Indian Stock Market Price prediction

Anant Sagar
IIITD
anant19232@iiitd.ac.in

Raj Kumar
IIITD
raj19084@iiitd.ac.in

Utkrisht Sikka
IIITD
utkrisht19215@iiitd.ac.in

1. Introduction

In this project , we are predicting closing price of NIFTY50 stocks on a given day based on the performance of the stock on previous days. We are using machine learning techniques to achieve our goal. Stock forecasting is a challenging task because it is inherently unpredictable. Anyone can sell stocks at any arbitrary price at any time.Also, trends in stock price keep changing. One cannot train the model on 10 year old data and use it to predict today's price. We have chosen an Indian stock as hardly any previous research used it as dataset.

2. Related Work

Singh et al [10] used deep learning to predict the price of the stock on the next day using (2D-PCA)+Deep Neural Network model and the data of the previous k days where they took k as a window size having values = [20,40,60,80,100]. 2D -PCA reduced the dimension and then they applied DNN on the reduced dimensional data. Their data had 36 normal features. Rene [7] predicts stock returns(instead of prices) of VolksWagon using stock returns of its suppliers.He employes Elastic Net, Decision Trees, XGBoost and LightGBM. Later, he fuses quant trading algorithms with his models to evaluate how much profit one can earn on a bankroll of 1000EUR. Chen et al [6] applied LSTM on a Chinese stock to predict stock returns.Rubi et al. [9] combined social media news with past prices to predict future price movement. Ayodele et al. [5] fused previous stock prices with company performance like rumors, growth, sales,etc. to predict its future price. Zhiqiang Guo [8]also implemented 2-directional 2 dimensional PCA on Shanghai Stock Market Index which had 36 input features in the data and they use a Radial Basis Function Neural Network (RBFNN) model to forecast stock market behavior.

One of the users [4] on kaggle reported RMSE of 10.9(we are considering this as state-of-the-art on our dataset) using his auto regressive model on our dataset. His target variable was opening price on each day while our target variable is closing price on each day. His train-test split is 80-20.He only feeds opening price of previous days in his model.

3. Dataset and Evaluation

The dataset has been taken from kaggle [3]. It contains Adaniports stock price from 2007 to 2021.There are 3322 days (or rows in our dataset). There are 15 columns in the dataset (namely Prev Close, Open, High, Close, etc.). Our train-validation-test split was 4year-1year-1year(900,200,200) , 8year-1year-1year(1900,200,200) and 12year-1year-1year(2800,200,200). From here on, 4-1-1 will imply 4year-1year-1year. We took window sizes of 5,10 and 15. A window size of x means that to predict close price of current day, the model is given features of previous x days, as a sample.

3.1. Extracted Features

Here we extracted some new features from the data which are Standard Moving average, Standard Deviation, Bollinger upper and lower band with window size 20, Momentum with standard window size 90, Williams index and Stochastic K index with window size 14 and lastly 3 Stochastic D index with window size 3. We also removed null value rows from the data which were created mainly because of rolling window in calculating SMA, STD and other indexes. The plot can be seen in Figure 11 . Adding new features also resulted in removal of 90 rows because of rolling window which we removed as they had null values.

3.2. Evaluation Metric

RMSE(Root Mean Square Error) is our evaluation metric. $RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$ where y_i and $\hat{y}_i$ are true closing price and predicted closing price of i^{th} day respectively. n is number of days in test data.

4. Methodology & Analysis & Results

4.1. MLR

MLR is Multiple Linear Regression. We used Linear regression with multiple variables. Here the loss function

which we used was mean squared error. Analysis : MLR works better when the number of window size is around 5 but the test-rmse and val-rmse grows with increasing window size and train-rmse decreases which shows that the model is overfitting when we increase the window size which can be inferred from figure 1

We used L2 regularization with alpha = 1.5 with Linear Regression as the results were not satisfactory without regularization and Train RMSE was also increasing with increasing number of windows. The plot of Linear Regression model without regularization can be seen in Figure 2

We also used PCA with Linear Regression to reduce the dimensionality. We found that $n_components = 55$ works best for most window sizes so we used $n_components = 55$ to reduce dimensionality which improved our results and our RMSE improved to 3.92 in testing and 3.831 in validation data for window size = 5 and 4year-1year-1year split as shown in Figure 3.

