Stock Market Prediction using LSTM

Masters in Computer Science
Major Research Project

Submitted
By
Surajit Das (Roll No:- 1723939)
Shreshth Saxena (Roll No:-1723936)

Submitted
To
Dr. Naveen Kumar
And
Mrs. Sheetal Taneja

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF DELHI
NEW DELHI- 110 007
May-2019

Certificate

This is to certify that the project report titled **'Stock Market Prediction using LSTM',** submitted by Surajit Das and Shreshth Saxena to the Department of Computer Science, Delhi University, is a bonafide record of the work done by them under the supervision of Dr. Naveen Kumar and Mrs. Sheetal Taneja, for the major project of M.Sc Computer Science(2017-2019).

Dr. Naveen Kumar
Dept. of Computer Science
University of Delhi

Mrs. Sheetal Taneja
Dept. of Computer Science
Dyal Singh College
University of Delhi

Acknowledgement

We would like to express our gratitude to all those who have guided us in the completion of the project.

We are very grateful to Dr. Naveen Kumar, for being a great guide and teacher. He provided immense support, constant encouragement and exemplary guidance at every stage of the project. He has always emphasized in thoroughness and clarity in our approach and we are thankful to him, for helping to put the project in perspective.

We are also very grateful to Mrs. Sheetal Taneja, for her immense support, exemplary guidance, encouragement and constant supervision, as well as for providing necessary information regarding the project. She has invested her full effort in guiding the team and achieving the goal.

I would also like to express my gratitude towards my friends, co-mates and seniors without whom the project would have remained incomplete.

Abstract

This project was carried out to determine a successful strategy for predicting stock prices based on past financial time series data. We started out with gathering domain-specific knowledge about financial data and stock markets, and building initial models for stock prediction and Algorithmic Trading.

The first model discussed here uses the "New York Stock Exchange" playground-dataset on Kaggle and an LSTM architecture with 2 layers containing 256 units each, followed by a fully connected layer (32 neurons) and an output layer (1 neuron).

Simultaneously, we developed an Algorithmic Trading strategy to which we later plugged in our LSTM model predictions and analysed the results. The Algorithmic Trading started with a "hello world" moving average crossover and moved ahead to more sophisticated strategies like Exponential Moving Average. The strategy was eventually back tested using data handlers, execution handler and portfolios implemented in python.

To further our understanding of LSTM models, we gathered and analysed a number of research papers with coinciding topics, and implemented two of them.

"Deep learning with long short-term memory networks for financial market predictions" Thomas Fischer Christopher Krauss

This paper deploys LSTM for predicting S& P 500 stocks and compares the performance with memory-free classification methods such as a random forest, a deep neural net and a logistic regression classifier. The data used here dates between 1992 and 2009.

"ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module"

Baek, Yujin; Kim, Ha Young

This paper talks about an unconventional approach for predicting S& P index through custom prediction models named SingleNet, ModAugNet-f and ModAugNet-c. SingleNet is the default LSMT prediction model taking 20-days window as input and predicting the index on 21st day. ModAugNets take this approach a little further and incorporates an overfitting-prevention module in addition to the prediction module. The prevention module aims to reduce overfitting and applies data augmentation (in ModAugNet-c) to increase the amount of training data. The ModAugNet has two variants, ModAugNet-c and ModAugNet-f discussed later in the report. The ModAugNet-c architecture gives better results than the ModAugNet-f and SingleNet.

Contents

1 OverView

1.1 Problem Statement

To implement and analyze various Machine Learning techniques used in the prediction of financial data such as stock prices or stock market index.
To understand and apply Algorithmic Trading strategies and generate good enough returns.

1.2 The Fuss about Predictability

The predictability of stock market is a highly controversial topic stimulating years of debate on the actual feasibility of this term. From the works of *Bachelier (1900)*, *Cootner (1964) and Fama(1965)* the Stock Market is affirmed to be independent of past and present information, and hence unpredictable, holding true to the **efficient-market hypothesis**. While a good number of researchers beg to differ and possess myriad tools and technologies which purportedly allow them to gain future stock price information.

1.3 'The Efficient Market Hypothesis"

The efficient market hypothesis postulates that stock prices are a function of information and rational expectations, and that newly revealed information about a company's prospects is almost immediately reflected in the current stock price.

Burton Malkiel, in his seminal 1973 work A Random Walk Down Wall Street stated "a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts." or in simpler terms stock prices prediction by looking at price history is not feasible. This engendered a number of articles and a group of "Research Affiliates" went ahead to simulate results of 100 monkeys throwing darts at the stock pages in a newspaper. They found the average monkey to outperform the index by an average of 1.7 percent per year since 1964.

Malkiel claimed that variations in stock prices correspond to a statistical process called a "random walk" meaning each day's deviations from the central value are completely random and unpredictable. He concluded that paying financial services persons to predict the market actually hurt, rather than helped, net portfolio return. A number of empirical tests also support the application of this theory, as most portfolios managed by professional stock predictors do not outperform the market average return after accounting for the managers' fees.

On the other hand a number of studies have attempted to throw away the efficient market hypothesis and demonstrated the predictability of stock markets to some extent, particularly for short windows. Also the fact that a probability of 52 percent is a considerable

increase from 50 percent in the stock trading world au contraire to a mathematical perspective, provides motivation to leverage these probabilities to generate better trading techniques (algorithms). To rest this debate and quench or curiosities on the subject let's conclude with the fact that the Wizard of investment, Warren Buffett himself rebutted the Efficient Market Hypothesis in 1984 during his speech at Columbia University.

1.4 Past Works

Researchers have applied parametric statistical methods such as vector autoregression, autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) that resulted to be efficient in describing and evaluating the relationships between variables by statistical inference but had their limitations while predicting financial time series. This owes to the model's presumption of a linear nature with a constant variance for stock prices which certainly is not the case in reality.

The issue of nonlinearity in financial data was acknowledged by making use of machine learning technologies such as ANNs (Artificial Neural Networks) and kernelized support vector machines. These models were better at generalization and nonlinear mapping, provided enough data. They exploit the **Universal Approximation Theorem** which states that any finite and continuous function could be approximated.

Hybrids of the two techniques were also proposed later such as using ARIMA model for extracting input features and RNNs (Recurrent Neural Network) for training md-term stock market trend prediction system. *Reference : Wang and Leu(1996).*

The computational results of such fusion models outperformed the single econometric models and other individual forecasts. This encouraged further hybrid models, combining different machine learning models such as ANN, random forest and support vector regression (SVR) by *Patel, Shah, Thakkar and Kotecha (2015).*

1.5 Recent Trends

Recently LSTM (Long Short-Term Memory) networks have been breaking the trends in time series predictions. They are structured to identify and learn temporal patterns. LSTMs are an improvement of conventional RNNs as they are better at learning long-term dependencies and also overcome the problem of vanishing/exploding gradients. LSTMs also mitigate the need of hand-crafted features such as economic or technical indicators.They automatically detect suitable patterns in the given data. LSTMs have evidently yielded better results than many baseline models like multi-layer perceptrons, random forests etc. LSTM-based odels have also been found to provide lower risks in trading strategies.

In addition, other sequential data can also be fed into LSTM networks to extend the available information set such as investor sentiment data as done by *Li, Bu and Wu (2017).*

Further improving the equation was the advent of attention mechanisms after the influential google research paper "Attention is All You Need" in 2017. Attention mechanisms

move forward in the goal of reducing sequential computations and provided more efficiency with better results.

1.6 Results

The different models applied and tested for stock prediction in this project have been evaluated on the metric of minimum squared error or mse of the model on test data. The predictions and actual data are compared by plotting against the time axis and hyper parameter tuning is done to achieve better performance.

2 Techniques

2.1 RNN (recurrent neural network)

Main motive to make an RNN model is to make use of sequential model.

2.1.1 How it is different than other neural network models

It is assumed for any traditional neural network that input is not dependent on output or vice versa. But it is not true for all the cases. Suppose we want to predict a day's return of a stock market, then it is necessary to know previous days' return index. Or suppose we want to predict the next word of a sentence, then we must know the word(s) that came before that.

What 'recurrent' means in RNN is repeating the same task for all the element in the sequence. Here output is determined by the previous calculations. In simpler words, it can be said that RNN has 'memory cell', which store the information about the previous elements.

Figure 1 describes RNN in a simple way

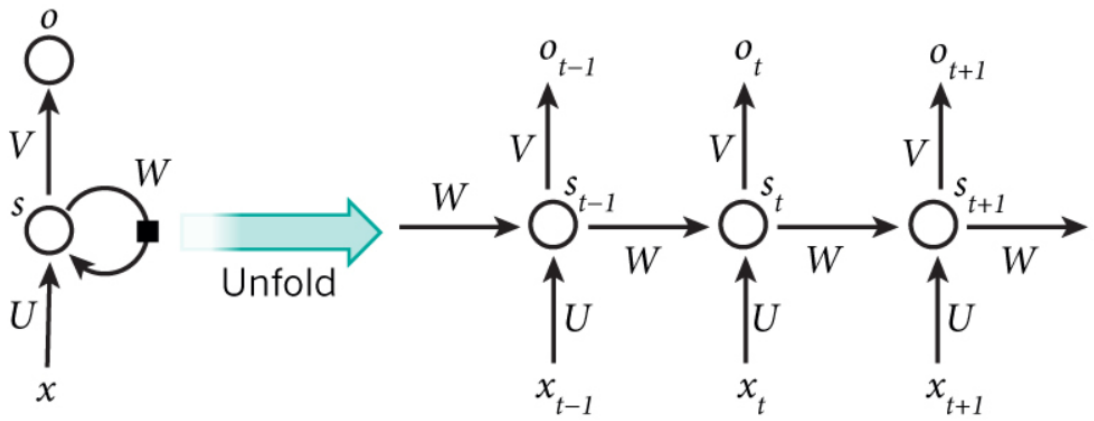


Figure 1: Simple Recurrent Neural Network

Here unfold means unrolling i.e. expanding into the full network. For example, if an RNN is to be used in a 10-word sentence, the RNN could be unrolled into 10 layered neural networks, each layer represents one layer.

RNN works best in field of Natural Language Processing and time series data.

2.1.2 Some applications of RNN are the following

2.1.2.1 Language modelling and prediction

in this method, the probability of a word occurring in a sentence in a particular position with respect to other words is considered. In language modelling, a sequence of words is the model and the output will be a sequence of predicted words by the model. The model basically computes how likely a sentence can be, which is an important step for machine translation module. Predicting the next word is the motive of generative model, which tries to generate new text by sampling from the output probabilities of the given input text.

One of the methods for language modelling is using one hot encoding.

2.1.2.2 Machine Translation

In this one language is the input and output will be any other target language. The difference between machine translation and language modelling is that the output starts after the model sees the input completely as shown in Figure 2.

