Short Term Stock Price Prediction with Machine Learning

Lucas Breinlinger[1]

Abstract
Accurately predicting stock prices is a highly sought after mastery. In recent developments Long Short Term Memory, a type of recurrent neural network, have proven to be highly promising in achieving better predictions. todo: add 3-10 sentences blabla + result + methodology

[1] *Contact: lucasbr@me.com*

Contents

1. Project Definition

The accurate forecast of stock prices, traded on an exchange, is inarguably one of the most challeging topics in the field of asset pricing. The stock market is described as a unpredictable, dynamic and non-linear construct. Predicting the prices of its stocks is a ambitious undertaking and depends on many variables including but not limited to global economy, company's metrics and local and global political situation.

Historically there are two main prediction metods. The fundamental analysis which is a qualitative analysis of the company is interested in finding the true value of a stock and compare it to the actual traded value. The evaluator utilizes several performance criterias e.g. the P/E ratio in order to truely assess the underlying stock. Secondly, the technical analysis, which is solely based on the past price of the stock e.g. in form of closing or opening prices as time-series. Its rather a short term predicing using factors like the Moving Average (MA) or the Exponential Moving Average (EMA). Its basic assumption is that every significant information about the stock is already considered in the stock price.

The fast computational development has led to the point that Machine learning techniques have a significant application in financial problem. The use of artificial neural networks have found more and its way into the field of stock price prediction. Here a recurrent neural network (RNN) has found very proven, more precisely the Long Short Term Memory (LSTM). Its advantage being able to process entire sequences of data rather than only one single data point. It has proven to be very practial with time series data such our historical stock prices.

In this project, I create an application which is going to predict the **closing price** of any given stock for which it is trained on. For the sake of convenience this report only considers the stock price predition of the **Apple** Inc. (*$AAPL*) stock.

A train-test cycle is used to test the accuracy of the prediction. The root mean squared error (1) is used to measure the accuracy of the mode:

$$RMSE = \sqrt{\frac{\Sigma_i(\hat{y}_i - y_i)}{N}} \tag{1}$$

where $\hat{y}_i$ is the prediction of the $i-th$ stock price y_i and N is the total number of predicted stock prices.

2. Data Analysis

The dataset was collected with the help of `yfinance` python plugin. In my case it is replacing the not working Yahoo Finance API. The collected dataframe contains

- 1 108 837 rows
- 10 columns
- price information for 457 unique stocks
- Data of the last 10 years (including 17/07/2022)

and was collected on the 18/07/2022. I use a list of 457 randomly selected stock tickers to generate the dataset.

The structure and the first 5 lines of the table filtered for *Apple Inc.* are displayed in table 1. For each day and each stock (denoted by its ticker (*Ticker*) and company name (*Name*) the dataset contains the price of the stock in dollar when opening the market (*Open*), closing the market (*Close*). Furthermore there the days lowest (*Low*) and highest intra-day price (*High* is available, as well as the daily traded volume (*Volume*).

Looking at the *Apple Inc.* stock there are 2514 columns regarding the apple stock with a minimum closing price of

Table 1. First lines of *Apple Inc.* stock prices dataset

Ticker	Name	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
AAPL	Apple Inc.	2012-07-19	18.77	18.90	18.61	18.87	436 861 600	0	0
AAPL	Apple Inc.	2012-07-20	18.83	18.87	18.54	18.56	397 471 200	0	0
AAPL	Apple Inc.	2012-07-23	18.25	18.61	18.05	18.54	487 975 600	0	0
AAPL	Apple Inc.	2012-07-24	18.65	18.72	18.38	18.45	565 132 400	0	0
AAPL	Apple Inc.	2012-07-25	17.64	17.84	17.51	17.66	877 312 800	0	0

Table 2. Statistics of the Apple stock prices. Which represent our used features

	Open	High	Low	Close
count	2514.0	2514.0	2514.0	2514.0
mean	55.0	55.6	54.4	55.0
std	46.0	46.6	45.4	46.0
min	12.1	12.5	12.0	12.2
25%	22.8	23.1	22.7	22.8
50%	36.5	36.7	36.2	36.5
75%	66.7	68.5	66.0	67.5
max	182.6	182.9	179.1	182.0

12.17 and a maximum closing price of 182.01. There are no missing values (see table 2) in the dataset, and the date and value columns are correctly set as a datetime respectively float columns. In table 2 the statistics of the distinctive variables are shown.

