Bitcoin Price Prediction Using Machine Learning

A Project Report

Submitted by

Arindam Sarkar(24MA60R26)

Mrinmoy Mondal(24MA60R08)

Greeshma Thadana(21MA25007)

Rohan Kharwar(24MA60R06)

Sougata Rana(24MA60R07)

Manoj Kumar Bag(24MA60R31)

For the Course Artificial Intelligence and Machine Learning

Department Of Mathematics

Indian Of Institute Technology Kharagpur West Bengal-721302, India

Table of Contents

Abstract:

Bitcoin price prediction is a challenging task due to its high volatility and non-linear price movements. In this study, we compare the effectiveness of **Convolutional Neural Networks (CNN)** and **Long Short-Term Memory (LSTM)** networks for forecasting Bitcoin prices. While CNNs are efficient at capturing local temporal patterns through convolutional layers, LSTMs are designed to handle long-term dependencies in sequential data using memory cells. We preprocess Bitcoin historical price data and extract key features, including Open, High, Low, Close, and Volume, before normalizing and structuring the dataset into time series sequences. The CNN model utilizes 1D convolutional layers to identify short-term price trends, whereas the LSTM model leverages recurrent units to learn long-term dependencies. Our experimental results show that **LSTM outperforms CNN** in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE),with Accuracy of approximately 98% over CNN's 95%, indicating its superior ability to model sequential dependencies in Bitcoin price data. However, CNN provides faster training times and is computationally less expensive. The findings suggest that while LSTM is better suited for accurate long-term Bitcoin price predictions, CNNs may be preferable for real-time or short-term trend analysis.

1 Introduction

In recent years, stock price prediction has gained significant attention due to its potential to maximize profits and minimize risks in financial markets. Accurate forecasting of stock prices is a complex task, as financial time series data exhibit high volatility, non-linearity, and intricate dependencies over time. Machine learning techniques, such as Long Short-Term Memory (LSTM) networks, have demonstrated their effectiveness in capturing long-range dependencies and patterns in sequential financial data.

This study explores the implementation of an LSTM-based model for stock price prediction, focusing on major global stock markets. The model leverages historical stock price data to predict future price movements, aiming to improve the accuracy of financial forecasting. The predicted stock prices are evaluated using performance metrics such as Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), which provide insights into the model's effectiveness.

Furthermore, the predictive performance of the LSTM model is compared against the Random Forest model, a widely used machine learning algorithm in financial forecasting. This comparative analysis helps assess the strengths and limitations of deep learning-based approaches versus traditional machine learning techniques. By employing LSTM for stock price forecasting, this study aims to contribute to the growing body of research on machine learning applications in financial markets, providing investors and analysts with more reliable tools for decision-making.

2 Some Keywords

Investor: An individual or entity that allocates capital with the expectation of receiving financial gains, typically through investments in assets such as stocks, bonds, or real estate etc.

Return : The profit or loss generated by an investment over a specific period, usually expressed as a percentage of the initial investment.

Risk: The potential for loss or uncertainty in investment returns, often measured by volatility of assets or by some variance function of return of asset.

Portfolio: A collection of financial assets, such as stocks, bonds, and cash equivalents, held by an individual or institution to meet investment goals.

3 Neural networks

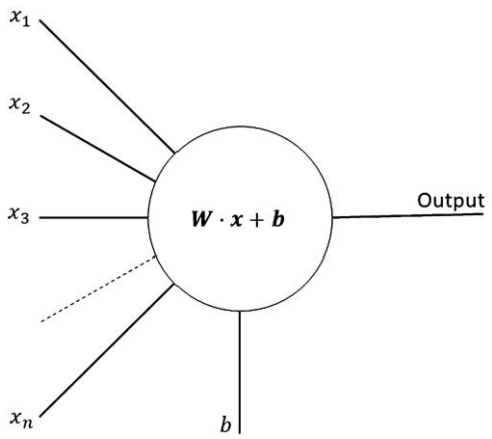


Figure 1: Schematic version of neuron.[2]

A neuron is a nonlinearity applied to an affine function.

The input features $x = (x1 , x2 , \ldots , xn)$ are passed through an affine function composed with a non- linearity φ:

$$T(x)=\varphi(\textstyle\sum_i W_i x + b_i) = \varphi(\boldsymbol{W}.\boldsymbol{x} + \mathbf{b})$$

with given *weights* $\boldsymbol{W}$ and *bias* $\boldsymbol{b}$.

