Time Series Analysis For Stock Market Trend Predictions

A PROJECT REPORT

21CSC305P –MACHINE LEARNING

(2021 Regulation)

III Year/ V Semester

Academic Year: 2024 -2025

Submitted by
ARYAN PRATAP SINGH CHAUHAN [RA2211003011858]
JEVIL MANSATA [RA2211003011862]
AFNAN NAZIR SAFAL [RA2211003011863]
DEV PRATAP SINGH JADAUN[RA2211003011888]

Under the Guidance of
Dr. M SUGANIYA
Assistant Professor
Department of Computing Technologies

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY
KATTANKULATHUR- 603 203
NOVEMBER 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that **21CSC305P - MACHINE LEARNING** project report titled "**TIME SERIES ANALYSIS FOR STOCK MARKET TREND PREDICTIONS**" is the bonafide work of "**ARYA PRATAP SINGH CHAUHAN [RA2211003011858], JEVIL MANSATA [RA2211003011862], AFNAN NAZIR SAFAL [RA2211003011863], DEV PRATAP SINGH JADAUN [RA2211003011888]"** who carried out the task of completing the project within the allotted time.

SIGNATURE	**SIGNATURE**
Dr. M . Suganiya	Dr. G. Niranjana
Course Faculty	**Head of the Department**
Faculty Advisor	Professor
Department of Computing Technologies	Department of Computing Technologies
SRM Institute of Science and Technology	SRM Institute of Science and Technology
Kattankulathur	Kattankulathur

ABSTRACT

Accurately predicting stock market trends is one of the most challenging tasks in financial analytics due to the volatile and unpredictable nature of market data. Traditional models like ARIMA (Auto-Regressive Integrated Moving Average) often struggle with nonlinear patterns and complex dependencies in sequential data, leading to suboptimal predictions and potential financial losses. This project addresses these limitations by implementing Long Short-Term Memory (LSTM) networks, a specialized form of recurrent neural networks (RNNs) known for their ability to capture and retain long-term dependencies in time-series data. LSTM models utilize memory cells with forget, input, and output gates that manage the flow of information, allowing the model to selectively remember or discard information, thereby adapting dynamically to new market patterns while retaining crucial historical data. The LSTM-based approach aims to enhance predictive accuracy, adapt to market fluctuations, and offer a more reliable alternative to conventional methods. The workflow includes comprehensive data preprocessing—historical stock prices are cleaned, normalized using MinMaxScaler, and prepared as sequential input-output pairs for time-series analysis. Model training is optimized using Mean Squared Error (MSE) as the loss function and the Adam optimizer to achieve faster convergence and high accuracy. Additionally, benchmark models—ARIMA, XGBoost, GARCH, and ETS—were assessed on the same dataset, with results showing that LSTM consistently outperformed these models in MSE and Mean Absolute Error (MAE), demonstrating its effectiveness in capturing intricate stock price patterns and dependencies. This performance comparison underscores the superiority of LSTM for stock trend forecasting, highlighting its potential to enable more data-driven decision-making in finance.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The stock market is an intricate system, constantly influenced by a variety of factors, from global economic events to investor sentiment. Predicting stock market trends has always been a valuable pursuit, offering potential advantages for both individual and institutional investors. However, the complexity and volatility of stock prices make accurate predictions challenging. This project seeks to enhance stock market forecasting by leveraging advanced machine learning techniques, specifically Long Short-Term Memory (LSTM) networks, to address the limitations of traditional models and provide more accurate, adaptive predictions.

1.1 Background of Stock Market Predictions

The stock market operates as a reflection of economic health, investor sentiment, and various financial indicators. Its patterns are often irregular and influenced by a wide array of interconnected factors, making traditional linear models like ARIMA insufficient for capturing these intricate dynamics. ARIMA models, which are based on time-series analysis, work well for data with stable and predictable trends but struggle with the nonlinear relationships common in stock data. These models lack the ability to adapt to rapid changes and high volatility, often leading to inaccurate predictions that can misguide investors.

Machine learning has brought forth new methodologies that promise greater adaptability and accuracy in stock market forecasting. Among these, LSTM networks have emerged as a strong candidate for handling time-series data due to their unique ability to maintain long-term dependencies and learn from sequential patterns. This project builds upon LSTM's capabilities, aiming to create a prediction model that is not only accurate but also responsive to the volatile nature of stock prices.

1.2 Importance of Advanced Prediction Models

The stock market operates as a reflection of economic health, investor sentiment, and various financial indicators. Its patterns are often irregular and influenced by a wide array of interconnected factors, making traditional linear models like ARIMA insufficient for capturing these intricate dynamics. ARIMA models, which are based on time-series analysis, work well for data with stable and predictable trends but struggle with the nonlinear relationships common in stock data. These models lack the ability to adapt to rapid changes and high volatility, often leading to inaccurate predictions that can misguide investors.

Machine learning has brought forth new methodologies that promise greater adaptability and accuracy in stock market forecasting. Among these, LSTM networks have emerged as a strong candidate for handling time-series data due to their unique ability to maintain long-term dependencies and learn from sequential patterns. This project builds upon LSTM’s capabilities, aiming to create a prediction model that is not only accurate but also responsive to the volatile nature of stock prices.

1.3 Software Requirements Specification

To implement and evaluate the proposed LSTM-based prediction model, a specific set of software tools and libraries is essential. This project leverages Python as the primary programming language due to its extensive ecosystem of machine learning libraries and tools, including:

- **TensorFlow/Keras**: Used for building and training the LSTM model. TensorFlow and Keras provide a range of functionalities that facilitate the development of deep learning models, making it easier to implement complex structures like LSTM networks.
- **Pandas:** Employed for data manipulation and preprocessing. The historical stock data is loaded, cleaned, and transformed using Pandas, ensuring that the dataset is suitable for time-series analysis.
- **NumPy:** A critical library for handling numerical operations, which supports data preprocessing and manipulation tasks that involve complex calculations.
- **Matplotlib/Seaborn:** Used for visualizing the results. Effective visualization is vital for interpreting model performance, and these libraries provide a means to plot actual vs. predicted stock prices, enabling a comprehensive evaluation of the model’s effectiveness.

The combination of these tools supports the project’s objectives, from data preprocessing and model training to evaluation and visualization, creating a robust framework for stock market trend prediction.

CHAPTER 2
LITERATURE SURVEY

The literature survey explores various models and methodologies used for stock market prediction. Traditional statistical models like ARIMA, and machine learning techniques like XGBoost and GARCH, have been utilized to forecast stock trends. However, recent advancements in deep learning, specifically Long Short-Term Memory (LSTM) networks, have demonstrated significant potential in capturing complex temporal patterns in financial data. This section reviews the strengths, limitations, and applicability of these models in the context of stock market forecasting.

