Predicting S&P500 ETF Closing Prices Using Historical Data and Evaluating It with The Mean Absolute Percentage Error

A. Matoug, J. Peeters, Ş. Saygılı

[[1]](#footnote-2)

***Abstract*—** **This work explores the use of Long Short-Term Memory (LSTM) networks to predict the closing prices of the S&P 500 ETF based on historical data. The dataset, comprising key price features such as opening, high, and low prices, was pre-processed to ensure temporal integrity and normalized for effective training. The LSTM model was implemented in PyTorch, with its architecture optimized through hyperparameter tuning, and trained using the Mean Squared Error (MSE) loss function. The evaluation, performed on unseen test data, used the Mean Absolute Percentage Error (MAPE) to measure predictive accuracy. Results indicate that the LSTM model successfully captures market trends, achieving low MAPE even in volatile conditions, though slight biases in predictions were observed. The study highlights the potential of deep learning models in financial forecasting and suggests future improvements, including the integration of macroeconomic indicators and additional technical features, to enhance model performance and adaptability to market dynamics.**

***Index Terms*—** **Closing price prediction, deep learning, financial forecasting, Long Short-Term Memory (LSTM), mean absolute percentage error (MAPE), PyTorch, S&P 500 ETF, time series analysis, volatility modelling**

# I. INTRODUCTION

T

he primary objective of this task is to predict the closing price of the S&P 500 ETF (Exchange-Traded Fund) for a specified number of days based on historical data.

This paper, which constitutes a report on the task performed, will first provide a comprehensive explanation of the S&P 500 ETF. Next, the report will discuss the algorithm that will be used for forecasting (and how it will be evaluated, why this specific algorithm is chosen, …), and will include a comparison with the state of the art for such problems. Thereafter, the results will be presented and discussed in detail. Finally, a conclusion will be drawn as to whether the choice of algorithm was appropriate and what could be improved for forecasting purposes.

## S&P500 ETF (Exchange-Traded Fund)

Because they provide diversified exposure to a variety of assets in a single, easily transferable instrument, exchange-traded funds, or ETFs, have changed the investing environment. Among these, the S&P 500 ETF is one of the most prominent, designed to track the performance of the S&P 500 Index, a benchmark comprising five hundred leading publicly traded companies in the United States. The S&P 500 Index is a theoretical construct that serves as a capitalization-weighted measure of the top U.S. corporations, selected based on criteria including market value, liquidity, and sector representation [1].

ETFs that track the S&P 500 Index, like the SPDR S&P 500 ETF (ticker: SPY), offer investors a way to participate in this benchmark, even though the index itself is used as a standard to evaluate the performance of the U.S. equities market. ETFs are tradable securities that mimic the index by owning the same or a representative sample of the underlying stocks, in contrast to the index, which is only a calculation [2]. Furthermore, the performance of ETFs may differ slightly from that of the index due to the introduction of real-world factors including tracking error, expense ratios, and liquidity [3].

The S&P 500 ETF's significance as a benchmark extends beyond its theoretical and practical distinctions, serving as a cornerstone of financial markets. Its capitalization-weighted design ensures that larger companies exert greater influence on the index’s performance, offering a snapshot of the U.S. economy’s largest players [4]. This dual role—as a benchmark and as an investable product—makes the S&P 500 ETF a vital tool for investors worldwide.

Historical data analysis of the S&P 500 ETF provides insights into market behaviour. By studying patterns of returns, volatility, and drawdowns, investors can refine their strategies and enhance decision-making [5]. For example, while the index data reflects theoretical market performance, ETF data incorporates the impact of trading costs, dividend distributions, and reinvestment policies, offering a more comprehensive picture of investable returns [6]. Such analyses are central to modern portfolio theory, enabling investors to optimize risk-adjusted returns through informed decisions.   
  
To sum up, the S&P 500 Index serves as a theoretical standard for the American stock market, and exchange-traded funds (ETFs) that track it offer a practical way to invest. Investors can close the gap between market performance and actual portfolio management by examining both the theoretical index and the actual ETF. This dual viewpoint emphasises how important the S&P 500 ETF is to risk management and contemporary financial markets. So, exploring algorithms for predicting this index would be of great value to investors.