A neural network can be modeled as a collection of neurons which are connected in an acyclic graph. That is, the output of some of the neurons become inputs to other neurons, and cycles where the output of a neuron maps back to an earlier intermediate input are forbidden. Commonly such neurons are organized in layers of neurons. Such a network consists of an input layer,

one or more *hidden layers* (a layer consisting of multiple perceptrons ,also called, Multi Linear Perceptron), and an output layer.

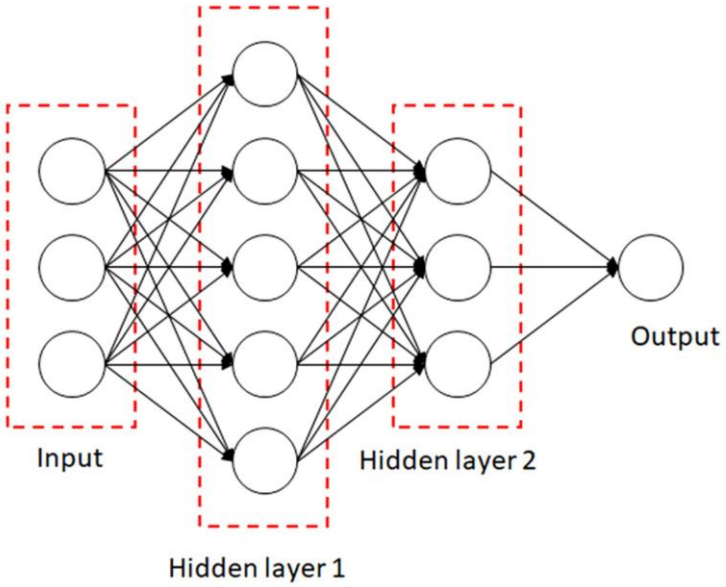


Figure 2: A 3-layer neural network with three inputs , two hidden layers of respectively 5 and 3 neurons, and one output layer[2]

To use a neural network for prediction, we need to find the proper values for the parameters (W,b) and define a function to map the output of the neural network to a prediction .

In order to assign the weight to each input of a perceptron, a backpropagation algorithm is used. This algorithm measures the network's output error (i.e., it uses a loss function that compares the desired output and the actual output of the network, and returns some measure of the error). Then it computes how much each output connection contributed to the error. This is done analytically by simply applying the *chain rule* (perhaps the most fundamental rule in calculus), which makes this step fast and precise. The algorithm then measures how much of these error contributions came from each connection in the layer below, again using the chain rule and so on until the algorithm reaches the input layer. As we explained earlier, this reverse pass efficiently measures the error gradient across all the connection weights in the network by propagating the error gradient backward through the network. Finally, the algorithm performs

a Gradient Descent step to tweak all the connection weights in the network, using the error gradients it just computed.

3.1 CNN

Convolutional neural networks (CNNs), or convnets for short, are a special case of feedforward neural networks. They are very similar to the neural networks. CNN architecture makes the implicit assumption that the inputs are grid structured, which allows us to encode certain properties in the architecture.

A convolution neural network consists of a sequence of layers, where every layer transforms the activations or outputs of the previous layer through another differentiable function. There are several such layers employed in CNNs. In our model we have included 1-D convolution layer, Pooling layer and Flattening layer. These layers are like feature extractors, dimensionality reduction

