Project Report

Group 51
Team members:

(1) Rishi Raj Bhagat (B23MT1034)
(2) Uday Shaw (B23CH1045)
(3) Sanjiban Sarkar (B23CH1041)
(4) Patel Nisarg Rakeshkumar (B23MT1029)

Project Title: INR Price Prediction

1. Introduction:

In this project, we aim to predict the closing INR (Indian Rupee) price using historical exchange rate data from the *HistoricalData.csv* file. This file includes daily exchange rate information, containing the following columns:

- Date: The date of the exchange rate data.
- Close/Last: The closing exchange rate (used as the target variable for prediction).
- Open, High, Low: The opening, highest, and lowest exchange rates for the day.
- Volume: This column has missing values and is therefore not used in our analysis.

Our approach involves using a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN), a model well-suited to sequence data such as time series. By training this model on historical INR closing prices, we can generate predictions that capture trends and patterns over time, allowing us to forecast future INR closing rates.

Overview of LSTM:

The LSTM model is a type of RNN that excels at learning long-term dependencies, making it ideal for time series forecasting. Unlike traditional RNNs, which struggle to retain information over extended sequences, LSTMs use a unique structure composed of:

- Forget Gate: Determines which information from the previous state is no longer relevant and can be discarded.
- Input Gate: Adds new, relevant information to the cell state.
- Output Gate: Controls which information is passed to the next layer as output.

This structure enables the LSTM to retain important past trends while ignoring irrelevant details, making it particularly effective in predicting exchange rates based on historical data patterns.

2. Code overview and structure:

1. Data Loading and Preprocessing

Data Import: Load historical data containing date, closing prices, and volume.

Date Formatting and Sorting: Convert dates to a standard datetime format, sort the data chronologically, and set dates as the index for easy time series manipulation.

Target Variable Selection: Extract the "Close/Last" column as the target variable for the prediction task.

Feature Adjustment: Drop the "Volume" column, focusing solely on price data for this model.

2. Data Scaling

Normalization: Apply MinMax scaling to rescale closing prices between 0 and 1, a necessary step for training neural networks like LSTM, which perform better on normalized data.

3. Exploratory Data Analysis

Distribution of Closing Prices

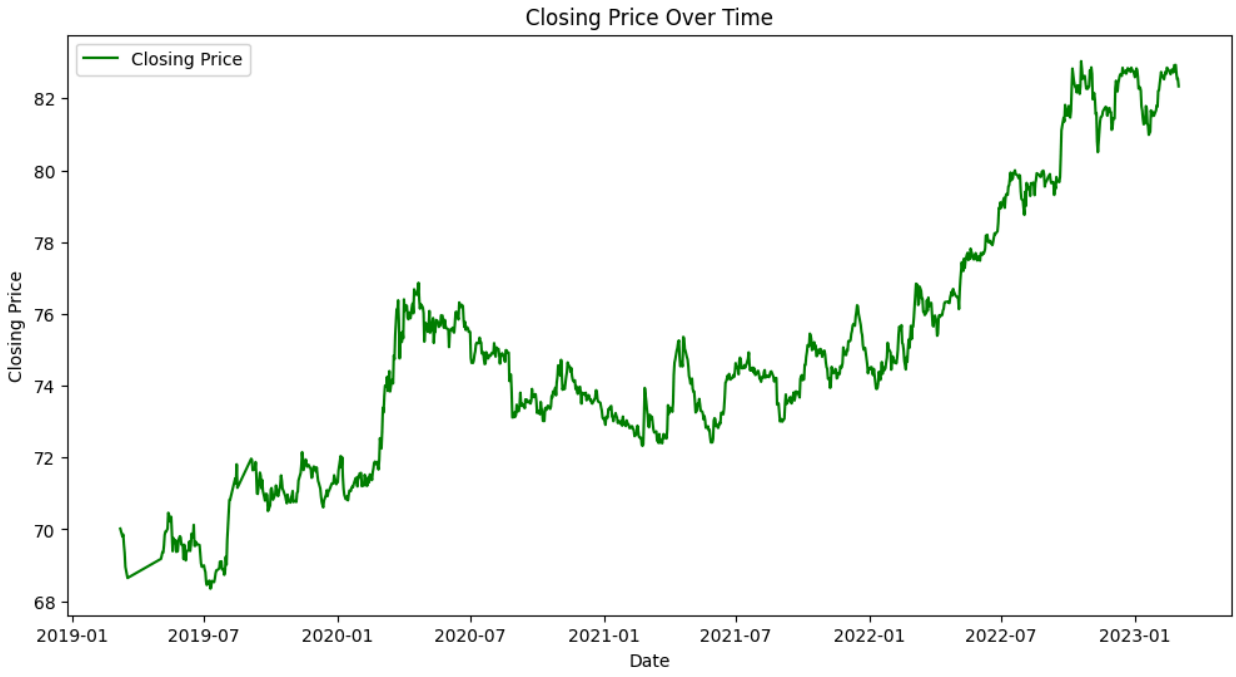
The histogram shows the frequency distribution of closing prices over the given time period.

