Machine Learning for Time Series Forecasting

Elena Novikova
Insitute of Computer Science
University of Tartu
elena.novikova@ut.ee

***Abstract* — In this project, I will focus on time series forecasting. Time series are usually characterized by nonstationary, strongly autocorrelation, and heteroscedasticity. Traditionally statistical methods or machine learning approaches like SVM and Regressions are used for forecasting purposes. However, in recent research, it is shown that neural networks are much better at handling non-linear models.**

Keywords – time series, forecasting, LSTM

Github:
https://github.com/ElenaSNovikova/DataScienceProject

I. DATA

While working on this project, I tried to work with various types of data. After initial analysis, I decided to focus on financial data that I have the most knowledge of. While looking to specify further, I noticed that this year at UT there were several Master's and bachelor's thesis dedicated to time series forecasting on financial data. As a result, I decided to continue working on the financial data from my previous project (Introduction to Data Science course). I focus on Amazon stock prices from 16.05.1997 till 10.11.2017. These time series are continuous time series without null values, duplicates or potential outliers. It shows characteristics specific to the majority of financial time series: non-stationarity, strong autocorrelation and heteroscedasticity.

TABLE I. INITIAL DATA

Fig. 1. Close stock prices from 16.05.1997 till 10.11.2017

II. PRE-PROCESSING

After concluding the initial exploratory analysis, I faced the problem of how I should correctly process this data. To solve the problem of non-stationarity, I tried several solutions. In most cases taking the first difference will correct this problem. However, it would be hard to draw conclusions from the resulting time series. Another approach is to calculate the return rates (arithmetical or log). It would provide us with new time series that could be easily understood from the financial perspective. However, when using machine learning techniques, it would lead to a higher mean squared error. In the end, I normalized the data with MinMaxScaler. I tried to smooth the data further with an exponential moving average, yet it only led to worse results.

TABLE II. NORMALIZED DATA

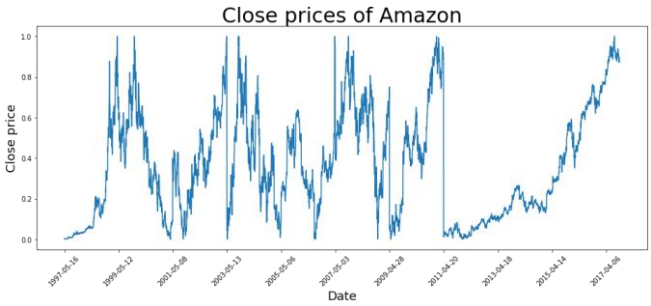

Fig. 2. Data normalized with MinMaxScaler

III. ONE-STEP PREDICTION

To show the possibility to predict the close prices only one step ahead, I used standard average (MSE 0.00510) and exponential moving average (MSE 0.00079).

TABLE III. ONE-STEP PREDICTION VIA STANDARD AVERAGE

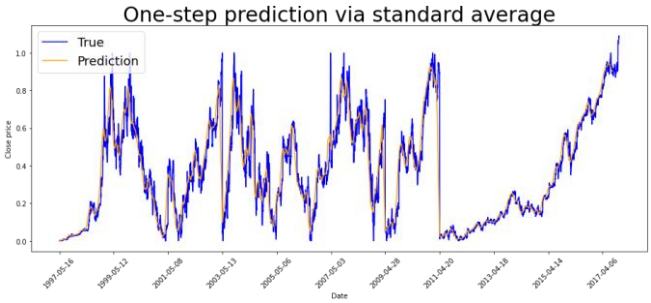

Fig. 3. MSE 0.00510

TABLE IV. ONE-STEP PREDICTION VIA EXPONENTIAL MOVING AVERAGE

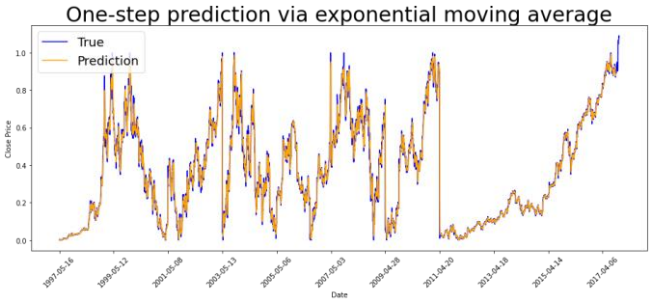

Fig. 4. MSE 0.00079

IV. ARIMA

To solve the problem of the non-stationarity I took the first difference. As a result, for the ARIMA model the term d is

equal to 1. Afterwards, I plotted autocorrelation and partial autocorrelation to determine other parameters of the ARIMA model.