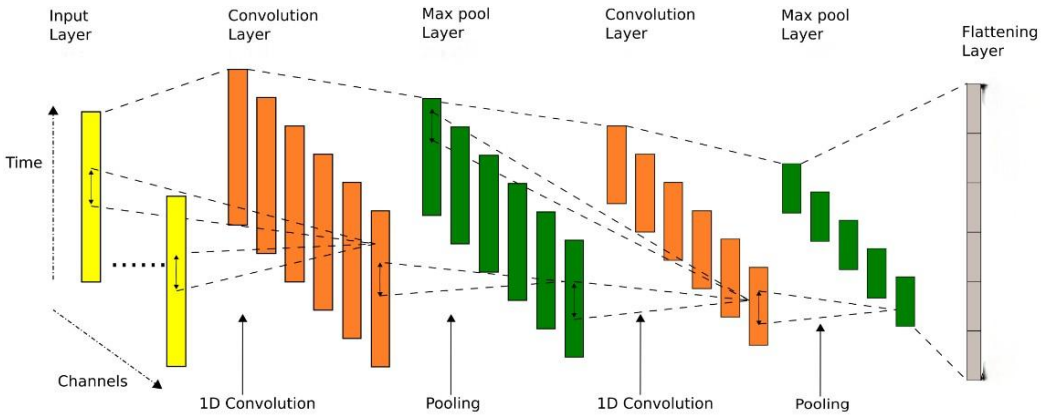


Figure : *Flowchart of the multi-channel CNN for time-series modeling.*[3]

3.2 Convolution layer

Convolutional layer is the most important component of any CNN architecture. It contains a set of convolutional kernels (also called filters), which gets convolved with the input data (N-dimensional metrics) to generate an output feature map. During a forward pass, a filter slides across the input volume and computes the activation map of the filter at that point by computing the dot product of each value and adding these to obtain the activation at the point. Such a sliding filter is naturally implemented by a convolution and, as this is a

linear operator, it can be written as a dot-product Such a sliding filter is naturally implemented by a *convolution* and, as this is a linear operator, it can be written as a dot-product for efficient implementation.
For example in Figure 4, there is an input volume and a filter of size 3X1 which is implemented on the input layer through dot-product and the result is being stored in a new array. After each iteration the filter slides through 1 step.

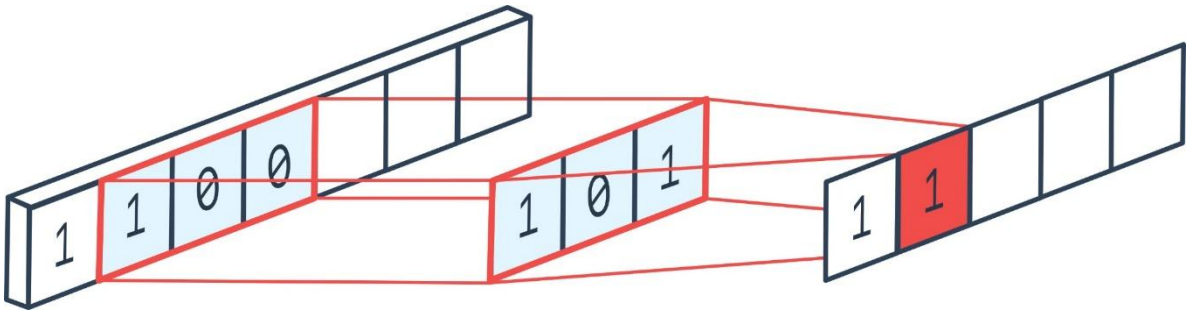


Figure 4: Dot-product between 1-D input layer and 3X3 filter

3.3 Pooling Layers

The goal of a pooling layer is to produce a summary statistic of its input and to reduce the spatial dimensions of the feature map. For this the max pooling layer reports the maximal values in each rectangular neighborhood of each point of each input feature.

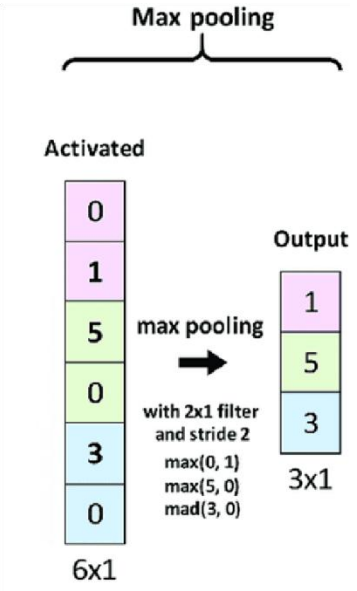


Figure 5: Example of MaxPooling layer with input layer and filter[5]

3.4 Flatten layer

Flattening layer is used to reshape the data from a higher-dimensional tensor into a 1D vector. This is necessary before passing the data to the LSTM because LSTM expects a specific 1D data format It takes the multi-dimensional output of the previous layers (produced Pooling layers) and converts it into a single one-dimensional vector. This step is crucial because LSTM layers require one-dimensional input. By flattening the data, the layer ensures that spatial or temporal patterns extracted by the convolutional layers are preserved in a form that the LSTM layer can process for prediction tasks, enabling the model to capture temporal dependencies across different regions while maintaining the local features extracted from the data.

