Stock Prediction Model

Aryan Gupta
Dept. of Computer Science and Engineerging
Chandigarh University
Mohali, India
19BCS3828@cuchd.in

Daksh Rawal
Dept. of Computer Science and Engineerging
Chandigarh University
Mohali, India
19BCS3849@cuchd.in

Prajwal
Dept. of Computer Science and Engineerging
Chandigarh University
Mohali, India
19BCS3831@cuchd.in

ABSTRACT

I. **In the era of big data, deep learning for predicting stock market prices and trends has become even more popular than before. We collected 2 years of data from Chinese stock market and proposed a comprehensive customization of feature engineering and deep learning-based model for predicting price trend of stock markets. The proposed solution is comprehensive as it includes pre-processing of the stock market dataset, utilization of multiple feature engineering techniques, combined with a customized deep learning-based system for stock market price trend prediction. We conducted comprehensive evaluations on frequently used machine learning models and conclude that our proposed solution outperforms due to the comprehensive feature engineering that we built. The system achieves overall high accuracy for stock market trend prediction. With the detailed design and evaluation of prediction term lengths, feature engineering, and data pre-processing methods, this work contributes to the stock analysis research community both in the financial and technical domains.**

II. INTRODUCTION

Earlier studies on stock market prediction are based on the historical stock prices. Later studies have debunked the approach of predicting stock market movements using historical prices. Stock market prices are largely fluctuating. The efficient market hypothesis (EMH) states that financial market movements depend on news, current events and product releases and all these factors will have a significant impact on a company's stock value [2]. Because of the lying unpredictability in news and current events, stock market prices follow a random walk pattern and cannot be predicted with more than 50% accuracy [1]. With the advent of social media, the information about public feelings has become abundant. Social media is transforming like a perfect platform to share public emotions about any topic and has a significant impact on overall public opinion. Twitter, a social media platform, has received a lot of attention from researchers in the recent times. Twitter is a micro-blogging application that allows users to follow and comment other users thoughts or share their opinions in real time [3]. More than million users post over 140 million tweets every day. This situation makes Twitter like a corpus with valuable data for researchers [4].Each tweet is of 140 characters long and speaks public opinion on a topic concisely. The information exploited from tweets are very useful for making predictions [5]. In this paper, we contribute to the field of sentiment analysis of twitter data. Sentiment classification is the task of judging opinion in a piece of text as positive, negative or neutral. There are many studies involving twitter as a major source for public-opinion analysis. Assur and Huberman [6] have predicted box office collections for a movie prior to its release based on public sentiment related to movies, as expressed on Twitter. Google flu trends are being widely studied along with twitter for early prediction of disease outbreaks. Eiji et al. [11] have studied the twitter data for catching the flu outbreaks. Ruiz et al. [7] have used time-constrained graphs to study the problem of correlating the Twitter micro-blogging activity with changes in stock prices and trading volumes. Borodino et al. [8] have shown that trading volumes of stocks traded in NASDAQ-100 are correlated with their query volumes (i.e., the number of users requests submitted to search engines on the Internet). Gilbert and Karahalios [9] have found out that increases in expressions of anxiety, worry and fear in weblogs predict downward pressure on the S&P 500 index. Bollen [10] showed that public mood analyzed through twitter feeds is well correlated with Dow Jones Industrial Average (DJIA). 1 arXiv:1610.09225v1 [cs.IR] 28 Oct 2016 All these studies showcased twitter as a valuable source and a powerful tool for conducting studies and making predictions. Rest of the paper is organized as follows. Section 2 describes the related works and Section 3 discusses the data portion demonstrating the data collection and pre-processing part. In Section 4 we discuss the sentiment analysis part in our work followed by Section 5 which examines the correlation part of extracted sentiment with stocks. In Section 6 we present the results, accuracy and precision of our sentiment analyzer followed by the accuracy of correlation analyzer.

III. OBJECTIVES

A. Data Collection

A total of 2,50,000 tweets over a period of August 31st, 2015 to August 25th,2016 on Microsoft are extracted from twitter API [16]. Twitter4J is a java application which helps us to extract tweets from twitter. The tweets were collected using Twitter API and filtered using keywords like $ MSFT, # Microsoft, #Windows etc. Not only the opinion of public about the company's stock but also the opinions about products and services offered by the company would have a significant impact and are worth studying. Based on this principle, the keywords used for filtering are devised with extensive care and tweets are extracted in such a way that they represent the exact emotions of public about Microsoft over a period of time. The news on twitter about Microsoft and tweets regarding the product releases were also included. Stock opening and closing prices of Microsoft from August 31st, 2015 to August 25th, 2016 are obtained from Yahoo! Finance [23].

B. Data Pre-Processing

Stock prices data collected is not complete understandably because of weekends and public holidays when the stock market does not function. The missing data is approximated using a simple technique by Goel [17]. Stock data usually follows a concave function. So, if the stock value on a day is x and the next value present is y with some missing in between. The first missing value is approximated to be (y+x)/2 and the same method is followed to fill all the gaps. Tweets consists of many acronyms, emoticons and unnecessary data like pictures and URL's. So tweets are preprocessed to represent correct emotions of public. For preprocessing of tweets we employed three stages of filtering: Tokenization, Stopwords removal and regex matching for removing special characters. 1) Tokenization: Tweets are split into individual words based on the space and irrelevant symbols like emoticons are removed. We form a list of individual words for each tweet. 2) Stop word Removal: Words that do not express any emotion are called Stopwords. After splitting a tweet, words like a,is, the, with etc. are removed from the list of words. 3) Regex Matching for special character Removal: Regex matching in Python is performed to match URLs and are replaced by the term URL. Often tweets consists of hashtags(#) and @ addressing other users. They are also replaced suitably. For example, #Microsoft is replaced with Microsoft and @Billgates is replaced with USER. Prolonged word showing intense emotions like coooooooool! is replaced with cool! After these stages the tweets are ready for sentiment classification.

C. Bench Mark Model

In machine learning, benchmarking is the practice of comparing tools to identify the best-performing technologies in the industry. However, comparing different machine learning platforms can be a difficult task due to the large number of factors involved in the performance of a tool.

