LEVERAGING BIG DATA ANALYTICS FOR STOCK MARKET ANALYSIS AND PREDICTIVE INSIGHTS

[1]Sanjana Alugani, [2]A Sahil Madan
VNRVJIET
[1]asanjana1502@gmail.com, [2]sahilmadan0805@gmail.com

ABSTRACT: Big data analytics are used primarily in various sectors for accurate prediction and analysis of the large data sets. They allow the discovery of significant information from large data sets, otherwise, it is hidden. In this paper, an approach of robust Cloudera-Hadoop based data pipeline is proposed to perform analyses for any scale and type of data, in which selected US stocks are analysed to predict daily gains based on real time data from Yahoo Finance. The Apache Hadoop big-data framework is provided to handle large data sets through distributed storage and processing, stocks from the US stock market are picked and their daily gain data are divided into training and test data set to predict the stocks with high daily gains using Machine Learning module of Spark.

Keywords – K-means and recurrent neural networks (RNN).

1. INTRODUCTION

Big data has been attached great importance for the proliferation of a lot of different sectors. It has been extensively employed by business organizations to formalize important business insights and intelligence. Furthermore, it has been utilized by healthcare sector to discover important patterns and knowledge so as to improve the modern healthcare systems. Besides, big data holds significant importance for the information, technology and cloud computing sector. Recently, the finance and banking sectors utilized big data to track the financial market activity. Big data analytics and network analytics were used to catch illegal trading in the financial markets. Similarly, traders, big banks, financial institutions and companies utilized big data for generating trade analytics utilized in high frequency trading. Besides, big data analytics also helped in the detection of illegal activities such as: money laundering and financial frauds. In this paper, we hope to build a system which analyses US oil stocks to predict daily gains in US stocks based on the real time data from Yahoo Finance. About all 13 stocks in US oil fund are picked up and their daily gain data are divided into training and test data set to predict the stocks with high daily gains using Machine Learning module of Spark. Based on our analysis we propose a robust ClouderaHadoop based data pipeline to perform this analyses for any type and scale of data. By means of studying a live stream data of US oil stock prices so that it can help us better understand how does US Oil index affects the stock price of other stocks in US oil funds exchange. Besides it can help us predict the profitable stocks for stock traders and provide profits to US Oil stocks trader community.

The goal is to harness the capabilities of big data by developing robust machine learning models, specifically ARIMA and LSTM algorithms. These models are intended to analyze vast quantities of data, with the overarching objective of predicting stock movements. Emphasizing the significance of big data processing, the project aims to uncover meaningful insights for stock traders and facilitate data-driven decision-making. The objective is to implement PYSPARK, a Python API for Apache Spark, with the goal of efficiently processing large volumes of stock data. Acknowledging the streaming nature of this data, the project aims to ensure its capacity to handle substantial datasets and maintain responsiveness to real-time changes in stock prices. The project's objective is to apply the XGBOOST algorithm to refine and enhance the relevance of features within the stock data. By systematically removing irrelevant variables, the goal is to improve the accuracy of predictions made by the machine learning models. This contributes to more reliable outcomes, aligning with the overarching goal of providing valuable insights for stock traders. The project aims to assess the performance of the ARIMA and LSTM algorithms, utilizing the R SQUARED metric. This objective provides a quantitative measure of how well the models predict stock movements. With a higher R SQUARED value signifying a more effective predictive capability, the goal is to aid in the selection of the most suitable algorithm for comprehensive stock analysis.

This project is to develop stock analysis prediction system using bigdata analytics. Illicit trading in the financial markets was detected through the use of network analytics and big data analytics. In a similar vein, traders, large banks, corporations, and financial organizations used big data to produce trade analytics used in high frequency trading.

2. LITERATURE REVIEW

Price Trend Prediction of Stock Market Using Outlier Data Mining Algorithm:

In this paper we present a novel data miming approach to predict long term behavior of stock trend. Traditional techniques on stock trend prediction have shown their limitations when using time series algorithms or volatility modelling on price sequence. In our research, a novel outlier mining algorithm is proposed to detect anomalies on the basis of volume sequence of high frequency tick-by tick data of stock market. Such anomaly trades always inference with the stock price in the stock market. By using the cluster information of such anomalies, our approach predict the stock trend effectively in the really world market. Experiment results show that our proposed approach makes profits on the Chinese stock market, especially in a long-term usage.

