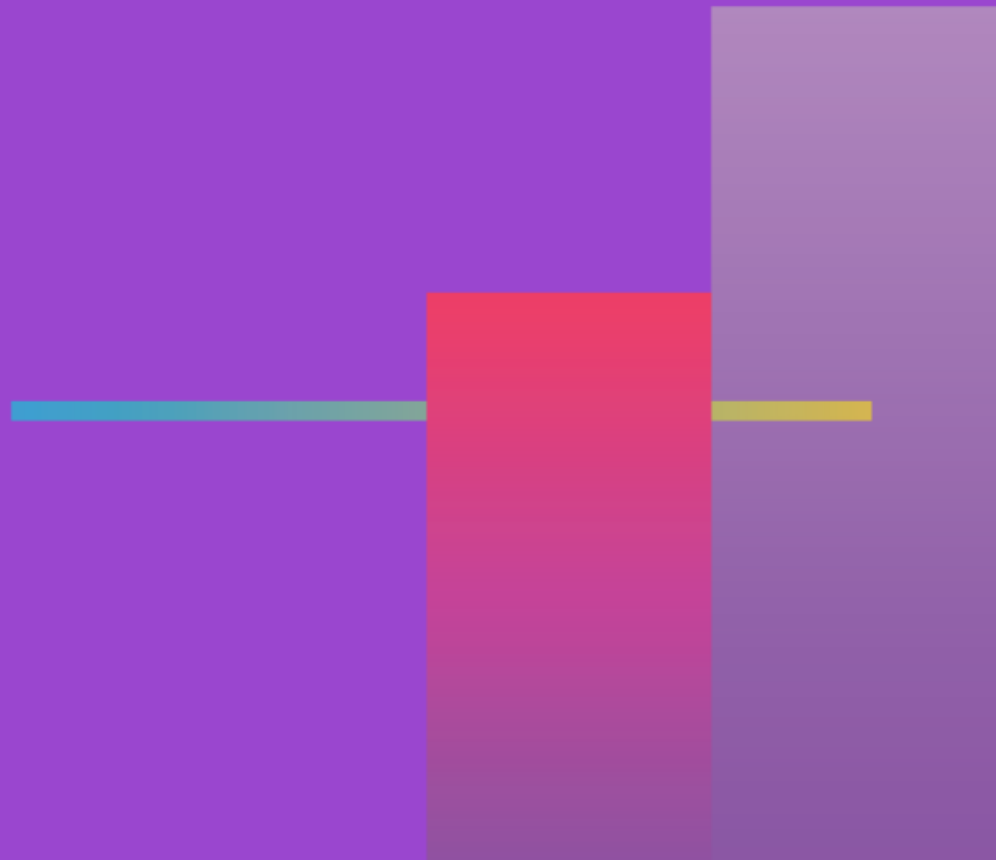
Application of Deep Learning Technique for Precise Stock Market Prediction

