PROJECT REPORT

TECHNICAL, FUNDAMENTAL AND SENTIMENTAL ANALYSIS OF STOCK MARKET

(Project Semester January 2018 to June 2018)

Submitted by:

Sharang Gupta

Aishwarya Dixit

Aayush Pant

Under the Guidance of

Faculty Mentor:

Mrs Pooja Kamat
Assistant Professor

Panel Member:

Mrs. Shruti Patil

Assistant Professor

Department of Computer Science and Information Technology

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

List of Diagrams

DECLARATION

I hereby declare that the project work entitled - Warren Stock Analyser is an authentic record of our own work carried out at as requirements of six months project semester for the award of B.Tech degree in Computer Science Engineering at Symbiosis. Institute of Technology Pune, affiliated to Symbiosis International University Pune, under the guidance of Prof.Pooja Kamat, during January 2018 to June 2018.

Sharang Gupta	Aishwarya Dixit	Ayush Pant
14070121116	14070121502	14070121512

Date:

Certified that the above statement made by the student is correct to the best of our knowledge and belief.

Prof. Shraddha Phansalkar

HoD CS and IT Department,

SIT, PUNE

Prof. Pooja Kamat

Assistant Professor

SIT, PUNE

ACKNOWLEDGEMENT

We owe our profound gratitude and special thanks to Prof. Pooja Kamat who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep us on the correct path by his untiring assistance, direction, encouragement, comments, suggestions, continuous guidance, support, ideas and constructive criticism throughout this project work.

We would also like to thank Professor Shruti Patil for assistance on our project and for suggestions that greatly improved the quality of our project.

We would like to thank our HOD, Prof. Shraddha Phansalkar for her valuable guidance, supervising this work and helpful suggestions.

We heartily thank our project coordinator, Prof. Prachi Jagtap for cooperating with us and giving us her valuable time and information.

We are immensely grateful to our organisation, Symbiosis Institute of Technology for giving us the opportunity, platform and the resources to work.

INDEX

ABSTRACT

THREE PRONG ANALYSIS OF STOCK MARKET DATA

Over time, the stock market tends to rise in value, though the prices of individual stocks rise and fall daily. Share prices change because of supply and demand. With an increase in the demand (buying) the prices rise. The investors try to read such patterns and invest accordingly. A number of factors may be responsible for the continuous fall or rise of share prices of a company and our project is aimed at analysing the patterns, financials and the general sentiment of at least 500 companies in the past 2 decades, to assist investment decisions by carrying out Technical, Fundamental and Sentimental Analysis.

1. LITERATURE SURVEY

The research phase is integral for the success of any project. The capabilities and strengths of a project depend on how strong the research is. We devoted 40% of our time towards research on various machine learning algorithms, scraping and cleaning techniques and various APIs.

- **Stock Market Prediction - Mark Dunne :** In this report we analyse existing and new methods of stock market prediction. We take three different approaches at the problem: Fundamental analysis, Technical Analysis, and the application of Machine Learning. We find evidence in support of the weak form of the Efficient Market Hypothesis, that the historic price does not contain useful information but out of sample data may be predictive. We show that Fundamental Analysis and Machine Learning could be used to guide an investor's decisions. We demonstrate a common flaw in Technical Analysis methodology and show that it produces limited useful information. Based on our findings, algorithmic trading programs are developed and simulated using Quantopian.
- **Sentiment analysis for stock exchange prediction - Milson L. Lima, Thiago P. Nascimento, Sofiane Labidi, Nadson S. Timbó, Marcos V. L. Batista, Gilberto N. Neto, Eraldo A. M. Costa and Sonia R. S. Sousa :** The economic growth is a consensus in any country. To grow economically, it is necessary to channel the revenues for investment. One way of raising is the capital market and the stock exchanges. In this context, predicting the behavior of shares in the stock exchange is not a simple task, as it involves variables not always known and can undergo various influences, from the collective emotion to high-profile news. Such volatility can represent considerable financial losses for investors. In order to anticipate such changes in the market, it has been proposed various mechanisms trying to predict the behavior of an asset in the stock market, based on previously existing information.Such mechanisms include statistical data only, without considering the collective feeling. This paper is going to use natural language processing algorithms (LPN) to determine the collective mood on assets and later with the help of the SVM algorithm to extract patterns in an attempt to predict the active behaviour.

There have been a lot of efforts made analyzing and predicting the stock market. The stock market is a complicated institution and no individual factor can do justice to an accurate prediction. Each of these individual analysis concentrate on very specific objects and hence, to get a more complete picture we need to combine these different factors and analyse them to produce a consolidated and improved results.

2. PROBLEM DEFINITION

2.1 OVERVIEW

Stocks across the globe have consistently earned more than bonds over the long term, despite regular ups and downs in the market. That is why investing in stocks, stock mutual funds, or ETFs, is important when saving for retirement or other far-off goals.

However, Predicting the Stock Market has been the bane and goal of investors since its existence. Everyday billions of dollars are traded on the exchange, and behind each dollar is an investor hoping to profit in one way or another. Entire companies rise and fall daily based on the behaviour of the market. Should an investor be able to accurately predict market movements, it offers a tantalizing promises of wealth and influence. It is no wonder then that the Stock Market and its associated challenges find their way into the public imagination every time it misbehaves

2.2 PURPOSE:

This particular project appealed to all of us for the sole reason of the magnitude of the data involved, and the potential of the various types of analysis associated. This project also pushes us beyond our limits, allowing us to explore new technologies and improve our knowledge base and experience tremendously. This project offered us opportunities to work with machine learning, sentiment analysis and data harvesting, which intrigues us all.

2.3 PROBLEM STATEMENT

Despite its prevalence, Stock Market prediction remains a secretive and empirical art. Few people, if any, are willing to share what successful strategies they have. A chief goal of this project is to add to the academic understanding of stock market prediction

The main objective of our project lies with the analysis of a stock as chosen by the ticker (the stock symbol of a company used to uniquely identify publicly traded shares in the stock market) provided.

Based on our analysis we will decide whether the stock is bullish (stock outperforms, i.e., the prices rise consistently in the market) or bearish (stock underperforms, i.e., the prices fall consistently in the market).

The bulk of our efforts will be directed towards gathering, cleaning and preparing the data, along with providing an overlying API.

The ultimate result after the analysis will help the investors decide whether they should be buying or selling the stocks of a particular company, of which the ticker is provided.

2.4 CONTRIBUTION

The main contribution of our project towards society and towards the technology is that it is equipping it with an analytics tool is that we are demonstrating how the power of Fundamental analysis, technical analysis and Natural Language Processing, Sentiment Analysis and Social Networking can be integrated together for the benefits of the general public.

2.5 FEATURES OF THE PROJECT

- User can enter a text containing the ticker of a company.
- User can then view the results of the three analysis based upon the ticker provided.
- User can view the combined results of the three analysis.

3. PLATFORM / TECHNOLOGY

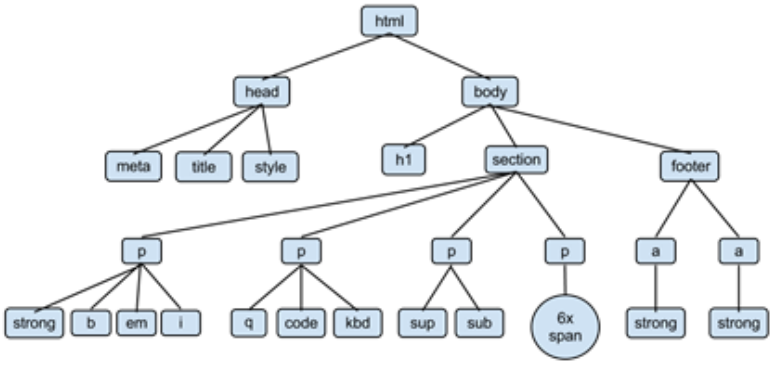
- **LANGUAGE : PYTHON 2.7**

MODULES :

- **PANDAS**

- **SKLEARN**

- **BEAUTIFULSOUP**



- **TWEEPY**

- **PYSENTIMENT**
- **REQUESTS**

4. SRS (SOFTWARE REQUIREMENT SPECIFICATION)

4.1. INTRODUCTION

This part of the document is a comprehensive description of the intended purpose and environment for software developed. This fully describes what the software will do and how it is expected to perform. This document also enlists enough and necessary requirements that are required for the project. This is a description of the behavior of a system and includes a set of use cases that describe interactions the users will have with the software. In addition this also contains nonfunctional requirements. Non-functional requirements impose constraints on the design or implementation such as performance engineering requirements, quality standards, or design constraints.

4.2. SYSTEM REQUIREMENTS

- PYTHON IDLE(SERVER)
- ALL MODULES INSTALLED(SERVER)
- INTERNET CONNECTIVITY(CLIENT AND SERVER)

4.3 EXTERNAL INTERFACE REQUIREMENTS

➔ **User Interfaces:**

1. **Introduction Page**: This Page will contain information about the functionality of the API. It will also have information about the team members, github link and the link to the main page.
2. **Analysis Page:** This is the key page of the API which contains a form in which user enters the ticker and receives the result.
3. **Modal**: This is another feature available in the API which displays the output or the results in the form of a modal.

➔ **Hardware Interface:**

1. Computer System
2. Server(absent in the demo)

➔ **Communication Interfaces:**

1. Internet Connectivity

➔ **Software Interfaces:**

1. Linux Operating System(Ubuntu OS)
2. Web Browser
3. Django API (web framework)

➔ **Prerequisite Softwares**

1. Python
2. Pip package
3. Web Browser(present by default)
4. Virtual Environment

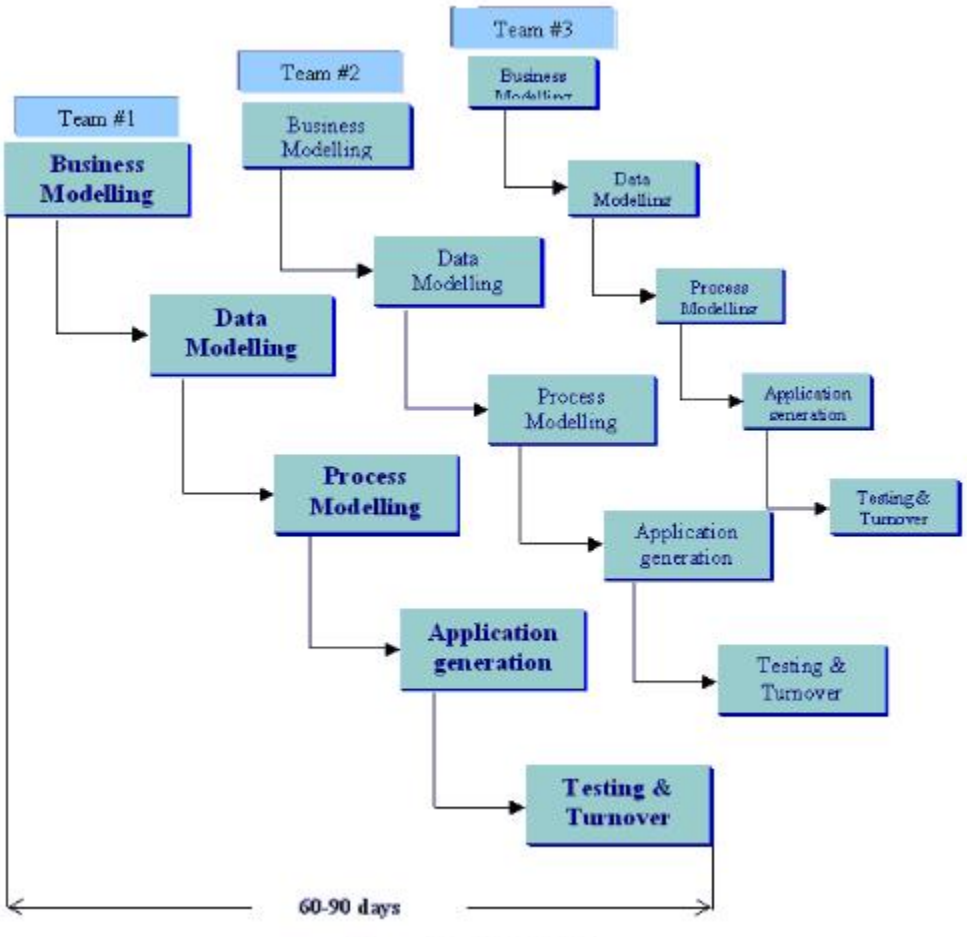
5 SOFTWARE PROJECT PLAN

5.1 SOFTWARE DEVELOPMENT PLAN

We selected the SDLC Rapid Application Development (RAD) Model while developing this project

RAD MODEL

RAD model is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.



Steps

Stage 1: Business Modeling

This step in the RAD model takes information gathered through many business related sources. The analysis takes all the pertinent information from the company. This info is then combined into a useful description of how the information can be used, when it is processed, and what is making this specific information successful for the industry.

Stage 2: Data Modeling

During the Data Modeling stage, all the information gathered during the Business Modeling phase is analyzed. Through the analysis, the information is grouped together into different groups that can be useful to the company. The quality of every group of information is carefully examined and given an accurate description. A relationship between these groups and their usefulness as defined in the Business Modeling step is also established during this phase of the RAD model.

Stage 3: Process Modeling

The Process Modeling phase is the step in the RAD model procedure where all the groups of information gathered during the Data Modeling step are converted into the required usable information. During the Process Modeling stage changes and optimizations can be done and the sets of data can be further defined. Any descriptions for adding, removing, or changing the data objects are also created during this phase.

Stage 4: Application Generation

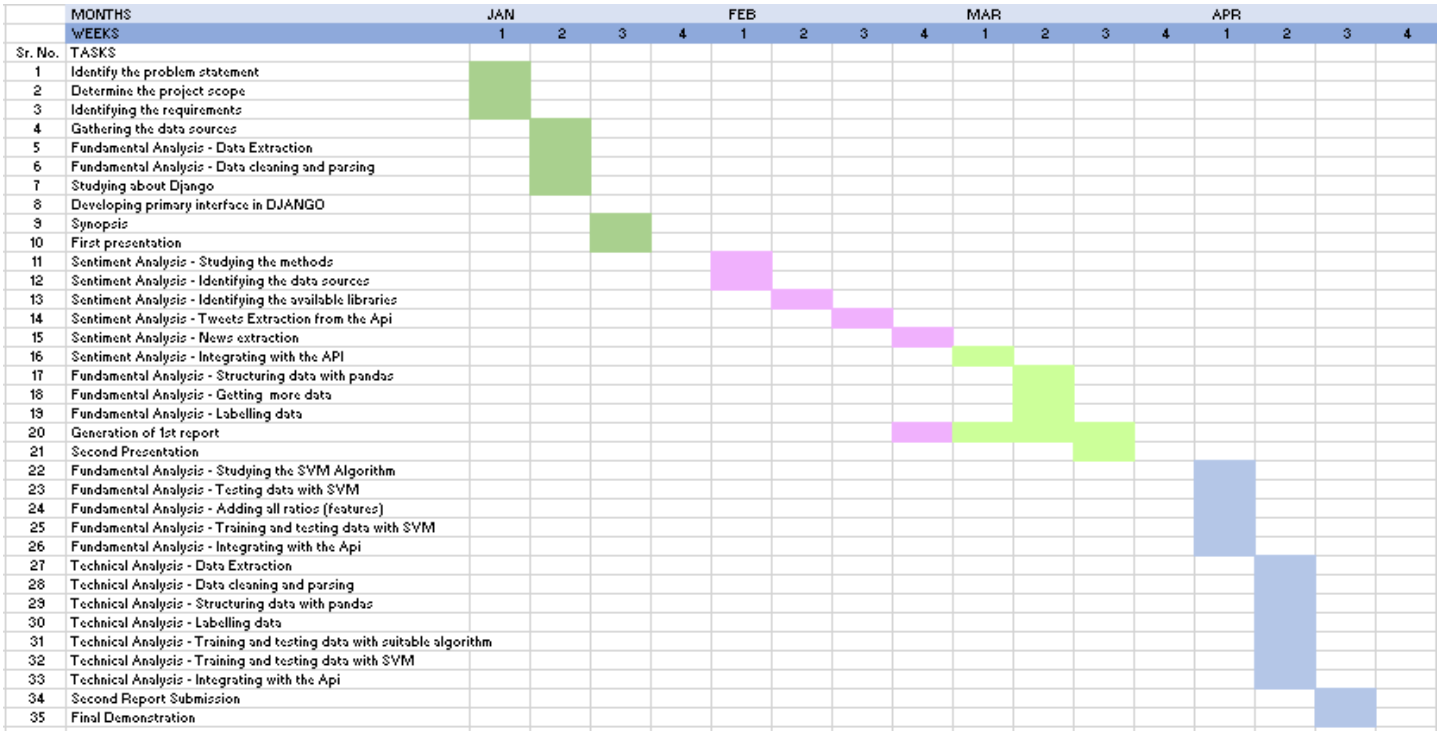
The Application Generation step is when all the information gathered is coded, and the system that is going to be used to create the prototype is built. The data models created are turned into actual prototypes that can be tested in the next step.

Stage 5: Testing and Turnover

The Testing and Turnover stage allows for a reduced time in the overall testing of the prototypes created. Every model is tested individually so that components can quickly be identified and

switched in order to create the most effective product. By this point in the RAD model, most of the components have already been examined, so major problems with the prototype are not likely.

