# Group 25 - Project Report

- Jiaqi Guo (jguo288@wisc.edu)
- Yutong Wei (ywei88@wisc.edu)
- Yuqi Zhang (yzhang2756@wisc.edu)
- Naomi Lim (nlim6@wisc.edu)

## 1. Introduction

In the rapidly evolving world of finance, accurate predictions of stock prices can be a substantial advantage for investors, financial analysts, and the companies themselves. Our project aims to leverage advanced machine learning techniques to predict the stock prices for real-world data, especially focusing on comparing the predicting performance of Long Short-Term Memory (LSTM) network and transformer model.

Given the complexities and the volatile nature of financial markets, traditional predictive models often fall short in capturing the dynamic temporal dependencies essential for accurate stock price forecasting. Long Short-Term Memory model can not only capture short-term time correlation but can also accurately predict long-term information. Besides, as the large language model like ChatGPT became increasingly popular, transformer architectures have also obtained attention and have been applied for time series forecasting. However, there is a gap of which models perform better at long-term time series prediction. To address the price prediction challenges, and also compare the performance of these two promising models, our project utilized a stock-market-Time Series dataset to implement these two deep learning models. To be specific, we choose the company Adani Ports for both models.

## 2. Literature Review

Time series forecasting involves analyzing past data to predict future outcomes. Many algorithms struggle to learn the long-term relationships inherent in time series data. However, LSTM (Long Short-Term Memory) networks, a specialized type of deep learning approach, are designed to effectively address this challenge by capturing these extended dependencies, actively being applied in predicting future trends (Hua et al., 2019).

Since its publication in 2017, the transformer model become a prevailing focus of deep learning research. It was originally developed to process long sentences. However, its scope has expanded beyond NLP applications to a variety of applications. According to research, about 650 Transformer-related models are used in various fields. (Islam et al., 2023).

## 3. Methodology

### 3.1 LSTM Model Description

The Long Short-Term Memory (LSTM) network is an advanced type of Recurrent Neural Network (RNN) designed to handle sequence prediction problems with input data where the context from the input is retained over time. This ability makes LSTMs particularly useful for time series forecasting tasks.

### 3.2 Mathematical and Statistical Foundations:

LSTMs mitigate the vanishing gradient problem inherent in traditional RNNs through a sophisticated cell structure that includes gates controlling the flow of information. Each LSTM unit consists of:

- Forget Gate: Determines the information to discard from the cell state.
- Input Gate: Decides which new information is added to the cell state.
- Output Gate: Produces the output based on the current input and the memory of the cell.

These gates utilize sigmoid activation functions to output values between 0 and 1, effectively controlling the extent to which information should be allowed through.