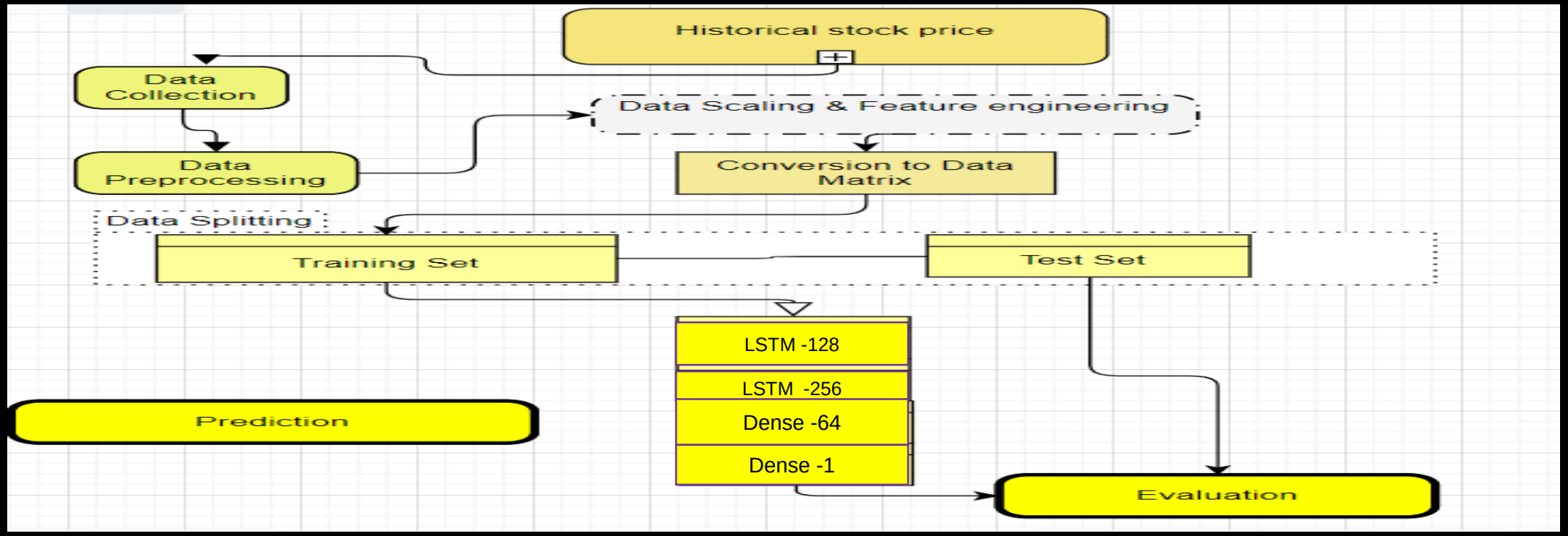


University of Hertfordshire UH

Introduction

In this study, we aim to develop an accurate stock price prediction model using LSTM networks using Keras. Our data-driven approach incorporates a comprehensive dataset, including historical prices, trading volumes. The LSTM model, trained through sequential learning. This task contributes significantly to financial forecasting, showcasing LSTM's potential for investors seeking a competitive edge in reliable predictions. In the process we will make use of EDA , feature engineering technique to explore and understand the data and the plot the data over period of 1000 trading days where last 200 trading days would be the prediction done by the model . In this journey we will also find some metrics like MSE, and MAE to evaluate the error percentage of our prediction. We will also split and compare the data as training , validation and test .

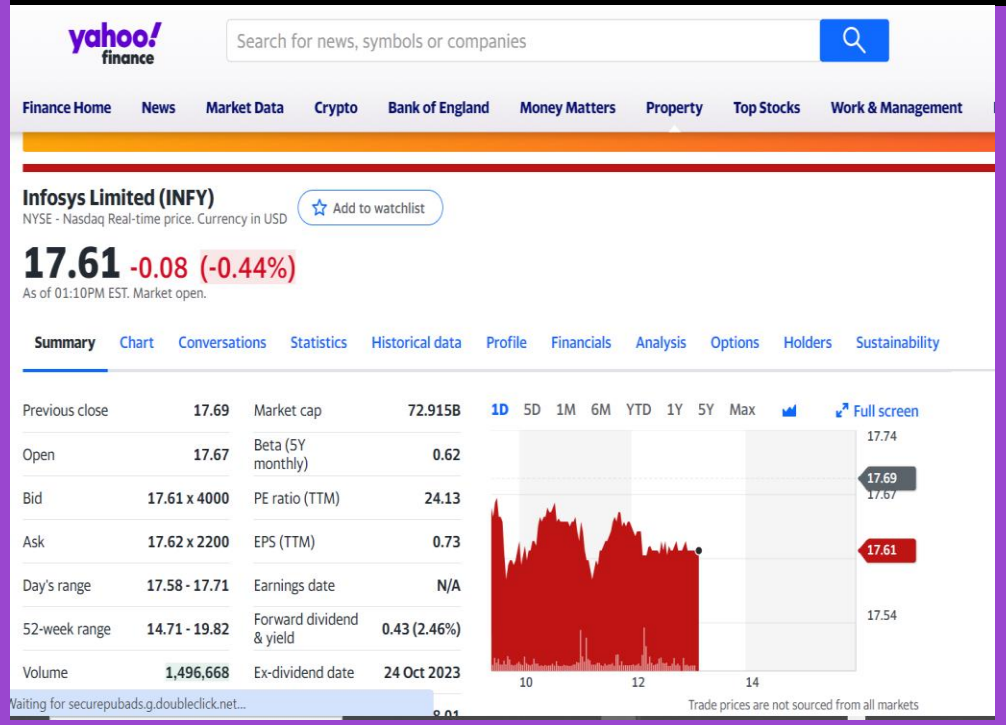
System architecture and data flow as in figure



Data Gathering

To gather the necessary market data for our stock prediction model, we will utilize the yFinance library in Python. This library is designed specifically for downloading relevant information on a given ticker symbol from the Yahoo Finance webpage.

