Exploratory Data Analysis and Predictive Modeling for BTCINR

Analysis of Candlestick Data and Technical Indicators for BTCINR

Author:
Bhavesh Nareshkumar Pabnani

Table of Contents

1. Introduction

The objective of this analysis is to explore historical BTCINR data, understand key technical patterns, and extract valuable insights using exploratory data analysis (EDA) and technical indicators. The assignment also aims to develop features that can serve as inputs to predictive models, particularly for forecasting the close price and analyzing the volatility patterns in cryptocurrency markets.

By performing in-depth data exploration and feature engineering, the final objective is to enhance predictive modeling for cryptocurrency, focusing on BTC/INR, through methodologies like rolling statistics, candlestick patterns, and technical indicators.

2. Objectives

The key objectives of this report are to:

- Conduct detailed exploratory data analysis on BTCINR data using candlestick charts.
- Analyze the relationships between key features like volume, volatility, and price.
- Implement and explain important technical indicators such as Moving Averages, Bollinger Bands, RSI, and MACD.
- Engineer features from raw candlestick data and compute technical indicators for predictive modeling.
- Develop a predictive framework for BTCINR price forecasting, validating it through backtesting and feature importance.

3. Dataset Description

3.1 Data Source

The dataset for this analysis was primarily obtained from the **Kline API** of the Pi42 cryptocurrency exchange, which provided historical candlestick data at minute intervals. This API offers a reliable and efficient way to retrieve historical data, making it suitable for our modeling purposes. The data includes open, high, low, and close (OHLC) prices, trading volume, and additional technical indicators calculated during the feature engineering phase.

In addition to the Kline API, I also implemented a script to collect data using the **WebSocket**. While WebSocket is beneficial for obtaining real-time data and is typically used for dynamic applications, it was taking a considerable amount of time and resources to scrape the data for modeling purposes. This is because WebSocket requires a continuous connection to fetch data as it streams, which can lead to higher computational overhead and complexity in handling the incoming data in real-time. Therefore, for this analysis and modeling, the Kline API was preferred due to its efficiency in retrieving historical data in a structured format.

I have included the WebSocket script in the submission for reference, demonstrating its potential application for real-time data monitoring and analysis in future work.

3.2 Feature Overview

The dataset consists of the following columns:

- **open, high, low, close**: Standard OHLC prices for BTCINR.

- **volume**: The trading volume during the specific interval.
- **startTime** and **endTime**: The starting and the ending time of the candles.
-

4. Exploratory Data Analysis (EDA)

4.1 Statistical Summary

The dataset provides a detailed statistical overview of the key features such as open, high, low, close prices, and trading volume for BTCINR. The following metrics are observed:

- **Mean:** The average close price is ₹5,461,709, with the mean open, high, and low prices being very close, indicating small fluctuations over the time period.
- **Standard Deviation:** The standard deviation for the close price is ₹52,163, reflecting moderate volatility. The standard deviation of the volume is around 5.27, signifying variations in trading activity.

Important insights include:

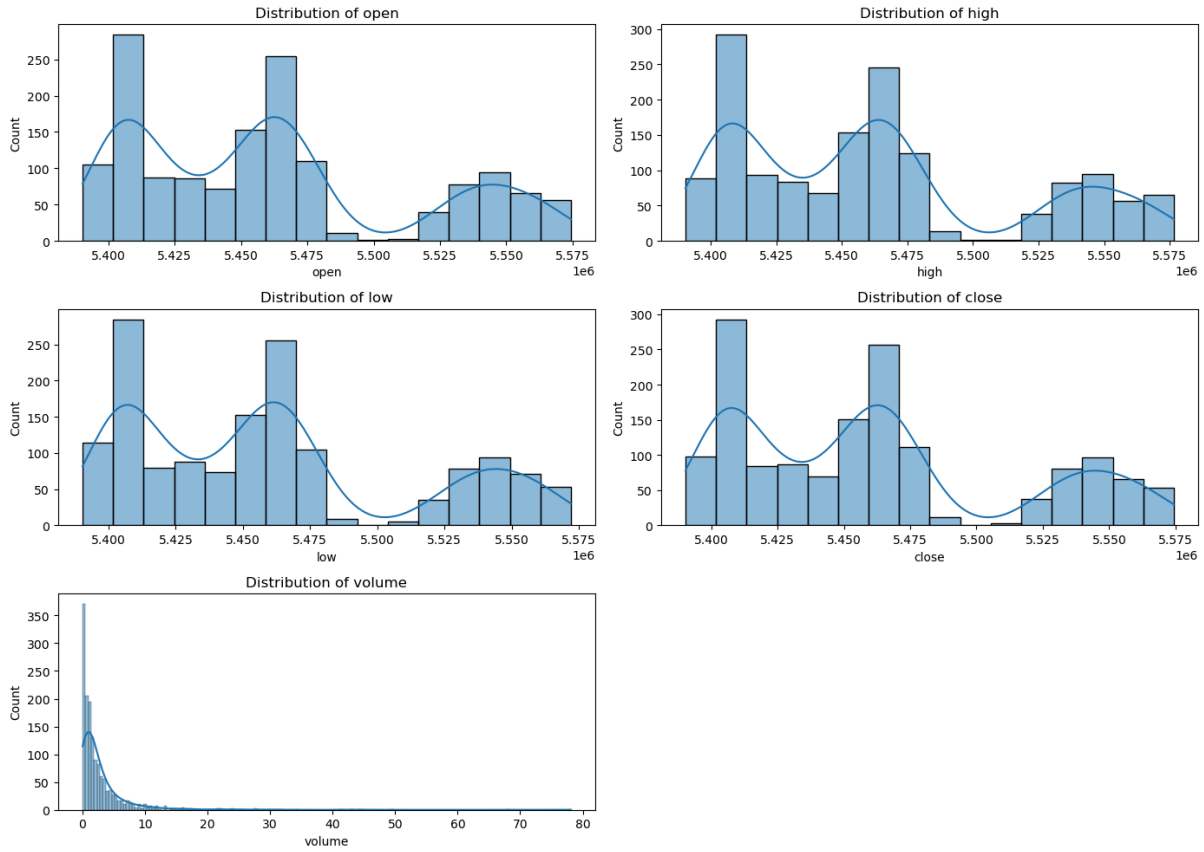
	open	high	low	close	volume
count	1.500000e+03	1.500000e+03	1.500000e+03	1.500000e+03	1500.000000
mean	5.461594e+06	5.463089e+06	5.460365e+06	5.461709e+06	2.885153
std	5.217482e+04	5.264062e+04	5.175818e+04	5.216323e+04	5.267420
min	5.390330e+06	5.390373e+06	5.390330e+06	5.390330e+06	0.002000
25%	5.412485e+06	5.413372e+06	5.411887e+06	5.412602e+06	0.481750
50%	5.457788e+06	5.459156e+06	5.456253e+06	5.457709e+06	1.359500
75%	5.478712e+06	5.480374e+06	5.476902e+06	5.478922e+06	3.150000
max	5.574322e+06	5.576536e+06	5.572002e+06	5.574322e+06	78.111000

4.2 Distribution Analysis

Histograms and density plots for key features such as high, low, open, close prices, and volume reveal interesting patterns in their distributions:

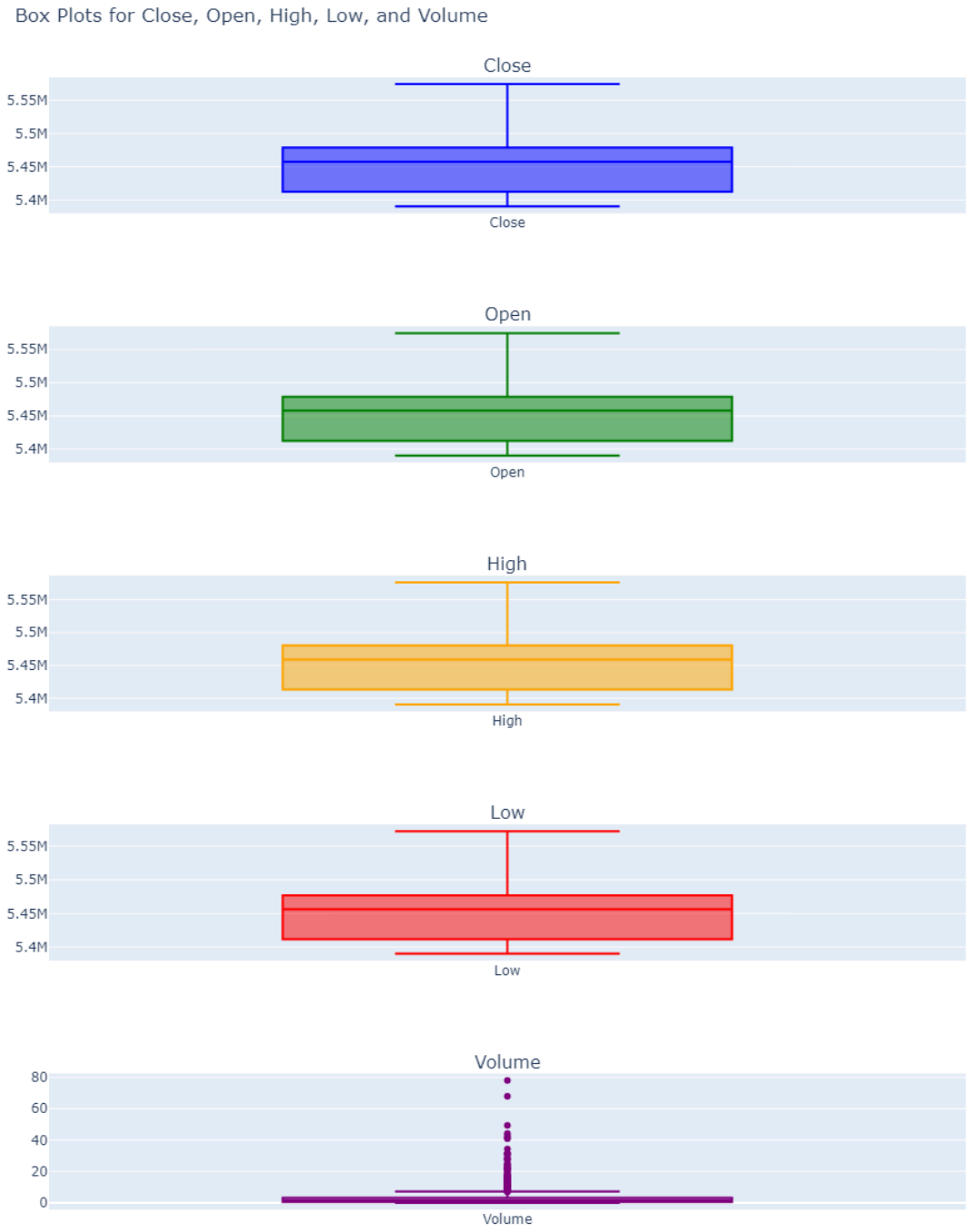
- **Price Distributions (Open, High, Low, Close):** The distributions are not normally distributed and exhibit a bimodal shape with two prominent peaks on the left and right, indicating the presence of price clusters at different levels. There is also a small peak towards the right, with a light tail, suggesting occasional higher price movements, but overall the distributions are more concentrated towards lower values.
- **Volume Distribution:** The volume distribution is heavily skewed towards zero, with a long right tail. This indicates that most trading activity happens at relatively low volumes, but there are occasional large spikes in volume, as reflected in the tail.

These distribution patterns provide insights into the underlying market dynamics, where prices tend to cluster around specific levels and volume is typically low with sporadic large trades.



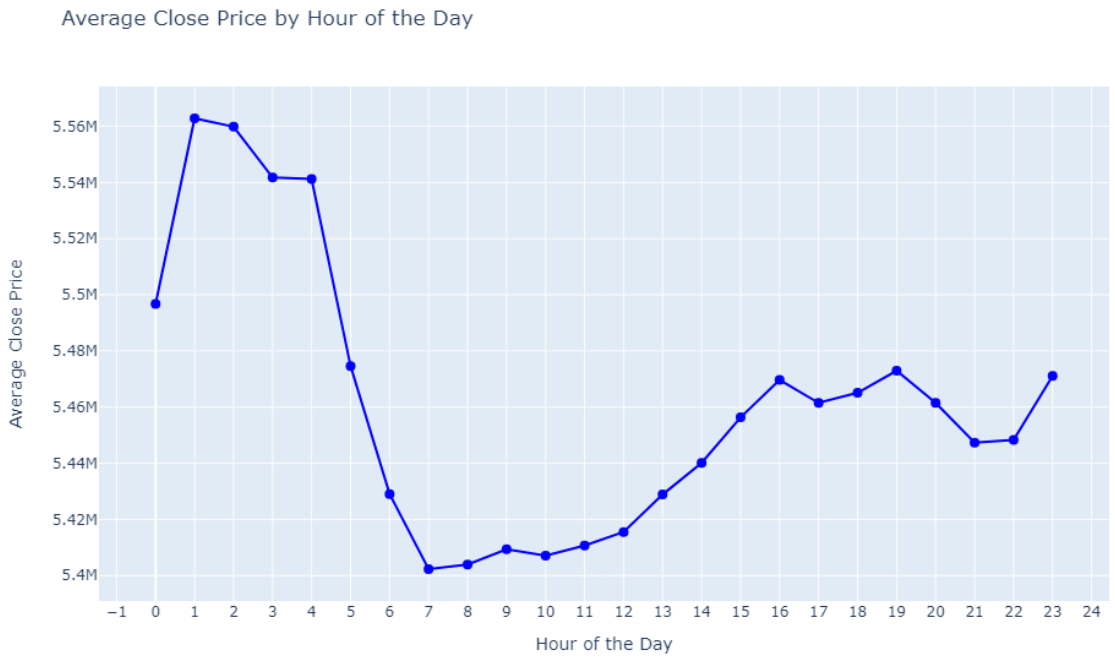
4.3 Anomalies Detection

Anomalies were detected using Z-scores and box-plots. The price spikes observed in these anomalous periods align with market events that are often triggered by significant buy/sell orders, news, or other external factors. Anomalies are critical in identifying outlier price movements that deviate significantly from normal market behavior. No anomalies were flagged using the z-score and box plots in the prices, whereas there were few outliers in the volume according to the box plot.


Box Plots for Close, Open, High, Low, and Volume
Close
5.55M
5.5M
5.45M
5.4M
Close
Open
5.55M
5.5M
5.45M
5.4M
Open
High
5.55M
5.5M
5.45M
5.4M
High
Low
5.55M
5.5M
5.45M
5.4M
Low
Volume
80
60
40
20
0
Volume

4.4 Average Hourly Price Chart The average hourly price chart was plotted to capture intra-day price patterns. Observations include:

- Price tends to show more variability in certain hours, particularly around market opening and closing times for major global exchanges.
- The average price is relatively stable during certain periods of the day, which is useful for identifying low-volatility trading windows where large price movements are less likely.



5. Time Series Analysis

5.1 OHLC Price Trends

The OHLC (open, high, low, close) plots provide valuable insights into price movements throughout the observed period. Key findings include:

- A **notable price spike** shortly after the start of October 7th, which corresponds to a sudden surge in trading activity, likely indicating significant buy/sell orders.
- An **upward trend** in prices, with a consistent increase in both highs and lows, suggesting a bullish market sentiment during the analysis period.

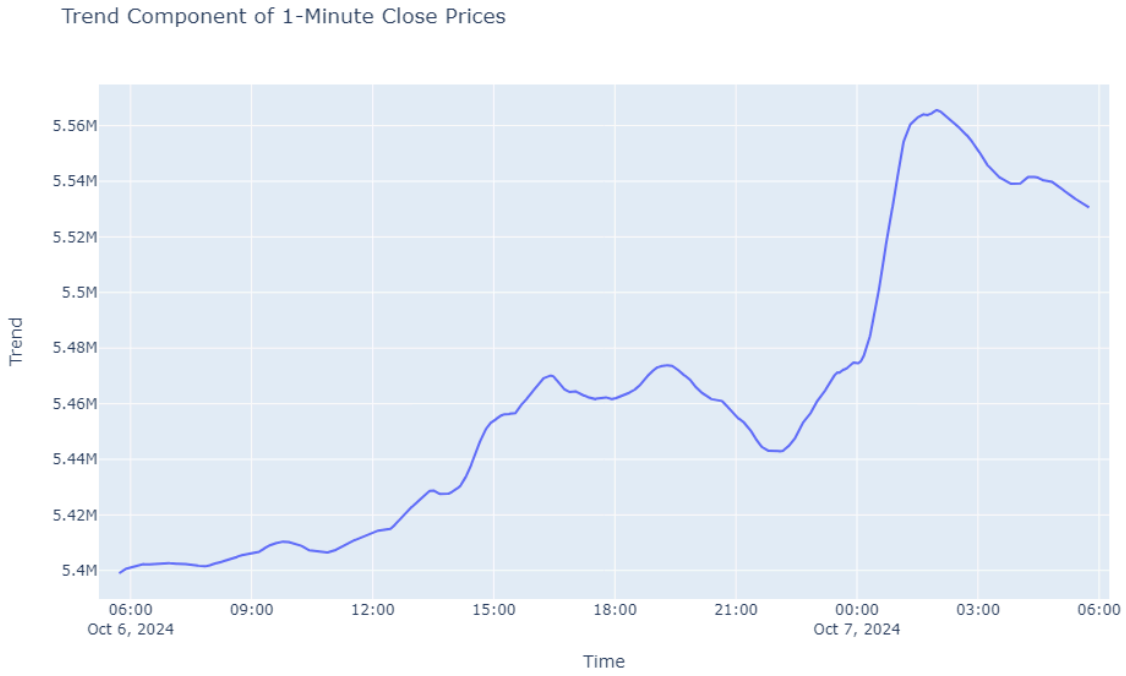
These trends highlight periods of strong price momentum and potential opportunities for market entry during upward movements.

