B.TECH. FINAL YEAR PROJECT REPORT

STOCKS ANALYSIS

SHARANG GUPTA

AISHWARYA DIXIT

AYUSH PANT

08.03.2018

PROBLEM STATEMENT:

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.

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 of at least 500 companies in the past 2 decades.

OBJECTIVE:

The 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).

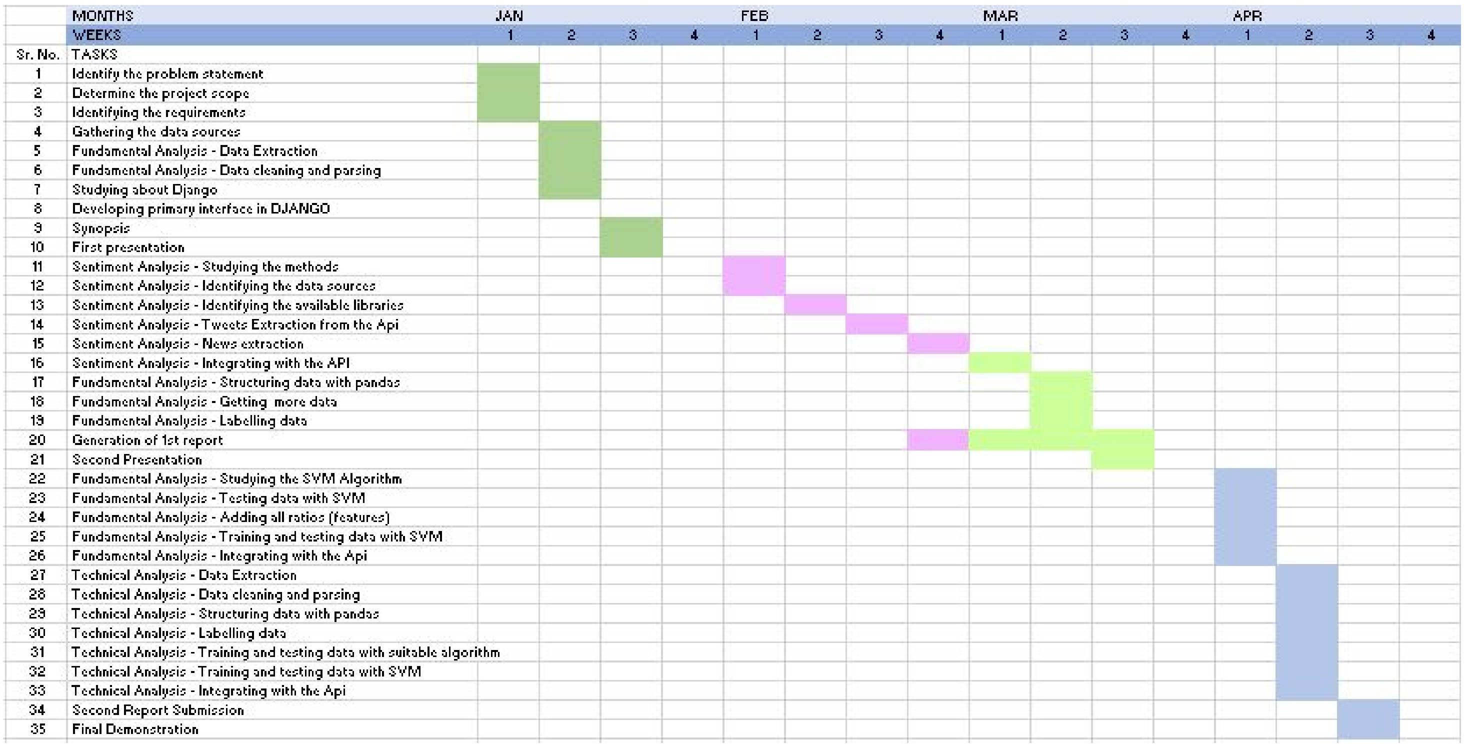
The bulk of our efforts will be directed towards gathering, cleaning and preparing the data, along with providing an overlying API.

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.

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.

GANTT CHART



	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															🟦	

PROCESS DESCRIPTION:

There are three kinds of analysis that we will be performing:

1. Sentiment Analysis

2. Technical Analysis

3. Fundamental Analysis

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.

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

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:

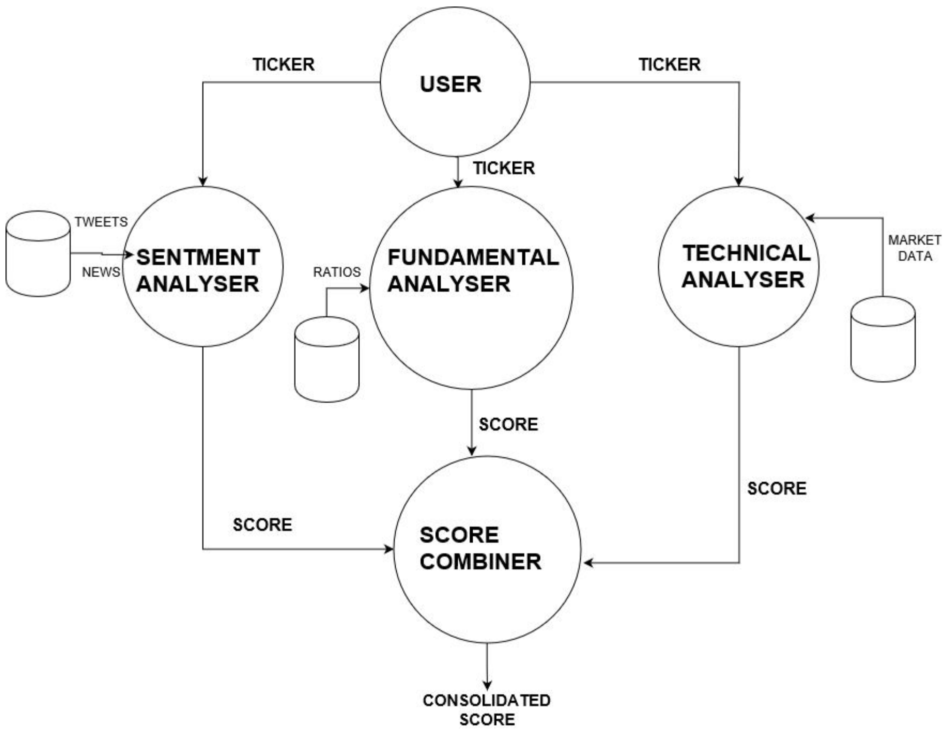
1) Market price discounts every factor that may influence a security's price and

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

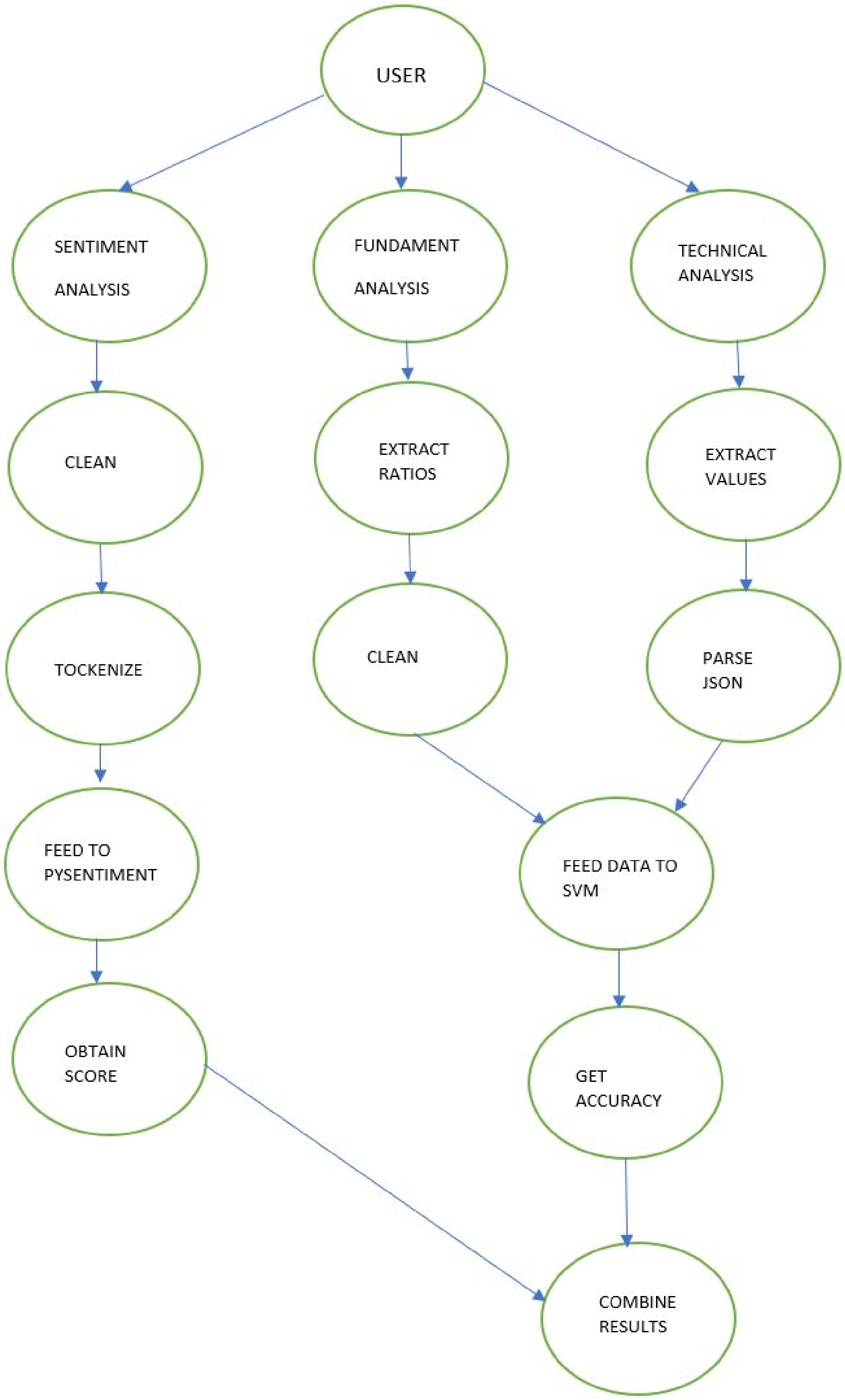
Technical analysis is used to attempt to forecast the price movement of virtually any tradable instrument that is generally subject to forces of supply and demand, including stocks, bonds, futures and currency pairs. In fact, technical analysis can be viewed as simply the study of supply and demand forces 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.