For our purposes, we will be using the ticker symbol "INFY", which is a well-known technology company. Here's an example screenshot of the ticker symbol on the Yahoo Finance page:



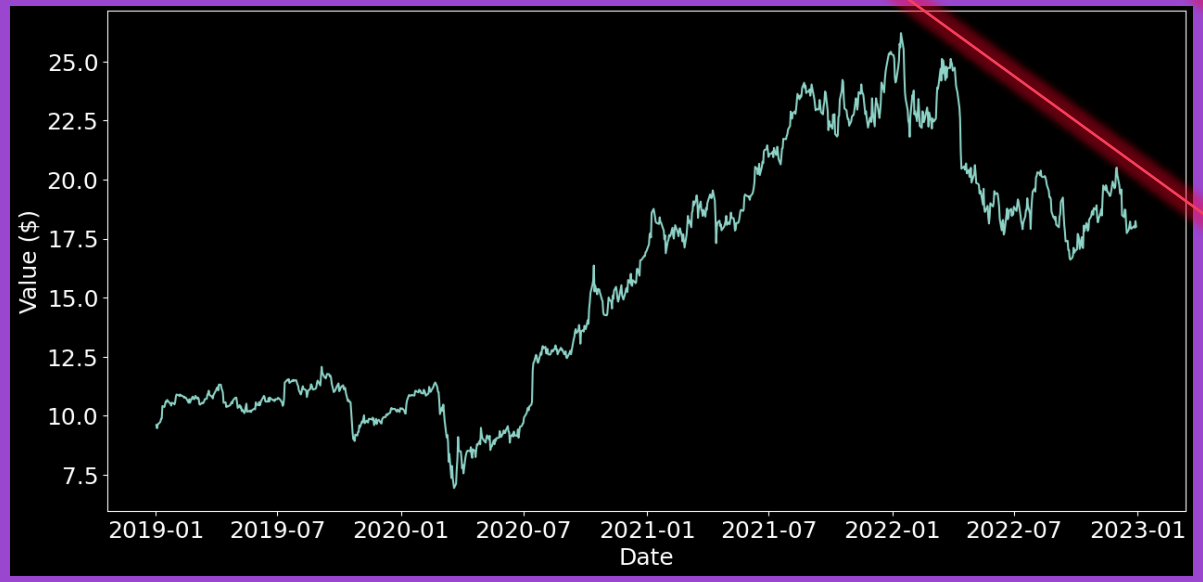
Details of data extracted from yFinance

Data Columns	Details
Open	Opening price of stock on that date
High	Highest value of the stock on that date
Low	Lowest prices of the stock on that date
Close	The closing price of the stock
Adj Close	Adjusted close is the price of the stock after dividends and other related manipulation
Volume	Volume of stock traded on that day

Data Preprocessing & Stock Visualization

- Data Cleaning – No null values present
- Data sorting – sorted the data
- Data scaling – scaled the data between 0 and 1
- Stats for the data for data exploration

- Data Visualization - Plotted the closing value of the stock to check the trends from 2019 till 2023




```
[4]    # Check the data for any null values
       data.isnull().sum()
```

```
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
dtype: int64
```

```
# Data Exploration
data = data.sort_values('Date')
print(data.head())
print(data.columns)
print(data.tail())
```

```
data['High'] = data['High'] / 100
data['Open'] = data['Open'] / 100
data['Low'] = data['Low'] / 100
data['Close'] = data['Close'] / 100
```

```
#Stats of the data
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1008 entries, 2019-01-02 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       1008 non-null   float64
 1   High       1008 non-null   float64
 2   Low        1008 non-null   float64
 3   Close      1008 non-null   float64
 4   Adj Close  1008 non-null   float64
 5   Volume     1008 non-null   int64
dtypes: float64(5), int64(1)
memory usage: 55.1 KB
```

	Open	High	Low	Close	Adj Close	Volume
count	1008.000000	1008.000000	1008.000000	1008.000000	1008.000000	1.008000e+03
mean	15.823175	15.962510	15.687282	15.828274	14.832637	9.967296e+06
std	5.160655	5.196915	5.123626	5.162797	5.124557	6.263134e+06
min	6.920000	7.190000	6.760000	6.940000	6.329522	1.320600e+06
25%	10.787500	10.827500	10.680000	10.760000	9.612980	6.101025e+06
50%	16.850000	16.915000	16.655000	16.725000	15.802841	8.416400e+06
75%	19.730000	19.895000	19.595000	19.719999	19.026180	1.206075e+07
max	26.150000	26.389999	25.580000	26.200001	24.964006	7.716540e+07

