

Linear Regression vs LSTM for Time Series Data

Rakshit Shah

*Department of Computer Engineering,
A.P.Shah Institute of Technology,
Thane(M.H), India 400615
Email: rakshitshah28@ieee.org*

Poojan Shah

*Department of Computer Engineering,
A.P.Shah Institute of Technology,
Thane(M.H), India 400615
Email: poojan.s@somaiya.edu*

Catherine Joshi

*Department of Computer Engineering,
A.P.Shah Institute of Technology,
Thane(M.H), India 400615
Email: catherine.joshi@gmail.com*

Rutuja Jain

*Department of Computer Engineering,
A.P.Shah Institute of Technology,
Thane(M.H), India 400615
Email: ajuturadeesh@gmail.com*

Rushikesh Nikam

*Department of Computer Engineering,
A.P.Shah Institute of Technology,
Thane(M.H), India 400615
Email: rnikam@apsit.edu.in*

Abstract—The sense of which model to use for prediction of time series data is very important. In this paper we have done a Project research on Linear Regression and Long short-term memory (LSTM) model both making a prediction using Indian Stock Market Index Nifty 50 which is a time series data. As we know the market changes daily and daily it touches new high's and low's and huge change in volume is seen as well, so can linear regression help us predict a better future prediction or Long short-term memory (LSTM) helps in predicting a better future prediction is what we have tested in this project. On the basis of the result of Root mean square error, we have given our conclusion. The project in a way also shows the strength of linear regression and that of Long short-term memory (LSTM) are different and shows a method of implementing time series data in both the models and checking their Root mean square error RMSE respectively.

Keywords—LSTM model, Linear Regression, epoch, batch size, Sequential, Open, Close, x_train, x_test, y_train, y_test

I. INTRODUCTION

This paper presents an overview of the comparison of Machine Learning models with a focus on Long short-term memory (LSTM) networks, which are been used for dynamic system modelling in diverse applications followed by Linear Regression. The aim in the scope of our problem is to set up a prediction model best suited for time series data. The vital part of machine learning is the dataset used which helps to predict the future. The dataset should be as concrete as possible as a little change in the data can led to massive changes in the outcome. In this project, Yahoo finance is used to obtain a dataset of the Nifty50. The dataset comprises of following variables : date, open, high, low, close, volume and adjusted close. The Nifty50 data is a time series data and proper data pre-processing is done before making it available for training the model and thus making the model fit for prediction

II. LITERATURE REVIEW

Stock price forecasting is not only an important topic but also a popular one in financial studies. As there are no significant rules to estimate or predict the worth of a share

within the share market, the share Market becomes an untidy place for predicting.

Ashutosh S. et al [4], predicts the stock prices using previous closing price and trading volumes to visualize both the anticipated price as well as values over time as well as the optimal parameters for the model. The model which we have used predicts 60 data points i.e the past two months' data based on the test data set and the last data point is pushed as the output. To add new information, it transforms the existing data completely by applying a function resulting in data modification, on the whole. LSTMs, a special kind of Recurrent Neural Nets (RNN), are capable of learning long-term dependencies. These are used specifically for learning the sequences and patterns of the time-series-based data. LSTMs make minor modifications to the existing data by performing a couple of multiplications and additions. This information flows through a mechanism known as cell states. This enables the LSTMs to selectively remember or forget things. The information at a particular cell state has three different dependencies such as the previous cell state, the previous hidden state, and input at the current time step. The mean balancing done over the processed LSTM provides better results and more accurate patterns over hysterical data sets.

Joshua B. et al [5]., defines three basic requirements of a recurrent neural network. This paper mentions that the system must be able to store information for an arbitrary duration and be resistant to noise i.e. fluctuations of the inputs that are random or irrelevant to predicting a correct output. and that the system parameters be trainable in a reasonable time. recurrent neural networks.

Polamuri S.R et al [6]. mentions implementing a linear statistical model (LSTM). In an LSTM Model, a linear model that is more or less ideal is primarily created and data is then added to it so that the linear model reflects the properties of the actual data. All the required pre-calculations for the prediction are used as an input for the prediction function.

It was observed from the literature survey that the world is looking forward to predicting the stock market using ma-

chine learning techniques. Technical analysis, traditional time series forecasting, and machine learning for prediction are the three conventional approaches for the prediction of the stock market.[3] The stock's price is an important indicator for a company. Many factors can affect the values of stock prices. Due to all these dependencies, on various factors, the stock prices do not remain static, but instead, they are dynamic, highly noisy, and nonlinear time series data. Machine learning has a great learning capacity for solving nonlinear time series prediction problems. Thus, it has been used greatly to research in this area.

