STOCK PREDICTION USING LSTM TECHINQUE: REVIEW

Kurian Tom Department of Computer Applications
Amal Jyothi College of Engineering Kanjirappally, India
kuriantom2017@gmail.com

Jetty Benjamin
Asst.professor Department of ComputerApplication,Amal Jyothi College of Engineering Kottayam, India
jettybenjamin@amaljyothi.ac.in

***Abstract*: A portfolio of assets has never been easy to invest in; the abnormalities of the financial market prevent simple models from accurately predicting future asset prices. In the current scientific study, one of the hottest topics in machine learning, which is teaching computers to do things that would otherwise require human intelligence. This paper aims to develop a model for predicting future stock market values using the Long-Short Term Memory Model (LSTM). Our goal is to find out how accurate Machine Learning algorithms can be, and how much the epochs can help us.**

I. INTRODUCTION

The use of machine learning algorithms in quantitative finance has been researched in several areas, including forecasting prices, managing and limiting a full portfolio of assets, and many other tasks. The general definition of machine learning is any process that uses computers to discover patterns through analysis of data, rather than instructions. With machine learning, several models can be used to predict future asset values, specifically for asset selection, in quantitative finance. An arrangement of AI methodologies, including acritical neural associations, point helped backslide trees, support vector machines, and sporadic guess have actually been refined using a blend of bits of knowledge and learning models. It is feasible to look at non-direct examples and uncover connections that can't be uncovered utilizing straight techniques. These computations moreover defeat straight backslides similar to practicality and multicollinearity. These calculations additionally beat direct relapses as far as adequacy and multicollinearity. An incredible number of explorations are presently being led on the issue of AI techniques in finance; some used tree-based models to conjecture portfolio returns, while others utilized profound figuring out how to anticipate future costs of monetary resources. In this unique circumstance, a group of AI strategies dependent on Recurrent Neural Networks have demonstrated to be especially gainful in monetary market value expectation and anticipating. When foreseeing time-series information, an examination analyzes the exactness of autoregressive incorporated moving normal ARIMA and LSTM as illustrative systems. These calculations were tried on a bunch of monetary information, and the discoveries uncovered that LSTM beat ARIMA by a landslide. Our review intends to gauge changed shutting costs for an arrangement of resources utilizing an AI calculation dependent on LSTM RNN. The primary objective is to track down the best precise prepared calculation to foresee future qualities for our portfolio.

II. RELATED WORKS

Stock market prediction exchange gauging has been a review issue for some time, and several survey papers have been issued relating to the turn of events and prospering of profound learning approaches in the past to our job. While their focus may also be on advanced learning applications, stock exchange forecasting may be simply one of several monetary difficulties addressed in previous overviews. In this section, we analyses our motivations and unmistakable viewpoints, and we record some of them in sequential order.

Shah et al. (2019) provide AI techniques for forecasting monetary market prices, whereas Sezer, Gudelek, and Ozbayoglu (2019) cover additional financial instruments. Regardless, our motivation is to stay aware of the research pattern of utilizing deep learning strategies to beat conventional AI methods, for example, support vector machines, in many distributions, with a couple of exceptions, for example, Ballings, lair Poel, Hespeels, and Gryp (2015), who discover that Random Forest is the best calculation, trailed by Support Vector Machines, Kernel Factory, AdaBoost, Neural Networks, K-Nearest Neighbors, With the accumulation of recorded costs and a variety of information kinds, for example, financial news and Twitter, we anticipate that the benefits of deep learning processes will expand, and future research should keep up with this pattern.

We focus on recent improvements in the previous three years (2017–2019) and a tighter scope of stock cost and market record forecasting than Sezer et al. (2019), who focus on profound learning for monetary time series estimation and a substantially longer time frame (from 2005 to 2019 unequivocally). We would recommend this book to those who are interested in various monetary instruments, for example, item costs, bond costs, digital money costs, and so on. We are also more concerned about the execution work method and the repeatability of previous studies' conclusions, for example, dataset and code accessibility, which has garnered the attention of AI analysts (Gundersen and Kjensmo, 2018). We'd also pay greater attention to how securities exchange expectation (or monetary time series forecasting) differs from other time series forecasting challenges, such as gauging productivity despite forecast accuracy..

DOI: 10.5281/zenodo.6391047
ISBN: 978-93-5607-317-3 @2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

III. LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory (LSTM) is one of a few types of Recurrent Neural Network RNNs that might catch input from past stages and use it to make forecasts later on. An Artificial Neural Network (ANN) is comprised of three layers:

1) input layer,
2) Hidden layers,
3) output layer.

