

ABSTRACT

Stock price forecasting is a popular and important domain in financial and academic studies. Share Market is an untidy place for predicting since there are no significant rules to estimate or predict the price of share in the share market. To understand and identify trends in the stock market has been the goal of numerous market analysts, but the patterns of stock trends are often difficult to detect. Recent attempt has been made by companies and individuals to utilize many methods like technical analysis, fundamental analysis, time series analysis, statistical analysis, and more for effectively predicting future stock values.

The proposed system performs stock market analysis by taking stock market historic data. This will be helpful in predicting the short term and medium term closing rate of the companies. The user can predict stock market closing price. The prediction will be done by considering statistical model and neural network.

INTRODUCTION

1.1 Stock Market

A stock market, equity market or share market is the aggregation of buyers and sellers (a loose network of economic transactions) of stocks (also called shares), which represent ownership claims on businesses; these may include securities listed on a public stock exchange, as well as stock that is only traded privately. Examples of the latter include shares of private companies which are sold to investors through equity crowdfunding platforms. Stock exchanges list shares of common equity as well as other security types.

A stock exchange is an exchange where stock brokers and traders can buy and sell shares of stock, bonds, and other securities. Many large companies have their stocks listed on a stock exchange. This makes the stock more liquid and thus more attractive to many investors. The exchange may also act as a guarantor of settlement. Other stocks may be traded "over the counter" (OTC), that is, through a dealer. Some large companies will have their stock listed on more than one exchange in different countries, so as to attract international investors.

1.2 Stock Market Prediction

1.2.1 Definition

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

1.2.2 Prediction methods

Fundamental analysis

Fundamental Analysts are concerned with the company that underlies the stock itself. They evaluate a company's past performance as well as the credibility of its accounts.

Many performance ratios are created that aid the fundamental analyst with assessing the validity of a stock, such as the P/E ratio is perhaps the most famous of all Fundamental Analysts.

What fundamental analysis in stock market is trying to achieve is finding out the true value of a stock, which then can be compared with the value it is being traded with on stock markets and therefore finding out whether the stock on the market is undervalued or not. Finding out the true value can be done by various methods with basically the same principle. The principle being that a company is worth all of its future profits added together. These future profits also have to be discounted to their present value. This principle goes along well with the theory that a business is all about profits and nothing else. Contrary to technical analysis, fundamental analysis is thought of more as a long-term strategy.

Technical Analysis

A time series analysis can also be used to predict the stock prices. Time-series analysis is a basic concept within the field of statistical learning that allows the user to find meaningful information in data collected over time. To demonstrate the power of this technique, it can be used applying it to the S&P 500 Stock Index in order to find the best model to predict future stock values. In investing, a time series tracks the movement of the chosen data points, such as a security's price, over a specified period of time with data points recorded at regular intervals. There is no minimum or maximum amount of time that must be included, allowing the data to be gathered in a way that provides the information being sought by the investor or analyst examining the activity.

Technical analysts or chartists are not concerned with any of the company's fundamentals. They seek to determine the future price of a stock based solely on the trends of the past price (a form of time series analysis). Numerous patterns are employed such as the head and shoulders or cup and Handle. Alongside the patterns, techniques are used such as the exponential moving average (EMA), oscillators, support and resistance levels or momentum and volume indicators. Candle stick patterns, believed to have been first developed by Japanese rice merchants, are nowadays widely used by technical analysts. Technical analysis is rather used for short-term strategies, than the long-term ones. And therefore, it is far more prevalent in commodities and forex markets where traders focus on short-term price movements. There are some basic assumptions used in

this analysis, first being that everything significant about a company is already priced into the stock, other being that the price moves in trends and lastly that history (of prices) tends to repeat itself which is mainly because of the market psychology.

Machine learning and Data Mining Technologies

Support Vector Machine is a machine learning technique used in recent studies to forecast stock prices. These are used as parameters to the SVM model. The model attempts to predict whether a stock price sometime in the future will be higher or lower than it is on a given day. SVM are one of the best binary classifiers. They create a decision boundary such that most points in one category fall on one side of the boundary while most points in the other category fall on the other side of the boundary

With the advent of the digital computer, stock market prediction has since moved into the technological realm.. The most common form of ANN in use for stock market prediction is the feed forward network utilizing the backward propagation of errors algorithm to update the network weights. These networks are commonly referred to as Backpropagation networks. Another form of ANN that is more appropriate for stock prediction is the time recurrent neural network (RNN) or time delay neural network (TDNN). Examples of RNN and TDNN are the Elman, Jordan, and Elman-Jordan networks. (See the Elman And Jordan Networks). For stock prediction with ANNs, there are usually two approaches taken for forecasting different time horizons: independent and joint. The independent approach employs a single ANN for each time horizon, for example, 1-day, 2-day, or 5-day. The advantage of this approach is that network forecasting error for one horizon won't impact the error for another horizon—since each time horizon is typically a unique problem. Since NNs require training and can have a large parameter space; it is useful to optimize the network for optimal predictive ability.

LITERATURE SURVEY

2.1 Tian Ye. “Stock Forecasting Method Based on Wavelet Analysis and ARIMA-SVR Model”, 3rd International Conference on Information Management ,2017

The stock price is a kind of time series which is extremely unstable and is influenced by many factors. There are many external factors affecting the stock market, mainly economic factors, political factors and the company's own situation in three areas. From the emergence of the stock market, there are a variety of methods used by various researchers to study stock price movements. From an economic point of view, investors generally use the traditional fundamental analysis, technical analysis and evolution analysis to predict. These traditional analytical methods are too theoretical and do not adequately reflect the correlation between the data.

ARIMA model is a linear time series model, and SVR is a highly efficient nonlinear prediction model. ARIMA model has good effect in the field of linear regression, and SVR has good effect in the field of nonlinear regression. Stock prices change irregularly and have strong non-linear characteristics, if using ARIMA model or SVR alone for stock price forecasting are likely to produce greater error. In this paper, the ARIMA-SVR prediction model based on wavelet analysis is proposed. Firstly, the original data are decomposed and reconstructed by wavelet. The general trend and unsteady beating of the data are obtained, ie the reconstructed part and the error part. Reconstruction part of the main trend is the long-term trend of data, the error part of the precision data reflects the deviation from the general trend, the original data is the sum of these two parts: the original data = reconstruction part + error part. Then the ARIMA model and the SVR model are used to forecast the reconstructed part and the error part respectively, and the final prediction results are synthesized by the prediction results respectively.

In this paper, the ARIMA-SVR model based on wavelet analysis is used to separate the general trend and error of the original data by wavelet decomposition and wavelet reconstruction. The ARIMA is used to predict the nonlinear part and SVR is used to predict the appropriate part. Use wavelet decomposition and reconstruction to enlarge the stock price of the abstract part and the specific part. The ARIMA time series prediction model is used to predict the abstraction, and use SVR to forecast specific

parts. It is helpful to master the general trend of the stock, and more efficiently to position to the stock price changes.

2.2 Mehak Usmani, Syrd Hasan Adil, Kamran raza and Syed SaadAzhar Ali “Stock Market Predictions Using Machine Learning Techniques”, 3rd International Conference On Computer And Information Sciences (ICCOINS), 2016

The main objective of this research is to predict the market performance of Karachi Stock Exchange (KSE) on dayclosing using different machine learning techniques. As the main focus in this study is to predict the market, there exist few theories that are valid as well as oppose each other. The theory of random walk says price of a security can't be predicted using the historical data. The prediction model uses different attributes as an input and predicts market as Positive & Negative. The attributes used in the model includes Oil rates, Gold & Silver rates, Interest rate, Foreign Exchange (FEX) rate, NEWS and social media feed. The old statistical techniques including Simple Moving Average (SMA) and Autoregressive Integrated Moving Average (ARIMA) are also used as input. The machine learning techniques including Single Layer Perceptron (SLP), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) and Support Vector Machine (SVM) are compared.

The results of this study confirm that machine learning techniques are capable of predicting the stock market performance. Karachi Stock Market with KSE-100 index does follow a behavior that can be predicted using machine learning techniques. The algorithm MLP performed best as compared to other techniques. The Multi-Layer Perceptron algorithm of machine learning predicted 77% correct market performance. Also the most related attribute to the KSE performance was found to be the petrol price and FEX proved to have no effect on the KSE performance.

2.3 Peihao Li, Chaoqun Jing, Tian Liang, Mingjia Liu, Zhenglin Chen, Li Guo “**Autoregressive Moving Average Modeling in the Financial Sector**”, 2nd Int. Conference on Information Technology, Computer and Electrical Engineering (ICITACEE), 2015

Time series modelling has long been used to make forecast in different industries with a variety of statistical models currently available. Methods for analyzing changing patterns of stock prices have always been based on fixed time series. Considering that these methods have ignored some crucial factors in stock prices, they use ARIMA model to predict stock prices given the stock-trading volume and exchange rate as independent variables to achieve a more stable and accurate prediction process. In this paper it will introduce the modeling process and give the estimate SSE (Shanghai Stock Exchange) Composite Index to see the model's estimation performance.

Autoregressive moving average model (ARIMA) is a statistical analysis model which utilizes time series data to predict future data. It is a form of regression analysis that seeks to predict future movements along the seemingly random walk taken by stocks and the financial market by examining the differences between values in the series instead of using the actual data values.

Time series ARIMA models are applied with time series data of variables measured over time. Since SSECI stationary series has no trend and its variations around its mean have a constant amplitude, and it wiggles in a consistent fashion, in this paper they use ARIMA model to simulate and estimate future stock market trend. ARIMA models are, in theory, the most general class of models for forecasting a time series which can be made to be “stationary” by differencing. The prediction equation is simply a linear equation that refers to past values of original time series and past values of the errors. The trend can be correctly predicted by the model meaning that both the model and independent variables are correctly selected.

2.4 Shashank Tiwari, Dr. Sudha Gupta, “Stock Price Prediction Using Data Analytics”, IEEE, 2017

Accurate financial prediction is of great interest for investors. This paper proposes use of Data analytics to be used in assist with investors for making right financial prediction so that right decision on investment can be taken by Investors. Two platforms are used for operation: Python and R. various techniques like Arima, Holt winters, Neural networks (Feed forward and Multi-layer perceptron), linear regression and time series are implemented to forecast the opening index price performance in R. While in python Multi-layer perceptron and support vector regression are implemented for forecasting Nifty 50 stock price and also sentiment analysis of the stock was done using recent tweets on Twitter.

The results of the same are then compared and the most accurate and efficient model is found. Another motivation for research in this field is that it possesses many theoretical and experimental challenges.

This research entailed the development of various methods to find the most accurate model for prediction of prices of the stock. They increased the nodes of neural network and got better results. From the above listed methods it was found that Feed Forward Neural network provides the highest accuracy for the opening price of stock. It was also observed that different method can be efficient for different type of stocks and prices. The result obtained was the opening price of the stock and that too was average for a full month. So an improvement in this system can be achieved by forecasting the opening price of each day.

