Deree - The American College of Greece
Finance department

Comparing Neural Networks to Statistical Approaches on Time-Series Forecasting

Quantitative Finance Project - FN 4555

Gavalas Nikolaos 245188

Professor: Dr. Nikiforos Laopodis

Athens, Apr. 2021

Table of Contents

Introduction

Forecasting of financial variables is a highly quantitative and interdisciplinary field, combining knowledge and methods from the fields of Finance, Economics, Statistics and Computer Science. This is due to its close linkage with Data Analysis, especially Time-Series Analysis.

Since recent advancements in Time-Series Analysis are focused around Machine Learning and Neural Networks, many researchers, institutions and companies have started exploring their capabilities in financial applications, sometimes with great success, outperforming more classical statistical approaches.

In this work, we will examine a particular class of Neural Networks-based models which have shown to be highly effective in Time-Series Forecasting according to the relevant literature, known as Recurrent Neural Networks (RNN) and more specifically, a subgenre of them called Long Short-Term Memory (LSTM). Then, we will proceed to compare them with two well-known more “traditional” statistical models, the Auto-Regressive Integrated Moving Average (ARIMA) and Vector Autoregression (VAR), from a financial standpoint.

Short Literature Review

Financial time-series forecasting, and principally stock price forecasting, is one of the most studied financial applications.

In earlier research, Chen, Zhou & Dai (2015) and Dezsi & Nistor (2016) have performed forecasting with LSTMs and raw stock price data. Samarawickrama & Fernando (2017) and Hiransha & Soman (2018) have done thorough comparisons of RNNs and other RNN-like models like LSTMs and GRUs for predicting stock price movements, whereas Selvin, Vinayakumar, Menon & Soman (2017), have done similar work but included the ARIMA model in their comparisons. Zhou (2019) has achieved good monthly returns after constructing portfolios based on LSTM models predicting stock price movements. Other researchers, like McNally, Roche & Caton (2018) have compared RNNs and LSTMs to ARIMA, and found that the former outperformed the latter, but used the price of cryptocurrencies instead of stocks. In another interesting work, Kraus & Feuerriegel (2017) have combined in an LSTM both textual data from automated analysis of news sources and stock price data. This is very important because it shows that Neural Networks can combine inputs of various forms.

To our best knowledge, there is no work including VAR in a comparison of statistical models and Neural Networks.

Theoretical and Empirical Methodology

Data

The comparison of the statistical models to RNNs that we will perform, will be on the basis of time-series forecasting, and more specifically, stock price forecasting. We chose stocks as financial assets and not indices, bonds or economic variables, because we believe that this is the kind of asset that is more relevant, since stock prices are more heavily influenced by technical analyses, and contain high amounts of noise. This is because of the impact of HFT (High-Frequency Trading) and short-term traders.

The selected stocks are: $AAPL, $AMZN, $JNJ, $XOM, $GM. These are all blue-chip companies, with high trading volume (which is important in this type of analysis), and from different industries. The time period of the data is March 2020 to March 2021, and the frequency is daily (relatively short-term), because this is where technical analysis is more relevant, as opposed to the long-term, where fundamentals are the most important factor. Finally, the source of the data is Yahoo Finance.

Theoretical Background

ARIMA

The Auto-Regressive Integrated Moving Average model is a statistical model used for Univariate Time-Series Forecasting. It combines an Auto-Regressive part (AR) with a Moving Average (MA) part, as well as an Integration part (I) which is used to transform the series into a series with certain statistical properties which are required for the model to work, turning them into *stationary* series.