This post aims to identify the most critical key performance indicators (KPIs) and define a consistent measurement process. Performance benchmarking. As we know, the volume, variety, and velocity of information stored in organizations are increasing significantly. Therefore, for machine learning tools to be efficient, they need to process large amounts of data in the shortest time possible. Key performance indicators typically measured here are data capacity, training speed, inference speed, and model precision. Benchmarking is used to measure performance using a specific indicator resulting in a metric that is then compared to others.This allows organizations to develop plans on making improvements or adapting specific best practices, usually to increase some aspect of performance. In this way, they learn how well the targets perform and, more importantly, the business processes that explain why these firms are successful. However, machine learning platforms may crash due to memory problems when building models with big datasets. Therefore, tools capable of processing these volumes of data are necessary.The data capacity of a machine learning platform can be defined as the biggest dataset that it can process. In this way, the tool should perform all the essential tasks with that dataset.

We can measure data capacity as the number of samples that a machine learning platform can process for a given number of variables.This metric depends on numerous factors:

The programming language in which it is written (C++, Java, Python...). The strategies used within the code for the efficient use of memory. The optimization algoritms it contains (SGD, Adam, LM...). To compare the data capacity of machine learning platforms, we follow the next steps:

Choose a reference computer (CPU, GPU, RAM...).

Choose a reference benchmark (data set, neural network, training strategy) Choose a reference model (number of layers, number of neurons...). Choose a reference training strategy (loss index, optimization algorithm...).Choose a stopping criterion (loss goal, epochs number, maximum time...). Note that the selection of a dataset suite is necessary. The following figure illustrates the result of a data capacity test with two platforms. One of the most critical factors in machine learning platforms is the time they need to train the models. Indeed, modeling big data sets is very expensive in computational terms. Training machine learning models with big datasets can take several hours. Moreover, before deploying a model, it is usually necessary to train many candidate models to select the best-performing one. This can make it impractical to use some platforms for some applications. The training speed of a machine learning platform depends on numerous factors: The programming language in which it is written (C++, Java, Python...).

The high performance computing (HPC) techniques that it implements (CPU parallelization, GPU acceleration...).

The optimization algorithms it contains (SGD, Adam, LM...).

Training speed is usually measured as the number of samples per second that the platform processes during training. To compare the training speed of machine learning platforms, we follow the next steps: Choose a reference benchmark (data set, neural network, training strategy...).

Choose a reference computer (CPU, GPU, RAM...).

Compare the training speed. The following figure illustrates the result of a training speed test with two platforms.

This post aims to define the most important KPIs in machine learning platforms. It also describes the most relevant factors that might affect those key performance indicators. Finally, it describes how to design and measure performance tests for data capacity, training speed, model precision, and inference speed. The machine learning platform Neural Designer implements high performance techniques so that you can get maximum productivity.

D. Long-Short Term Memory Model

Long Short-Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory. Let's say while watching a video you remember the previous scene or while reading a book you know what happened in the earlier chapter. Similarly, RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they cannot remember long term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems. Note: If you are more interested in learning concepts in an Audio-Visual format, we have this entire article explained in the video below. If not, you may continue reading. The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp. These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.

E. Improved Long-Short Term Memory Model

An improved model is proposed to solve the above problems. The input variables are divided into the delay variables and no-delay variables. The delay variables participate in LSTM model (see Figure 3) calculation and generate the memory block output , while the no-delay variables merge with to form new input. Then, a feedforward neural network is established for the new input. One more hidden layer 2 is added to enhance the nonlinear expression ability of the network. So the no-delay variables can directly use aging factor time at the last time step without complex transformation and the form of subsequence. The improved LSTM model also can save training time because, for no-delay variables, there is no need for recurrent calculation inside the memory block. However, it increases the time consumption due to the additional hidden layer 2. Therefore, the neurons' number and the transfer function of hidden layer 2 should be appropriately set to control the time consumption. Figure 4 shows the improved LSTM model calculating process.

IV. DISCUSSION

should be appropriately set to control the time consumption. Figure 4 shows the improved LSTM model calculating process. This section gives an overview of accuracy rates of the trained classifiers. All the calculations are done in Weka tool which runs on java virtual machine [20]

A. Sentiment Analyzer Results

The above sections discussed the method followed to train the classifier used for sentiment analysis of tweets. The classifier with features as Word2vec representations of human annotated tweets trained on Random Forest algorithm with a split percentage of 90 for training the model and remaining for testing the model showed an accuracy of 70.2%. With N-gram representations, the classifier model with same algorithm and with same dataset showed an accuracy of 70.5%. Though the results are very close, model trained with word2vec representations is picked to classify the nonhuman annotated tweets because of its promising accuracy for large datasets and the sustainability in word meaning. Numerous studies have been conducted on people and they concluded that the rate of human concordance, that is the degree of agreement among humans on the sentiment of a text, is between 70% and 79% [21]. They have also synthesized that sentiment analyzers above 70% are very accurate in most of the cases. Provided this information, the results we obtained from the sentiment classification can be observed as very good figures while predicting the sentiments in short texts, tweets, less than 140 characters in length. Table-2 depicts the results of sentiment classification including accuracy, precision, F-measure and recall when trained with different machine learning algorithms. ROC curves are plotted for detailed analysis.

B. Stock Price and Sentiment Correlation Results

A classifier is presented in the previous sections that is trained with aggregate sentiment values for 3-day period as features and the increase/decrease in stock price represented by 1/0 as the output. Total data is split into two parts, 80 percent to train the model and remaining for testing operations. The classifier results show an accuracy value of 69.01% when trained using Logistic regression algorithm and the accuracy rate varied with the training set. When the model with LibSVM is trained with 90 percent of data, it gave a result of 71.82%. These results give a significant edge to the investors and they show good correlation between stock market movements and the sentiments of public expressed in twitter. This trend shows that with increasing dataset the models are performing well.

We would like to incorporate more data in our future work.

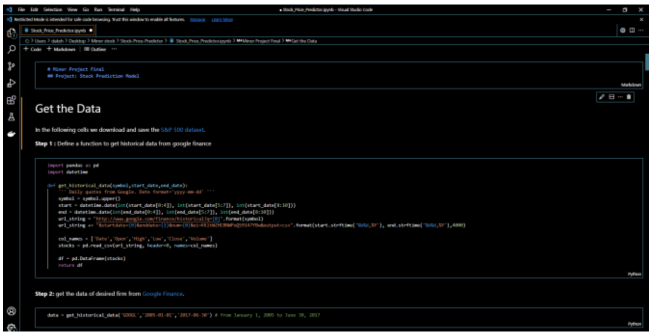
V. ABOUT THE DATASET

This section details the data that was extracted from the public data sources, and the final dataset that was prepared. Stock market-related data are diverse, so we first compared the related works from the survey of financial research works in stock market data analysis to specify the data collection directions. After collecting the data, we defined a data structure of the dataset. Given below, we describe the dataset in detail, including the data structure, and data tables in each category of data with the segment definitions. Description of our dataset In this section, we will describe the dataset in detail. Tis dataset consists of 3558 stocks from the Chinese stock market. Besides the daily price data, daily fundamental data of each stock ID, we also collected the suspending and resuming history, top 10 shareholders, etc. We list two reasons that we choose 2 years as the time span of this dataset: (1) most of the investors perform stock market price trend analysis using the data within the latest 2 years, (2) using more recent data would benefit the analysis result. We collected data through the open-sourced API, namely Tushare [43], mean-while we also leveraged a web-scraping technique to collect data from Sina Finance web pages, SWS Research website.

VI. ABOUT THE MODEL

A. Getting the Data

One of the questions I hear often is, “Where can I get data?” I wish I heard it a lot more often. The question means different things to different people. Some are on a quest for information that will drive business decisions. Others want practice to develop technical skills. Still others are interested in furthering social causes or understanding science. While some need the kind of detailed data that fuels statistical analysis, many are better off if they can find a source that has already done some of the data analysis for them, providing reports, data in aggregate form or even just specific facts.



```
# Minor Project Final
## Project: Stock Prediction Model
```

Get the Data

In the following cells we download and save the S&P 500 dataset.

Step 1 : Define a function to get historical data from google finance

```
import pandas as pd
import datetime

def get_historical_data(symbol,start_date,end_date):
    ''' Daily quotes from Google. Date format='yyyy-mm-dd' '''
    symbol = symbol.upper()
    start = datetime.date(int(start_date[0:4]), int(start_date[5:7]), int(start_date[8:10]))
    end = datetime.date(int(end_date[0:4]), int(end_date[5:7]), int(end_date[8:10]))
    url_string = "http://www.google.com/finance/historical?q={0}".format(symbol)
    url_string += "&startdate={0}&enddate={1}&num={0}&ei=KKltWZHCBNWPuQS9147YBw&output=csv".format(start.strftime('%b%d,%Y'), end.strftime('%b%d,%Y'),4000)

    col_names = ['Date','Open','High','Low','Close','Volume']
    stocks = pd.read_csv(url_string, header=0, names=col_names)

    df = pd.DataFrame(stocks)
    return df
```

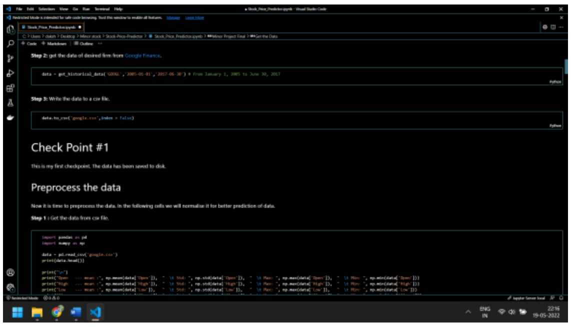
Step 2: get the data of desired firm from Google Finance.

```
data = get_historical_data('GOOGL','2005-01-01','2017-06-30') # from January 1, 2005 to June 30, 2017
```

Nearly all can obtain useful data to help meet their goals. Loads of data is available today, both privately within businesses, and through public sources. A little effort can yield a wealth of information. What worries me is knowing that many people who ought to be looking for data aren't. They're making decisions based on just personal opinions, or something in the news, or using some data, but neglecting data types or sources that would add value for them. What a waste. The key to getting the data you need is to have well-defined goals and a clear sense of purpose. The better than you can define what information you need and what you're going to do with it, the more easily you will be able to locate appropriate resources.

B. Data Preprocessing

Companies can use data from nearly endless sources – internal information, customer service interactions, and all over the internet – to help inform their choices and improve their business. But you can't simply take raw data and run it through machine learning and analytics programs right away. You first need to preprocess your data, so it can be successfully "read" or understood by machines.



Step 2: get the data of desired firm from Google Finance.

```
data = get_historical_data('GOOGL','2005-01-01','2017-06-30') # from January 1, 2005 to June 30, 2017
```

Step 3: Write the data to a csv file.

```
data.to_csv('google.csv',index = False)
```

Check Point #1

This is my first checkpoint. The data has been saved to disk.

Preprocess the data

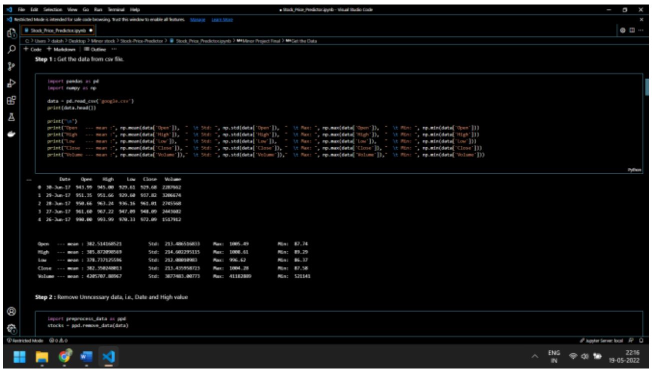
Now it is time to preprocess the data. In the following cells we will normalise it for better prediction of data.

Step 1 : Get the data from csv file.

```
import pandas as pd
import numpy as np

data = pd.read_csv('google.csv')
print(data.head())

print("\n")
print("Open   --- mean :", np.mean(data['Open']),  "  \t Std: ", np.std(data['Open']),  "  \t Max: ", np.max(data['Open']),  "  \t Min: ", np.min(data['Open']))
print("High   --- mean :", np.mean(data['High']),  "  \t Std: ", np.std(data['High']),  "  \t Max: ", np.max(data['High']),  "  \t Min: ", np.min(data['High']))
print("Low    --- mean :", np.mean(data['Low']),   "  \t Std: ", np.std(data['Low']),   "  \t Max: ", np.max(data['Low']),   "  \t Min: ", np.min(data['Low']))
```

Data Preprocessing includes the steps we need to follow to transform or encode data so that it may be easily parsed by the machine.

