Capstone Project
Data Science Nanodegree

Petar Valchev
October 2022

Stock prediction with LSTM

Definition

Project Overview

This project will explore the possibility of analysing time-series data in the form of stock prices in order to predict future growth or decline. For this purpose, a Long-Short Term Memory (LSTM) methodology will be adopted. The methodology will use a user-defined set of consecutive days that will train a model to predict the following days' closing price of Google. In order to do that, the model will be trained on readily available data from Yahoo Finance. In conjunction with that, additional features will be added to increase the model accuracy. Finally, different time-frames of look-backs will be compared to each other, in the attempt to find the most optimal solution.

Problem Statement

The task is to create an LSTM model that will predict the next day closing price as accurately as possible, based on pre-defined variables and in a user-defined look-back time frame. This will be achieved by doing the following:

1. Using Yahoo Finance API to extract data for Google
2. Pre-process the dataset, including additional features and splitting the dataset into sets of a length defined by the user, containing a tensor for the chosen variables and a matrix for the prices that match the variables
3. Build an LSTM model
4. Train the model on existing data
5. Predict the later part of the dataset by using the model
6. Compare the predicted values with the actual values
7. Tune model parameters and user inputs to improve results

Metrics

Mean Square Error - this metric measures the amount of error in the statistical model. Although it is not a determining metric for the success of the model due to the nature of the problem, keeping the MSE as low as possible gives an indication of a good model.

Trend – The more critical metric for the model evaluation will be the trend. A model which follows the trend of the actual price is able to predict growth or decline, which in the nature of the problem is considered to be satisfactory.

Data exploration

Stock data for Google has been taken from Yahoo Finance for the period of 01/10/2004 – 01/10/2022.

```
def get_stock_data(symbol, start_date, end_date):
    '''
    download stock data over from yahoo api form start date to end date
    input
        stock - String representing stock symbol eg APPL
        start - datetime object represent start date; default Jan 1, 2010
        end - datetime object represent end date; default: Jan 1, 2020
    output
        historical stock data pulled from yahoo finance stock api from start to end dates
    '''
    stockData = web.DataReader(symbol, 'yahoo', start_date, end_date)

    return stockData

start_date = dt.datetime(2004, 10, 1)
end_date = dt.datetime(2022, 10, 1)
df_GOOG = get_stock_data('GOOG',start_date, end_date)
```

A description of the values contained in the dataset have been shown in *Table 1*.

	high	low	open	Close	volume
count	4511	4511	4511	4511	4.51E+03
mean	37.5763	36.8236	37.2006	37.2058	1.23E+08
std	35.1829	34.4453	34.8091	34.8156	1.51E+08
min	4.22168	4.01769	4.0964	4.11209	1.58E+05
25%	12.8265	12.5388	12.6953	12.6769	3.01E+07
50%	23.1126	22.822	23.0212	22.9888	6.86E+07
75%	53.5588	52.3621	52.916	53.011	1.53E+08
max	152.1	149.887	151.863	150.709	1.65E+09

Table 1: Description of the dataset

Looking at the count, it can be seen that there are no missing values. Furthermore, we can see the sharp difference between the mean, 50% and 75% and the maximum values for high, low and open. Giving an early indication of the potential spike in stock price to be observed.

Figure 1 shows the correlation between the different variables in the database. It can be seen that every variable apart from the Volume is perfectly correlated to one another.

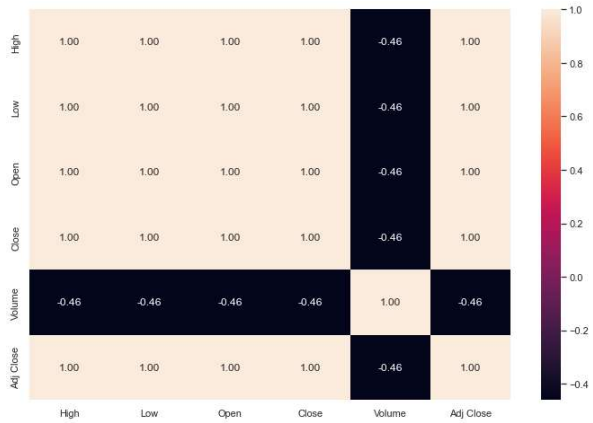

Figure 1: Correlation matrix

Figure 2: Stock price

In figure 2, the stock price has been plotted with respect to time. It can be observed that for the beginning of the analysed period, the price of the stock has been steadily increasing. However, towards the end, volatility can be observed, where the price is going through periods of sharp declines and sharp upward trends.