Stock price prediction using data analytics:

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. Nifty 50 (A NSEI) stock indices is considered as a data input for methods which are implemented. 9 years of data is used. The accuracy was calculated using 2-3 years of forecast results of R and 2 months of forecast results of Python after comparing with the actual price of the stocks. Mean squared error and other error parameters for every prediction system were calculated and it is found that feed forward network only produces 1.81598342% error when opening price of stock is forecasted using it.

Stock market: Statistical analysis of its indexes and its constituents:

The ever-changing realm of the stock market is constantly thriving under the process of modifications and alterations. Thus, making a profit from it is hard and requires intensive planning. It is in the context of this fact that makes Stock Market analysis the first and foremost priority for any financial investment. Considering the behavioural aspects of stock prices which have a tendency to rise and fall unexpectedly, leads to a volatile scenario. However, to acquire some insight, intellectual wit and smartness to extract the best, a thorough and consistent analysis is most popular and tested way. This paper aims to determine top high performing stocks having good returns under given index that would be most safe and beneficial for investment. Using historical data we were able to obtain top stocks that are advisable for investment. We also verified our results by analyzing contemporary data similarly and found out that the performance and returns of these stocks were still high irrespective of volatility.

"Stocks Analysis and Prediction Using Big Data Analytics:

Big data is a new and emerging buss word in today's times. Stock market is an up and ever evolving, volatile, uncertain and intriguingly potential niche, which is an important extension in finance and business growth and prediction. Stock market has to deal with a large amount of vast and distinct data to function and draw meaningful conclusions. Stock market trends depend broadly on two analyses; technical and fundamental. Technical analysis is carried out using historical trends and market values. On the other hand, fundamental analysis is done based on the sentiments, values and social media data and responses. Since large, complex and complicated and exponentially growing data is involved, we use big data analysis to help assist in the prediction and drawing accurate business decisions and profitable investments.

Stock market prediction: A big data approach:

The Stock market process is full of uncertainty and is affected by many factors. Hence the Stock market prediction is one of the important exertions in finance and business. There are two types of analysis possible for prediction, technical and fundamental. In this paper both technical and fundamental analysis are considered. Technical analysis is done using historical data of stock prices by applying machine learning and fundamental analysis is done using social media data by applying sentiment analysis. Social media data has high impact today than ever, it can aide in predicting the trend of the stock market. The method involves collecting news and social media data and extracting sentiments expressed by individual. Then the correlation between the sentiments and the stock values is analyzed. The learned model can then be used to make future predictions about stock values. It can be shown that this method is able to predict the sentiment and the stock performance and its recent news and social data are also closely correlated.

3. METHODOLOGY

Recently, the finance and banking sectors utilized big data to track the financial market activity. Big data analytics and network analytics were used to catch illegal trading in the financial markets. Similarly, traders, big banks, financial institutions and companies utilized big data for generating trade analytics utilized in high frequency trading. Besides, big data analytics also helped in the detection of illegal activities such as: money laundering and financial frauds.

Disadvantages of existing system:

1. In existing systems there is no accurate prediction in data
2. and also existing systems unable to analysis of the large data sets

Proposed System:

In this paper, we hope to build a system which analyses US oil stocks to predict daily gains in US stocks based on the real time data from Yahoo Finance. About all 13 stocks in US oil fund are picked up and their daily gain data are divided into training and test data set to predict the stocks with high daily gains using Machine Learning module of Spark. Based on our analysis we propose a robust ClouderaHadoop based data pipeline to perform this analyses for any type and scale of data.

Advantages of proposed system:

1. It support all sort of complex analysis
2. faster
3. With lot of ML tools available, deciding the tool that can perform analysis and implement ML algorithms efficiently has been a daunting task.
4. Provides a flexible platform for implementing

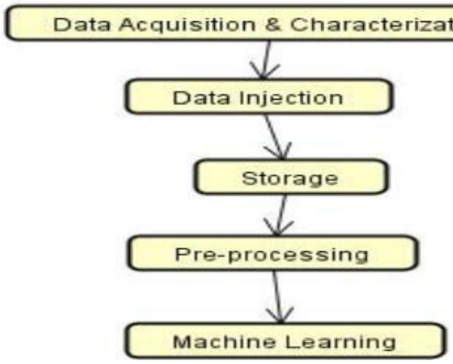


Fig.2: System architecture

MODULES:

1. Read Data using PySpark:

- Read stock data from a dataset file using PySpark.
- Initialize Spark and set up a Spark Streaming Context.
- Create a Spark session.
- Read the dataset as a stream or in batches using PySpark classes.
- Display the dataset to ensure it has been loaded correctly.