Neural networks are being applied in many ways and in many fields in the present world. [1] Prof. S.P. Pimpalkar, J. Karia, M. Khan, S. Anand, and T. Mukherjee have made the use of an Artificial Neural Network to predict share market price with given input parameters of the share market. Data of any number of years can be predicted by Artificial Neural Networks and it can predict the future based on the past data. [1] uses feedforward architecture for prediction. The network was trained using ten years of data. The network succeeded in predicting the trends of the stock market even though it was not able to predict exact values. [3] used proposes an enhanced learning-based method for stock price prediction that considers the effect of news sentiments. Using Neural Network as a case learning-based method, the news sentiments relevant to the stock market is used to enhance the predictor. For the financial market value dataset, daily DJIA data was downloaded from Yahoo! Finance.

It is seen frequently that the prediction is chaotic rather than random. This means it can be predicted by carefully analyzing the history of the respective stock market [2]. [2] Machine learning increases accuracy by predicting a market value close to the tangible value. The dataset used is the vital part. [2] The dataset used is obtained from Yahoo Finance. In this dataset, supervised machine learning is employed. Open, close, low, high, and volume are the five variables contended in the dataset. They are the different bid prices for the stock at separate times. Regression and LSTM models are separately used. Minimizing error is done by regression and remembering the data and results for the long run is done by LSTM. [2] The dataset that was picked consisted of approximately 9 lakh records of the required stock prices and other relevant values. The data was in CSV format. Pandas library was used to transform the data into a data frame. Normalization of data was performed using the sklearn library in Python. The data were divided into training and testing sets. 20% of the dataset was kept for testing. [2] For predicting continuous values through some given independent values, the Regression-based Model is used. The gradient descent linear regression algorithm was used for predicting correct values by minimising the error function. Linear regression was performed on the data and then the relevant predictions were made. To determine the confidence score, the R-square confidence test was used. The predictions were shown by the help of graph with prices vs time. [2] The information belonging to previous states persists in LSTM. LSTM is an advanced version of RNN(Recurrent-

Neural-Networks). LSTM gives aid to the RNNs and regulates error by retaining information for older stages making the prediction more accurate. LSTM prevents the problem of Vanishing Gradient from happening. LSTM consists of remembering cells, an input gate, an output gate and a forget gate. The gates regulate the values remembered by the cell for long-term propagation. [2] They made a sequential model which consist of two layers added over each other with the output value of 256. 0.3 was the dropout value that was fixed. This means that 0.3 out of the total nodes will be frozen during the training process. This avoided overfitting of data and increased the speed of the training process. The mean square cost function was used to compile the model to maintain the error throughout the process and accuracy was chosen as a metric for the prediction. [2] LSTM model was proven to be more efficient than the linear regression model for determining stock market predictions.

III. PROBLEM STATEMENT

Many of the projects using time series dataset try to make prediction on the basis of linear regression and thus by doing that reduce the prediction efficiency. In our proposed method we showcase the use of both Linear Regression and Long short-term memory (LSTM) and show Root mean square error of both to show the difference between both the model and which model is more efficient.

IV. METHODOLOGY

In order to prove the point of LSTM being better suited for Time Series Data than Linear Regression, we have considered the following example of India stock market data and fetching its index Nifty 50's data from the year 2018 upto 14 April 2022.

Linear Regression: Linear Regression is a part of supervised machine learning which is used for simple predictions of a variable known as dependent variable, relying on the input value of another variable also known as independent variable. The independent variable takes the X-axis and Dependent variable takes the Y-axis respectively.

In order to train the linear regression model we first pre process the data and then make it available for the model to learn and thus predict. Beginning with the preprocessing of data, we first fetch the data from Yahoo finance and give the abbreviation of the Nifty50 name to be fetched, in this case for Nifty 50 we give the code "NSEI" and provide the start date as 2018-01-01 and end date as 2022-04-14 and download the data and store it in a form of csv file, for further usage. Further we clean the dataset on the basis of what we need and in this case we are giving inputs of only two variables, one which will act as dependent variable and other as independent variable. So we separate out two columns namely "Open" and "Close" of the market and we do this by the help of Pandas library.

The Pandas library is used mainly to perform data manipulation, playing with data table and performing analysis as well on the basis of given data. Once the columns are distinct,

we now import the sklearn library also known as Scikit-learn and from that we choose the `train_test_split` method and give inputs of “Open” and “Close” column values, along with that we have given a test size of 0.2. The Test size is a percentage indicator of the data that should be available for testing, in this case 20% of data is provided for the testing part and 80% of data is provided for the training part. The `train_test_split` method will now create 4 variables namely, `X_TRAIN`, `X_TEST`, `Y_TRAIN`, `Y_TEST` and store values of “Open” and “Close” in them. The `X_TRAIN` will consist of the 80% “Open” column values and `X_TEST` will contain the rest 20% “Open” column values. Similarly, `Y_TRAIN` will contain the 80% value of “Close” column values and `Y_TEST` will contain 20% remaining data value of “Close” column values.