## Challenges of Predicting the Closing Price

Several factors, such as the non-stationary character of financial time series, the existence of seasonality, and the unpredictability of markets, make it difficult to forecast closing prices in financial datasets. Since statistical characteristics like mean and variance can fluctuate over time, financial data is usually non-stationary, making it challenging to find consistent patterns for forecasting [7]. Furthermore, the modelling process is made more difficult by outside variables like market sentiment, macroeconomic events, and geopolitical developments, which frequently generate abrupt and unforeseen fluctuations [8].

Seasonality, which reflects repeating patterns or cycles over specific intervals, also poses a challenge. While some financial instruments exhibit clear seasonal trends—such as increased volatility around earnings reports or fiscal year-end activities—such patterns are often masked by noise in daily or intraday data [9]. In the case of the S&P 500 ETF, the presence of both short-term volatility and long-term macroeconomic influences makes it difficult to disentangle meaningful signals from noise.   
  
Additionally, predicting the closing price for the next month amplifies the difficulty since the model must generalize well to unseen data. Unlike retrospective analyses, where historical data is fully available, future data introduces uncertainty that models must approximate based on historical patterns. This task is particularly demanding in financial contexts, where the assumption of past performance being indicative of future behaviour is often invalidated by structural breaks or market shifts [10].

The combination of non-stationarity, and market randomness requires some considerations for the model design and feature selection. For example, models must balance between overfitting historical trends and capturing patterns that hold across different timeframes. These challenges highlight the importance of error metrics, such as mean absolute percentage error (MAPE), which account for variations in data scale and enable evaluation of predictive performance [11].

In conclusion, predicting the closing price for the following month is a difficult task impacted by the unstable and ever-changing nature of financial markets. Precise modelling techniques and a well-informed modelling of features are necessary to address these issues.

## Role of Historical Price Features

In this section, the given features for this project will be discussed.

The dataset provided for this project includes four features:

* date,
* opening price,
* high price,
* low price.

These collectively offer insights into the dynamics of daily market behaviour and play a role in predicting the closing price of the S&P 500 ETF.

First, the date feature, which not directly influences the closing price, can be important for identifying temporal trends and seasonal patterns. Financial markets often exhibit cyclical behaviours, with certain times of the year, month, or week displaying distinct trends. For example, the so-called "end-of-month effect" or quarterly earnings cycles may impact investor behaviour, indirectly influencing price movements [9]. Of course, relative to the other features, the date will realistically play a minor role in this project.

Secondly, the opening price, which represents the first transaction price of the day, setting the baseline for the day’s trading activity. The price is influenced by pre-market trading and after-hours market movements. It is often indicative of investor sentiment and external factors such as news or economic developments occurring prior to the market opening. In trending markets, the opening price often aligns with the closing price. Consequently, the opening price serves as a reference point for subsequent intraday fluctuations [10].

Then there is the high price and low price of the day. The high price and low price indicate the highest and lowest points of trading during the day, respectively, offering a measure of the day’s volatility. These features are helpful in capturing the variety of market swings, which might indicate uncertainty and mood in the market. For example, a wide difference between the high and low prices might be an indicator of high market volatility, which is sometimes caused by news or investor responses to fresh data [12]. These characteristics are necessary for modelling since high volatility days are frequently linked to a closing price that differs from the beginning price. But of course, these assumptions will have to be detected by the model (programmed in Python), so going further into this is beyond the scope.

When combined, these features offer a basis for closing price prediction. Even while each parameter has a distinct importance, it's possible that their correlations with the closing price are not linear, requiring other modelling techniques to accurately capture dependencies, instead of simple linear regression models, and interactions. Building a precise predictive model requires an understanding of how the opening price establishes the day's baseline, how the high and low represent volatility, and how temporal patterns affect trading behaviours. The exact method and which modelling technique was used is discussed in the next section.

## Time-Series Analysis and Regression in Financial Forecasting

The modelling and estimation of financial measures, including the closing price of an asset, is important because they are temporary and can sometimes be quite erratic. There are two main approaches that can be used to analyse such data: time series analysis and regression modelling. Both offer advantages depending on the characteristics of the data and the forecasting objectives.