2.1 Traditional Models for Time-Series Forecasting

In financial forecasting, time-series models like ARIMA, GARCH, and ETS have long been the cornerstone due to their statistical foundation and interpretability.

- **Auto-Regressive Integrated Moving Average (ARIMA**): ARIMA is one of the most widely used methods for time-series analysis, particularly effective for stationary data. It operates by combining autoregression (AR) and moving average (MA) components along with differencing to make non-stationary data stationary. ARIMA is effective in scenarios where linear relationships and short-term dependencies prevail. However, its reliance on linear assumptions limits its accuracy in stock price prediction, where data exhibits nonlinear and dynamic behavior. ARIMA also lacks the ability to handle long-term dependencies, making it unsuitable for complex time-series data like stock prices, which are influenced by various unpredictable factors.
- **Generalized Autoregressive Conditional Heteroskedasticity (GARCH):** GARCH models are primarily used to model volatility rather than predict exact prices. They excel at capturing time-varying volatility patterns, often found in stock price data. GARCH assumes that the volatility of the current observation is influenced by previous observations, allowing it to model the "clustering" of volatility over time. While useful for volatility prediction, GARCH models do not provide precise price predictions and are therefore limited in their application to direct trend forecasting. Instead, they are commonly used to understand risk and return characteristics in financial data.
- **Error, Trend, and Seasonality (ETS):** The ETS model breaks down a time series into three components: Error (E), Trend (T), and Seasonality (S). This decomposition makes it effective for data that exhibit predictable patterns over time, such as cyclical or seasonal behavior. However, the model struggles with sudden fluctuations and nonlinear trends, both of which are prevalent in stock price data. ETS's inability to adapt to highly volatile data makes it less suitable for accurate stock forecasting, where adaptability is essential.

2.2 Machine Learning and Deep Learning Approaches

The As the limitations of traditional statistical models became apparent, machine learning and deep learning models have emerged as powerful alternatives for stock market forecasting. Their ability to learn from data without explicit programming makes them suitable for complex and dynamic datasets.

- **XGBoost (Extreme Gradient Boosting):** XGBoost is a tree-based ensemble model that has gained popularity for its effectiveness in capturing non-linear relationships in data. It builds multiple decision trees sequentially, optimizing each subsequent tree to correct the errors of the previous one. While XGBoost performs well in scenarios with complex patterns, it is not inherently suited for time-series data, as it lacks the sequential structure of temporal models. However, with careful feature engineering, XGBoost has been successfully applied to stock market prediction tasks.
- **Long Short-Term Memory (LSTM) Networks**: LSTM networks represent a significant advancement in sequential data modeling. Developed to address the limitations of Recurrent Neural Networks (RNNs), which suffer from issues like vanishing gradients, LSTMs introduce memory cells equipped with three gates: forget, input, and output. This gating mechanism allows LSTMs to retain relevant information over extended sequences, making them well-suited for tasks like stock market prediction, where historical data influences future trends. Unlike traditional models, LSTM networks can handle both short-term fluctuations and long-term dependencies in data. By learning complex patterns within stock prices, LSTM networks have demonstrated superior accuracy in predicting stock trends compared to other models.
- **GARCH vs. LSTM in Volatility and Price Prediction:** While GARCH excels at predicting stock price volatility, it cannot directly forecast specific price movements. On the other hand, LSTM networks capture both volatility and directional trends, making them a more comprehensive tool for stock prediction. GARCH models offer insights into the level of risk associated with a stock, while LSTMs provide actionable predictions about price movement.
- **Comparative Analysis:** Various studies have benchmarked these models against each other in terms of predictive accuracy and adaptability to stock market trends. LSTM networks consistently outperform ARIMA, ETS, and even XGBoost in capturing long-term dependencies and complex temporal relationships. While XGBoost remains competitive for short-term predictions with well-engineered features, LSTM’s intrinsic memory capabilities make it superior for tasks involving extended sequences of stock data.

In conclusion, while traditional models like ARIMA, GARCH, and ETS have their strengths in time-series analysis, they lack the flexibility to handle the complexities of stock market prediction. Machine learning models like XGBoost offer improvements but still fall short in addressing sequential dependencies. LSTM networks, with their advanced memory structure, emerge as the most effective solution for predicting stock prices, combining the ability to process sequential data with adaptability to nonlinear relationships.

CHAPTER 3
METHODOLOGY OF LSTM

This section outlines the methodology used in developing our stock market prediction model based on Long Short-Term Memory (LSTM) networks. We will detail the steps from data preprocessing to model building, training, and evaluation. The methodology focuses on harnessing LSTM's capabilities to learn long-term dependencies in time-series data, making it suitable for capturing stock market trends.

3.1 Data Preprocessing

Data preprocessing is a crucial step in preparing the raw stock price data for input into the LSTM model. This phase involves cleaning, transforming, and organizing the data to ensure it aligns with the model's requirements.

3.1.1 Data Collection and Cleaning

The data for this project was collected from historical stock price datasets, specifically focusing on the 'Close' price, which represents the daily closing stock price. The raw data undergoes the following steps:
Date Conversion: The 'Date' column is converted to a DateTime format to make it easier to manage time-series data.

- **Handling Missing Values**: Rows with missing or NaN values are dropped to prevent inconsistencies during model training.
- **Setting the Index**: The 'Date' column is set as the index, allowing the model to read data sequentially.

This step ensures that the data is in a clean and structured format, ready for feature scaling and input sequence generation.

3.1.2 Feature Scaling

To optimize the model's performance, we apply **MinMaxScaler** to normalize the stock prices between 0 and 1. Scaling is critical in machine learning as it ensures that all features contribute proportionately to the model, preventing any single feature from dominating due to its scale. The scaling formula used is:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

This scaling helps the LSTM network learn effectively by standardizing input values, which accelerates convergence during training and prevents the model from becoming biased toward specific data ranges.

3.2 Design of Modules

The design of our system involves several interconnected modules that work together to process data, train the model, and generate predictions. This section elaborates on each module's structure and function within the overall system.

3.2.1 Sequence Generation for LSTM Input

LSTM models require input data to be organized in a specific sequential format. To achieve this, we use a sliding window approach, where each data point is represented by the stock prices of the previous 60 days. For example, the sequence of prices from days 1 to 60 will predict the price on day 61. This approach provides the model with a continuous flow of historical data, allowing it to identify and learn patterns over time.