The main agenda for a model to be accurate and precise in predictions is that the algorithm should be able to easily interpret the data's features.



Step 1 : Get the data from csv file.

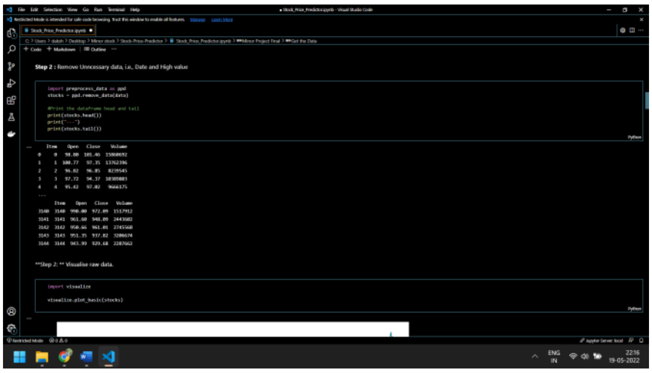
```
import pandas as pd
import numpy as np

data = pd.read_csv('google.csv')
print(data.head())

print("\n")
print("Open   --- mean :", np.mean(data['Open']),  "  \t Std: ", np.std(data['Open']),  "  \t Max: ", np.max(data['Open']),  "  \t Min: ", np.min(data['Open']))
print("High   --- mean :", np.mean(data['High']),  "  \t Std: ", np.std(data['High']),  "  \t Max: ", np.max(data['High']),  "  \t Min: ", np.min(data['High']))
print("Low    --- mean :", np.mean(data['Low']),   "  \t Std: ", np.std(data['Low']),   "  \t Max: ", np.max(data['Low']),   "  \t Min: ", np.min(data['Low']))
print("Close  --- mean :", np.mean(data['Close']), "  \t Std: ", np.std(data['Close']), "  \t Max: ", np.max(data['Close']), "  \t Min: ", np.min(data['Close']))
print("Volume --- mean :", np.mean(data['Volume']),"  \t Std: ", np.std(data['Volume']),"  \t Max: ", np.max(data['Volume']),"  \t Min: ", np.min(data['Volume']))
```

```
        Date    Open    High     Low   Close   Volume
0  30-Jun-17  943.99  945.00  929.61  929.68  2287662
1  29-Jun-17  951.35  951.66  929.60  937.82  3206674
2  28-Jun-17  950.66  963.24  936.16  961.01  2745568
3  27-Jun-17  961.60  967.22  947.09  948.09  2443602
4  26-Jun-17  990.00  993.99  970.33  972.09  1517912


Open   --- mean : 382.514168521       Std:  213.486516833     Max:  1005.49         Min:  87.74
High   --- mean : 385.872098569       Std:  214.602295115     Max:  1008.61         Min:  89.29
Low    --- mean : 378.737125596       Std:  212.08080983      Max:  996.62          Min:  86.37
Close  --- mean : 382.350248013       Std:  213.435958723     Max:  1004.28         Min:  87.58
Volume --- mean : 4205707.80967       Std:  3877483.00773     Max:  41182889        Min:  521141
```

Step 2 : Remove Unncessary data, i.e., Date and High value

```
import preprocess_data as ppd
stocks = ppd.remove_data(data)
```

Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning. Raw, real-world data in the form of text, images, video, etc., is messy.



Step 2 : Remove Unncessary data, i.e., Date and High value

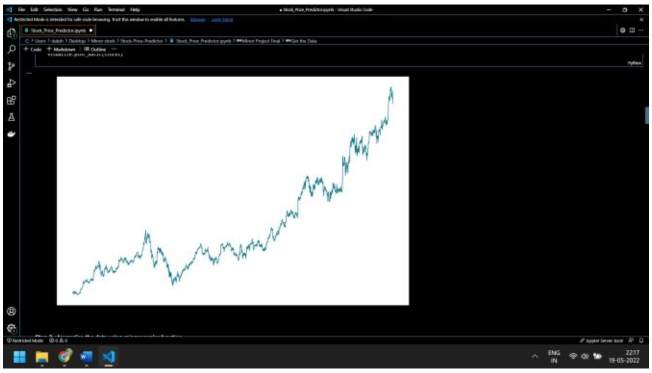
```
import preprocess_data as ppd
stocks = ppd.remove_data(data)

#Print the dataframe head and tail
print(stocks.head())
print("---")
print(stocks.tail())
```

```
   Item    Open   Close    Volume
0     0   98.80  101.46  15860692
1     1  100.77   97.35  13762396
2     2   96.82   96.85   8239545
3     3   97.72   94.37  10389803
4     4   95.42   97.02   9666175
---
      Item    Open   Close   Volume
3140  3140  990.00  972.09  1517912
3141  3141  961.60  948.09  2443602
3142  3142  950.66  961.01  2745568
3143  3143  951.35  937.82  3206674
3144  3144  943.99  929.68  2287662
```

**Step 2: ** Visualise raw data.

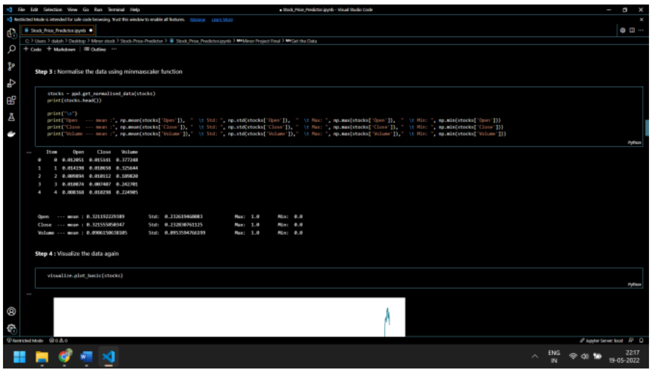
```
import visualize

visualize.plot_basic(stocks)
```

Not only may it contain errors and inconsistencies, but it is often incomplete, and doesn't have a regular, uniform design. Machines like to process nice and tidy information – they read data as 1s and 0s. So calculating structured data, like whole numbers and percentages is easy. However, unstructured data, in the form of text and images must first be cleaned and formatted before analysis.



```
visualize.plot_basic(stocks)
```

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.