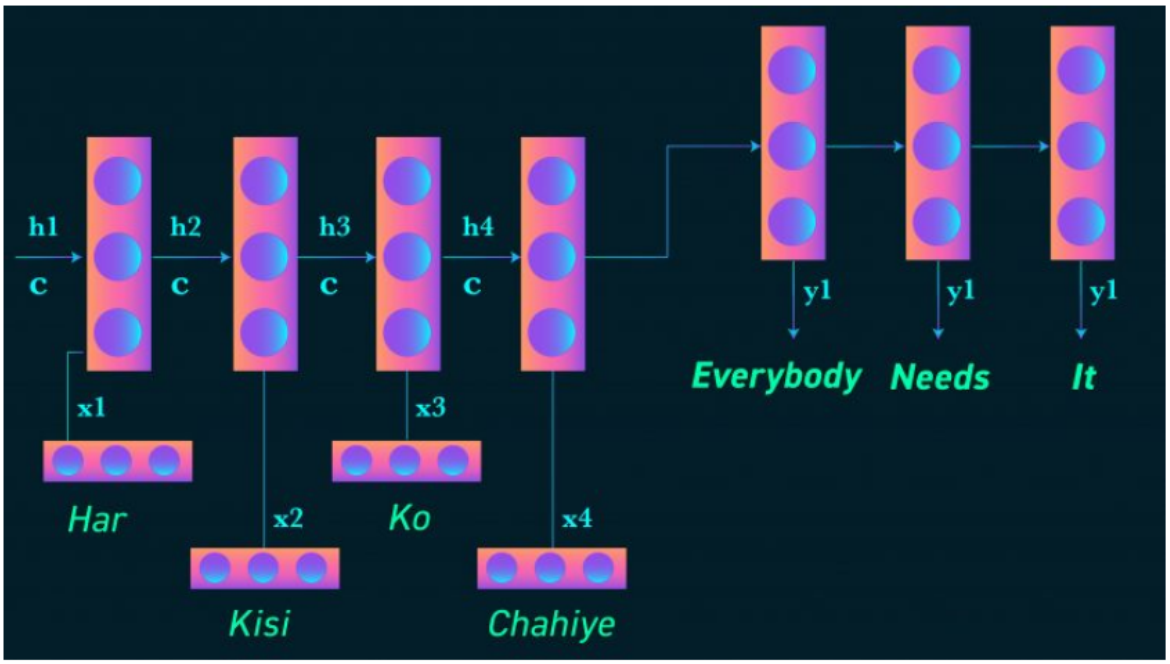


Figure 2: Language Translation from 'Hindi' to 'English' by RNN

2.1.2.3 Image recognition and generating description

Convolution neural network (CNN) can be used with RNN to generate image description. Results can be very impressive after constructing the model properly.

2.1.2.4 Speech Recognition

Given phoneme (sequence of acoustic signal from sound wave) as input, it predicts the sequence of phonetic segments.

2.1.3 Training RNN

Like the other neural network RNN needs to be trained. Here backpropagation algorithm is also used with a slight modification as not only the current time step is required to compute the output but also all the previous time steps are also important.

2.1.4 RNN extension

Researchers have been developing so many things in RNN. Some of the developments are as follows.

2.1.4.1 Bidirectional RNN

Let's assume a case when a missing word of a sentence is to be predicted. In this case left and right side of the missing place in the sentence should be considered. It is not possible in simple RNN because there the inputs are fed in a unidirectional way.

So basically, bidirectional RNN requires elements of previous time steps as well as elements of future time steps. Bidirectional RNN comprises two RNNs stacked on the top of each other.

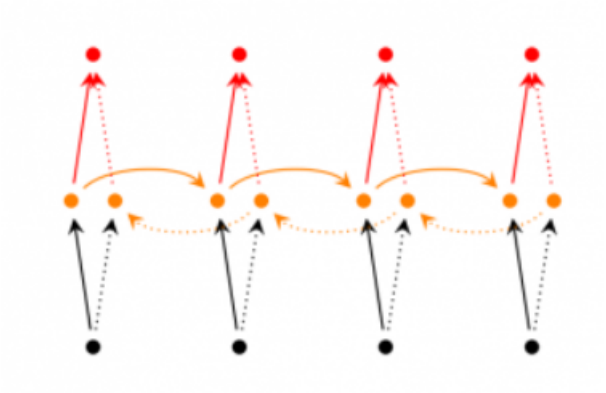


Figure 3: Working of a Bidirectional RNN

2.1.4.2 Deep (Bidirectional) RNN

It is similar to the bidirectional RNN with multiple layers are stacker per time step. This requires lots of training data but also gives higher learning rates.

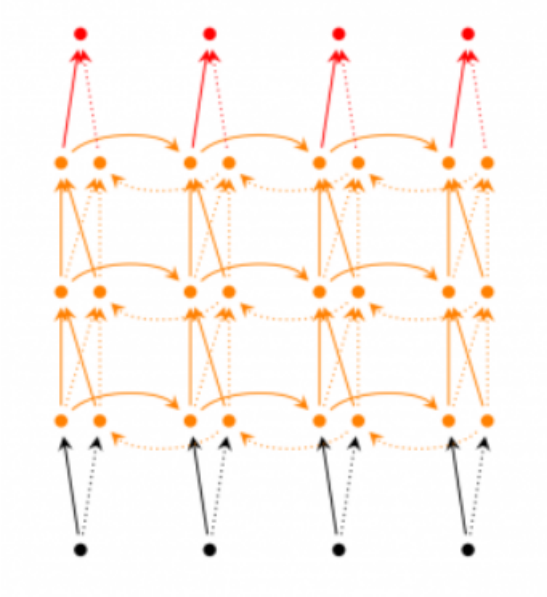


Figure 4: Working of a Deep (Bidirectional) RNN

2.1.5 Different types of RNN

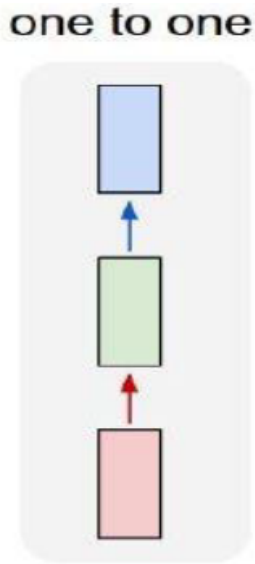


Figure 5: one to one RNN

2.1.5.1 One to one(vanilla)

it is the basic model of RNN

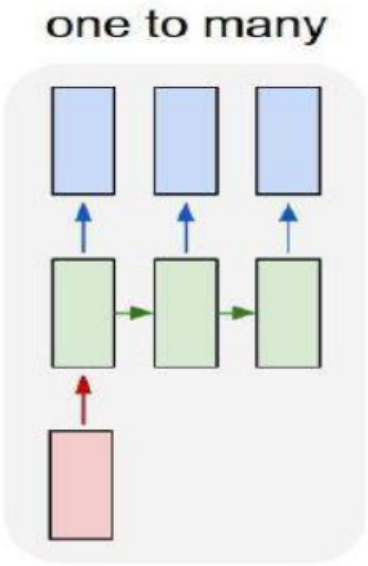


Figure 6: one to many RNN

2.1.5.2 One to many

Here number of inputs is one and number of output is more than one.

Example: **image captioning**

Here an image is given and task is to predict some sequence of words as the description of the model.

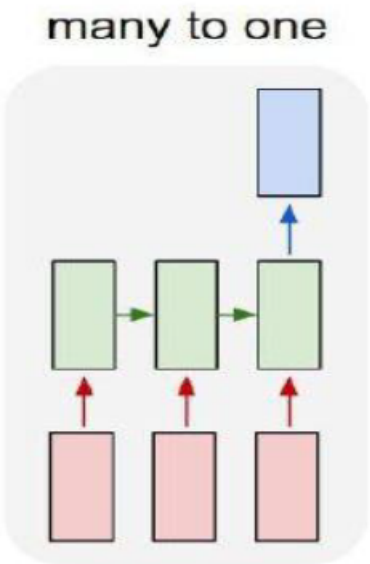


Figure 7: many to one RNN

2.1.5.3 Many to one

Here more than one input layer is used and only output is available.

Example: **Sentiment classification**

Suppose a paragraph i.e. a bulk of words is given, and the sentiment of the para has to be determined. In this type of case many to one model is used.

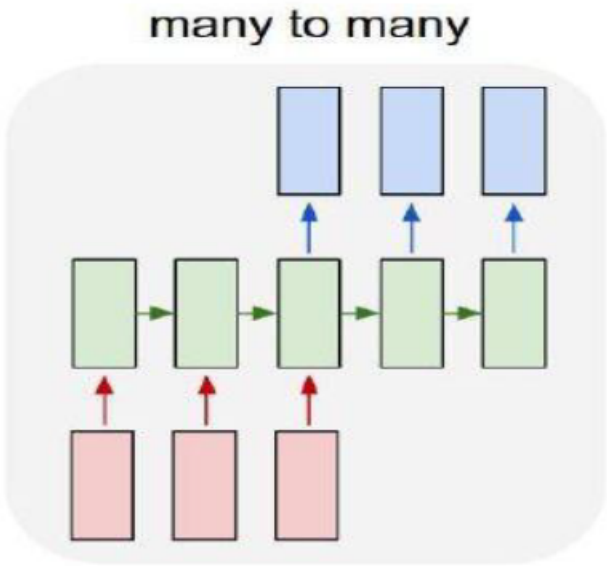


Figure 8: many to many RNN

2.1.5.4 Many to many (where number of input is not equal to number of output)

Here number of input is more than one and number of output is more than one, but the number of input and output may or may not be equal.

Example: **Machine Translation**

Here input is sequence of words in a particular language and output is sequence of words of another language. Here the number of words of given language may not be equal to number of words in translated language.

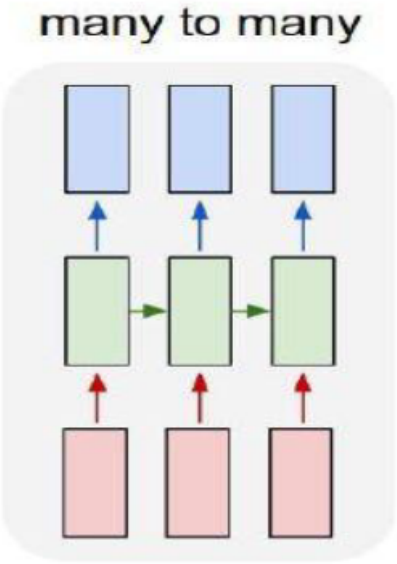


Figure 9: many to many RNN (np. of input = no. of output)

2.1.5.5 Many to many (where number of input is equal to number of output)

Here the number of input is equal to number of output.

Example: **Video classification on frame level**

Here first video is divided into frames and then frames are labelled.

2.1.6 Mathematical formulas of RNN

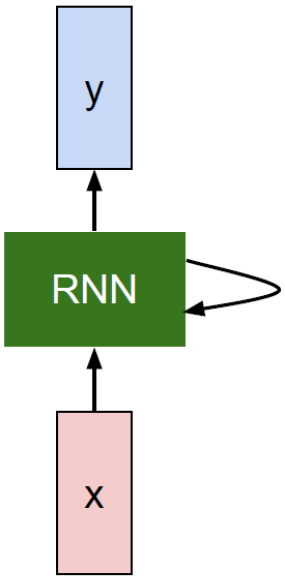


Figure 10: Flow diagram of an RNN

$$h_t = f_W(h_{t-1}, x_t)$$

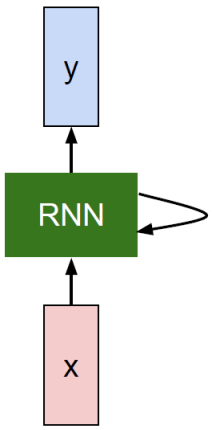
new state — old state — input vector at some time step

some function with parameters W

$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.