The quantity of hubs in the information layer of a neural network with just one secret layer is constantly controlled by the component of the information, and the hubs of the info layer associate with the secret layer by means of linkages called 'synapses.' The weight coefficient is the leader for signals in the connection between each two hubs from (contribution to the secret layer). In the wake of learning, the Artificial neural network will have ideal loads for every neurotransmitter since the learning system is innately a nonstop change of loads.

The enactment work, which is applied by the concealed layer hubs to the number of loads from the info layer, is a sigmoid or digression exaggerated (tanh) work; this change produces esteems with a negligible blunder rate between the train and test information utilizing the SoftMax work

The qualities got after this changing structure the result layer of our NN; in any case, these qualities may not be awesome; for this situation, a back proliferation interaction will be utilized to focus on the ideal mistake esteem. The back engendering process interfaces the result layer to the secret layer, conveying a message adjusting to the best weight with the ideal mistake for the quantity still up in the air. This methodology will be rehashed with expectations of working on our gauges and lessening forecast blunders.

The model will be prepared when this progression is finished. A repetitive Neural Network (RNN) is a kind of neural organization that predicts future qualities dependent on past arrangements of perceptions. This sort of Neural Network utilizes prior stages to learn information and expect future patterns

To expect and figure future qualities, the past phases of information ought to be recollected; in the present circumstance, the secret layer goes about as a storage facility for the past information from the consecutive information. The term intermittent is utilized to depict the method involved with anticipating future information utilizing pieces from past arrangements.

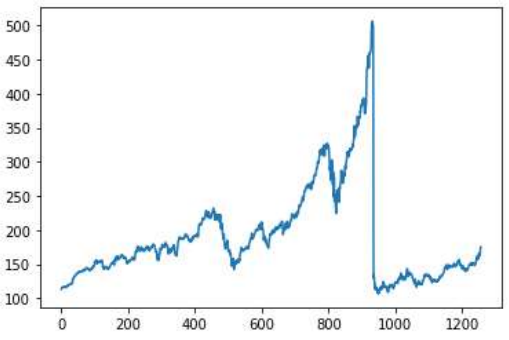
IV. IMPLEMENTAION

We will collect the data from apple company. This is the important for the stock prediction.

Because stock data changes over time, we process it using time series data. The data is then preprocessed as train and test. The index value is then chosen from OHLC. We take the close index value from stock data.

```
In [16]: df1=df.reset_index()['close']
```

```
In [18]: import matplotlib.pyplot as plt
         plt.plot(df1)
Out[18]: [<matplotlib.lines.Line2D at 0x21f302a3d90>]
```




```
In [24]: ##splitting dataset into train and test split
         training_size=int(len(df1)*0.65)
         test_size=len(df1)-training_size
         train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]

In [25]: training_size,test_size
Out[25]: (817, 448)
```

create a dataset matrix from an array of values

DOI: 10.5281/zenodo.6391047
ISBN: 978-93-5607-317-3 @2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

```
In [27]: import numpy
         # convert an array of values into a dataset matrix
         def create_dataset(dataset, time_step=1):
             dataX, dataY = [], []
             for i in range(len(dataset)-time_step-1):
                 a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
                 dataX.append(a)
                 dataY.append(dataset[i + time_step, 0])
             return numpy.array(dataX), numpy.array(dataY)
```

reshape into X=t, t+1, t+2, t+3 and Y=t+4.
For train data and test data.

```
]: # reshape into X=t,t+1,t+2,t+3 and Y=t+4
   time_step = 100
   X_train, y_train = create_dataset(train_data, time_step)
   X_test, ytest = create_dataset(test_data, time_step)
```

reshape input to [samples, time steps, and features] as required by LSTM

```
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)
```

Build the Stacked LSTM model.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

```
model.summary()
```

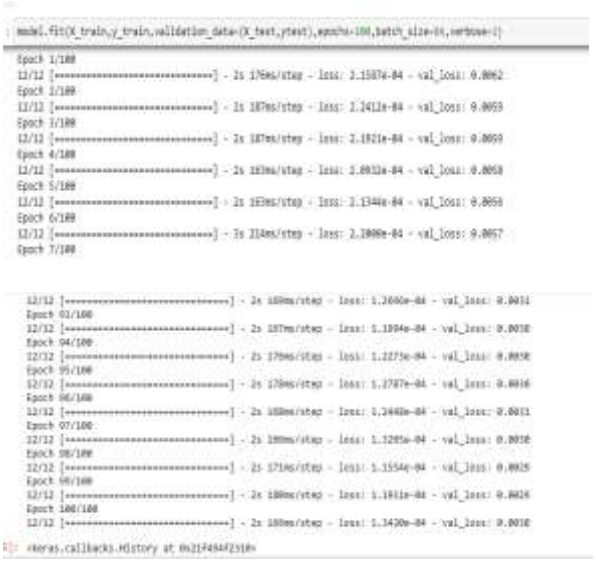
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```



Let's create a prediction and examine the performance metrics. Calculate the RMSE performance metrics.