This is where we end our pre-processing part and now we move to model training, and for that we import Linear-Regression model from the `sklearn.linear_model` library and we now store it in a variable named `model` for further use. After successfully importing the model we use `model.fit()` method and pass the values of `X_TRAIN` and `Y_TRAIN` in order to teach the model. Model fitting is a very important part of the training phase, as it enables the model to choose its environment best suited for learning the data. After the model fitting we now have our model all set with the knowledge of the training dataset and on the basis of this we now make the model predict future values. For that we create a variable `Y_PRED` and use the `model.predict()` method and pass the `X_TEST` data to see the prediction that model does on the basis of training dataset. Once we have the value of `Y_PRED` we now compare it with the values of `Y_TEST` and see if they are equal or different. In our case we found a huge difference in the values of both the variables and furthermore to check the error we take the Root Mean Square Error of the same. In order to do that we import `mean_squared_error` method from the `sklearn.metrics` library and pass the variable both `Y_TEST` and `Y_PRED` and thus by keeping the squared function `False`, we obtain the RMSE value of the model. The RMSE value came out to be 109.5948895818484. Here we end our Linear Regression part and now we move ahead with the LSTM model.

LSTM: LSTM is the abbreviation for Long Short Term Memory and it is associated with Recurrent Neural Network also known as RNN. LSTM is known for its feedback mechanism which makes it one of the best tools for prediction of time series data. Its basic structure consists of an input layer, hidden layers, dense layer and an output layer.

In our project to begin with the LSTM model we first have to pre-process the data as we did for the Linear Regression model. To begin with we again import data from yahoo finance and store it in a variable named `df`. The data remains the same used in the Linear Regression model ranging from the same date and same index of nifty 50. Now we will filter the dataset on the basis of our requirement and use the “Close” column to analyse the data. Once we do that we now proceed to divide the data into training and testing parts. The difference between

LSTM and Linear regression is that we will not require any dependent and independent variables, all we need is a constant one variable whose prediction is to be done purely on the basis of time frame. So now we check the length of our “Close” column and we have divided 80% of the data for training and the rest 20% for testing. To do that we have imported a library named `math` and used `math.ceil()` function and we pass the updated dataset with “Close” column in the `math.ceil()` and store the result in a variable named `training_dataset_length`.

The reason why we did not use `train_test_split` here is that we do not wish to jumble our data but just divide it and on the other hand what `train_test_split` does is, it randomly splits the data and hence jumbles the data in the process. So to avoid that we manually split the data here. Once the length of training and testing is defined we now normalise the data using `MinMaxScaler` which is a part of `sklearn.preprocessing` library. The need for Normalising the data is because the values of “Close” column ranges are very high and if the LSTM model is allowed to access such high values, there are high chances of the model getting a wrong training. So in order to avoid such mistakes we normalise the data in the range of (0,1) where in the least value in the data will be given 0 as its value and highest value in the data will be given 1 as its replaced value, and all the other values will range in between 0 to 1. And to achieve this we use `MinMaxScaler` and give its feature range from 0 to 1 and fit the transform on the dataset and now we call this data set as `new_scaled_data`.

Post data scaling we then append the data in two different arrays namely `x_train` and `y_train` and we append them in a manner that for every 60 inputs in `x_train` we correspond 1 output in `y_train` for training purpose and then we increment the loop variable in which now the next 60 element array will be created and added in `x_train`, but this time the output data previously present in `y_train` will now be a part of second array of `x_train` and new output will be added in `y_train` for this corresponding array. Lets understand this with the following example:

TABLE I
EXAMPLE OF ARRAY GENERATION

No. Of Iterations	X_Train	Y_Train
1	Array-1[1,2,3,4,5,6,7,8,9,10]	Array-1[11]
2	Array-2[2,3,4,5,6,7,8,9,10,11]	Array-2[12]
3	Array-3[3,4,5,6,7,8,9,10,11,12]	Array-3[13]

Once we have these `X_train` and `Y_train` data ready we convert them into a numpy array by using the `numpy` library. `Numpy` library helps in order to deal with large arrays or metrics. The LSTM model takes input of only a three dimensional array so hence again by using `numpy` we reshape the `x_train` data. Here we end our Data pre-processing and now we move on to build the LSTM model.

First and foremost we import the `model sequential()` from `Keras.model` library. `Sequential` model is a linear stack of layers and it has a method of creating deep learning models and allows the instance to add layers to it in order to train the

model. Sequential allows the data to flow from one layer to another and even allows the layer to perform a feedback mechanism. So we create a model Sequential() and we add our first layer of LSTM with 50 units of nodes and return_sequence as True. return_sequence allows the layers to perform back tracking and hence enabling feedback mechanism. Along with this we give the input_shape as x_train in the first layer. Along with that we even add a Dropout layer which will randomly drop a few input units and set them to zero, this helps to avoid overfitting of the model and even increases the accuracy of your model.

