Using Stock Market Data and News Articles Data to Predict Stock Movements

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

In

Advanced Computing and Data Science

Prepared By

Fakhre Alam

179305004

MANIPAL UNIVERSITY
JAIPUR

(In collaboration with **C-DAC**)

Computer Science & Enineering

School of Computing & Information Technology

MANIPAL UNIVERSITY JAIPUR
JAIPUR-303007
RAJASTHAN, INDIA

May – 2019

MANIPAL UNIVERSITY
JAIPUR

MANIPAL UNIVERSITY JAIPUR

CANDIDATE'S DECLARATION

I herby certify that the work is being presented in the dissertation entitled "USING STOCK MARKET DATA AND NEWS ARTICLES DATA TO PREDICT STOCK MOVEMENTS" in partial fulfilment of the requirements for the award of the Degree of Master of Advanced Computing and Data Science submitted to the Department of Computer Science and Engineering, School of Computing and Information Technology, Manipal University Jaipur is an authentic record of my own work carried out during the period from July 18 to April 19 under the supervision of Dr. Ashish Kumar and Prof. Arun Vela Department of computer science and engineering.
The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

Date: 16-05-2019 Signature

Place: Manipal University Jaipur FAKHRE ALAM

Certificate

This is to certify that the above statement made by the candidate is correct to the best of m y knowledge and belief.

Date: 16-05-2019 Signature

Place: Manipal University Jaipur Dr. Ashish Kumar

April 30, 2019

To Whomsoever It May Concern

This to certify that **Mr. Fakhre Alam** has completed his internship starting June 26, 2018 to April 30, 2019 under the guidance of Mr Vishnu Kumar Madan.

He was working on the Project: **"Subrogation stage models and automating the process by integrating it with the system"**

We wish Fakhre all the very best for his future endeavors.

On behalf of **exl Service.com (India) Private Limited**

Anjali Dargar
Lead Assistant Manager– Human Resources

exl Service.com (India) Private Limited
A-37, Sector 64, Noida 201 301, UP, India. Tel : +91.120.244.4422. Fax.: +91.120.240.4455 www.exlservice.com
Registered Office: 414, 4th Floor, DLF Jasola Tower B, Plot no. 10 & 11, DDA District Centre, Jasola, New Delhi - 110 044
CIN: U72200DL1999PTC098888

ACKNOWLEDGEMENT

I owe my gratitude and appreciation towards **Dr. Ashish Kumar** for his kind co-operation and encouragement which helped me in completion of this Dissertation.

I would also like to thank all the staff members of **Manipal University Jaipur** for their valuable assistance and support during my academia.

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents for their blessings, my friends/classmates for their help and wishes for the successful completion of this Dissertation.

- **FAKHRE ALAM**

ABSTRACT

The behavior of data today allows a shareholder at any scale to make better investment decisions. The challenge is ingesting and interpreting the data to determine which data is useful, finding the signal in this sea of information. So, the question is can we use news article content to forecast the stock price movements. This paper shows the analysis of news data to predict stock prices moments with using machine learning algorithm Extreme Gradient Boosting and Light Gradient Boosting. This unique opportunity will advance the state of research in understanding the predictive power of the news. This power if harnessed could help financial outcomes and generate significant economic impact all over the world. Here the target is to predict the 10-day market residualized returns depending upon various price movements recorded at the close and open time of the market associated with that stock using news articles.

Keywords *- Stock, Investment, News, Market close time, Market open time, Market residuals return.*

TABLE OF CONTENT

LIST OF FIGURES

TABLES

ABBREVIATIONS

Acronym	Abbreviation
ANN	Artificial Neural Network
MKL	Multiple Kernel Learning
ML	Machine Learning Algorithm
MLR	Multi Linear Regression
SVM	Support Vector Machine
TF-IDF	Term frequency–inverse document frequency

CHAPTER 1

INTRODUCTION

The stock market follows very random walk and is dynamic because of this instability the price of stock has very complex behaviour, in financial market profit opportunities are exploited as soon as they arise [18] hence there are very important need to explore this enormous amount of data of stock market [19]. Traders decision are based how a news articles is influencing the market. These financial news articles have information about organisation vision, mission, coming project, ongoing project, activity in which it is involved and expectation from the other competitor and looks for financial market information like the trading volume inflation, demand for product or services offered by the organisation opening price and closing price, calculated return etc. It has been shown in the previous research work. The fluctuation in the stock prices is strongly related to the publication of related news article and financial market information of the stock. So, for most of the research papers either it is determined the best time to buy or sell the stock or whether the stock price will increase or decrease or may be near about same. So, this research paper we are predicting the signed confidence value between (-1, +1) which will be multiplied by the market adjusted value of the given stock over a 10-day market residualized return value.

1.1 Signed Confidence Value

- If we expect stock to have large positive return compared to the broad market over the next 10 days, we might assign it is large positive confidence value i.e. near (1.0).
- If we are expecting stock to have a negative return, we might assign it is large negative confidence value i.e. near (-1.0).
- If we are insuring, we might assign value zero then the confidence value zero.

The output of this research will be a model which will be used for short term prediction. Different forecasting methodologies will be applied each methodology has its own important features and is applied to specific dataset prepared after employing various data mining techniques on two raw datasets (Market data & News data).

Market Data: Contains financial information such as opening price, closing price, trading values etc.

News Data: Contain information about news articles/alert published about stocks such as article details, sentiment and other commentary.

In this model different forecasting techniques that are employed are Statistical, Regression and predictive.

1.12 Residualized Return Value

Return means investment return. Example percentage return of stock price Market and residualized means a stock's excess return over a benchmark. Example Apple return today is 5%, and S&P 500 return is 3%, the residualized return for Apple using S&P 500 is 5-3 = 2%.

1.2 Stock Market

A stock market or equity market is the aggregation of buyers and sellers of stocks (also called shares); these may include securities listed on a stock exchange as well as those only traded privately. Stocks can be categorised in various ways. One common way is by the country where the company is domiciled. For example, Nestlé and Novartis are domiciled in Switzerland, so they may be considered as part of the Swiss stock market, although their stock may also be traded at exchanges in other countries. At the close of 2012, the size of the world stock market (total market capitalisation) was about US$55 trillion. By country, the largest market was the United States (about 34%), followed by Japan (about 6%) and the United Kingdom (about 6%). This went up more in 2013. A stock exchange is a place or organisation by which stock traders (people and companies) can trade stocks. Companies may want to get their stock listed on a stock exchange. Other stocks may be traded "over the counter", that is, through a dealer. A large company will usually have its stock listed on many exchanges across the world. Exchanges may also cover other types of security such as fixed interest securities or interest derivatives. Trade in stock markets means the transfer for money of a stock or security from a seller to a buyer. This requires these two parties to agree on a price. Equities (stocks or shares) confer an ownership interest in a company. Participants in the stock market range from small individual stock investors to larger traders investors, who can be based anywhere in the world, and may include banks, insurance companies or pension funds, and hedge funds. Their buy or sell orders may be executed on their behalf by a stock exchange trader. An example of such an exchange is the New York Stock Exchange. The other type of stock exchange is a virtual kind, composed of a network of computers where trades are made electronically by traders. An example of such an exchange is the NASDAQ [15].

1.3 Text Mining

Text mining is considered a set of methodologies to derive useful information from text content. For this purpose, it is necessary to transform unstructured text content into a structured format readable by other algorithms. Text mining is derived from data mining research started during the 80s. It is considered a multidisciplinary field that involves information retrieval, natural language processing, data mining, statistics and linguistics.

The main activities of text mining are entity extraction, taxonomy extraction, sentiment analysis, document summarization, text categorisation, text clustering, entity relationship, and visualisation. Most of these activities rely on data mining algorithms, but these algorithms are not able to deal directly with unstructured data, as they need a structured format, generally in a matrix shape [2].

A text mining system normally has the architecture described in figure to be explained the following sections.

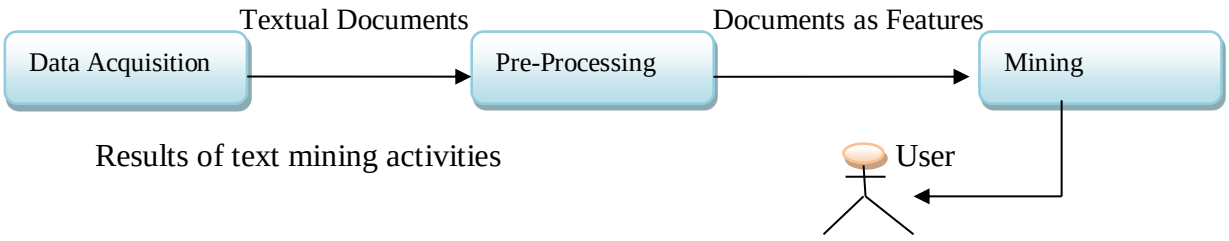


Figure 1: Text mining system architecture

1.31 Data Acquisition

The set of textual documents, also known as corpus, can be collected from internet resources using a web crawler mechanism [1] or another automated mechanism to collect unstructured data from email and messaging systems, databases, or textual files existing in a file system (e.g., log files, digitized books, speech to text, etc.). The selection of excellent and reliable sources of textual content is fundamental to obtain a successful text mining system.

1.32 Pre-Processing

To transform unstructured data in features, the textual documents must be parsed into simple words, with the blank spaces and punctuation used to distinguish and separate the words. This process is also known as tokenization. A list with all existing words and the respective number of occurrences in the corpus can also be generated during this phase. After this, the words or terms are selected to form features. In this context, a feature can be understood as a value, and the feature name is the meaning of this value. Features can represent a word, a sequence of words or *n*-grams, which consists in a series of consecutive *n* words [5] types of entities (e.g., company names, stock symbols), quantitative values (e.g., stock prices, date, time), syntactical structures like noun-phrases and part-of-speech, etc.

Not all the words carry information in the textual content. The stop words are terms with low importance for information retrieval (normally prepositions), and its removal is recommended. Terms with occurrence per document lower or above a specified threshold are also recommended for removal, because a few numbers of words have no representation, and do not carry significant information in the document. The same applies to repeated and abundant words. The min/max thresholds must be adjusted according to the problem under study, but normally values lower than ~ 5% or greater than ~ 90% are reported in the literature.

The use of stemming reduces the number of words, by replacing a word to its base or stem (Lovins, 1968), (Porter, 1980), e.g., fruit = fructify, fruity, fruitful. The use of stemming requires caution and must be adjusted according to the problem under study, as it may remove important information existing in the original words.

The most common type of feature representation is the Bag of Words (BOW), first mentioned by [16] and still a predominant technique nowadays [7] [15].

A BOW is basically a matrix, where each document is represented as a vector row, and the features (normally words) as the columns of this matrix. The columns of this matrix must contain not only the existing terms in the document, but also all the existing terms in the corpus. Not all the documents share the same terms, then the missing terms in a document are filled with zero or null, which can result in a sparse matrix.

The feature values can be represented as categorical, binary (i.e., existence, nonexistence of a feature in a document), and numerical values. The numerical values can contain any integer or continuous value extracted from the textual content (e.g., prices, counting, etc.), or some measurement or weighting regarding that feature. For example, the Term Occurrence (TO), is the number of times a term occurs in a document, Term Frequency (TF) is the TO divided by the total number of terms in the document (22), since every document has a different length, it is possible that a term would appear many more times in long documents than shorter ones, then the division is a way of normalization.

$$TF(t,d) = \frac{\text{Number of occurrence of t in d}}{\text{Number of terms in d}}$$

Where:

- t is the term;
- d is the document

When using TF, all terms have the same importance; however, to account for the fact that some words appear more frequently than others in all documents, the TF is inversely weighted by the frequency of the same word along the corpus, also known as the Term Frequency-Inverse Document Frequency (TF-IDF) [17]. Nowadays, TF-IDF continues being the most common approach for feature representation in text mining [7] [15].

1.33 Mining

Once the data existing in the textual documents are readable in terms of features and values, they can be processed by a sort of algorithms. Documents can be grouped and associated using unsupervised learning (document clustering, association rules) to identify, visualize, and understand communities, concepts, taxonomies, and sentiments. Entities, groups of documents, taxonomies, sentiments, concepts, and meta-data can be used to assign a category (also known as label) to each document, to be used in supervised learning (document classification and regression) and recommendation systems [2][7][15].

The architecture and the most common techniques for text mining were presented in this section, and they will be referred frequently in this work. Nevertheless, text mining is an extensive and evolving area, and one section is not enough to describe all this branch of research.

1.4 Machine Learning

1.41 Artificial Neural Network (ANN)

An ANN has several advantages, but one of the most recognised of these is the fact that it can learn from observing data sets. In this way, Artificial Neural Network is known for as a random function approximation tool. These types of tools help estimate the most cost-effective and ideal methods for arriving at solutions while defining computing functions or distributions. ANN use data samples instead of complete data sets to arrive at solutions, which saves both time and money. ANNs are considered relatively simple mathematical models to improve the effectiveness of the available data analysis technologies. ANNs includes three layers. These layers are connected to each other. The first node consists of input neurons. Those neurons send data on to the second node, which in turn sends the output neurons to the third node. Training an ANN involves choosing from allowed models different algorithm.

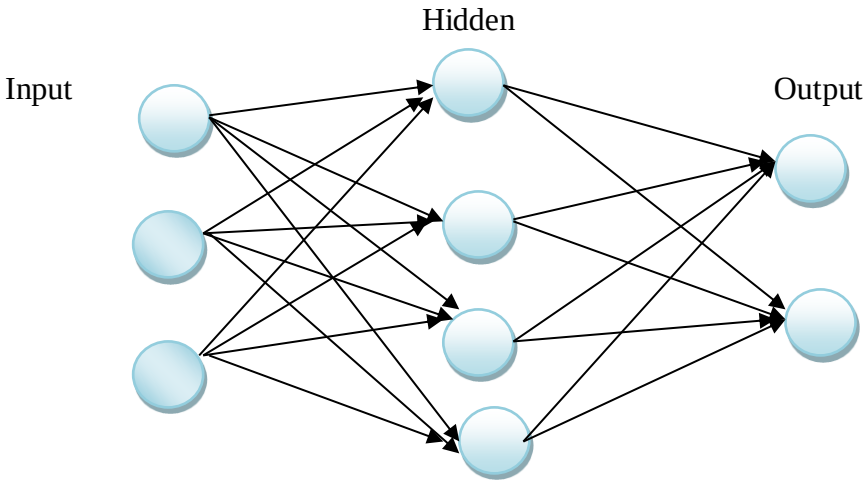


Figure 2: Artificial Neural Network

Phua et al. [24] applied Neural Networks to the financial prediction. He tested the influence of volume data on Stock price prediction. Khan et al. [25] applied the Neural Networks with different number of hidden layers to analyze the prediction of the Stock prices.

1.42 Light Gradient Boosting Machine (LightGBM)

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

Benefitting from these advantages, LightGBM is being widely-used in many winning solutions of machine learning competitions. Comparison experiments on public datasets show that LightGBM can outperform existing boosting frameworks on both efficiency and accuracy, with significantly lower memory consumption. What's more, parallel experiments show that LightGBM can achieve a linear speed-up by using multiple machines for training in specific settings.

1.43 Multi-Linear Regression (MLR)

Most common analytical tool for today's stock market is the regression method. It uses essential values to estimate the future. Multiple Linear Regression is a study on the relationship between one continuous dependent variable and two or more independent variables. Many factors determined the price of a stock and based on "a guess of experts", several economic factors had been identified to influence the stock prices and also some of the insiders and the news is part of price fluctuation. The applied model of Multiple Linear Regression will be used to predict the future stock price [22].

The model for MLR, given n observations, is:

$y_i = B_0 + B_1x_{i1} + B_2x_{i2} + ... + B_px_{ip} + E$ where i = 1,2, ..., n

Where y_i = dependent variable

x_{i1} = independent variable – interest rates

x_{i2} = independent variable – oil price

x_{i3} = independent variable – value of S&P 500 index

x_{i4}= independent variable – price of oil futures

E = random error in prediction, that is variance that cannot be accurately predicted by the model. Also known as residuals.

B_0 = y-intercept at time zero.

B_1 = regression coefficient that measures a unit change in the dependent variable when x_{i1} changes – change in XOM price when interest rates change

B2 = coefficient value that measures a unit change in the dependent variable when x_{i2} changes – change in XOM price when oil prices change.

1.44 XGBoost (Extreme Gradient Boosting)

Ever since its introduction in 2014, Extreme Gradient Boosting with XGBoost has been admired as the holy grail of machine learning competitions. From predicting ad click-through rates to classifying high energy physics events, XGBoost has proved its one of the best algorithms in terms of performance and speed. I always used to XGBoost as my first algorithm of building machine learning model because it saves time and gives better accuracy and provide better solution than other machine learning algorithm and it is optimized and distributed gradient boosting library. It uses gradient boosting framework at core.

XGBoost was created by Tianqi Chen, PhD Student, University of Washington. It is used for supervised ML problems. Features of XGBoost algorithm.

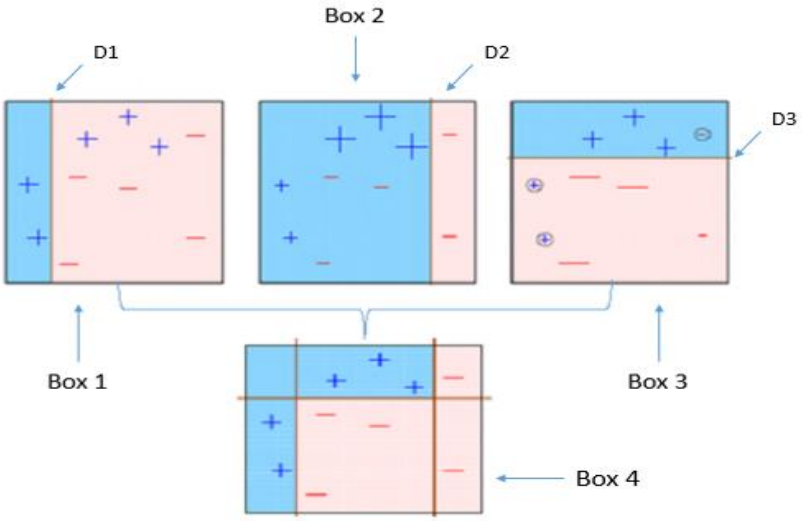
1. **Parallel Computing:** It is enabled with parallel processing (using OpenMP); i.e., when you run XGBoost, by default, it would use all the cores of your laptop/machine.
2. **Regularization:** I believe this is the biggest advantage of XGBoost. GBM has no provision for regularization. Regularization is a technique used to avoid overfitting in linear and tree-based models.
3. **Enabled Cross Validation:** In R, we usually use external packages such as caret and mlr to obtain CV results. But, XGBoost is enabled with internal CV function (we'll see below).
4. **Missing Values:** XGBoost is designed to handle missing values internally. The missing values are treated in such a manner that if there exists any trend in missing values, it is captured by the model.
5. **Flexibility:** In addition to regression, classification, and ranking problems, it supports user-defined objective functions also. An objective function is used to measure the performance of the model given a certain set of parameters. Furthermore, it supports user defined evaluation metrics as well.
6. **Availability:** Currently, it is available for programming languages such as R, Python, Java, Julia, and Scala.

7. **Save and Reload:** XGBoost algorithm gives us a feature to save our data matrix and model and reload it later. Suppose, we have a large data set, we can simply save the model and use it in future instead of wasting time redoing the computation.
8. **Tree Pruning:** Unlike GBM, where tree pruning stops once a negative loss is encountered, XGBoost grows the tree upto max_depth and then prune backward until the improvement in loss function is below a threshold.

How does XGBoost work

XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing.

Boosting: Boosting is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations. Let's look at a classic classification example:



Four classifiers (in 4 boxes), shown above, are trying hard to classify + and - classes as homogeneously as possible. Let's understand this picture well.

1. **Box 1:** The first classifier creates a vertical line (split) at D1. It says anything to the left of D1 is + and anything to the right of D1 is -. However, this classifier misclassifies three + points.
2. **Box 2:** The next classifier says don't worry I will correct your mistakes. Therefore, it gives more weight to the three + misclassified points (see bigger size of +) and creates a vertical line at D2. Again, it says, anything to right of D2 is - and left is +. Still, it makes mistakes by incorrectly classifying three - points.

3. **Box 3:** The next classifier continues to bestow support. Again, it gives more weight to the three -misclassified points and creates a horizontal line at D3. Still, this classifier fails to classify the points (in circle) correctly.
4. Remember that each of these classifiers has a misclassification error associated with them.
5. Boxes 1,2, and 3 are weak classifiers. These classifiers will now be used to create a strong classifier Box 4.
6. **Box 4:** It is a weighted combination of the weak classifiers. As you can see, it does good job at classifying all the points correctly.

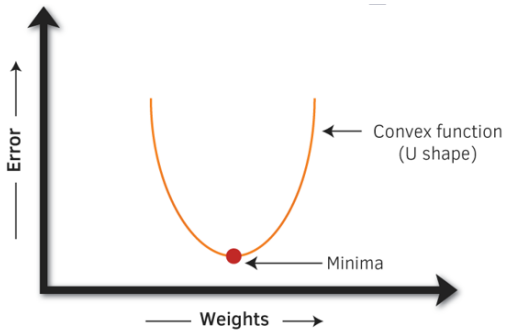
That's the basic idea behind boosting algorithms. The very next model capitalizes on the misclassification/error of previous model and tries to reduce it. Now, **let's come to XGBoost**.

As we know, XGBoost can used to solve both regression and classification problems. It is enabled with separate methods to solve respective problems.

Classification Problems: To solve such problems, it uses booster = gbtree parameter; i.e., a tree is grown one after other and attempts to reduce misclassification rate in subsequent iterations. In this, the next tree is built by giving a higher weight to misclassified points by the previous tree (as explained above).

Regression Problems: To solve such problems, we have two methods: booster = gbtree and booster = gblinear. You already know gbtree. In gblinear, it builds generalized linear model and optimizes it using regularization (L1,L2) and gradient descent. In this, the subsequent models are built on residuals (actual - predicted) generated by previous iterations. Are you wondering what is gradient descent? Understanding gradient descent requires math, however, let me try and explain it in simple words:

- **Gradient Descent:** It is a method which comprises a vector of weights (or coefficients) where we calculate their partial derivative with respective to zero. The motive behind calculating their partial derivative is to find the local minima of the loss function which is convex in nature. In simple words, gradient descent tries to optimize the loss function by tuning different values of coefficients to minimize the error.



XGBoost Parameters Tuning

Every parameter has a significant role to play in the model's performance. Before hyper tuning, let's first understand about these parameters and their importance.

XGBoost parameters can be divided into three categories (as suggested by its authors):

- **General Parameters:** Controls the booster type in the model which eventually drives overall functioning
- **Booster Parameters:** Controls the performance of the selected booster
- **Learning Task Parameters:** Sets and evaluates the learning process of the booster from the given data

1. General Parameters

1. **Booster[default=gbtree]**
    - Sets the booster type (gbtree, gblinear or dart) to use. For classification problems, you can use gbtree, dart. For regression, you can use any.
2. **nthread[default=maximum cores available]**
    - Activates parallel computation. Generally, people don't change it as using maximum cores leads to the fastest computation.
3. **silent[default=0]**
    - If you set it to 1, your R console will get flooded with running messages. Better not to change it.

2**. Booster Parameters**

As mentioned above, parameters for tree and linear boosters are different. Let's understand each one of them:

Parameters for Tree Booster

1. **rounds[default=100]**
    - It controls the maximum number of iterations. For classification, it is similar to the number of trees to grow.
    - Should be tuned using CV
2. **eta[default=0.3] [range: (0,1)]**
    - It controls the learning rate, i.e., the rate at which our model learns patterns in data. After every round, it shrinks the feature weights to reach the best optimum.
    - Lower eta leads to slower computation. It must be supported by increase in nrounds.
    - Typically, it lies between 0.01 - 0.3
3. **gamma[default=0] [range: (0,Inf)]**

- It controls regularization (or prevents overfitting). The optimal value of gamma depends on the data set and other parameter values.
- Higher the value, higher the regularization. Regularization means penalizing large coefficients which don't improve the model's performance. default = 0 means no regularization.
- *Tune trick:* Start with 0 and check CV error rate. If you see train error >>> test error, bring gamma into action. Higher the gamma, lower the difference in train and test CV. If you have no clue what value to use, use gamma=5 and see the performance. Remember that gamma brings improvement when you want to use shallow (low max_depth) trees.