2. Data Normalization:

- Normalize dataset values to ensure consistency in scale.
- Apply data normalization techniques to scale the values of the dataset.
- Normalization is crucial when dealing with features that may have different units or ranges, ensuring that the algorithms can learn effectively from the data.

3. Feature Engineering With XGBoost:

- Apply XGBoost algorithm to perform feature engineering and select relevant features. while excluding irrelevant ones.
- XGBoost can help enhance the model's ability to make accurate predictions by focusing on the most impactful features.

4. Training ARIMA Model:

- Train the ARIMA model on the preprocessed dataset.
- Use the preprocessed dataset to train an ARIMA model, specifying the order of the model (p, d, q).
- The ARIMA model is a time series model that captures temporal dependencies in the data.

5. Evaluate ARIMA Model:

- Evaluate the performance of the ARIMA model.
- Apply the trained ARIMA model to the test dataset.
- Evaluate the model's performance using metric R-squared.
- R-squared measures how well the model explains the variance in the test data.

6. Training LSTM Model:

- Train the LSTM model on the preprocessed dataset.
- Set up a Sequential model using Keras with LSTM layers.
- Train the LSTM model on the preprocessed dataset.

- LSTM models are effective for capturing long-term dependencies in sequential data.

7. Evaluate LSTM Model:

- Evaluate the performance of the LSTM model.
- Apply the trained LSTM model to the test dataset.
- Evaluate the model's performance using metrics like R-squared.
- Compare the R-squared value with the ARIMA model to determine which model performs better.

4. IMPLEMENTATION

ALGORITHMS:

LSTM (Long Short-Term Memory):

Definition: LSTM is a type of recurrent neural network (RNN) architecture designed to overcome the vanishing gradient problem in traditional RNNs. It excels at capturing and learning patterns in sequential data over extended time periods by maintaining a cell state that can be selectively updated, allowing for better retention of long-term dependencies.

Why Used in the Project: LSTM is employed in the project for its capability to analyze and learn patterns in time-series data, which is crucial for predicting stock movements. Its ability to capture long-term dependencies in the sequential nature of stock prices makes it well-suited for forecasting stock

ARIMA (AutoRegressive Integrated Moving Average):

Definition: ARIMA is a statistical method used for time-series analysis and forecasting. It consists of three main components: AutoRegressive (AR), Integrated (I), and Moving Average (MA). ARIMA models are effective in capturing trends and seasonality within time-series data.

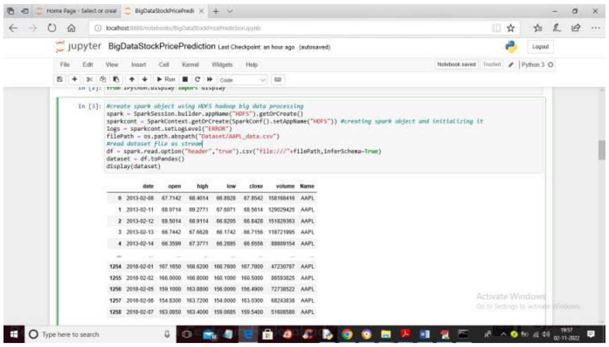
Why Used in the Project: ARIMA is applied in the project due to its effectiveness in modeling time-series data, making it suitable for predicting stock price movements over time. Its ability to account for trends and seasonality in the data complements the project's goal of forecasting daily gains or losses in the stock market.

XGBOOST (Extreme Gradient Boosting):

Definition: XGBOOST is a powerful machine learning algorithm based on gradient boosting frameworks. It builds a series of decision trees and combines their predictions to produce a final output. XGBOOST is known for its efficiency, speed, and high performance in various data science applications.

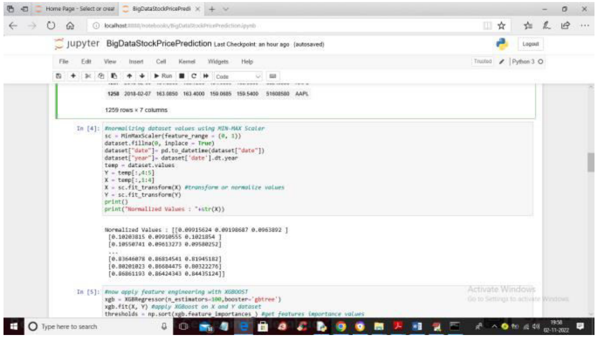
Why Used in the Project: XGBOOST is utilized in the project for feature refinement within the stock data. Its capability to handle large datasets and prioritize the most influential features makes it suitable for improving the accuracy of predictions. By removing irrelevant variables, XGBOOST contributes to enhancing the overall performance of the machine learning models in the project.

