**Timeseries Forecasting using LLMS &Generative AI models**

**Abstract:**

This project investigates the application of Large Language Models (LLMs) for time series forecasting, specifically in predicting Tesla's stock price. Leveraging a historical data set of Tesla stock prices, we design and implement a Time Series Forecasting Framework using various LLM models, including Time GPT, and compare their performance with traditional time series forecasting techniques. Our objective is to evaluate the effectiveness of LLM models in capturing complex patterns and trends in financial data, identify the most accurate model for stock price prediction, and analyze the impact of hyperparameters on LLM models in financial forecasting. Our results show that Gen AI and LLM models can outperform traditional techniques in forecasting accuracy, providing insights into their potential in revolutionizing financial forecasting and analysis. This study contributes to the growing field of AI in finance, with implications for investors, analysts, and researchers seeking to leverage advanced machine learning techniques for stock price prediction.

**Introduction:**

Time series forecasting is a crucial task in finance, allowing investors and analysts to make informed decisions about stock prices. This project explores the application of Large Language Models (LLM) and General Artificial Intelligence (Gen AI) methods, specifically Time GPT, to forecast Tesla's stock price. We evaluate the performance of various models on the Tesla stock price data set.

**Problem Statement:**

Traditional time series models often struggle with the complexities of financial data, leading to inaccurate forecasts and missed opportunities for investors and financial institutions.

**Description:**

This project explores the use of Large Language Models (LLM) and Generative AI (Gen AI), such as Time GPT, in forecasting Tesla's stock price. By comparing these advanced models with traditional techniques, the goal is to identify the most accurate and reliable approach for financial forecasting, highlighting the potential of LLM/Gen AI in transforming financial analysis.By focusing on business use cases, this project aims to demonstrate the practical applications and benefits of LLM/Gen AI in financial forecasting, ultimately driving better decision-making and performance in the financial industry.

Traditional models struggle with financial data  
LLMs are emerging to revolutionize time series forecasting, capturing complex patterns, handling noisy data, and adapting to changing markets, leading to:

-Enhanced decision-making

-Improved risk management

-New predictive analytics opportunities

-Transformation of financial forecasting

**Objective of the Project:**

* Explore and Compare various LLM Models for time series forecasting to implement them alongside traditional models, and compare their performance to identify strengths and weaknesses.
* Develop Accurate LLM-based Forecasting Model, leveraging their potential in capturing complex patterns and trends in financial data.
* Evaluate and Select Best-fit Model by assessing the performance of different time series forecasting models, including LLMs and traditional techniques, and select the most suitable model for accurate and reliable predictions.

**Tools used:**

**Python** was chosen as the primary tool for this analysis due to its versatility, flexibility, and extensive libraries make it an ideal choice for data analysis, machine learning, and time series forecasting.

**Key Challenges:**

* **Computational Resource Challenges**: Addressing the constraints of limited memory, processing power, and runtime in free Google Colab and local desktop environments, to ensure successful model implementation and training.
* **Processor Constraint Limitations:** Difficulty in setting and tuning hyperparameters due to processor constraints, leading to reliance on default parameter settings.
* Implementing transformer models and LLM-based models in local IDEs due to space constraints and runtime errors.
* Handling noisy financial data
* Capturing **complex patterns** and non-linear relationships

**Expected Outcomes:**

* A comprehensive evaluation of Time GPT and other LLM/Gen AI models in forecasting Tesla's stock price
* A Time Series Forecasting framework using LLM/Gen AI models
* A comparison of the performance of LLM/Gen AI models with traditional time series forecasting techniques
* Insights into the potential of LLM/Gen AI models in revolutionizing financial forecasting and analysis.
* Evaluate the Performance of Time GPT and other LLM/Gen AI models in forecasting Tesla's stock price, using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2).

**Data Source:**

The project utilizes a data set obtained from [Yahoo Finance](https://finance.yahoo.com/quote/TSLA/history/" \t "_new). The data set includes daily historical stock prices for Tesla (TSLA) and consists of the following features:

Date: The trading date.

Open: The opening price of the stock on the given date.

High: The highest price of the stock on the given date.

Low: The lowest price of the stock on the given date.

Close: The closing price of the stock on the given date.

Volume: The number of shares traded on the given date.

Adj Close: The closing price adjusted for dividends and stock splits.

The data set is well-structured and provides sufficient data points for training and testing the predictive models.

**Data Description:**

The data set used in this project comprises Tesla's historical stock prices, including daily opening, closing, high, and low prices, along with volume traded. The data set consists of historical daily stock prices of Tesla from June 29, 2010, to June 6, 2024.

The data set contains 3509 rows.

**Data Collection and Preprocessing:**

Data is taken from the previously mentioned link and Relevant features were extracted from the data. Rigorous data pre-processing techniques, such as feature engineering, and scaling, were applied to ensure data quality and integrity.

1. **Exploratory Data Analysis (EDA):**

* **Data Pre-processing :**The data is sorted by date, and the 'Close' price is selected for analysis.
* **Handling Non-Continuous Time Stamps:**

No Missing Values: The data set did not contain any missing values.

Non-Trading Days: However, stock market data is not continuous due to weekends and public holidays, resulting in gaps in the time stamps.

Model Limitations: Large Language Models (LLMs) and foundational models are unable to inherently understand these gaps.

Solution: To address this, business days were used as the index, and the Forward Fill (Ffill) method was employed to fill gaps, assuming the previous closing price would be the opening price on the next trading day.

* **Feature Selection:**

For this analysis, two primary features have been selected:

**Time (Index):** Time has been used as the index to capture the temporal dependencies and patterns in the data, which is essential for time series forecasting. By setting time as the index, the closing price data can be effectively analyzed in relation to its temporal context.

**Closing Price:** The closing price has been identified as the most critical feature for forecasting stock prices, as it represents the final price of the stock at the end of each trading day. The closing price over time has been analyzed to identify trends, patterns, and seasonality, which are crucial for predicting future price movements.

By focusing on these two features, the following benefits can be achieved:

The temporal dynamics of the stock price can be captured

Patterns and trends in the closing price can be identified

A robust forecasting model can be developed that leverages the relationships between time and closing price

This feature selection allows for the creation of a concise and effective model that targets the most critical aspects of stock price forecasting.

* **Scaling of Closing Price:**

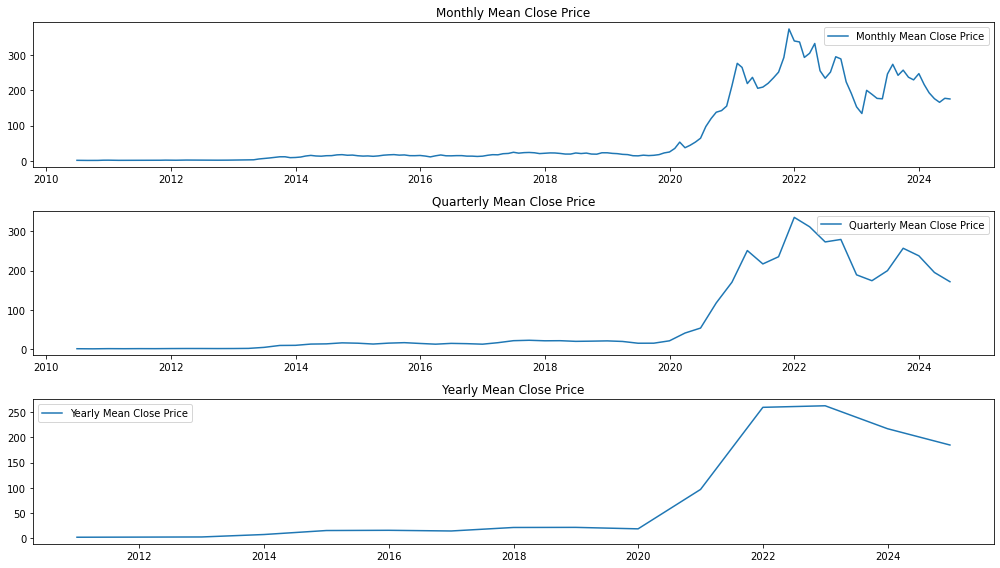
The closing price data was normalized as a preprocessing step. - using standard scalar

* **Data Visualization:** Visualizations and statistical methods are used to understand the data.

1. **Time Series Plot:** To visualize the trend and seasonality in Tesla's stock prices.

Moving Averages: To smooth out the data and identify underlying trends.

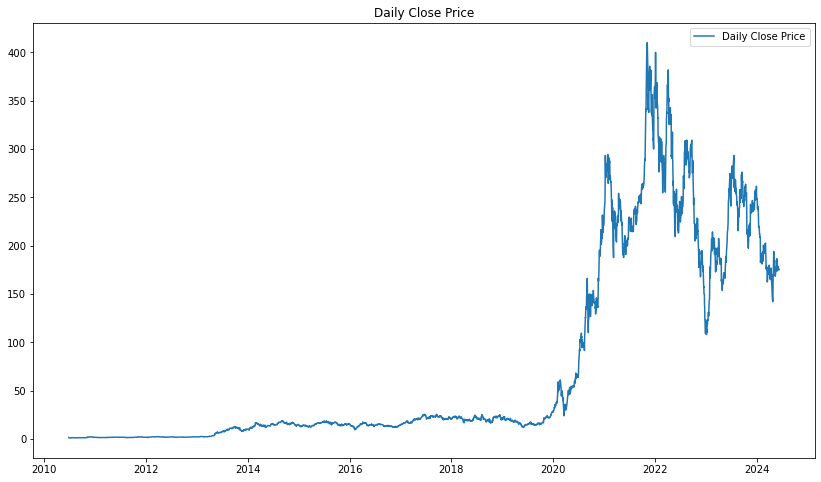
**Monthly, Quarterly, and Yearly Mean Close Prices:** This plot shows the resampled data at different frequencies, providing insights into the trends over different time periods.



*#Monthly Mean Close Price:T*here are noticeable fluctuations, indicating periods of rapid growth and occasional declines.

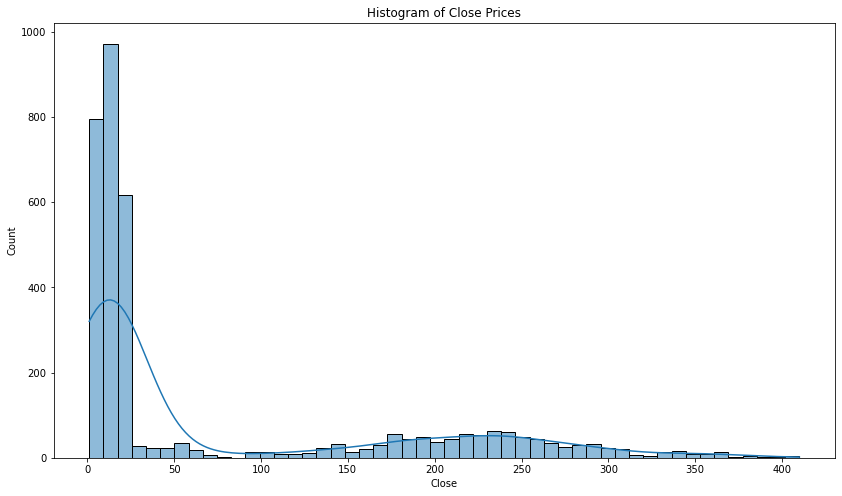
*#Quarterly Mean Close Price:* The quarterly resampled data smooths out some of the short-term volatility seen in the monthly data. This plot is useful for identifying medium-term trends. The general upward trend in Tesla's stock price is more evident in this plot.

