Big Data Content Analytics

Alexiou Dimitrios F2821802
Fotopoulos Spyridon F2821810

May 2020

1 Description

In the past years, the deep learning sector has seen a huge increase in interest, not only academically but in many business implementation. One of the sectors that has been known to take particular interest in various deep learning and machine learning technologies is the financial sector. Large amounts of money and work are thrown in research and development, in pursuit of a small edge in the market. A small edge in these sector could be translated into many millions, if not billions, in gains. Particularly, in this exercise, we engage with the scenario of providing a reliable prediction, about the next day close price of a particular stock, traded in a stock exchange, on behalf of an actively managed fund.

Active investing, as its name implies, takes a hands-on approach and requires that someone act in the role of portfolio manager. The goal of active money management is to beat the stock market's average returns and take full advantage of short-term price fluctuations. It involves a much deeper analysis and the expertise to know when to pivot into or out of a particular stock, bond, or any asset. A passively managed fund (usually referred as index fund), by contrast, simply follows a market index. It does not have a management team making investment decisions. Active funds make it possible to beat the market index and several funds have been known to post huge returns. Although it is worth noting that, statistically speaking, most actively managed funds tend to under-perform, or do worse than, the market index.

If our implementation is considered success full and gets used by an actively managed fund, it will provide an insight(prediction), on the closing price of a stock for the next business day. This way the fun can for example, either buy more stocks in the markets opening expecting a gain by the end of the day, or sell a position they already ahead of the expected price reduction, in order to protect their gains.

2 Mission

Any publicly traded company's stock, in any of the stock exchanges around the world, is characterized mainly by the following 5 metrics:

- Open Price: The price of the stock at the start a trading day, not always equal to the previous trading day Close Price, due to after hours trading.
- Close Price: The last price with which a stock was agreed to change hands before the closing "bell" of the exchange.
- High Prices: The highest prices achieved in a transaction of a stock during a trading day.
- Low Prices: The lowest prices achieved in a transaction of a stock during a trading day.
- Volume: The total amount of individual stocks that changed hands during a trading day.

In our implementation, we focus to predict the Close price of a stock, using the other 4 metrics as predictors. Our goal is to provide a reliable prediction about the trend (upwards, downwards or unchanged) that the stock's Close price will have in the next trading session, as well as a prediction about the exact Close price in will reach.

3 Data

The largest historical dataset of stocks prices and their metrics we were able to find, consisted of stocks included in the S&P 500 index, from 1980 up to 2018. This dataset contained over 30 years of trading data (depending on the stock, considering that some the companies did not exist or were not trading in 1980), with thousands trading days for each stock. Since 500 stocks over 30 years of trading translates to a huge dataset, we extracted 25 of those stocks, chosen by either their technology focus or by their size and history.

The data went through various stages of cleaning and reformatting, in order to achieve a consistent dataset, especially in the formatting of dates, as well as dropping any non conforming data. After the cleaning the dataset looked as follows:

	Date	Close	Volume	Open	High	Low	Company
12322	01-02-1970	7.140914917	22500	7.140914917	7.225019932	7.140914917	AA
34201	01-02-1970	6.699775219	24100	6.699775219	6.770052433	6.699775219	ARNC
81427	01-02-1970	11.16975021	24400	11.0994997	11.20487499	10.99412537	CNP
93750	01-02-1970	0.683144331	1109700	0.688280761	0.689564824	0.683144331	DIS
141129	01-02-1970	0.798177063	2227200	0.807291687	0.80859375	0.796875	GE
...	...	...	...	...	...	...	...
212939	01-02-1976	1.448489547	21200	1.448489547	1.498437524	1.448489547	MATX
224420	01-02-1976	1.114583373	1108800	1.104166627	1.114583373	1.104166627	MO
236743	01-02-1976	1.934027791	680400	1.923611164	1.9375	1.916666627	MRK
249066	01-02-1976	10.83227444	81100	10.77013874	10.83227444	10.77013874	MRO
269620	01-02-1976	0.475710779	16600	0.475710779	0.545668244	0.475710779	PNR

The most important transformation applied on the data is the MinMaxScaler, used to transform the target variable and all the features into the same scale and into the same range (0 to 1), a process especially useful considering that Neural Network are sensitive to input values, and even though most of our inputs were dollars amounts, the Volume feature was a completely different quantity. The scaler used for each variable, is saved, and used to reverse scale in the end of the process.

4 Methodology

Theory

Many algorithms have been developed through the years in order to predict the stock prices. Moving Average, Linear regression and ARIMA are only a few of them. However, in the later years a lot of deep learning algorithms have also implemented in order to solve this difficult and complicated problem. We decide to use a Long Short-Term Memory (LSTM) Artificial Neural Network(ANN), to predict the stock prices since LSTM have proved to be one of the most powerful and effective models in processing sequential data.

LSTM is a type of Recurrent Neural Network (RNN) that follows a different architectural approach from a traditional RNN. In general ANNs are computational systems inspired by the way the human brain works and they are used in various tasks such as speech recognition, time series prediction, image processing etc. The main architecture of an ANN consists of an input layer, a hidden layer and an output layer. Each layer have some preceptors which called neurons. Each neuron receive inputs x multiplied by some constant values w called weights. The output of the neuron will be one or zero depending on if the value of weighted sum is greater than a specified threshold.

RNNs are multi-layer type of networks that use internal memory in order to process sequential data [fig.1].For each time step t , the input $x(t)$ and the activation $a(t-1)$ produce the output $y(t)$. Although RNN are very effective in sequential data their major disadvantage is that they are suffering for vanishing gradient. Vanishing gradient has to do with the fact that even big changes in the parameters' values of early layers has not any serious effect in the output of the neural network.

LSTM solve this problem with different approach in its architecture [fig.2] and its basic elements are:

- Forget Gate: The forget gate decides what information should be thrown away or kept. This has been succeeded through a sigmoid function where information from the previous hidden state and information from the current input is passed and transformed into one or zero values (1-pass/0-forget).

- Input gate: The input gate filters the new information and updates the current cell state. The input gate consists of a sigmoid function which decides which values are important and a *tanh* function that squishes the values to $[-1, 1]$. The results of sigmoid and tanh functions are multiplied in order to take at the final output of input gate only the important information.

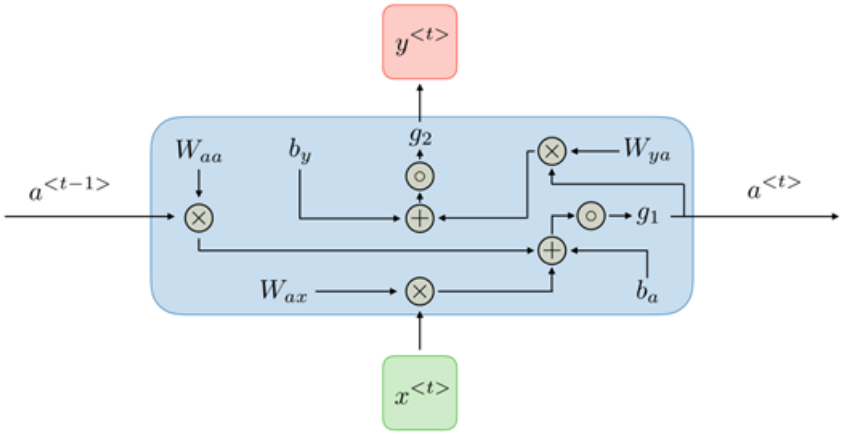


Figure 1: Recurrent Neural Network Architecture

- Cell state: Cell state is like the "memory" of the LSTM. At first, there is the $Ct-1$ state which is multiplied with the output of forget gate in order to drop the values which are closed to zero. Then an addition with the output of input gate is taken place in order to update the values of the cell state and create the new state Ct in the memory.
- Output Gate: The output gate refers to the output of the LSTM. The Ct of memory is the input of the *tanh* function. The $[h+x]$ (hidden state h, new input x) are the input of sigmoid function. Then we multiple the 2 outputs to decide what information the new hidden state should carry.