2.5 David M. Q. Nelson, Adriano C. M. Pereira, Renato A. de Oliveira, “Stock Market’s Price Movement Prediction with LSTM Neural Networks”, 2017 International Joint Conference on Neural Networks (IJCNN), 2017

One common approach is to use Machine Learning algorithms to learn from price historic data to predict future prices. This article goes in that direction but studying an specific method using recurrent neural networks. Such networks have a short term memory capability and the hypothesis to explore here is that this feature can present gains in terms of results when compared to others more traditional approaches in

Machine Learning field. The objective of this project is to study the applicability of recurrent neural networks, in particular the LSTM networks, on the problem of stocks market prices movements prediction. Assess their performance in terms of accuracy and other metrics through experiments on top of real life data and analyze if they present any sort of gain in comparison to more traditional machine learning algorithms.

The main contributions of this work are the following: (1) a new price movement prediction model for stock markets using deep learning based technique; (2) the validation of the model using real data from Brazilian stock exchange; (3) evaluation of the model by comparing and analyzing it against some typical baselines.

When compared to the other machine learning models it has displayed considerable gains in terms of accuracy, but in another hand we believe that variance could be lower and that would contribute for a more reliable model. When it comes to the financial results it's important to note that it was able to keep it positive for all stocks, even though it didn't necessarily had the best results when compared to the baselines. Another positive aspect is that it had a high return ratio per operation, meaning that it had more success on detecting high variations which is extremely important when account transactions costs and taxes are taken into account. The LSTM based model offers less risks when compared to the other strategies.

2.6 Tingwei Gao, Yueting Chai, Yi Liu, “Applying Long Short Term Momory Neural Networks for Predicting Stock Closing Price”, 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017

Forecasting the movement of stock market is the use of a model to predict future values behavior based on previously observed values. In effect, the usage of machine learning (ML) algorithms for stock market behavior modelling and prediction is an active research topic during the past decade. And one of the most important and most widely used method is the model based on artificial neural network (ANN). RNN is a special kind of ANN, which is designed to learn sequential or time varying patterns. LSTM recurrent neural network is a special and important architecture of RNN that excels at remembering values for either long or short durations of time.

In this research, they hypothesize that combining RNNs with informative input variables can provide a more effective and reliable stock market forecast system. Furthermore,

they propose and validate a novel stock closing price prediction model based on LSTM and stock basic data. At last, the main contributions can be related to the streams (a)-(b) as follows: (a) effectively apply LSTM network, which is a most popular structure of RNN, to design a stock prediction system. The study may support further advances in financial time series analysis, especially on how to use RNNs for stock closing price forecast; (b) properly designed comparison experiments are conducted to evaluate the model. And a series of performance metrics are utilized to provide an overall evaluation. To evaluate the performance of the model and other models, they conducted a series of experiments. And the results were measured by some criteria. The first and the simplest measure of predictive value accuracy is mean absolute error (MAE). MAE is a common approach in the forecasting domain. The root mean square error (RMSE) is used to compute the root means square error of a variable. The mean absolute percentage error (MAPE) computes the percentage of error between the actual values and predicted values. To get more evaluation information, introduced the average mean absolute percentage error (AMAPE). To demonstrate the utility of the system, they tested five different models namely moving average (MA), exponential moving average (EMA), support vector machine (SVM) and LSTM. The forecast target is the closing price of next day. All predictive experiments were conducted using to predict the next day's closing price movements in S&P500 stocks.

PROBLEM STATEMENT

The problem statement can be defined as:

“To provide users the accurate closing rate using the historic data and the stock analysis”

Input: Dataset of stock history with **Timestamp** of various companies.

Output: Predicted closing price of a stock and stock trend analysis.

- An Accurate prediction of closing price of the stock based on historical data needs to be done so as to provide accurate value of closing price on how the stock price fluctuated previously.
- The best of the regressive model which is the ARIMA model and Support Vector Regression are compared for the prediction accuracy.

SYSTEM REQUIREMENTS

4.1 Functional Requirements

The various functional requirements of the system are:-

- The system should predict the closing price for the next 1 to 20 days.
- The user interface should display the graph of variation in stock price over the period of time.

4.2 Non Functional Requirements

- **Functionality** :Provide customers with predictive data on the stock market
- **Usability** : Site will be simple and clean so that the customer can view everything easily.
- **Reliability** : Prediction models will be updated daily. Data backed up in the case of a site failure.
- **Performance**: Current data transfer rates should allow for quick and easy navigation of the site.

4.3 Software Requirements

The software requirements cover software required for development and running the application. The software requirements are specified in the following table:-

Operating System	Windows 7 or higher
Development Platform	Jupyter notebook, Django
Third party Software	Python 3.6 , Anaconda IDE, Notepad++,
Input	Forecast time period and model to be used (ARIMA/LSTM)

4.4 Hardware Requirements

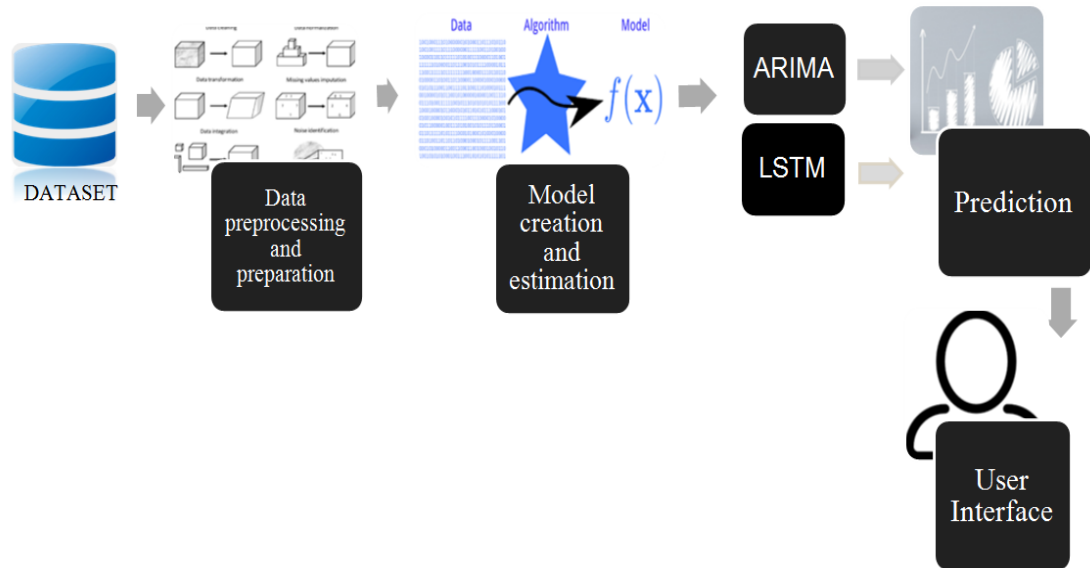
The hardware requirements are based on the amount of hardware resources needed to run all the software applications needed for the development.

Requirement	Minimum	Recommended
Memory	1.5 GB	2GB or more
Free Disk Space	512 MB	1GB or more
Processor speed	1 GHz	2 GHz or faster

ARCHITECTURE

5.1 Architecture

The proposed architecture for the stock market prediction and analysis is as shown below:



5.2 Modules

5.2.1 Data Acquisition and Pre-processing

The historical data available on National stock exchange of India and the stock prices along with its metadata is extracted from the companies of interest such as AMAZON, UBER ,GOOGLE,AAL etc. The data is scraped and extracted into a csv file.

- Inaccurate data (missing data) - There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics and much more.
- The presence of noisy data (erroneous data and outliers) - The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.
- Inconsistent data - The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more.

The process of organizing data into groups and classes on the basis of certain characteristics is known as the classification of data. Classification helps in making comparisons among the categories of observations. It can be either according to numerical characteristics or according to attributes. So here we need to visualize the prepared data to find whether the training data contains the correct label, which is known as a target or target attribute.

Next, we will slice a single data set into a training set and test set.

- **Training set**—a subset to train a model.
- **Test set**—a subset to test the trained model.

You could imagine slicing the single data set as follows:



Fig: slicing a single data set into a training set and test set

Make sure that your test set meets the following two conditions:

- ✓ Is large enough to yield statistically meaningful results.
- ✓ Is representative of the data set as a whole? In other words, don't pick a test set with different characteristics than the training set.

5.2.2 Model Building and training

The term ML model refers to the model artefact that is created by the training process. The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

Two machine learning models to be compared that is the ARIMA model and Long short term Model.

ARIMA model is built using Box Jenkins Methodology and dicky fuller test is conducted to decide the lag parameters and check for stationarity. The model is built and the rolling mean is estimated for a given window period (12). The residuals, seasonality and

stationarity are checked. When the user gives a window period for which he needs an estimation a graph comparison of predicted and original closing price is shown along with the confidence interval.

LSTM neural network is trained by providing the input gate, forget gate parameters using keras. The number of epochs is defined and set to 40 for this project. Whenever a user sets a particular company which he wishes to buy the stock of, closing price for the next day and variation of the stock closing price is shown in the form of a graph on the user interface. The backend training of neural network and change of lag , accuracy parameters for each iteration of the training the network could be seen on the command prompt.

5.2.3 Model Validation and Result Analysis

In testing phase, the model is applied to new set of data. The training and test data are two different datasets. The goal in building a predictive model is to have the model perform well. On the training set, as well as generalize well on new data in the test set. Once the build model is tested then we will pass real time data for the prediction. Once prediction is done then we will analyse the output to find out the crucial information.

Never train on test data. If you are seeing surprisingly good results on your evaluation metrics, it might be a sign that you are accidentally training on the test set. For example, high accuracy might indicate that test data has leaked into the training set.

The one-period ahead forecast error is, the distinction between a forecast error and a residual is the same as between a forecast and a predicted value:

- a residual is “in-sample”
- a forecast error is “out-of-sample” – the value of Y_{T+1} isn’t used in the estimation of the regression coefficients.

$$\text{forecast error} = Y_{T+1} - \hat{Y}_{T+1|T}$$

We will analyse the model using following method:

The mean squared forecast error (MSFE) is,

$$E(Y_{T+1} - \hat{Y}_{T+1|T})^2 = E(u_{T+1})^2 + E[(\hat{\beta}_0 - \beta_0) + (\hat{\beta}_1 - \beta_1)Y_T + (\hat{\beta}_2 - \beta_2)X_T]^2$$

- $MSFE = \text{var}(u_{T+1}) + \text{uncertainty arising because of estimation error}$
- If the sample size is large, the part from the estimation error is (much) smaller than $\text{var}(u_{T+1})$, in which case $MSFE \approx \text{var}(u_{T+1})$
- The *root mean squared forecast error (RMSFE)* is the square root of the MS forecast error:

$$RMSFE = \sqrt{E[(Y_{T+1} - \hat{Y}_{T+1|T})^2]}$$

The root mean squared forecast error (RMSFE)

$$\text{RMSFE} = \sqrt{E[(Y_{T+1} - \hat{Y}_{T+1})^2]}$$

- The RMSFE is a measure of the spread of the forecast error distribution.
 - The RMSFE is like the standard deviation of u_t , except that it explicitly focuses on the forecast error using estimated coefficients, not using the population regression line.
 - The RMSFE is a measure of the magnitude of a typical forecasting “mistake”
-
- The two models ARIMA and LSTM are checked for prediction accuracy with respect to forecast window period.
 - The respective outputs from both the models are plotted and compared to the original closing price.

5.2.4 User Interface

The User interface is built using Django API and view is built in HTML, SCSS. The form validations are done using JavaScript.

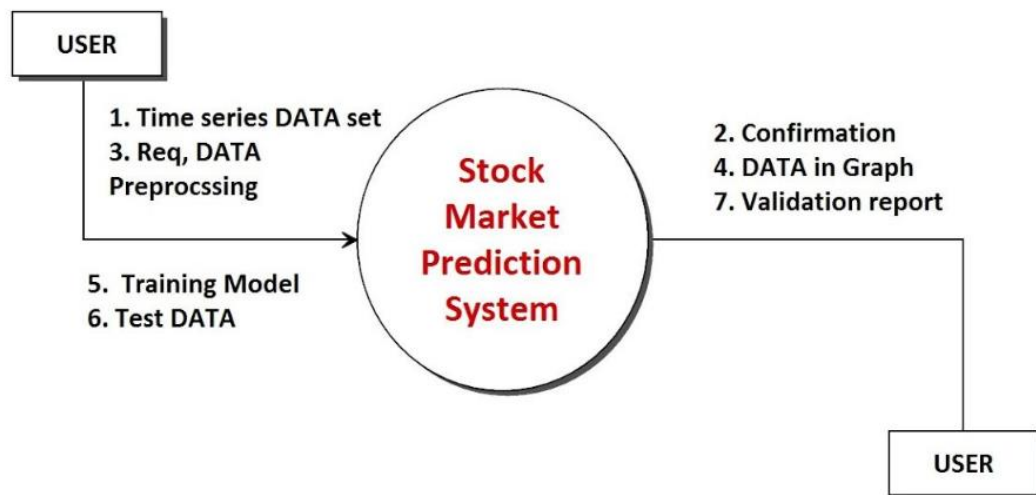
The user interface consists of registration, login pages, home page and result pages.

Once the user logs in the user is asked to select a particular company whose stock price they are interested in. the neural network model is then trained for that particular company change in the stock closing price over a time period is displayed.

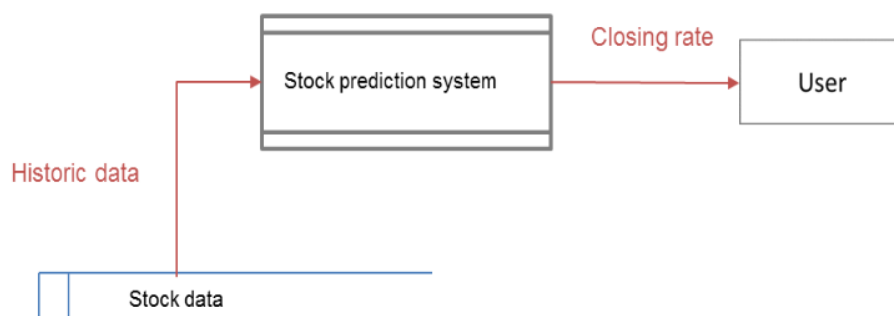
DESIGN

6.1 Data-flow Diagram

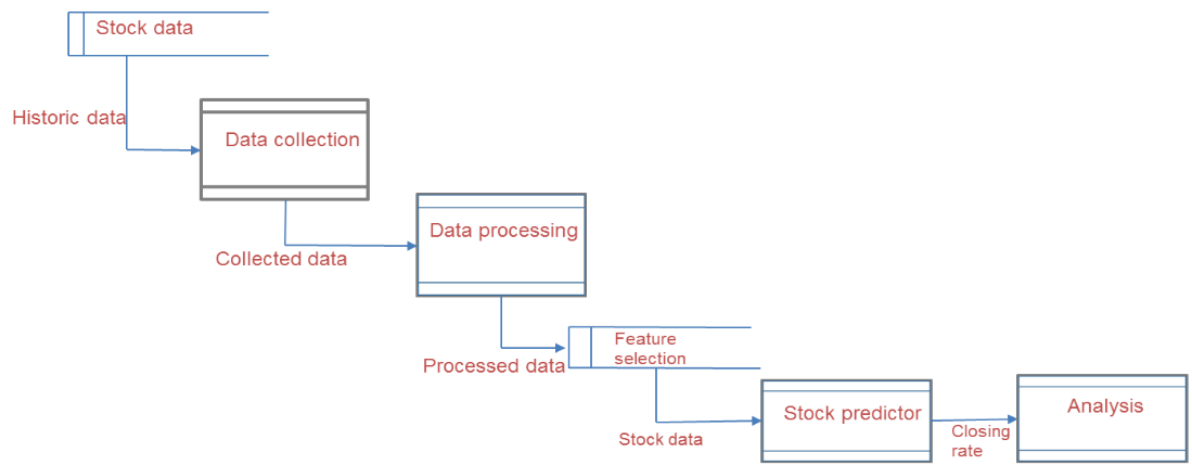
The data flow diagram shows the flow of data between each of the data and the repositories.



DFD (Level 0)



DFD (Level 1)



6.2 Algorithms

6.2.1 ARIMA

ARIMA stands for Autoregressive Integrated Moving Average. ARIMA is also known as Box-Jenkins approach. Box and Jenkins claimed that non-stationary data can be made stationary by differencing the series, Y_t . The general model for Y_t is written as,

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} \dots \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots \theta_q \epsilon_{t-q}$$

Where Y_t is the differenced time series value, ϕ and θ are unknown parameters and ϵ are independent identically distributed error terms with zero mean. Here, Y_t is expressed in terms of its past values and the current and past values of error terms.

The ARIMA model combines three basic methods:

- AutoRegression (AR) – In auto-regression, the values of a given time series data are regressed on their own lagged values, which is indicated by the “p” value in the ARIMA model.
- Differencing (I-for Integrated) – This involves differencing the time series data to remove the trend and convert a non-stationary time series to a stationary one. This is indicated by the “d” value in the ARIMA model. If $d = 1$, it looks at the difference between two-time series entries, if $d = 2$ it looks at the differences of the differences obtained at $d = 1$, and so forth.
- Moving Average (MA) – The moving average nature of the ARIMA model is represented by the “q” value which is the number of lagged values of the error term.

This model is called Autoregressive Integrated Moving Average or ARIMA(p,d,q) of Y_t .

Algorithm:

1. Plot(dataset)

while(graph is non-stationary = True)

(smoothen graph to make it stationary)

2. ACF/PACF(stationary Graph)

3. Estimate all possible model parameters

4. Calculate AIC values of all model parameters

5. Plot (with model parameters)

if(no lag)

(Forecast Dataset with these parameters)

else choose other estimated model parameters and repeat step 3

6.3.2 Box Jenkins

Stationary and seasonality

The first step in developing a Box–Jenkins model is to determine if the time series is stationary and if there is any significant seasonality that needs to be modelled.

Detecting stationary

Stationary can be assessed from a run sequence plot. The run sequence plot should show constant location and scale. It can also be detected from an autocorrelation plot. Specifically, non-stationary is often indicated by an autocorrelation plot with very slow decay.

Detecting seasonality

Seasonality (or periodicity) can usually be assessed from an autocorrelation plot, a seasonal subseries plot, or a spectral plot.

Differencing to achieve stationary

Box and Jenkins recommend the differencing approach to achieve stationary. However, fitting a curve and subtracting the fitted values from the original data can also be used in the context of Box–Jenkins models.

Seasonal differencing

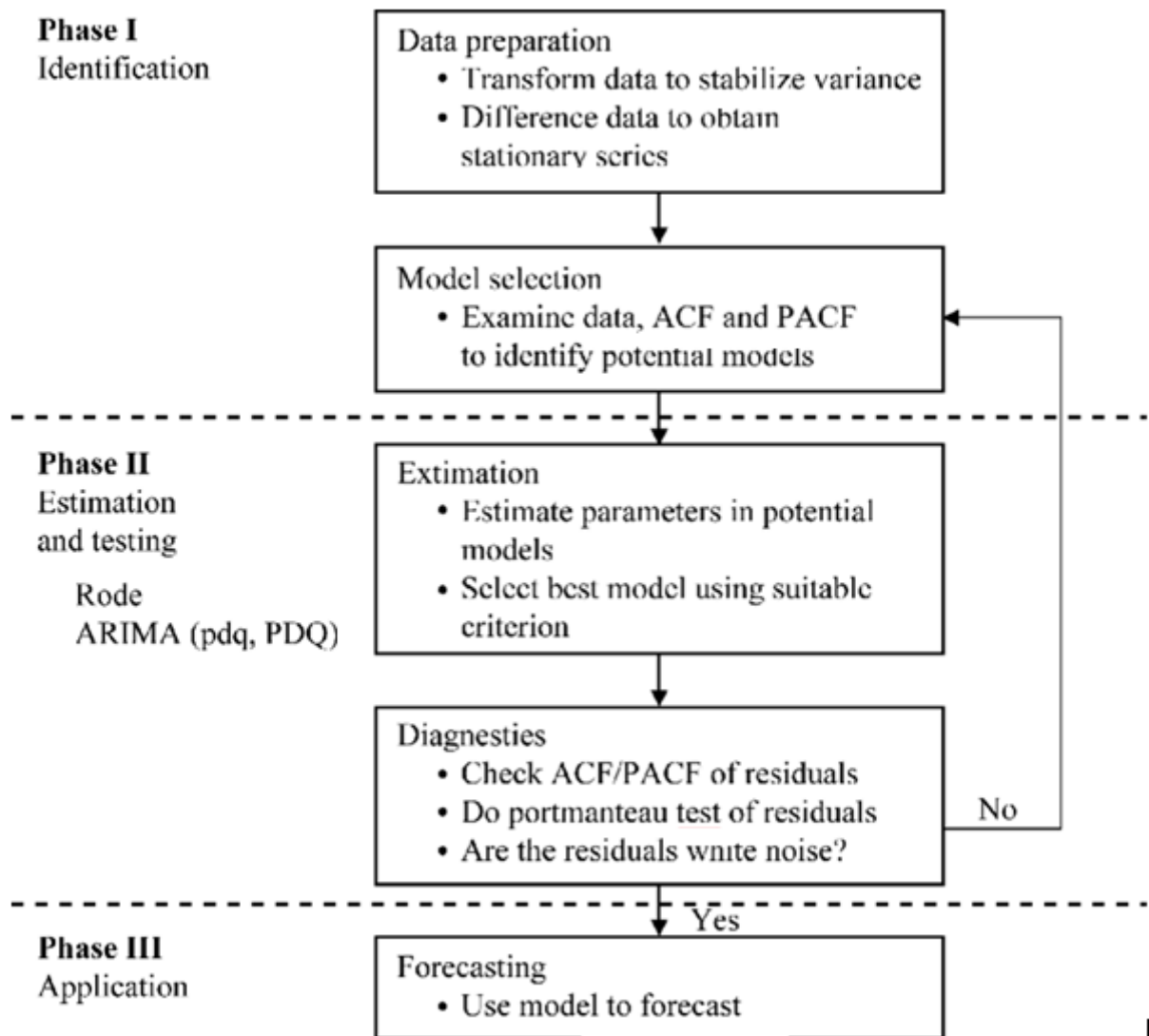
At the model identification stage, the goal is to detect seasonality, if it exists, and to identify the order for the seasonal autoregressive and seasonal moving average terms. For many series, the period is known and a single seasonality term is sufficient. For example, for monthly data one would typically include either a seasonal AR 12 term or a seasonal MA 12 term. For Box–Jenkins models, one does not explicitly remove seasonality before fitting the model. Instead, one includes the order of the seasonal terms in the model specification to the ARIMA estimation software. However, it may be helpful to apply a seasonal difference to the data and regenerate the autocorrelation and partial autocorrelation plots. This may help in the model identification of the non-seasonal component of the model. In some cases, the seasonal differencing may remove most or all of the seasonality effect.

Identify p and q

Once stationary and seasonality have been addressed, the next step is to identify the order (i.e. the p and q) of the autoregressive and moving average terms. Different authors have different approaches for identifying p and q. Brockwell and Davis (1991)^[3] state "our prime criterion for model selection [among ARMA(p,q) models] will be the AICc", i.e. the Akaike information criterion with correction. Other authors use the autocorrelation plot and the partial autocorrelation plot, described below.

Autocorrelation and partial autocorrelation plots

The sample autocorrelation plot and the sample partial autocorrelation plot are compared to the theoretical behavior of these plots when the order is known.



6.2.3 LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM Pseudo code

Step 1: Decide the data parameters.

Step 2: Model those parameters.

Step 3: Load and save the data.

Performed stop word elimination and clean data (remove special characters).

Step 4: Save parameters to file.

Step 5: Performed cross validation of data (used k-fold cross validation).

Step 6: Train data for classifier.

Step 7: Used LSTM classifier.

Step 8: Repeat step 5 to 7 until the end of training data.

IMPLEMENTATION CODE

7.1 ARIMA Implementation

```

import os

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from pylab import rcParams

from statsmodels.tsa.stattools import adfuller

from statsmodels.tsa.stattools import acf, pacf

from statsmodels.tsa.seasonal import seasonal_decompose

from statsmodels.tsa.arima_model import ARIMA

import pandas as pd


train_df=pd.read_csv('stock_data.csv')

train_df.head()

train_df['date']=pd.to_datetime(train_df['date'])

train_df.dtypes


company_data = train_df.loc[train_df['Name'] == 'GOOGL']

company_data.head()

company_data.drop(company_data.loc[company_data['volume'].isnull()].index)

company_data =

company_data.drop(company_data.loc[company_data['open'].isnull()].index)

company_data.isnull().sum()


rcParams['figure.figsize'] = 20, 10

fig = plt.figure()

ax1 = fig.add_subplot(221)

ax1 = sns.lineplot(x="date", y="open",

```



```

        markers=True, dashes=False, data=company_data)
ax2 = fig.add_subplot(222, sharex=ax1, sharey=ax1)
ax2 = sns.lineplot(x="date", y="high",
        markers=True, dashes=False, data=company_data)
ax3 = fig.add_subplot(223, sharex=ax1, sharey=ax1)
ax3 = sns.lineplot(x="date", y="low",
        markers=True, dashes=False, data=company_data)
ax4 = fig.add_subplot(224, sharex=ax1, sharey=ax1)
ax4 = sns.lineplot(x="date", y="close",
        markers=True, dashes=False, data=company_data)
company_cols=company_data[['date','open','high','low','close']]
company_cols=company_cols.set_index('date')
#Determine rolling statistics
rolling_mean = company_cols.rolling(window=12).mean() #window size 12 denotes 12
months, giving rolling mean at yearly level
rolling_std_deviation = company_cols.rolling(window=12).std()

print(rolling_mean,rolling_std_deviation)

#Plot rolling statistics
original_plot = plt.plot(company_cols['high'], color='black', label='Original Plot')
mean_plot = plt.plot(rolling_mean['high'], color='red', label='Rolling Mean Plot')
std_plot = plt.plot(rolling_std_deviation['high'], color='green', label='Rolling Std Plot')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show()

# Perform Augmented Dickey–Fuller test:
print('Results of Dickey Fuller Test:')

```

```
dfctest = adfuller(company_cols['high'], autolag='AIC')
dfoutput = pd.Series(dfctest[0:4], index=['Test Statistic','p-value','#LagsUsed','Number of
Observations Used'])
for key,value in dfctest[4].items():
dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
#The below transformation is required to make series stationary
company_cols_high=company_cols[['high']]
company_cols_high
company_cols_high_logScale = np.log(company_cols_high)
plt.plot(company_cols_high_logScale, color='green')
plt.show()
```

```
movingAverage = company_cols_high_logScale.rolling(window=12).mean()
movingSTD = company_cols_high_logScale.rolling(window=12).std()
plt.plot(company_cols_high_logScale)
plt.plot(movingAverage, color='red')
plt.show()
```

```
logScaleMinusMovingAverage =company_cols_high_logScale - movingAverage
logScaleMinusMovingAverage.head(12)
```

```
#Remove NAN values
logScaleMinusMovingAverage.dropna(inplace=True)
logScaleMinusMovingAverage.head(10)
def test_stationarity(timeseries):
```

```
    #Determine rolling statistics
    movingAverage = timeseries.rolling(window=12).mean()
```

```

movingSTD = timeseries.rolling(window=12).std()

#Plot rolling statistics
orig = plt.plot(timeseries, color='blue', label='Original')
mean = plt.plot(movingAverage, color='red', label='Rolling Mean')
std = plt.plot(movingSTD, color='black', label='Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)

#Perform Dickey-Fuller test:
print('Results of Dickey Fuller Test:')
dftest = adfuller(timeseries['high'], autolag='AIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#LagsUsed','Number of
Observations Used'])
for key,value in dftest[4].items():
dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
test_stationarity(logScaleMinusMovingAverage)
exponentialDecayWeightedAverage = company_cols_high_logScale.ewm(halflife=12,
min_periods=0, adjust=True).mean()
plt.plot(company_cols_high_logScale)
plt.plot(exponentialDecayWeightedAverage, color='red')
plt.show()

datasetLogScaleMinusExponentialMovingAverage = company_cols_high_logScale -
exponentialDecayWeightedAverage
test_stationarity(datasetLogScaleMinusExponentialMovingAverage)
datasetLogDiffShifting = company_cols_high_logScale -
company_cols_high_logScale.shift()
plt.plot(datasetLogDiffShifting)
plt.show()

```

```
decomposition = seasonal_decompose(company_cols_high_logScale,freq=52)
```

```
trend = decomposition.trend
```

```
seasonal = decomposition.seasonal
```

```
residual = decomposition.resid
```

```
plt.plot(company_cols_high_logScale, label='Original',color='red')
```

```
plt.plot(trend, label='Trend',color='black')
```

```
plt.plot(seasonal, label='Seasonality',color='blue')
```

```
plt.plot(residual, label='Residuals',color='green')
```

```
plt.legend(loc='best')
```

```
plt.show()
```

```
decomposedLogData = residual
```

```
decomposedLogData.dropna(inplace=True)
```

```
test_stationarity(decomposedLogData)
```

```
decomposedLogData = residual
```

```
decomposedLogData.dropna(inplace=True)
```

```
test_stationarity(decomposedLogData)
```

```
datasetLogDiffShifting.describe()
```

```
datasetLogDiffShifting.dropna(inplace=True)
```

```
#ACF & PACF plots
```

```
from statsmodels.tsa.stattools import acf, pacf
```

```
lag_acf = acf(datasetLogDiffShifting, nlags=20)
```

```
lag_pacf = pacf(datasetLogDiffShifting, nlags=20, method='ols')
```

```
#Plot ACF:
```

```
plt.subplot(121)
```

```

plt.plot(lag_acf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-1.96/np.sqrt(len(datasetLogDiffShifting)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(datasetLogDiffShifting)), linestyle='--', color='gray')
plt.title('Autocorrelation Function')

#Plot PACF
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-1.96/np.sqrt(len(datasetLogDiffShifting)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(datasetLogDiffShifting)), linestyle='--', color='gray')
plt.title('Partial Autocorrelation Function')

plt.tight_layout()

model = ARIMA(company_cols_high_logScale, order=(2,1,2))
results = model.fit()
plt.plot(datasetLogDiffShifting)
plt.plot(results.fittedvalues, color='red')
plt.title('RSS: %.4f'%sum((results.fittedvalues - datasetLogDiffShifting['high'])**2))
plt.show()

print('Plotting ARIMA model')
vexponentialDecayWeightedAverage = company_cols_high_logScale.ewm(halflife=12,
min_periods=0, adjust=True).mean()
plt.plot(company_cols_high_logScale)
plt.plot(exponentialDecayWeightedAverage, color='red')
plt.show()

predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()

```

```

print(predictions_ARIMA_diff_cumsum)

predictions_ARIMA_log = pd.Series(company_cols_high_logScale['high'].iloc[0],
index=company_cols_high_logScale.index)

predictions_ARIMA_log =
predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum, fill_value=0)
predictions_ARIMA_log.head()

# Inverse of log is exp.
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(company_cols_high_logScale)
plt.plot(predictions_ARIMA)
plt.show()

company_cols_high_logScale.head()

results.plot_predict(start=1,end=500)
next_day_forecast=results.forecast()[0].

```

7.2 LSTM implementation

7.2.1 code to run server

```

#!/usr/bin/env python

import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"Stock_Market_ANN.settings")

    try:

```

```

from django.core.management import execute_from_command_line

except ImportError:

    # The above import may fail for some other reason. Ensure that the
    # issue is really that Django is missing to avoid masking other
    # exceptions on Python 2.

    try:
        import django
    except ImportError:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        )
    raise

execute_from_command_line(sys.argv)

```

7.2.2 LSTM network training

```

from django.shortcuts import render
from StockMarketApp.forms import *
import openpyxl
import pandas as pd

from itertools import count
import matplotlib.pyplot as plt
import numpy as np
import collections
from random import randint
from matplotlib import style
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import seaborn as sns

import itertools
import os

from django.shortcuts import render_to_response
from django.http import HttpResponseRedirect
from django import forms
from django.template import RequestContext
#import django_excel as excel
from .forms import *

```

```
# Extra Imports for the Login and Logout Capabilities
from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponseRedirect, HttpResponse
from django.urls import reverse
from django.contrib.auth.decorators import login_required

import django.contrib.sessions

import math
from matplotlib.pyplot import rcParams
from keras.models import Sequential
from keras.layers import Dense , BatchNormalization , Dropout , Activation
from keras.layers import LSTM , GRU
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.optimizers import Adam

# Create your views here.
# Create your views here.
# Create your views here.
def index(request):
    return render(request, 'StockMarketApp/index1.html')

@login_required
def special(request):
    # Remember to also set login url in settings.py!
    # LOGIN_URL = '/basic_app/user_login/'
    return HttpResponse("You are logged in. Nice!")

@login_required
def user_logout(request):
    # Log out the user.
    logout(request)
    # Return to homepage.
    return HttpResponseRedirect(reverse('index'))

def user_login(request):
    #print("=====Its Came Inside user
login=====")
    #username = request.POST.get('username')
    #password = request.POST.get('password')

    #return render(request, 'StockMarketApp/home.html', { })

    if request.method == 'POST':
        # First get the username and password supplied
        username = request.POST.get('username')
        password = request.POST.get('password')

        print(username)
        print(password)
```



```

# Django's built-in authentication function:
user = authenticate(username=username, password=password)

print(user)

# If we have a user
if user:
    #Check if the account is active
    if user.is_active:
        # Log the user in.
        login(request,user)
        # Send the user back to some page.
        # In this case their homepage.
        print("Login success")
        request.session['username'] = username
        #return HttpResponseRedirect(reverse('home'))
        return render(request, 'StockMarketApp/home.html', {"username":"Welcome! "
+username})
    else:
        # If account is not active:
        return HttpResponseRedirect("Your account is not active.")
    else:
        print("Someone tried to login and failed.")
        print("They used username: {} and password: {}".format(username,password))
        return HttpResponseRedirect("Invalid login details supplied.")
else:
    #Nothing has been provided for username or password.
    return render(request, 'StockMarketApp/login.html', {})

def register(request):
    print("=====Its Came Inside register=====")

    registered = False

    if request.method == 'POST':

        # Get info from "both" forms
        # It appears as one form to the user on the .html page
        user_form = UserForm(data=request.POST)
        profile_form = ProfileInfoForm(data=request.POST)

        # Check to see both forms are valid
        if user_form.is_valid() and profile_form.is_valid():

            # Save User Form to Database
            user = user_form.save()

            # Hash the password
            user.set_password(user.password)

            # Update with Hashed password
            user.save()

            # Now we deal with the extra info!

            # Can't commit yet because we still need to manipulate