4. **max_depth[default=6][range: (0,Inf)]**
    - It controls the depth of the tree.
    - Larger the depth, more complex the model; higher chances of overfitting. There is no standard value for max_depth. Larger data sets require deep trees to learn the rules from data.
    - Should be tuned using CV
5. **min_child_weight[default=1][range:(0,Inf)]**
    - In regression, it refers to the minimum number of instances required in a child node. In classification, if the leaf node has a minimum sum of instance weight (calculated by second order partial derivative) lower than min_child_weight, the tree splitting stops.
    - In simple words, it blocks the potential feature interactions to prevent overfitting. Should be tuned using CV.
6. **subsample[default=1][range: (0,1)]**
    - It controls the number of samples (observations) supplied to a tree.
    - Typically, its values lie between (0.5-0.8)
7. **colsample_bytree[default=1][range: (0,1)]**
    - It control the number of features (variables) supplied to a tree
    - Typically, its values lie between (0.5,0.9)
8. **lambda[default=0]**
    - It controls L2 regularization (equivalent to Ridge regression) on weights. It is used to avoid overfitting.
9. **alpha[default=1]**
    - It controls L1 regularization (equivalent to Lasso regression) on weights. In addition to shrinkage, enabling alpha also results in feature selection. Hence, it's more useful on high dimensional data sets.

Parameters for Linear Booster

Using linear booster has relatively lesser parameters to tune, hence it computes much faster than gbtree booster.

1. **nrounds[default=100]**

- It controls the maximum number of iterations (steps) required for gradient descent to converge.
- Should be tuned using CV

2. **lambda[default=0]**
    - It enables Ridge Regression. Same as above
3. **alpha[default=1]**
    - It enables Lasso Regression. Same as above

3. Learning Task Parameters

These parameters specify methods for the loss function and model evaluation. In addition to the parameters listed below, you are free to use a customized objective / evaluation function.

1. **Objective[default=reg:linear]**
    - reg:linear - for linear regression
    - binary:logistic - logistic regression for binary classification. It returns class probabilities
    - multi:softmax - multiclassification using softmax objective. It returns predicted class labels. It requires setting num_class parameter denoting number of unique prediction classes.
    - multi:softprob - multiclassification using softmax objective. It returns predicted class probabilities.
2. **eval_metric [no default, depends on objective selected]**
    - These metrics are used to evaluate a model's accuracy on validation data. For regression, default metric is RMSE. For classification, default metric is error.
    - Available error functions are as follows:
        - mae - Mean Absolute Error (used in regression)
        - Logloss - Negative loglikelihood (used in classification)
        - AUC - Area under curve (used in classification)
        - RMSE - Root mean square error (used in regression)
        - error - Binary classification error rate [#wrong cases/#all cases]
        - mlogloss - multiclass logloss (used in classification)

CHAPTER -2

MOTIVATION

Adequate Market Hypothesis is one of the popular theories in financial economics. Prices of the stock indicate all the information that is already available, and it is difficult to outperform the market consistently. There are three exceptions of the Efficient Market System, namely weak form, semi-strong form and the active form. The soft way says that the securities reflect all the information that is publicly available in the past. The semi-strong form says that the price indicates all the publicly accessible data and also, they change spontaneously to reflect the recently available data. The active form would include even insider or hidden information. But this method is often disputed and highly questionable. The original illustration would be investors such as Warren Buffet, who have made huge profits over a long time by consistently outperforming the market. Even difficult to predict the trend of the stock market price by manually translating the chaotic stock market data is a slow task, with the approach of the machine learning algorithm, artificial intelligence, big data and increased computational capabilities, automated systems of forecasting the stock market prices are becoming achievable. So, the best way to predict stock market prices using machine learning algorithm, machine learning models are more capable of learning a function by viewing the data without explicitly being programmed. The time series of a stock is not a function that can be easily mapped because of the stock market follows a very random walk and is dynamic because of this instability the price of the stock has very complex behaviour which makes the feature engineering and prediction much harder. Using machine learning techniques to train the data and build the model to predict and identify the trend of large amounts of stock data, but still, there is no such algorithm or model which could consistently predict the price of future stock price accurately. A lot of research is going on both in academia and organisation on this challenging problem.

CHAPTER -3

LITERATURE REVIEW

2.1 Research Paper - 1

Paper: Forecasting Stock Market Movement: A Neural Network Approach

.

Authors: Ms. Ashwini Pravin Navghane, Dr. Pradeep Mitharam Patil.

Journal: IJECET

Year: 2013

Literature Review: With the increasing accuracy and precision of the analytical measuring methods show that all result that is of interest cannot be described by simple univariate and even not by the linear multivariate correlations precise. However, in recent researches, it is found a set of methods said to be the artificial neural networks gives the optimised result. In this research paper, the stock market price prediction using the neural network approach is discussed in the following sections.

Prediction Using Feed-forward neural network

A feed-forward neural network was the first and simplest type of artificial neural network devised. Feed-Forward Neural Network consists of at least three nodes of neurons: an input node, intermediate hidden node, and an output node. Usually, neurons are connected in a feed-forward fashion with input units fully connected to neurons in the hidden node and hidden neurons fully connected to neurons in the output layer. There are a few parameters, which we have to use during the training period. They are the error, threshold, tolerance, etc. The accurate results depend on these parameters. To achieve our target rapidly, the error is pre-processed.

Training Using Back Propagation Algorithm

Backpropagation algorithm is a supervised machine learning algorithm, for training Multi-layer Perceptrons. Here Backpropagation algorithm is used for future prediction of stock market rates. In the back-propagation algorithm the steepest-descent minimisation method is used.

Conclusion

The Feed-forward neural network is used to train the dataset and back propagation algorithm is used for prediction. Using this method gives the optimised result.

2.2 Research Paper - 2

Paper: Predicting Stock Market behaviour using Data Mining Technique and News Sentiment Analysis.
Authors: Ayman E. Khedr, S.E. Salama, Nagwa Yaseen.

Journal: MECS

Year: 2017

Literature Review: In this paper, building a useful model to predict stock market future trends with small error ratio and improve the accuracy of prediction and this model is based on sentiment analysis of financial news and historical stock market prices and provide better accuracy of all previous studies. Using data mining techniques divide the task into two important part descriptive and predictive. So, the author takes predictive part, and classification task is used to predict stock movement behaviour Naive Bayes, and KNN algorithm is used to build the predictive model.

In these studies, three categories of news data were considered: news relevant to market, company news and financial reports that were published by financial experts about stocks. The proposed model consists of two phases, the first stage is to determine the news polarities to be either positive or negative using Naive Bayes algorithm, and the second stage incorporates the output of the first phases as input along with the processed historical numeric data attributes to predict the future stock trend using KNN algorithm. The results of our proposed model achieved higher accuracy.

Naive Bayes

Naive Bayes classifier is used to classify the stock news to be either positive or negative sentiment based on TF-IDF values. The Naïve Bayes algorithm allows that the result of an attribute value on a given class is independent of the values of the different attributes. This assumption is called class conditional Independence. Naïve Bayes classifier has been used to predict the polarity of each document because of its simplicity and speed in text classification. It assigns each document into the positive or negative class: Using text analysis of the stock market news to determine the two views of the news articles.

KNN Algorithm

K-NN classifier is used to predict the stock trend up or down. KNN classifier is a method for classifying objects based on the closest training examples in the feature space. The class label is assigned the same class as the nearest K instances in the training set. KNN is a type of lazy learner strategy that delays the process of applying

the model on training data only if it is necessary to classify test data. KNN classifier is considered a flexible and simple classification technique where information about the training data distribution is available [3], [4].

Goal of this model

- The Model helps investor avoid risk and financial crises when making investment decisions.
- Predicting the stock market behaviour, whether it is falling or rising.
- Using text analysis of the stock market news to determine the two views of the news articles either positive or negative.
- Stock market historical prices opening, closing, high and low are analyzed to predict the future growth.

Conclusion

The studies of this paper is Investigated the simultaneous effect of analysing different types of news along with historical numeric attributes for understanding stock market behaviour. It improved the prediction accuracy for the future trend of the stock market, by considering different types of daily news with different values of numeric attributes during a day.

2.3 Research Paper: 3

Paper: Predicting Stock Price Movements Based on Different Categories of News Article.

Author: Yauheniya Shynkeyich, T.M.Micginnity, Sonya Coleman, Ammar Belatreche.
Journal: IEEE
Year: 2015

Literature Review: This research paper studies how to improve stock price movements based on different categories of the news articles using the multiple kernel learning techniques for predicting the price movements and how the simultaneous use of financial news articles with different levels of relevance to the target stock can give an advantage in financial news-based forecasting. To achieve this goal, the Global Industry Classification Standard is employed for dividing stocks by industries and sectors and for assigning their corresponding news articles to five categories:
Stock-Specific, Sub-Industry-Specific, Industry-Specific, Group-Industry-Specific and Sector-Specific news items.

Multiple Kernel Learning

In recent years, many researchers have used a method to deal with the problem of selecting suitable kernels for features from multiple sources. This method is called Multiple Kernel Learning [6].
Some researchers have applied Multiple Kernel Learning to prediction in the FX rate or stock prices; Fletcher, Hussain & Shawe-Taylor [12] applied MKL to the limit order book for predicting and trading on the FX market; Luss & d'Aspremont [13] applied Multiple Kernel Learning for predicting abnormal returns from news using text classification Shie-Jue Lee et al. [14] applied Multiple Kernel Learning to the prediction of prices on the Taiwan stock market, although the results looked a little better than some conventional methods, the prediction also looked like stating the current value; Deng & Sakurai [13] applied MKL on the prediction and trading of FX rate and obtained excellent results.

Advantage of Multiple Kernel Learning

- It allows us to combine different kernels when it is better to use different kernels for different input features.
- Multiple Kernel Learning mitigates the risk of erroneous kernel selection to some degree by taking a set of kernels and deriving a weight for each kernel such that predictions are made based on the weighted sum of the kernels.

Conclusion

This research study explores whether the simultaneous usage of different financial news categories can provide an advantage in financial prediction system based on news. Five categories of news articles were considered: news relevant to a target stock and news relevant to its sub-industry, industry, group industry and sector. Each category of news articles was pre-processed independently, and five different subsets of data were constructed. The MKL approach was utilised for learning from different news categories; independent kernels were employed to learn from each subset. A number of different types of kernels and kernel combinations were used. It gives highest prediction accuracy and accurate result.

2.4 Research Paper: 4

Paper: Prediction Model for Stock Market using News based different Classification, Regression and Statistical Techniques

Authors: Hiral R. Pate, Satyen M. Parikh, Dhara N. Darji

Journal: IEEE

Year: 2016

Literature Review: Principle goals of this model for better investment decision for stock market price. The decision policy would be driven by analysing stock price change based on sector choice and news released. This model works with three different

types of forecasting techniques: Predictive, Statistical and Regression. These methods are selected by performing comparative analysis that published in the earlier research paper. As a Regression Analysis, the Multiple Linear Regression is used and as a Predictive method neural network is applied. As a Statistical Technique the Exponential smoothing, double exponential smoothing and moving average are used.

Multi-Linear Regression: Most common analytical tool for today's stock market is the regression method. It uses essential values to estimate the future. Multiple Linear Regression is a study on the relationship between a single dependent variable and one or more independent variables. Many factors determined the price of a stock and based on "a guess of experts", several economic factors had been identified to influence the stock prices and also some of the insiders and the news is part of price fluctuation. The applied model of Multiple Linear Regression will be used to predict the future stock price.

Neural Network: Neural Networks have great the explosion in few years and also cover wide diversified domain areas such as finance, medicine, engineering, geology and physics. The biological neuronal structure inspires artificial neurons. The way of a signal from one neuron to another neuron through synapses is a complicated chemical method in which specific transmitter substances are delivered from the sending side of the junction. The result is to higher or lowers the electrical potential within the body of the taking cell. If here evaluated potential reaches an edge, the neuron activates. The artificial neuron model regenerates as it is characteristic. So for this study neural network is applied for forecasting.

Result: Using Multi Linear Regression and Neural network algorithm for developing the model and it gives the better result. 82% of accuracy achieved for Multi Linear Regression model and the Neural Network model gives 70% accuracy for prediction. Using the same model with different tools and different methods the neural network with back propagation gives a better and accurate result.

2.5 Research Paper - 5

Paper: Stock market prediction from news on the Web and a new evaluation approach in trading
Author: Kazutaka Shimada, Ko Ichinose
Journal: IEEE

Year: 2016

Literature Review: The primary approach of this paper to propose the method to predict the one-day stock price performance from web news. The target is the Nikkei Stock Average, which is one of the most important stock market indexes and also focuses on the evaluation method of the prediction. The contributions of this paper are:

- Proposing some one-day stock price prediction models, and then compares them.
- Evaluate the prediction models regarding the improvement of an actual return in trading.
- Propose a criterion based on Property Changes in the experiment. By using the criterion that can automatically obtain an objective score to evaluate a property change graph.

Proposed Method

Our method consists of two parts

(1) One-day classifiers

(2) Strategies for trading.

The purpose of one-day classifiers is to classify the next day into "raising" or "dropping" by using the articles of a day. We explain two strategies for trading after generating one-day classifiers.

A. Classifier

Applying machine learning techniques to find out the one-day classifiers with the help of support vector machine and linear perceptron and introduce three types of features to this method.

- BOW: Bag-of-word with binary features.
- VOL: Bag-of-words that are weighted by the absolute value of volatility score (V S).
- AVS: Bag-of-words that are weighted by the actual value of the V S.

We use nouns, verbs and adjectives as the BOW features. The volatility score (*V S*) denotes stock market volatility in each day. It is computed as follows:

$$Vs = hp - lp$$

Where hp and lp are the highest and the lowest prices of the day. We combine the two methods and the three features. As a result, we compare six classifiers SVM (BOW, VOL, AVS), Perceptron (BOW, Vol, AVS) in the experiment. The target value of each classifier is +1 (Rising in the next day) or -1 (Dropping in the next day).

B. Trading settings and strategy

Evaluate method with simulated trading using real stock prices. For the evaluation, we need to set a virtual market and a virtual investor with a strategy.

- Virtual market: We use the past data on the Nikkei Stock Average. Each datum of a day consists of the opening price, closing price, highest price, and lowest price.
- We set the initial money of each investor to 2 million yen (1 million yen for Spot buying and 1 million yen for Short selling).The investor buys stocks if the classifier answers that the price of the next day will be raised and the investor sells stocks if the classifier answers that the price of the next day will be dropped.

Conclusion

In this paper, we proposed six methods and two strategies to predict the Nikkei Stock Average. The best accuracy of one-day stock price prediction models, namely "raising" or "dropping", was approximately 60 %. We also evaluated the prediction models in terms of the improvement of an actual return in trading. In the simulated trading, a simple strategy produced good results for the six one-day classifiers. In the trading situation, we introduced a new criterion to evaluate the models; the property changes *CPC*. By using the criterion *CPC,* we can easily understand the effectiveness of one-day classifiers in terms of the real trading situation.

2.6 Research Paper - 6

Paper: Stock Market Prediction using Data Mining Techniques
Author: Sahaj Singh Maini, Govinda.K
Journal: IEEE
Year: 2017
Literature Review: The primary approach of this paper is towards prediction of stock market trends using machine learning models like Random forest and Support vector machine give the better performance model and optimised the result. These two models are used to forecast whether the price of a stock in the future will be higher than its price on a given day based on the historical data.
This paper studies the use of sentiment analysis of text from the content derived from news sources for understanding the natural language and inferring whether the message it provides is positive, negative or neutral towards its influence on the stock price.

Random Forest

Random Forest model is an ensemble learning method that creates multiple decision trees based on random subsets of data during the training and outputs the mode of classes that have resulted from these decision trees using an input for classification. Random Forest Model involves a general technique called Bootstrap Aggregation which leads to better model performance by decreasing the variance.

Support Vector Machine

Support Vector Machine algorithm is used for classification and regression. Support Vector Machine is predominantly used for classification problems. It is a maximum margin classification algorithm. In Support Vector Machine each data item is plotted on an n-dimensional space, where n is equal to the number of features, and then an attempt is made to classify these points by finding the most suitable plane that differentiates these points better than all the other planes

Conclusion

Using two machine learning model Random Forest and Support Vector Machine are providing a reliable prediction of stock market trends based on historical data. Random Forest using 1-gram model for text analysis produced accuracy of 84.3% and on using a 2 –gram model produced an accuracy of 86.6%, The linear Support Vector Machine using a 1-gram model and 2-gram model for text analysis produced predictions with the accuracy of 82.2% and 84.6%, while the nonlinear Support Vector Machine produced predictions with an accuracy of 85.1% for both 1-gram and 2-gram models. We have observed that the Random Forest Model outperforms the Support Vector Machine while using the given dataset. However, Support Vector Machine is a favourable substitute for financial forecasting when the data is in time-series format

2.7 Research Paper - 7

Paper: Analysing News for Stock Market Prediction

Author: V. V. Ramalingam, A. Pandian, Shivam Dwivedi and Jigar P. Bhatt

Journal: National Conference on Mathematical Techniques and its Applications

Year: 2018

Literature Review: In this paper using the statistical model for predicting the future stock market values. Using Quindl, it provides alternative data platform and the method comprises of completing the Natural Language Processing of the data and then making it easier for the system to understand, finds and identifies the correlation in between this data and the stock market fluctuations.

Quindl: Collecting online data from a social media platform is known as Quindl.

The size of data is very large and therefore the data is linked to form a csv (Comma Separated Value) file and is then parsed to obtain cleaned data, which is the required characteristic data for predicting the stock market values for the user. Sales, Marketing, Manufacturing are other areas where this data can be used. Since the data collected from Quindl also includes public opinions, it cannot be completely relied on.
Opinions mining are computational techniques used for extracting useful characteristic data from public opinions on social media platforms or from the news.
Using the same technique, the market moods monitored from Quindl can be used to predict the flow of stock prices.

Conclusion: Introducing another method of focusing on deriving the best statistical learning model for predicting the future values. Stock markets vary over different time periods and analysing the news makes it easier for determining the trends of the market. Thus, it can be said they have a relationship in between them. The model was developed based on this relationship and python language was chosen to make the model lesser complex. The accuracy of the model is also determined along with the forecasting results over the time period range and Quindl provides reliable and authentic dataset.

Table 1: Literature Review Comparison

Title	Author	Year	Method	Parameter	Remark
Forecasting Stock Market Movement: A Neural Network Approach	Ms. Ashwini Pravin Navghane, Dr. Pradeep Mitharam Patil	2013	Prediction Using Feed-forward neural network, Training Using Back Propagation Algorithm	Three-layer feed forward neural network is used.	Feed forward NN is used to train the dataset and back propagation algorithm is used for prediction.
Predicting Stock Market behaviour using Data Mining Technique and News Sentiment Analysis	Ayman E. Khedr, S.E. Salama, Nagwa Yaseen	2017	Naive Bayes and KNN algorithm are used	Positive, Negative, Falling and Raising	Using Naïve Bayes algorithm up to 86.21%. This achieved the highest accuracy compared to other previous researches, our model for predicting the future behaviour of stock market obtained accuracy up to 89.80% using KNN.
Predicting Stock Price Movements Based on Different Categories of News Article	Yauheniya Shynkeyich, T.M. Micginnity, Sonya Coleman, Ammar Belatreche	2015	SVM, KNN, Multiple Kernel Learning	SS, SIS, IS and GIS data. Kernel type	The MKL approach was used for learning from different news categories; independent kernels were employed to learn from each subset. A number of different types of kernels and kernel combinations were used.
Prediction Model for Stock Market using News based different Classification, Regression and Statistical Techniques.	Hiral R. Pate, Satyen M. Parikh, Dhara N. Darji	2016	Multi linear Regression, Neural Network	Open price is output parameter and another variable are input parameter.	Using Multi Linear Regression and Neural network algorithm for developing the model and it gives the better result. 82% of accuracy achieved for Multi Linear Regression model and the Neural Network model gives 70% accuracy for prediction

Stock Market Prediction using Data Mining Techniques	Sahaj Singh Maini, Govinda.K	2017	Random forest and Support vector machine	unigram(n=2), bigram(n=2)	Random Forest model using a 1- and 2-gram model and produced 84.3%accuracy. Support Vector Machine using 1-gram model and 2-gram model for text analysis produced predictions with an accuracy of 82.2% and 84.6%.
Analysing News for Stock Market Prediction	V. V. Ramalingam, A. Pandian, Shivam Dwivedi and Jigar P. Bhatt	2018	Quindl and opinion Mining	Text	The accuracy of the model is determined along with the forecasting results over the time period range and also Quindl provide authentic and reliable data set.

Chapter - 4

Objective of the work

This paper shows the analysis of news data to predict stock prices moments with using machine learning algorithm Extreme Gradient Boosting and Light Gradient Boosting. This unique opportunity will advance the state of research in understanding the predictive power of the news. This power if harnessed could help financial outcomes and generate significant economic impact all over the world. Here the target is to predict the 10-day market residualized returns depending upon various price movements recorded at the close and open time of the market associated with that stock using news articles.

The stock market traces the sequence of stochastic process and fluctuates frequently. The main cause of variations is lack of stability in stock prices which is extremely complicated, in financial market profit exploitation occurs simultaneously with the price differences, that's why it is crucial to investigate stock market data tremendously. The stockholders take their decision by considering the financial news articles that impart all those factors influencing the market. The financial newscast discloses the necessary details about organization views, mission, requirements and bulging out the image of activities concerned with the organization and expectancy build up by anticipators and inspecting relevant information related to financial market such as the trading volume inflation, market demand of the specific product or services provided by the organization opening price and closing price, calculated return etc. It has already declared in the previous research work that the stock prices variations is correlated with the publication of related news journals and financial market information of the stock. Generally, it is observed that the purchasing or selling of the stock impact with the fluctuations in stock price. Stock prices move up and down due to the variations in supply and demand. In this research paper we are predicting the signed confidence value between (-1, +1) which will be multiplied by the market adjusted value of the given stock over a 10-day market residualized return value. The output of this research will be a model which will be used for short term prediction.

CHAPTER- 5

Functional Partitioning of Project

- **Proposed Model**
- **Data Set Description**
- **Exploratory Data Analysis**
- **Pre-processing of data (Stock Market Data and News Data)**
- **Text Mining of News Headline data**
- **Join Features of News Data and Market Data**
- **Build Model**

PROPOSED MODEL

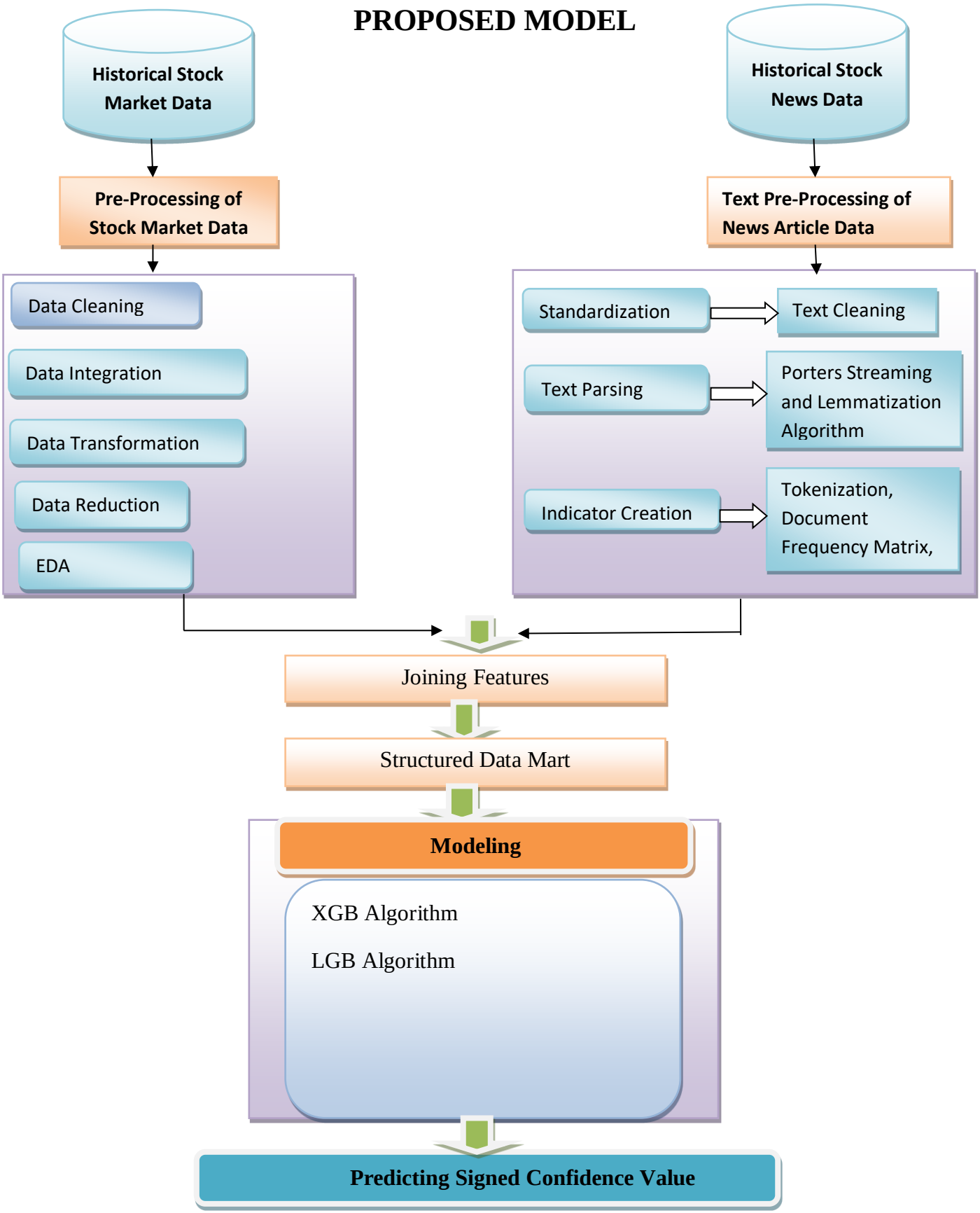


Figure 4: Proposed Model

So, for most of the research papers either it is determined the best time to buy or sell the stock or whether the stock price will increase or decrease or may be near about same.

