Forecasting The Stock Prices For AAPL Stocks Using 2006-2018 Data

K.Olivier, D.Partha, W.Bright.

December 23, 2024

1 Introduction

Forecasting financial time series, such as stock prices, has long been a challenging and important area of research due to the inherent complexity and volatility of financial markets. Stock prices are influenced by a multitude of factors, including macroeconomic indicators, market sentiment, geopolitical events, and company-specific news, which make their prediction highly non-linear and dynamic. Traditional statistical approaches, such as autoregressive models, have provided a foundation for understanding time-dependent patterns, but recent advancements in machine learning and deep learning have opened new possibilities for capturing intricate relationships and dependencies in financial data. These models leverage the sequential nature of time series to enhance prediction accuracy, offering a powerful tool for investors and analysts aiming to navigate the uncertainties of stock market behavior.

The task of forecasting stock prices for Apple Inc. (AAPL) using historical data from 2006 to 2018 involves applying statistical and machine learning techniques to predict future price movements based on past trends. This analysis seeks to identify patterns, correlations, and trends within the stock's historical price data, including factors such as daily closing prices, trading volume, and market behavior. By leveraging advanced forecasting models—such as time series analysis, regression models, and neural networks—the goal is to develop a reliable predictive framework that can inform investment strategies, risk management, and decision-making in the stock market. In this study, we employ three machine learning models Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and the Autoformer model as they represent key advancements in time-series forecasting and sequential data modeling. We explain a bit of these methods in the literature review section [3].

1.1 Data Description

The dataset consists of daily stock price data for Apple Inc. (AAPL) over 12 years, from January 1, 2006, to January 1, 2018. The data includes attributes related to Apple's stock performance, recorded for each trading day. The total number of records is 3,019 and the attributes are; Date, Open, High, Low, Close, Volume, and Name describing the trading date, price of the stock at market open, highest price of the stock during the trading day, lowest price of the stock during the trading day, stock's closing price at the end of the trading day, number of shares traded during the day and the ticker symbol (AAPL) of the stock respectively.

Table 1: Summary Statistics of AAPL Stock Prices (2006-2018)

	Open	High	Low	Close	Volume
Mean	64.673	65.257	64.033	64.663	1.311×10^8
Std	44.525	44.812	44.231	44.536	1.019×10^8
Min	7.390	7.560	7.170	7.240	1.148×10^7
Max	175.110	177.200	174.860	176.420	8.433×10^8

2 Exploratory Analysis

Figure 1 shows a line plot of the closing prices over time showing a clear upward trend, indicating that Apple's stock generally appreciated over the 12 years.

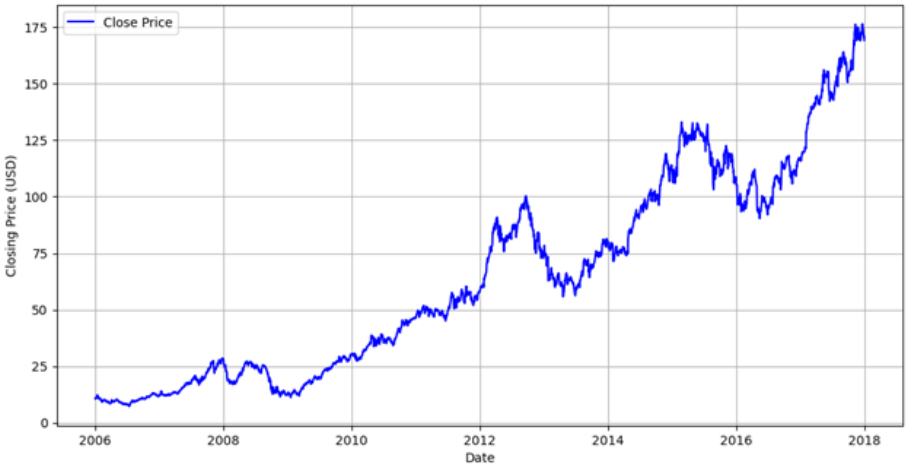


Figure 1: AAPL Closing Prices (2006-2018)

The trading volume of Apple Inc. (AAPL) from 2006 to 2018 shows a noticeable decline over the years, with significant fluctuations during the earlier part of the timeline, particularly between 2006 and 2008 as presented in Figure 2. This initial high trading activity might be associated with major company events, such as the launch of the first iPhone. After 2010, the volume exhibits a downward trend, indicating a potential stabilization in investor trading behavior as Apple matured and became a more established player in the market. However, there are occasional sharp spikes in trading volume throughout the years, suggesting event-driven activities, likely tied to product announcements, quarterly earnings reports, or broader market events. Overall, the data reflects a shift from high speculative trading in Apple's earlier growth phase to more stable trading patterns in later years.

2.1 Data Pretreatment Strategy

In this stage, we will prepare the time-series data by first handling missing values through forward fill or interpolation to ensure no gaps in the time indices, resampling if necessary. Next, we will apply

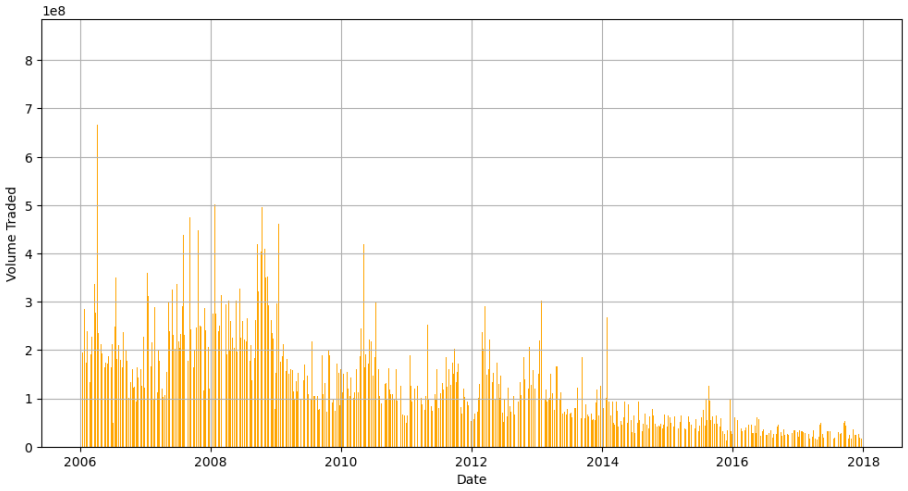


Figure 2: Daily Trading Volume of Apple Inc. (AAPL) from 2006 to 2018

transformations such as Min-Max normalization, z-score standardization, or log transformation to stabilize variance and scale the data effectively. We will then extract additional features like moving averages (MA), exponential moving averages (EMA), Bollinger Bands, and temporal features such as the day of the week, month, or quarter if relevant. Finally, we will split the data into training (70%), validation (15%), and test sets (15%) using a temporal split to maintain the sequence integrity.

Normalization:

$$p'_t = \frac{p_t - \min(p)}{\max(p) - \min(p)} \tag{1}$$

Moving Average:

$$\text{MA}_t = \frac{1}{k} \sum_{i=t-k+1}^{t} p_i \tag{2}$$

2.2 Missing Values, Synchronous and Outlier Detection

There is no missing data however one thing to know is that STL (Seasonal-Trend Decomposition using LOESS) can help identify outliers by isolating the residual component containing irregular variations.

The pair plot in 3 presents the relationships between the variables Open, High, Low, and Close. Its off-diagonal is a scatter plots revealing a strong linear relationship between all pairs of variables, indicating that they are highly correlated and move in sync with one another. This high degree of correlation is typical for stock market data, where daily prices across different metrics (Open, High, Low, and Close) tend to be closely related. The almost perfect positive correlation between these variables suggests that when one price metric increases, the others tend to increase proportionally.

Additionally, the scatter plots show a clear diagonal alignment with minimal scatter, implying there are no major outliers that could indicate disruptions in the synchrony among these variables.

This alignment suggests that the data is consistently aligned across all variables, with no visible signs of irregularities, missing data, or misaligned timestamps. If there were synchrony issues, we would expect to see more dispersed points or non-linear patterns, which are absent in these plots.

The histograms along the diagonal of the pair plot provide insights into the distribution of each variable. While the distributions are slightly skewed, which is common in financial time series data, they do not exhibit irregular spikes or gaps. This further indicates that the data is well-synchronized, without significant anomalies that could affect the alignment of variables over time.

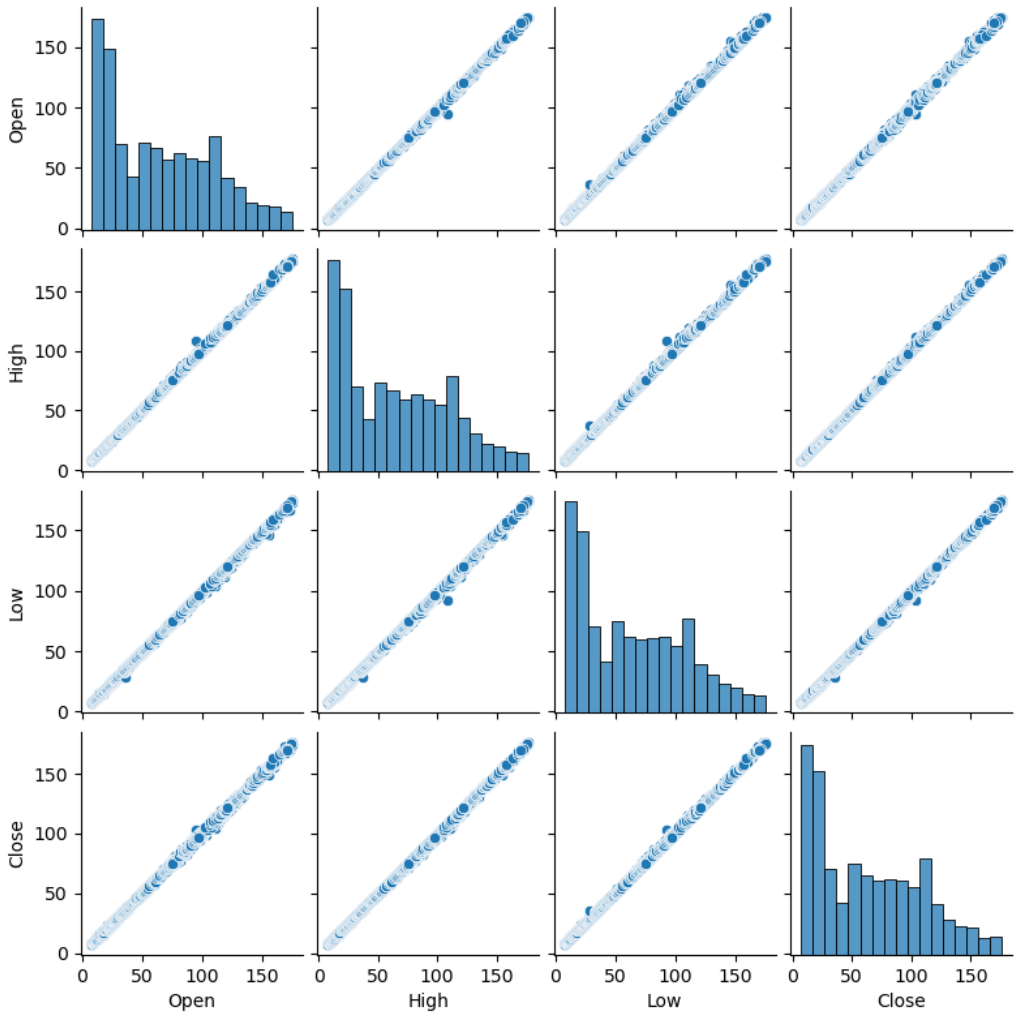


Figure 3: Pair plot of stock prices

2.3 Time Series Decomposition

We decomposed the time series into three components: *trend*, *seasonality*, and *residuals* using the multiplicative model. This can be expressed using formula 1 [5]:

$$y_t = S_t \times T_t \times R_t \tag{1}$$

where y_t represents the observed data, S_t denotes the seasonal component, T_t is the trend-cycle component, and R_t refers to the remainder (residual) component, all at time t.

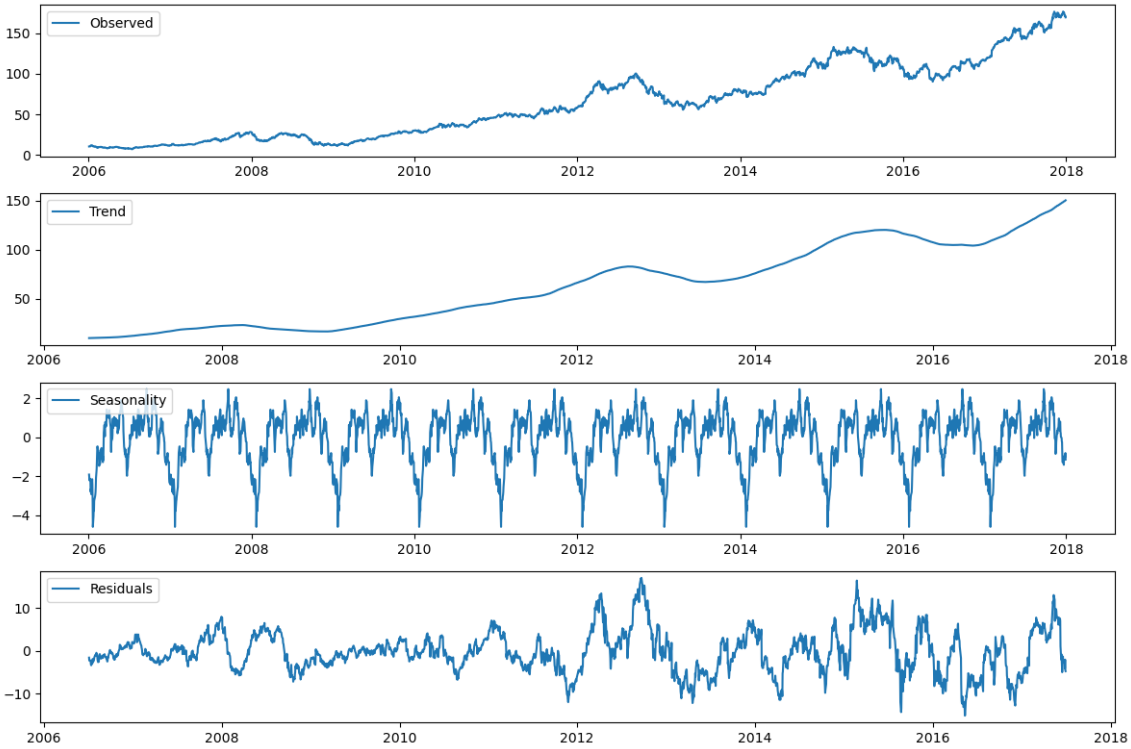


Figure 4: The AAPL stock price data and its three components.

The three components are shown separately in the bottom three panels of Figure 4. These components can be multiplied to reconstruct the data in the top panel.

Notice that the trend in panel 2 highlights a consistent upward trajectory, especially after 2013, indicating a long-term growth in Apple's stock price. There are also visible periods of slowdown, particularly around 2008 and 2015. The third panel shows the repeating seasonal patterns, suggesting that Apple's stock price exhibits regular cycles within each year. The consistent peaks and troughs indicate a strong seasonal effect, possibly driven by predictable annual events like product launches or holiday sales. The fourth panel is errors or residuals which appear to vary over time, with some larger deviations around 2008, 2014, and 2014, possibly reflecting market anomalies or macroeconomic impacts during those years.

2.4 Correlation and Autocorrelation analysis

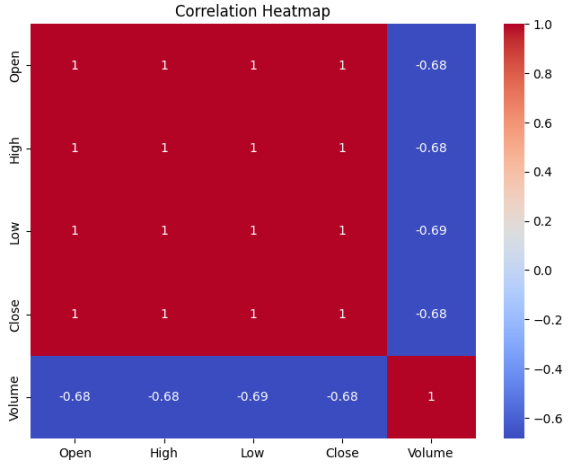


Figure 5: Correlation coefficients between variables

Based on the correlation heat map presented in 5, variables that have perfect or near-perfect correlation carry redundant information. Including all of them won't add value to the model and could even introduce collinearity issues (leading to instability in model training).

The ACF plot presented in Figure 6 shows a very slow decay, which indicates non-stationarity in the data. The high autocorrelation values across many lags suggest a strong trend component, as is typical in non-stationary time series. The PACF on the left part of Figure 6 shows a significant spike at lag 1, followed by values near zero for subsequent lags. This is characteristic of a time series that could be well-captured by an autoregressive (AR) process of low order, such as AR(1).

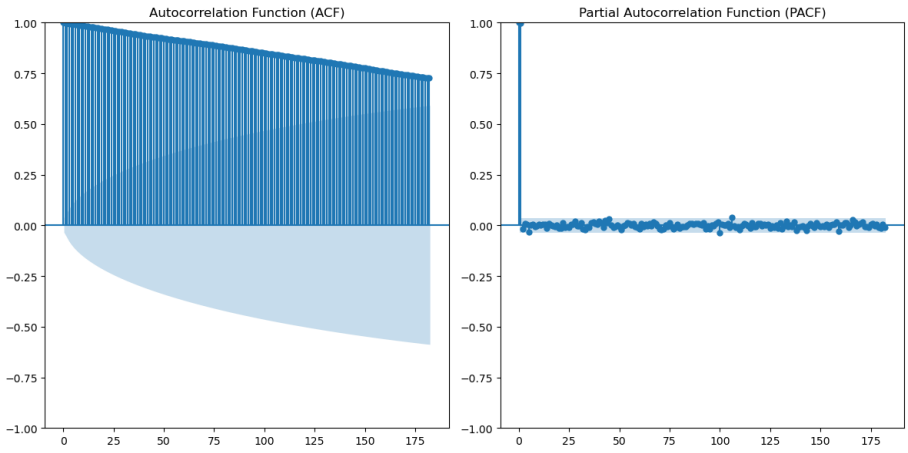


Figure 6: Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) with 182 Lags

3 Literature Review

RNNs are designed to handle sequential data by maintaining a hidden state that captures temporal dependencies, but they suffer from vanishing and exploding gradient problems when modeling long-term dependencies [3]. LSTMs, an enhancement of RNNs, address these limitations through gated mechanisms—namely, input, forget, and output gates—which regulate the flow of information and allow for effective long-term dependency learning [4]. Despite their effectiveness, both RNNs and LSTMs face computational inefficiencies with long sequences. Autoformer, a transformer-based model specifically designed for time-series forecasting, introduces an auto-correlation mechanism and seasonal-trend decomposition, enabling more efficient and accurate long-range time-series predictions [8]. Autoformer outperforms traditional LSTM and RNN models in scalability and capturing complex temporal patterns, marking a significant leap in sequence modeling for time-series data.

3.1 Sesonality Effects on Sampling and Sub-sampling Techniques for Time Series Data

Time series data must be divided in a way that prevents data leaking and maintains the chronological sequence. These time series splitting strategies can help you maintain the accuracy and dependability of your models[1].

TimeSeriesSplit

It is done by dividing your data into successive folds so that it guarantees that each test set is created from future data and each training set is created from historical data.

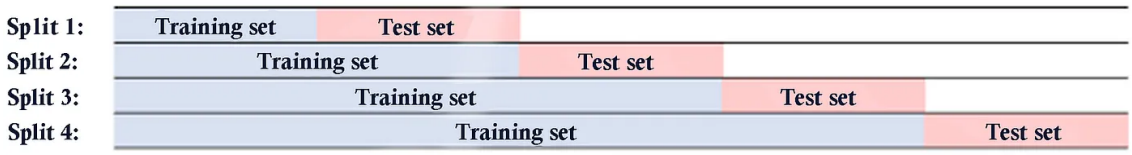


Figure 7: Time series split

Sliding/Rolling Window Split

With the rolling window technique, a fixed-size training window rolls along your dataset as your model advances in time. It's similar to moving forward while always monitoring the past. There are more versions for this method like Expanding Window Split and sliding window with gap where you leave some gap between train and test set.

[1]More explanation these methods can be found on https://medium.com/@mouadenna/time-series-splitting-techniques-ensuring-accurate-model-validation-5a3146db3088

Figure 8: Rolling Window Split

Group Time Series Split

For non-overlapping groups, GroupTimeSeriesSplit is a scikit-learn compatible version of time series validation with groups. In order to prevent data leaking in time series data, this technique makes sure that the training and test sets do not overlap.

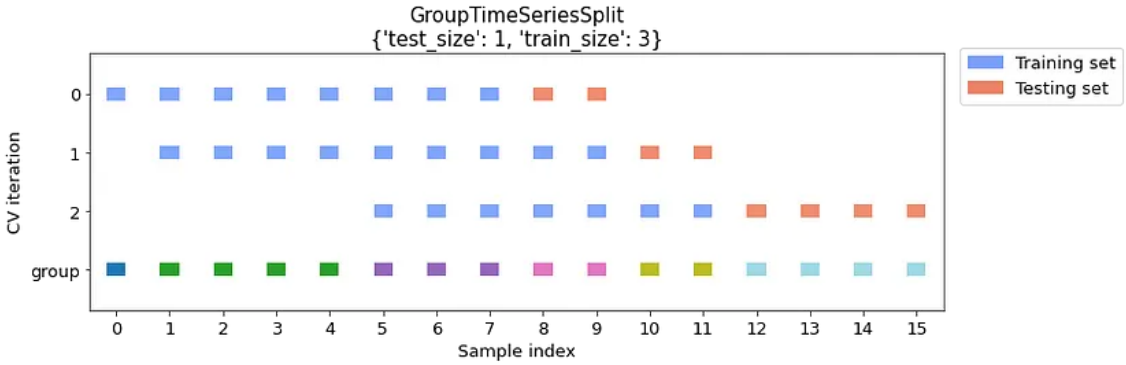


Figure 9: Group Time Series Split

Seasonality plays a critical role in how sub-sequencing is approached. Ignoring seasonality can lead to misleading results, especially if different segments of the series exhibit strong seasonal variations. Seasonal adjustments are often performed prior to sub-sequencing to normalize the patterns across different sub-series, making it easier for models like LSTM to learn temporal dependencies without being biased by repetitive seasonal effects. Research [6] indicates that seasonality can significantly influence forecasting accuracy. For instance, methods that adjust for seasonality prior to modeling, such as the Theta method, have shown strong performance in competitions like M4 forecasting competition [7]. These methods involve decomposing the time series into trend, seasonal, and residual components before applying machine learning models like LSTM to each component separately.

LSTM models are capable of learning long-term dependencies in time series data, but their performance can be enhanced by explicitly considering trends and seasonality [6]. One strategy is to use external feature engineering, where seasonal indices or trend components extracted from the original series are fed as additional inputs to the LSTM model. Another approach is to use seasonal differencing to remove seasonality from the series before training the model, thus making it easier for the LSTM to capture the underlying patterns.

4 Methodology

In this section, we present the methodology employed in this study, which includes data standardization techniques to ensure uniform scaling across features, evaluation metrics for assessing model performance, and a structured workflow outlining the key steps from data preprocessing to model evaluation.

4.1 Strategy and workflow for the Model Creation RNN and LSTM

The diagram below shows the general strategy for the task. The modeling strategy involves preprocessing raw data and splitting it into three subsets: training (70%), validation (15%), and testing (15%).

The training and validation sets form the calibration set, used for training and tuning the LSTM and RNN models to optimize their performance. Metrics like Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and training time are used to evaluate model performance during training. After training, the models are tested on the reserved testing set to assess their generalization ability using MSE and MAPE.

Finally, the results from both models are compared based on their training time and performance metrics across both the calibration and testing phases, helping to identify the best-performing model.

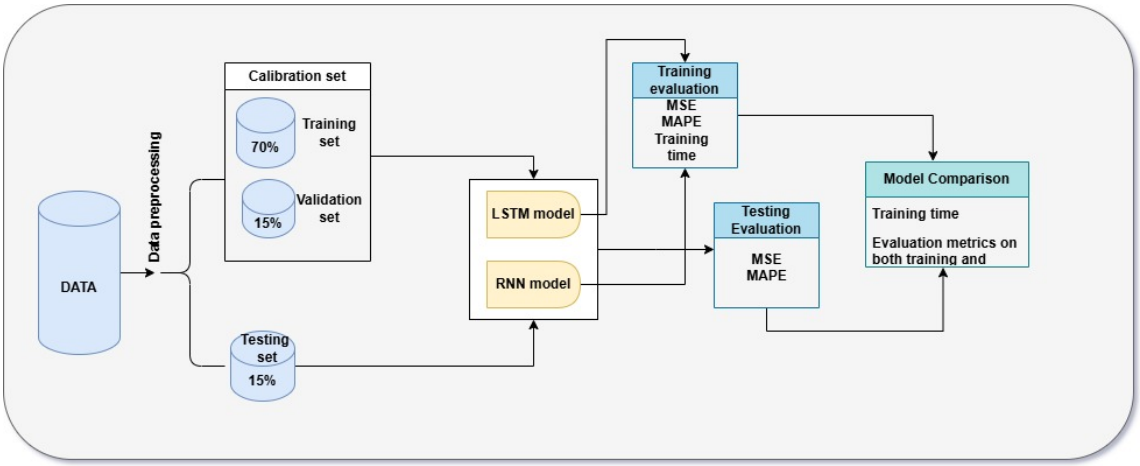


Figure 10: Mainstream Modeling Strategy Overview

4.2 Standardization Methods

Min-Max Scaling:

This method is often used to normalize data between 0 and 1, which aids LSTM models in converging faster during training. It is especially effective when the data values are on different scales.

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

Z-score Normalization:

Involves transforming the data to have a mean of 0 and a standard deviation of 1. This is useful for stabilizing variance and is widely used when features are normally distributed.

$$z = \frac{x - \mu}{\sigma}, \quad \text{where } \mu \text{ is mean and } \sigma \text{ is standard deviation}$$

Robust Scaling:

This technique uses the interquartile range (IQR) to scale data, making it robust to outliers. It is particularly beneficial for datasets that contain significant outlier values, ensuring that the model training is not skewed.

4.3 Evaluation metrics

We will use key metrics and a structured evaluation strategy to evaluate the model's performance. Mean Squared Error (MSE) will be used for our tasks to quantify the average squared difference between the predicted and actual values. We will use the Mean Absolute Percentage Error (MAPE) to assess the relative prediction error, which measures the error as a percentage of the actual values.

To ensure robust evaluation, we will implement a backtesting framework using a rolling window approach. In this method, we will first train the model on data from t_1 to t_{k-1} and validate on t_k. Subsequently, we will slide the window forward, training on t_2 to t_k and validating on t_{k+1}. This approach allows us to simulate real-world performance by iteratively testing on unseen data while maintaining the time-series order.

Additionally, we will perform residual analysis by plotting the residuals (the differences between predicted and actual values) to identify any patterns or unexplained variance. This step helps to check for model biases and ensure that the residuals are randomly distributed.

The mathematical formulations for the evaluation metrics are as follows. The Mean Squared Error (MSE) is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where $\hat{y}_i$ represents the predicted value and y_i the actual value for the i-th instance. The Mean Absolute Percentage Error (MAPE) is given by:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

5 Implementation and Results

5.1 Baseline Model: Autoregressive (AR) Model

We have chosen the AR model as our baseline model because it is a fundamental part of time series forecasting [1] and is often used as a baseline due to its simplicity. It's particularly suitable for modeling data that shows temporal dependence [2]. The AR model's performance can be further improved with seasonal adjustments or integrated autoregressive models (ARIMA), which are widely used in time series forecasting in finance and economics.

We implemented a baseline autoregressive (AutoReg) model for time series to forecast close price. We splited the dataset into training (80%) and testing (20%) sets. The AutoReg model is trained using 730 lags (representing roughly 2 years of daily data). Predictions are made for the test set, and the model's performance is evaluated using Root Mean Squared Error (RMSE). The results are visualized by comparing the predicted and actual stock prices over time as presented in Figure 11. The root mean squared error was estimated to be 10.53.

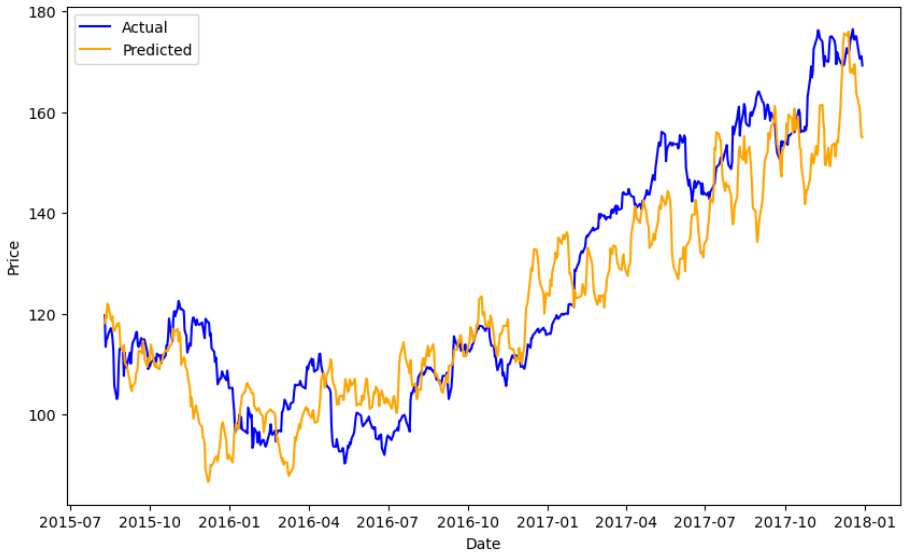


Figure 11: Autoregressive Model: Predictions vs Actual

5.2 RNN and LSTM models

5.2.1 Models Architecture with Layer Graph

The LSTM model architecture plays a crucial role in capturing and learning temporal patterns from the sequential data. we have designed a stacked LSTM model with dropout regularization to prevent overfitting. Stacking multiple LSTM layers enables the model to learn complex relationships in the data. Figure 12 presents LSTM layers with corresponding units and dropout layers with certain portions.

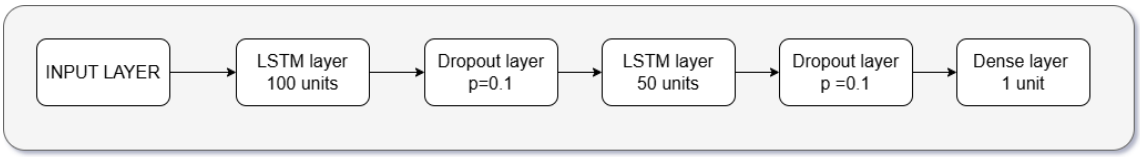
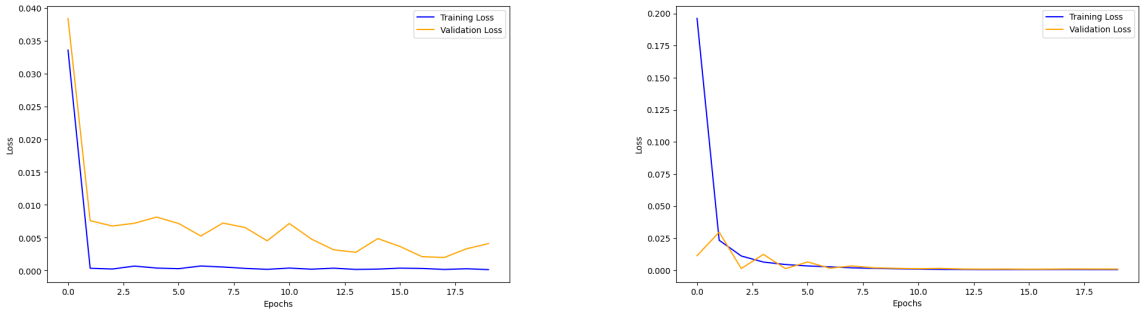


Figure 12: The architecture of the LSTM model

As for the RNN model, we will also stack the simpleRNN layers in the same manner. To get the structure, we only replace the LSTM layers by simpleRNN layers.

In addition to the two models, we will explore a model-based transformer approach, specifically utilizing Autoformer as an innovative decomposition architecture featuring an Auto-Correlation mechanism. Autoformer retains the residual and encoder-decoder structure but transforms the traditional Transformer into a decomposition-based forecasting architecture. By incorporating our proposed decomposition blocks as core operators, Autoformer can gradually extract long-term trend information from the predicted hidden variables. This design enables the model to iteratively decompose and refine intermediate results throughout the forecasting process. [8]

5.2.2 Track of training and validation losses



(a) RNN Loss

(b) LSTM Loss

Figure 13: Comparison of RNN and LSTM Loss Curves

The RNN model exhibits a steep decrease in training and validation loss during the initial epochs, with validation loss stabilizing at a slightly higher value than the training loss. This behavior suggests that the RNN successfully learns the patterns but might exhibit mild overfitting as the validation loss plateaus above the training loss.

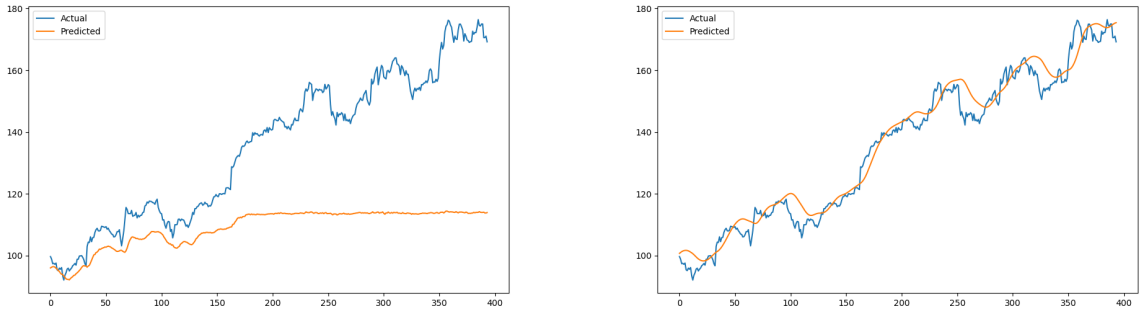
The LSTM model, on the other hand, shows a significantly sharper reduction in loss during the first few epochs, with the training and validation loss curves closely aligned throughout the training process. This alignment indicates that the LSTM model generalizes better than the RNN, likely due to its enhanced capacity to capture long-term dependencies and mitigate vanishing gradients. Overall, the LSTM demonstrates superior performance with lower overall loss values, suggesting its greater effectiveness.

5.2.3 Evaluation metrics for RNN and LSTM models

Model	Training time	MSE	MAPE
RNN	190.00 seconds	820.1351	15.4948%
LSTM	8.02 seconds	16.5660	2.2950%

Table 2: Summary results

5.2.4 Comparing predictions on the test set



(a) RNN: predictions vs.actual test data

(b) LSTM: predictions vs. actual test data

Figure 14: Comparison of RNN and LSTM Loss Curves

The RNN struggles to accurately predict stock prices over time due to its inherent limitations in capturing long-term dependencies. Initially, in the starting window, the RNN's predictions are relatively close to the actual values. However, as time progresses, the RNN fails to follow the upward trend and volatility of the stock prices, resulting in predictions that plateau at a narrow range. This is a typical limitation of RNNs, as they suffer from issues like vanishing gradients, which restrict their ability to learn and retain patterns over extended sequences.

The LSTM significantly outperforms the RNN by better capturing the stock price trends and fluctuations over time. In the starting window, the LSTM predictions align closely with the actual values, and as the test period progresses, the LSTM continues to track the upward trend while partially capturing the inherent volatility. This improvement is due to the LSTM's ability to handle long-term dependencies through its gating mechanisms, which selectively retain relevant information and discard irrelevant data. Although the LSTM exhibits some lags and slight mismatches in predicting peaks and troughs, its overall performance demonstrates its superiority in modeling the non-linear and dynamic nature of stock prices compared to the RNN.

The alignment of the two lines indicates the model's performance in capturing the trends and patterns of the time series data. Based on the visual, the predictions closely follow the actual prices, suggesting that the model effectively leveraged Close and Volume, to improve forecasting accuracy.

5.3 AUTOFORMER MODEL

5.3.1 Architecture

The Autoformer design consists of an encoder-decoder architecture that leverages an innovative decomposition mechanism for time series forecasting. The encoder processes the input time series by first applying an Auto-Correlation block to capture dependencies, followed by a series decomposition block that separates the seasonal and trend-cyclical components. The residual information is then passed through a feed-forward layer and another decomposition step, refining the extracted features. The decoder mirrors this process, beginning with initial seasonal and trend-cyclical values, iteratively applying Auto-Correlation, series decomposition, and feed-forward operations. This progressive decomposition-refinement cycle enables the model to effectively extract long-term trends

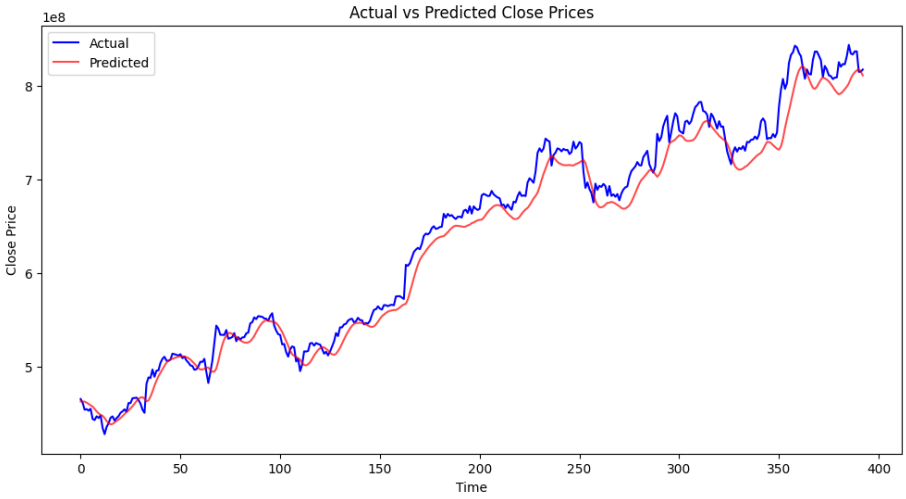


Figure 15: Close price predictions using multivariate data

and seasonal patterns from the data, enhancing forecasting accuracy by alternately refining intermediate results.

Building on the hyperparameters used by the author, which demonstrated high accuracy, we aim to apply transfer learning with the L2 loss function, utilizing the ADAM optimizer with an initial learning rate of 10^{-4}. The batch size is set to 32, and early stopping is applied within 10 epochs. The hyperparameter c for Auto-Correlation is tuned within the range of 1 to 3 to balance performance and efficiency. The Autoformer will model consists of 2 encoder layers and 1 decoder layer.

5.3.2 Training and Validation loss

The loss curve of the Autoformer model in Figure 17 demonstrates efficient training and generalization across 50 epochs. Initially, the validation loss peaks significantly higher than the training loss, indicating a high error in predictions on unseen data. However, within the first 10 epochs, both losses exhibit a sharp decline, with the validation loss converging rapidly, signifying the model's ability to generalize effectively in the early stages of training. The training loss, though initially fluctuating, follows a similar downward trajectory, highlighting the optimization process's effectiveness.

Beyond epoch 10, both training and validation losses stabilize at near-zero levels, with minimal fluctuations. This suggests that the model has successfully captured the underlying patterns in the data without overfitting, as evidenced by the close alignment of the losses. The results indicate excellent performance and robust generalization capabilities of the Autoformer model.

5.3.3 Autoformer predictions on the test set

Figure 18 illustrates the performance of the Autoformer model in predicting stock prices compared to the actual values over time. The blue line represents the actual stock price, while the orange line

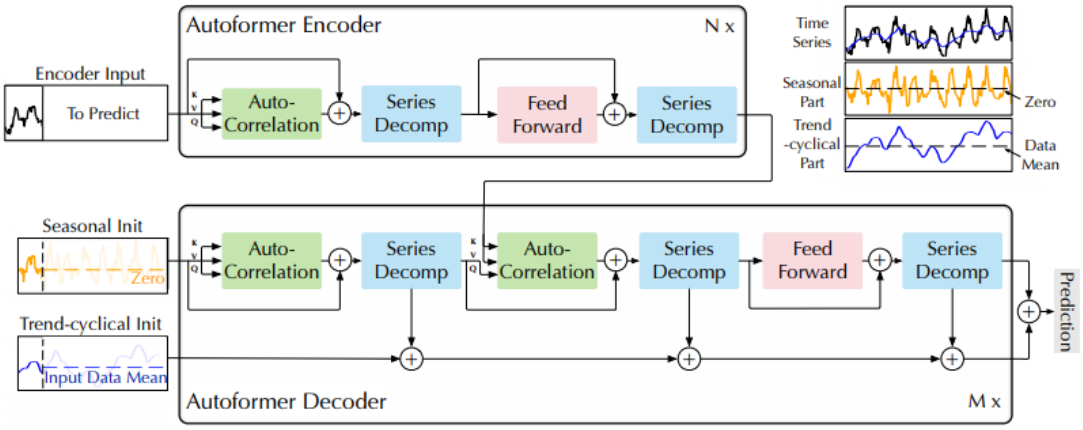


Figure 16: Autoformer architecture, Source: [8]

corresponds to the model's predictions.

The Autoformer captures the overall upward trend effectively, with its predictions closely tracking the general direction of the actual values. However, it appears to underperform in capturing the short-term fluctuations and sudden spikes, particularly during periods of rapid changes in stock prices. This behavior suggests that while the model is adept at learning long-term trends, it might smooth out short-term volatility, which could limit its applicability for highly dynamic market scenarios. Nonetheless, its strong alignment with the general trajectory indicates robust long-term forecasting capability.

5.3.4 Analysis of prediction power for Autoformer

Analysis of the Autoformer model's predictive power, assessed through quarterly Mean Squared Error (MSE) 19 on distinct 3-month periods, offers insights into its localized performance. The observed trend in MSE, which may show fluctuations across quarters, highlights the model's ability to capture patterns and dynamics specific to each period. An increase in MSE during certain quarters could indicate challenges in predicting market movements due to increased volatility, unforeseen events, or shifts in underlying market trends. Conversely, a decrease in MSE in other quarters might suggest a better alignment between the model's learned patterns and the prevailing market conditions during those periods.

However, it is crucial to avoid over-interpreting isolated fluctuations in quarterly MSE. Randomness in data and model predictions can contribute to minor variations in MSE, potentially obscuring the underlying performance trend.

Evaluation of the Autoformer model's predictive power using cumulative period Mean Squared Error (MSE) in Figure 20 provides a comprehensive understanding of its performance across varying time horizons. The observed trend in MSE, typically exhibiting an increase with the lengthening of cumulative periods, aligns with the expectation of diminishing predictive accuracy as the forecast

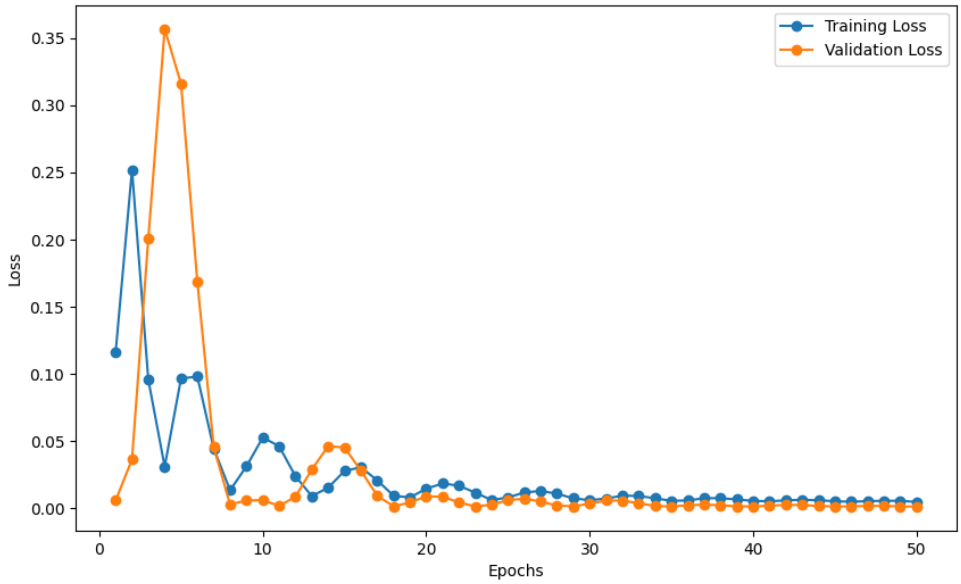


Figure 17: Autoformer track of training and validation loss

horizon extends. This behavior stems from the inherent challenges associated with forecasting further into the future, where model assumptions and learned patterns may become less relevant. Moreover, the accumulation of errors from earlier periods can contribute to an overall larger MSE in longer cumulative periods.

It is crucial to acknowledge potential deviations from a strictly monotonic increase in MSE. Factors such as model stability and inherent accuracy can influence the MSE trend. A robust model demonstrating consistent performance over longer periods might display a plateau or even a slight decrease in MSE, indicating sustained predictive power despite increasing forecast horizons. Additionally, the presence of seasonality or cyclical patterns in the data can introduce fluctuations in MSE across different cumulative periods, reflecting the model's ability to capture or struggle with these temporal dynamics.

6 Discussion

6.1 Model Improvement Strategy: RNN and LSTM

To improve the performance of the models, a focused approach on advanced architectures and parameter tuning is necessary. For the RNN, replacing it with an LSTM or Autoformer can address its inability to capture long-term dependencies, while incorporating additional features like technical indicators or sentiment data can enhance its learning capacity. For the LSTM, fine-tuning hyperparameters such as learning rate, batch size, and sequence length is crucial to better capture temporal patterns and volatility that can remove the smoothness in the LSTM prediction. Introducing attention mechanisms can help the model focus on relevant time steps, improving its predictive accuracy. Additionally, employing rolling-window cross-validation and regularization techniques like dropout

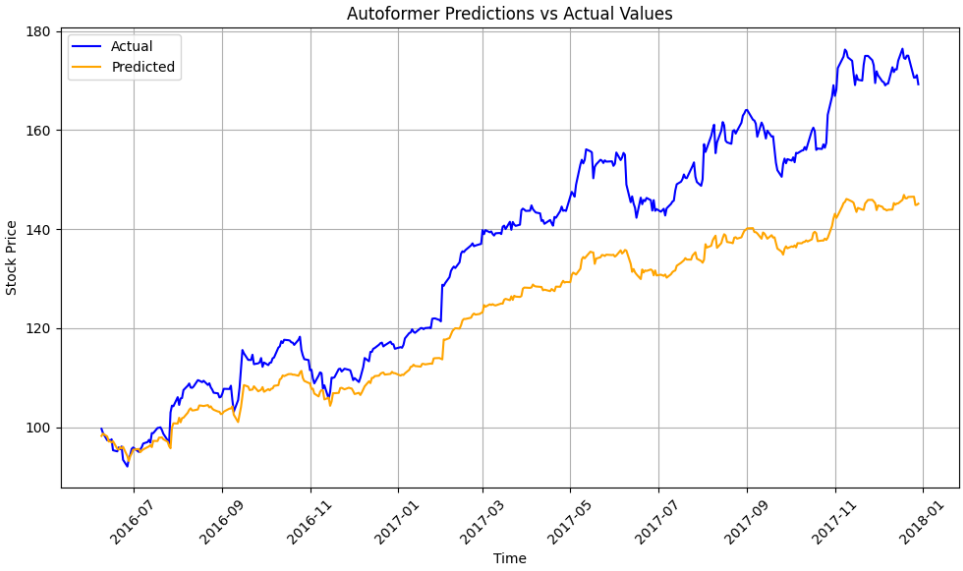


Figure 18: Autoformer predictions and actual values

can enhance the robustness and generalizability of predictions. Together, these steps can address overfitting, improve trend tracking, and better handle the inherent noise and complexity of stock price data.

6.2 Complexity Analysis For Multivariate modeling in LSTM

Using multivariate data introduces several complexity points compared to univariate data. The input size grows significantly, as the model now processes multiple features (Close and Volume) instead of just one, doubling the data size per sequence and increasing the number of model parameters. This increased computational costs and training time. Additionally, the model captured not only temporal dependencies within each feature but also interactions between features, such as the negative correlation between Close and Volume, which is critical for accurate predictions.

However, this added complexity comes with challenges like a higher risk of overfitting, especially with smaller datasets. Careful feature selection, such as focusing on Close and Volume to reduce redundancy, alongside regularization techniques, can mitigate this. Preprocessing also became more demanding; features with different ranges like Volume required normalization, and redundant features, like High and Low, were removed to avoid collinearity and ensure the model learns meaningful patterns efficiently.

6.3 Multivariate complexity analysis in Autoformer

In the implementation of the Autoformer model, a crucial step involved decomposing the time series data into its seasonal and trend components. This decomposition aimed to isolate the recurring patterns and underlying long-term movements in the data, allowing the model to focus on capturing these distinct dynamics separately. However, this decomposition process introduced certain

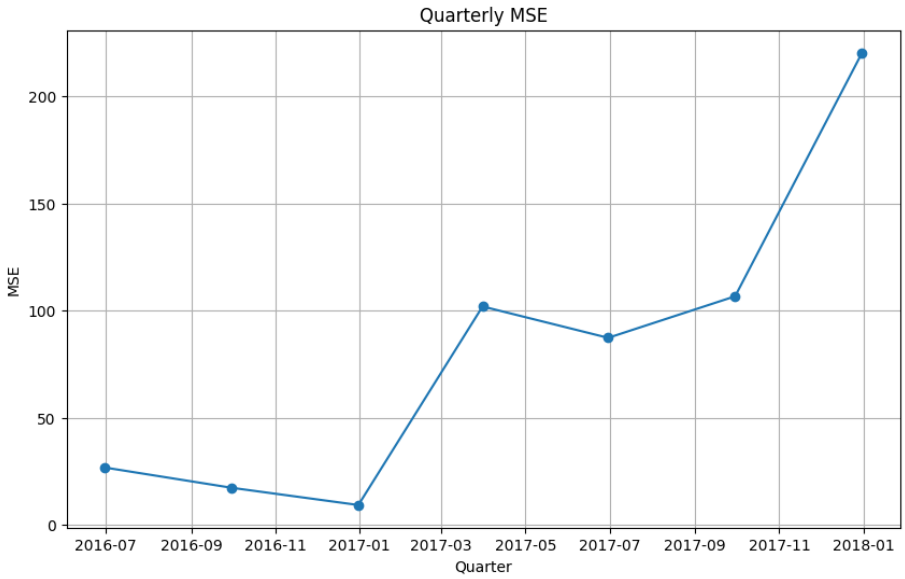


Figure 19: A localizedQuarterly MSE: Independent 3-Month Windows

complexities when it comes to multivariate.

Time series data often exhibit non-stationarity, meaning that its statistical properties change over time. Decomposition methods may assume stationarity, requiring preprocessing steps like differencing or transformation to stabilize the data before decomposition. This adds complexity to the data preparation process and becomes worse when dealing with more variables.

Selecting an appropriate decomposition method is crucial, as different methods have varying assumptions and may be better suited for specific types of data and seasonality patterns. However, the choice of method based on different variables with different trends can influence the accuracy of the decomposition and subsequently impact the model's performance.

6.4 Suggestions for improvement and plan for further parameter tuning

Based on the observed behavior of the Autoformer model, particularly the increasing mean squared error (MSE) trend over longer cumulative periods, several strategies for improvement and parameter tuning can be proposed to enhance its performance.

Incorporating Multivariate Data: Expanding the model to handle multivariate data can improve predictive accuracy by providing a holistic view of market dynamics. This necessitates careful feature engineering, feature selection, and preprocessing to ensure the data is consistent, relevant, and suitable for the model.

Hyperparameter Tuning: Fine-tuning key hyperparameters—such as the number of layers, attention heads, or learning rate—can profoundly affect performance. Employing optimization strategies like grid search or Bayesian optimization may identify optimal configurations tailored to specific datasets and forecast horizons.

Regularization Techniques: To enhance model generalization and prevent overfitting, regular-

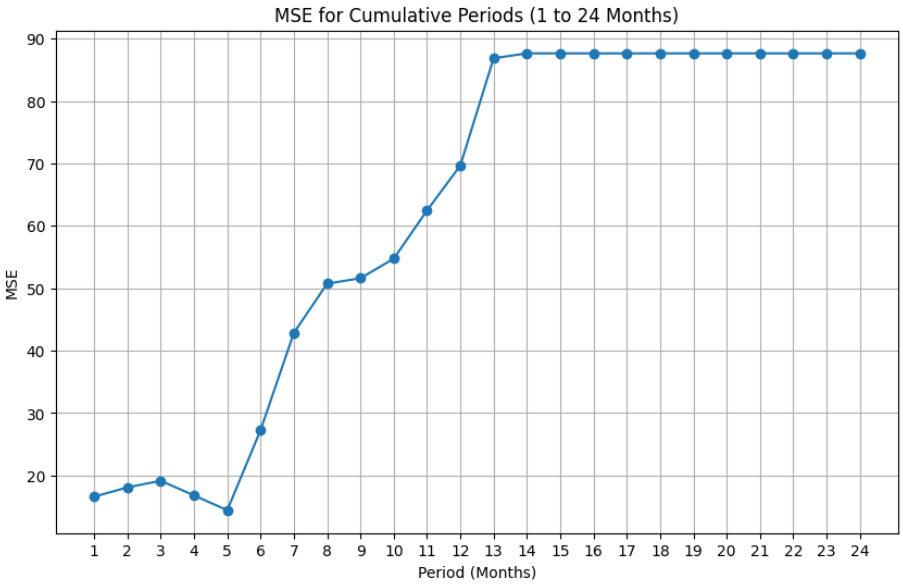


Figure 20: Evaluating Predictive Power Across Expanding Time Windows

ization methods such as dropout or weight decay can be applied. These approaches encourage the model to learn robust patterns rather than memorizing the training data, thereby improving its performance on unseen data.

7 Conclusion

In conclusion, enhancing time series models like RNN, LSTM, and Autoformer requires a combination of advanced architectures, careful hyperparameter tuning, and robust preprocessing techniques. Transitioning from RNN to LSTM or Autoformer, integrating attention mechanisms, and applying regularization strategies can improve predictive accuracy and address long-term dependency challenges. In multivariate modeling, managing increased complexity through effective feature selection, normalization, and decomposition methods is essential to mitigate overfitting and computational costs. Ultimately, a balanced approach to architecture, data preprocessing, and parameter optimization is key to building accurate and generalizable predictive models.

References

[1] G.E.P. Box and G.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day, 1976.

[2] Richard A Davis, Pengfei Zang, and Tian Zheng. Sparse vector autoregressive modeling. *Journal of Computational and Graphical Statistics*, 25(4):1077–1096, 2016.

[3] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[5] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

[6] Xixi Li, Fotios Petropoulos, and Yanfei Kang. Improving forecasting by subsampling seasonal time series, 2021.

[7] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. M4 Competition.

[8] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2022.