3.5 Regional-CNN

A Regional CNN (R-CNN) is to handle tasks that require processing regions within an image or sequence. In our case it is applied to extract features from specific time windows or regions of the data. In a Regional-CNN, input data is divided into regional size or windows. These windows can be defined based on specific criteria, such as fixed-size windows or sliding windows. Each region is then fed into a CNN to extract features. CNN learns to identify patterns and characteristics within the region. The extracted features are then used for classification or regression tasks. For example, in stock price prediction, the extracted features can be used to predict the future price of the stock.

R-CNNs can capture local patterns and dependencies within the data, making them suitable for time series analysis where short-term trends and anomalies might be important. R-CNNs have shown improved performance compared to traditional CNNs in certain tasks, especially when dealing with complex spatial relationships. In our model, the input data is divided into overlapping time windows.

Now let us move to LSTM architecture.

4 LSTM

In order to understand the LSTM, first let us see how an RNN (Recurrent Neural Network) works.

4.1 RNN

RNNs are a type of neural network designed to process sequential data, such as time series data, natural language, and audio. Unlike traditional feedforward neural networks, RNNs have feedback connections that allow them to maintain a memory of previous inputs, making them suitable for tasks that require understanding context and sequential dependencies. RNNs are specifically designed to process sequential data, while traditional neural networks are better suited for static data. RNNs have recurrent connections that allow information to persist from previous time steps, enabling them to learn from past data.

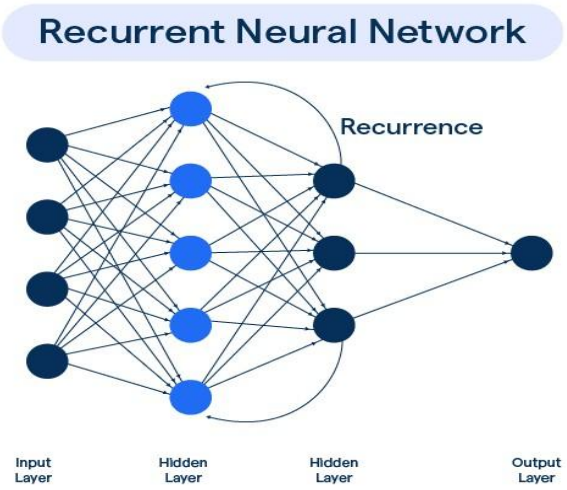


Figure 6: RNN architecture[6]

An RNN receives a sequence of inputs, one at a time and then at each time step, the RNN updates its hidden state based on the current input and the previous hidden state. This hidden state captures the memory of the network as shown in above figure. RNN produces an output at each time step, which can be used for tasks like prediction or classification.

Let's now look how an LSTM is different from a RNN.

4.2 LSTM

LSTM networks are a special type of recurrent neural network (RNN) designed to address the vanishing gradient problem that can occur in traditional RNNs when dealing with long sequences. This problem arises because the gradients of the error signal can become very small or large as they propagate through the network, making it difficult for the network to learn long-term dependencies.

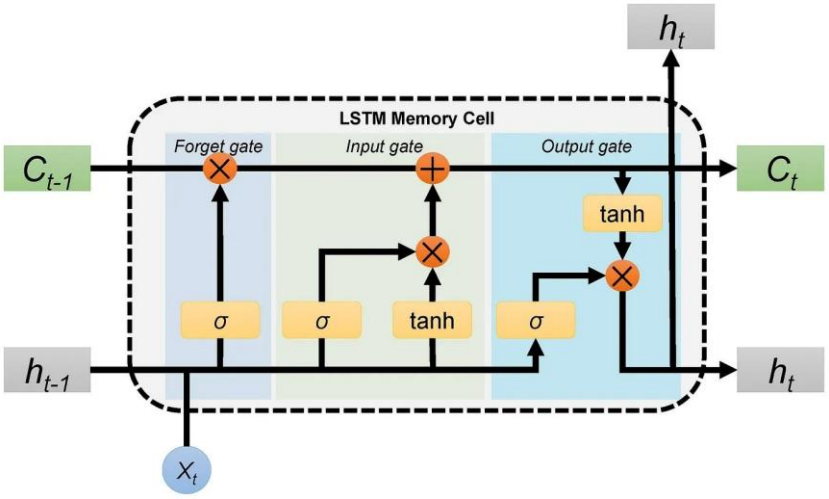


Figure 7: LSTM Architecture[7]

LSTM networks have a unique structure, which includes three fundamental gates:

1. **Forget Gate**: This gate determines what information from the previous cell state should be forgotten or retained.