DATA FLOW DIAGRAM

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

PROCEDURE:

Considering the changes suggested in the first presentation we have prioritized sentiment analysis and started with it first.

Data sources

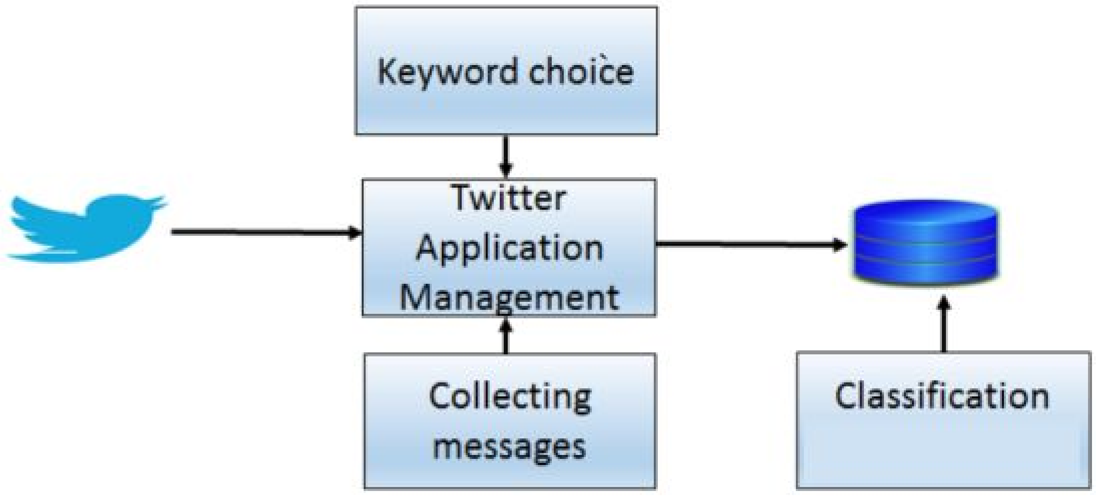
After thoroughly searching for sufficiently vast data sources based on various companies' sentiments we arrived at the following viable sources:

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).

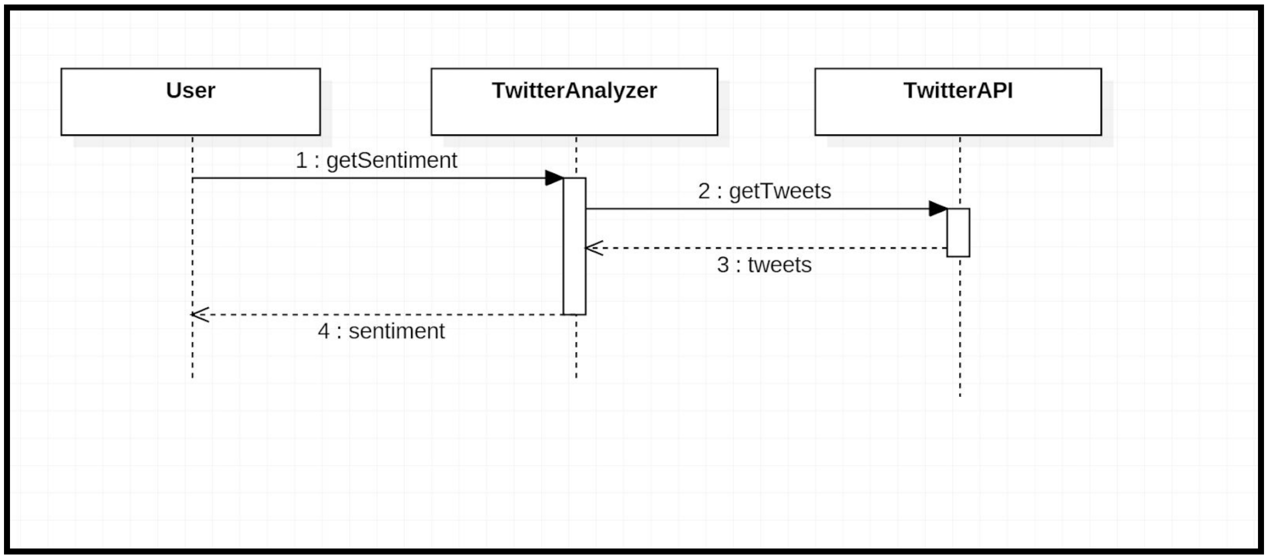
Lang

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.

