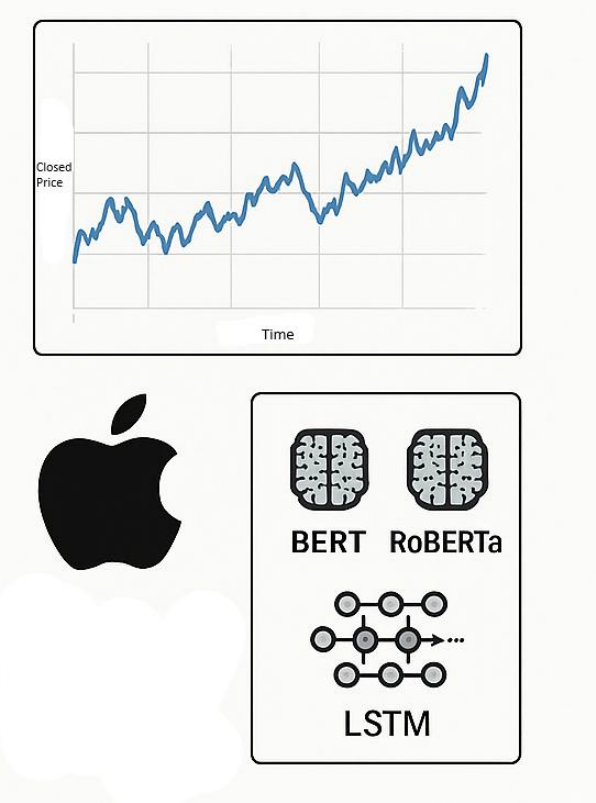


# APPLE CLOSING PRICE FORECAST USING DEEP LEARNING

Mahdi Mohammadi
Big Data Processing

## Dataset Selection

For this task, the main objective is to predict the closing price of Apple Inc. To ensure sufficient data, we also use data from other companies to train the model. Specifically, we include stocks of companies whose closing price has a correlation greater than 83% with Apple's closing price. For the test dataset, we first sort Apple's data chronologically and take the last 40% as the test set. The remaining 60% is combined with the data from other companies for training. The training set is then split into training and validation subsets in an 80% to 20% ratio. Further explanation on this split is provided later.

## Extracting Financial Market Indicators

After identifying the relevant companies and before splitting the data, we calculate the RSI and MACD indicators for each company's data. These indicators are used as additional features for model training.

## Data Preprocessing

After splitting the data into training and test sets, we scale the training feature vectors—["Close", "MACD_Line", "RSI"]—using the MinMax method. The same min and max values from the training set are then used to scale the test set. Only after this step do we perform the train/validation split.

## Preparing Data for BERT and RoBERTa Models

Before feeding the data into models that use BERT or RoBERTa as encoders, the data format must be adjusted. The following steps are taken:

- **Sorting by Date**
  The data is sorted by the date column to ensure the sequence is in chronological order.
- **Creating Time Windows (Windowing)**
  For each record, a time window of fixed size is created. This window includes the Close, MACD_Line, and RSI values for the past 5 days. The goal is to extract a sequence of these features to predict the Close value for the next day.
- **Rounding Values to Fixed Decimal Places**
  Each feature value in the time window is rounded to a fixed number of decimal places to simplify the values and ensure proper functioning of the Attention mechanism.
- **Converting Data to Text Format**
  The time window data is converted into a format understandable by BERT. This involves combining each row's values into text strings and adding [CLS] and [SEP] tokens.
  Example:

[CLS] 100.5 0.25 45.3 [SEP] 101.2 0.30 46.8 [SEP] 102.1 0.35 47.2 [SEP] 103.0 0.40 48.1 [SEP] 103.7 0.45 49.5 [SEP]

- This is a sample input with a window size of 5.
- The function `create_bert_ready_data` repeats this process for all companies in the dataframe. The prepared data includes sequences and target values, ordered chronologically for each company.
- **Tokenizing the Data (Tokenization)**
  The prepared text strings are passed to the `encode_bert_input` function to be converted into tokens readable by BERT.
- **Tokenization**
  BERT converts the text into individual tokens using `BertTokenizer`.
- **Attention Mask**
  A mask is created to indicate which tokens should be processed by the model. PAD tokens are ignored.
- **Truncation & Padding**
  Sequences longer than the maximum allowed length (512) are truncated, and shorter sequences are padded with PAD tokens.