It appears to have a roughly normal distribution with the majority of prices concentrated between **72 and 76**, indicating this range is the most common.

Trend of Closing Prices Over Time

The time series plot illustrates a general upward trend in closing prices over the observation period.

Significant periods of growth are visible, particularly around **mid-2020 to 2022**, likely reflecting positive market performance or external factors influencing stock or asset prices.

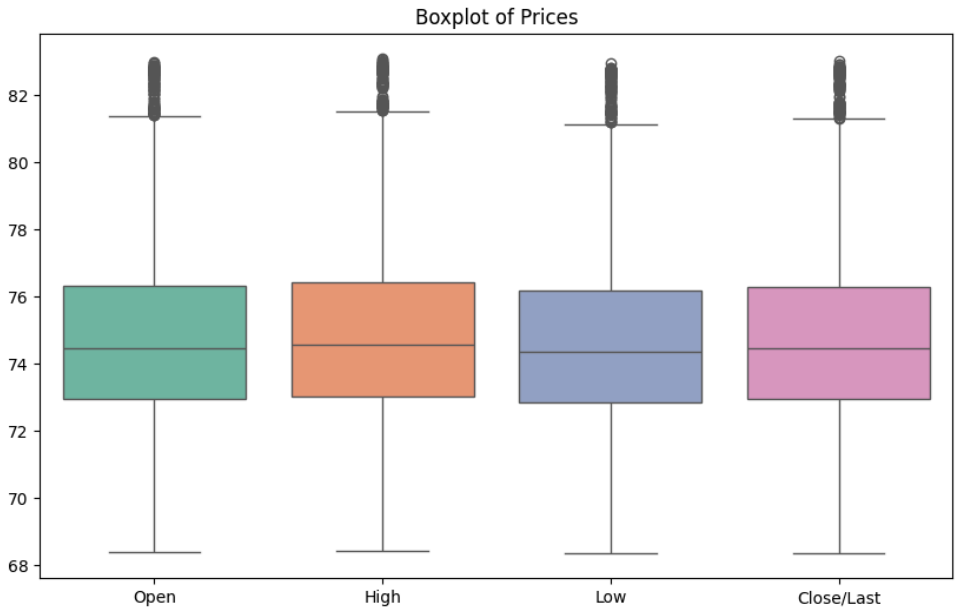


Boxplot Analysis (Prices)

Median and Spread: Similar median values and interquartile ranges across Open, High, Low, and Close/Last prices.

Outliers: Outliers are present above the upper whiskers, indicating occasional high price deviations.