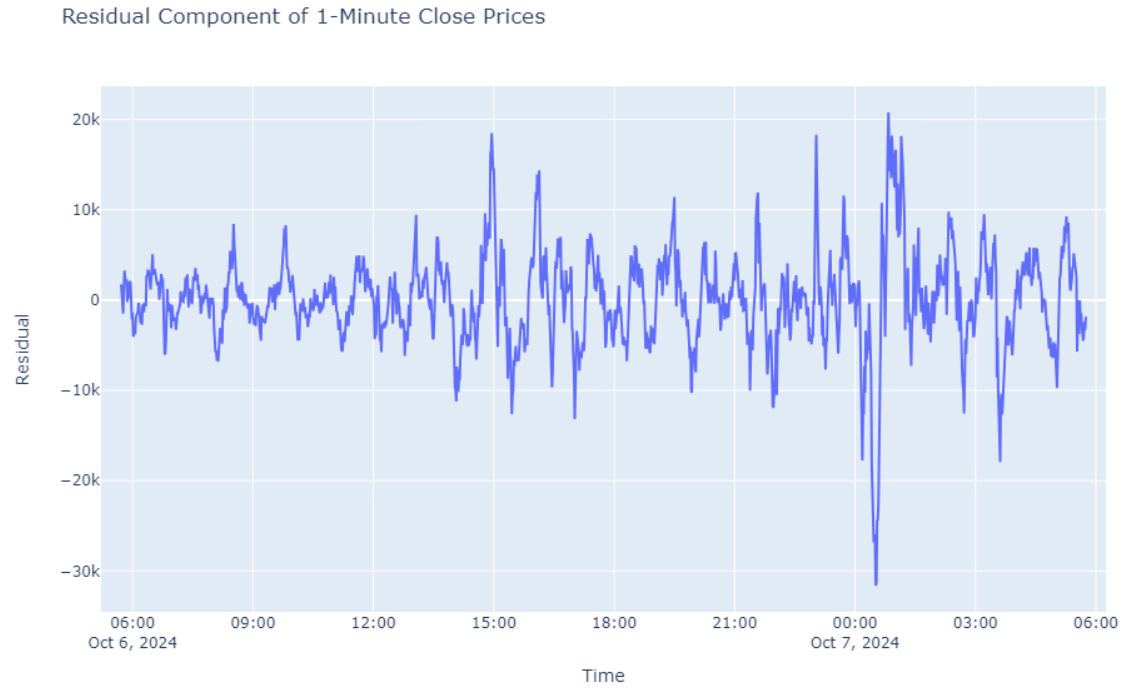
5.2 Seasonality and Trend Decomposition

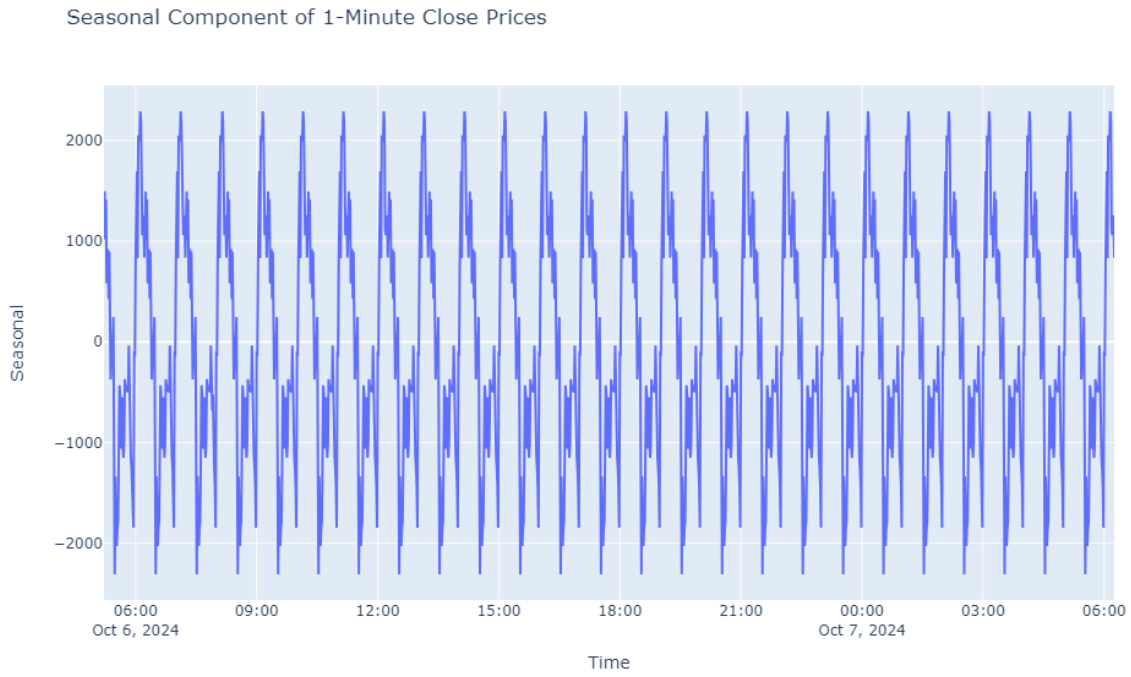
Using **Seasonal and Trend decomposition using Loess (STL)**, the time series was decomposed into three components:

- **Trend**: A smooth representation of long-term price movements, removing the short-term fluctuations.

- **Seasonality**: Repeating patterns observed at fixed intervals. In this case, subtle seasonality patterns were identified, although cryptocurrency markets exhibit less clear-cut seasonality compared to traditional markets.
- **Residual**: The noise or irregular component, which includes any unexpected price movement not explained by trend or seasonality







This decomposition helped isolate long-term trends from short-term seasonal effects, providing better insights into underlying market conditions and price trends.

5.3 Partial Autocorrelation Function (PACF) and Autocorrelation Function (ACF) Analysis

The **Autocorrelation Function (ACF)** measures how the current price is correlated with past prices over various time lags. For the BTCINR dataset, the ACF plot revealed significant autocorrelation at shorter lags, with the correlation gradually decreasing as the lag increased. This indicates that recent prices strongly influence subsequent price movements, with diminishing effects as time goes on.

In contrast, the **Partial Autocorrelation Function (PACF)** helps isolate the direct influence of each lag on the current price. The PACF plot showed a sharp drop after the second lag, with significant partial autocorrelations observed at lag 1 and lag 60. These findings were used to create lagged features:

- **Close_Lag_1**: Captures the immediate price momentum from the previous minute.
- **Close_Lag_60**: Reflects the trend from one hour earlier, offering a broader perspective on price changes over a longer timeframe.

The steep decline in the PACF after the second lag further supports the inclusion of these specific lags, as they provide the most predictive value for the model.

5.4 Rolling Statistics

The rolling mean, maximum, and minimum were computed and plotted using a 30-minute time window. These rolling statistics are valuable for identifying:

- **Long-term price trends** while filtering out short-term volatility.

- At certain time steps, the **rolling minimum and maximum values became constant**, indicating price stability during those periods.
- The **rolling mean closely followed the close price**, providing a smoother representation of the overall price movement.

This analysis helps to distinguish stable price periods from more volatile ones, offering a clearer view of the underlying trend.

5.5 Stationarity Analysis

To ensure the reliability of the time series modeling, a **stationarity analysis** was conducted using the **Augmented Dickey-Fuller (ADF) test**. This statistical test is used to determine whether a time series is stationary or has a unit root, indicating non-stationarity.

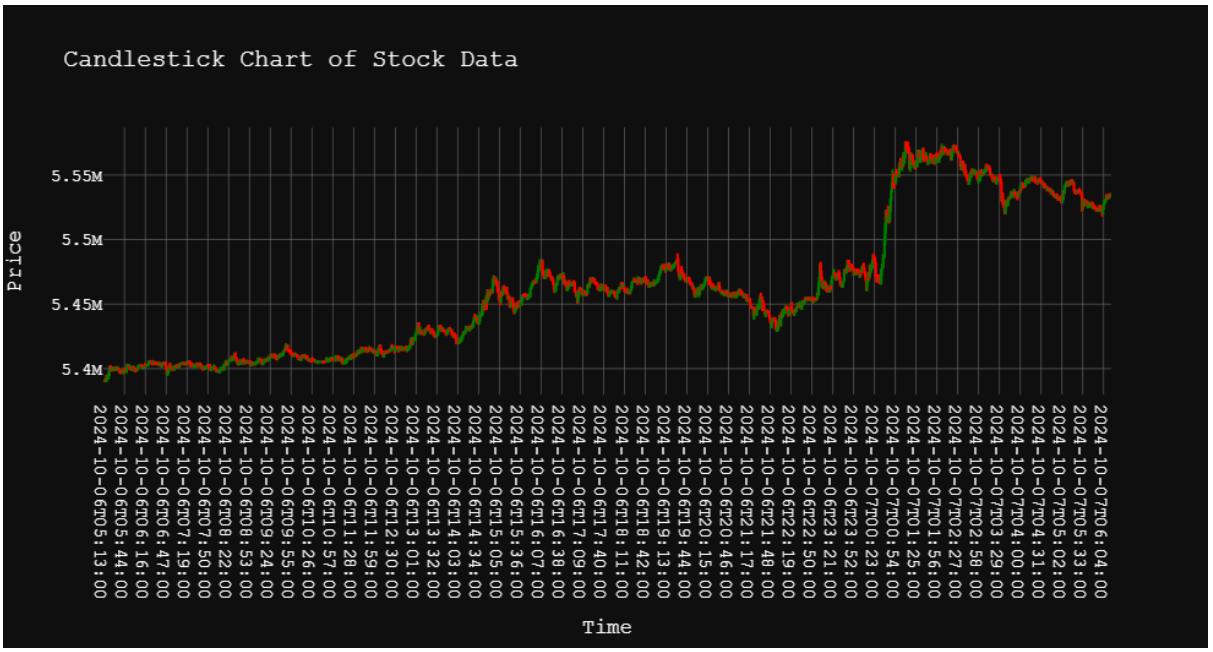
The results of the ADF test revealed that the p-values were greater than 0.05 or 0.1, suggesting that the null hypothesis of non-stationarity could not be rejected. This indicates that the time series exhibits trends or seasonality, which can adversely affect the model's performance.

To address this issue, further transformations, such as differencing or detrending, may be required in subsequent analyses to achieve stationarity. Ensuring stationarity is critical for accurate predictions, as many time series models assume a constant mean and variance over time.

	Values	Metric
0	-0.911561	Test Statistics
1	0.784141	p-value
2	3.000000	No. of lags used
3	1496.000000	Number of observations used
4	-3.434729	Critical value (1%)
5	-2.863474	Critical value (5%)
6	-2.567800	Critical value (10%)

6. Candlestick Analysis

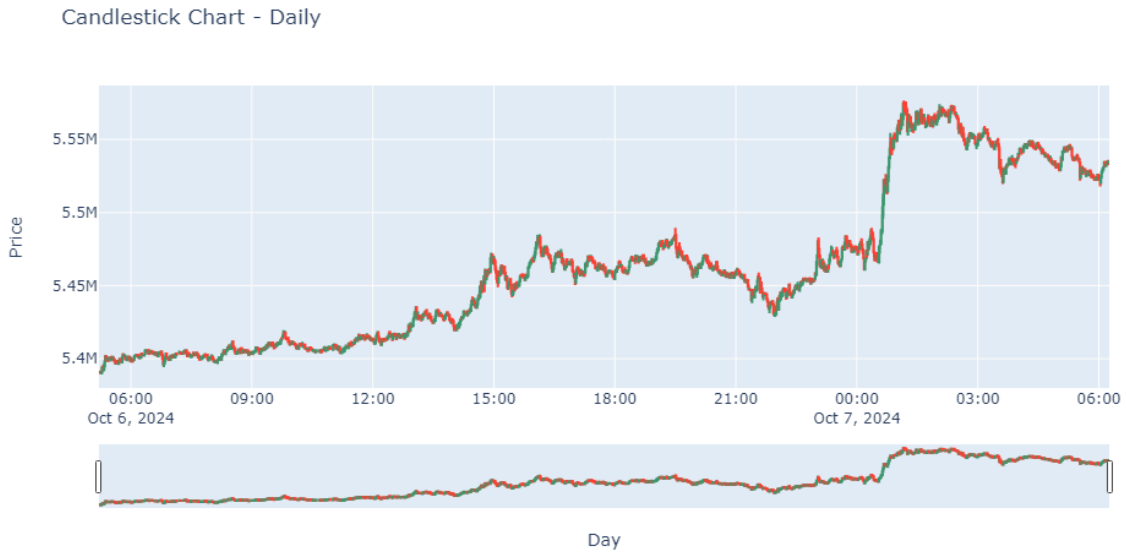
6.1 Hourly and Daily Charts

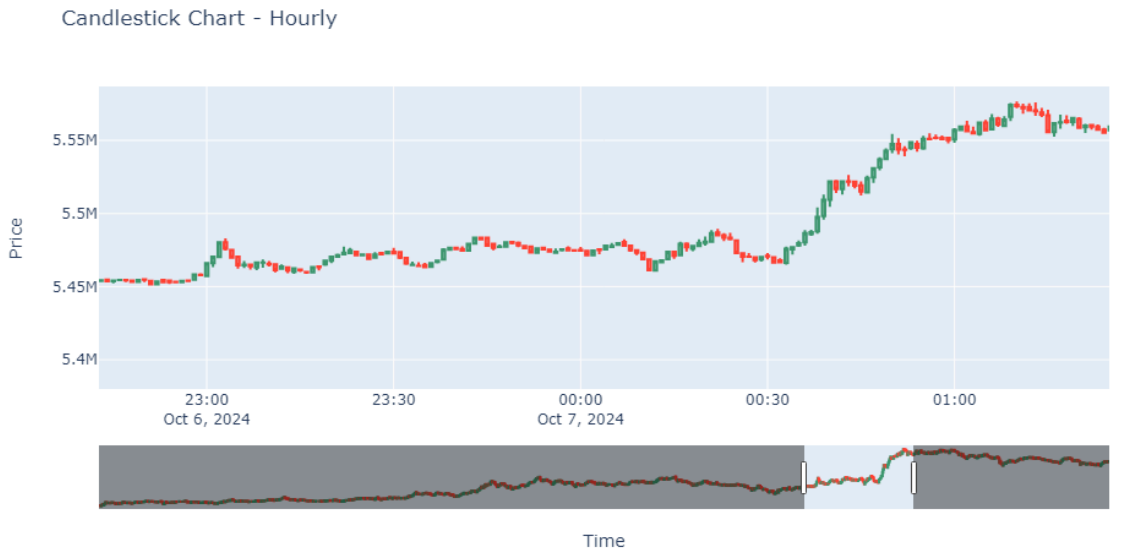


I plotted minute, hourly, and daily candlestick charts, which primarily showed **small red candles** with a few **larger green candles** indicating upward momentum. The overall trend was **bullish**, with prices steadily moving upward. These candlestick patterns suggest:

- **Small red candles** indicate minor price corrections or temporary downward movements.
- **Larger green candles** signal stronger upward momentum, suggesting potential buying pressure.
- The **upward trend** in these charts provides valuable insights into market conditions and potential buy/sell opportunities.

These candlestick charts serve as a foundation for understanding price reversals and identifying buy/sell opportunities.





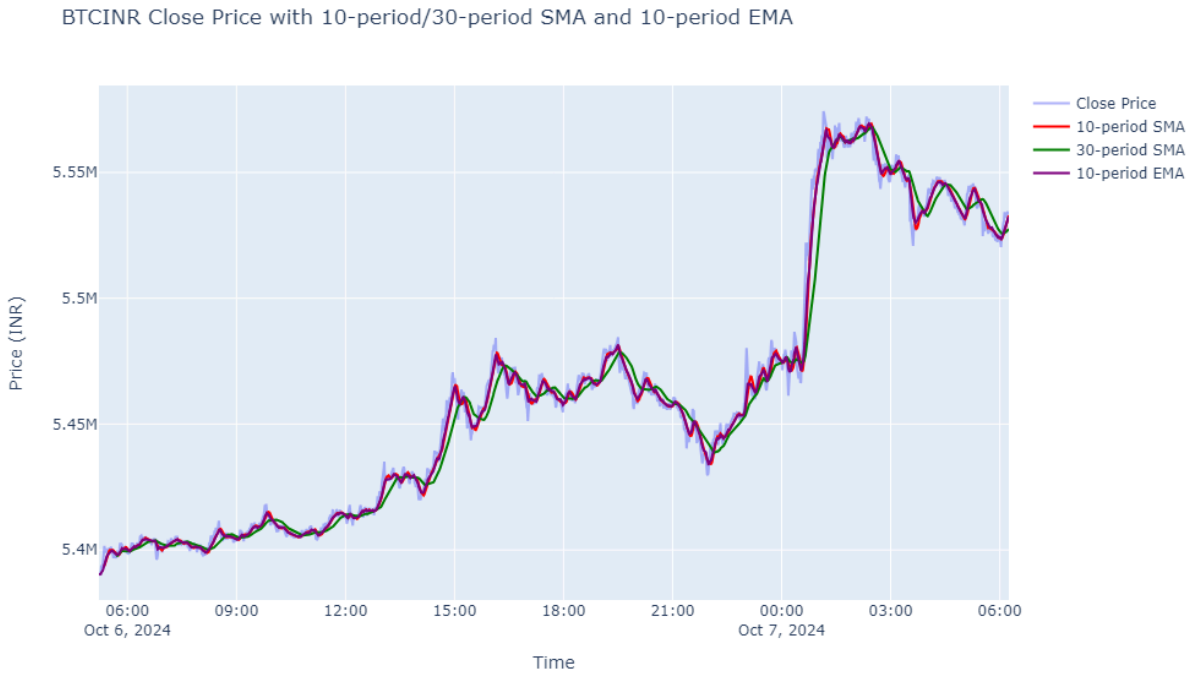
7. Technical Indicators

In this analysis, 10 key technical indicators were computed to provide a comprehensive understanding of price movements, momentum, and volatility. These indicators are essential tools for identifying potential trading signals, market conditions, and price trends.