2. **Input Gate**: It controls what new information should be stored in the cell state.

3. **Output Gate**: This gate defines the output of the LSTM cell, considering the current input and the updated cell state.

LSTMs have a cell state that acts as a memory unit, allowing the network to store information over long periods of time. The gates in LSTMs help to regulate the flow of information, preventing the vanishing gradient problem.

LSTM can effectively capture long-term dependencies in sequential data. LSTMs have been shown to outperform traditional RNNs in tasks that involve long sequences.

5 Methodology

5.1 Data Collection and Preprocessing

- The data used for modelling was collected from the BSE website. This data was collected for the period of 1st January, 2024 to 01st October, 2024 for 8 stocks. These 8 stocks are those which are/were on Sensex index.

Table 1: Stocks details

Ticker	Company Name	Sector
BITCOIN	Bitcoin core	Cryptocurrency
RELIANCE.BO	Reliance Industries Ltd	Oil & Gas, Diversified
TCS.BO	Tata Consultancy Services Ltd	IT Services & Consulting
HDFCBANK.BO	HDFC Bank Ltd	Banking
INFY.BO	Infosys Ltd	IT Services & Consulting
ICICIBANK.BO	ICICI Bank Ltd	Banking
HINDUNILVR.BO	Hindustan Unilever Ltd	Consumer Goods (FMCG)
ITC.BO	ITC Ltd	Consumer Goods (FMCG)
KOTAKBANK.BO	Kotak Mahindra Bank Ltd	Banking
LT.BO	Larsen & Toubro Ltd	Infrastructure & Engineering
SBIN.BO	State Bank of India	Banking
BAJFINANCE.BO	Bajaj Finance Ltd	Financial Services (NBFC)
AXISBANK.BO	Axis Bank Ltd	Banking

Table 2: Timeline of dataset

	Dataset	Training Dataset	Testing Dataset
Time Interval	01/01/2024 12/10/2024	01/01/2024 02/08/2024	03/08/2024 01/10/2024

○ Missing values were handled using linear interpolation.

○ The data was normalized using MinMaxScaler.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

○ Stock prices were segmented into regional windows for model input.

5.2 Model Architecture

- The LSTM model is designed to capture temporal dependencies in stock price data, making it well-suited for time-series forecasting.
- The input stock data is structured as sequential time-series data, which is fed into an LSTM layer to learn long-term dependencies and trends.
- Multiple LSTM layers are stacked to enhance the model's ability to extract meaningful patterns from historical stock prices.
- Dropout layers are incorporated to prevent overfitting and improve generalization.
- A Dense layer is used as the final output layer to predict stock prices for each time step.
- Model parameters are detailed in

Model Parameters are –

Table 3: LSTM Parameters

Category	Parameters	Value
LSTM Layer	Number of LSTM layers	2
	Number of LSTM units per layer	128
	Activation function	ReLU
Dropout Layer	Dropout rate	0.2
Dense Layer	Number of Dense layers	1
	Activation function	Linear
Training Parameters	Number of epochs	50
	Batch size	32
	Optimizer	Adam
	Loss function	Mean Squared Error (MSE)
	Learning rate	0.001

5.3 Training and Evaluation

- The model is trained using 80% of the data, with the remaining 20% used for testing. The performance was measured using RMSE and MAPE.
- The predicted prices are inverse-transformed to their original scale for comparison with real prices.

