STOCK TREND PREDICTION MODEL

Authors: Emanuele G. Renda, Kindra R. Chiappinelli

ABSTRACT

We attempted to predict the future trend of Goldman Sachs' (GS) stock price per share seven days in advance. We utilized the Long-Short Term Model (LSTM) as our neural network architecture and we came up with different approaches in order to achieve our goal. At first, we built two models that are able to predict one day in advance, then we used one of them to predict iteratively the next day by feeding the new prediction into the dataset. Secondly, we moved to a more reliable approach that uses a seven dense layer. We came to decent results, but have come to the conclusion that there are many other variables that could be included to fine tune the results to a better accuracy.

INTRODUCTION

Aim of this report is to present all the strategies implemented in order to find a suitable deep learning model for predicting the future trend price of a GS's share a week-ahead. This means that if the model is run today, it will predict the next seven days of future prices.

For our purposes we downloaded directly from the Yahoo! finance package our dataset which consists of six variables: Adjusted Close (Adj. Close), Open, High, Low, Close, Volume. The target variable is the Adj. Close; while, in regards to the features used, as it is explained later, we decide to not use all the variables as inputs for our model. Instead, we assumed that the stock price history of changes can be a valuable indication of that stock's future price movements, a typical assumption of the technical analysis.

METHODOLOGY AND APPROACHES

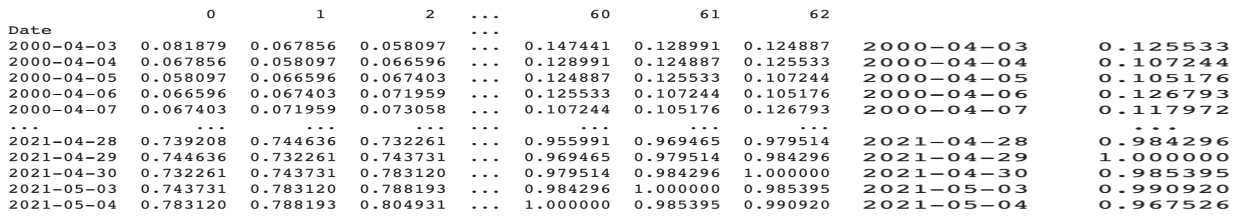
The Approach. In order to achieve our project goal of developing a model that is able to predict a week-ahead, the approach that we used was to start building at first a one-day ahead model. As you will see in the next few pages, we start from a one-day ahead model that has two features as inputs, Adj. Close and Volume. Then, we move on to a second one-day ahead model that uses one feature as input, the Adj. Close. Specifically, this last model is known as the univariate model, which is the base case for our final model, the week-ahead one uses a single feature.

The dataset was divided into three parts, because we wanted to validate our final model with the validation set: the training set (Dates: 01/01/2000 - 06/30/2020), the validation set (Dates: 07/01/2020 - 03/31/2021), and the test set (Dates: 04/01/2021 to present date. Thus, when the model will be run again, we used datetime.now() to accomplish this particular characteristic. Therefore, you will be able to make predictions for your "present time" about your "future time"). All our dataset was normalized, so all the real values were "translated" in values between 0 and 1, this was accomplished by using MinMaxScalar functionality.

In order to reach a model that is able to predict seven days ahead, we used two particular approaches. The first one, we used the univariate model as a model for making predictions one day ahead, then we fed the new prediction into the test set and we did the same thing iteratively for fourteen days ahead. Unfortunately, a similar approach is not as good as we thought and it is demonstrated in the results section. The second approach is more reliable in which we considered a new input shape as well as the target shape, (-1,63,1) and (-1,7,1) respectively, which is the LSTM with a seven dense layer at the end.

<u>The Methodology</u>. For our purposes the best fit architecture is a LSTM suitable for processing time-series data and known to have the strengths of recurrent neural networks (RNN), with the additional advantage to resolve gradient vanished problems typical of RNNs. All of our models use the same architecture, two hidden layers of LSTM and one dense layer that labels our outputs. Another choice that we decided to implement was the utilization of EarlyStopping and CheckPoint functionality. We let the neural network model decide which epoch is the best for our model, and so we put a high number of epochs (150) allowing the model the time to decide which one is the best. One major change that you can see as you move from the first model to the final model is that we changed the way we feed data into the neural networks. Indeed, the first model is fed with data that has a window size of five days, this means that the model uses five preceding days in order to predict the next day's price. While, for the other two models the data is fed with a window size (or timesteps) of sixty-three days. The following image sums up visually what we have just explained in this last part; for the univariate model we have:

x_train y_train



```
                   0         1         2  ...        60        61        62
Date                                      ...
2000-04-03  0.081879  0.067856  0.058097  ...  0.147441  0.128991  0.124887   2000-04-03    0.125533
2000-04-04  0.067856  0.058097  0.066596  ...  0.128991  0.124887  0.125533   2000-04-04    0.107244
2000-04-05  0.058097  0.066596  0.067403  ...  0.124887  0.125533  0.107244   2000-04-05    0.105176
2000-04-06  0.066596  0.067403  0.071959  ...  0.125533  0.107244  0.105176   2000-04-06    0.126793
2000-04-07  0.067403  0.071959  0.073058  ...  0.107244  0.105176  0.126793   2000-04-07    0.117972
...              ...       ...       ...  ...       ...       ...       ...                    ...
2021-04-28  0.739208  0.744636  0.732261  ...  0.955991  0.969465  0.979514   2021-04-28    0.984296
2021-04-29  0.744636  0.732261  0.743731  ...  0.969465  0.979514  0.984296   2021-04-29    1.000000
2021-04-30  0.732261  0.743731  0.783120  ...  0.979514  0.984296  1.000000   2021-04-30    0.985395
2021-05-03  0.743731  0.783120  0.788193  ...  0.984296  1.000000  0.985395   2021-05-03    0.990920
2021-05-04  0.783120  0.788193  0.804931  ...  1.000000  0.985395  0.990920   2021-05-04    0.967526
```

In the week-ahead model the y_train is composed of seven values for each date. We accomplish this objective by using the function target_y(y_set). You can also notice in the week-ahead model that we do not have a target dataset for the test set, the reason behind this choice is that the test set will serve as a set to make real-time predictions as we explained before. Therefore, the test targets of this model are not delayed by the period that we want to predict.

One-Day Ahead Models

- LSTM 1 dense, input shape (-1, 5, 2), no drop-out, two LSTM hidden layers with 50 units, activation function “ReLU”, optimizer Adam, batch size 150, and Mean Squared Error (MSE) as a loss function. The input shape has to be in the way shown before, because LSTM requires the shape of our input data to be: (number of observations, timesteps, number of features), where timesteps could be seen as “how much we want the neural network to remember from the past”, or window size.
- LSTM 1 dense univariate, input shape (-1, 63,1), no drop-out, two LSTM hidden layers with 50 units, activation function “ReLU”, optimizer Adam, batch size 150 and MSE as a loss function.

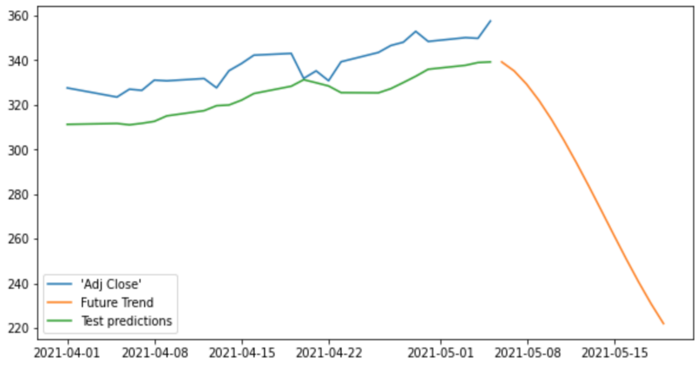
Week Ahead Model

- LSTM 7 dense, input shape (-1, 63,1), no drop out, two LSTM hidden layers with 60 units (originally there were 50 units, but as you will see in the results section, the model improved with 60 units for each layer), activation function “ReLU”, optimizer Adam, batch size 140, and MSE as a loss function, patience of 20 epochs.

RESULTS AND ANALYSIS

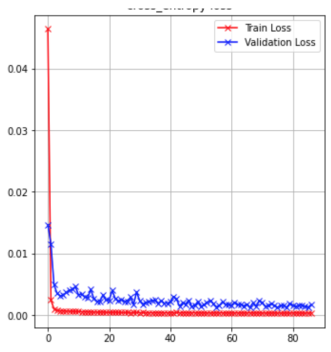
Because of the limited space available, we will present only some results which are related to the aim of predicting seven days ahead. The rest of the results can be seen once you run each model.