Step 3 : Normalise the data using minmaxscaler function

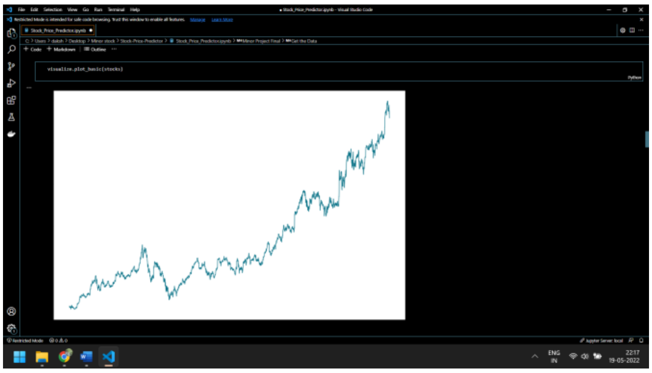
```
stocks = ppd.get_normalised_data(stocks)
print(stocks.head())

print("\n")
print("Open   --- mean :", np.mean(stocks['Open']),  "  \t Std: ", np.std(stocks['Open']),  "  \t Max: ", np.max(stocks['Open']),  "  \t Min: ", np.min(stocks['Open']))
print("Close  --- mean :", np.mean(stocks['Close']), "  \t Std: ", np.std(stocks['Close']), "  \t Max: ", np.max(stocks['Close']), "  \t Min: ", np.min(stocks['Close']))
print("Volume --- mean :", np.mean(stocks['Volume']),"  \t Std: ", np.std(stocks['Volume']),"  \t Max: ", np.max(stocks['Volume']),"  \t Min: ", np.min(stocks['Volume']))
```

```
   Item      Open     Close    Volume
0     0  0.012051  0.015141  0.377248
1     1  0.014198  0.010658  0.325644
2     2  0.009094  0.010112  0.189820
3     3  0.010074  0.007407  0.242701
4     4  0.008368  0.010298  0.224905


Open   --- mean : 0.321192229389      Std:  0.232619460083       Max:  1.0      Min:  0.0
Close  --- mean : 0.321555050347      Std:  0.232830761125       Max:  1.0      Min:  0.0
Volume --- mean : 0.0906150630105     Std:  0.0953594766199      Max:  1.0      Min:  0.0
```

Step 4 : Visualize the data again

```
visualize.plot_basic(stocks)
```

It’s important to understand what “features” are when preprocessing your data because you’ll need to choose which ones to focus on depending on what your business goals are. Later, we’ll explain how you can improve the quality of your dataset’s features and the insights you gain with processes like feature selection.

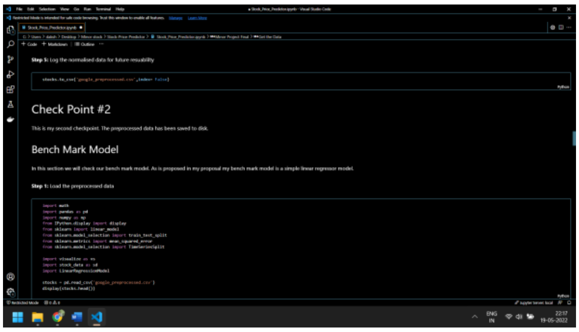


```
visualize.plot_basic(stocks)
```

The human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than poring over spreadsheets or reports. Data visualization is a quick, easy way to convey concepts in a universal manner – and you can experiment with different scenarios by making slight adjustments.

C. *Bench Mark Model*

The term benchmarking is used in machine learning (ML) to refer to the evaluation and comparison of ML methods regarding their ability to learn patterns in ‘benchmark’ datasets that have been applied as ‘standards’. Benchmarking could be thought of simply as a sanity check to confirm that a new method successfully runs as expected and can reliably find simple patterns those existing methods are known to identify it.



Step 5: Log the normalised data for future reusability

```
stocks.to_csv('google_preprocessed.csv',index= False)
```

Check Point #2

This is my second checkpoint. The preprocessed data has been saved to disk.

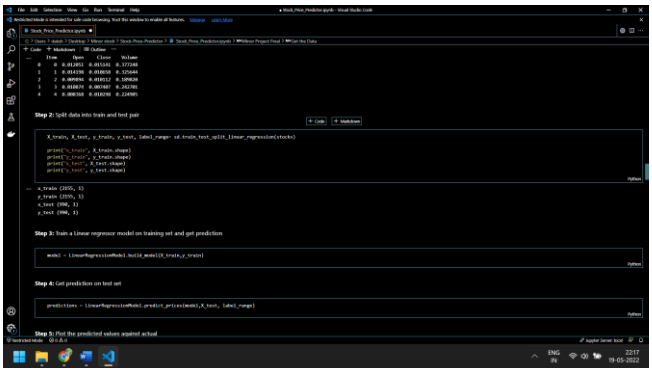
Bench Mark Model

In this section we will check our bench mark model. As is proposed in my proposal my bench mark model is a simple linear regressor model.

Step 1: Load the preprocessed data

```
import math
import pandas as pd
import numpy as np
from IPython.display import display
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import TimeSeriesSplit

import visualize as vs
import stock_data as sd
import LinearRegressionModel

stocks = pd.read_csv('google_preprocessed.csv')
display(stocks.head())
```

A more rigorous way to view benchmarking is as an approach to identify the respective strengths and weaknesses of a given methodology in contrast with others [2]. Comparisons could be made over a range of evaluation metrics, e.g., power to detect signal, prediction accuracy, computational complexity, and model interpretability. This approach to benchmarking would be important for demonstrating new methodological abilities or simply to guide the selection of an appropriate ML method for a given problem.



```
   Item      Open     Close    Volume
0     0  0.012051  0.015141  0.377248
1     1  0.014198  0.010658  0.325644
2     2  0.009094  0.010112  0.189820
3     3  0.010074  0.007407  0.242701
4     4  0.008368  0.010298  0.224905
```

Step 2: Split data into train and test pair

```
X_train, X_test, y_train, y_test, label_range= sd.train_test_split_linear_regression(stocks)

print("x_train", X_train.shape)
print("y_train", y_train.shape)
print("x_test", X_test.shape)
print("y_test", y_test.shape)
```

```
x_train (2155, 1)
y_train (2155, 1)
x_test (990, 1)
y_test (990, 1)
```