We also analysed that the range of closing price in which the rmse is higher. For that we calculated normalized score of each true price and clubbed them into of three types: when normalized score is less than 0.35(low) , between 0.35 and 0.7(mid), and greater than and equal to 0.7(high) And after analysing this we found that generally predicting high price results in higher rmse compared to lower and mid values . Results can be seen in figure 4. Also we concluded that Linear Regression models are high bias with low variance as they assume the data to be linear but as the size of window increases bias decreases and variance increases as the training error decreases but test error and validation error increases significantly.

4.2. LSTM

Long Short Term Memory is a Recurrent Neural Network used in times series/sequential data. Before feeding data into the model, we normalised all numeric columns(i.e mapped them to a value between 0 and 1). Architecture [2] constructed in LSTM is given in Figure 5.

Configuration was as follows: loss function = mean squared error, optimizer = adam, learning rate = 0.001, beta 1 = 0.9, beta 2 = 0.99 (parameters in adam optimizer). RMSE values are given in TABLE 2. In any specific data split, overall RMSE is almost equal for each window size. Highest RMSE is seen for high close price values. We plotted validation and train error with epochs for each LSTM model. Figure 8 shows such plot for 4-1-1 split and window size 5. Clearly, there are a lot of ups and downs in the train and validation error. This can be attributed to quite high learning rate such that model parameters oscillate around optimal parameters. Note that even though train error is more than validation error, its still not a case of high bias because RMSE usually increases with the increase in samples, observed in our case. In another model, we incorporated regularization by dropping out 20 percent inputs between LSTM layers 'lstm_8' and 'lstm_9', and also between 'lstm_9' and 'lstm_10'But RMSE increased. RMSE was 15.47(overall), 32.95(on high close prices), 14.54(mid close prices) and 10.16(low close prices), on 12-1-1 split and window 5.

4.3. Elastic Net Regression

We used ElasticNet of scikit library. Note that since we found training error more than test error in initial ElasticNet models, we dropped the idea of taking validation set. During GridSearchCV, we took training set and chose best model that had least rmse on training set. Then we tested it on test set. For elastic net , the splits were 5year-1year, 9year-1year and 13year-1year. Here, 5year-1year means that 1st five years of data was used as training data and next 1 year of data was used as testing data and similarly for other notations. Objective function was

$$1/(2 * nsamples) * ||y - Xw||_2^2$$
$$+alpha * l1ratio * ||w||_1$$
$$+0.5 * alpha * (1 - l1ratio) * ||w||_2^2$$

First, we trained the model on normal data with regularization parameter 0.01, l1ratio 0.5, window size 5. RMSEs were 3.77(for 5-1 split), 6.29(for 9-1 split) and 8.53(for 13-1 split). Next we took window sizes of 5, 10 and 15. For window 10, test RMSEs were 5.15(for 5-1 split), 6.45(for 9-1 split) and 8.65(for 13-1 split). For window 15, test RMSEs were 6.35(for 5-1 split), 6.75(for 9-1 split) and 8.87(for 13-1 split). Clearly, increasing window size doesn't help. Next we applied gridsearchCV on Regularization parameter (values being 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.0, 1.0, 10.0, 100.0). Best model had test rmse 3.42(for 5-1 split), 6.04(for 9-1 split) and 8.43(for 13-1 split). Further improvement was obtained when we applied grid search CV on l1ratio (values being 0,0.1,0.2,..0.9). Best model had test rmse 3.42(for 5-1 split), 5.99(for 9-1 split) and 8.43(for 13-1 split). We tried taking only closing price as column and nothing else. But that didn't improve the rmse. Note that till now, train error was above test error, so model was not overfit. Rather, it could more likely be said that it was under-fit as training error is above our expectations.