The objective of this modelling practice is to predict the signed confidence value between (-1, +1) which will be multiplied by the market adjusted value of the given stock over a 10 day market residualized return value of a stock by applying different machine learning techniques to capture how a stock return value is affected by various kinds of financial news articles, political features, economic features etc.

5.1.2 Data Set Description

This dataset contains two different datasets and both datasets are taken from Kaggle. (https://www.kaggle.com)

5.1.2.1 Stock Market Data Set:

Stock Market Data Set contains a variety of returns calculated over different time spans. All of the returns in this set of market data have these properties.

- Returns are always calculated either open-to-open (from the opening time of one trading day to the open of another) or close-to-close (from the closing time of one trading day to the open of another).
- Returns are either raw, meaning that the data is not adjusted against any benchmark, or market-residualized, meaning that the movement of the market as a whole has been accounted for, leaving only movements inherent to the instrument.
- Returns can be calculated over any arbitrary interval. Provided here are 1 day and 10-day horizons.
- Returns are tagged with 'Prev' if they are backwards looking in time, or 'Next' if forwards looking.

Stock Market data has the following features:

- **time(datetime64[ns, UTC]) -** The current time (in market data, all rows are taken at 22:00 UTC)
- **assetCode (object) -** A unique id of an asset
- **assetName (category) -** The name that corresponds to a group of assetCodes. These may be "Unknown" if the corresponding assetCode does not have any rows in the news data.
- **universe(float64) –** A boolean indicating whether or not the instrument on that day will be included in scoring. This value is not provided outside of the training data time period. The trading universe on a given date is the set of instruments that are available for trading (the scoring function will not consider instruments that are not in the trading universe). The trading universe changes daily.
- **volume(float64) -** Trading volume in shares for the day
- **close(float64) -** The close price for the day (not adjusted for splits or dividends)
- **open(float64) -** The open price for the day (not adjusted for splits or dividends)
- returnsClosePrevRaw1(float64)

- returnsOpenPrevRaw1(float64)
- returnsClosePrevMktres1(float64)
- returnsOpenPrevMktres1(float64)
- returnsClosePrevRaw10(float64)
- returnsOpenPrevRaw10(float64)
- returnsClosePrevMktres10(float64)
- returnsOpenPrevMktres10(float64
- returnsOpenNextMktres10(float64)
  10 day, market-residualized return, this is the target variable. The market data has been filtered such that returnsOpenNextMktres10 is always not null.

5.1.2.2 News data

The news data contains information at both the news article level and asset level (in other words, the table is intentionally not normalized).

- **time(datetime64[ns, UTC])** - UTC timestamp showing when the data was available on the feed (second precision)
- **sourceTimestamp(datetime64[ns, UTC])** - UTC timestamp of this news item when it was created
- **firstCreated(datetime64[ns, UTC])** - UTC timestamp for the first version of the item
- **sourceId(object)** - An Id for each news item
- **headline(object)** - The item's headline
- **urgency(int8)** - differentiates story types (1: alert, 3: article)
- **takeSequence(int16)** - The take sequence number of the news item, starting at 1. For a given story, alerts and articles have separate sequences.
- **provider(category)** - identifier for the organization which provided the news item (e.g. RTRS for Reuters News, BSW for Business Wire)
- **subjects(category)** - topic codes and company identifiers that relate to this news item. Topic codes describe the news item's subject matter. These can cover asset classes, geographies, events, industries/sectors, and other types.
- **audiences(category)** - identifies which desktop news product(s) the news item belongs to. They are typically tailored to specific audiences. (e.g. "M" for Money International News Service and "FB" for French General News Service)
- **bodySize(int32)** - the size of the current version of the story body in characters
- **companyCount(int8)** - the number of companies explicitly listed in the news item in the subjects field
- **headlineTag(object)** - the Thomson Reuters headline tag for the news item
- **marketCommentary(bool)** - Boolean indicator that the item is discussing general market conditions, such as "After the Bell" summaries
- **sentenceCount(int16)** - the total number of sentences in the news item. Can be used in conjunction with firstMentionSentence to determine the relative position of the first mention in the item.

- **wordCount(int32) -** the total number of lexical tokens (words and punctuation) in the news item
- **assetCodes (category)** - list of assets mentioned in the item
- **assetName(category) -** name of the asset
- **firstMentionSentence(int16) -** the first sentence, starting with the headline, in which the scored asset is mentioned.

  1: Headline

  2: First sentence of the story body

  3: Second sentence of the body, etc

  4: The asset being scored was not found in the news item's headline or body text. As a result, the entire news item's text (headline + body) will be used to determine the sentiment score.
- **relevance(float32) -** A decimal number indicating the relevance of the news item to the asset. It ranges from 0 to 1. If the asset is mentioned in the headline, the relevance is set to 1. When the item is an alert (urgency == 1), relevance should be gauged by firstMentionSentence instead.
- **sentimentClass(int8) -** Indicates the predominant sentiment class for this news item with respect to the asset. The indicated class is the one with the highest probability.
- **sentimentNegative(float32) -** Probability that the sentiment of the news item was negative for the asset
- **sentimentNeutral (float32)** - Probability that the sentiment of the news item was neutral for the asset
- **sentimentPositive (float32) -** Probability that the sentiment of the news item was positive for the asset
- **sentimentWordCount (int32) -** The number of lexical tokens in the sections of the item text that are deemed relevant to the asset. This can be used in conjunction with wordCount to determine the proportion of the news item discussing the asset.
- **noveltyCount12H(int16) -** The 12-hour novelty of the content within a news item on a particular asset. It is calculated by comparing it with the asset-specific text over a cache of previous news items that contain the asset.
- **noveltyCount24H(int16) -** Same as above, but for 24 hours
- **noveltyCount3D(int16) -** Same as above, but for 3 days
- **noveltyCount5D(int16) -** Same as above, but for 5 days
- **noveltyCount7D(int16) -** Same as above, but for 7 days

- **volumeCounts12H (int16) -** the 12-hour volume of news for each asset. A cache of previous news items is maintained and the number of news items that mention the asset within each of five historical periods is calculated.
- **volumeCounts24H(int16) -** Same as above, but for 24 hours
- **volumeCounts3D(int16) -** Same as above, but for 3 days
- **volumeCounts5D(int16) -** Same as above, but for 5 days
- **volumeCounts7D(int16) -** Same as above, but for 7 days

5.2 Exploratory Data Analysis

High level analysis of the data	
Market Data	News Data
1. Time Frame : 2017-01-01 to 2018-07-31	1. Time Frame : 2017-01-01 to 2018-07-31
2. Number Of Observations: 4072965	2. Number Of Observations: 9328827
3. Number of columns: 16	3. Number of columns: 35

Table 2

Importing Library

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt ## Graphical representation

import seaborn as sns ### Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

%matplotlib inline

import datetime      ### for date and time
import lightgbm as lgb  ## importing lightgbm algorithm for building model
from scipy import stats  ### it uses for statistical analysis(it specializes in random variables and probability distributions)
from scipy.sparse import hstack, csr_matrix ## sequence of sparse matrices with compatible shapes
from sklearn.model_selection import train_test_split ## splliliting datat in train and test
from wordcloud import WordCloud ## making word cloud for text data to check which text frequency is higher
from collections import Counter ### A counter is a container that stores elements as dictionary keys, and their counts are stored as dictionary values.
from nltk.corpus import stopwords ## nltk is the one of the best library to use for text mimng(removing stopwords)
from nltk.util import ngrams ## ngram
from sklearn.feature_extraction.text import TfidfVectorizer ## Term frequncy and inverse frequncy
from sklearn.preprocessing import StandardScaler ## scaling data
stop = set(stopwords.words('english'))
```

```
import plotly.offline as py  ### plot
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls

from xgboost import XGBClassifier   ### XGBoost
import lightgbm as lgb
from sklearn import model_selection
from sklearn.metrics import accuracy_score  ## accuracy
### Importing data
    market_train_df = pd.read_csv("C:\\Users\\FAKHRE\\Music\\Data\\market.csv")
    news_train_df = pd.read_csv("C:\\Users\\FAKHRE\\Music\\Data\\news.csv")
```

5.2.1 Stock Market Data

5.2.1.1 Top Head Value of Stock Market Data

```
1 market.head(5)
```

	time	assetCode	assetName	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1	returnsClosePrevMktres1	returnsOpenPrevMktr
0	2007-02-01 22:00:00+00:00	A.N	Agilent Technologies Inc	2606900	32.19	32.17	0.005938	0.005312	NaN	I
1	2007-02-01 22:00:00+00:00	AAI.N	AirTran Holdings Inc	2051600	11.12	11.08	0.004517	-0.007168	NaN	I
2	2007-02-01 22:00:00+00:00	AAP.N	Advance Auto Parts Inc	1164800	37.51	37.99	-0.011594	0.025648	NaN	I
3	2007-02-01 22:00:00+00:00	AAPL.O	Apple Inc	23747329	84.74	86.23	-0.011548	0.016324	NaN	I
4	2007-02-01 22:00:00+00:00	ABB.N	ABB Ltd	1208600	18.02	18.01	0.011791	0.025043	NaN	I

5.2.1.2 Top Tail Value of Stock Market Data

```
1 market.tail(5)
```

	time	assetCode	assetName	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1	returnsClosePrevMktres1	returnsOpenPrevMk
995	2007-02-01 22:00:00+00:00	PEG.N	Public Service Enterprise Group Inc	841500	67.21	67.00	0.002685	-0.006966	NaN	
996	2007-02-01 22:00:00+00:00	PEI.N	Pennsylvania Real Estate Investment Trust	278200	42.82	42.74	0.002810	0.000702	NaN	
997	2007-02-01 22:00:00+00:00	PENN.O	Penn National Gaming Inc	706366	44.60	44.17	0.017800	0.003636	NaN	
998	2007-02-01 22:00:00+00:00	PEP.N	PepsiCo Inc	4382800	65.38	65.00	0.002146	-0.001536	NaN	
999	2007-02-01 22:00:00+00:00	PER.N	SandRidge Permian Trust	304400	16.48	16.33	0.008568	0.004923	NaN	

5.2.1.3 Columns of Market Data

```
1 market.columns
```

```
Index(['time', 'assetCode', 'assetName', 'volume', 'close', 'open',
       'returnsClosePrevRaw1', 'returnsOpenPrevRaw1',
       'returnsClosePrevMktres1', 'returnsOpenPrevMktres1',
       'returnsClosePrevRaw10', 'returnsOpenPrevRaw10',
       'returnsClosePrevMktres10', 'returnsOpenPrevMktres10',
       'returnsOpenNextMktres10', 'universe'],
      dtype='object')
```

5.2.2 News Data Set

5.2.2.1 Top Head Value of News Data

```
1 News.head(5)
```

	time	sourceTimestamp	firstCreated	sourceId	headline	urgency	takeSequence	provider	subjects	audiences	...	noveltyCount12H
0	2007-01-01 04:29:32+00:00	2007-01-01 04:29:32+00:00	2007-01-01 04:29:32+00:00	e58c6279551b85cf	China's Daqing pumps 43.41 min tonnes of oil i...	3	1	RTRS	{'ENR', 'ASIA', 'CN', 'NGS', 'EMRG', 'RTRS', '...	{'Z', 'O', 'OIL'}	...	0
1	2007-01-01 07:03:35+00:00	2007-01-01 07:03:34+00:00	2007-01-01 07:03:34+00:00	5a31c4327427f63f	FEATURE-In kidnapping, finesse works best	3	1	RTRS	{'FEA', 'CA', 'LATAM', 'MX', 'INS', 'ASIA', 'I...	{'PGE', 'PCO', 'G', 'ESN', 'MD', 'PCU', 'DNP',...	...	1
2	2007-01-01 11:29:56+00:00	2007-01-01 11:29:56+00:00	2007-01-01 11:29:56+00:00	1cefd27a40fabdfe	PRESS DIGEST - Wall Street Journal - Jan 1	3	1	RTRS	{'RET', 'ENR', 'ID', 'BG', 'US', 'PRESS', 'IQ'...	{'T', 'DNP', 'PSC', 'U', 'D', 'M', 'RNP', 'PTD...	...	0
3	2007-01-01 12:08:37+00:00	2007-01-01 12:08:37+00:00	2007-01-01 12:08:37+00:00	23768af19dc69992	PRESS DIGEST - New York Times - Jan 1	3	1	RTRS	{'FUND', 'FIN', 'CA', 'SFWR', 'INS', 'PUB', 'B...	{'T', 'DNP', 'PSC', 'U', 'D', 'M', 'RNP', 'PTD...	...	0
4	2007-01-01 12:08:37+00:00	2007-01-01 12:08:37+00:00	2007-01-01 12:08:37+00:00	23768af19dc69992	PRESS DIGEST - New York Times - Jan 1	3	1	RTRS	{'FUND', 'FIN', 'CA', 'SFWR', 'INS', 'PUB', 'B...	{'T', 'DNP', 'PSC', 'U', 'D', 'M', 'RNP', 'PTD...	...	0

5.2.2.2 Top Tail Value of News Data

```
1 News.tail()
```

	time	sourceTimestamp	firstCreated	sourceId	headline	urgency	takeSequence	provider	subjects	audiences	...	noveltyCount12H
995	2007-01-03 07:17:15+00:00	2007-01-03 07:17:15+00:00	2007-01-03 07:17:15+00:00	0b0bd0bcb5aa9cef	Poland - Factors to Watch on Jan 3	3	1	RTRS	{'PL', 'EUROPE', 'STX', 'GVD', 'DBT', 'PRESS',...	{'T', 'D', 'M', 'RNP', 'PTD', 'PMF'}	...	
996	2007-01-03 07:18:35+00:00	2007-01-03 07:18:35+00:00	2007-01-03 07:00:57+00:00	fd91e635b0bfbbf7	Ireland's CRH sees 2006 pretax profit up 23 pct	3	1	RTRS	{'MRG', 'EUROPE', 'IE', 'RESF', 'WEU', 'BLD', ...	{'UKI', 'PCO', 'DNP', 'PSC', 'U', 'RNP', 'EMK',...	...	
997	2007-01-03 07:21:48+00:00	2007-01-03 07:21:48+00:00	2007-01-03 07:21:48+00:00	ca42f600971d7ad5	Spanish stocks - Factors to watch on Jan 3	3	1	RTRS	{'EUROPE', 'FR', 'DE', 'WEU', 'ES', 'ELG', 'TE...	{'DNP', 'PSC', 'RNP', 'EMK', 'MF', 'PTD', 'PMF...	...	
998	2007-01-03 07:21:48+00:00	2007-01-03 07:21:48+00:00	2007-01-03 07:21:48+00:00	ca42f600971d7ad5	Spanish stocks - Factors to watch on Jan 3	3	1	RTRS	{'EUROPE', 'FR', 'DE', 'WEU', 'ES', 'ELG', 'TE...	{'DNP', 'PSC', 'RNP', 'EMK', 'MF', 'PTD', 'PMF...	...	
999	2007-01-03 07:22:31+00:00	2007-01-03 07:22:31+00:00	2007-01-03 07:21:48+00:00	bf1faf76b5322913	Spanish stocks - Factors to watch on Jan 3	3	2	RTRS	{'EUROPE', 'FR', 'DE', 'WEU', 'ES', 'ELG', 'TE...	{'DNP', 'PSC', 'RNP', 'EMK', 'MF', 'PTD', 'PMF...	...	

5.2.2.3 Columns of News Data

```
1 News.columns
```

```
Index(['time', 'sourceTimestamp', 'firstCreated', 'sourceId', 'headline',
       'urgency', 'takeSequence', 'provider', 'subjects', 'audiences',
       'bodySize', 'companyCount', 'headlineTag', 'marketCommentary',
       'sentenceCount', 'wordCount', 'assetCodes', 'assetName',
       'firstMentionSentence', 'relevance', 'sentimentClass',
       'sentimentNegative', 'sentimentNeutral', 'sentimentPositive',
       'sentimentWordCount', 'noveltyCount12H', 'noveltyCount24H',
       'noveltyCount3D', 'noveltyCount5D', 'noveltyCount7D', 'volumeCounts12H',
       'volumeCounts24H', 'volumeCounts3D', 'volumeCounts5D',
       'volumeCounts7D'],
      dtype='object')
```

5.3 EDA (Exploratory Data Analysis)

It refers to the critical processing of informing initial investigation on data so as to discover pattern, to spot anomalies, to test hypothesis and to check assumption with summary statistics

5.3.1 Market Data

5.3.2 At first plotting 10 random assets

```
data = []
for asset in np.random.choice(market_train_df['assetName'].unique(), 10):
    asset_df = market_train_df[(market_train_df['assetName'] == asset)]

    data.append(go.Scatter(
        x = asset_df['time'].dt.strftime(date_format='%Y-%m-%d').values,
        y = asset_df['close'].values,
        name = asset
    ))
layout = go.Layout(dict(title = "Closing prices of 10 random assets",
                  xaxis = dict(title = 'Month'),
                  yaxis = dict(title = 'Price (USD)'),
                  ),legend=dict(
                 orientation="h"))
py.iplot(dict(data=data, layout=layout), filename='basic-line')
```

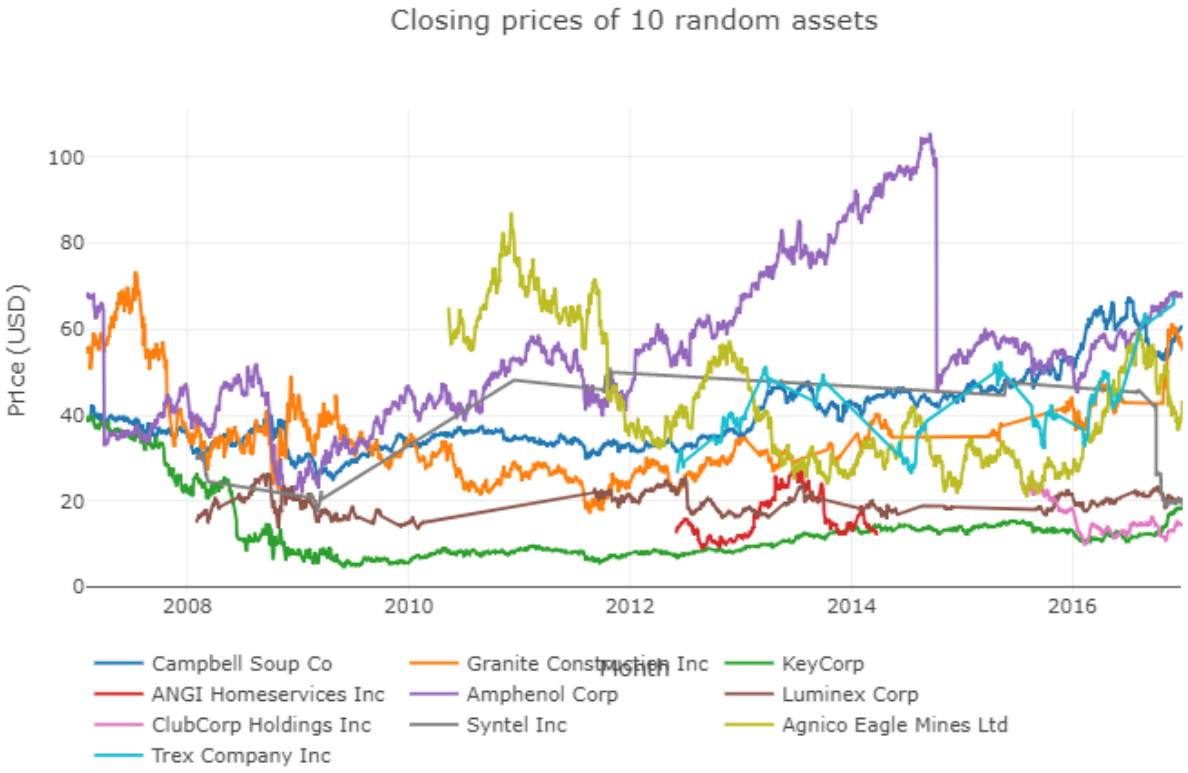


Figure 5: Closing Price of Random Asset

I plot data for all periods because I'd like to show long-term trends. Assets are sampled randomly, but you should see that some companies' stocks started trading later, some disappeared. Disappearance could be due to bankruptcy, acquisition or other reasons.

5.33 Well, these were some random companies. But it would be more interesting to see general trends of prices

```
data = []

#market_train_df['close'] = market_train_df['close'] / 20

for i in [0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95]:

    price_df = market_train_df.groupby('time')['close'].quantile(i).reset_index()
```

```
    data.append(go.Scatter(
        x = price_df['time'].dt.strftime(date_format='%Y-%m-%d').values,
        y = price_df['close'].values,
        name = f'{i} quantile'
    ))

layout = go.Layout(dict(title = "Trends of closing prices by quantiles",
                  xaxis = dict(title = 'Month'),
                  yaxis = dict(title = 'Price (USD)'),
                 ),legend=dict(
                orientation="h"))
py.iplot(dict(data=data, layout=layout), filename='basic-line')
```

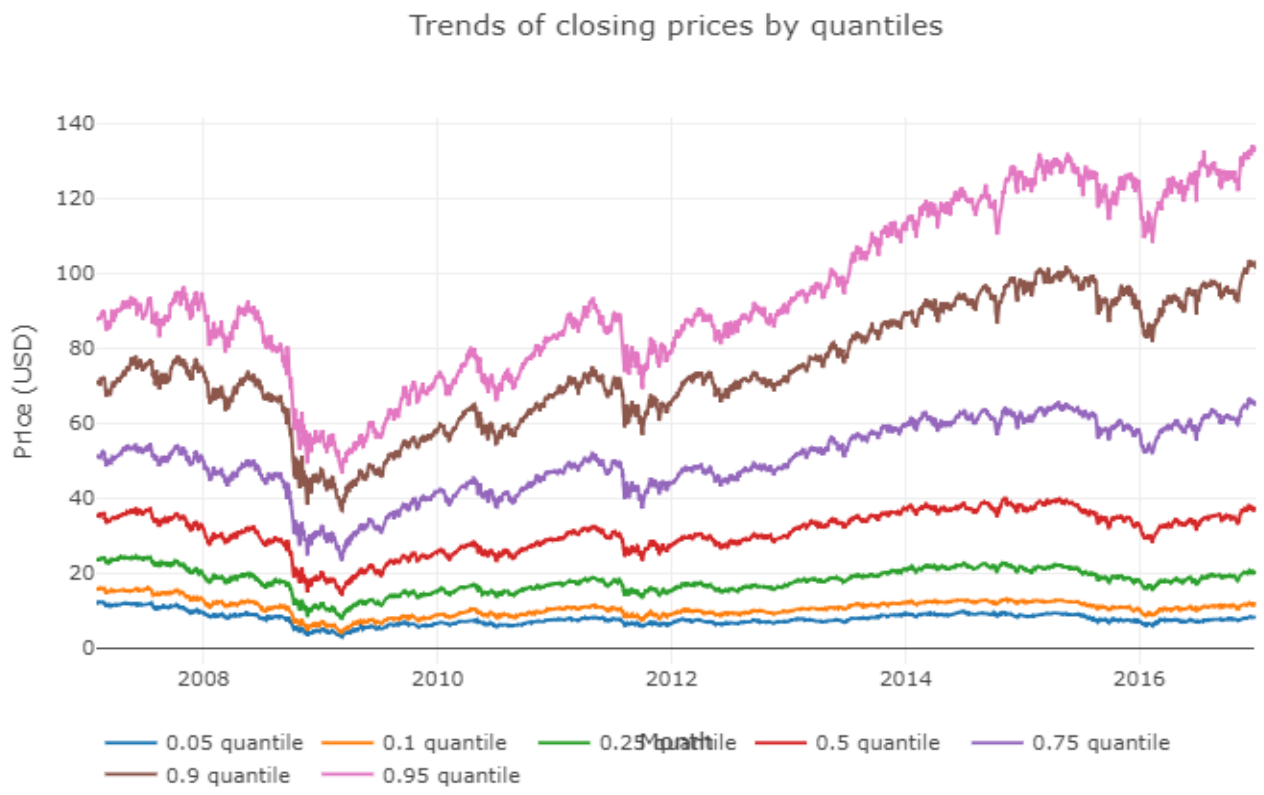


Figure 6: Trend of Closing Price

Explanation

It is cool to be able to see how markets fall and rise again. You could also notice that higher quantile prices have increased with time and lower quantile prices decreased. Maybe the gap between poor and rich increases, on the other hand may be more "little" companies are ready to go to market and prices of their shares isn't very high.

5.3.4 Let's look at these price drops in details.