```

```

profile = profile_form.save(commit=False)

# Set One to One relationship between
# UserForm and UserProfileInfoForm
profile.user = user

# Now save model
profile.save()

# Registration Successful!
registered = True
else:
    # One of the forms was invalid if this else gets called.
    print(user_form.errors, profile_form.errors)

else:
    # Was not an HTTP post so we just render the forms as blank.
    user_form = UserForm()
    profile_form = ProfileInfoForm()

# This is the render and context dictionary to feed
# back to the registration.html file page.
return render(request, 'StockMarketApp/registration.html', {
    'user_form': user_form,
    'profile_form': profile_form,
    'registered': registered})

def user_home(request):
    if request.session.has_key('username'):
        username = request.session['username']
        #return render(request, 'basic_app/menu.html', {'username': username})
        print("Username: ", username)
        dic = {'username': "Welcome! " + username}
        return render(request, 'StockMarketApp/home.html', dic)

def stockMarket(request):
    print("=====Its came to
water_fraud()=====")
    if request.method == "POST":
        print("Its Came To If")
        BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
        #Read Dataset
        sdata = os.path.join(BASE_DIR, 'Dataset\stock_data.csv')
        print(sdata)
        s_data = pd.read_csv(sdata)
        print(s_data.head())
        #print(wdata.shape)
    return render(request, 'StockMarketApp/prediction.html')

def process_data(data, n_features):
    dataX, dataY = [], []
    for i in range(len(data)-n_features-1):
        a = data[i:(i+n_features), 0]

```

```

dataX.append(a)
dataY.append(data[i + n_features, 0])
return np.array(dataX), np.array(dataY)

def model_score(model, X_train, y_train, X_test, y_test):
    trainScore = model.evaluate(X_train, y_train, verbose=0)
    print('Train Score: %.5f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))
    testScore = model.evaluate(X_test, y_test, verbose=0)
    print('Test Score: %.5f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))
    return trainScore[0], testScore[0]

def forecasting(request):
    print("=====Its came to forecasting()=====")
    if request.method == "POST":
        print("=====Its came inside forecasting if=====")
        companyName = request.POST.get('companyName')
        print(companyName)
        BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
        #Read Dataset
        sdata = os.path.join(BASE_DIR, 'Dataset\stock_data.csv')
        #print(df.head())
        #print(df.shape)
        df = pd.read_csv(sdata)
        df.Name.unique()[0:20]
        df['date']=pd.to_datetime(df['date'])
        df.dtypes

        company_df = df.loc[df['Name'] == companyName ]
        company_df.head()

        company_df = company_df.drop(company_df.loc[company_df['open'].isnull()].index)
        print(company_df.isnull().sum())

        company_df=company_df[['date','open','high','low','close']]
        company_df=company_df.set_index('date')

        company_df_close = company_df[['close']]
        print(company_df_close.head())
        scaler = MinMaxScaler(feature_range=(0, 1))
        stocks = scaler.fit_transform(company_df_close)

        train = int(len(stocks) * 0.80)
        test = len(stocks) - train

        print(train, test)

        train = stocks[0:train]
        print(train)

        test = stocks[len(train) : ]
        train = train.reshape(len(train) , 1)
        test = test.reshape(len(test) , 1)

        print(train.shape , test.shape)

```

```

#Building the RNN
n_features = 2

trainX, trainY = process_data(train, n_features)
testX, testY = process_data(test, n_features)

print(trainX.shape , trainY.shape , testX.shape , testY.shape)
trainX = trainX.reshape(trainX.shape[0] , 1 ,trainX.shape[1])
testX = testX.reshape(testX.shape[0] , 1 ,testX.shape[1])
model = Sequential()
model.add(GRU(256 , input_shape = (1 , n_features) , return_sequences=True))
model.add(Dropout(0.4))
model.add(LSTM(256))
model.add(Dropout(0.4))
model.add(Dense(64 , activation = 'relu'))
model.add(Dense(1))

print(model.summary())
adam = Adam(lr = 0.0005)
model.compile(loss='mean_squared_error', optimizer= adam, metrics = ['accuracy'])

history = model.fit(trainX, trainY, epochs=40 , batch_size = 128 , validation_data =
(testX,testY))
model_score(model, trainX, trainY , testX, testY)

pred = model.predict(testX)
pred = scaler.inverse_transform(pred)
pred[:10]

testY = testY.reshape(testY.shape[0] , 1)
testY = scaler.inverse_transform(testY)
testY[:10]

print("Red - Predicted Stock Prices , Blue - Actual Stock Prices")
plt.rcParams["figure.figsize"] = (15,7)
plt.plot(testY , 'b')
plt.plot(pred , 'r')
plt.xlabel("Time")
plt.ylabel('Stock Prices')
plt.title('Check the accuracy of the model with time')
plt.grid(True)
plt.savefig(os.path.join(BASE_DIR, 'static/img/predicted_result.png'))
#plt.show()
imagePath = os.path.join(BASE_DIR, "static/img/predicted_result.png")

print("=====imagePath=====")
print(imagePath)

dic = {'companyName' : companyName}
return render(request, 'StockMarketApp/result.html', dic)

```

7.3 Django User Interface

```

from django.contrib import admin

from StockMarketApp.models import ProfileInfo

# Register your models here.


admin.site.register(ProfileInfo)

from django.apps import AppConfig


class StockmarketappConfig(AppConfig):
    name = 'StockMarketApp'
from django import forms
from django.contrib.auth.models import User
from StockMarketApp.models import ProfileInfo


class UserForm(forms.ModelForm):
    #password = forms.CharField(widget=forms.PasswordInput())
    password = forms.CharField(widget=forms.PasswordInput())


    class Meta():
        model = User
        fields = ('username', 'email', 'password')


class ProfileInfoForm(forms.ModelForm):
    class Meta():
        model = ProfileInfo
        fields = ('phone_no', 'address')


class UploadFileForm(forms.Form):
    phone_no = forms.CharField(widget=forms.TextInput(attrs={'class':'form-control',
'placeholder':'Phone No'}), required=True, max_length=50)
    address = forms.CharField(widget=forms.TextInput(attrs={'class':'form-control',
'placeholder':'Address'}), required=True, max_length=50)


from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class ProfileInfo(models.Model):
    user = models.OneToOneField(User)

    #additional
    phone_no = models.CharField(max_length=128)
    address = models.CharField(max_length=128)

    def __str__(self):
        return self.user.username
from django.conf.urls import url

```

```
from StockMarketApp import views

#TEMPLATE URLS!
app_name = 'StockMarketApp'

urlpatterns = [
    url(r'^user_login/', views.user_login, name='user_login'),
    url(r'^register/', views.register, name='register'),
    url(r'^home/', views.user_home, name='home'),
    url(r'^stockMarket', views.stockMarket, name='stockMarket'),
    url(r'^forecasting/', views.forecasting, name='getCompanyName'),
]
```

VIEWS

PREDICTION PAGE (HTML)

```
{% extends "StockMarketApp/menu.html" %}

{% block body_block %}
```

```
<style type="text/css">
```

```
.button {
position: relative;
display: block;
width: 200px;
height: 36px;
margin-top:30px;
border-radius: 18px;
background-color: #1c89ff;
border: solid 1px transparent;
color: #fff;
font-size: 18px;
font-weight: 300;
cursor: pointer;
transition: all .1s ease-in-out;
}

.button:hover {
background-color: #135dc4;
```

```
border-color: #90c60f;  
border-width: 2px 2px 2px 2px;  
transition: all .1s ease-in-out;  
}
```

```
.loader {  
display: flex;  
justify-content: center;  
align-items: center;  
width: 50px;  
height: 50px;  
background: transparent;  
margin: 30px auto 0 auto;  
border: solid 2px #424242;  
border-top: solid 2px #1c89ff;  
border-radius: 50%;  
opacity: 0;  
}
```

```
.check {  
width: 100%;  
height: 100%;  
display: flex;  
flex-direction: column;  
justify-content: center;  
align-items: center;  
transform: translate3d(-4px,50px,0);  
opacity: 0;  
span:nth-child(1) {  
display: block;  
width: 10px;  
height: 2px;  

```

```
background-color: #fff;
transform: rotate(45deg);
}
span:nth-child(2) {
display: block;
width: 20px;
height: 2px;
background-color: #fff;
transform: rotate(-45deg) translate3d(14px, -4px, 0);
transform-origin: 100%;
}
}
```

```
.loader.active {
animation: loading 2s ease-in-out;
animation-fill-mode: forwards;
}
```

```
.check.active {
opacity: 1;
transform: translate3d(-4px,4px,0);
transition: all .5s cubic-bezier(.49, 1.74, .38, 1.74);
transition-delay: .2s;
}
```

```
@keyframes loading {
30% {
opacity:1;
}
```

```
85% {
opacity:1;
```



```

transform: rotate(1080deg);
border-color: #262626;
}
100% {
opacity: 1;
transform: rotate(1080deg);
border-color: #1c89ff;
}
}
</style>

<div class="container">
  <div class="jumbotron">
    <h4>Enter Company Name</h4>

    { % csrf_token % }

    <div class="content">
      <div class="content_resize">
        <form action="{ % url 'StockMarketApp:getCompanyName' % }" method="post" >
          <br>
          <input id="name" style="padding:5px" name="companyName" required
placeholder="Enter Company Name" class="text" />
          { % csrf_token % }
          <button class="button">Submit</button>

        </form>

        <div class="clr"></div>
      </div>
    </div>

  </div>

```

</div>

{% endblock %}

RESULT PAGE (HTML)