Algorithms and Techniques

Long short-term memory is a type of recurrent neural network that can process not only single data points but also entire sequences of data. The name is derived from having the ability to store and take into consideration information not only from short-term memory but also retained information from previous states called ‘long-term memory‘.

LSTM models have 3 gates and 1 state:

The cell state is used to store the ‘long-term memory’. It starts from the beginning of a cycle, gets updated if necessary, and then is used in conjunction with the input to produce an output.

The forget gate is used to update the cell state in the beginning. It takes into consideration the input and the 'short-term memory' from the previous event and updates the cell state accordingly.

The input gate is used to add information from the new event to the previous knowledge in the cell state.

The output gate takes into consideration information fed from the cell state combined with the previous output data to return a new output.

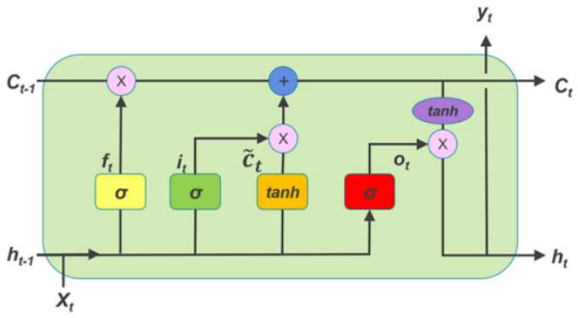

Figure 3: LSTM model

Methodology

Importing relevant libraries

Figure 4 shows the libraries that have been imported in order to proceed with the analysis. Libraries include common additions like pandas, numpy, seaborn, sklearn and keras. One library to note is the *finta* library used to calculate the technical indicators which will be added to the dataset, to be discussed later.

```
import datetime as dt
from finta import TA

import pandas as pd
from pandas_datareader import data as web
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import preprocessing
from sklearn.metrics import mean_squared_error
from numpy import size
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

import tensorflow as tf

import keras
from keras.models import Sequential
from keras.models import Model
from keras.layers import Dense, Dropout, LSTM, Input, Activation, concatenate
from keras import optimizers
from keras.callbacks import History
```

Figure 4: Relevant Libraries

Data Pre-processing

After loading the dataset, there are some necessary steps which need to be performed in order to ensure firstly, the accuracy of the prediction is as high as possible and secondly, making sure that parameters are passed into the LSTM model in their correct form.

After the initial exploration of the dataset it has been determined that additional features will be required in order to increase the accuracy of the prediction. The features are in the form of technical indicators used to measure the performance of the stock in various ways. Technical indicators are split into different types, namely: trend , momentum, volatility and volume. Even though it might be tempting to include as many of those indicators as possible, adding such requires caution. If many indicators of the same type are added, the model will suffer from multicollinearity, overblowing the results in one direction or another. For this reason, it is advised that a mixture of technical indicator types are used in a way that will complement each other. Different types of strategies exist when one is looking to interpret the stock market in an attempt to make predictions on the future price of a stock. One such includes using Relative Strength Index together with Bollinger Bands. Attention needs to be placed on observing if the prices hit any of the upper or lower bands whilst looking after the RSI value to assess if the stock is overbought or oversold. Such indicators can server as an early signal for investors for a reversal in the trend of a stock.

For the purposes of this analysis the following technical indicators have been chosen:

- Relative Strength Index (RSI) – Used as a momentum indicator, commonly used to determine the strength of price changes
- On Balance Volume – Used as a volume indicator, calculating the buying and selling pressures of a stock
- Bollinger Bands – Used as a volatility and trend indicator, plots three trend lines, determining whether a stock is overbought or oversold

Once all the technical indicators are included, the dataset would need to be split into a dataset used to train the model and one that will be used to test the model on and compare its predictive power. The split will follow the standard practices by allowing for 70% of the dataset for training and 30% for testing.

Once the split has occurred, the datasets need to be normalized in order to facilitate the different ranges of values such as volume and price. It is also important to normalize the train and test datasets separately in order to prevent data leakage from one dataset to another. Sklearns' function MinMaxScaler will be used to normalize the data of the train set. Then that scaler will be used to transform the variables of the testing dataset.

```
#Split data
train_data = df[:split_index]
test_data = df[split_index:]

#Normalize the variables
normaliser = preprocessing.MinMaxScaler(feature_range=(0,1))
#Creating a specific scaler for the y values
y_normaliser = preprocessing.MinMaxScaler(feature_range=(0,1))
#Normalizing the training dataset by using fit_transform
train_normalised = normaliser.fit_transform(train_data)
```