```
market_train_df['price_diff'] = market_train_df['close'] - market_train_df['open']

grouped=market_train_df.groupby('time').agg({'price_diff':['std', 'min']}).reset_index()


g = grouped.sort_values(('price_diff', 'std'), ascending=False)[:10]

g['min_text'] = 'Maximum price drop: ' + (-1 * g['price_diff']['min']).astype(str)

trace = go.Scatter(
    x = g['time'].dt.strftime(date_format='%Y-%m-%d').values,
    y = g['price_diff']['std'].values,
    mode='markers',
    marker=dict(
        size = g['price_diff']['std'].values,
        color = g['price_diff']['std'].values,
        colorscale='Portland',
        showscale=True
    ),
    text = g['min_text'].values
    #text = f"Maximum price drop: {g['price_diff']['min'].values}"
    #g['time'].dt.strftime(date_format='%Y-%m-%d').values
)
```

```
data = [trace]


layout= go.Layout(
    autosize= True,
    title= 'Top 10 months by standard deviation of price change within a day',
    hovermode= 'closest',
    yaxis=dict(
        title= 'price_diff',
        ticklen= 5,
        gridwidth= 2,
    ),
    showlegend= False
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig,filename='scatter2010')
```

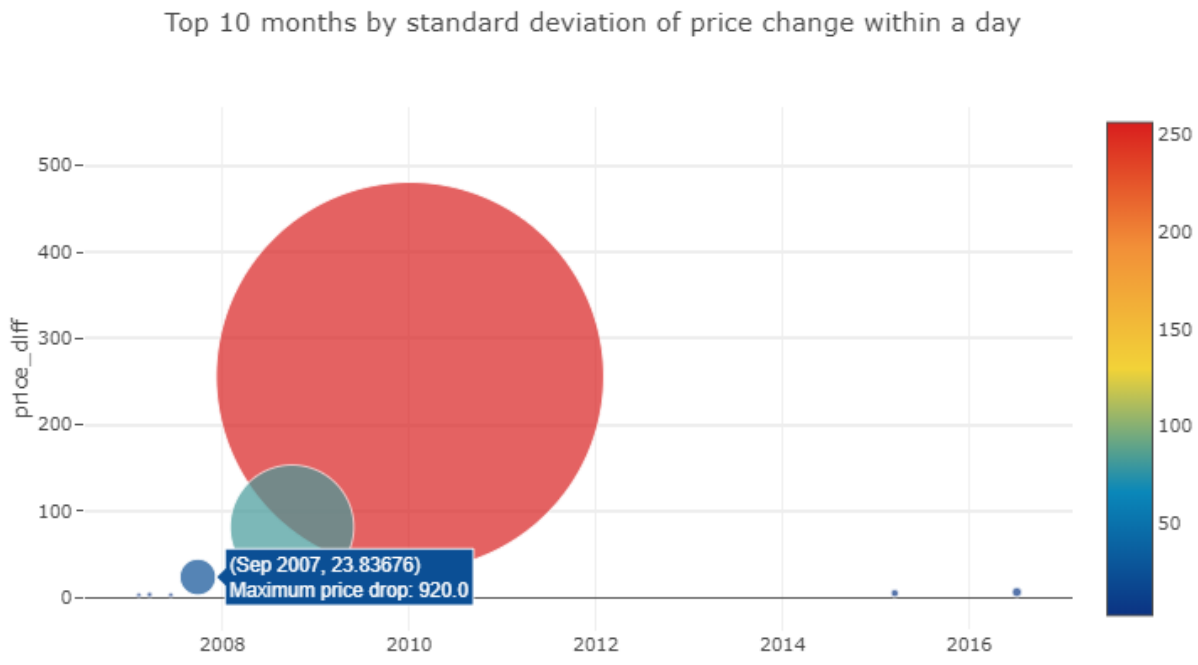
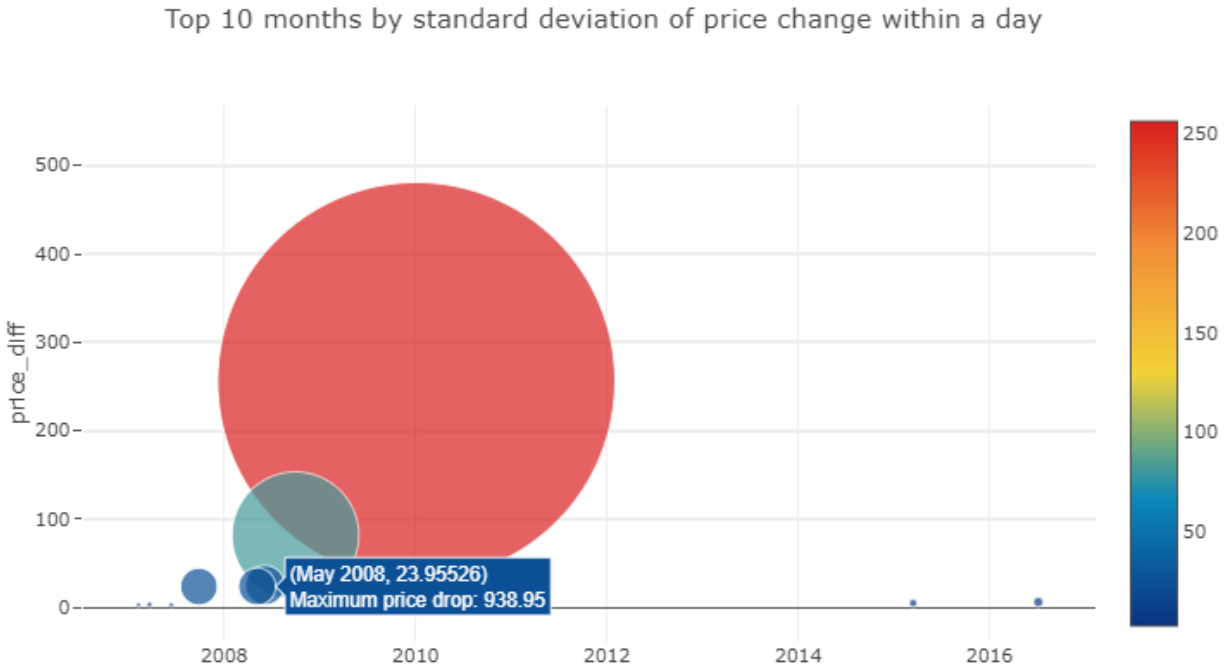
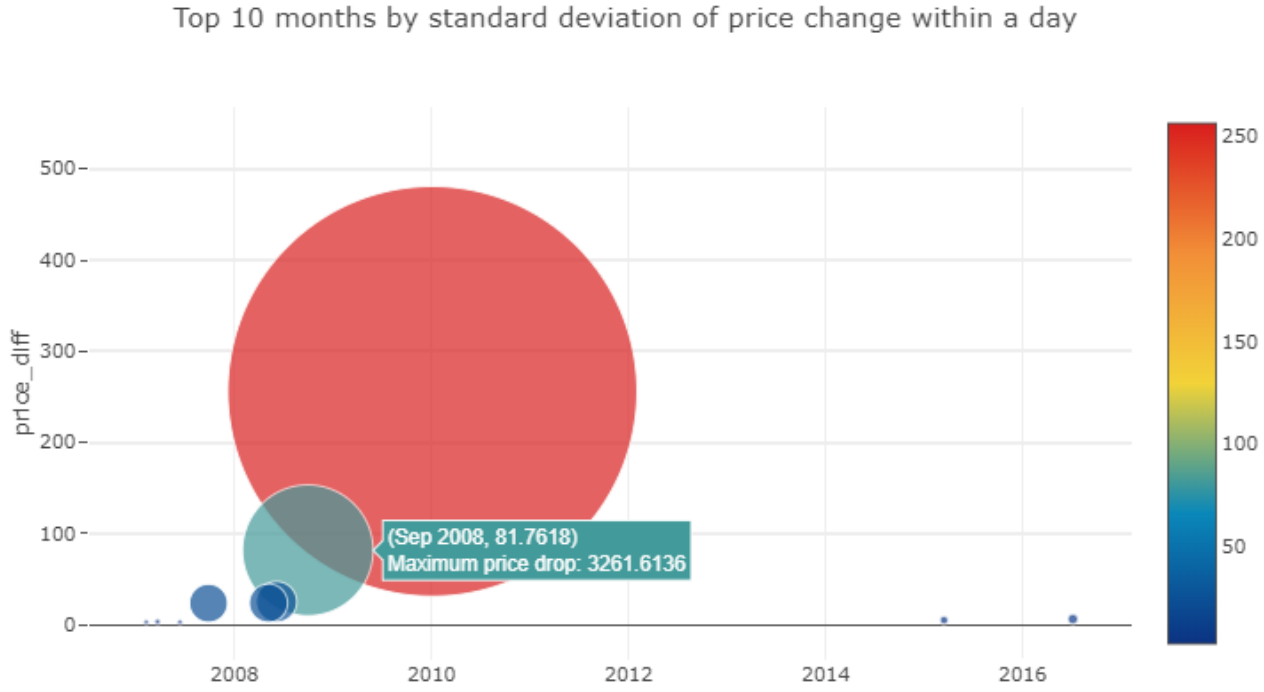
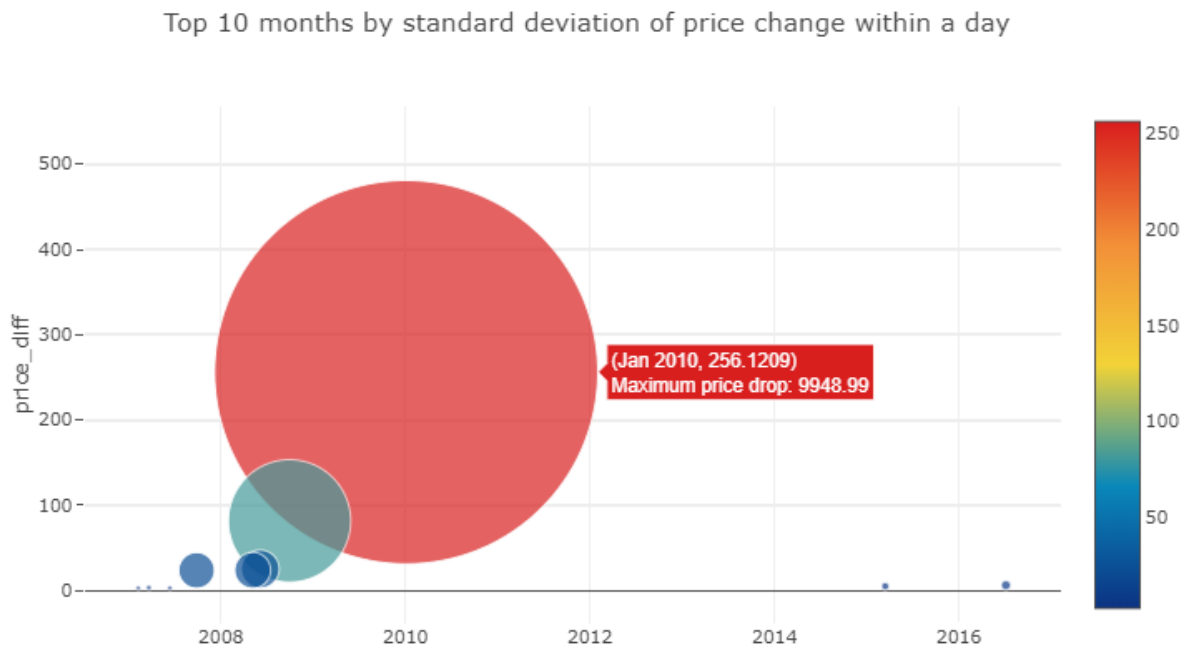


Figure 7: Price Change Within a Day


Top 10 months by standard deviation of price change within a day
price_diff
500
400
300
200
100
0
2008
2010
2012
2014
2016
(May 2008, 23.95526)
Maximum price drop: 938.95
250
200
150
100
50


Top 10 months by standard deviation of price change within a day
price_diff
500
400
300
200
100
0
2008
2010
2012
2014
2016
(Sep 2008, 81.7618)
Maximum price drop: 3261.6136
250
200
150
100
50



We can see huge price fluctuations when market crashed. Just think about it... **But this is wrong!** There was no huge crash on January 2010... Let's dive into the data!

5.3.5 Possible data errors

At first let's simply sort data by the difference between open and close prices.

```
market_train_df.sort_values('price_diff')[:10]
```

1]:

	time	assetCode	assetName	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1	re
1127598	2010-01-04 22:00:00+00:00	TW.N	Towers Watson & Co	223136.0	50.00	9998.9900	-0.058470	185.988360	-(
627547	2008-09-29 22:00:00+00:00	BK.N	Bank of New York Mellon Corp	18718479.0	26.50	3288.1136	-0.271578	99.125262	-(
502997	2008-06-05 22:00:00+00:00	AHG.N	Apria Healthcare Group Inc	801892.0	17.29	999.9900	0.009930	58.523214	-(
471381	2008-05-06 22:00:00+00:00	CEPH.O	Cephalon Inc	4846.0	61.04	999.9900	0.014628	15.547907	0
242847	2007-09-27 22:00:00+00:00	EXH.N	Archrock Inc	490100.0	79.99	999.9900	0.022236	11.658101	0
3264631	2015-03-16 22:00:00+00:00	TECD.O	Tech Data Corp	674385.0	56.59	263.8000	0.036447	3.868057	0
7273	2007-02-08 22:00:00+00:00	BA.N	Boeing Co	5155700.0	89.52	200.0000	-0.009186	1.207749	-(
375899	2008-02-06 22:00:00+00:00	CME.N	CME Group Inc	8676040.0	485.25	583.4900	-0.175866	-0.061536	-(
628075	2008-09-29 22:00:00+00:00	IBM.N	International Business Machines Corp	9586679.0	114.46	191.9800	-0.041534	0.637915	0
3565205	2015-11-20 22:00:00+00:00	CMG.N	Chipotle Mexican	5023617.0	536.19	612.0000	-0.123171	0.027863	-(

So, price of "Towers Watson & Co" shares was almost 10k... I think this is simply an error in data.

What about Bank of New York Mellon Corp?

Let's see data by Yahoo

The Bank of New York Mellon Corporation (BK) ☆ Add to watchlist

NYSE - NYSE Delayed Price. Currency in USD

50.99 -0.55 (-1.07%)

At close: September 28 4:00PM EDT

Buy Sell

Summary Chart Conversations Statistics Historical Data Profile Financials Analysis Options Holders Sustainability

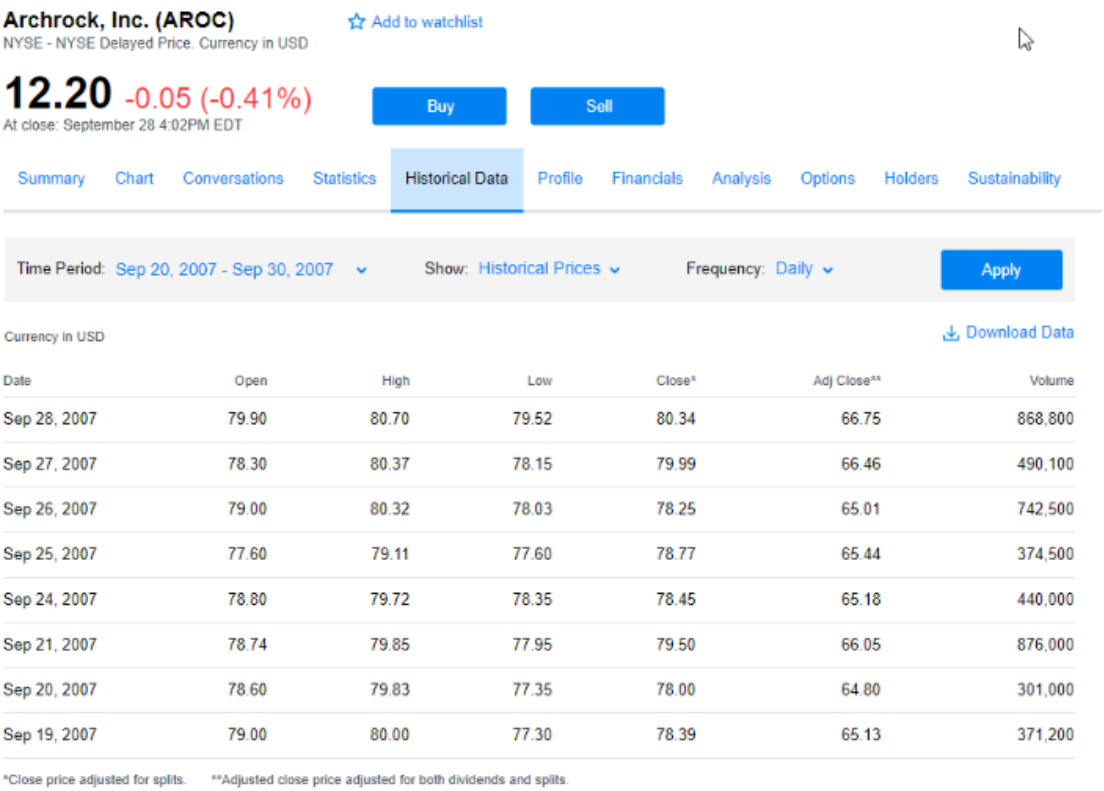
Time Period: Sep 20, 2008 - Oct 05, 2008 Show: Historical Prices Frequency: Daily Apply

Currency in USD Download Data

Date	Open	High	Low	Close*	Adj Close**	Volume
Oct 03, 2008	32.61	33.82	29.46	29.78	24.63	10,708,200
Oct 02, 2008	33.55	34.05	30.18	30.65	25.35	8,422,600
Oct 01, 2008	31.50	34.71	30.75	33.66	27.84	9,929,400
Sep 30, 2008	29.68	33.00	28.02	32.58	26.95	16,897,000
Sep 29, 2008	34.37	35.39	24.18	26.50	21.92	18,718,500
Sep 26, 2008	32.84	36.86	32.80	36.38	30.09	7,547,400
Sep 25, 2008	33.25	36.48	32.31	35.44	29.31	9,375,900
Sep 24, 2008	33.15	33.15	30.90	32.53	26.90	7,793,900
Sep 23, 2008	32.58	33.59	31.70	31.81	26.31	7,538,700

There were no spikes.

Another case is with cost equal to 999, such numbers are usually suspicious. Let's look at Archrock Inc - no spikes there as well.



Archrock, Inc. (AROC)
NYSE - NYSE Delayed Price. Currency in USD
Add to watchlist

12.20 -0.05 (-0.41%)
At close: September 28 4:02PM EDT

Buy | Sell

Summary | Chart | Conversations | Statistics | Historical Data | Profile | Financials | Analysis | Options | Holders | Sustainability

Time Period: Sep 20, 2007 - Sep 30, 2007 | Show: Historical Prices | Frequency: Daily | Apply

Currency in USD | Download Data

Date	Open	High	Low	Close*	Adj Close**	Volume
Sep 28, 2007	79.90	80.70	79.52	80.34	66.75	868,800
Sep 27, 2007	78.30	80.37	78.15	79.99	66.46	490,100
Sep 26, 2007	79.00	80.32	78.03	78.25	65.01	742,500
Sep 25, 2007	77.60	79.11	77.60	78.77	65.44	374,500
Sep 24, 2007	78.80	79.72	78.35	78.45	65.18	440,000
Sep 21, 2007	78.74	79.85	77.95	79.50	66.05	876,000
Sep 20, 2007	78.60	79.83	77.35	78.00	64.80	301,000
Sep 19, 2007	79.00	80.00	77.30	78.39	65.13	371,200

*Close price adjusted for splits. **Adjusted close price adjusted for both dividends and splits.

5.3.6 So, let's try to find strange cases

In [12]:
```
market_train_df['close_to_open'] =  np.abs(market_train_df['close'] / market_train_df['open'])
```

In [13]:
```
print(f"In {(market_train_df['close_to_open'] >= 1.2).sum()} lines price increased by 20% or mo
re.")
print(f"In {(market_train_df['close_to_open'] <= 0.8).sum()} lines price decreased by 20% or mo
re.")
```

```
In 1211 lines price increased by 20% or more.
In 778 lines price decreased by 20% or more.
```

Well, this isn't much considering we have more than 4 million lines and a lot of these cases are due to price falls during market crash. Well just need to deal with outliers.

[14]:

```
print(f"In {(market_train_df['close_to_open'] >= 2).sum()} lines price increased by 100% or mor
e.")
print(f"In {(market_train_df['close_to_open'] <= 0.5).sum()} lines price decreased by 100% or m
ore.")
```

```
In 38 lines price increased by 100% or more.
In 16 lines price decreased by 100% or more.
```

5.3.7 For a quick fix I'll replace outliers in these lines with mean open or close price of this company

```
market_train_df['assetName_mean_open'] = market_train_df.groupby('assetName')['o
pen'].transform('mean')
```

```
market_train_df['assetName_mean_close'] = market_train_df.groupby('assetName')['c
lose'].transform('mean')
```