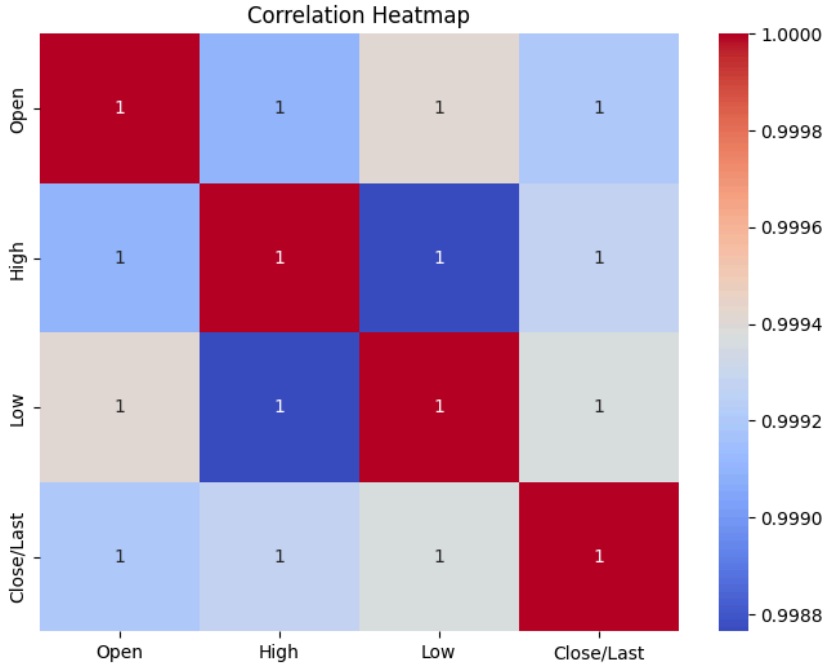
Symmetry: Distributions are mostly symmetric, showing balanced price variations, except for outliers.



Correlation Heatmap Analysis

Strong Correlation: Near-perfect positive correlations (close to 1) among all price categories.

Interpretation: Prices move together consistently, reflecting strong market interdependence.



4. Data Preparation for LSTM Model

Defining Sequence Length: Set a sequence length of 70, meaning the model will use the last 70 days to predict the next day's price.

Creating Training Data:

- Split data into sequences (inputs) of 70 past values with the following value as the target output.
- Convert lists of inputs and targets into NumPy arrays, reshaping inputs to the required 3D shape for LSTM.

Preparing Test Data: Separate the last 150 values for testing to evaluate model performance. Create similar sequences for the test set, reshaping them for the LSTM.

5. LSTM Model Building

Model Architecture: Construct a sequential neural network with:

- Two LSTM layers, the first returning sequences for the next layer.
- A dense layer with a single output for predicting the next day's price.

Compilation: Use the Adam optimizer and mean squared error as the loss function.

6. Model Training

Early Stopping: Implement early stopping with patience set to prevent overfitting if no loss improvement occurs after 10 epochs.