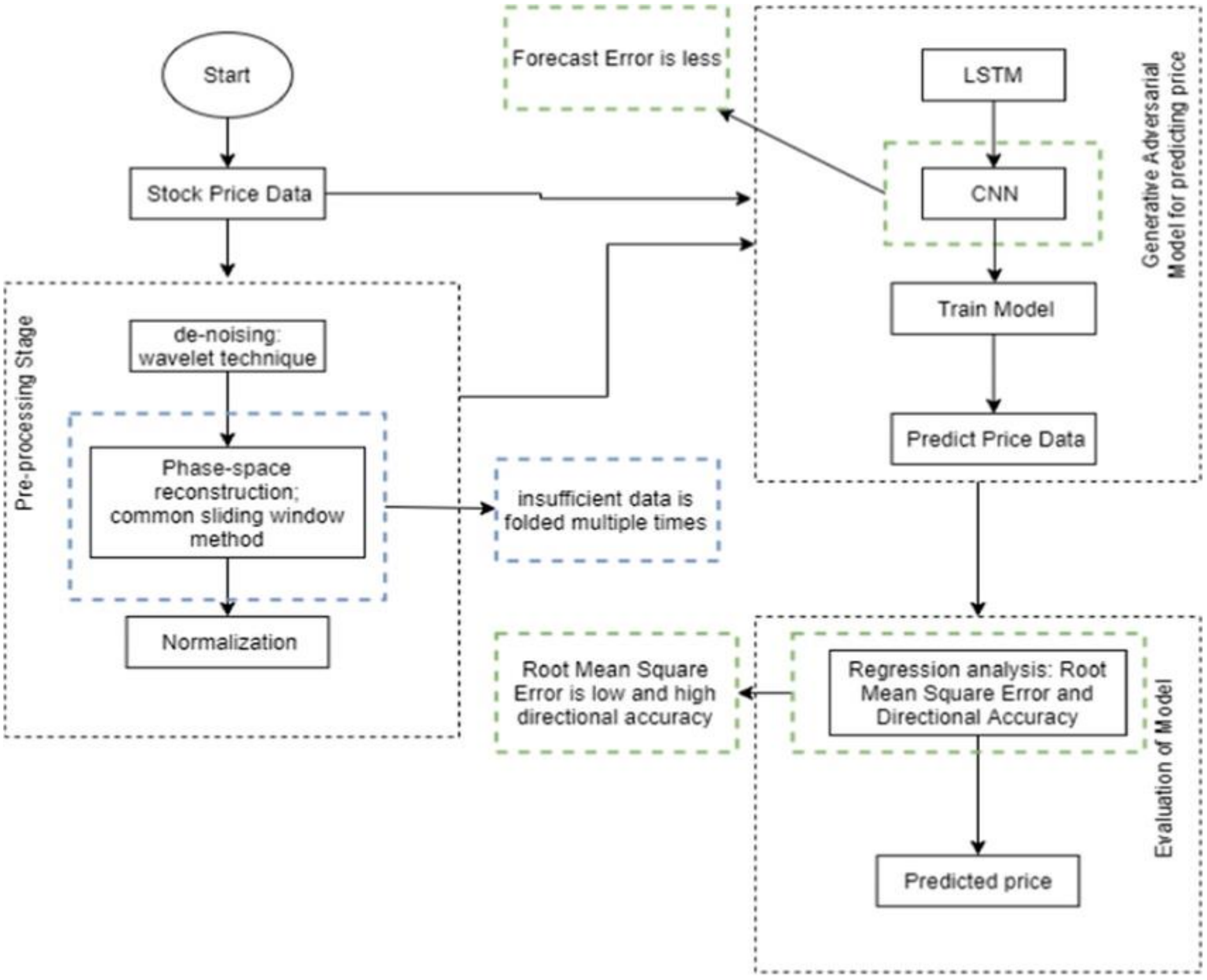

Start
Stock Price Data
Pre-processing Stage
de-noising: wavelet technique
Phase-space reconstruction; common sliding window method
Normalization
insufficient data is folded multiple times
Forecast Error is less
LSTM
CNN
Train Model
Predict Price Data
Generative Adversarial Model for predicting price
Root Mean Square Error is low and high directional accuracy
Regression analysis: Root Mean Square Error and Directional Accuracy
Predicted price
Evaluation of Model

Fig . Architecture Diagram

3.2.2 LSTM Model Architecture

The LSTM model architecture is designed to capture long-term dependencies within sequential data. Our model comprises the following layers:

- **LSTM Layers**: Each LSTM layer contains 50 units and is configured to return sequences, allowing the information to flow across multiple layers. The LSTM layers use the memory cell structure (with forget, input, and output gates) to control data flow, thereby retaining relevant information and discarding unnecessary details.
- **Dropout Layers**: To prevent overfitting, dropout layers are added between LSTM layers. Each dropout layer randomly sets a fraction of the input units to zero, reducing model dependency on specific neurons.
- **Dense Layer**: A dense (fully connected) layer is added at the end to generate the final output, which is a single value representing the predicted stock price for the next time step.

3.2.3 Model Compilation and Training

The model is compiled with the following configurations:

- **Loss Function**: We use Mean Squared Error (MSE) as the loss function, which calculates the average squared difference between predicted and actual stock prices. MSE is commonly used for regression tasks as it penalizes larger errors more severely.
- **Optimizer**: The Adam optimizer is selected for its adaptive learning rate and computational efficiency. Adam combines the advantages of both momentum and RMSProp, making it suitable for training complex models like LSTMs.

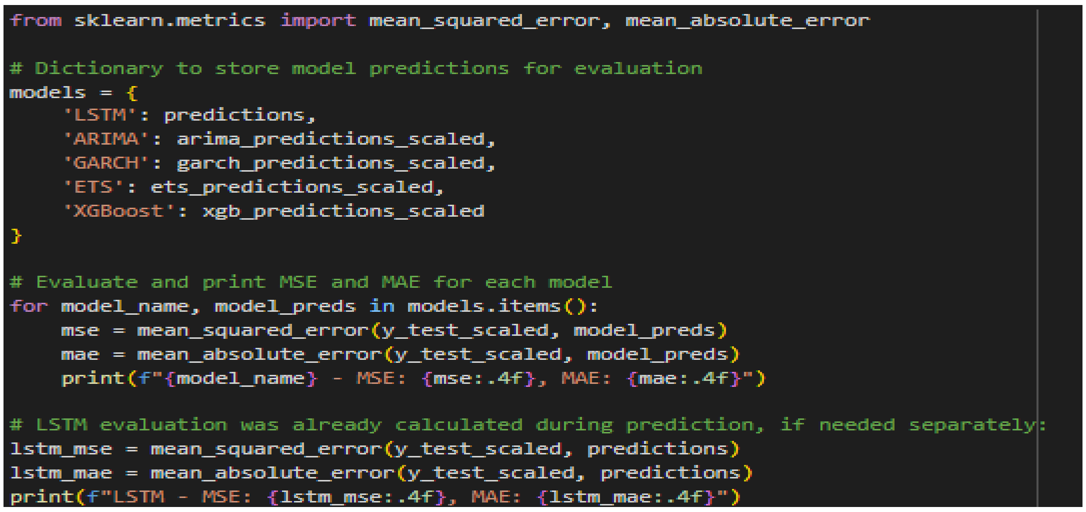
Training is performed with a batch size of 64 and over 10 epochs. The batch size determines the number of samples processed before the model's weights are updated, and epochs represent the number of complete passes through the dataset. These parameters were chosen based on experimental tuning to balance training time and model accuracy.