```
# if open price is too far from mean open price for this company, replace it. Otherwise
replace close price.
```

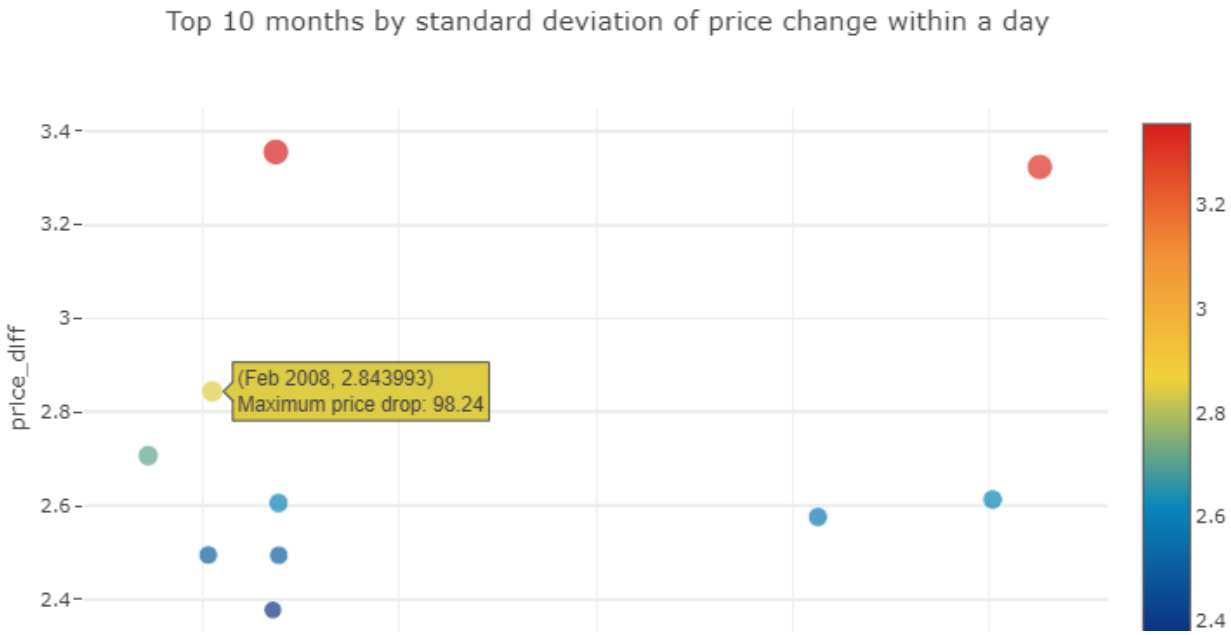
```
for i, row in market_train_df.loc[market_train_df['close_to_open'] >= 2].iterrows():
    if np.abs(row['assetName_mean_open'] - row['open']) > np.abs(row['assetName_me
an_close'] - row['close']):
        market_train_df.iloc[i,5] = row['assetName_mean_open']
    else:
        market_train_df.iloc[i,4] = row['assetName_mean_close']
```

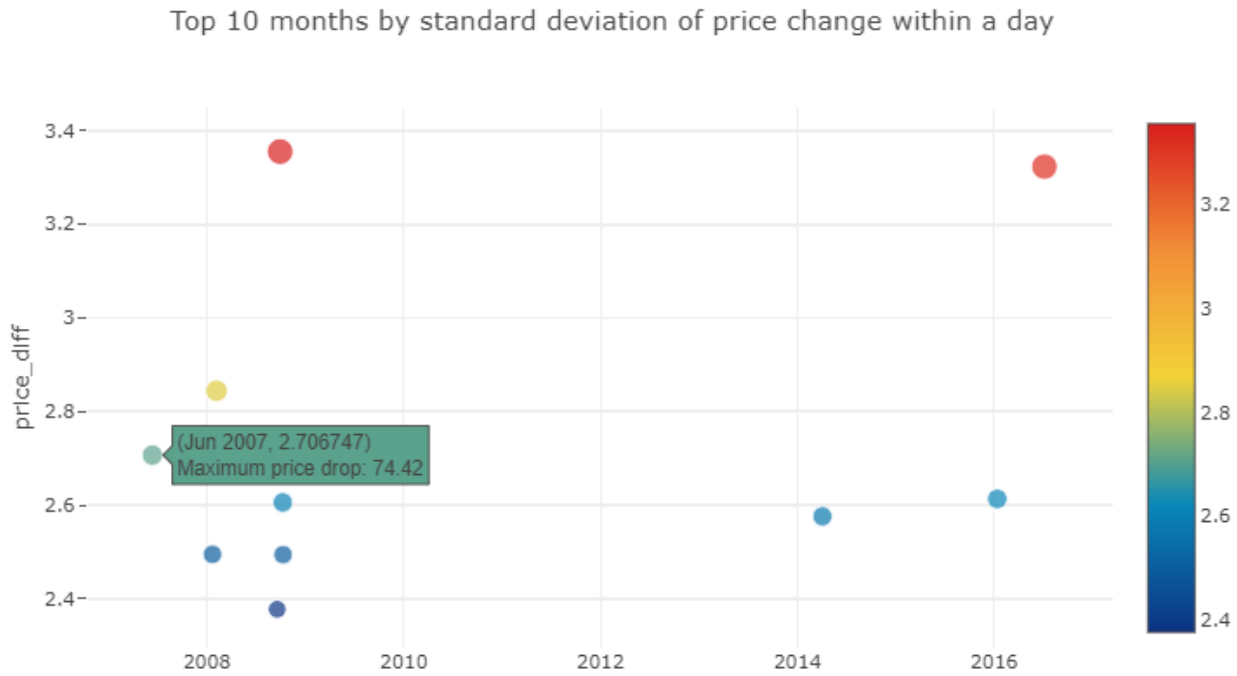
```
for i, row in market_train_df.loc[market_train_df['close_to_open'] <= 0.5].iterrows():
    if np.abs(row['assetName_mean_open'] - row['open']) > np.abs(row['assetName_me
an_close'] - row['close']):
        market_train_df.iloc[i,5] = row['assetName_mean_open']
    else:
        market_train_df.iloc[i,4] = row['assetName_mean_close']
```

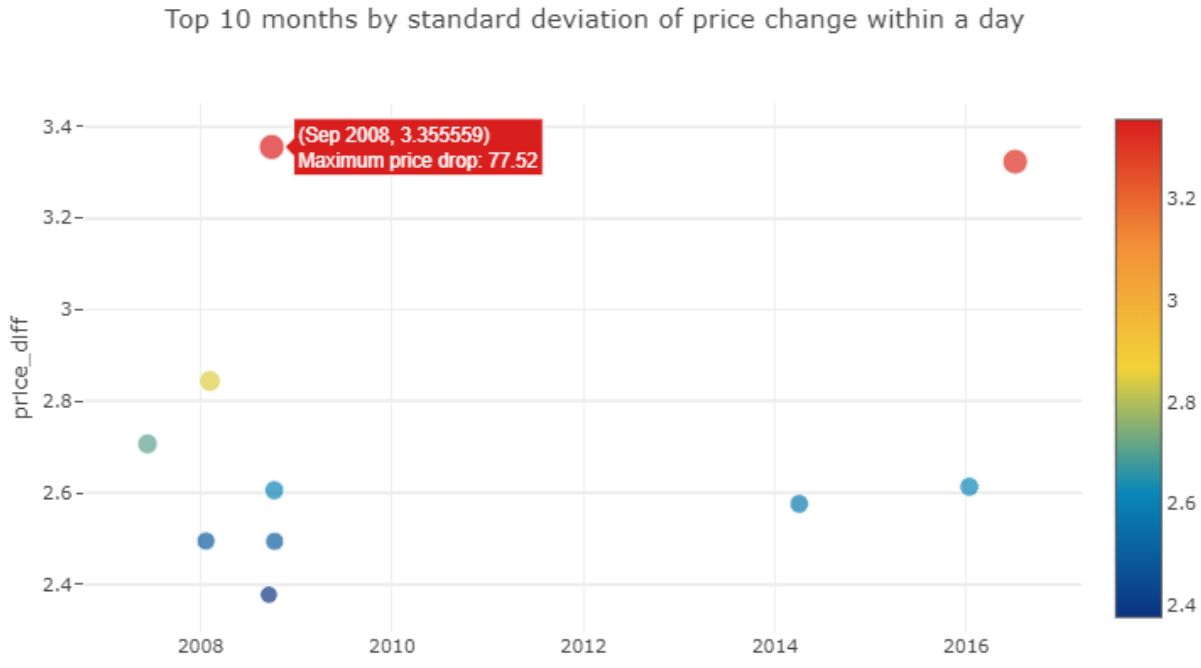
Now let's try to build that graph again.

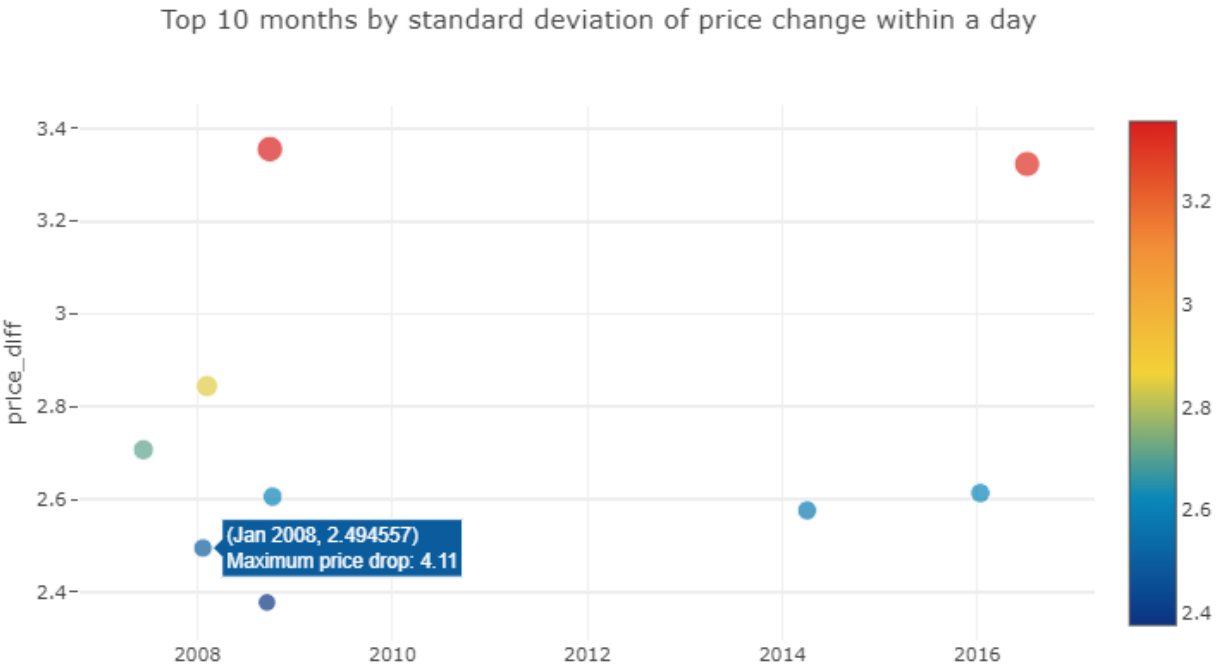
```
market_train_df['price_diff']=market_train_df['close']-market_train_df['open']

grouped=market_train_df.groupby(['time']).agg({'price_diff':['std','min']}).reset_index
()

g = grouped.sort_values(('price_diff', 'std'), ascending=False)[:10]

g['min_text'] = 'Maximum price drop: ' + (-1 * np.round(g['price_diff']['min'],
2)).astype(str)


trace = go.Scatter(

 x = g['time'].dt.strftime(date_format='%Y-%m-%d').values,

 y = g['price_diff']['std'].values,

 mode='markers',

 marker=dict(

size = g['price_diff']['std'].values * 5,

color = g['price_diff']['std'].values,

colorscale='Portland',

 showscale=True),

text = g['min_text'].values

#text = f"Maximum price drop: {g['price_diff']['min'].values}"

#g['time'].dt.strftime(date_format='%Y-%m-%d').values

)

data = [trace]

layout= go.Layout(

        autosize= True,

        title= 'Top 10 months by standard deviation of price change within a day',
```

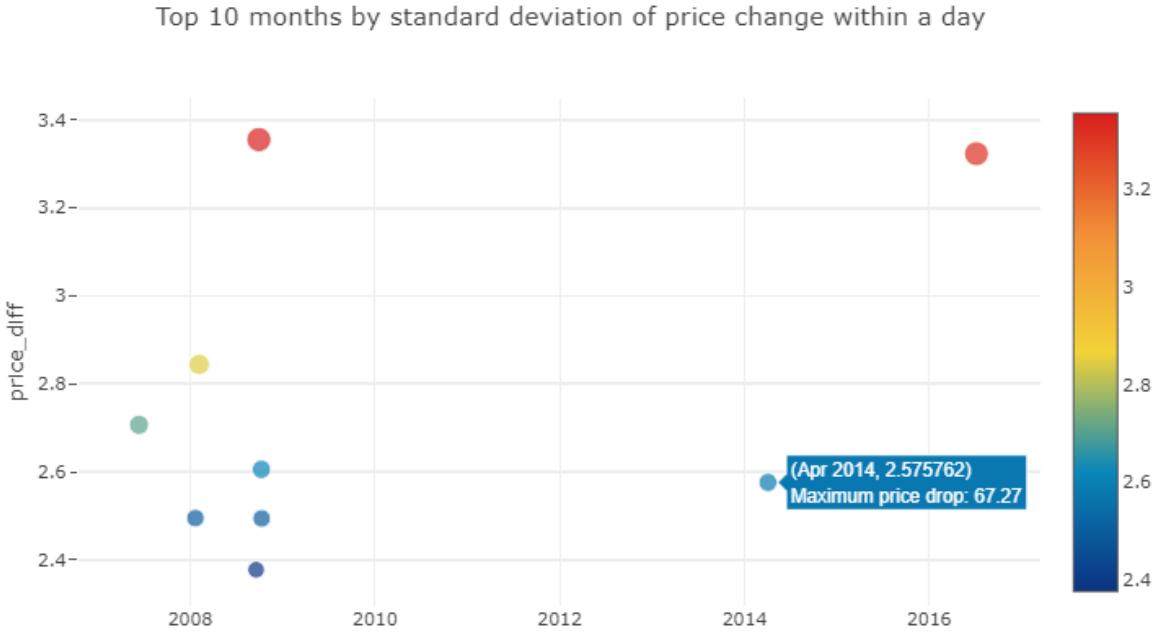
```
        hovermode= 'closest',

        yaxis=dict(

          title= 'price_diff',

       ticklen= 5,
      gridwidth= 2,
    ),

    showlegend= False
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig,filename='scatter2010')
```

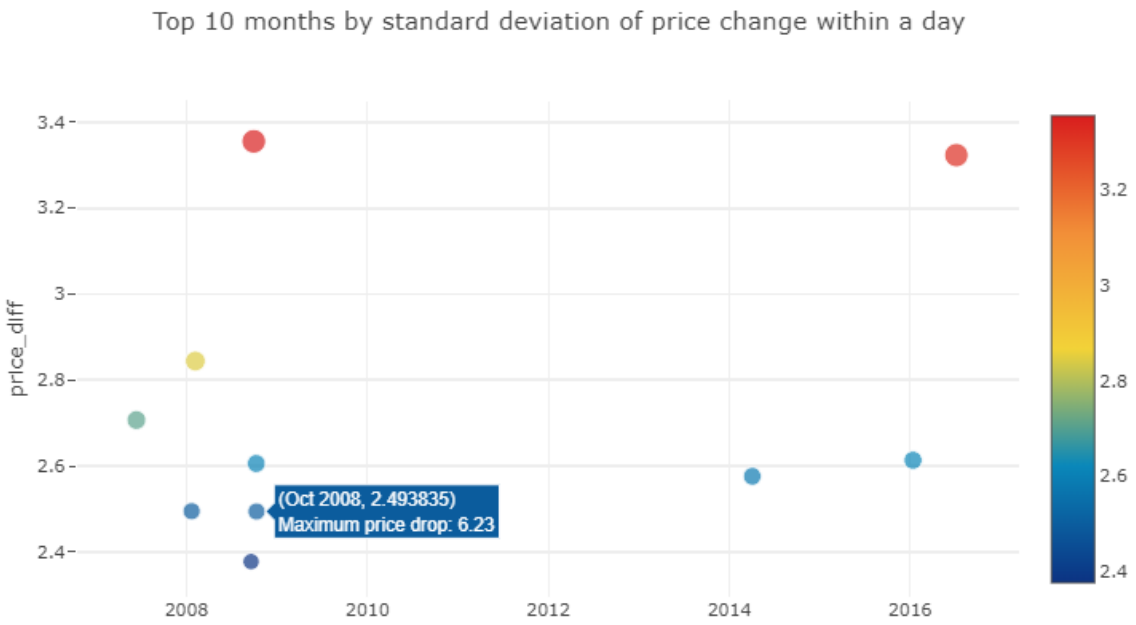


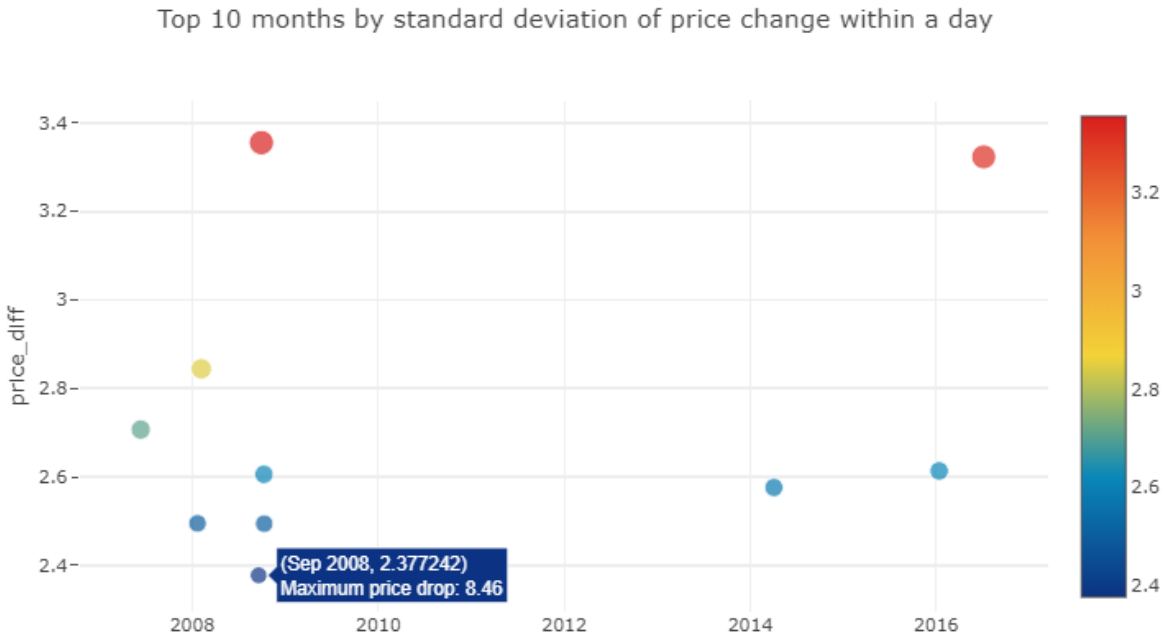

Top 10 months by standard deviation of price change within a day
price_diff
3.4
3.2
3
2.8
2.6
2.4
2008
2010
2012
2014
2016
(Jun 2007, 2.706747)
Maximum price drop: 74.42
3.2
3
2.8
2.6
2.4

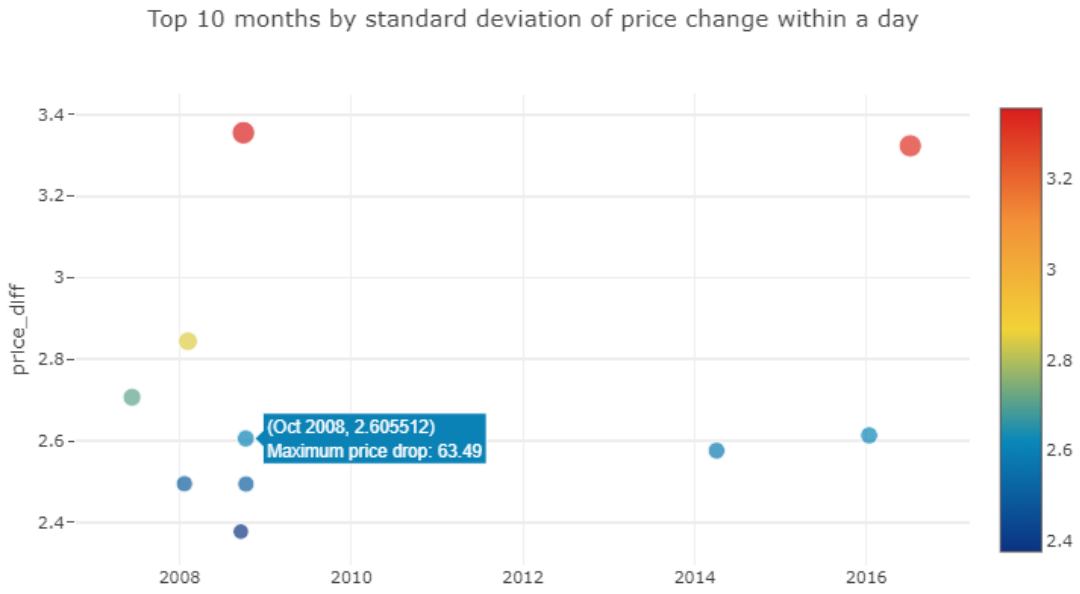

Top 10 months by standard deviation of price change within a day
price_diff
3.4
3.2
3
2.8
2.6
2.4
2008
2010
2012
2014
2016
(Sep 2008, 3.355559)
Maximum price drop: 77.52
3.2
3
2.8
2.6
2.4

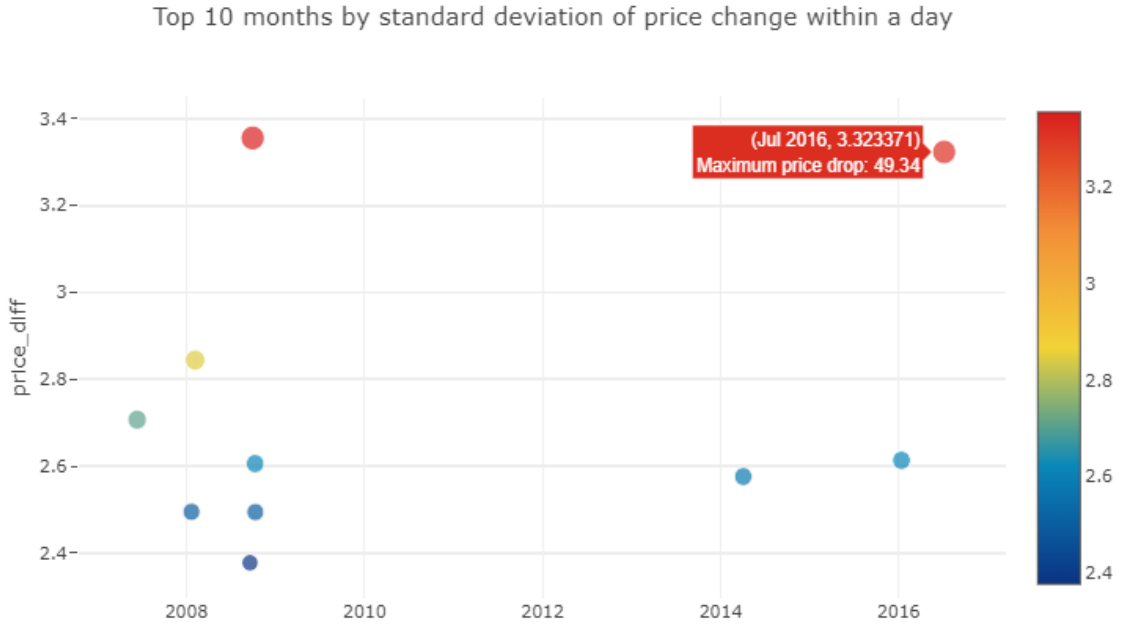

Top 10 months by standard deviation of price change within a day
price_diff
3.4
3.2
3
2.8
2.6
2.4
2008
2010
2012
2014
2016
(Jan 2008, 2.494557)
Maximum price drop: 4.11
3.2
3
2.8
2.6
2.4


Top 10 months by standard deviation of price change within a day
price_diff
3.4
3.2
3
2.8
2.6
2.4
2008
2010
2012
2014
2016
(Apr 2014, 2.575762)
Maximum price drop: 67.27
3.2
3
2.8
2.6
2.4


Top 10 months by standard deviation of price change within a day
price_diff
3.4
3.2
3
2.8
2.6
2.4
2008
2010
2012
2014
2016
(Oct 2008, 2.493835)
Maximum price drop: 6.23
3.2
3
2.8
2.6
2.4


Top 10 months by standard deviation of price change within a day
price_diff
3.4
3.2
3
2.8
2.6
2.4
2008
2010
2012
2014
2016
(Sep 2008, 2.377242)
Maximum price drop: 8.46
3.2
3
2.8
2.6
2.4





Now the graph is much more reasonable

5.3.8 Let's look at out target variable

```
data = []

for i in [0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95]:

    price_df = market_train_df.groupby('time')['returnsOpenNextMktres10'].quantile(i).reset_i
ndex()

    data.append(go.Scatter(
        x = price_df['time'].dt.strftime(date_format='%Y-%m-%d').values,
        y = price_df['returnsOpenNextMktres10'].values,
        name = f'{i} quantile'
    ))


layout = go.Layout(dict(title = "Trends of returnsOpenNextMktres10 by quantiles",
                  xaxis = dict(title = 'Month'),
                  yaxis = dict(title = 'Price (USD)'),
                  ),legend=dict(
                orientation="h"),)


py.iplot(dict(data=data, layout=layout), filename='basic-line')
```

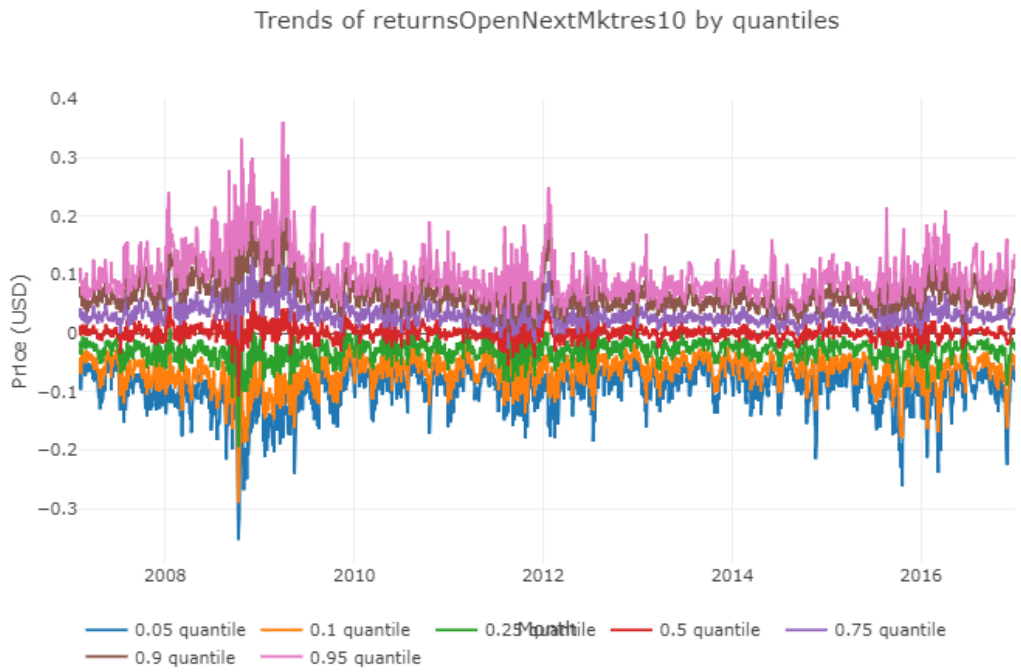


Figure 8: Trend of returns Open Next Market

We can see that quantiles have a high deviation, but mean value doesn't change much.

5.3.9 I think it is time to throw an old part of dataset. Let's leave only data since 2010 year, this way we will get rid of the data of the biggest crisis.

Let's look at the target variable now.