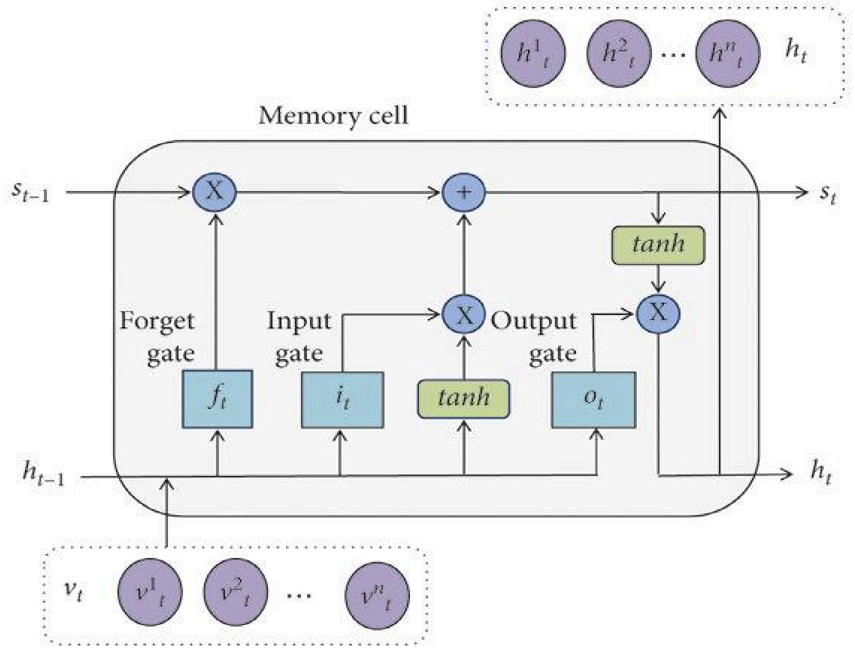


**Figure 1**: Architectural Diagram of the LSTM Model (ResearchGate, n.d.)

## 3.3 Step-by-Step Data Processing in LSTM:

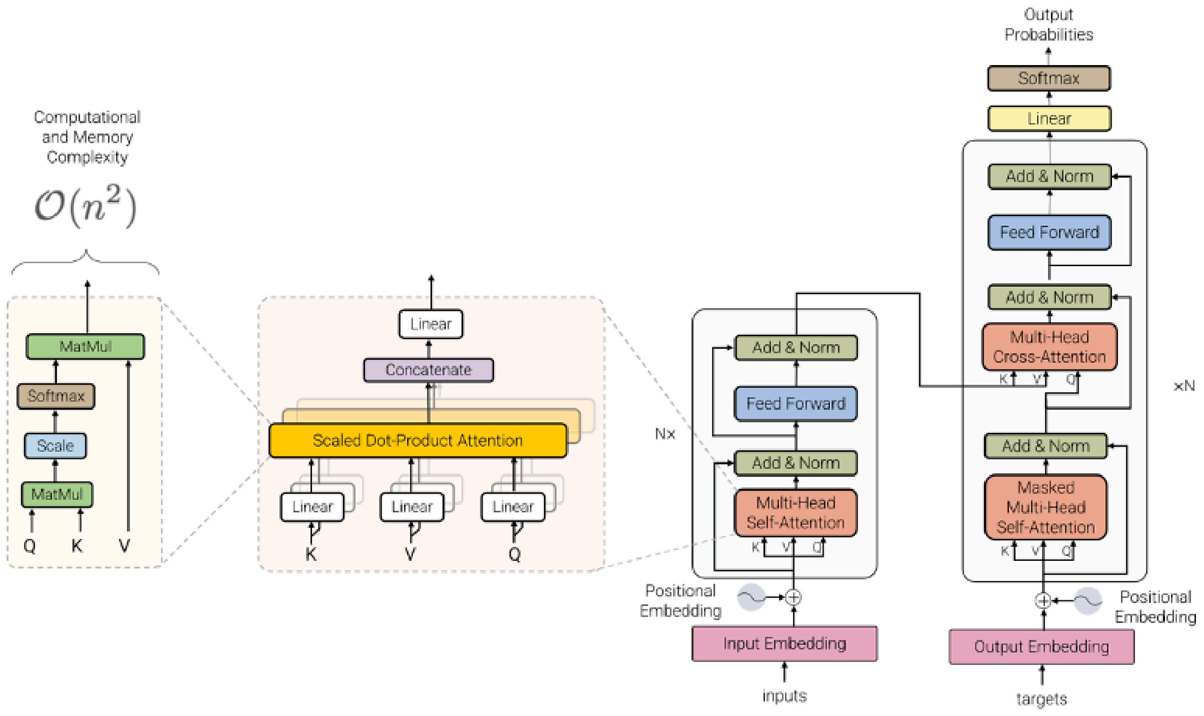
1. Input Integration: At each time step, input data and the output from the previous LSTM unit are processed.
2. Gates Operations: Data flow through the gates—determined by the weight matrices and bias vectors of the model—modifies the cell state, which holds information over arbitrary time intervals.
3. Output Generation: The final output of the LSTM unit is calculated based on the current state of the cell, potentially passing through more layers or directly influencing the model output.

## 3.4 Transformer Overview

Transformer is a deep learning architecture that utilizes a self-attention mechanism to capture contextual relationships within sequential data. The development of the Transformer model was mainly motivated and entirely relied on the attention mechanism to draw global dependencies between input and output (Vaswani et al., 2017).