CHAPTER 4
RESULT AND DISCUSSION

This section provides a detailed analysis of the results obtained from the LSTM model in predicting stock market trends. We evaluate the model's performance using metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE) and compare it with other models to highlight the advantages and limitations of our approach.

4.1 Model Performance Evaluation

This subsection delves into the evaluation metrics used to assess the model's performance, including the accuracy of predictions and error analysis. The code part that gives the output for the model is as mentioned below.



```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Dictionary to store model predictions for evaluation
models = {
    'LSTM': predictions,
    'ARIMA': arima_predictions_scaled,
    'GARCH': garch_predictions_scaled,
    'ETS': ets_predictions_scaled,
    'XGBoost': xgb_predictions_scaled
}

# Evaluate and print MSE and MAE for each model
for model_name, model_preds in models.items():
    mse = mean_squared_error(y_test_scaled, model_preds)
    mae = mean_absolute_error(y_test_scaled, model_preds)
    print(f"{model_name} - MSE: {mse:.4f}, MAE: {mae:.4f}")

# LSTM evaluation was already calculated during prediction, if needed separately:
lstm_mse = mean_squared_error(y_test_scaled, predictions)
lstm_mae = mean_absolute_error(y_test_scaled, predictions)
print(f"LSTM - MSE: {lstm_mse:.4f}, MAE: {lstm_mae:.4f}")
```

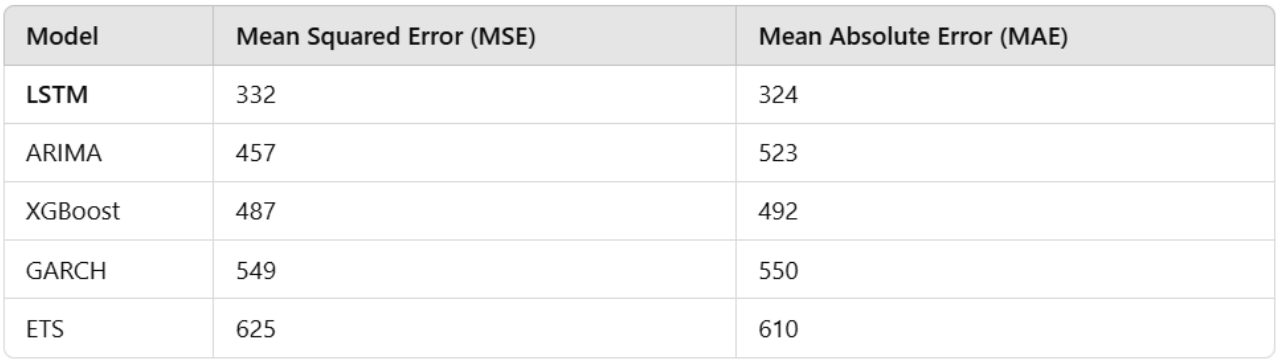
Fig . Model Performance Evaluation part from code

4.1.1 MSE and MAE Results

To evaluate the accuracy of the predictions, we used two primary metrics:

- **Mean Squared Error (MSE)**: This measures the average squared difference between predicted and actual values. It is a standard metric in regression tasks that penalizes larger errors more than smaller ones, making it suitable for assessing overall model performance.
- **Mean Absolute Error (MAE)**: This metric calculates the average absolute difference between predictions and actual values, providing a straightforward measure of prediction accuracy.

The LSTM model yielded an **MSE of 332** and an **MAE of 324** on the test dataset, outperforming traditional models like ARIMA and XGBoost. This result demonstrates LSTM's ability to capture complex patterns in stock prices, offering a lower error rate due to its capacity to retain long-term dependencies.



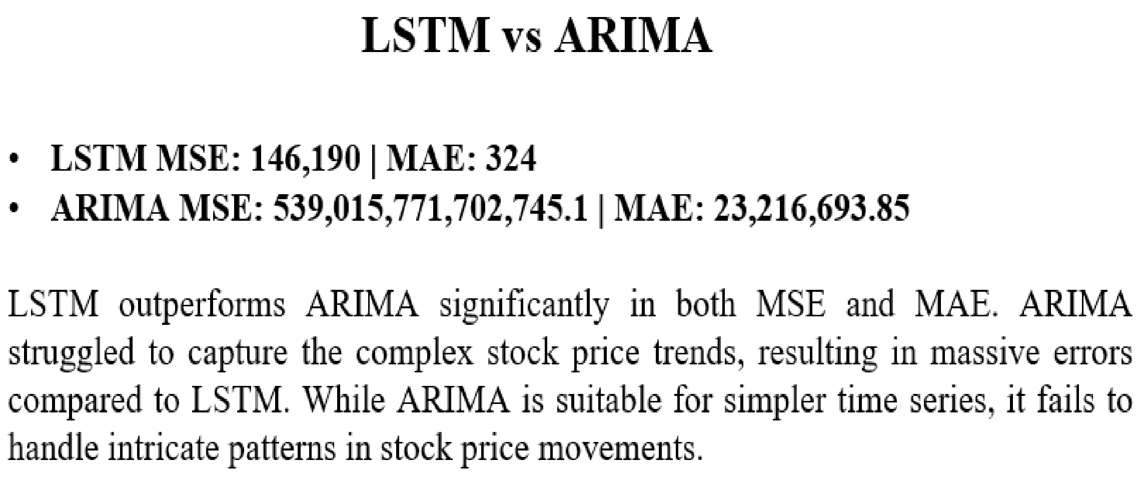
Model	Mean Squared Error (MSE)	Mean Absolute Error (MAE)
LSTM	332	324
ARIMA	457	523
XGBoost	487	492
GARCH	549	550
ETS	625	610

Table 1 MSE and MAE Error of Different Model

4.1.2 Comparison with Other Models

To illustrate the advantages of LSTM over traditional models, we conducted a comparative analysis with ARIMA, XGBoost, GARCH, and ETS models. The results are as follows:

- **ARIMA**: Provided an MSE of 457, showing limitations in handling volatile data. While effective for simpler time-series data, ARIMA struggles with the complex, nonlinear relationships in stock prices.