The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

2.1.7 RNN: Computation Graphs

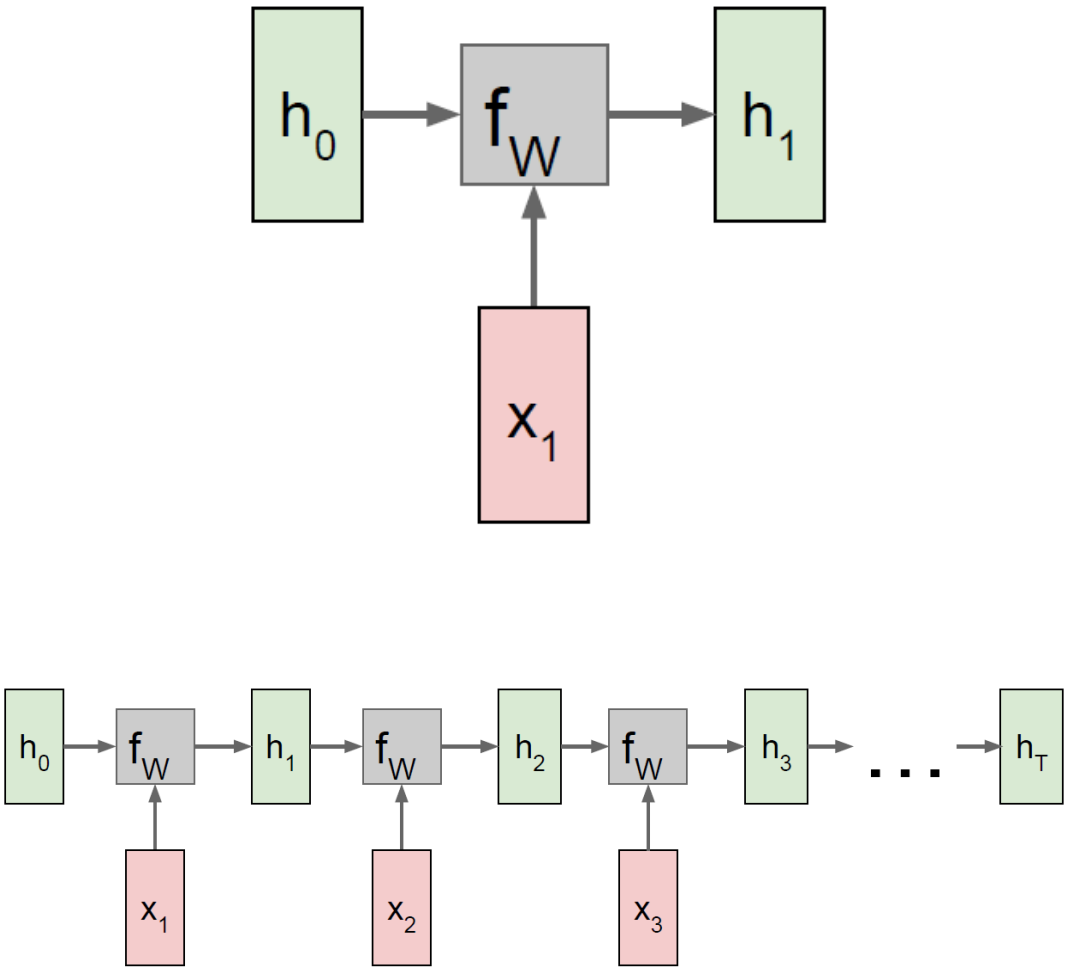


Figure 11: Computational Graphs of RNN a)one time step b)multiple time steps

2.1.8 Problems with RNN

RNN suffers from Gradient flow problem.

First it is important to know what is gradient descent. In a neural network a very basic but important task is to optimize the network. And by optimization it is meant to minimize the cost of learning and find the values of different parameters and hyper parameters of the model like learning rate, batch size, number of epochs etc.

Let's discuss the different between parameters and hyperparameters.

- Parameters are the properties of the training data that are learnt during training. These define the skill of the model. These are not usually set by the developer. Example: weights of a network.
- Hyperparameters are those which cannot be learnt from the model. But these are to be set before training. They help to estimate the parameters. Hyperparameters have to set by developer. Tuning hyperparameters helps the model to work more efficiently. Example: learning rate, decay, number of hidden layers, number of epochs, batch size etc.

To understand gradient descent, it is easy if a bowl is considered as in Fig 11.

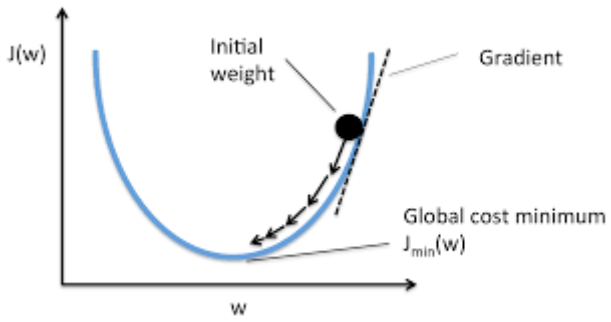


Figure 12: Gradient Descent

First, a random position into the bowl is considered. Which is assumed to be the current values of the coefficients(weights). In the above graph x axis represents the weights and y axis represents the cost of the training. So, the minimum of the cost will be achieved when the weight is in the bottom of the bowl.

To reach the bottom of the bowl, different values of the coefficients are tried, cost is evaluated and then select other coefficients that gives slightly lower(better) cost. Continuing this process will lead to the bottom of the bowl.

2.1.8.1 Mathematical procedure

Suppose the initial values of the coefficient for the function is 0.0.

Coefficient = 0.0

And suppose cost function is f, i.e. cost = f(coefficient)

We know derivative is used to calculate the optimization point. So here the derivation of the cost is calculated, i.e. basically the slope of a function at a given point. In order to get the lower cost in the next iteration, the sign of the slope is important to know.

diff = derivative(cost)

Now if the value is decreasing, the coefficient values can be updated. A parameter is used to control over the update in each iteration. This parameter is called learning rate.

coefficient = coefficient – (lr∗ diff)

here lr stands for learning rate.

The above process is repeated until the cost of the coefficients is nearly zero.

Now the gradient descent algorithm often has problems of exploding gradient and vanishing gradient.

Though gradient descent works quite similarly for RNN's, there are some add ons:

- The information propagates through timesteps in RNN.
- At each time step cost function has to calculated.

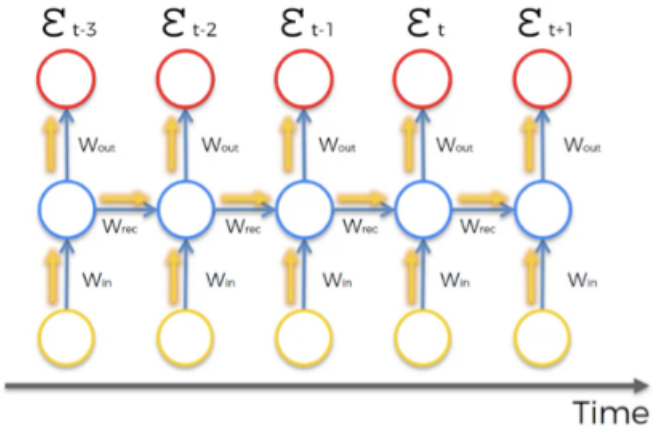


Figure 13: Timesteps in an RNN

Here et stands for error term.

After calculating et, it is propagated back into the network to update the weights.

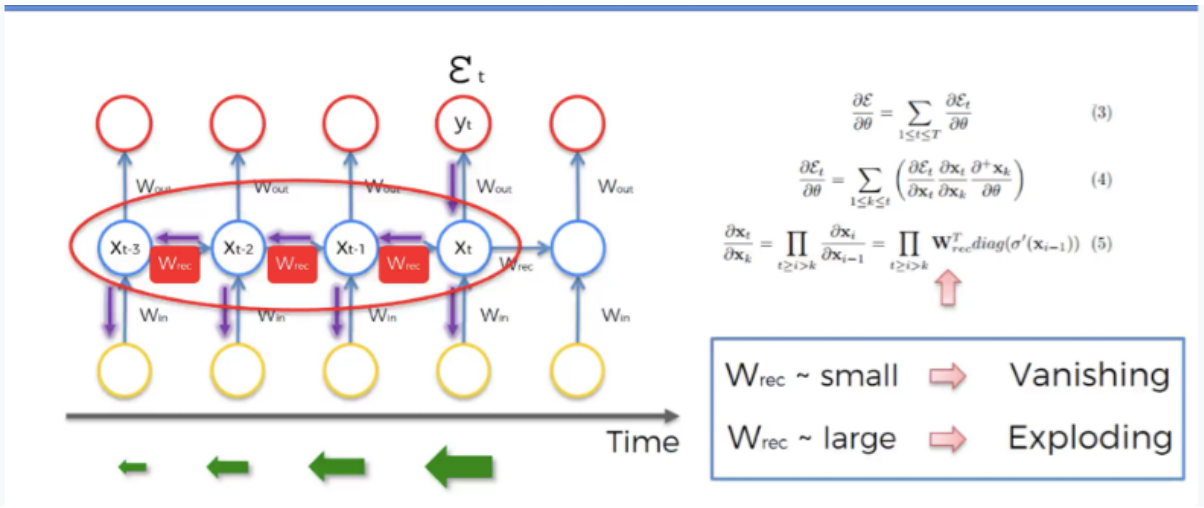


Figure 14: The problem of Vanishing and Exploding gradients in RNN

So, if the gradient is very small and as it is multiplied back in each timesteps it will just result near zero result and if the gradient is very high it will just explode in same way.

2.1.8.2 Solutions to the vanishing gradient problems

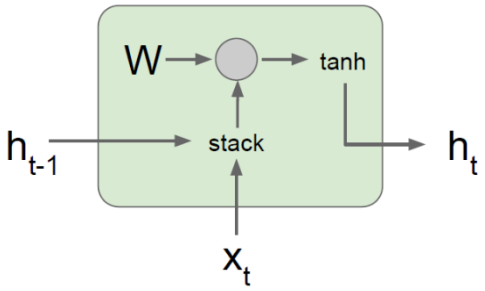
In case of exploding gradient:

- Back propagating is stopped after some point.
- Reduce gradient in some way.
- Put a limit on gradient.

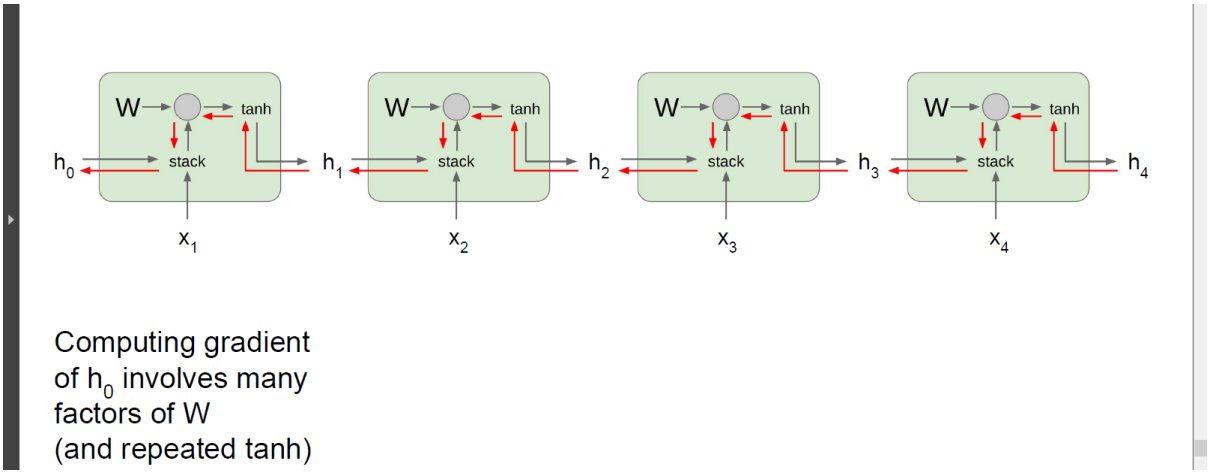
In case of vanishing gradient:

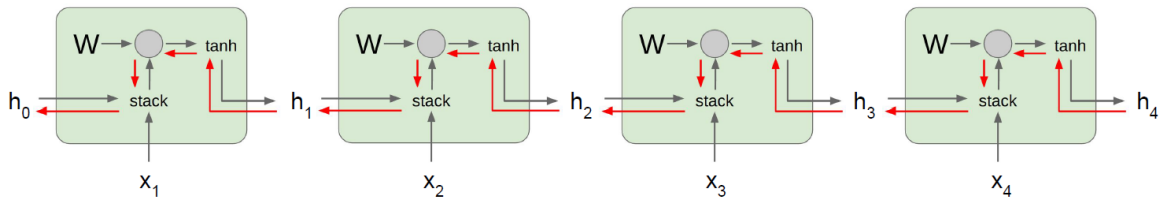
- Initial weight in a proper manner so that vanishing gradient is minimized.
- Having long short-term memory (LSTM) networks.