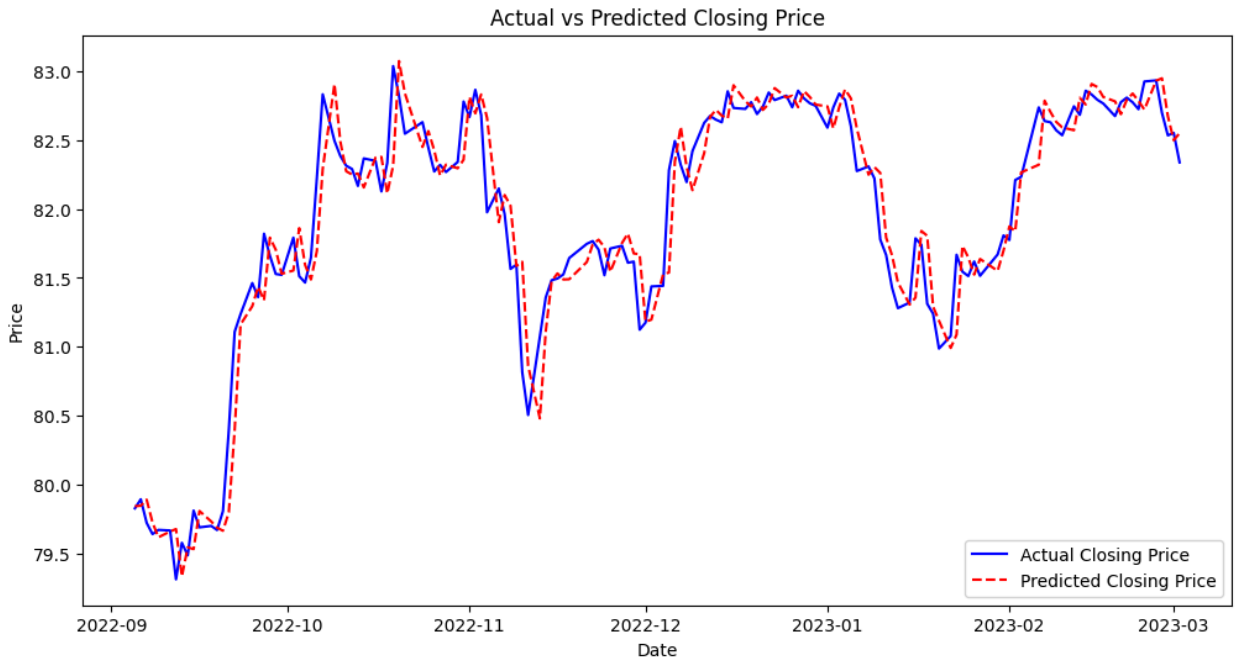
Training: Train the model on the prepared training data with batch size 32, running for up to 100 epochs or until early stopping.

7. Model Prediction and Evaluation

Prediction on Test Set: Use the trained model to predict prices for the test set. Inverse transform these predictions to match the original scale of the closing prices.

Visualization: Plot the actual vs. predicted prices over the test period to visually assess prediction accuracy.

It gives the following graph:



Future Prediction Function:

1. Function Purpose: Predict the closing prices for a specified number of future days using the trained LSTM model.

2. Starting Sequence:

- Uses the last available scaled sequence of data (of length 120) as the base input for future predictions.

3. Prediction Loop:

- For each future day:
  - Reshapes the last sequence to match the input format for the LSTM model.
  - Predicts the next day's scaled closing price.
  - Inverse-transforms the predicted value back to the original price scale and stores it in the list of future predictions.

4. Updating the Sequence:
- Appends each new prediction to the end of the sequence while removing the first value, maintaining a fixed sequence length of 120 for consistent input to the model.

5. Return:
- Outputs a list of predicted prices for the specified number of days.

Future Price Prediction and Visualization:

1. User Input for Prediction Duration
- Prompts the user to specify the number of future days to predict.
- Stores this input as `days_to_predict`, which is then passed to the prediction function.