TABLE V. AUTOCORRELATION OF THE FIRST DIFFERENCE

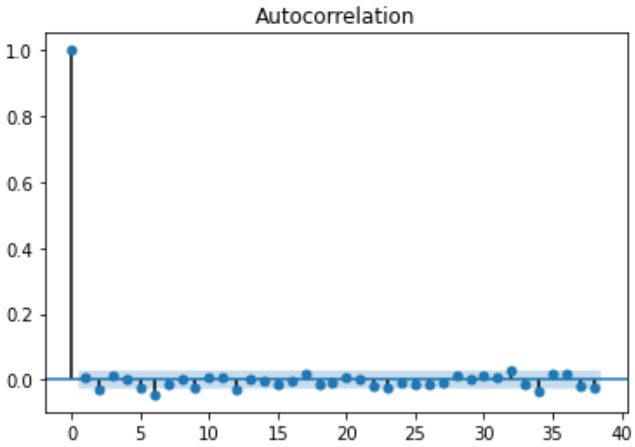

Fig. 5. For the model ARIMA the term q as 6

TABLE VI. PARTIAL AUTOCORRELATION OF THE FIRST DIFFERENCE

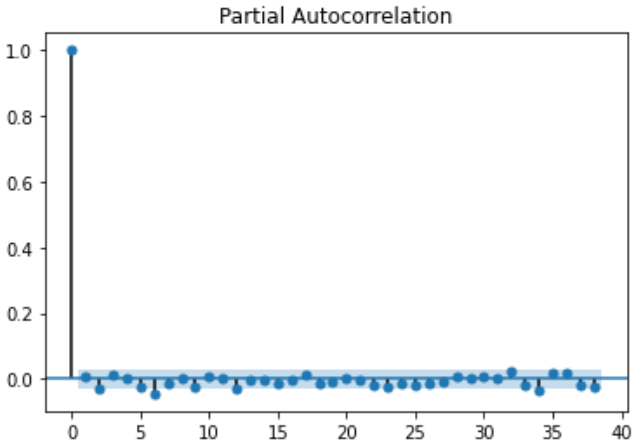

Fig. 6. For the model ARIMA the term p as 2

After plotting autocorrelation and partial autocorrelation, I determined the optimal model as ARIMA (6,1,2).

I also tried different combinations of AR and MA terms. According to AIC and BIC criteria, the statistical model ARIMA (6,1,2) was of higher quality. I also tried to determine the terms of the ARIMA model automatically through the package pmdarima. However, the resulting optimal model ARIMA (0,1,0) obviously provided a worse forecast (MSE > 0.01). While comparing forecast from ARIMA (6,1,2) with the test dataset, it is evident that the model allows for good estimates of future close prices.

TABLE VII. IMPLEMENTATION OF ARIMA(6,1,2)

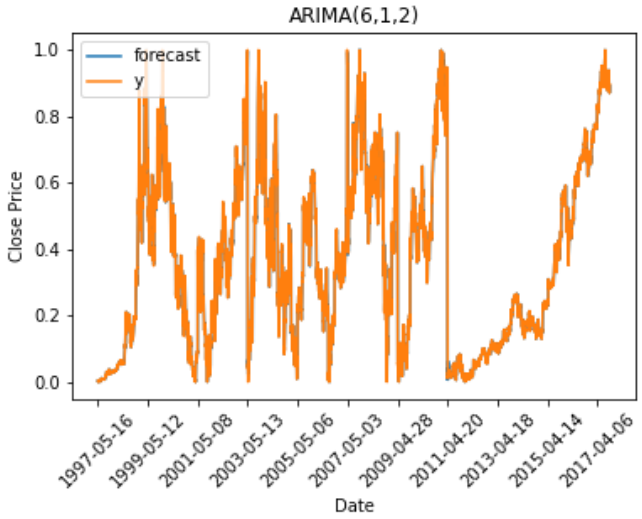

Fig. 7. ARIMA(6,1,2) on the train data

TABLE VIII. PREDICTION VIA ARIMA(6,1,2)

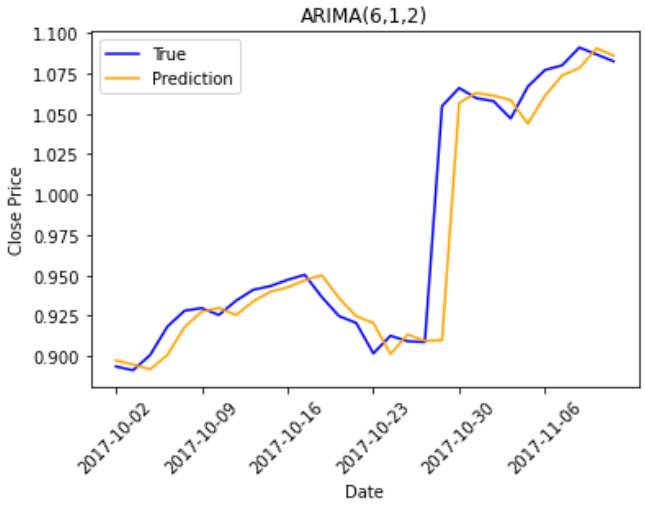

Fig. 8. MSE 0.00079

V. LSTM

After implementing LSTM via regression MSE for the training data was equal to 0.00130, while MSE for the test data was equal to 0.00089.

TABLE IX. IMPLEMENTING LSTM

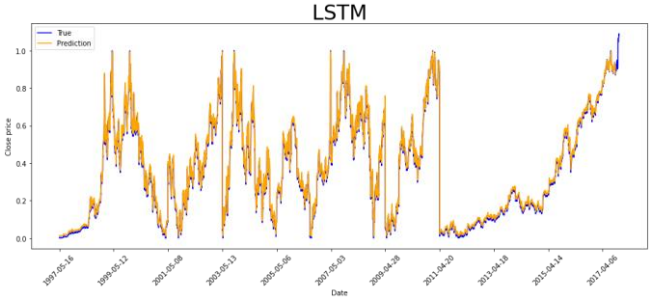

Fig. 9. MSE 0.00130

TABLE X. PREDICTION VIA LSTM

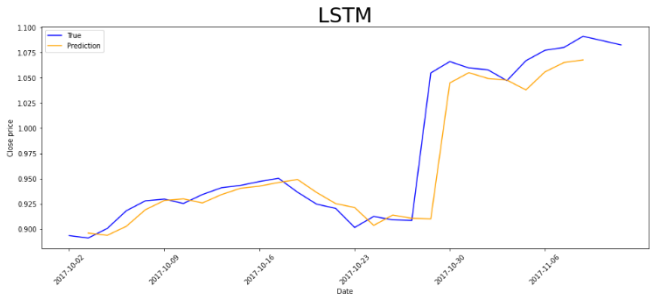

Fig. 10. MSE 0.00089

TABLE XI. TRAINING AND VALIDATION LOSS

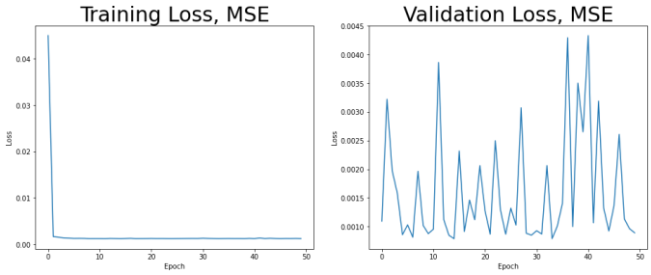

Fig. 11. Training and validation loss for 50 epochs

VI. CONCLUSIONS

The initial data showed the standard problems of non-stationarity, strong autocorrelation and heteroscedasticity. The data was normalized with MinMaxScaler, so that machine learning could be used correctly. One-step prediction, especially via exponential moving average provided a good estimate for a one day ahead prediction. ARIMA(6,1,2) gave estimates with the same MSE(0.00079). After implementing LSTM MSE on the test data was equal to 0.00089. Although it is higher than MSE for the ARIMA model, LSTM for regression only took into account the previous value of the close stock prices.