2.1.8.3 Gradient flow in RNN



$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$





Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

Figure 15: Solving Exploding and Vanishing Gradients

2.2 LSTM (Long Short-Term Memory)

LSTM is a special kind of RNN, which extends memory. That's why it is perfect to learn from the patterns or trend that have very long-time difference in them. It also solves the Vanishing Gradient/ Exploding Gradient problem.

Basic difference between RNN and LSTM as following.

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix}\right)$$

LSTM

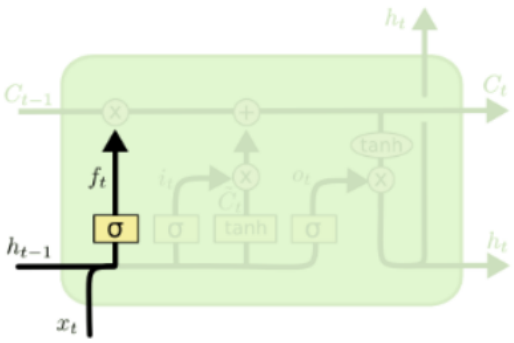
$$\begin{pmatrix}i\\ f\\ o\\ g\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix} W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

A simple LSTM cell has four gates. Those are forget gate, input gate, gate to write the cell, output gate.

2.2.1 Forget Gate



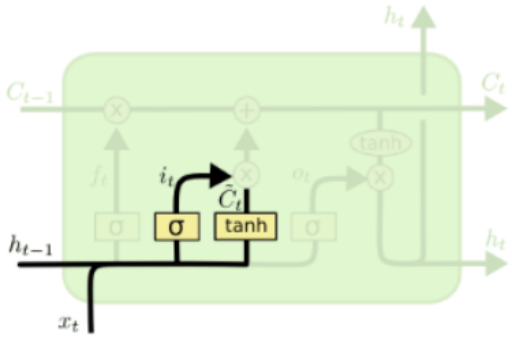
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fig 15 : Forget Gate in LSTM

After getting h(t-1) i.e. the output of previous step, this gate helps to take decision about how much information is to forget from the previous state. In this way it tries to keep

only relevant stuff. It uses sigmoid function to keep the input between 0 and 1 both inclusive. The above diagram represents the Forget Gate.

2.2.2 Input Gate

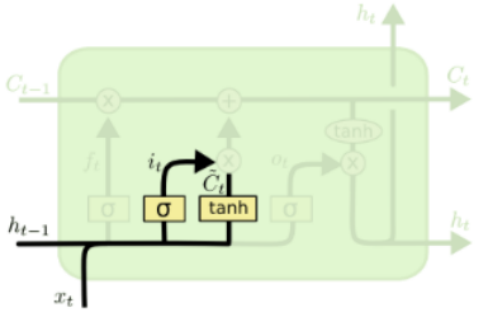


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig 16 : Input Gate in LSTM

This gate decides whether to write the present cell from the present cell or not. tanh layer helps to create a vector to add to present state and sigmoid layer decides the values to be updated. The above picture describes the Input Gate.

2.2.3 Output Gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig 17 : Output Gate in LSTM

It decides how much to revel as output. The following picture describe this cell.

2.3 Trading

A trading strategy states when to buy, sell or hold our financial assets in order to achieve better returns. It tells us to go long or short in markets at the given circumstances. There are two common trading strategies we deal with in this project:

Momentum Strategy

Also known as divergence or trend trading, this strategy believes in the movement or lack of it, of a quantity i.e. it expects the movement to continue its trend. Stocks with an upward trend tend to move up and those with downward seem to go further too as per the intuition of this strategy.

Common examples of this strategy include :

- Moving Average Crossover
- Dual Moving Average Crossover
- Turtle Trading (Richard Dennis)

Reversion Strategy

Also known as convergence or cycle trading, this strategy begs to differ from the current trend and believes that the movement of an asset will eventually reverse. For example the Mean Reversion strategy insists that stocks return to their mean prices and aims to exploit when stock prices deviate away from the mean.Some Common strategies in this category are :

- Mean Reversion Strategy
- Pairs trading Mean Reversion Strategy

We implemented the moving average crossover strategy of quantitative trading which uses two separate Simple Moving Averages(SMA) of a time-series with a short and a long lookback period. If the short moving average exceeds the long moving average then we buy a stock or we go long and if the reverse happens then we exit or we go short(sell the stock).

Fig 18 : Plotting signals on moving averages

3 Finance Glossary

To get a better understanding of all this lets get done with the basic terminologies first

- Share

  A unit of ownership that entitles the holder to an equal proportion of the company's capital.

- Stock

  A collection of shares owned by a member that signifies his/her ownership of the company. Unlike shares, stocks can have different denominations.

- Stock Trading

  Buying and selling of stocks in the stock market. Orders are provided to the stock breaker regarding the trade and traders generally have a stock trading plan with which they wish to achieve the best returns on their investment. There are many types of stock trading including Short-term, Intra-day, Long-term, swing trading etc.

- Algorithmic Trading

  The application of automated pre-programmed algorithms to execute orders in stock trading. It makes use of various strategies like esimated moving averages, mean reversion etc.

- Backtesting

  The process of applying an analytical method or a trading strategy to real-time or historic data in order to check the efficiency of the strategy or method. Backtesting is used for checking the viability of a trading stratgy and compare it with other trading strategies. It is done on historical data to attain confidence of applying the strategy on real-time data.

- Time Series data

  A sequence of data points recorded at regular time intervals, over a given period of time. Instead of having different value of features for each training/testing instance, time series data has different values of the same variable at different time steps. Time is hence the primary axis here. For example, Tracking and eventing applications and the Financial trading systems render time series data.

- Sharpe Ratio

  The term is coined by William F. Sharpe. This ratio helps investors to understand the return of an investment in an algorithmic trading given its risk. The ratio is the average return earned in excess of the risk-free rate per unit of volatility or total risk.

- Buy and Sell

  Buy means buying stock(s) and Sell means selling stock(s). Investors take help of different algorithmic trading strategy to take decision when to Buy or Sell. Ultimate motive is to make profit though these.

- Portfolio

  Portfolio is collection of different financial assets such as stocks, commodities, bonds, currencies, cash equivalents etc.

- Uptrend and Downtrend

  Uptrend is basically when a company's stock return is increasing day by day. And Downtrend is just the opposite.

4 Models and Strategies

For each of the subsequent sections there are two subsections each for the two projects we aim to document, namely

1) Kaggle NYSE stock prediction

2) Research Paper Implementation: Baek and Ha Young Kim

4.1 Data

The financial data is an indicator of business performance and the efficiency of strategies/tactics applied in the business. It consists of sets of information related to the financial status of a business such as the price of shares of the company. The stock price comes under the category of Time Series data in which the movement of certain data points is recorded at regular gaps or intervals over a given time.

1) For the first LSTM model we built, we used the "New York Stock Exchange" kaggle dataset which consists of historical prices and fundamental data about the S& P 500 companies.
   The data can be downloaded from : https://www.kaggle.com/dgawlik/nyse

   Contents :

   - **prices.csv**: original, unaltered daily prices. Most of data ranges from 2010 to the end 2016 except for companies added late to the stock market index in which case the time frame is shorter. The data doesn't account for stock splits which count to about 140 in the said time.
   - **prices-split-adjusted.csv**: same as prices with added adjustment for stock splits .
   - **securities.csv**: general information about each company with division on its sectors
   - **fundamentals.csv**: metrics extracted from annual SEC 10K filings (2012-2016). These an be used to derive the popular financial indicators that are needed.

```
df = pd.read_csv("../input/prices-split-adjusted.csv", index_col = 0)
df["adj close"] = df.close # Moving close to the last column
df.drop(['close'], 1, inplace=True) # Moving close to the last column
df.head()
```

	symbol	open	low	high	volume	adj close
date						
2016-01-05	WLTW	123.430000	122.309998	126.250000	2163600.0	125.839996
2016-01-06	WLTW	125.239998	119.940002	125.540001	2386400.0	119.980003
2016-01-07	WLTW	116.379997	114.930000	119.739998	2489500.0	114.949997
2016-01-08	WLTW	115.480003	113.500000	117.440002	2006300.0	116.620003
2016-01-11	WLTW	117.010002	114.089996	117.330002	1408600.0	114.970001

Figure 16: first 5 records of the dataset used from Kaggle

2) The SingleNet model proposed in the paper uses only the historical closing prices of S& P index while the ModAugNet models use the S& P index as well well as data from selected 10 companies in S& P500. The closing prices are collected from January 4, 2000 to July 27, 2017 on working day basis. The selected 10 companies retrieved from the S& P index are listed on the exchange before 2000 and presently rank in the top 25 companies by market capitalizations. The closing prices of these companies are all highly correlated with the S& P500 stock market index. We calculated the numerical values of this correlation using Pearson's correlation coefficient and only the companies with correlation greater than 0.8 were selected (under 5 percent significance level and all p-values less than 0.05).
All the data is retrieved from the Yahoo! Finance API and the S& P500 components are listed on the NASDAQ stock market.

```
snp_data.head()
```

	Adj Close
Date	
2000-01-04	1399.420044
2000-01-05	1402.109985
2000-01-06	1403.449951
2000-01-07	1441.469971
2000-01-10	1457.599976

```
d
```

Ticker	AAPL	ADBE	AMGN	AMZN	ASML
Date					
2000-01-04	2.440975	14.791295	48.689911	81.937500	26.826624
2000-01-05	2.476697	15.083735	50.365246	69.750000	26.085730
2000-01-06	2.262367	15.206868	51.202930	65.562500	24.480459
2000-01-07	2.369532	15.945663	56.961960	69.562500	25.036133
2000-01-10	2.327857	16.561329	60.417358	69.187500	27.413177
2000-01-11	2.208785	15.422350	55.653084	66.750000	26.517927

```
d.corrwith(snp_data['Adj Close'], axis=0, drop=False).sort_values(ascending = False)

Ticker
ASML    0.939346
ADBE    0.931655
AMGN    0.920196
MSFT    0.920161
FISV    0.915146
SBUX    0.913389
CELG    0.913010
INTU    0.909671
AAPL    0.899090
REGN    0.898842
```

```
from scipy.stats.stats import pearsonr
for i in d.columns:
  print(i, pearsonr(d[i], snp_data['Adj Close']))

ASML (0.9393456110059639, 0.0)
ADBE (0.9316554196208873, 0.0)
AMGN (0.9201959633264232, 0.0)
MSFT (0.9201605242557374, 0.0)
FISV (0.9151462492481346, 0.0)
SBUX (0.9133890277443993, 0.0)
CELG (0.9130104913548708, 0.0)
INTU (0.9096707807385662, 0.0)
AAPL (0.8990895194220256, 0.0)
REGN (0.8988415016799999, 0.0)
```

Figure 17: (Top) Loading S& P index and company closing prices
(Bottom) Correlations and p-values for top 10 companies

4.2 Model Architecture

1) For the Keras NYSE dataset we extract the closing prices for a particular company's ticker and normalize the data using Scikit's min-max scaler to scale and translate each feature to make it lie between 0 and 1.
After normalization the data is divided into consecutive windows of 22 days and the total data is then split between train and test sets by allocating 90% to the train and 10% to test data. The training data is further split 10:90 to give 10% validation data and use the remaining 90 for trainin.
LSTMs consume input in the format [batch_size, time_steps, Features].
Corresponding to which train_X has the shape : [1565,22,5]

   The model consists of two LSTM layers with 256 units each with the first layer returning outputs at each time step (return_sequence = True) and the second returning only the final output (return_sequence = False). The output is then fed into a Fully Connected layer with 32 neurons and relu activation that further feeds the output to a single neuron output layer. The final layer has a linear activation function. Other parameters used with the model include
   Batch size : 512
   Epochs : 90
   Dropout rate : 0.3
   Validation split : 0.1
   Optimizer : Adam
   Loss : Mean Square Error
   Metrics : Accuracy
   No. of features : 5 (Open, Low, High, Volume, Adj_close)

   The output predictions are finally denormalized using inverse_transform() function of Scikit min-max scaler. Model is evaluated using Mean Square Error(MSE) and Root Mean Square Error (RMSE) methods as illustrated in the Results section.