5.2 PROJECT TIMELINE



	MONTHS	JAN				FEB				MAR				APR			
	WEEKS	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Sr. No.	TASKS																
1	Identify the problem statement																
2	Determine the project scope																
3	Identifying the requirements																
4	Gathering the data sources																
5	Fundamental Analysis - Data Extraction																
6	Fundamental Analysis - Data cleaning and parsing																
7	Studying about Django																
8	Developing primary interface in DJANGO																
9	Synopsis																
10	First presentation																
11	Sentiment Analysis - Studying the methods																
12	Sentiment Analysis - Identifying the data sources																
13	Sentiment Analysis - Identifying the available libraries																
14	Sentiment Analysis - Tweets Extraction from the Api																
15	Sentiment Analysis - News extraction																
16	Sentiment Analysis - Integrating with the API																
17	Fundamental Analysis - Structuring data with pandas																
18	Fundamental Analysis - Getting more data																
19	Fundamental Analysis - Labelling data																
20	Generation of 1st report																
21	Second Presentation																
22	Fundamental Analysis - Studying the SVM Algorithm																
23	Fundamental Analysis - Testing data with SVM																
24	Fundamental Analysis - Adding all ratios (features)																
25	Fundamental Analysis - Training and testing data with SVM																
26	Fundamental Analysis - Integrating with the Api																
27	Technical Analysis - Data Extraction																
28	Technical Analysis - Data cleaning and parsing																
29	Technical Analysis - Structuring data with pandas																
30	Technical Analysis - Labelling data																
31	Technical Analysis - Training and testing data with suitable algorithm																
32	Technical Analysis - Training and testing data with SVM																
33	Technical Analysis - Integrating with the Api																
34	Second Report Submission																
35	Final Demonstration																

5.3 RMMM PLAN:

The primary benefit of risk management is to contain and mitigate threats to project success. We need to identify and plan, and then be ready to act when a risk arises—drawing upon the experience and knowledge of the entire team to minimize the impact to the project.

RMMM includes Mitigation, Monitoring and Management:

Following are the categories of risks involved in the project which needs to be dealt with:

- Project Risk : Schedule of different type of analysis and the API integration needs to be followed strictly. This also includes problems in importing several python modules.
- Technical Risks: This includes all the design issues associated with the Web Framework UI.

Other Risks:

- Server Risk:

  This includes risk involved if the server which contains the API crashes.

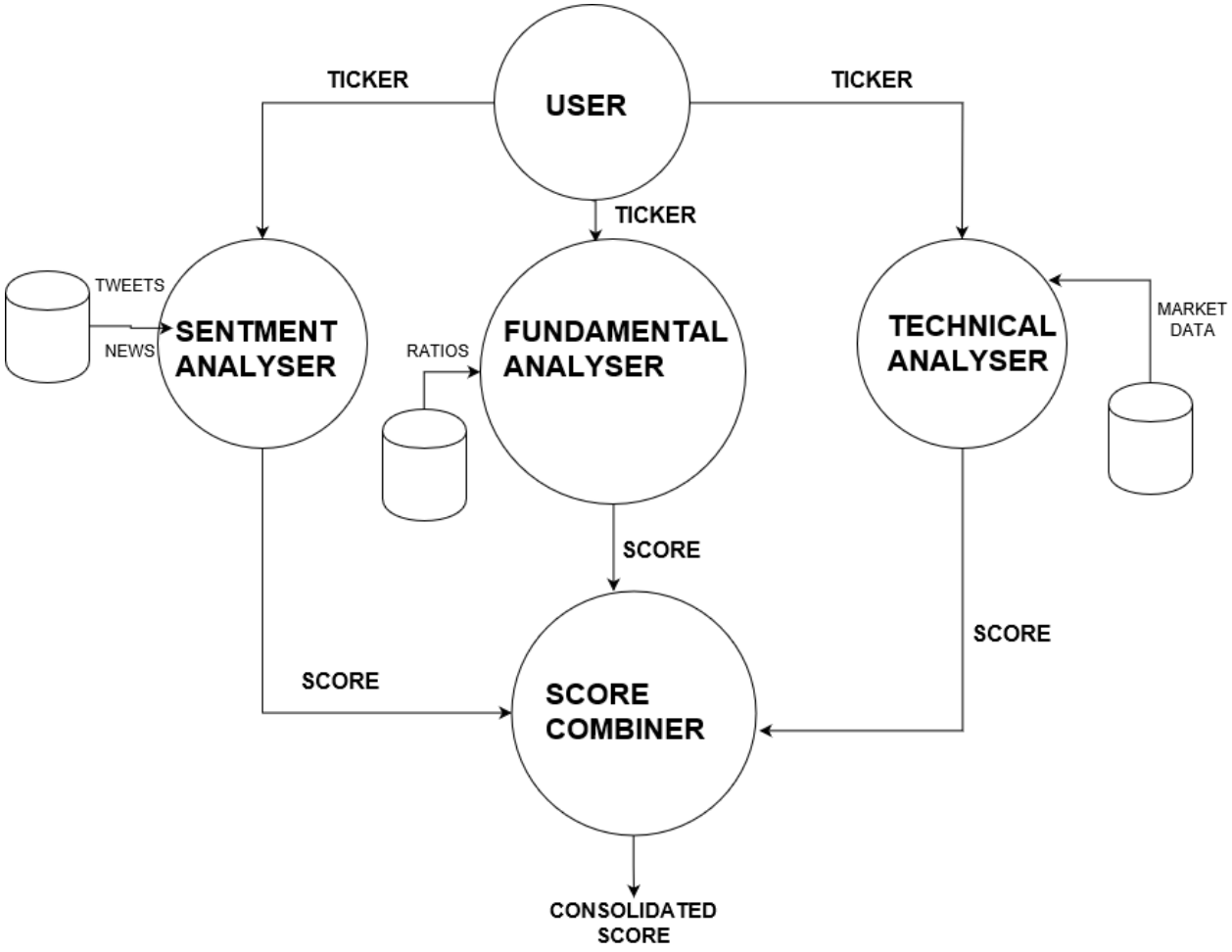
- Unpredictable risk:

  These kind of risks may arise when there is a error in displaying the results.

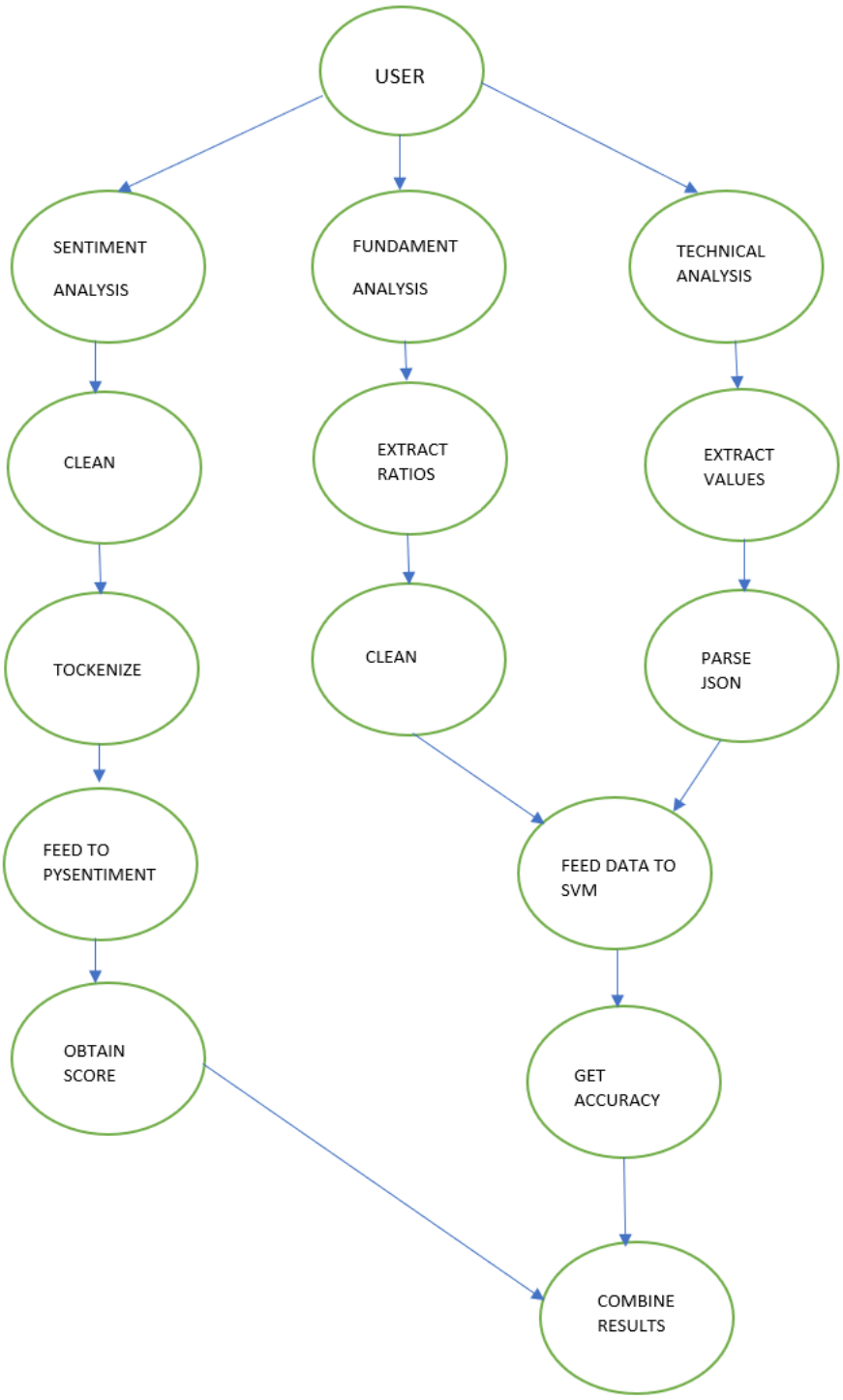
6. HIGH LEVEL DESIGN

6.1. DATA FLOW DIAGRAMS

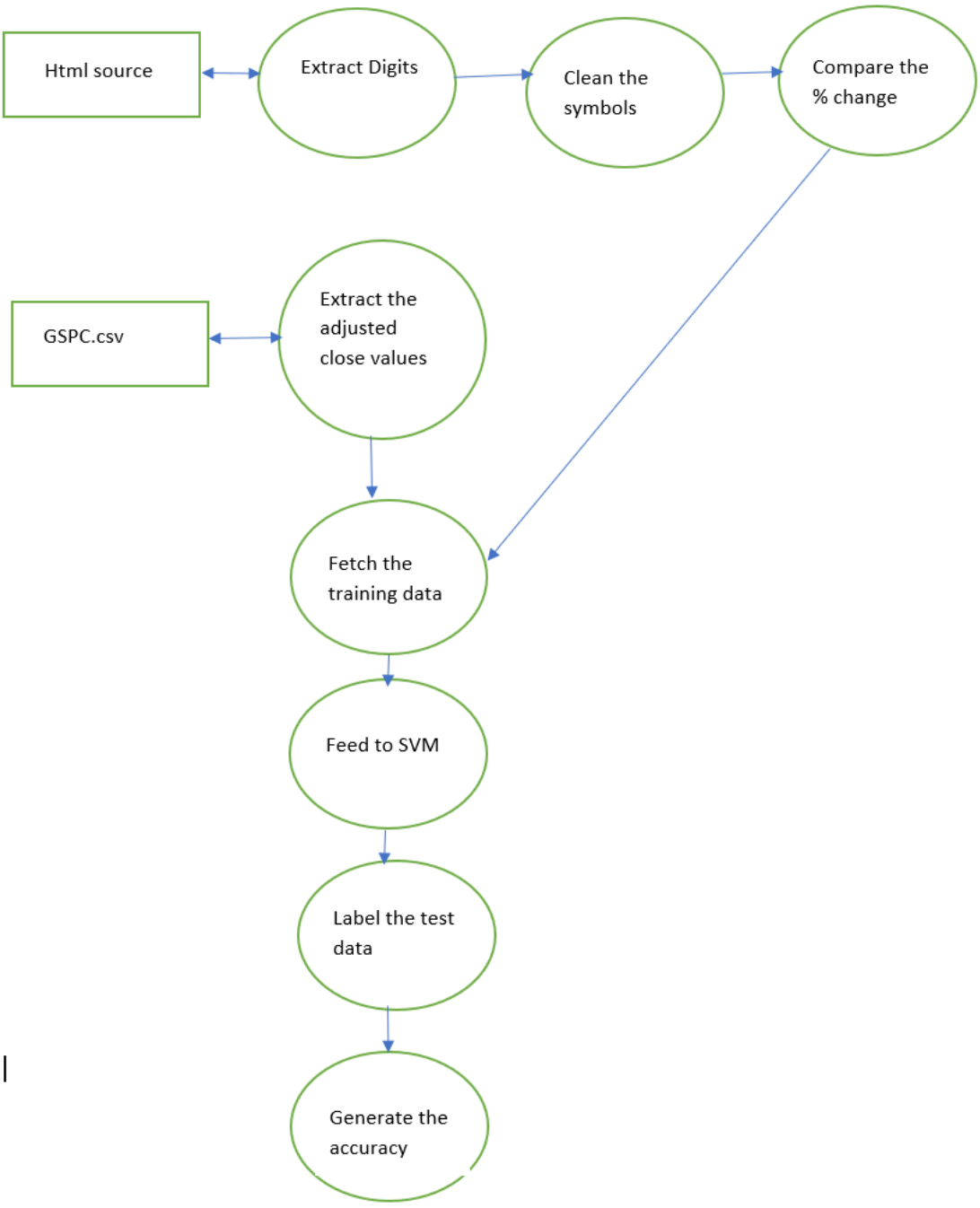
Level 0:



Level 1:


USER
SENTIMENT ANALYSIS
FUNDAMENT ANALYSIS
TECHNICAL ANALYSIS
CLEAN
EXTRACT RATIOS
EXTRACT VALUES
TOCKENIZE
CLEAN
PARSE JSON
FEED TO PYSENTIMENT
FEED DATA TO SVM
OBTAIN SCORE
GET ACCURACY
COMBINE RESULTS

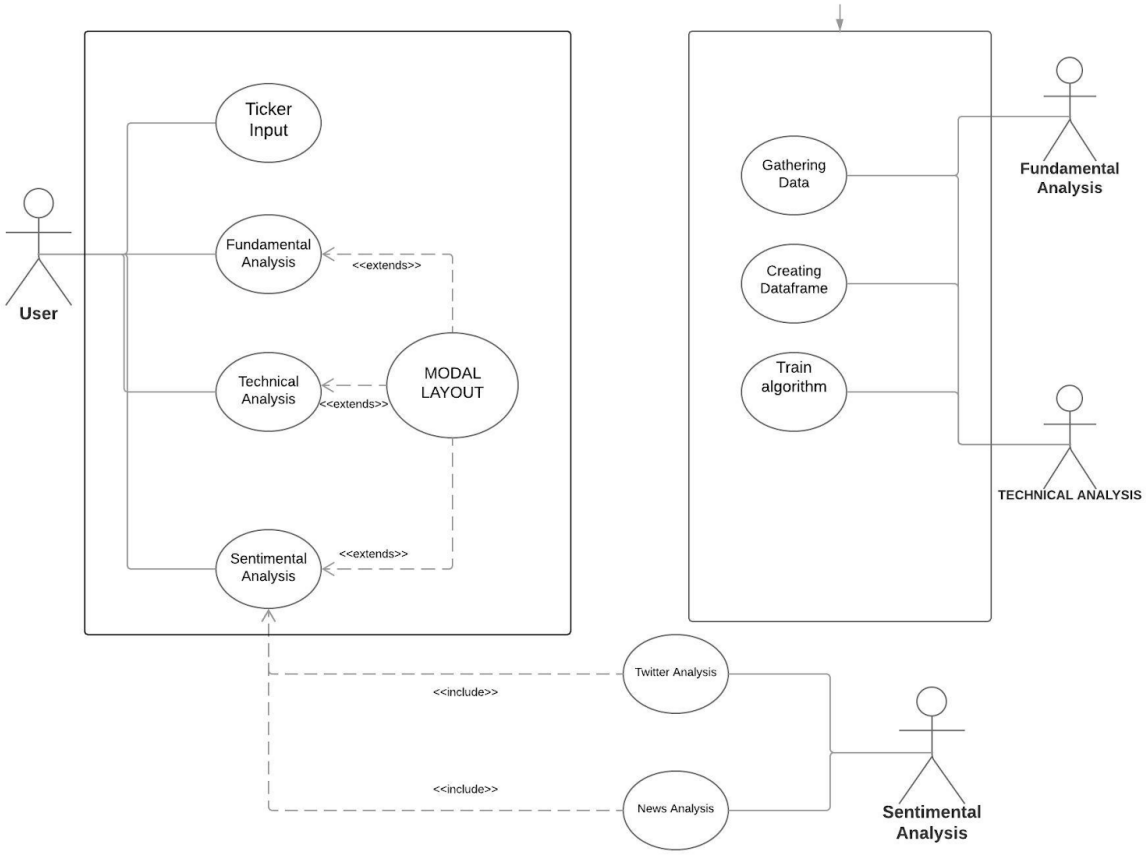
Level 2


Html source
Extract Digits
Clean the symbols
Compare the % change
GSPC.csv
Extract the adjusted close values
Fetch the training data
Feed to SVM
Label the test data
Generate the accuracy