Step 3: Train a Linear regressor model on training set and get prediction

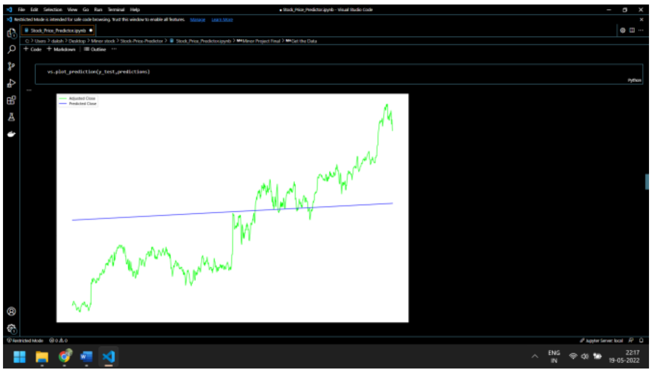
```
model1 = LinearRegressionModel1.build_model(X_train,y_train)
```

Step 4: Get prediction on test set

```
predictions = LinearRegressionModel1.predict_prices(model1,X_test, label_range)
```

Step 5: Plot the predicted values against actual

Benchmark datasets typically take one of three forms. The first is accessible, well-studied real-world data, taken from different real-world problem domains of interest. The second is simulated data, or data that has been artificially generated, often to ‘look’ like real-world data, but with known, underlying patterns. For example, the GAMETES genetic-data simulation software generates epistatic patterns of association in ‘mock’ single nucleotide polymorphism (SNP) data [3, 4].

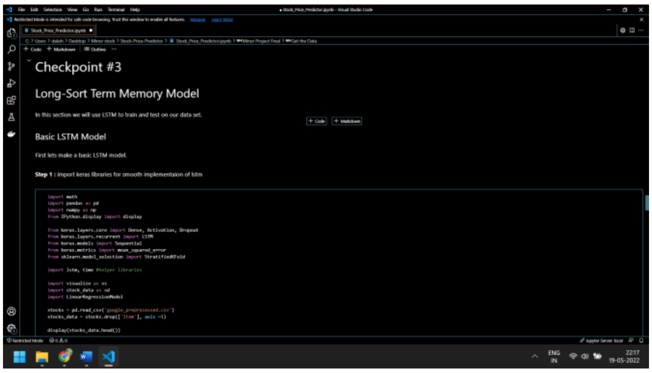


```
vs.plot_prediction(y_test,predictions)
```

The third form is toy data, which we will define here as data that is also artificially generated with a known embedded pattern but without an emphasis on representing real-world data, e.g., the parity or multiplexer problems [5, 6]. It is worth noting that the term ‘toy dataset’ has often been used to describe a small and simple dataset such as the examples included with algorithm software.

D. *Long-Short Term Memory Model*

Our proposed solution is a unique customization as compared to the previous works because rather than just proposing yet another state-of-the-art LSTM model, we proposed a fine-tuned and customized deep learning prediction system along with utilization of comprehensive feature engineering and combined it with LSTM to perform prediction. By researching into the observations from previous works, we fill in the gaps between investors and researchers by proposing a feature extension algorithm



Checkpoint #3

Long-Sort Term Memory Model

In this section we will use LSTM to train and test on our data set.

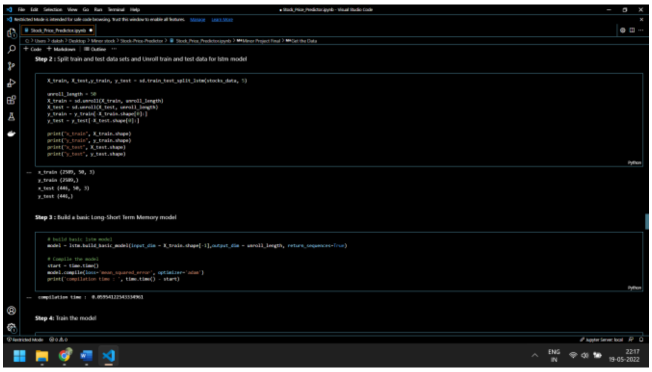
Basic LSTM Model

First lets make a basic LSTM model.

Step 1 : import keras libraries for smooth implementaion of lstm

```
import math
import pandas as pd
import numpy as np
from IPython.display import display

from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Sequential
from keras.metrics import mean_squared_error
from sklearn.model_selection import StratifiedKFold

import lstm, time #helper libraries

import visualize as vs
import stock_data as sd
import LinearRegressionModel

stocks = pd.read_csv('google_preprocessed.csv')
stocks_data = stocks.drop(['Item'], axis =1)

display(stocks_data.head())
```

before recursive feature elimination and get a noticeable improvement in the model performance. Tough we have achieved a decent outcome from our proposed solution, this research has more potential towards research in future. During the evaluation procedure, we also found that the RFE algorithm is not sensitive to the term lengths other than 2-day, weekly, biweekly



Step 2 : Split train and test data sets and Unroll train and test data for lstm model

```
X_train, X_test,y_train, y_test = sd.train_test_split_lstm(stocks_data, 5)

unroll_length = 50
X_train = sd.unroll(X_train, unroll_length)
X_test = sd.unroll(X_test, unroll_length)
y_train = y_train[-X_train.shape[0]:]
y_test = y_test[-X_test.shape[0]:]

print("x_train", X_train.shape)
print("y_train", y_train.shape)
print("x_test", X_test.shape)
print("y_test", y_test.shape)
```

```
x_train (2589, 50, 3)
y_train (2589,)
x_test (446, 50, 3)
y_test (446,)
```

Step 3 : Build a basic Long-Short Term Memory model

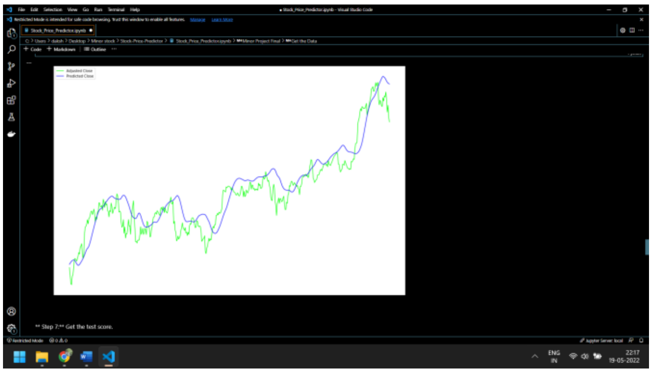
```
# build basic lstm model
model = lstm.build_basic_model(input_dim = X_train.shape[-1],output_dim = unroll_length, return_sequences=True)

# Compile the model
start = time.time()
model.compile(loss='mean_squared_error', optimizer='adam')
print('compilation time : ', time.time() - start)
```

```
compilation time :  0.05954122543334961
```

Step 4: Train the model

.