2) The formerly stated methods yielded splendid performance and some really close predictions which led to the suspicion of Overfitting. To deal with overfitting the third approach put to use a separate 'Prevention Module' that augments data and sends additional data that is highly correlated to the the target value in hopes of

increasing the generalization power of our model.
The Prediction module in these modules is fed the target S& P 500 index value while the Prevention module takes as input the closing price of a combination of 5 companies (from the selected 10) at a time. Before feeding, the data is normalized to ensure that larger inputs do not dominate in training process. The original dataset is normalized by using the following equation at each data point :

$$\bar{X} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where, X os the original value at data point
Xmax is minimum value of the feature (closing price) that X belongs to
and Xmin is the minimum of that feature correspondingly.
Consequently, every feature (closing price) can be translated in the range [0,1].

The normalized data is then fed into a model. The paper specifies three models with following configurations:

SingleNet
The SingleNet consists only the LSTM prediction module that predicts the index as its target value. The lack of data point (approx 2000) leads to overfitting and hence a reduced prediction accuracy but the simple architecture of SingleNet still yields better test results than the ModAugNet-f. The model uses a lookback window of 20 days.
SingleNet has a LSTM layer with 4 units and ReLU activation unit followed by the output fully connected layer with a single neuron and linear activation, providing index predictions. Other parameters related to the model include:
Droupout rate :0.3
Metrics : Mean Square Error
Optimizer: 'Adam'
Number of variables/features : 1 (S& P500 Index value)

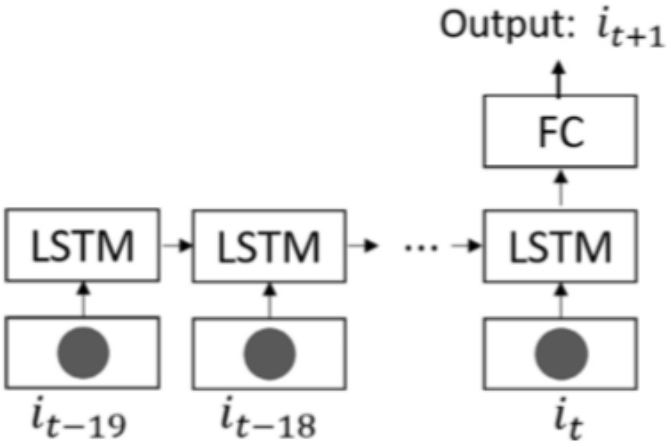


Figure 18: Architecture of SingleNet

ModAugNet

The ModAugNet uses a similar input for the Prediction module and a set of 5 companies from the selected 10 companies for Prevention module. These 5 companies are a fixed combination from the total possible $\boldsymbol{10C5 = 252}$ combinations in case of ModAugNet-f whereas ModAugNet-f uses every possible combination of selecting 5 companies from the available 10 with a particular window of Prediction input. A particular combination is chosen every 200 epochs in ModAugNet-c. The Prevention module is separated to input different feature values so that the model is not overwhelmed by indirect information fed to it.The input for Prevention module has five times more dimensions than that of Prediction module. The model should learn to give higher importance to the direct information (Prediction module input). If the model was not divided into two modules and the six inputs were collectively fed, the network would treat all the features equally.

Each module takes sequence data with a window of 20 days. The daily closing price of 5 companies for the Prevention module and the closing price of index as Prediction module input.The window is slid over the dataset.

ModAugNet consists of 5 LSTM units followed by a fully connected layer with 3 neurons in the Prevention module and 4 LSTM units followed by 2 neuron in the corresponding fully connected layer. The output of both these modules (fully connected layers) is further concatenated and taken as input or the next fully connected layer to predict the output. The activation function used for all fully connected layers except the final output layer is 'ReLU', for the final layer 'linear' activation is used. Remaining parameters used in the model are as follows:

Batch size : 32
Epochs : 200
Validation split : 0.2
Metrics : Mean Square Error
Optimizer: 'Adam'
Learning rate : 0.00005
Regularization : L2
Regularization constant : 0.001
Number of variables/features : 1 (S& P500 Index value) + 5 (stock closing prices)
Loss functions :