Feature engineering

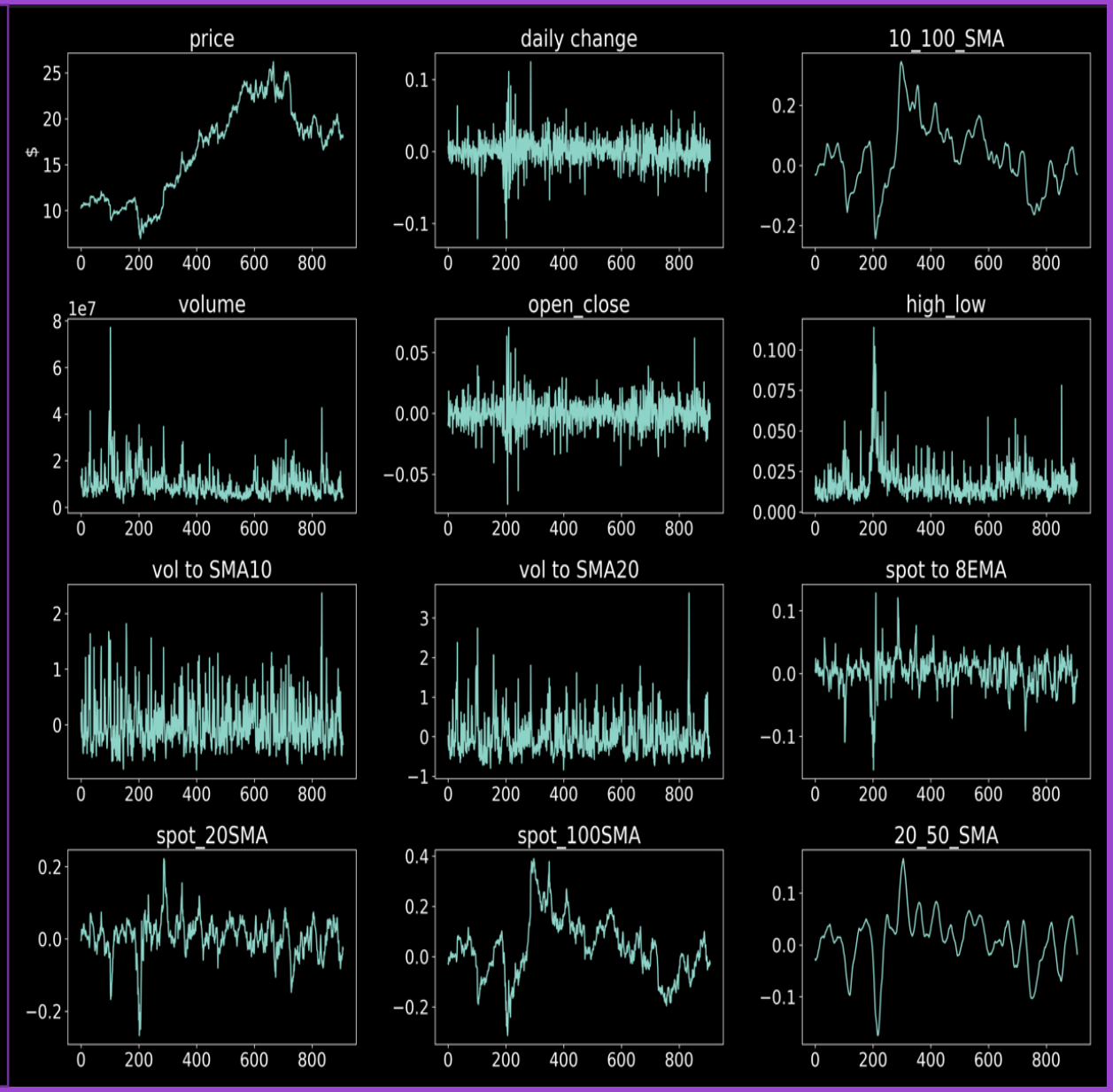
Feature construction – Used column “Volume” to derive the moving averages and open/close price comparison

Moving Averages (v_10 and v_20):
Calculates the 10-day,20 Days moving average calculated on the trading volume data over 10 periods.

Open-Close and High-Low Ratios:
Computes the percentage change between 'Close' and 'Open' and between 'High' and 'Low', which are then converted to lists.
Volume Moving Average with a window of 10 periods

Exponential Moving Average (EMA_8):
Computes the 8-day Exponential Moving Average (EMA) for the 'df' DataFrame. EMA gives more weight to recent data points, and the weight decreases exponentially as you move back in time.

Simple moving average SMA_10, SMA_20 , SMA_50 ,SMA_100 :
SMA calculates the average of the specified number of periods equally. It gives equal weight to all data points in the window.



Long Short Term Memory - Description

LSTM or Long Short-Term Memory is an improvement over traditional RNN or Recurrent Neural Network in the sense that it can effectively "remember" long sequence of events in the past. Just like humans can derive information from the previous context and can chart his future actions, RNN and LSTM tends to imitate the same. The difference between RNN and LSTM is that RNN is used in places where the retention of memory is short, whereas LSTM is capable of connecting events that happened way earlier and the events that followed them.