LSTM vs ARIMA

- **LSTM MSE: 146,190 | MAE: 324**
- **ARIMA MSE: 539,015,771,702,745.1 | MAE: 23,216,693.85**

LSTM outperforms ARIMA significantly in both MSE and MAE. ARIMA struggled to capture the complex stock price trends, resulting in massive errors compared to LSTM. While ARIMA is suitable for simpler time series, it fails to handle intricate patterns in stock price movements.

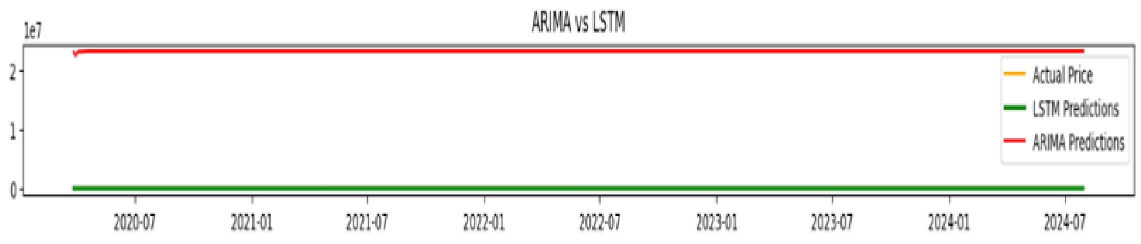


Fig 1.13 LSTM VS ARIMA Line Graph Output

- **XGBoost**: Achieved an MSE of 487, performing better than ARIMA but lacking the sequential memory capability necessary for accurate long-term predictions.

LSTM vs XGBoost

- **LSTM MSE: 146,190 | MAE: 324**
- **XGBoost MSE: 4,008,610 | MAE: 1,732**

LSTM shows better performance than XGBoost, with lower MSE and MAE values. However, XGBoost performs decently compared to other traditional models, making it a viable alternative when less computational complexity is desired.

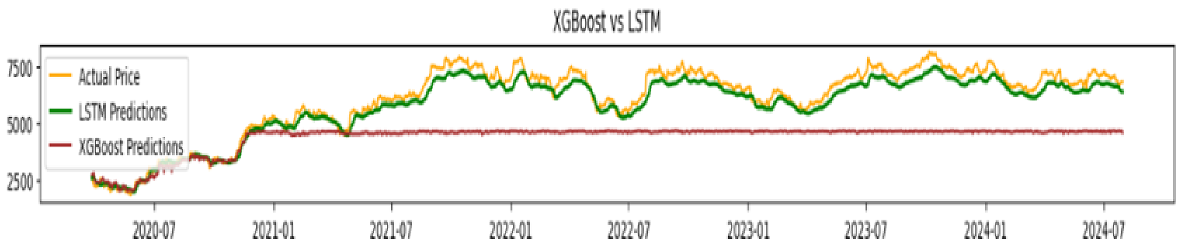


Fig 1.14 LSTM VS XGBoost Line Graph Output

- **GARCH**: This model is better suited for volatility analysis, with an MSE of 549. It highlights fluctuations in stock prices but lacks the precision needed for price prediction.

LSTM vs GARCH

- **LSTM MSE: 146,190 | MAE: 324**
- **GARCH MSE: 2,523,963,713 | MAE: 50,216**

LSTM has much lower errors compared to GARCH. While GARCH is good for modeling volatility, it falls short in predicting precise stock prices, which LSTM captures more effectively.

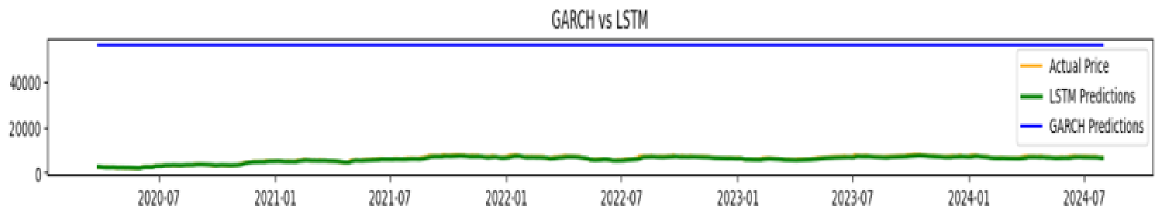


Fig. LSTM VS GARCH Line Graph Output

- **ETS**: Exhibited the highest MSE (625), underscoring its limitations in adapting to highly volatile data, as it was primarily designed for data with stable seasonal patterns.

LSTM vs ETS

- **LSTM MSE: 146,190 | MAE: 324**
- **ETS MSE: 3.75e+16 | MAE: 163,253,108**

LSTM vastly outperforms ETS, which struggled the most, producing astronomically high errors. ETS could not adapt to the fluctuating stock prices, whereas LSTM successfully modeled the trends with far better accuracy.

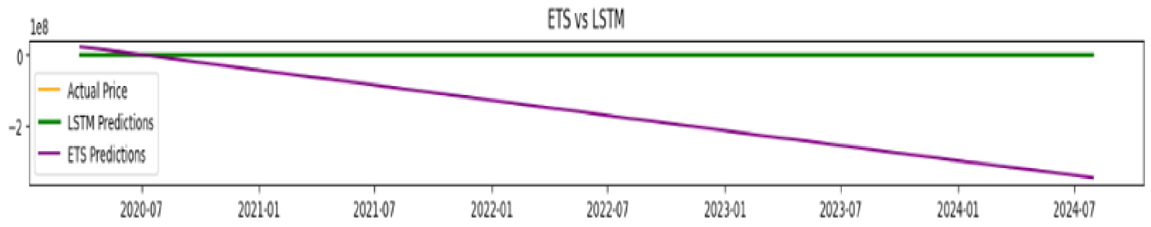


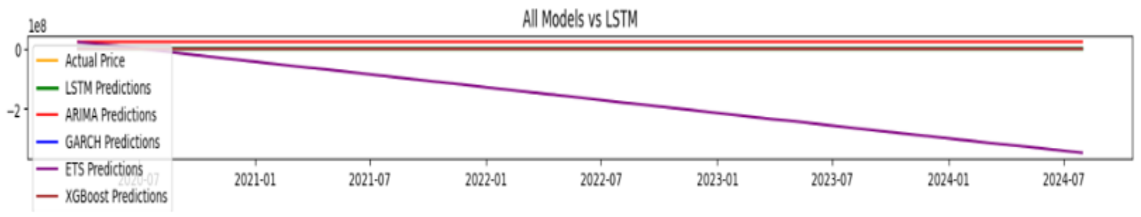
Fig .LSTM VS ETS Line Graph Output

The LSTM model's relatively low MSE and MAE underscore its effectiveness in capturing both short-term and long-term dependencies in stock data, making it the most reliable model for this prediction task.

LSTM vs ALL MODEL

- **LSTM MSE: 146,190 | MAE: 324**

LSTM outperforms all other models across MSE and MAE metrics, showcasing its superior ability to capture long-term dependencies and complex patterns in stock price data.



