Stock Price Forecasting using LSTM

June 17, 2021

Leonardo Emili Alessio Luciani

1. Introdution

Stock price prediction is an active field of study where the objective goal is to predict the trend of the stock market, typically based on its historical evidence. Despite the theory of *efficient markets* claims that there cannot be any pattern in the trend of financial assets due to the overall knowledge of the mass, some affirm that periods of human behavioral irrationality lead to strong operation correlation reflected in the markets (Lo, 2017). Therefore, the task of predicting stock prices based on past data seems not to be completely inapproachable. Over the years, many methods have been proposed to tackle the task: ranging from the Naive Forecast approach, which trivially forecasts the stock value to be the last observed one, up to the most recent ones that heavily rely on machine learning techniques. In this context, we will delve into the topic of stock price prediction using fundamental data to assess whether a stock is attractive to investors. This technique, as opposed to technical analysis, considers prices, as well as financial reports, to model the problem. Here, the underlying assumption is that one cannot tell whether it's worth buying a stock only by looking at its current price and volumes. Indeed, quarterly released financial reports about companies and external information (e.g. the common sentiment on a stock ticker) may be crucial to assess the quality of a given stock. In this project, we experimented with the effectiveness of deep learning techniques applied to the stock price prediction task. The key idea is the use of recurrent neural networks that are able to exploit temporal dependencies of events, hence conditioning the presence of an event at timestep *t* on the previous events at *t-1, t-2, ...*

2. Related work

Some approaches have been explored in this direction, such as in (Zou & Qu, 2020) where the authors apply LSTM, Stacked-LSTM, and Attention-Based LSTM into the prediction of stock prices. The authors also propose an evaluating framework to assess the quality of their models based on the return of the trading strategy. In another work (Mehtab et al., 2020), the authors predict the open value of NIFTY 50 using different machine learning and deep learning models. They also demonstrate that using one-week prior data as input leads to good results.

@affilnum1¡ [0]**AUTHORERR: Missing \dlaiaffiliation.** Email: Leonardo Emili <emili.1802989@studenti.uniroma1.it>, Alessio Luciani <luciani.1797637@studenti.uniroma1.it>.

Deep Learning and Applied AI 2021, Sapienza University of Rome, 2nd semester a.y. 2020/2021.

3. Dataset

Data wrangling operations have been crucial for this task. In fact, we start considering the S&P 500 stock data, which is a collection of daily stock prices for all companies from the S&P 500 index. This dataset provides us with the following features: open price, highest price, lowest price, close price, volume, stock ticker, and date. We also consider an auxiliary dataset that provides us with fundamental data from Yahoo Finance. The dataset contains the following columns: Forward P/E, DE Ratio, Earnings Growth, Enterprise Value/EBITDA, EBITDA, Current Ratio, Cash Flow, Trailing P/E, Beta, PEG Ratio, Gross Profit, Total Debt, Price, Return on Equity, Return on Assets, Price/Book, Revenue Growth, Operating Margin, Enterprise Value/Revenue, Revenue, Total Cash, Enterprise Value, Total Cash Per Share, Profit Margin, Price/Sales, Book Value Per Share, Diluted EPS, Market Cap, Revenue Per Share, Net Income Avl to Common, Ticker, Date. Based on the date column, we can align the two datasets such that for each event in the S&P 500 dataset, we have the latest available financial report. From now on, we will refer to the dataset obtained from the alignment process simply as the dataset.

3.1. Feature engineering

We fill forward values whenever possible, we otherwise replace missing values with constant values (i.e. zero paddings). The intuition here is that if there are missing values at the beginning of the history of a stock, it means that the feature is not available, otherwise referring to the latest value as the most up-to-date one. As an example, consider holidays when markets are closed, and the price does not change since no orders are placed. We also fill the dataset with missing working days using the same forward-

fill strategy. Furthermore, we perform significant feature engineering steps adding technical indicators such as SMA and RSI. The SMA is computed by averaging the prices of a given number of multiple contiguous time steps. The RSI was, instead, computed using the relative strength formula (rsi).

$$U = max(0, closeNow - closePrevious)$$

$$D = max(0, closePrevious - closeNow)$$

$$RS = SMMA(U, n)/SMMA(D, n)$$

$$RSI = 100 - 100/(1 + RS)$$

It takes into account the differences in contiguous prices, with respect to the exponentially smoothed moving average. Therefore, it makes it easy to spot overbought and oversold events. In fact, using this technical indicator, we could extract the overbought and oversold binary features too. According to the RSI definition, a value that goes higher than 70 can be interpreted as a situation of overbought. Similarly, a value that goes below 30 is seen as a situation of oversold. An articulated model could of course extract this kind of information directly from the price, but manually adding these knownly meaningful indicators definitely simplifies the work of the network. Another step of the feature engineering process involved scaling the features to put them on similar scales. This step was very important considering that we are using gradient descent to optimize the models and having features with completely different scales would have made the convergence task much harder. Therefore, we globally scaled prices, volumes, technical indicator values, fundamentals, etc. At this point, all the features were centered on the same scale. Before training the model, we split the dataset into three subsets: *train set*, *dev set* and *test set*. These are respectively meant for training the model, tuning its hyperparameters and testing its performance. Since the core idea of the project is forecasting future events by looking at past data, we cannot leak future information in the train set. Therefore, we split the dataset by years. Having temporal data ranging from 2004 to 2013, we dedicate the years 2004 - 2011 to the train set, 2012 to the dev set, and 2013 to the test set. This way, the procedure is similar to backtesting the model strategy on present events. In order to predict the target adjusted close price, we consider the dataset according to the sliding window approach, also known as the lag method. In this way, we decompose the original dataset with overlapping windows and condition the target value only on its lag. During the training phase, we treat both the step size (i.e. the number of days before a new window) and the window size (i.e. the size of the lag) as hyperparameters and tune them accordingly.

4. Models

In this project, we apply the sliding window approach with deep recurrent neural networks. As our first models, we start with naive recurrent networks, respectively LSTM and GRU models. As opposed to vanilla RNN, LSTM and GRU networks partially solve the problem of vanishing gradient by employing a gating mechanism to regulate the amount of information to carry from previous time steps (Hu et al., 2018). Our input has the following shape: (*batch_size*, *window_size*, *feature_size*). Moreover, we assume that there exists a local pattern that we can leverage to predict future prices. In fact, in a third model, we make use of CNN layers to extract such information across time steps. It consists of a convolutional layer followed by an LSTM layer, then connected to two dense layers. The convolutional layer is meant to extract high level patterns that form across time steps inside a window. This information is then also passed through the recurrent unit in order to get enriched by means of temporal sequencial correlation. As a fourth model, we employ the attention mechanism over our input data and then feed it to an LSTM layer. In particular, we employ a multi-head self-attention mechanism *to jointly attend to information from different representation subspaces at different positions* (Vaswani et al., 2017). Similarly to the previous model, here the idea is to process the raw input sequence by enriching it via a preliminary layer, and then extracting its sequential information via the LSTM layer. However, in this case the attention layer operates differently compared to the CNN one. In fact, it works by putting the focus on specific parts of the input sequence. For this task, it can be useful since it can understand that specific subsequences of financial data are more meaningful than others and give those more importance.

5. Hyperparameters

In this section, we present a subset of the hyperparameters used. However, it is not intended to be an exhaustive list of all the hyperparameter. The list of runs is logged using wandb (Biewald, 2020) and can be reached at link.

Table 1. List of hyperparameters.

Hyperparameter	Value
Window size	20
Step size	1
Optimizer	SGD
Batch size	1024

6. Experimental results

As an evaluation framework, we consider multiple metrics in order to assess the quality of a particular model. Since

we are dealing with a regression task it comes naturally to adopt the Mean Squared Error (MSE) measure as a proxy to indicate how well a model performs. We also consider the R-squared statistical measure to check how similar the predicted and the observed data are similar. However, since we are aware that R-squared increases when adding more independent variables, we use the adjusted r-squared. Its formulation is similar to the one of r-square and only differs from the degrees of freedom. Finally, we measure the real performances of our system according to the return of a trading strategy. The system is designed to buy a fixed amount of stocks whenever the predicted price of the following day is higher than the current one. Then, it closes the position after one day and registers the gain or the loss. This way, the profit expectancies can be compared among all the models. Another metric that we took into consideration was the operation's accuracy. That is an unconventional metric that we decided to adopt to understand the fraction of times that the model predicted the correct trend. In other words, when the model predicted a price gain and the real outcome was actually a gain. The reason why we implemented these additional evaluation steps is because regression metrics were not enough to assess the goodness of the model in performing this task. In fact, a model could simply replicate the price of the previous day by applying the identity function and obtain decent MSE. This is because the relative change in price between two subsequent days is very small on average. However, such a model would make very careless predictions that would be of no help to a potential investor. **TODO:** explain what are our best performing models and the reasons why they perform so well.

Table 2. Performance comparison.

	MSE	R2	Adjusted R2	Operation accuracy (%)	Return (%)
Naive LSTM	0.045	0.985	0.955	**85.79**	419.12
Naive GRU	0.062	0.988	0.970	85.11	**422.87**
CNN-LSTM	0.033	0.988	0.968	83.52	390.87
Attention-based LSTM	**0.022**	**0.991**	**0.977**	0.8102	420.26

7. Conclusions

TODO: Add conclusions here ..

References

RSI – Relative Strength Index. https://en.wikipedia.org/wiki/Relative_strength_index.

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Hu, Y., Huber, A. E. G., Anumula, J., and Liu, S. Overcoming the vanishing gradient problem in plain recurrent networks. *CoRR*, abs/1801.06105, 2018. URL http://arxiv.org/abs/1801.06105.

Lo, A. W. *Adaptive Markets: Financial Evolution at the Speed of Thought*. Princeton University Press, 2017. ISBN 9780691135144. URL http://www.jstor.org/stable/j.ctvc77k3n.

Mehtab, S., Sen, J., and Dutta, A. Stock price prediction using machine learning and lstm-based deep learning models, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

Zou, Z. and Qu, Z. Using lstm in stock prediction and quantitative trading, 2020.