7.1 Moving Averages (SMA & EMA)

The **Simple Moving Average (SMA)** and **Exponential Moving Average (EMA)** are used to smooth price data and highlight trends.

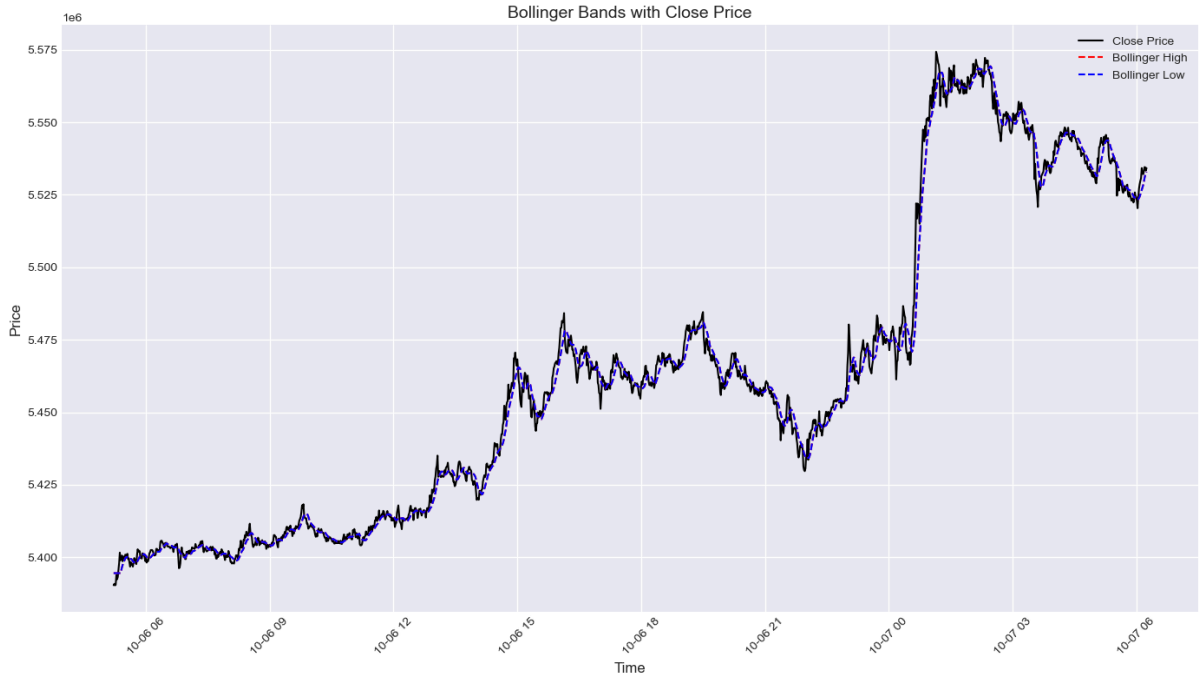
- **SMA** (5-period) crossing above a longer-term **SMA** (20-period) can signal a potential buy opportunity.
- **EMA**, with more weight given to recent data, is particularly useful for detecting **recent price movements** and tends to react faster than the SMA to market changes.



7.2 Bollinger Bands

Bollinger Bands plot an upper and lower boundary around the price based on standard deviation, indicating volatility.

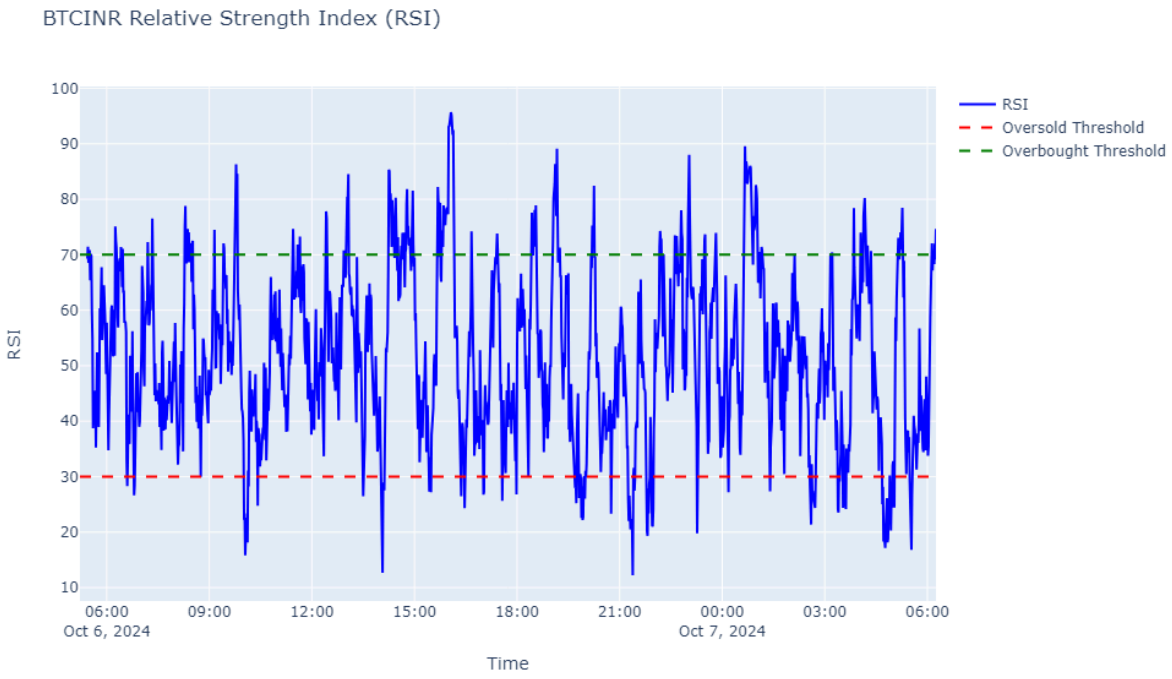
- When the price touches the **upper band**, it may signal **overbought** conditions.
- When it touches the **lower band**, it suggests **oversold** conditions, possibly indicating a reversal.



7.3 Relative Strength Index (RSI)

The **RSI** is a momentum oscillator used to determine whether a security is **overbought** or **oversold**:

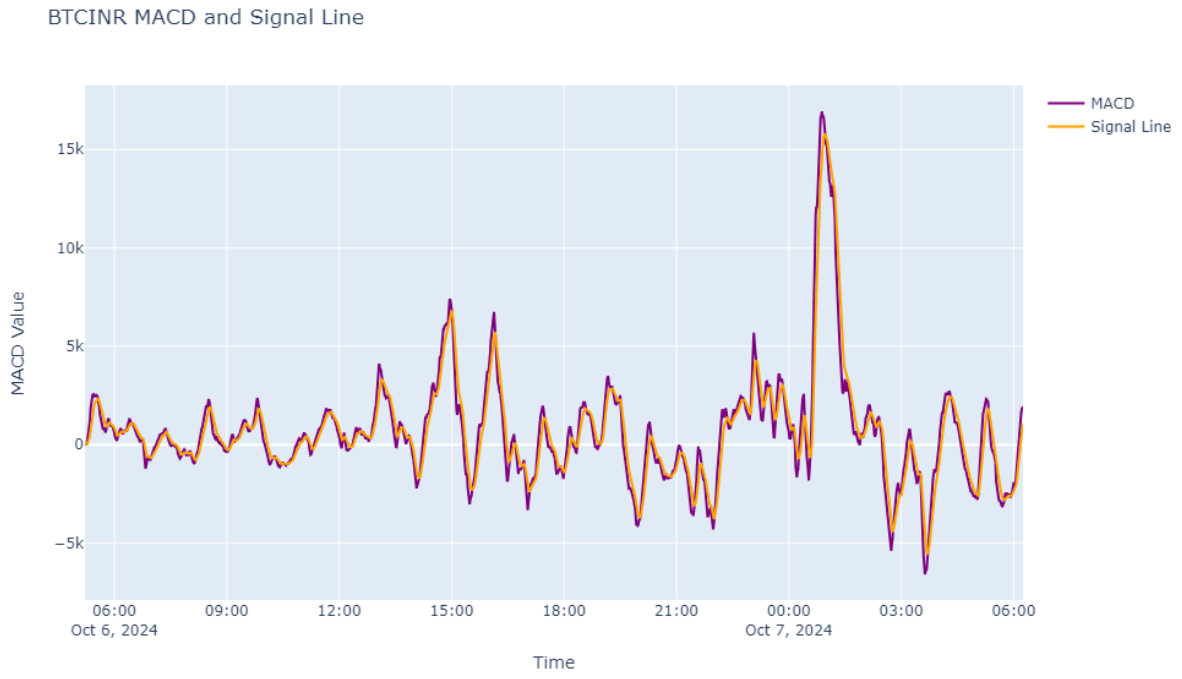
- **RSI > 70** indicates overbought conditions, potentially signaling a correction.
- **RSI < 30** indicates oversold conditions, suggesting the price might rebound.



7.4 Moving Average Convergence Divergence (MACD)

The **MACD** measures the difference between short-term and long-term EMAs to identify trend strength:

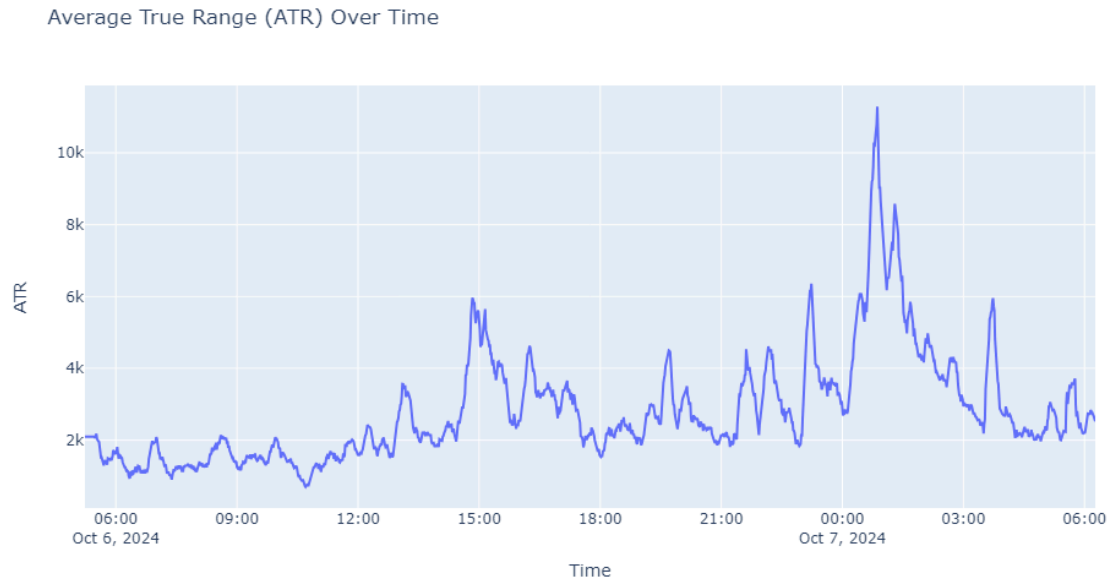
- A **bullish signal** occurs when the MACD line crosses **above** the signal line.
- A **bearish signal** occurs when the MACD line crosses **below** the signal line.



7.5 Average True Range (ATR)

The **ATR** measures market volatility by analyzing the range of price movement over a period.

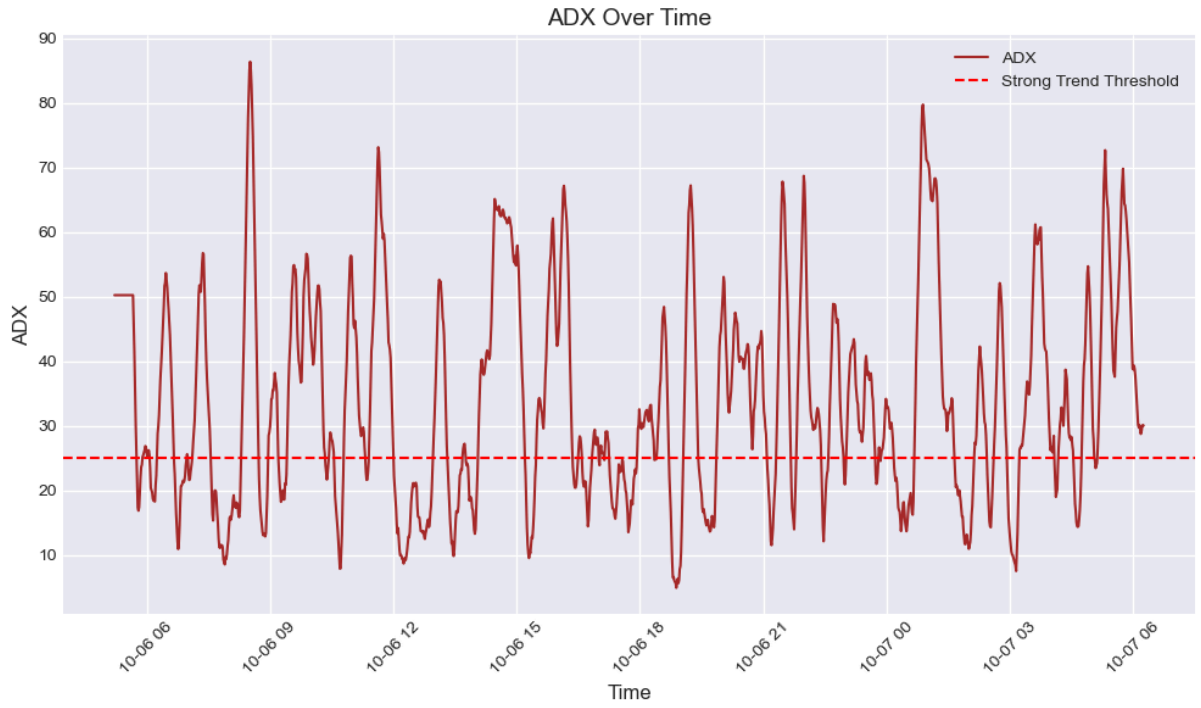
- Higher ATR values indicate **greater volatility** and potential risk, while lower values suggest quieter markets.



7.6 Average Directional Movement Index (ADMI)

The **ADMI** is used to quantify the strength of a trend:

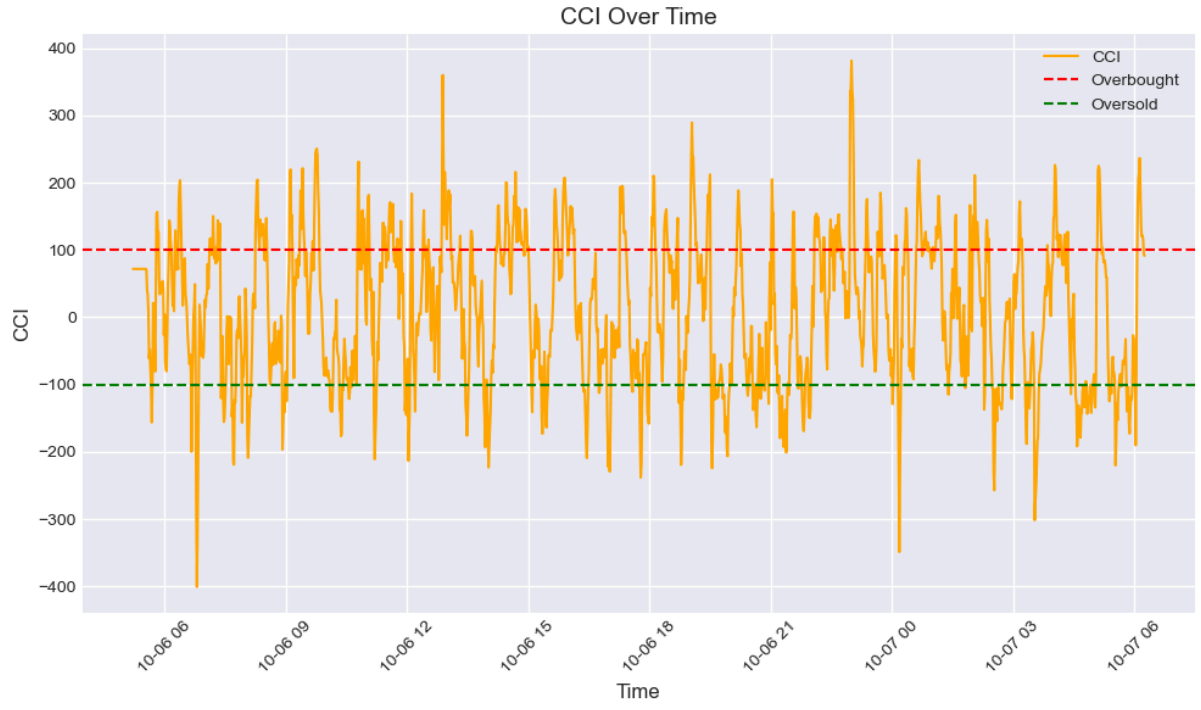
- A reading above **25** suggests a **strong trend**, while values below that indicate weak or nonexistent trends.