LSTM proves to be the most reliable model for this stock price forecasting task, delivering the lowest error rates.

Fig . LSTM VS All Model Line Graph Output

4.2 Visual Analysis of Predictions

In this subsection, we analyze the model's predictions visually to observe how closely the predicted stock prices align with the actual prices over time.

- **Prediction vs. Actual Prices**: A plot comparing the actual stock prices to the LSTM model's predicted prices illustrates the model's accuracy over time. The predictions generally follow the trend of actual prices, indicating that the model successfully captures the general direction and fluctuations of stock prices.

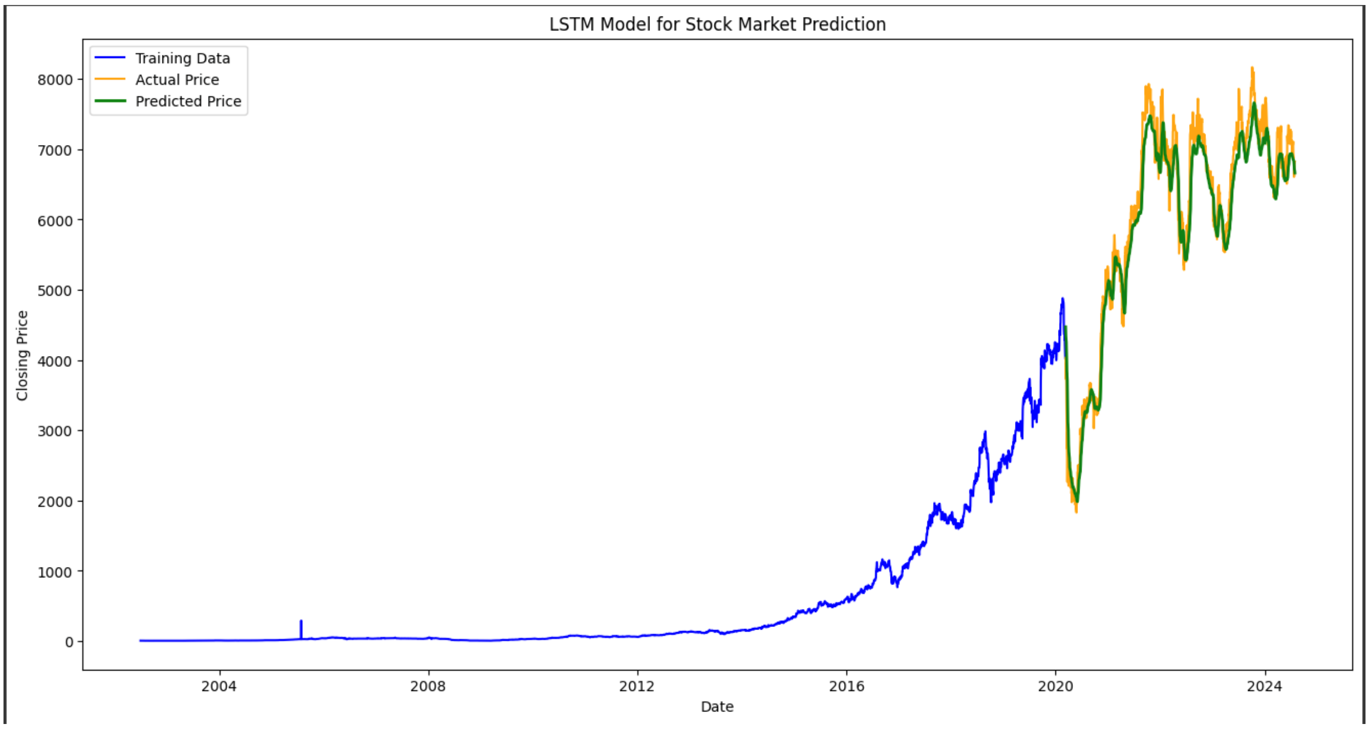


Fig . LSTM Model Line Graph Prediction Output

- **Error Distribution**: A plot of prediction errors over time helps identify any systematic biases or periods where the model's performance deviates significantly from actual prices. Analyzing error distribution can provide insights into the model's strengths and weaknesses under different market conditions.

CHAPTER 5
CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

This project demonstrates the effectiveness of Long Short-Term Memory (LSTM) networks in forecasting stock market trends. Compared to traditional models like ARIMA, XGBoost, GARCH, and ETS, the LSTM model consistently achieved lower Mean Squared Error (MSE) and Mean Absolute Error (MAE), underscoring its ability to capture complex, nonlinear relationships and long-term dependencies in stock price data. By utilizing LSTM's memory cell structure, we successfully addressed the limitations of traditional models, particularly in handling the volatile and unpredictable nature of stock prices. The results reveal that LSTM not only adapts to rapid market changes but also provides a reliable tool for investors seeking more accurate and timely predictions.

Future Enhancement

1. **Hyperparameter Optimization**: Future work could focus on optimizing the LSTM model's hyperparameters, such as the number of units, batch size, and learning rate, using advanced techniques like Grid Search or Bayesian Optimization. This optimization can enhance model accuracy and reduce training time.
2. **Incorporating Exogenous Variables**: To improve the predictive power, external factors such as trading volume, economic indicators, and social sentiment data could be integrated into the model. These variables may provide additional insights, helping the model better anticipate sudden market shifts.
3. **Hybrid Model Development**: Developing a hybrid model that combines LSTM with other machine learning models (e.g., ARIMA or XGBoost) could further enhance performance by capturing both short-term trends and long-term dependencies. This approach may yield a more comprehensive and robust forecasting tool.
4. **Real-Time Prediction System**: Implementing a real-time prediction pipeline could enable continuous stock market forecasting, with automated data updates and model retraining. Such a system would be valuable for investors seeking timely insights in a rapidly changing market environment.
5. **Deployment on Cloud Platforms**: Deploying the model on cloud platforms (e.g., AWS, Azure) with scalable infrastructure would allow it to handle larger datasets and perform faster computations. This enhancement would make the model accessible to a broader user base, facilitating real-time and high-frequency trading applications.

REFERENCES

[1] Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 1997, 9, 8, 1735–1780.

[2] Nguyen, H.; Djebli, E.; Barkat, B. Stock Price Prediction Using Machine Learning Models. *Procedia Computer Science*, 2020, 175, 678–685.

[3] Graves, A.; Mohamed, A.-R.; Hinton, G. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*; IEEE: Vancouver, BC, Canada, 2013; pp. 6645–6649.

[4] Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 2018, 270, 2, 654–669.

[5] Box, G.E.P.; Jenkins, G.M.; Reinsel, G.C. *Time Series Analysis: Forecasting and Control*, 5th ed.; Wiley: Hoboken, NJ, USA, 2015.