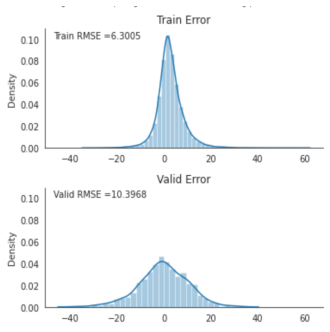
One-Day Ahead Model Result (feeding the prediction to the test set approach): As mentioned earlier, in this approach we predicted tomorrow's price by using the univariate model (one-day ahead) and then we fed this prediction into our test set. We then remade a prediction in order to get the prediction price for the day after tomorrow, and we did this iteratively for fourteen days. This is the result of such intuition for today (05/04/2021):

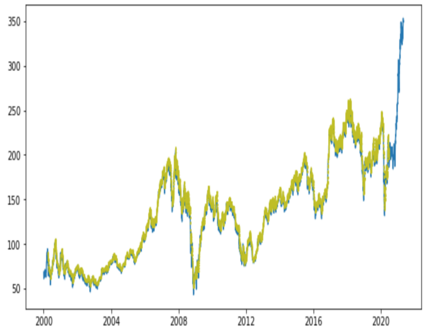


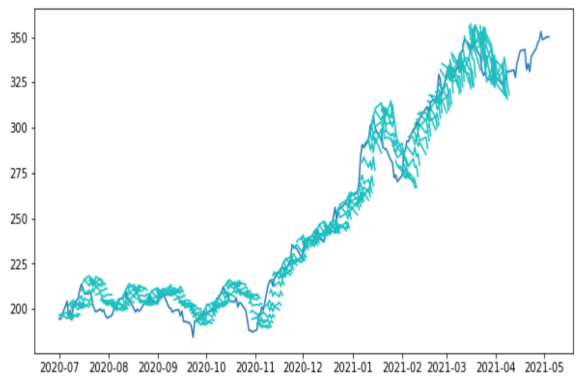
The drawback of such intuition is that we do not know if the prediction that we are feeding into our test set is correct, potentially causing even further wrong predictions.

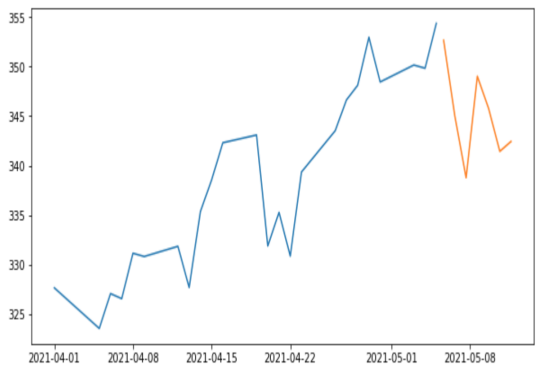
Week Ahead Model Results: With our configuration the improvement stopped at the 69th epoch. The following images show the validation loss vs. training loss, the distribution of errors for the training and validation set, predictions for the training, predictions for the validation set, and prediction for the next seven days from today (05/04/2021), respectively.







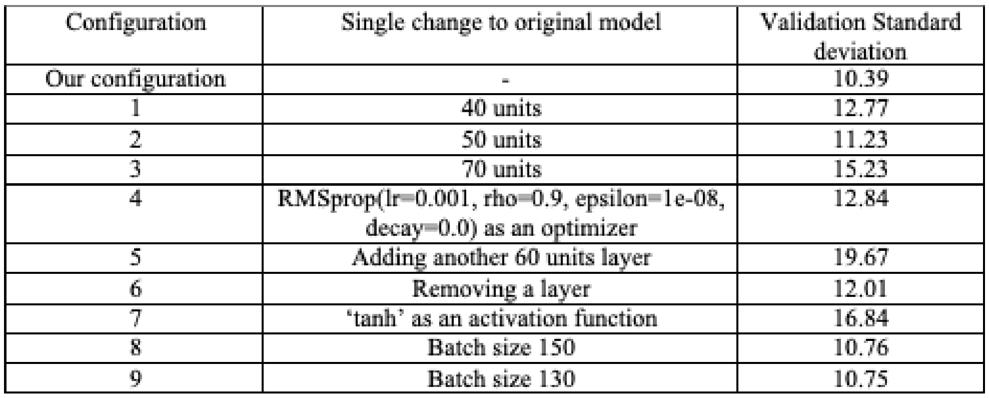




5

As it is quite noticeable from the distribution of the model's errors, we achieved a good result. The validation set absolute errors vary approximately with a range of 20, but we should also consider the fact that our validation set is not so huge, and this could be a limitation. The standard deviation increased from 6.30 of the training set to 10.39 of the validation set.