## Preparing Data for the LSTM Model

For this part, the data undergoes several transformations. First, lagged features are added to the dataset. Missing values (caused by lagging) are filled using the backfill method (i.e., using future values). Next, the `singleStepSampler` function is used to generate LSTM-compatible input sequences of a specified window size. The window size determines how far back the LSTM looks in time.

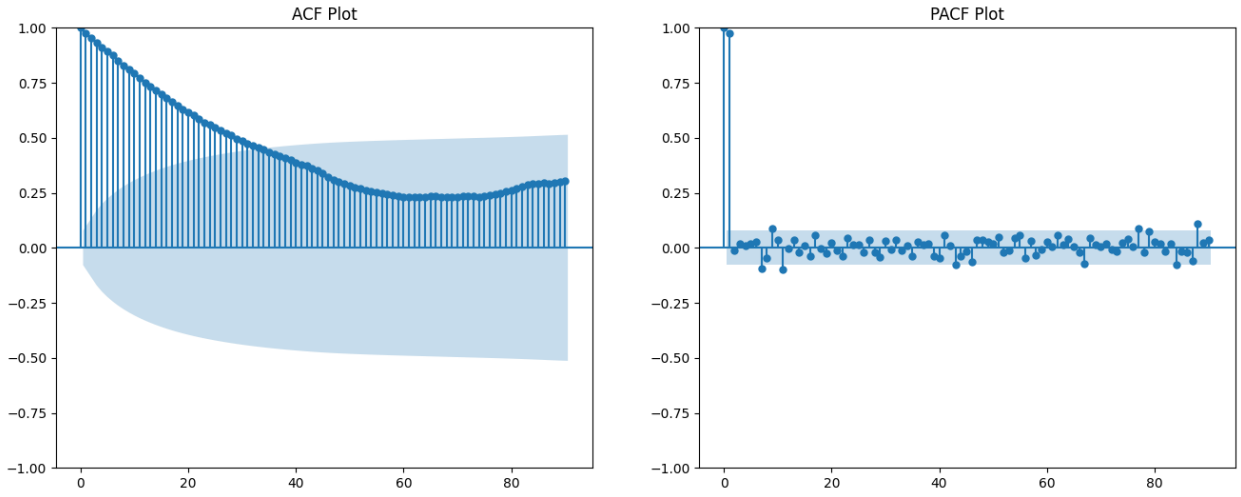


*Figure 1 – ACF and PACF Plots*

Based on the ACF plot in Figure 1, which shows autocorrelation, a window size of 15 was chosen—corresponding to points with at least ~70% correlation. The LSTM model follows a seq2vec design, predicting the 16th value based on the previous 15 inputs.

## Model Architectures

For the BERT and RoBERTa-based models, the following structure is used:

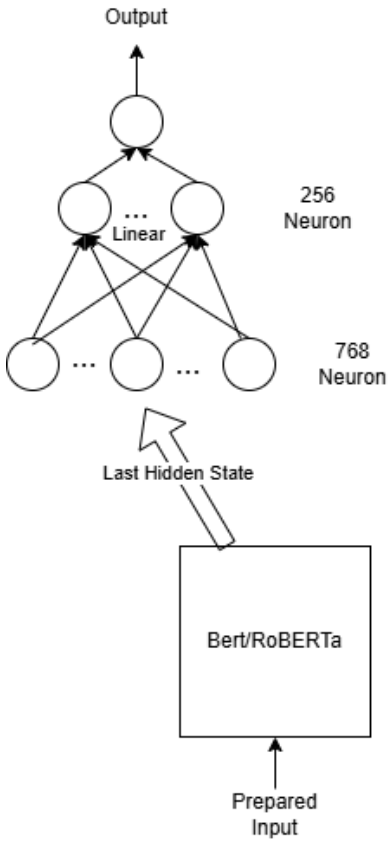


*Figure 2 – Architecture of BERT and RoBERTa-Based Models*

To prevent overfitting, a dropout rate of 15% is applied between the FC input and the last hidden state, as well as between the two FC layers.

For the LSTM-based model, the structure is as follows:

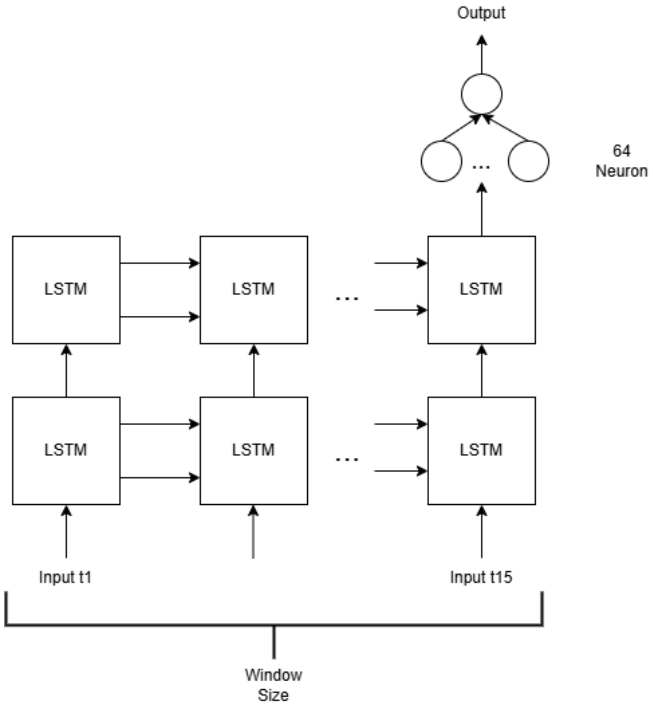


*Figure 3 – Architecture of the LSTM-Based Model*

It's worth noting that all structural and training hyperparameters were determined through extensive trial and error.

## Training Process

Due to their complexity, all models used in this project are highly prone to overfitting. Therefore, early stopping and weight decay were applied during training. The training and validation loss curves are fully available in the attached notebook. To avoid unnecessary length, only one example is shown here:

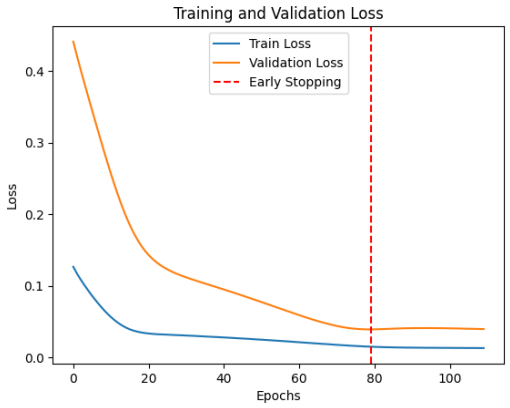


*Figure 4 – Training and Validation Accuracy per Epoch for LSTM with Lag 3*

## Model Comparison and Results

All results are summarized in the table below. The evaluation metric, as specified in the task, is MSE. The best values for each prediction horizon are bolded.

*Table 1 – Summary and Comparison of Results*

| Model / Encoder | Lag | 1-Day MSE | 7-Day MSE | 30-Day MSE |
|---|---|---|---|---|
| **BERT** | 3 | 0.036 | 0.028 | 0.041 |
| | 7 | 0.037 | 0.011 | 0.042 |
| | 15 | 0.045 | 0.033 | 0.071 |
| | 30 | 0.055 | 0.110 | 0.120 |
| | 90 | 0.017 | 0.006 | 0.013 |
| **RoBERTa** | 3 | 0.260 | 0.036 | 0.210 |
| | 7 | 0.260 | 0.267 | 0.270 |
| | 15 | 0.265 | 0.260 | 0.350 |
| | 30 | 0.267 | 0.413 | 0.477 |
| | 90 | 0.236 | 0.203 | 0.268 |
| **LSTM** | 3 | 0.008 | 0.004 | 0.159 |
| | 7 | 0.008 | 0.007 | 0.012 |
| | 15 | 0.009 | 0.003 | 0.002 |
| | 30 | 0.017 | 0.021 | 0.020 |
| | 90 | 0.028 | 0.030 | 0.022 |

As shown, the LSTM model significantly outperforms the Transformer-based models. We conclude that the best model is the LSTM with a lag of 15.

For multi-step forecasting, since the model input is multivariate, we use a recursive approach: the first test input is fed into the model, the output replaces the next test point's Close value, and this new point is fed back into the model. This process continues until the desired forecast horizon is reached. While this may involve some simplification (since indicator values should also change with the Close price), the results are reasonably accurate.

Below are sample plots comparing the predicted and actual values for the LSTM model with lag 3 (other plots are available in the notebook):

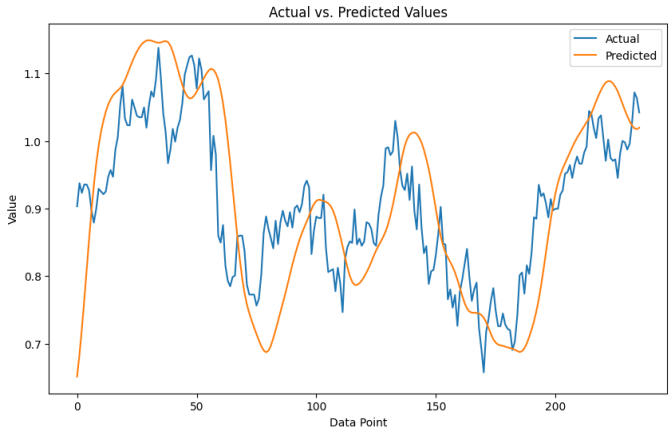


*Figure 5 – Actual vs. Predicted Close Price for Next Day (LSTM, Lag 3)*

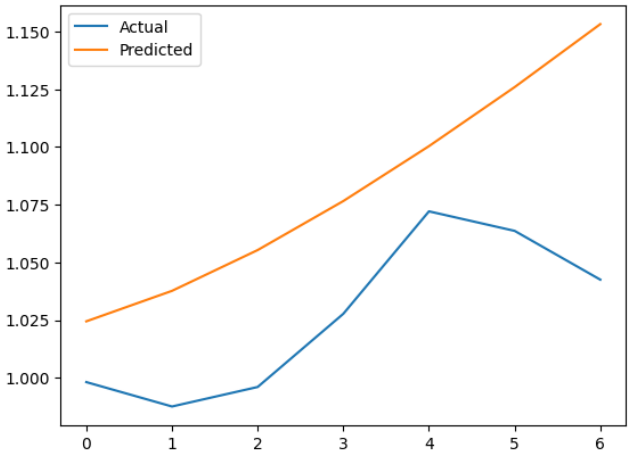


*Figure 6 – Actual vs. Predicted Close Price for Next Week (LSTM, Lag 3)*

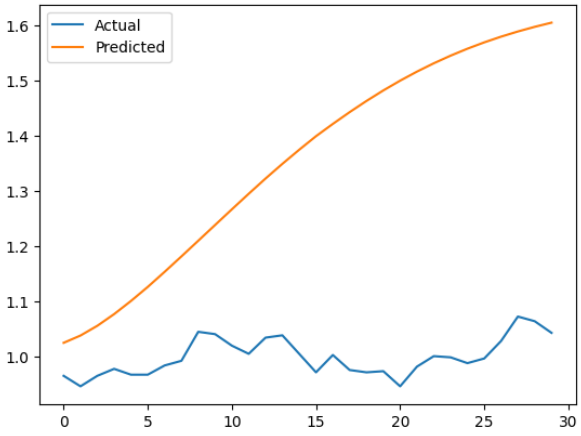


*Figure 7 – Actual vs. Predicted Close Price for Next Month (LSTM, Lag 3)*