*#Yearly Mean Close Price:* The yearly resampled data provides a long-term view of Tesla's stock performance. This plot shows a clear upward trend, indicating significant growth over the years. The yearly averages help in understanding the overall performance and long-term trends.

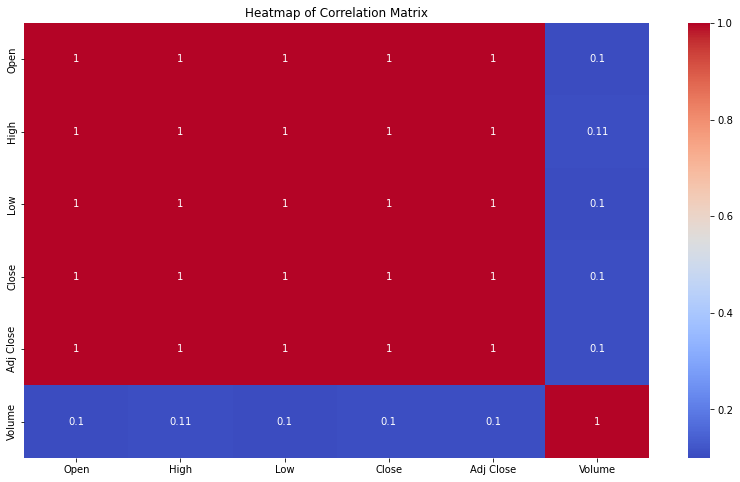


+

Daily Close Price Line Plot: This plot shows the daily closing prices of Tesla stock over the entire data set.#The overall trend is upward, but there are several periods of significant volatility, especially in recent years.



1. **Histogram of Close Prices:** This histogram shows the distribution of the closing prices, with a kernel density estimate (KDE) overlay #The histogram indicates that the majority of closing prices are concentrated in the lower range, with a long tail extending to higher prices. This suggests that while Tesla's stock has experienced significant growth, it has spent a considerable amount of time at lower price levels.
2. Heatmap of Correlation Matrix: This heatmap shows the correlation between different columns in the data set .



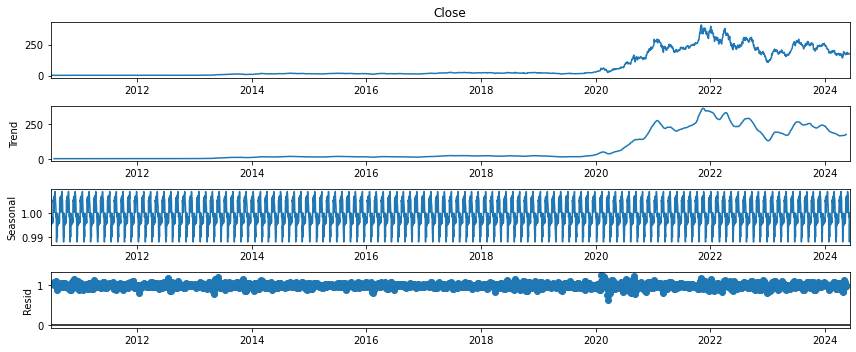
High correlation between Open, High, Low, and Close prices: This is expected as these prices are closely related to each other within a trading day.

The trading volume does not have a strong linear relationship with the stock prices. This suggests that other factors, such as market sentiment and external events, may play a significant role in determining the trading volume.

**Time Series Analysis:**

ACF, PACF, and seasonal decomposition are applied to identify patterns.

1. **STL Decomposition of the Data:**



Trend: The trend component shows a clear upward movement in Tesla's stock prices over the years, indicating significant growth.

The plot reveals an overall upward trend with significant volatility, particularly during specific periods like 2020 and 2021, which coincide with major company announcements and market events.

Seasonal: The seasonal component shows repeating patterns, suggesting the presence of seasonality in the data. The periodic fluctuations indicate that there are certain times of the year when Tesla's stock prices tend to be higher or lower.

Residual: The residual component appears to be random noise, with no clear patterns. This indicates that most of the systematic variation in the data has been captured by the trend and seasonal components.

1. **ACF plot with different lags**

**Determining Optimal Lag Size for Time Series Analysis**

To effectively capture the underlying patterns in the Tesla data set, spanning approximately 14 years of daily data (June 2010 to June 2024), it's essential to determine the ideal lag size.

**Seasonality Considerations**

Annual Seasonality: Expecting seasonal patterns, such as yearly fluctuations, necessitates examining longer lags to accurately capture these trends.

Lag Size Options:

Short-term dependencies: Initial analysis with 50 lags

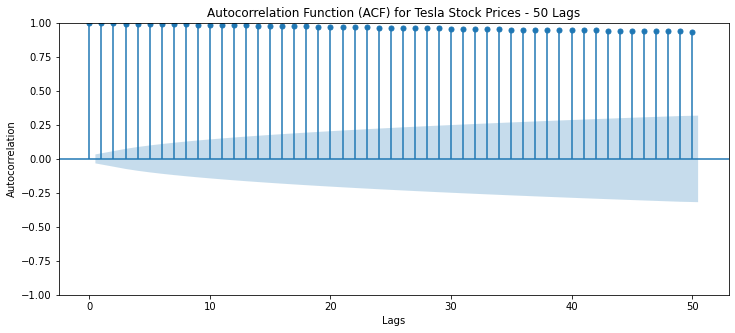
Longer-term dependencies: Expanded analysis with 100 lags

Annual seasonality: Comprehensive analysis with 365 lags to fully capture yearly patterns.

Optimal Lag Size Identification

By considering these factors and iteratively exploring different lag sizes, the optimal lag size can be identified to uncover meaningful insights from the time series data.

50 lags:

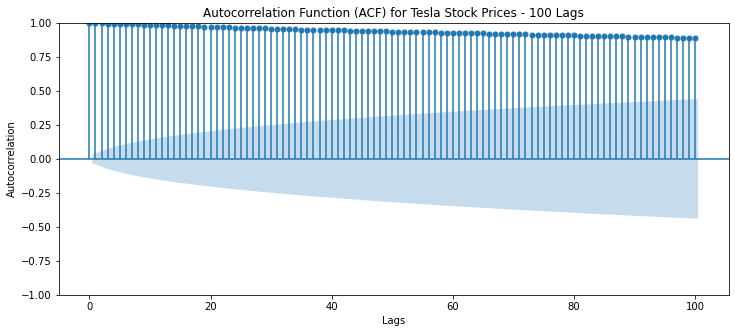


Findings for ACF with 50 lags:

There is a very strong positive autocorrelation for small lags, and the autocorrelation slowly decreases as the lag increases. This indicates that the Tesla stock price is highly persistent, and that past prices are a good predictor of future prices.

Also, the slowly decaying ACF suggests that the Tesla stock price may not be stationary, and that there may be a trend in the data.

100 lags:



Findings for ACF with 100 lags:

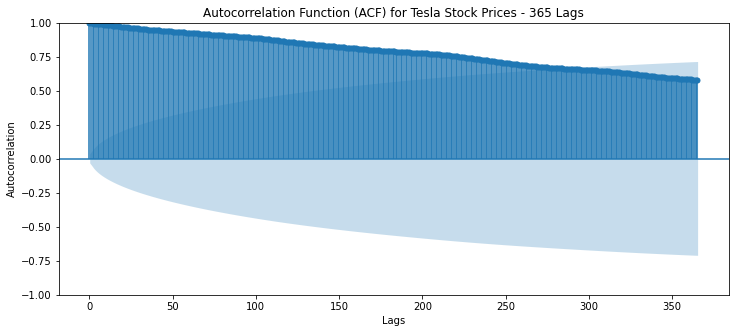
- Gradual Decay: The ACF gradually decays, hinting at potential non-stationarity in the time series, suggesting that the data may not be constant over time.

- Seasonal Patterns: Significant autocorrelations at larger lags (approximately multiples of 30) indicate possible monthly or quarterly seasonality, implying recurring patterns within these time frames.

- Longer-term Insights: This 100-lag plot offers a more comprehensive view of dependencies compared to the 50-lag plot, uncovering potential longer-term patterns and relationships in the data.

These findings suggest that the time series may exhibit non-stationarity and seasonality, emphasizing the need for further analysis and potential differencing or seasonal adjustment to achieve stationarity.

Annual lags:



Findings: With 365 lags, the potential annual seasonality becomes more apparent

6. **PACF (Partial Autocorrelation Function) Analysis**

The PACF chart is a valuable tool in time series analysis, enabling the identification of the direct relationship between an observation and its lagged values, while accounting for the effects of intermediate lags.

Key Benefits of PACF:

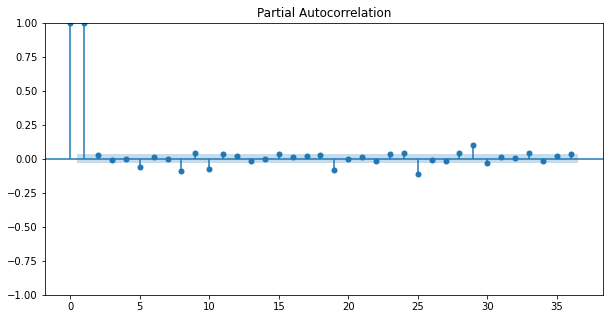
Identify Direct Relationships: PACF helps identify the direct correlation between an observation and its lagged values, uncoupling the effects of intermediate lags.

*Determine AR Model Order:* The number of significant lags in the PACF plot indicates the appropriate number of Autoregressive (AR) terms to include in your model, ensuring optimal model specification.

*Complementary Insights:* PACF provides additional insights beyond what the ACF plot alone reveals, offering a more comprehensive understanding of the time series dynamics.

*AR Model Suitability:* PACF helps determine if an AR model is suitable for your time series data and, if so, what order of the model is appropriate.

By applying PACF analysis, you can gain a deeper understanding of your time series data, make informed decisions about AR model specification, and ultimately improve your forecasting accuracy.



Findings: The number of significant spikes before the cut-off can give you an indication of the potential order for your AR model. The above plot shows that the potential order can be 1 or 2.