### Time-Series Analysis

The main objective of time series analysis is to analyse the temporal characteristics of the data and use historical information and sequences to make future projections. Some of the widely used methods include statistical models such as Autoregressive Integrated Moving Average (ARIMA) and more advanced techniques such as Long Short-Term Memory (LSTM) networks.

* ARIMA predicts data with the of help its previous values and previous forecast errors. It performs particularly well where there is high autocorrelation or trends in the data and can be used to forecast future values [13]. However, ARIMA model has difficulties with non-linearity and cannot incorporate other variables that affect the system, which limits its use in analysing complex financial data.
* LSTM is a type of Recurrent Neural Network (RNN) that has been developed to deal with sequences and particularly with long-term dependencies. Its capability of capturing complex correlations and temporal structure makes it appropriate for the financial forecasting [14]. They can capture various features in the data series, for instance trends, seasonal patterns and sudden changes which are common in financial markets.

### Regression-based models

Regression models provide an alternative approach, focusing on relationships between dependent and independent variables. Using features such as the opening, high, and low prices, regression models aim to establish direct correlations with the closing price.

* Linear regression is a simple, effective approach, but only (not always) when the relationships between features are linear. However, it may fail to capture non-linearities in financial datasets.
* Non-linear models, such as decision trees or support vector regression, address this limitation by modelling more complex relationships. These methods can improve accuracy but require careful tuning to avoid overfitting [15]. Also, polynomial regression is an option but it’s the same as described before. Regularization can help here but methods like this require some effort the finetune the parameters.

## Feature engineering and training

Feature engineering enhances model performance by converting raw data into useful inputs. Examples include daily price range (high minus low), percentage change from the opening price, and rolling averages. Temporal features, such as the day of the week or month, can also reveal patterns linked to seasonality or investor behaviour.

Model training uses a portion of historical data to fit parameters, while the rest is reserved for validation and testing to ensure generalization. Techniques like cross-validation, hyperparameter tuning, and regularization can also be applied to improve accuracy and avoid overfitting. Further details, whether these techniques will be applied or not, will be given in the next section.

Time-series analysis and regression are widely used in financial forecasting, each suited to different data characteristics. Effective feature engineering and training strategies help models capture market complexity and make reliable predictions. Exactly which method will be used in this project will become clear in the next section, as mentioned earlier.

# II. Method

The provided dataset for this task consists of four key features: the date, opening price, high price of the day, and low price of the day. This methodology emphasizes using the historical data for training, selecting the optimal model through grid search evaluated with a separate part of the training data, called evaluation set, and testing the model on unseen data without relying on the features of the test data, which are assumed to be unavailable.

## Data Preprocessing

Regardless of what model is used, the first step is to preprocess the raw data to ensure it is suitable for training and testing. Before any analysis can begin, the dataset must be inspected for any missing values or anomalies. Given that the stock market is closed on weekends and public holidays, there are reoccurring gaps in the data. The nature of the data is sequential, so it’s important that it is ordered chronologically, from old to new. This is critical for maintaining the time series structure and ensuring that the model only uses past data to predict future values. The task involves predicting future closing prices based on historical patterns, therefore the data was transformed into fixed-length sequences. Instead of predicting a day’s closing price, the system generates input sequences of past closing prices, which allows the model to learn patterns from historical data. All numerical features were normalized to a range of 0 to 1 using the MinMaxScaler. This normalization prevents features with larger magnitude from dominating the learning process. To mimic real-world forecasting scenarios where future data is completely unavailable, the historical data was split into a training set, consisting of historical data up to September 2024, and a validation set consisting of only the last month, October 2024. This validation set was used for hyperparameter tuning. The test data was reserved exclusively for final evaluation of the prediction and was not used during the entire training and validation process. This is important to prevent data leakage, which would mean that any information from the test set influences the training process, leading to overly optimistic evaluation results that don’t reflect the model’s true performance on unseen data [].