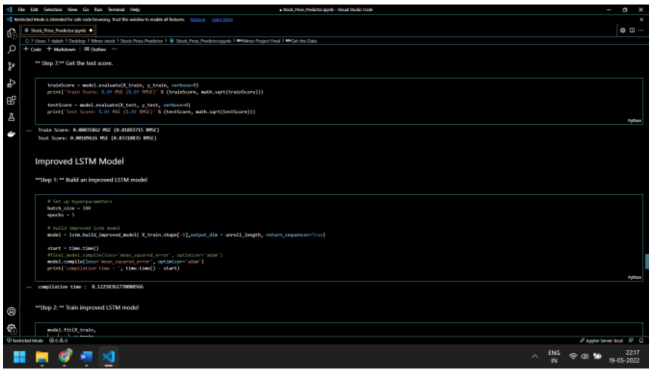
Getting more in-depth research into what technical indices would influence the irregular term lengths would be a possible future research direction. Moreover, by combining latest sentiment analysis techniques with feature engineering and deep learning model, there is also a high potential to develop a more comprehensive prediction system which is trained by diverse types of information such as tweets, news, and other text-based data.



** Step 7:** Get the test score.

E. Improved LSTM Model

The performance of the LSTM model and its improved model for the radial displacement prediction of the Dongjiang arch dam is demonstrated. The MLR, MLP, SVM, and BRT models are used as contrast models.



** Step 7:** Get the test score.

```
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))
```

```
Train Score: 0.00035862 MSE (0.01893715 RMSE)
Test Score: 0.00109616 MSE (0.03310835 RMSE)
```

Improved LSTM Model

**Step 1: ** Build an improved LSTM model

```
# Set up hyperparameters
batch_size = 100
epochs = 5

# build improved lstm model
model = lstm.build_improved_model( X_train.shape[-1],output_dim = unroll_length, return_sequences=True)

start = time.time()
#final_model.compile(loss='mean_squared_error', optimizer='adam')
model.compile(loss='mean_squared_error', optimizer='adam')
print('compilation time : ', time.time() - start)
```

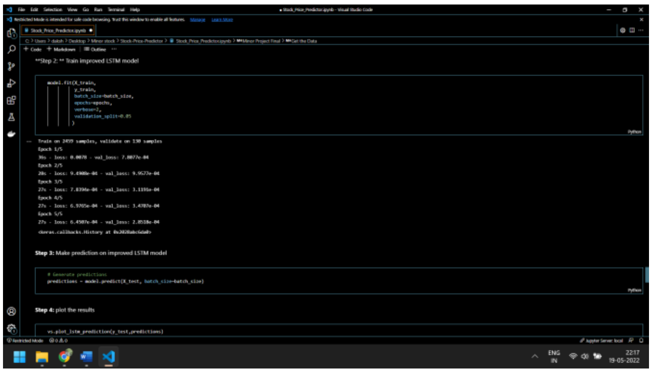
```
compilation time :  0.12258362770080566
```

**Step 2: ** Train improved LSTM model

```
model.fit(X_train,
```

.

The modeling process using MLR, MLP, SVM, and BRT is carried out using an application program interface (API) scikit-learn (v0.19.0). LSTM algorithm is implemented using TensorFlow (v1.8.1) developed by Google.



**Step 2: ** Train improved LSTM model

```
model.fit(X_train,
          y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=2,
          validation_split=0.05
         )
```

```
Train on 2459 samples, validate on 130 samples
Epoch 1/5
36s - loss: 0.0078 - val_loss: 7.8077e-04
Epoch 2/5
28s - loss: 9.4988e-04 - val_loss: 9.9577e-04
Epoch 3/5
27s - loss: 7.8394e-04 - val_loss: 3.1191e-04
Epoch 4/5
27s - loss: 6.9765e-04 - val_loss: 3.4707e-04
Epoch 5/5
27s - loss: 6.4587e-04 - val_loss: 2.8518e-04
<keras.callbacks.History at 0x2028abc6da0>
```

Step 3: Make prediction on improved LSTM model

```
# Generate predictions
predictions = model.predict(X_test, batch_size=batch_size)
```

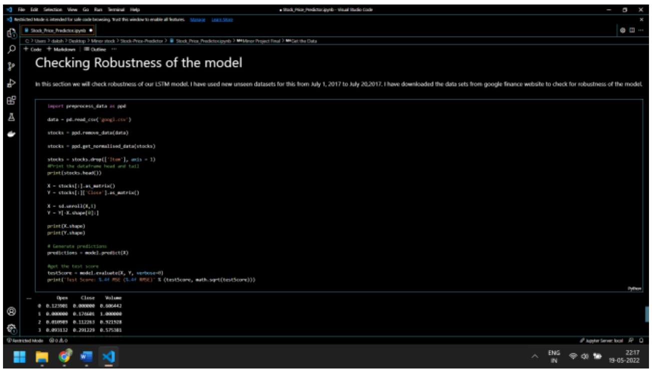
Step 4: plot the results

```
vs.plot_lstm_prediction(y_test,predictions)
```

All the algorithms are implemented in Python 3.6 environment All these algorithms are used in their original version and are performed via a laptop computer with detailed technical parameters shown as one CPU of Intel core i5-4200U@1.60 Hz and 2 core processors, one RAM of 8GB, and 64-bit system type. Before running these models, data were normalized into the range.

F. Checking robustness

Robustness is the evaluation of an analytical method wherein the results obtained are found to be reliable even when performed in a slightly varied condition. It is the ability of a method to remain unaffected when slight variations are applied. A more general example of this parameter would be to imagine a footballer practising spot kicks using certain technique under given conditions. Now the same footballer is asked to take the kicks using a different sized ball, smaller target, higher atmospheric temperature or different boots. If the results are similar (statistically), the technique could be termed as robust.