7.7 Commodity Channel Index (CCI)

The **CCI** helps identify cyclical trends by comparing the current price to its moving average:

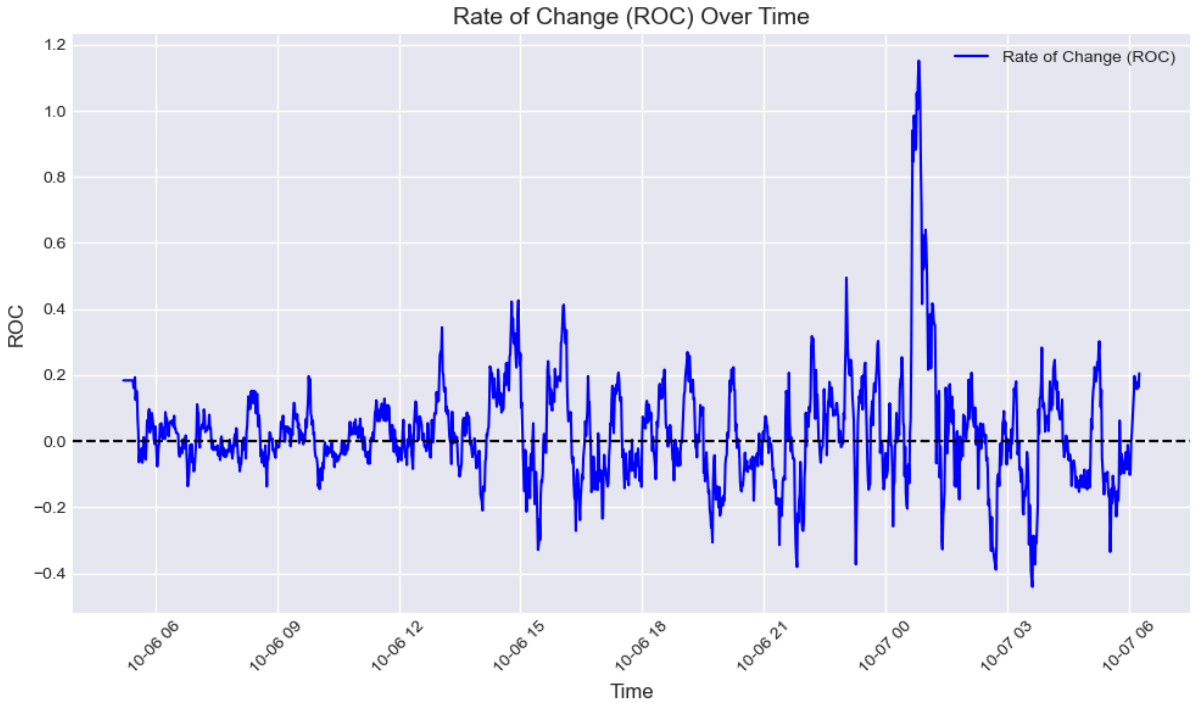
- Values above **100** suggest an **overbought** market, while values below **-100** signal an **oversold** market.



7.8 Price Rate of Change (ROC)

The **ROC** measures the percentage change in price over a specific period:

- A positive ROC indicates **upward momentum**, while a negative ROC points to **downward momentum**.



7.9 William's %R Oscillator

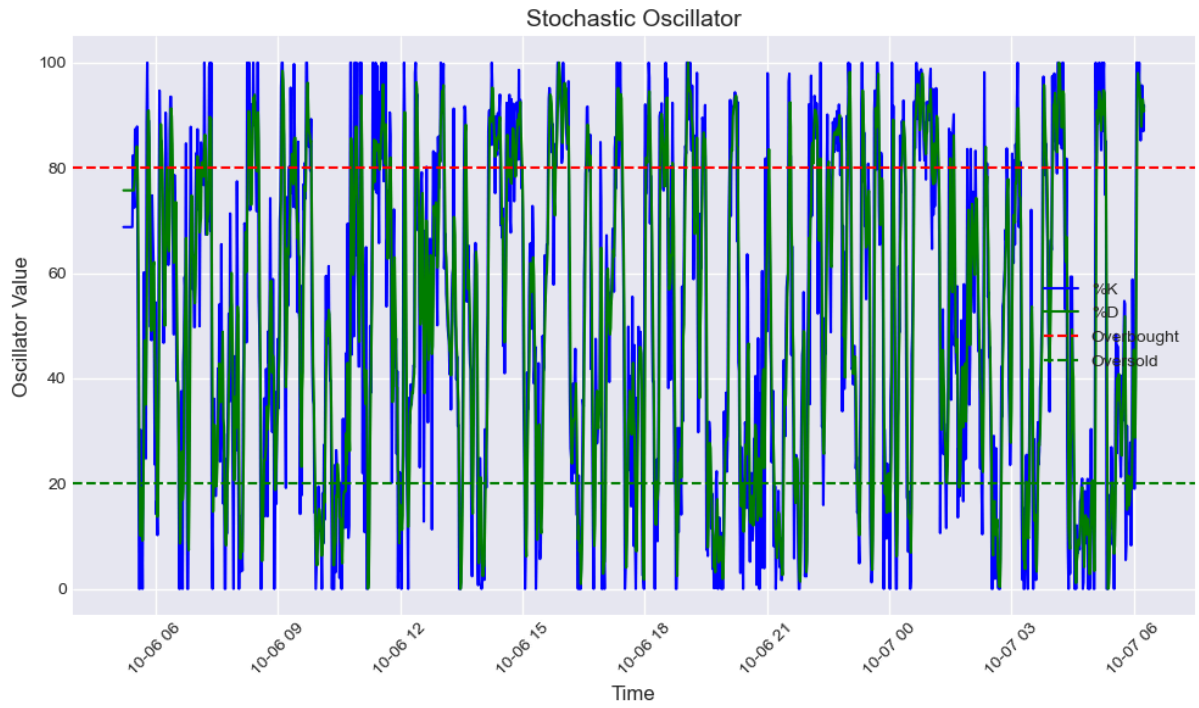
The **William's %R** oscillator determines whether the market is **overbought** or **oversold**:

- Values between **-20** and **-0** suggest an **overbought** condition.
- Values between **-80** and **-100** indicate an **oversold** condition.

7.10 Stochastic Oscillator (%K and %D)

The **Stochastic Oscillator** compares the closing price of a security to its price range over a specific time period:

- **%K** measures the current closing price relative to the high-low range.
- **%D**, the **3-period moving average** of %K, helps smooth out fluctuations.
- A %K crossing **above %D** suggests a **buy signal**, while crossing **below %D** signals a potential **sell**.

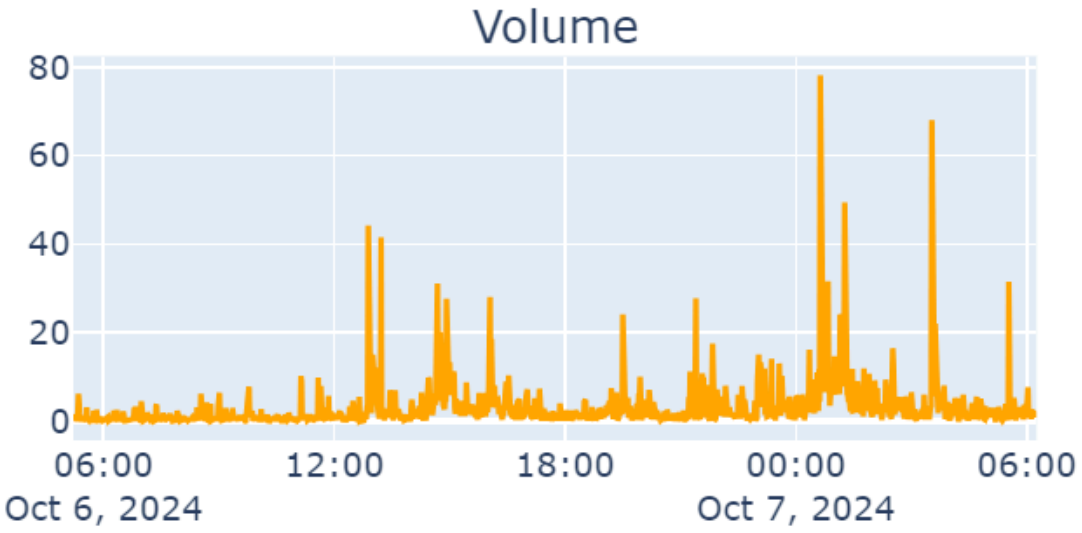


These technical indicators were used in combination to identify critical patterns, volatility shifts, and momentum changes that can signal potential trading opportunities in BTCINR.

8. Volume and Returns Analysis

8.1 Volume Trends

Volume trends were analyzed to determine periods of heightened market activity. A sudden increase in volume often precedes significant price movements, highlighting the importance of volume as a predictive feature.

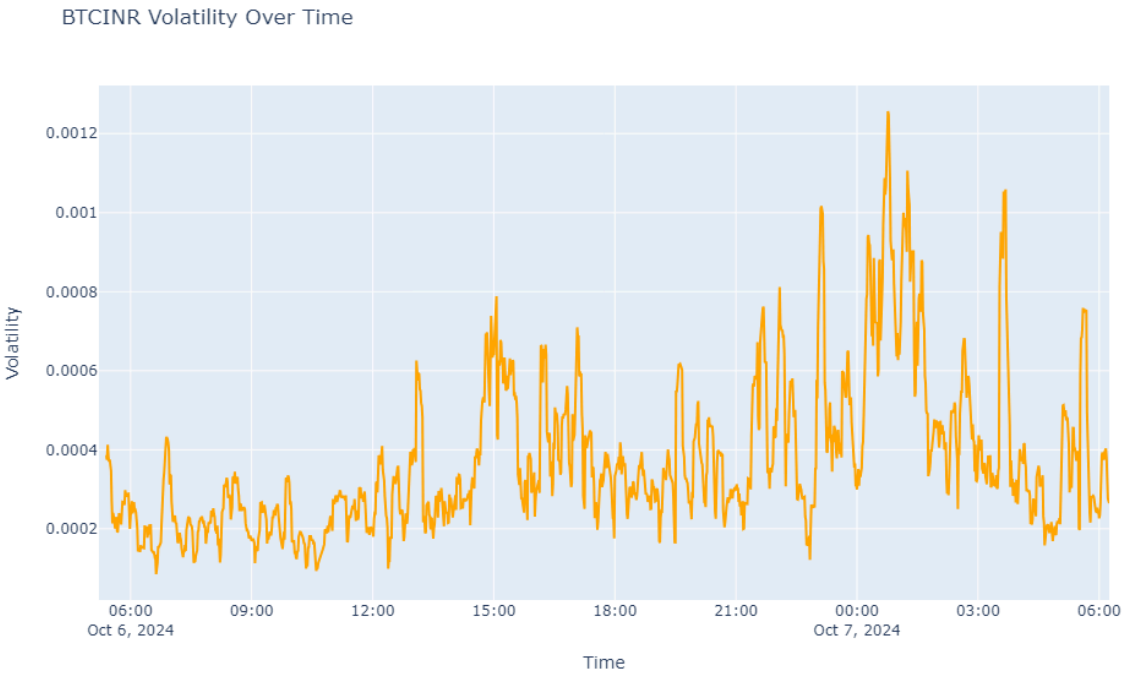


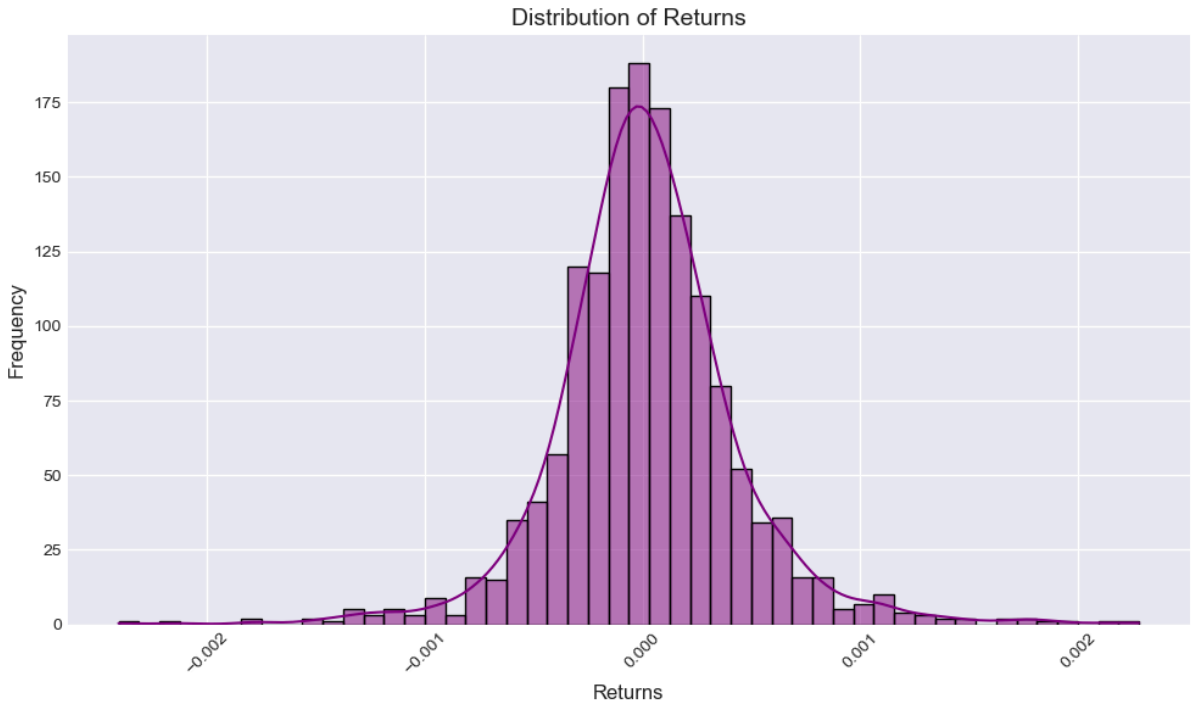
Again, the insights from the graph convey that the new day resulted in an increased volume of trades with abrupt spikes and high volatility.

8.2 Returns and Volatility

Returns are calculated as the logarithmic change in closing prices, and volatility is computed using rolling standard deviations. Important observations include:

- High returns are often followed by periods of increased volatility, consistent with the notion of volatility clustering in financial markets.

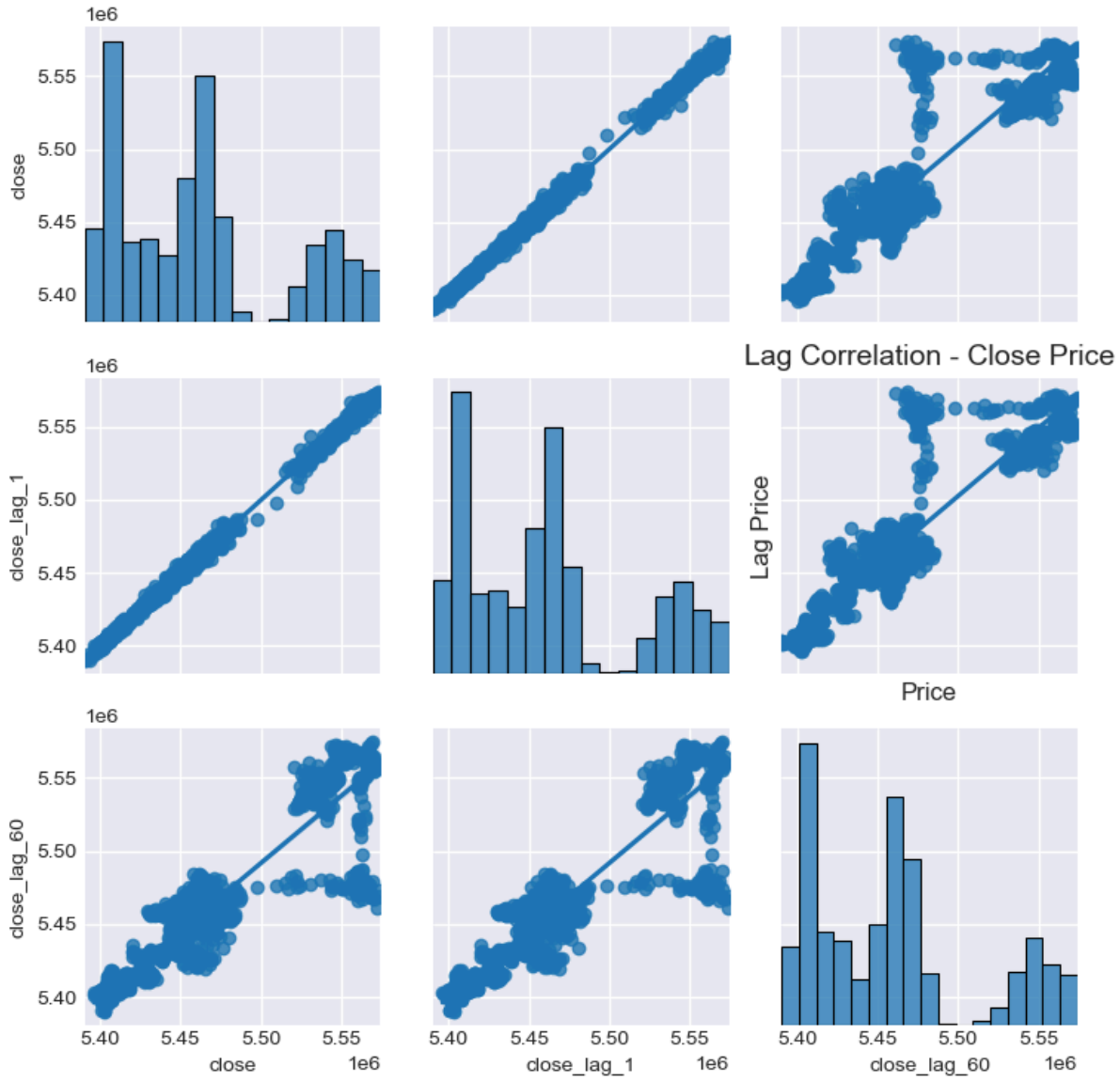




9. Advanced Feature Engineering

9.1 Lag Features and Momentum Indicators

Lag features such as `close_lag_1` and `close_lag_60` provide insight into price momentum. **Momentum indicators** like the **Rate of Change (ROC)** and **Williams %R** were also engineered to capture momentum trends.