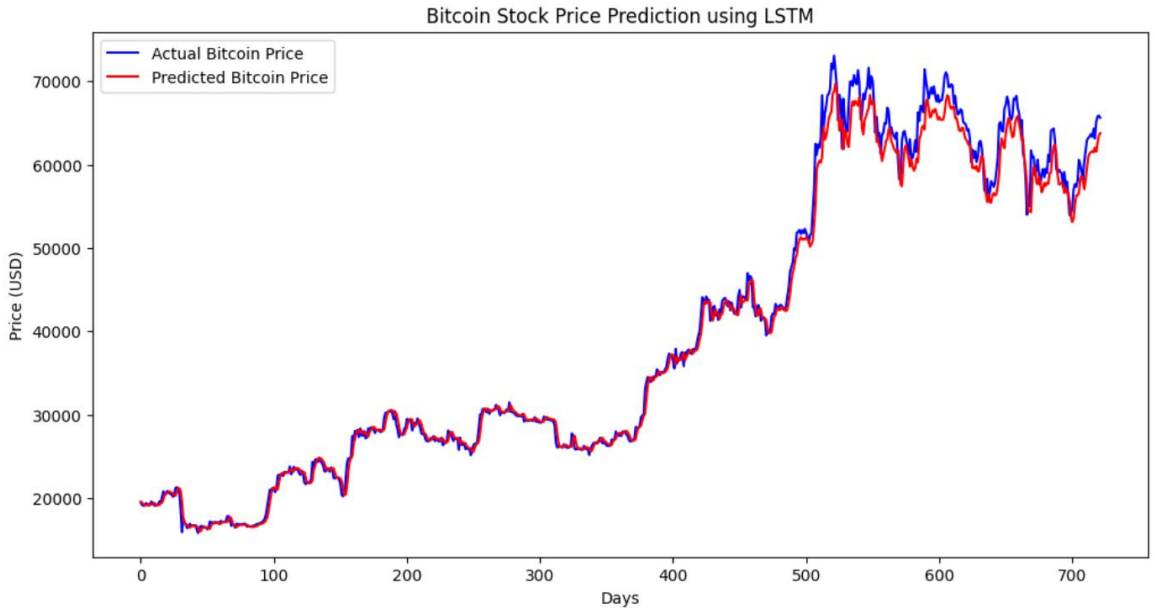
5.4 Comparison with Other Models

- CNN models is implemented to compare performance against the Regional LSTM model.
- RMSE and MAPE metrices are used for comparison across models

6 Results

6.1 Predicted Stock Price

Below are the predicted stock's prices vs actual stock's price drawn on an histogram for few companies.



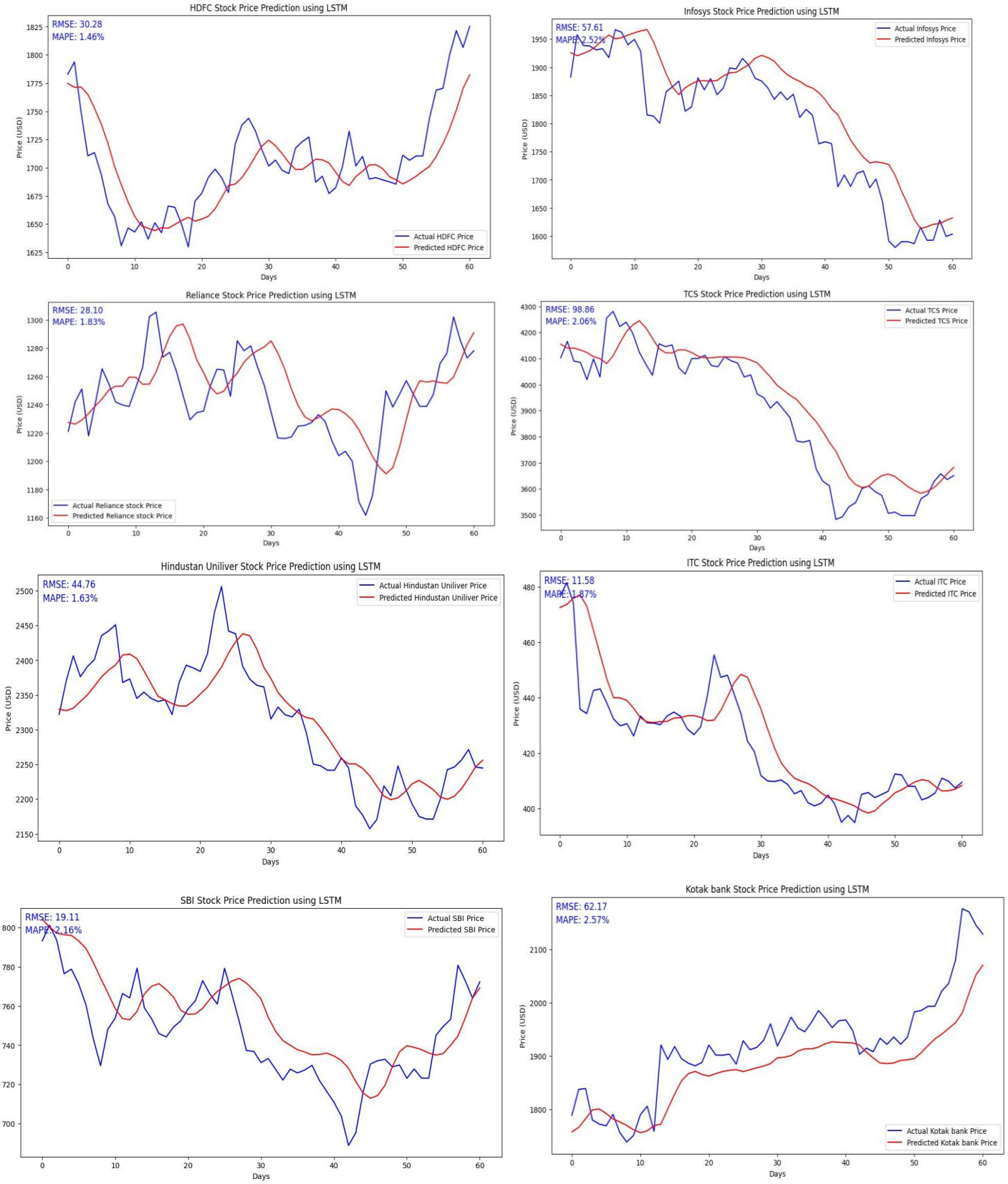


Figure : Plot of predicted stock price vs actual stock price for few companies. For all other companies please refer

6.2 Model Comparison

- Performance Metric for LSTM, and CNN

Table 5: Performance Metrices

Stock	LSTM		CNN	
	RMSE	**Accuracy(%)**	**RMSE**	**Accuracy(%)**
BITCOIN.CO	1721.50	97.87	2741.79	96.24
HDFC.CO	30.28	98.54	52.45	97.43
INFOSYS.CO	57.61	97.48	70.67	96.87
SBI.BO	19.11	98.84	26.93	97.43
TCS.BO	98.86	98.94	135.44	98.01
ITC.CO	11.58	98.13	19.13	97.15
HINDUNILVR.BO	44.76	98.37	59.98	97.88
KOTAK.BO	62.17	97.43	73.97	96.56

7 Discussion

- The Regional-LSTM model successfully combines the benefits of regionalization and neural network's learning for stock price prediction, showing strong predictive power compared to traditional models like CNN and Random Forest.

- There are many limitations in stock's price modelling since their prediction is very complex depending on so many factors. Limitation of our model is its reliance on a fixed regional window size, which may not always capture variations in stock price behaviour. Future work could explore adaptive window sizing or ensemble methods for improved predictions.

8 Conclusion & Future Scope

This report demonstrates the successful application of LSTM model for stock price prediction. The LSTM model outperformed CNN models in predicting stock prices, resulting in better portfolio performance.

Stock market is very complex. It depends on many factors involving government, seasonal trends, country and world's politics, individual company's performance etc. Since in this work only closing price of stocks were included as a feature for LSTM model, involving other features will improve the model's performance. There is a large scope to incorporate various features such as weekly moving average, 14-Days moving average, 30-Days moving average etc. Involving data from stock exchanges of other countries will also help to predict the market mood. Sentiment analysis can also be added by making a separate model for
sentiment analysis

9 References

- Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron(2017)
- Time-series analysis with smoothed Convolutional Neural Network, (https://doi.org/10.1186/s40537%E2%80%90022%E2%80%9000599%E2%80%90y)
- Yahoo Finance(https://finance.yahoo.com/))
- Embedded Features for 1D CNN-based Action Recognition on Depth Maps, Jacek Trelinski and Bogdan Kwolek, DOI: 10.5220/0010340105360543
- Handbook of Medical ImageComputing and Computer Assisted Intervention (https://doi.org/10.1016/B978-0-12-816176-0.00025-9)
- Optimization Methods in Finance, Gerard Cornuejols, Reha Tutu¨ncu¨