Figure 5: Splitting and normalizing

Once splitting and normalizing is accounted for, the training and testing datasets need to be further split into 2 arguments. One includes the list of preceding events before a price prediction and the other one includes the price prediction. The above is done via a method to convert the dataset into a tensor containing all the lists of preceding events, the length of which is defined by the user, where each entry in the list keeps all of its technical indicators. Then from the passed dataset, only the next day prices are collected following the list of events, ensuring a 1-day lag. In the end, a combination of a next day price combined with its n preceding events is achieved.

```
X, y = list(), list()
for i in range(len(sequences)):
    # find the end of this pattern
    end_ix = i + n_steps
    # check if we are beyond the dataset
    if end_ix > len(sequences):
        break
    # gather input and output parts of the pattern
    seq_x, seq_y = sequences[i:end_ix, :-1], sequences[end_ix-1, -1]
    X.append(seq_x)
    y.append(seq_y)
return np.array(X), np.array(y)
```

Figure 6: Splitting train and test datasets into a tensor and a matrix

Model

The model that has been built includes 4 LSTM layers, each followed by a Dropout. All 4 LSTM layers have been modelled to read the input data and output 70 features and have a *relu* activation function. The optimizer used for the model is *adam*. The model is also trained for 25 epochs with a batch size of 32.

```
model = Sequential()
#Adding the first LSTM layer and some Dropout regularisation
model.add(LSTM(units = 70, return_sequences = True, input_shape = (X_train.shape[1], 10)))
model.add(Dropout(0.2))
# Adding a second LSTM layer and some Dropout regularisation
model.add(LSTM(units = 70, return_sequences = True))
model.add(Dropout(0.2))
# Adding a third LSTM layer and some Dropout regularisation
model.add(LSTM(units = 70, return_sequences = True))
model.add(Dropout(0.2))
# Adding a fourth LSTM layer and some Dropout regularisation
model.add(LSTM(units = 70))
model.add(Dropout(0.2))
# Adding the output layer
model.add(Dense(units = 1))
model.add(Activation('relu'))

# Compiling the RNN
model.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
model.fit(X_train, y_train, epochs = 25, batch_size = 32)
```

Figure 7: LSTM model

Once the model is trained, then stock price predictions are being generated based on the data from the testing dataset.

```
X_train, X_test, y_train, y_test, y_test_norm, split_index, days_prior = data_preprocessing(df_GOOG, 21, 0.7)
model = build_lstm(X_train, y_train)
predicted_stock_price = model.predict(X_test)
```

Figure 8: Pre-processing, building the LSTM model and predicting the stock price

```
Epoch 1/25
99/99 [==============================] - 8s 27ms/step - loss: 0.0067
Epoch 2/25
99/99 [==============================] - 2s 25ms/step - loss: 0.0021
Epoch 3/25
99/99 [==============================] - 3s 25ms/step - loss: 0.0016
Epoch 4/25
99/99 [==============================] - 3s 26ms/step - loss: 0.0016
Epoch 5/25
99/99 [==============================] - 2s 24ms/step - loss: 0.0015
Epoch 6/25
99/99 [==============================] - 2s 25ms/step - loss: 0.0027
Epoch 7/25
99/99 [==============================] - 2s 25ms/step - loss: 0.0012
Epoch 8/25
99/99 [==============================] - 3s 25ms/step - loss: 0.0011
Epoch 9/25
99/99 [==============================] - 2s 25ms/step - loss: 0.0011
Epoch 10/25
99/99 [==============================] - 2s 25ms/step - loss: 0.0022
Epoch 11/25
99/99 [==============================] - 2s 25ms/step - loss: 0.0012
Epoch 12/25
99/99 [==============================] - 2s 25ms/step - loss: 8.9086e-04
Epoch 13/25
...
99/99 [==============================] - 3s 25ms/step - loss: 6.3515e-04
Epoch 25/25
99/99 [==============================] - 2s 25ms/step - loss: 6.1164e-04
42/42 [==============================] - 1s 8ms/step
```

Figure 9: Epochs

It can be seen that the model has a low loss, indicating that it is doing a good job at guessing the training dataset.