```
data = []

market_train_df = market_train_df.loc[market_train_df['time'] >= '2010-01-01 22:00:00+000
0']

price_df = market_train_df.groupby('time')['returnsOpenNextMktres10'].mean().reset_index()

data.append(go.Scatter(
    x = price_df['time'].dt.strftime(date_format='%Y-%m-%d').values,
    y = price_df['returnsOpenNextMktres10'].values,
    name = f'{i} quantile'
))


layout = go.Layout(dict(title = "Treand of returnsOpenNextMktres10 mean",
```

```
            xaxis = dict(title = 'Month'),
            yaxis = dict(title = 'Price (USD)'),
            ),legend=dict(
           orientation="h"),)
py.iplot(dict(data=data, layout=layout), filename='basic-line')
```

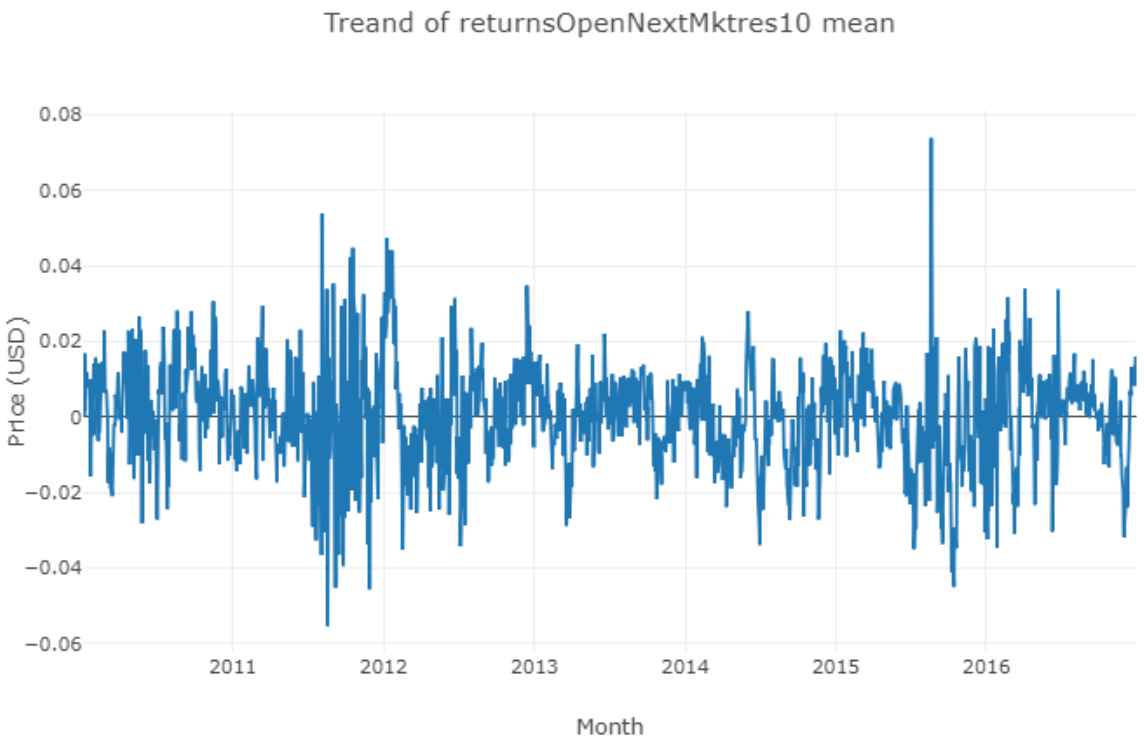


Figure 8: Trend of return open next market

Fluctuations seem to be high, but in fact they are lower that 8 percent. In fact, it looks like a random noise...

Now let's remember the description:

The market data contains a variety of returns calculated over different timespans. All of the returns in this set of market data have these properties:

- Returns are always calculated either open-to-open (from the opening time of one trading day to the open of another) or close-to-close (from the closing time of one trading day to the open of another).
- Returns are either raw, meaning that the data is not adjusted against any benchmark, or market-residualized (Mktres), meaning that the movement of the

market as a whole has been accounted for, leaving only movements inherent to the instrument.

- Returns can be calculated over any arbitrary interval. Provided here are 1 day and 10-day horizons.

Returns are tagged with 'Prev' if they are backwards looking in time, or 'Next' if forwards looking.

5.3.10 The market data contains a variety of returns. Let's have a look at means of these variables.

```
data = []

for col in ['returnsClosePrevRaw1', 'returnsOpenPrevRaw1',
      'returnsClosePrevMktres1', 'returnsOpenPrevMktres1',
      'returnsClosePrevRaw10', 'returnsOpenPrevRaw10',
      'returnsClosePrevMktres10', 'returnsOpenPrevMktres10',
      'returnsOpenNextMktres10']:

   df = market_train_df.groupby('time')[col].mean().reset_index()

   data.append(go.Scatter(
      x = df['time'].dt.strftime(date_format='%Y-%m-%d').values,
      y = df[col].values,
      name = col
   ))

layout = go.Layout(dict(title = "Treand of mean values",
                  xaxis = dict(title = 'Month'),
                  yaxis = dict(title = 'Price (USD)'),
                  ),legend=dict(
                orientation="h"),)


py.iplot(dict(data=data, layout=layout), filename='basic-line')
```

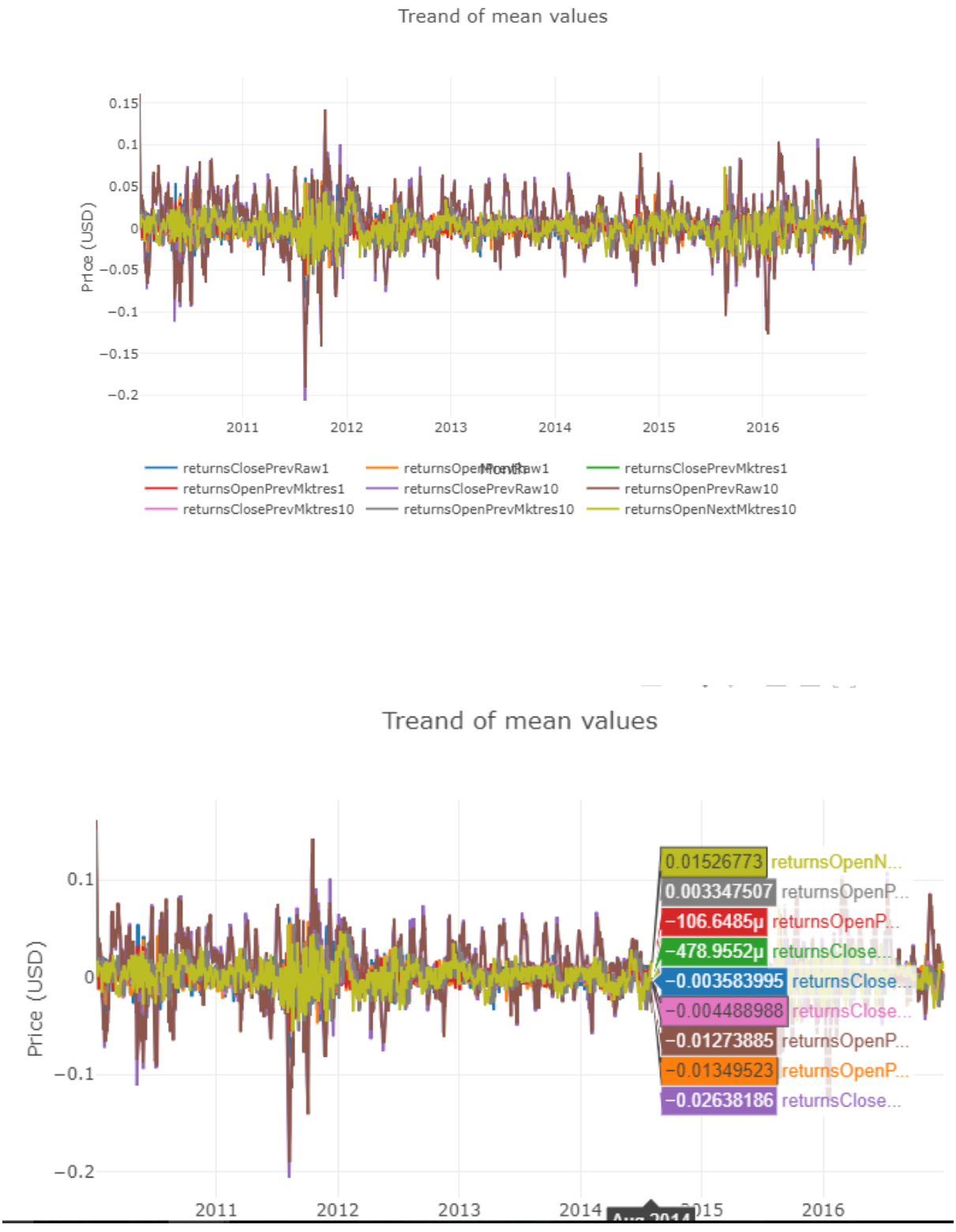


Figure 9: Trend of Mean Value

Well, for me it is difficult to interpret this, but it seems that returns for previous 10 days fluctuate the most.

5.4. News Data EDA

5.4.1 News Headlines

The file is too huge to work with text directly, so let's see a wordcloud of the last 100000 headlines.

```
text = ' '.join(news_train_df['headline'].str.lower().values[-1000000:])

wordcloud = WordCloud(max_font_size=None, stopwords=stop, background_color='white', width=1200, height=1000).generate(text)


plt.figure(figsize=(12, 8))
plt.imshow(wordcloud)
plt.title('Top words in headline')
plt.axis("off")
plt.show()
```

Figure 10: Word Cloud

```
# Let's also limit the time period
news_train_df = news_train_df.loc[news_train_df['time'] >= '2010-01-01 22:00:00+0000']



(news_train_df['urgency'].value_counts() / 1000000).plot('bar');
plt.xticks(rotation=30);
plt.title('Urgency counts (mln)');
```

5.4.2 Urgency Column

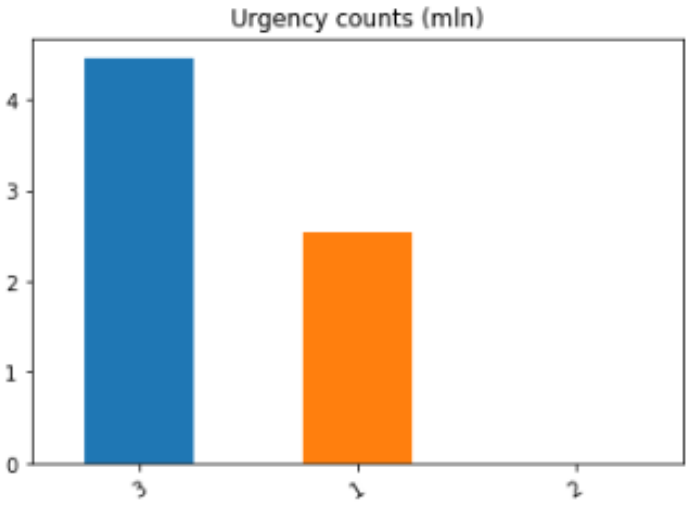


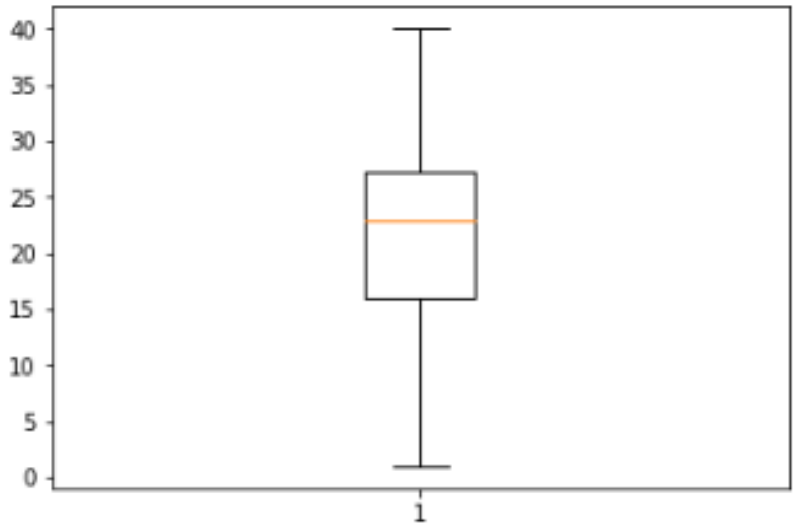
Figure 11: Urgency columns

Well, it seems that in fact urgency "2" is almost never used

5.4.3 SentenceCount

```
news_train_df['sentence_word_count'] =  news_train_df['wordCount'] / news_train_df['sentenceCount']

plt.boxplot(news_train_df['sentence_word_count'][news_train_df['sentence_word_count'] < 40]);
```

There are some big outliers but sentences mostly have 15-25 words in them.



5.4.4 Provider

```
news_train_df['provider'].value_counts().head(10)
```

```
RTRS    5517624
PRN      503267
BSW      472612
GNW      145309
MKW      129621
LSE       64250
HIIS      56489
RNS       39833
CNW       30779
ONE       25233
Name: provider, dtype: int64
```

It isn't surprising that Reuters is the most common news provider

5.4.5 Head Line Tag

```
(news_train_df['headlineTag'].value_counts() / 1000)[:10].plot('barh');
plt.title('headlineTag counts (thousands)');
```

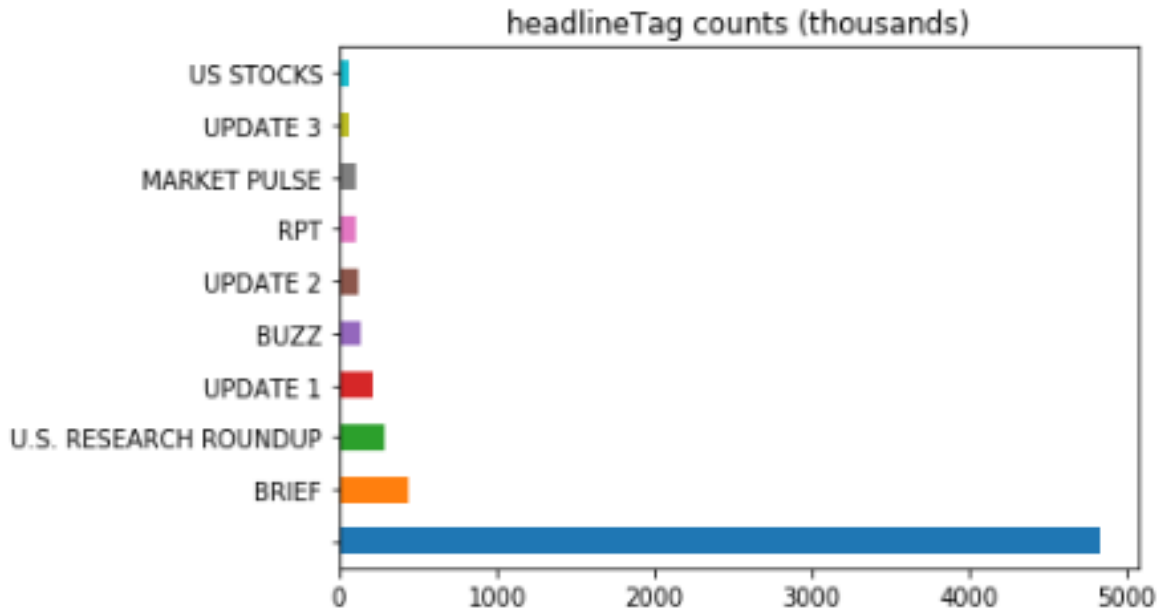


Figure 12: Headline Tag

Well, most news are tagless.

5.4.6 Sentiments

```
for i, j in zip([-1, 0, 1], ['negative', 'neutral', 'positive']):
    df_sentiment = news_train_df.loc[news_train_df['sentimentClass'] == i, 'assetName
']
    print(f'Top mentioned companies for {j} sentiment are:')
    print(df_sentiment.value_counts().head(5))
    print('')
```

```
Top mentioned companies for negative sentiment are:
Apple Inc                 22518
JPMorgan Chase & Co       20647
BP PLC                    19328
Goldman Sachs Group Inc   17955
Bank of America Corp      17704
Name: assetName, dtype: int64

Top mentioned companies for neutral sentiment are:
HSBC Holdings PLC    19462
Credit Suisse AG     14632
Deutsche Bank AG     12959
Barclays PLC         12414
Apple Inc            10994
Name: assetName, dtype: int64

Top mentioned companies for positive sentiment are:
Apple Inc               19020
Barclays PLC            18051
Royal Dutch Shell PLC   15484
General Electric Co     14163
Boeing Co               14080
Name: assetName, dtype: int64
```

I think it is quite funny that Apple is a company with most both negative and positive sentiments.

At first, I was sad that we don't have access to the texts of the news, but I have realized that we won't be able to use them anyway due to kernel memory limitations.

5.5 Pre-processing of Data

5.5.1 Changing Time Format

Change time in year, months, date, hour and in minute format

```
market_train_df['time'] =pd.to_datetime(market_train_df['time'], format='%Y-%m-%d %H:%M')

news_train_df['time'] = pd.to_datetime(news_train_df['time'], format='%Y-%m-%d %H:%M')
```

5.5.2 Market Data

- **Impute missing value**

- **Market values:** All null data comes from market adjusted columns. We fill them up with the raw values in the same row.

```
column_market=['returnsClosePrevMktres1','returnsOpenPrevMktres1','returnsClosePrevMktres10', 'returnsOpenPrevMktres10']

column_raw=['returnsClosePrevRaw1','returnsOpenPrevRaw1','returnsClosePrevRaw10', 'returnsOpenPrevRaw10']

for i in range(len(column_raw)):

    market_train_df[column_market[i]]=market_train_df[column_market[i]].fillna(market_train_df[column_raw[i]])
```

- **Outliers-Returns:** Return should not exceed 50% or falls below 50%. If it does, it is either noise, or extreme data that will confuse our prediction later on. We remove these extreme data.

```
print ('Removing outliers ...')

column_return = column_market + column_raw + ['returnsOpenNextMktres10']

orig_len = market_train_df.shape[0]
```

```
for column in column_return:
    market_train_df = market_train_df.loc[market_train_df[column]>=-2]
    market_train_df = market_train_df.loc[market_train_df[column]<=2]
new_len = market_train_df.shape[0]
rmv_len = np.abs(orig_len-new_len)
print ('There were %i lines removed' %rmv_len)
```

- **Remove strange data**:

  Here we remove data with unknown asset name or asset codes with strange behaviour.

```
print('Removing strange data ...')
orig_len = market_train_df.shape[0]
market_train_df=market_train_df[~market_train_df['assetCode'].isin(['PGN.N','EBRYY.OB'])]
#market_train_df= market_train_df[~market_train_df['assetName'].isin(['Unknown'])]
new_len = market_train_df.shape[0]
rmv_len = np.abs(orig_len-new_len)
print('There were %i lines removed' %rmv_len)
```

5.5.3 News data

- **Remove outliers**: apply a clip filter to reduce too extreme data

```
# Function to remove outliers
def remove_outliers(data_frame, column_list, low=0.02, high=0.98):
    for column in column_list:
        this_column = data_frame[column]
        quant_df = this_column.quantile([low,high])
```

```
        low_limit = quant_df[low]

        high_limit = quant_df[high]


        data_frame[column]=data_frame[column].clip(lower=low_limit,upper=high_l
        imit)

    return data_frame
```

- **Remove outlier**

```
columns_outlier=['takeSequence','bodySize','sentenceCount','wordCount','sentimentW
ordCount','firstMentionSentence','noveltyCount12H','noveltyCount24H','noveltyCount
3D','noveltyCount5D','noveltyCount7D','volumeCounts12H','volumeCounts24H',\'vol
umeCounts3D','volumeCounts5D','volumeCounts7D']

print('Clipping news outliers ...')

news_train_df = remove_outliers(news_train_df, columns_outlier)
```

5.5.4 Features engineering

Data processing function

Here we make a function process both market and news data, then merge them.

