Time Series Modelling using Recurrent Neural Network - LSTM

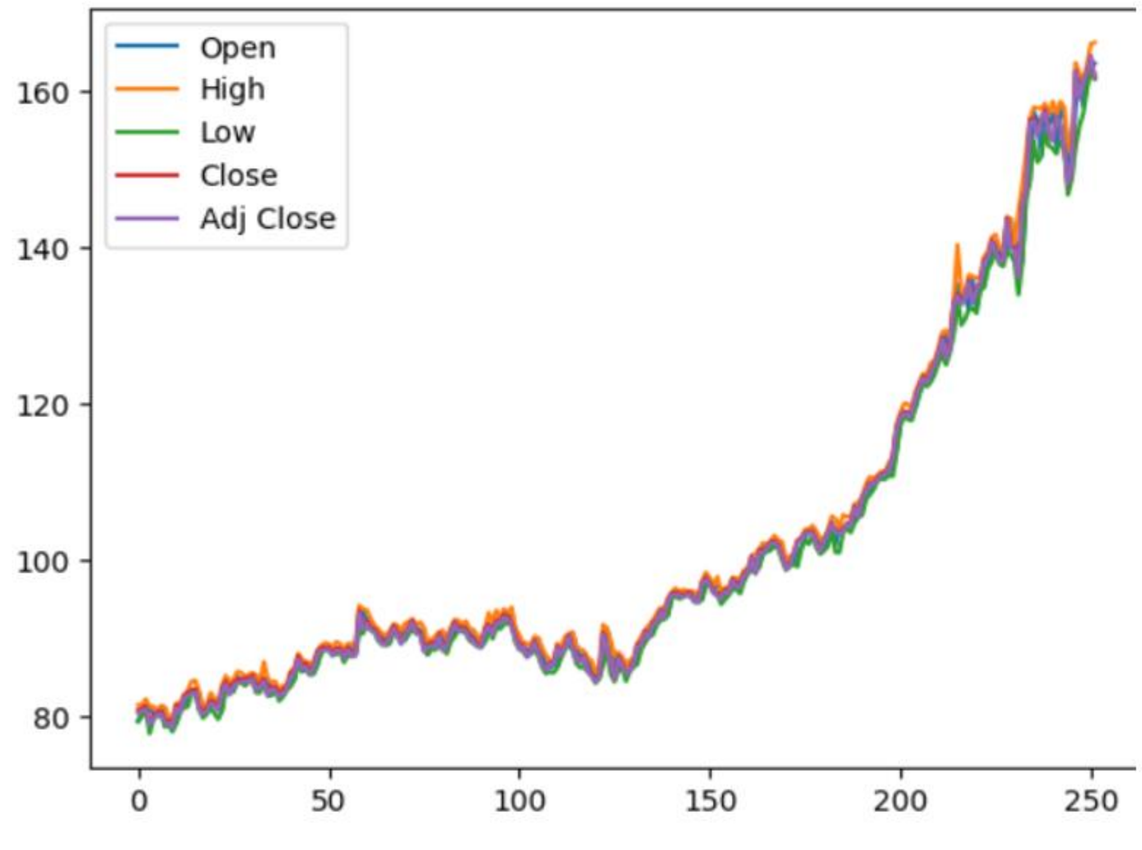
1. Abstract

This report is based on a comprehensive analysis of time series modeling using the Recurrent Neural Network (RNN), specifically focusing on the Long Short-Term Memory (LSTM) model. The study is conducted on the General Electric Company's historical stock prices, sourced from Yahoo Finance, to predict future stock movements. It also explains the effectiveness of LSTM over traditional RNN by comparing their performance across different window sizes.

2. Data and Motivation

2.1 Dataset Description

The dataset comprises the historical stock prices of General Electric Company (GE) obtained from Yahoo Finance. This dataset includes daily records of the opening, highest, lowest, and closing stock prices, along with the adjusted closing prices.



2.2 Advantages and disadvantages of using LSTM

Pros of Using LSTM:

- Long-Term Dependencies: LSTM models are well-equipped to remember information for extended periods, unlike traditional RNNs. This is important for financial time series forecasting where past information significantly influences future trends.
- Robustness to Window Size: LSTMs demonstrate consistent performance even as the input window size varies, indicating their capability to capture dependencies over longer sequences without a significant drop in accuracy.

Cons of Using LSTM:

- Computational Complexity: LSTMs are more resource-intensive compared to RNNs due to their complex architecture. This can lead to increased computational costs and time, especially with large datasets.
- Parameter Tuning: The performance of LSTM models is highly sensitive to hyperparameter settings, including the window size and the number of LSTM units. This requires careful tuning, which can also be time-consuming.

2.3 Advantages and disadvantages of using RNN

Pros of Using RNNs

- Good for Sequential Data: RNNs are great for tasks where the order and context of your data matter, like text or time series analysis.
- Handles Variable Length: They can work with inputs and outputs of different lengths, which is useful for sentences of varying lengths or time series of different durations.
- Efficient Use of Parameters: RNNs reuse the same weights while processing a sequence, which means they have fewer parameters and are more memory efficient than fully connected networks.

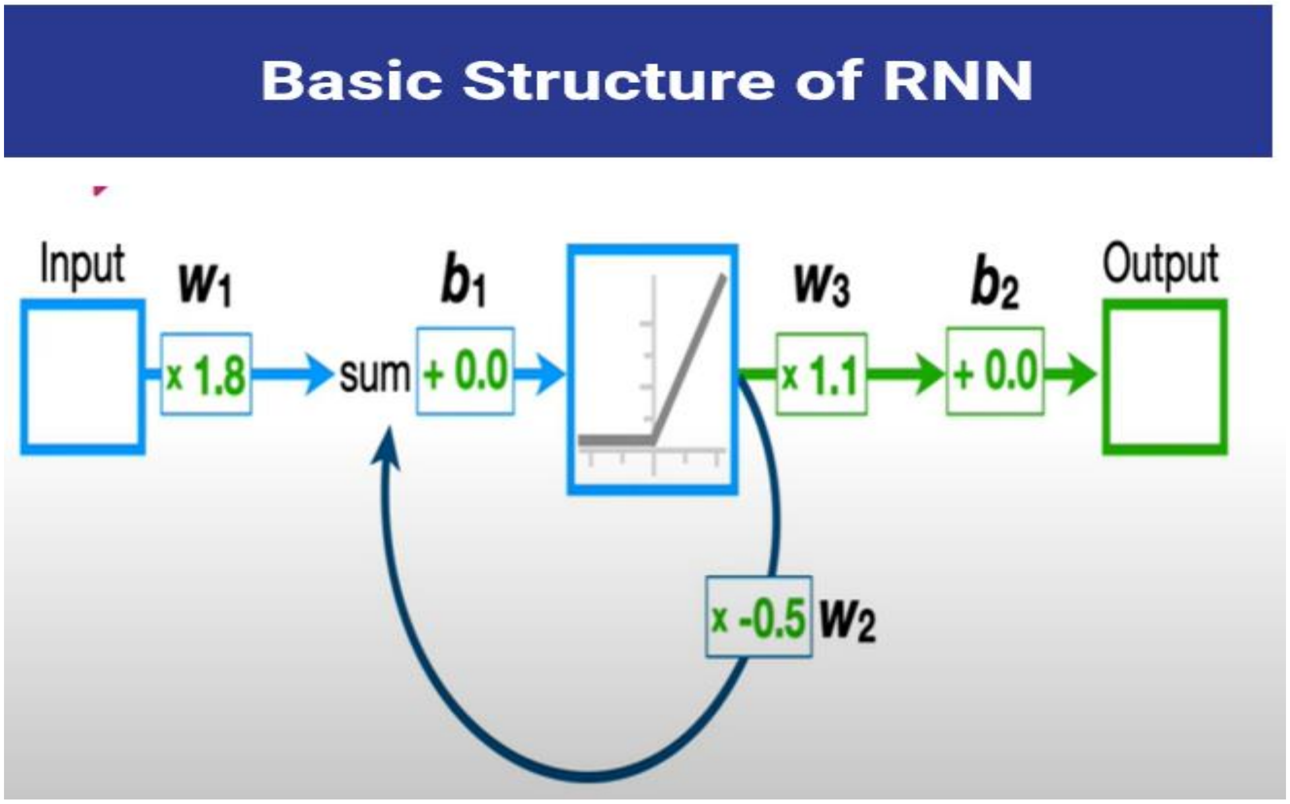
Cons of Using RNNs

- Vanishing and Exploding Gradients: RNNs often struggle to learn long-term dependencies because the gradients can either become too small (vanish) or too large (explode) as they propagate through time, making it hard to train.
- Slow Training: The sequential nature of RNNs prevents parallel processing of the data points in a sequence, leading to slower training compared to other neural networks.
- Short-Term Focus: Standard RNNs have trouble remembering information from the early parts of the sequence, making them less effective for tasks that need long-term memory.

3. Methods

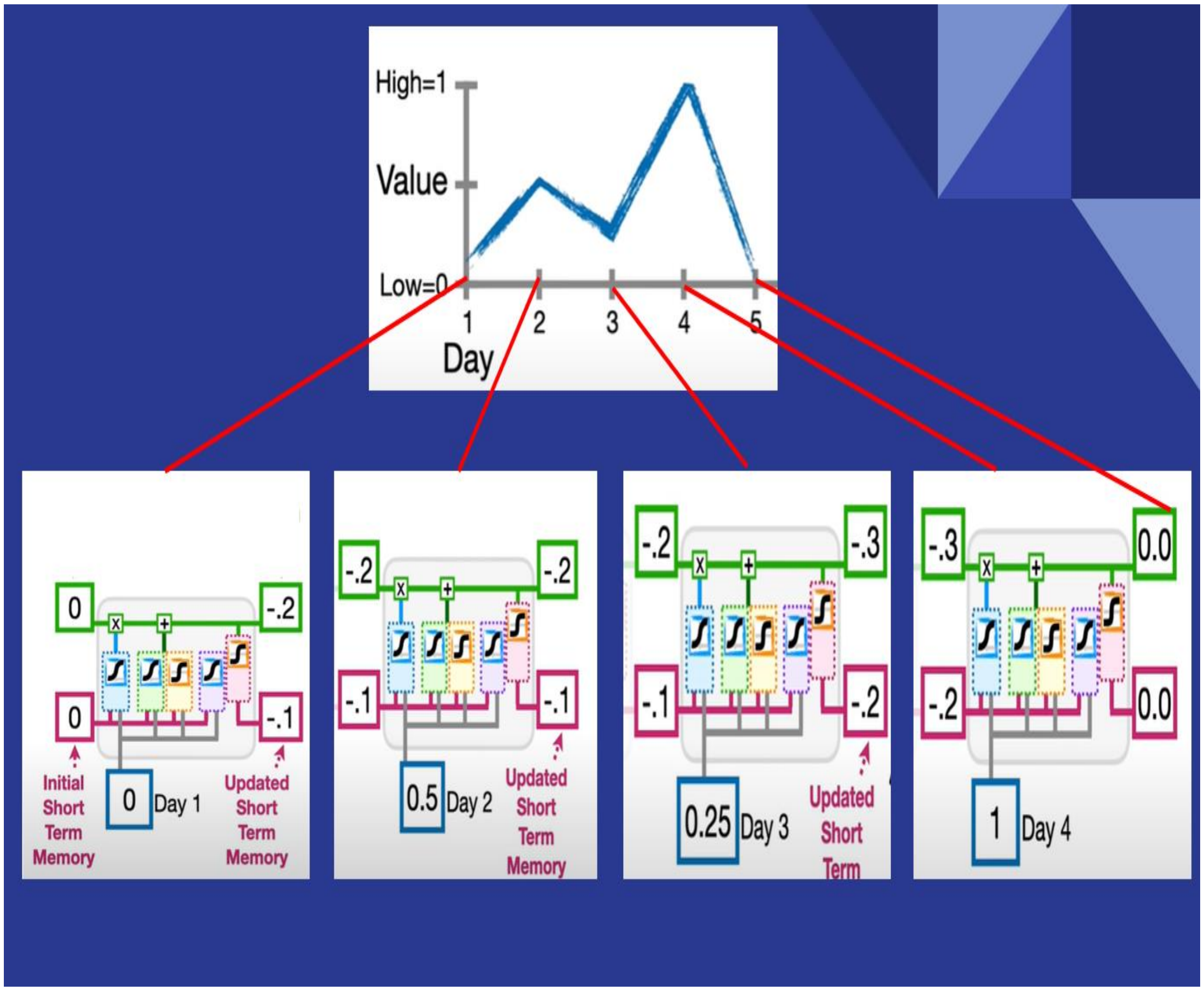
3.1 Theory of RNN and LSTM

Recurrent Neural Network (RNN):



- RNNs are a type of neural network specially designed to handle sequences of data, like sentences, time series data, or video. They are like a chain where each link can pass information to the next. This design helps RNNs remember information from previous steps in the sequence, which helps them understand everything better. This memory feature makes RNNs good at tasks where context and history are important, like predicting the next sequence or understanding the trend in stock market prices.

Long Short-Term Memory (LSTM):



- LSTMs are an extension of RNNs designed to avoid the long-term dependency problem. They contain mechanisms called gates that regulate the flow of information. These include:

- Forget Gate: Decides what proportion of the past information is to be forgotten.
- Input Gate: Updates the cell state by regulating the flow of new information into the cell.
- Output Gate: Determines what next to put out to the hidden state.

3.2 Usage During Analysis

3.2.1 LSTM:

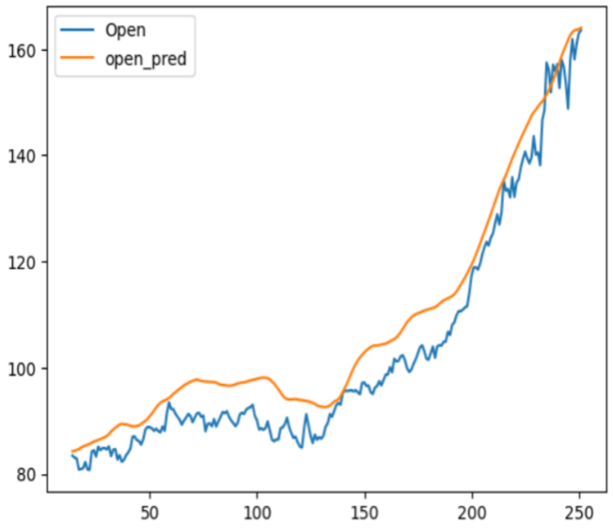
1. LSTM Model Building: Constructed an LSTM model with 3 layers using a specified window size. For the project, window sizes of 5 and 14 days were used.

2. Training: Trained the LSTM model using the historical stock prices.

3. Prediction: Used the trained model to predict future stock prices.

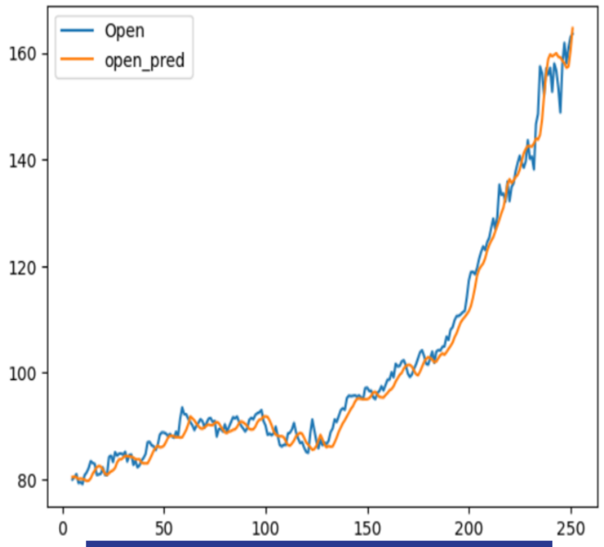
4. Performance Evaluation: Assessed the model's performance using metrics like accuracy and Root Mean Square Error (RMSE).

LSTM
Window size = 14



Accuracy : 92.03
Test RMSE: 0.06

LSTM
Window size = 5

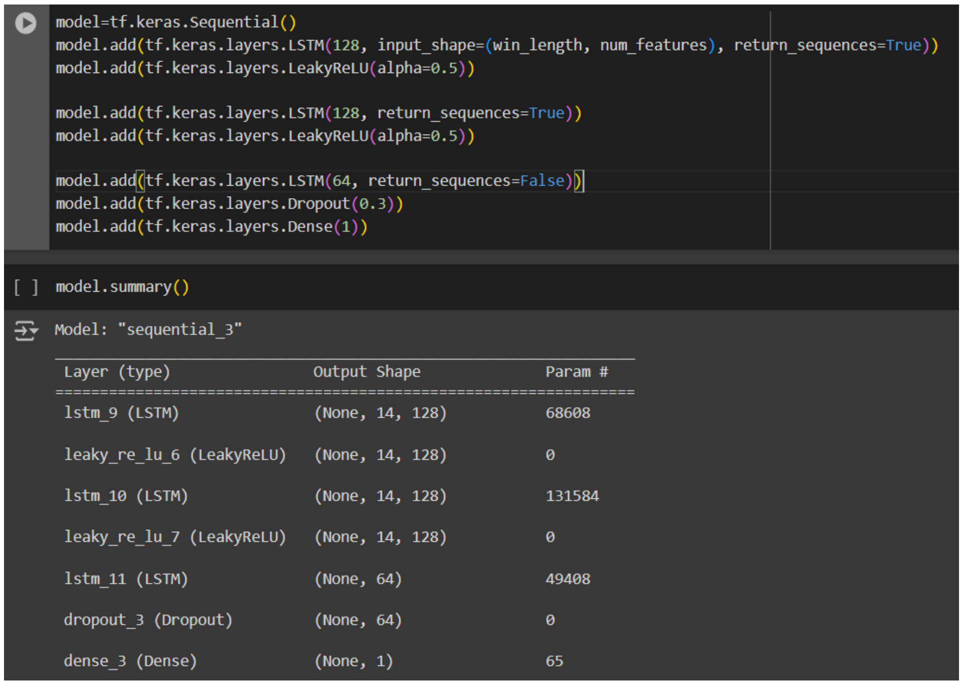


Accuracy : 98.6
Test RMSE: 0.02

Documentation of R Code for LSTM:

Code link:

https://colab.research.google.com/drive/13lFl7oK3HtcVutTSA_VwHXhEqkVoDyGz?usp=sharing



```
model=tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape=(win_length, num_features), return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))

model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))

model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(1))
```

```
[ ] model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_9 (LSTM)               (None, 14, 128)           68608

 leaky_re_lu_6 (LeakyReLU)   (None, 14, 128)           0

 lstm_10 (LSTM)              (None, 14, 128)           131584

 leaky_re_lu_7 (LeakyReLU)   (None, 14, 128)           0

 lstm_11 (LSTM)              (None, 64)                49408

 dropout_3 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65
```

Figure 1: LSTM Code Snippet

3.2.2 RNN:

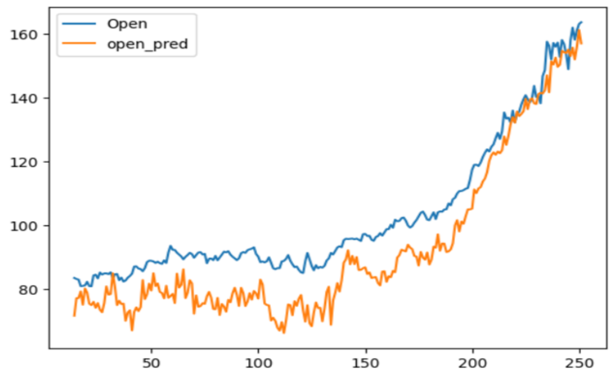
1. RNN Model Building: Constructed a SimpleRNN model with 2 layers using a specified window size. For the project, window sizes of 5 and 14 days were used.

2. Training: Trained the RNN model using the historical stock prices.

3. Prediction: Used the trained model to predict future stock prices.

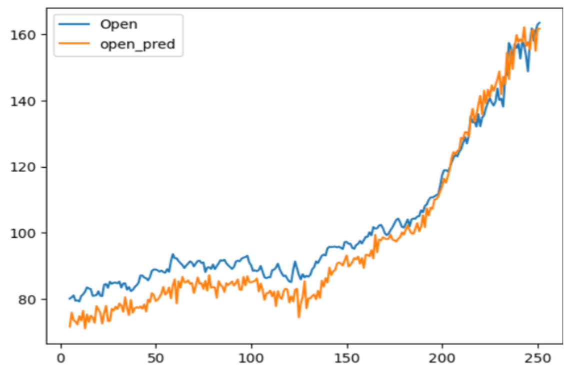
4. Performance Evaluation: Assessed the model's performance using metrics like accuracy & Root Mean Square Error (RMSE).

RNN
Window size = 14



Accuracy : 73.03
Test RMSE: 0.13

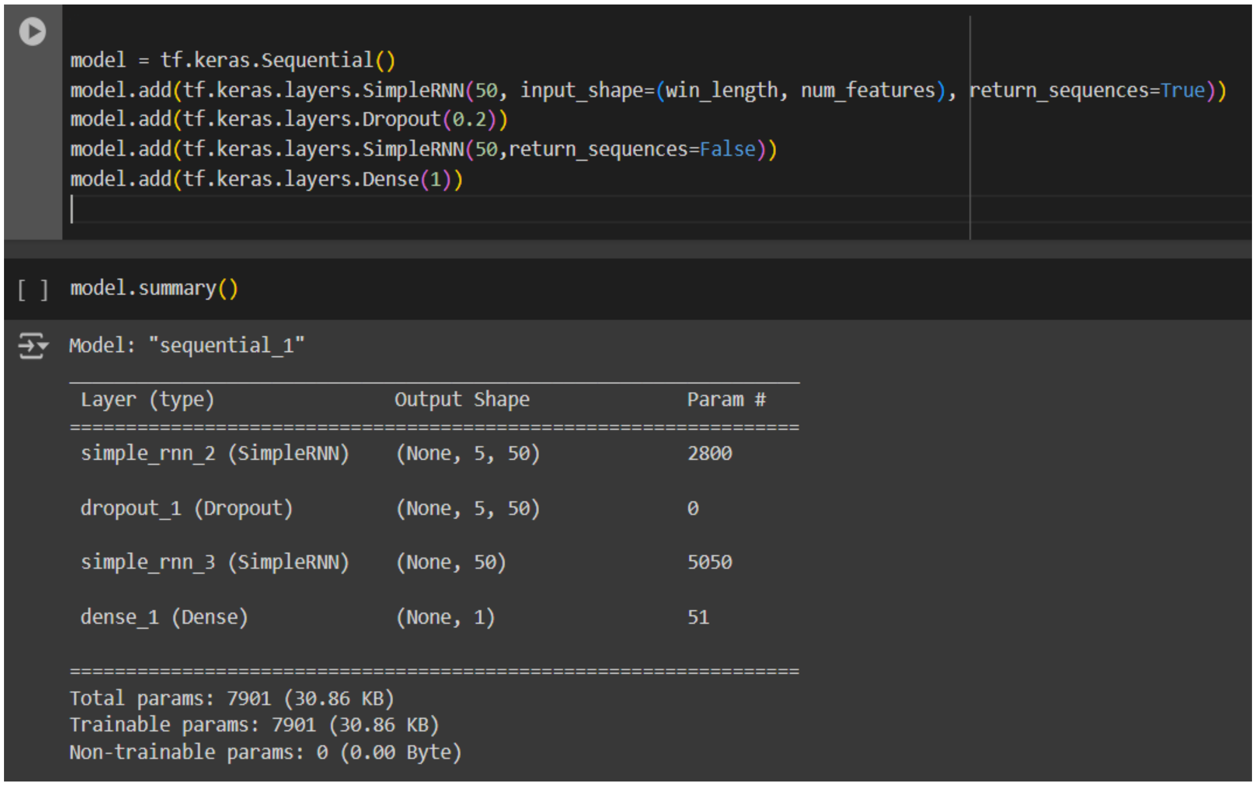
RNN
Window size = 5



Accuracy : 91.8
Test RMSE: 0.07

Documentation of R Code for RNN:

Code Link:
https://colab.research.google.com/drive/1bWs7ZyPYK9jSFKpYuQwRsxQlgtoTXr6D?usp=sharing



```
model = tf.keras.Sequential()
model.add(tf.keras.layers.SimpleRNN(50, input_shape=(win_length, num_features), return_sequences=True))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.SimpleRNN(50,return_sequences=False))
model.add(tf.keras.layers.Dense(1))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_2 (SimpleRNN)    (None, 5, 50)             2800

 dropout_1 (Dropout)         (None, 5, 50)             0

 simple_rnn_3 (SimpleRNN)    (None, 50)                5050

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 7901 (30.86 KB)
Trainable params: 7901 (30.86 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 2: RNN Code Snippet

4. Bibliography

1. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

2. Olah, C. (2015). Understanding LSTM Networks. [online] colah's blog. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3. Brownlee, J. (2016). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

4. Yahoo Finance. General Electric Company (GE) Stock Historical Prices & Data. Available at: https://finance.yahoo.com/quote/GE/history/