LSTM is capable of performing three main operations: Forget, Input and Output. These operations are made possible with the help of trained neural network layers like the tanh layer and the sigmoid layer. These layers decide whether an information needs to be forgotten, updated and what values need to be given as output. LSTM learns which parameter to learn, what to forget and what to be updated during the training process. That is the reason why LSTM requires a huge amount of dataset during its training and validation process for a better result. The sigmoid layer decides the flow of information. Whether it needs to allow all of the information or some of it or nothing.

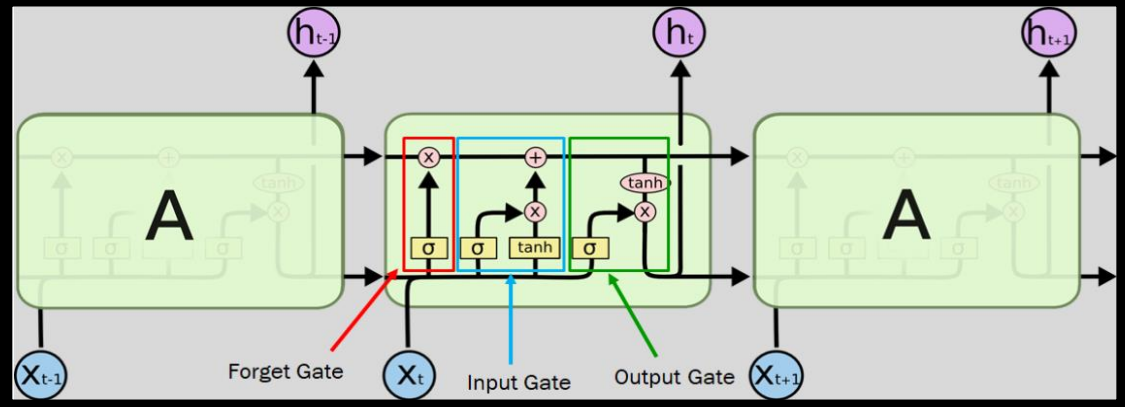
Advantages of using LSTM for stock market prediction

Sequential Data Handling : Ideal for modeling historical prices, trading volumes, and technical indicators.

Memory and Context Preservation:
Captures long-term dependencies, vital for understanding market dynamics.

Non-Linearity and Complex Patterns:
Excels at modeling non-linear and complex behaviors in financial markets

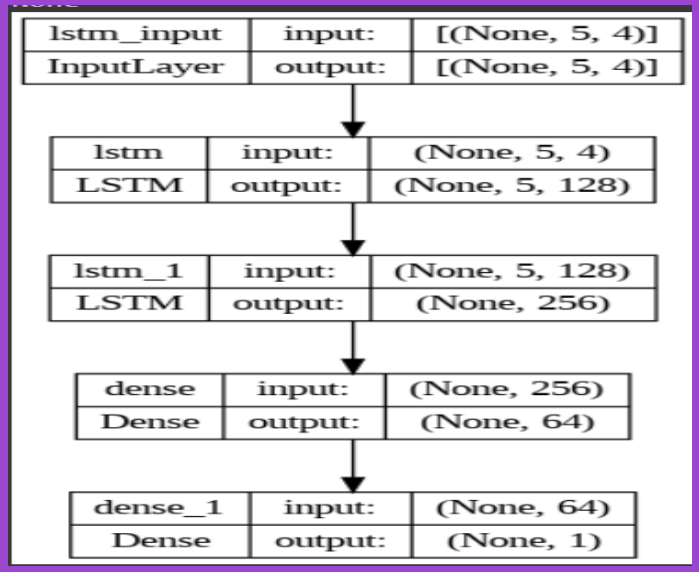
LSTM unit



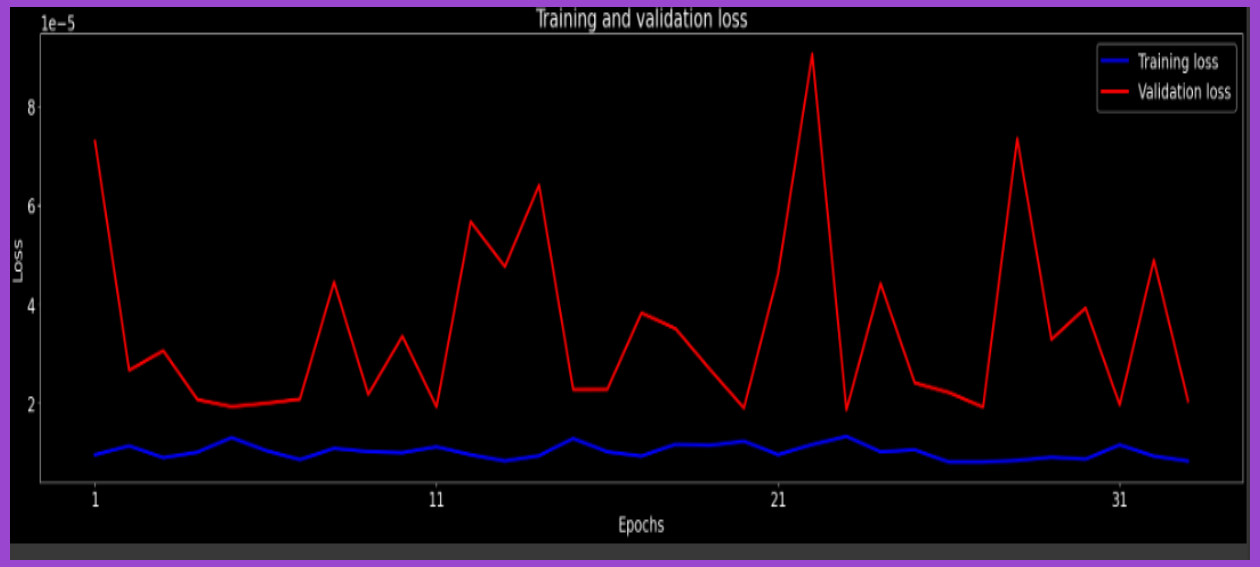
Contains a cell state that carries information from one state to another states.
Gates manipulate the info in the cell state. Three main gates used to do this are:
Forget Gate: Responsible for removing useless information.
Input Gate: Responsible for adding new information.
Output Gate: Responsible for filtering out the information.