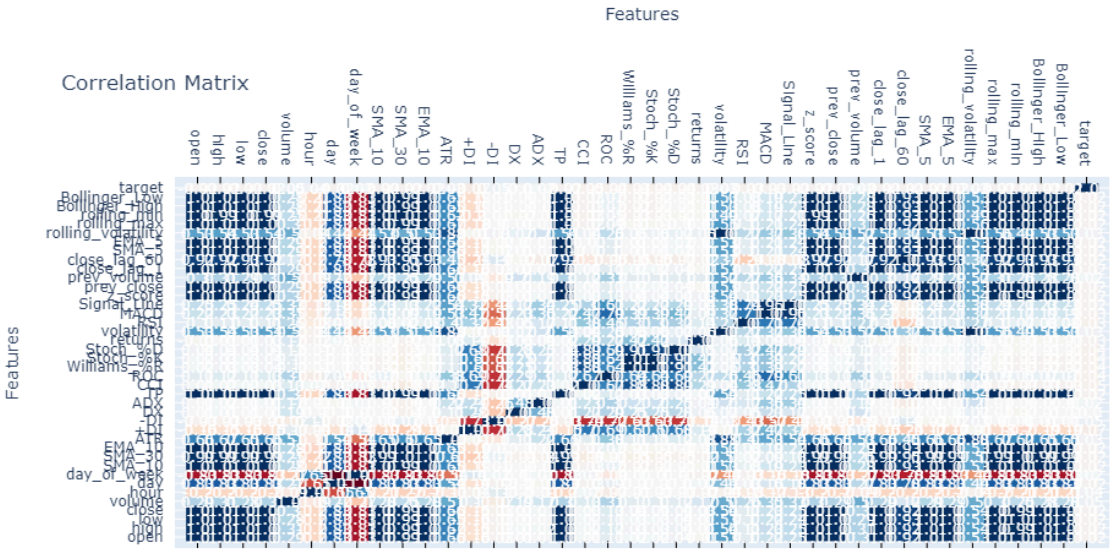


9.2 Rolling Volatility

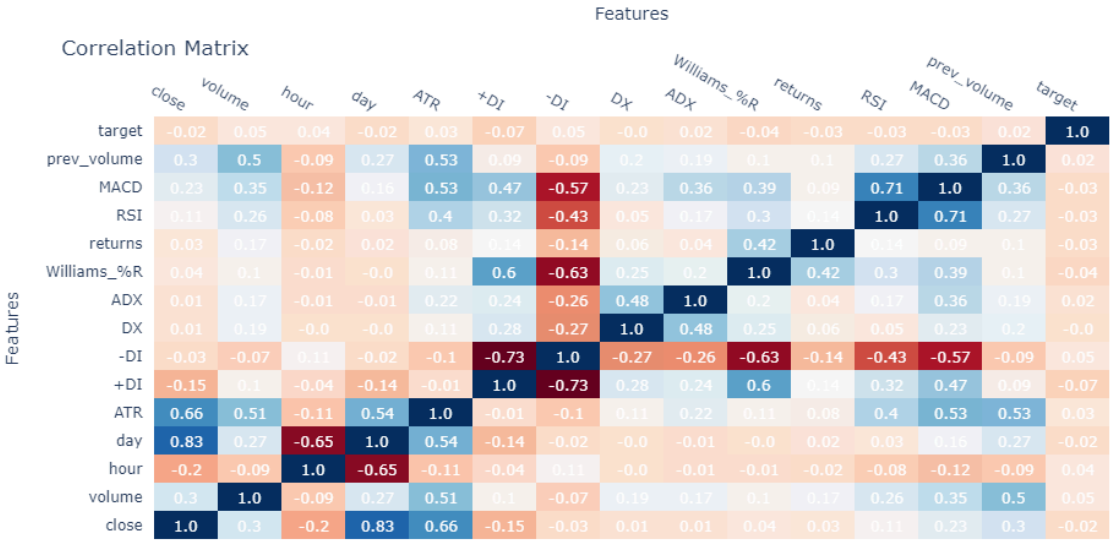
Rolling volatility was computed over different time windows to capture both short-term fluctuations and long-term market trends. High volatility periods are often associated with uncertainty and risk in the market.

10. Correlation Analysis

To avoid multicollinearity and redundancy in the dataset, a thorough **correlation analysis** was performed on the engineered features. Using a correlation matrix, highly correlated features (with correlation values above 0.75) were identified and removed to prevent model overfitting and ensure more accurate predictions. This step was crucial in simplifying the model without losing significant predictive power.



After eliminating these highly correlated features, a refined subset of the DataFrame was obtained, which retained only the most informative and independent features. This subset became the foundation for further model training, helping streamline feature selection for optimal performance.



11. Data Preprocessing

In the **preprocessing** phase, the data was scaled using the **MinMax Scaler** to normalize the range of all features between 0 and 1. This step is essential for ensuring that the model learns efficiently and that features with larger ranges do not dominate those with smaller ranges.

To address the **NaN values** present in the dataset—primarily resulting from the calculation of indicators at the initial rows—a **backward fill (bfill)** method was employed. This approach

filled the missing values with the next valid observation in the column, ensuring that the dataset remained continuous and without gaps.

To prepare the dataset for model training and evaluation:

- **1400 timesteps** were used for **training**, allowing the model to capture the underlying patterns and trends over a substantial period.
- The remaining **100 timesteps** were reserved for **testing/validation**, ensuring a reliable evaluation of the model's predictive performance.

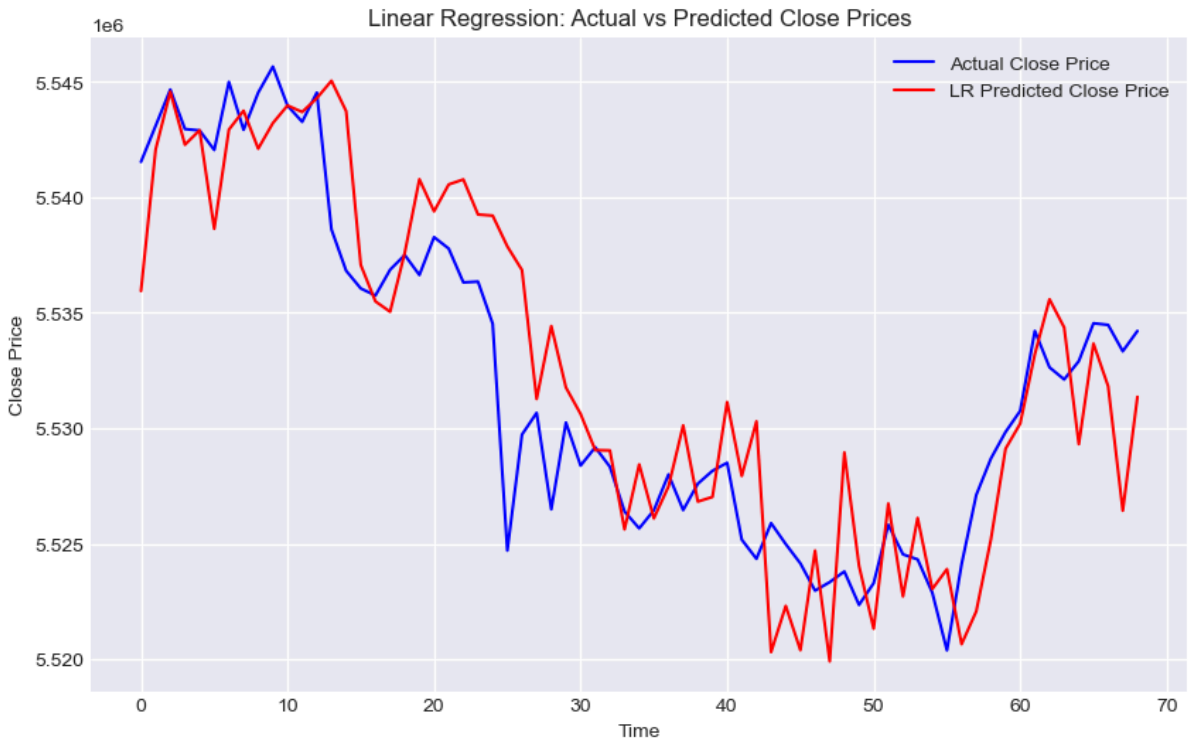
By scaling the data and appropriately splitting it into training and testing sets, the model was able to generalize well to unseen data, while maintaining a robust and effective training process.

12. Predictive Modeling

In this section, various predictive models were implemented to forecast the close price of BTCINR. Each model required specific data manipulation and architecture configurations to optimize their performance.

12.1 Linear Regression (LR)

- **Data Manipulation**: The dataset was divided into training and testing subsets, with the first 1400 rows used for training and the remaining 100 for testing. The features were selected based on their correlation with the target variable. 30 previous time steps were used to predict the next close price.
- **Model Architecture**: A simple linear model that predicts the target variable as a linear combination of input features.

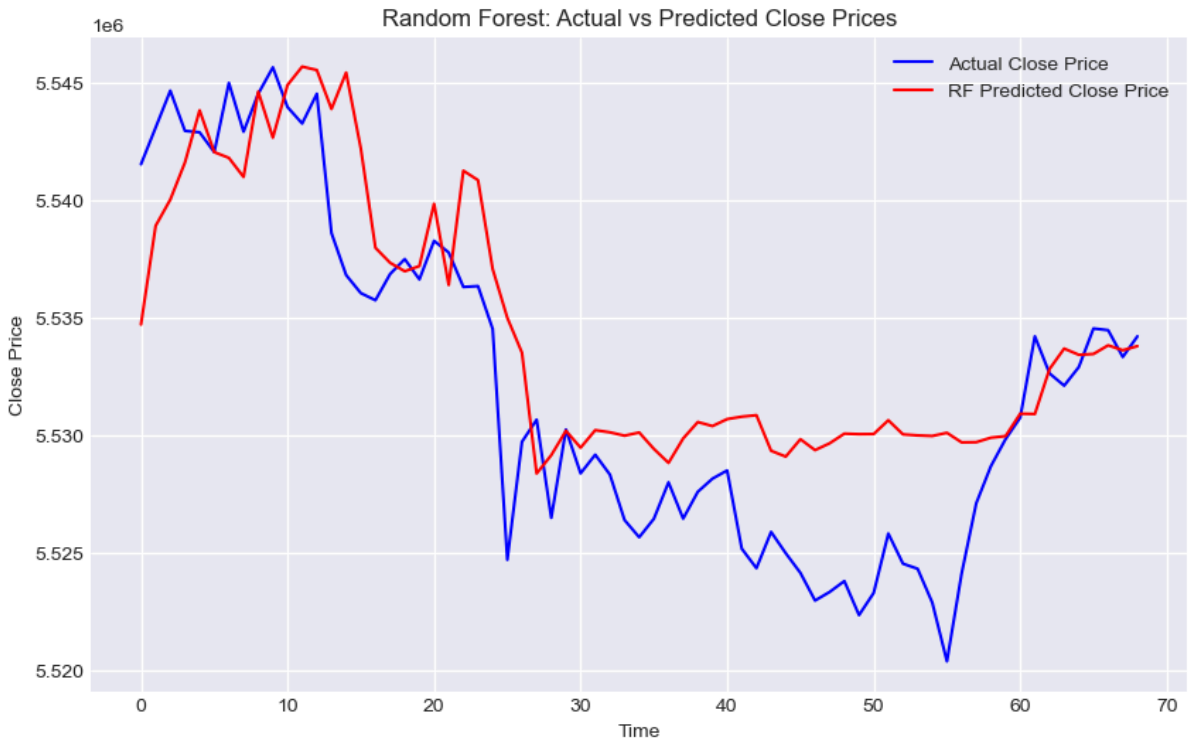



```
Linear Regression MSE: 12515024.260224765
```

12.2 Random Forest Regressor (RF)

- **Data Manipulation**: Similar to LR, the data was split into training and testing sets, and features were selected based on their significance.
- **Model Architecture**: An ensemble model consisting of multiple decision trees, each trained on different subsets of the data to improve accuracy and reduce overfitting.

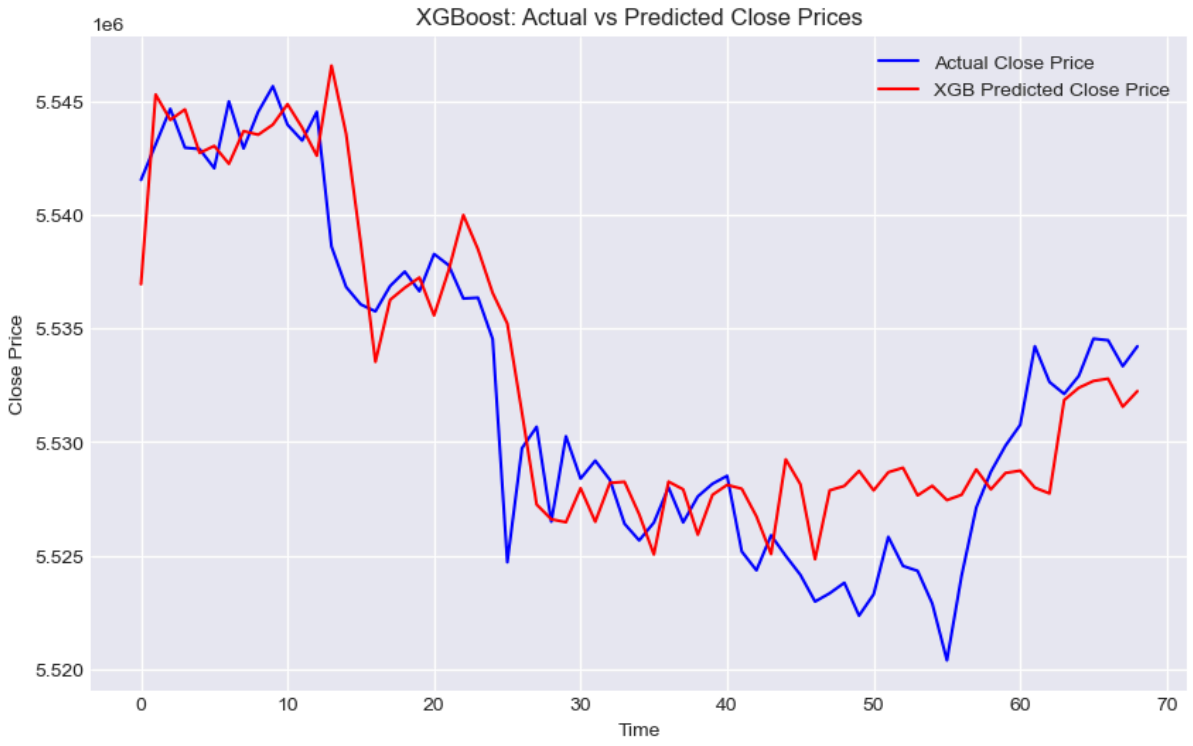
```
Random Forest MSE: 17075294.499639448
```



12.3 XGBoost (XGB)

- **Data Manipulation**: The data underwent the same preprocessing steps as previous models, with careful feature selection to enhance model performance.
- **Model Architecture**: An optimized gradient boosting framework that sequentially builds trees and reduces bias, allowing for high predictive accuracy.

```
XGBoost MSE: 10463505.157760512
```

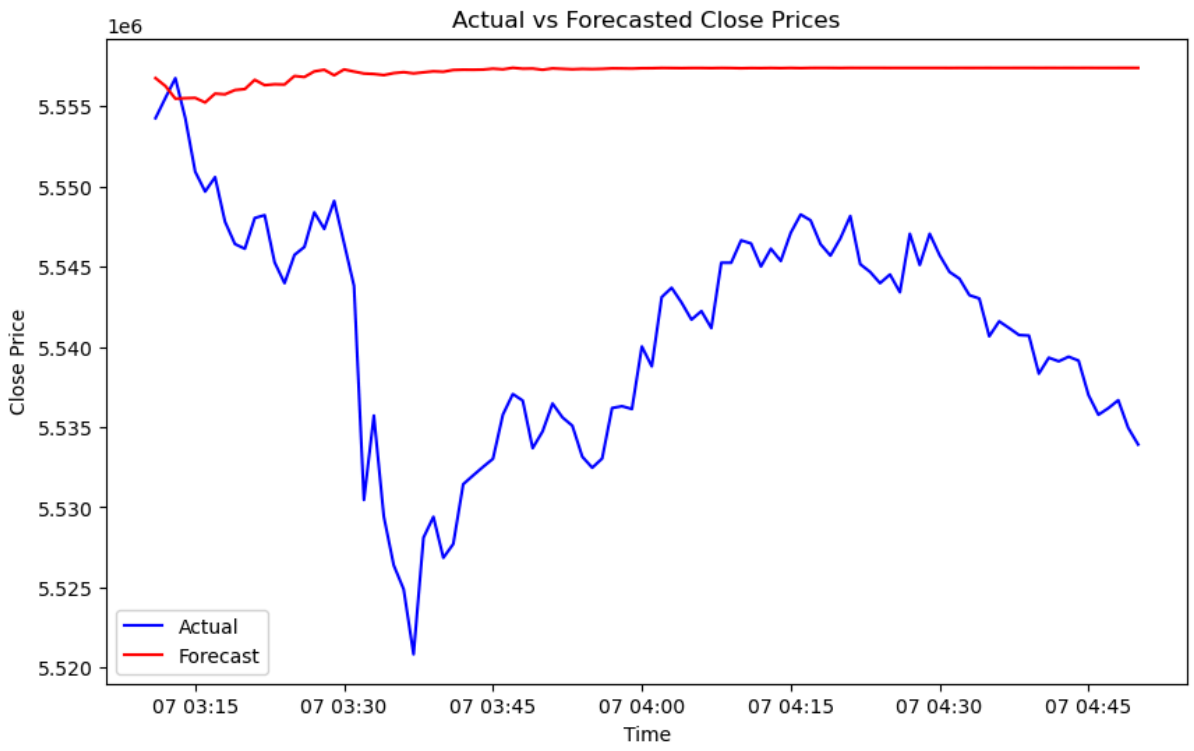


12.4 AutoRegressive Integrated Moving Average (ARIMA)

- **Data Manipulation**: The close prices were scaled using MinMaxScaler. The ARIMA model was fitted on the scaled training data with varying hyperparameters (p, d, q) to determine the best configuration.