This model essentially is based on the idea that each forecast value of a series is a linear combination of the series past values (or *lagged* values) and forecast errors (or *residuals*). For p lags of the AR part and q lags of the MA part, the model for the integrated stationary variable y is specified as:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

VAR

The Vector Autoregression model is another popular statistical model. It is inspired by the ARIMA model, as it is also based on the assumption of the series' values being linearly combined with the lags and the residuals, but differs in some points. Specifically, it is a Multivariate model that works with systems of variables, which we assume that they influence each other (without needing however to specify *how* or *why* they do so). The model is specified by the following system of equations, for p lags and k variables:

$$y_{1,t} = c_1 + a^1_{1,1} y_{1,t-1} + a^1_{1,2} y_{2,t-1} + \cdots + a^1_{1,k} y_{k,t-1} + \cdots + a^p_{1,1} y_{1,t-p} + a^p_{1,2} y_{2,t-p} + \cdots + a^p_{1,k} y_{k,t-p} + e_{1,t}$$

$$y_{2,t} = c_2 + a^1_{2,1} y_{1,t-1} + a^1_{2,2} y_{2,t-1} + \cdots + a^1_{2,k} y_{k,t-1} + \cdots + a^p_{2,1} y_{1,t-p} + a^p_{2,2} y_{2,t-p} + \cdots + a^p_{2,k} y_{k,t-p} + e_{2,t}$$

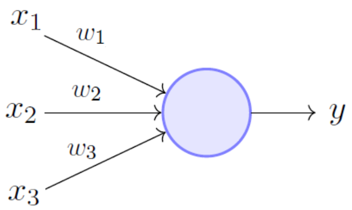
$$\vdots$$

$$y_{k,t} = c_k + a^1_{k,1} y_{1,t-1} + a^1_{k,2} y_{2,t-1} + \cdots + a^1_{k,k} y_{k,t-1} + \cdots + a^p_{k,1} y_{1,t-p} + a^p_{k,2} y_{2,t-p} + \cdots + a^p_{k,k} y_{k,t-p} + e_{k,t}$$

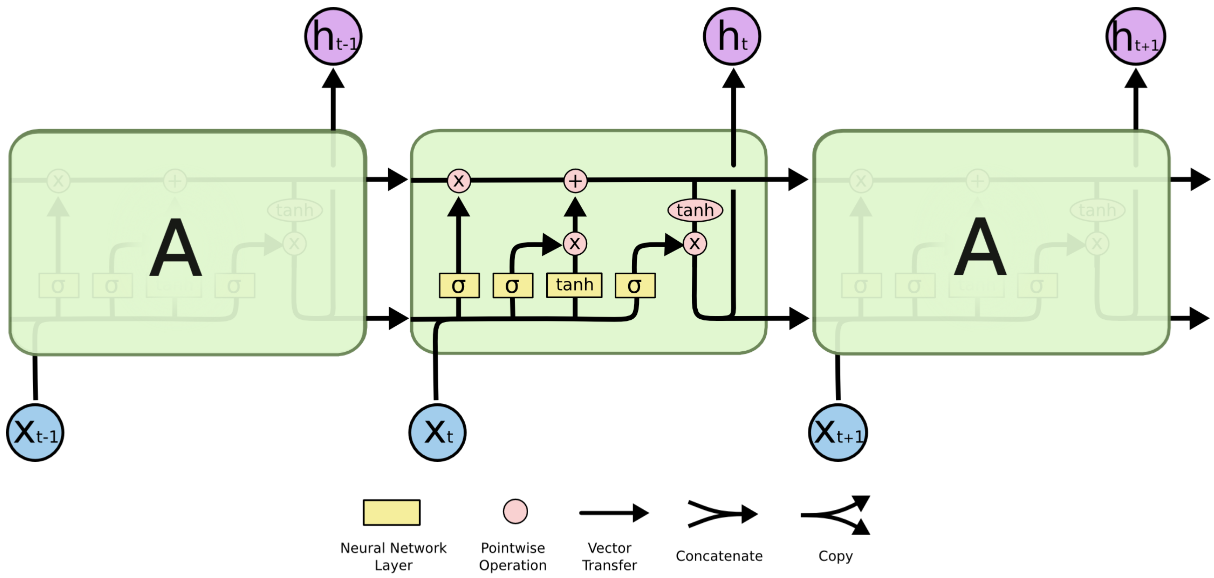
RNN

The Recurrent Neural Network is a special type of Artificial Neural Network (ANN) which "feeds" the outputs of the model back into the input, in order to achieve a "memory" effect and capture patterns of sequences of the data. This type of ANN, due to its nature, has been applied and found most effective on predicting sequential data, such as text or time-series.

An ANN consists of interconnected layers of artificial neurons (*perceptrons)*, like the ones shown in the figure below. Each neuron receives the inputs, applies weights, aggregates them, and maps them to the output through an *activation function*, inspired by the way the actual neurons work.



The RNN on the other hand, feeds the output *y* back into the cell. In this work, instead of using simple RNNs, we will use the Long Short-Term Memory architecture. The LSTM is a specialized type of RNN that works exceptionally well with time-series, because it does not suffer from an issue known as "long-term dependency problem". Unrolled in time, the LSTM cell looks like the figure below:



Neural Networks in general learn a function by adjusting their *weights* to fit the training data. This is done through a process called *backpropagation*, which is a numerical optimization procedure based on the numerical gradient of the *loss function*, a function that measures how far are the predicted outputs from the expected outputs.

The main advantage of Neural Networks over other classes of models, is that they can learn functions without any assumption on the characteristics of the input data. That is why they are well-suited for nonlinear functions.

Methodology

For the statistical models, we check the series for stationarity using the Augmented Dickey–Fuller test (ADF). If they are not stationary, we transform them into log returns (because we have stock price data):

$$y_t = \log \frac{y_t}{y_{t-1}} = \log y_t - \log y_{t-y}$$

For selecting the optimal models, we follow the Box-Jenkins approach, and we select the model with the lag order(s) that minimize the Akaike Information Criterion (AIC).

We split the data into two sets, a *training* set and a *test* set. We want the length of the prediction to be 7 days into the future, so the test set will contain the last 7 observations of our initial data. The test set will be used to compare the results with the actual values and evaluate the forecast.

After performing forecasting, we revert the transformations, plot them, and evaluate them using three different metrics.

Concerning the tools we used, we worked in Python, a powerful programming language, using the software libraries *statsmodels* for the statistical models and *tensorflow* for the ANNs, which are both considered state-of-the-art.

Evaluation

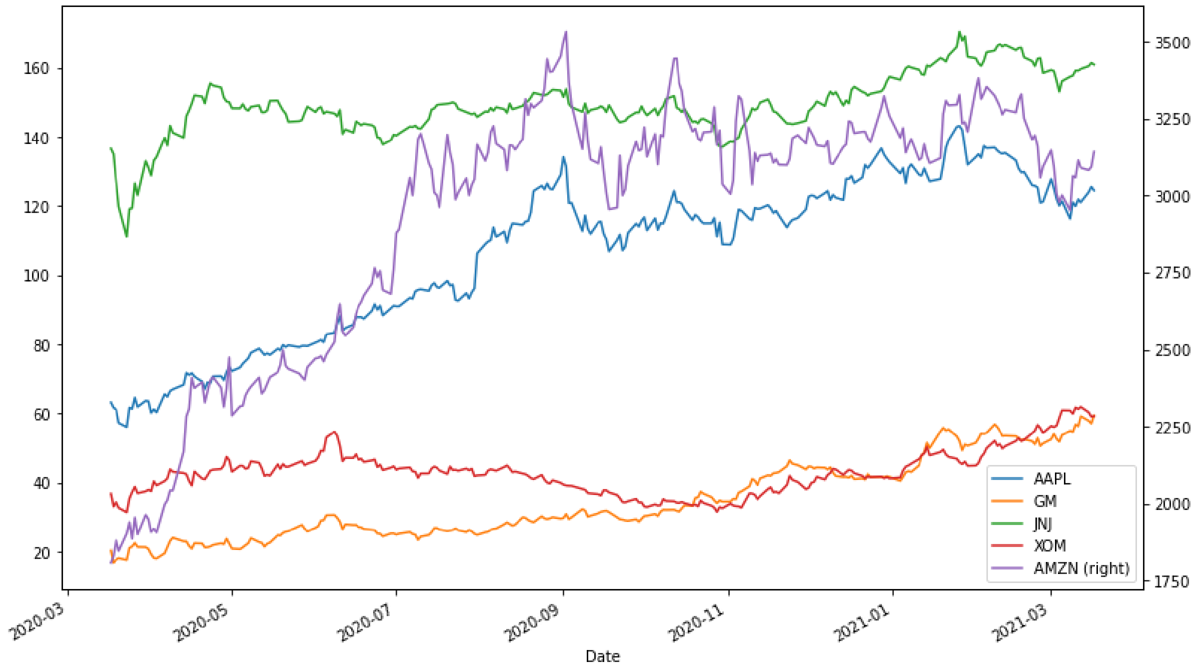
The metrics we use for evaluation are the RMSE, MAE and MSE, which stand for Root Mean Square Error, Mean Absolute Error, and Mean Square Error respectively:

$$RMSE = \sqrt{\sum_{i=0}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$$MAE = \sum_{i=0}^{n} \frac{|\hat{y}_i - y_i|}{n}$$

$$MSE = \sum_{i=0}^{n} \frac{(\hat{y}_i - y_i)^2}{n}$$

Empirical Results and Discussion



Plot of the stocks' prices we used.

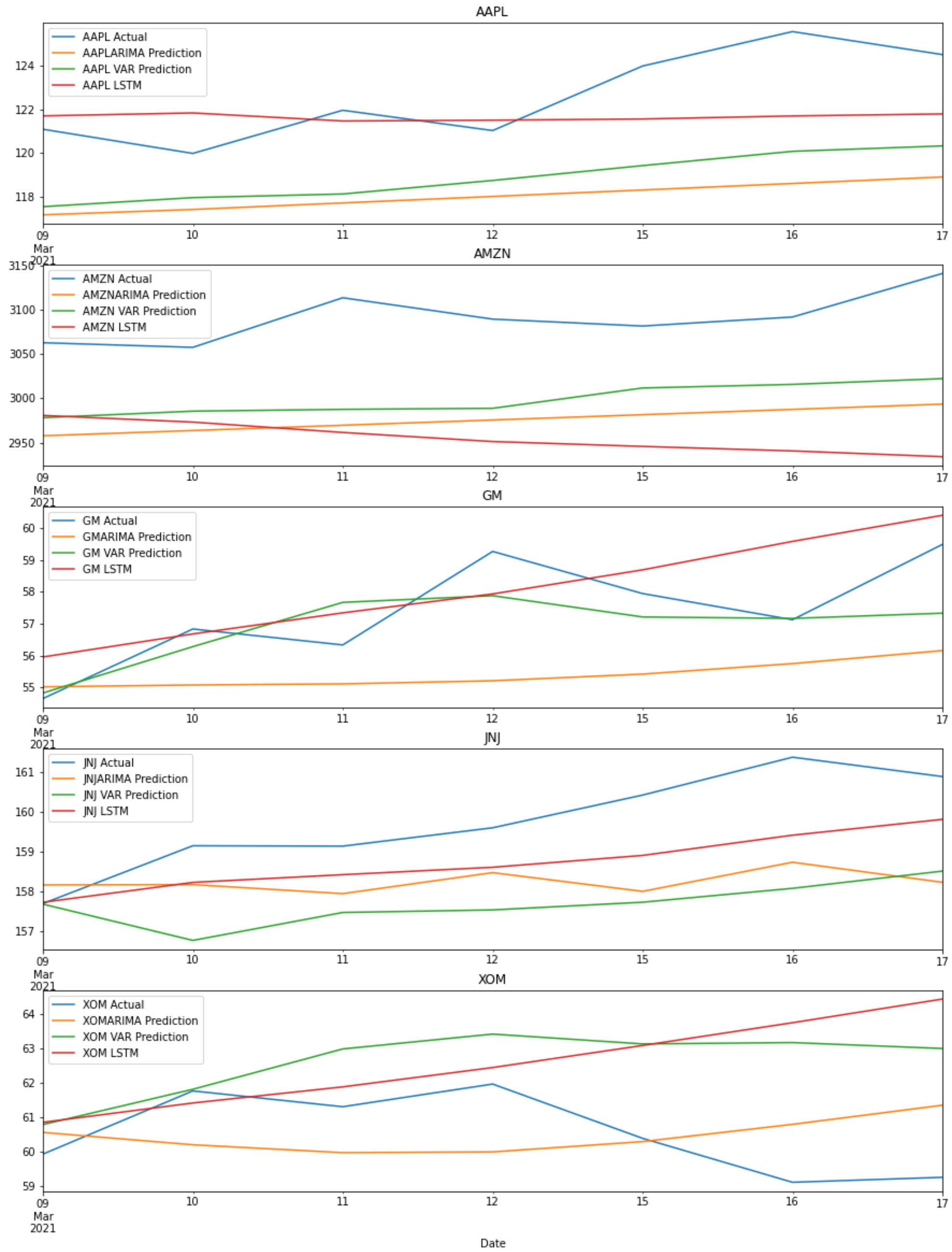
Firstly, the ADF Tests failed, so none of the series was initially stationary (as expected). We converted them to log-returns and then they were passing the ADF test.

The optimal lag order for the VAR model indicated by the AIC was 5 lags, and for the ARIMA were:

Stock	*Optimal ARMA Order*
AAPL	(1, 0)
AMZN	(0, 0)
GM	(3, 2)
JNJ	(4, 2)
XOM	(2, 2)

All the outputs of the models and shown in the accompanying file.

The forecasts were the following:



The resulting evaluation metrics are presented below:

Stock	*RMSE*	*MAE*	*MSE*
AAPL	3.87	3.70	15.05

AMZN	94.93	92.54	9012.54
GM	1.14	0.91	1.31
JNJ	2.27	2.06	5.18
XOM	2.50	2.08	6.26

VAR model

Stock	*RMSE*	*MAE*	*MSE*
AAPL	4.80	4.57	23.0
AMZN	117.16	115.45	13726.74
GM	2.40	2.08	5.77
JNJ	1.83	1.63	3.37
XOM	1.50	1.34	2.25

ARIMA model

Stock	*RMSE*	*MAE*	*MSE*
AAPL	2.15	1.78	4.65
AMZN	141.45	135.75	20010.61
GM	1.30	1.12	1.70
JNJ	1.17	1.02	1.37
XOM	2.86	2.12	8.19

RNN LSTM model

It is evident that the results are very close for all the models, with the ARIMA and the LSTM performing slightly better than the VAR. The plots show that all of the models were relatively close to the actual price movement, but failed to capture its movements with precision.

Summary and Conclusions

Recurrent Neural Networks may be promising, but in order to reach a more compelling conclusion about the cases where they are superior to statistical analysis, or the opposite, it is evident that more rigorous empirical research is needed. We recommend experimentation with diverse data including stocks, bonds, indices, and macroeconomic time-series data, in order to reach a sound conclusion about the time-series forecasting scenarios where each model is more suitable.

Bibliography

Chen, K., Zhou, Y., & Dai, F. (2015, October). A LSTM-based method for stock returns prediction: A case study of China stock market. In 2015 IEEE international conference on big data (big data) (pp. 2823-2824). IEEE.

Dezsi, E., & Nistor, I. A. (2016). Can deep machine learning outsmart the market? A comparison between econometric modelling and long-short term memory. Romanian Economic and Business Review.

Samarawickrama, A. J. P., & Fernando, T. G. I. (2017, December). A recurrent neural network approach in predicting daily stock prices and application to the Sri Lankan stock market. In 2017 IEEE International Conference on Industrial and Information Systems (ICIIS) (pp. 1-6). IEEE.

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2018). NSE stock market prediction using deep-learning models. Procedia computer science, 132, 1351-1362.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-sliding window model. In 2017 international conference on advances in computing, communications and informatics (icacci) (pp. 1643-1647). IEEE.

Zhou, B. (2019). Deep learning and the cross-section of stock returns: Neural networks combining price and fundamental information. Available at SSRN 3179281.

McNally, S., Roche, J., & Caton, S. (2018, March). Predicting the price of bitcoin using machine learning. In 2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP) (pp. 339-343). IEEE.

Kraus, M., & Feuerriegel, S. (2017). Decision support from financial disclosures with deep neural networks and transfer learning. Decision Support Systems, 104, 38-48.

Wordcount: 1852 (1609 without including the bibliography)