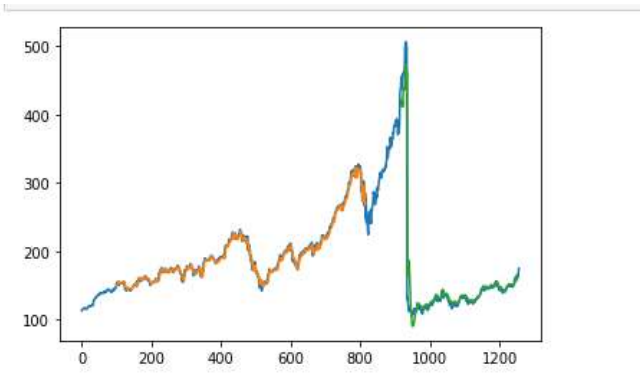
```
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

```
]: 199.872786340688
```

```
### Test Data RMSE
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
168.4254164539451
```

After we got the upsides of the train and test information. following the 100-venture highlight. The worth of the train and test information is utilized to decide the rightness of the worth. assuming that they are more like each other so the better the accuracy



We can see from the graph that the orange color represents train data and the green color represents test data. The complete data is represented by the blue
provide a forecast for the next 10 days

DOI: 10.5281/zenodo.6391047
ISBN: 978-93-5607-317-3 @2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

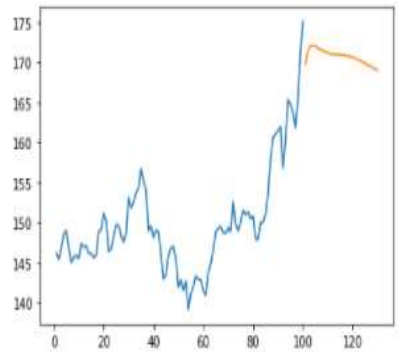
```
In [77]: # demonstrate prediction for next 10 days
from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

```
Out[82]: [<matplotlib.lines.Line2D at 0x21f303082e0>]
```



The expected values for the following 10 days are shown in orange color.

V. FUTURE WORK

Profound learning needs a great deal of handling capability. The flow investigation was carried out with the assistance of Google Colab, which has eight Tensor handling units (TPUs). If dedicated PC assets are available, additional analysis with more data and longer ages should be achievable. This might make hyper-boundary setup more visible.
The present effort is focused on calibrating the LSTM design's essential components. LSTM models of various types may be altered in a similar way.

Cross-market Analysis

The majority of present research focuses on just one stock market, in the sense that stock markets differ from one another due to trading regulations, although various markets may have certain common feature that may be exploited for prediction using techniques like transfer learning. Most surviving research focuses on a single stocks exchange, as financial exchanges differ from one another due to exchanging guidelines, however diverse company sectors may have some common factor that may be used for forecasting employing tactics, for example, move learning.

VI. CONCLUSION

This study proposes an RNN based on LSTM to predict future APPL values; the results of our model have been encouraging. The results of the tests suggest that our computation can track the evolution of resource opening costs. Later on, we anticipate determining the optimum sets for session information span and preparing ages that best suit our resources and improving our anticipation precision

VII. References

[1] .Batres-Estrada, B. (2015). Deep learning for multivariate financial time series

[2]. Patterson J., 2017. Deep Learning: A Practitioner's Approach, O'Reilly Media.

[3]. Sezer Omer Berat, Murat Ozbayoglu, Erdogan Dogdu A Deep Neural-Network Based Stock Trading System Based on Evolutionary Optimized Technical Analysis Parameters

[4].Siami-Namini, S., & Namin, A. S. (2018). Forecasting economics and financial time series: Arima vs. lstm. arXiv preprint arXiv:1803.06386

[5]. Bouktif Salah, Ali Fiaz, Ali Ouni, Mohamed Adel Serhani Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches

DOI: 10.5281/zenodo.6391047
ISBN: 978-93-5607-317-3 @2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam