CLERS-NET: A DEEP LEARNING ENSEMBLE MODEL FOR REAL TIME STOCK PRICE PREDICTION

Saikrishnan S[1*†], Kiruthika J[2†], Madhumithaa S[3†] and Dr. Saritha M[4*]

[1*]Avalara Technologies, Pune, 411045, Maharashtra, India.
[2]Citicorp Services, Chennai, 600087, Tamil Nadu, India.
[3]Workato, Hyderabad, 500081, Telangana, India.
[4*]Department of Computer Science, SSN College of Engineering, Chennai, 603110, Tamil Nadu, India.

*Corresponding author(s). E-mail(s): saikrish0108@gmail.com; sarithamadhesh@ssn.edu.in;
Contributing authors: kiru2405@gmail.com; satmadhu01@gmail.com;
†These authors contributed equally to this work.

Abstract

In recent years, stock price prediction has become prevalent because of its dynamic and unpredictable nature. The accurate prediction of stock prices can be of great use for investors to maximize their profits. Deep Learning techniques have been proved to provide accurate results in stock price prediction. The stock price fluctuation depends on a variety of factors like political and economic situations, news about the company, etc. This work aims to provide a real time web application that predicts the closing price of the NIFTY-50 companies for the next 7 days. Our proposed CLERS-Net model is trained using the historical stock price data collected from Kaggle and news articles obtained using pygooglenews from 2011 to 2021. The proposed architecture is composed of a Flair, CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory) and Random Forest Regressor models. The Pre-trained Flair model is used to compute sentiment scores from the news articles about the companies. The sentiment scores are appended to the

numeric stock price data and fed as inputs to LSTM and CNN models. The outputs obtained are fed to a random forest regressor which predicts the next day's stock price. The ensembled model performs exceedingly well with a negligible 2% error. The stock price of a selected company is forecasted and visualized in a real time web application.

Keywords: Stock price prediction, deep learning, ensemble learning, sentiment analysis.

1 Introduction

The stock market is very volatile and difficult to understand. Predicting the future stock price is a very challenging task as the stock market data is highly time-variant and depends on a variety of factors like politics, economic situation, society, etc. The accurate forecast of stock price movement will be useful for investors to maximize the profit. From [7] we find a very strong correlation between stock prices and tweets about the company. Since news about a company directly reflects the company's growth, it becomes a very important KPI (Key Performance Indicator) in stock price movement.

There are two main approaches for stock price prediction. The first approach is technical analysis which makes use of quantitative data to predict the future stock price. The main quantitative data includes historical stock prices like opening and closing price, previous close price, volume traded, and so on. The second one is fundamental analysis which uses qualitative data that includes external factors such as political and economic factors, market situation, information about the company from news articles, social media, blogs, etc. In recent years, advanced intelligent techniques like machine learning and deep learning are used to predict stock prices using technical or fundamental analysis which have given significant results.

From an investor's perspective, it is necessary that he is well informed about both the technical and the fundamental aspects of a company so that he can decide when and in which stock to invest.

The main objectives of our work are to:

- Perform sentiment analysis to obtain sentiment scores from google news articles.
- Provide a 2 tier model (CLERS-Net) which predicts the next day closing price using historical stock price data.
- Perform an ablation study to show the importance of different components in CLERS-Net.
- Analyze the impact of COVID on different sectors of stock market.
- Build a real time web application to visualize our predictions.

2 Related Works

There have been several attempts to predict stock price with machine learning techniques. Existing works showed that more progress has been made in near-term [1, 2] and long-term [4] price prediction. Lucas compared Linear Regression, Stochastic Gradient Descent and Support Vector Regression in predicting long-time stock prices in 2014 [18]. It was observed that SVR provided more accurate predictions. In 2017, Support Vector Machines [2] were applied to build a regression model using historical stock data. However, the accuracies of these regression models were observed to be low. Stacking regression is an ensemble learning technique which was used to combine multiple regressors. The stock prices were predicted combining both SVM and ExtRa regressors [15]. It was observed that the ensemble models performed significantly better than the individual regressors. The advent of deep learning in stock price forecasting has resulted in a radical improvement in this domain. The sections below describe the earlier researches that are related to stock price predictions using LSTM, CNN and sentiment analysis.

2.1 Long Short-Term Memory

Long Short Term Memory is a special kind of Recurrent Neural Network which can handle long-term dependencies. With the introduction of LSTM [3], the analysis of time dependent data became more efficient. This model was used in stock price prediction by [4, 5]. LSTM neural networks were modeled to learn patterns from historical data which was exploited in forecasting stock prices.

Apart from the application of classical LSTM in stock price prediction, researches pertaining to LSTM variations in the same area have also been identified. LBL-LSTM [4] based model was used to predict the short-term fluctuations of stock prices. LBL-LSTM was used to forecast stock prices considering external fluctuations in [14]. Gaurav[14], had discussed the effectiveness of using LBL-LSTM to predict stock prices during COVID-19. This model predicted short-term fluctuations when external factors were involved. Although LSTM is a popular technique used in this field, it is observed that it takes a long time to train and is prone to overfitting.

2.2 Convolutional Neural Network

Convolutional Neural Network is known for its ability to automatically learn relevant features without supervision. Mehtab and Sen [16], compared both CNN and LSTM in predicting stock prices. Further works showed that using multivariate data with CNN was more accurate than univariate analysis [17].

Selvin [1] introduced the application of sliding window method with data overlap. This method was used with three models namely, CNN, RNN and LSTM in order to analyze the inter relation within the data. CNN was identified as the best model as it was able to capture the trends in data within the current window. CNN networks were able to automatically extract features from the past values of stock price time series and predict future prices.

Most research works showed that CNN was faster compared to other models. However, in most cases it requires a large training set in order to give more accurate predictions.

2.3 Sentiment Analysis

The fluctuations in stock market are determined by technical indicators and various external factors. In recent years, it has been found that there is a strong correlation between the movement of stock prices and the publication of news articles. There has been an increasing research attention on studying investors' sentiments and their impact on the stock market. We can easily identify investors' views on stock market and its trends from news articles, group forums and social media. With the recent advancements in deep learning for Natural Language Processing (NLP), researchers use various text mining algorithms to evaluate the investors' sentiments from news articles and tweets.

Bharadwaj [8], considered news articles about Nifty and Sensex indices, and studied the effect of sentiment analysis on stock indices. [11], combined both fundamental and technical analysis of stock prices. LSTM was used to predict stock prices and Textblob(NLP) for sentiment analysis to validate the predictions. It was observed that the addition of sentiment scores effectively improved the accuracy of the forecast. Our proposed system unlike the existing systems, uses an ensemble model along with sentiment analysis to predict stock prices. While most models are trained for only a few companies, our model is trained for all the NIFTY-50 companies. Moreover, we predict the stock prices in real-time.

3 Dataset

The stock prices can be predicted by both fundamental and technical analysis. For technical analysis, historical stock price data from 2011 to 2021 for the NIFTY-50 companies is extracted from Kaggle. The feature set contains everyday prices which include Open, High, Low, Last, Prev Close, VWAP, Volume, Turnover, Deliverable Volume and Deliverable %. This dataset contains 1,37,544 datapoints out of which 80% is used for training and 20% is used for testing.

For fundamental analysis, news articles for the 50 companies are collected using the GoogleNews package from the library pygooglenews. Pygooglenews allows us to obtain news articles that contain a given keyword within the specified date range. A dataframe is created for each company and the title and published date of each article about that company is appended to the dataframe. The collected news data is cleaned by removing white spaces, anchors, links and other special characters. Algorithm 1 shows the steps involved in collecting the news articles about the companies. A pre-trained Flair model is used to predict the sentiment scores of each company using the collected news articles. These scores are added as another feature to the numeric dataset to obtain the final dataset.

Algorithm 1 Collect GoogleNews data

```
df = Empty dictionary
while company in list − of − companies do
    Create dataframe for each company, df[company]
    while year between 2011 − 2021 do
        search = gn.search(company data for every month in that year)
        while each in search[entries] do
            d = empty dictionary
            d['title'] = each['title']
            d['Published_at'] = each['Published_at']
            df[company].append(d)
        end while
        year = year + 1
    end while
end while
```

4 System Architecture

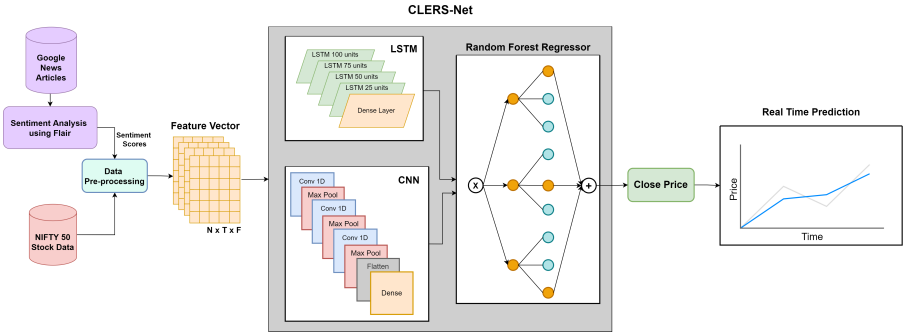


Fig. 1 Architecture Diagram

The overall architecture of our proposed system is shown in Figure 1 and it comprises of collecting the news articles and stock data of the companies, model building and making predictions. News articles about the NIFTY 50 companies are collected from Google News and pre-processed by removing special characters. The pre-trained Flair NLP model is used to perform sentiment analysis on the pre-processed text data to obtain sentiment scores for all the companies. The historical stock price data is collected from Kaggle. The numeric data is pre-processed and merged with the sentiment scores obtained from Flair model. The final dataset is converted into a feature vector and fed into the proposed CLERS-NET model. The model is evaluated with real time data which are obtained using NSEpy library and the closing price of the

stock is predicted for the next seven days. The predictions are visualized using Streamlit.

4.1 Sentiment Analysis

In natural language processing (NLP), neural networks do not operate on words or sentences, but on their numerical representation. The process of converting texts into their numerical form is known as word embeddings which is one of the key elements for sentiment analysis. Flair is a state-of-the-art NLP framework built on PyTorch for sentiment analysis. It comprises of popular state-of-the-art word embeddings, such as GloVe, BERT, ELMo, Character Embeddings which uses contextual word embeddings that capture latent syntactic-semantic data that goes beyond standard word embedding.

Flair uses a character-level LSTM neural network for sentiment classification, which takes the sequences of letters and words into account when predicting sentiments. Since the dataset is unlabelled, a pre-trained Flair model is used to perform sentiment analysis. The pre-trained Flair model classifies the news articles into positive or negative along with a sentiment score ranging from 0 to 1.

4.2 Data Preprocessing

The stock prices for NIFTY-50 companies range from 200 to 20,000 and hence the prices are normalized using min-max normalization. For every feature x_i,

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Once the dataset is normalized, it is converted into a feature vector as shown in Algorithm 2 and fed into the proposed CLERS-Net model. Now shape of X is (batch size, number of timestamps, number of indicators) and shape of Y is (batch size,).

Algorithm 2 Convert features to vectors

Require: *Normalizeddf* //Dataframe after values are normalized

days ⇐ 30
dfscaled ⇐ *Normalizeddf*
$X = []$ //X contains all feature vectors for previous 30 days
$y = []$ //y contains the label, i.e closing price
while $days \leq len(dfscaled)$ **do**
    $i \Leftarrow days$
    X.append($dfscaled[i - days: i, :]$)
    y.append($dfscaled$[i, 5]) // close price is the 5th column.
    $i = i + 1$
end while

Table 1 List of Features

Feature Index	Name
0	Prev Close
1	Open
2	High
3	Low
4	Last
5	Close
6	VWAP
7	Volume
8	Turnover
9	Trades
10	Deliverable Volume
11	%Deliverable
12	Company_Code
13	score

4.3 CLERS-NET Architecture

CNN and LSTM Ensembled using Random Forest Regressor with Sentiment analysis (CLERS-Net) is the proposed architecture that builds a model by ensembling CNN and LSTM with Random Forest Regressor. The sentiment scores of the news articles, along with the stock related numerical data are formed as feature vector and fed as input to the ensembled architecture. It consists of three components namely

- LSTM
- CNN
- Random Forest Regressor

4.3.1 Long Short Term Memory Cells(LSTM)

In recent years, LSTM has played a crucial role in time series predictions. LSTM is an advanced version of RNN which was designed to remember long term dependencies. Recurrent neural networks (RNN) have the ability to store previous information and use it for processing current input. The drawback of RNN is that they can not store long term dependencies due to vanishing gradient. Just like RNN, LSTM has a hidden state (H_t) for short term memory. It also has a Cell state which is known as long term memory. The LSTM consists of three parts, and each part performs an individual function. The first part is the Forget gate, the second part is the Input gate and the last one is the Output gate.

In a cell of the LSTM network, the forget gate chooses to store or discard the information received from previous timestamp.

$$f_t = \sigma(x_t * u_f + H_{t-1} * w_f)$$
$$C_{t-1} * f_t = 0 \quad if f_t = 0 \; [Forget\ everything]$$

$$C_{t-1} * f_t = C_{t-1} \quad if f_t = 1 \; [Forget \; nothing] \tag{2}$$

Input gate adds important information acquired from the current timestamp. The equation of the input gate is

$$\begin{aligned} i_t &= \sigma(x_t * u_i + H_{t-1} * w_i) \\ N_t &= \tanh(x_t * u_c + H_{t-1} * w_c) \; [New \; information] \\ C_t &= f_t * C_{t-1} + i_t * N_t \; [Update \; cell \; state] \end{aligned} \tag{3}$$

Output gate yields the final output of a particular LSTM cell. The equation of the output gate is,

$$\begin{aligned} o_t &= \sigma(x_t * u_o + H_{t-1} * w_o) \\ H_t &= o_t * \tanh(C_t) \\ output &= softmax(H_t) \end{aligned} \tag{4}$$

where,

- u_f, u_i, u_o: Weight associated with the input.
- H_{t-1}: Hidden state of the previous timestamp
- H_t: Hidden state of current timestamp
- w_i, w_f, w_o: Weight matrix associated with hidden state
- C_{t-1}: Cell state of previous timestamp.
- C_t: Cell state of current timestamp.
- N_t: New information
- σ : sigmoid activation function [0,1]
- tanh : tanh activation function [0,1]

Figure 2 is a more intuitive diagram of the LSTM cell.

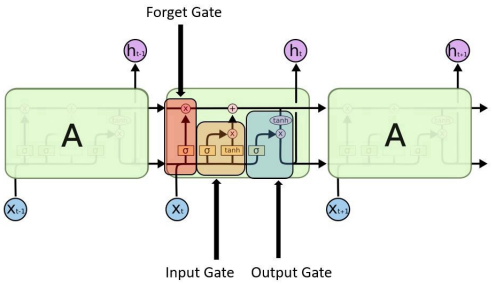


Fig. 2 LSTM cell diagram [21]

In this work, we have proposed a 4-layered stacked LSTM model. Each LSTM layer is followed by a Dropout layer with a dropout rate 0.1. The dropout technique is used to randomly drop selected neurons during training to avoid overfitting. The first LSTM layer consists of 100 LSTM units. The feature vector containing stock prices of the past 30 days is fed into each of the 100 cells simultaneously. Each cell in this layer gives an output for each timestamp which is fed into the dropout layer. Similarly, the output is passed to the second layer which consists of 75 LSTM cells followed by the third layer with 50 cells and the fourth layer 25. Each cell in the fourth layer gives one output. After the final dropout layer, a Dense layer is added to produce a single dimensional output. We define the loss function as MSE and use Adam optimizer for efficient predictions. Figure 3 gives a complete view of our LSTM architecture.

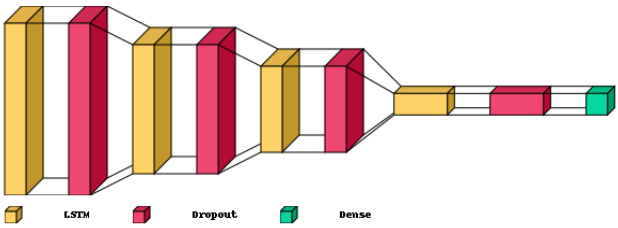


Fig. 3 LSTM Architecture

4.3.2 1D Convolution Neural Network

One-dimensional Convolutional Neural Network is a CNN model which has a convolutional hidden layer that operates over a single dimensional sequence. So in recent years, 1D CNNs have been used for time series data. A CNN has 3 main Layers: Convolutional layer, Pooling layer and Flatten layer.

Convolutional Layer

Convolutional layers are the major building block of CNNs. Convolution is the process of merging input vector with the filter to produce a feature map. A filter is a 2-D array of weights. A dot product is applied between a filter-sized patch of the input and the filter. A filter smaller than the input is used so that the same set of weights can be multiplied by the input array many times at different points on the input. The filter is applied across the input data, left to right, top to bottom. The stride(s) decides the distance filter moves over the input sequence. Padding is used to ensure that input matrix and the output matrix are of the same size. So, given the number of input features (n_{in}), padding size (p), stride (s), kernel size (k), number of output features can be calculated using the formula

$$n_{out} = \frac{n_{in} + 2p - k}{s} + 1 \tag{5}$$

Pooling Layer
A Pooling layer generally follows the convolutional layer to reduce the dimensionality of the convolution output. It higlights the notable features of the convolution output and plays a major role in reducing overfitting. We've used maxpooling which picks a window of size f and slides it over the input with stride s. For each pooling operation, the cell with the maximum pixel value is chosen and highlighted in the feature map.

Flatten
Flatten layer converts the 2-D output of the pooling layer into a 1-D array. Generally this flatten layer output is inserted into an artificial neural network later on. But since in our case we're using it for time series prediction, our flatten layer output is passed into a dense layer which produces a single output (in our case, next day's close price).

In our CNN component, we have 8 layers. There are 3 1D Convolutional layers which are followed by 3 Max pooling layers respectively. The first Conv1D layer consists of 16 filters each having a kernel size 3. Same padding is used so that the output layer of each conv1D has the same size as their input layer. The second and third conv1D layers have 32 and 64 filters respectively. Default pooling size of 2 is used for all the 1D pooling layers. The activation function used is Relu.

$$R(z) = \max(0, z) \tag{6}$$

The output from the final pooling layer is passed through a flatten layer to make it one dimensional and then passed through a dense layer which returns the predicted price. The model uses Adam optimizer a gradient descent technique for efficient predictions. The loss function is defined as mean squared error (MSE) during compilation. Figure 4 gives an overview of our CNN component.

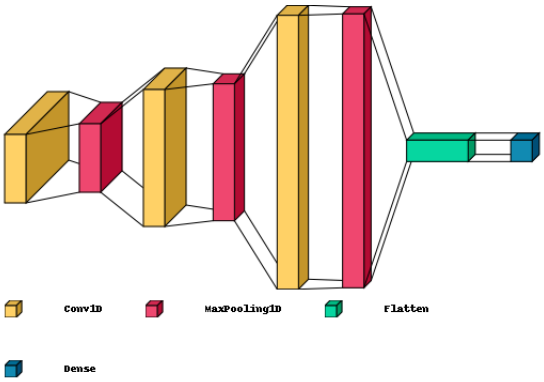


Fig. 4 CNN architecture

4.3.3 Random Forest Regression

Random forest is an ensemble Machine Learning technique which can solve both classification and regression problems. This algorithm constructs a multitude of decision trees during the training time and uses a statistical technique called **Bootstrap Aggregation**, also called as **Bagging**.

Bootstrap Aggregation
Bootstrapping is a statistical re-sampling technique to infer results for a dataset from results found on a collection of smaller random samples of that dataset, using replacement during the sampling process.

Bootstrap Aggregation is a powerful ensemble technique that combines the outputs from multiple models together to improve the accuracy of predictions compared to the individual models. Bagging in Random Forest entails training each decision tree on a different data sample. Here sampling is done with replacement. This technique is a general procedure to reduce the variance of high variance models.

Working
Figure 5 shows the architecture of Random Forest Regression. Random forests follow the wisdom of the crowds concept. Here combining predictions from a number of uncorrelated or weakly correlated models working together as a group will outperform any of the individual models. The number of trees in the forest is specified by the n_estimators argument. The trees are built following the specified hyperparameters like maximum depth of a tree, minimum number of samples to be a leaf node, etc. Each of the decision tree in the forest is trained by a random sample of the training set. The samples are drawn using the bootstrapping method. Each tree considers all attributes and values and determines which attribute would lead to the best split. This process is repeated iteratively until it reaches terminal nodes or a specified stopping criteria (eg. max depth).

The main idea behind Random Forest is to combine decisions from many trees to determine the output instead of depending on a single decision tree. A single decision tree can easily overfit and have high variance. This might lead to high test errors. Random forest algorithm randomly selects subsets of features for splitting. This prevents multiple decision trees to rely on the same set of features and produces uncorrelated trees. Overall, the entire forest will have lower variance and hence will make more accurate predictions.

The proposed CLERS-Net model uses a Random Forest Regressor as the final estimator. The predictions from the LSTM network and CNN model are normalized and zipped together to form a n * 2 vector. This vector is fed into the Random forest regressor as input and the final output is obtained.

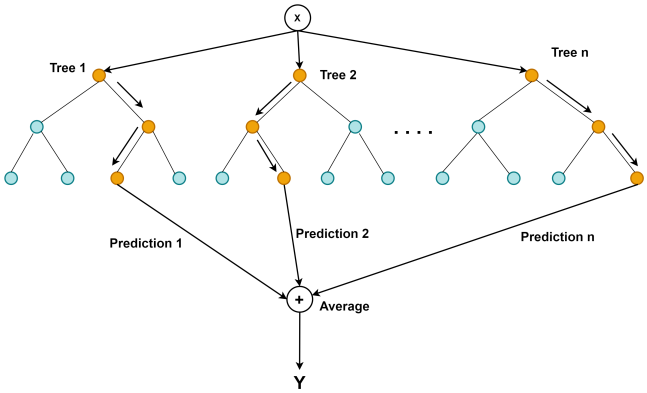


Fig. 5 Random Forest Regression Architecture

5 Performance Analysis

5.1 Performance Metrics

The 3 main performance metrics used for evaluating our model are:

- **RMSE:** Root mean squared error is the square root of the mean of the square of all of the error

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(x - x_i)^2} \tag{7}$$

where,
n = number of non-missing data points
x_i = actual observations time series
x = estimated time series

- **MAE:** Mean absolute error is the average of the difference between the predicted value and the true value

$$MAE = \frac{\Sigma_{i=1}^{n} y_i - x_i}{n} \tag{8}$$

where,
y_i = predicted value
x_i = true value
n = total number of data points

- **MAPE:** Mean absolute percentage error measures the accuracy of the forecast. It is the average absolute percent error for each time period minus actual values divided by actual values

$$MAPE = \frac{100}{n} \Sigma_{i=1}^{n} \frac{y_i - x_i}{x_i} \tag{9}$$

where
y_i = prediction
x_i = true value
n = total number of data points

Since our data ranges from 200 to 20,000, MAPE is considered as the premier metric. For time series data, R^2 is mostly inflated, hence we haven't considered R^2 score.

5.2 Model Testing

Our model was trained and validated using 2011-2020 data and tested using 2021 data for NIFTY-50 companies. As seen in Table 2, Both LSTM and CNN were trained for 50 epochs. Upon tuning the batch size and timestamps to 64 and 30 respectively, the model produced best results. Early stopping callback is enabled to stop training if there is no change in the loss function for 3 epochs. The number of estimators for random forest regression is 100.

Table 2 Hyper-Parameter Tuning

Hyper-parameter	**Value**
Epochs	50
Timestamps	30
Batch Size	64
Optimizer	Adam
Loss function	Mean Squared Error
callback	Early Stopping
n-estimators	100

Mean squared error for LSTM component is plotted during training. As seen from Figure 6, training stops after 18 epochs because of early stopping. We can see a gradual decrease in loss for every iteration. Similarly Figure 7 shows the mean squared error plot for CNN component. CNN is trained for 21 epochs and it is seen that CNN with a loss of 0.00002 performs better than LSTM which has a loss of 0.00010. After the prices are inverse transformed, CNN has a lower MAE compared to LSTM.

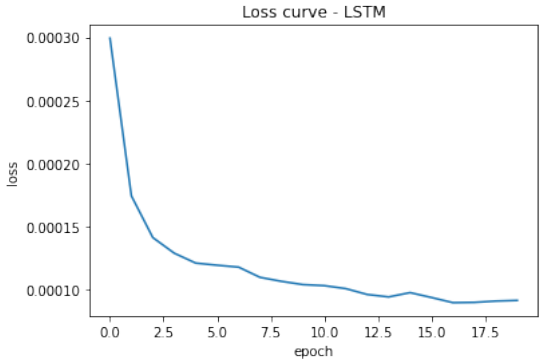


Fig. 6 LSTM loss

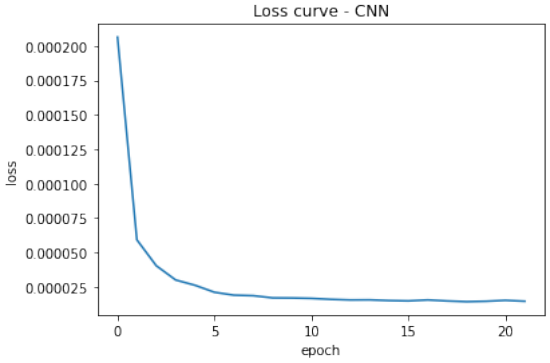


Fig. 7 CNN loss

The output from the first level estimators CNN and LSTM are fed to the random forest regressor which predicts the next day's closing price. The predictions are inverse transformed and absolute error for 2021 unseen data is plotted in descending order. If we analyze figure 8, it is clear that absolute error is less than 30 for more than 80% of the records. The mean absolute error is 29.73. The stock prices in our dataset range from 200 to 30,000. The mean stock price of the dataset is 1523.45. So an MAE of 29.73 is negligible considering our data range.

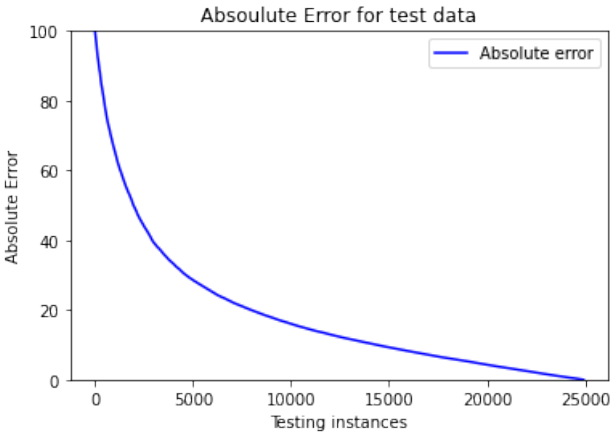


Fig. 8 Mean absolute error

5.3 Ablation Study

Our model was trained and validated using 2011-2020 data and tested using 2021 data for NIFTY-50 companies. The experiment was conducted for 4 models namely

- LSTM
- CNN
- LSTM + CNN
- CLERS-Net

5.3.1 LSTM

The model performs reasonably well when we consider only the LSTM component. As seen in table 3, LSTM has an MAPE of 6% and MAE of 65 which is genuinely good.

Table 3 LSTM Results

Metric	Result
RMSE	210.45
MAE	65.77
MAPE	6.64

5.3.2 CNN

The 1-D CNN component outperforms the LSTM component. This is mainly because convolutions consider spatial structure of the data. And a time coordinate works just as well as a spatial coordinate, which is why CNN produces an MAPE of 4.03% as seen in Table 4.

Table 4 CNN Results

Metric	Result
RMSE	101.23
MAE	50.04
MAPE	4.03

5.3.3 CNN + LSTM

In this model, we combine both LSTM and CNN output using ensemble learning. The outputs are passed to random forest regressor which in its turn gives the final combined output. As seen in table 5, this model produces an MAPE of 3.35% and MAE of 37 which is nearly a 2% increase over LSTM.

Table 5 Random Forest Regressor Results

Metric	Result
RMSE	89.23
MAE	37.28
MAPE	3.35

5.3.4 CLERS-Net

CNN and LSTM Ensembled using Random Forest Reggressor with Sentiments analysis(CLERS-Net) is our final model which performs exceedingly well with an MAPE of 2% and MAE of 29.7 as shown in table 6. This shows that sentiments play a very crucial role in predicting stock price. All small variations that might happen on a day are captured when we add sentiments to our model.

Table 6 CLERS-Net Results

Metric	Result
RMSE	80.45
MAE	29.73
MAPE	2.03

5.4 Visualizing Predictions

The errors from different models are plotted in figure 9. It is clearly seen that with every new component the MAE and MAPE decreases gradually. When we compare LSTM and LSTM with sentiments(LSTM-S) MAE reduces by 17 and there's a 1% drop in MAPE. Similarly there is a significant drop in scores when we compare CNN with CNN-S, which indicates that sentiments have improved the model performance immensely. When we compare CNN & LSTM, 1D CNN performs better because of its ability to capture spacial

structure of the data. So, combining all the components, CLERS-Net performs best with an MAPE of 2.03%.

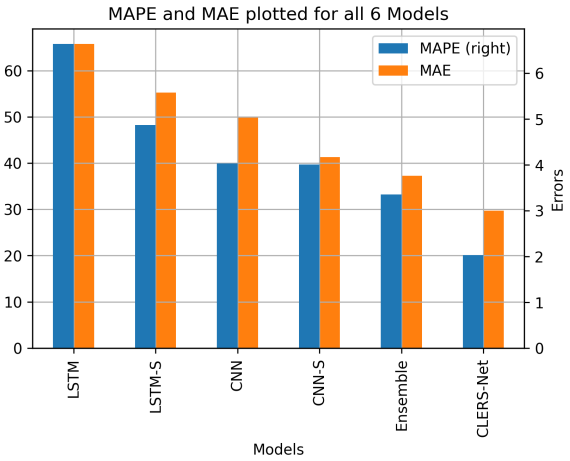


Fig. 9 Errors plotted

In this experiment, prediction graphs for Britannia were plotted to analyze the different models. Figure 10 shows LSTM predictions for Britannia in the year 2021. From the graph, it is seen that there is high difference between actual and predicted price. This can be correlated to a comparatively higher MAE of 63.

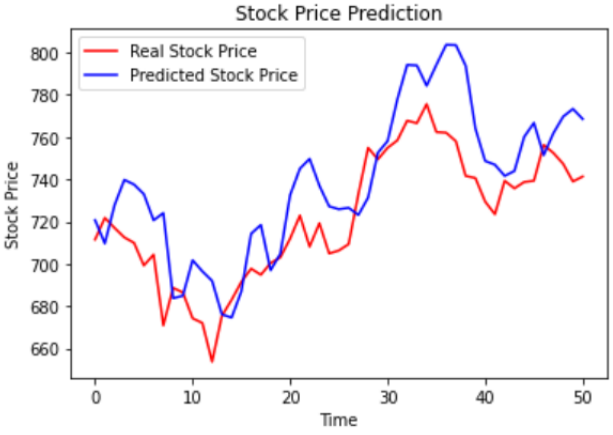


Fig. 10 Britannia LSTM Output

Figure 11 shows CNN predictions for Britannia in the year 2021. Even though the predictions are better than LSTM, there are some variations seen in the graph. Figure 12 shows CNN+LSTM ensemble predictions in the year 2021. This graph looks more accurate and there are very less differences between the predicted and actual price.

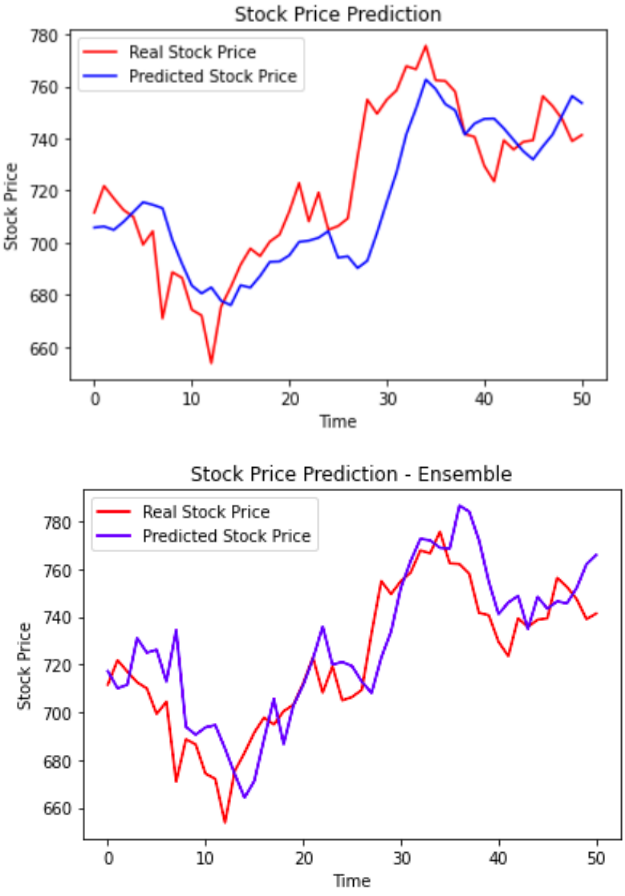


Fig. 12 Britannia Ensemble output

Figure 13 shows CLERS-Net model prediction for 2021. From LSTM to CLERS-Net we have seen a gradual improvement in our predictions and CLERS-Net outperforms all other models with a very low MAE of 29. Considering the range of the dataset (200 - 30, 000), an MAE of 29 is very less. Our model has produced a mere 2% error and can be clearly relied on to buy NIFTY-50 stocks.

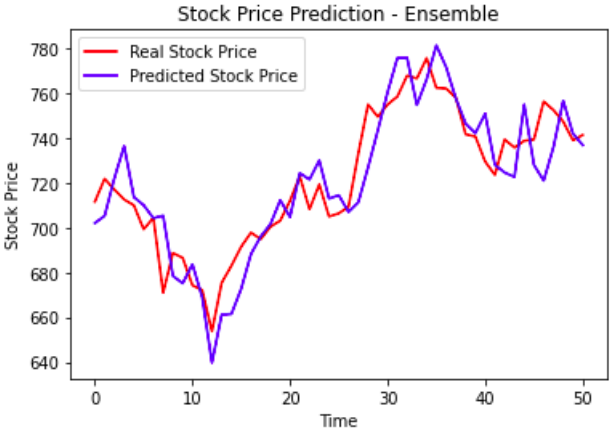


Fig. 13 Britannia CLERS-Net Output

6 Effect of COVID on stock market

The global pandemic of COVID-19 caused by SARS-CoV-2 was first identified from an outbreak in Wuhan, China in November 2019. This deadly virus has affected more than 527 million people across the world and has taken the lives of 6.28 million people. Previous contagions like SARS, Influenza, etc. have caused similar impacts. Pandemics cause long-term economic ramifications on the nations around the world. COVID-19 has negatively impacted various sectors of industries in the past two years. As many transport services, shops, industries were shuttered during the lockdown, the Indian economy took a devastating downturn. The surveys conducted by CMIE shows that there has been a sheer rise in the unemployment rates during the April-June quarter of 2021. The nation's Gross Domestic Product contracted by 7.3% in 2021-2022. This contraction could be above 8% if informal sectors are taken into account. This is by far the worst economic performance the nation has seen since 1947.

The economic disruption due to COVID-19 has had severe impacts on stock markets, crude oil and gold markets globally. The MSCI world index which consists of stocks from well developed companies and emerging markets, lost 10.7% of its value between January and March 2020. The Indian stock market was also drastically affected. When the nationwide lockdown was announced in March, 2020, the productivity of industries reduced, increasing their cost of operation and decreasing their revenue. Such an unprecedented change caused by the pandemic led to market abnormality. The Indian stock indices took a big hit with values down by 40%. The Indian markets crashed to levels which haven't been witnessed since the Global Financial Crisis of 2008. The BSE Sensex and NIFTY 50 indices dropped by 38%.

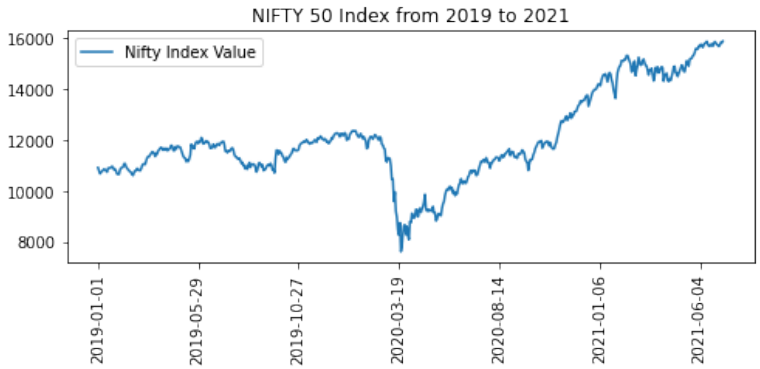


Fig. 14 Nifty Index from 2019 to 2021

The Figure 14, shows the everyday closing price of the NIFTY 50 index. This value is calculated by taking the last half an hour weighted average closing prices of the companies present in the index. As can be seen from the graph, there is a steep decline in the index values from March 2020, which is when the nationwide lockdown was first announced. Profits of Nifty 50 index members fell about 15% in that quarter. More then 90% of the Nifty index companies have hit their multi-year low in the midst of the volatility caused by the pandemic.

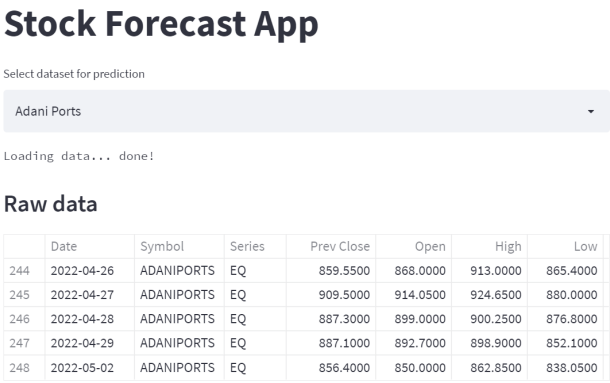
7 Real time stock prediction

Real time in finance markets refers to the instantaneous price of a security which is crucial for investors. NSEpy is a python library that allows us to extract historical and real time stock price from NSE's website. NSEpy aims to provide analysis ready data-series which can be obtained by passing the company's code, start date and end date. Prices upto the current date can be obtained using this library. The gethistory() function from NSEpy returns 15 features including Open, Close, High, Low, etc.

Streamlit is an open-source Python framework used for developing web apps for data science and machine learning. This framework is extremely simple to use and allows us to develop and deploy web apps in a short period of time. Streamlit supports a number of popular Python libraries used in data science like Keras, Matplotlib, NumPy, etc. Streamlit allows us to visualize the results in an interactive manner using the web app.

In our work, real time stock price prediction is performed using NSEpy and the predictions are visualized using Streamlit. NSEpy is used to fetch the real time stock prices for the last 30 days for the selected company. The trained model is tested using this real time data to predict the closing price of the company for the next 7 days. The close price of the first day is predicted using the collected 30 days data. The predicted price of the first day is appended to the data of the previous 29 days to predict the close price of the second day

and so on up to the seventh day. The predictions are plotted in an interactive manner using Streamlit.



	Date	Symbol	Series	Prev Close	Open	High	Low
244	2022-04-26	ADANIPORTS	EQ	859.5500	868.0000	913.0000	865.4000
245	2022-04-27	ADANIPORTS	EQ	909.5000	914.0500	924.6500	880.0000
246	2022-04-28	ADANIPORTS	EQ	887.3000	899.0000	900.2500	876.8000
247	2022-04-29	ADANIPORTS	EQ	887.1000	892.7000	898.9000	852.1000
248	2022-05-02	ADANIPORTS	EQ	856.4000	850.0000	862.8500	838.0500

Fig. 15 Input: Real time stock price for Adani Ports

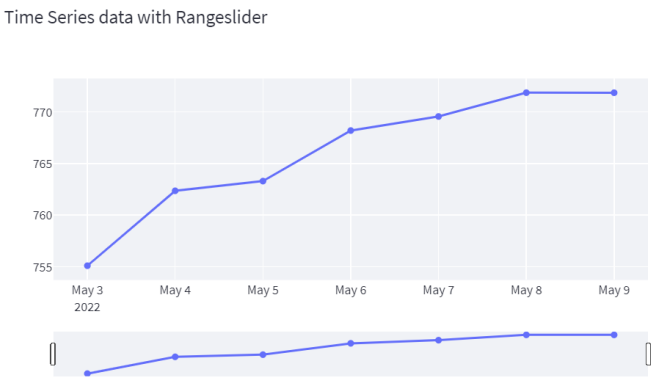


Fig. 16 Output: 7 days close price prediction for Adani Ports

8 Conclusion and future work

We have proposed a methodical approach to solve the stock price prediction problem using ensemble learning and sentiment analysis. Our proposition includes 4 models, LSTM, CNN, Random Forest Regressor and the final CLERS-Net model. This research uses state of the art NLP technique, Flair to perform sentiment analysis on the collected news articles related to individual stocks. Our final CLERS-Net model makes use of two Deep Learning models,

LSTM and CNN as the first level estimators and a Random Forest Regressor as the final estimator.