Model Architecture: The model captures the linear dependencies in the time series data through autoregressive and moving average components.

```
model = ARIMA(train_scaled, order=(30, 1, 1))
model_fit = model.fit()
forecast_scaled = model_fit.forecast(steps=100)
```

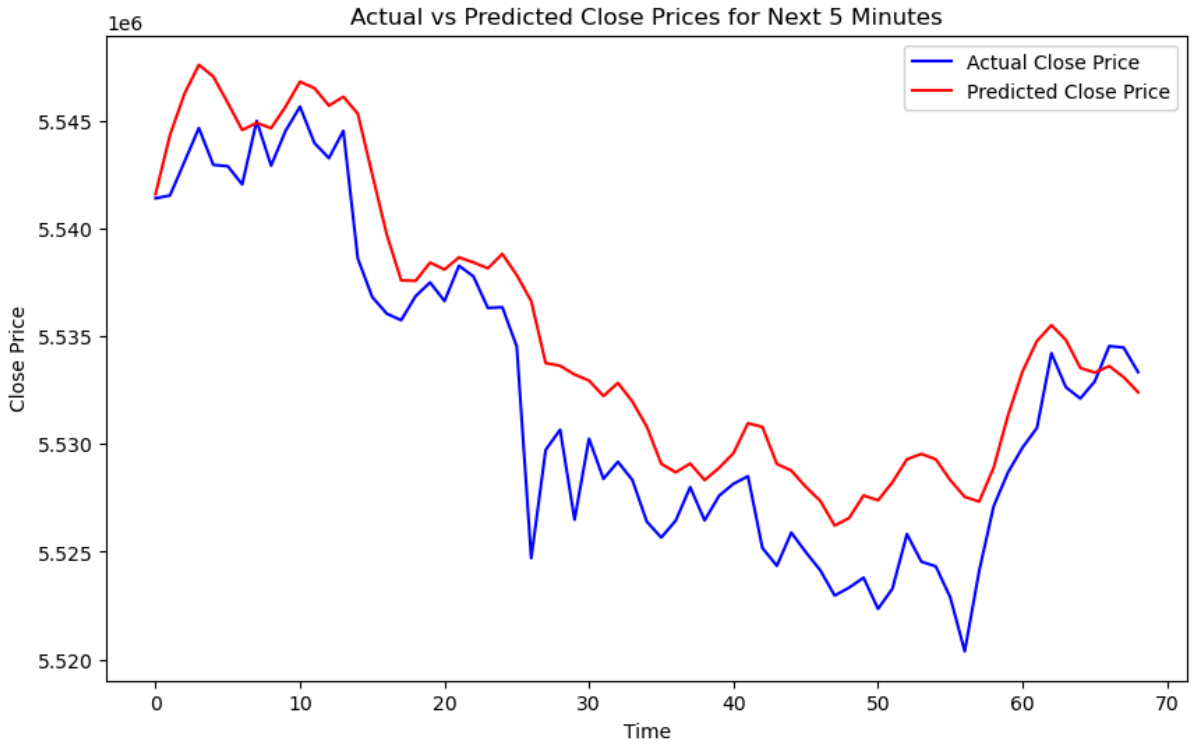


12.5 Long Short-Term Memory (LSTM)

- **Data Manipulation**: Features were engineered, and sequences were created from the scaled dataset. Two versions were implemented: one with engineered features and one without. The model utilized the features from the **past 30 timesteps to predict the next close price.**
- **Model Architecture**:
    - **Input Layer**: Shape defined by timesteps and features.
    - **LSTM Layer**: Captured sequential dependencies with dropout for regularization.
    - **Dense Output Layer**: Predicted the close price.

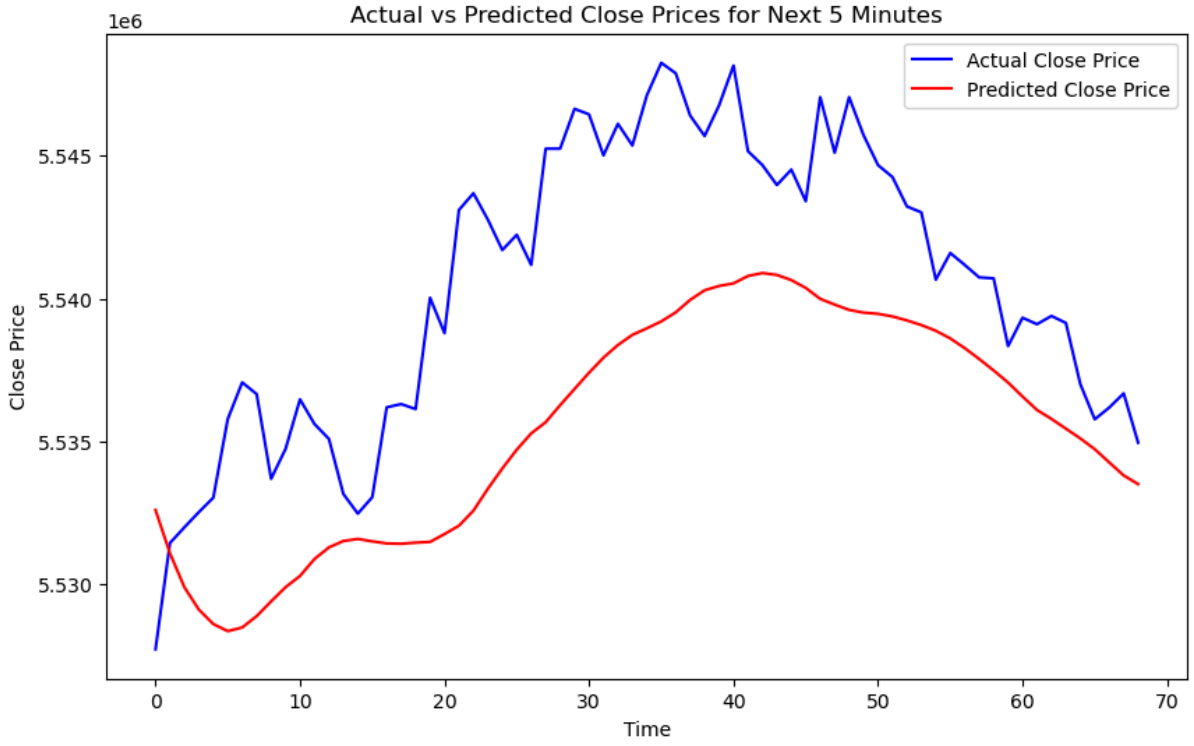
```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(output_timesteps))
```

With Engineered Features



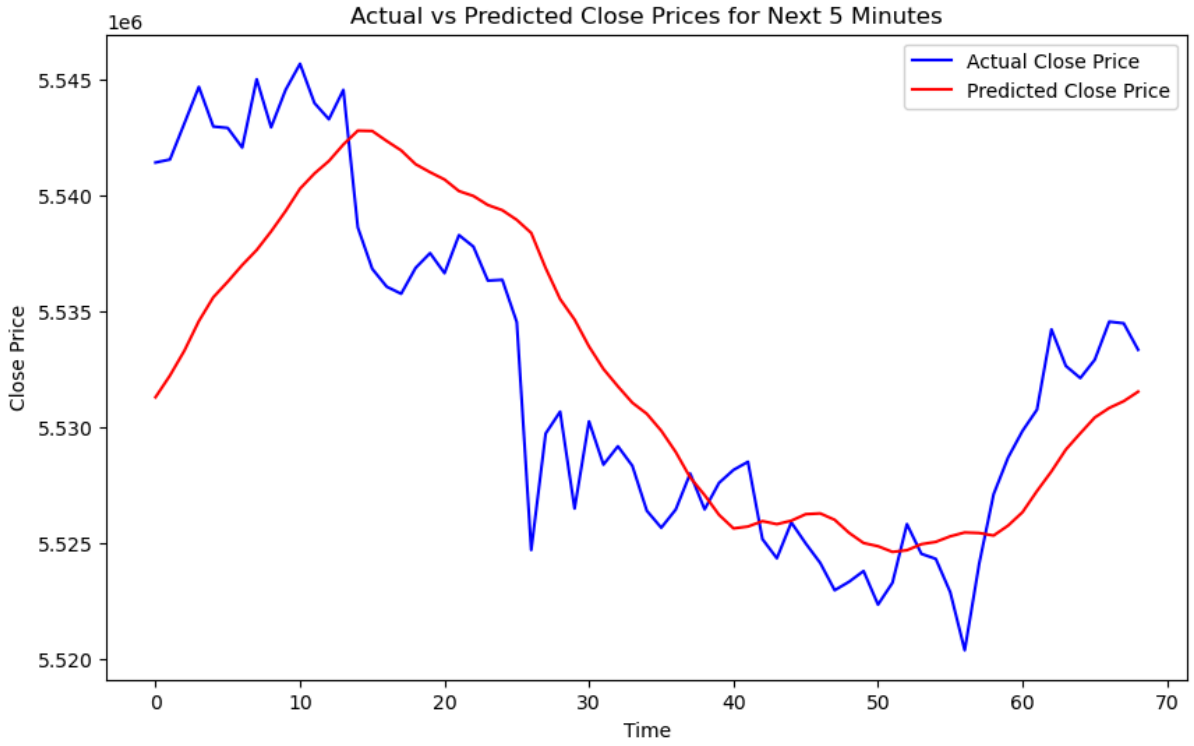

```
Epoch 20/20
44/44 [==============================] - 2s 54ms/step - loss: 0.0020 - val_loss: 3.7756e-04
3/3 [==============================] - 2s 18ms/step
```

Without Engineered Features



12.6 LSTM with Attention

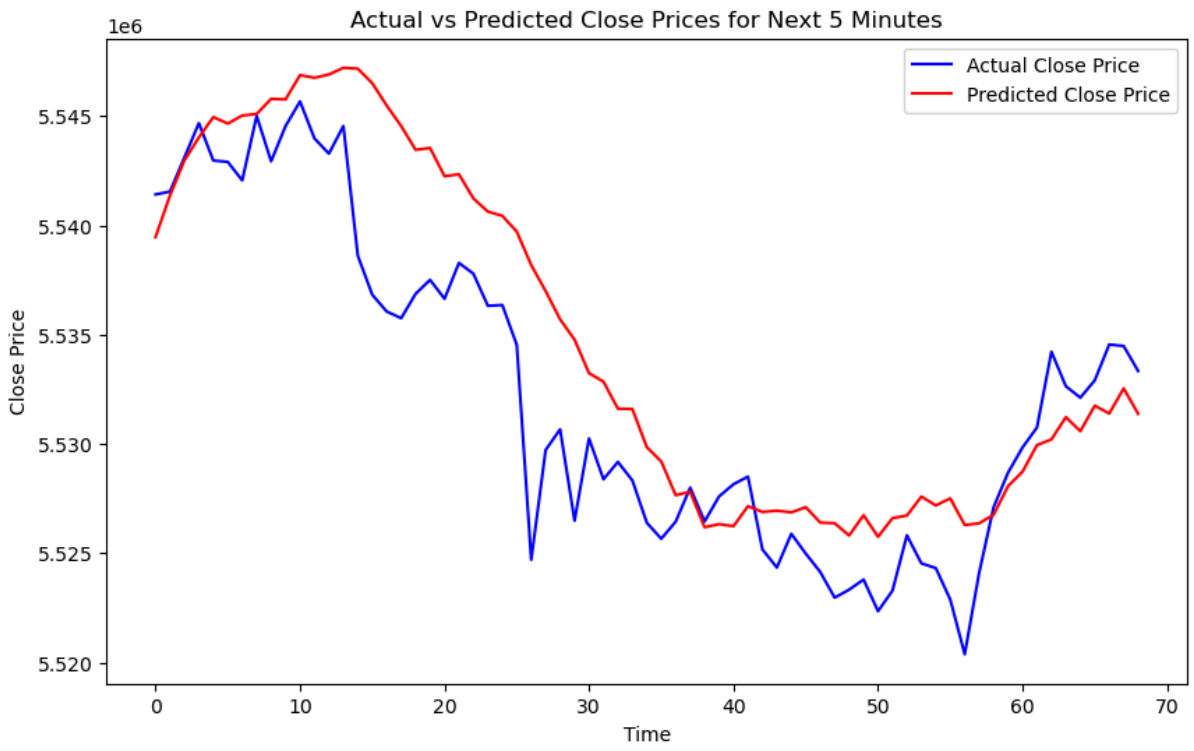
- **Data Manipulation**: Similar to the basic LSTM, but included an attention mechanism to focus on important time steps in the input sequence.
- **Model Architecture**: Combined LSTM layers with an attention layer to enhance predictive capabilities.



12.7 CNN + LSTM

- **Data Manipulation**: The same feature set was used, with the data prepared in sequences suitable for convolutional operations followed by LSTM layers.
- **Model Architecture**:
    - **CNN Layer**: Extracted features from the input sequence.
    - **LSTM Layer**: Captured temporal patterns in the features.
    - **Dense Output Layer**: Predicted the close price.

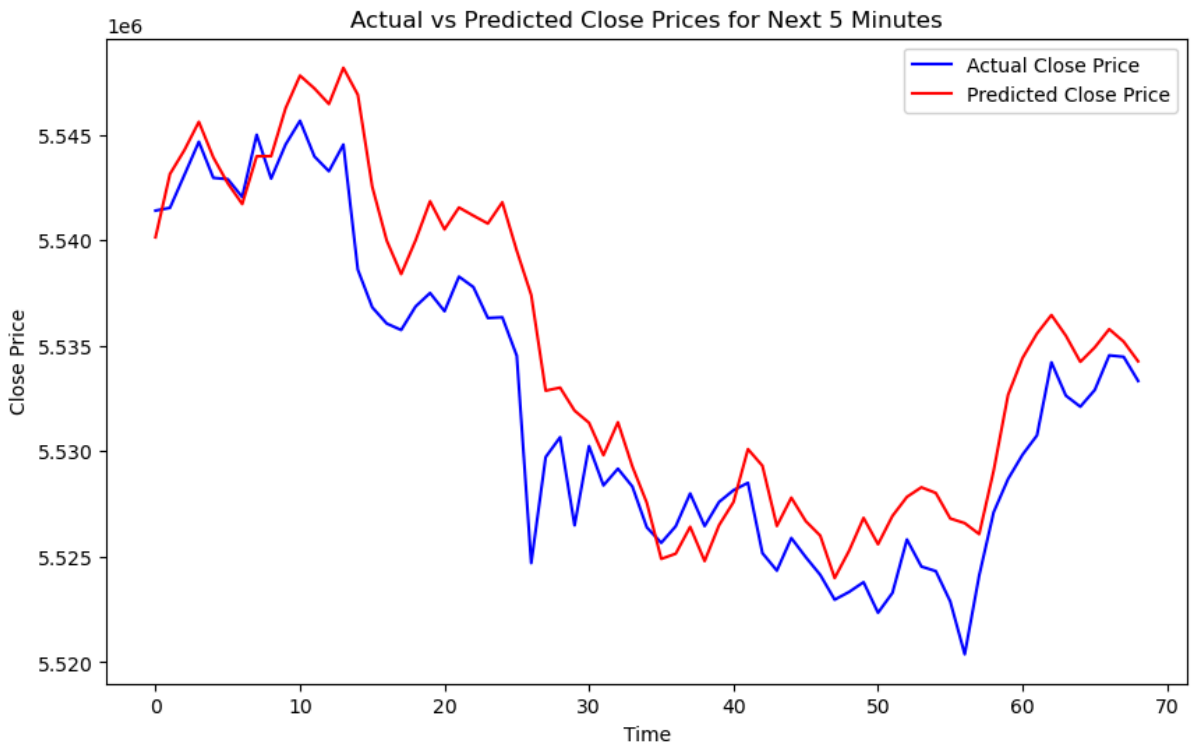
```
model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(units=50))
```



12.8 Bidirectional LSTM (BiLSTM)

- **Data Manipulation**: The data was prepared similarly to the LSTM, capturing relationships from both forward and backward directions.
- **Model Architecture**:
    - **Two LSTM Layers**: One processing sequences from start to end and the other in reverse, allowing the model to capture complex relationships more effectively.

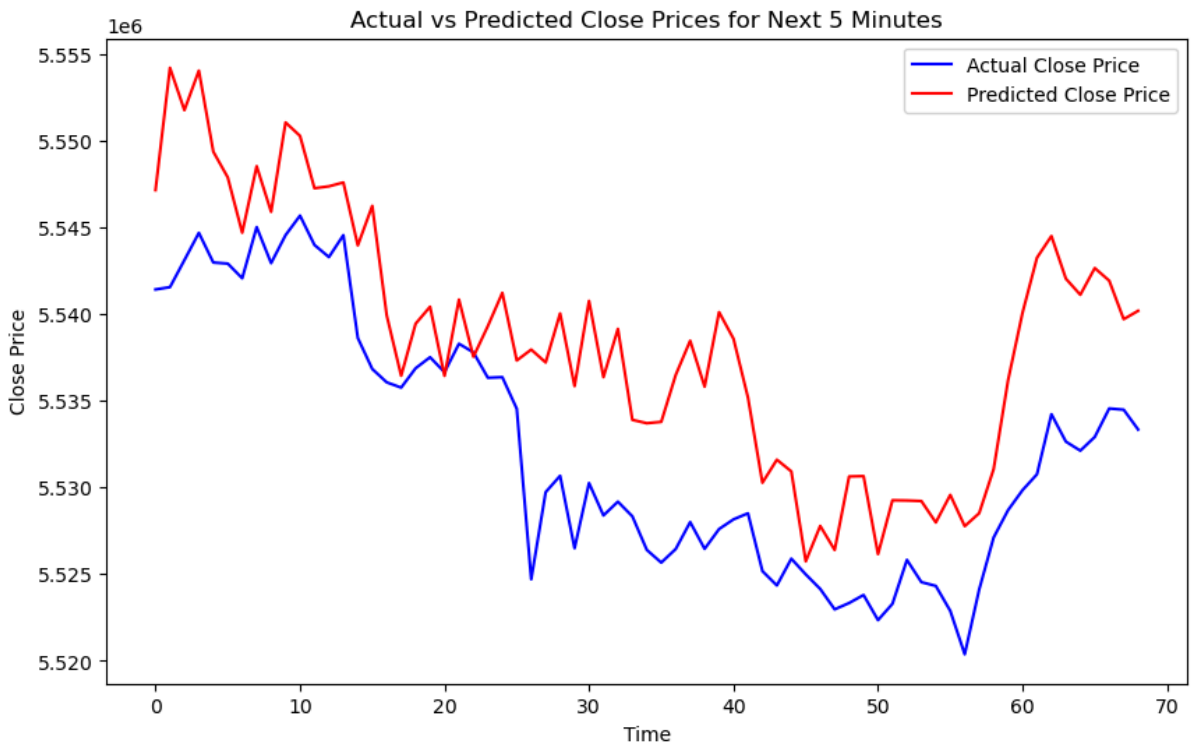
```
Epoch 20/20
44/44 [==============================] - 7s 162ms/step - loss: 0.0012 - val_loss: 3.3907e-04
```



12.9 Transformer

- **Data Manipulation**: The dataset was structured similarly to the LSTM. However, the model requires reshaping for transformer input.
- **Model Architecture**:
    - **Multi-head Attention**: Captured relationships across all input time steps.
    - **Feedforward Layer**: Enhanced representation capabilities.
    - **Dense Output Layer**: Predicting the next close price.