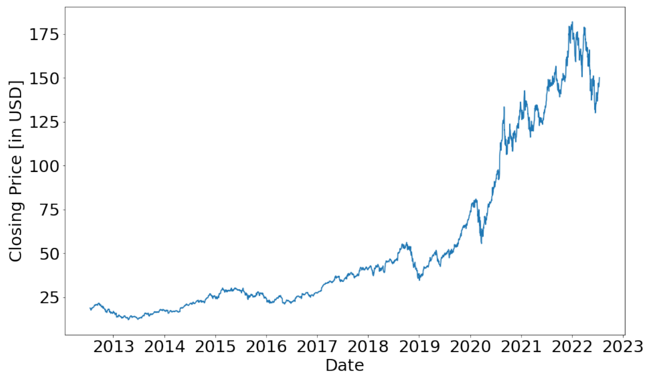

Figure 1. Closing price of the *Apple Inc.* in the last 10 years.

In figure 1 the highly volatile closing price of the stock is shown for its last 10 years. In recent years the stocke prices have incresead roughly sevenfold.

3. Methodology

Here, I am using a Recurrent Neural Network for the stock price prediction. More specifically a Long Short Term Memory Network is used to build the model. These networks are capable of learning long-term dependencies and are commonly used for handling and forecasting time-series data. In figure 2 a recurrent cell of a LSTM network is shown. Each cell consists of three gates, an input gate, an output gate and a forget gate.

Those cells are linked together to a chain, where each cell looks at the input gate at the previous hidden state h_{t-1},the current input x_t, and the previous cell state c_{t-1}. The forget cell is the first part where h_{t-1} and x_t are passed through the sigmoid activation function and pointwise multiplied with the cell state c_{t-1}. The input gates combines a sigmoid and tanh layer, here it is decided which parts are going to be updated. Multiplying both layers determines which information is used to update the cell state by pointwise adding it the the output of the forget gate, resulting in the final c_t. In the output gate the same input (h_{t-1} and x_t) is passed through a sigmoid function and finally multiplied with the current cell state c_t, yielding the final output the hidden state h_t.

In order for the LSTM to work properly the stock closing prices have to be normalized. This will help our model to converge faster and be more stable. Furthermore when using the scaler on more than one input features with different scales all features will contribute equally to the fitting and there will be no bias due to a different scaling. I am using the MinMaxScaler (2) of sci-kit learn.

$$x_{i,scaled} = \frac{x_i - min(x)}{max(x) - min(x)} \tag{2}$$

Here, I am trying to predict future stock prices y_t (target) with the help of the previous n stock prices. For example I am predicting the stock price from day 11 using day 1 to 10 as my features, for day 12, using day 2-11 and so on. In order to do this I defined a splitting function which generates me a data sample consisting of n samples (the previous stock prices) and the target value $n+1$. n is denoting the size of my window which I am using to predict the prices. This function is applied to my previously scaled stock prices.

Finally I split this data set in a training and a test set, since we are dealing with a time-series we cannot randomly select

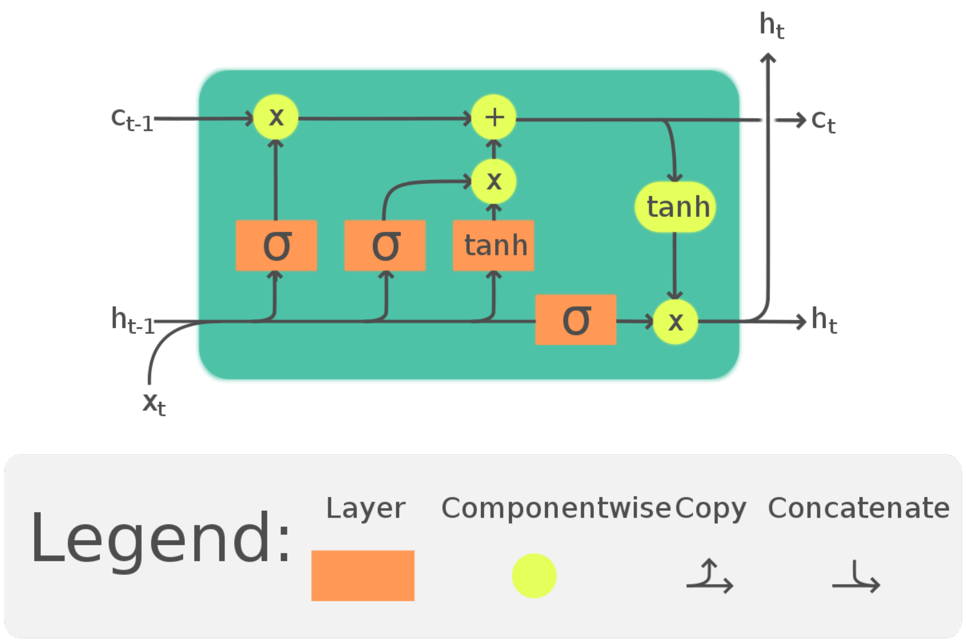

Figure 2. The reccurent cell of a LSTM network [1].