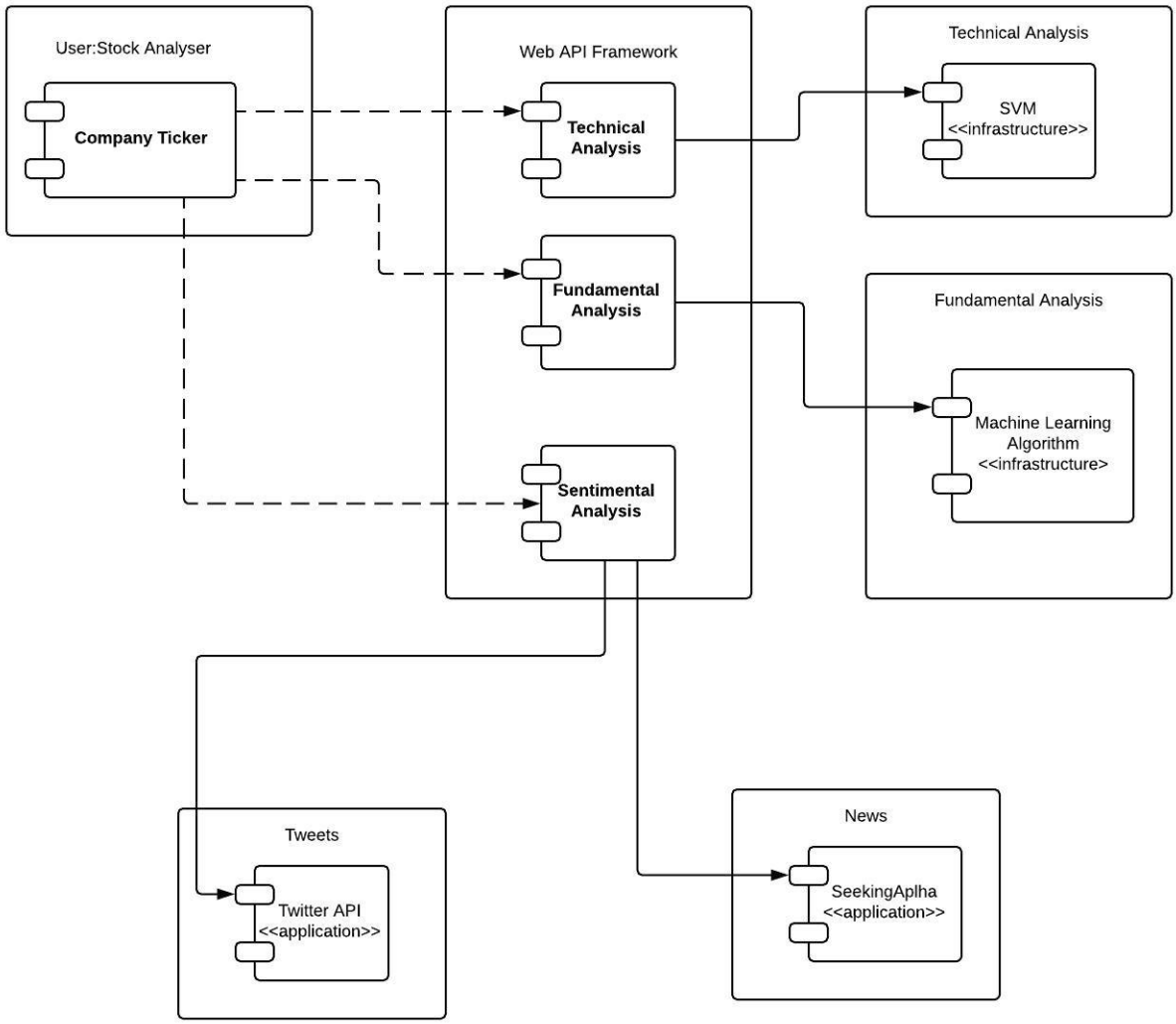
6.2 UML DIAGRAMS

I. USE CASE DIAGRAM:

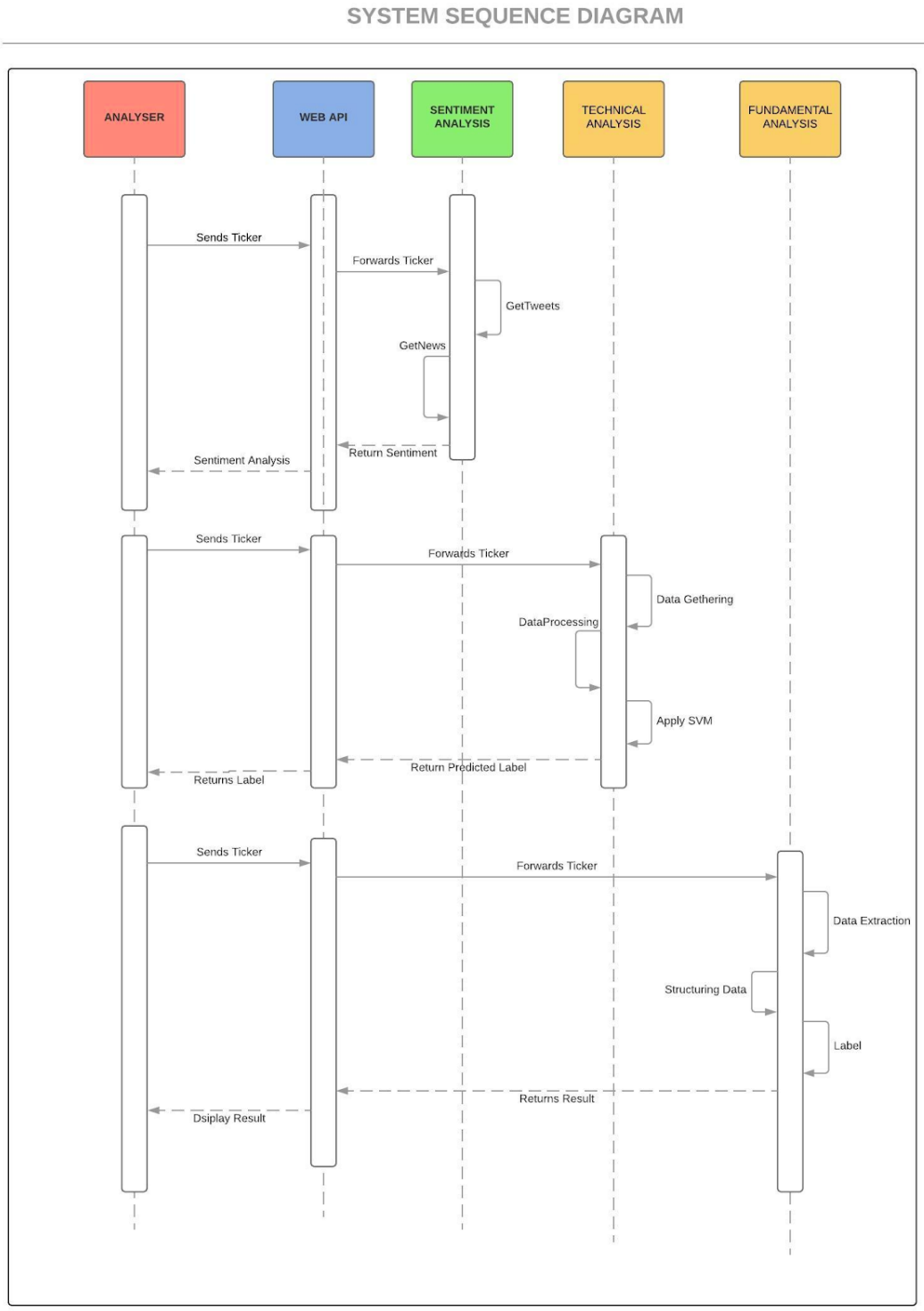
Actors: Analyzer, Fundamental Analysis, Technical Analysis



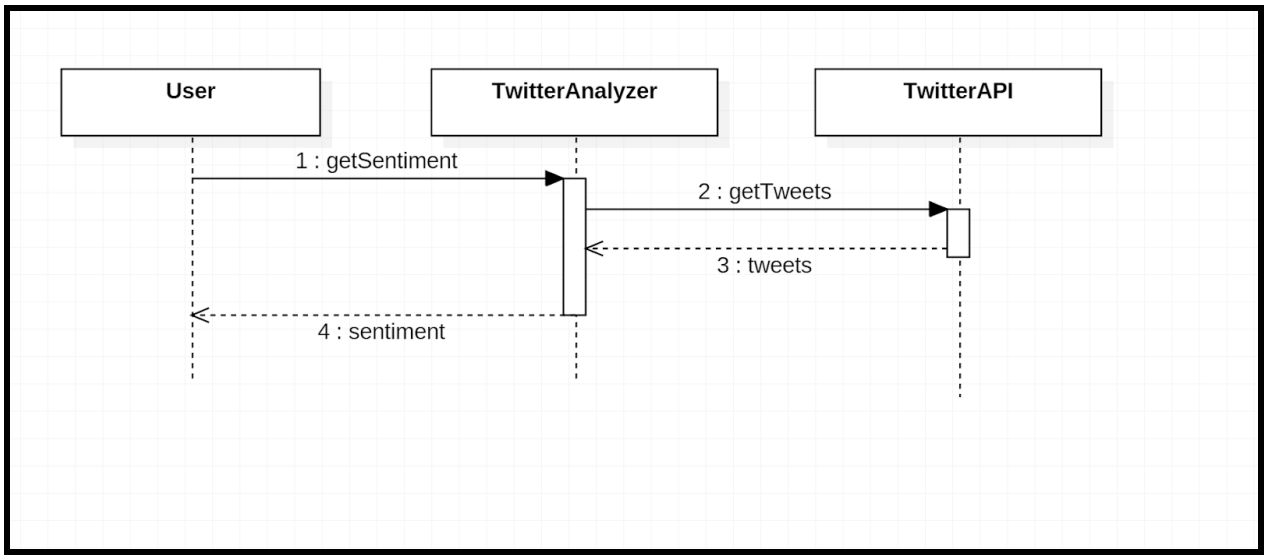
2. Component Diagram:

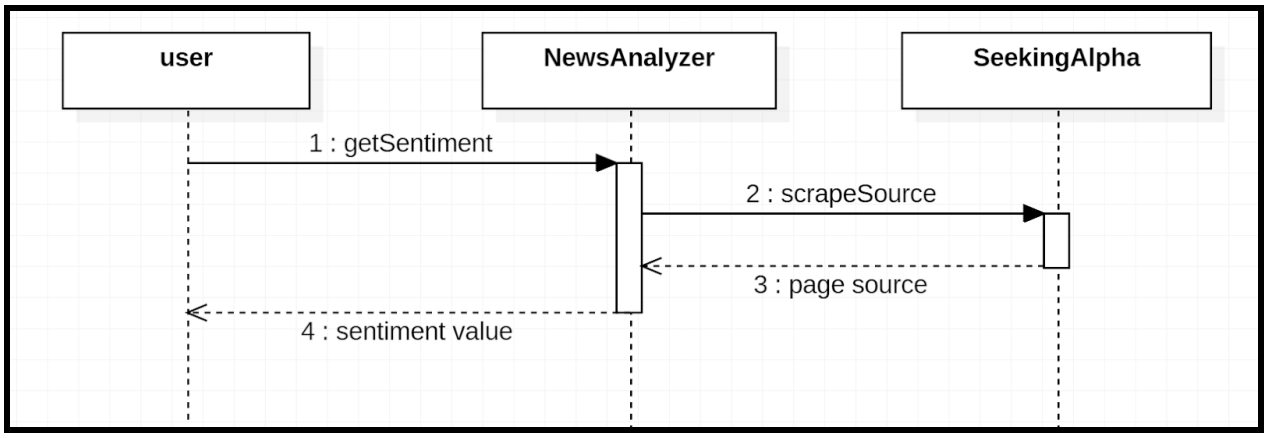


3. Sequence diagram:



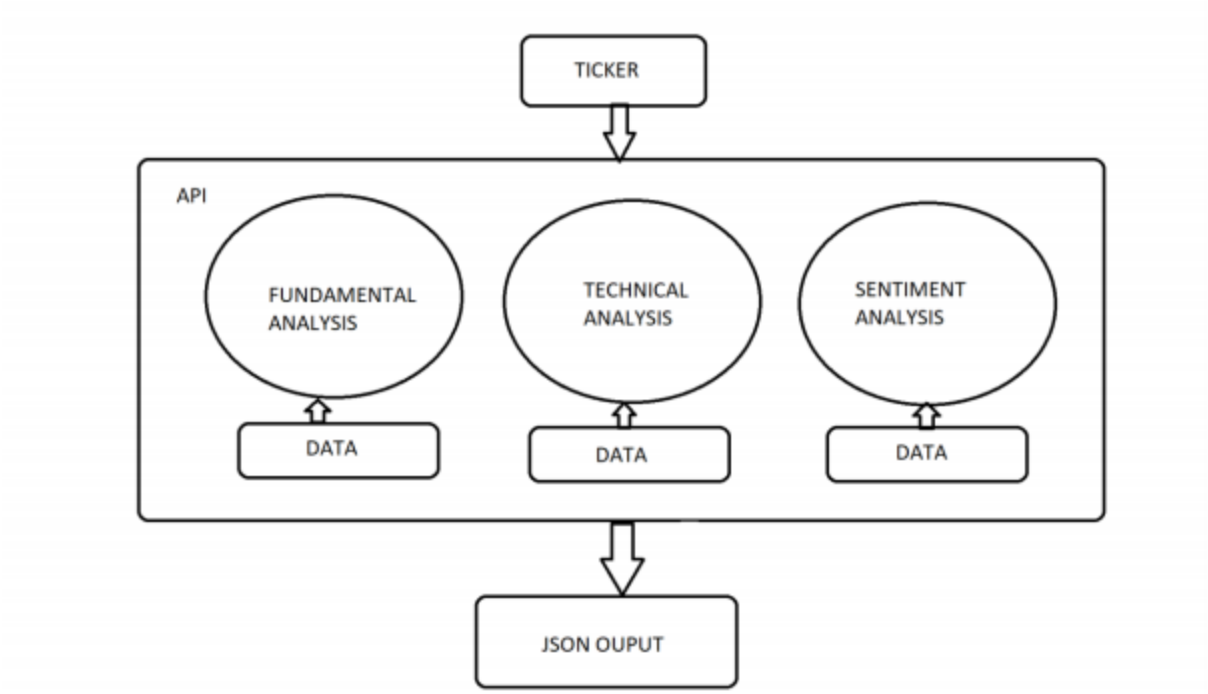
Interaction diagram depicting the sequence of interactions with the twitter API in the sentiment analysis module.





7 IMPLEMENTATION

7.1 SYSTEM MODEL



This system proposes a three phase model where each phase involves working on a particular analysis:

1. Sentiment Analysis
2. Technical Analysis
3. Fundamental Analysis

The brief description of the three analysis models is as follows:

7.1.1 Sentiment Analysis:

In addition to the "usual" tricks of statistical arbitrage, trend-following and fundamental analysis, many retail quants engage in natural language processing (NLP) techniques to build systematic strategies. Such techniques fall under the banner of Sentiment Analysis. The goal of sentiment analysis is, generally, to take large quantities of data in the form of blog posts, newspaper articles, research reports, tweets, etc and use NLP techniques to quantify positive or negative "sentiment" about certain assets. For equities this often amounts to a statistical machine learning analysis of the language utilised and whether it contains bullish or bearish phrasing. This phrasing can be quantified in terms of strength of sentiment, which translates into numerical values.
Often this means positive values reflecting bullish sentiment and negative values representing bearish sentiment.

7.1.2 Fundamental Analysis:

Fundamental analysis is a method of evaluating a security to measure its intrinsic value, by examining related economic, financial and other qualitative and quantitative factors. The end goal of fundamental analysis is to produce a quantitative value that an investor can compare with a security's current price, thus indicating whether the security is undervalued or overvalued.

For example, an investor can perform fundamental analysis on a bond's value by looking at economic factors such as interest rates and the overall state of the economy. He can also look at information about the bond issuer, such as potential changes in credit ratings.

For stocks and equity instruments, this method uses revenues, earnings, future growth, return on equity, profit margins and other data to determine a company's underlying value and potential for future growth. In terms of stocks, fundamental analysis focuses on the financial statements of the company being evaluated.

It involves studying and comparing various ratios such as: p/b ratio, p/e ratio, cash flow debt load, margin price, multiple book value

7.1.3 Technical Analysis:

Technical analysis is a trading tool employed to evaluate securities and attempt to forecast their future movement by analysing statistics gathered from trading activity, such as price movement and volume. Unlike fundamental analysts who attempt to evaluate a security's intrinsic value, technical analysts focus on charts of price movement and various analytical tools to evaluate a security's strength or weakness and forecast future price changes.

Technical Analysis uses the concept of Dow Theory. Two basic assumptions of Dow Theory that underlie all technical analysis are:

1) Market price discounts every factor that may influence a security's price and

2) Market price movements are not purely random but move in identifiable patterns and trends that repeat over time.

Technical analysis is used to attempt to forecast the price movement of virtually any tradable instrument that is generally subject to forces of supply and demand, including stocks, bonds, futures and currency pairs. In fact, technical analysis can be viewed as simply the study of supply and demand forces as reflected in the market price movements of a security. It is most commonly applied to price changes, but some analysts may additionally track numbers other than just price, such as trading volume or open interest figures.

7.2 MODULES

7.2.1 Sentimental Analysis

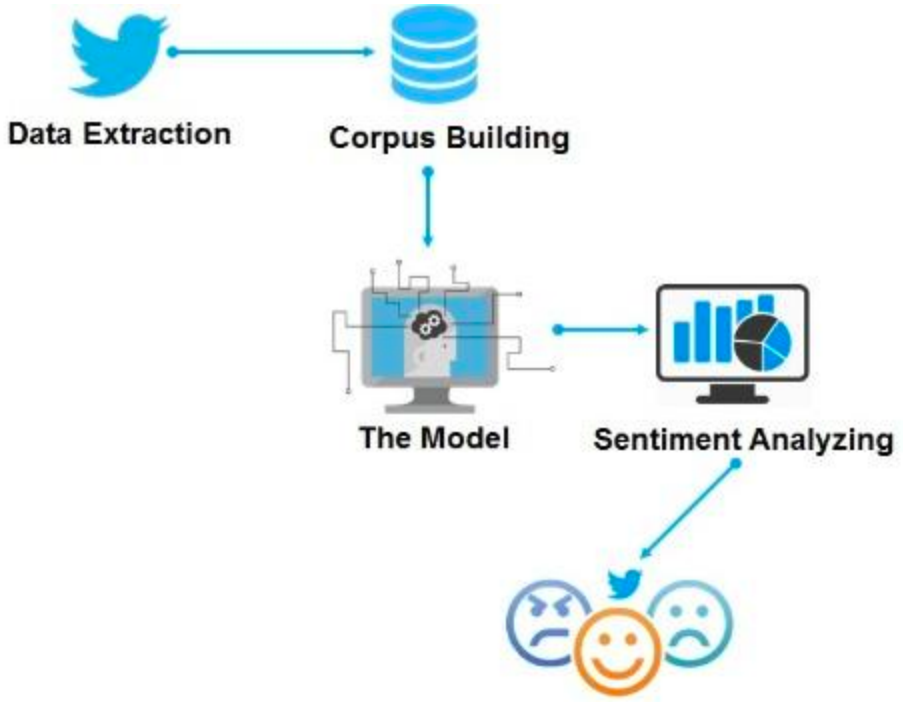
Data sources

After thoroughly searching for sufficiently vast data sources based on various companies' sentiments we arrived at the following viable sources:

1. Tweets
2. News related to the stocks of companies

Tweets

The Twitter's standard search API (search/tweets) allows simple queries against the indices of recent or popular Tweets and behaves similarly to, but not exactly like the Search UI feature available in Twitter mobile or web clients. The Twitter Search API searches against a sampling of recent Tweets published in the past 7 days.



To begin the process we need to register our client application with Twitter. We created a new application and obtained the consumer token and secret which is necessary to request for tweets. We used a module tweepy to query the tweets.
before we can start using the API. We must complete the following steps:

1. Get a request token from twitter
2. Redirect user to twitter.com to authorize our application
3. If using a callback, twitter will redirect the user to us. Otherwise the user must manually supply us with the verifier code.

4. Exchange the authorized request token for an access token.

The application must be authenticated with the help of 4 tokens, viz-a-viz,

- Consumer key
- Consumer secret
- Access secret
- Access token

OAuth is a bit more complicated initially than Basic Auth, since it requires more effort, but the benefits it offers are very lucrative:

- Tweets can be customized to have a string which identifies the app which was used.
- It doesn't reveal user password, making it more secure.
- It's easier to manage the permissions, for example a set of tokens and keys can be generated that only allows reading from the timelines, so in case someone obtains those credentials, he/she won't be able to write or send direct messages, minimizing the risk.
- The application doesn't reply on a password, so even if the user changes it, the application will still work.

Once authorised the tweepy module is used to fetch the tweets.This module uses a cursor to query the tweets with parameters like :

query(q)
This specifies the word to be searched. Here we pass the ticker and the company name as keywords whose tweets are being sought.

Since
The starting date from when the tweets are to be fetched (This is set as the date that was 7 days before the current system date, i.e., system date - 7 days).

Until
The last date (We have set this as the current date fetched from the system).

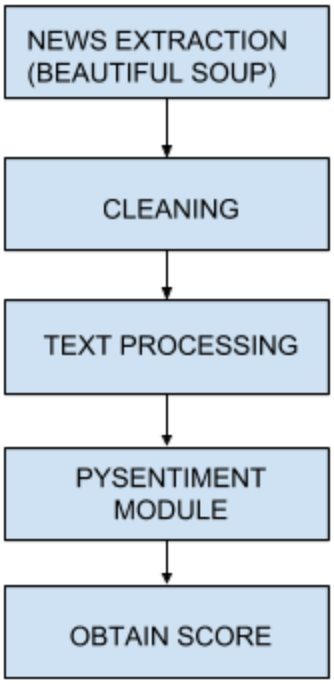
Lang
To obtain tweets in the preferred language (Set as English).

The data provided by Twitter APIs are made up of data objects and their attributes rendered in JavaScript Object Notation (JSON). JSON is based on key-value pairs, with named attributes and associated values. These attributes, and their state, are used to describe objects. Twitter serves many objects as JSON, including *Tweets and Users*. These objects all encapsulate core attributes that describe the object. Each Tweet has an author, a message, a unique ID, a timestamp of when it was posted, and sometimes geo metadata shared by the user. Each User has a Twitter name, an ID, a number of followers, and most often an account bio.

With each Tweet twitter also generates 'entity' objects, which are arrays of common Tweet contents such as hashtags, mentions, media, and links. If there are links, the JSON payload can also provide metadata such as the fully unwound URL and the webpage's title and description. So, in addition to the text content itself, a Tweet can have over 140 attributes associated with it.We fetch the text from the tweets and keep on appending to the forthcoming tweets dynamically.

The next step is the cleaning of the tweets. The text often has hyperlinks and abbreviations like 'RT' for retweets, punctuations, numbers, articles, prepositions and other commonly used words that must be eliminated due to their irrelevance.
We manually removed 'RT' by using the replace function and hyperlinks with the help of regular expressions. The cleaning of the common words and punctuations and numbers is done with the help of **pysentiment** module. Once all the tweets are appended and cleared of links and retweets representation RT, they are fed to this module where they are tokenized. Then according to the predefined dictionary of positive and negative words which are stored in pysentiment, each word is given a score. The result is combined for the entire appended and cleaned set of tweets for that particular company and returned as POSITIVE SCORE and NEGATIVE SCORE.

News



While using news for sentiment analysis, our first step involved identifying the right source of stock related news. While exploring the various sources, we found seekingalpha to be the most viable option, fulfilling all our requirements. Seekingalpha is a dedicated finance focused website, which provides the latest updates and trends in stocks, analysis, as well as news, which we are focused on.

To start with, we decided to use **requests** module in python to send an http get request to the page, with the ticker as parameter, to obtain the page source. After a few requests, on their detection of a number of requests, they stopped returning the page source.

This urged us to another module, selenium. This is most powerful scraping tool available, with certain concerning drawbacks, mainly the scraping time required. But this can be overlooked by us, for we are retrieving past data(one week), and in hindsight, the scraping time is negligible.

Selenium uses **firefox webdriver** and opens the browser in headless mode and accesses the specified link to obtain the page source.

The page source is then passed as a parameter to **beautifulsoup**. This is a module to parse the web page and navigate the elements tag by tag.
With the use of beautifulsoup we navigate the span tag and **extract the text** of the news and append it to the forthcoming news, which is fed to the module pysentiment where they are cleaned off the punctuations, numbers, articles, prepositions and other common irrelevant words and then tokenized. Then according to the predefined dictionary of positive and negative words which are stored in pysentiment, **each word is given a score.** The result is combined for the entire appended and cleaned set of news for that particular company and returned as **POSITIVE SCORE** and **NEGATIVE SCORE.**

7.2.2 FUNDAMENTAL ANALYSIS

For stocks and equity instruments, this method uses revenues, earnings, future growth, return on equity, profit margins and other data to determine a company's underlying value and potential for future growth. In terms of stocks, fundamental analysis focuses on the financial statements of the company being evaluated.
It involves studying and comparing various ratios such as :

- p/B RATIO
- P/E RATIO
- CASH FLOW
- DEBT LOAD
- MARGIN PRICE MULTIPLE
- BOOK VALUE

There are 35 ratios which will become the features of our Machine Learning Model.

Extraction And Parsing

We have stored the data set of the companies offline so that there is no need to visit the pages online, thereby saving the bandwidth and time. We have used the HTML source codes so that it is just like parsing the website. We need to know the corresponding dates to the data and then we pull the actual data.

Modules imported :

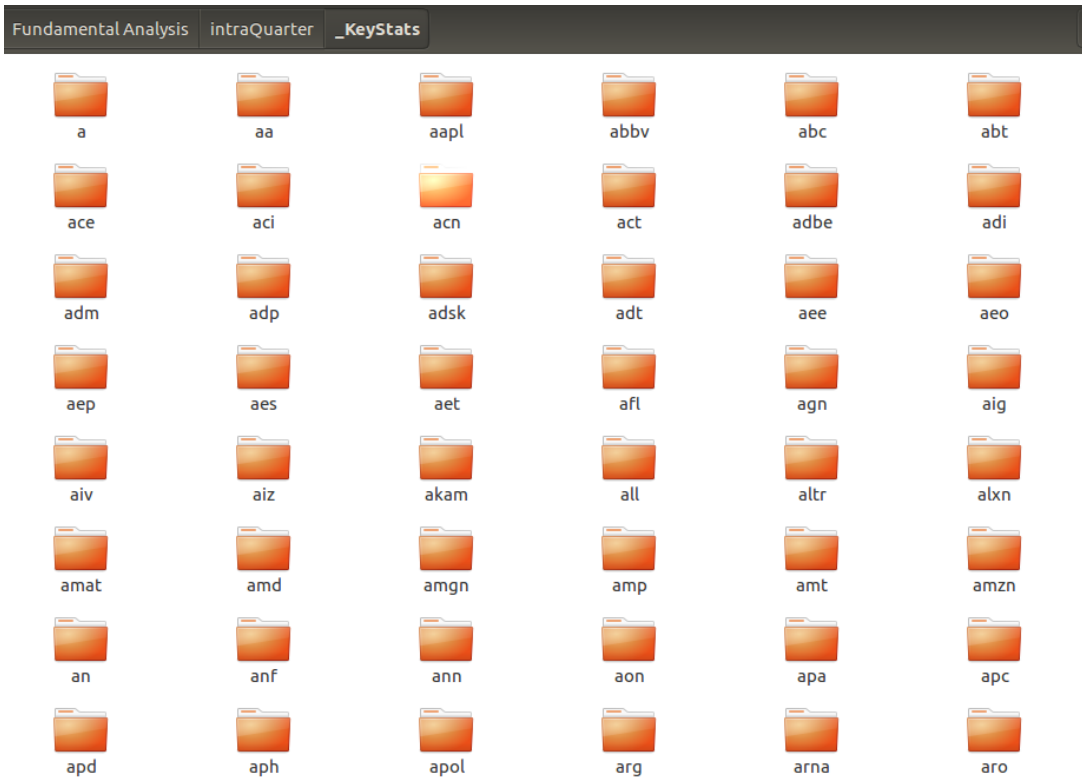
- pandas for the Pandas module
- os so that we can interact with directories
- time and datetime for managing time and date information

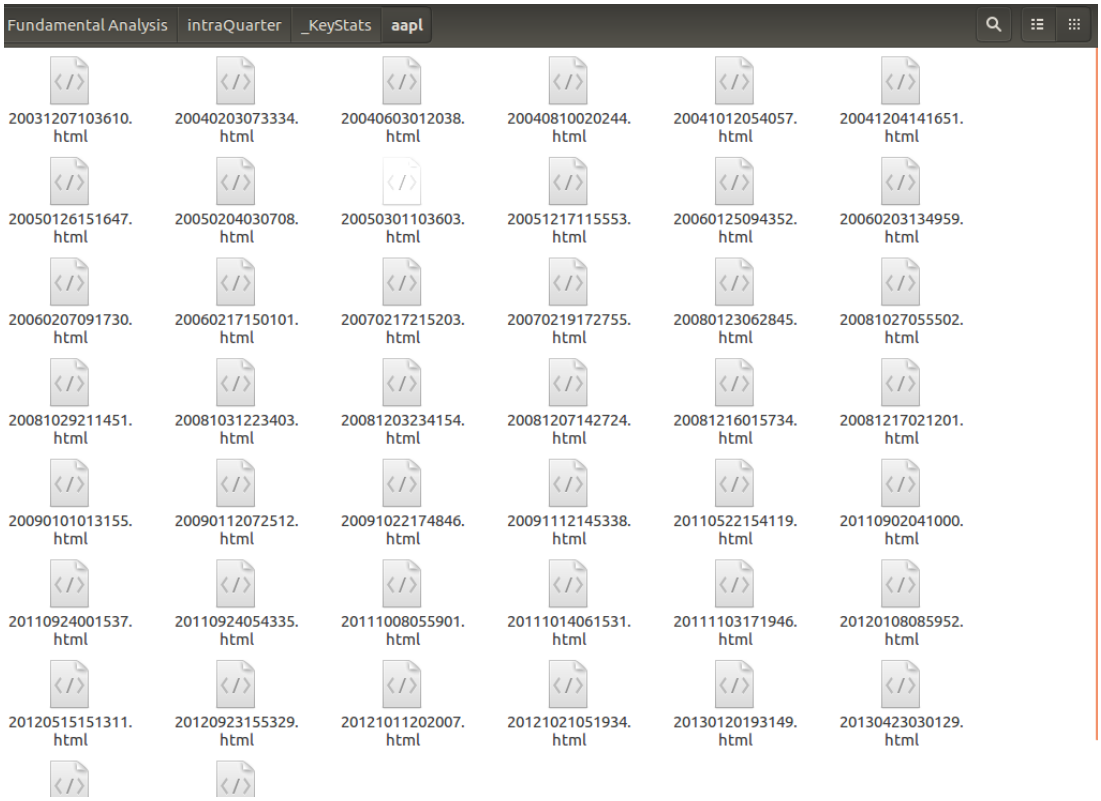
Directory Structure

The feature name list is pickled, i.e., converted into a byte stream for easy access. This pickle is stored in the config.ini file under the name ‘gather’, ‘features’ and ‘useful’.

The **os** module allows us to traverse through a gain directory structure. We can either move up or down the hierarchy.

We have defined "path," which is the path to the intraQuarter folder where the data set is located as shown.





It is clearly visible from this figure that we have the records for each ticker of about a decade. Inside each ticker directory, the fundamental data is saved with the date of that record as its name.

- **Prepeare_ratios.py :** Stores all the pickled features.
- **Prepare_dataset :** Prepares a .csv file consisting of the fundamental features of all the tickers.
- **Fundamental_analysis.py :** The previously prepared data set is fetched and fed to a Machine learning algorithm. It uses neural networks algorithm to train and test and predict the label for a ticker provided.

ITERATION 1

We started only one feature at first i.e., with collecting the Debt/Equity value.

We iterated through every directory of every stock ticker with the help of the os module. Then,

we list "each_file" which is each file within that stock's directory. If the length of each_file, which is a list of all of the files in the stock's directory is greater than 0, then we want to proceed. Some stocks have no files/data. Finally, we run a for loop that pulls the date_stamp from each file. Our files are stored under their ticker, with a file name of the exact date and time from the information being pulled. From there, we specify the format of date stamp then convert it to a unix time-stamp.

Now we can access the file and save the full source code HTML contents to the "source" variable.Then do a quick search for the "**gather**" term, which is the name of the feature we need to consider, available as a pickle then we split by the opening of the table data tag and the table data closing tag to find the required value.

Structuring Data

After extraction, the next step naturally is the structuring and organizing this data into a standard readable format that we can use.

We have used the **Pandas** module for this which is known for its quick, efficient and easy data manipulation abilities. After importing the pandas module we set up the dataframe as ‘df’. The df variable is used to store the creation of a new "DataFrame" object from Pandas, where we initially specify the columns to be date, unix, ticker, and DE ratio.

A **DataFrame** like a spreadsheet or a relational database consisting of a table with rows and columns.

The dataframe is stored as a csv file using the to_csv function of pandas. We want to save the data since we really just need to access and store the data once.

Comparing With The Market

Our end goal with "labeling" data is to categorize it into two categories, outperform or underperform to decide whether it would be better or not to invest in the company,

We are going to compare to "market," or the S&P 500 index. We extracted the S&P 500 value from Yahoo Finance.

The data can simply be downloaded as a csv file for the specified range of duration. The next step is to read the data and load it into the DataFrame with Pandas.

This csv can be loaded into a dataframe by using from_csv, as well as initiating our data frame with a few new columns.

Some of our stock data may have been pulled on a weekend day. If we search for a weekend day's value of the S&P 500, that date just simply wont exist in the dataset, since there is only data for Monday-Friday. This we handle by using a try-except statement, containing the definition of the sp500 date and value and looking for the value of the S&P 500 index at the same time as the date for our stock file.

Labelling

The next step is to now label our data by comparing the stock's percentage change to the S&P 500's percentage change. If the stock's percent change is less than the S&P 500, then the stock is and under-performing stock. If the percentage change is more, than the label is out-perform. The starting_stock_value and the matching sp500 version are set as false so, that as we go, we want to calculate % change, we need to start over with the % change each time the stock itself changes. To handle for this, we set these values.

So now we set the starting value if we don't have one. Then just need to calculate % change (new-old)/old * 100

With supervised learning, the labelling of the data will be done. We use machine learning. This concept is based on: First "show" and "teach" the machine some examples of "what is." Then, eventually show the machine some data, without the label, expecting it to predict the result.

First feed the data with labels to train the machine. Then test the machine by feeding information and seeing what the answer is compared to the known answer.

Kindly Note:

It is not possible for the machine to be extremely accurate, especially in stocks prediction. Therefore, we look if the results are at least above coin toss accuracy, otherwise the machine requires improvement. -As suggested by experts.

It must be noted that with investing in mind, not only is accuracy important, but so is the degree by which the companies perform.

The next step is labeling the data as a 0 or a 1, or as underperform or outperform considering the

S&P 500 index as comparison, considering the "Adjusted close" value which is a closing price that accounts for any stock splits and adjusts historical prices for accuracy.