Once the predictions have been made, they are plotted against the actual values observed in the testing dataset.

```
# Visualising the results
plt.figure(figsize=(15,8))
plt.plot(y_test_norm, color = "red", label = "Real Price - scaled")
plt.plot(predicted_stock_price, color = "blue", label = "Predicted Price - scaled")
plt.title('Google Stock Price Prediction - 21-day look-back')
plt.xlabel('Time range')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
print("Mean absolute error of {0}".format(mean_squared_error(y_test_norm[:-20], predicted_stock_price, squared = False)))
```

Figure 10: Plotting the chart

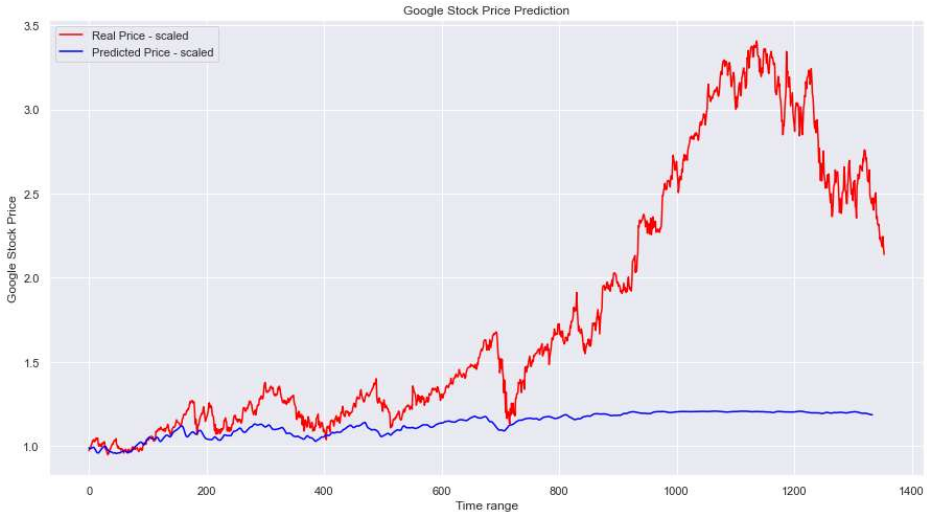

Figure 11: Predicted vs Actual values

MSE = 0.9545

Even though the MSE of the model is around 95, it can be seen that the model has done a relatively decent job at predicting the direction at which the stock is going. However, technical indicators have performed well up until the drop in 2020. Afterwards, it can be observed that the technical indicators lose track of the rapid increase of the stock. This serves as proof that there are other factors influencing the rise of the stock – either other technical indicators or external factors like media hype or the phenomenon of fear of missing out etc.

Refinement

After observing the values in Figure 11, an attempt has been made to refine the LSTM model by fine tuning the parameters such as the number of layers, the amount of units contained in them, number of epochs etc. Another parameter which has been tweaked with is the number of days preceding the price prediction. The above results have been achieved after looking at 21 days prior to the predicted price. Attempts will be made with a shorter time-frame of 7 days and a longer time-frame of 50 days.

7-days

```
Epoch 1/25
99/99 [==============================] - 6s 12ms/step - loss: 0.0118
Epoch 2/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0021
Epoch 3/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0017
Epoch 4/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0015
Epoch 5/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0013
Epoch 6/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0012
Epoch 7/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0011
Epoch 8/25
99/99 [==============================] - 1s 13ms/step - loss: 0.0012
Epoch 9/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0012
Epoch 10/25
99/99 [==============================] - 1s 12ms/step - loss: 9.4632e-04
Epoch 11/25
99/99 [==============================] - 1s 12ms/step - loss: 0.0010
Epoch 12/25
99/99 [==============================] - 1s 12ms/step - loss: 9.7206e-04
Epoch 13/25
...
99/99 [==============================] - 1s 12ms/step - loss: 6.3261e-04
Epoch 25/25
99/99 [==============================] - 1s 12ms/step - loss: 6.4328e-04
43/43 [==============================] - 1s 3ms/step
```

Figure 12: Epochs – 7-days

Figure 13: Predicted vs Actual – 7-days

MSE = 1.009

50-days

```
Epoch 1/25
98/98 [==============================] - 11s 55ms/step - loss: 0.0059
Epoch 2/25
98/98 [==============================] - 6s 59ms/step - loss: 0.0017
Epoch 3/25
98/98 [==============================] - 6s 64ms/step - loss: 0.0014
Epoch 4/25
98/98 [==============================] - 6s 65ms/step - loss: 0.0014
Epoch 5/25
98/98 [==============================] - 6s 62ms/step - loss: 0.0013
Epoch 6/25
98/98 [==============================] - 6s 66ms/step - loss: 0.0012
Epoch 7/25
98/98 [==============================] - 6s 63ms/step - loss: 0.0014
Epoch 8/25
98/98 [==============================] - 6s 62ms/step - loss: 0.0012
Epoch 9/25
98/98 [==============================] - 6s 62ms/step - loss: 0.0011
Epoch 10/25
98/98 [==============================] - 6s 62ms/step - loss: 9.3813e-04
Epoch 11/25
98/98 [==============================] - 6s 61ms/step - loss: 9.4177e-04
Epoch 12/25
98/98 [==============================] - 6s 62ms/step - loss: 0.0012
Epoch 13/25
...
98/98 [==============================] - 6s 63ms/step - loss: 6.4541e-04
Epoch 25/25
98/98 [==============================] - 6s 63ms/step - loss: 6.4075e-04
41/41 [==============================] - 2s 16ms/step
```