the samples, rather I split the sample at some point. In my experiment I am using 80% as a training set and the remaining 20% as my test set. Using the predicted stock price $\hat{y}_i$ and the real stock price y_t of the test set the RMSE can be easily calculated using eq. 1.

The first **simple model** I build, consisted of a single LSTM layer with 10 neurons and a dense Layer with 1 neuron. The dense layer is needed to get the dimensionality of the output of the LSTM h_t back to a 1D output. I am using the previous thirty close prices ($n = 30$) as my input x_t.

My refined model is a deeper neural network, since the relations between stock prices are quite complicated. Thus, I added an additional LSTM layer and a second dense layer with 25 neurons. The refined model is run with different parameters, the first run is 10 neurons for both LSTM layers, the next run is 50 neurons for both layers and in my last model I am using 150 neurons and 5 epochs . All the time n is kept at 30 days.

In table 3 the model parameters are shown in an overview.

4. Results

As expected the *RMSE* for the refined model is smaller than for the simple model. Furthermore the refined model 3 with 150 neurons per LSTM and 5 epochs also performed better than the refined model with less neurons. For comparison see the values for all four models in table 4.

Table 4. The resulting *RMSE* of the four models.

Model	*RMSE*
simple	37.83
refined 1	28.34
refined 2	9.59
refined 3	3.74

The poor performance of the simple model can also seen in figure 3 and 4. The prediction follows loosely the real price, however is much higher than the real price.

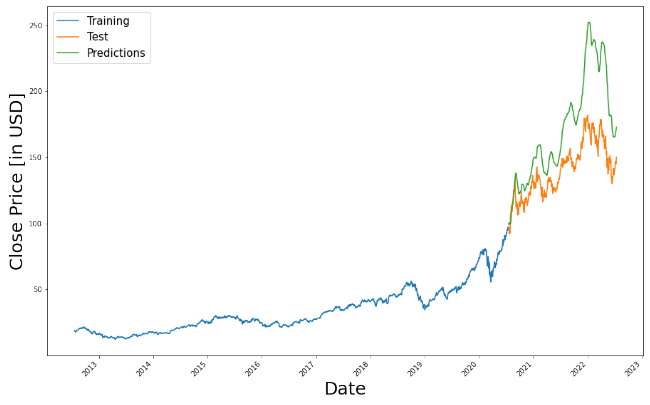

Figure 3. The prediction close price of the simple model versus the real close price.

5. Conclusion

Table 3. Overview of the model parameters.

Model	LSTM 1	LSTM 2	Dense 1	Dense 2	Epoch	n
simple	10	-	1	-	1	30
refined 1	10	10	25	1	1	30
refined 2	50	50	25	1	1	30
refined 3	150	150	25	1	5	30

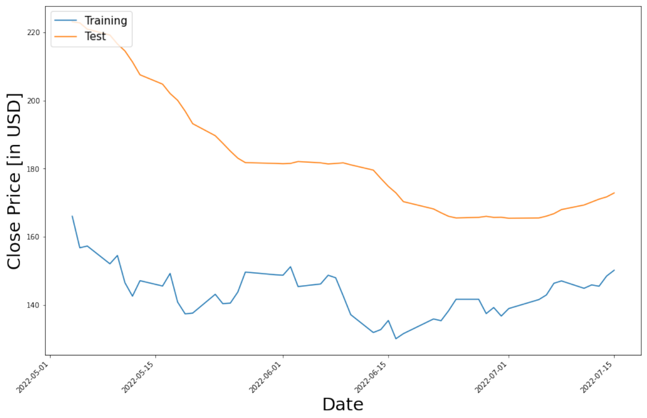

Figure 4. The prediction close price for the last 60 days of the simple model versus the real close price.

References

[1] Long short-term memory. https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg. Accessed: 2022-07-19.