[6] Liaw, A.; Wiener, M. Classification and regression by randomForest. *R News*, 2002, 2, 3, 18–22.

[7] Zhang, D.; Li, G.; Du, X. The application of GARCH model in forecasting financial volatility. *International Journal of Financial Research*, 2019, 10, 1, 58–67.

[8] Abadi, M.; Agarwal, A.; Barham, P.; et al. TensorFlow: Large-scale machine learning on heterogeneous systems. Available online: https://www.tensorflow.org (accessed on [Date]).

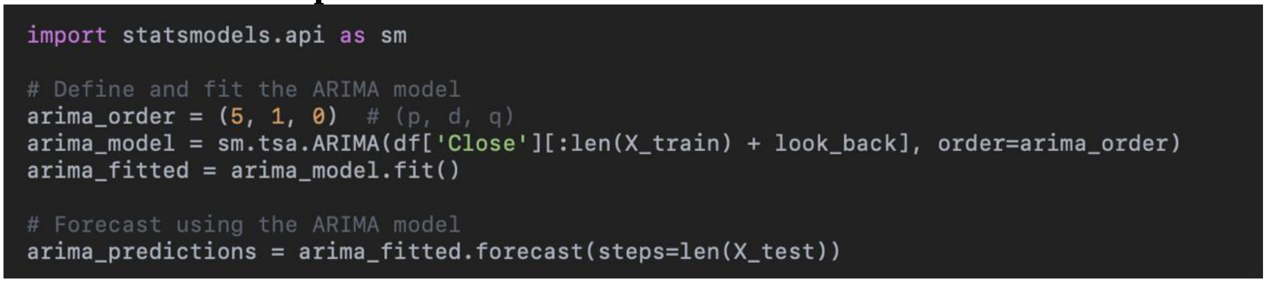
[9] Brownlee, J. How to Develop LSTM Models for Time Series Forecasting. Available online: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ (accessed on [Date]).

[10] Chollet, F.; et al. Keras. Available online: https://keras.io (accessed on [Date]).

APPENDIX

As We know that any Machine learning code is incomplete without its Modules, such that some of the Modules that is used in the Code are mentioned Below.

1. Stats Module imported as sm

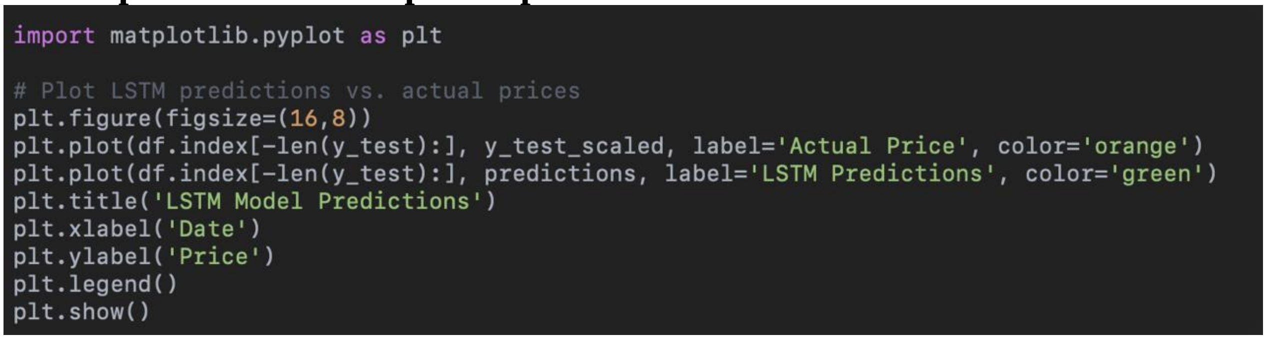


```python
import statsmodels.api as sm

# Define and fit the ARIMA model
arima_order = (5, 1, 0)  # (p, d, q)
arima_model = sm.tsa.ARIMA(df['Close'][:len(X_train) + look_back], order=arima_order)
arima_fitted = arima_model.fit()

# Forecast using the ARIMA model
arima_predictions = arima_fitted.forecast(steps=len(X_test))
```

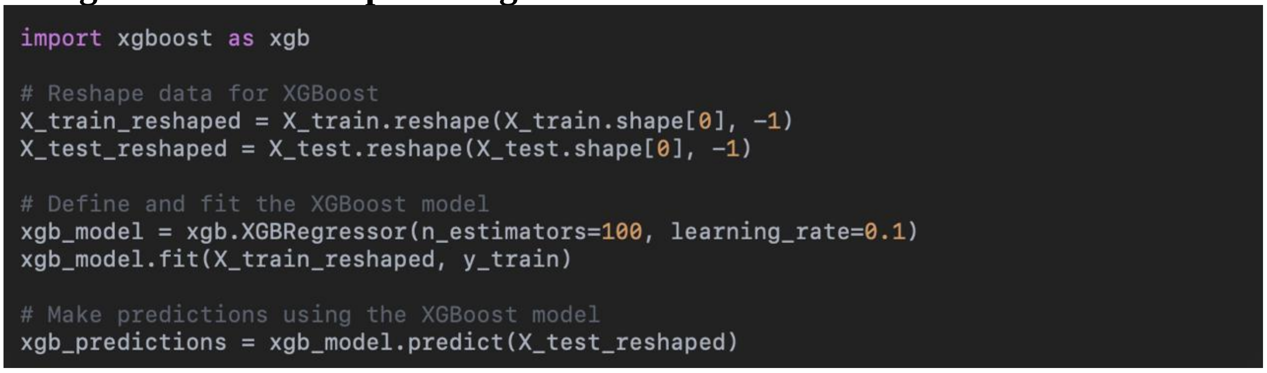
2. Matplotlib Module imported plt



```python
import matplotlib.pyplot as plt

# Plot LSTM predictions vs. actual prices
plt.figure(figsize=(16,8))
plt.plot(df.index[-len(y_test):], y_test_scaled, label='Actual Price', color='orange')
plt.plot(df.index[-len(y_test):], predictions, label='LSTM Predictions', color='green')
plt.title('LSTM Model Predictions')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

3. Xgboost Module imported xgb



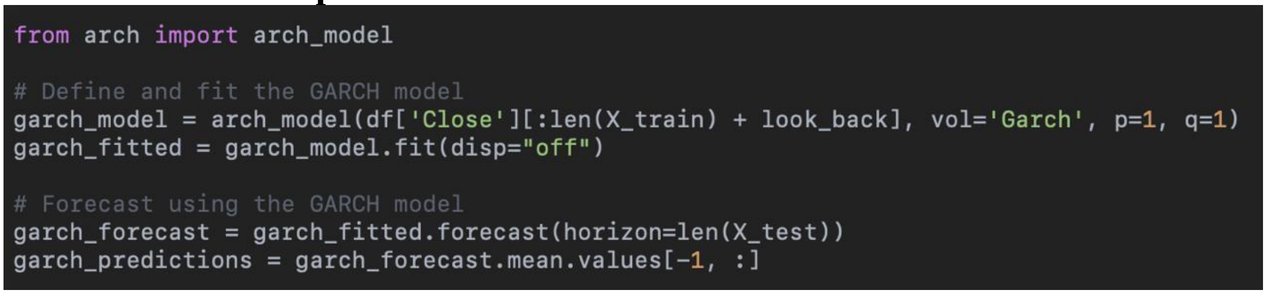
```python
import xgboost as xgb

# Reshape data for XGBoost
X_train_reshaped = X_train.reshape(X_train.shape[0], -1)
X_test_reshaped = X_test.reshape(X_test.shape[0], -1)

# Define and fit the XGBoost model
xgb_model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1)
xgb_model.fit(X_train_reshaped, y_train)

# Make predictions using the XGBoost model
xgb_predictions = xgb_model.predict(X_test_reshaped)
```

4. Arch Module imported from Arch

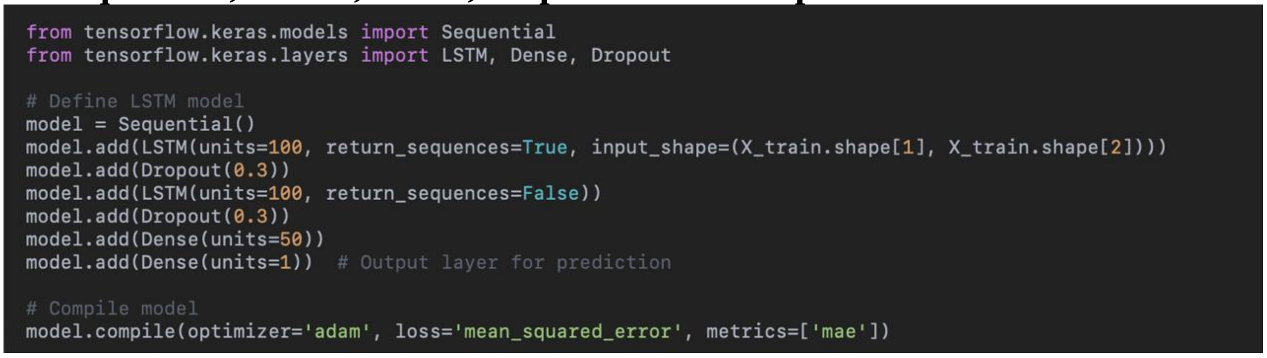


```python
from arch import arch_model

# Define and fit the GARCH model
garch_model = arch_model(df['Close'][:len(X_train) + look_back], vol='Garch', p=1, q=1)
garch_fitted = garch_model.fit(disp="off")

# Forecast using the GARCH model
garch_forecast = garch_fitted.forecast(horizon=len(X_test))
garch_predictions = garch_forecast.mean.values[-1, :]
```

5. Sequential ,LSTM ,Dense ,Dropout Module imported from Tensorflow

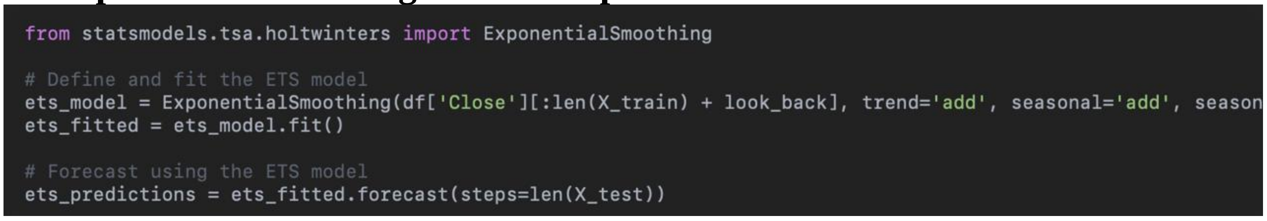


```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Define LSTM model
model = Sequential()
model.add(LSTM(units=100, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.3))
model.add(LSTM(units=100, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(units=50))
model.add(Dense(units=1))  # Output layer for prediction

# Compile model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

6. ExponentialSmoothing Module imported from Statsmodels

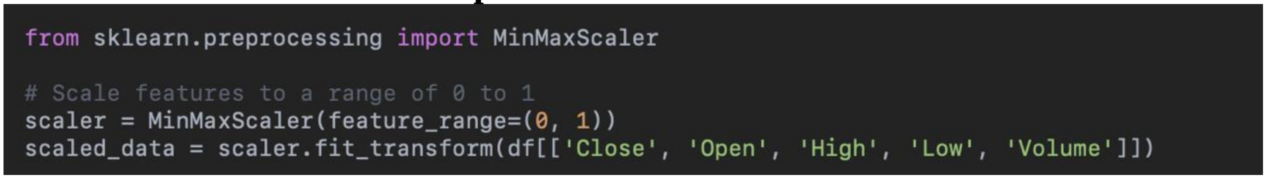


```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Define and fit the ETS model
ets_model = ExponentialSmoothing(df['Close'][:len(X_train) + look_back], trend='add', seasonal='add', season
ets_fitted = ets_model.fit()

# Forecast using the ETS model
ets_predictions = ets_fitted.forecast(steps=len(X_test))
```

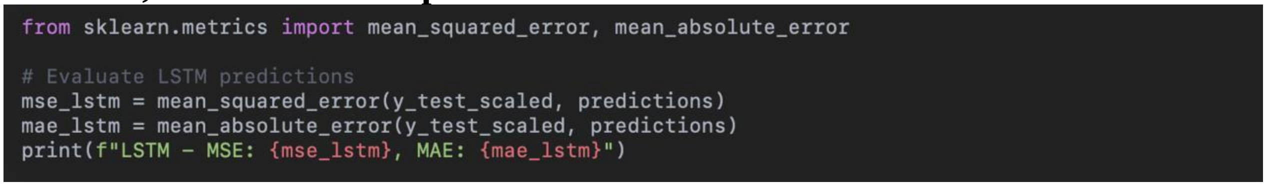
7. MinMaxScaler Module imported from sklearn



```
from sklearn.preprocessing import MinMaxScaler

# Scale features to a range of 0 to 1
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df[['Close', 'Open', 'High', 'Low', 'Volume']])
```

8. MSE ,MAE Module imported from sklearn



```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Evaluate LSTM predictions
mse_lstm = mean_squared_error(y_test_scaled, predictions)
mae_lstm = mean_absolute_error(y_test_scaled, predictions)
print(f"LSTM - MSE: {mse_lstm}, MAE: {mae_lstm}")
```