1. **Rolling mean and SD**

**Rolling Mean:**

-Shows the overall trend and direction of the data

-Reduces noise and fluctuations, making it easier to see the pattern

-Can help adjust for seasonality

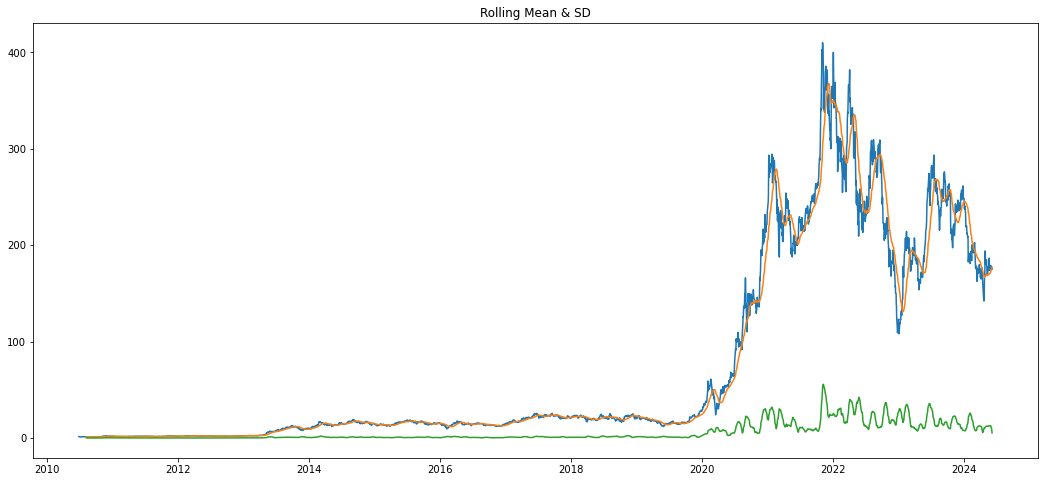
-Rolling Standard Deviation (SD):

-Measures the risk and uncertainty of forecasting

-Helps identify unusual patterns or outliers

-Informing forecasting models with this information can improve accuracy

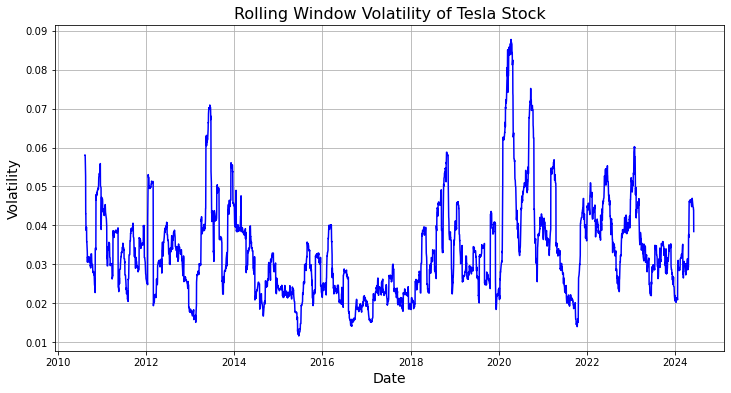
In simpler terms, Rolling Mean helps us see the big picture and trends, while Rolling Standard Deviation helps us understand the risk and potential surprises in the data.



**Key Findings:**

- Upward Trend: Tesla's closing price has a general upward trend over time, as shown by the rolling mean (blue line).

- Volatility Changes: The rolling standard deviation (orange line) shows that volatility increases during certain periods, indicating potential market uncertainty.

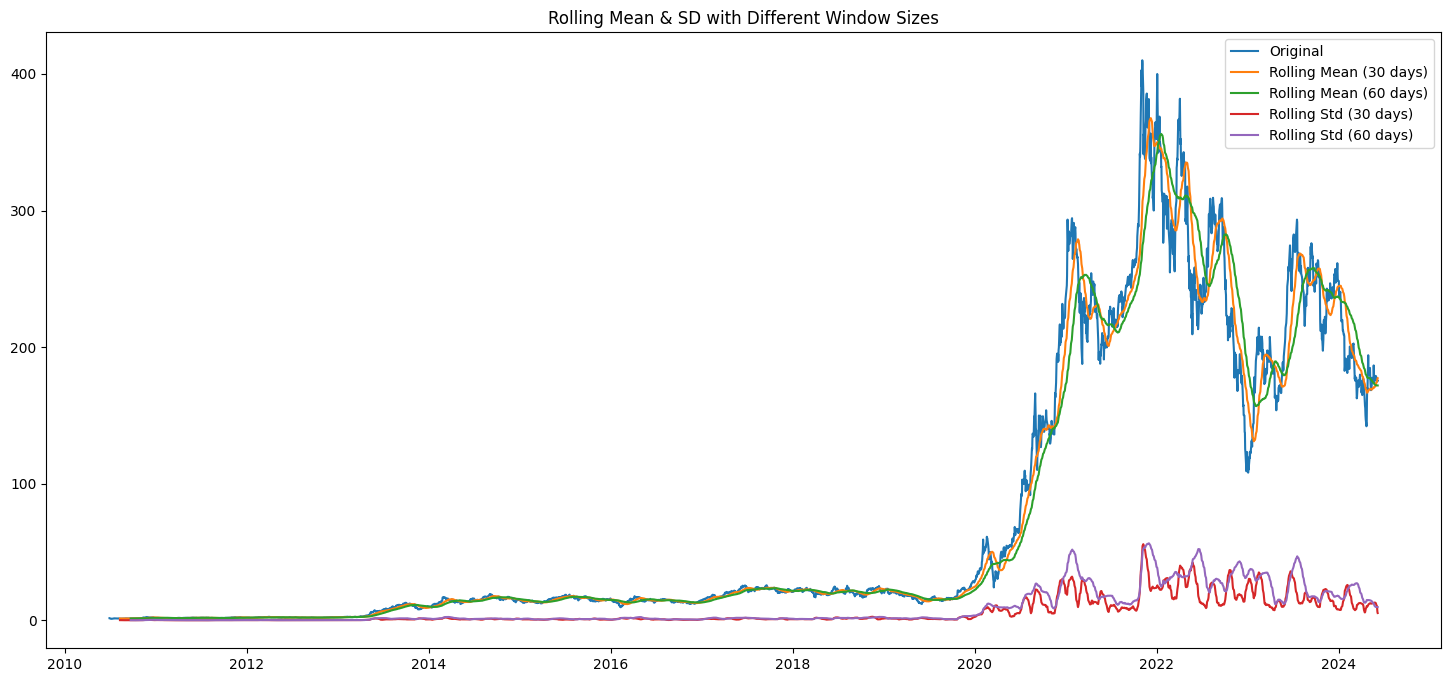


- Price Movement & Volatility: When the price changes rapidly (up or down), volatility tends to increase, suggesting a connection between the two.

In simpler terms, Tesla's stock price has been going up over time, but with periods of increased uncertainty and volatility, especially when the price changes quickly.

-The actual stock prices show significant volatility, especially from 2020

1. **Experiment with Window Size:**



**Key Findings:**

Smoother Trends: Using a 60-day window smooths out the curves, giving a clearer view of the overall trend and volatility.

Less Volatility: The 60-day window shows less volatility compared to the 30-day window, indicating a more stable measure.

Delayed Response: Larger windows make the rolling mean less responsive to short-term changes, as they consider more past data.

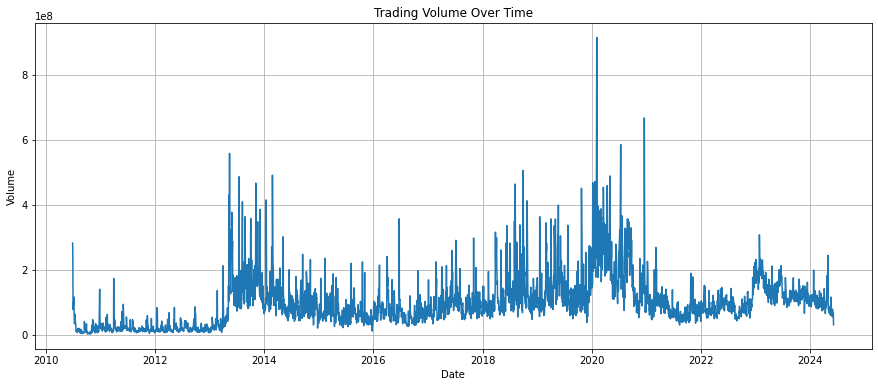
Choosing the Right Window:

Small window (30-day): Best for short-term analysis and trends

Large window (60-day): Best for long-term analysis and a smoother view

Using a larger window size gives a more stable and smoother view of the trend and volatility, but may not capture short-term changes quickly.

1. **Trading Volume:**



* 2020-2021 is the peak of Trading.
* The highest trading volume: 2020-02-04 ---> 914082000.0
* The highest opening price:2021-11-04 ---> 411.470001
* The highest closing price --->2021-11-04 409.970001

1. **Stationarity Check**

Stationarity check helps to confirm if the data has a consistent pattern over time, without any significant changes in its underlying structure, making it easier to model and forecast.

**Stationarity Test Results**

**Level Data:**

The Augmented Dickey-Fuller (ADF) test was conducted on the level data to assess its stationarity. The results are as follows:

ADF Statistic: -1.3107

p-value: 0.6243

Critical Values:

1%: -3.4322

5%: -2.8624

10%: -2.5672

Since the p-value (0.6243) is greater than the significance level of 0.05, we fail to reject the null hypothesis, indicating that the time series is not stationary at level.

First Differenced Data:

To achieve stationarity, the time series was differenced once, and the ADF test was re-conducted. The results are as follows:

ADF Statistic: -11.1558

p-value: 2.867786251443141e-20 (approximately 0)

Critical Values:

1%: -3.4322310380892618

5%: -2.862371135477015

10%: -2.5672124281161035

With a p-value of approximately 0, we reject the null hypothesis, confirming that the time series is stationary after first differencing.

In conclusion, the time series data exhibits non-stationarity at level, but becomes stationary after applying **first differencing**. This preprocessing step is crucial for subsequent time series analysis and forecasting.

**Overall Findings of EDA:**

1. Time Series Plot:  
\* Stock prices show significant volatility, especially from 2020 onwards.  
\* Overall upward trend in stock prices.

1. Histogram Plot:  
   \* Stock prices follow a positively skewed distribution.  
   \* Most prices are concentrated between $300 and $400.
2. Volatility:  
   \* Stock prices show significant volatility, especially from 2020 onwards.  
   \* The 60-day window smooths out the curves, giving a clearer view of the overall trend and volatility.
3. Trading Volume:  
   \* 2020-2021 is the peak trading period.  
   \* The highest trading volume was on 2020-02-04, with 914,082,000 shares traded.  
   \* The highest opening and closing prices were on 2021-11-04, at $411.47 and $409.97, respectively.
4. Stationarity:  
   \* The Augmented Dickey-Fuller (ADF) test indicates that the time series is not stationary at level (p-value = 0.6243).  
   \* First differencing makes the time series stationary (p-value ≈ 0).
5. Correlation:  
   \* No significant correlation between trading volume and stock prices.

7 . Trends:  
\* The 60-day window shows a smoother trend, indicating a more stable measure.  
\* The rolling mean with a 60-day window gives a clearer view of the overall trend.

These findings provide valuable insights into the stock price data, including its volatility, trading volume, stationarity, correlation, and trends. They can inform the development and evaluation of predictive models.

**Model Building and Evaluation:**

Several models were built and evaluated for their performance in predicting stock prices

**Traditional Models:**

1. ARIMA

**ML Models:**

1. Linear Regression
2. XG Boost
3. Random forest

**Deep learning Model:**

1. LSTM

**LLM and Generative AI Models:**

1. TimesFM
2. Chronos
3. Time GPT
4. NBeats Model
5. GPT 2

**Evaluation metrics**:

Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R-squared (R2) are used to compare model performance.