## Model Selection

A Long Short-Term Memory (LSTM) model was selected for its ability to effectively capture temporal dependencies in sequential data, making it ideal for time-series forecasting tasks. The LSTM model was implemented using PyTorch and its architecture is configured as follows:

* The Input Layer, which accepts a 3D tensor of shape (batch size, sequence length and number of features).
* The Hidden Layers, which are stacked LSTM layers that use learnable parameters to capture dependencies in the data.
* The Output layer, which maps the final hidden state from the last LSTM layer to a scalar value, representing the predicted target for the next timestep.

Fig. 1 shows the principal architecture of the model.

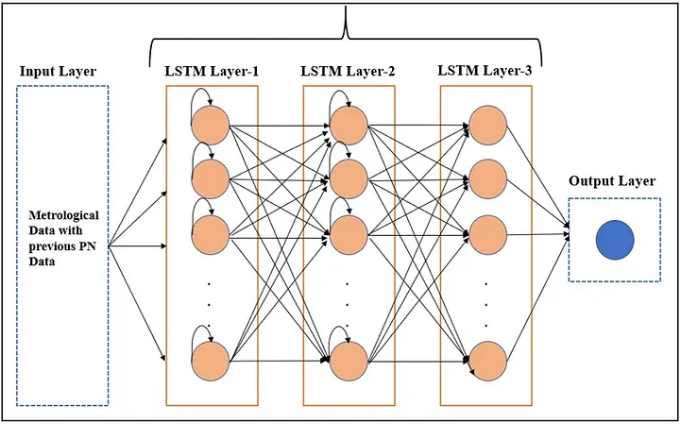


Fig. 1: LSTM architecture [17]

This architecture allows for easy adjustments based on task requirements, making it a very flexible model. The number of LSTM layers, the size of the hidden states, and the activation functions can all be adjusted to optimize performance for specific datasets. Deeper networks with higher hidden dimensions are used for more complex patterns, while simpler architectures are used for smoother or less variable series [18].

During training, the model processes data in mini batches, the hidden and cell states are reset to zero at the start of each batch to ensure independence between sequences and avoid information leakage across batches. This approach is consistent with truncated backpropagation through time (TBPTT), where backpropagation is limited to the sequence length within a batch [19]. By resetting the states for every batch, we ensure that the model only focuses on the dependencies within the current batch.

The architecture leverages PyTorch’s GPU acceleration to scale efficiently for larger datasets, this ensures that the training remains computationally feasible for average users' systems.

## Training process

The model is trained using the Mean Squared Error (MSE) loss function. The Adam optimizer is used for training, it adapts the learning rate during training and works faster than the traditional stochastic gradient descent. The training data is fed into the model in mini batches. The batch size is another hyperparameter that can be adjusted during training. This is done automatically using the DataLoader class from PyTorch. The model saves the best-performing weights, determined by the lowest validation loss. The training process is stopped when there is no improvement in validation loss over a predefined number of epochs, ensuring that the model does not overfit, and still generalizes well to unseen data.

To find the optimal hyperparameters for the model, a grid search is performed. Hyperparameters such as the number of LSTM layers, hidden layer size, learning rate, batch size, number of epochs, and the sequence length are explored. The grid search tests all combinations of these parameters and selects the configuration that results in the lowest Mean Absolute Percentage Error (MAPE) on the validation set.

## Model Evaluation

After training, the model is evaluated using unseen test data. The test data, which has not been used in training or validation, is pre-processed similarly to the training data. The trained model is used to generate predictions on this unseen test set. The model makes multi-step forecasts, where the output of a prediction is used as input for the next. The predictions are compared to the actual values (found online) to assess the model’s performance. This performance is evaluated using the MAPE, which is commonly used for regression tasks, providing a percentage error between the predicted and actual values. Those actual versus predicted values are plotted over time, providing a visual representation of model prediction accuracy. This allows for a more intuitive understanding of the model’s forecasting ability.