### 3.5 Mathematical and Statistical Foundations: Transformers

Transformers address the challenges of sequence dependency in time series data without the sequential computation of RNNs, through a unique architecture built around the mechanism of self-attention. Each Transformer model consists of: Self-Attention Mechanism, Positional Encoding, Multi-Head Attention, Feed-Forward Neural Networks, Normalization and Dropout. These components work together within an encoder and decoder framework. The Transformer processes entire sequences in parallel during training, significantly reducing computation time and enabling the model to capture complex patterns in large datasets effectively.



## 4. Implementation

### 4.1 Transformer Architecture

We initiated our project by loading and preparing the time-series data crucial for training our deep learning model. The 'Date' column was converted to datetime format, and the data was sorted chronologically. To ensure consistent training dynamics, the 'VWAP' values were normalized using MinMaxScaler to a [0, 1] range. This step, along with dividing the data into 70% training, 15% validation, and 15% testing sets, was fundamental for preparing the model inputs and evaluating performance across different datasets.

After normalization, we converted the data into sequences that predict the next day's 'VWAP', aligning with common practices in time series forecasting. The data preprocessing step involved using a window size of eight timesteps, creating sequences where each input contains the previous eight data points to predict the target value for the following day.

To optimize our machine learning pipeline, we designed a transformer model specifically tuned for our dataset with dimensions (8, 1). The model leverages a custom-built transformer encoder, defined by parameters such as a head size of 46, 60 attention heads, and a feed-forward network dimension of 55. The transformer encoder begins with a LayerNormalization step, followed by a MultiHeadAttention layer that identifies relationships within the data. This configuration was embedded within a series of five transformer blocks to adequately capture the complexities of the input data.

For training, the transformer model was compiled using mean squared error as the loss metric and optimized with the Adam optimizer for efficient gradient handling. We implemented a custom learning rate scheduler of 0.0004 to enhance convergence speed and introduced an early stopping mechanism to prevent overfitting, ensuring optimal performance and generalization on unseen data.

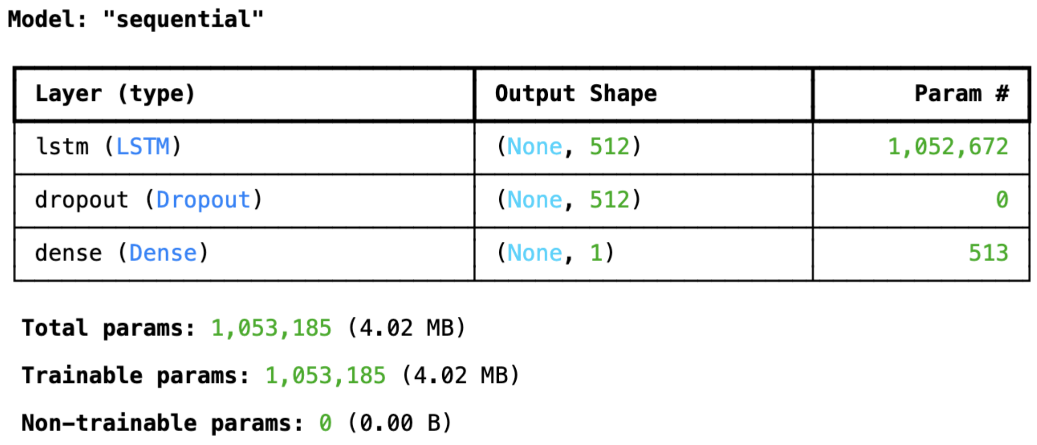
Finally, the trained model was evaluated on both validation and test datasets to assess its predictive accuracy and generalization ability outside of the training data scope. This step is crucial for verifying the model's effectiveness in real-world scenarios, where it needs to operate reliably on unseen data.

## 4.2 LSTM

Before building and training the LSTM Model, we first conducted data cleaning steps as those used for the Transformer model, to ensure that it is comparable in its model performance with the Transformer model in the end. The same columns, 'Date' and 'VWAP' were selected, and the MinMaxScalar method was implemented to normalize the ‘VWAP’ values. To ensure consistency when comparing the model with the Transformer, the dataset was split into 70% for training, 15% for validation, and 15% for testing. Unlike typical random splitting methods used for other models, the data splitting for this time series data is based on their chronological order by calculating the indices for the data. This ensures that the sequence and time dependencies to time series are continuously maintained.

For hyperparameter tuning, we use Random Search via the KerasTuner library to efficiently find a good hyperparameter combination. This method explores various hyperparameter combinations randomly from predefined values. The optimal hyperparameter combination found through this process includes setting the number of units(neurons) to 512, the number of return sequences(i.e. number of hidden states) to be zero, and adding a dropout layer after an LSTM layer, which helps prevent overfitting by randomly omitting some of the features during training. One unique parameter we set for the LSTM is the look-back value, which is the number of past time steps the LSTM model will look at in order to predict stock price at the subsequent time step.

The model is configured with the Adam optimizer and the mean squared error as the loss function, as it is in the Transformer. During training, early stopping regularization is implemented to avoid overfitting and save training time. This technique monitors the model’s performance and stops the training process when the performance no longer improves or starts to degrade. At the end, a dense layer outputs the prediction.



```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm (LSTM)                     │ (None, 512)            │     1,052,672 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 512)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 1)              │           513 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 1,053,185 (4.02 MB)

 Trainable params: 1,053,185 (4.02 MB)

 Non-trainable params: 0 (0.00 B)
```

**Figure 2**: LSTM Model Summary

## 5. Results and Discussion

### 5.1 Transformer Architecture

| RMSE | MAE | MSE | Returns RMSE | Returns MAE | returns MSE |
|---|---|---|---|---|---|
| 12.12 | 5.36 | 146.88 | 376.15 | 375.32 | 141,490.12 |

**Table 2**: Performance Metrics for VWAP Predictions and Returns

The model's performance was evaluated through a comprehensive set of error metrics, including root mean squared error (RMSE), mean absolute error (MAE), and mean squared error (MSE) for both stock price and returns predictions. For price forecasts, the model achieved a reasonably low RMSE of 12.12, reflecting accurate predictions that closely matched actual values. The MAE of 5.36 indicates a consistent predictive capability, with an average error of just over five units, highlighting the model's reliability in forecasting price trends. The price MSE of 146.88, which amplifies larger prediction errors, remains within an acceptable range, showcasing stable performance despite the inherent variability of financial data.

However, for returns predictions, the model faced challenges due to the inherently volatile nature of returns data. The returns RMSE of 376.15 reveals the difficulty of accurately predicting returns, which are subject to high variability. The MAE of 375.32 further emphasizes the model's sensitivity to abrupt market shifts that are characteristic of returns. Additionally, the returns MSE of 141,490.12 reflects the unpredictable nature of returns data and the significant discrepancies in predicting volatile market movements. These metrics underscore the inherent complexity of capturing returns patterns effectively.

The results demonstrate the transformer model's potential in recognizing and predicting intricate stock price patterns but also reveal its struggle with the nuanced and abrupt shifts of market returns. While the model identifies broader trends accurately, its difficulty in predicting returns highlights the need for feature engineering approaches that incorporate more comprehensive market indicators. Further refinement in hyperparameter tuning or an ensemble model approach could improve returns prediction. Despite these challenges, the results point to the promising application of transformer models in financial forecasting, providing clear directions for future research and refinement.

## 5.2 LSTM

To evaluate the performance of the LSTM model, the same metrics are chosen; MSE, MAE, and RMSE. As shown in the table below, the model has an underfitting problem, where the train score is greater than the test score.

| | MSE | MAE | RMSE |
|---|---|---|---|
| Train | 548.87 | 9.74 | 23.43 |
| Test | 71.06 | 6.15 | 8.43 |

**Table 2**: Train vs. Test Score of the LSTM model(Underfitting)

The underfitting problem occurs due to the irregular trends in the dataset used for training. As mentioned above, the data are split into three different sets for training, validation, and testing based on their time indices in chronological order. Looking at the plot illustrating the actual dataset and predictions based on each splitted dataset, we can see that there were huge stock price dynamics starting from 2008 to 2011. By using them to train the model, it may become more used to predict dynamic trends in the future. However, the trend becomes more stable as time goes forward, which would have made predictions easier for the test data. This transition to a stable phase might not be well captured by the model due to the higher volatility and lead to underfitting during these more stable periods.

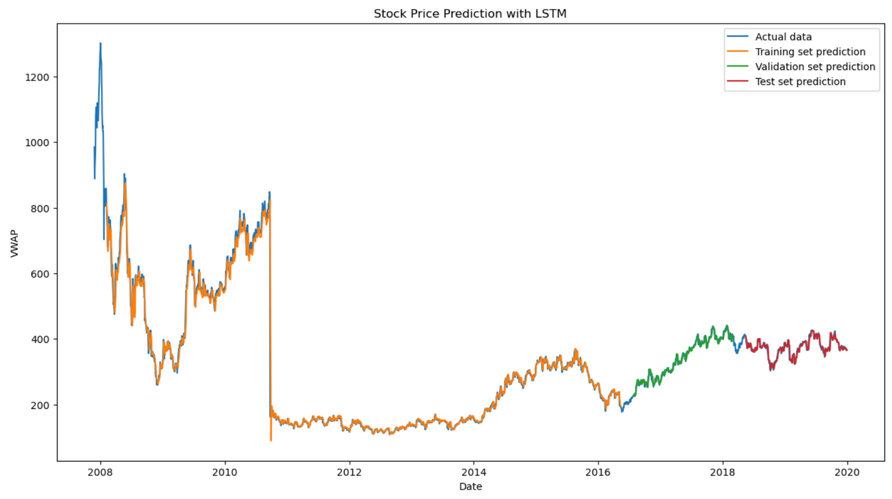


**Figure 2**: Stock Price Prediction with LSTM

Despite this underfitting issue, the model remains useful and predicts stock prices well. Therefore, while underfitting is generally viewed as a drawback in a view of deep learning techniques, it can still hold significant value from a business perspective because of the fairly low MSE, MAE, and RMSE scores to be applied in real business scenarios. Even though the model does not capture every nuance of market behavior, its predictions could still be meaningful to inform profitable investment decisions in practice.

## 5.3. Comparison of LSTM and Transformer Performance

By comparing the three evaluation scores(MSE, MAE, and RMSE) between the LSTM and the Transformer models, we could conclude which model works better or not. As the chart shows below, the LSTM has lower values for MSE and RMSE while the Transformer has a lower value for MAE. This means that the LSTM model is more effective for this project, and it is likely due to the relatively small size of our dataset.

|  | MSE - test | MAE - test | RMSE - test |
|---|---|---|---|
| LSTM | 71.07 | 6.15 | 8.43 |
| Transformer | 146.88 | 5.36 | 12.12 |

**Table 3**: LSTM and Transformer Performance using MSE, MAE, and RMSE

Transformers are typically useful with larger datasets for their full potential because they depend heavily on extensive data to capture complex patterns and dependencies accurately. Therefore, for future projects involving larger datasets with longer periods of data, the Transformer model would be more effective.

## 6. Conclusion

This project aimed to assess the effectiveness of LSTM and Transformer models in predicting stock prices, particularly for the stock of Adani Ports. Through comparative analysis, we observe that the LSTM model shows better performance in terms of mean squared error(MSE), mean absolute error(MAE), and root mean squared error(RMSE). On the other hand, as mentioned above when comparing the two models, Transformer would be more effective if we use Transformer for larger datasets due to its ability to capture long-term dependencies.

Given those findings, future projects could benefit from deploying Transformer models on larger datasets, where their ability to dissect and learn from extensive and complex data sequences can be fully utilized. For datasets similar in scope to the one used in the study, LSTM models may continue to provide more reliable and accurate forecasts.

Ultimately, this project highlights the strengths and weaknesses of both LSTM and Transformer models in the context of financial forecasting. It provides a foundation for future research and practical applications which utilize more diverse datasets to optimize the predictive capabilities of machine learning models in the financial industry.

## 7. Contribution

Everyone contributed equally to the project.

## Reference

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2019). Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, *57*(6), 114-119.

Islam, S., Elmekki, H., Elsebai, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. (2023). A comprehensive survey on applications of transformers for deep learning tasks. Expert Systems with Applications, 122666.