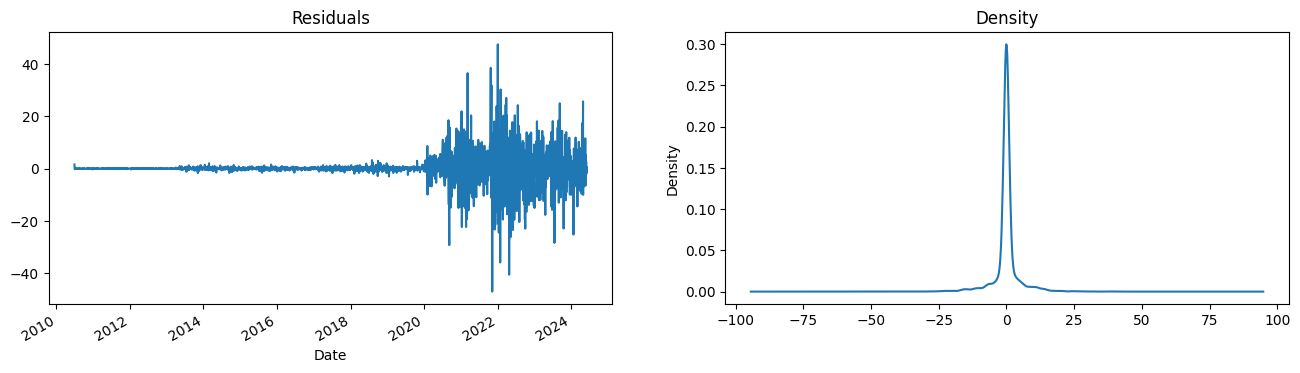
**Models:**

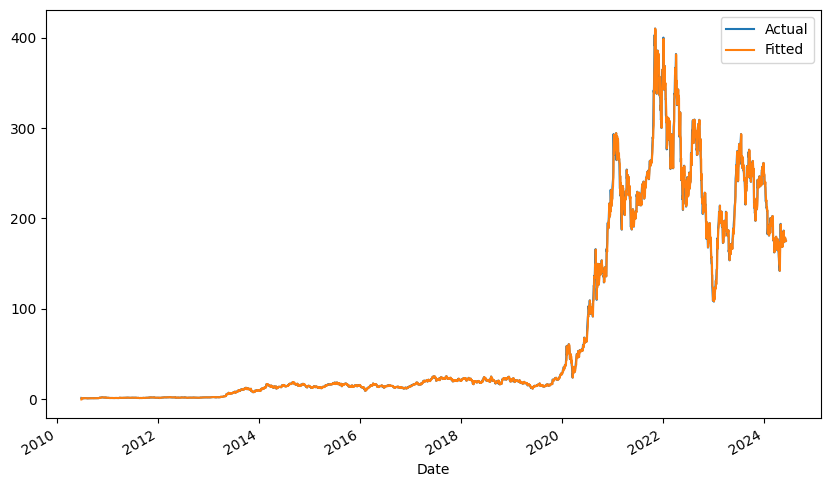
1. **ARIMA**

Model Selection:

Grid search was performed to determine the best parameters for the ARIMA model.

The selected model was ARIMA(1, 1, 0) based on the lowest Akaike Information Criterion (AIC).





**Model Evaluation:**

MSE: 22.20394718100934

RMSE: 4.712106448395382

MAE: 2.0158184429299877

MAPE: 0.02499498839622135

R-squared: 0.9978616950599498

**-ARIMA Model Fitting Results**

Best Model: ARIMA(1, 1, 0) with AIC value of 20835.906654659448

Model Summary:

Coefficients: AR term (ar.L1) and variance (sigma2)

Residual Analysis:

No significant autocorrelation (Ljung-Box test)

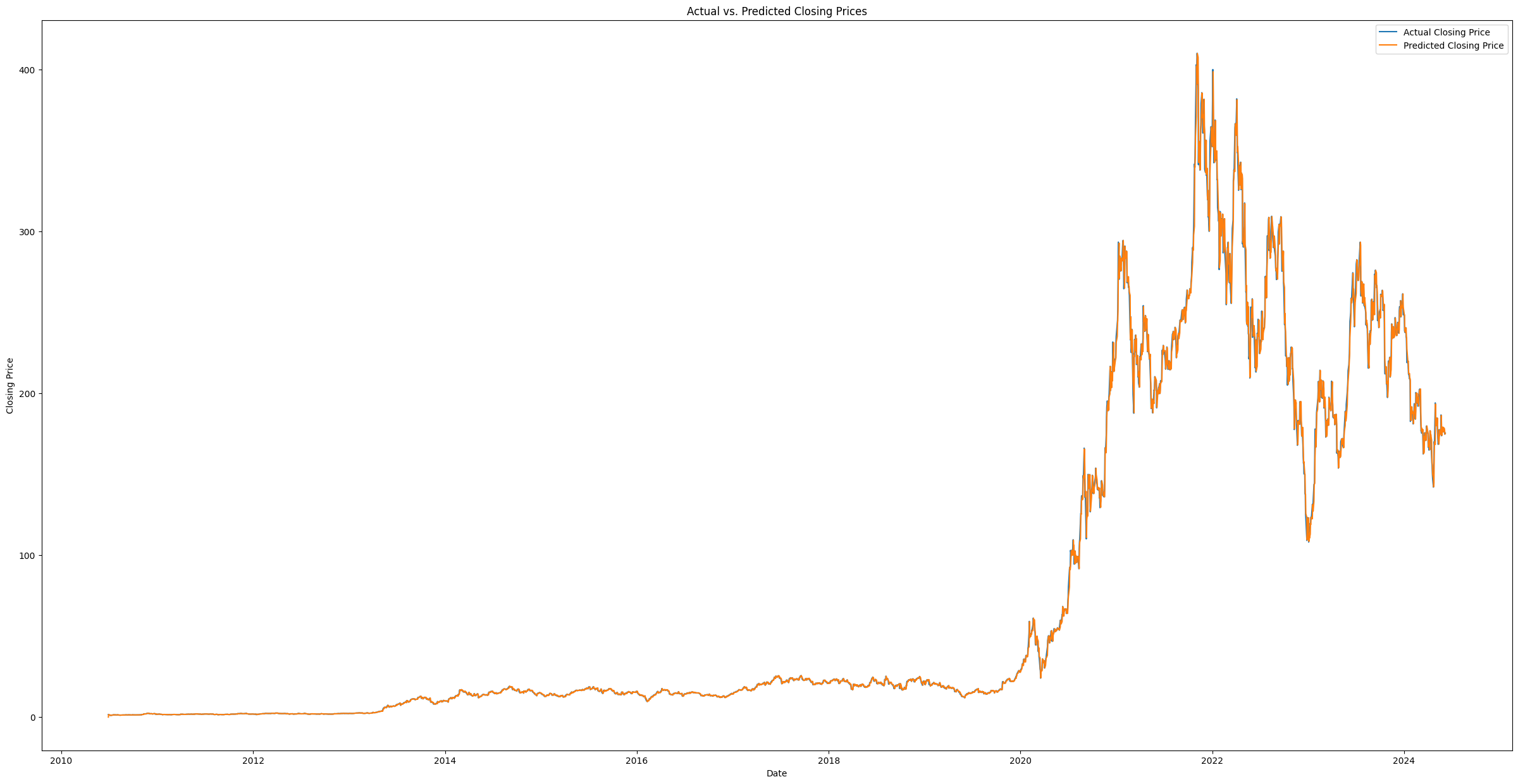
Non-normal distribution of residuals (Jarque-Bera test)

Heteroskedasticity in residuals (Heteroskedasticity test)

**-Model Performance:**

Actual vs Fitted Values: Plots show a good fit

Residual Plots: Density and plots indicate some patterns, but overall acceptable



**Training Set:**

Excellent fit with low errors and high R-squared value

**Test Set:**

Higher errors and lower R-squared value indicate poor model performance

MSE: 268.57 (higher than full data)

RMSE: 16.39 (higher than full data)

MAE: 15.52 (average absolute prediction error)

MAPE: 0.087 (average percentage error, higher than full data)

R-squared: -8.71 (negative value indicates poor performance)

**Key Findings:**

ARIMA model is overfitting the training data

Model struggles to generalize to new, unseen data points

Residuals show no significant autocorrelation, but are not normally distributed and exhibit heteroskedasticity

**Various approach on ARIMA:**

Tried the simplified model But model dint perform well. Hence tried creting sythetic data using **GAN method :**

MSE: 3773.7414586185228

RMSE: 61.43078591893907

MAE: 49.32236858887279

MAPE: 0.21969237150907636

1. squared: -0.0071511784006126344

Original Data: Mean: 74.0632059373282

Variance: 10394.237301897214

Standard Deviation: 101.95213240485563

Correlation: nan

Skewness: 1.2915225033796172

Kurtosis: 0.19821289024117839

**Synthetic Data:**

Mean: tensor(0.0305, grad\_fn=<MeanBackward0>)

Variance: tensor(0.9513, grad\_fn=<VarBackward0>)

Standard Deviation: tensor(0.9754, grad\_fn=<StdBackward0>)

Correlation: nan Skewness: -0.13790271038655963

Kurtosis: -0.31208529436498766

The synthetic data has a much lower mean and variance compared to the original data, indicating that the generated data may not be representative of the original data. The skewness and kurtosis values also differ between the two datasets.

Hence Hyper parameter tuning performed on this GAN model using grid search.

Best Hyperparameters:

Learning Rate (lr): 0.01

Batch Size: 32

Number of Epochs: 50

Optimizer: Adam

Best MSE: 52060.8070176427

This means that the GAN model with the above hyper-parameters achieved the lowest Mean Squared Error (MSE) on the test data.

Some observations:

The learning rate of 0.01 seems reasonable for this problem.

A batch size of 32 is a good trade-off between computational efficiency and model performance.

50 epochs might be a relatively small number of iterations, but it's a good starting point.

Adam optimizer is a popular choice for GANs, and it seems to be working well here.

Finally Implemented ARIMA on sythetic data and the results are:

MSE (Mean Squared Error): 37211.25

RMSE (Root Mean Squared Error): 192.90

MAE (Mean Absolute Error): 182.93

MAPE (Mean Absolute Percentage Error): 0.74 (or 74%)

*The model shows the poor performance of test data in all the possibilities.*

1. **Linear Regression:**

Linear Regression is a widely used statistical model for predicting continuous outcomes. In the context of time series forecasting, Linear Regression assumes a linear relationship between the past values of the time series and its future values.

Advantages: Interpretability, Computational Efficiency, Flexibility

Limitations: Stationarity, Overfitting, Sensitivity to Outliers, Limited Ability to Capture Complex Patterns

Linear Regression Model Performance

MSE: 11.77

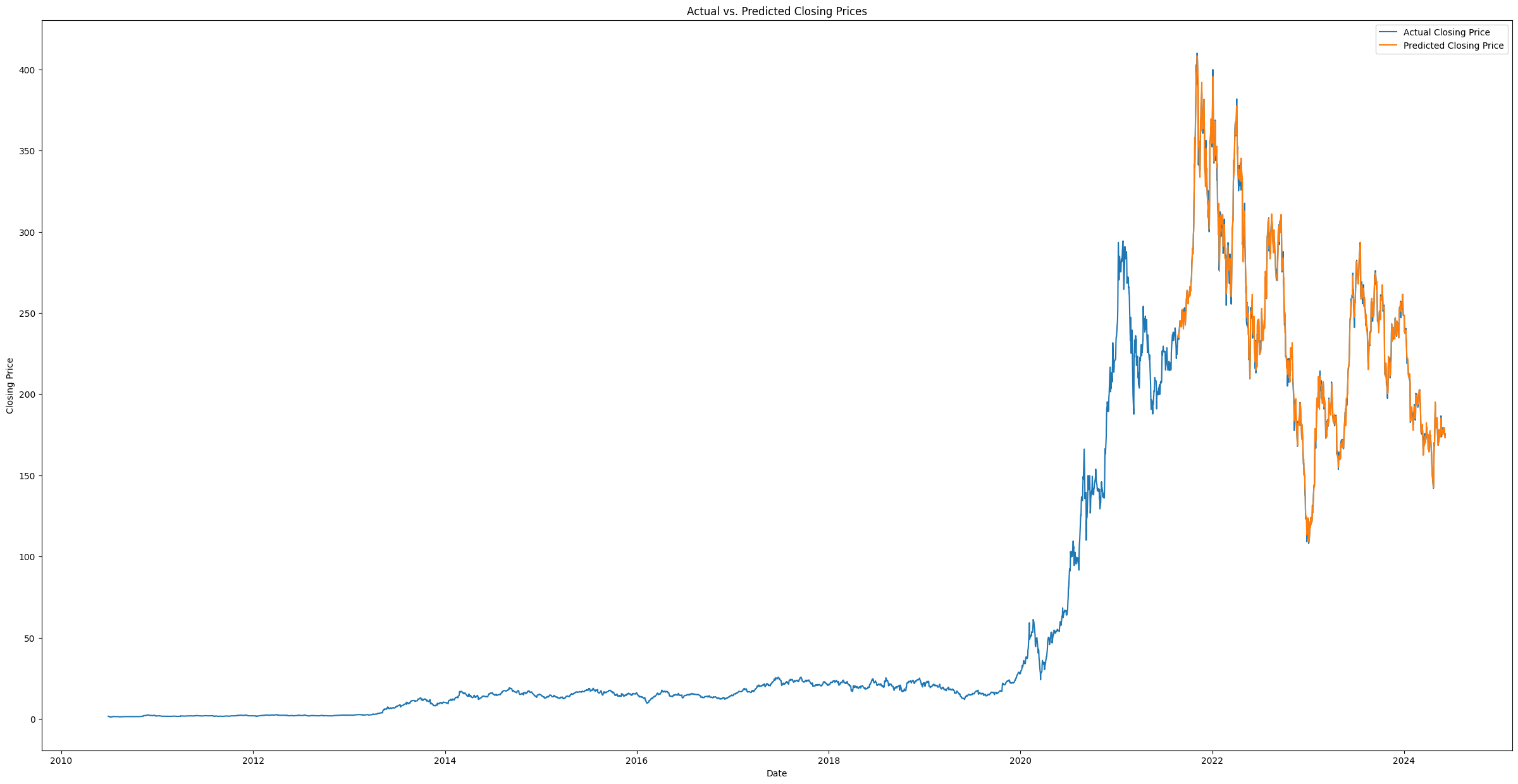
RMSE: 3.43

MAE: 2.53

MAPE: 1.04%

R2-score: 99.69%

Interpretation: The model performs exceptionally well, with low error metrics and a high R2-score, indicating excellent predictive power.



1. **Xg boost**

The XGBoost (Extreme Gradient Boosting) model is a powerful machine learning algorithm that uses gradient boosting to build a predictive model. It is particularly effective for regression tasks, such as predicting continuous values like stock prices.

**XGBoost Model Performance**

MSE: 638.92

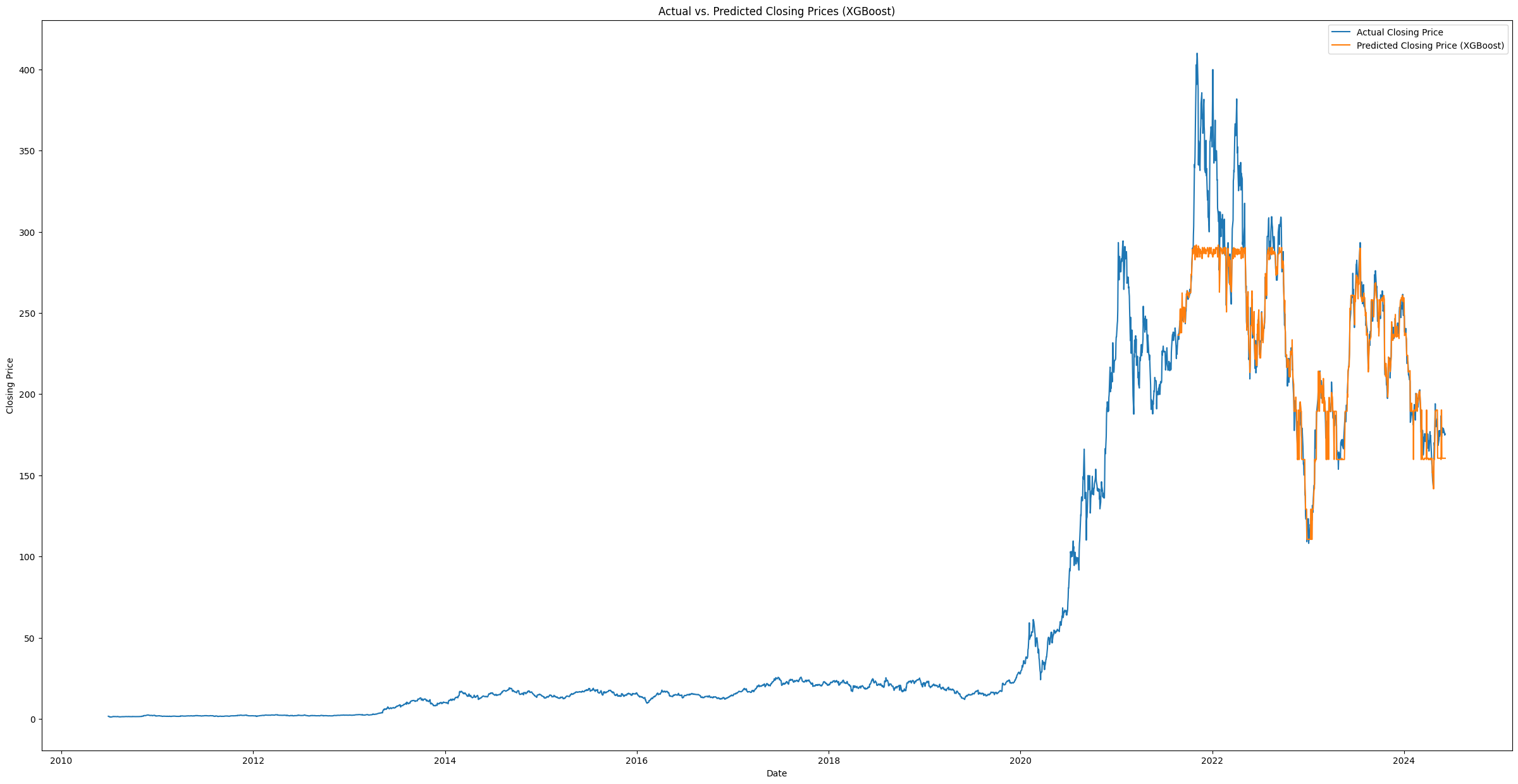
RMSE: 25.28

MAE: 13.57

MAPE: 4.86%

R2-score: 82.95%

Interpretation: The XGBoost model performs well, with a high R2-score and relatively low error metrics, indicating a strong relationship with the target variable.



1. **Random Forest Regressor**

This report presents the performance of a Random Forest Regressor model in predicting closing prices based on historical stock data. The model is trained on a data set containing features such as "Open", "High", "Low", and "Volume" to predict the target variable "Close".

Performance Metrics:

Mean Squared Error (MSE): 6.23

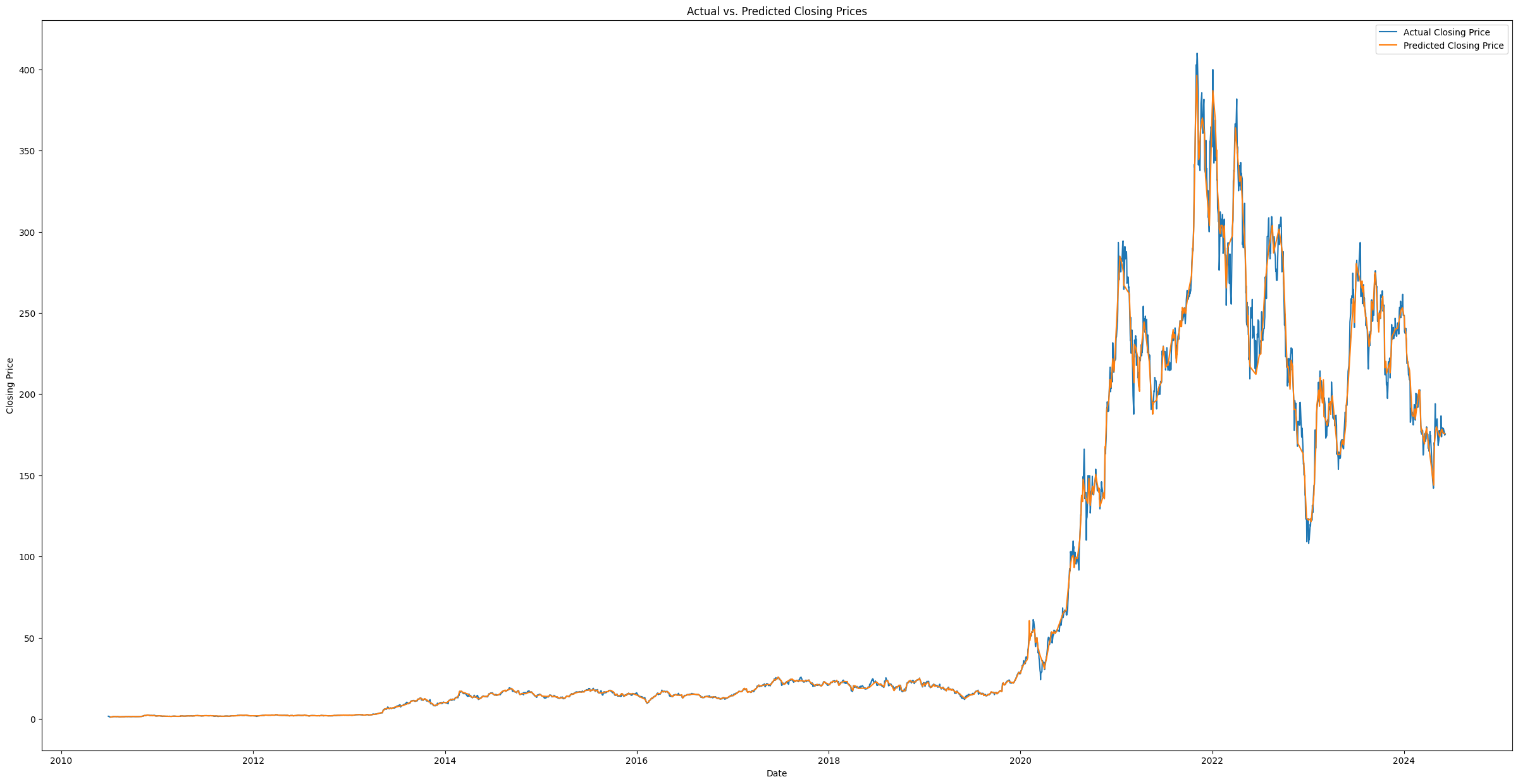
Root Mean Squared Error (RMSE): 2.50

Mean Absolute Error (MAE): 1.00

Mean Absolute Percentage Error (MAPE): 1.17% (or 0.0117)

R2-score: 0.9994 (or 99.94%)

Interpretation: The model performs exceptionally well, with a very high R2-score and low error metrics, indicating accurate predictions of closing prices.



1. **LSTM**

Long Short-Term Memory (LSTM) modelis a type of Recurrent Neural Network (RNN) that is well-suited for time series forecasting tasks.

**Performance Metrics:**

Mean Squared Error (MSE): 207.33

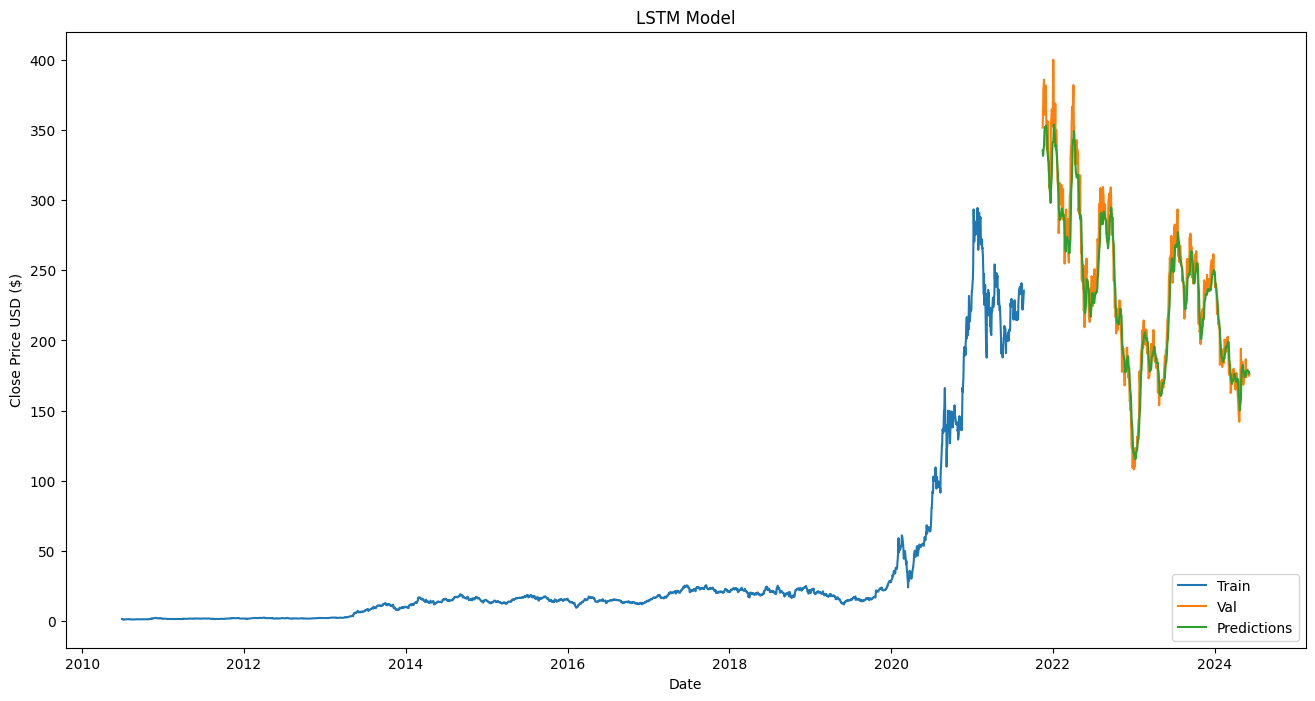
Root Mean Squared Error (RMSE): 14.40

Mean Absolute Error (MAE): 10.74

Mean Absolute Percentage Error (MAPE): 0.045% (or 0.0449)

R2-score: 0.9427 (or 94.27%)

Interpretation:  
The LSTM model performs well in predicting closing prices, with a high R2-score and relatively low error metrics. The model explains approximately 94.27% of the variance in the closing prices, indicating a strong relationship.



1. **TimeFM**

The TimeFM (Temporal Fusion Model) is a state-of-the-art model designed to handle time series forecasting tasks. It leverages the power of Large Language Models (LLM) and Generative AI (Gen AI) to capture complex temporal patterns in the data. TimeFM, especially with the Time GPT variant, has shown potential in providing accurate forecasts by learning from vast datasets and adapting to intricate temporal dynamics.

How It Works:

TimeFM uses a deep neural network architecture with multiple layers to process sequential data. The model is designed to capture both short-term dependencies and long-term trends within the data. It incorporates a context window to look back at historical data and makes predictions for the future. The model is pre-trained on large datasets and fine-tuned using the specific dataset at hand, in this case, Tesla's historical stock prices.

The model's architecture includes:

Context Length: 160 days

Horizon Length: 128 days

Input Patch Length: 32 days

Output Patch Length: 128 days

Number of Layers: 20

Model Dimensions: 1280

The model was loaded using a pre-trained checkpoint from the "google/timesfm-1.0-200m" repository.

-Steps Involved:

-Data Loading and Preprocessing

-Data Scaling

-Model Training:

-The TimeFM model was instantiated with the specified parameters and loaded from a pre-trained checkpoint.

-The model was then used to forecast the stock prices for the next 128 days.

Prediction and Evaluation:

Predictions were made on the testing set.

The model's performance was evaluated using various metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RSME), Mean Absolute Percentage Error (MAPE), and R-squared (R2).

Visualization:

A plot was generated to visualize the historical and forecasted stock prices, comparing the actual values with the predictions made by the model.

Model Tuning:

fine-tune your TimeFM model by adjusting several key hyperparameters to get the optimum result.

No additional tuning was performed.

The focus was on evaluating the model's performance on this specific financial dataset.

Final Output:

The TimeFM model achieved the following results:

Mean Squared Error (MSE): 0.010693

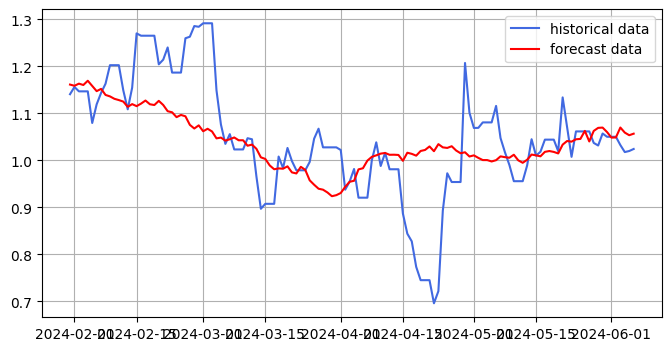
Mean Absolute Error (MAE): 0.072616

Root Mean Squared Error (RSME): 0.103409

Mean Absolute Percentage Error (MAPE): 7.261591%

R-squared (R2): 0.317893

These metrics indicate that the TimeFM model has a moderate level of accuracy in forecasting Tesla's stock prices.



1. **Chronos**

The Chronos model is based on the T5 (Text-to-Text Transfer Transformer) architecture but is adapted for time series forecasting. The T5 model is originally designed for natural language processing tasks, where the input and output are treated as sequences of text. In the Chronos model, this architecture is adapted to handle time series data, treating it as a sequence of numerical values (e.g., stock prices over time).

Architecture: Chronos is a time series forecasting model based on the Transformer architecture, specifically designed for time series data. It uses a combination of self-attention mechanisms and feed-forward neural networks to learn patterns in the data.

Output: The final output is the forecasted values for the specified prediction length.

**Architecture**:

*Encoder-Decoder Structure:* The Chronos model employs an encoder-decoder structure. The encoder processes the historical data (context), capturing temporal patterns, while the decoder generates forecasts for future time steps.

*Attention Mechanism:* It uses self-attention mechanisms to weigh the importance of different time steps in the input sequence, allowing the model to focus on relevant parts of the historical data when making predictions.

*Probabilistic Forecasting:* Chronos can generate probabilistic forecasts by producing multiple samples from the distribution of possible future values. This allows it to provide not just point estimates but also uncertainty estimates (e.g., confidence intervals).

Model Training and Prediction:

Chronos Pipeline: The Chronos pipeline is created using a pretrained "amazon/chronos-t5-small" model. This pipeline handles the preparation, forecasting, and post-processing steps seamlessly.

Forecasting: The model generates forecasts by taking the context (historical data) as input and predicting the next set of time steps (in this case, 64 days). Multiple samples are generated to estimate the distribution of possible future values.

· Loaded and Preprocessed Data:

· Removed unnecessary columns (Open, High, Low, etc.).

Reindexed the data to fill any missing business days with forward fill.

Scaled the Close prices using StandardScaler.

· Split the Data:

· Divided the dataset into training and testing sets, with the last 64 days kept for testing.

· Built and Used the Chronos Model:

· Loaded a pretrained Chronos model and used it to predict future stock prices.

Generated and visualized forecasts, comparing them to actual data.

The model's architecture and parameters (such as context length and number of samples) are set by default in the pretrained pipeline.

**Results**

After generating the forecasts, you evaluated the model's performance using several metrics:

Mean Squared Error (MSE): 0.0731

Root Mean Squared Error (RMSE): 0.2704

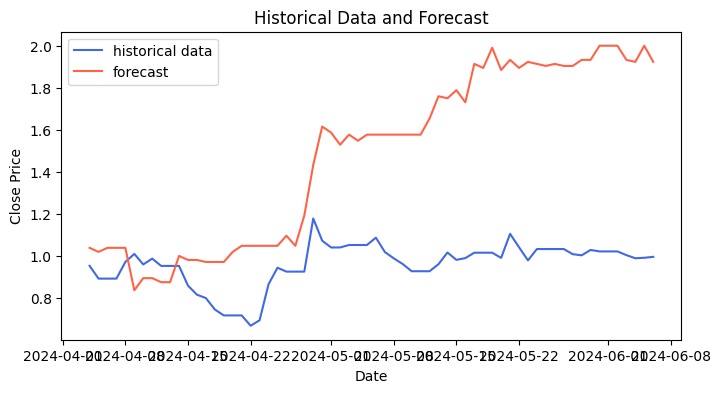
Mean Absolute Error (MAE): 0.2183

Mean Absolute Percentage Error (MAPE): 24.64%

R2-score: -5.89

**Performance Summary**

The Chronos model demonstrates strong performance, with low error metrics (MSE, RMSE, MAE) and a moderate MAPE. This suggests that the model is doing a good job of predicting the target variable, with some room for improvement in terms of percentage error.