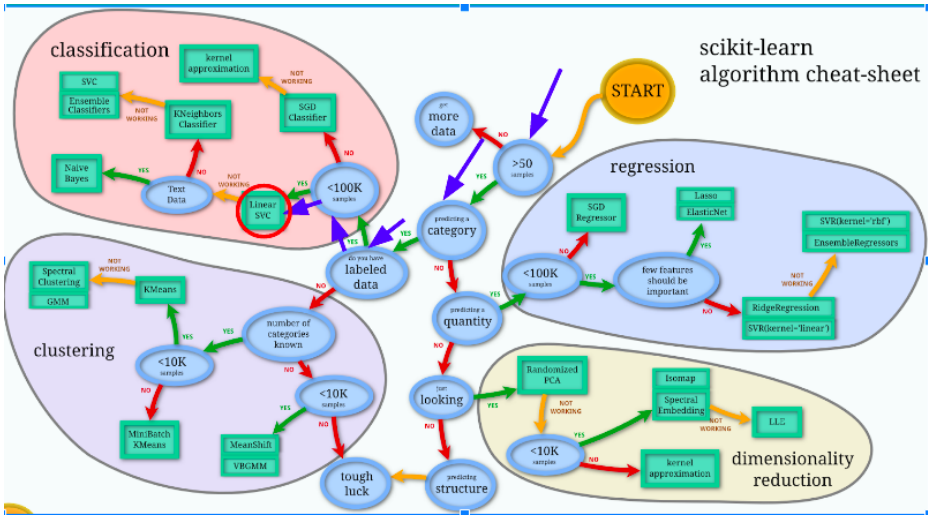
Before we started with this step we needed to understand the following concept Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

Methods:

Machine learning is categorized as supervised or unsupervised learning :

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- **Unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

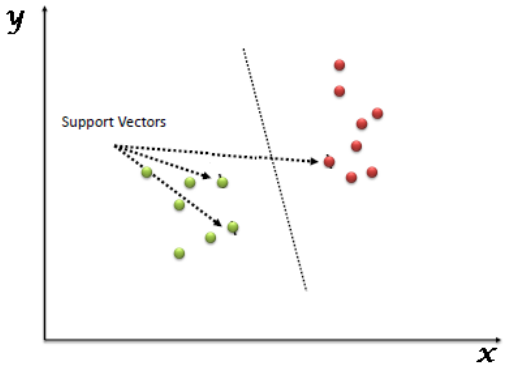
For fundamental analysis we have used support vector machine algorithm which is a supervised learning algorithm and is available in python scikit learn module.The reason for selecting SVM can be demonstrated from the scikit cheat sheet shown below:



This figure shows the parameters to be taken into consideration before choosing the appropriate algorithm (viz-a-viz, samples>50, predicting a category, labelling a data, with < 10k samples).

Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiate the two classes very well (look at the below snapshot).



Support Vectors are simply the coordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

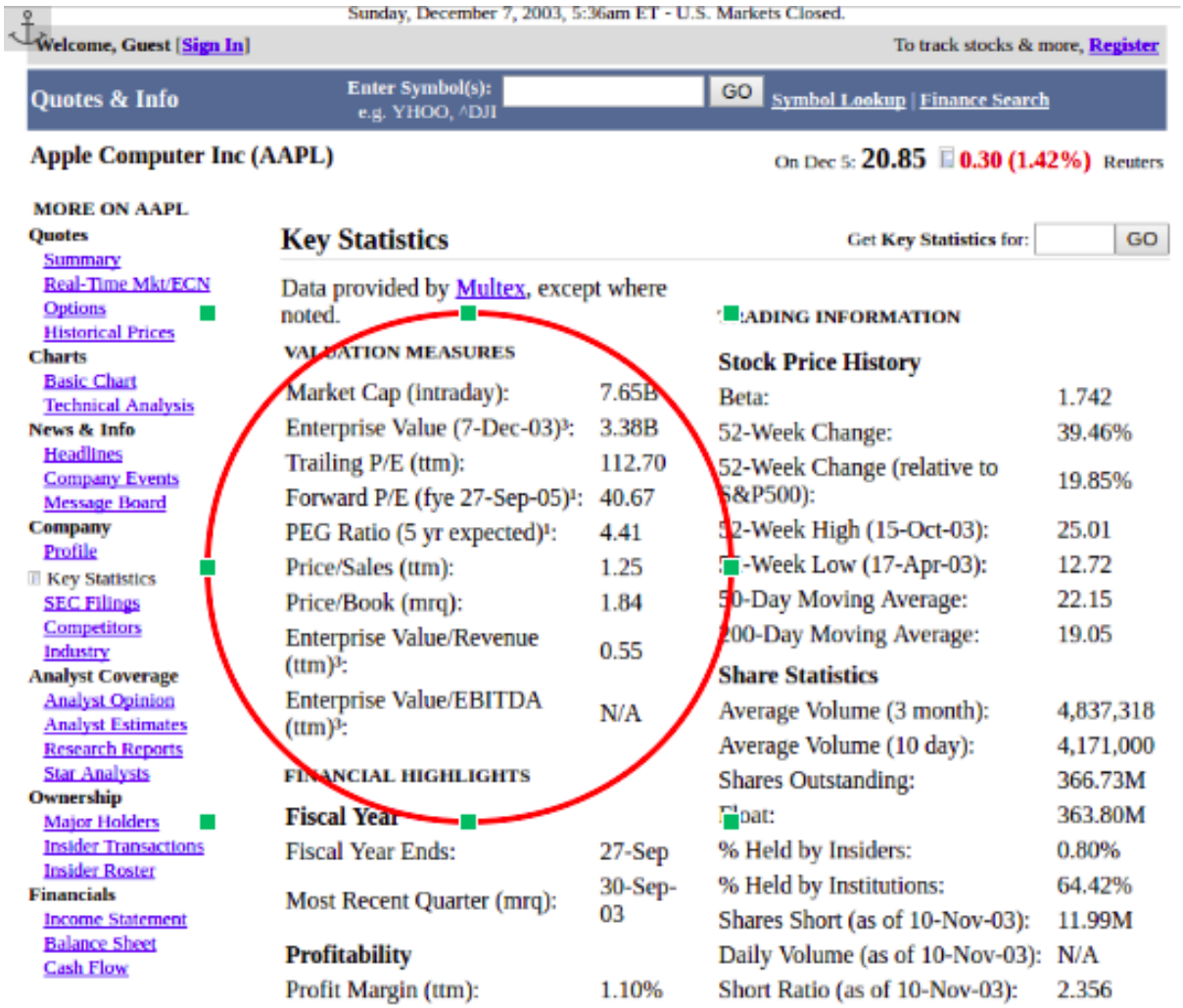
For the labelling of our data, we are going to compare the stock's percentage change to the S&P 500's percentage change. If the stock's percent change is less than the S&P 500, then the stock is and “under-performing” stock. If the percentage change is more, than the label is “out-perform”.

ITERATION - 2

Before we apply SVM to the data we must first gather more features. Uptil now we have only considered the Debt/Equity values or the companies. The other features that must be taken into consideration are:

DE Ratio, Trailing P/E, Price/Sales, Price/Book, Profit Margin, Operating Margin, Return on Assets, Return on Equity, Revenue Per Share, Market Cap, Enterprise Value, Forward P/E, PEG Ratio, Enterprise Value/Revenue, Enterprise Value/EBITDA, Revenue, Gross Profit, EBITDA, Net Income

Avl to Common , Diluted EPS, Earnings Growth, Revenue Growth, Total Cash, Total Cash Per Share, Total Debt, Current Ratio, Book Value Per Share, Cash Flow, Beta, Held by Insiders, Held by Institutions, Shares Short (as of, Short Ratio, Short % of Float, Shares Short.


Sunday, December 7, 2003, 5:36am ET - U.S. Markets Closed.

Welcome, Guest [Sign In] To track stocks & more, Register

Quotes & Info Enter Symbol(s): e.g. YHOO, ^DJI GO Symbol Lookup | Finance Search

Apple Computer Inc (AAPL) On Dec 5: 20.85 0.30 (1.42%) Reuters

MORE ON AAPL
Quotes
Summary
Real-Time Mkt/ECN
Options
Historical Prices
Charts
Basic Chart
Technical Analysis
News & Info
Headlines
Company Events
Message Board
Company
Profile
Key Statistics
SEC Filings
Competitors
Industry
Analyst Coverage
Analyst Opinion
Analyst Estimates
Research Reports
Star Analysts
Ownership
Major Holders
Insider Transactions
Insider Roster
Financials
Income Statement
Balance Sheet
Cash Flow

Key Statistics

Get Key Statistics for: GO

Data provided by Multex, except where noted.

VALUATION MEASURES

Market Cap (intraday):	7.65B
Enterprise Value (7-Dec-03)³:	3.38B
Trailing P/E (ttm):	112.70
Forward P/E (fye 27-Sep-05)¹:	40.67
PEG Ratio (5 yr expected)¹:	4.41
Price/Sales (ttm):	1.25
Price/Book (mrq):	1.84
Enterprise Value/Revenue (ttm)³:	0.55
Enterprise Value/EBITDA (ttm)³:	N/A

FINANCIAL HIGHLIGHTS

Fiscal Year

Fiscal Year Ends:	27-Sep
Most Recent Quarter (mrq):	30-Sep-03

Profitability

Profit Margin (ttm):	1.10%

TRADING INFORMATION

Stock Price History

Beta:	1.742
52-Week Change:	39.46%
52-Week Change (relative to S&P500):	19.85%
52-Week High (15-Oct-03):	25.01
52-Week Low (17-Apr-03):	12.72
50-Day Moving Average:	22.15
200-Day Moving Average:	19.05

Share Statistics

Average Volume (3 month):	4,837,318
Average Volume (10 day):	4,171,000
Shares Outstanding:	366.73M
Float:	363.80M
% Held by Insiders:	0.80%
% Held by Institutions:	64.42%
Shares Short (as of 10-Nov-03):	11.99M
Daily Volume (as of 10-Nov-03):	N/A
Short Ratio (as of 10-Nov-03):	2.356

Cleaning:

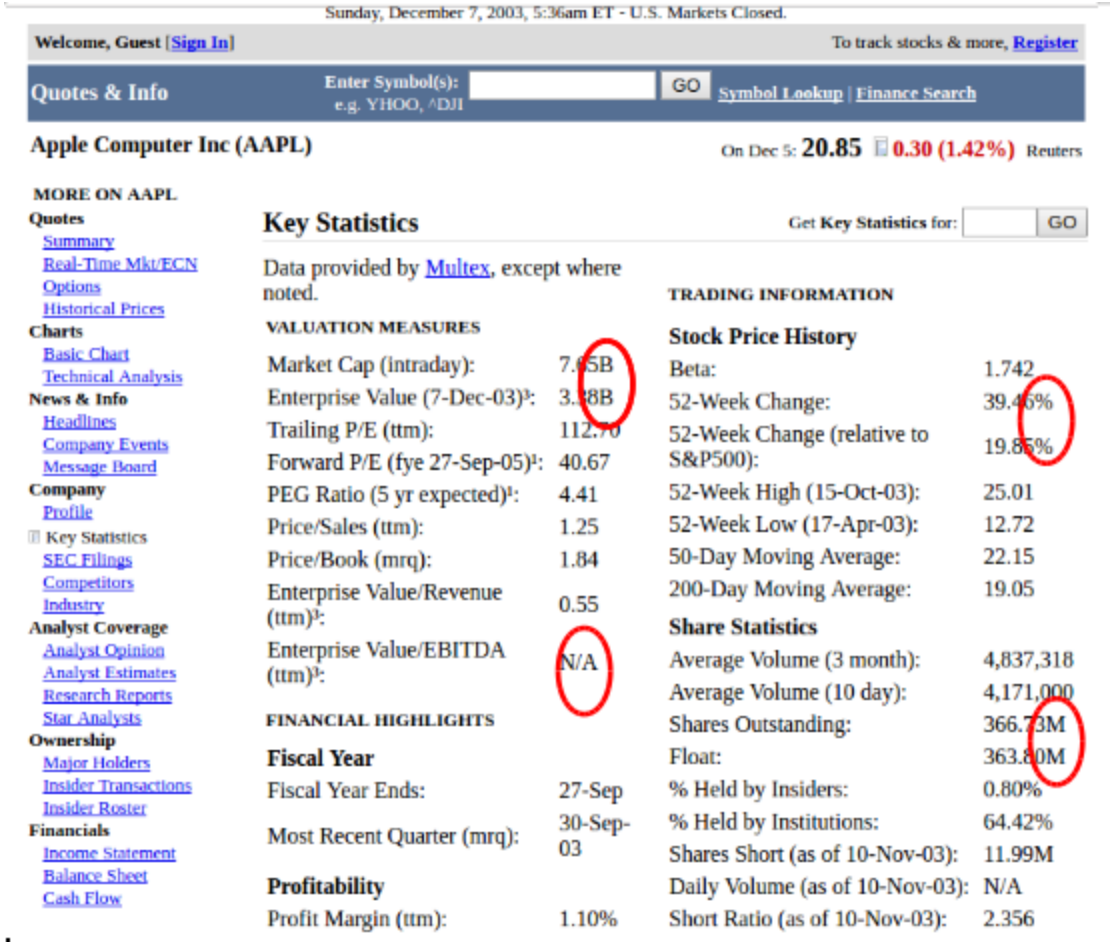
There are various symbols and abbreviations like percentages, superscripts, subscripts, "M" for million, or "B" for billion which must be removed.

If the row contains any missing, or N/A, value, we're just going to ignore it. We simply will not store the information.

In general, with machine learning, ideally want the data normalized, which means all features are

on a similar scale.

The data might range from 0 to 12 typically, where other features go from 0 to 50 billion typically. ALthough it is possible to feed data features that vary this much through the machine learning algorithm, the results will likely be less accurate and less useful than if you were to scale it.


Sunday, December 7, 2003, 5:36am ET - U.S. Markets Closed.

Welcome, Guest [Sign In] To track stocks & more, Register

Quotes & Info Enter Symbol(s): e.g. YHOO, ^DJI GO Symbol Lookup | Finance Search

Apple Computer Inc (AAPL) On Dec 5: 20.85 0.30 (1.42%) Reuters

MORE ON AAPL
Quotes
Summary
Real-Time Mkt/ECN
Options
Historical Prices
Charts
Basic Chart
Technical Analysis
News & Info
Headlines
Company Events
Message Board
Company
Profile
Key Statistics
SEC Filings
Competitors
Industry
Analyst Coverage
Analyst Opinion
Analyst Estimates
Research Reports
Star Analysts
Ownership
Major Holders
Insider Transactions
Insider Roster
Financials
Income Statement
Balance Sheet
Cash Flow

Key Statistics Get Key Statistics for: GO

Data provided by Multex, except where noted.

VALUATION MEASURES

Market Cap (intraday):	7.65B
Enterprise Value (7-Dec-03)³:	3.38B
Trailing P/E (ttm):	112.70
Forward P/E (fye 27-Sep-05)¹:	40.67
PEG Ratio (5 yr expected)¹:	4.41
Price/Sales (ttm):	1.25
Price/Book (mrq):	1.84
Enterprise Value/Revenue (ttm)³:	0.55
Enterprise Value/EBITDA (ttm)³:	N/A

FINANCIAL HIGHLIGHTS

Fiscal Year

Fiscal Year Ends:	27-Sep
Most Recent Quarter (mrq):	30-Sep-03

Profitability

Profit Margin (ttm):	1.10%

TRADING INFORMATION

Stock Price History

Beta:	1.742
52-Week Change:	39.46%
52-Week Change (relative to S&P500):	19.85%
52-Week High (15-Oct-03):	25.01
52-Week Low (17-Apr-03):	12.72
50-Day Moving Average:	22.15
200-Day Moving Average:	19.05

Share Statistics

Average Volume (3 month):	4,837,318
Average Volume (10 day):	4,171,000
Shares Outstanding:	366.73M
Float:	363.80M
% Held by Insiders:	0.80%
% Held by Institutions:	64.42%
Shares Short (as of 10-Nov-03):	11.99M
Daily Volume (as of 10-Nov-03):	N/A
Short Ratio (as of 10-Nov-03):	2.356

I

Pre processing

Applying changes to the data before running it through the machine learning algorithm is generally referred to as "pre-processing." This involves more than just scaling and normalizing the data.

Scikit-Learn has a pre-built in functionality under sklearn.preprocessing for data preprocessing. We have used sklearn.preprocessing.scale.

This will scale the data for. Scaling requires to have an accurate range, but its not possible to just do a simple conversion using the minimum and maximum of the features and then convert based on the scaling differences, because any significant outliers will significantly cause problems. We want the scales to still accurate reflect as best as possible.

We use a function to build our data set, with the use of preprocessing.scale()

test_size is being used as a variable to determine how much data that we would like to reserve for testing.

Then unpack the returned values of the data-set building functions to X,y, then printing out the length of the data.Next we specify the classifier we intend to use, and then fit the data to that classifier.

We fit the data to the negative index of "test_size", to make sure we do not train on the amount of data that we specify that we want to test against.This is done in order to prevent training and testing against identical data.

Next, we begin testing our data with a simple for-loop, which will check predictions of all of the testing data and then compare the predictions to reality.

Upon running this script, the resultant accuracy is found to be something between ~53-63%.

It's important to note that the data we are looking at is not really black and white. We are converting a spectrum of data (performance, which could be anything from -15% to +15% on average) to black and white. Our predictions are right more often than not, we're actually more curious about the "spectrum" in the end.

CHALLENGE

The accuracy obtained with the above model was between 53% to 59%. In order to improve the accuracy of the model we migrated to neural networks supervised learning classifier called Multilayer Perceptron (MLP) classifier.

Reason for change

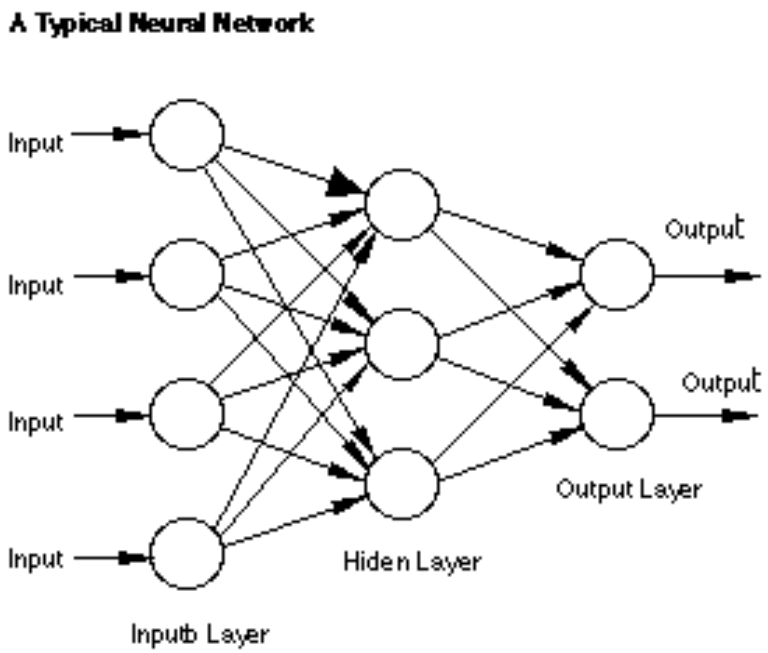
ALthough SVMs are known to give a more accurate prediction with a faster processing rate, SVMs do not outperform NNs currently in many tasks, especially when the dimensionality is high. Therefore, as the data size goes on increasing the accuracy of SVM algorithm decreases. Since,

this system requires around 35 features with a large number of tickers, being studied for over 2 decades, it was necessary to adapt to a more aptly suited algorithm for the prediction.