Checking Robustness of the model

In this section we will check robustness of our LSTM model. I have used new unseen datasets for this from July 1, 2017 to July 20,2017. I have downloaded the data sets from google finance website to check for robustness of the model.

```
import preprocess_data as ppd

data = pd.read_csv('googl.csv')

stocks = ppd.remove_data(data)

stocks = ppd.get_normalised_data(stocks)

stocks = stocks.drop(['Item'], axis = 1)
#Print the dataframe head and tail
print(stocks.head())

X = stocks[:].as_matrix()
Y = stocks[:]['Close'].as_matrix()

X = sd.unroll(X,1)
Y = Y[-X.shape[0]:]

print(X.shape)
print(Y.shape)

# Generate predictions
predictions = model.predict(X)

#get the test score
testScore = model.evaluate(X, Y, verbose=0)
print('Test Score: %.4f MSE (%.4f RMSE)' % (testScore, math.sqrt(testScore)))
```

```
       Open     Close    Volume
0  0.123901  0.000000  0.606442
1  0.000000  0.174601  1.000000
2  0.018989  0.112263  0.921928
3  0.093132  0.291229  0.575381
```

File Edit Selection View Go Run Terminal Help
• Stock_Price_Predictor.ipynb - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Stock_Price_Predictor.ipynb
+ Code + Markdown Outline
Python
Restricted Mode
Jupyter Server: local
ENG IN
22:17 19-05-2022

Long short-term memory (LSTM) is an effective solution to time sequence prediction. Considering the data perturbations, in this letter, a variant model of LSTM is proposed to achieve robustness of prediction. Specifically, data processing procedure in the recurrent unit of proposed model is reformulated, the gates are controlled by only one variable, and the variable is the sum of long-term memory and the current input. Due to the simplified two-gate structure of proposed model, the speed of prediction is improved as well. The experiments on three datasets verify that the proposed model with simplified structure has higher robustness and shorter running time than the traditional LSTM model.

VII. Acknowledgment

The authors would like to thank Dr. Rahul Rastogi for supporting this research with her guidance and Chandigarh University, Gharuan for giving us the opportunity and platform for completing this research.

References

[1] [1] S. A. R. Nai-Fu Chen and Richard Roll, Economic Forces and the Stock Market, The Journal of Business, vol. 59, no. 3, pp. 383–403, (1986).

[2] [Online]. Available: http://www.jstor.org/stable/2352710.

[3] [2] E. F. Fama, Random Walks in Stock Market Prices, Financial Analysts Journal, vol. 51, no. 1, pp. 75–80, (1995).

[4] [Online]. Available: http://www.jstor.org/stable/4479810.

[5] [3] S. J. Grossman and R. J. Shiller, The Determinants of the Variability of Stock Market Prices, National Bureau of Economic Research, Working

[6] Paper 564, October (1980) [Online]. Available: http://www.nber.org/papers/w0564.

[7] [4] A. W. Lo and A. C. MacKinlay, Stock Market Prices do not Follow Random Walks: Evidence from a Simple Specification Test, Review of

[8] Financial Studies, vol. 1, no. 1, pp. 41–66, (1988).

[9] [5] P. P¨a¨akk ¨onen and D. Pakkala, Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems,

[10] Big Data Research, vol. 2, no. 4, pp. 166–186, (2015).

[11] [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214579615000027.

[12] [6] P. A. G. Xue Zhang and Hauke Fuehres, Predicting Stock Market Indicators through Twitter I Hope it is not as Bad as I Fear,

[13] Procedia – Social and behavioral Sciences, vol. 26, pp. 55–62, (2011).

[14] [7] J. Bollen, H. Mao and X. Zeng, Twitter Mood Predicts the Stock Market, Journal of Computational Science, vol. 2, no. 1, pp. 1–8, (2011).

[15] [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187775031100007X.

[16] [8] K. Mizumoto, H. Yanagimoto and M. Yoshioka, Sentiment Analysis of Stock Market News with Semi-Supervised Learning,

[17] In 2012 IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS), pp. 325–328, May (2012).

[18] [9] M. Z. F. Werner Antweiler, Is all that Talk Just Noise? the Information Content of Internet Stock Message Boards, The Journal of Finance,

[19] vol. 59, no. 3, pp. 1259–1294, (2004). [Online]. Available: http://www.jstor.org/stable/3694736.

[20] [10] R. Ahuja, H. Rastogi, A. Choudhuri and B. Garg, Stock Market Forecast Using Sentiment Analysis, In 2015 2nd International Conference

[21] on Computing for Sustainable Global Development (INDIACom), pp. 1008–1010, March (2015).

[22] [11] N. Lin, J. Yuan, W. Xu, L. Wei and X. Wang, How web News Media Impact Futures Market Price Linkage?, In 2013 Sixth International

[23] Conference on Business Intelligence and Financial Engineering (BIFE), pp. 562–566, November (2013).

[24] [12] M. Hagenau, M. Liebmann, M. Hedwig and D. Neumann, Automated News Reading: Stock Price Prediction Based on Financial News Using

[25] Context-Specific Features, In 2012 45th Hawaii International Conference on System Science (HICSS), pp. 1040–1049, January (2012).

[26] [13] J. Gong and S. Sun, A New Approach of Stock Price Prediction Based on Logistic Regression Model, In 2009. NISS '09. International

[27] Conference on New Trends in Information and Service Science, pp. 1366–1371, June (2009).

[28] [14] R. F. W. Robert Tumarkin, News or Noise? Internet Postings and Stock Prices, Financial Analysts Journal, vol. 57, no. 3, pp. 41–51, (2001).

[29] [Online]. Available: http://www.jstor.org/stable/4480315.

[30] [15] G. W. Schwert, Why does Stock Market Volatility Change Over Time? The Journal of Finance, vol. 44, no. 5, pp. 1115–1153, (1989).

[31] [Online]. Available: http://dx.doi.org/10.1111/j.1540-6261.1989.tb02647.x.

[32] [16] G. V. Attigeri, M. P. M. M, R. M. Pai, and A. Nayak, Stock Market Prediction: A Big Data Approach, In TENCON 2015 - 2015 IEEE Region

[33] 10 Conference, pp. 1–5, November (2015).

[34] [17] P. S. Michael Rechenthin and W. Nick Street, Stock Chatter: Using Stock Sentiment to Predict Price Direction, Algorithmic Finance, (2013).

[18] T. H. Nguyen, K. Shirai and J. Velcin, Sentiment Analysis on Social Media for Stock Movement Prediction, Expert Systems with Applicat