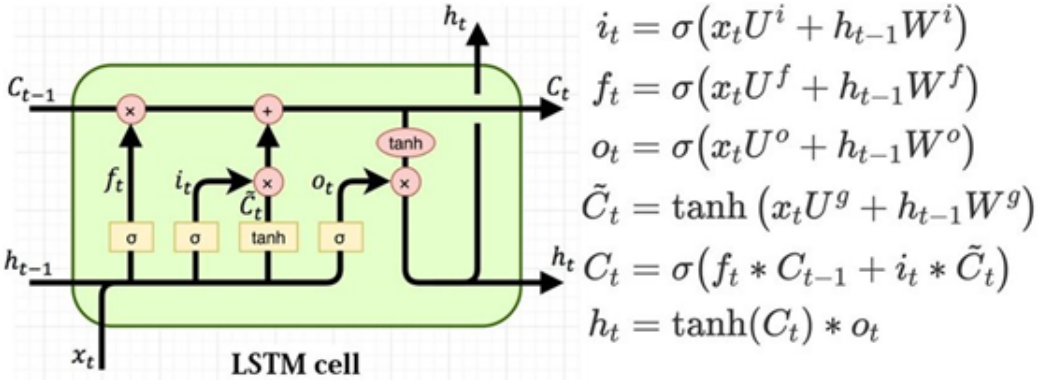


Figure 2: Long Short-Term Memory Neural Network Architecture

Tools Used

The initial data cleansing was made using bash and any further cleaning, processing and analysis has been made using python code and the some python frameworks such as Pandas, Numpy and Scikit-Learn. For the visualization we used the matplotlib and the seaborn which are both also python frameworks. In order to train our stacked-LSTM model we used Keras

which is a Python framework, that provides and API for neural networks running on top of TensorFlow, the well-known free software library focused on machine learning. Finally, the tunning of models have been made using Talos, a package provided by *Autonomio*. Talos is a solution that helps finding hyperparameter configurations for Keras models.

Design & Implementation

We created three different LSTM architectures in order to predict the stock market price. The main difference of these three architectures is the percentage of the previous information that is used in order to predict the next closing price. The reason for these different approaches is that we want to examine if the previous values are affected the output in a positive or negative way.

The input of our LSTM models are 3D tensors (*No.* of sequences, length of sequence, *No.* of features) and in order to reduce overfitting we have used drop out , which is a value that indicates the number of cells to be dropped out during the process.

The first model, labeled "MLP", takes as an input 1 sequence with all the values of the training dataset with 5 features and the top level architecture can be seen in [fig.3a]. In this model, we have 3 stacked lstm layers and time distributed dense layer. The time distributed wrapper applies the same task to every time step and provides hooks to get output after each time step instead of waiting for the whole sequence to be processed.

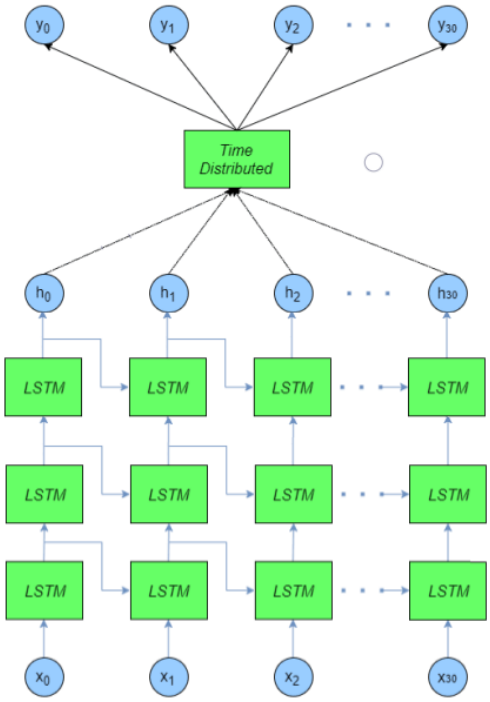
In the second model, labeled "Window" we have 3 stacked LSTM layers and dense layer. The dense layer has as an output only one feature which is the close price. The choice of the appropriate length of the sequence is a crucial factor for this model[fig.3b].

The third model, labeled "All-In" uses the architecture of the first model, but we change the input format in order to eliminate the influence of the previous values of the sequence. So, the input format for this model is (*No.* of Close Prices the dataset, 1, 5).

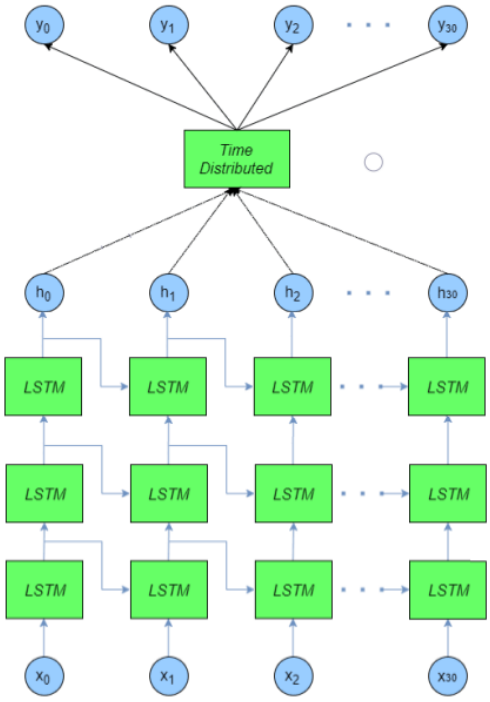
We found from the literature [1], [2], [3] that only some parameters are important in order to predict the future price and these are the parameters that we have used in Talos framework to tune our models. Talos provides many optimization strategies such as random search, grid search and probabilistic reduction. We chose to use grid search and for specific values we have created the parameter space that follows:

Number of units of the first LSTM layer	[8, 16, 32]
Number of units of the second LSTM layer	[8, 16, 32]
Number of units of the third LSTM layer	[8, 16, 32]
Dropout	[0.1 0.2]

Table 1: Talos Parameters Optimization Results



(a) LSTM Top Level Architecture Model 1 & 3.



(b) LSTM Top Level Architecture Model 2

Figure 3: Models Architecture

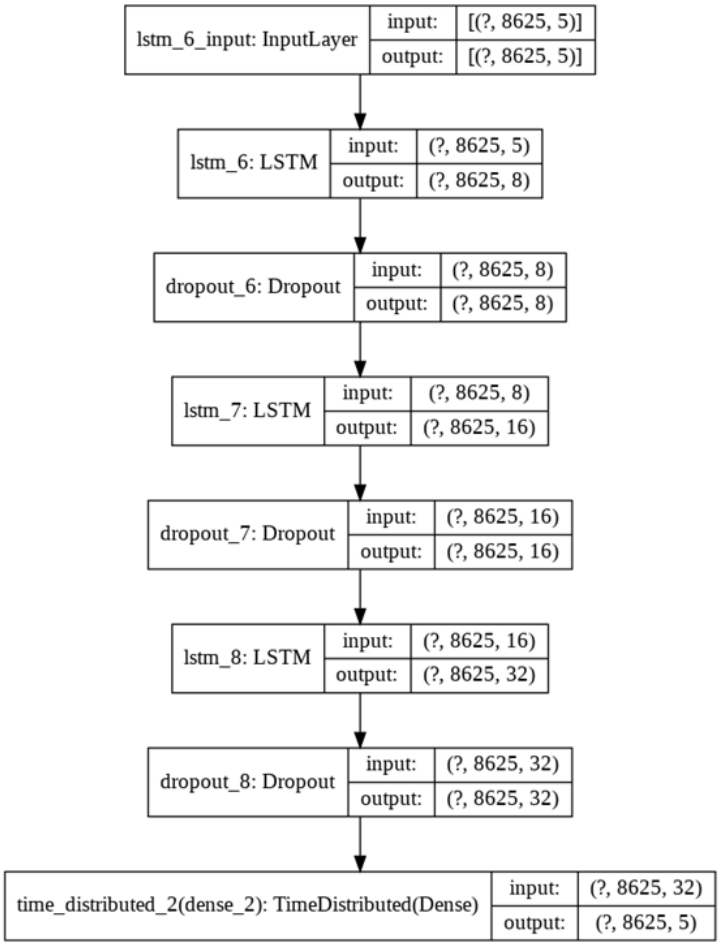


Figure 4: Model Layers

In all LSTM layers we used the default activation function which is tanh and the default recurrent activation function which is the sigmoid. Moreover, the metric that we used was the Mean Squared Error (MSE) for the validation loss and the optimization algorithm was the RMSprop. Finally, with trained the all the models using the historical values of one of the stocks only and then used the weights generated to predict the Close Price of all the other stocks we are testing. The logic behind this decision was that, we would first avoid any hindering of over-fitting, and second if the results were positive, we would have a solution that is significantly faster to deploy.

Results

As we explained before we use on stock for training and validation of the model, and all the other stocks that are passed through are in fact all test sets, since their datasets, in their entirety, have never been seen from the model. In general we see from all the models a relatively good performance, especially when considering that they at least follow the pattern of the close price, if not always the scale, even for stocks that have never been trained for, and follow different motives. The stock we chose for the training was $ARNC$.

Obviously, as can been seen in exact metrics in the Tables[2], [3], [4], with all the models we get the best performance from the stock we trained for. There are stocks suck as GE, that more or less have the same performance in the model as $ARNC$, and others that have more mediocre performanc ($MSFT$, $APPL$). In general though for every stock we tested we found that there was not a single one that any of the models gave an output completely detached from reality for the full time period, but there are instances were for large periods of time it fails to follow accurately predict the close price of a stock, such as the stock of IBM.

As can been seem from some of the figures, a common time period were the model would fail was the last 5 years of testing. When failing to predict the real closing price with significant margin, the output of the model was usually underestimating rather than over estimating.

Figures [5,11,17] present the original and the predicted values of the stock price of ARNC as calculated by the three models. The model, labeled MLP, that uses all the previous information in order to predict the next working day's close price seems to underestimate the close price in its prediction. The reason for this is that the past values, which are lower than the next ones, influence the model and when there is a steep increase in the close price the model makes less accurate predictions. So it is clear that the disadvantage of this model seems to be its inability to be adapted in the sudden changes of the close price.

The next model, labeled Window, which uses only a window of past values for the prediction have the ability to be more flexible and not so much affected by the sudden changes of the close price.

The third model, labeled All-In, that uses only the information of the previous day in order to make the prediction succeed to have the smallest RMSE (RMSE = 0.0220) than the first (RMSE= 0.0488) and the second model (RMSE= 0.0368).

All_In		ARNC	MSFT	GE	AAPL	IBM
Train Score	RMSE	0.026	0.0379	0.0263	0.0304	0.0528
	MSE	0.0007	0.0014	0.0007	0.0009	0.0028
Test Score	RMSE	0.0488	0.0420	0.043	0.0450	0.0467
	MSE	0.0024	0.0018	0.0018	0.002	0.0022

Table 2: RMSE and MSE Scores of All In model

All_In		ARNC	MSFT	GE	AAPL	IBM
Train Score	RMSE	0.0029	0.2712	0.0115	0.0181	0.0019
	MSE	0.0000	0.0736	0.0001	0.003	0.0000
Test Score	RMSE	0.0220	0.0482	0.0609	0.0598	0.0524
	MSE	0.0005	0.0023	0.0037	0.0036	0.0027

Table 3: RMSE and MSE Scores of MLP model

All_In		ARNC	MSFT	GE	AAPL	IBM
Train Score	RMSE	0.0253	0.0102	0.0259	0.0032	0.0135
	MSE	0.0006	0.0001	0.0007	0.0000	0.0002
Test Score	RMSE	0.0368	0.0384	0.0328	0.0365	0.0633
	MSE	0.0014	0.0015	0.0011	0.0013	0.0040

Table 4: RMSE and MSE Scores of Window model

MLP Model Plots

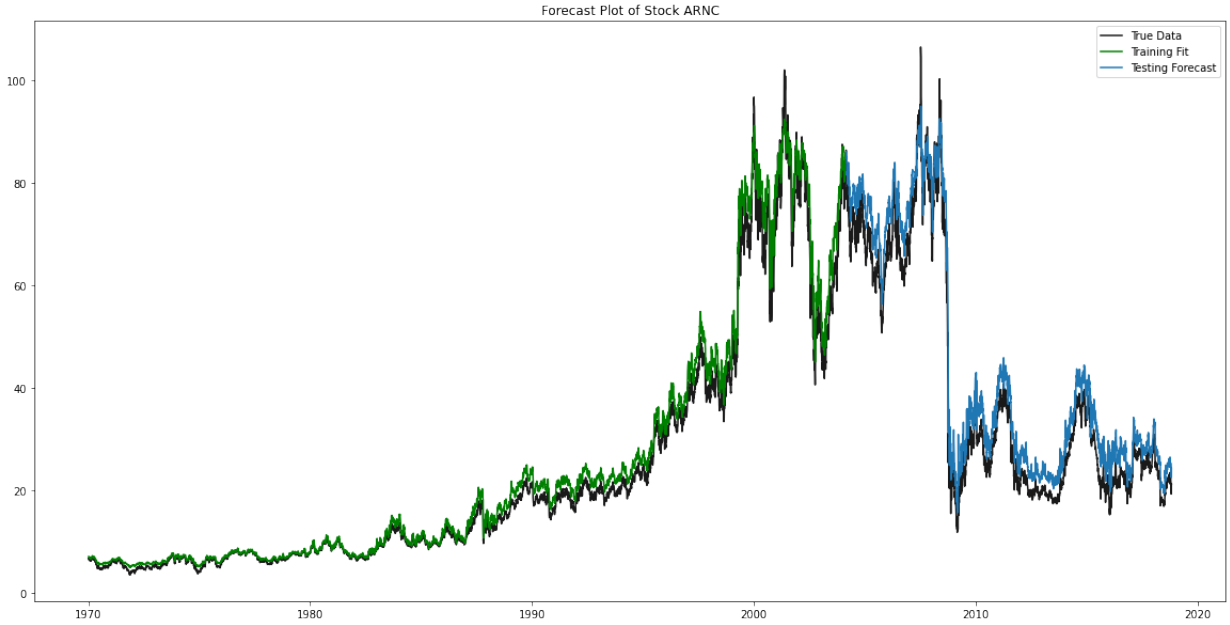


Figure 5: MLP Model Stock : ARNC

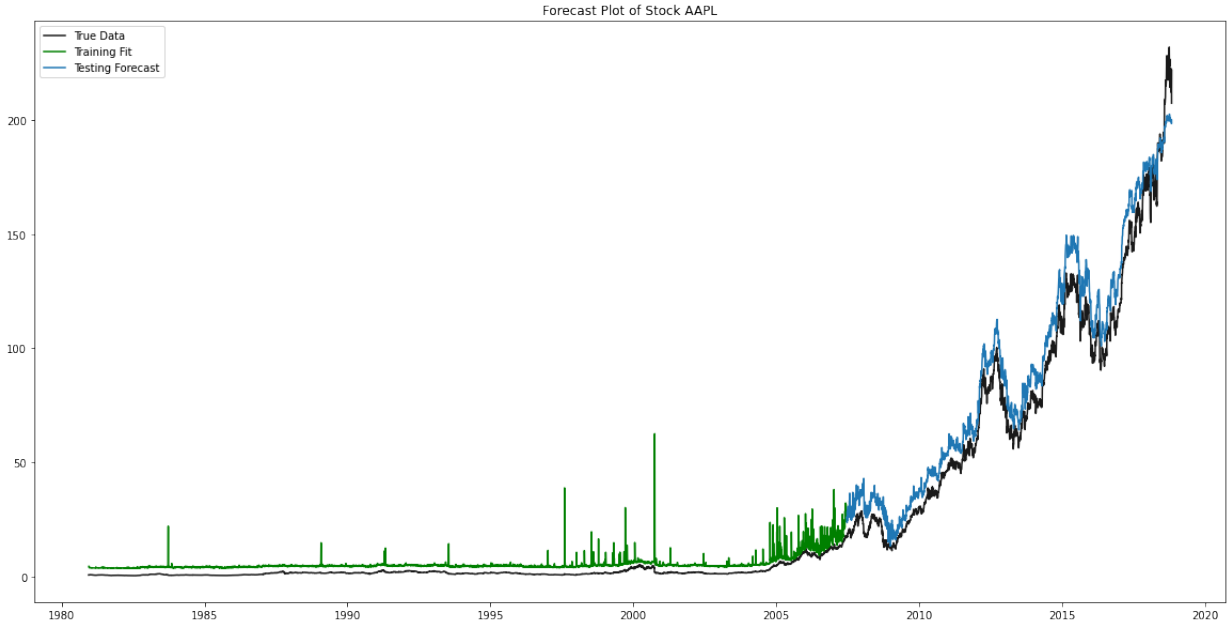


Figure 6: MLP Model Stock : APPL

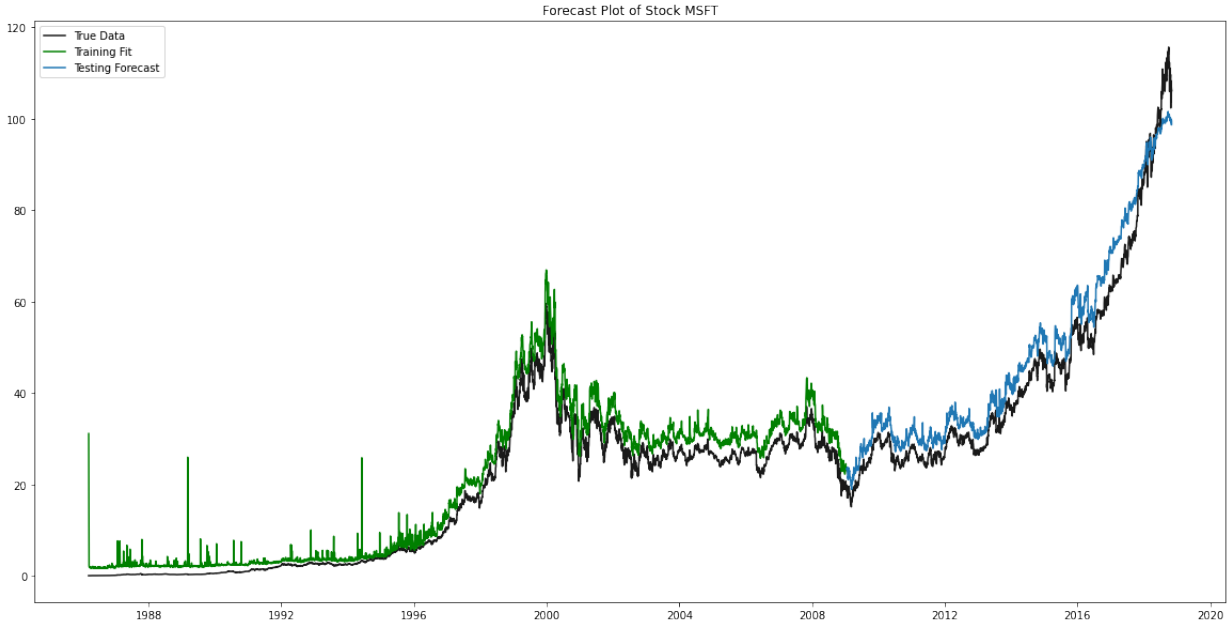


Figure 7: MLP Model Stock : MSFT

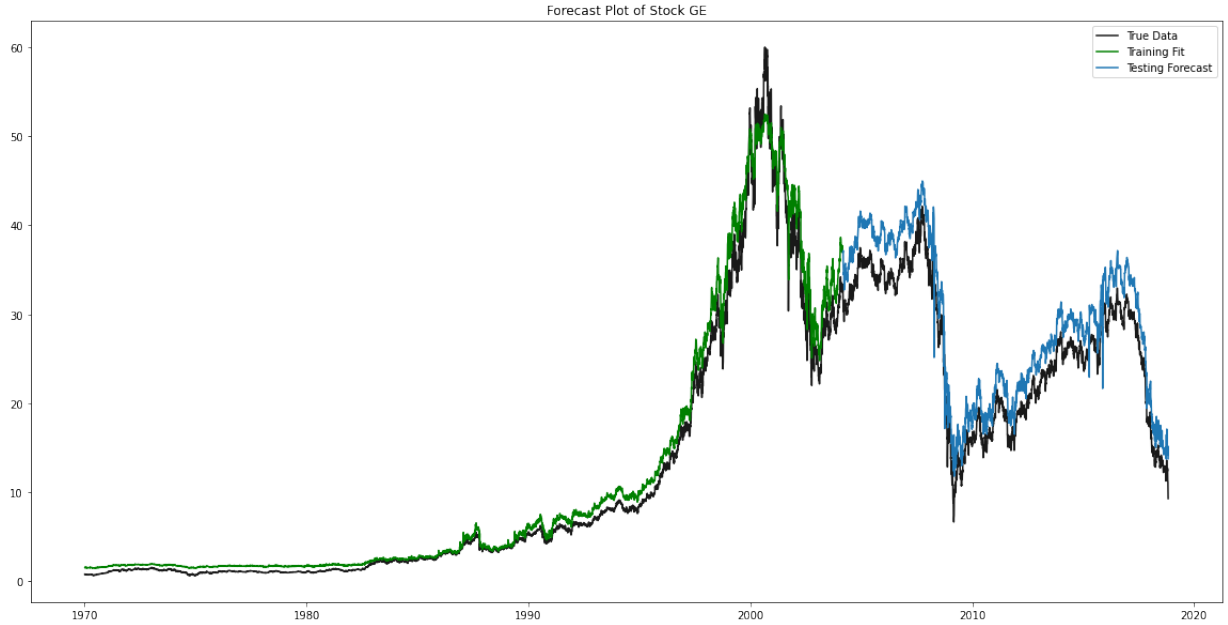


Figure 8: MLP Model Stock : GE

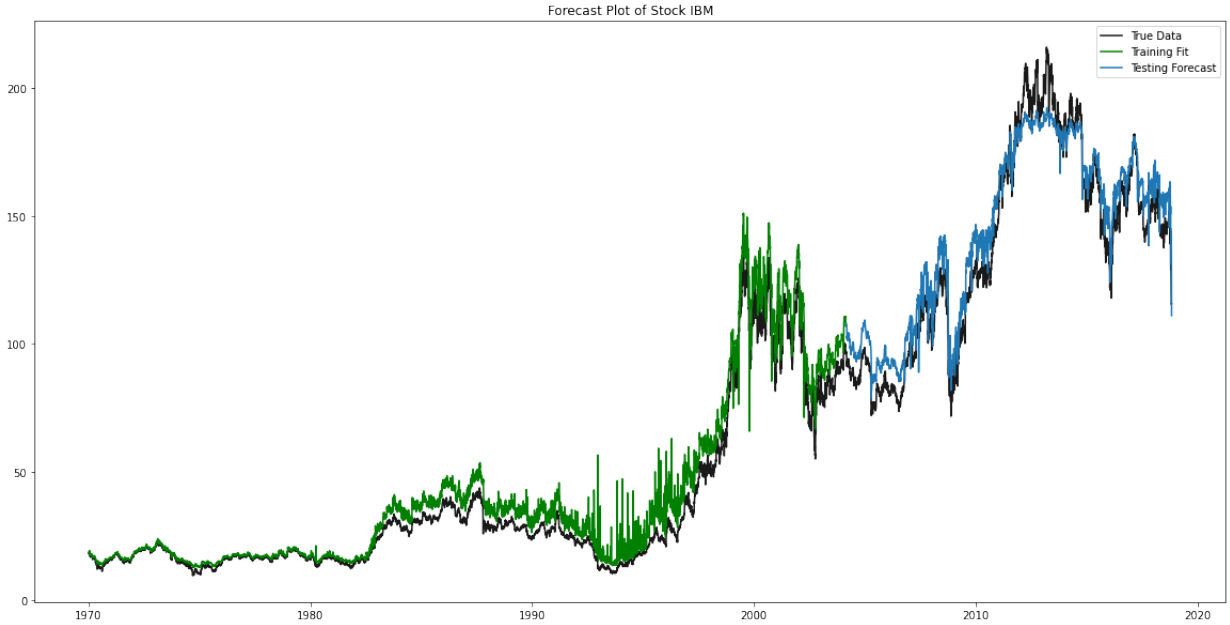


Figure 9: MLP Model Stock : IBM

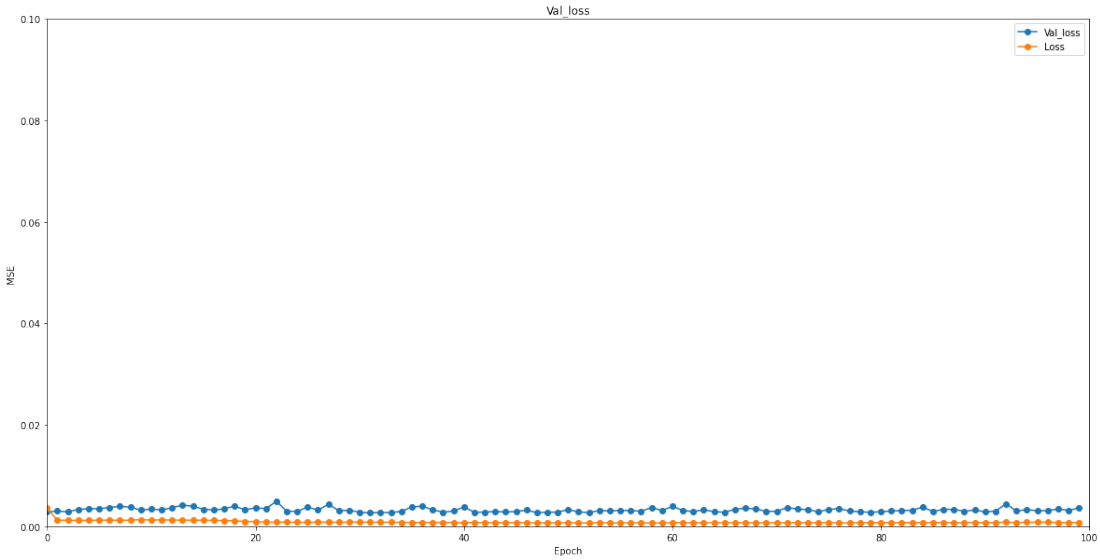


Figure 10: MLP Model Loss vs Epochs

Window Model Plots

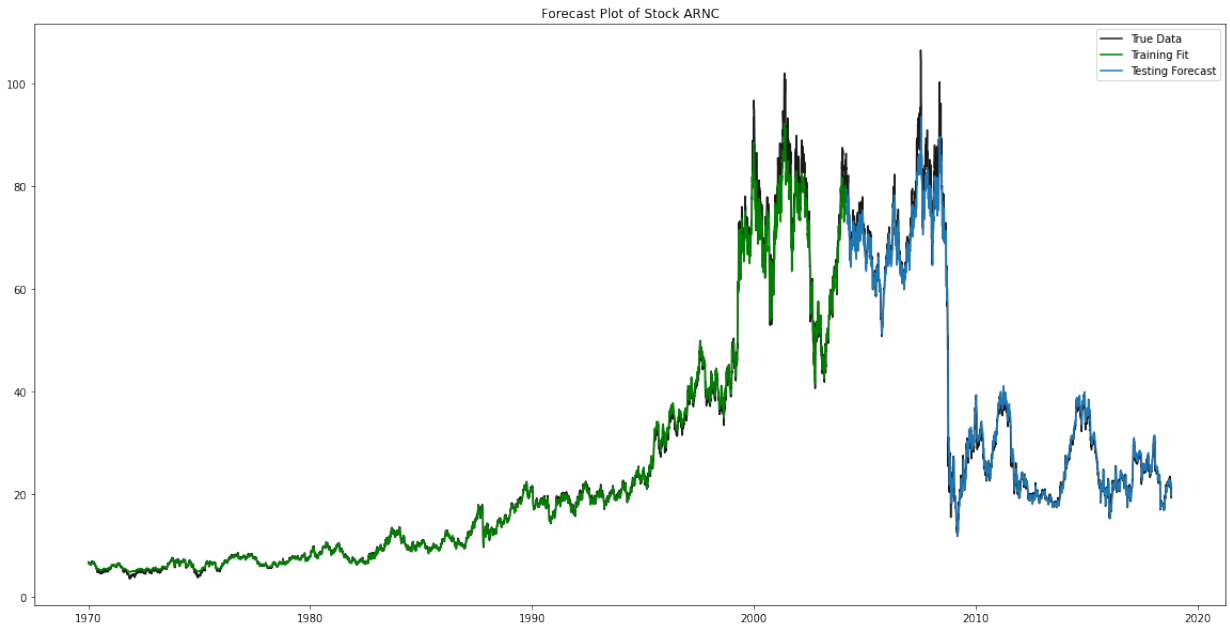


Figure 11: Window Model Stock : ARNC

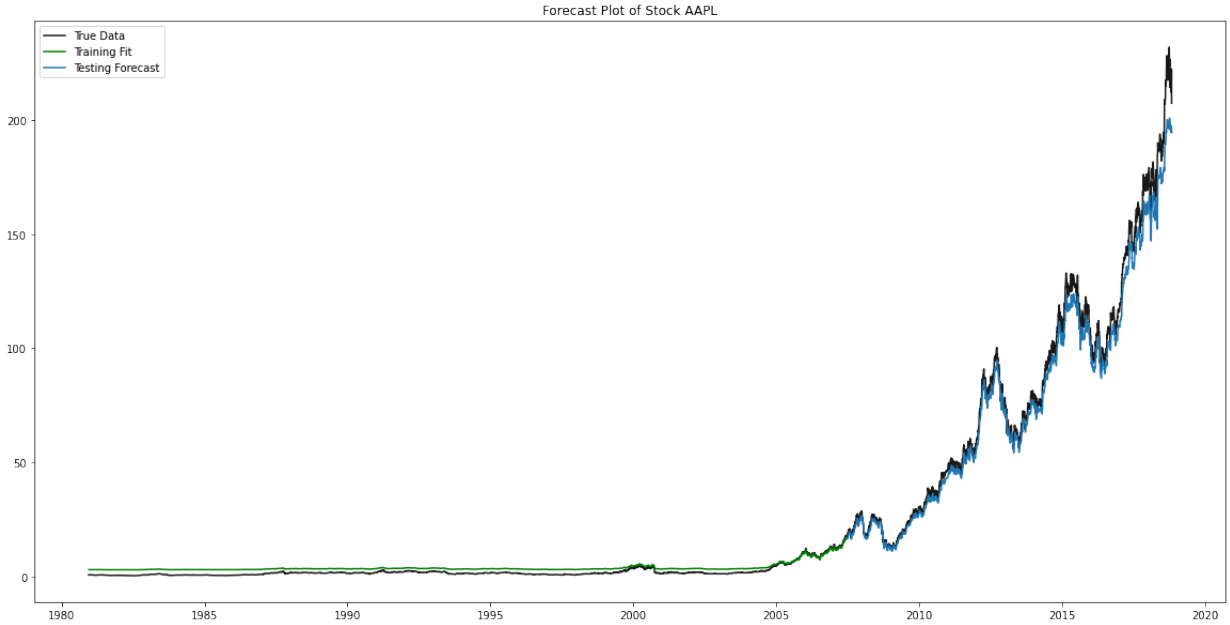


Figure 12: Window Model Stock : APPL

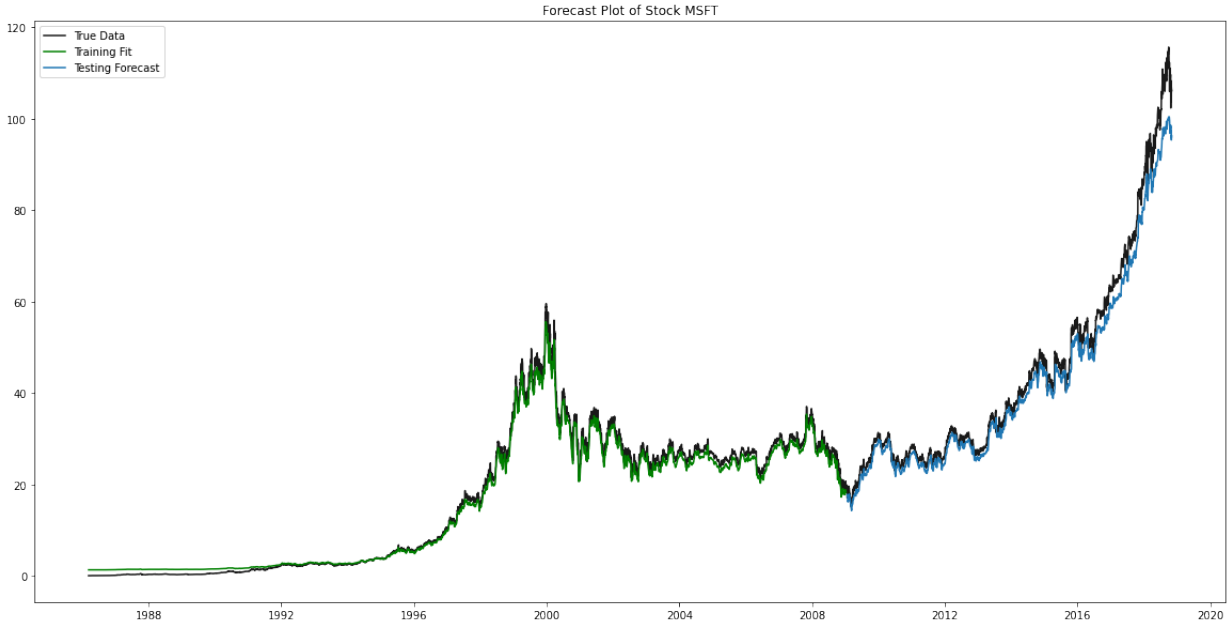


Figure 13: Window Model Stock : MSFT

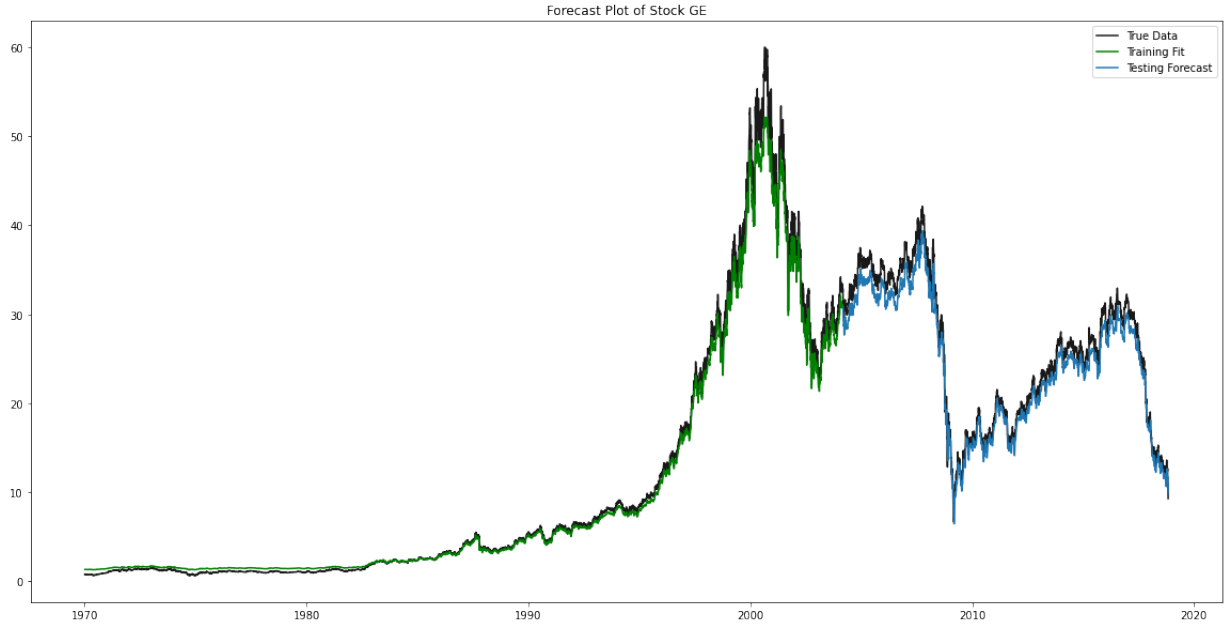


Figure 14: Window Model Stock : GE

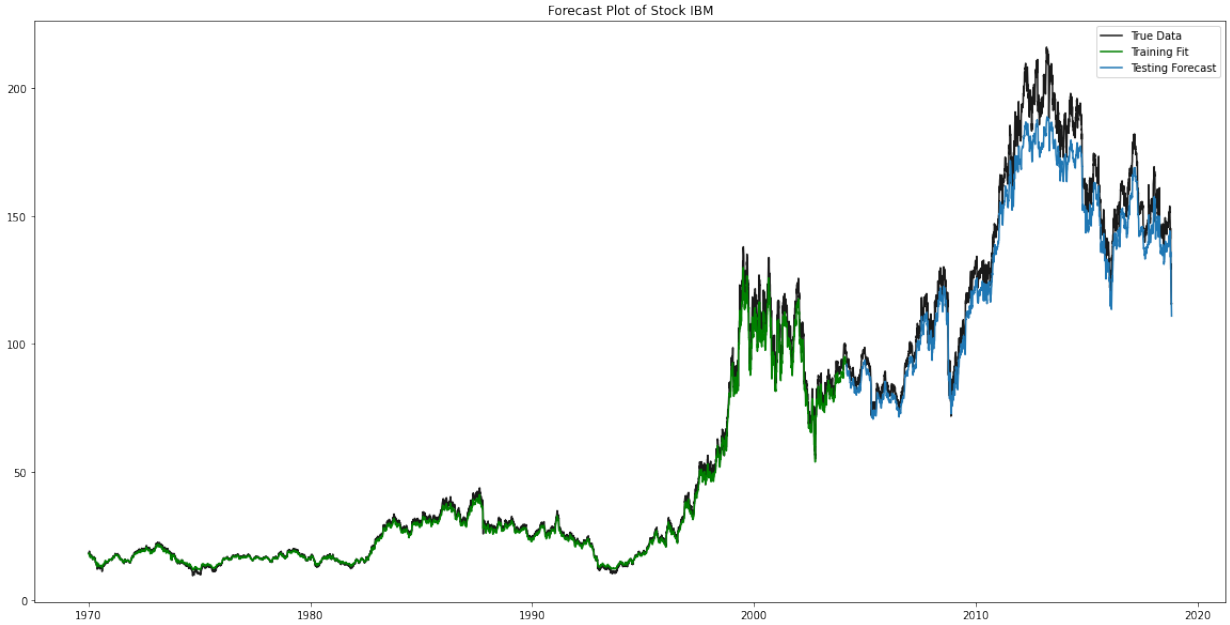


Figure 15: Window Model Stock : IBM

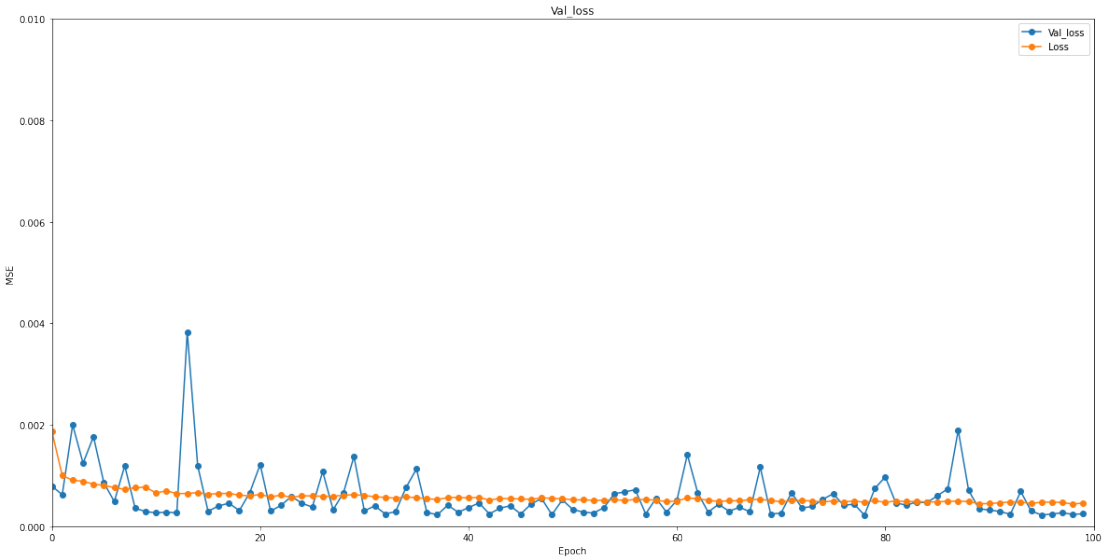


Figure 16: Window Model Loss vs Epochs

All-In Model Plots

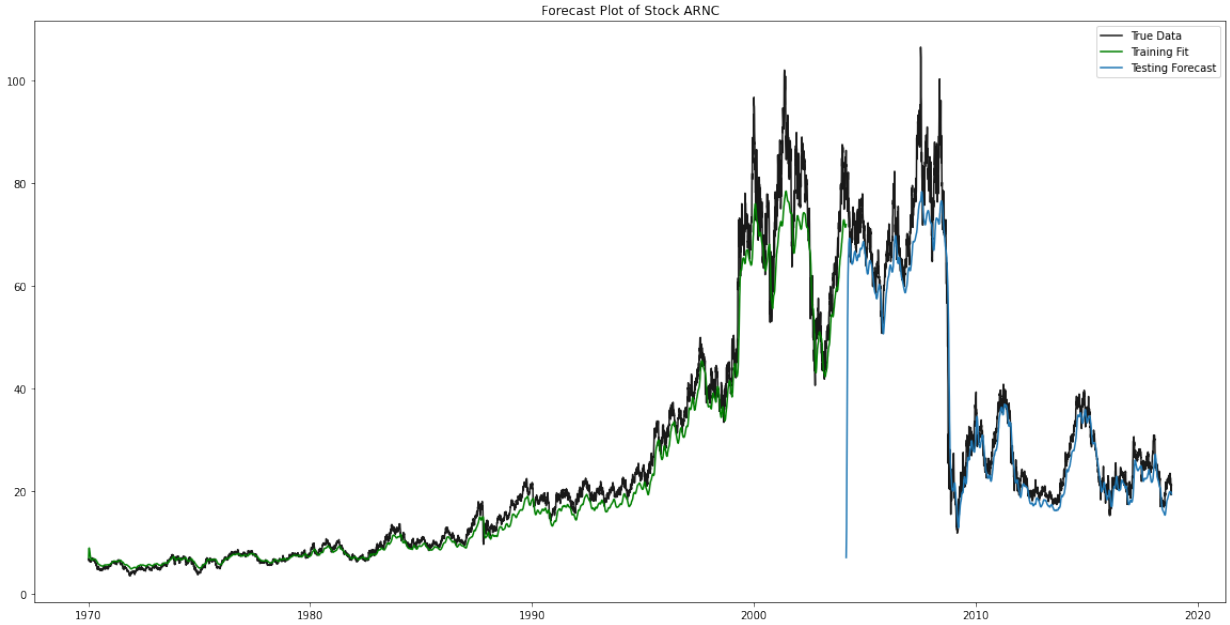


Figure 17: All-In Model Stock : ARNC

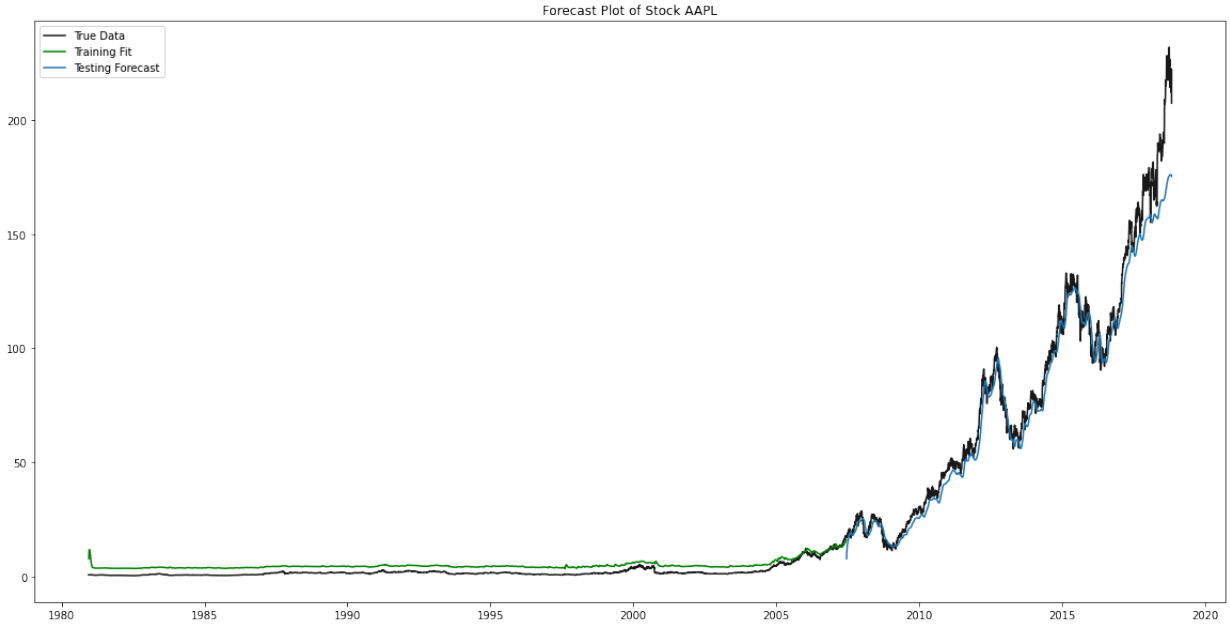


Figure 18: All-In Model Stock : APPL

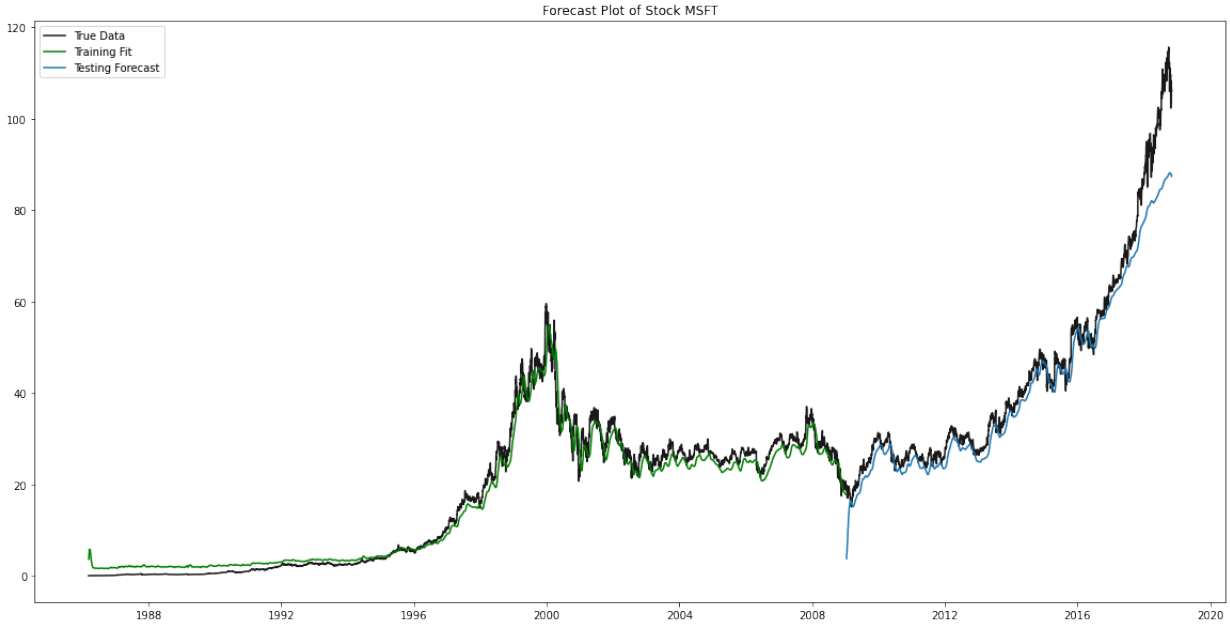


Figure 19: All-In Model Stock : MSFT

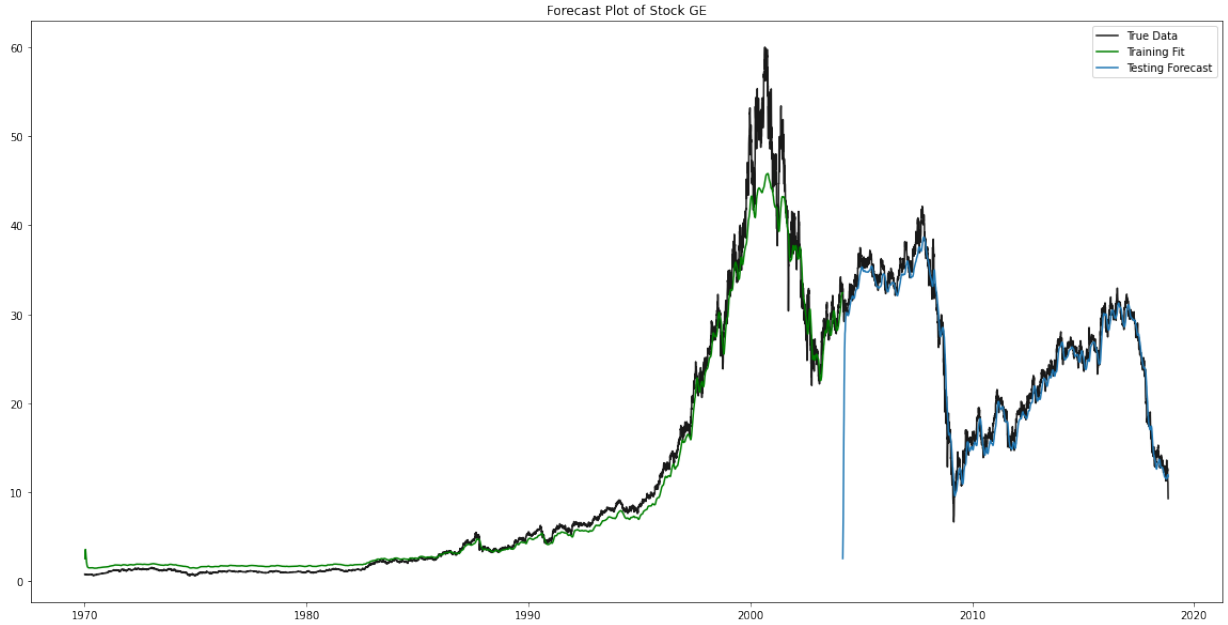


Figure 20: All-In Model Stock : GE

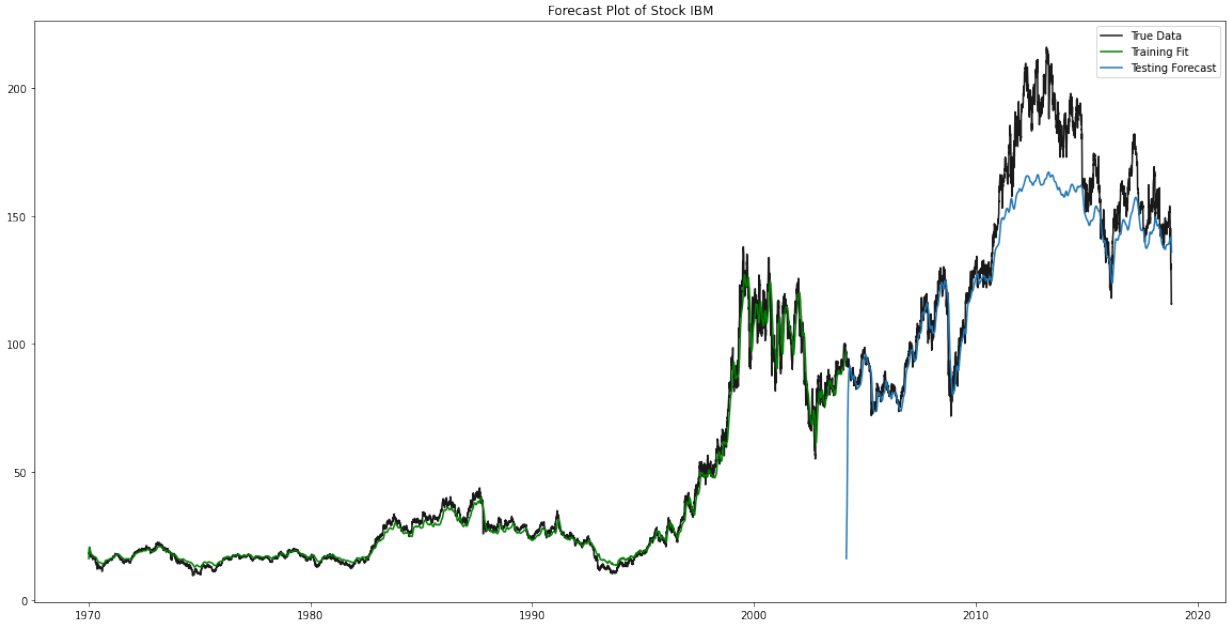


Figure 21: All-In Model Stock : MSFT

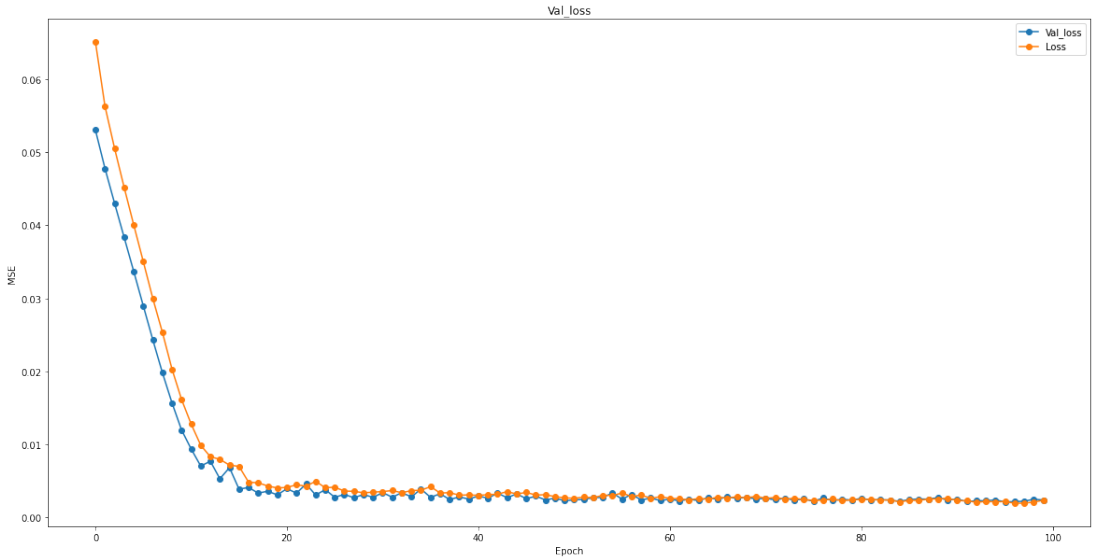


Figure 22: All-In Model Loss vs Epochs

Members and Roles

Data Analyst , Data scientist, Data engineer : Alexiou Dimitrios
Business Analyst, Data engineer, Machine learning Engineer: Fotopoulos Spyridon

References

[1] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[2] M. Moocarme, M. Abdolahnejad, and R. Bhagwat. *The Deep Learning with Keras Workshop: An Interactive Approach to Understanding Deep Learning with Keras, 2nd Edition*. Packt Publishing, 2020.

[3] Dipanjan Sarkar, Raghav Bali, and Tushar Sharma. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*. Apress, USA, 1st edition, 2017.

[4] Shervine Amidi Afshine Amidi. *Recurrent Neural Networks cheatsheet*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[5] Guangyu Ding and Liangxi Qin. Study on the prediction of stock price based on the associated network model of lstm. *International Journal of Machine Learning and Cybernetics*, 11 2019.

[6] Jason Brownlee. *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

[7] Jason Brownlee. *How to use Learning Curves to Diagnose Machine Learning Model Performance*. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

[8] J. Brownlee. *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Jason Brownlee, 2017.

[9] Achyut Ghosh, Soumik Bose, Giridhar Maji, Narayan Debnath, and Soumya Sen. Stock price prediction using lstm on indian share market. In Quan Yuan, Yan Shi, Les Miller, Gordon Lee, Gongzhu Hu, and Takaaki Goto, editors, *Proceedings of 32nd International Conference on Computer Applications in Industry and Engineering*, volume 63 of *EPiC Series in Computing*, pages 101–110. EasyChair, 2019.

[10] *Autonomio Talos [Computer software]*. http://github.com/autonomio/talos.

[11] Svetlana Borovkova and Ioannis Tsiamas. An ensemble of lstm neural networks for high-frequency stock market classification. *Journal of Forecasting*, 03 2019.

[12] *Keras Documentation*. https://keras.io/.

[13] David Benjamin Lim and Justin Lundgren. Algorithmic trading using lstm-models for intraday stock predictions.

[14] Junming Yang, Yaoqi Li, Xuanyu Chen, Jiahang Cao, and Kangkang Jiang. Deep learning for stock selection based on high frequency price-volume data. *arXiv preprint arXiv:1911.02502*, 2019.

[15] *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[16] Paula Pant. *Actively vs. Passively Managed Funds*. https://www.thebalance.com/actively-vs-passively-managed-funds-453773.

[17] Pam Krueger. *Active vs. Passive Investing: What's the Difference?* https://www.investopedia.com/news/active-vs-passive-investing/.

[18] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017.