```
asset_code_dict = {k: v for v, k in enumerate(market_train_df['assetCode'].unique())}

drop_columns = [col for col in news_train_df.columns if col not in ['sourceTimestamp',
'urgency','takeSequence','bodySize','companyCount','sentenceCount','firstMentionSent
ence', 'relevance','firstCreated', 'assetCodes']]

columns_news =
['firstCreated','relevance','sentimentClass','sentimentNegative','sentimentNeutral',
'sentimentPositive','noveltyCount24H','noveltyCount7D','volumeCounts24H','volume
Counts7D','assetCodes','sourceTimestamp', 'assetName','audiences', 'urgency',
'takeSequence', 'bodySize', 'companyCount',    'sentenceCount',
'firstMentionSentence','time']
```

- **Data processing function**

```
def data_prep(market_df, news_df):

   market_df['date'] =pd.to_datetime(market_train_df['time'], format='%Y-%m-%d
%H:%M')

   market_df['close_to_open'] = market_df['close'] / market_df['open']

   market_df.drop(['time'], axis=1, inplace=True)



   news_df = news_df[columns_news]

    news_df['sourceTimestamp']=pd.to_datetime(news_df['sourceTimestamp'],
format='%Y-%m-%d %H:%M')

   news_df['firstCreated'] = pd.to_datetime(news_df['firstCreated'], format='%Y-%m-
%d %H:%M')

   news_df['assetCodesLen'] = news_df['assetCodes'].map(lambda x: len(eval(x)))

   news_df['assetCodes'] = news_df['assetCodes'].map(lambda x: list(eval(x))[0])


news_df['asset_sentiment_count']=news_df.groupby(['assetName','sentimentClass'])['t
ime'].transform('count')

   news_df['len_audiences'] = news_train_df['audiences'].map(lambda x: len(eval(x)))

   kcol = ['firstCreated', 'assetCodes']

   news_df = news_df.groupby(kcol, as_index=False).mean()

   market_df = pd.merge(market_df, news_df, how='left', left_on=['date', 'assetCode'],

                        right_on=['firstCreated', 'assetCodes'])

   del news_df

   market_df['assetCodeT'] = market_df['assetCode'].map(asset_code_dict)

  market_df =
market_df.drop(columns=['firstCreated','assetCodes','assetName']).fillna(0)

   return market_df
```

5.5.5 Merging Market_Data and News_Data

```
print('Merging data ...')

market_train_df = data_prep(market_train_df, news_train_df)
```

5.5.6 Data selection

Looking at the statistics, most data behave homogeneously after 2010 (volume increase, price increase, etc.). However, before 2010, due to the burst of the housing bubble that leads to the financial crisis in 2008, the data behaves differently. So, the question to make the right prediction for this problem is: **Will there be a financial crisis in the next 6 months?** If the answer is **Yes**, then we include data before 2010. If the answer is **No**, then we exclude them.

I choose **No** as the answer and proceed from that.

We then perform feature selection. Feature scaling is not needed since we plan to use XGBoost, lightgbm - a tree-based model, which do not require standardization.

I tried using a regressor model, but a problem is that it gives close-to-0 values for most of prediction. Thus, I convert this problem into a classification problem: 0 for negative return and 1 for positive return.

Separating Numerical columns and categorical columns

5.5.6.1 Numerical Columns: -

Numerical data is information that is something that is measurable. It is always collected in number form, although there are other types of data that can appear in number form. An example of numerical data would be the number of people that attended the movie theatre over the course of a month.

5.5.6.2.Categorical Columns: -

Categorical variables represent types of data which may be divided into groups. Examples of categorical variables are race, sex, age group, and educational level. While the latter two variables may also be considered in a numerical manner by using exact values for age and highest grade completed, it is often more informative to categorize such variables into a relatively small number of groups.

Select numerical features and categorical features from data

```
num_columns=['volume','close','open','returnsClosePrevRaw1','returnsOpenPrevRaw1
','returnsClosePrevMktres1','returnsOpenPrevMktres1','returnsClosePrevRaw10',
'returnsOpenPrevRaw10','returnsClosePrevMktres10','returnsOpenPrevMktres10',
'close_to_open','urgency','companyCount', 'takeSequence', 'bodySize', 'sentenceCount',
'relevance','sentimentClass','sentimentNegative','sentimentNeutral','sentimentPositive',
'noveltyCount24H','noveltyCount7D','volumeCounts24H','volumeCounts7D','assetCo
desLen',  'asset_sentiment_count', 'len_audiences']

cat_columns = ['assetCodeT']

feature_columns = num_columns + cat_columns
```

5.5.7 Data Scaling

It is a step of Data Pre-Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a range. Sometimes, it also helps in speeding up the calculations in an algorithm.

$$z = \frac{x - \mu}{\sigma}$$

Formula used in backend, standardisation replaces the value by their z scores.

```
# Scaling of data

from sklearn.preprocessing import StandardScaler, MinMaxScaler


data_scaler = StandardScaler()

#market_train_df[num_columns]                                    =
data_scaler.fit_transform(market_train_df[num_columns])

#data_scaler = MinMaxScaler()

market_train_df[num_columns]                                    =
data_scaler.fit_transform(market_train_df[num_columns])
```

5.5.8 Making Dependent variable

In this problem dependent variable is continuous so I am converting dependent variables into 0 and 1.

```
### making dependent variables
market_train_df['returnsOpenNextMktres10']=np.where(market_train_df.returnsOpen
NextMktres10 <= 0,0,1)

market_train_df.returnsOpenNextMktres10.value_counts()
```

5.5.9 Balancing the data Set

```
### counting 0 and 1
c = market_train_df['returnsOpenNextMktres10'].value_counts()
c
```

Output:

```
1    8364
0    6632
Name: returnsOpenNextMktres10, dtype: int64
```

So, we can see that ones are more than zero, so balancing zero and ones.

- **balancing zero and once...**

```
import random
random.seed(10)

a = market_train_df[market_train_df['returnsOpenNextMktres10']==1].sample(n
= c[0])

b = market_train_df[market_train_df['returnsOpenNextMktres10']==0]

market_train_df = b.append(a,ignore_index=True)
```

- **Now Checking zero and Once**

```
c = market_train_df['returnsOpenNextMktres10'].value_counts()
c
```

Output:

```
1    6632
0    6632
Name: returnsOpenNextMktres10, dtype: int64
```

- **Checking Missing value present or not in data set**

  ### verify is that missing value present or not
  market_train_df['returnsOpenNextMktres10'].isna().value_counts()
- **Imputing Missing value**

  #### imputing missing value
  market_train_df['returnsOpenNextMktres10']=np.where(market_train_df.returns
  OpenNextMktres10.isna==True,0,market_train_df.returnsOpenNextMktres10)

  ### cross verify is that missing value present or not
  market_train_df['returnsOpenNextMktres10'].isna().value_counts()

  Output:

```
False    13264
Name: returnsOpenNextMktres10, dtype: int64
```

- **Hot-Encoding of Asset Code**
  Asset code is categorical variables so here I have done hot-encoding of asset code

  # one hot-Encoding with categorical variables
  columns = ['assetCode']

  one_hot1 = pd.get_dummies(market_train_df[['assetCode']])

  market_train_df = market_train_df.drop(columns,axis = 1)

  market_train_df=market_train_df.join(one_hot1)

5.6 Dealing with News headlines text Data (Using Text Mining Approach)

5.6.1 Pre-processing of News headlines Text Data

making copy of news_train_orig data

news_df = news_train_df.copy()

5.6.2 Lower Case

The first pre-processing step which we will do is transform our news headline data into lower case. This avoids having multiple copies of the same words. For example, while calculating the word count, ‘Analytics’ and ‘analytics’ will be taken as different words.

converting all text data into Lower case

```
news_df['headline'] = news_df['headline'].apply(lambda x: " ".join(x.lower() for x in x.split()))

news_df['headline'].head()
```

5.6.3 Removing Punctuation

The next step is to remove punctuation, as it doesn’t add any extra information while treating text data. Therefore, removing all instances of it will help us reduce the size of the training data.

```
## Removing Punctuation

news_df['headline'] = news_df['headline'].str.replace('[^\w\s]','')

news_df['headline'].head()
```

5.6.4 Removal of Stop Words

As we discussed earlier, stop words (or commonly occurring words) should be removed from the text data. For this purpose, we can either create a list of stopwords ourselves or we can use predefined libraries.

```
### Removing Stopwords

from nltk.corpus import stopwords

stop = stopwords.words('english')
```

```
news_df['headline'] = news_df['headline'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))

news_df['headline'].head()
```

5.6.5 Removal of numerical digit

The next step is to remove numerical digit.

```
## Removing numerical Value from text

news_df['headline'] = news_df['headline'].str.replace('\d+', ' ')

news_df['headline'].head()
```

5.6.6 Spelling correction

We've all seen news headline with a plethora of spelling mistakes. Our timelines are often filled with hastly sent news headline that are barely legible at times.

In that regard, spelling correction is a useful pre-processing step because this also will help us in reducing multiple copies of words. For example, "Analytics" and "analytcs" will be treated as different words even if they are used in the same sense.

To achieve this, we will use the *textblob* library. If you are not familiar with it, you can check my previous article on 'NLP for beginners using *textblob'*.

```
from textblob import TextBlob

news_df['headline'] = news_df['headline'].apply(lambda x: str(TextBlob(x).correct()))
```

5.6.7 Stemming

Stemming refers to the removal of suffices, like "ing", "ly", "s", etc. by a simple rule-based approach. For this purpose, we will use *PorterStemmer* from the NLTK library.

```
### Streaming text data with help of portar Stemmer library from nltk

from nltk.stem import PorterStemmer

st = PorterStemmer()

news_df['headline'] = news_df['headline'].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

Advance Text Processing

Up to this point, we have done all the basic pre-processing steps in order to clean our data. Now, we can finally move on to extracting features using NLP techniques.

5.6.8 N-grams

N-grams are the combination of multiple words used together. Ngrams with N=1 are called unigrams. Similarly, bigrams (N=2), trigrams (N=3) and so on can also be used.

Unigrams do not usually contain as much information as compared to bigrams and trigrams. The basic principle behind n-grams is that they capture the language structure, like what letter or word is likely to follow the given one. The longer the n-gram (the higher the *n*), the more context you have to work with. Optimum length really depends on the application – if your n-grams are too short, you may fail to capture important differences. On the other hand, if they are too long, you may fail to capture the "general knowledge" and only stick to particular cases.

So, let's quickly extract bigrams from our news headline using the *ngrams* function of the textblob library.

5.6.9 Term frequency

Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence.

Therefore, we can generalize term frequency as:

TF = (Number of times term T appears in the particular row) / (number of terms in that row)

5.6.10 Inverse Document Frequency

The intuition behind inverse document frequency (IDF) is that a word is not of much use to us if it's appearing in all the documents.

Therefore, the IDF of each word is the log of the ratio of the total number of rows to the number of rows in which that word is present.

> **IDF = log(N/n)**, where, N is the total number of rows and n is the number of rows in which the word was present.

5.6.11 Term Frequency – Inverse Document Frequency (TF-IDF)

TF-IDF is the multiplication of the TF and IDF

making n grams & calculate tf-idf

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=1000, lowercase=True, analyzer='word',

stop_words= 'english',ngram_range=(2,2))

train_vect = tfidf.fit_transform(news_df['headline'])
```

5.6.12 Singular-Value Decomposition

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler.

Some author defined SVD, bellow the following details are-

The Singular Value Decomposition is a highlight of linear algebra. — Page 371, Introduction to Linear Algebra, Fifth Edition, 2016

The singular value decomposition (SVD) provides another way to factorize a matrix, into singular vectors and singular values. The SVD allows us to discover some of the same kind of information as the eigendecomposition. However, the SVD is more generally applicable.

— Pages 44-45, Deep Learning, 2016.

```
from sklearn.decomposition import PCA, FastICA, TruncatedSVD
```

define svd function and pass argument

```
svd = TruncatedSVD(n_components=240, n_iter=5)
```

fitting tfidf values in svd function

data1_svd = svd.fit_transform(train_vect)

making data frame because svd function return values in array

data1_svd = pd.DataFrame(data1_svd)

5.7 Join text data with main Market data

market_train_df = pd.concat([market_train_df, data1_svd], axis=1)

- **Checking Missing Value**

  ##checking missing values in data

  market_train_df['returnsOpenNextMktres10'].isna().value_counts()

  Output:

```
True     16736
False    13264
Name: returnsOpenNextMktres10, dtype: int64
```

- **Imputing Missing Value**

  ## imputing missing values
  market_train_df['returnsOpenNextMktres10'] =
  np.where(market_train_df.returnsOpenNextMktres10.isna()==True,0,market_t
  rain_df.returnsOpenNextMktres10)

- **Creating Dependent and Independent variables**
  ### defing X and Y

  columns = ['returnsOpenNextMktres10','date']

  X = market_train_df.drop(columns, axis=1)

  y = market_train_df['returnsOpenNextMktres10']

- **Splitting data set into Train and Test**

```
#### Splitting data set into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

5.8 BUILDING MODEL

1. XGBoost (Extreme Gradient Boosting)
2. LGB (Light Gradient Boosting)

- **Importing Library for modeling**

```
import gc
import matplotlib
from sklearn.metrics import confusion_matrix
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
#from sklearn import cross_validation
from sklearn.model_selection import cross_validate
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold, RandomizedSearchCV
from sklearn.metrics import accuracy_score
#import pandas as pd
#import numpy as np
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix
from sklearn.feature_selection import VarianceThreshold
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout
from keras import regularizers
import matplotlib.pyplot as plt
import xgboost as xgb
import lightgbm as lgb
from sklearn.model_selection import train_test_split,GridSearchCV, RandomizedSearchCV
from sklearn.metrics import get_scorer
from sklearn.metrics import f1_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.ensemble import VotingClassifier
import lightgbm as lgb
from sklearn.externals.joblib import Parallel, delayed
from sklearn.base import clone
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve
import pickle
```

- **XGBoost (Extreme Gradient Boosting)**

define xgboost classifier

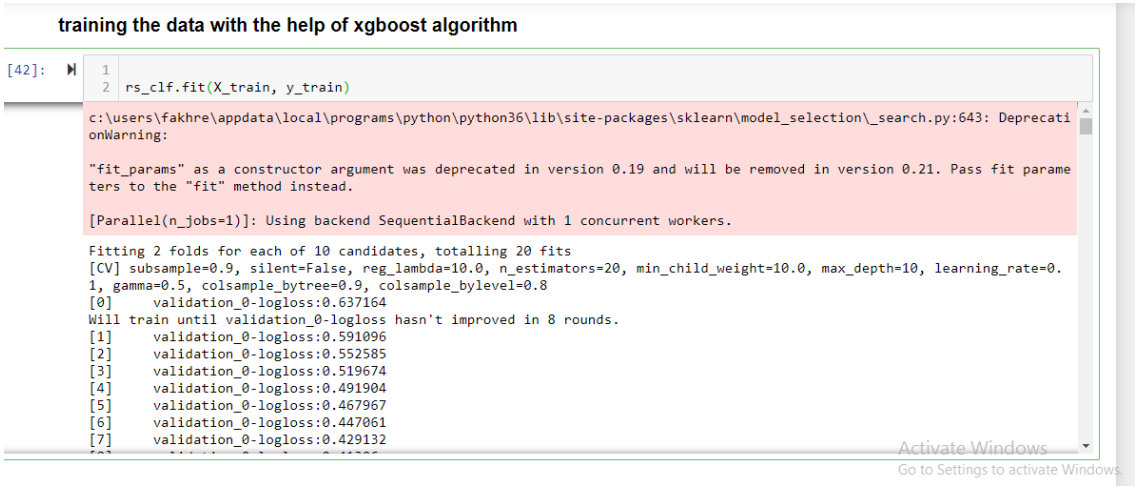
```
clf = xgb.XGBClassifier()
```

Parameter tuning code for prevent Overfitting

```
param_grid = {
'silent': [False],
'max_depth': [5, 10],##15,20
'learning_rate': [0.1, 0.2, 0.3],
'subsample': [0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
'min_child_weight': [0.5, 1.0, 3.0, 5.0, 7.0, 10.0],
'gamma': [0, 0.25, 0.5, 1.0],
'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0],
'n_estimators': [20]}

fit_params = {'eval_metric': 'logloss',
    'early_stopping_rounds': 8, ##10
     'eval_set': [(X_test, y_test)]}

rs_clf = RandomizedSearchCV(clf, param_grid, n_iter=10,
                    n_jobs=1, verbose=2, cv=2,
                    fit_params=fit_params,
                    scoring='neg_log_loss', random_state=50) #, refit=False
```

- **Training the data with the help of xgboost algorithm**

```
rs_clf.fit(X_train, y_train)
```



- **Defining function for calculating threshold**

```
from sklearn.metrics import accuracy_score

from sklearn.metrics import roc_curve

def Find_Optimal_Cutoff(target, predicted):

    fpr, tpr, threshold = roc_curve(target, predicted)

    i = np.arange(len(tpr))

    roc = pd.DataFrame({'tf' : pd.Series(tpr-(1-fpr), index=i), 'threshold' : pd.Series(threshold, index=i)})

    roc_t = roc.ix[(roc.tf-0).abs().argsort()[:1]]

    return list(roc_t['threshold'])
```

- **Predicting Probability on Train Data**

```
nn_train_pred = rs_clf.predict_proba(X_train)[:,1]

nn_train_pred
```

- **Output**

Predicting Probability on Train Data

```
In [44]:
1  nn_train_pred = rs_clf.predict_proba(X_train)[:,1]
2  nn_train_pred

Out[44]: array([0.47086924, 0.43957728, 0.53854614, ..., 0.00905202, 0.39770377,
                0.00905202], dtype=float32)
```

- **Find optimal probability threshold**

threshold = Find_Optimal_Cutoff(y_train, nn_train_pred)

print(threshold)

Output:

```
[0.43181565403938293]
```

- **Confusion Matrix**

In the case of binary classification, the confusion matrix is a 2-by-2 matrix laying out correct and incorrect predictions made in each label as follows:

```
+-----------------------------------+
| Predicted label |
+--------------------+--------------------+
| (+1) | (-1) |
+-------+-----+--------------------+--------------------+
| True |(+1) | # of true positives | # of false negatives |
| label +-----+--------------------+--------------------+
| |(-1) | # of false positives | # of true negatives |
+-------+-----+--------------------+--------------------+
```

from sklearn.metrics import confusion_matrix

nn_train_pred1 = np.where(nn_train_pred >=threshold,1,0)

cm1=confusion_matrix(y_train, nn_train_pred1)

cm1

Building Confusion Matrix

In [46]:
```
from sklearn.metrics import confusion_matrix
nn_train_pred1 = np.where(nn_train_pred >=threshold,1,0)
cm1=confusion_matrix(y_train, nn_train_pred1)
cm1
```

Out[46]:
```
array([[15987,  2688],
       [  765,  4560]], dtype=int64)
```

Accuracy

One performance metric I will use for our more advanced exploration is accuracy. Recall that the accuracy is given by

Accuracy=correctly classified data points/ TotalData point

Calculating Sensitivity, Specificity and Accuracy

specificity = cm1[0,0]/(cm1[0,0]+cm1[0,1])

sensitivity= cm1[1,1]/(cm1[1,0]+cm1[1,1])

print(sensitivity)

print(specificity)

accuracy_score(y_train, nn_train_pred1)

Calculating Sensitivity, Specificity andn Accuracy

In [47]:
```
specificity = cm1[0,0]/(cm1[0,0]+cm1[0,1])
sensitivity= cm1[1,1]/(cm1[1,0]+cm1[1,1])
print(sensitivity)
print(specificity)
accuracy_score(y_train, nn_train_pred1)
```

```
0.856338028169014
0.8560642570281124
```

Out[47]:
```
0.856125
```

Result on Train Data

Accuracy: 86%

Sensitivity: 86%

Specificity: 86%

- **Predict Probability on Test Data**

nn_test_pred = rs_clf.predict_proba(X_test)[:,1]

nn_test_pred

```
array([0.00905202, 0.00905202, 0.23425336, ..., 0.29610324, 0.00905202,
       0.00905202], dtype=float32)
```

- **Confusion Matrix**

nn_test_pred1 = np.where(nn_test_pred >=threshold,1,0)

cm2=confusion_matrix(y_test, nn_test_pred1)

cm2

Confusion Matrix

```
In [49]:
1 nn_test_pred1 = np.where(nn_test_pred >=threshold,1,0)
2 cm2=confusion_matrix(y_test, nn_test_pred1)
3
4 cm2

Out[49]: array([[3961,  732],
                [ 212, 1095]], dtype=int64)
```

calculating Sensitivity Specificity and Accuracy

```
In [50]:
1 specificity = cm2[0,0]/(cm2[0,0]+cm2[0,1])
2 sensitivity= cm2[1,1]/(cm2[1,0]+cm2[1,1])
3 print(sensitivity)
4 print(specificity)
5 accuracy_score(y_test, nn_test_pred1)

0.837796480489671
0.8440230129980822

Out[50]: 0.8426666666666667
```

Result on Test Data

Accuracy: 84%

Sensitivity: 84%

Specificity: 84%

2. Light Gradient Boosting Model

Importing Important Library for model Building

```
import lightgbm as lgb

from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV

from sklearn.metrics import get_scorer

from sklearn.metrics import f1_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score

from sklearn.ensemble import VotingClassifier

import lightgbm as lgb

from sklearn.externals.joblib import Parallel, delayed

from sklearn.base import clone

import pickle
```

Defining Learning Rate

 # Set up decay learning rate

```
def learning_rate_power(current_round):

    base_learning_rate = 0.19000424246380565

    min_learning_rate = 0.01
```

```
    lr = base_learning_rate * np.power(0.995,current_round)
    return max(lr, min_learning_rate)
```

Parameter Tuning code to prevent Overfitting

```
from scipy.stats import randint as sp_randint
from scipy.stats import uniform as sp_uniform

# tune_params = {'n_estimators': [200,500,1000,2500,5000],
#              'max_depth': sp_randint(4,12),
#               'learning_rate' : 0.005,
#               'bagging_fraction' : 0.7,
#               'feature_fraction' : 0.5,
#               'bagging_frequency' : 6,
#               'bagging_seed' : 42,
#              'colsample_bytree':sp_uniform(loc=0.8, scale=0.15),
#              'min_child_samples':sp_randint(60,120),
#              'subsample': sp_uniform(loc=0.75, scale=0.25),
#              'reg_lambda':[1e-3, 1e-2, 1e-1, 1]}
tune_params = {
    'task': 'train',
    'boosting_type': 'gbdt',
    'objective': 'binary',
    'metric': {'binary_logloss', 'auc'},
    'metric_freq': 1,
    'is_training_metric': True,
    'max_bin': 255,
    'learning_rate': 0.1,
```

```
    'num_leaves': 63,
    'tree_learner': 'serial',
    'feature_fraction': 0.8,
    'bagging_fraction': 0.8,
    'bagging_freq': 5,
    'min_data_in_leaf': 50,
    'min_sum_hessian_in_leaf': 5,
    'is_enable_sparse': True,
    'use_two_round_loading': False,
    'is_save_binary_file': False,
    'output_model': 'LightGBM_model.txt',
    'num_machines': 1,
    'local_listen_port': 12400,
    'machine_list_file': 'mlist.txt',
    'verbose': 0,
    # parameters to keep the exactly the same
    'subsample_for_bin': 200000,
    'min_child_samples': 20,
    'min_child_weight': 0.001,
    'min_split_gain': 0.0,
    'colsample_bytree': 1.0,
    'reg_alpha': 0.0,
    'reg_lambda': 0.0
}

fit_params = {'early_stopping_rounds':40,
              'eval_metric': 'accuracy',
```

```
            'eval_set': [(X_train, y_train), (X_test, y_test)],
            'verbose': 20,
            'callbacks': [lgb.reset_parameter(learning_rate=learning_rate_power)]}


lgb_clf = lgb.LGBMClassifier(n_jobs=4, objective='binary',random_state=1)
gs = RandomizedSearchCV(estimator=lgb_clf,
                        param_distributions=tune_params,
                        n_iter=10,
                        scoring='f1',
                        cv=5,
                        refit=True,
                        random_state=1,
                        verbose=True)


lgb_clf = lgb.LGBMClassifier(n_jobs=4,
                         objective='multiclass',
                        random_state=100)
opt_params = {'n_estimators':500,
            'boosting_type': 'dart',
            'objective': 'binary',
            'num_leaves':2452,
            'min_child_samples':212,
            'reg_lambda':0.01}
lgb_clf.set_params(**opt_params)
```

Train the Model

lgb_clf.fit(X_train, y_train,**fit_params)

```
In [55]:
lgb_clf = lgb.LGBMClassifier(n_jobs=4,
                             objective='multiclass',
                            random_state=100)
opt_params = {'n_estimators':500,
              'boosting_type': 'dart',
              'objective': 'binary',
              'num_leaves':2452,
              'min_child_samples':212,
              'reg_lambda':0.01}
lgb_clf.set_params(**opt_params)
lgb_clf.fit(X_train, y_train,**fit_params)
```

```
c:\users\fakhre\appdata\local\programs\python\python36\lib\site-packages\lightgbm\callback.py:189: UserWarning:

Early stopping is not available in dart mode
```

```
[20]    training's binary_logloss: 0.579053     valid_1's binary_logloss: 0.581023
[40]    training's binary_logloss: 0.589611     valid_1's binary_logloss: 0.591549
[60]    training's binary_logloss: 0.595046     valid_1's binary_logloss: 0.596921
[80]    training's binary_logloss: 0.593917     valid_1's binary_logloss: 0.595834
[100]   training's binary_logloss: 0.58821      valid_1's binary_logloss: 0.590276
[120]   training's binary_logloss: 0.580101     valid_1's binary_logloss: 0.582348
[140]   training's binary_logloss: 0.570942     valid_1's binary_logloss: 0.573409
[160]   training's binary_logloss: 0.561592     valid_1's binary_logloss: 0.564298
[180]   training's binary_logloss: 0.552528     valid_1's binary_logloss: 0.555491
[200]   training's binary_logloss: 0.544012     valid_1's binary_logloss: 0.547225
[220]   training's binary_logloss: 0.536165     valid_1's binary_logloss: 0.539624
[240]   training's binary_logloss: 0.529034     valid_1's binary_logloss: 0.53272
```