5. EXPERIMENTAL RESULTS



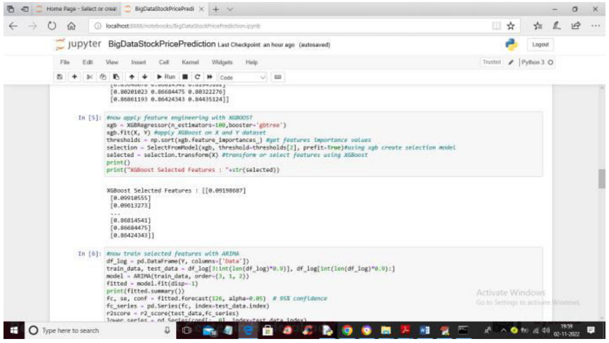
```
In [2]: from IPython.display import display

In [3]: #create spark object using HDFS hadoop big data processing
        spark = SparkSession.builder.appName("HDFS").getOrCreate()
        sparkcont = SparkContext.getOrCreate(SparkConf().setAppName("HDFS")) #creating spark object and initializing it
        logs = sparkcont.setLogLevel("ERROR")
        filePath = os.path.abspath("Dataset/AAPL_data.csv")
        #read dataset file as stream
        df = spark.read.option("header","true").csv("file:///"+filePath,inferSchema=True)
        dataset = df.toPandas()
        display(dataset)

              date     open      high      low     close     volume  Name
   0  2013-02-08   67.7142   68.4014   66.8928   67.8542  158168416  AAPL
   1  2013-02-11   68.0714   69.2771   67.6071   68.5614  129029425  AAPL
   2  2013-02-12   68.5014   68.9114   66.8205   66.8428  151829363  AAPL
   3  2013-02-13   66.7442   67.6628   66.1742   66.7156  118721995  AAPL
   4  2013-02-14   66.3599   67.3771   66.2885   66.6556   88809154  AAPL
 ...         ...       ...       ...       ...       ...        ...   ...
1254  2018-02-01  167.1650  168.6200  166.7600  167.7800   47230787  AAPL
1255  2018-02-02  166.0000  166.8000  160.1000  160.5000   86593825  AAPL
1256  2018-02-05  159.1000  163.8800  156.0000  156.4900   72738522  AAPL
1257  2018-02-06  154.8300  163.7200  154.0000  163.0300   68243838  AAPL
1258  2018-02-07  163.0850  163.4000  159.0685  159.5400   51608580  AAPL
```

In above screen using SPARK classes we are reading dataset as stream, here we don't have any online streaming so we are reading data from file and then displaying



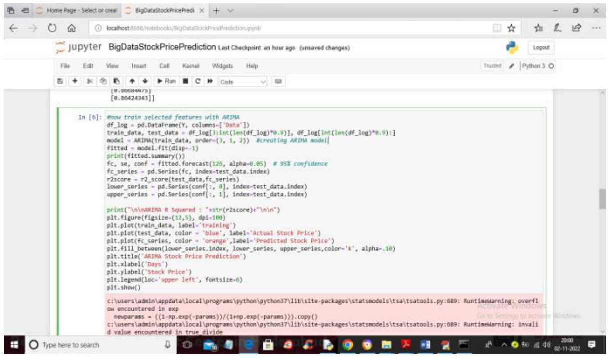
```
1258  2018-02-07  163.0850  163.4000  159.0685  159.5400   51608580  AAPL

1259 rows × 7 columns

In [4]: #normalizing dataset values using MIN-MAX Scaler
        sc = MinMaxScaler(feature_range = (0, 1))
        dataset.fillna(0, inplace = True)
        dataset["date"]= pd.to_datetime(dataset["date"])
        dataset["year"]= dataset['date'].dt.year
        temp = dataset.values
        Y = temp[:,4:5]
        X = temp[:,1:4]
        X = sc.fit_transform(X) #transform or normalize values
        Y = sc.fit_transform(Y)
        print()
        print("Normalized Values : "+str(X))

        Normalized Values : [[0.09915624 0.09198687 0.0963892 ]
         [0.10203815 0.09910555 0.1021854 ]
         [0.10550741 0.09613273 0.09580252]
         ...
         [0.83646078 0.86814541 0.81945182]
         [0.80201023 0.86684475 0.80322276]
         [0.86861193 0.86424343 0.84435124]]

In [5]: #now apply feature engineering with XGBOOST
        xgb = XGBRegressor(n_estimators=100,booster='gbtree')
        xgb.fit(X, Y) #apply XGBoost on X and Y dataset
        thresholds = np.sort(xgb.feature_importances_) #get features importance values
```

Using above screen code we are normalizing dataset values



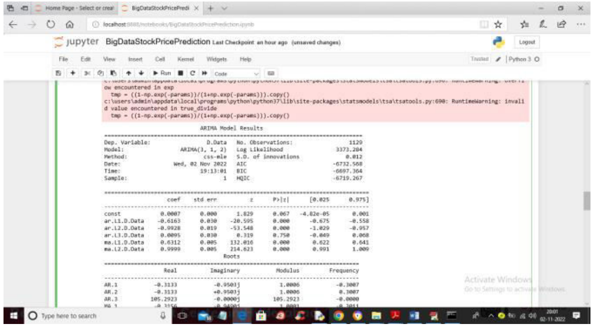
```
         [0.80201023 0.86684475 0.80322276]
         [0.86861193 0.86424343 0.84435124]]

In [5]: #now apply feature engineering with XGBOOST
        xgb = XGBRegressor(n_estimators=100,booster='gbtree')
        xgb.fit(X, Y) #apply XGBoost on X and Y dataset
        thresholds = np.sort(xgb.feature_importances_) #get features importance values
        selection = SelectFromModel(xgb, threshold=thresholds[2], prefit=True)#using xgb create selection model
        selected = selection.transform(X) #transform or select features using XGBoost
        print()
        print("XGBoost Selected Features : "+str(selected))

        XGBoost Selected Features : [[0.09198687]
         [0.09910555]
         [0.09613273]
         ...
         [0.86814541]
         [0.86684475]
         [0.86424343]]

In [6]: #now train selected features with ARIMA
        df_log = pd.DataFrame(Y, columns=['Data'])
        train_data, test_data = df_log[3:int(len(df_log)*0.9)], df_log[int(len(df_log)*0.9):]
        model = ARIMA(train_data, order=(3, 1, 2))
        fitted = model.fit(disp=-1)
        print(fitted.summary())
        fc, se, conf = fitted.forecast(126, alpha=0.05)  # 95% confidence
        fc_series = pd.Series(fc, index=test_data.index)
        r2score = r2_score(test_data,fc_series)
```

In above screen we are applying XGBOOST algorithm to apply features engineering and then displaying selected values from dataset. Now processed data will be input to both algorithm



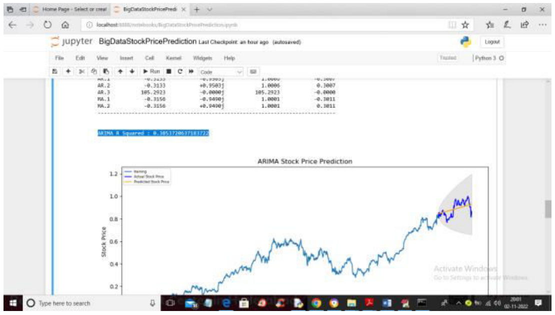
```
         [0.86424343]]

In [6]: #now train selected features with ARIMA
        df_log = pd.DataFrame(Y, columns=['Data'])
        train_data, test_data = df_log[3:int(len(df_log)*0.9)], df_log[int(len(df_log)*0.9):]
        model = ARIMA(train_data, order=(3, 1, 2))  #creating ARIMA model
        fitted = model.fit(disp=-1)
        print(fitted.summary())
        fc, se, conf = fitted.forecast(126, alpha=0.05)  # 95% confidence
        fc_series = pd.Series(fc, index=test_data.index)
        r2score = r2_score(test_data,fc_series)
        lower_series = pd.Series(conf[:, 0], index=test_data.index)
        upper_series = pd.Series(conf[:, 1], index=test_data.index)

        print("\n\nARIMA R Squared : "+str(r2score)+"\n\n")
        plt.figure(figsize=(12,5), dpi=100)
        plt.plot(train_data, label='training')
        plt.plot(test_data, color = 'blue', label='Actual Stock Price')
        plt.plot(fc_series, color = 'orange',label='Predicted Stock Price')
        plt.fill_between(lower_series.index, lower_series, upper_series,color='k', alpha=.10)
        plt.title('ARIMA Stock Price Prediction')
        plt.xlabel('Days')
        plt.ylabel('Stock Price')
        plt.legend(loc='upper left', fontsize=6)
        plt.show()

        c:\users\admin\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\tsatools.py:689: RuntimeWarning: overfl
        ow encountered in exp
          newparams = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
        c:\users\admin\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\tsatools.py:689: RuntimeWarning: invali
        d value encountered in true_divide
```

Using above screen code we are training dataset with ARIMA class and below is the ARIMA stock prediction output



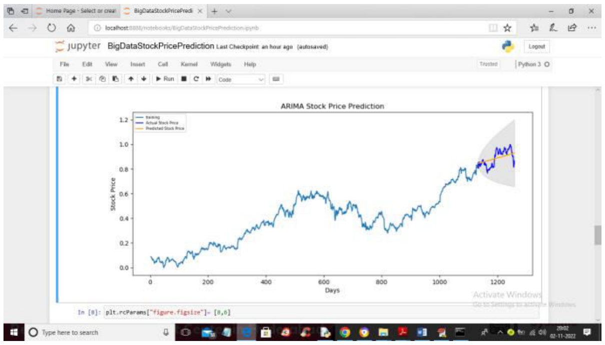
```
ow encountered in exp
  tmp = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
c:\users\admin\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\tsatools.py:690: RuntimeWarning: invali
d value encountered in true_divide
  tmp = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
                             ARIMA Model Results
==============================================================================
Dep. Variable:                 D.Data   No. Observations:                 1129
Model:                 ARIMA(3, 1, 2)   Log Likelihood                3373.284
Method:                       css-mle   S.D. of innovations              0.012
Date:                Wed, 02 Nov 2022   AIC                          -6732.568
Time:                        19:13:01   BIC                          -6697.364
Sample:                             1   HQIC                         -6719.267

================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0007      0.000      1.829      0.067   -4.82e-05       0.001
ar.L1.D.Data    -0.6163      0.030    -20.595      0.000      -0.675      -0.558
ar.L2.D.Data    -0.9928      0.019    -53.548      0.000      -1.029      -0.957
ar.L3.D.Data     0.0095      0.030      0.319      0.750      -0.049       0.068
ma.L1.D.Data     0.6312      0.005    132.016      0.000       0.622       0.641
ma.L2.D.Data     0.9999      0.005    214.623      0.000       0.991       1.009
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.3133           -0.9503j            1.0006           -0.3007
AR.2           -0.3133           +0.9503j            1.0006            0.3007
AR.3          105.2923           -0.0000j          105.2923           -0.0000
```

In above screen ARIMA starts building model and will get below output



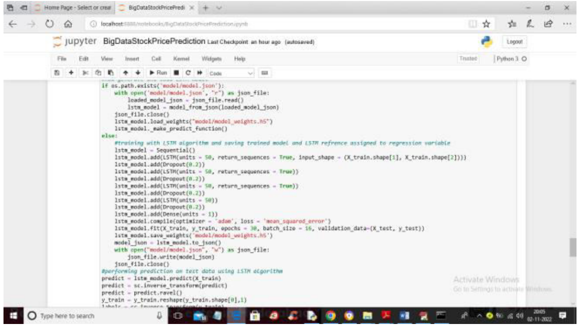
```
AR.2           -0.3133           +0.9503j            1.0006            0.3007
AR.3          105.2923           -0.0000j          105.2923           -0.0000
MA.1           -0.3156           -0.9490j            1.0001           -0.3011
MA.2           -0.3156           +0.9490j            1.0001            0.3011
-----------------------------------------------------------------------------

ARIMA R Squared : 0.3851720637183722
```

In above screen in blue color text we can see ARIMA R SQUARED as 0.38 and below is the graph


jupyter BigDataStockPricePrediction Last Checkpoint: an hour ago (autosaved)
ARIMA Stock Price Prediction
training
Actual Stock Price
Predicted Stock Price
Stock Price
Days
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0
1.2