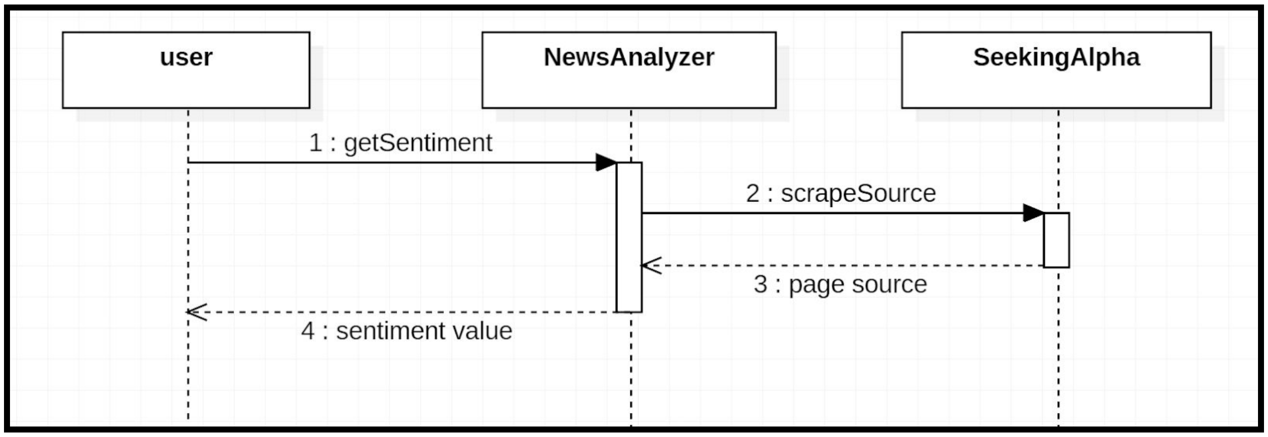
With each Tweet twitter also generates 'entity' objects, which are arrays of common Tweet contents such as hashtags, mentions, media, and links. If there are links, the JSON payload can also provide metadata such as the fully unwound URL and the webpage's title and description. So, in addition to the 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.**



FUNDAMENTAL ANALYSIS PROCESS DESCRIPTION

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

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

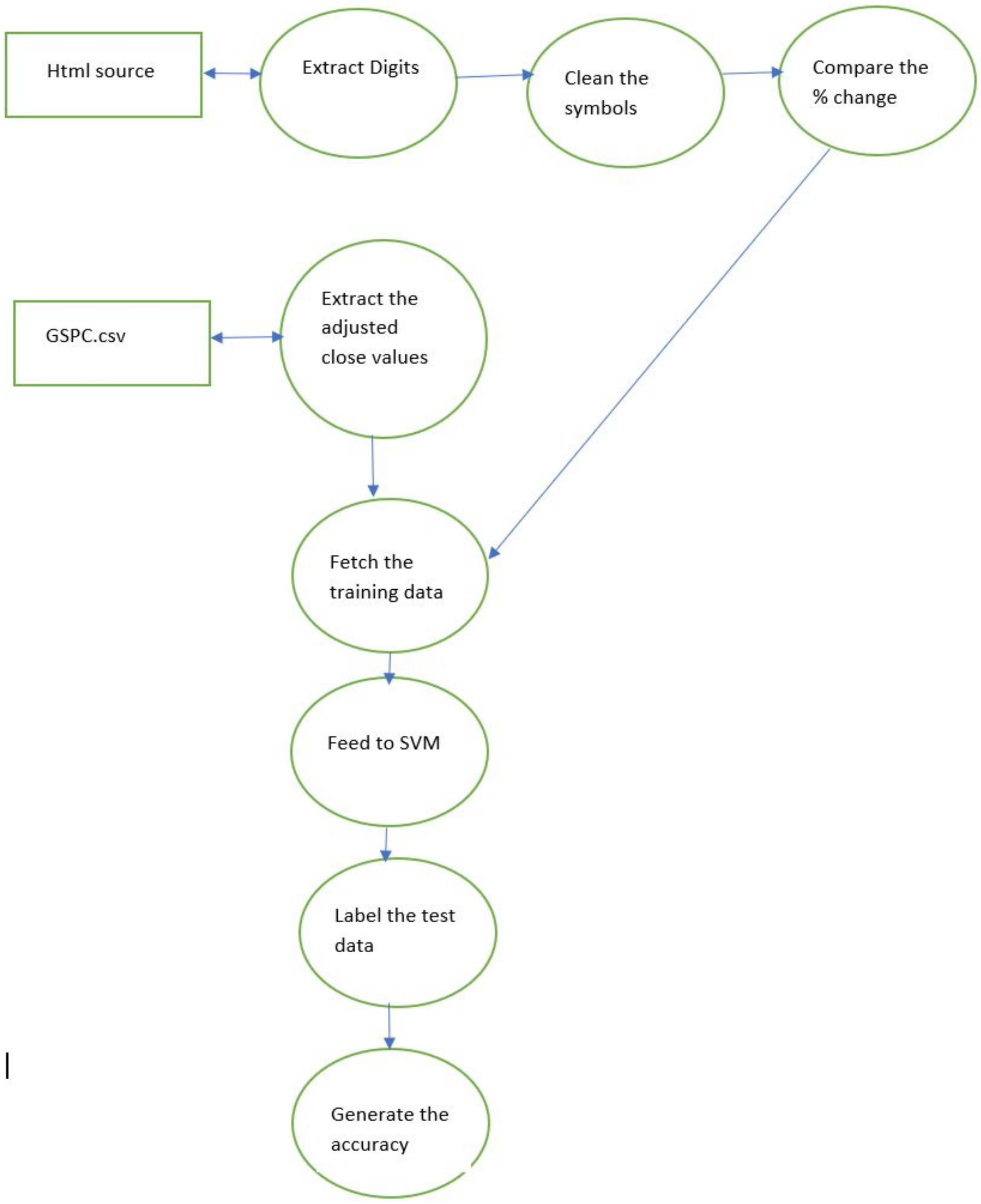
We have defined "path," which is the path to the intraQuarter folder where the data set is located. We start with collecting the Debt/Equity value.Statspath is the path to the stats directory.stock_list is a quick one-liner for loop that uses os.walk to list out all contents within a directory.