Predicting Probability on Train Data

nn_train_pred = lgb_clf.predict_proba(X_train)[:,1]

nn_train_pred

Output:

```
array([0.48797667, 0.49387833, 0.52861003, ..., 0.29747057, 0.44331762,
       0.29747056])
```

Find optimal probability threshold

threshold = Find_Optimal_Cutoff(y_train, nn_train_pred)

print(threshold)

```
Output: [0.46296081481892276]
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
nn_train_pred1 = np.where(nn_train_pred >=threshold,1,0)
cm1=confusion_matrix(y_train, nn_train_pred1)
cm1
```

Output:

```
array([[15838,  2837],
       [  809,  4516]], dtype=int64)
```

Calculating Accuracy, Sensitivity and Specificity

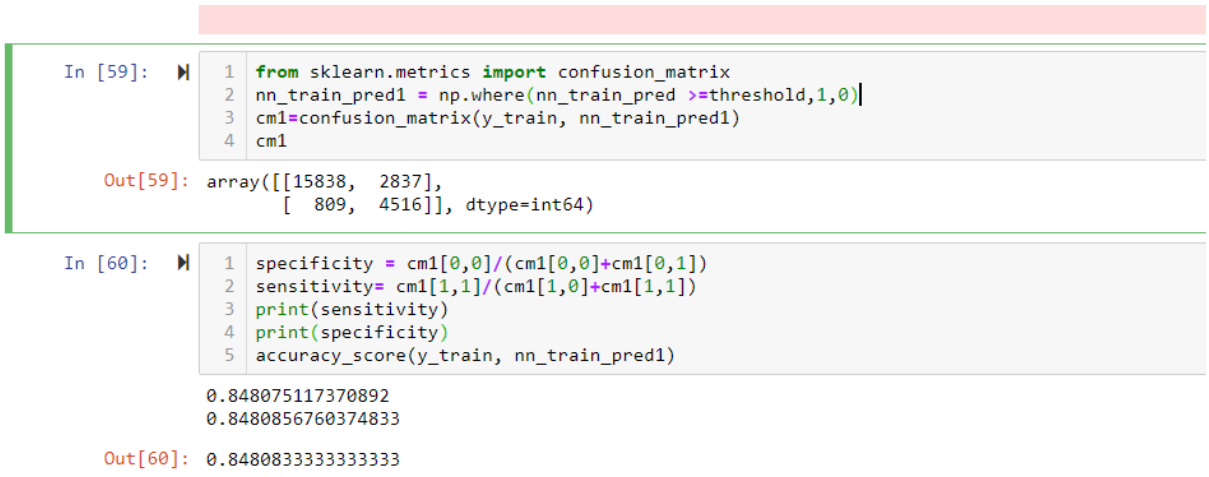
specificity = cm1[0,0]/(cm1[0,0]+cm1[0,1])

sensitivity= cm1[1,1]/(cm1[1,0]+cm1[1,1])

print(sensitivity)

print(specificity)

accuracy_score(y_train, nn_train_pred1)



Predict Probability on Test Data

nn_test_pred = lgb_clf.predict_proba(X_test)[:,1]

nn_test_pred

Output:

```
array([0.29802105, 0.29747057, 0.42032811, ..., 0.40774574, 0.29747056,
```

```
                0.29747056])
```

Confusion Matrix

```
nn_test_pred1 = np.where(nn_test_pred >=threshold,1,0)
cm2=confusion_matrix(y_test, nn_test_pred1)
cm2
```

```
array([[3936,  757],
       [ 225, 1082]], dtype=int64)
```

Calculating Accuracy Sensitivity and Specificity

```
specificity = cm2[0,0]/(cm2[0,0]+cm2[0,1])
sensitivity= cm2[1,1]/(cm2[1,0]+cm2[1,1])
print(sensitivity)
print(specificity)
accuracy_score(y_test, nn_test_pred1)
```

In [61]:
```
nn_test_pred =  lgb_clf.predict_proba(X_test)[:,1]
nn_test_pred
```

Out[61]:
```
array([0.29802105, 0.29747057, 0.42032811, ..., 0.40774574, 0.29747056,
       0.29747056])
```

In [62]:
```
nn_test_pred1 = np.where(nn_test_pred >=threshold,1,0)
cm2=confusion_matrix(y_test, nn_test_pred1)

cm2
```

Out[62]:
```
array([[3936,  757],
       [ 225, 1082]], dtype=int64)
```

In [63]:
```
specificity = cm2[0,0]/(cm2[0,0]+cm2[0,1])
sensitivity= cm2[1,1]/(cm2[1,0]+cm2[1,1])
print(sensitivity)
print(specificity)
accuracy_score(y_test, nn_test_pred1)
```

```
0.827850038255547
0.8386959301086725
```

Out[63]:
```
0.8363333333333334
```

CHAPTER 6

Tools Required

- ✓ Detailed specifications of the various components, measuring devices, software tool box, data sheet references etc

Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python provides variety of packages for data exploration and analysis including: Pandas, NumPy, SciPy, a helping hand from Python's Standard Library. For data visualization python provides a self-explanatory name. Taking data and turning it into something colorful diagram. Matplotlib, Seaborn, Datashader are very good library for data visualization.

Jupyter-Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Getting Up and Running with Jupyter Notebook

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language. This article will focus on just two of them for the purposes of installing Jupyter Notebook. The most popular is CPython, which is the reference version of Python that you can get from their website. It is also assumed that you are using **Python 3**.

Installation

If so, then you can use a handy tool that comes with Python called **pip** to install Jupyter Notebook like this:

Open window command prompt and type

- Pip install Jupyter

Starting the Jupyter Notebook Server

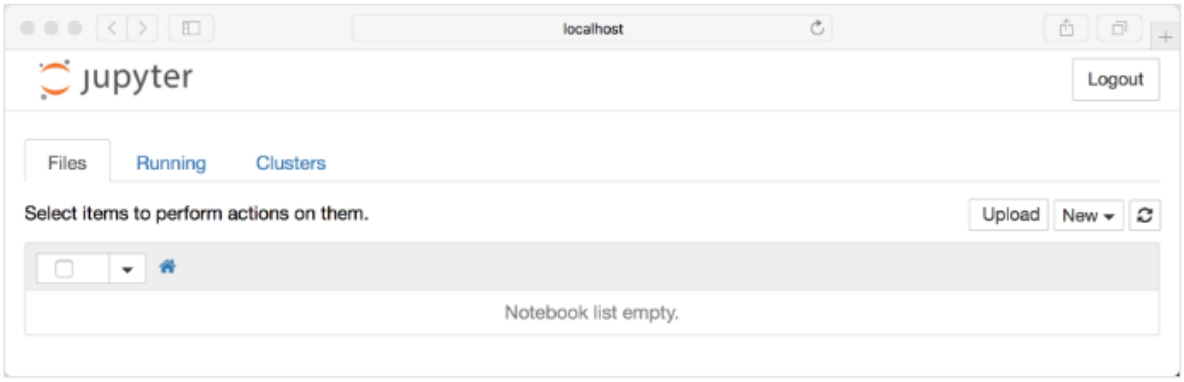
Now that you have Jupyter installed, let's learn how to use it. To get started, all you need to do is open up your terminal application and go to a folder of your choice. I recommend using something like your Documents folder to start out with and create a subfolder there called Notebooks or something else that is easy to remember.

Then just go to that location in your terminal and run the following command:

- Jupyter-notebook

This will start up Jupyter and your default browser should start to the following URL: http://localhost:8888/tree

Your browser should now look something like this:



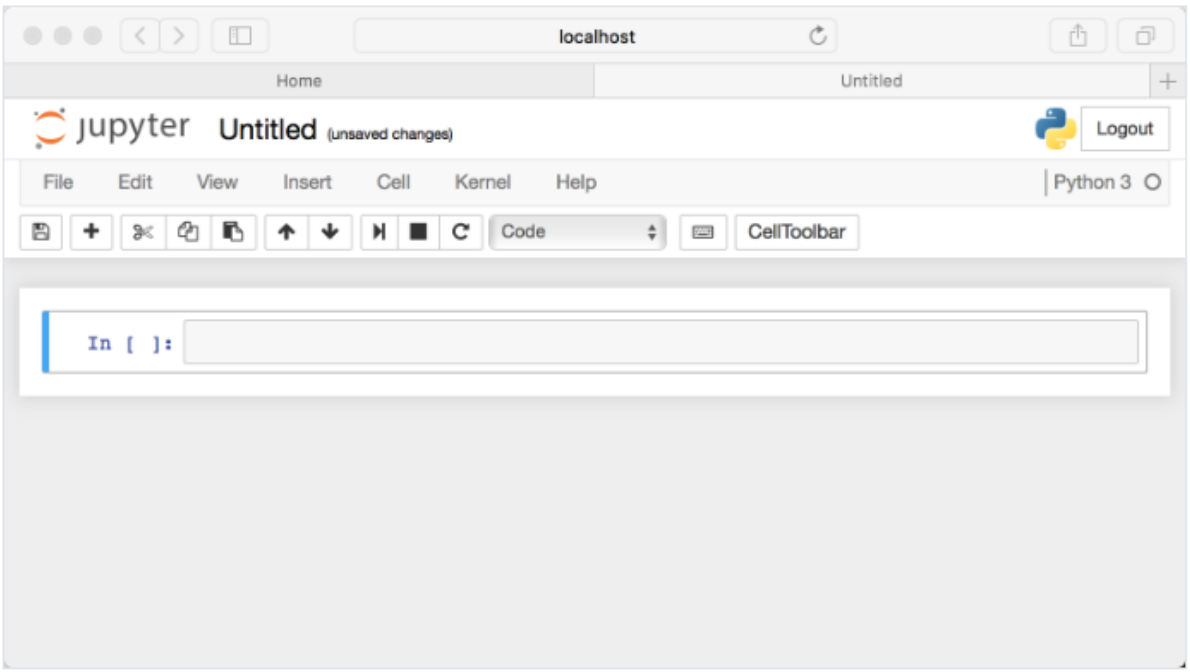
Note that right now you are not actually running a Notebook, but instead you are just running the Notebook server. Let's create a Notebook now!

Creating a Notebook

Now that you know how to start a Notebook server, you should probably learn how to create an actually Notebook document.

All you need to do is click on the New button (upper right), and it will open up a list of choices. On my machine, I installed Python 3, so I can create a Notebook that uses either of these. For simplicity's sake, let's choose Python 3.

Your web page should now look like this:



Running Cells

A Notebook's cell defaults to using code whenever you first create one, and that cell uses the kernel that you chose when you started your Notebook.

In this case, you started yours with Python 3 as your kernel, so that means you can write Python code in your code cells. Since your initial Notebook has only one empty cell in it, the Notebook can't really do anything.

Thus, to verify that everything is working as it should, you can add some Python code to the cell and try running its contents.

Running a cell means that you will execute the cell's contents. To execute a cell, you can just select the cell and click the *Run* button that is in the row of buttons along the top. It's towards the middle. If you prefer using your keyboard, you can just press shift + enter button.

Data Source

This dataset contains two different datasets and both datasets are taken from Kaggle. (https://www.kaggle.com)

CHAPTER – 7

Results Analysis

XGBoost was created by Tianqi Chen, PhD Student, University of Washington. It is used for supervised ML problems Ever since its introduction in 2014. From predicting ad click-through rates to classifying high energy physics events, XGBoost has proved its one of the best algorithms in terms of performance and speed. I always used to XGBoost as my first algorithm of building machine learning model because it saves time and gives better accuracy and provide better solution than other machine learning algorithm and it give optimized and robust result and it has distributed gradient boosting library and also it uses gradient boosting framework at core. Second algorithm I have used to LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages: Faster training speed and higher efficiency, lower memory uses, better accuracy, support of parallel and GPU learning, Capable of handling large-scale data

Later training our model, we go on to an application at the correlation within news and stock market prices on each day range. To do so, we have received stock data as well as news data for the same timeline as described above. Also, business stocks collected daily data for each day. After maintaining a strong correlation, we can predict the stock values.

Result

Algorithm	**Accuracy**	**Sensitivity**	**Specificity**
Extreme Gradient Boosting (XGBoost)	86%	86%	86%
Light Gradient Boosting (LightGBM)	85%	85%	85%

CHAPTER - 8

Deploy Python Flask Application with Apache on a Windows Server

5.1 SETUP: -

4 Steps

Install Python

5.1.2 Download python 3.6.6 (amd 64 bit) and install it.

5.1.3 After installation add the python path.

Step to set the python path

control panel -> System and Security -> System -> Advanced System Settings -> Environment Variables -> Selecting Path -> Edit

(i) C:\Python3.6.6

(ii) C:\Python3.6.6\Scripts.

5.1.4 Install Pip

- As of python3.6.6 version pip is already installed automatically and will be available in your script folder.
- Pip is a Package manager for python which we will use to load in modules/libraries into our environments.
- To test that Pip is installed open a command prompt (win+r->'cmd'->Enter) and try 'pip help'.

5.1.5 Install virtualenv

Now that you have pip installed and a command prompt open installing virtualenv to our root Python installation is as easy as typing

- pip install virtualenv

5.1.6 Install virtualenvwrapper-win

- pip install virtualenvwrapper-win

5.2 USAGE: -

Steps:

5.2.1 Make a Virtual Environment

Open command prompt and enter

- mkvirtualenv Hanover (we can give any name).

This will create a folder with python.exe, pip, and setuptools all ready to go in its own environment. It will also activate the Virtual Environment which is indicated with the (stock_project) on the left side of the prompt.

Anything we install now will be specific to this project. And available to the projects we connect to this environment.

5.2.2 Connect our project with our Environment

Now we want our code to use this environment to install packages and run/test code. First let's create a directory with the same name as our virtual environment in our preferred development folder. In this case mine is 'dev'. And then create directory stock_project inside the dev folder.

5.2.3 Set Project Directory

Now to bind our virtualenv with our current working directory we simply enter

- 'setprojectdir.'

Now next time we activate this environment we will automatically move into this directory.

5.2.4 Deactivate

If we to deactivate this environment, then type deactivate.

5.2.5 Workon

Open up the command prompt and type 'workon stock_project r' to activate the environment and move into your root project folder.

5.2.6 Pip Install

To use flask, we need to install the packages and to do that we can use pip to install it into our stock_project virtual environment.

- pip install flask

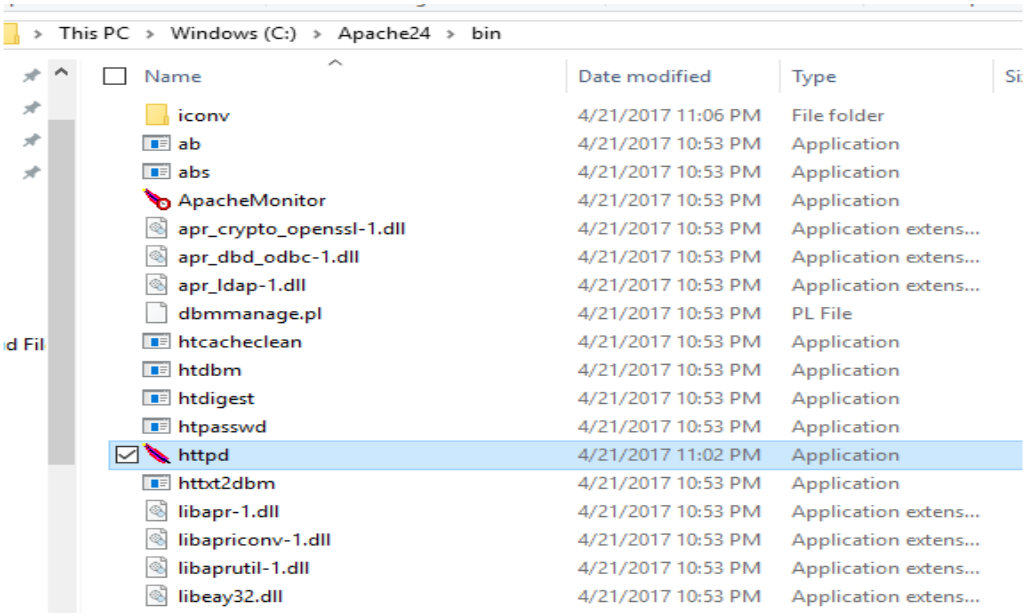
5.2.7 Flask

Now that you have flask installed in your virtual environment you can start coding and save it in your stock_project Directory, and then save file as stock_project.py.

Once the code is in place I can start the server using 'python stock_project.py' this will run the python instance from your virtual environment that has flask.

Now we have to configure apache to run python.

- Go to this link https://httpd.apache.org/docs/2.4/platform/windows.html and download wamp or Apache Standalone from here or https://www.apachelounge.com/download/.
- Extract the files and copy Apache24 to C:\Apache24
- Navigate to C:\Apache24\bin and run or click on httpd.exe to start apache



You can now navigate with your browser to http://127.0.0.1:5000/ (or you can use any ip in my case I gave http://10.53.6.80:9000/)and see your new site.

CHAPTER - 9

CONCLUSION

The stock market follows very random walk and is dynamic because of this instability the price of stock has very complex behaviour, in financial market profit opportunities are exploited as soon as they arise hence there are very important need to explore this enormous amount of data of stock market. Trader decision are based how a news articles is influencing the market. These financial news articles have information about organization vision, mission, coming project, ongoing project, activity in which it is involved and expectation from the other competitor and also looks for financial market information like the trading volume inflation, demand for product or services offered by the organization opening price and closing price, calculated return etc. So, analysing of news data to predict stock prices. This unique opportunity will advance the state of research in understanding the predictive power of the news. This power if harnessed could help financial outcomes and generate significant economic impact all over the world. Using XGBoost and LightGBM algorithm for developing the model and it gives the better result. 86% of accuracy achieved for XGBoost model and the LightGBM model gives 85% accuracy for prediction.

.

CHAPTER – 10

STAGE PALN

These are the milestone planning to achieve my Dissertation with relative duration of the months.

Milestone 1(Aug-Sept): Background History, Problem Formulation.

Milestone 2(Oct-Nov): Literature Review, Research Direction Requirement, Getting and Processing dataset, writing of the Survey Paper.

Milestone 3(Dec-Jan): Gaining insight from the data, Implementation of the System.

Milestone 4(Feb-March): Experiment the implemented system, Analyse and improve the results found in the experiment.

Milestone 5(Apr-May): Result and Conclusion, Writing of the Final Paper, Writing of the Thesis.

Table 3: Time Line Chart for Dissertation

No.	Task	Aug Sept		Oct Nov	Dec Jan	Feb March	Apr May
1	Background History						
2	Problem Formulation						
3	Literature Review						
4	Research Direction						
5	Getting and Processing Dataset						
6	Writing Survey Paper						
7	Gaining insight from the data						
8	Implementation of the System						
9	Experiment the implemented System						
10	Analyse and improve the results found in the experiment						
11	Stimulation and Verification						
12	Result and Conclusion						
13	Writing of the Final Paper and Thesis						

Milestones Achieved

REERENCES

[1] Dhaka, V., Kausar, M. & Singh, S., 2013. Web Crawler: A Review. International Journal of Computer Applications, Vol 63, No. 2, pp. 31-36.

[2] Weiss, S. M., Indurkhya, N. & Zhang, T., 2010. Fundamentals of Predictive Text Mining. s.l.: Springer Publishing Company, Incorporated

[3] Future K. Tan, steinbach, Introduction to data mining. 2006.

[4]. S. B. Imandoust and M. Bolandraftar, ―Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background, ‖ vol. 3, no. 5, pp. 605–610, 2013.

[5] Sidorova, G. et al., 2014. Syntactic N-grams as machine learning features for natural language processing. Expert Systems with Applications, Vol. 41, Issue 3, pp. 853-860.

[6]. F.R. Bach, G.R.G. Lanckriet, M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st International Conference on Machine Learning (2004), pp. 6-13.

[7] Miner, G. et al., 2014. Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications. s.l.: Elsevier.

[8]. S. Deng, T. Mitsubuchi, K. Shioda, T. Shimada, and A. Sakurai, "Combining Technical Analysis with Sentiment Analysis for Stock Price Prediction," in Proc. of the IEEE 9th Intl Conference on Dependable, Autonomic and Secure Computing, 2011, pp. 800–807.

[9]. X. Li, C. Wang, J. Dong, and F. Wang, "Improving stock market prediction by integrating both market news and stock prices," Database and Expert Systems Applications, Lecture Notes in Computer Science, vol. 6861, pp. 279–293, 2011.

[10] F. Wang, L. Liu, and C. Dou, "Stock Market Volatility Prediction: A Service Oriented Multi-kernel Learning Approach," in Proc. of IEEE 9th Intl Conference on Services Computing, 2012, vol. d, pp. 49–56.

[11] C.-Y. Yeh, C.-W. Huang, and S.-J. Lee, "A multiple-kernel support vector regression approach for stock market price forecasting," Expert Systems with Applications, vol. 38, no. 3, pp. 2177–2186, 2011.

[12] T. Fletcher, Z. Hussain, J. Shawe-Taylor, Multiple kernel learning on the limit order book. JMLR: Workshop and Conference Proceedings 11 (2010) 167-174. Workshop on Applications of Pattern Analysis.

[13] R. Luss, A. d'Aspremont, Predicting Abnormal Returns From News using Text Classification, arXiv:0809.2792v3, 2009

[14] Yeh, C.-Y., Huang, C.-W., Lee, S.-J., A multiple-kernel support vector regression approach for stock market price forecasting. Expert Systems with Applications (2010), doi: 10.1016/j.eswa.2010.08.004.

[15] Zhai, C. & Massung, S., 2016. Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining. s.l.: ACM.

[16] Harris, Z., 1954. Distributional Structure. Word, 10, p. 146–162.

[17] Robertson, S., 2004. Understanding inverse document frequency: On theoretical arguments for IDF. Journal of Documentation. 60 (5), p. 503–520.

[18] La Jolla, CA 92037, June 15-2001, using news articles to predict stock price movements

[19] Ayman E. Khedr, S.E. Salama, Nagwa Yaseen, 2017. Predicting Stock Market behaviour using Data Mining Technique and News Sentiment Analysis: MECS

[20] Banking Inclusion-A Gateway to Financial Inclusion Sibi. M. S, Dr. A. A. Ananth SUMEDHA Journal of Management, Jan-March 2017.

[21] Vivek Rajput, Sarika Bobde "Stock market forecasting techniques: Literature Survey", IJCSMC, Vol. 5, Issue. 6, June 2016, pg.500 – 506.

[22] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Hongjun Lu, "The Predicting Power of Textual Information on Financial Markets", IEEE Intelligent Informatics Bulletin, June 2005 Vol.5 No.1.

[23] Ms. Ashwini Pravin Navghane, Dr. Pradeep Mitharam Patil, "Forecasting Stock Market Movement: A Neural Network Approach", IJECET2013.

[24] X. Wang, P. K. H. Phua, and W. Lin, "Stock market prediction using neural networks: does trading volume help in short-term prediction" in Neural Networks, 2003. Proceedings of the International Joint Conference on, vol. 4. IEEE, 2003, pp. 2438–2442.

[25] A. U. Khan et al., "Stock rate prediction using back propagation algorithm: Analyzing the prediction accuracy with different number of hidden layers," Glow gift, Bhopal, 2005.

[26] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," Omega, vol. 33, no. 6, pp. 497 505, 2005.

[27] Vivek Rajput, Sarika Bobde, "STOCK MARKET FORECASTING TECHNIQUES", IJCSMC, Vol. 5, Issue. 6, June 2016, pg.500 – 506.