The LSTM architecture and Trainable Parameters


```
Model: "sequential_41"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_84 (LSTM)              (None, 5, 128)            68096

 lstm_85 (LSTM)              (None, 256)               394240

 dense_83 (Dense)            (None, 64)                16448

 dense_84 (Dense)            (None, 1)                 65

=================================================================
Total params: 478849 (1.83 MB)
Trainable params: 478849 (1.83 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

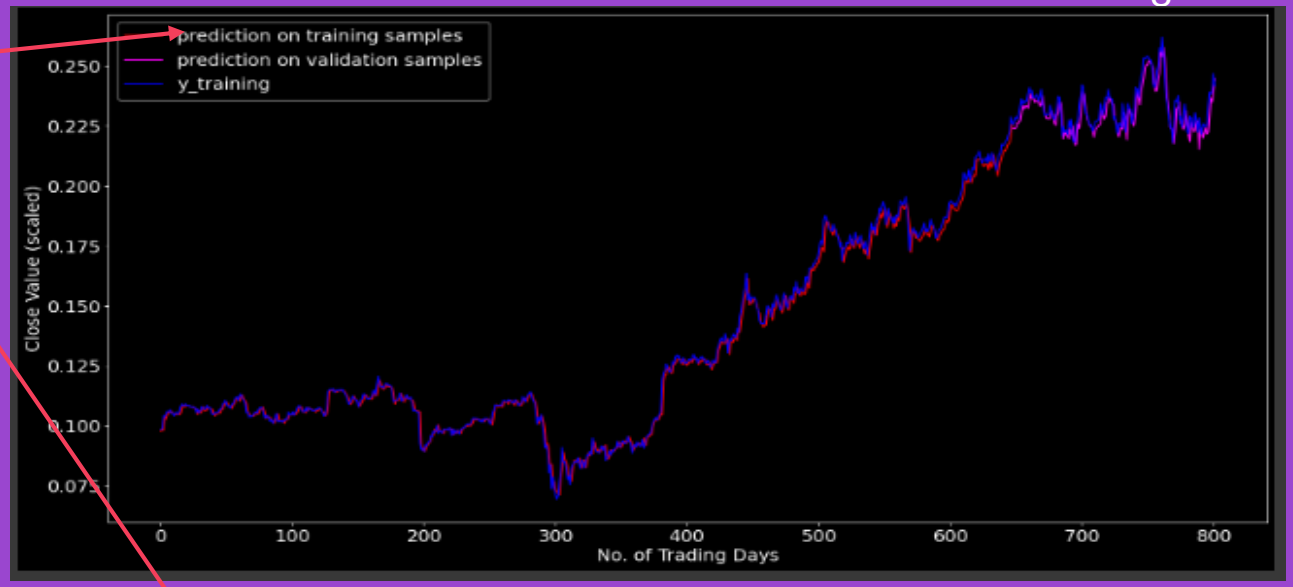


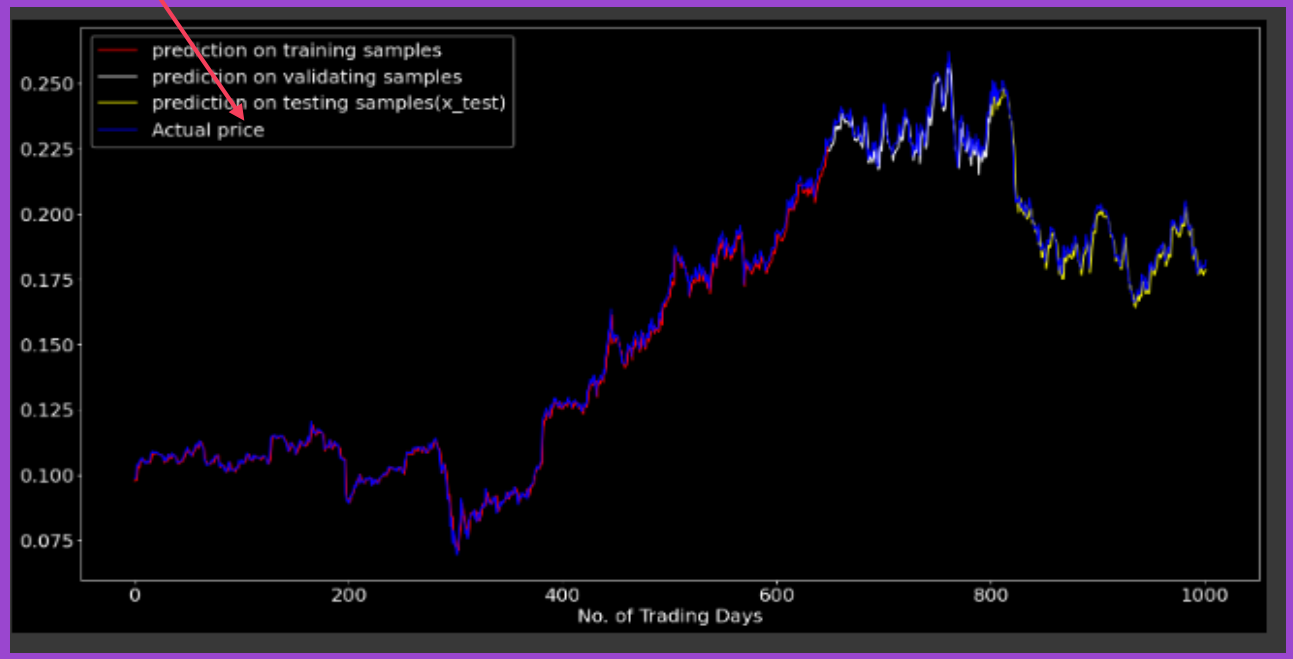
Loss and error graphs



Results and predictions from the model –Via Graph

1.Validation and training data was used to compare values across 800 trading days .
2. Prediction was made for the next 200 trading days and compared with the actual price movement for the stock
3. Finally based on various parameter changes and tuning we reached a value of MAE percentage within 1% range





Parameter tuning for best model

EPOCS	Learning rate	Batch Size	mean absolute error for predicted Value	mean absolute percentage error for predicted Value
100	0.001	16	0.45289	2.33766
150	0.001	16	0.39458	2.04746
200	0.001	16	0.32013	1.64976
50	**0.0001**	**8**	**0.29721**	**1.53169**

Result Analysis & Challenges faced

During the initial training of our model, we utilized the Stochastic Gradient Descent (SGD) optimizer. Unexpectedly, the result accuracy was high. The high accuracy on training data set indicates that the model is overfitting & not perform well on unseen data(test data), Additionally, the Mean Absolute Error (MAE) percentage was notably high(4.53169). Acknowledging the imperative for enhancement, we made a transition to a different optimizer and implemented additional strategies aimed at improving accuracy.

Below techniques were used to improvise the model and reach a outcome within **1 % error margin**.

Data Scaling:- We scale our input features to a similar range to prevent any feature from dominating the learning process, ensuring faster model convergence.

Optimizer Switch:-We switched from **SGD** to a more advanced optimizer like **Adam** to take advantage of better learning rates and improved training progress.

Learning Rate Adjustment:- We tried out different learning rates to make the model learn better. If the rates are too high, the model might jump past the optimal solution, and if they're too low, the learning process can become too slow. Tried learning rate like, 0.01,0.02,0.001,0.002, 0.0001,0.0002

Also experimented with the below parameters to reach an optimum model

1. **Early stopping**
2. **Batch Normalization**
3. **Dropouts – 02. 0.3**
4. **Increased the LSTM layer from 1 to 2**
5. **Kept dense layer to 2**
6. **Tried different activation function like RELU, Linear**

After implementing these modifications, our model experienced substantial enhancement. The transition from SGD to Adam, along with fine-tuning learning rates and implementing data scaling, effectively reduced Mean Absolute Error (MAE). This reduction in MAE signifies improved accuracy highlighting the model's enhanced ability to make predictions that closely align with actual outcomes.

Conclusion

Analysis of Training and Validation Loss/Error

The graph depicting the training and validation loss/error provides crucial insights into the performance of our LSTM-based stock market prediction model. The loss/error metrics are visualized over epochs, allowing us to understand the convergence and generalization capabilities of the model. Key observations from the graphs include:

Training Loss/Error:

- The blue line represents the training loss or error, it shows how well the model fits the training data.
- A gradual decrease in the training loss signifies effective learning, capturing underlying patterns in the training dataset.

Validation Loss/Error:

- The red line represents the validation loss or error, indicating the model's performance on unseen data.
- A close alignment between training and validation loss implies that the model generalizes well to new data.