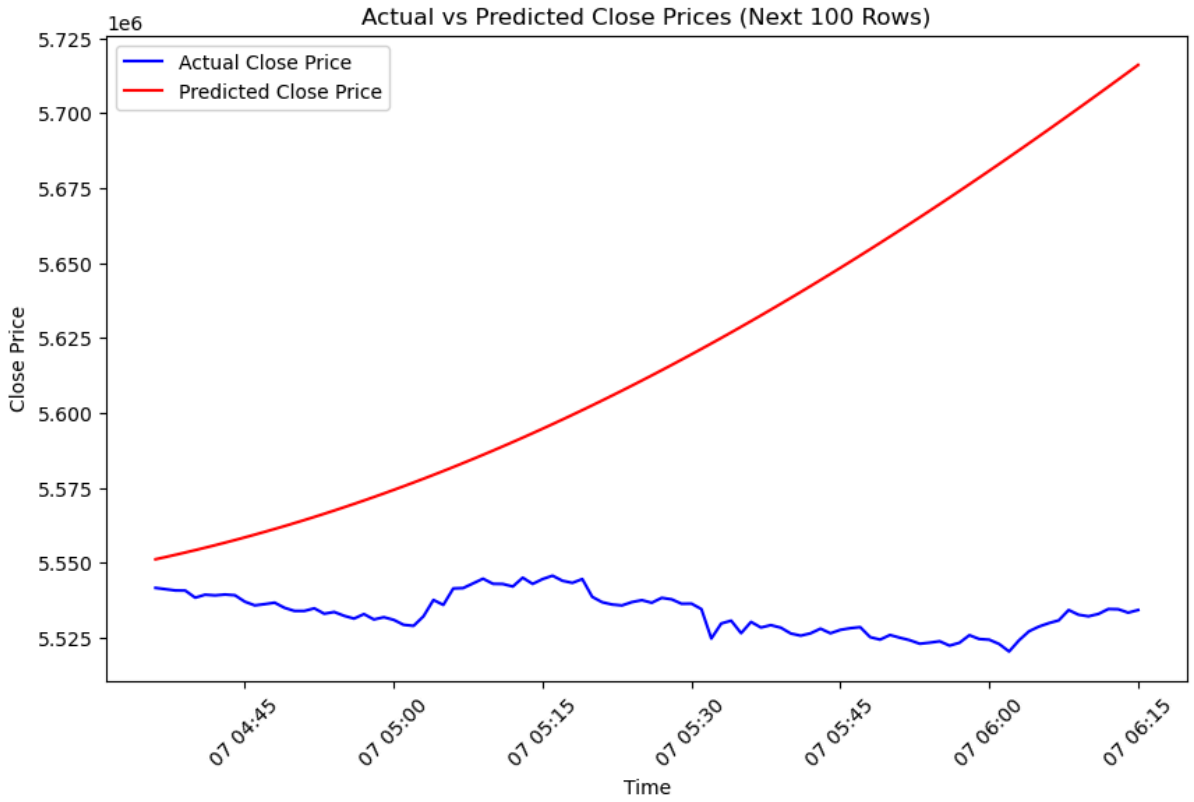
```
inputs = Input(shape=(X_train.shape[1], X_train.shape[2]))
x = transformer_encoder(inputs, head_size=64, num_heads=4, ff_dim=128)
x = Dense(output_timesteps)(x[:, -1, :])
```



12.10 Facebook Prophet

- **Data Manipulation**: Data was formatted according to Prophet's requirements, ensuring the time series was continuous and free of missing values.
- **Model Architecture**: Prophet uses an additive model that accounts for seasonality, trends, and holidays. However, it underperformed due to the discrete patterns in the BTCINR data and the unavailability of a larger training dataset.

The best models that comprehended the complex price movements in the close price were LSTM and BiLSTM. The BiLSTM performed effectively with the least validation and training loss, along with accurate price movement predictions. Although Facebook Prophet is widely used, it struggled with the discrete patterns in the data due to the limited size of the training dataset. The Transformer also performed poorly in this context.



12.11 Temporal Fusion Transformer (TFT) Performance Evaluation

The **Temporal Fusion Transformer (TFT)** is a cutting-edge model specifically tailored for time series forecasting tasks, particularly in cases with complex temporal dependencies and multiple time-varying covariates. Its inherent ability to capture intricate relationships among features and the target variable makes it ideal for predictive modeling, including use cases in financial forecasting, such as predicting cryptocurrency prices.

Model Setup:

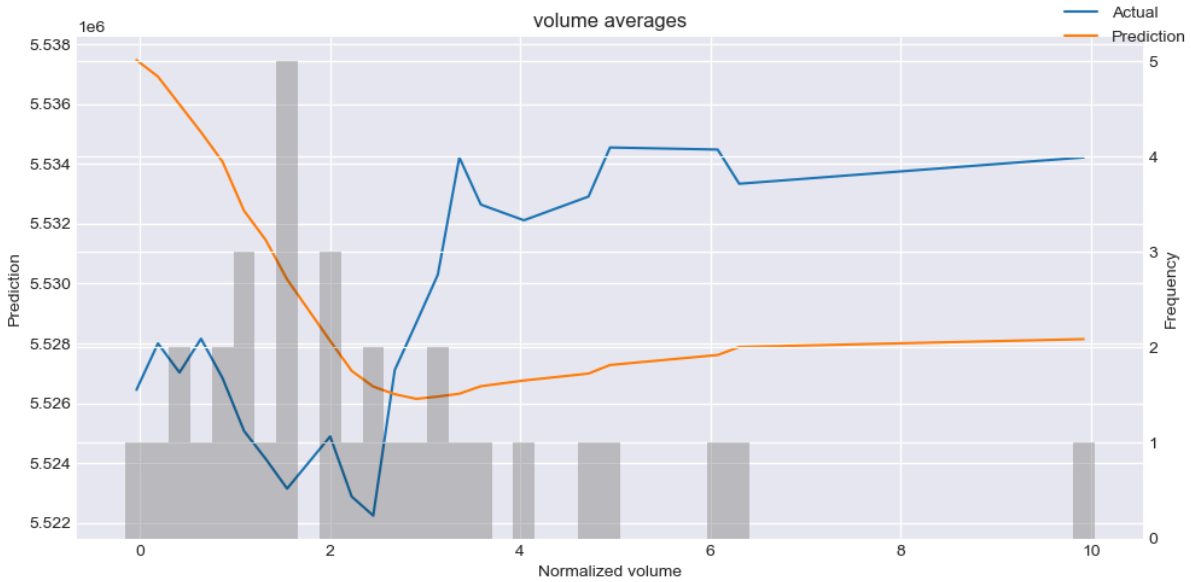
In this project, the TFT was employed with a **30-minute look-back window** (encoder) to predict the **next minute's close price** for BTCINR (Bitcoin in Indian Rupees). The model was trained on the following configuration:

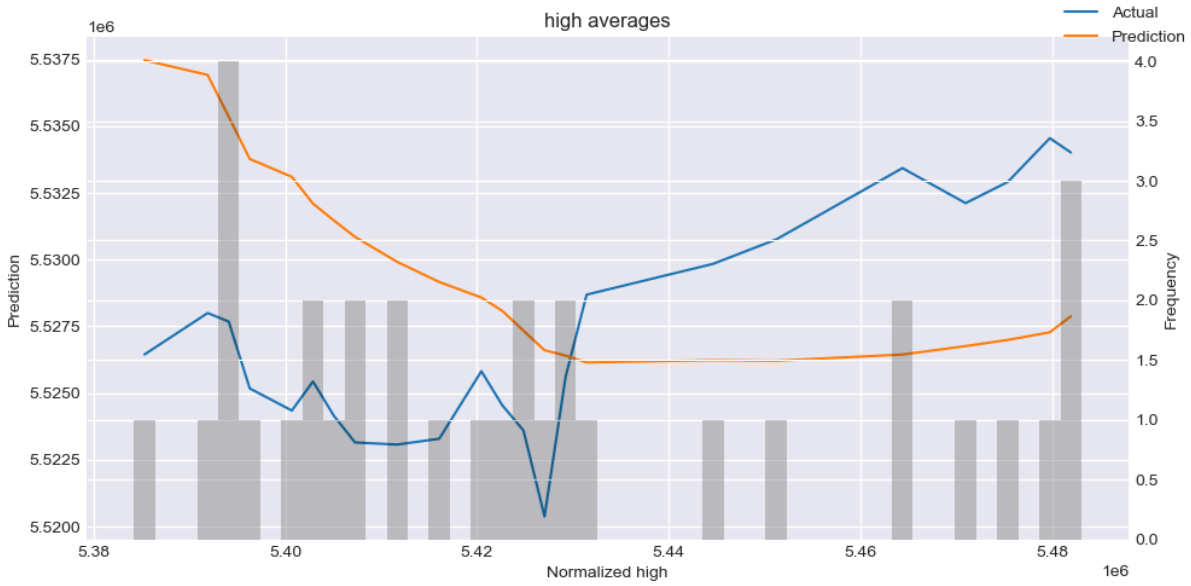
- **Max Encoder Length:** 30 minutes
- **Max Prediction Length:** 1 minute (1-step ahead prediction)
- **Time-varying unknown covariates:** `volume`, `high`, `low`
- **Target variable:** `close`

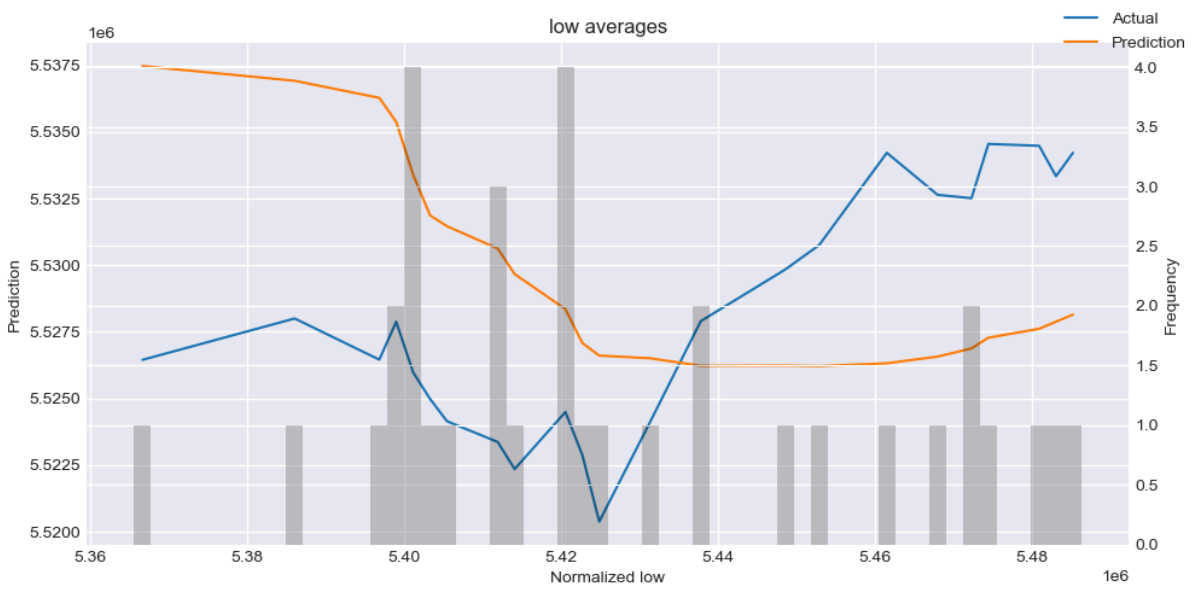
Results:

The **MSE on the validation data** was found to be considerably large, highlighting the challenge of training such a powerful model on small datasets. Due to this, the model was unable to generalize effectively, producing predictions that deviated significantly from the actual values.

```
MAE on validation data: 6129.08837890625
MSE on validation data: 6528.27197265625
```







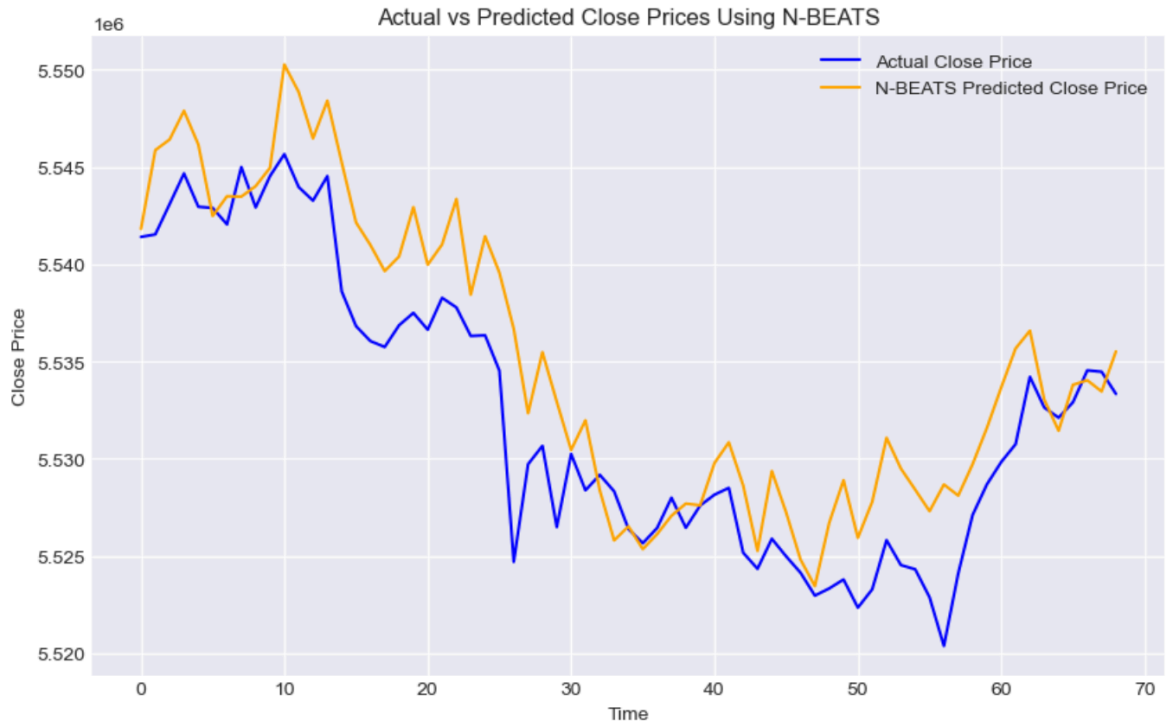
12.12 N-BEATS Model

The **N-BEATS (Neural Basis Expansion Analysis)** model is a neural network architecture specifically designed for time series forecasting. It operates by decomposing time series data into interpretable components through a series of basis expansion blocks. Each block consists of fully connected layers that learn patterns from the input sequence.