<style>

```
.center {  
    display: block;  
    margin-left: auto;  
    margin-right: auto;  
    width: 100%;  
}
```

```
.button {  
    border-radius: 4px;  
    background-color: #f4511e;  
    border: none;  
    color: #FFFFFF;  
    text-align: center;  
    font-size: 28px;  
    padding: 20px;  
    width: 200px;  
    transition: all 0.5s;  
    cursor: pointer;  
    margin: 5px;  
}
```

```
.button span {  
    cursor: pointer;  
    display: inline-block;  
    position: relative;  
    transition: 0.5s;
```

```
}
```

```
.button span:after {
  content: '\00bb';
  position: absolute;
  opacity: 0;
  top: 0;
  right: -20px;
  transition: 0.5s;
}
```

```
.button:hover span {
  padding-right: 25px;
}
```

```
.button:hover span:after {
  opacity: 1;
  right: 0;
}
```

```
</style>
```

```
<div class="content">
```

```
<div class="content_resize">
```

```
<h2>Predicted Result</h2>
```

```
<img src= "../static/img/predicted_result.png" alt="predicted_result" class="center">
```

```
</div>
```

```
</div>
```

```
{% endblock %}
```

LOGIN PAGE(HTML)

```
{% extends "StockMarketApp/base.html" %}
```

```
{% block body_block %}
```

```
<div class="container">
```

```
  <div class="jumbotron">
```

```
    <h2 class="text-center text-uppercase">User Login</h2>
```

```
  <div class="login-form">
```

```
    <div class="main-div">
```

```
      <div class="panel">
```

```
        <p>Please enter your Username and password</p>
```

```
      </div>
```

```
    <form action="{% url 'StockMarketApp:user_login' %}" method="post">
```

```
      {% csrf_token %}
```

```
      {# A more "HTML" way of creating the login form#}
```

```
    <div class="form-group">
```

```
      <input type="text" class="form-control" id="inputEmail" name="username"
placeholder="User Name">
```

```
    </div>
```

```
    <div class="form-group">
```

```
      <input type="password" class="form-control" id="inputPassword" name="password"
placeholder="Password">
```

```
    </div>
```

```
    <button type="submit" class="btn btn-primary">Login</button>
```

```
  </form>
```

```
</div>
```

```
</div>
```

```
</div>
```

</div>

{% endblock % }

Performance Analysis, Testing and Results

8.1 Testing

The testing is done module wise to check the if the correct results are being obtained. And finally an integration test is performed to check if the entire system is performing as necessary.