We iterate through every directory of every stock ticker. 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, then we split by the opening of the table data tag and the table data closing tag to find the required value.

With this method we have pulled the Debt/Equity ratios for all of the companies.

DFD 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

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 th to_csv function of pandas.We want to save the data since we really just need to access and store the data once. When we begin doing analysis, we're going to want to frequently modify the data, restructure it, and do all sorts of testing with it. If we were re-pulling the data every single time, it would add a significant amount of time to the overall process.

Comparing with the market (Meshing)

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.

In this case, the answer we are looking for is in fact quite ambiguous.

It is not plausible for the machine to be extremely accurate, especially in stocks prediction.Therefore, we look if the results are at least above 50%, otherwise the machine requires improvement.

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.

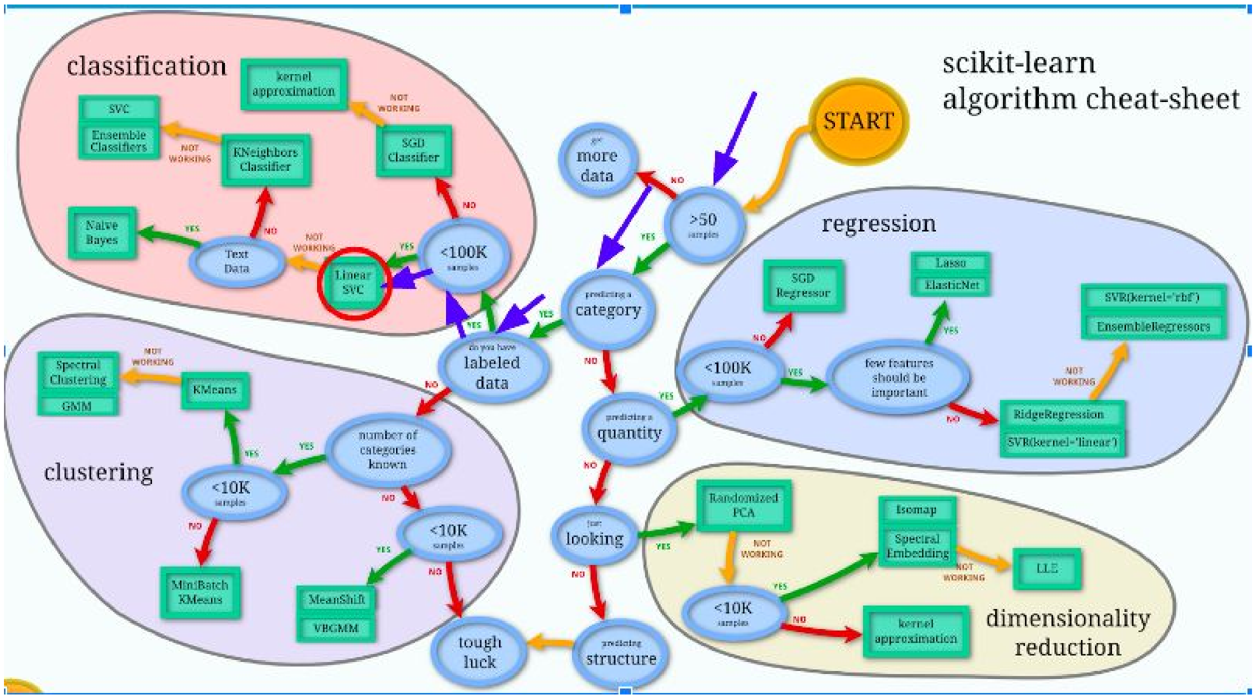
Methods:

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.

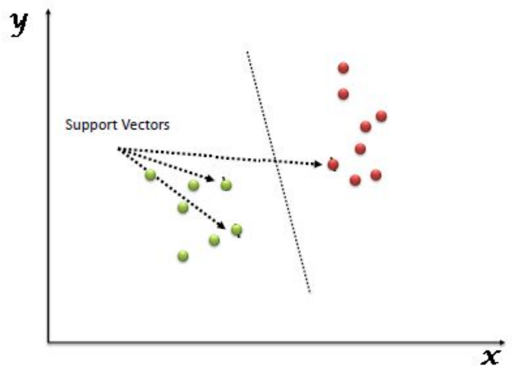
The reason for selecting SVM can be demonstrated from the scikit cheat sheet shown below:



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

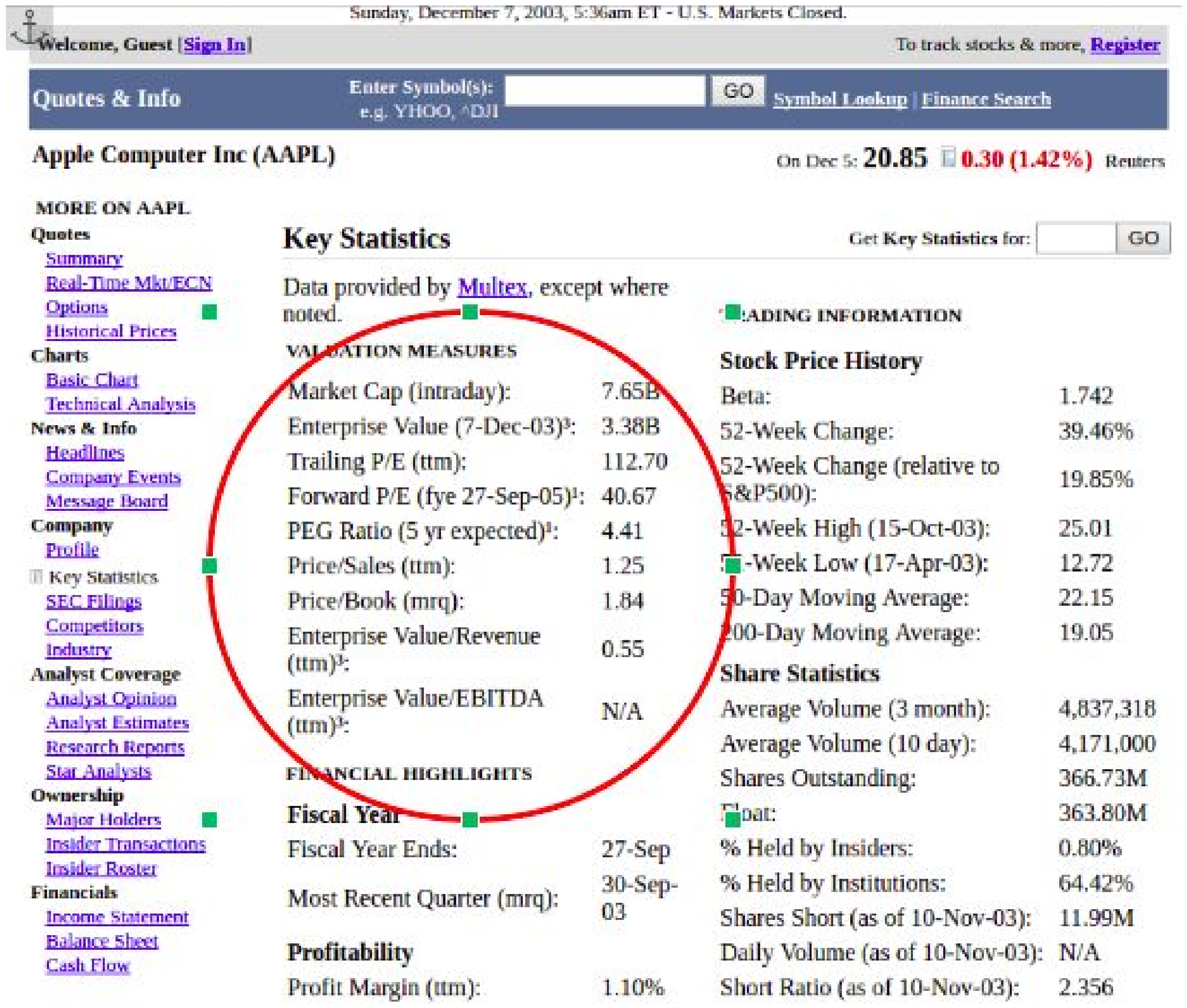
Machine is a frontier which best segregates the two classes (hyper-plane/ line).

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”.

NEXT ITERATION

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

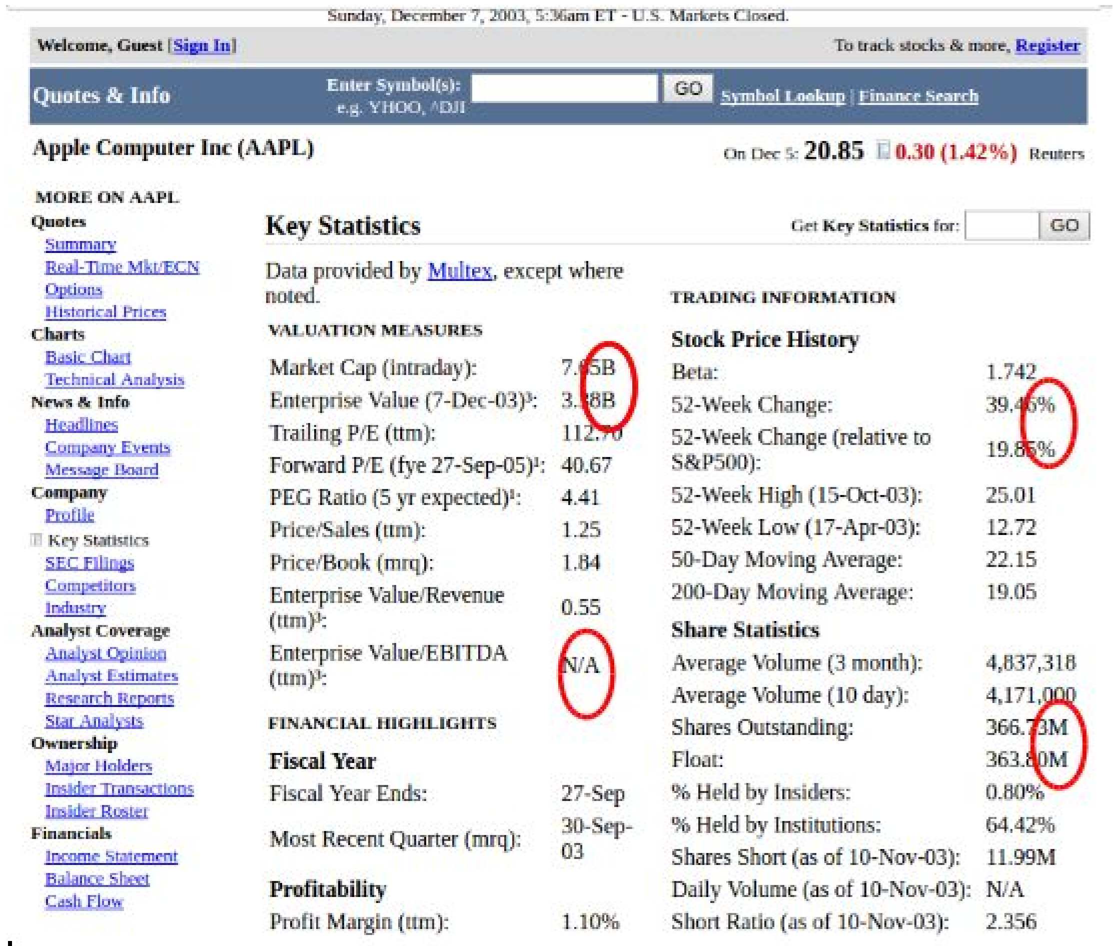
Cleaning:

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):	19.86%
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

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.

The goal is that our "loser" stocks that we pick are not not extremely bad. Consider for example:

Machine Learning Algorithm is 85% accurate, and chooses 100 stocks to invest in:

15 stocks perform an average -88% compared to market.

85 stocks perform an average +2% compared to market.

This equates to an average of -11.5% in the end compared to market.

Next, instead, consider:

Machine Learning Algorithm is 52% accurate, and chooses 100 stocks to invest in:

48 stocks perform an average of -2% compared to market.

52 stocks perform an average of +2% compared to market.

This equates to an average +8% in the end compared to market.

Which of the two above is clearly the ideal in the end? This is an exaggerated example, of course, but it illustrates a major imperative within the field of investing or trading. Many people get hyper-focused on accuracy, rather than performance. When people focus on performance, however, they tend to hyper-focus there as well, ignoring the importance of risk.

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`

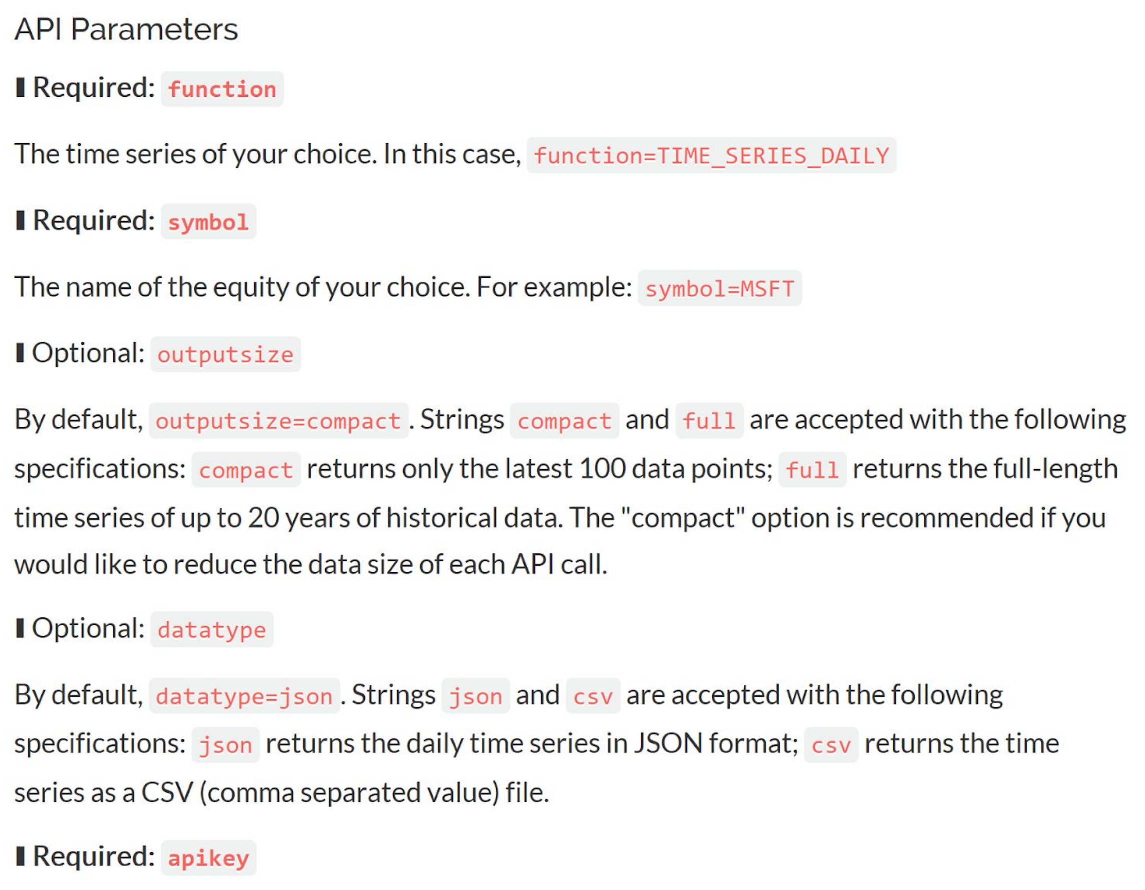
❚ Required: `interval`

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:

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.