## Model Deployment and Prediction

Once the best model is selected and trained, it is saved in a .pth file for future use. The saved model includes the optimal weights, the model configuration, and the model architecture. Additionally, the scaler used for normalizing the training data is also saved within the same .pth file. This ensures that the model and the scaling procedure are tightly coupled, and the same scaling transformations are applied to new data as were applied during training. The saved model can be used for making predictions on new, unseen test data. This provides a lot of flexibility as the trained model is a file that can transferred across different environments or machines, enabling predictions to be made without requiring the original training data or model retraining. The saved model can also be reloaded and used for additional predictions as needed, without the need to repeat the entire training process. Moreover, there is also the option to retrain the model with new data or with more parameters whenever desired. This flexibility is especially useful in dynamic environments where the data might evolve over time. By simply retraining the model, it can be adapted to account for these changes, improving its predicting performance.

# III. Results

Before diving into the results, it is important to note a key distinction about how this model was trained and evaluated. Unlike typical machine learning models, which are trained to predict the target variable (in this case, the closing price) based on features available in the test set, our model follows a different approach. The S&P 500 ETF’s future data is not available at the time of predicting, which means that the model does not have access to the features of the test data. This would give unrealistically low errors, such as MAPE of approximately 1% because the model would be predicting based on future information.

Instead, the model is trained using only the historical data available up to the prediction point, ensuring that it is unable to “cheat”. The test data is predicted purely based on the historical data, making the evaluation much more challenging. Depending on how different or unpredictable the market was during the range of time covered by the test data, compared to the training data, a Mean Absolute Percentage Error between 5% and 20% could be expected. Financial markets are inherently volatile and subject to external factors such as economic events, geopolitical developments, … Therefore, the model’s prediction accuracy can vary significantly based on the stability and predictability of the market during the test period.

This section presents the results of the model’s performance on predicting the closing prices of the S&P 500 ETF. The training and validation loss over epochs are examined to evaluate the model’s learning process and convergence. The predictions of the LSTM model are compared against the actual values to assess accuracy. Finally, the residuals (prediction errors) are analysed to identify any systematic patterns or biases in the model’s predictions.

## Training and validation loss

Fig. 2 shows a graph which plots the loss against the epochs.

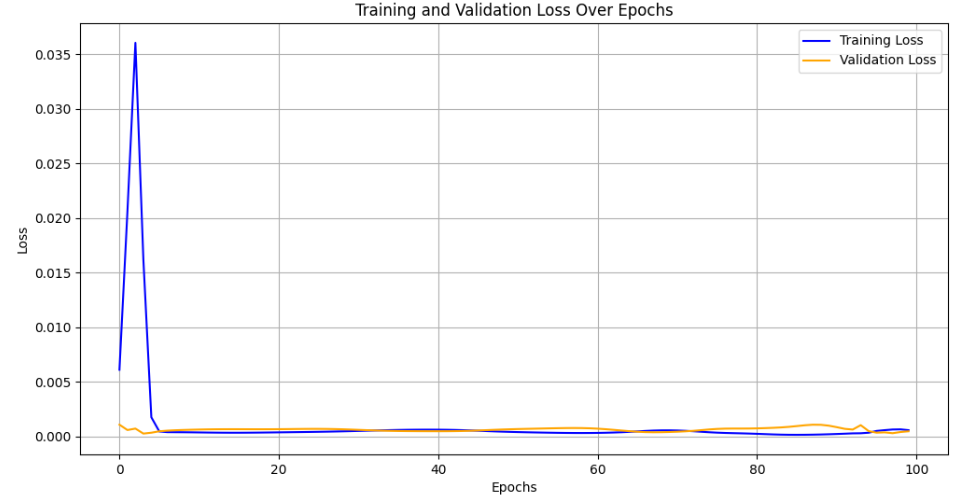


Fig. 2: Training and validation loss over epochs

The training and validation loss curves show that during the initial epochs, there is a sharp rise in training loss, which is expected as the model begins adjusting its weights. However, both losses stabilize quickly, and the validation loss remains close to the training loss throughout the rest of the process. This suggests that the model generalizes well and does not overfit the training data. Slight oscillations in the validation loss occur, but the training process demonstrates that the chosen hyperparameters allow the model to converge effectively.

## Prediction vs Actual values

The main goal of this assignment was to predict the closing price of the S&P 500 ETF. To evaluate the effectiveness of the model, we compare the predicted closing prices generated by the LSTM model with the actual closing prices from the test set.

Fig. 3 shows the comparison between the model’s predictions and the actual closing prices for the month November.

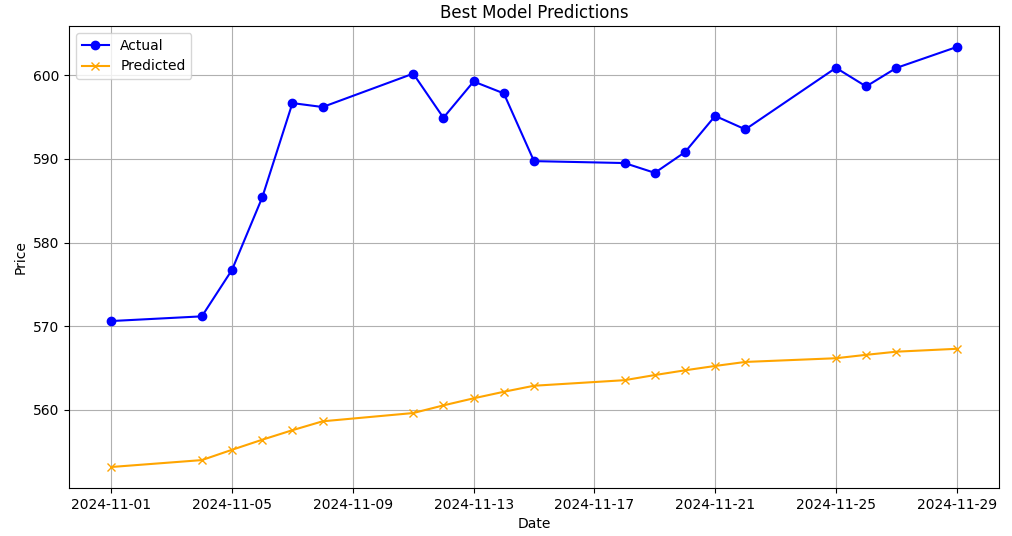


Fig. 3: Predicted vs actual values of November 2024

As seen in the figure above, the prediction for November 2024 has an upward trend, a bit lower than the actual values of that month. This is expected because the model is trained on past data starting in 2018, and its parameters are tuned on the validation set consisting October 2024. Both have lower closing prices than November and follow an upward trend over extended periods.

The Mean Absolute Percentage Error (MAPE) for November 2024 is 5.11%, which is very low given the circumstances. The reason for this low MAPE is that we use October, the month before, as validation data to tune the parameters, this is obviously going to be in a comparable price range as November. The second reason is because the test month follows a similar trend (slightly upward) as the historical data.

Next, Fig. 4 shows the comparison between the model’s predictions for the month December.

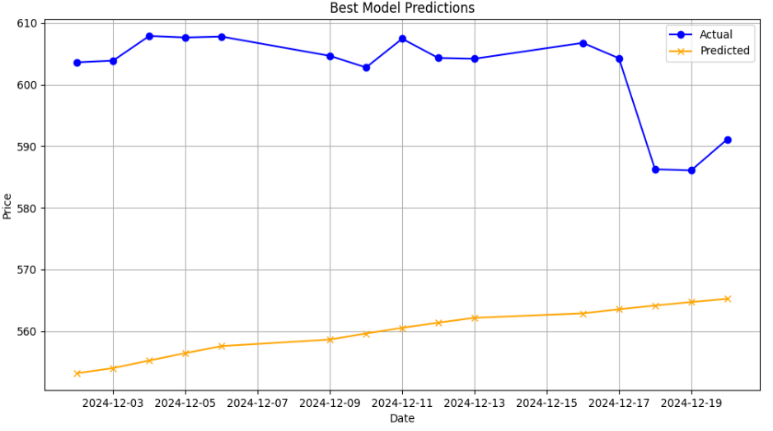


Fig. 4: Predicted vs actual values of December 2024

In the figure, the prediction for December 2024, shows an extremely similar trend as the prediction for November 2024. This is normal, because the predictions are made using the same model as used for November, so same training data and same validation data.

The Mean Absolute Percentage Error for December 2024 is 6.95%, which is a little bit higher than for November. This is simply because the December closing prices are more distant from our training data which ends in October. If we used November 2024 in our training data, for example as validation set, we would get a better prediction.

Fig. 5 shows the graph which plots the residuals for the month November.

As seen in figure 5, the residuals are consistently positive for November, indicating that the predicted values are systematically underestimating the actual values throughout the month. This systematic underestimation suggests a bias in the model predictions, which is due to the limited complexity of the model.

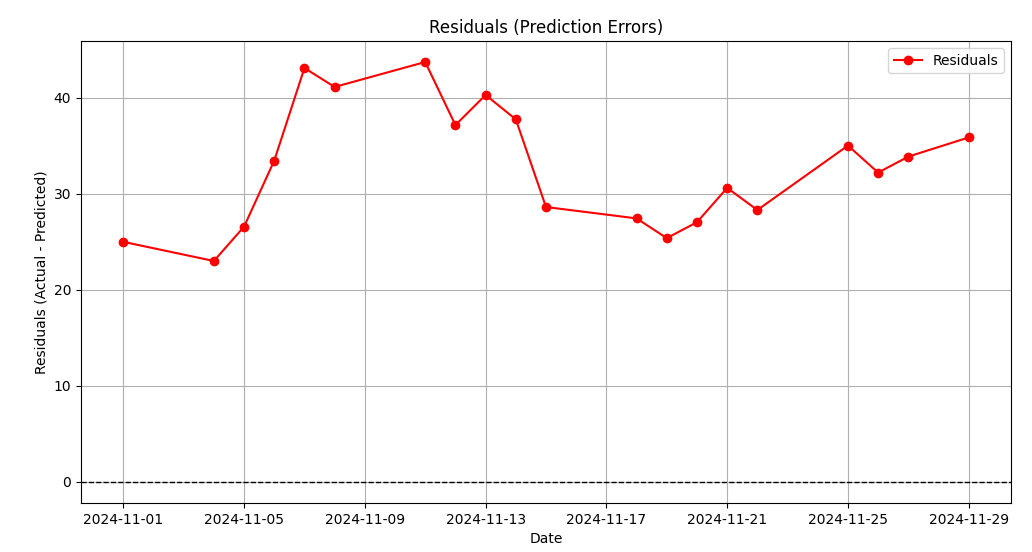


Fig. 5: Difference between actual and predicted values for November

# Conclusion

In conclusion, the LSTM model demonstrates solid performance in predicting the closing prices of the S&P 500 ETF, with low Mean Absolute Percentage Error (MAPE) in both November (5.11%) and December (6.95%) 2024. Given the volatile nature of financial markets, these results are quite good. The model was also able to capture the general trend of the market.

However, while the predictions show reasonable accuracy, the model could be further improved by incorporating more sophisticated features such as macroeconomic indicators and other advanced technical indicators. Additionally, the performance could be improved by using more historical data in the training process. This model also assumes that historical patterns will persist, which might not be the case in the future, which is why adding macroeconomic indicators can be valuable. Future work could explore other models, such as hybrid approaches that combine multiple machine learning algorithms, as well as deeper hyperparameter tuning and extra data sources to improve the accuracy of predictions [20]. But despite its limitations, this LSTM model is a promising tool for forecasting stock market trends.

# Citations

[1] “S&P U.S. Indices Methodology | S&P Dow Jones Indices.” Accessed: Dec. 24, 2024. [Online]. Available: https://www.spglobal.com/spdji/en/methodology/article/sp-us-indices-methodology/

[2] R. A. . Ferri, “The ETF Book : All You Need to Know About Exchange-Traded Funds,” p. 574, 2011, Accessed: Dec. 24, 2024. [Online]. Available: https://www.wiley.com/en-nl/The+ETF+Book%3A+All+You+Need+to+Know+About+Exchange-Traded+Funds%2C+Updated+Edition-p-9781118160770

[3] B. G. Malkiel, “The efficient market hypothesis and its,” *Journal of Economic Perspectives*, vol. 17, no. 1, pp. 59–82, Dec. 2003, doi: 10.1257/089533003321164958.

[4] E. F. Fama *et al.*, “The Cross-Section of Expected Stock Returns,” *The Journal of Finance*, vol. 47, no. 2, pp. 427–465, Jun. 1992, doi: 10.1111/J.1540-6261.1992.TB04398.X.

[5] J. Y. Campbell, A. W. Lo, and A. C. MacKinlay, “The econometrics of financial markets,” *The Econometrics of Financial Markets*, pp. 1–611, Jun. 2012, doi: 10.1017/S1365100598009092.

[6]

W. F. Sharpe, “The Sharpe Ratio,” *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, Oct. 1994, doi: 10.3905/JPM.1994.409501.

[7] B. G. Malkiel and E. F. Fama, “EFFICIENT CAPITAL MARKETS: A REVIEW OF THEORY AND EMPIRICAL WORK\*,” *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, May 1970, doi: 10.1111/J.1540-6261.1970.TB00518.X.

[8] R. Shiller, “Do Stock Prices Move Too Much to Be Justified by Subsequent Changes in Dividends?,” *American Economic Review*, vol. 71, pp. 421–436, Jan. 1981.

[9] C. Chatfield, “The Analysis of Time Series : An Introduction, Sixth Edition,” *The Analysis of Time Series*, Jul. 2003, doi: 10.4324/9780203491683.

[10] R. S. Tsay, “Analysis of financial time series,” *Analysis of Financial Time Series*, pp. 1–677, Jan. 2010, doi: 10.1002/9780470644560.

[11] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and Machine Learning forecasting methods: Concerns and ways forward,” *PLOS ONE*, vol. 13, no. 3, p. e0194889, Mar. 2018, doi: 10.1371/JOURNAL.PONE.0194889.

[12] JOHN. HULL, “PLP\_OPTIONS, FUTURES, AND OTHER DERIVATIVES (BOOK+PLP), GLOBAL EDITION,” 2020.

[13] M. Epaphra and M. Epaphra, “Modeling Exchange Rate Volatility: Application of the GARCH and EGARCH Models,” *Journal of Mathematical Finance*, vol. 7, no. 1, pp. 121–143, Dec. 2016, doi: 10.4236/JMF.2017.71007.

[14] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[15] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, “An Introduction to Statistical Learning,” 2023, doi: 10.1007/978-3-031-38747-0.

[16] M. A. Bouke, S. A. Zaid, A. Abdullah, and M. Aly Bouke, “Implications of Data Leakage in Machine Learning Preprocessing: A Multi-Domain Investigation,” Jul. 2024, doi: 10.21203/RS.3.RS-4579465/V1.

[17] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, “Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model,” *2nd Novel Intelligent and Leading Emerging Sciences Conference, NILES 2020*, pp. 87–92, Oct. 2020, doi: 10.1109/NILES50944.2020.9257950.

[18] S. Shen, H. Jiang, and T. Zhang, “Stock Market Forecasting Using Machine Learning Algorithms”.

[19] G. Sonkavde, D. S. Dharrao, A. M. Bongale, S. T. Deokate, D. Doreswamy, and S. K. Bhat, “Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications,” *International Journal of Financial Studies 2023, Vol. 11, Page 94*, vol. 11, no. 3, p. 94, Jul. 2023, doi: 10.3390/IJFS11030094.

[20] S. Siami-Namini, N. Tavakoli, and A. S. Namin, “The Performance of LSTM and BiLSTM in Forecasting Time Series,” *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, pp. 3285–3292, Dec. 2019, doi: 10.1109/BIGDATA47090.2019.9005997.

1. This work is supported by dr. Tsiogkas Nikolaos.

   A. Matoug, J. Peeters & S. Saygili are with Hasselt University/KU Leuven, Agoralaan, 3590 Diepenbeek, Belgium (emails: abdelmalek.matoug@student.uhasselt.be, joris.peeters@student.uhasselt.be, sukru.saygili@student.uhasselt.be) [↑](#footnote-ref-2)