8.2 Results

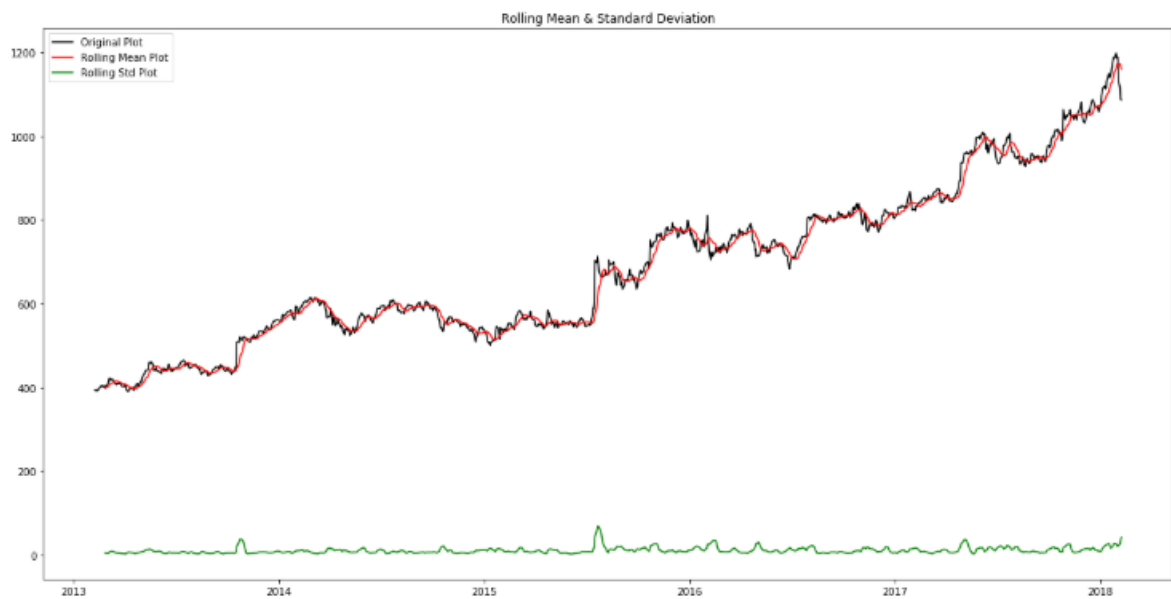


Figure 7. Calculation of Rolling Mean and Standard Deviation

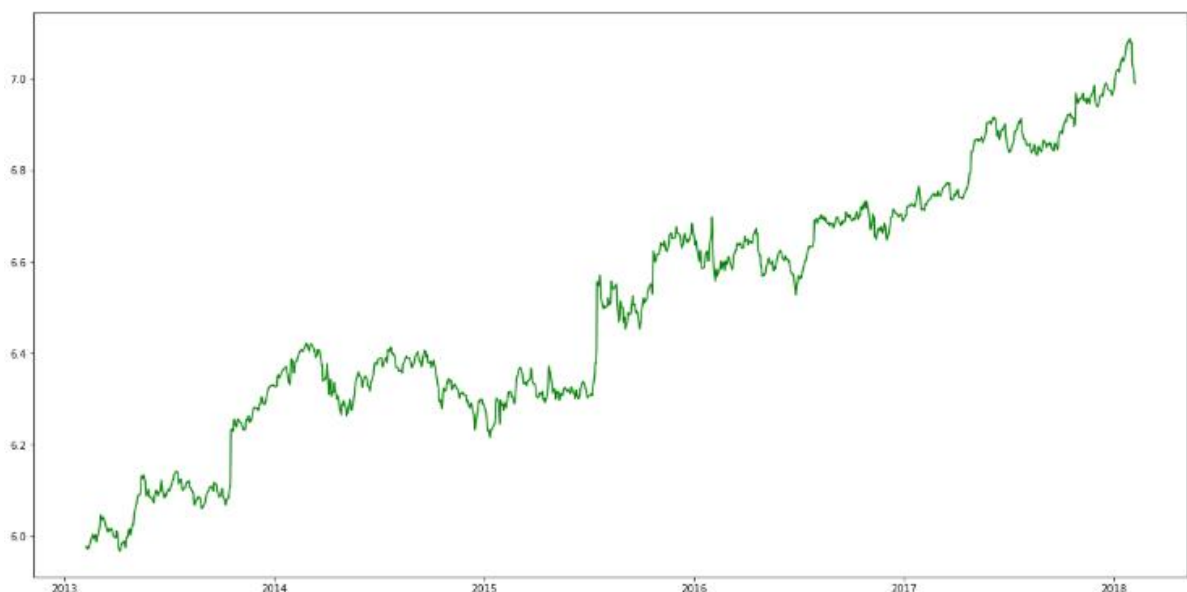


Figure 8. Closing Price Log Scale

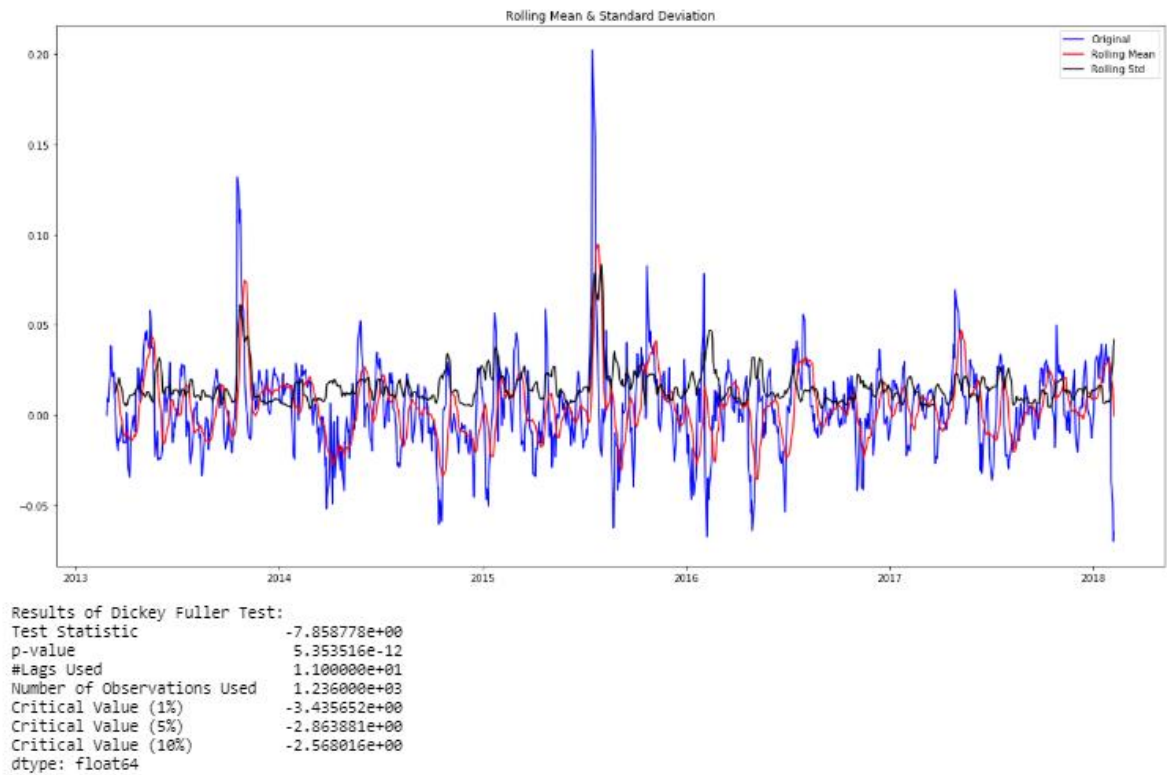


Figure 9. Dicky Fuller Test Stationarity testing

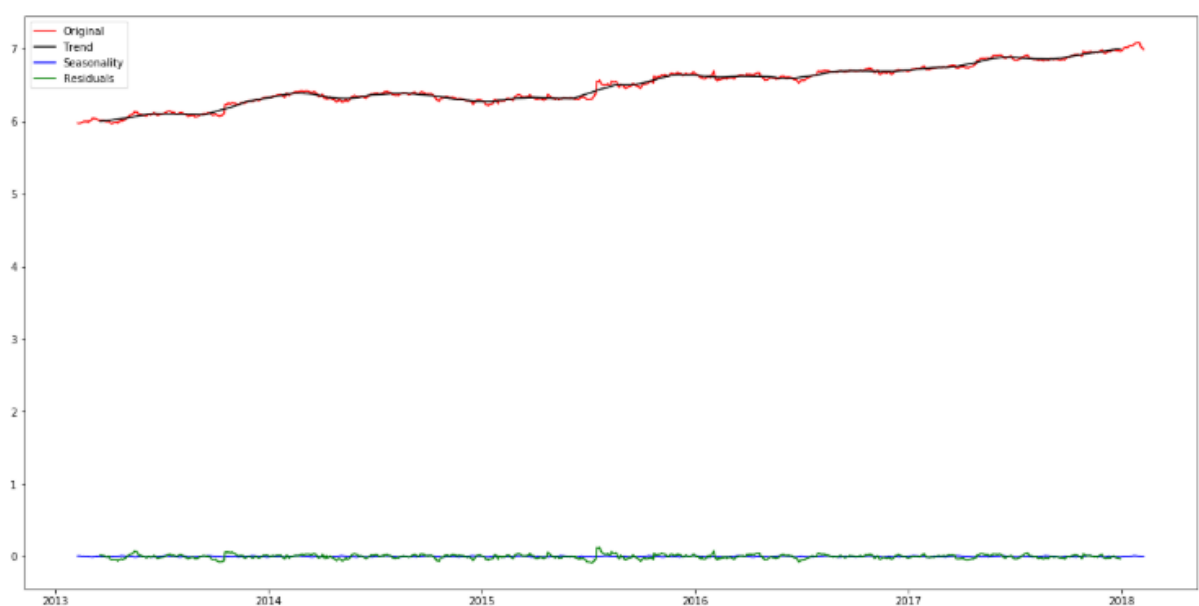
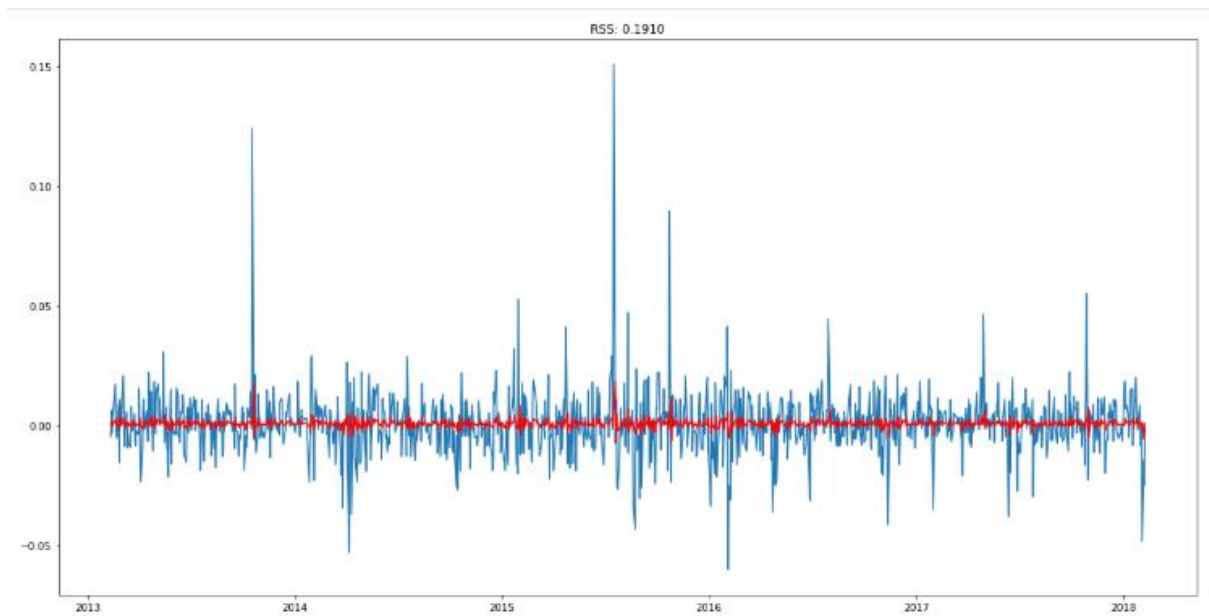


Figure 10. Checking the output data of Dicky Fuller Test for Seasonality
Trend and Residuals.



Plotting ARIMA model

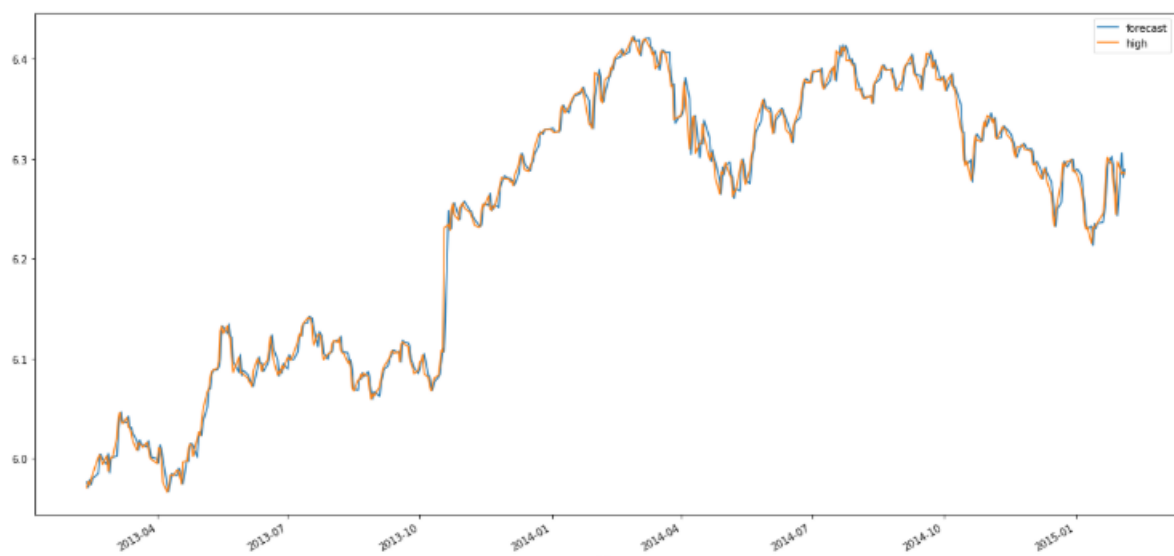


Figure 11. Predicted Change in the Closing Price with Time.

date	
2013-02-08	5.975661
2013-02-11	5.970985
2013-02-12	5.977224
2013-02-13	5.973982
2013-02-14	5.978289

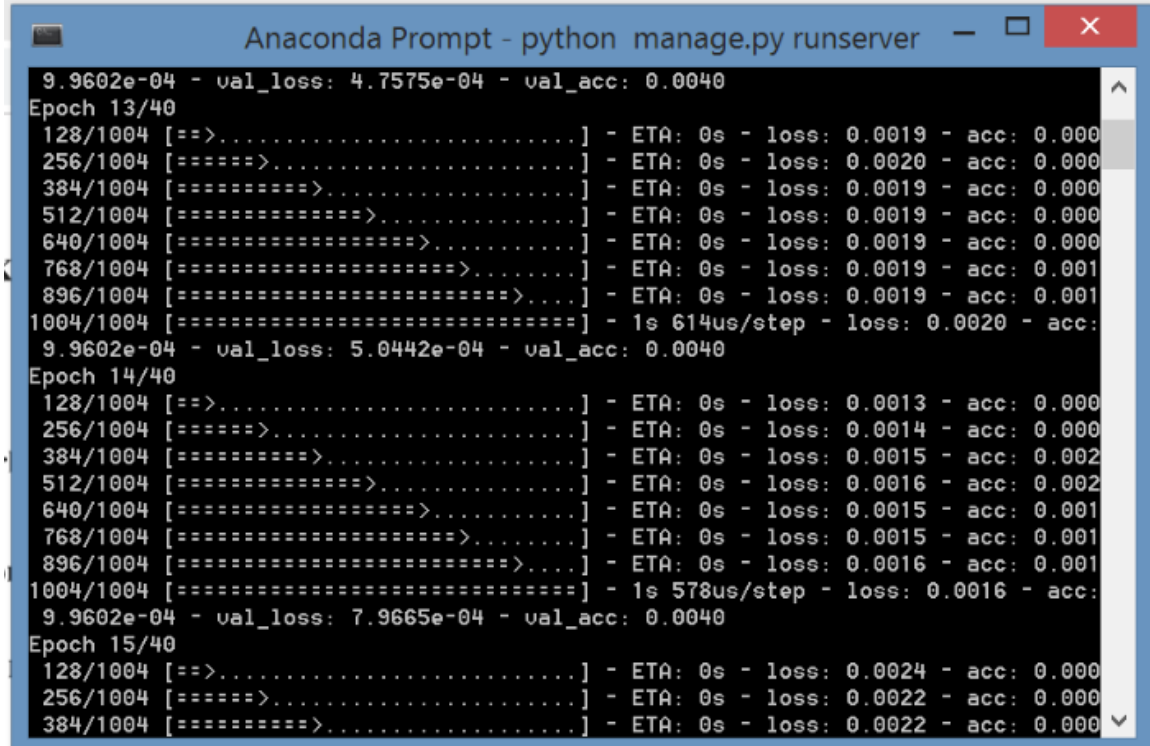
Figure 12. Previous day closing price

```
print(next_day_forecast)
```

```
[6.99266676]
```

Next Day Forecast

LSTM



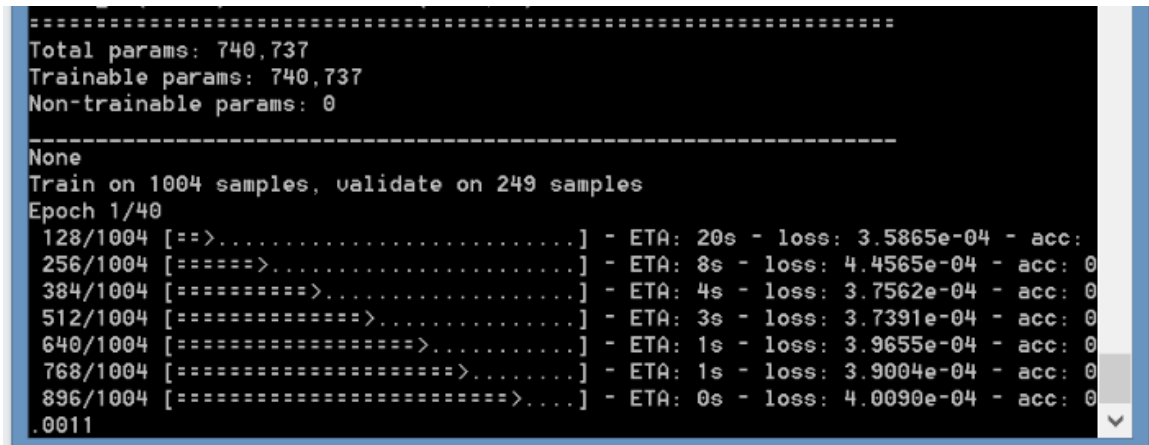
```

Anaconda Prompt - python manage.py runserver
9.9602e-04 - val_loss: 4.7575e-04 - val_acc: 0.0040
Epoch 13/40
128/1004 [==>.....] - ETA: 0s - loss: 0.0019 - acc: 0.000
256/1004 [=====>.....] - ETA: 0s - loss: 0.0020 - acc: 0.000
384/1004 [=====>.....] - ETA: 0s - loss: 0.0019 - acc: 0.000
512/1004 [=====>.....] - ETA: 0s - loss: 0.0019 - acc: 0.000
640/1004 [=====>.....] - ETA: 0s - loss: 0.0019 - acc: 0.000
768/1004 [=====>.....] - ETA: 0s - loss: 0.0019 - acc: 0.001
896/1004 [=====>.....] - ETA: 0s - loss: 0.0019 - acc: 0.001
1004/1004 [=====] - 1s 614us/step - loss: 0.0020 - acc:
9.9602e-04 - val_loss: 5.0442e-04 - val_acc: 0.0040
Epoch 14/40
128/1004 [==>.....] - ETA: 0s - loss: 0.0013 - acc: 0.000
256/1004 [=====>.....] - ETA: 0s - loss: 0.0014 - acc: 0.000
384/1004 [=====>.....] - ETA: 0s - loss: 0.0015 - acc: 0.002
512/1004 [=====>.....] - ETA: 0s - loss: 0.0016 - acc: 0.002
640/1004 [=====>.....] - ETA: 0s - loss: 0.0015 - acc: 0.001
768/1004 [=====>.....] - ETA: 0s - loss: 0.0015 - acc: 0.001
896/1004 [=====>.....] - ETA: 0s - loss: 0.0016 - acc: 0.001
1004/1004 [=====] - 1s 578us/step - loss: 0.0016 - acc:
9.9602e-04 - val_loss: 7.9665e-04 - val_acc: 0.0040
Epoch 15/40
128/1004 [==>.....] - ETA: 0s - loss: 0.0024 - acc: 0.000
256/1004 [=====>.....] - ETA: 0s - loss: 0.0022 - acc: 0.000
384/1004 [=====>.....] - ETA: 0s - loss: 0.0022 - acc: 0.000

```

Figure 13. LSTM Training

The above figure shows the training of the LSTM network which is undergoing 14th and 15th iteration and shows the variation in accuracy.



```

=====
Total params: 740,737
Trainable params: 740,737
Non-trainable params: 0
-----
None
Train on 1004 samples, validate on 249 samples
Epoch 1/40
128/1004 [==>.....] - ETA: 20s - loss: 3.5865e-04 - acc:
256/1004 [=====>.....] - ETA: 8s - loss: 4.4565e-04 - acc: 0
384/1004 [=====>.....] - ETA: 4s - loss: 3.7562e-04 - acc: 0
512/1004 [=====>.....] - ETA: 3s - loss: 3.7391e-04 - acc: 0
640/1004 [=====>.....] - ETA: 1s - loss: 3.9655e-04 - acc: 0
768/1004 [=====>.....] - ETA: 1s - loss: 3.9004e-04 - acc: 0
896/1004 [=====>.....] - ETA: 0s - loss: 4.0090e-04 - acc: 0
.0011

```

Figure 14. LSTM Train Test split

The above figure shows the parameters trained and the split taken by the pre network automatically.

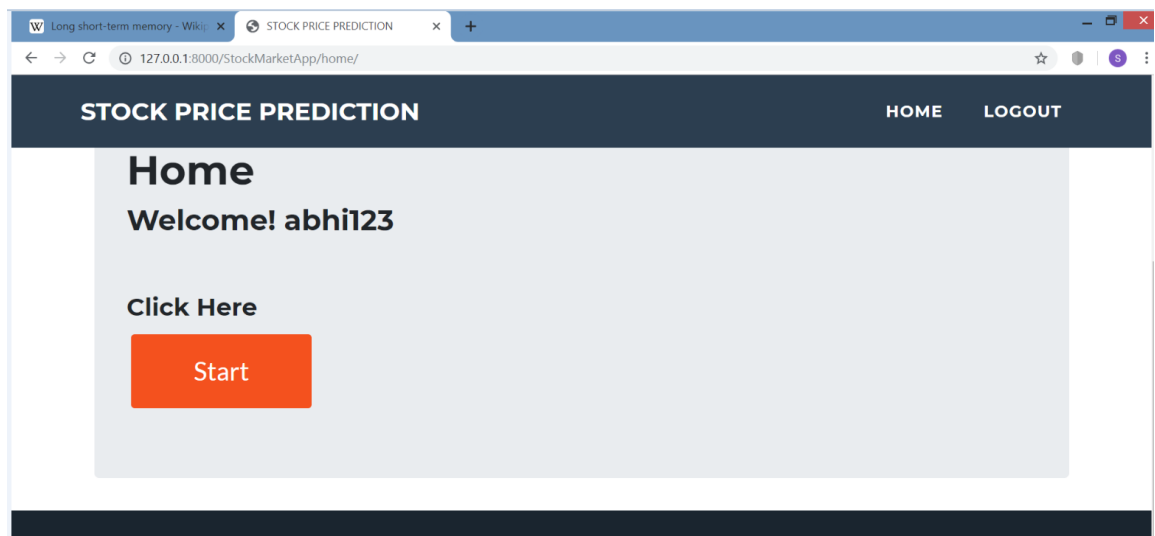


Figure 15. User interface homepage

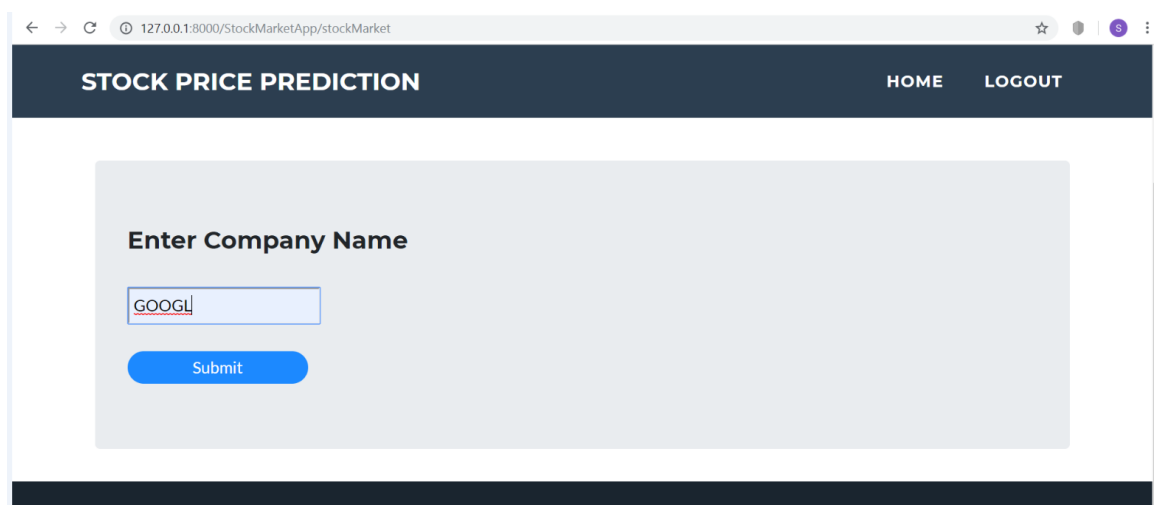


Figure 15. Choosing the company whose stock the user is interested.

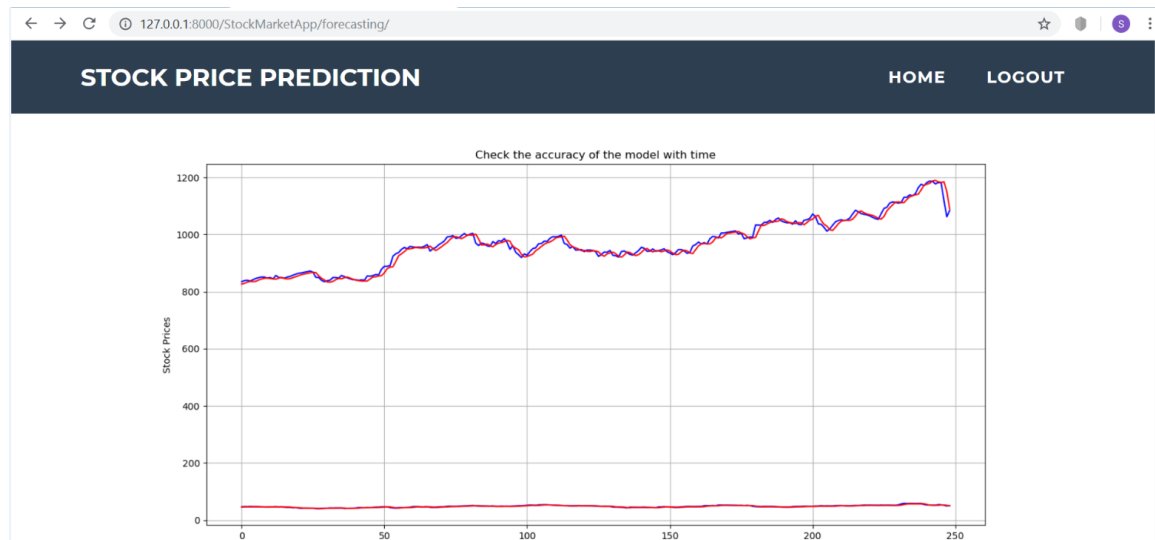


Figure 16. Original v/s predicted closing price graph

FUTURE WORK

Using neural networks to forecast stock market prices will be a continuing area of research as researchers and investors strive to outperform the market, with the ultimate goal of bettering their returns. It is unlikely that new theoretical ideas will come out of this applied work. However, interesting results and validation of theories will occur as neural networks are applied to more complicated problems.

Financial neural networks must be trained to learn the data and generalize, while being prevented from overtraining and memorizing the data. Also, due to their large number of inputs, network pruning is important to remove redundant input nodes and speed-up training and recall. The major research thrust in this area should be determining better network architectures. The commonly used back propagation network offers good performance, but this performance could be improved by using recurrence or reusing past inputs and outputs. The architecture combining neural networks and expert systems shows potential. Currently, implemented neural networks have shown that the Efficient Market Hypothesis does not hold in practice, and that stock markets are probably chaotic systems. Until we more fully understand the dynamics behind such chaotic systems, the best we can hope for is to model them as accurately as possible.

CONCLUSION

Predicting the direction of movements of the stock market index is important for the development of effective market trading strategies. It usually affects a financial trader's decision to buy or sell an instrument. Successful prediction of stock prices may promise attractive benefits for investors. These tasks are highly complicated and very difficult. We can use the principle of artificial neural networks to correctly predict the future value of a particular stock. This is done in steps, first we collect the past data of a stock. Then we create a neural network & train the network on the basis of the past data. Generally 80-90% of the data is taken to train the model. Then we use our model to predict the value of the stock & compare the value with the pre obtained data set. Although neural networks are not perfect in their prediction, they outperform all other methods and provide hope that one day we can more fully understand dynamic, chaotic systems such as the stock market.

BIBLIOGRAPHY

- [1] Tom M. Mitchell., 2013, “Machine Learning”, McGraw Hill Education.
- [2] Ayodele A. Adebisi, Aderemi O. Adewumi and Charles K. Ayo., “Stock Price Prediction Using the ARIMA Model”., 2014.
- [3] Peihao Li, Chaoqun Jing, Tian Liang, Mingjia Liu, Zhenglin Chen, Li Guo., “Autoregressive Moving Average Modeling in the Financial Sector”., 2015.
- [4] ”. MehakUsmani,Syrd Hasan Adil,Kamranraza and Syed SaadAzhar Ali., “Stock Market Predictions Using Machine Learning Techniques”., 2016.
- [5] Shashank Tiwari, AkshayBharadwaj and Dr.Sudha Gupta., “Stock Price Prediction Using Data Analytics”., 2017.
- [6] Tian Ye., “Stock Forecasting Method Based on Wavelet Analysis and ARIMA-SVR Model”., 2017.
- [7] EthemAlpaydın, “Introduction to machine learning”, second edition, MIT press.2017