

**2022-23**

**CMSE11475- Financial Machine Learning**

**Topic:** Forecasting Stock Volatility

**B214479**

**Word Count :**

1. **Abstract**

This report presents a comprehensive approach for forecasting stock volatility using 7 input features, with the target variable being volatility. The aim of this report is to address the questions posed in the Assessment and compare the results of three NASDAQ-listed stocks. The model has shown robust generalization performance for [INTC], an older listed stock with more training data. However, the convergence quality of predictions for the other two stocks, TSLA and MSFT, is comparatively lower.

1. **Introduction**

Forecasting stock volatility is a challenging task, and several studies have been conducted in this area. The current understanding and background information about the topic suggest that predicting stock volatility is critical for investors to make informed decisions about their investment portfolios.

The purpose of this project is to forecast stock volatility using 7 input features. To achieve this objective, several techniques and models have been employed. The report provides a detailed explanation of the model architecture and the steps involved in implementing the LSTM model consisting of multiple layers with regularization, activation, and other means to reduce underfitting/overfitting. The report also explains the determination of optimal look back and horizon parameters through repeated iterations of the primary model for different [Look-back, Horizon] pairs.

The techniques implemented also include hyperparameter tuning using GridSearchCV and time-series split cross-validation. Walk-forward optimization is achieved by walk-forward rolling time series CV (also called Anchored - Walk). The LSTM autoencoders incorporate dimensionality reduction with a different number of compressed features and a different lookback window from the originally used lookback. Additionally, XGBoost can be used to combine the predictions from multiple LSTM models and improve overall performance. XGBoost is a gradient boosting algorithm that can handle complex non-linear relationships and capture interactions between variables.

The potential outcomes of this project include accurate and reliable forecasts of stock volatility, which can be used by investors to make informed decisions. The report is organized into different sections, including the data preparation, model development, hyperparameter tuning, model evaluation, and conclusion, to provide a detailed description of the methodology used and the results obtained.

**Constraints:** Interpretability has not been implemented in the model.

1. **Data**

Data is downloaded from Yahoo Finance using yfinance library. The dataset contains top 50+ NASDAQ stocks and the user can input different stock symbols. With INTEL Corporation, being the oldest listed stock – the dataset has historical prices of all stocks since its inception in the markets.

* 1. **Raw Structure:**

The dataset has info on Open, High, Low, Close, Adj. Close, Volume, Stock Name as seen in the below figure.

Table

Description automatically generated

**Fig 3.1:** Raw Data

**Source:**Colab Notebook of Author

* 1. **Feature Engineering and Selection:**

The 30-day arithmetic return (𝑟t) of the stock represents the average daily return of the stock over the past 30 trading days. The volatility of the stock is determined by calculating the standard deviation of the 30-day arithmetic returns over a rolling 30-day period. The data frame is filtered to have the following columns as displayed in below code snippet and in Fig 3.2.1.

*df = chosen\_stock[['Open', 'High', 'Low', 'Close', 'Adj Close', '30 day arithmetic return', 'volatility']]*

Text, letter

Description automatically generated

**Fig 3.2.1:** Definition of Feature Xt and Volatility VOLt

**Source:**Assignment Brief

**Normalization:** Standard Scaling (z-score) helps the model specifically time series data to converge faster and improves the training process. It is almost mandatory that we need to implement in any model.

**Train – Test split and Validation data:** The trained data and test data are split by both ways – slicing and train\_test\_split function. Validation split argument ensures a percentage of train data is allocated to improve the model performance.

1. **Model Architecture**

The model architecture appears like a “Hybrid Comparative Modelling” approach that implements all techniques and compares the performance of the resultant models with one another.

|  |  |  |
| --- | --- | --- |
| Diagram  Description automatically generated with low confidence | Diagram  Description automatically generated | Diagram, timeline  Description automatically generated |
| **LSTM model** | **LSTM Autoencoder** | **Encoder** |
| **Fig 4.1:** Model Architecture | | |
| **Tool Used:** <https://netron.app/> | | |

**LSTM Model:** The first one is the primary model implemented at the beginning and this will serve as the basic block throughout the project. This is the representation of one of the calls made to the model from Walk – Forward Optimization section [discussed below].

**Layers and Hyperparameters:** The Exponential Linear Unit (ELU) was specifically chosen here since it gave better results than ReLu in implementing non-linearity in the system. ELU can better interpret negative values especially in case of stock prices where the prices and indicators can change swiftly. It also solves Bias shift issue by controlling the weights and distribution of inputs to the LSTM layers. In the case of stock volatility prediction, it is important to have a model that can capture the temporal dependencies in the time series data, and that can generalize well to new data. The early stopping criteria, dropout layers, used in this model can prevent overfitting and help to generalize the model better. Since the output needs to be a continuous value here, a linear activation output layer has been used instead of Sigmoid/SoftMax. Adam is the primary optimizer chosen in this model. Adam uses adaptive learning rates that change as the training progresses and by keeping a moving average of the past gradients. This allows the optimizer to quickly converge to a good solution and handle different types of data [[1]](#endnote-1).

**Hyperparameter tuning:**

The parameters are tuned in a two-step process here. The optimal look-back, horizon pairs are first determined so that the comparative analysis use the same pair of values throughout the comparative model. In most cases, optimal pairs are found at look\_back = 7 or 14. In most cases where horizon > 3, the prediction is so poor and has been overshot at peaks.



Secondly with limited experimentation and research (Table 1) [[2]](#endnote-2), it was found that it is better to tune the following parameters to improve the model performance and reduce underfitting/overfitting. All the parameters discussed in earlier section (Generalization/Overfitting) regulate fairly by themselves since multiple criteria is involved such as early stopping that monitors the values, stops the training, and restores weights from the best epoch.

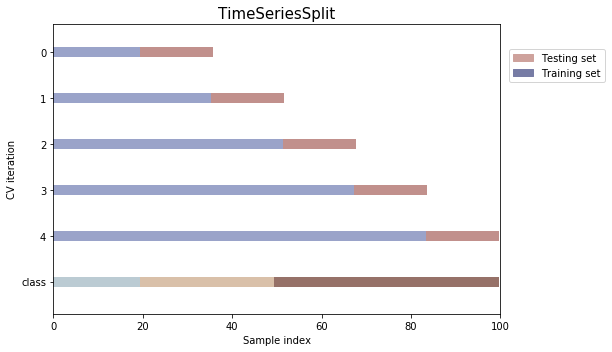
**

The GridSearchCV function is used to perform a grid search with 5-fold time series cross-validation to find the best hyperparameters for the LSTM model. A randomized search might not be helpful here since it can miss some pair combinations and can significantly affect the predictions. An optimal level of neurons and regularization [dropout] prevents underfitting/overfitting.



**Input – Output Pairs Generator:** This is one of the critical elements in the model where the function then creates input-output pairs by sliding a window of size look\_back over the data and taking the next horizon time steps as the output. This process is repeated until the end of the data is reached. This is essential for sequential time series data to capture long term and temporal dependencies in data.

**Walk – Forward Optimization:** This technique uses rolling time series cross validation (anchored walk forward) where the data is split into ‘k’ walks (folds). Using rolling time series cross-validation allows us to evaluate the performance of the model on multiple testing sets and provides a more realistic estimate of the model's performance. Moreover, by shifting the training and testing periods forward, we can assess how well the model generalizes to future data, which is the goal of time series forecasting. While it is expected to get a better result at kth walk, that’s not always a guarantee. The same can be seen in MSFT and TSLA stock prediction where the 1st walk didn’t converge and comparatively converged better at 4th walk. But the 5th walk test\_loss increased abruptly. The predictions at the 5th walk were better in the INTC stock [Refer Appendix and Results Analysis]. Nevertheless, the technique is useful since it incorporates arising patterns as the model moves through the training data across walks [[3]](#endnote-3).



**Fig 4.2:** Rolling Time Series Cross Validation

**Source:** <https://goldinlocks.github.io/>

**Autoencoder and Dimensionality Reduction:**

|  |  |
| --- | --- |
| Calendar  Description automatically generated | Chart, calendar  Description automatically generated |

From the matrices, we can clearly conclude that [ Open, High, Low, Close, Adj. Close ] are highly correlated with each other. Reducing dimensions using LSTM autoencoder can help remove noise and reduce computational costs during training. Therefore, the author has assigned reduced dimension range of [ 4, 5, 6] since that would be more reasonable and efficient to make predictions. Autoencoder is trained with a different lookback window (LB – 1) and the compressed X\_train - X\_test pairs are extracted from the Encoder. ReLu activation functions are commonly preferred, and it applies non-linearity to the series. The compressed input train and test data are fed into the LSTM model and target variable Volatility is determined.

**XGBoost – Ensemble Model:**

For better predictions, an ensemble method XGBoost with 5 iterations of LSTM is implemented with weighted averages of final predictions. The hyperparameters in XGBoost are tuned to prevent overfitting and to ensure the model generalizes well to new data. For example, the max\_depth and min\_child\_weight hyperparameters control the complexity of model. If these hyperparameters are too high, the model can become too complex and start fitting the noise in the data. Similarly, the gamma, reg\_alpha, and reg\_lambda hyperparameters control the regularization of the model, which can prevent overfitting.

1. **Result Analysis:**

Background pattern

Description automatically generated with medium confidence

**Fig 5.1:** Walk – Forward Optimization: Walks 1 to 5 for ‘MSFT’ Stock

As discussed earlier, this clearly shows how the test loss has increased for walk 5 but remained a little over 0 until walk 4. While walk forward optimization captures temporal patterns efficiently, it might not decrease the test loss consistently in upcoming walks. In ‘INTC’ stock, it can be observed that the predictions converge consistently across all walks [Plots attached in Appendix].

A picture containing text

Description automatically generated

**Fig 5.2:** Loss of models with different (L, H) for INTC Stock

From Fig. 5.2, we can clearly see that the look\_back and the horizon should not be too large for a good model. The Appendix plots contains all indicators and values inside the plots.

**APPENDIX :**

**Figure 1 :** Normalized and Denormalized Values of MSFT stock data

Diagram

Description automatically generated

**Figure 2 :** Predictions for Look\_back – Horizon Pair (INTC)

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Figure 3 :** Predictions for Look\_back – Horizon Pair (MSFT)

|  |
| --- |
| Graphical user interface, chart, histogram  Description automatically generated |
| Graphical user interface, chart, histogram  Description automatically generated |
| Graphical user interface, chart, line chart, histogram  Description automatically generated |
| Chart  Description automatically generated |

|  |
| --- |
| Chart, line chart  Description automatically generated |
| Chart, line chart  Description automatically generated |
|  |
| Chart, line chart  Description automatically generated |

**Figure 4:**  Prediction with different look\_back , Horizon pairs – TSLA

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Figure 5:**  Prediction after Walk Forward Optimization – INTC

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Figure 6:**  Prediction after Walk Forward Optimization – MSFT

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Figure 7:**  Prediction after Walk Forward Optimization – TSLA

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Figure 8:**  Prediction after Dimensionality Reduction Autoencoder – INTC

|  |
| --- |
|  |
|  |
|  |

**Figure 9:**  Prediction after Dimensionality Reduction Autoencoder – MSFT

|  |
| --- |
|  |
| Graphical user interface, chart, scatter chart  Description automatically generated |

**Figure 8:** Ensemble Model – INTC followed by MSFT

|  |
| --- |
| **Graphical user interface, chart, line chart  Description automatically generated** |
| **Graphical user interface  Description automatically generated** |

**References:**

1. Chang, Z., Zhang, Y. and Chen, W. (2019). Electricity price prediction based on hybrid model of adam optimized LSTM neural network and wavelet transform. *Energy*, 187, p.115804. doi:https://doi.org/10.1016/j.energy.2019.07.134. [↑](#endnote-ref-1)
2. Konar, J., Khandelwal, P. and Tripathi, R. (2020). *Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network*. [online] IEEE Xplore. doi:https://doi.org/10.1109/SCEECS48394.2020.94. [↑](#endnote-ref-2)
3. alphascientist.com. (n.d.). *Stock Prediction with ML: Walk-forward Modeling — The Alpha Scientist*. [online] Available at: https://alphascientist.com/walk\_forward\_model\_building.html [Accessed 23 Mar. 2023].

   ‌ [↑](#endnote-ref-3)