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

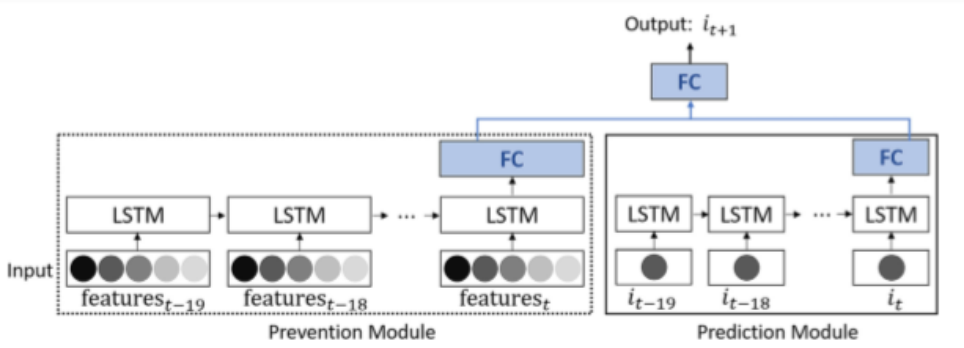


Figure 19: Architecture of the [proposed ModAugNet model

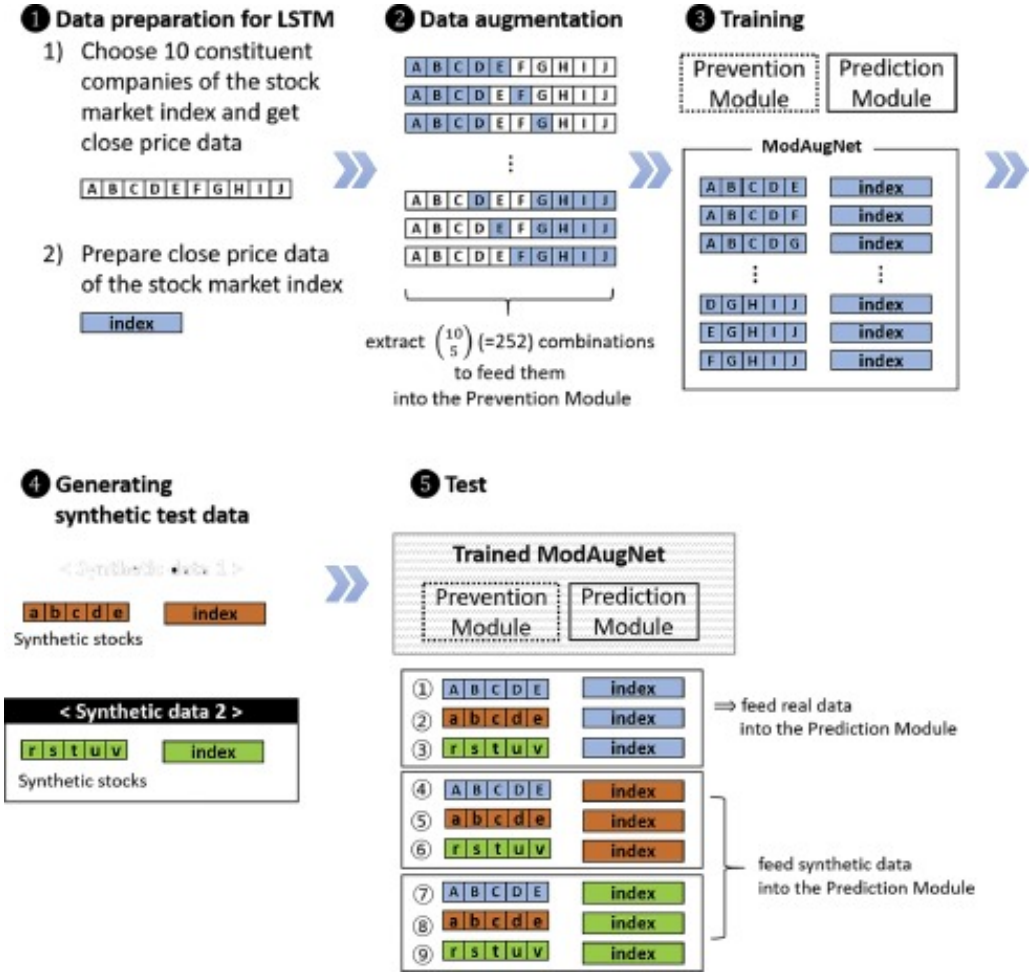


Figure 20: process of forecasting framework for stock market index

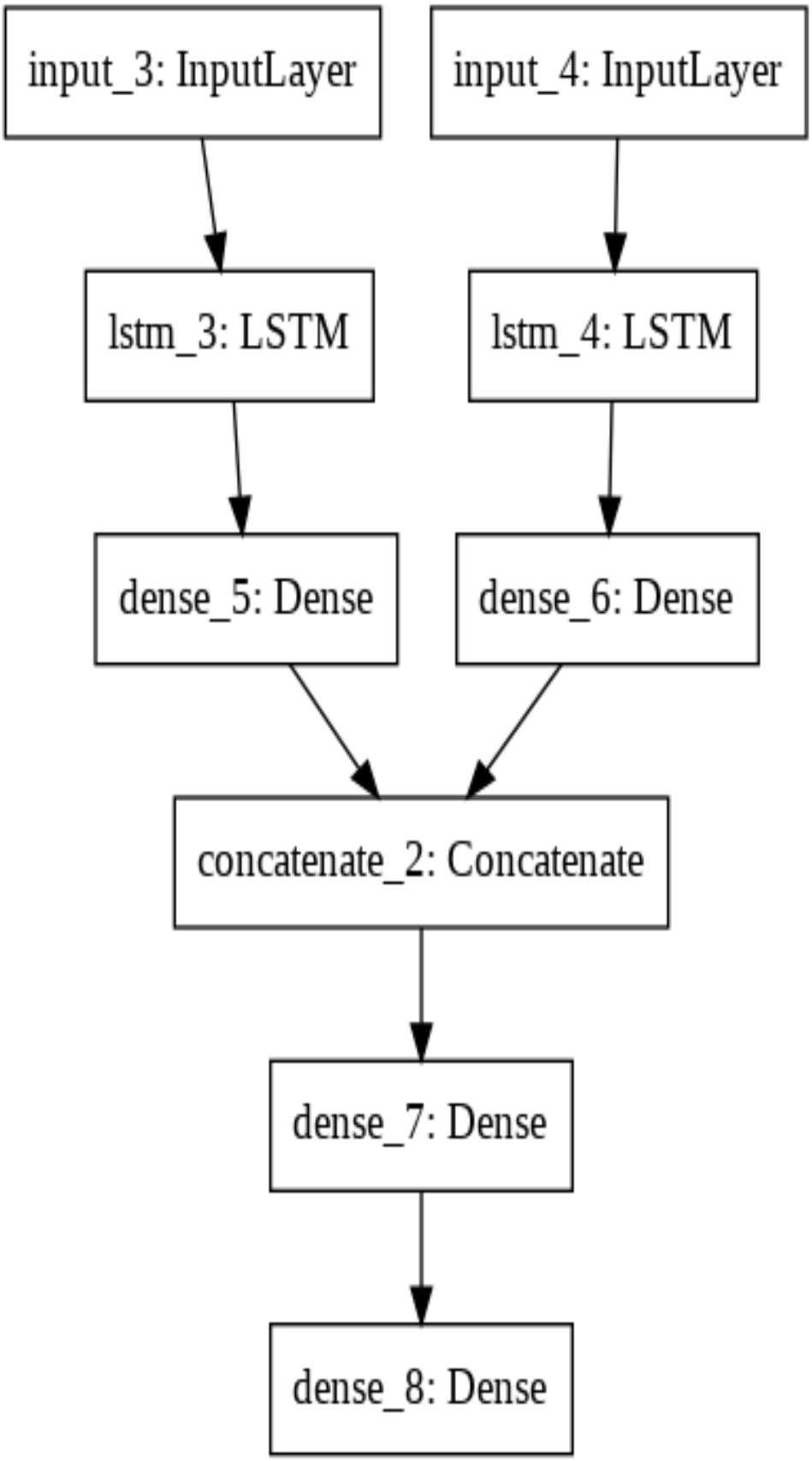


Figure 21: data flow in ModAugNet

4.3 Testing

For testing the model, two types of Synthetic data were generated.

Synthetic data 1 is composed by randomly choosing a number from a Uniform distribution ranging between the maximum and minimum values of the given feature. For prediction module data a number is randomly chosen from a uniform distribution between maximum and minimum S& P500 index value and correspondingly for the 5 stocks of Prevention module Synthetic data 1 is generated.

Synthetic data 2 is generated by arbitrarily choosing value of 10 for all data points.
The Synthetic data (1 and 2) is also normalized in the same fashion as Original data.The values of Synthetic data will be computed to 0 after normalization since the data consists of a constant value i.e. 10.

The two types of Synthetic data combined with the Original data give total nine possible test input sets, that can be illustrated as in Figure 20.
The data synthesized is demonstrated in figure 21.

<Synthetic data 1>

Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
30.2700*	19.3400*	8.6000*	7.3200*	25.0600*
95.0500**	58.5000**	80.0625**	46.3125**	107.2300**
93.9773	35.6673	42.8646	21.5123	42.5319
86.3682	40.0774	71.9137	32.7380	36.4135
90.8230	37.6242	32.6242	42.2130	50.0072
42.0761	32.7588	49.5146	8.0059	65.7760
54.4551	23.6496	47.7760	30.3630	60.7512
78.3708	22.1363	16.7528	7.4331	42.8505
45.1511	53.7986	29.9641	40.6810	87.0722
80.9823	58.3798	62.5843	25.7652	90.0763
52.9144	21.6238	13.5990	11.8126	75.0358
58.8015	32.4412	79.2704	21.6826	32.8648
80.6418	53.8477	72.9355	27.1330	66.6216
90.1018	41.7594	69.5793	7.8128	91.6469
50.9637	20.0238	32.1205	39.1383	64.3313
55.7780	47.6092	66.8461	30.8644	44.3372
74.5079	29.5871	11.4764	41.3768	57.5522
33.7314	44.9736	12.8056	26.9845	82.0032
88.5760	40.9349	15.5002	23.7163	69.8982
65.4187	53.4469	26.3636	15.0815	79.6973
83.7899	45.5478	16.6434	8.9368	39.2880
80.1729	46.2699	8.7943	46.0460	91.7732

<Synthetic data 2>

Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10

Figure 22: Example of Synthetic Data

5 Results and Performance Comparison

1) **Kaggle NYSE dataset**

The model scores were evaluated as :

Score	MSE	RMSE
Train	0.00109	0.03 RMSE
Test	0.00938	0.10 RMSE

Plotting the actual and predicted prices together we get Figure 23

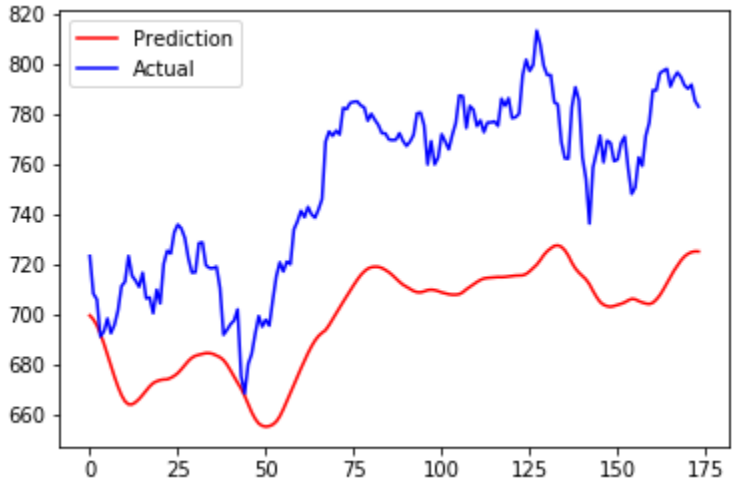


Figure 23: model predictions(red) and actual closing price values(blue)

The error resulted by this model can be reduced further by increasing the amount of data available from the current timeline of 2010-2016.

2) **The model scores were evaluated as following:**

SingleNet

Score	**Test**
MSE	**63832.03337786394**

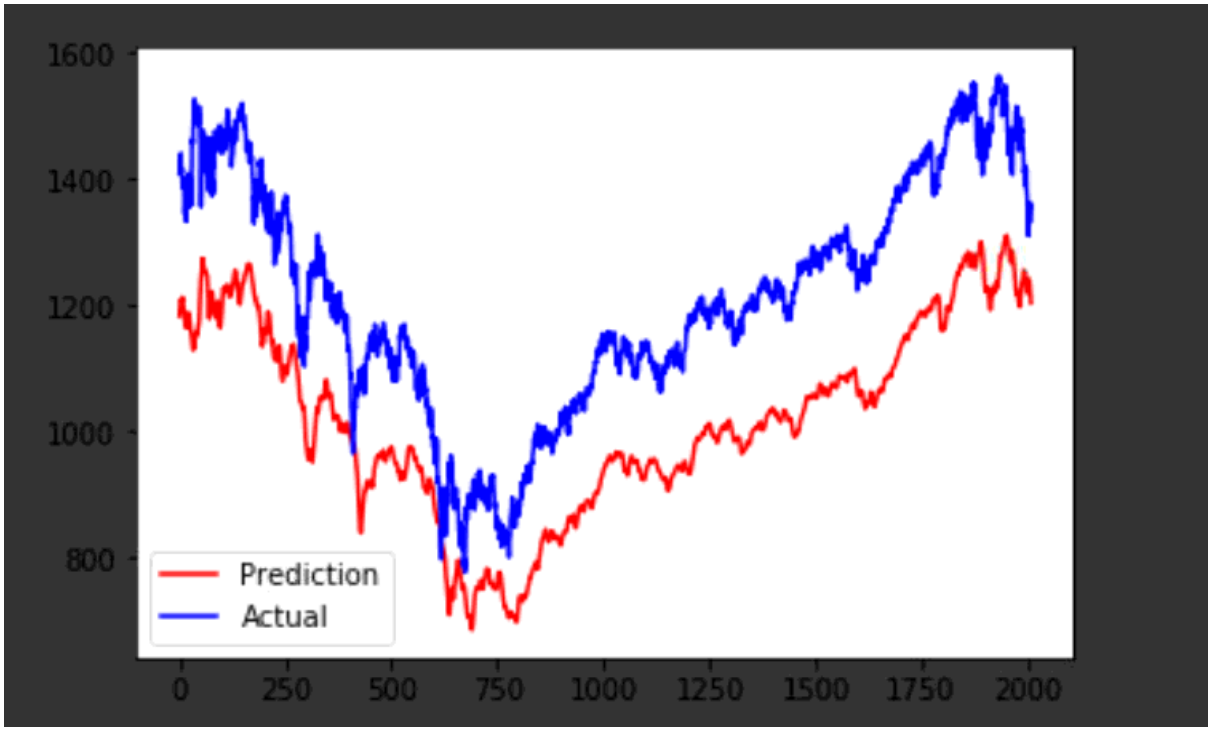


Figure 24: Training Predictions (SingleNet)

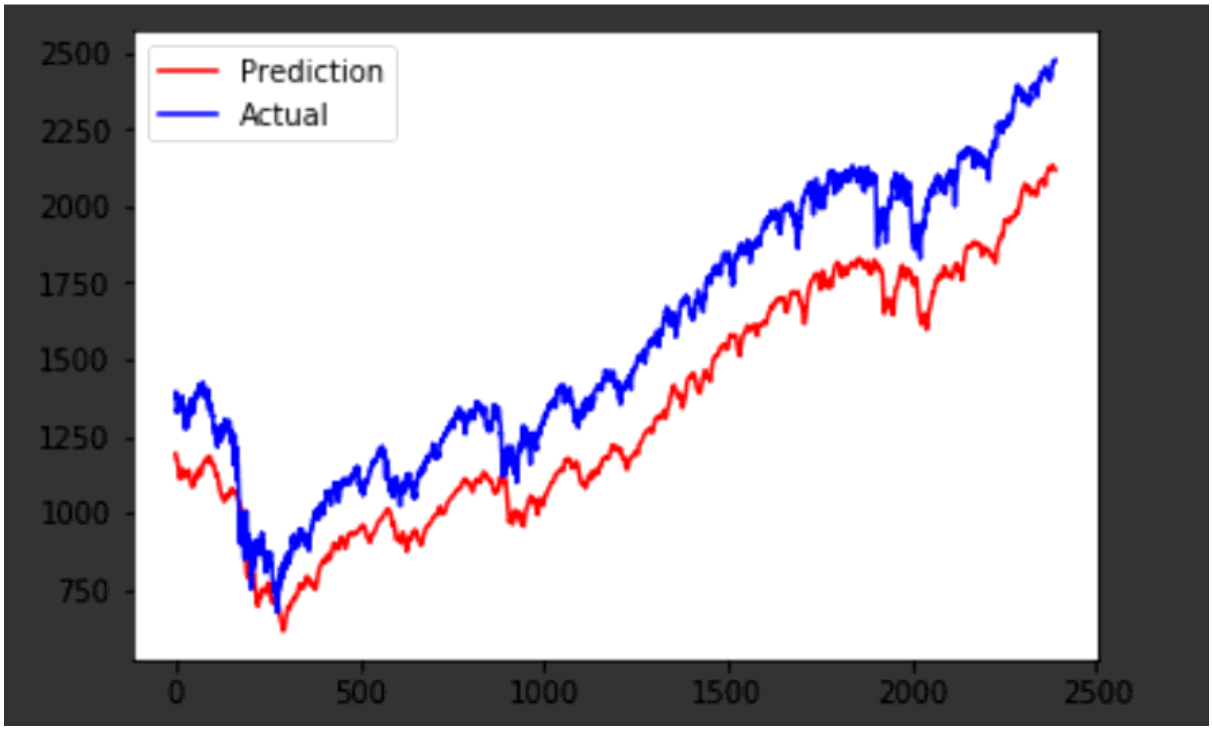


Figure 25: Test Predictions (SingleNet)

ModAugNet-f

Prevention Module	Prediction Module	Test MSE
5 stocks	**S& P 500**	302936.4565972191
Synthetic data 1	**S& P 500**	**2090688.4180960732**
Synthetic data 2	**S& P 500**	**432184.0252289429**
5 stocks	**Synthetic data 1**	**302936.4565972191**
Synthetic data 1	**Synthetic data 1**	**2090688.4180960732**
Synthetic data 2	**Synthetic data 1**	**432184.0252289429**
5 stocks	**Synthetic data 2**	**302936.4565972191**
Synthetic data 1	**Synthetic data 2**	**2090688.4180960732**
Synthetic data 2	**Synthetic data 2**	**432184.0252289429**

Plotting the actual and predicted prices together we get(for training Data) Figure 26.