Figure 14: Epochs – 50-days

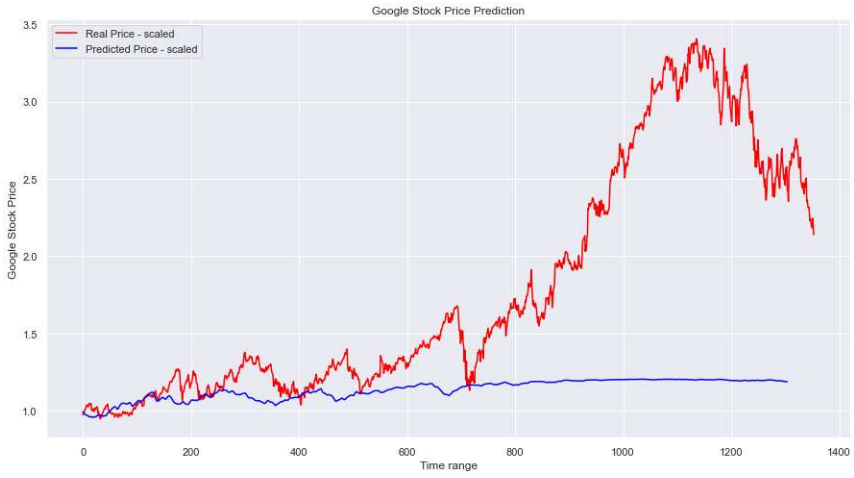

Figure 15: Predicted vs Actual – 50-days

MSE = 0.943

From the graphs above, it can be seen that the 7-day model follows a similar trend to the 21-day one. However, it has a higher mean square error, making it slightly worse than the 21-day model. The 50-day model on the other hand provides us with a better mean square error than the 21-day. It can, however, be seen that the predicted line does not follow the trend as well as in the 21-day model, lagging behind by a noticeable amount. The lower mean square error could be a result of the blue line having more overlapping parts with the red line, which does not necessarily result in a better model as it can be seen from the figures above.

Those 3 cases were further tested with different configurations of the LSTM model, including a different number of LSTM layers, dense layers, activation functions and units. However, it did not provide any significant improvements on the ones on display. Out of the activation functions, the *relu* was outperforming the other available ones by a large margin. Epochs were also increased in size, though it did not result in sizeable differences.

Conclusion

An attempt was made to predict the next day price of the stock of Google based on its preceding n-days. This was achieved by fetching the data from Yahoo Finances' API and then applying an LSTM technique in order to take advantage of its ability to manipulate time-series data.

Apart from the indicators which came with the dataset from Yahoo Finance, further technical indicators have been added in an attempt to increase the accuracy of the predictions. Such additions include the Relative Strength Index (RSI) used as a momentum indicator, the On Balance Volume – used as a volume indicator and the Bollinger Bands – used as a volatility and trend indicators.

The LSTM model used included 4 LSTM layers with a relu activation function splitting the inputs into 70 features. They were followed by a Dense layer whilst the optimizer that has been used is adam.

Dataset has been split and converted into tensors at a user defined length which correspond to the number of preceding days the prediction takes into account. Lengths tested include 7-days, 21-days and 50-days. Out of the ones tested the 21-day scenario provided the best results.

All in all, the model did provide reasonable predictions on the trends of growth and decline, however, it was far from perfect. Furthermore at the point at which volatility increased, the model did not perform well, not being able to recognize the upward trend of the stock price.

Improvement

Improvements on the model include researching and testing out different technical indicators in an attempt to find the ones with the highest predictive power. More combinations within the LSTM model can be explored including layers, number of epochs etc. Furthermore, as mentioned previously, external factors do also influence the change in price of the stock. Incorporating an algorithm which would be able to track the sentiment of the market and applying a weight based on positive or negative news trends might prove to be beneficial in increasing accuracy. A natural

language processing algorithm can be used for this purpose to analyse articles or other sources of text information. Once a desirable accuracy is achieved, an algorithm can be written that incorporates a buy/sell strategy which would maximize profits.

Summary:

- Incorporate different combinations of technical indicators in order to find the most suitable
- Use NLP to analyse text data and incorporate sentiment into price prediction
- Create an algorithm that incorporates buy/sell strategies which uses the predictive power of the model to maximize profits