167 seconds

SparkSQL

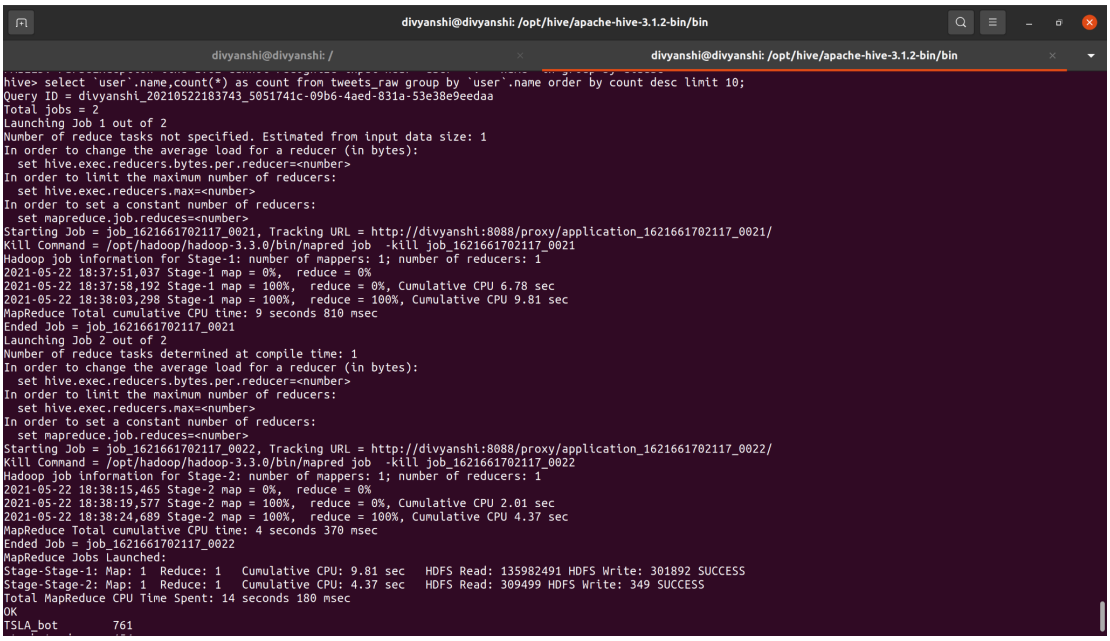
```
In [21]: start = time.time()
q3 = spark.sql("SELECT user.name, max(user.followers_count) as followers_count FROM tweets1 WHERE text like '%AAPL%'
q3.show()
end = time.time()
print("Time taken = " + str(end - start))
```

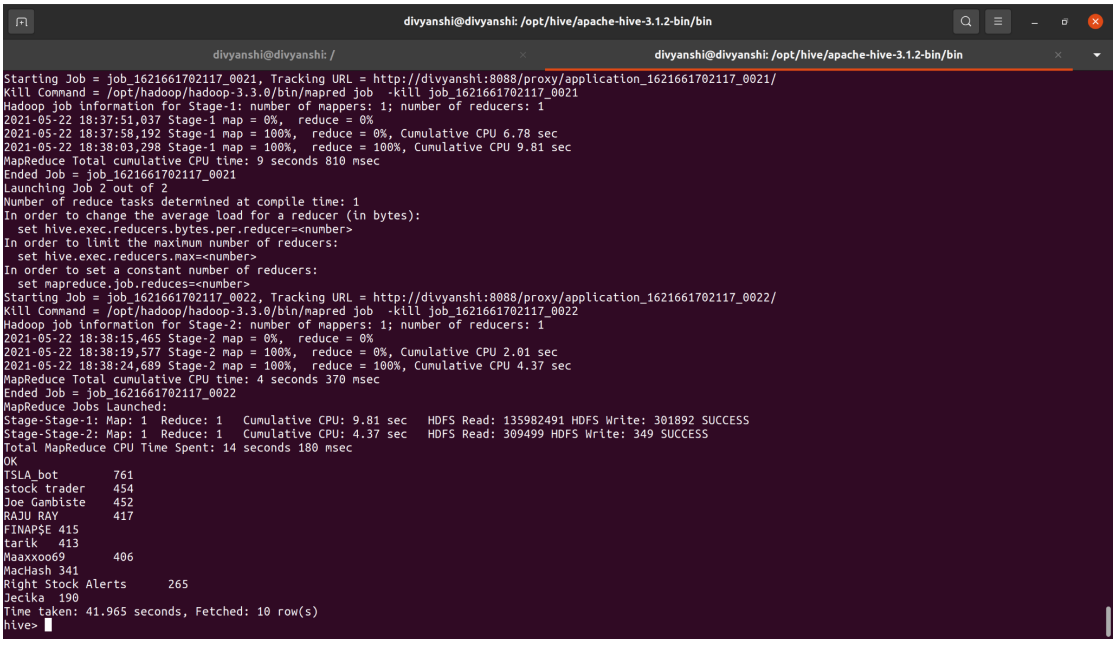
```
+----------------+---------------+
|            name|followers_count|
+----------------+---------------+
|            CNBC|        4150493|
|     MarketWatch|        4021394|
|       TheStreet|         782944|
| Charles V Payne|         556900|
|     Steve Burns|         363566|
|Carl Quintanilla|         300922|
|   Hank Lockwood|         274053|
|    Jon Najarian|         262431|
|   Investors.com|         252172|
|    Amit Bhawani|         249170|
| Cheddar News 🧀|         249027|
|    Ramp Capital|         243000|
|   FXStreet News|         204190|
|      LiveSquawk|         194892|
| Morning Brew ☕|         187974|
+----------------+---------------+

Time taken = 1.4111835956573486
```

Time Taken: 1.4111835956573486 seconds

- **Query: select `user`.name,count(*) as count from tweets_raw group by `user`.name order by count desc limit 10;**

Hive





Time Taken: 41.965 seconds

SparkSQL

```
In [22]: start = time.time()
q5 = spark.sql("SELECT count(*) as count, user.name from tweets1 where user.name is not null group by user.name orde
q5.show()
end = time.time()
print("Time taken = " + str(end - start))
```

```
+-----+-----------------+
|count|             name|
+-----+-----------------+
|  761|         TSLA_bot|
|  454|      stock trader|
|  452|      Joe Gambiste|
|  417|          RAJU RAY|
|  415|           FINAP$E|
|  413|            tarik|
|  406|        Maaxxoo69|
|  341|          MacHash|
|  265|Right Stock Alerts|
|  190|           Jecika|
+-----+-----------------+

Time taken = 1.505272388458252
```

Time Taken: 1.505272388458252

- **Query: select `user`.name,retweet_count from tweets_raw where retweet_count in (select max(retweet_count) from tweets);**

Hive

```
divyanshi@divyanshi: /opt/hive/apache-hive-3.1.2-bin/bin
divyanshi@divyanshi: /   divyanshi@divyanshi: /opt/hive/apache-hive-3.1.2-bin/bin
Total MapReduce CPU Time Spent: 3 seconds 80 msec
OK
Time taken: 13.037 seconds
hive> select `user`.name,retweet_count from tweets_raw where retweet_count in (select max(retweet_count) from tweets);
Query ID = divyanshi_20210522183413_6ee6d0e5-8d68-4bb5-816e-3ab60836f977
Total jobs = 4
Launching Job 1 out of 4
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1621661702117_0019, Tracking URL = http://divyanshi:8088/proxy/application_1621661702117_0019/
Kill Command = /opt/hadoop/hadoop-3.3.0/bin/mapred job  -kill job_1621661702117_0019
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2021-05-22 18:34:22,390 Stage-2 map = 0%,  reduce = 0%
2021-05-22 18:34:26,515 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.27 sec
2021-05-22 18:34:31,619 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.37 sec
MapReduce Total cumulative CPU time: 5 seconds 370 msec
Ended Job = job_1621661702117_0019
Stage-6 is selected by condition resolver.
Stage-7 is filtered out by condition resolver.
Stage-1 is filtered out by condition resolver.


SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2021-05-22 18:34:37     Uploaded 1 File to: file:/tmp/divyanshi/a1ee8113-c5ec-465e-a373-3a0bd3b9816d/hive_2021-05-22_18-34-13_519_2948164305926656225-1/-local-1
0005/HashTable-Stage-3/MapJoin-mapfile31---.hashtable (280 bytes)
2021-05-22 18:34:37     End of local task; Time Taken: 0.987 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 3 out of 4
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1621661702117_0020, Tracking URL = http://divyanshi:8088/proxy/application_1621661702117_0020/
Kill Command = /opt/hadoop/hadoop-3.3.0/bin/mapred job  -kill job_1621661702117_0020
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2021-05-22 18:34:46,753 Stage-3 map = 0%,  reduce = 0%
2021-05-22 18:34:53,915 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 7.47 sec
```

```
MapReduce Total cumulative CPU time: 7 seconds 470 msec
Ended Job = job_1621661702117_0020
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.37 sec   HDFS Read: 11098591 HDFS Write: 116 SUCCESS
Stage-Stage-3: Map: 1   Cumulative CPU: 7.47 sec   HDFS Read: 135978774 HDFS Write: 113 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 840 msec
OK
Raoul Pal       623
Time taken: 41.486 seconds, Fetched: 1 row(s)
hive>
```

Time Taken: 41.486 seconds

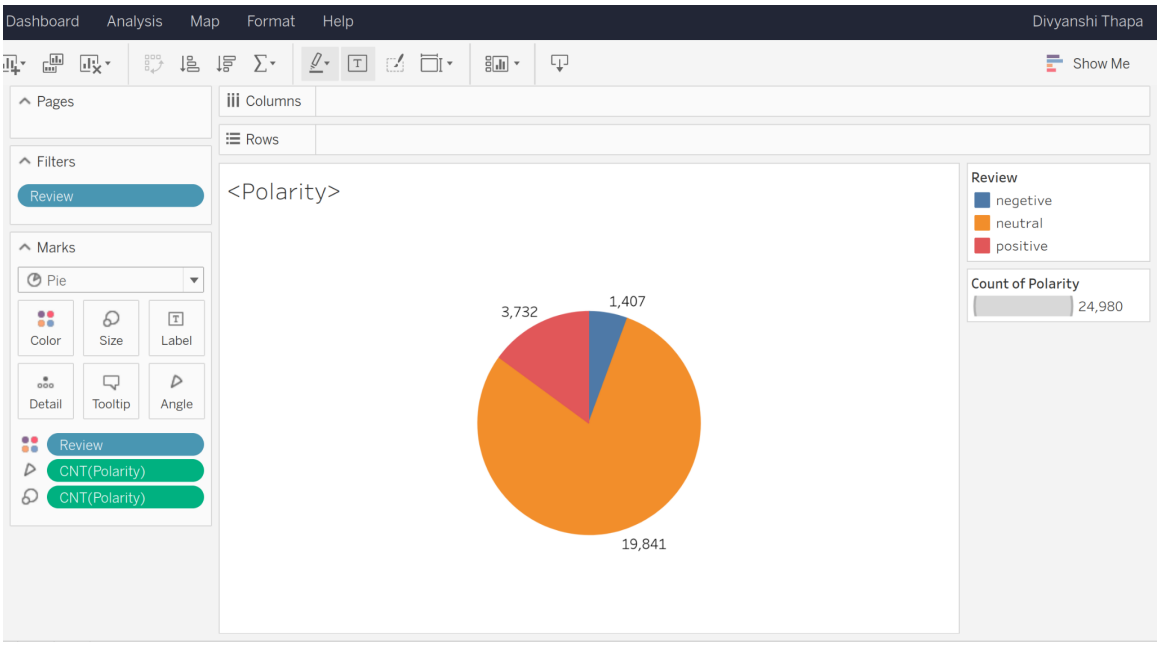
SparkSQL

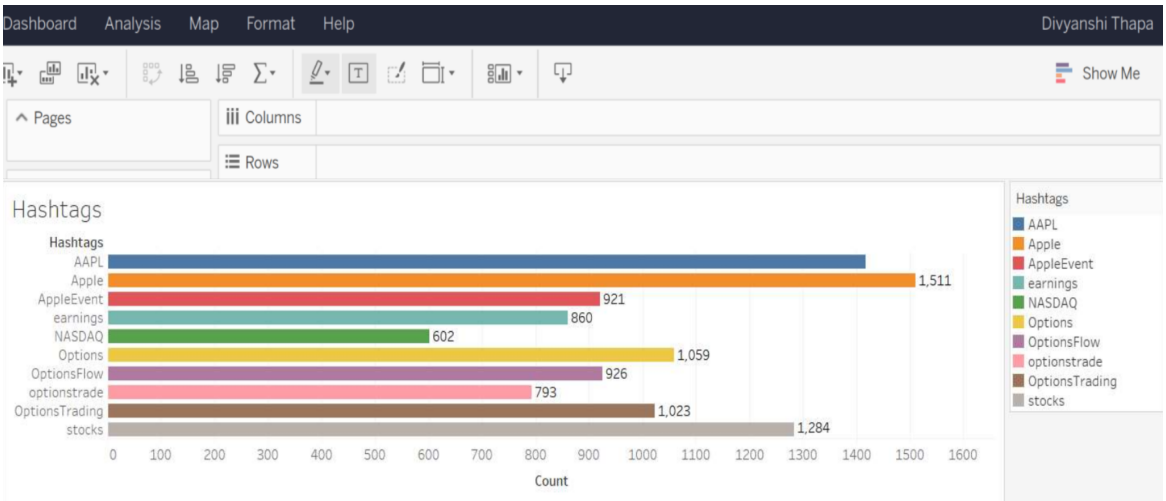
```
In [24]: ime.time()
         k.sql("select user.name,retweet_count from tweets1 where retweet_count in (select max(retweet_count) from tweets1);"

         e.time()
         me taken = " + str(end - start))
```

```
+---------+-------------+
|     name|retweet_count|
+---------+-------------+
|Raoul Pal|          623|
+---------+-------------+

Time taken = 1.3617329597473145
```