The model obtained now is better than state of the art because we applied Grid Search over hyperparameters. However, in state-of-the-art, the developer has taken only 1 value of each parameter passed to ARIMA function. One limitation of my approach is that it might be the case that the closing prices we are predicting could be a higher degree function of previous closing prices. When we test our best model on closing prices which were of low value, intermediate value, and high value, the obtained RMSE are given in Table 1.Out of high,low and intermediate prices, elastic net failed much on intermediate close prices.

4.4. CNN

Here we used a CNN model using Conv layer of tensorflow.keras on univariate timeseries data to predict the price on ith day using the prices of previous 5 days(window size 5). Configuration was as follows : loss function = mean squared error, optimizer = adam. Then we trained the model and got an RMSE of 3.61, 8.36 and 10.04 for window size 20. AS increasing window size didn't help and since CNN is computationally expensive So we didn't try CNN on window size 10 and 15. Here too, the training loss was more than validation loss. So it can't be said that the model is overfitting, rather we can say that the model is not able to fit the data in training data because of the high noise. Thus, the model gives high RMSE. The CNN model is not able to fit the training data because of highly noisy timeseries data but it is able to make nice predictions on test data . The curve for validation loss can be seen in Figure 6. The curve for training loss against number of epochs can be seen if Figure 7. We referred how to use CNN for univariate timeseries forecasting data from here [1].

4.5. Challenges Faced

a) MLR model gives poor results when the window size is increasing so it was tougher to find suitable dimensionality reduction to improve the rmse error. b) The hyperparameter tuning was also very difficult in Ridge regression especially to tune alpha value.

a) LSTM model takes around 3 hours to train for complete dataset and 400 epochs. Also, knowledge of LSTM and its internal working is difficult to study. b) We first used sgd as optimizer which was slow in converging so we swicthed to adam optimizer.

a) It was tough to find the appropriate kernel size and sequence of layers to use CNN on timeseries forecasting data.

4.6. Difference b/w our model and state-of-the-art

a) He targetted opening price column. b) He used Autoregressive model with Moving average features to predict the opening price of next day. c) He trained on 80 precent data and tested on 20 percent data with no validation split. d) We got RMSE of 10.1 with L2+Linear Regularization for n_components = 55 and window size = 5 as compared to his RMSE of 10.9.

5. Contributions

5.1. Raj

Planned Deliverables: Performing EDA on dataset and making CNN model and also contributed in doing error analysis of Multiple Linear Regression Model.

References & Citations: 1. https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/ 2. https://medium.com/codex/how-to-calculate-bollinger-bands-of-a-stock-with-python-f9f7d1184fc3 3. https://towardsdatascience.com/moving-averages-in-python-16170e20f6c

Individual Contributions: All the graphs/code used in Exploratory Data Analysis and extraction of new features. All code/models of CNN were also made by him.

5.2. Utkrisht

Planned Deliverables: Making LSTM and Elastic Net Regression model and refining them.

References & Citations: Reference 1(see references section) provided a brief start as to how and what libraries can be used for LSTM. We borrowed architecture of LSTM from there and tweaked it. Preprocessing ideas like normalising data by scaling were also learnt from there. https://machinelearningmastery.com/elastic-net-regression-in-python/ was referenced to know what is regression and which library can be used to implement it. Individual Contributions: All the models of Elastic Net regression and LSTM(and its graphs) in the code files. Since Elastic Net is very similar to Linear Regression, we decided that it could be given to 1 member and Anant be given task of unit testing.

5.3. Anant

Planned Deliverables: Making baseline classifier and elastic net regression.