By using the MLP Classifier, the accuracy range increased from [53% to 59%] by SVM to [64% to 70%] by the MLP Classifier.

Neural Network

A neural network has several input, hidden, and output nodes. Each node applies a function some data (could be softmax, linear, logistic), and returns an output. Every node in the proceeding layer takes a weighted average of the outputs of the previous layer, until an output is reached. The reasoning is that multiple nodes can collectively gain insight about solving a problem (like classification) that an individual node cannot. The cost function differs for this type of model -- the weights between nodes adjust to minimize error.



MLP Classifier

The multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers and each layer is

fully connected to the following one. The nodes of the layers are neurons using nonlinear activation functions, except for the nodes of the input layer. There can be one or more non-linear hidden layers between the input and the output layer.

This classifier is also available in the sklearn module of python, just like SVM.

Implementation of MLP Classifier

Parameters (from the code)

```
-----------------------------------------------------------------------------------------------------------
clf = MLPClassifier(hidden_layer_sizes=1500, alpha=0.03, max_iter=6000, random_state=42)
scaler = StandardScaler()
#scaling the features to increase the accuracy
x_train_scaled = scaler.fit(x_train).transform(x_train)
x_test_scaled = scaler.fit(x_test).transform(x_test)
model = clf.fit(x_train_scaled, y_train)
print(clf.score(x_train_scaled, y_train)) #Prints the accuracy of the training data model
print(clf.score(x_test_scaled, y_test))   #Prints the accuracy of the test data model
-----------------------------------------------------------------------------------------------------------
```

clf : classifier variable

- **hidden_layer_sizes :** tuple, length = n_layers - 2, default (100,)

  The ith element represents the number of neurons in the ith hidden layer.

- **alpha** : float, optional, default 0.0001

  L2 penalty (regularization term) parameter.

- **max_iter** : int, optional, default 200

Maximum number of iterations. The solver iterates until convergence (determined by ‘tol’) or this number of iterations. For stochastic solvers (‘sgd’, ‘adam’), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

- **random_state** : int, RandomState instance or None, optional, default None

  If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

Features and Labels:

The features are the 35 fundamental ratios.

The labels are [0, 1, where 0 - “UNDERPERFORM”, 1 - “OVERPERFORM”.

(X, Y) : X contains the list of all the 35 features.

Y contains the labels [0, 1].

Scaling :

We scaled the data with the help of the inbuilt function available in python - **StandardScaler().** This scales the values, for example one set of values range from [1, 5000] and another set of values range from [1000, 2000], scaling enables us to scale these vastly varying values and normalize them for a better accuracy.

Prediction :

def predict(url, ticker, data_dir)

This function takes as input

1 url : The url of the fundamental ratios of a particular company from the yahoo finance. It is specified in the config.ini file : https://in.finance.yahoo.com/quote/

E.g. https://in.finance.yahoo.com/quote/AAPL/key-statistics?p=/AAPL

2 ticker: The ticker of the company whose fundamental data is to be analysed.

3 data_dir: Path to the data directory having the stored set of pickled features

The ‘predict’ function calls the ‘extract’ function which extracts the following values :

page = requests.get(url + ticker.upper() + **'/key-statistics?p='** + ticker.upper()).text

The html page of the ticker with the fundamental ratios of the current day is fetched with the help of the python requests module. It will have the most recent features of the company for the present day from the HTML source code.

gather = pickle_get(os.path.join(data_dir, **'ratios'**))

The list of the features to be considered from the pickle.

tables = pd.read_html(page)

The variable **‘tables‘** fetches all of the ratios as a data frame of python pandas module.

This variable has the value for each of the 35 features to b considered as of current day, fetched from the url.

value_list = []

This list is filled with the ‘feature name’ (from gather and their ‘Corresponding values’ from the tables variable.

Then the value list is cleaned of the various symbols like ‘B’, ‘M’, %’, etcetera and replaces the ‘N/A’’ with null.

The extract function returns this value_list which has the values of the fundamental ratios of the specified ticker for the present day.

The predict function then uses these current variables to predict the performance based upon the model developed.

The model used is developed using MLP classifier, with the parameters as described earlier.

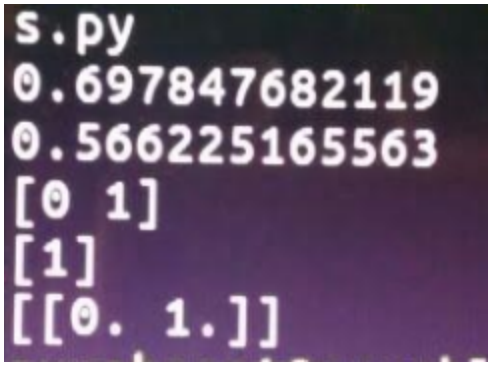
The (X,Y) have the features and the label respectively .

The accuracy of the training and testing data model was determined with the fit_algo function as described above. Now, the predict function uses

print(model.predict(X)) to predict the label for the given company ticker.

print(model.predict_proba(X)) to determine the probability of this prediction.

OUTPUT:



%ACCURACY OF TRAINING SET: 69%

%ACCURACY OF TESTING SET : 56%

Class labels : [o and 1]

Predicted label : 1

Prediction probability: 100%

TECHNICAL ANALYSIS PROCESS

DATA GATHERING

For the data gathering phase, we use Alpha Vantage's API, which provides all the data we require for technical analysis, including the open-close values for each day, along with other ratios like the simple moving average(SMA), exponential moving average(EMA), moving average convergence / divergence (MACD), stochastic oscillator (STOCH), relative strength index (RSI), Aroon (AROON) etc.

Below are some of the parameters of the get request we send :

API Parameters

- Required: `function`

The technical indicator of your choice. In this case, `function=AROON`

- Required: `symbol`

The name of the equity of your choice. For example: `symbol=MSFT`

- Required: `interval`

Time interval between two consecutive data points in the time series. The following values are supported: `1min`, `5min`, `15min`, `30min`, `60min`, `daily`, `weekly`, `monthly`

- Required: `time_period`

Number of data points used to calculate each AROON value. Positive integers are accepted (e.g., `time_period=60`, `time_period=200`)

- Required: `apikey`

API Parameters

- Required: `function`

The time series of your choice. In this case, `function=TIME_SERIES_DAILY`

- Required: `symbol`

The name of the equity of your choice. For example: `symbol=MSFT`

- Optional: `outputsize`

By default, `outputsize=compact`. Strings `compact` and `full` are accepted with the following specifications: `compact` returns only the latest 100 data points; `full` returns the full-length time series of up to 20 years of historical data. The "compact" option is recommended if you would like to reduce the data size of each API call.

- Optional: `datatype`

By default, `datatype=json`. Strings `json` and `csv` are accepted with the following specifications: `json` returns the daily time series in JSON format; `csv` returns the time series as a CSV (comma separated value) file.

- Required: `apikey`

As we can see, the API is well defined to our requirements, with the time period, symbol/ticker and the data format all adjustable. We shall use the JSON format and parse the data for further processing.

DATA PROCESSING

We use the json module in python to parse the data we receive as input from the API, and use each key as a column, and the values as the corresponding rows. We extract the days separately so as to use it for the further ratios to append to the dataframe.

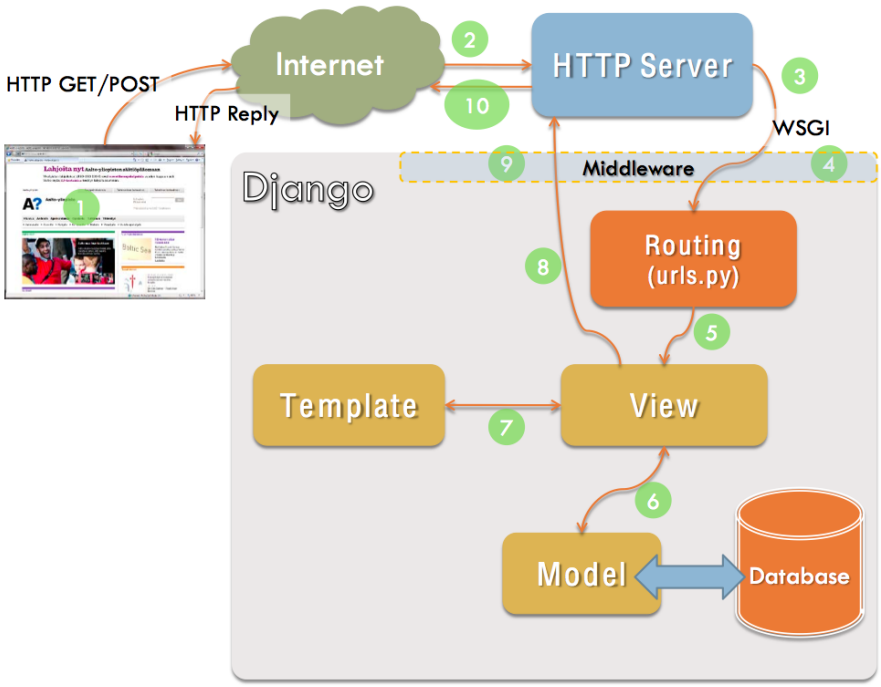
MACHINE LEARNING

Now that we have our dataset ready, we apply the SVM algorithm to the dataset, with the standard 80-20 training and testing set. Before applying the algorithm, we also scale the data for better results. Further, certain parameter adjustments can be made to improve the performance. The output of the entire procedure for a ticker is a predicted label, saying if the stock will underperform or outperform.

API INTEGRATION

After all the three analysis are performed, we developed a Graphical User Interface(GUI) needs to be developed which presents the functionality of the project to the user.

The web framework used for integrating the Application Programming Interface(API) with fundamental, sentimental and technical analysis is Django API.



Introduction to Django:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Prerequisites for Django:

- Python
- Pip Tool
- Setting up Virtual Environment.
- Setting up the Developer Environment

Steps Involved in making the API:

The integration of the analysis of functions involves of three basic steps:

1. Creating App :

The first and foremost step of a Django Project is creating the app. App in a django project is the basic unit of representing functionalities of a project. The website which appears in the front-end of the project uses app as a basic element.

In Stock Analysis, a single app is created which provides a combined result of sentimental, technical and fundamental Analysis. Name of the app is visible in the url of the browser.

2. URL Mapping:

Now, as the view is created. Mapping of the url to our custom html page is done using urls.py file present in the main project.

3. Integrating Functions:

- Sentimental Analysis:

  Ticker of the company will be passed as a parameter to the function. The output of the

functions will contain a score(negative or positive) based on several factors.

News and Twitter analysis is displayed to the user in terms of Polarity, Negativity and Positivity.

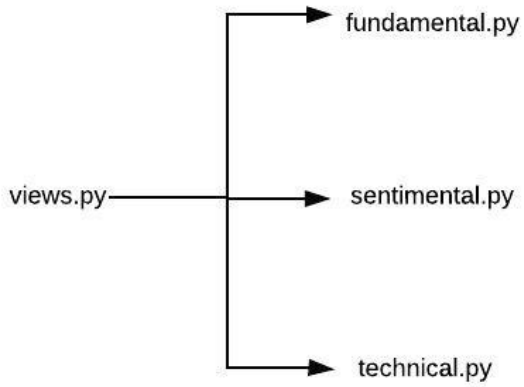
- Fundamental Analysis:

  The outcome of fundamental Analysis is to determine whether the stock has overperformed or underperformed. Following are the results displayed to the user:

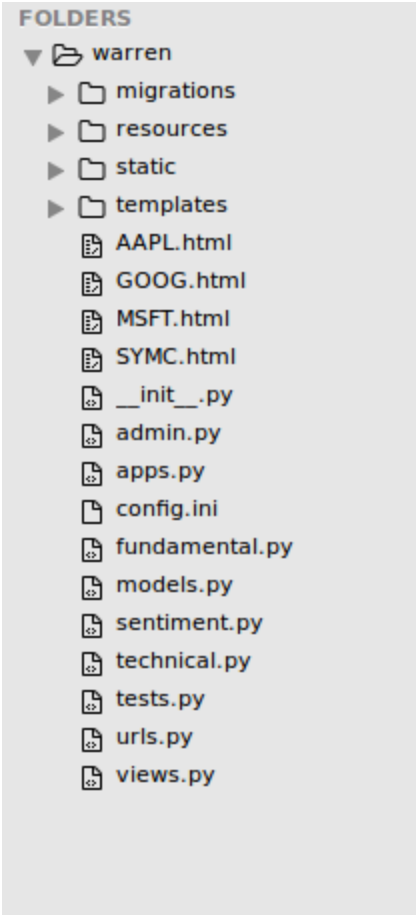
  - ❑ Training Model Accuracy
  - ❑ Testing Model Accuracy
  - ❑ Predicted Label
  - ❑ Probability of Prediction
- Technical Analysis:
  - ❑ Training Model Accuracy
  - ❑ Testing Model Accuracy

4.Segregating Functions:

Now, since the functions for Technical, Fundamental and Sentimental Analysis are already integrated, the directory structure of the project can be enhanced by segregating the functionalities into different individual files.