Model Evaluation Scores

The model evaluation scores provide quantitative measures of the model's performance on both the training and testing datasets:

Metric	Value
Root Mean Squared Error (RMSE): [Train RMSE]	0.003317397
Mean Absolute Error (MAE): [Train MAE]	0.002583009
Root Mean Squared Error (RMSE): [Test RMSE]	0.003889267
Mean Absolute Error (MAE): [Test MAE]	0.002974000

Conclusion…

Predictive Visualization: The predictive visualization gives a 360 degree view of the predicted and actual stock prices for both training and testing datasets. Below are the observations

Training Samples	Validation Samples	Testing Samples (x_test)
• The red line in the graph represents the model's predictions on the training samples. • Close alignment with the blue line (actual prices) indicates accurate predictions on the training set.	• The magenta line in the graph illustrates the model's predictions on the validation samples. • The model's ability to generalize is evident through its performance on previously unseen validation data.	• The yellow line in the graph showcases the model's predictions on the testing samples. • A comparison with the blue line (actual prices) demonstrates the model's effectiveness in predicting future stock prices.

Overall Observations and Conclusions:In conclusion, our LSTM-based stock market prediction model exhibits promising performance. The model perform well on the unseen data as well as predicting the stock closing values for the next 200 trading days with MAE percentage around 1.5 %. The price of the stock "INFY" would be between **$17.5 to $20.**

Key observations to conclude

a)The model demonstrates effective learning and generalization, as evident from the convergence of training and validation loss/error.

b) Evaluation scores such as RMSE and MAE indicate a reasonable level of accuracy in predicting both training and testing datasets.

c)Predictive visualizations showcase the model's ability to capture trends and patterns in stock prices.

Future Enhancement

There are several avenues for future enhancements and research in the field of stock market prediction using LSTM algorithms

Feature Engineering: Further investigation can be done on identifying and incorporating additional relevant features that might impact stock prices, such as news sentiment, corporate events, or geopolitical factors. Enhancing feature engineering techniques could lead to more accurate predictions

Model Architecture: Experimenting with different LSTM model architectures, such as stacked or bidirectional LSTMs, may improve the performance and prediction capabilities. Exploring other variants of RNNs, such as Gated Recurrent Units (GRUs), could also be beneficial.

Ensemble Methods: Employing ensemble methods, such as combining predictions from multiple LSTM models or integrating with other machine learning techniques like random forests or gradient boosting, could potentially enhance the overall predictive power and robustness of the model

Hyper parameter Tuning: Conducting a systematic search for optimal hyper parameters, including the number of hidden layers, learning rate, batch size, and regularization techniques, can further optimize the LSTM model's performance.

References

1. Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

2. Gao, P., Zhang, R., and Yang, X. (2020). The application of stock index price prediction with neural network. Mathematical and Computational Applications, 25(3), 53. http://dx.doi.org/10.3390/mca25030053

3. Zou, Z., and Qu, Z. (2020). Using LSTM in Stock prediction and Quantitative Trading. CS230: Deep Learning, Winter, 1-6.

4. Shen, J., and Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. Journal of big Data, 7(1), 1-33. https://doi.org/10.1186/s40537-020-00333-6

5. Dong, Y., Yan, D., Almudaifer, A. I., Yan, S., Jiang, Z., and Zhou, Y. (2020, December). Belt: A pipeline for stock price prediction using news. In 2020 IEEE International Conference on Big Data, 1137-1146. https://doi.org/10.1109/BigData50022.2020.9378345