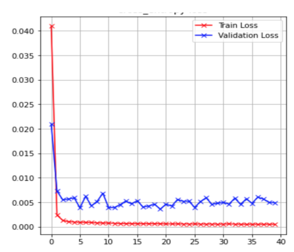
The configuration that we presented here is the best that we found after varying settings.

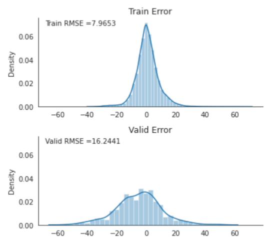


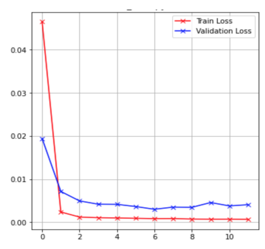
Configuration	Single change to original model	Validation Standard deviation
Our configuration	-	10.39
1	40 units	12.77
2	50 units	11.23
3	70 units	15.23
4	RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0) as an optimizer	12.84
5	Adding another 60 units layer	19.67
6	Removing a layer	12.01
7	'tanh' as an activation function	16.84
8	Batch size 150	10.76
9	Batch size 130	10.75

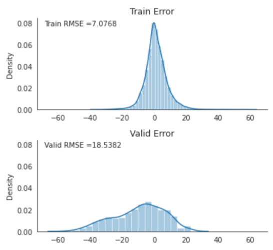
One major thing that you are not able to see in the model is a dropout. Let us discover the reason why, we decided to not use it with another model with the following configuration and here we explore it's results. Configuration:

- LSTM 7 dense, input shape (-1, 63,1), dropout of 0.20 between the first and the second hidden layer , two LSTM hidden layers with 60 units, activation function "ReLU", optimizer Adam, and MSE as a loss function. In the first image we have the results when the model is run with a patience of 5 epochs because we do not want to overfit too much the training set, while in the second the patience is the same as the previous model, 20. The first configuration stopped at the 12th epoch (best model at the 7th epoch), while the second one at the 40th epoch (best model at the 20th epoch).

- 







- As we can see the less we fit the training set, the higher the standard deviation of the errors, from 10.39 of our model to 16.24 and 18.54 of these two last configurations. The reason is particularly linked to our hypothesis. In other words, by fitting our data we are learning the past behaviour of the GS's price changes in the training set, once the model is done it is ready to predict the future. The results demonstrate, at least for this particular case, the reliability of our assumption. Therefore, learning the past behavior of the stock price could lead us to some useful results.

CONCLUSION

Although our team achieved great results, there are a lot of improvements that need to be made in order to achieve even better results. The main one is linked to the assumption that the past performance could be a good indicator of future price movements. Although some investment philosophies rely on the past performance in order to get some buy or sell signals, for other investment philosophies (such as the fundamental technique) the past performance is not an indicator of future price movements. Therefore, we do think that relying only on one single feature, the price itself, is not enough to achieve a greater result. So, the next implementation of our model would be to have more features fed into the neural network.

BIBLIOGRAPHY

Dr. Dataman. “A Technical Guide on RNN/LSTM/GRU for Stock Price Prediction.” *Medium*, 6 December 2020, https://medium.com/swlh/a-technical-guide-on-rnn-lstm-gru-for-stock-price-prediction-bce2f7f30346.

Jansen, Stefan. *Machine Learning for Algorithmic Trading*. Second ed.

Jansen, Stefan. “Machine Learning for Algorithmic Trading Second Edition.” *GitHub*, 26 June 2020, Recurrent Neural Networks Univariate Time Series Regression.

Müller, Florian. “Time Series Forecasting – Creating a Multi-Step Forecast in Python.” *Relataly*, 19 April 2020, https://www.relataly.com/multi-step-time-series-forecasting-a-step-by-step-guide/275/#h-prerequisites.

Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications Hardcover. 1999.