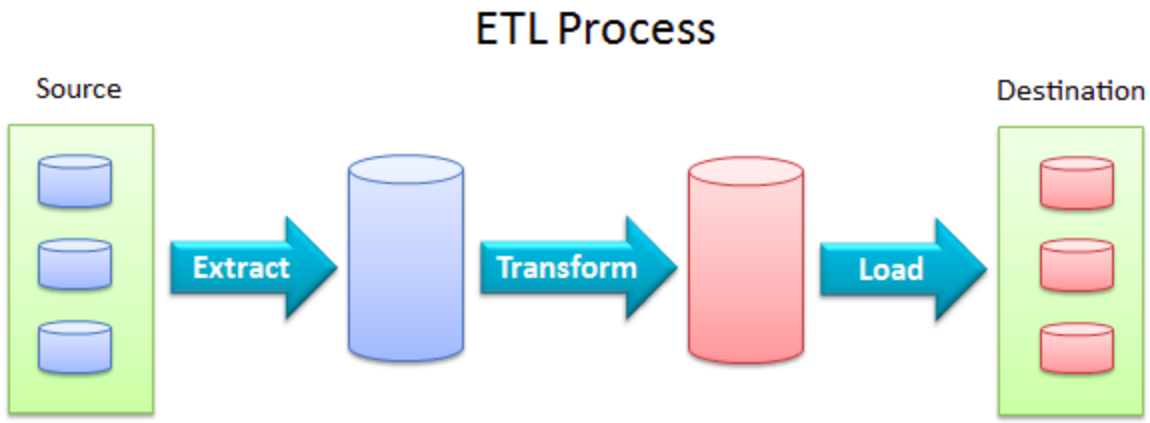


5. Directory Structure:

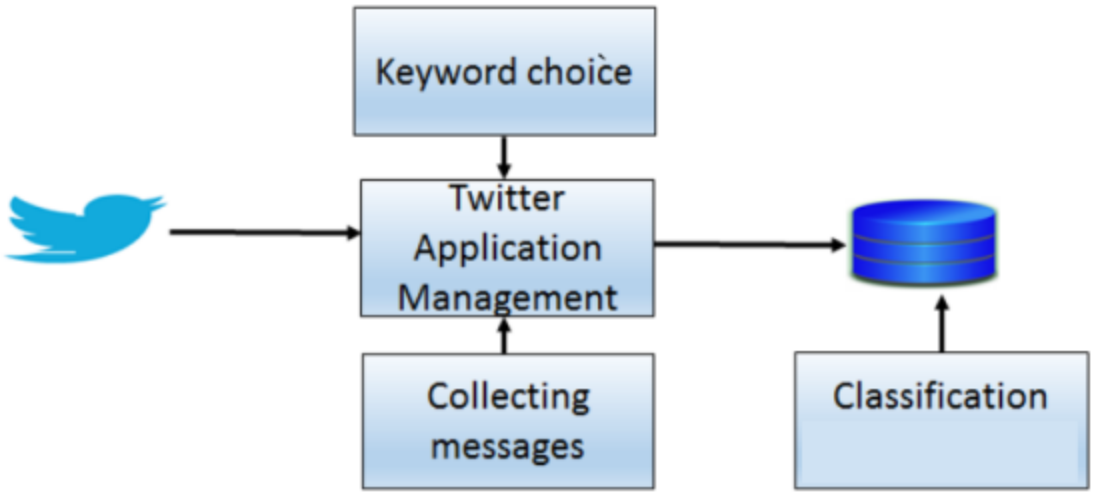


7.3 PROCESS FLOW

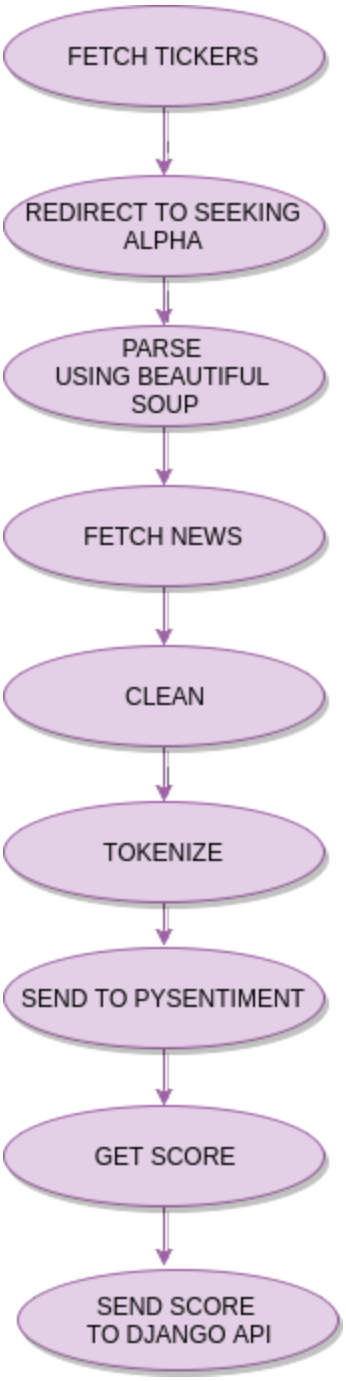
ETL PROCESSING



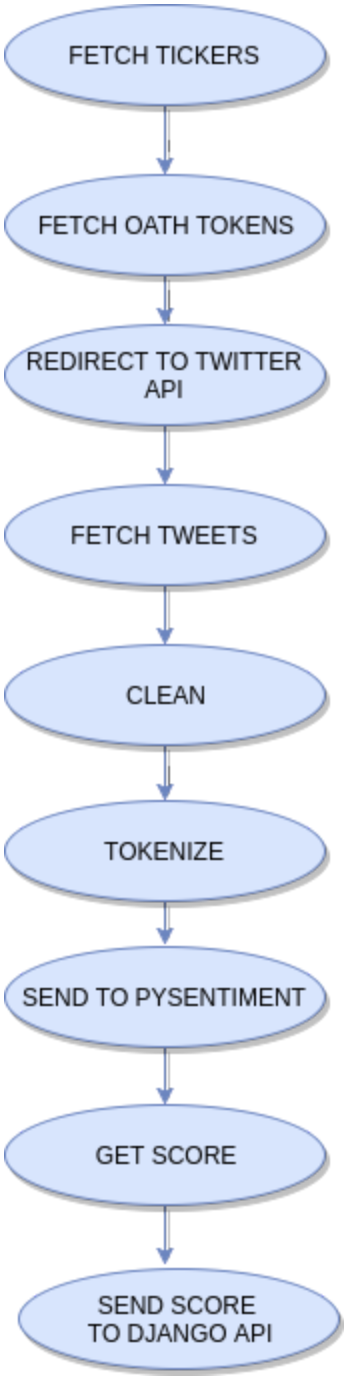
TWEETS



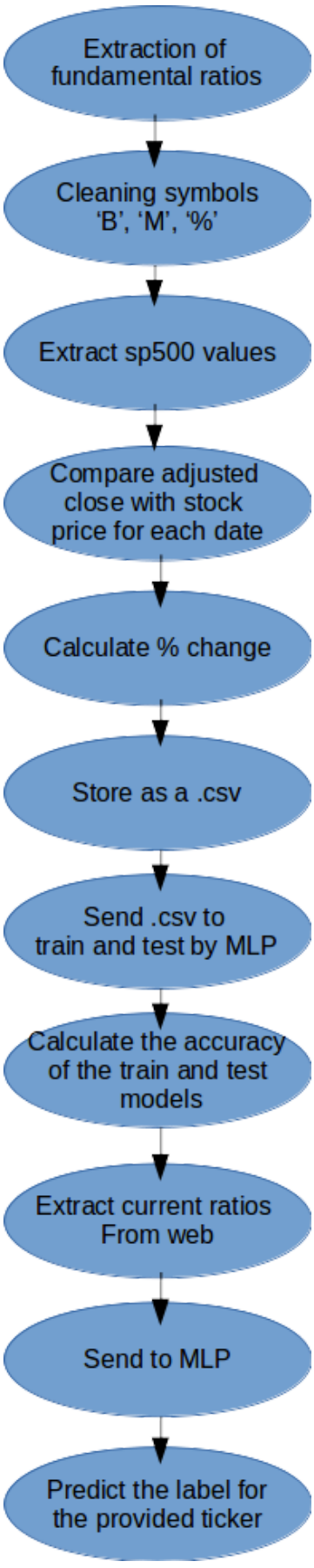
Process FLow of Tweets



NEWS


FETCH TICKERS
FETCH OATH TOKENS
REDIRECT TO TWITTER API
FETCH TWEETS
CLEAN
TOKENIZE
SEND TO PYSENTIMENT
GET SCORE
SEND SCORE TO DJANGO API

FUNDAMENTAL ANALYSIS



7.4. SOFTWARE SYSTEM ATTRIBUTES

7.4.1. Reliability

The whole reliability of our system depends on :

- sentiment analysis of the extracted tweets
- Fundamental ratios of the tickers from yahoo finance
- Technical data of the company

7.4.2. Availability

The application is available 24*7 to the user but internet connectivity is a requirement for availability.

7.4.3. Security

PYTHON is a secured language and neither the application nor the user requires much security in our case.

7.4.4. Maintainability

There is not much maintenance required. The application can be debugged, modified or upgraded easily.

8. RESULTS AND ANALYSIS

8.1 Codes

1 Sentimental Analysis

```
from datetime import datetime, timedelta

import pickle

import pysentiment as ps

import tweepy

import time

import re

import ConfigParser

from selenium import webdriver

from selenium.webdriver.firefox.options import Options

from bs4 import BeautifulSoup

def get_config(config_file_name):

  # Read config file and return config object

  options = ConfigParser.ConfigParser()

  options.read(config_file_name)

  return options

def get_twitter_sentiment(ticker, company):

  # fetch oath tokens from config.ini to secure them

  conf = get_config('config.ini')
```

```
    ckey = conf.get('twitter', 'ckey')
    csecret = conf.get('twitter', 'csecret')
    atoken = conf.get('twitter', 'atoken')
    asecret = conf.get('twitter', 'asecret')
    auth = tweepy.OAuthHandler(ckey, csecret)
    auth.set_access_token(atoken, asecret)
    api = tweepy.API(auth)


    hiv4 = ps.HIV4()
    ss = '!!'  # cleaning mark
    t = ""
    s = ''
    d = datetime.today()
    d7 = d - timedelta(days=7)
    d = d.strftime("%Y-%m-%d")
    d7 = d7.strftime("%Y-%m-%d")
    try:
        for tweet in tweepy.Cursor(api.search, q=company, since=str(d7), until=str(d),
lang="en").items():
            s = ss + tweet.text
            print(s)
            # cleaning the tweets
            s = s.replace(ss + 'RT ', '')
```

```
            result = re.sub(r"http\S+", "", s)
            # http  matches literal characters
            # \S+ matches all non-whitespace characters (the end of the url)
            t = t + result + '\n'
        except Exception as e:
            time.sleep(10)
            try:
                for tweet in tweepy.Cursor(api.search, q=ticker, since=d7, until=d, lang="en").items():
                    s = ss + tweet.text
                    print(s)
                    # cleaning the tweets
                    s = s.replace(ss + 'RT ', '')
                    result = re.sub(r"http\S+", "", s)
                    # http  matches literal characters
                    # \S+ matches all non-whitespace characters (the end of the url)
                    t = t + result + '\n'
            except Exception as e:
                time.sleep(10)
        tokens = hiv4.tokenize(t)
        score = hiv4.get_score(tokens)
        return score
    # print(get_twitter_sentiment('aapl', 'apple'))
```

```
def get_news_sentiment(ticker):
  ticker = ticker.upper()
  hiv4 = ps.HIV4()
  # -- Setup
  options = Options()
  options.add_argument("--headless")
  browser = webdriver.Firefox(firefox_options=options)
  # -- Parse
  #browser.get("https://seekingalpha.com/symbol/" + ticker +
"/analysis-and-news?analysis_tab=focus&news_tab=news-all")
  browser.get("https://simulationstock.000webhostapp.com/MSFT.html")
  soup = BeautifulSoup(browser.page_source, "html5lib")
  x = ''
  for div_tag in soup.find_all('div', attrs={"class": "mc_list_texting right bullets"}, limit=7):
    for span_tag in div_tag.find_all('span', attrs={"class": "general_summary light_text
bullets"}):
      x = x + span_tag.text
  print(x)
  tokens = hiv4.tokenize(x)
  score = hiv4.get_score(tokens)
  browser.close()
  return score
score = get_news_sentiment('msft')
```

```
print(score)

ratio = float(float(score['Positive'])/ float(score['Negative']))

print('\n SENTIMENT RATIO : ' + str(ratio))
```

2 Fundamental Analysis

Fundamental_analysis.py

```
import re

import numpy as np

import pandas as pd

import sklearn

from sklearn import svm

from sklearn.neural_network import MLPClassifier

from sklearn.preprocessing import StandardScaler

import logging

import ConfigParser

import os

import pickle

from datetime import datetime

import requests

import prepare_dataset

import scipy

import lxml

import html5lib
```

```
def get_config(config_file_name):
   # Read config file and return config object
   options = ConfigParser.ConfigParser()
   options.read(config_file_name)
   return options


def pickle_get(path):
   """This function loads the pickled data
   structure and returns it"""
   logging.info('loading the pickle')
   if os.path.isfile(path):
      return pickle.load(open(path, 'rb'))
   else:
      logging.exception('No pickle found')
      return []


def build_feature_label_set(data_dir):
   features = pickle_get(os.path.join(data_dir, 'useful_f'))
   data_df = pd.DataFrame.from_csv(os.path.join(data_dir, 'key_stats_acc_perf_NO_NA.csv'))
   data_df = data_df.reindex(np.random.permutation(data_df.index))
   # randomizing the data frame so that the trained set is shuffled.
```

```
    X = np.array(data_df[features].values)
    y = (data_df["Status"]
         .replace("underperform", 0)
         .replace("outperform", 1)
         .values.tolist())
    return X, y


def fit_algo(data_dir):
    X,y = build_feature_label_set(data_dir)
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X,  y, test_size=0.2,
random_state=33)
    #clf = svm.SVC(kernel="linear", C=1.0)
    clf = MLPClassifier(hidden_layer_sizes=1500, alpha=0.03, max_iter=6000,
random_state=42)
    scaler = StandardScaler()
    #scaling the features to increase the accuracy
    x_train_scaled = scaler.fit(x_train).transform(x_train)
    x_test_scaled = scaler.fit(x_test).transform(x_test)
    model = clf.fit(x_train_scaled, y_train)
    print(clf.score(x_train_scaled, y_train))
    print(clf.score(x_test_scaled, y_test))
    return model, clf
```

```
def extract(url, ticker, data_dir):
    page = requests.get(url + ticker.upper() + '/key-statistics?p=' + ticker.upper()).text
    gather = pickle_get(os.path.join(data_dir, 'ratios'))
    tables = pd.read_html(page)
    value_list = []
    for each_data in gather:
        value = 'N/A'
        try:
            if each_data == 'Total Debt/Equity':
                value = tables[5].at[3, 1]
            elif each_data == 'Trailing P/E':
                value = tables[0].at[2, 1]
            elif each_data == 'Price/Sales':
                value = tables[0].at[5, 1]
            elif each_data == 'Price/Book':
                value = tables[0].at[6, 1]
            elif each_data == 'Profit Margin':
                value = tables[2].at[0, 1]
            elif each_data == 'Operating Margin':
                value = tables[2].at[1, 1]
            elif each_data == 'Return on Assets':
                value = tables[3].at[0, 1]
```

```
elif each_data == 'Return on Equity':
    value = tables[3].at[1, 1]
elif each_data == 'Revenue Per Share':
    value = tables[4].at[1, 1]
elif each_data == 'Market Cap':
    value = tables[0].at[0, 1]
elif each_data == 'Enterprise Value':
    value = tables[0].at[1, 1]
elif each_data == 'Forward P/E':
    value = tables[0].at[3, 1]
elif each_data == 'PEG Ratio':
    value = tables[0].at[4, 1]
elif each_data == 'Enterprise Value/Revenue':
    value = tables[0].at[7, 1]
elif each_data == 'Enterprise Value/EBITDA':
    value = tables[0].at[8, 1]
elif each_data == 'Revenue':
    value = tables[4].at[0, 1]
elif each_data == 'Gross Profit':
    value = tables[4].at[3, 1]
elif each_data == 'EBITDA':
    value = tables[0].at[8, 1]
```

```
elif each_data == 'Net Income Avl to Common ':
    value = tables[4].at[5, 1]
elif each_data == 'Diluted EPS':
    value = tables[4].at[3, 1]
elif each_data == 'Earnings Growth':
    value = tables[4].at[7, 1]
elif each_data == 'Revenue Growth':
    value = tables[4].at[2, 1]
elif each_data == 'Total Cash':
    value = tables[5].at[0, 1]
elif each_data == 'Total Cash Per Share':
    value = tables[5].at[1, 1]
elif each_data == 'Total Debt':
    value = tables[5].at[2, 1]
elif each_data == 'Current Ratio':
    value = tables[5].at[4, 1]
elif each_data == 'Book Value Per Share':
    value = tables[5].at[5, 1]
elif each_data == 'Cash Flow':
    value = tables[6].at[0, 1]
elif each_data == 'Beta':
    value = tables[7].at[0, 1]
```

```
            elif each_data == 'Held by Insiders':
                value = tables[8].at[4, 1]
            elif each_data == 'Held by Institutions':
                value = tables[8].at[5, 1]
            elif each_data == 'Shares Short (as of':
                value = tables[8].at[6, 1]
            elif each_data == 'Short Ratio':
                value = tables[8].at[7, 1]
            elif each_data == 'Short % of Float':
                value = tables[8].at[8, 1]
            elif each_data == 'Shares Short (prior ':
                value = tables[8].at[9, 1]
            if "B" in value:
                value = float(value.replace("B", '')) * 1000000000
            elif "M" in value:
                value = float(value.replace("M", '')) * 1000000
            elif "%" in value:
                value = float(value.replace("%", ''))
            value_list.append(value)
        except Exception as e:
            value = "N/A"
            value_list.append(value)
```

```
    return value_list


def predict(url, ticker, data_dir):
    value_list = extract(url, ticker, data_dir)
    df = pd.DataFrame(columns=pickle_get(os.path.join(data_dir, 'useful_f')))
    df = df.append({
                'DE Ratio': value_list[0],
                'Trailing P/E': value_list[1],
                'Price/Sales': value_list[2],
                'Price/Book': value_list[3],
                'Profit Margin': value_list[4],
                'Operating Margin': value_list[5],
                'Return on Assets': value_list[6],
                'Return on Equity': value_list[7],
                'Revenue Per Share': value_list[8],
                'Market Cap': value_list[9],
                'Enterprise Value': value_list[10],
                'Forward P/E': value_list[11],
                'PEG Ratio': value_list[12],
                'Enterprise Value/Revenue': value_list[13],
                'Enterprise Value/EBITDA': value_list[14],
                'Revenue': value_list[15],
```

```
            'Gross Profit': value_list[16],

            'EBITDA': value_list[17],

            'Net Income Avl to Common ': value_list[18],

            'Diluted EPS': value_list[19],

            'Earnings Growth': value_list[20],

            'Revenue Growth': value_list[21],

            'Total Cash': value_list[22],

            'Total Cash Per Share': value_list[23],

            'Total Debt': value_list[24],

            'Current Ratio': value_list[25],

            'Book Value Per Share': value_list[26],

            'Cash Flow': value_list[27],

            'Beta': value_list[28],

            'Held by Insiders': value_list[29],

            'Held by Institutions': value_list[30],

            'Shares Short (as of': value_list[31],

            'Short Ratio': value_list[32],

            'Short % of Float': value_list[33],

            'Shares Short (prior ': value_list[34],

            },

           ignore_index=True)

X = np.array(df.values)
```

```
    model, clf = fit_algo(data_dir)
    print(clf.classes_)
    print(model.predict(X))
    print(model.predict_proba(X))


def main():
    project_path = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    conf_rel = os.path.join(project_path, 'etc', 'config.ini')
    conf = get_config(conf_rel)
    var_dir = os.path.join(project_path, 'var')
    data_dir = os.path.join(var_dir, 'data')
    url =conf.get('fundamental', 'url')
    date = datetime.now().strftime('%Y_%m_%d')
    log_file = os.path.join(var_dir, 'log', 'fundamental_analysis' + date)
    logging.basicConfig(filename=log_file, level=logging.DEBUG)
predict(url, 'AAPL', data_dir)
if __name__ == "__main__":  # pragma: no cover
    main()
```