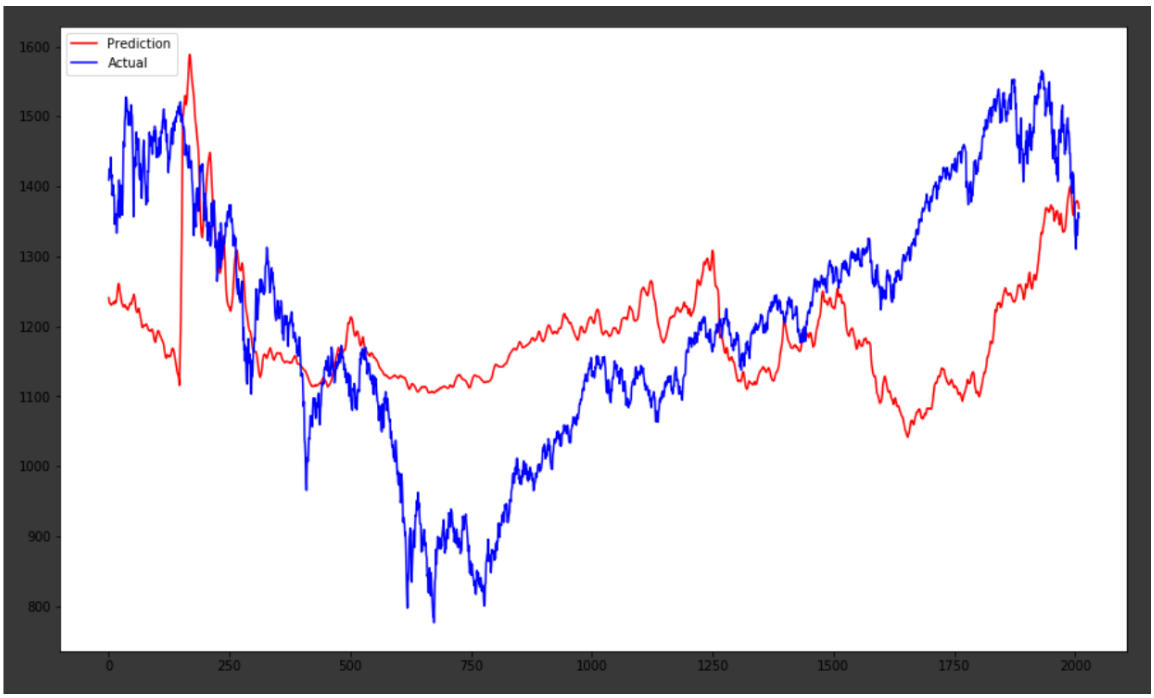


Figure 26

Plotting the actual and predicted prices together we get(when prediction module has S& P 500 testing Data):

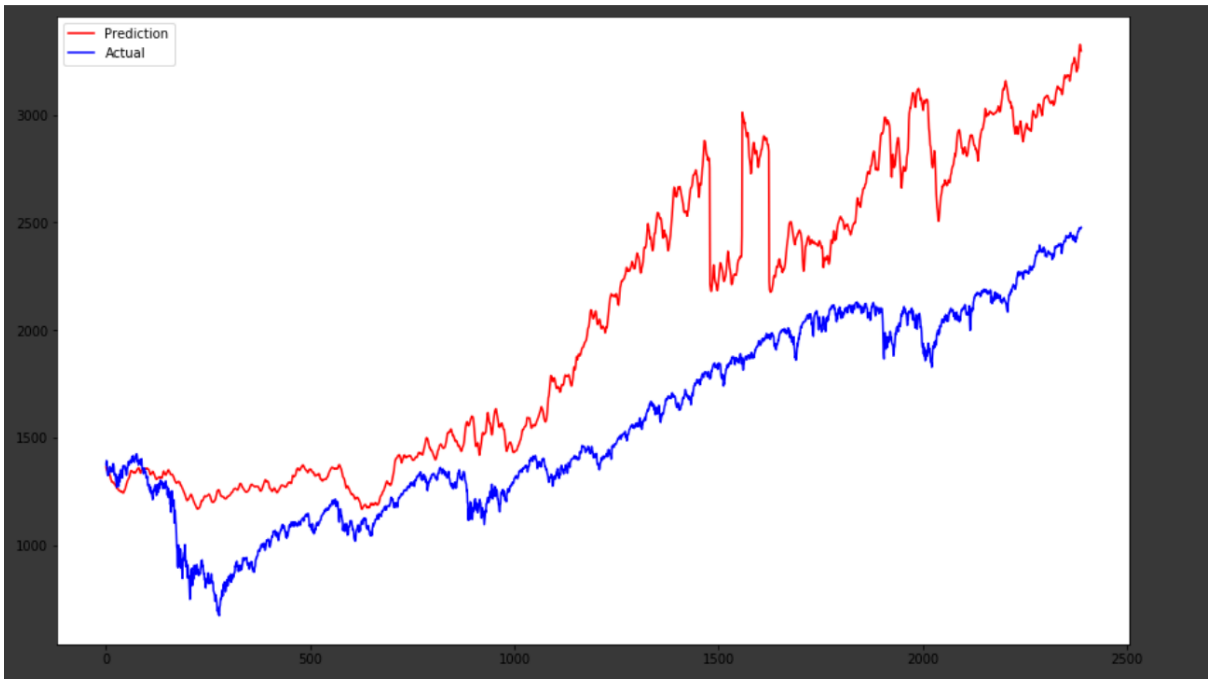


Figure 27: Prevention Module has actual stock index

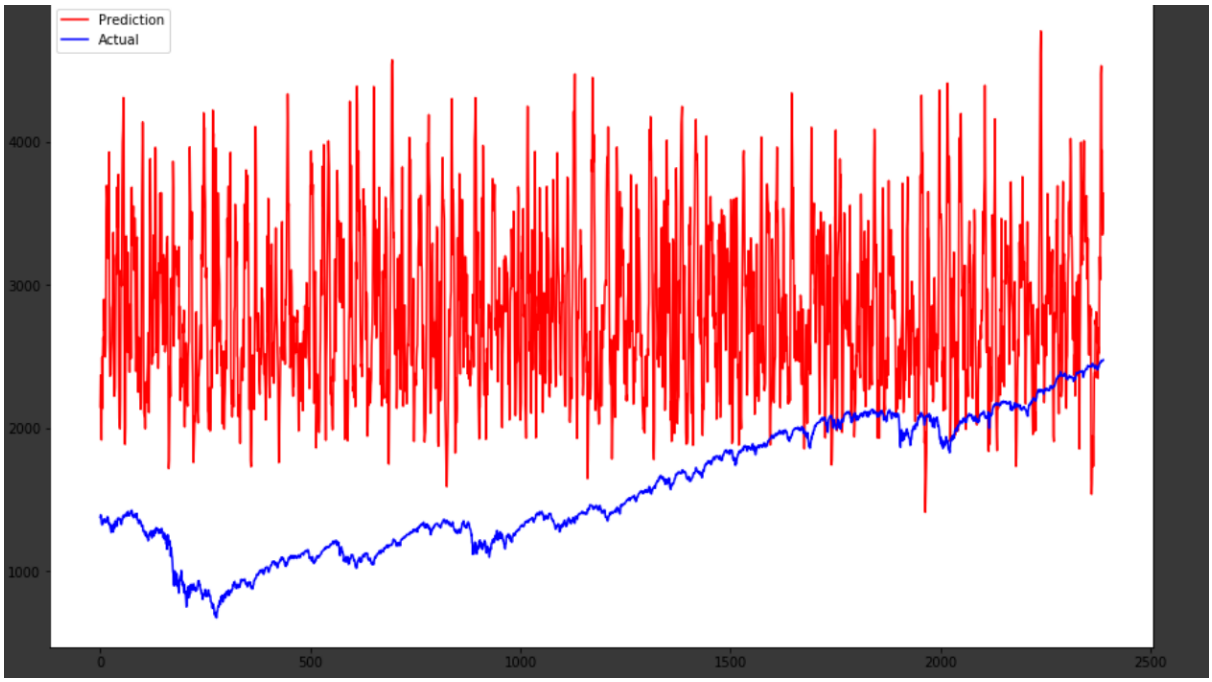


Figure 28: Prevention Module has Synthetic data 1

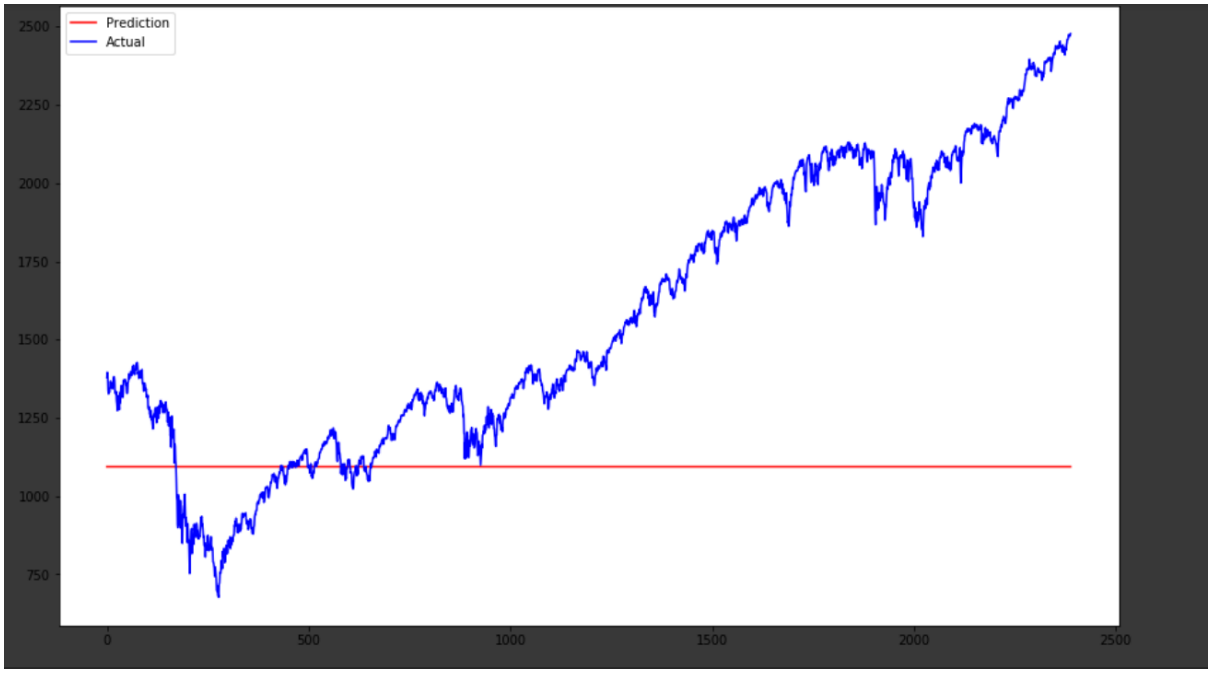


Figure 29: Prevention Module has Synthetic data 2

Plotting the actual and predicted prices together we get(when prediction module has Synthetic data1).