References & Citations: 1. https://datatofish.com/multiple-linear-regression-python/ This tutorial provided a complete overview about how to perform multiple linear regression for stock market prediction or for other time-series data. 2. https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/ 3. https://towardsdatascience.com/time-series-forecasting-with-2d-convolutions-4f1a0f33dff6 Reference 2 and 3 were used for further research in the domain of CNN on time-series data with deeper networks and for multivariate data.

Individual Contributions: The baseline models of Multiple Linear Regression along with regularization. Creation of trivial data-sets for sanity checking of the developed models. Resource collection and research for deeper CNNs.

data	RMSE on high close	RMSE on intermediate close	RMSE on low close	RMSE on overall close price
5-1	5.23	4.88	2.64	3.42
9-1	6.22	6.4	4.39	5.99
13-1	5.38	8.29	7.32	15.41

Table 1. RMSE values of LSTM in different on different groups of closing price

data	window	RMSE on high close	RMSE on intermediate close	RMSE on low close	RMSE on overall close price
4-1-1	5	8.37	5.54	3.18	5.43
8-1-1	5	4.9	8.33	6.75	7
12-1-1	5	28.69	13.48	7.32	13.03
4-1-1	10	8.95	6.31	3.41	6.08
8-1-1	10	5.36	8.44	6.76	7.11
12-1-1	10	29.72	12.75	7.5	13.95
4-1-1	15	9.01	5.99	3.65	5.99
8-1-1	15	5.83	7.79	6.73	6.89

Table 2. RMSE values of LSTM in different on different groups of closing price

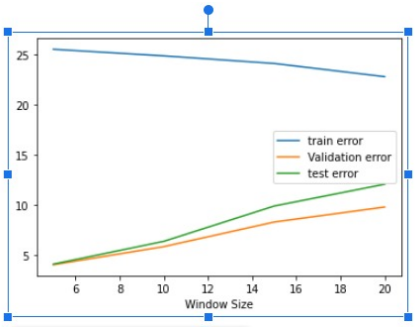

Figure 1. Train , Validation , Test RMSE + L2 regularization vs number of windows

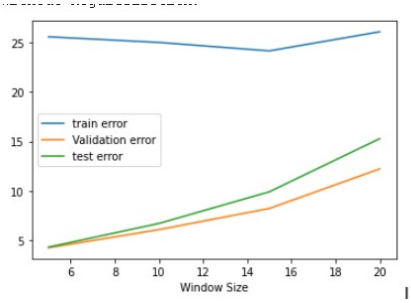

Figure 2. Train , Validation , Test RMSE + No regularization vs number of windows

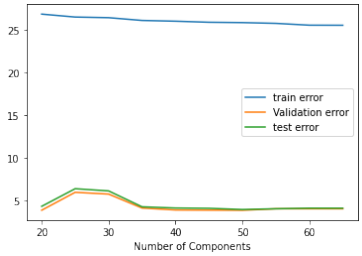

Figure 3. Linear Regression + L2 with pca plot vs Number of Components for window size = 5

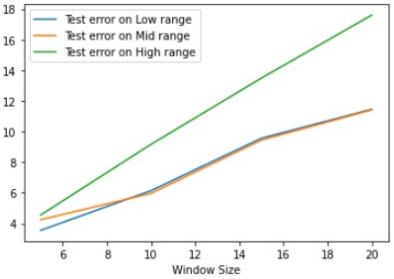

Figure 4. High , mid , low RMSE vs Window size plot

Layer (type)	Output Shape	Param #
lstm_8 (LSTM)	(None, 160, 50)	10400
lstm_9 (LSTM)	(None, 160, 50)	20200
lstm_10 (LSTM)	(None, 160, 50)	20200
lstm_11 (LSTM)	(None, 50)	20200
dense_2 (Dense)	(None, 1)	51

Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0

Figure 5. architecture in LSTM

References

[1] Cnn on univariate timeseries data. https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/. 3