3 Technical Analysis

```
import requests
import sklearn
from sklearn import svm
```

```
from sklearn.neural_network import MLPClassifier

import numpy as np

from sklearn.preprocessing import StandardScaler

import logging

import ConfigParser

import os

from datetime import datetime


def get_config(config_file_name):

  # Read config file and return config object

  options = ConfigParser.ConfigParser()

  options.read(config_file_name)

  return options


def prepare_dataset(base_url, api_key, ticker):

  url = base_url + 'function=TIME_SERIES_DAILY&symbol=' + ticker.upper() +
'&outputsize=full&apikey=' + api_key

  package_json = requests.get(url).json()

  dates = list(package_json['Time Series (Daily)'].keys())

  l = package_json['Time Series (Daily)']


  url_s = base_url + 'function=SMA&symbol=' + ticker.upper() +
'&interval=daily&time_period=50&series_type=close&apikey=' + api_key
```

```
  package_json_s = requests.get(url_s).json()


  url_e = base_url + 'function=EMA&symbol=' + ticker.upper() +
'&interval=daily&time_period=50&series_type=close&apikey=' + api_key

  package_json_e = requests.get(url_e).json()


  url_m = base_url + 'function=MACD&symbol=' + ticker.upper() +
'&interval=daily&series_type=close&apikey=' + api_key

  package_json_m = requests.get(url_m).json()


  url_st = base_url + 'function=STOCH&symbol=' + ticker.upper() + '&interval=daily&apikey=' +
api_key

  package_json_st = requests.get(url_st).json()


  url_r = base_url + 'function=RSI&symbol=' + ticker.upper() +
'&interval=daily&time_period=40&series_type=close&apikey=' + api_key

  package_json_r = requests.get(url_r).json()


  url_a = base_url + 'function=ADX&symbol=' + ticker.upper() +
'&interval=daily&time_period=14&apikey=' + api_key

  package_json_a = requests.get(url_a).json()


  url_c = base_url + 'function=CCI&symbol=' + ticker.upper() +
'&interval=daily&time_period=20&apikey=' + api_key

  package_json_c = requests.get(url_c).json()
```

```
  url_ar = base_url + 'function=AROON&symbol=' + ticker.upper() +
'&interval=daily&time_period=25&apikey=' + api_key

  package_json_ar = requests.get(url_ar).json()


  url_b = base_url + 'function=BBANDS&symbol=' + ticker.upper() +
'&interval=daily&time_period=20&series_type=close&apikey=' + api_key

  package_json_b = requests.get(url_b).json()


  url_ad = base_url + 'function=AD&symbol=' + ticker.upper() +
'&interval=daily&time_period=50&series_type=close&apikey=' + api_key

  package_json_ad = requests.get(url_ad).json()


  url_o = base_url + 'function=OBV&symbol=' + ticker.upper() + '&interval=daily&apikey=' +
api_key

  package_json_o = requests.get(url_o).json()

  labels = []

  arr = []

  for i in range(len(dates) - 1):

    ts = dates[i + 1]


    gap = float(l[dates[i]]['1. open']) - float(l[ts]['4. close'])

    high = float(l[ts]['2. high'])

    low = float(l[ts]['3. low'])
```

```
        volume = float(l[ts]['5. volume'])
        sma = get_sma(package_json_s, ts)
        ema = get_ema(package_json_e, ts)
        macd = get_macd(package_json_m, ts)
        stoch1, stoch2 = get_stoch(package_json_st, ts)
        rsi = get_rsi(package_json_r, ts)
        adx = get_adx(package_json_a, ts)
        aroon1, aroon2 = get_aroon(package_json_ar, ts)
        bbands1, bbands2, bbands3 = get_bbands(package_json_b, ts)
        ad = get_ad(package_json_ad, ts)
        obv = get_obv(package_json_o, ts)
        cci = get_cci(package_json_c, ts)
        temp = [gap, high, low, volume, sma, ema, macd, stoch1, stoch2, rsi, adx, cci, aroon1,
aroon2, bbands1, bbands2, bbands3, ad, obv]
        arr.append(temp)
        if float(l[dates[i]]['1. open']) >= float(l[dates[i]]['4. close']):
            labels.append(0)
        else:
            labels.append(1)
    X = np.array(arr)
    Y = labels
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X, Y, test_size=0.2,
random_state=73)
```

```
    scaler = StandardScaler()
    x_train_scaled = scaler.fit(x_train).transform(x_train)
    x_test_scaled = scaler.fit(x_test).transform(x_test)
    clf = MLPClassifier(hidden_layer_sizes=1500, alpha=1, max_iter=6000, random_state=42)
    model = clf.fit(x_train_scaled, y_train)
    print(clf.score(x_train_scaled, y_train))
    print(clf.score(x_test_scaled, y_test))


def get_sma(package_json, ts):
    l = package_json['Technical Analysis: SMA']
    if ts not in l:
        return 0.0
    return float(l[ts]['SMA'])


def get_ema(package_json, ts):
    l = package_json['Technical Analysis: EMA']
    if ts not in l:
        return 0.0
    return float(l[ts]['EMA'])
def get_macd(package_json, ts):
    l = package_json['Technical Analysis: MACD']
    if ts not in l:
```

```
        return 0.0
    return float(l[ts]['MACD'])


def get_stoch(package_json, ts):
    l = package_json['Technical Analysis: STOCH']
    if ts not in l:
        return 0.0, 0.0
    return float(l[ts]['SlowD']), float(l[ts]['SlowK'])


def get_rsi(package_json, ts):
    l = package_json['Technical Analysis: RSI']
    if ts not in l:
        return 0.0
    return float(l[ts]['RSI'])


def get_adx(package_json, ts):
    l = package_json['Technical Analysis: ADX']
    if ts not in l:
        return 0.0
    return float(l[ts]['ADX'])
def get_cci(package_json, ts):
    l = package_json['Technical Analysis: CCI']
```

```
  if ts not in l:
    return 0.0
  return float(l[ts]['CCI'])


def get_aroon(package_json, ts):
  l = package_json['Technical Analysis: AROON']
  if ts not in l:
    return 0.0, 0.0
  return float(l[ts]['Aroon Up']), float(l[ts]['Aroon Down'])


def get_bbands(package_json, ts):
  l = package_json['Technical Analysis: BBANDS']
  if ts not in l:
    return 0.0, 0.0, 0.0
  return float(l[ts]['Real Upper Band']), float(l[ts]['Real Lower Band']), float(l[ts]['Real Middle
Band'])


def get_ad(package_json, ts):
  l = package_json['Technical Analysis: Chaikin A/D']
  if ts not in l:
    return 0.0
  return float(l[ts]['Chaikin A/D'])
```

```
def get_obv(package_json, ts):
    l = package_json['Technical Analysis: OBV']
    if ts not in l:
        return 0.0
    return float(l[ts]['OBV'])


def main():  # pragma: no cover
    """This function is where execution starts"""

    project_path = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    conf_rel = os.path.join(project_path, 'etc', 'config.ini')
    conf = get_config(conf_rel)
    var_dir = os.path.join(project_path, 'var')
    date = datetime.now().strftime('%Y_%m_%d')
    log_file = os.path.join(var_dir, 'log', 'technical_analysis_logfile' + date)
    logging.basicConfig(
        filename=log_file,
        level=logging.DEBUG)

    api_key = conf.get('technical', 'api_key')
    url = conf.get('technical', 'base_url')
```

```
    prepare_dataset(url, api_key, 'AAPL')


if __name__ == "__main__":  # pragma: no cover
    main()
```

config.ini

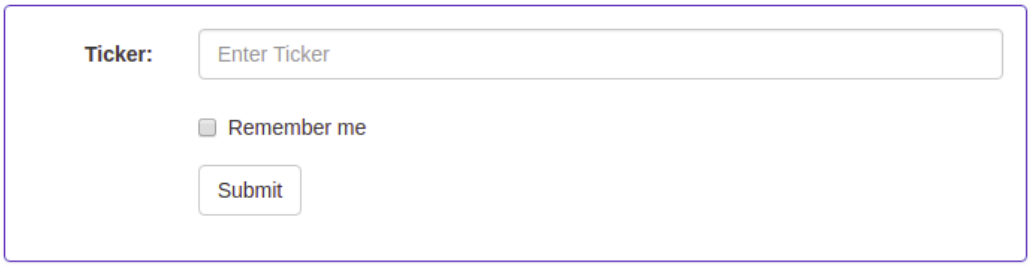
```
[twitter]
ckey=y9k1f4YMu6HmLIZhIR3aryE6K
csecret=NMMOm6zevNLt6CfnOAqK9gQOaC0Vg9Lts9sGwihkFzBtWQXlH2
atoken=828244911845580800-4BRFbk42yVRlXc0sYxCvgUJqNwjsRSL
asecret=CLArQyixUwnOWT6y7TRJwqsmvUaDxxllGy8fN8Sk2EBO9
[technical]
base_url=https://www.alphavantage.co/query?
api_key=LBQ67F59FL7NECLE
[fundamental]
url = https://in.finance.yahoo.com/quote/
```

4. API integration

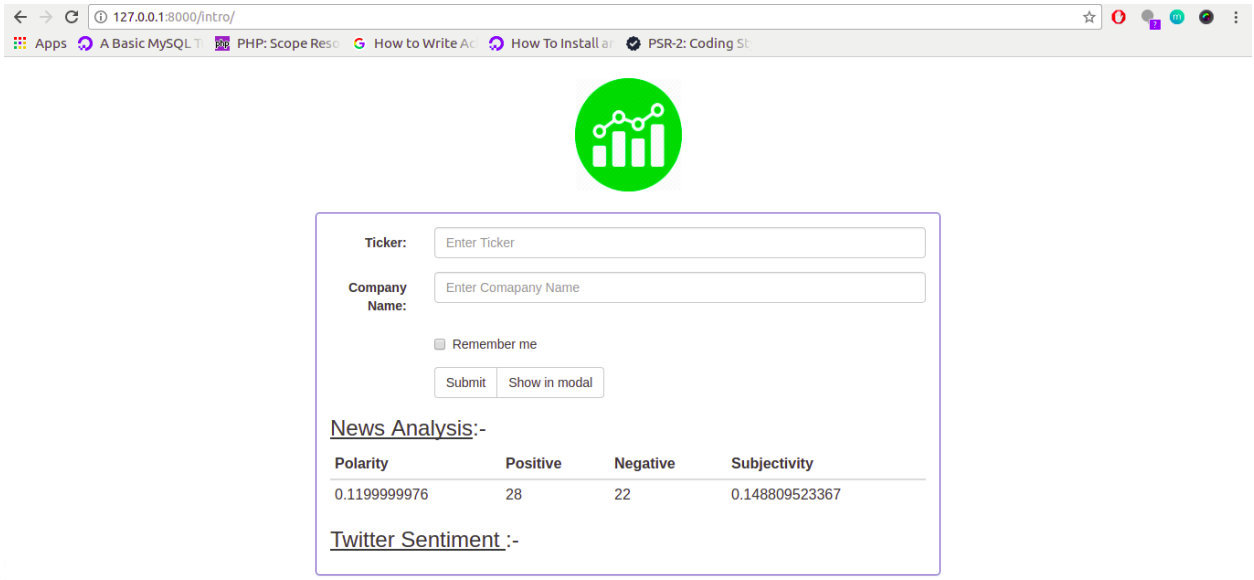
```
    project_path = os.path.dirname(os.path.abspath(__file__)) + '/resources'
    conf_rel = os.path.join(project_path, 'etc', 'config.ini')
    conf = get_config(conf_rel)
    var_dir = os.path.join(project_path, 'var')
    data_dir = os.path.join(var_dir, 'data')
    url =conf.get('fundamental', 'url')
    date = datetime.now().strftime('%Y %m %d')
    log_file = os.path.join(var_dir, 'log', 'fundamental_analysis' + date)
    logging.basicConfig(
    filename=log_file,
    level=logging.DEBUG)
    print(request.POST['tickr'])
    print('URL:' + url)
    fund = predict(url, request.POST['tickr'], data_dir)

    return render(request, 'index.html',
           context={
               'Polarity' : 'Polarity',
               'Positive' : 'Positive',
               'Negative' : 'Negative',
               'Subjectivity' : 'Subjectivity',
               'project_path' : project_path,
               'conf_path' : conf_rel,
               'url' : url,
               'fund0' : fund[0],
               'fund1' : fund[1],
               'fund2' : fund[2],
               'fund3' : fund[3]
           })

return render(request, 'index.html',
              context={'name_of_creator':'Ayush Pant'})
```

8.2 Screenshots

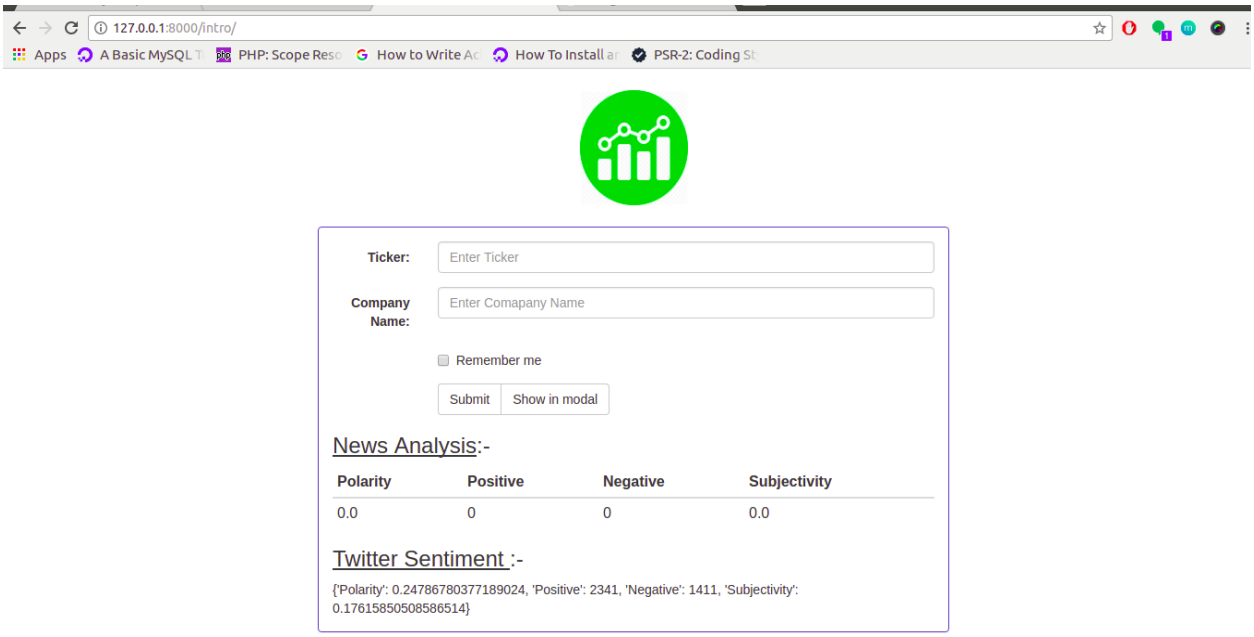
- Template:



- News Analysis:



- Twitter Analysis:



- Fundamental Analysis:

Ticker: Enter Ticker

Company Name: Enter Comapany Name

Remember me

Submit | Show in modal

News Analysis:-

Twitter Sentiment :-

0.7255794701986755
0.5728476821192053
[0 1]
[0]
[[1. 0.]]

- Technical Analysis:

News Analysis:

Twitter Sentiment:

Fundamental Analysis :

Training Model Accuracy	
Testing model Accuracy	
Predicted Label	
Probability of Prediction	

Technical Analysis :

Training Model Accuracy	0.5537459283387622
Testing model Accuracy	0.511400651465798

9 Resources And Limitations

Analysis Language: python 2.7, with modules pandas, sklearn, beautifulsoup, requests and textprocessing for sentiment analysis

Web Framework for API : Django Hardware: Linux server

10. CONCLUSION AND FUTURE ENHANCEMENTS

10.1. CONCLUSION

Development of this project titled “Warren - Stock Market Analysis‘ has been a useful experience. We gained knowledge about the API‘s, Natural Language Processing (NLP), Python programming language, Machine Learning, Pycharm IDE, Django web development API and various other technologies and platforms. We‘ve learnt new concepts of working. The project aims to demonstrate how the stocks of a company are bought and sold based upon:

- The sentiments of the people towards the company
- The fundamental features of a company
- The technical data of a company

The results of all the individual analysis are combined in a weighted manner to yield a final outcome which is demonstrated via a custom developed API designed in Django. The project was analyzed, designed, developed and deployed successfully.

10.2. FUTURE SCOPE

Prediction of stock market is an extremely complicated kind of analysis as the data is time series type. Which means the variance is immediate, within seconds. Stocks behaviour depends upon so many factors that keep changing so frequently that their prediction becomes very difficult and time consuming.

For quite some years coders have been working on many algorithms to predict the stock market however, the accuracy and performance has been only moderate.

Therefore, more analysis needs to be done to improve the accuracy of the prediction. Moreover the results obtained in this project depend highly on the real time data, bandwidth and the amount of the data. The algorithms also have quite an impact upon the accuracy. However, in stocks prediction, getting an accuracy even above fifty percent is considered good enough. We have tried our results with two algorithms and chose MLP as the better one. Perhaps, more better and faster algorithms can be applied to obtain refined and more accurate predictions.

11. GLOSSARY

- **API (Application Programming Interface):** It is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact and are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.
- **Django:** Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Ridiculously fast.
- **Fundamental Analysis:** Fundamental analysis is a method of evaluating a security to measure its intrinsic value, by examining related economic, financial and other qualitative and quantitative factors. The end goal of fundamental analysis is to produce a quantitative value that an investor can compare with a security's current price, thus indicating whether the security is undervalued or overvalued.
- **Machine Learning:** Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.
- **MLP Classifier:** A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.
- **NLP (Natural Language Processing):** It is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. NLP is related to the area of human–computer interaction.
- **Sentiments:** a view or opinion that is held or expressed.
- **Sentiment analysis:** It refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

- **Stocks:** A stock is a share in the ownership of a company. Stock represents a claim on the company's assets and earnings. As you acquire morestock, your ownership stake in the company becomes greater.
- **SVM:** “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. ... Support Vectors are simply the coordinates of individual observation.
- **Technical Analysis:** Technical analysis is a trading tool employed to evaluate securities and attempt to forecast their future movement by analysing statistics gathered from trading activity, such as price movement and volume.
- **Tickers:** A ticker symbol is an arrangement of characters (usually letters) representing a particular security listed on an exchange or otherwise traded publicly. When a company issues securities to the public marketplace, it selects an available ticker symbol for its securities that investors use to place trade orders.
- **Twitter**: It is an online social networking and micro blogging service that enables users to send and read short 140-character text messages, called "tweets". Registered users can read and post tweets, but unregistered users can only read them.

12 REFERENCES AND BIBLIOGRAPHY

PAPERS

- International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 7, No. 1, January 2016 USING SENTIMENT ANALYSIS FOR STOCK EXCHANGE PREDICTION
- Stock Market Prediction by Mark Dunne
- Pandas: a Foundational Python Library for Data Analysis and Statistics by Wes McKinney.

BOOKS

Neural Network Library project in C#

LINKS

https://finance.yahoo.com/quote/MSFT/key-statistics

https://seekingalpha.com/symbol/MSFT/financials/income-statement

https://www.alphavantage.co/documentation/

http://blog.hackerearth.com/simple-tutorial-svm-parameter-tuning-python-r

http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

https://www.investopedia.com/terms/f/fundamentalanalysis.asp

http://m.businesstoday.in/story/key-financial-ratios-analyze-company-stock-investment/1/209789.html