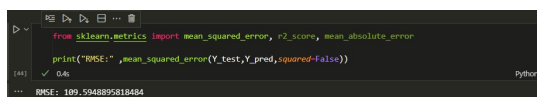
We then add two more similar layers to the model with return_sequence as True and one layer after that with return_sequence as False. That means after this layer the model will not be able to backtrack and the output generated in this layer will proceed further. Now we add one final layer and that will be the Dense layer. A dense layer stores all the outcomes and outputs of the previous layer and stores them in its neurons and these neurons provide the final result of the model.

Here we end with the creation of the model and now we move towards Compilation of the model. In order to do that we use a method known as compile and use the optimizer “adam” to do the compilation. “Adam” stands for “adaptive moment estimation” and it is best suited for problems with large datasets and it also possesses little memory requirement along with computational efficiency. Adam also has a good adaptive learning rate making it one of the best optimizers.

Once the model is compiled we now Fit the model giving it x_train and y_train data along with batch_size 32 and epochs=1. Batch size refers to the number of training examples used by the model in one iteration, and epoch refers to the number of passes of the dataset that the model has to complete. By this step we complete our model training part successfully and moving ahead we use model.predict() method to make the prediction on the basis of x_test data, post this step we do the inverse transformation of the predicted data. We do the inverse transform in order to revise the normalised data, which now converts all the data ranging from 0 to 1 to its original values. Once the data is inverse transformed and predictions have been made, we now check the Root Mean Square error as we did for the linear regression model.

The RMSE value came out to be just 16.372621673145932. And here we end with our LSTM model and hence we see that LSTM has a lower RMSE value in comparison to Linear Regression for time series data.

V. RESULTS

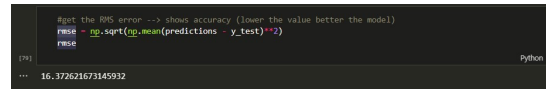


```

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
print("RMSE:", mean_squared_error(Y_test, Y_pred, squared=False))
RMSE: 109.5948895818464

```

Fig. 1. RMSE value of Linear Regression.



```

#Get the RMS error --> shows accuracy (lower the value better the model)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
rmse
16.372621673145932

```

Fig. 2. RMSE value of LSTM Model .

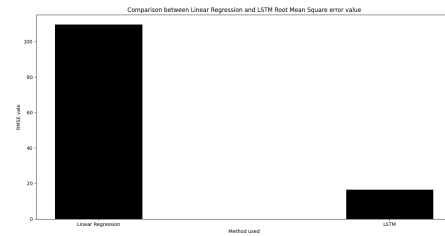


Fig. 3. Difference Between Both the RMSE .

VI. CONCLUSION

By performing the following method and seeing the result and checking the RMSE of both Linear Regression and Long short-term memory (LSTM) and the difference between them we can conclude that Long short-term memory (LSTM) model is best fit for Time Series Dataframe and can give a better accurated result.And hence we also conclude that Linear Regression is more suited for non Time series data and should be used for that. We also conclude that data when given with respect to time can be unpredictable hence a Recurrent Neural Network such as Long short-term memory (LSTM) model who has feedback mechanism available shall be used for better accuracy and predictions.

REFERENCES

- [1] Prof. S .P. Pimpalkar, Jenish Karia, Muskaan Khan, Satyam Anand, Tushar Mukherjee.(2017) “Stock Market Prediction using Machine Learning”, International Journal of Advance Engineering and Research Development, Volume 4
- [2] I. Parmar, N. Agarwal, S. Saxena, R. Arora, S. Gupta, H. Dhiman, L. Chouhan, “Stock Market prediction using Machine Learning”, 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC).
- [3] Z. WANG, Z. LIN. (2018) IEEE International Conference on Data Mining Workshops (ICDMW)
- [4] A. Sharma, S. Modak, Eashwaran Sridhar, “Data Visualization and Stock Market and Prediction” in International Research Journal of Engineering and Technology (IRJET), 2019
- [5] Y. Bengio. R. Pascanu, T. Mikolov “Learning Long-Term Dependencies with Gradient Descent”, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 5, NO. 2, MARCH 1994.
- [6] Polamuri S.R, Shrinivas K, A. Krishna M. “A Survey on Stock Market Prediction Using Machine Learning Techniques, 2020
- [7] Naeini MP, Tareimian H, Hashemi HB ”(2010) Stock market value prediction using neural networks.” IEEE
- [8] Aparna Nayak, M. M. Manohara Pai, Radhika M.Pai “Prediction Models for Indian Stock Market” (2016)
- [9] Nisha Shetty, Ashish Pathak “Indian Stock Market Prediction Using Machine Learning and Sentiment Analysis” , December 2017