[2] Hands-on guide to lstm recurrent neural network for stock market prediction. https://analyticsindiamag.com/hands-on-guide-to-lstm-recurrent-neural-network-for-stock-market-prediction/. 2

[3] Nifty-50 stock market data (2000 - 2021). https://www.kaggle.com/datasets/rohanrao/nifty50-stock-

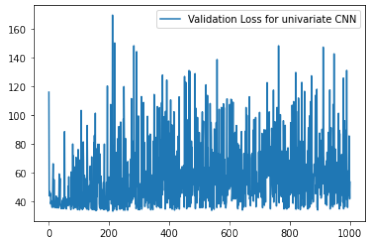

Figure 6. CNN Validation Loss

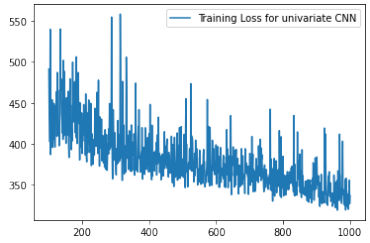

Figure 7. CNN Training Loss

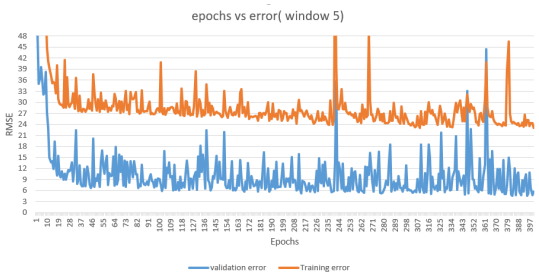

Figure 8

Figure 9. Close Price(in Rupees) vs day

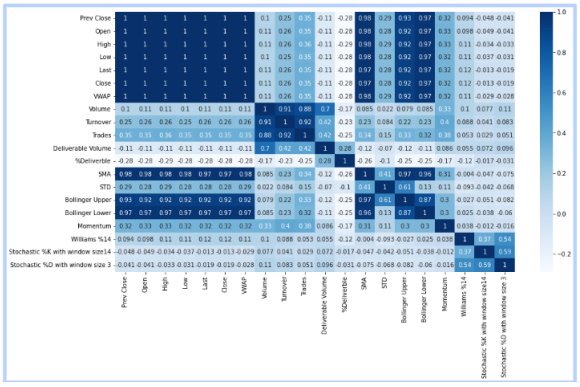

Figure 10. Correlation Matrix of features

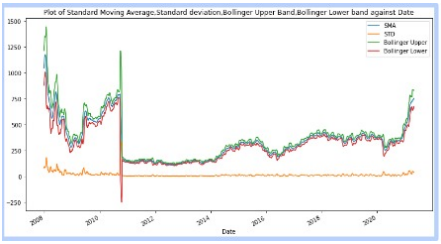

Figure 11. SMA , STD , Bollinger Upper and Lower band plots

market-data. 1

[4] Stock prediction code. https://www.kaggle.com/code/meow62k/stock-prediction-code. 1

[5] Adebiyi Ayodele A., Ayo Charles K., Adebiyi Marion O., and Otokiti Sunday O. Stock price prediction using neural network with hybridized market indicators. *Journal of Emerging Trends in Computing and Information Sciences*, 2012. 1

[6] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction : A case study of china stock market. *IEEE International Conference on Big Data (Big Data)*, 2015. 1

[7] Rene M. Glawion. Quantitative trading with machine learning. *Stanford University*. 1

[8] Zhiqiang Guo, Huaiqing Wang, Jie Yang, and David J. Miller. A stock market forecasting model combining two-directional two-dimensional principal component analysis and radial basis function neural network. 2015. 1

[9] Rubi Gupta and Min Chen. Sentiment analysis for stock price prediction. *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2020. 1

[10] Ritika Singh and Shashi Srivastava. Stock prediction using deep learning. *Springer Science+Business Media New York*, 2016. 1