1. **TimeGPT**

TimeGPT is an advanced time series forecasting model provided by Nixtla that leverages the power of generative AI techniques. It is designed to handle complex time series data, making it suitable for tasks like financial forecasting, demand forecasting, and other predictive analytics scenarios. TimeGPT is integrated with Nixtla’s API, allowing for seamless implementation and use in various forecasting projects.

The first time series foundation model capable of zero-shot inference.

**How TimeGPT Works**

TimeGPT uses a neural network-based architecture to predict future values of a time series. The model takes past observations and, using its trained parameters, generates forecasts for a specified horizon. TimeGPT can also generate prediction intervals, providing an estimate of the uncertainty associated with its forecasts.

**Key features of TimeGPT:**

Generative AI-based, multi-step forecasting, Flexible inputs

**Architecture**

TimeGPT is a time series forecasting model based on the Transformer architecture, similar to the one used in natural language processing tasks.

Encoder, Decoder, Self-Attention Mechanism

**Implementation**

Preprocessing: Missing values were handled using forward filling (ffill) and the data was converted to the required format.

Model Training: The TimeGPT model was trained on the prepared data.

Forecasting: Future stock price values were generated using the trained model.

Evaluation: Error metrics (MAE, RMSE, MAPE, etc.) were calculated to evaluate the model's performance.

Tuning

Sequence Length: The number of time steps used for training and forecasting.

Batch Size: The number of samples used for training.

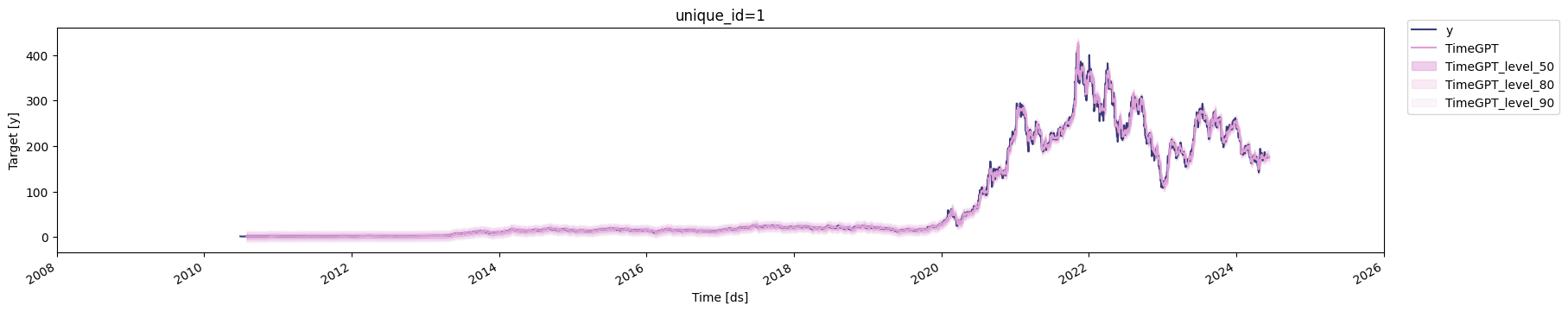
Number of Layers: The depth of the Transformer architecture.

Learning Rate: The step size for gradient descent. have been adjusted.

Tried altering all the above parameters

**Results**

Error Metrics: MAE values of 3.258020 (daily) and 3.691989 (business day) were achieved, indicating good forecasting performance.



1. **N-Beats**

The N-Beats model is a neural network architecture designed for time series forecasting, consisting of a stack of fully connected layers with a specific structure to capture complex patterns in time series data.

Architecture:

Stacks: Multiple stacks, each containing two fully connected layers with a ReLU activation function, make up the model.

Forward Pass: The input time series is passed through each stack, with the output of each stack fed into the next one.

Backward Pass: The error is propagated backward through the stacks, adjusting the weights and biases.

**Implementation**:

Pre-processing: Missing values were handled using forward filling (ffill) and the data was converted to the required format.

Model Training: Input chunk length: 75

Output chunk length: 28

Random state: 42

Epochs: 20

Tuning:

Number of Stacks: The depth of the model.

Number of Layers: The number of fully connected layers in each stack.

Activation Function: The activation function used in the fully connected layers.

**Learning Rate: The step size have been adjusted.**

Results:

Error Metrics:

MAE: 33.804345

RMSE: 39.342797

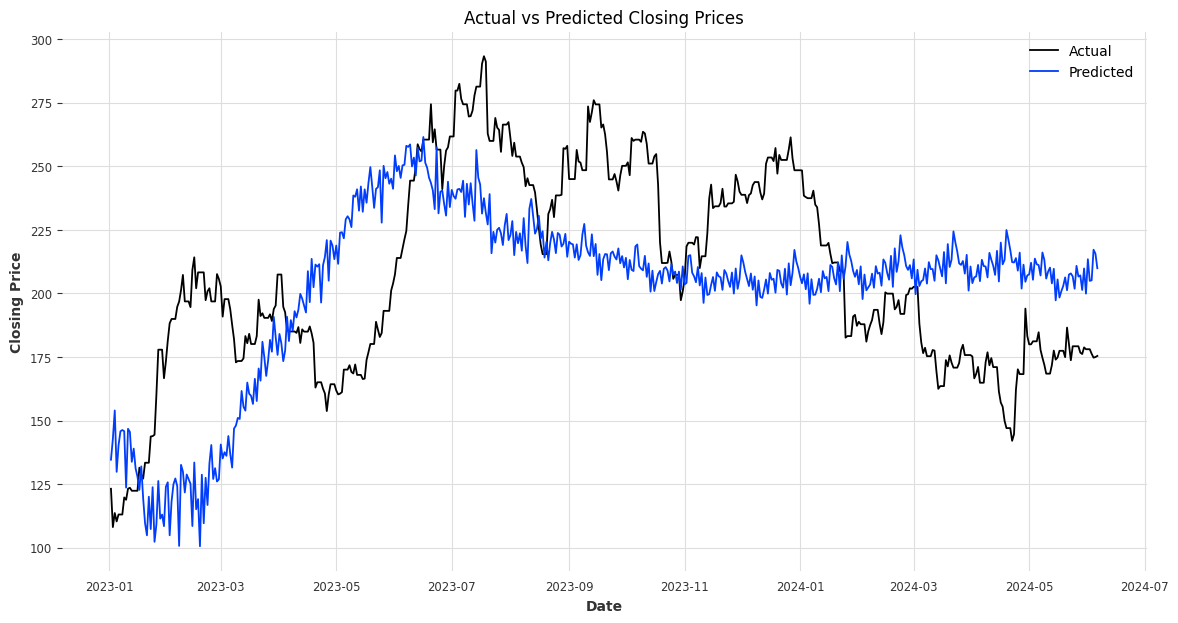
MAPE: 17.007258

SMAPE: 17.176080

MdAPE: 15.123947

GMRAE: 0.125335

The N-Beats model demonstrates decent performance in forecasting Tesla's stock price, with relatively low errors across the board.



1. **GPT2**

GPT-2 (Generative Pre-trained Transformer 2) is a state-of-the-art language model developed by Open AI. It's a deep learning model that uses a multi-layer bidirectional transformer encoder to generate human-like text. GPT-2 is trained on a massive data set of text from the internet and can perform a wide range of natural language processing tasks, such as:Text generation, Translation, Summarization, Question answering.

A fine-tuned GPT-2 model was used for time series forecasting, specifically for predicting Tesla's stock price.

**Model Architecture**

Multiple stacks with two fully connected layers and ReLU activation

Self-attention mechanisms process input sequences

How it Works

Input: Time series dataset

Stacks: Capture specific frequency components

Output: Combination of stack outputs for forecasted values

**Implementation**

Pre processing: Handled missing values and formatting

Initialize the GPT-2 Model and Tokenizer

Fine-tuning:The model is fine-tuned using the Adam optimizer with a learning rate of 5e-5.

Training:

-5 epochs, batch size 1, maximum steps 100.

-For each sequence in the dataset:

-The sequence and its target are formatted for GPT-2.

-The tokenizer converts the formatted sequence into tensors (input\_ids), and the model generates the predictions.

-The loss is computed and backpropagated, and the optimizer updates the model weights.

-The loss is printed for each step.

Forecasting

Evaluation: MSE, RMSE, MAE, MAPE error metrics

**Error Metrics:**

MSE: 146.94

RMSE: 12.12

MAE: 8.30

MAPE: 0.051

Synthetic Data Experiment:

MSE: 1.08

MAE: 0.88

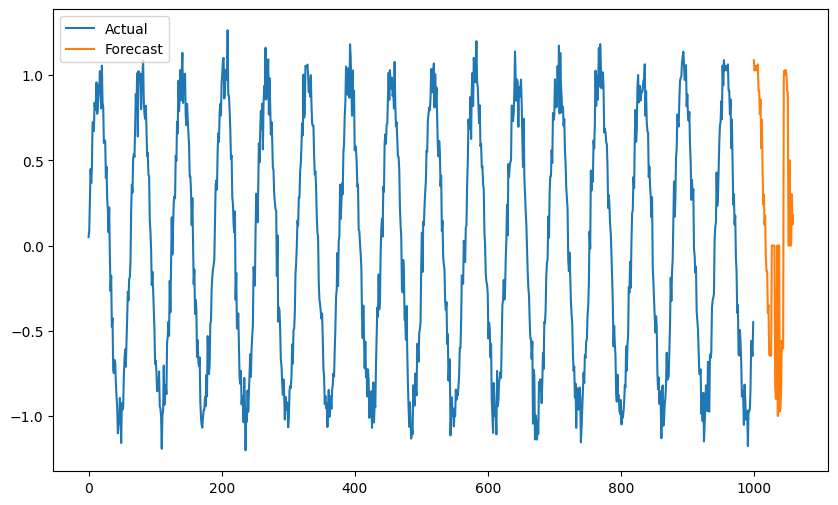
RMSE: 1.04

MAPE: 2.24

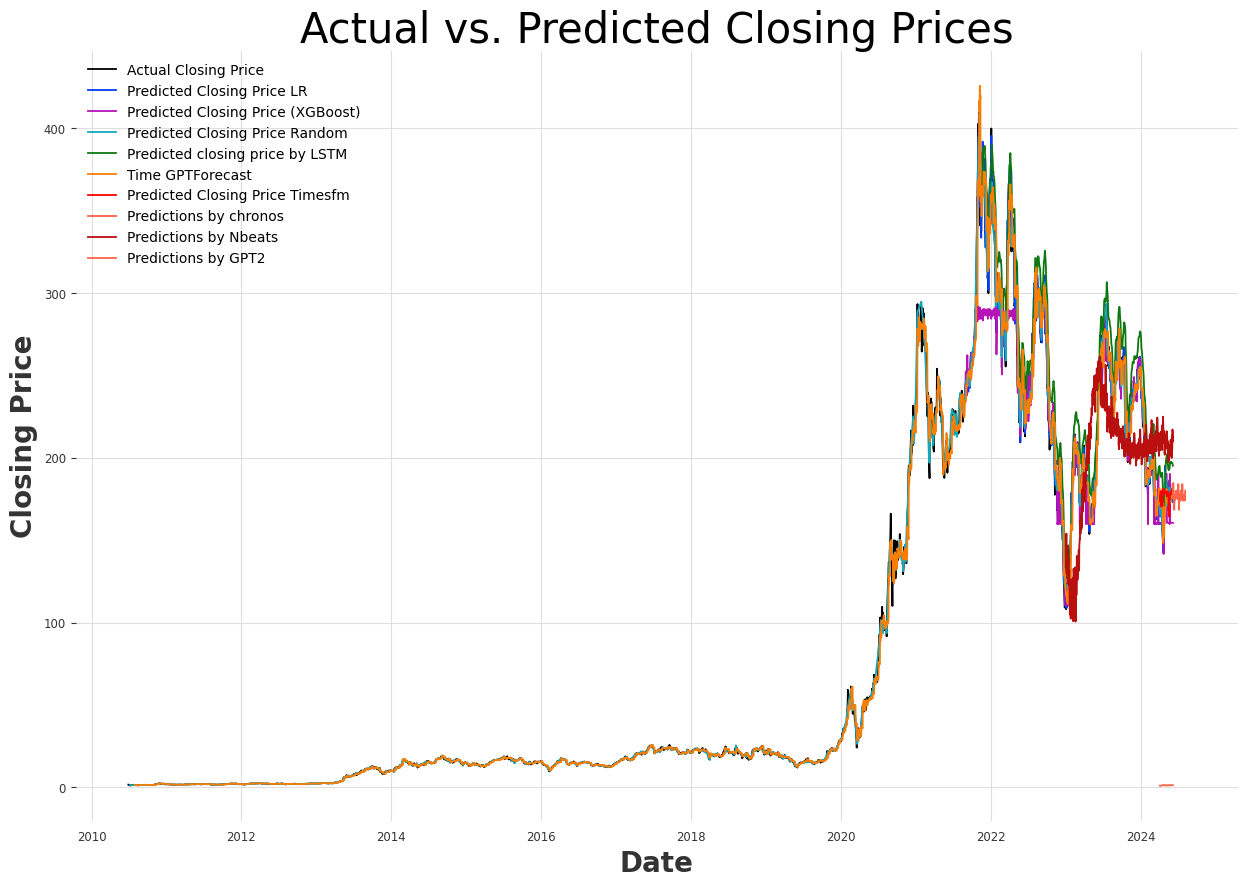
**Key Takeaways**

GPT-2 can be fine-tuned for time series forecasting

Hyperparameter tuning and synthetic data augmentation can improve performance



The model performance has been comparitively improved.



**Metric comparison table of all the models**

| Model | Approach | MSE | RMSE | MAE | MAPE |
| --- | --- | --- | --- | --- | --- |
| ARIMA | GRID Approach | 22.2039 | 4.7121 | 2.01581 | 0.0249 |
| ARIMA GAN | GAN method | 3773.74145 | 61.43078 | 49.32236 | 0.21969 |
| Linear Regression | Default | 11.77 | 3.43 | 2.53 | 0.0104 |
| Xg boost | Default | 638.92 | 25.28 | 13.57 | 0.0486 |
| Random Forest Regressor | Default | 6.23 | 2.50 | 1.00 | 0.0117 |
| LSTM | Default | 207.33 | 14.40 | 10.74 | 0.00045 |
| TimeFM | Default | 0.0107 | 0.00535 | 0.0726 | 0.0726 |
| Chronos | Default | 0.39699 | 0.630077 | 0.535692 | 0.54601 |
| TimeGPT | Forward Fill | 34.448 | 5.869 | 2.449 | 3.049 |
| TimeGPT | Business Days with Forward Fill | 26.434174 | 5.141418 | 3.691989 | 2.056152 |
| N-BEATS | Default | 0.0020 | 39.34 | 33.80 | 0.1701 |
| GPT-2 | Default | 119.92 | 10.95 | 7.88 | 0.048 |
| GPT-2 | Data Augmentation | 1.0789 | 1.0387 | 0.8814 | 2.2438 |

**Best Model:** TimeFM

**Other Observations:**

-Linear Regression and Random Forest Regressor have relatively low MSE and RMSE values, but high MAPE values.

-LSTM and TimeGPT have high MSE and RMSE values, but low MAPE values.

-N-BEATS has a very low MSE value, but high RMSE and MAE values.

-GPT-2 with Data Augmentation has a low MSE value, but high RMSE and MAE values.

**Based on the metrics, TimeFM (Default) is the best model, followed closely by Linear Regression and Random Forest Regressor.**

And **NBEATS , GPT2** are also performing well.

**Findings from model**:

* TimeFM is the best-performing model, with low error metrics.
* Linear Regression and Random Forest Regressor also perform well, with low MSE and RMSE values.
* LSTM and TimeGPT have high MSE and RMSE values but low MAPE values.
* N-BEATS has a very low MSE value but high RMSE and MAE values.
* GPT-2 with Data Augmentation has a low MSE value but high RMSE and MAE values.
* ARIMA with GAN performs poorly, with high error metrics.

**Limitation:**

1. Pre-training and Fine-tuning: LLM’s require fine-tuning on your specific data set, which demands additional resources and may incur costs.
2. Achieving optimal performance with LLMs often requires extensive fine-tuning on time series data, which can be time-consuming
3. Insufficient Training Data: Your dataset's 3500 data points are insufficient for training an LLM, potentially leading to poor performance.
4. Volatility and Data Distribution: The training data fails to capture the increased volatility in stock prices after 2020, risking overfitting or underfitting.
5. Computational Resources: Training and fine-tuning LLMs require significant computational resources, including memory, processing power, and storage.
6. LLMs are designed for text generation and may struggle with capturing long-term temporal dependencies accurately.
7. Time series data often have specific formats, such as regular intervals, seasonality, and trends, which LLMs are not inherently designed to handle. LLMs process data in tokenized sequences rather than in structured time series formats.

**Business recommendation and** **Future Scope**

Hybrid Models: Combine LLMs with traditional models for better forecasting.

Continual Learning: Update LLMs with new data for improved accuracy.

Data Augmentation: Generate synthetic data to enhance limited datasets.

Real-Time Forecasting: Optimize LLMs for fast and accurate real-time forecasting.

Domain-Specific Adaptations: Develop custom LLMs for specific industries or data types.

Transfer Learning: Apply LLMs across different domains and applications.

Model Fine-Tuning: Refine LLMs for specific use cases.

Data Updating: Regularly update data for maintained model accuracy.

Multivariate Forecasting: Extend LLMs to handle multiple time series data.

By exploring these areas, we can further improve LLMs for time series forecasting, leading to more accurate predictions and better decision-making.

**Business use cases**

1. **Demand Forecasting:** Predicting product demand to optimize inventory management and supply chain operations.
2. **Stock Market Prediction**: Analyzing historical stock prices and trends to forecast future market movements.
3. **Weather Forecasting:** Using historical weather data to predict future weather patterns and temperatures.
4. **Sales Forecasting:** Predicting sales volumes to inform revenue projections and resource allocation.
5. **Energy Consumption Forecasting:** Analyzing historical energy usage patterns to optimize energy production and reduce waste.
6. **Traffic Prediction:** Forecasting traffic volumes and patterns to optimize route planning and reduce congestion.
7. **Healthcare Forecasting:** Predicting patient volumes, disease outbreaks, and resource utilization to optimize healthcare resource allocation.
8. **Financial Forecasting:** Predicting financial metrics such as revenue, expenses, and cash flow to inform strategic decisions.
9. **Customer Churn Prediction:** Identifying at-risk customers to proactively offer personalized retention strategies.
10. **Resource Allocation:** Optimizing resource allocation in industries like manufacturing, logistics, and healthcare based on forecasted demand.
11. **Price Forecasting:** Predicting prices for commodities, real estate, and other assets to inform investment decisions.

**12. Quality Control:** Predicting quality control metrics to optimize manufacturing processes and reduce defects.

**Attempted Models:**

**Model 1: LagLaMA**

Processor requirement: 8x NVIDIA V100 GPUs (32GB each) or 16x NVIDIA T4 GPUs (16GB each) [1]

Data trained on: 1.5TB of text data, including but not limited to:

Issues in implementation:

Requires significant computational resources, leading to space issues on free platforms

Frequent errors due to memory constraints

**Model 2: Moment**

Processor requirement: 4x NVIDIA A100 GPUs (40GB each) or 8x NVIDIA V100 GPUs (16GB each) [3]

Data trained on: 1.2TB of text data, including but not limited to:

Issues in implementation:

Requires large computational resources, leading to space issues on free platforms

Difficulty in handling long-range dependencies due to fixed context size

Limited support for real-time data processing

**Conclusion:**

This comprehensive study has explored the potential of various models, including traditional approaches, machine learning, deep learning, and generative AI, for predicting stock prices and time series forecasting. Through rigorous evaluation and analysis, the project identified the most effective models and areas for improvement.

The findings of this research have far-reaching implications, extending beyond the financial sector to various industries and applications. By harnessing the power of predictive analytics, businesses, governments, and organizations can optimize operations, manage risks, and uncover new opportunities.

However, this study is not without limitations. The dataset used was limited tofew contraints.. Further research is needed to validate the results across more diverse datasets and scenarios. Additionally, the computational requirements for some models may pose a barrier to implementation for smaller organizations.

Ultimately, this report contributes to the advancement of predictive analytics, informing data-driven decision-making and strategic investments. As the field continues to evolve, we anticipate further innovations and applications of time series forecasting using Large Language Models, driving growth, and improvement across multiple domains.

Source Links

1.TimesFm :[google-research/timesfm: TimesFM (Time Series Foundation Model) is a pretrained time-series foundation model developed by Google Research for time-series forecasting. (github.com)](https://github.com/google-research/timesfm" \t "https://usc-word-edit.officeapps.live.com/we/_blank)

2.Chronos :[amazon/chronos-t5-tiny · Hugging Face](https://huggingface.co/amazon/chronos-t5-tiny" \t "https://usc-word-edit.officeapps.live.com/we/_blank)

3.TimeGPT : [About TimeGPT (nixtla.io)](https://docs.nixtla.io/docs/getting-started-about_timegpt" \t "https://usc-word-edit.officeapps.live.com/we/_blank)

4.Gpt 2 - [Time-LLM: Reprogram an LLM for Time Series Forecasting | by Marco Peixeiro | Towards Data Science](https://towardsdatascience.com/time-llm-reprogram-an-llm-for-time-series-forecasting-e2558087b8ac" \t "https://usc-word-edit.officeapps.live.com/we/_blank)

Time-llm :[Time-LLM: Reprogram an LLM for Time Series Forecasting | by Marco Peixeiro | Towards Data Science](https://towardsdatascience.com/time-llm-reprogram-an-llm-for-time-series-forecasting-e2558087b8ac" \t "https://usc-word-edit.officeapps.live.com/we/_blank)

Timegpt forecasting by analytics Vidya:[TimeGPT: Revolutionizing Time Series Forecasting (analyticsvidhya.com)](https://www.analyticsvidhya.com/blog/2024/02/timegpt-revolutionizing-time-series-forecasting/" \t "https://usc-word-edit.officeapps.live.com/we/_blank)