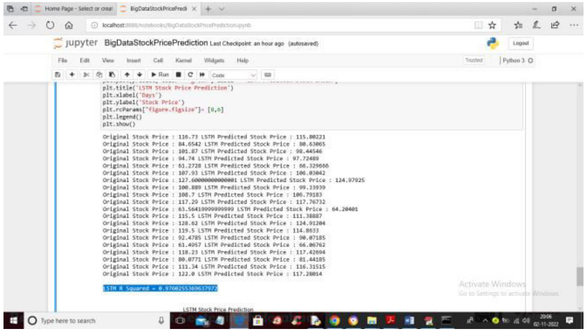
```
In [8]: plt.rcParams["figure.figsize"]= [8,6]
```

In above graph x-axis represents training stock days and y-axis represents stock prices and big light blue line represents training stock prices and dark blue line in the last is the TEST prices and orange line is the predicted prices and orange line is not as zigzag as train and test data so its prediction is not accurate and due to that reason we got its R SQUARED as 0.38%. In below screen we are showing LSTM code


jupyter BigDataStockPricePrediction Last Checkpoint: an hour ago (autosaved)

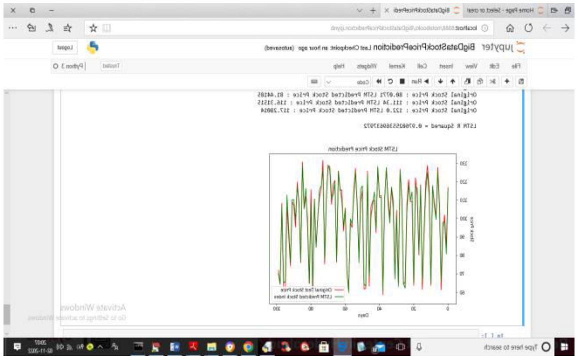
```
if os.path.exists('model/model.json'):
    with open('model/model.json', "r") as json_file:
        loaded_model_json = json_file.read()
        lstm_model = model_from_json(loaded_model_json)
    json_file.close()
    lstm_model.load_weights("model/model_weights.h5")
    lstm_model._make_predict_function()
else:
    #training with LSTM algorithm and saving trained model and LSTM refrence assigned to regression variable
    lstm_model = Sequential()
    lstm_model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], X_train.shape[2])))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(LSTM(units = 50, return_sequences = True))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(LSTM(units = 50, return_sequences = True))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(LSTM(units = 50))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(Dense(units = 1))
    lstm_model.compile(optimizer = 'adam', loss = 'mean_squared_error')
    lstm_model.fit(X_train, y_train, epochs = 30, batch_size = 16, validation_data=(X_test, y_test))
    lstm_model.save_weights('model/model_weights.h5')
    model_json = lstm_model.to_json()
    with open("model/model.json", "w") as json_file:
        json_file.write(model_json)
    json_file.close()
#performing prediction on test data using LSTM algorithm
predict = lstm_model.predict(X_train)
predict = sc.inverse_transform(predict)
predict = predict.ravel()
y_train = y_train.reshape(y_train.shape[0],1)
```

In above screen we are training dataset with LSTM algorithm and below is the prediction output


jupyter BigDataStockPricePrediction Last Checkpoint: an hour ago (autosaved)
LSTM Stock Price Prediction

```
plt.title('LSTM Stock Price Prediction')
plt.xlabel('Days')
plt.ylabel('Stock Price')
plt.rcParams["figure.figsize"]= [8,6]
plt.legend()
plt.show()

Original Stock Price : 116.73 LSTM Predicted Stock Price : 115.80221
Original Stock Price : 84.6542 LSTM Predicted Stock Price : 88.63065
Original Stock Price : 101.87 LSTM Predicted Stock Price : 98.44546
Original Stock Price : 94.74 LSTM Predicted Stock Price : 97.72489
Original Stock Price : 61.2728 LSTM Predicted Stock Price : 66.329666
Original Stock Price : 107.93 LSTM Predicted Stock Price : 106.03042
Original Stock Price : 127.60000000000001 LSTM Predicted Stock Price : 124.97925
Original Stock Price : 100.889 LSTM Predicted Stock Price : 99.33939
Original Stock Price : 108.7 LSTM Predicted Stock Price : 106.79183
Original Stock Price : 117.29 LSTM Predicted Stock Price : 117.76732
Original Stock Price : 63.5641999999999 LSTM Predicted Stock Price : 64.20401
Original Stock Price : 115.5 LSTM Predicted Stock Price : 111.38887
Original Stock Price : 128.62 LSTM Predicted Stock Price : 124.91204
Original Stock Price : 119.5 LSTM Predicted Stock Price : 114.8633
Original Stock Price : 92.4785 LSTM Predicted Stock Price : 90.07185
Original Stock Price : 61.4057 LSTM Predicted Stock Price : 66.06762
Original Stock Price : 118.23 LSTM Predicted Stock Price : 117.42694
Original Stock Price : 80.0771 LSTM Predicted Stock Price : 81.44185
Original Stock Price : 111.34 LSTM Predicted Stock Price : 116.31515
Original Stock Price : 122.0 LSTM Predicted Stock Price : 117.28014

LSTM R Squared = 0.9760255369637972
```