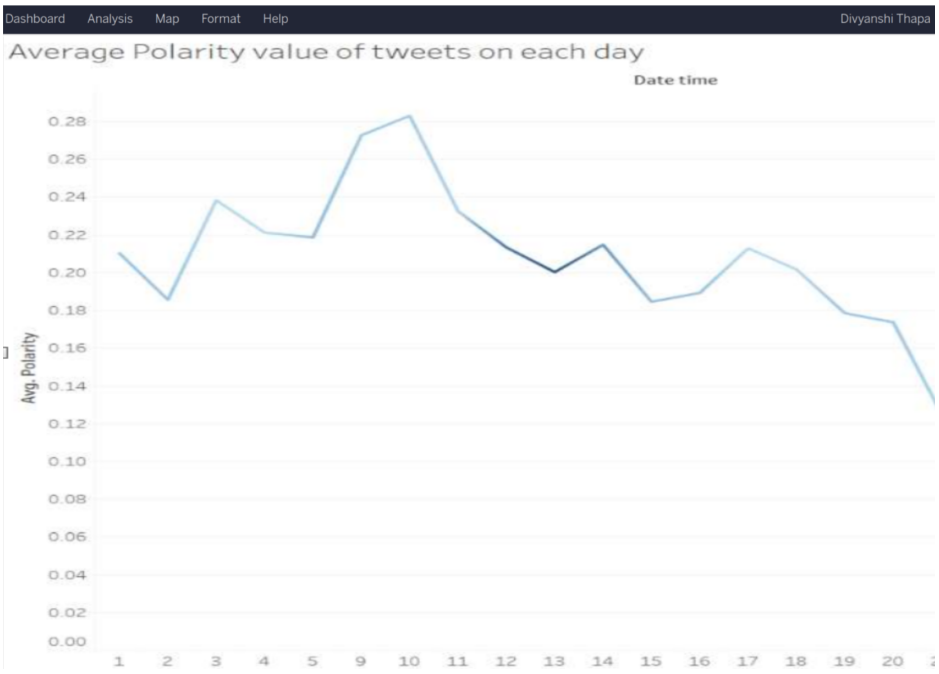
Time Taken: 1.3617329597473145 seconds

Visualization

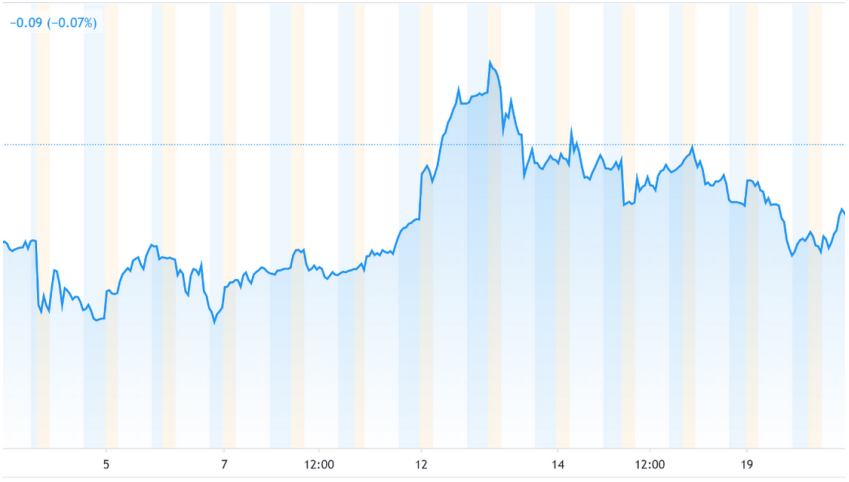




Our predicted trend



Original trend



Conclusion

In general, the task of stock market prediction is quite challenging, as the data keeps on changing. Data Analysis and Big Data are on the cusp of completely revolutionising how the stock markets will function. Using this system, we successfully analyzed the data extracted from the twitter social opinions based on stock ticker with the help of hadoop ecosystem. In the above results, we are able to get the trend after analysis which can be used for predictions of movement of stock. This trend might help for the investors in future investments. Along with prediction we also compared the efficiency of different big data tools and deduced the best among them. In our case, Apache SparkSQL performed better than Hive when it came to querying big data. This comparative analysis can be of use for other big data prediction models also.

References

1) https://www.ijert.org/big-data-analysis-in-stock-market-prediction#:~:text=Abstract%20Big%20data%20analytics%20can,data%2C%20which%20is%20hidden%20otherwise.
2) https://www.smartdatacollective.com/big-data-analytics-has-potential-to-massively-disrupt-stock-market/
3) https://ieeexplore.ieee.org/document/8669649
4) https://github.com/topics/twitter-sentiment-analysis
5) https://www.intechopen.com/books/e-business-higher-education-and-intelligence-applications/recent-advances-in-stock-market-prediction-using-text-mining-a-survey
6) https://link.springer.com/article/10.1007/s12652-020-01839-w