Benefits of Django :

1. Incredibly fast.
2. Fully Loaded
3. Reassuringly Secure

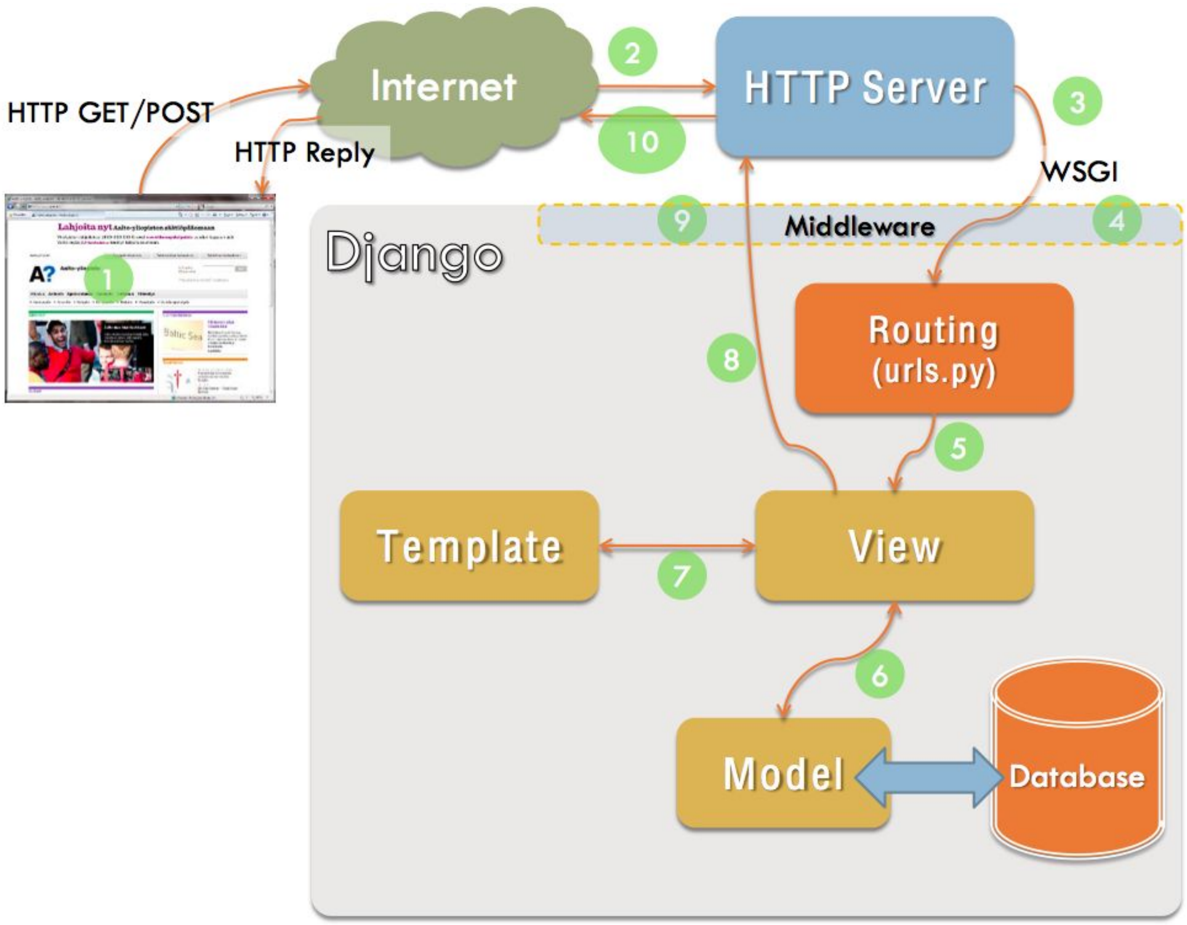
Representational State Transfer(REST):

Django Rest Framework (or simply DRF) is a powerful module for building web APIs. It's very easy to build model-backed APIs that have authentication policies and are browsable.

- **Authentication** – From basic and session-based authentication to token-based and Oauth2 capabilities, DRF is king.

- **Serialization** – It supports both ORM and Non-ORM data sources and seamlessly integrates with your database.
- **Great Documentation** – If you get stuck somewhere, you can refer to it's vast online documentation and great community support
- **Heroku, Mozilla, Red Hat, and Eventbrite** are among the top companies using DRF in their APIs.

Workflow of Django:



Introduction to Apps in Django:

Once a project is successfully created, the next step in the API is creating an app or application. 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 other words, A Django app is a group of related functionality used to complete or maintain one aspect of a site.

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.

Creating app in Django :

The command used for creating an app in django :

python manage.py startapp stock_analysis

Introduction to Views :

A view function, or *view* for short, is simply a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really. The view itself contains whatever arbitrary logic is necessary to return that response.

Example of a simple view :

```
from django.http import HttpResponse
import datetime
def current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body>It is now %s.</body></html>" % now
    return HttpResponse(html)
```

URL mapping in Django :

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.

Eg:

```
urlpatterns = patterns('',
   #Examples
   #url(r'^$', 'myproject.view.home', name = 'home'),
   #url(r'^blog/', include('blog.urls')),

   url(r'^admin', include(admin.site.urls)),
)
```

Integrating Functions:

1. **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.

API will hence generate output in JSON format.

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

Testing Technologies used : Nose for unit testing

CONCLUSION

The results of all the individual analysis will be combined in a weighted manner to yield a final outcome which will be demonstrated via a custom developed API designed in Django. The API will generate responses in JSON data format, providing either cleaned data or the analysis results, as requested by the user depending on the URL