Shahid Beheshti University
Department of Computer Science, Faculty of Mathematical Sciences

Master Student of Computer Science , Data Mining

Stock Prediction Using Recurrent Neural Networks LSTM & Bi-LSTM

Tahsin Ilkhas Zadeh 400422034 Zeinab Khosravi 99422067

NEURAL NETWORKS COURSE, Dr.Katan Foroush

Contents

Introduction

Stock Prediction Using Neural Network

Machine Learning became very useful to the Stock Market Forecasting over the last years, and today, many investment companies are using Machine Learning to make decisions in the Stock Market.

The successful prediction of a stock's future cost could return noteworthy benefit.

Stock Prediction Using Neural Network

- Different types of RNNs are taken in forecasting stock trend in the previous years.

- In this approach, a stock price prediction method is proposed with Recurrent Neural Networks (RNN) models.

Datasets

The datasets used to fit to models:

Apple dataset

This dataset contains Apple's (AAPL) stock data for the last 10 years (from 2010 to date) reached from Nasdaq

There are 2515 rows and 5 columns in apple Historical Quotes dataset

Datasets

The datasets used to fit to models:

Google dataset

This dataset contains Google's stock data for the last years (to 2017)

There are 1258 rows and 5 columns in Google_Stock_Price_Train dataset & 20 rows and 5 columns in Google_Stock_Price_Test dataset

Datasets

Since the data is sent as a list of candlesticks, it has 6 values:

Date	• The date represents the transactions date
Open	• The stock price at the time of opening sales
Close/Last	• The stock price at the time of Closing sales
High	• The highest price of stock during sale
Low	• The lowest price during sale
Volume	• the Volume indicates total volume of stock being traded each day

Datasets

The Apple dataset head lines:

Date	Close/Last	Volume	Open	High	Low
2022-06-22	$135.35	73409230	$134.79	$137.76	$133.91
2022-06-21	$135.87	81000490	$133.42	$137.06	$133.32
2022-06-17	$131.56	134520300	$130.065	$133.079	$129.81
2022-06-16	$130.06	107961500	$132.08	$132.39	$129.04
2022-06-15	$135.43	91532970	$134.29	$137.34	$132.16

Datasets

The google dataset head lines:

Date	Open	High	Low	Close	Volume
2012-01-03	325.25	332.83	324.97	663.59	7,380,500
2012-01-04	331.27	333.87	329.08	666.45	5,749,400
2012-01-05	329.83	330.75	326.89	657.21	6,590,300
2012-01-06	328.34	328.77	323.68	648.24	5,405,900
2012-01-09	322.04	322.29	309.46	620.76	11,688,800

Problem statement

Understanding the Problem

forecast stock price is about use Sequential data, where the past data matters and influence directly in our predictions.

We are going to give to our model a Stock Price history of datasets, it will work on this data to identify patterns and make a lot of calculus to, and give us the predictions.

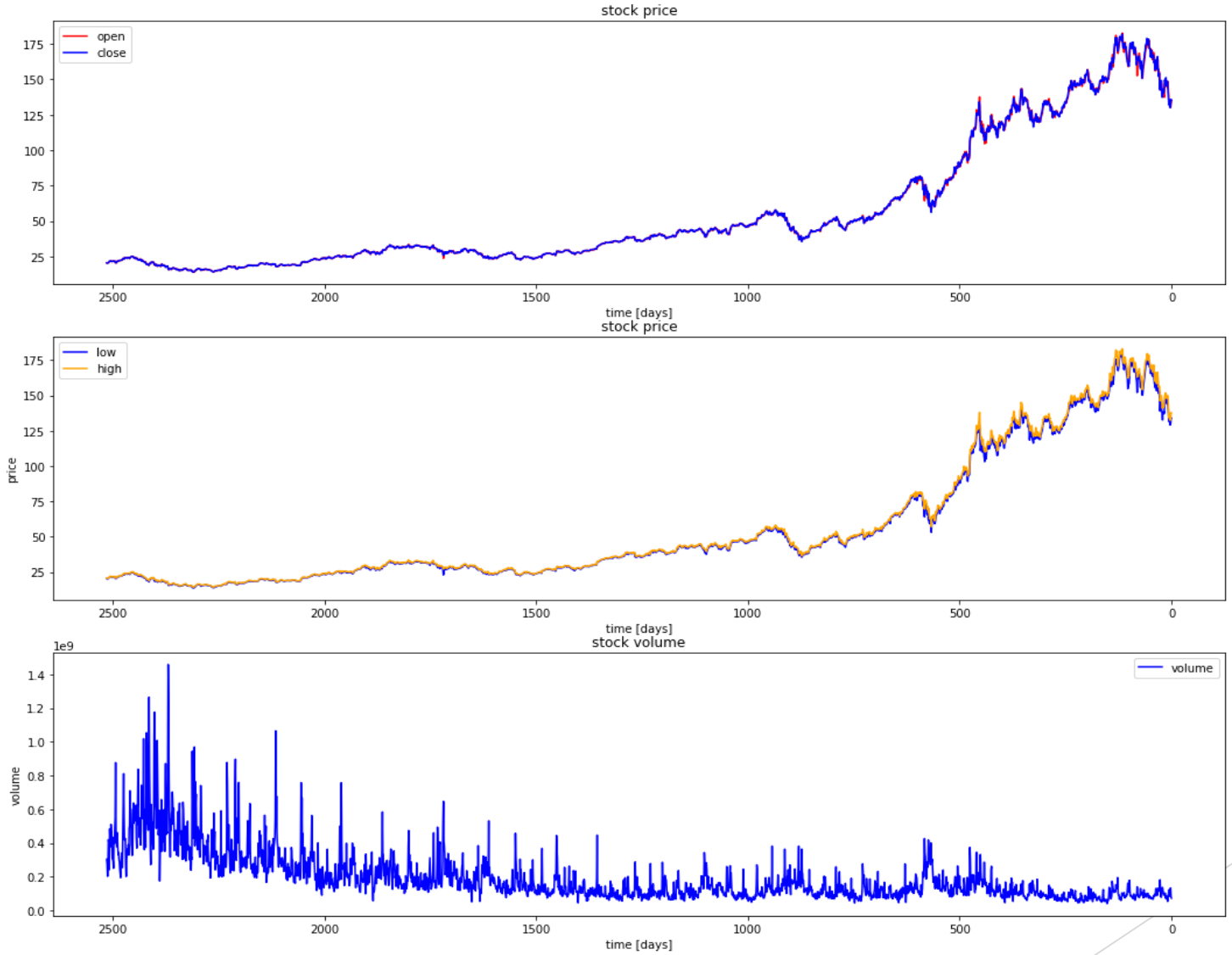
Here we will predict Closing price (Y) using past closing prices (as input vector X)

Understanding the Problem

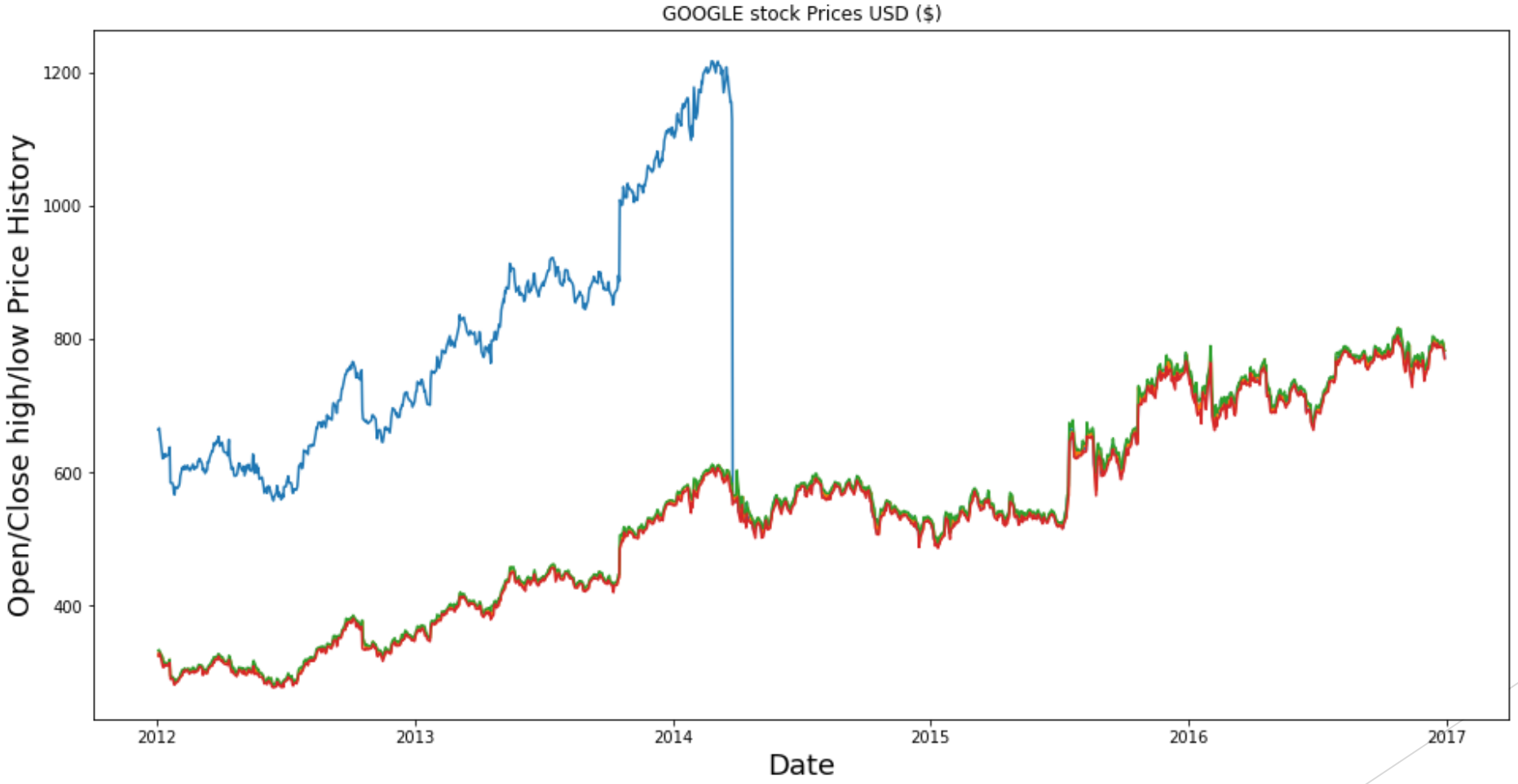
- In addition to the start and end values given by the start and end dates, values such as interval and period can be given.

Understanding the Problem

Apple open/close and high/low graph



Understanding the Problem



Understanding the Problem

The time interval inside each candle, that is from open to close, is a few minutes, called the **interval**.

With the help of the **period** value, it is possible not to give the start, end dates, and for example request the data of the last 24 hours.

The risk of the stock

We are going to analyze the risk of the stock.

In order to predict the risk, we need to inspect the daily changes of the stock, and not just its absolute value

Then find the daily Closing value of the stock by model prediction

Moving Average (MA) Lines

Moving averages are one of the most commonly used technical indicators in stock, futures and forex trading.

Market analysts and traders use moving averages to help identify trends in price fluctuations, smoothing out the noise and short-lived spikes (from news and earnings announcements, for example) for individual securities or indexes.

Most Commonly-Used Moving Averages

The 5 -, 10 -, 20 - and 50 - day moving averages are often used to spot near-term trend changes.

Changes in direction by these shorter-term moving averages are watched as possible early clues to longer-term trend changes.

Crossovers of the 50-day moving average by either the 10-day or 20-day moving average are regarded as significant.

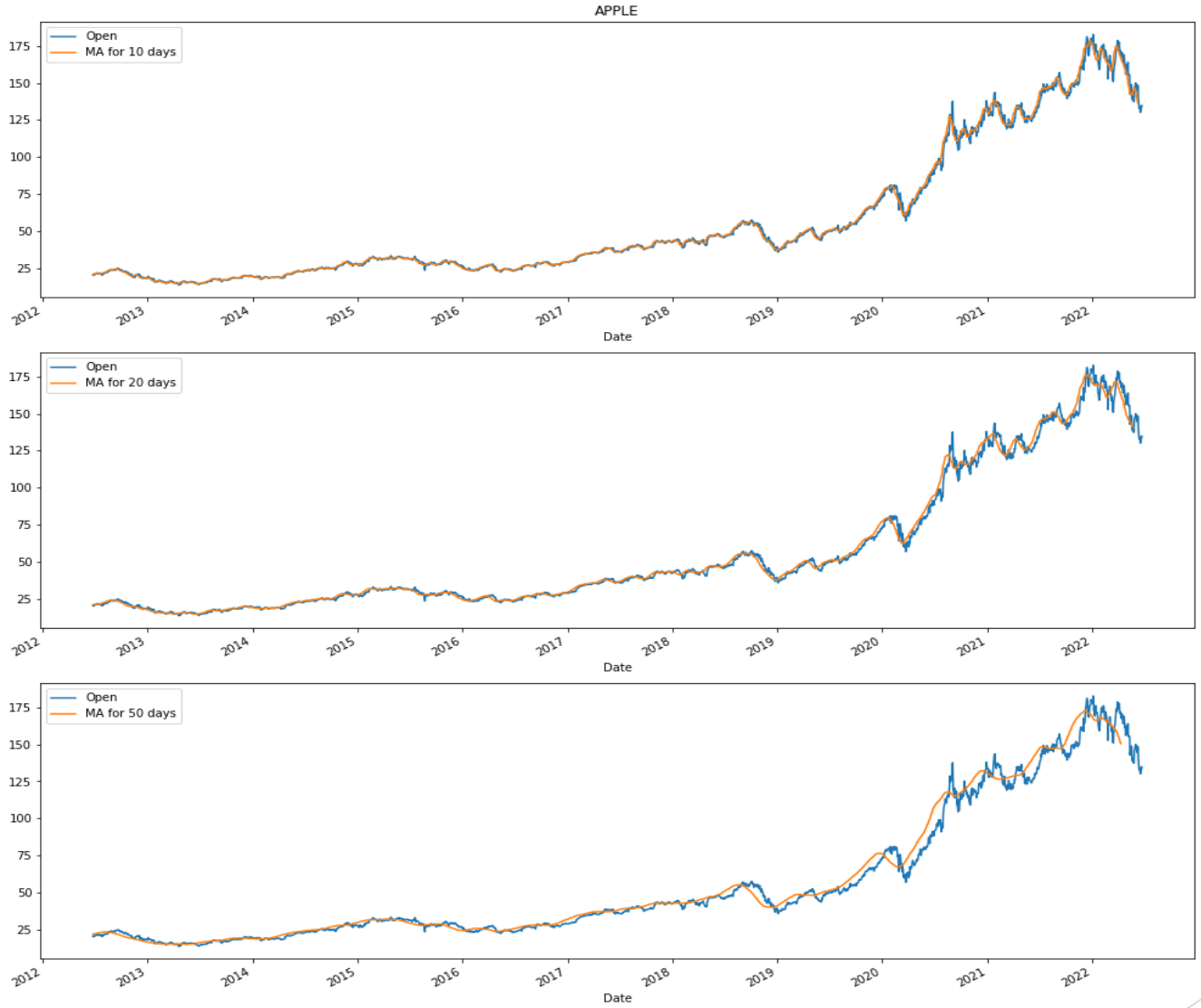
The 10-day moving average plotted on an hourly chart is frequently used to guide traders in intraday trading.

Types of Moving Averages

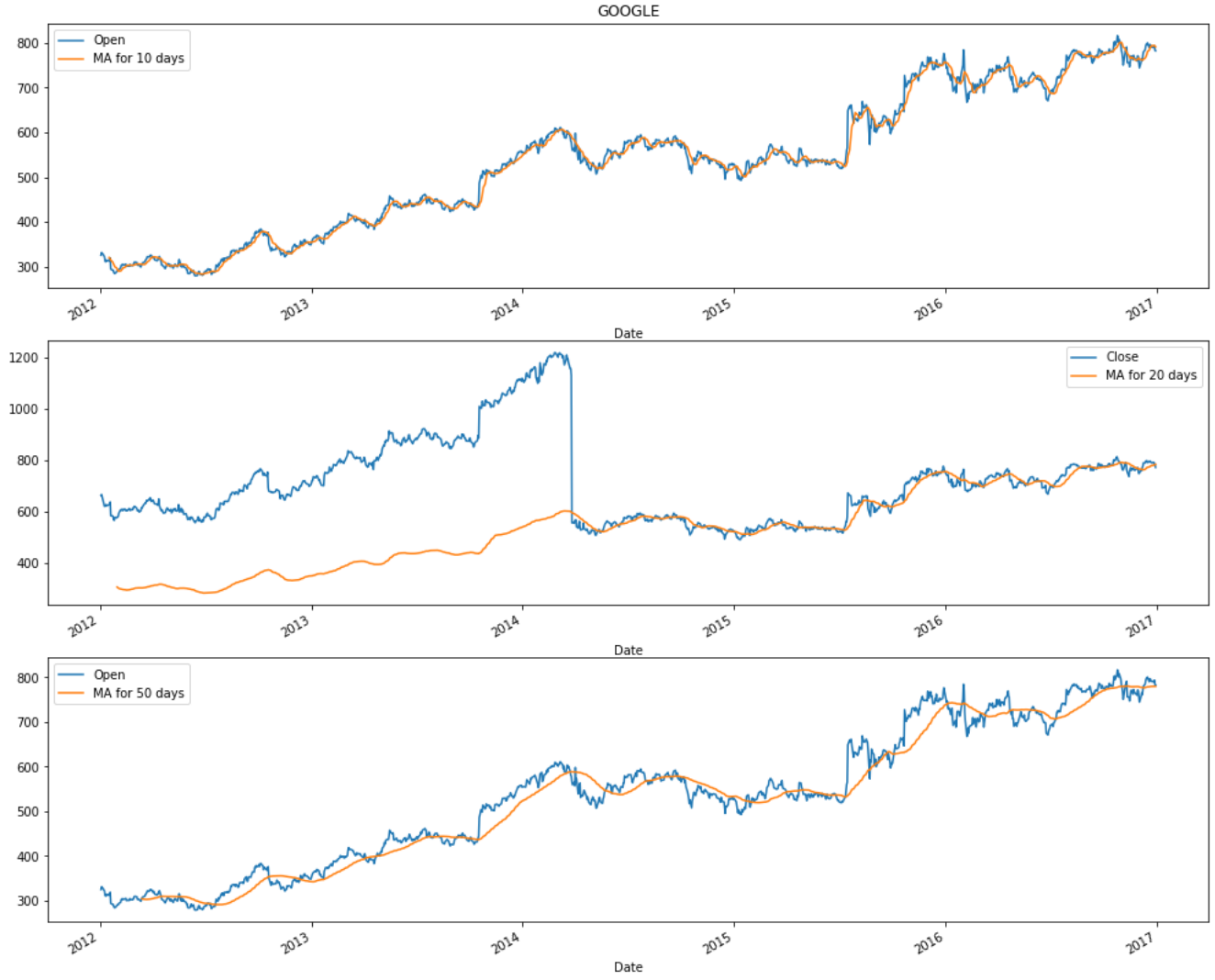
Most moving averages are some form of the simple moving average (SMA), which is the average price over a given time period

As a general guideline, **if the price is above a moving average, the trend is up. If the price is below a moving average, the trend is down.**

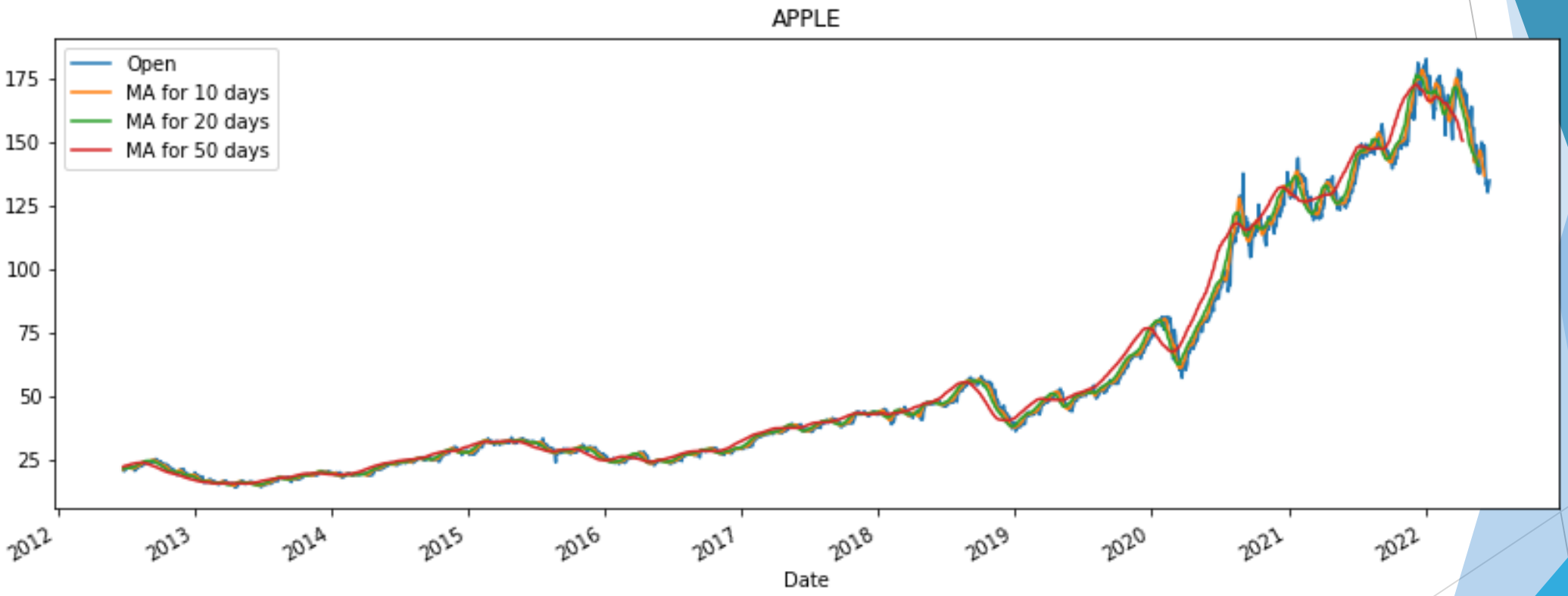
Moving Averages for Apple dataset



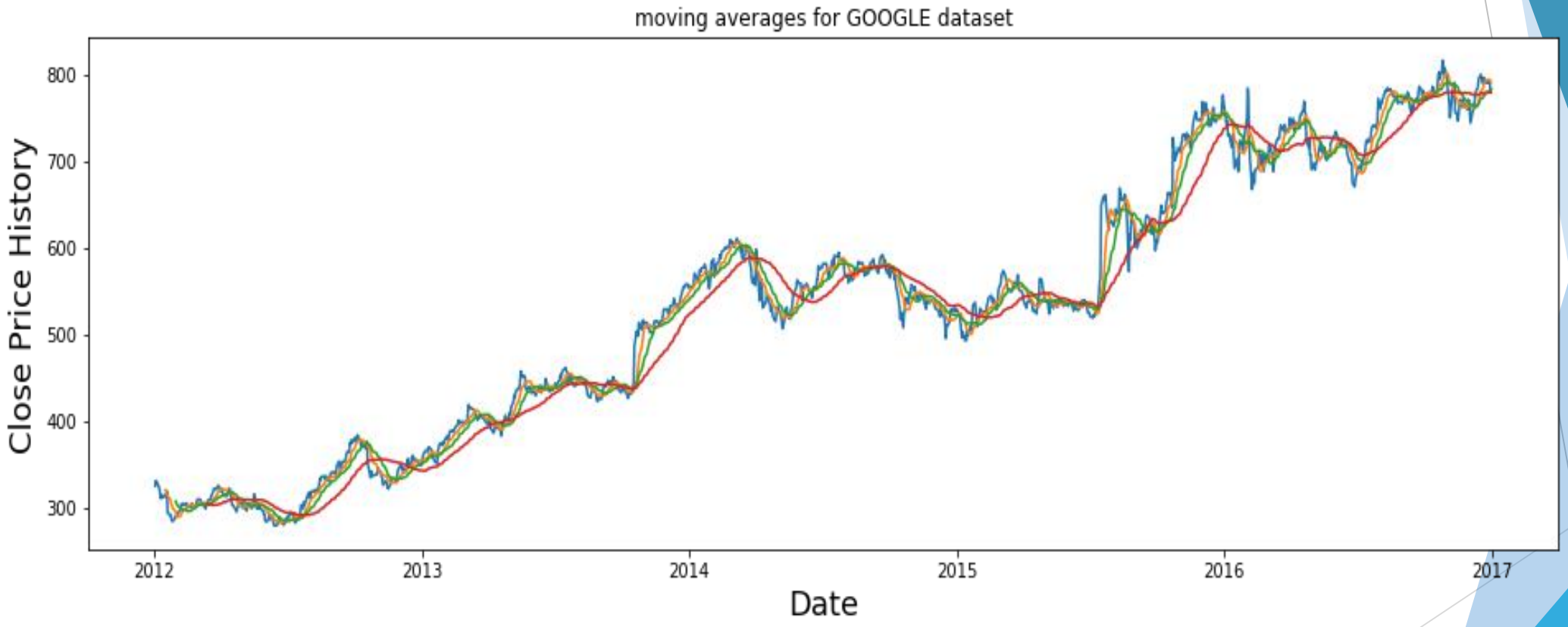
Moving Averages for Google dataset



Moving Averages for Apple dataset



Moving Averages for Google dataset



Models

RNN models

RNN Is a type of artificial neural network where connections between nodes form a sequence. This allows temporal dynamic behavior for time sequence.

Long Short Term Memory models

From the simulation results, it can be noted that using RNN models i.e. LSTM, and BI-LSTM with proper hyper-parameter tuning, can forecast future stock trend with high accuracy.

Recurrent Neural Network (RNN) models

we have tested 2 kind of models for each dataset:

1. Long Short Term Memory (LSTM) model

2. Bidirectional long short term memory, Bi-LSTM model

LSTM model

LSTM Models

There are 3 types of vanilla recurrent neural network:

- the simple (RNN)
- gated recurrent unit (GRU)
- long short term memory unit (LSTM).

LSTM Models

In regular RNN, the problem frequently occurs when connecting previous information to new information

Long short term memory networks, usually called LSTM

are a special kind of RNN.

They were introduced to avoid the

long-term dependency problem.

LSTM Models

Recurrent networks have an **internal state** that can represent context information. In this model we call it **history**.

The history **keeps information about past inputs** for an amount of time.

LSTM Models

Only the hidden state is passed into the output layer. The memory cell is internal.

LSTMs have three types of gates that control the flow of information.

- input gates
- forget gates
- output gates

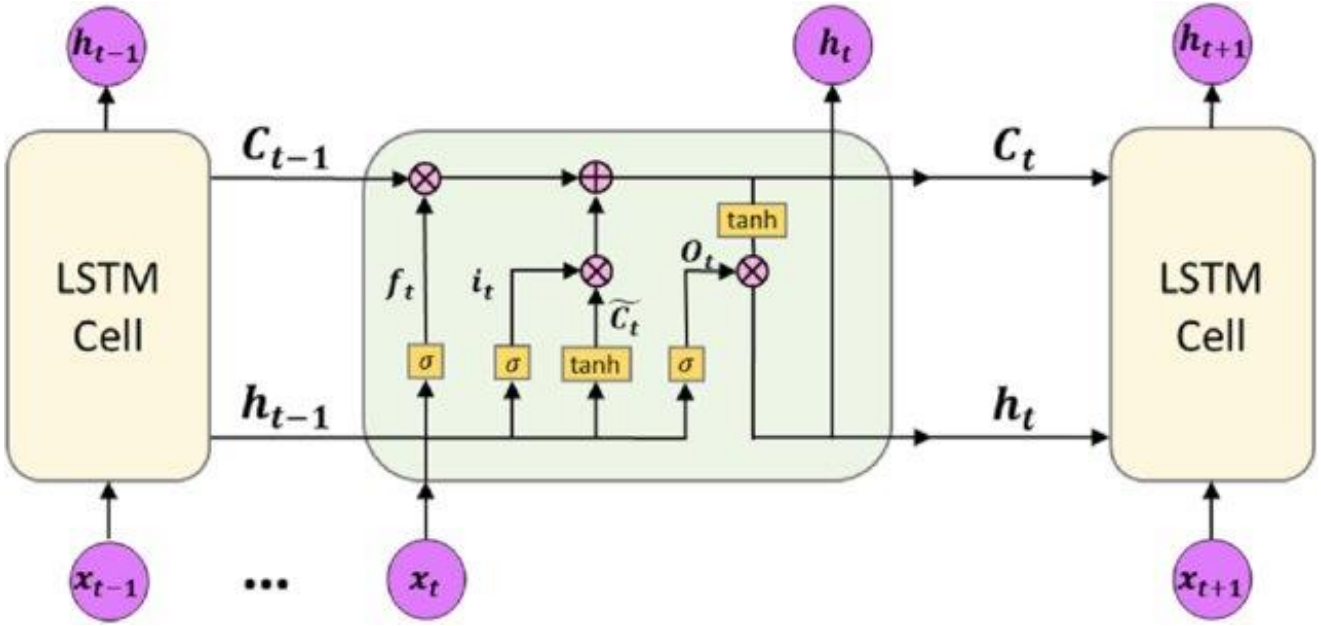
LSTM Models

The hidden layer output of LSTM includes the hidden state and the memory cell

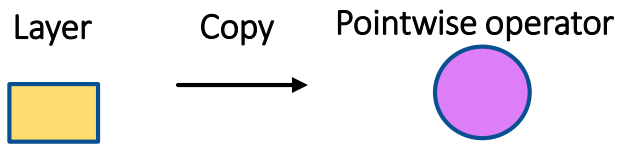
LSTMs can alleviate vanishing and exploding gradients.

LSTM Model formulation

A LSTM layer structure for a time step, with a detailed calculation illustration shown in the LSTM cell at time step t.



$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$



LSTM Model formulation

The parameters in a LSTM unit is associated with the functions involved in calculation

A LSTM Cell has four functional units:

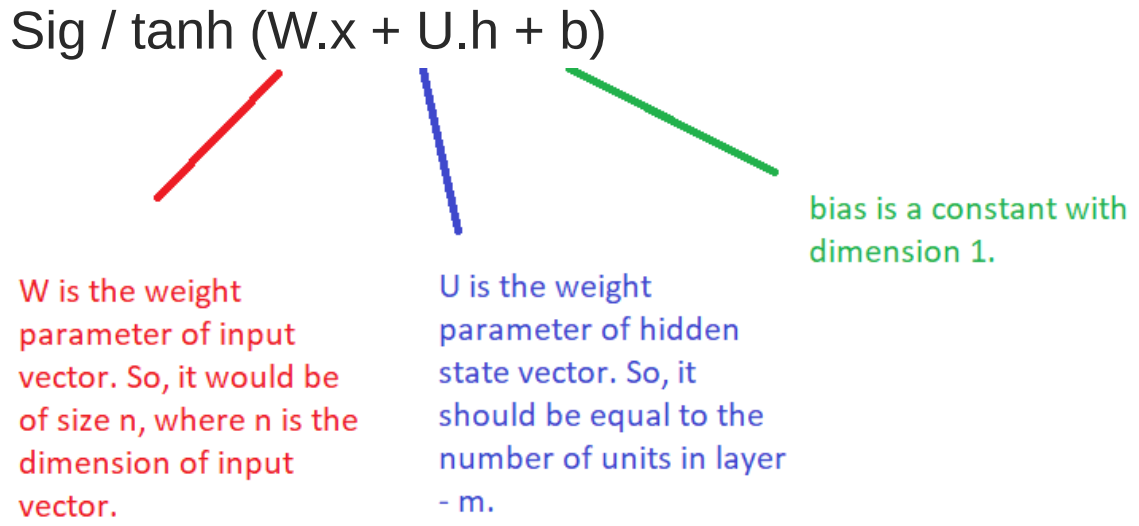
3 Sigmoids (f, i, o functions below) | 1 Tanh (c function below)

We can generalize all these 4 functions :

Sig / tanh (W.x + U.h + b)

Where, W, U, b are the parameters and x is the input and h is the hidden layer values calculated using above parameters.

LSTM Model formulation



Total number of parameters for a function within LSTM unit would be n+m+1

where m is dimension of input vector, m is number of LSTM units in a layer and 1 is bias parameter

LSTM unit has 4 functions (3 sigmoid + 1 tanh).

So, total number of params will be 4*(n+m+1)

If LSTM has m units, total number of params will be 4*(n+m+1)*m

LSTM Model formulation

Formula for calculating current state:

$$h_t = f(h_{t-1}, x_t)$$

h_t -> current state

h_{t-1} -> previous state

x_t -> input state

Formula for applying Activation function(tanh):

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

w_{hh} -> weight at recurrent neuron

w_{xh} -> weight at input neuron

Formula for calculating output:

$$y_t = W_{hy}h_t$$

Y_t -> output

W_{hy} -> weight at output layer

LSTM Model No.1

```
# Build the LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape = (x_train.shape[1],1)))
model.add(Dropout(0.2))
model.add(LSTM(64 , return_sequences=True,activation='tanh'))
model.add(Dropout(0.2))
model.add(LSTM(32, return_sequences=False))
model.add(Dense(1,activation='linear'))
model.summary()

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error',metrics=['accuracy'])
```

LSTM Model No.1

1. We defined a sequential model which consists of a linear stack of layers.

2. Add a LSTM layer by 100 network units. setting return_sequence = true means the output of the layer will be another sequence of the same length.

3. Adding each dropout layer to prevent overfitting. when defining drop out layers we specify 0.2,meaning that 20% of the layers will be dropped.

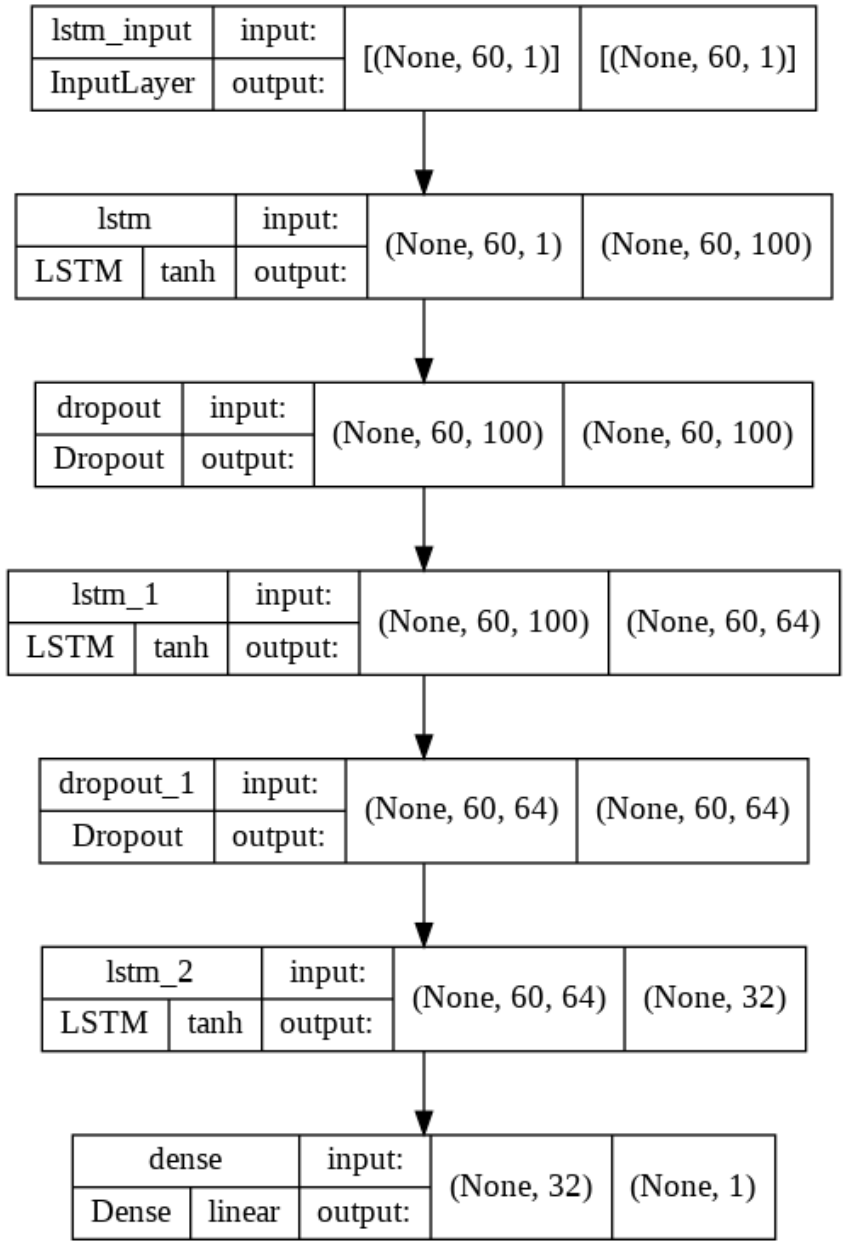
4. Return-sequences is True as we need to add another LSTM layer after the current one.

5. add another LSTM layer with 100 units and return_sequence = false means that this time just the last output returns in the output unit.so, the last LSTM layer, Return-sequences is False as we will not add more LSTM layers.

6. adding a densely connected layer that specifies the output of 1 network unit. The output dimension is 1 since we are predicting 1 price each time.

LSTM Model No.1

Total params: 95,489
Trainable params: 95,489
Non-trainable params: 0



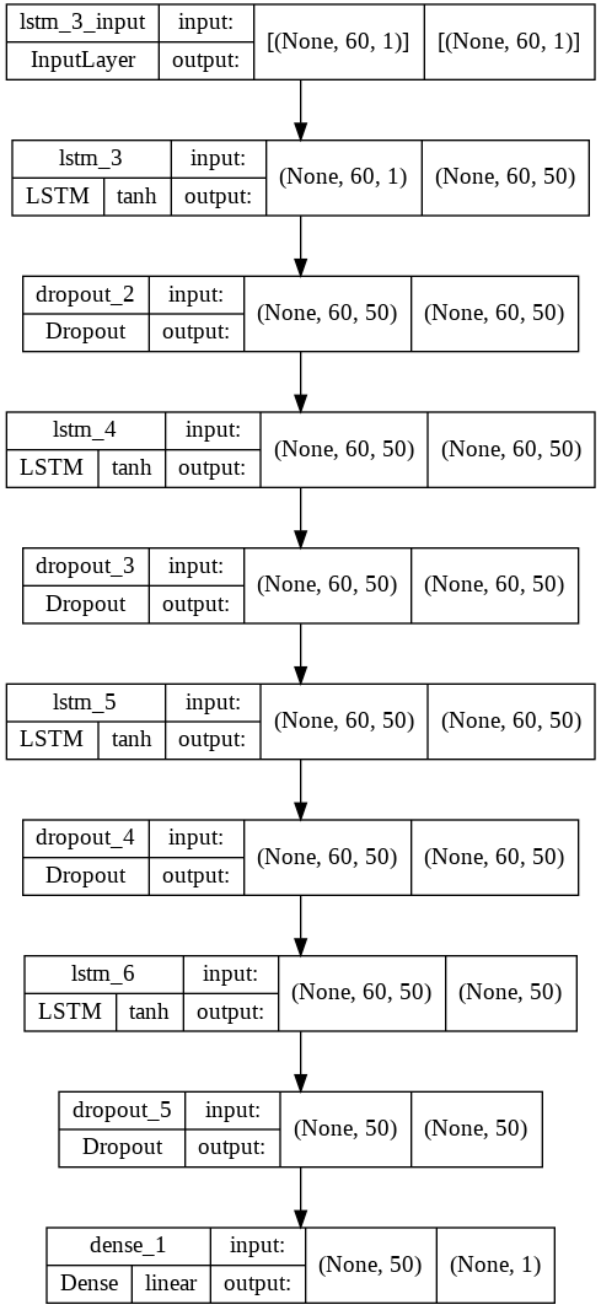
LSTM Model NO.2

We changed model layer architecture and parameters to compare the results and fitted it over same condition on both datasets. Thus, the second model is as below:

```
# Building Model:
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.LSTM(units=50, return_sequences=True,input_shape=(x_train.shape[1], 1)))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.LSTM(units=50, return_sequences=True))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.LSTM(units=50, return_sequences=True))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.LSTM(units=50))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(units=1))
model.summary()
```

LSTM Model NO.2

Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0



Bi-LSTM model

BI-LSTM Model

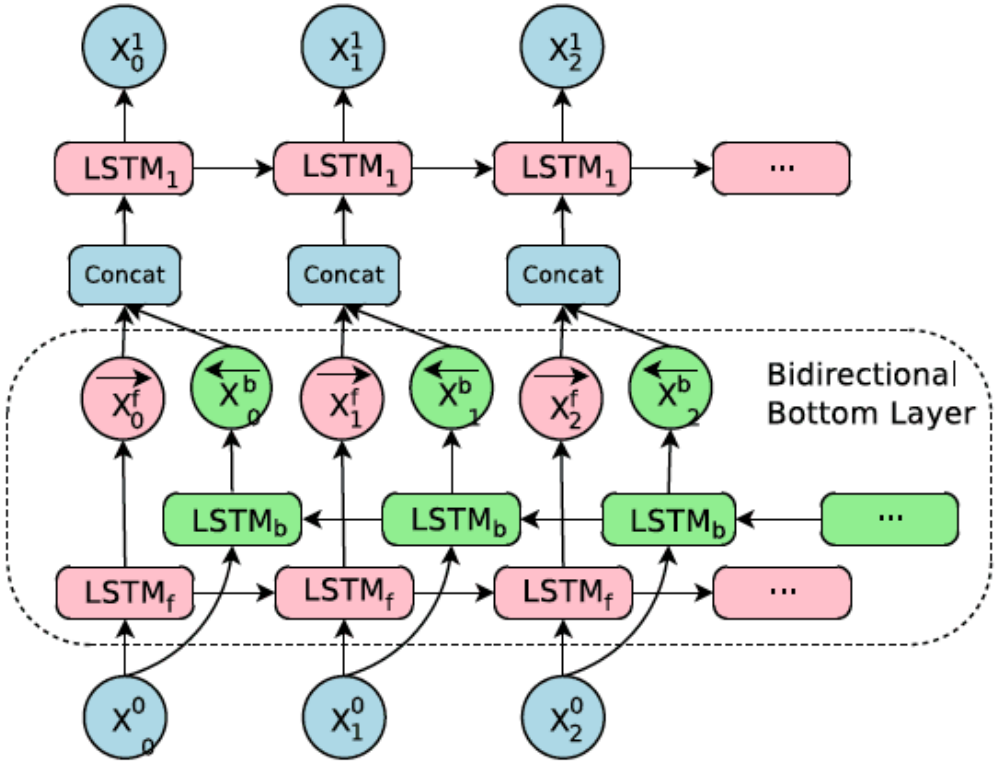
Bidirectional long-short term memory (bi-lstm)

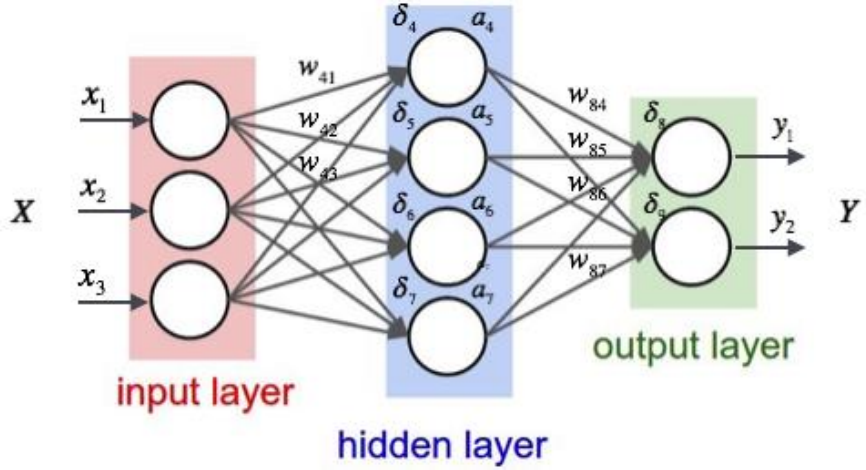
the process of making any neural network to have the sequence information in both directions

backwards (future to past)

Or forward (past to future)

BI-LSTM Model



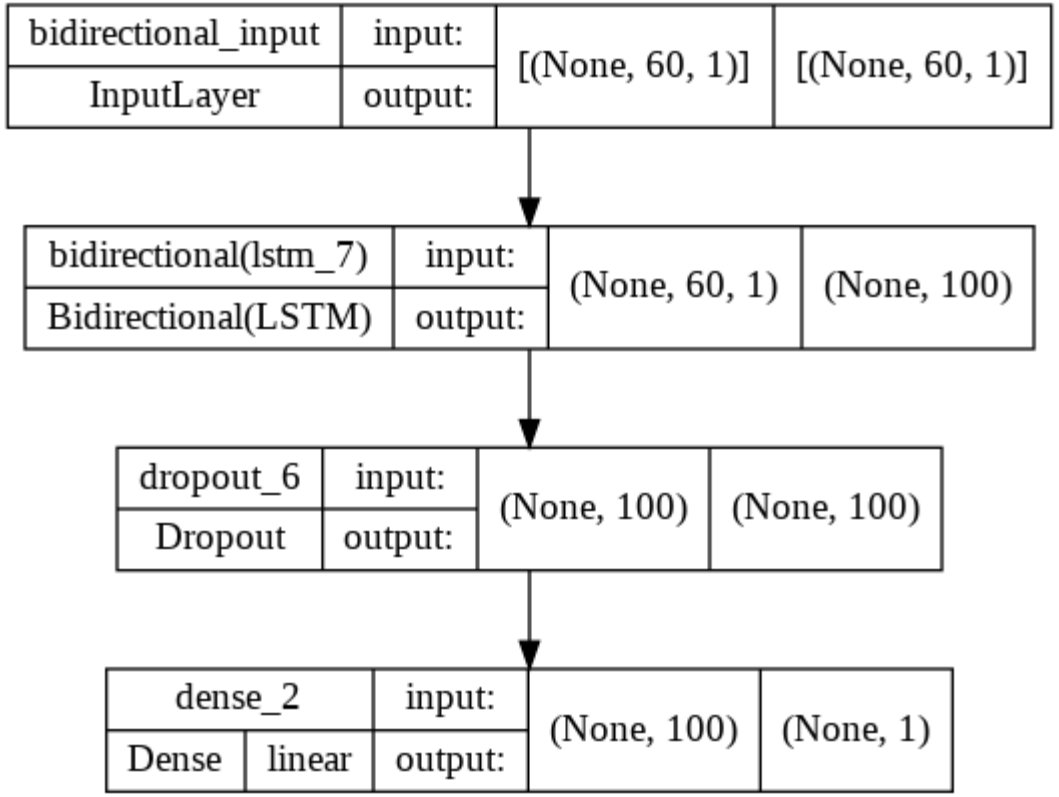


BI-LSTM Model

- The bidirectional model used for both datasets:

```
# define model
model = Sequential()
model.add(Bidirectional(LSTM(50, activation='tanh'), input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.build((x_train.shape[1],1))
model.add(Dense(1))
```

BI-LSTM Model



Model Evaluation

we evaluate our model with test set and then apply following metrics to examine the performance of the model:

- mean squared error (MSE)
- Root mean squared error(RMSE)
- R_squared

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

$$R^2 = 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}},$$

$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

Model Evaluation

MSE is beneficial when the spread of prediction values is significant and larger values must be punished.

When the spread is important and bigger values need to be penalized, RMSE (NRMSE) is also useful. When compared to MSE, RMSE is easier to interpret because the RMSE number is on the same scale as the projected values.

When the relation between the forecasted and actual value is to be known then R^2 is used.

The procedure

The procedure

- now we choose a specific stock feature
- drop feature: volume
- normalize stock data
- Normalization is a very important part for any Recurrent Neural Network. For our LSTM model, normalization will play an important role. Normalization usin g MinMaxScaler will bring our entire datapoints between a minimum and a ma ximum value. For this purpose, we will use the values (0,1).
- then create train, validation and test data sets

The procedure

At First call the libraries and read the dataset

then normalize the data

get the moving averages of the stocks

get their weight, which changes by a few percent.

Find the changes and match the data to the model.

In the next step, using the min max scaler, we cut the data in the range of zero and one.

In the model, we use the normalization batch, train it, and then validate it.

We use this model to make predictions and store the predictions.

Finally, we use the inverse scaler to inverse the scales as before,

obtain the values of the loss and the accuracy of the model.

evaluate model with MSE , RMSE and R2 metrics to examine performance of the model.

The model setups

define the shape of the input dataset:

- Number of time steps, the number of lags in the data frames we set in Step #2.

define the number of units, and the number of hyper parameters of the LSTM.

- The higher number, more parameters in the model.

The model setups

Define the **dropout** rate, which is used to prevent overfitting.

Specify the output layer to have a linear activation function.

Then we also define the optimization function and the loss function.

- Tuning these hyper parameters to find the best option would be a better practice.

Apple Dataset Results

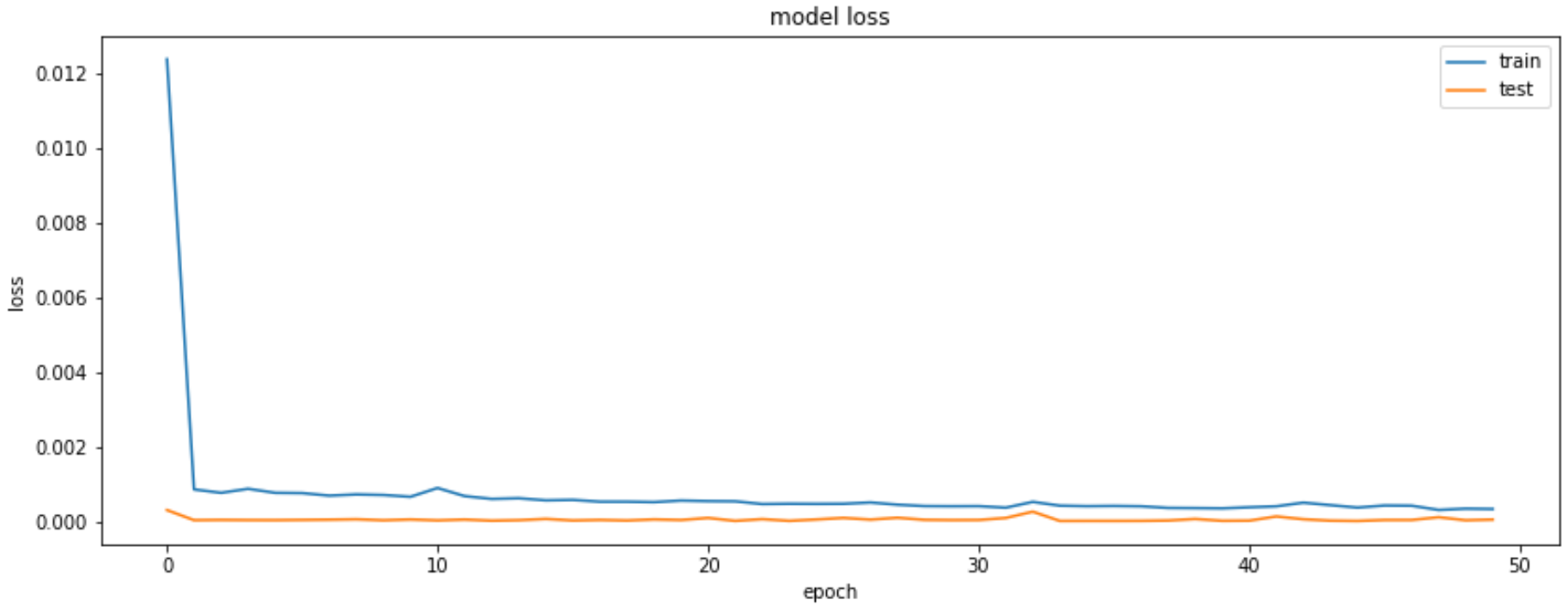
LSTM Model NO.1 for Apple dataset

We trained our model with the training data over bellow parameters :

- epochs = 50
- with a batch size = 32
- optimizer= Adam
- loss= mean squared error (MSE)
- Validation split=0.25
- Drop out = 0.6

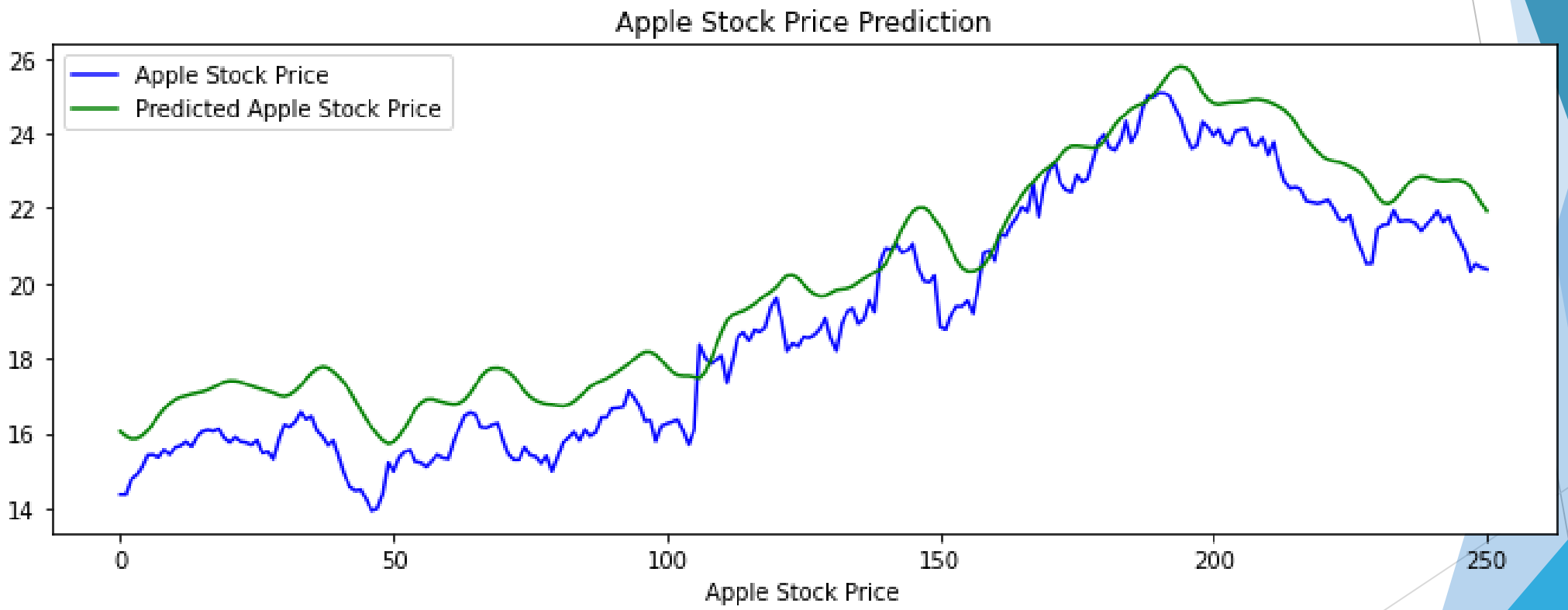
LSTM Model NO.1 for Apple dataset

- The Loss graph for LSTM model no.1



LSTM Model No.1 for Apple dataset

- The prediction

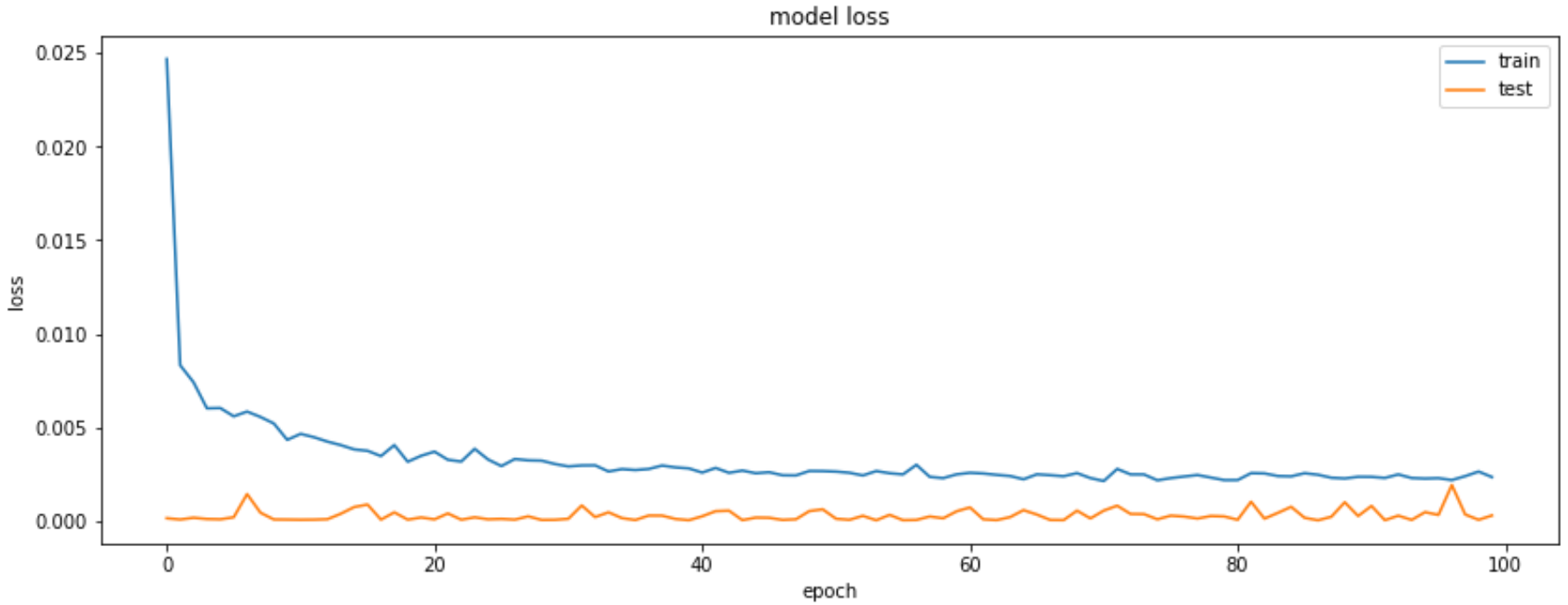


LSTM Model No.1 for Apple dataset

RMSE	• 1.1292269819160854
MSE	• 1.275153576687311
R2	• 0.8409089295587304
Accuracy	• 0.00088417332153767 35

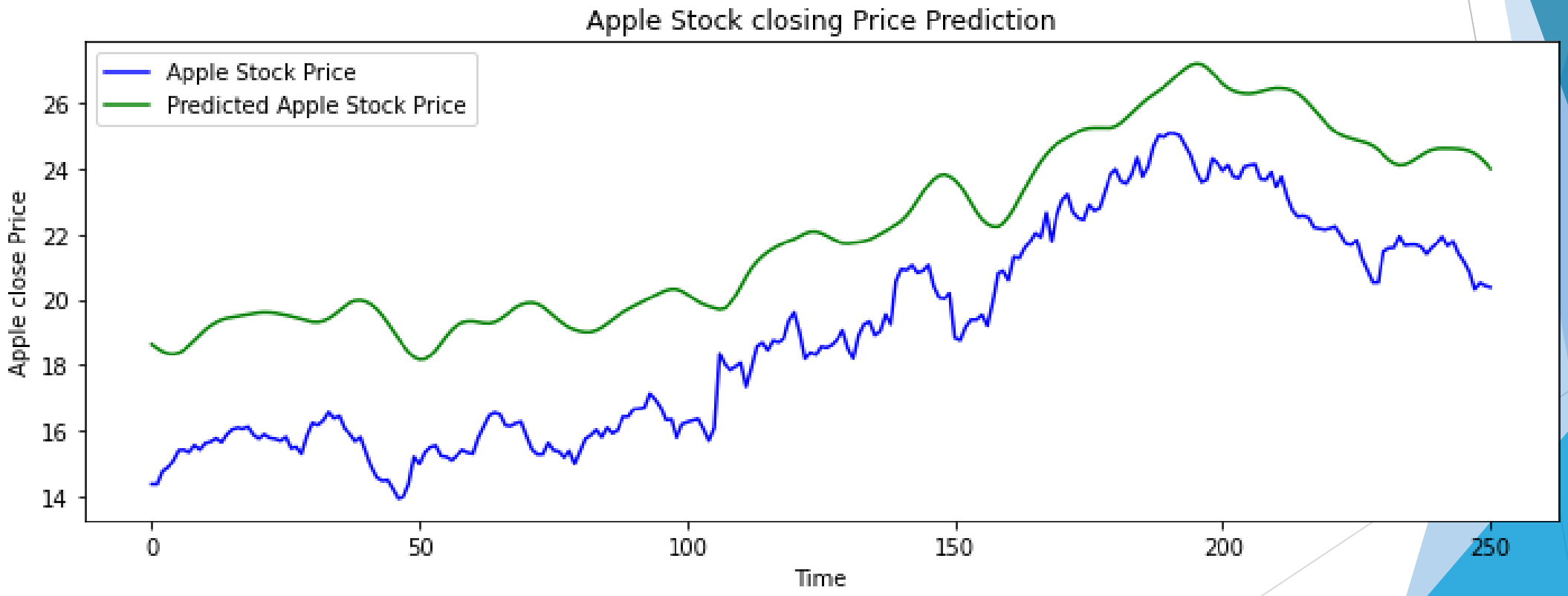
LSTM Model No.2 for Apple dataset

- The Loss graph for LSTM model no.2 over 100 epochs and batch size 32:



LSTM Model No.2 for Apple dataset

- The prediction



LSTM Model No.2 for Apple dataset

RMSE	• 3.120903760823406
MSE	• 9.740040284321678
R2	• 0.0006180498835880455
Accuracy	• 0.0006049606599844992

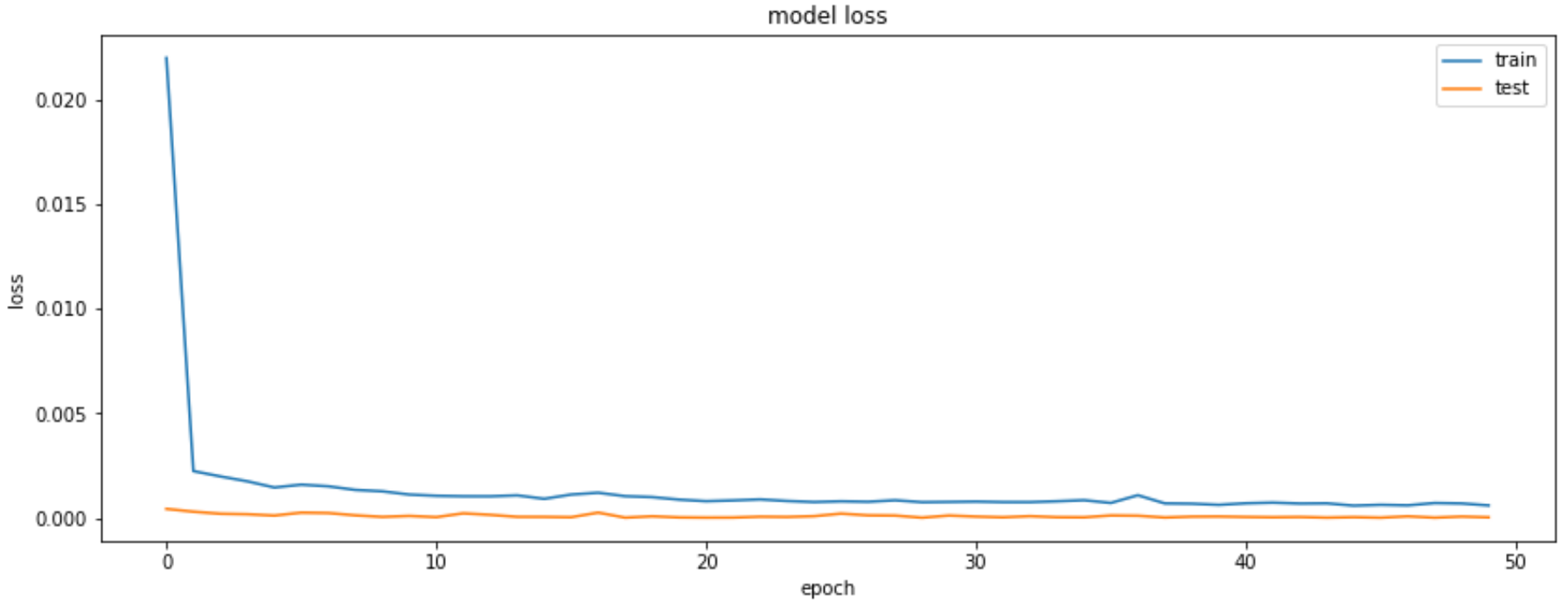
Bi-LSTM Model for Apple dataset

We trained our model with the training data over bellow parameters :

- epochs = 50
- with a batch size = 32
- optimizer= Adam
- loss= mean squared error (MSE)
- Validation split=0.2
- Drop out = 0.2

Bi-LSTM Model for Apple dataset

- The Loss graph



Bi-LSTM Model for Apple dataset

- The prediction

Bi-LSTM Model for Apple dataset

RMSE	• 0.39104942785012287
MSE	• 0.1529196550219084 6
R2	• 0.9340726278285816
Accuracy	• 0.0005672149709425867

Google Dataset Results

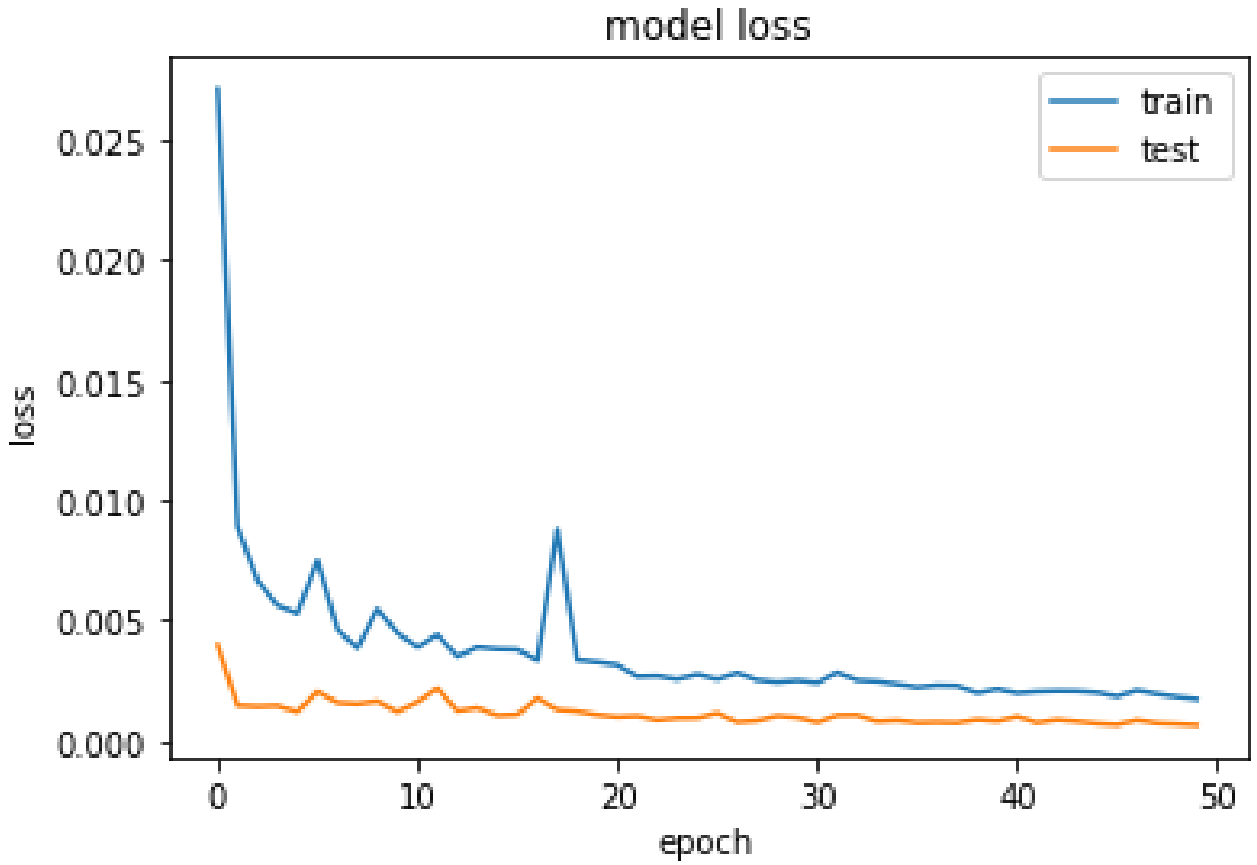
LSTM Model No.1 for Google dataset

We trained model no.1 with the training data over bellow parameters :

- epochs = 100
- with a batch size = 64
- optimizer= Adam
- loss= mean squared error (MSE)
- Validation split=0.10
- Drop out = 0.6

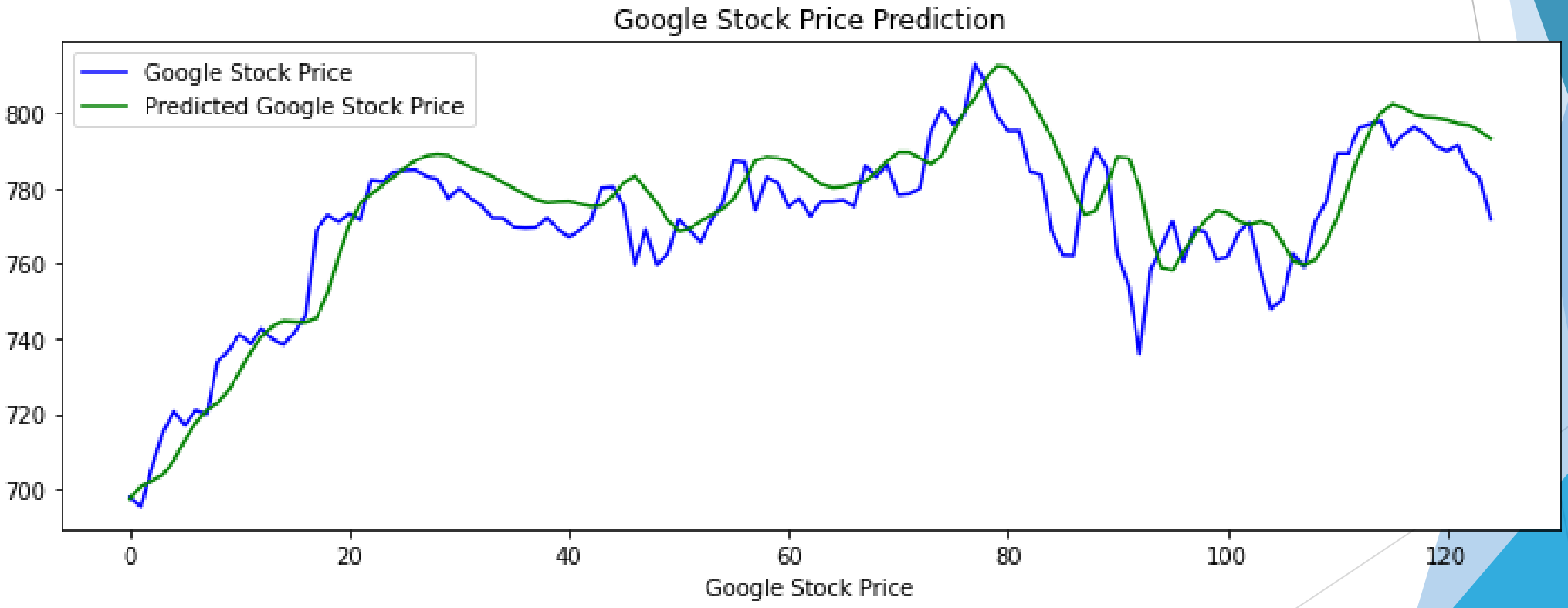
LSTM Model No.1 for Google dataset

- The Loss graph for LSTM model no.1



LSTM Model No.1 for Google dataset

- The prediction

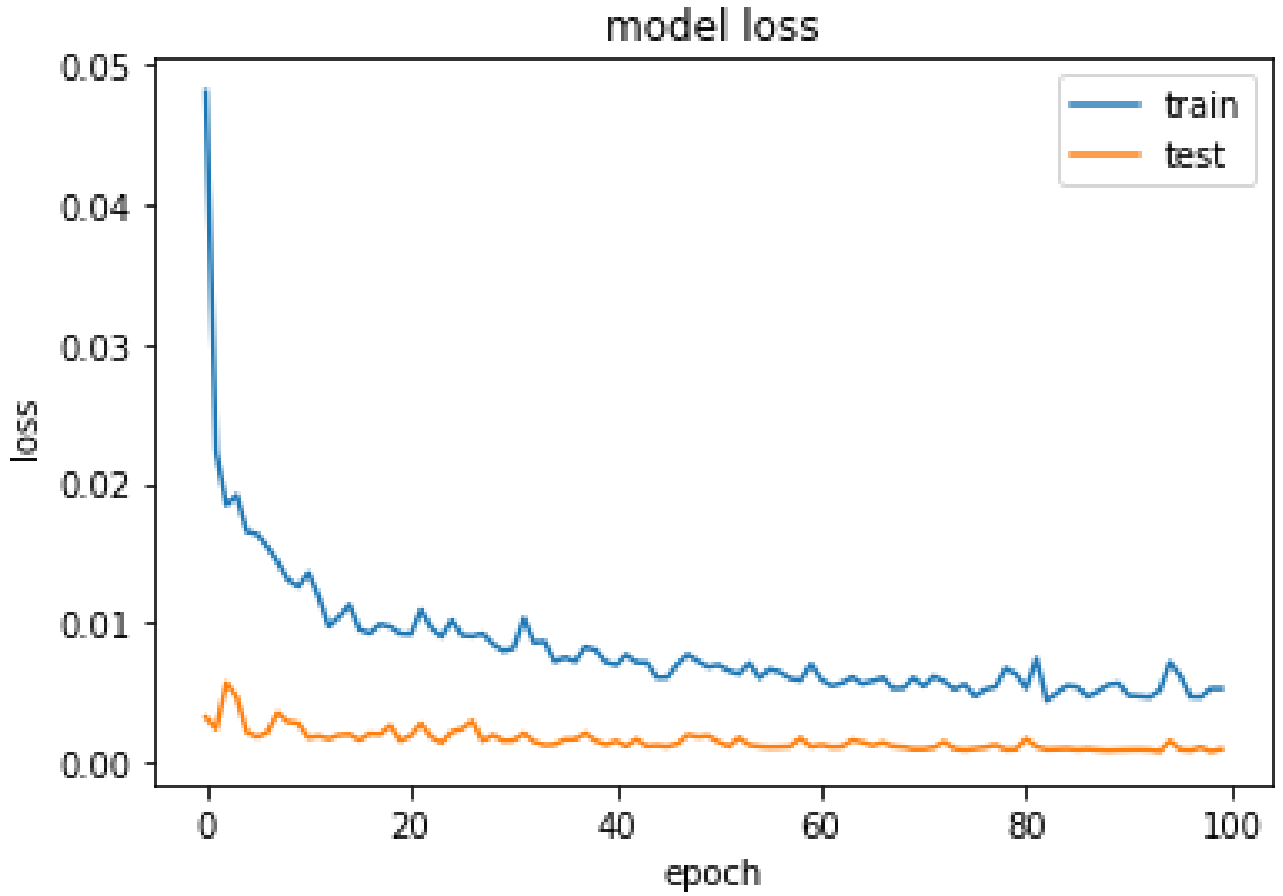


LSTM Model No.1 for Google dataset

RMSE	• 3.729775312500001
MSE	• 13.911223881734448
R2	• 0.7389681205217664
Accuracy	• 0.002487562131136656

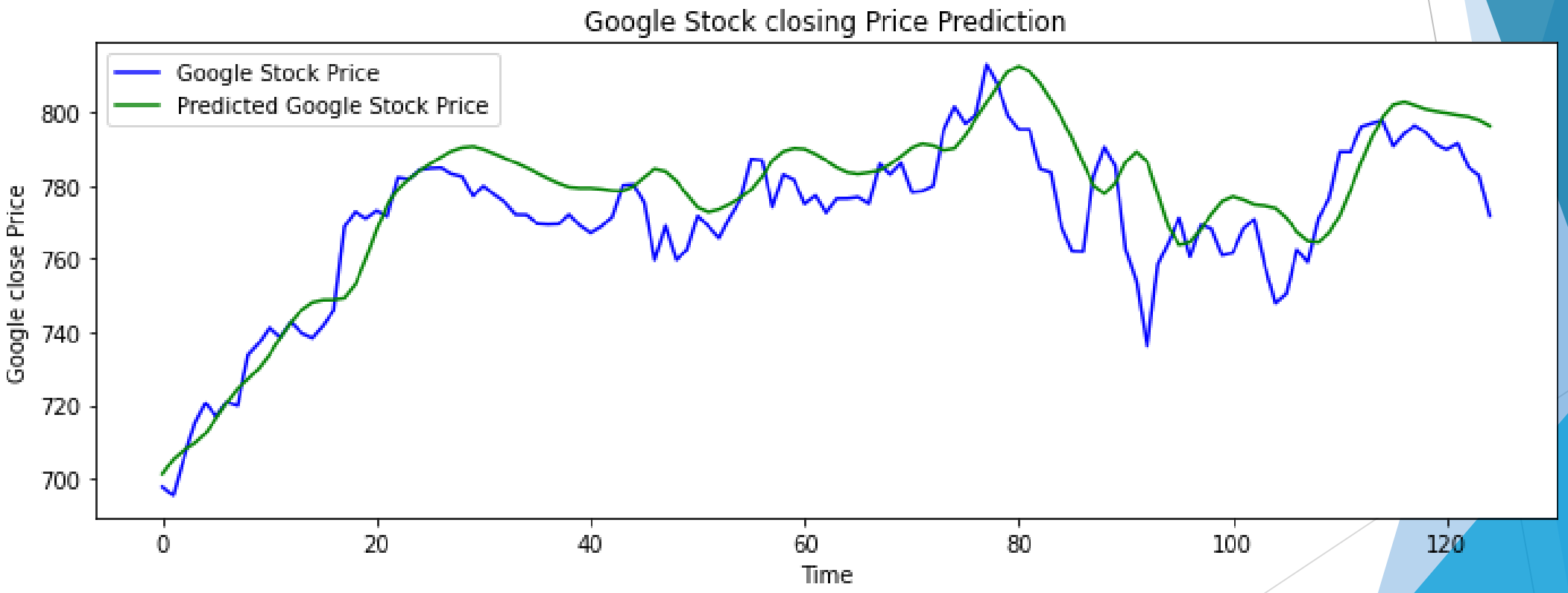
LSTM Model No.2 for Google dataset

- The Loss graph for LSTM model no.2



LSTM Model No.2 for Google dataset

- The prediction



LSTM Model No.2 for Google dataset

RMSE	• 6.235203535156251
MSE	• 38.87776312482501
R2	• 0.6655010188246948
Accuracy	• 0.002487562131136656

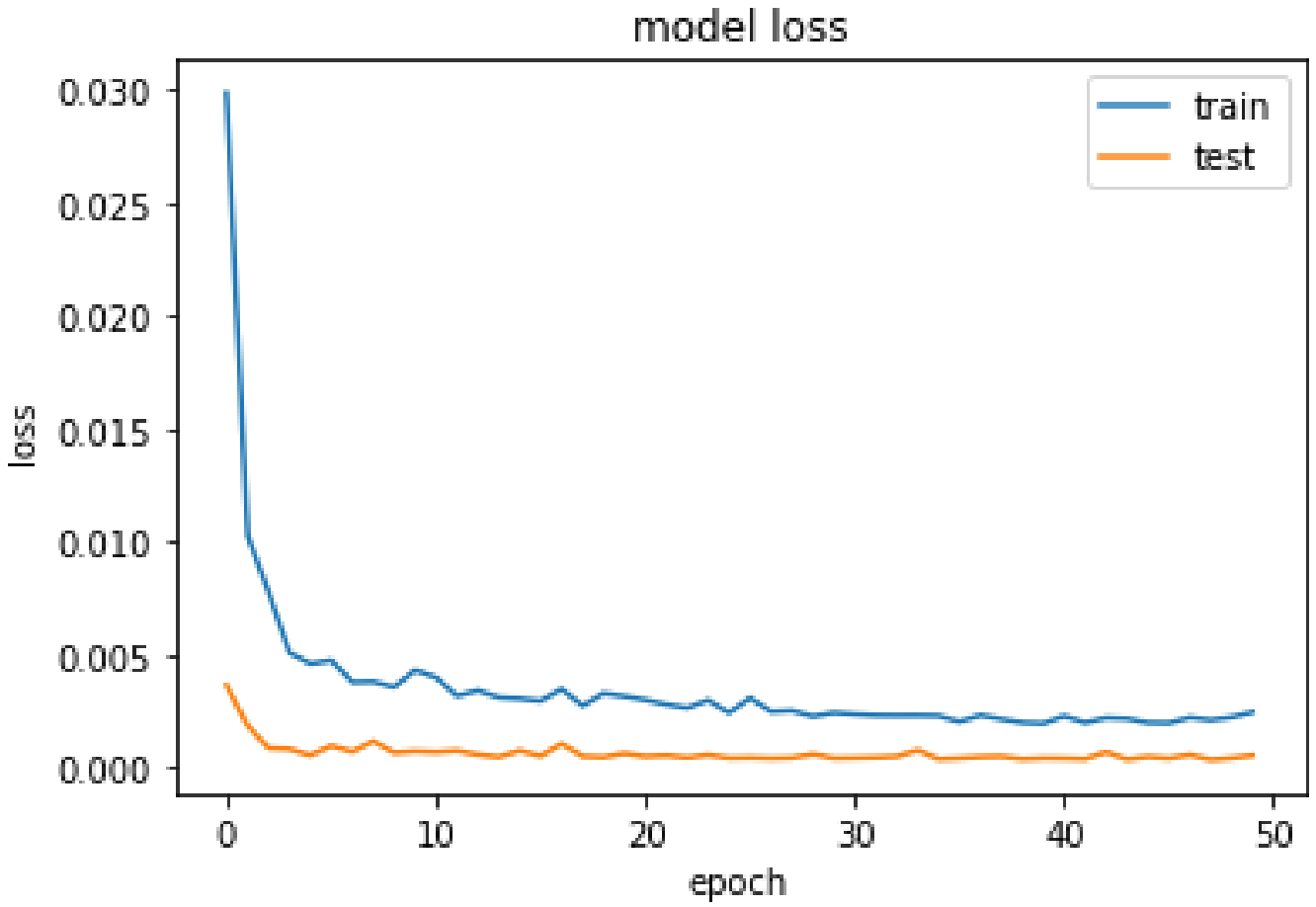
Bi-LSTM Model for Google dataset

We trained our model with the training data over bellow parameters :

- epochs = 50
- with a batch size = 32
- optimizer= Adam
- loss= mean squared error (MSE)
- Validation split=0.2
- Drop out = 0.6

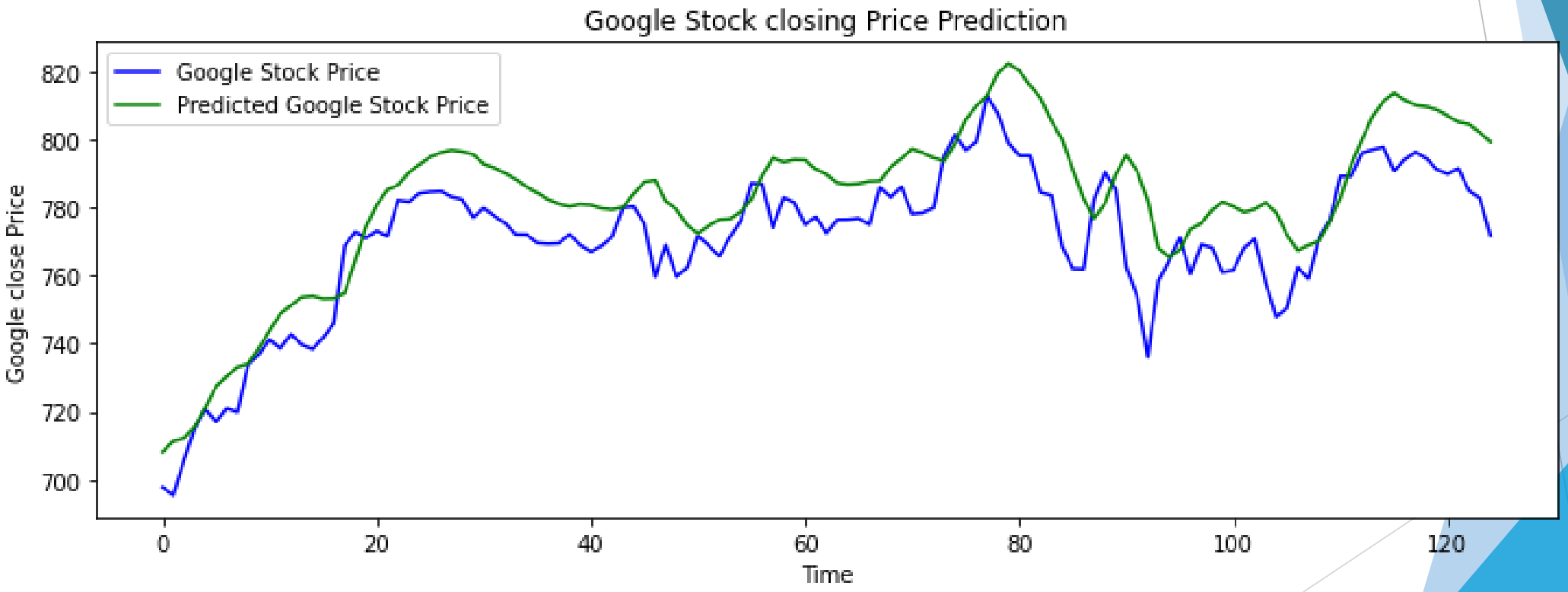
Bi-LSTM Model for Google dataset

- The Loss graph



Bi-LSTM Model for Google dataset

- The prediction



Bi-LSTM Model for Google dataset

RMSE	• 11.538846113281252
MSE	• 133.14496962598585
R2	• 0.5392159314681311
Accuracy	• 0.0023310023825615644

Conclusion

- In this approach, the goal was to predict stock prices using a recurrent neural network.
- For this purpose, several model architectures for stock price forecasting were presented. Also, tried to test different model parameters such as number of layers, nodes and drop outs and fitting over several batch size, epoch size, optimizer and different activation functions to determine the effect of different parameters in accuracy of the model prediction.
- With proper layers and hyper parameters, the LSTM and the Bi-LSTM models give good results for stock prediction.

References

- Long short-term memory – Wikipedia
- What are Recurrent Neural Networks? | IBM
- LSTM layer (keras.io)
- Bi-LSTM. What is a neural network? Just like our… | by Raghav Aggarwal | Medium
- Analysis of Stock Price Predictions using LSTM models | by Yu Hao Lee | Analytics Vidhya | Medium
- Stock Prices Prediction Using Long Short-Term Memory (LSTM) Model in Python | by Bee Guan Teo | The Handbook of Coding in Finance | Medium
- 9.4. Bidirectional Recurrent Neural Networks — Dive into Deep Learning 0.17.5 documentation (d2l.ai)
- Calculating-number-of-parameters-in-a-lstm-unit-layer

Thank you

Tahsin Ilkhas Zadeh : 400422034

Zeinab Khosravi : 99422067