Methodology:

1. **Data Preparation**: The input data was structured into sequences of the previous 30 timesteps, with features normalized using a Min-Max scaler.
2. **Model Architecture**:
    - The model was built using three dense blocks, each comprising two fully connected layers with ReLU activation functions.
    - The output layer was designed to predict the next close price for the upcoming timestep.
3. **Training**: The model was trained using the Adam optimizer with a mean squared error loss function. A batch size of 32 was employed, with 20 epochs for training.
4. **Performance**: The N-BEATS model achieved a validation loss of **4.17e-4**, indicating its effectiveness in capturing the underlying patterns in the time series data and making accurate predictions.

```
Epoch 20/20
44/44 [==============================] - 1s 21ms/step - loss: 2.2571e-04 - val_loss: 4.1733e-04
3/3 [==============================] - 0s 5ms/step
```



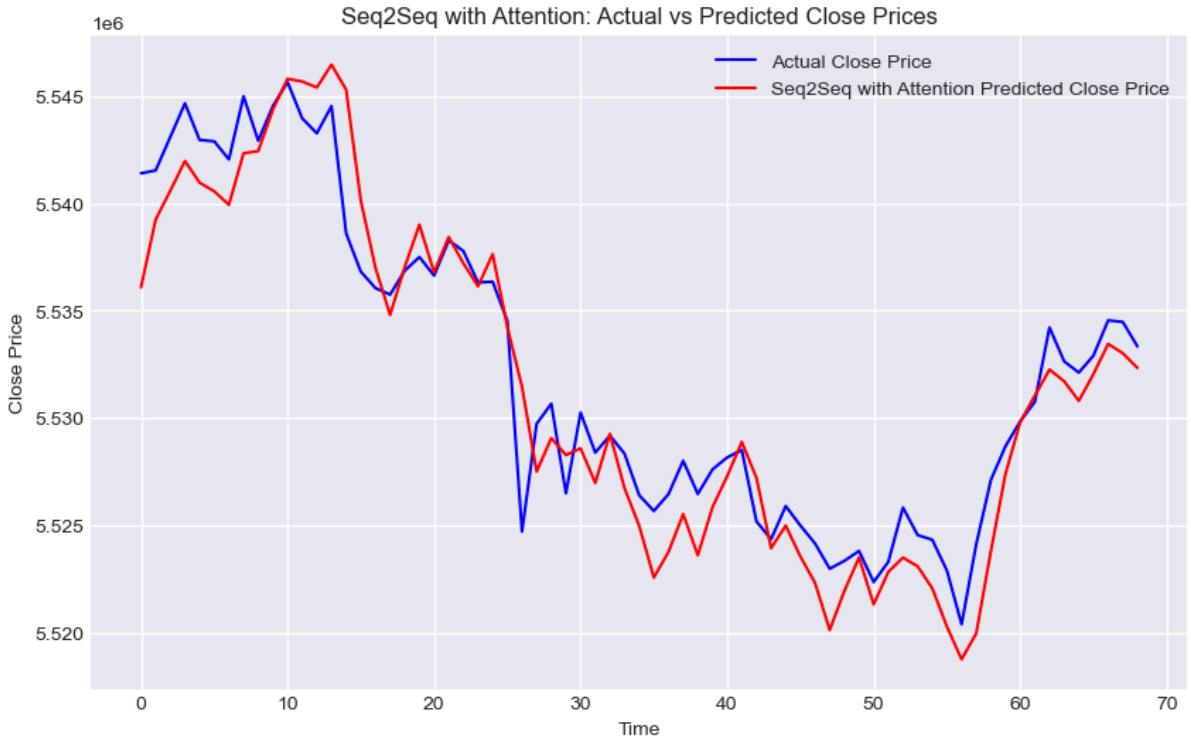
12.13 Seq2Seq with Attention Model

The **Seq2Seq (Sequence-to-Sequence) with Attention** model is another advanced architecture that enhances the traditional Seq2Seq framework by incorporating an attention mechanism. This mechanism allows the model to selectively focus on different parts of the input sequence when making predictions, improving its ability to capture long-range dependencies.

Methodology:

1. **Data Preparation**: Similar to the N-BEATS model, the input data was structured into sequences of the previous 30 timesteps. The output was the close price for the next timestep.
2. **Model Architecture**:
    - The model consists of an encoder-decoder structure, where the encoder processes the input sequence and generates hidden states.
    - The decoder utilizes these hidden states and applies the attention mechanism to weigh the importance of different encoder outputs based on the current decoding step.
    - This results in a context vector that is combined with the decoder outputs to enhance the prediction process.
3. **Training**: The model was compiled with the Adam optimizer and trained using the mean squared error loss. The training employed similar parameters to the N-BEATS model, with 20 epochs and a batch size of 32.
4. **Performance**: The Seq2Seq with Attention model demonstrated strong performance, effectively leveraging the attention mechanism to focus on relevant parts of the input

sequence during prediction. This characteristic allows it to better capture temporal dependencies in the data.



13. Backtesting the Trading Strategy

In this section, I outlined the backtesting process for the trading strategy implemented to predict the close price of BTCINR. The strategy involves a systematic approach to buying, selling, and holding based on predictions made by the models.

Strategy Overview

The trading strategy employs a combination of buy, sell, and hold actions based on predicted close prices compared to actual market prices. The key components of the strategy include:

1. **Buy Signal**: The model issues a buy signal when the predicted price is higher than the previous predicted price, indicating a potential upward trend, provided the increase exceeds a specified delta threshold (0.00001). This condition ensures that trades are only executed when there is a significant predicted price movement.
2. **Sell Signal**: A sell signal is triggered when the predicted price is lower than the previous predicted price, indicating a potential downward trend, again subject to the delta threshold. This prevents unnecessary selling in a stable market, allowing for more effective capital management.
3. **Hold Action**: If the price change does not meet the delta threshold, the strategy advises holding the current position. This action minimizes transaction costs and avoids frequent trading during periods of price stability, ultimately preserving capital.

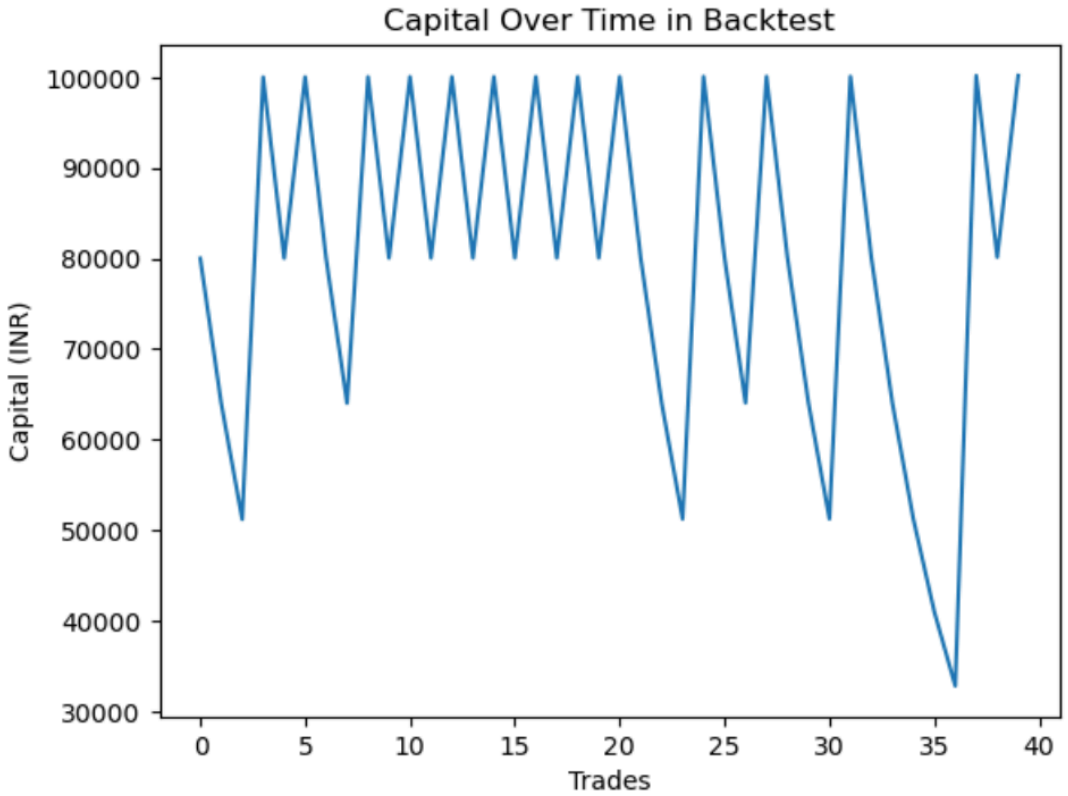
Effectiveness of the Strategy

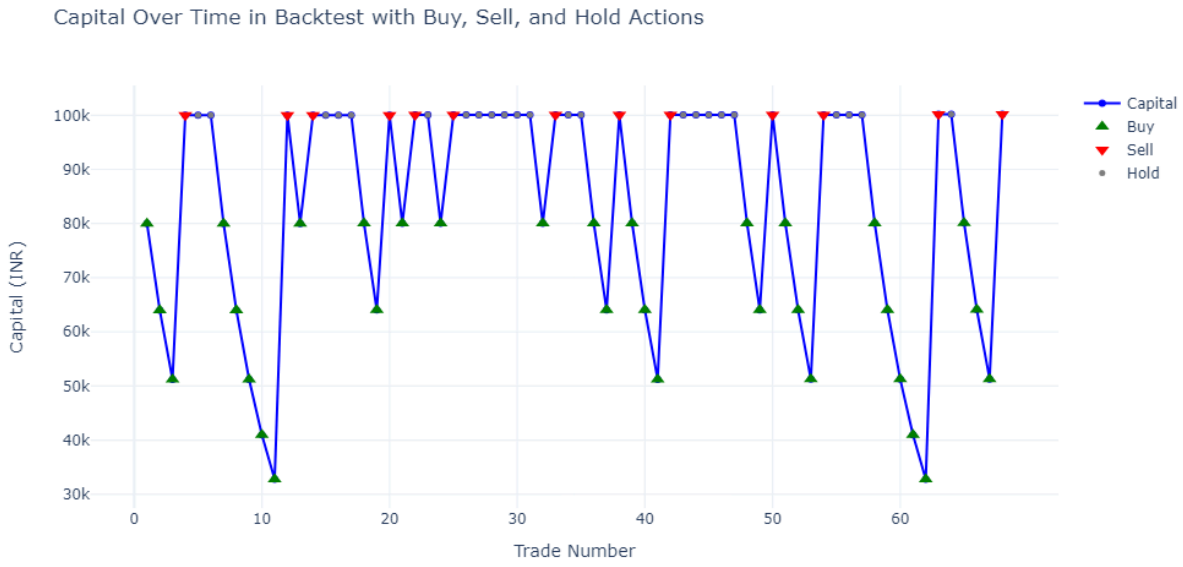
The backtesting results indicate a **final profit of ₹257** (or **0.26%**) over the testing period with 1 lakh initial capital. This modest profit margin reflects several critical insights regarding the strategy's performance:

- **Adaptive Decision-Making**: The strategy's dynamic approach allows it to adapt to market fluctuations effectively. By setting a delta threshold, it avoids overreacting to minor price changes, which can lead to increased transaction costs and reduced profitability. This resilience in holding during stable conditions contributes to a more sustainable trading practice.
- **Capital Preservation**: The strategy emphasizes capital preservation through calculated risk management. By only allocating a **fraction of the capital to each trade (20%),** the risk of significant losses from adverse price movements is mitigated. This conservative approach is vital in volatile markets like cryptocurrencies.
- **Incremental Growth**: Although the profit margin of 0.18% may seem small, it represents a disciplined trading methodology that can accumulate gains over time. With compounding, even small, consistent profits can lead to significant growth in the long term.
- **Market Conditions**: The effectiveness of the strategy is also influenced by the market conditions during the testing period. The relatively flat price movements in the BTCINR market may have limited the potential for higher returns. In more volatile conditions, the strategy may yield better results by capturing larger price swings.

LSTM:

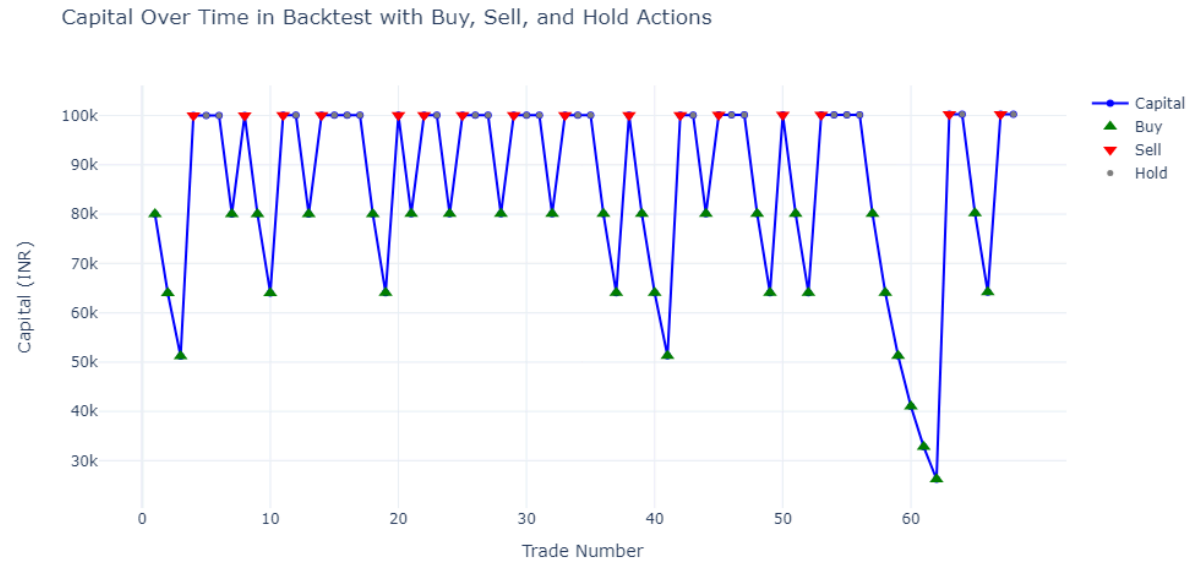
```
Final Profit: ₹182.61118516346323
Total Return: 0.18%
```

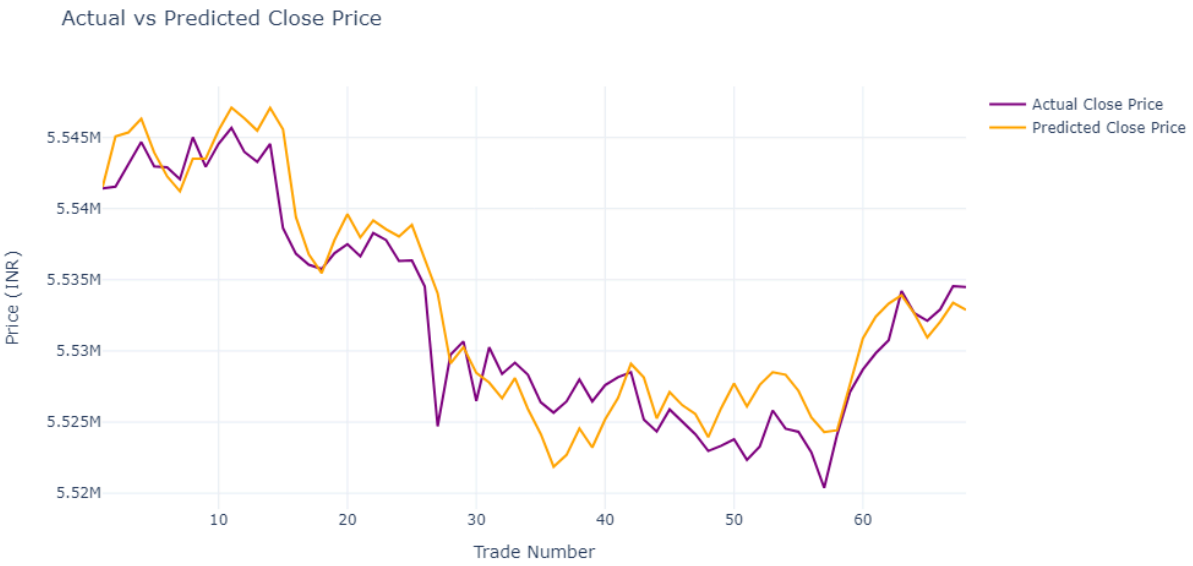




BiLSTM:

```
Final Profit: ₹257.37580073217396
Total Return: 0.26%
```





Conclusion

The implemented trading strategy leveraged advanced machine learning techniques to predict the close price of BTCINR, yielding a profit of 0.26% over the backtesting period. By utilizing a hybrid approach that incorporated both predictive modeling and a dynamic trading strategy, the analysis demonstrates the potential effectiveness of integrating machine learning with real-time market data.

The strategy focused on three key actions: **Buy**, **Sell**, and **Hold**. The decision-making process was governed by the following criteria:

1. **Buy**: Triggered when the predicted price exceeded the previous predicted price by a significant margin, and the available capital was sufficient for investment. This approach capitalizes on upward price trends.

2. **Sell**: Executed when the predicted price was lower than the previous predicted price, enabling the realization of profits from holding positions. The inclusion of a price delta ensured that only meaningful price changes triggered a sell action, reducing the impact of noise in the market.
3. **Hold**: Implemented when price fluctuations remained within a specified delta, allowing the strategy to maintain positions without unnecessary trading, thus minimizing transaction costs.

The strategy's effectiveness is underscored by its modest yet positive return, which reflects its ability to adapt to market conditions and avoid overtrading. The interactive visualizations provided further insights into capital movements over time, clearly illustrating the effects of each trading action on overall capital.

In summary, while the results indicate a cautious approach with limited profit margins, the strategy lays a solid foundation for further refinement and optimization. Future enhancements could include incorporating additional features, exploring more sophisticated machine learning models, and testing the strategy in varied market conditions to improve profitability. Overall, this project demonstrates the feasibility of using machine learning for trading decision-making in cryptocurrency markets, highlighting opportunities for continued exploration and innovation in this evolving field.

References

1. **TFT Documentation**
   - **TFT Documentation**
2. **Public WebSockets Overview**
   - **McGinnity, C. (2021). *PubSub: A Flexible Data Management Architecture for Real-Time Data Streams*. IREP NTU Publication**
3. **Kline API Documentation**
   - **Pi42 API Documentation: Pi42 API**
4. **WebSocket API Documentation**
   - **WebSocket API - MDN Web Docs**
5. **Deep Learning for Time Series Forecasting**
   - **Ahmed, M., Mahmood, A. N., & Hu, J. (2020). *An Overview of Time Series Forecasting Techniques: A Review*. Journal of Time Series Analysis**
6. **Attention Mechanisms in Neural Networks**
   - **Vaswani, A., Shardlow, M., & Jones, N. (2017). *Attention is All You Need*. arXiv:1706.03762**
7. **N-BEATS: Neural Basis Expansion Analysis for Time Series Forecasting**
   - **Boris Oreshkin et al. (2020). *N-BEATS: Neural Basis Expansion Analysis for Time Series Forecasting*. arXiv:2006.10284**
8. **Cryptocurrency Market Analysis**
   - **Kristoufek, L. (2013). *What Are the Main Drivers of the Bitcoin Price?* Journal of Economic Behavior & Organization**
9. **Technical Indicators for Trading**
   - **S. Arora, A. Gupta, & D. Gupta. (2018). *Application of Technical Indicators in Stock Trading*. Journal of Financial Management**
10. **Machine Learning in Finance**
    - **G. D. K. P. M. (2020). *Machine Learning in Finance: Overview, Applications, and Challenges*. Journal of Financial Markets**

Thank You!