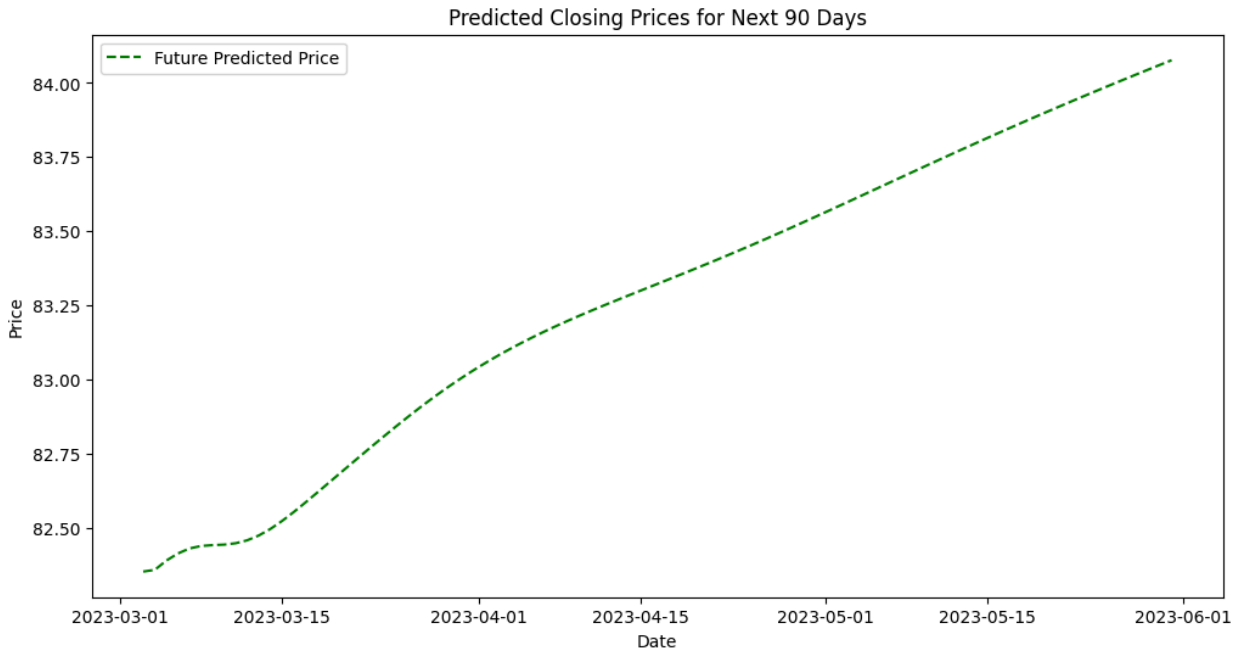
2. Future Prediction
- Calls the `predict_future` function with `days_to_predict` to generate the list of predicted prices for the specified days.

3. Generating Future Dates
- Creates a date range starting from the day after the last date in the historical data, with a length equal to `days_to_predict`.
- This range serves as the timeline for plotting the predicted future prices.

4. Plotting Future Predictions

- Plots only the predicted future prices against the generated future dates to visualize the model's forecast.
- The plot includes:
  - A green dashed line representing predicted prices.
  - Titles, labels, and legends for clarity, showing the predicted closing prices for the specified number of days.

5. Displaying Future Predictions
- Iterates through each date and its corresponding predicted price.
- Prints the predicted closing price for each future date in a readable format.



LSTM Model:

An epoch is one complete pass through the entire training dataset. When training a model, each epoch means the model sees every sample in the dataset once and adjusts its weights based on the error between its predictions and the actual values.

In the context of machine learning models, including LSTMs, training typically involves running multiple epochs. This allows the model to refine its predictions gradually by repeatedly updating its weights, aiming to reduce the loss function with each epoch.

How Epochs Work in LSTM Training

1. Data Shuffling: Each epoch may start with the data shuffled (for non-time-series tasks) or in sequence (for time-series like stock prediction).

2. Forward Pass: For each sample in the dataset, the model performs a forward pass through the LSTM layer, generating predictions based on the input sequence.

3. Loss Calculation: After each forward pass, the model calculates the error, or loss, between its prediction and the actual value. A common loss function for regression (e.g., stock price prediction) is Mean Squared Error (MSE)

4. Backward Pass (Gradient Descent): The model then performs a backward pass, where it calculates gradients with respect to each weight in the LSTM cells using backpropagation through time (BPTT). These gradients indicate the direction and magnitude by which each weight needs to be adjusted to minimize the error.

5. Weight Update: Using an optimizer (such as Adam or SGD), the model updates the weights in small steps. Each epoch helps refine the model's ability to capture patterns in the data by making these adjustments.

6. Convergence: With every additional epoch, the loss should ideally decrease, showing that the model is learning. However, if the loss stops improving or starts increasing, it may indicate overfitting, which can often be managed by reducing the number of epochs or applying techniques like dropout.

Multiple Epochs usage:

The reason for running multiple epochs is to allow the model enough iterations to find the optimal set of weights for making accurate predictions.