In above screen first we can see Original Test data stock price and then we can see LSTM predicted prices and we are displaying few records where you can see there is very close difference between original test prices and predicted prices and in blue color text we can see LSTM R SQUARED as 0.97%. so we can say LSTM prediction is accurate and in below screen we can see LSTM graph



In above graph x-axis represents Number of Days and y-axis represents STOCK PRICES and red line represents original TEST stock prices and green line represents LSTM predicted prices and we can see both lines are fully overlapping so test prices and predicted prices are very close.

So from above result we can say LSTM is accurate.

6. CONCLUSION

In this paper, the big data analytics are used for efficient stock market analysis and prediction. Generally, stock market is a domain that uncertainty and inability to accurately predict the stock values may result in huge financial losses. Through our work we were able to propose a approach to help us identify stocks with positive everyday return margins, which can be suggested to be the potential stocks for enhanced trading. Such appoach will act as a Hadoop based pipeline to learn from past data and make decisions based on streaming updates which the US stocks are profitable to trade in. We also try to find scope of improvements to our study in future directions. We intend to further our study by automating the analysis processes using scheduling module, then obtain periodic recommendations for trading the US stocks. We also plan to test some Neural Network model based learning rather than linear regression aims to accurately predict the US stock prices.

REFERENCES

[1] L. Zhao and L. Wang, “Price Trend Prediction of Stock Market Using Outlier Data Mining Algorithm,” in 2015 IEEE Fifth International Conference on Big Data and Cloud Computing, Dalian, China, 2015, pp. 93–98.

[2] M.D. Jaweed and J. Jebathangam, “Analysis of stock market by using Big Data Processing Environment” in International Journal of Pure and Applied Mathematics, Volume 119

[3] S. Tiwari, A. Bharadwaj, and S. Gupta, “Stock price prediction using data analytics,” in 2017 International Conference on Advances in Computing, Communication and Control (ICAC3), Mumbai, 2017, pp. 1–5

[4] P. Singh and A. Thakral, “Stock market: Statistical analysis of its indexes and its constituents,” in 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, 2017, pp. 962–966.

[5] Z. Peng, “Stocks Analysis and Prediction Using Big Data Analytics,” in 2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Changsha, China, 2019, pp. 309–312.

[6] G. V. Attigeri, Manohara Pai M M, R. M. Pai, and A. Nayak, “Stock market prediction: A big data approach,” in TENCON 2015 - 2015 IEEE Region 10 Conference, Macao, 2015, pp. 1–5.

[7] W.-Y. Huang, A.-P. Chen, Y.-H. Hsu, H.-Y. Chang, and M.-W. Tsai, “Applying Market Profile Theory to Analyze Financial Big Data and Discover Financial Market Trading Behavior - A Case Study of Taiwan Futures Market,” in 2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China, 2016, pp. 166–169.

[8] S. Jeon, B. Hong, J. Kim, and H. Lee, "Stock Price Prediction based on Stock Big Data and Pattern Graph Analysis:," in Proceedings of the International Conference on Internet of Things and Big Data, Rome, Italy, 2016, pp. 223–231.

[9] R. Choudhry and K. Garg, "A Hybrid Machine Learning System for Stock Market Forecasting," vol. 2, no. 3, p. 4, 2008.

[10] K. Kim, "Financial time series forecasting using support vector machines," Neurocomputing, vol. 55, no. 1–2, pp. 307–319, Sep. 2003.

[11] M. Makrehchi, S. Shah, and W. Liao, "Stock Prediction Using EventBased Sentiment Analysis," in 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Atlanta, GA, USA, 2013, pp. 337–342.

[12] H. Pouransari and H. Chalabi, "Event-based stock market prediction," p. 5.