The sentiment scores obtained for each stock are appended to the historical data and a 30-day sliding window mechanism is applied to obtain the feature vector for each instance. This vector is passed to both LSTM and CNN, the predictions of those models are combined and piped to the Random forest regressor. The predictions of this CLERS-Net model was observed to be more accurate than the individual models. The experimental results found that integrating sentiment scores of stock related news with numeric stock data can reduce the overall percentage error of the CLERS-Net model. Adding sentiment scores to the feature vector reduced the mean absolute percentage error from 3.35% to 2.03%. Therefore, it can be inferred that public emotion plays an important role in the stock market movement.

In future works, risk analysis factor can be incorporated. Stock prices can be highly volatile, some companies tend to have highly fluctuating stocks while others show a steady pattern. This volatility factor can be analyzed and integrated. The risk factor associated with a particular stock can be determined by measuring the volatility of the stock. The risk factor when presented to the investors will help them make much safer investments.

References

[1] Selvin S, Vinayakumar R, Gopalakrishnan EA, Menon VK, and Soman KP (2017) *Stock price prediction using LSTM, RNN and CNN - sliding window model* , International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India. https://doi.org/10.1109/ICACCI.2017.8126078

[2] Zheng A, and Jin J (2017) *Using AI to make predictions on Stock Market*, Stanford University, Stanford, CA.

[3] Pramod BS, and Shastry M (2020) *Stock Price Prediction Using LSTM.*

[4] Gaurav UP, and Sirbi K (2020) *Impact of COVID 19 on Stock Market Performance using Efficient and Predictive LBL-LSTM based Mathematical Model* , International Journal on Emerging Technologies11 (4), 108-115.

[5] Ding G, and Qin L (2020) *Study on the prediction of stock price based on the associated network model of LSTM*, International Journal of Machine Learning and Cybernetics, 11(6), 1307-1317.

[6] Kalyani J, Bharathi HN, and Jyothi R (2016) *Stock Trend Prediction Using News Sentiment Analysis* , arXiv:1607.01958 https://doi.org/10.48550/arXiv.1607.01958 .

[7] Mohan S, Mullapudi S, Sammeta S, Vijayvergia P and Anastasiu CD (2019) *Stock Price Prediction Using News Sentiment Analysis* , IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), https://doi.org/10.1109/BigDataService.2019.00035.

[8] Bharadwaj A, Narayan Y, Dutta M, Vanraj, Pawan (2015) *Sentiment Analysis for Indian Stock Market Prediction Using Sensex and Nifty* , Procedia computer science, 70, 85-91, https://doi.org/10.1016/j.procs.2015.10.043.

[9] Kordonis J, Symeonidis S and Arampatzis A (2016) *Stock Price Forecasting via Sentiment Analysis on Twitter* , Proceedings of the 20th Pan-Hellenic Conference on Informatics, https://doi.org/10.1145/3003733.3003787 .

[10] Mittal A and Goel A (2011) *Stock Prediction Using Twitter Sentiment Analysis* , Standford University, CS229 .

[11] Panday V, Vijayarajan V, Manhendran A, Krishnamoorthy A and Surya VB (2020) *Stock Prediction using Sentiment analysis and Long Short Term Memory* , European Journal of Molecular & Clinical Medicine, 7(2), 5060-5069.

[12] Cheng LC, Huang YH, and Wu ME (2018)*Applied attention-based LSTM neural networks in stock prediction*, IEEE International Conference on Big Data (Big Data) (pp. 4716-4718). IEEE, https://doi.org/10.1109/BigData.2018.8622541.

[13] Ho TT and Huang Y (2021) *Stock Price Movement Prediction Using Sentiment Analysis and CandleStick Chart Representation* , Sensors 21(23), 7957, https://doi.org/10.3390/s21237957.

[14] Gaurav UP, and Sirbi K (2020) *Lbl - Lstm : Log Bilinear And Long Short Term Memory Based Efficient Stock Forecasting Model Considering External Fluctuating Factor* , International Journal of Engineering and Advanced Technology (IJEAT), ISSN, 2249-8958.

[15] Pasupulety U, Anees AA and Anmol S, Mohan R (2019) *Predicting Stock prices using Ensemble Learning and Sentiment Analysis*, IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE) (pp. 215-222). IEEE, https://doi.org/10.1109/AIKE.2019.00045.

[16] Mehtab S and Sen J (2020)*Stock Price Prediction Using CNN and LSTM-Based Deep Learning Models*, International Conference on Decision Aid Sciences and Application (DASA) (pp. 447-453). IEEE, https://doi.org/

10.1109/DASA51403.2020.9317207.

[17] Mehtab S and Sen J (2020) *Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Timeseries*, arXiv preprint arXiv:2001.09769, https://doi.org/10.48550/arXiv.2001.09769.

[18] Nunno, L (2014) *Stock Market Price Prediction Using Linear and Polynomial Regression Models* , Computer Science Department, University of New Mexico: Albuquerque, NM, USA.

[19] Staudemeyer RC and Morris E (2019) *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks* , arXiv preprint arXiv:1909.09586.

[20] Jayanth K (2016) *Understanding Convolutional Neural Networks* , arXiv preprint arXiv:1605.09081, https://doi.org/10.48550/arXiv.1605.09081.

[21] *https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm*