## Implementation LSTM Neural Networks for Stock Market Prediction

Diptiranjan Harichandan

Department of Computer Science and Engineering

VSS University of Technology, Burla, India-768018

## ABSTRACT

Stock exchanges are considered major players in financial sectors of many countries. Most Stockbrokers, who execute stock trade, use technical, fundamental or time series analysis in trying to predict stock prices, so as to advise clients. However, these strategies do not usually guarantee good returns because they guide on trends and not the most likely price. It is therefore necessary to explore improved methods of prediction.

Here, we use a special type of Neural Network, Long Short Term Memory (LSTM) to predict testing data given a sequence of time-series input. And from the results it is clear that the model is thus capable of prediction on typical stock markets.

**Key words:**

RNN, LSTM, Beautiful Soup, Standard & Poor's 500, ADBE

**INTRODUCTION**

Machine Learning has been a significant but it is nothing but a form of computing which is basically automation of **feature engineering.**

There are many applications where this form of computing is superior to human intelligence. Properly weighing and analysing all aspects is simply done better with less bias, and far quicker, by computers.

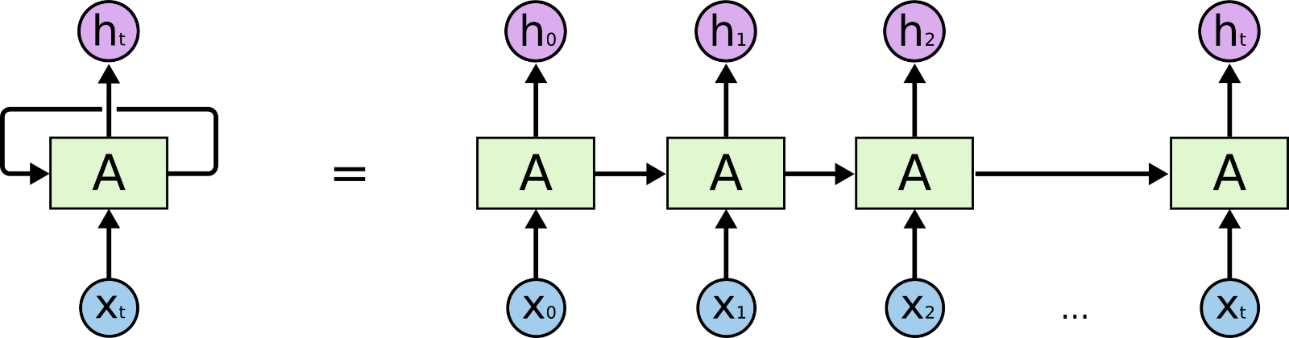
**OVERVIEW**

**Recurrent Neural Networks**

RNNs make use of sequential information. Information where we have to predict the next word in a sentence, for this we need to know what came before it.

Basically, RNN take in previous hidden layer output along with present input to make prediction. It

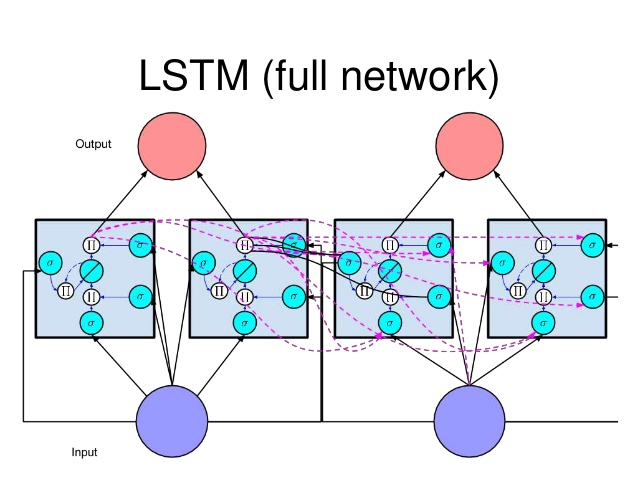
can be thought of as a memory unit.



**LSTM (Long Short Term Memory)**

It is a type of Recurrent Neural Network (RNN). LSTM’s whole architecture is same as RNN except the node part. Here Memory Cells are used in place of nodes. These nodes have input gate, output gate and an internal state which together cleverly allows LSTM networks to -

1. Selectively choose what it remembers
2. Decide to forget
3. Select how much of its memory it should output.



## IMPLEMENTATION

## Automating getting the S&P 500 list

First of all we're going to be working on how we can go about grabbing pricing information en masse for a larger list of companies, and then how we can work with all of this data at once.

To begin, we need a list of companies. We can just get a list from any website manually, but actually acquiring a list of stocks can be just one of the many challenges one might encounter. In our case, we want a Python list of the **S&P 500 companies**.

Whether we are looking for the Dow Jones companies, the S&P 500, BSE or NSE, we always want to make sure it is up-to-date, but chances are it's not already in the perfect format for you. In our case, we're going to grab the list from Wikipedia: <http://en.wikipedia.org/wiki/List_of_S%26P_500_companies>.

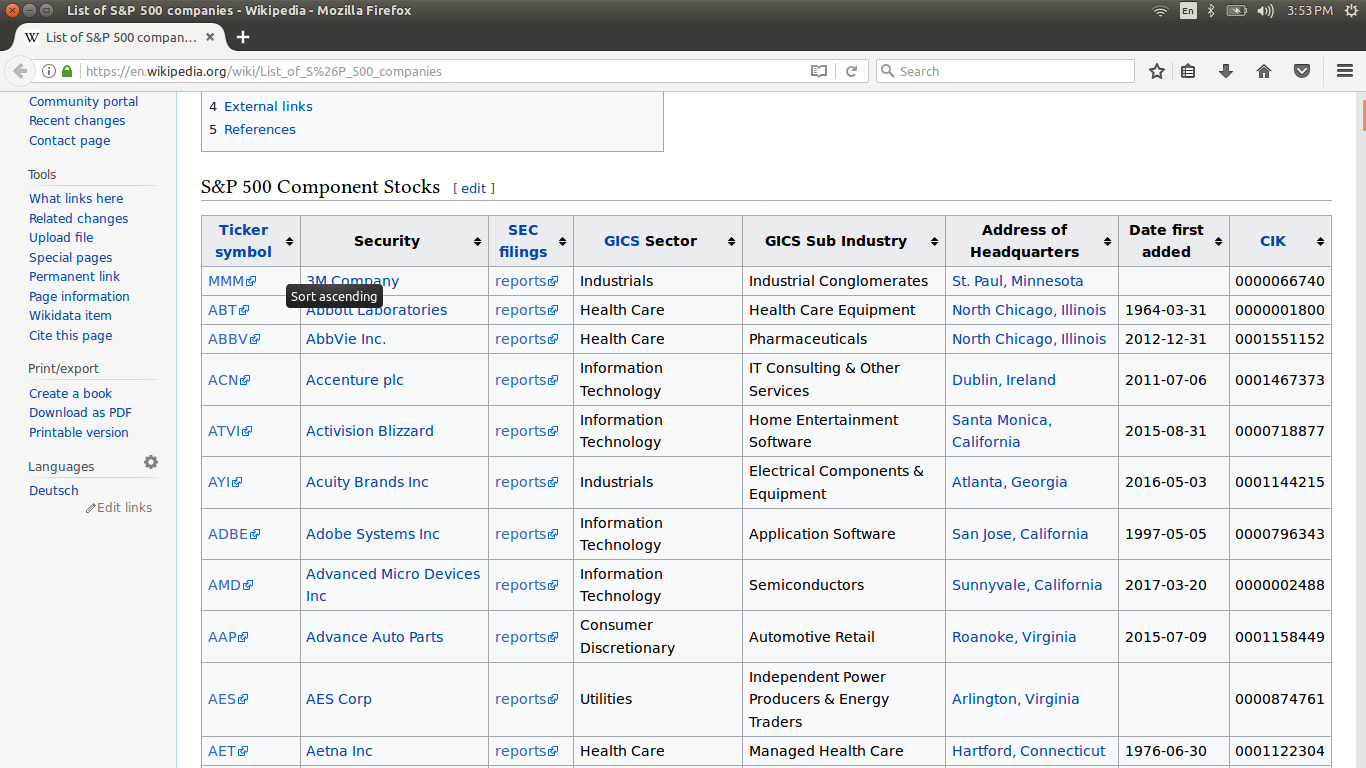
The tickers (or the symbols of companies) in Wikipedia are organized on a table. To handle for this, we're going to use the HTML parsing library, Beautiful Soup.

We need **pickle** so we can easily just save this list of companies, rather than hitting Wikipedia every time we run. But occasionally we have to update the list.

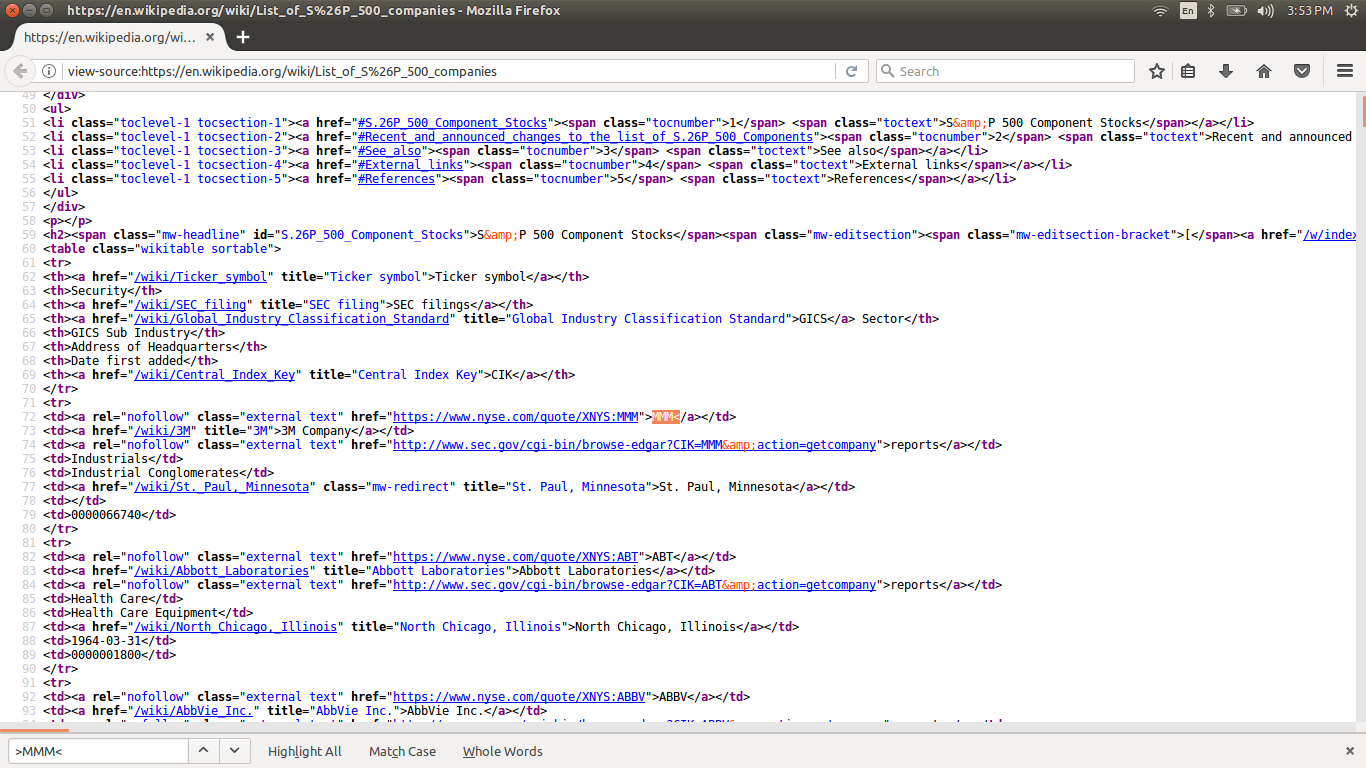
The **datetime** module supplies classes for manipulating dates and times in both simple and complex ways.

**Scrapping list from Wikipedia**

First, we visit the Wikipedia page, which contains the list of all S&P500 companies and their tickers, then we turn its source code to BeautifulSoup object.



Once we have our soup, we can find the table of stock data by simply searching for the wikitable sortable classes. This class contains the Tickers of the companies.



## Getting all company pricing data in the S&P 500

After getting all the S&P500 ticker list now it is time to pull stock pricing data on all of them.

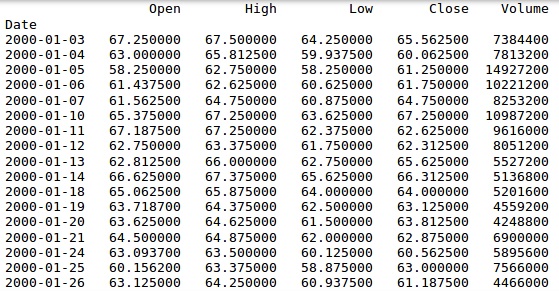
Many operations that will be conducted on the data might require different set of parameters as inputs. This requires more than once downloading of tickers and searching for particular parameters.

Thus, we pull everything we can from what Yahoo returns to us for every stock and save it locally. For this, we create a new directory and save each company’s stock pricings in individual .csv file.

**DATASET**

Here we have used stock data of ADBE (Adobe Systems). But due to locally storing the individual stock prices of all the S&P500 companies we can directly predict values of any company by just feeding in the desired company’s ticker.

Here is what single dataframe of stock looks like –



Where,

Open = Opening price of the Day

High = Highest price of the Day

Low = Lowest price of the Day

Close = Closing price of the Day

Volume = Volume of stocks sold

Adjusted Close = Closing price which is obtained by removing Dividends and including out of market investment and trade splits.



*Graph – Closing prices vs time and Volume vs time*

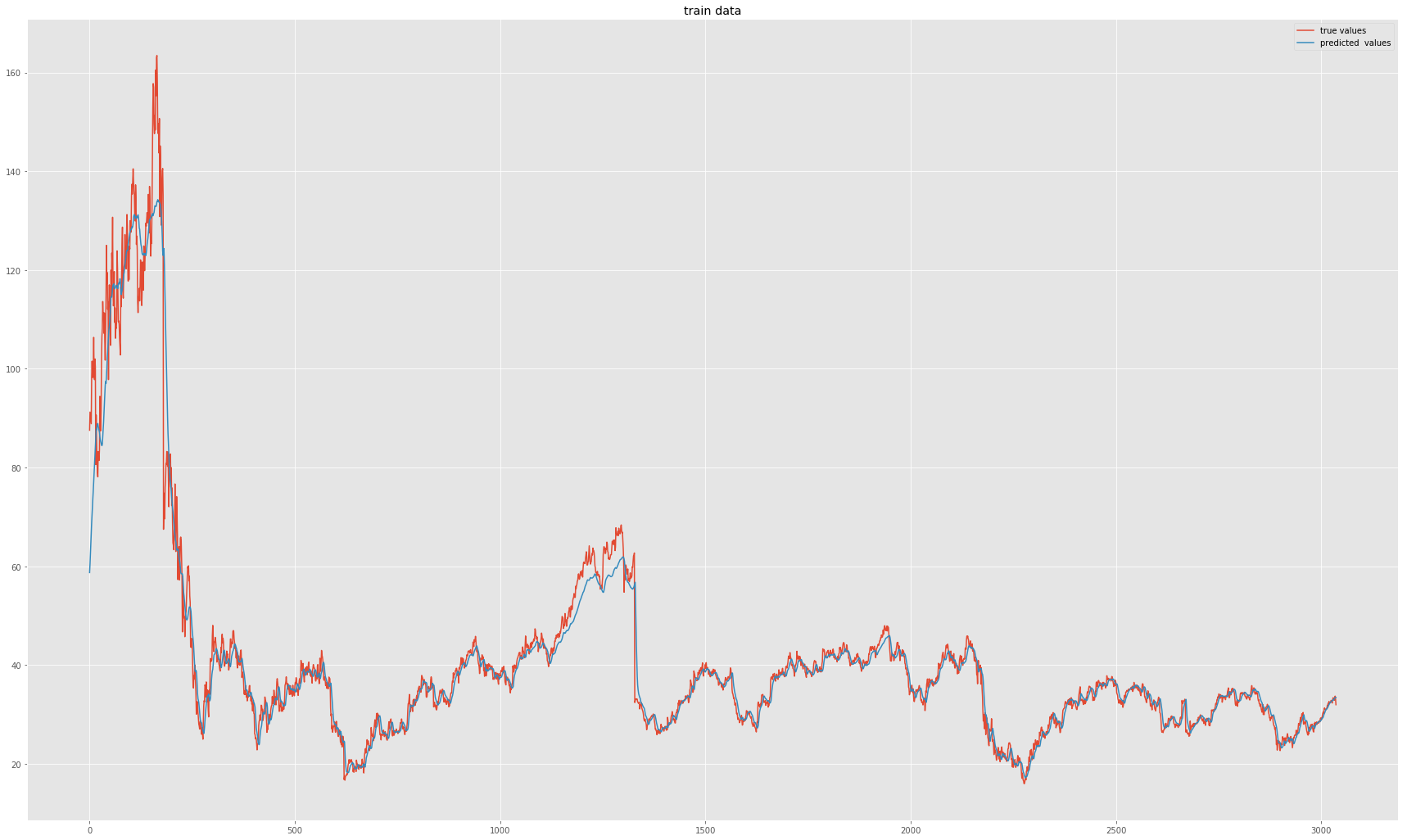


*Graph – Candlestick graph*

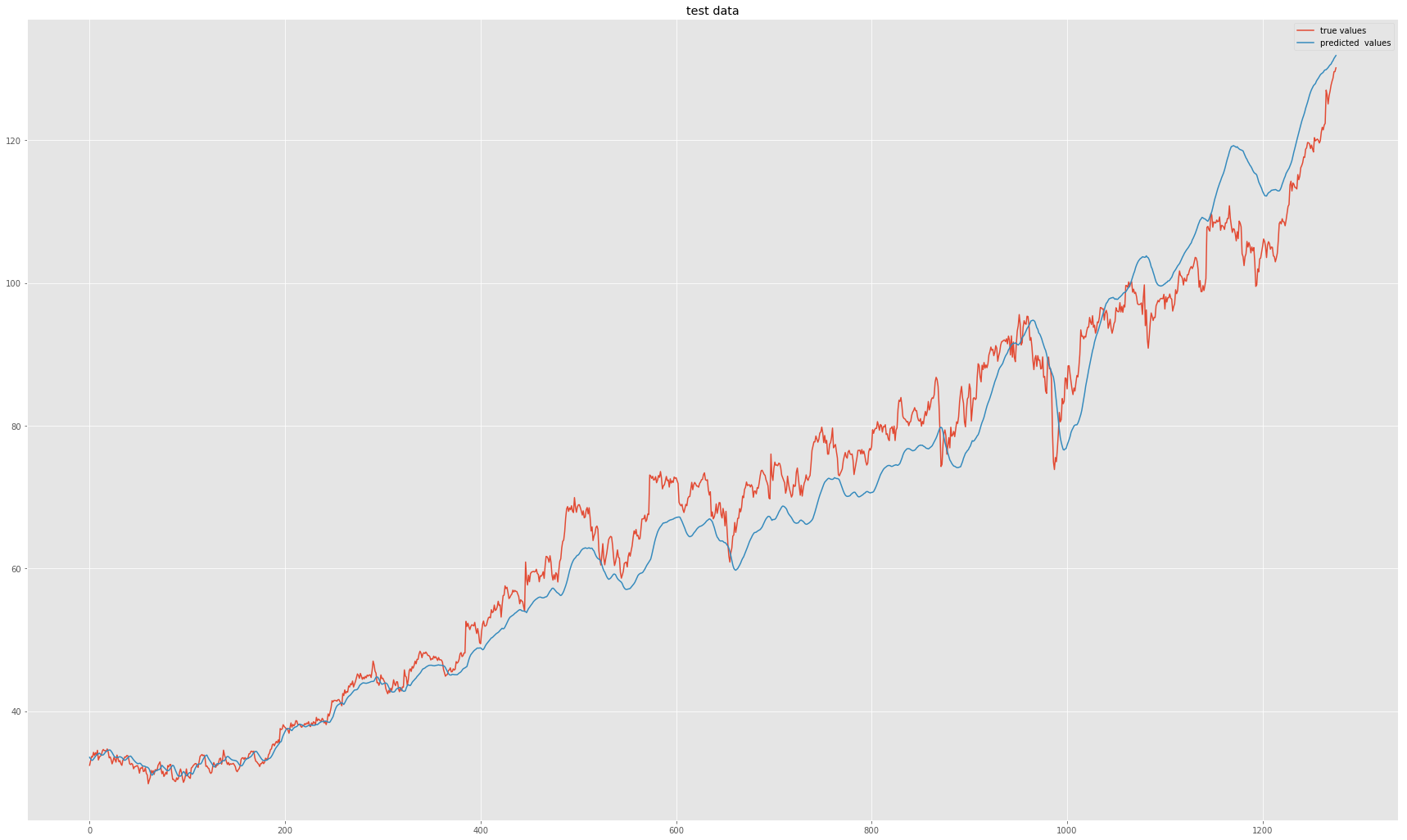
**RESULTS**

From the below graphs it is clear that our model has been able to predict Testing dataset efficiently. The testing score was **4.987,** while the training score was **4.389**.

All scores are Root mean square error (RMSE).



*Graph – training*



*Graph – Testing*

**REFERENCES –**

* Learning Long term Dependencies with Gradient Descent is Difficult <http://www.dsi.unifi.it/~paolo/ps/tnn-94-gradient.pdf>
* Investopedia.com