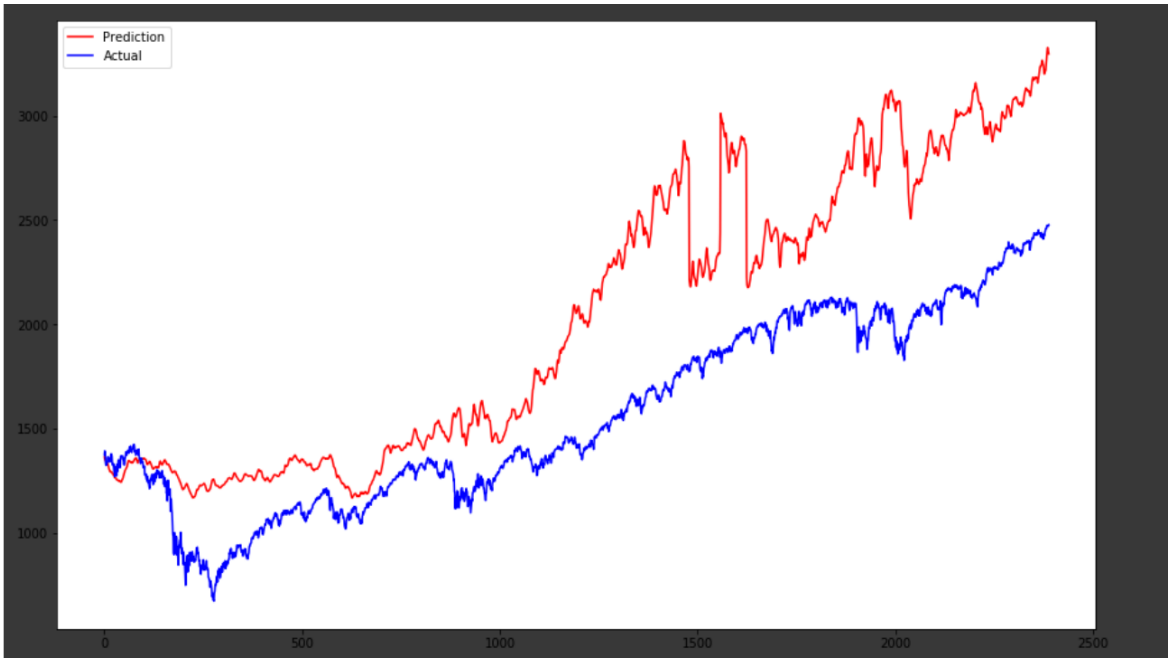


Figure 30: Prevention Module has actual stock index

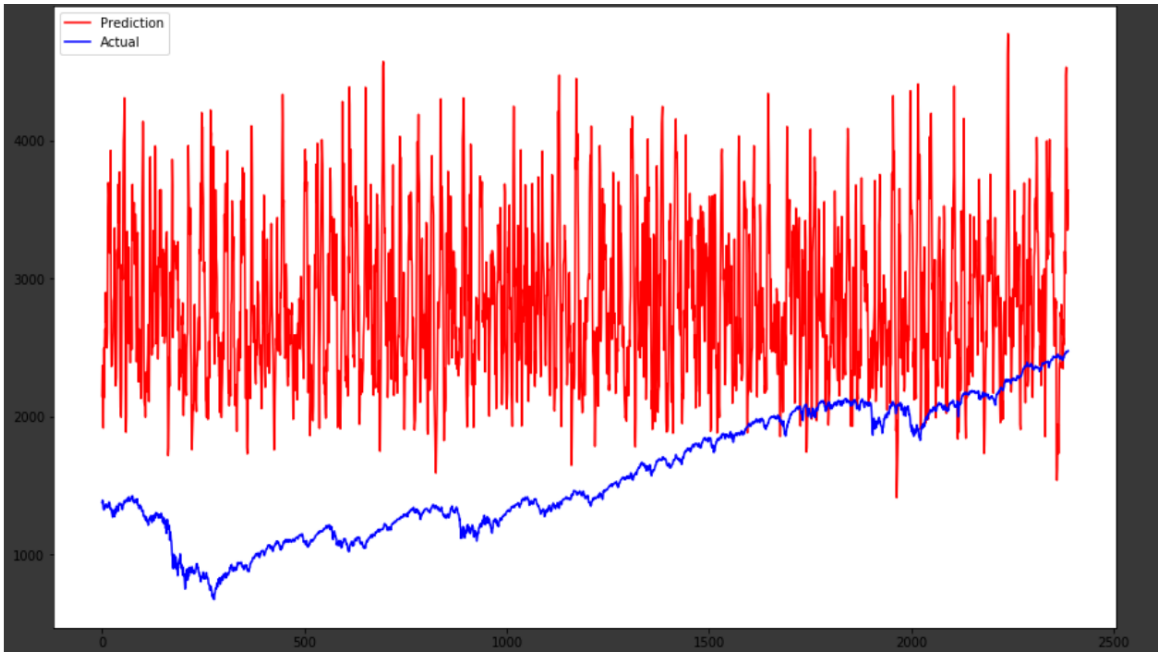


Figure 31: Prevention Module has Synthetic data 1

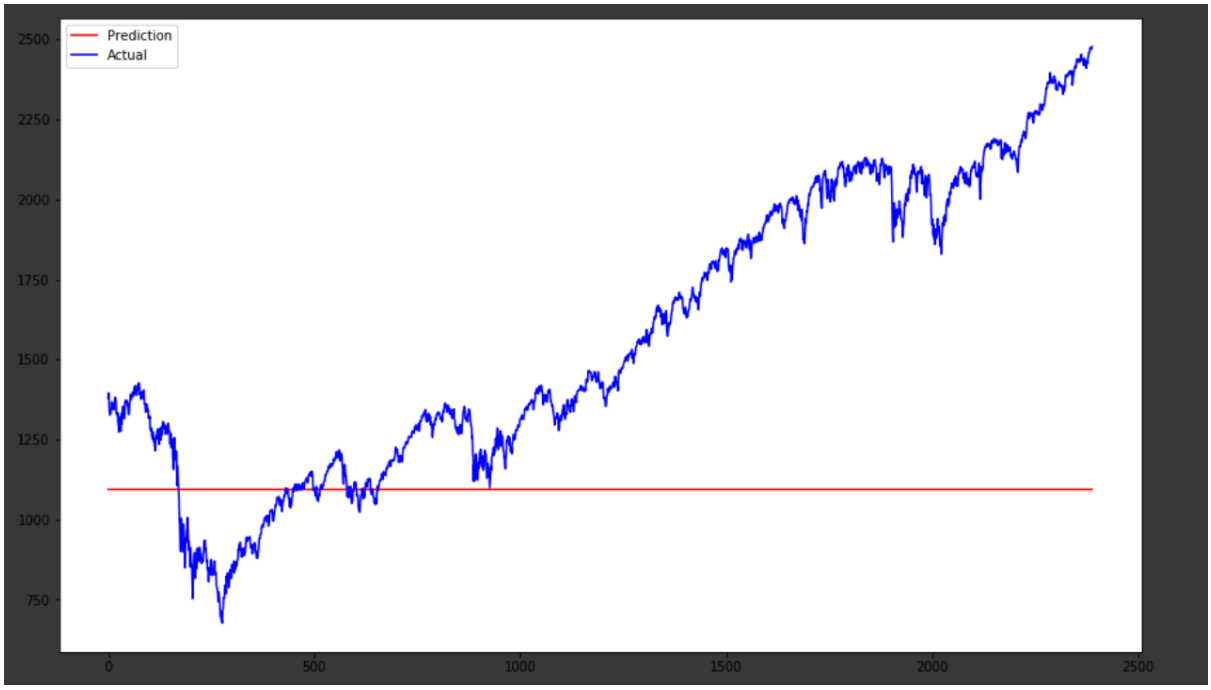


Figure 32: Prevention Module has Synthetic data 2

Plotting the actual and predicted prices together we get(when prediction module has Synthetic data2).

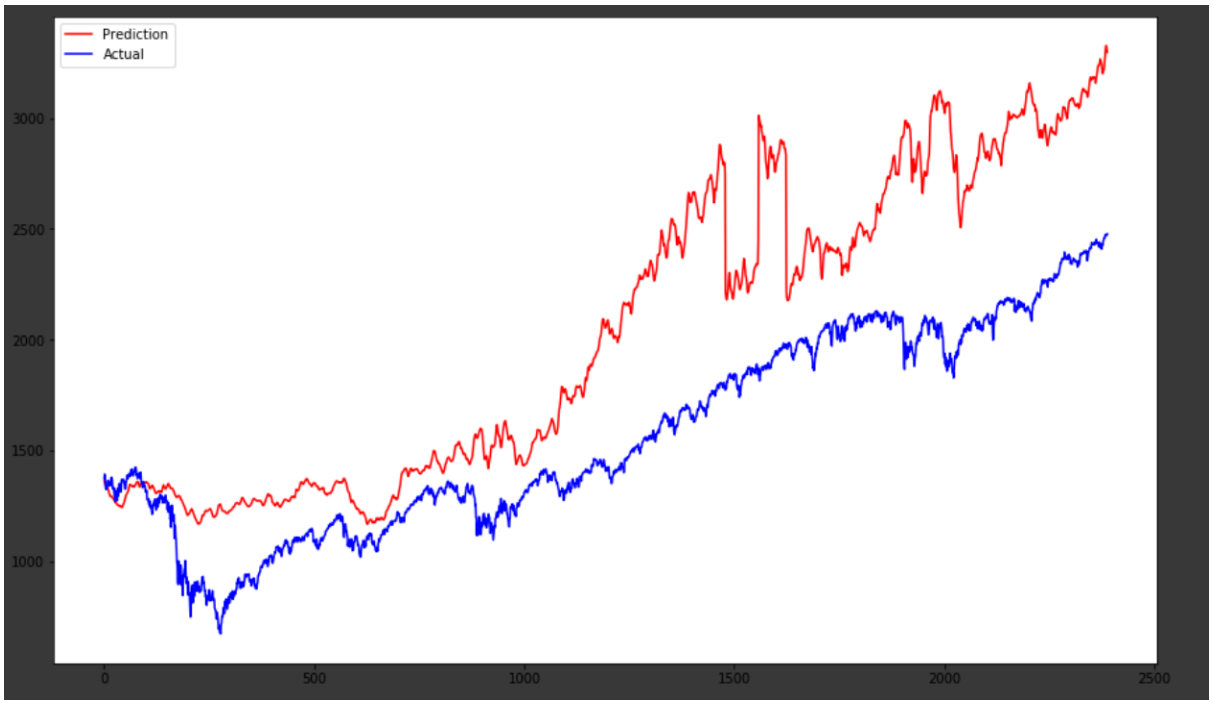


Figure 33: Prevention Module has actual stock index

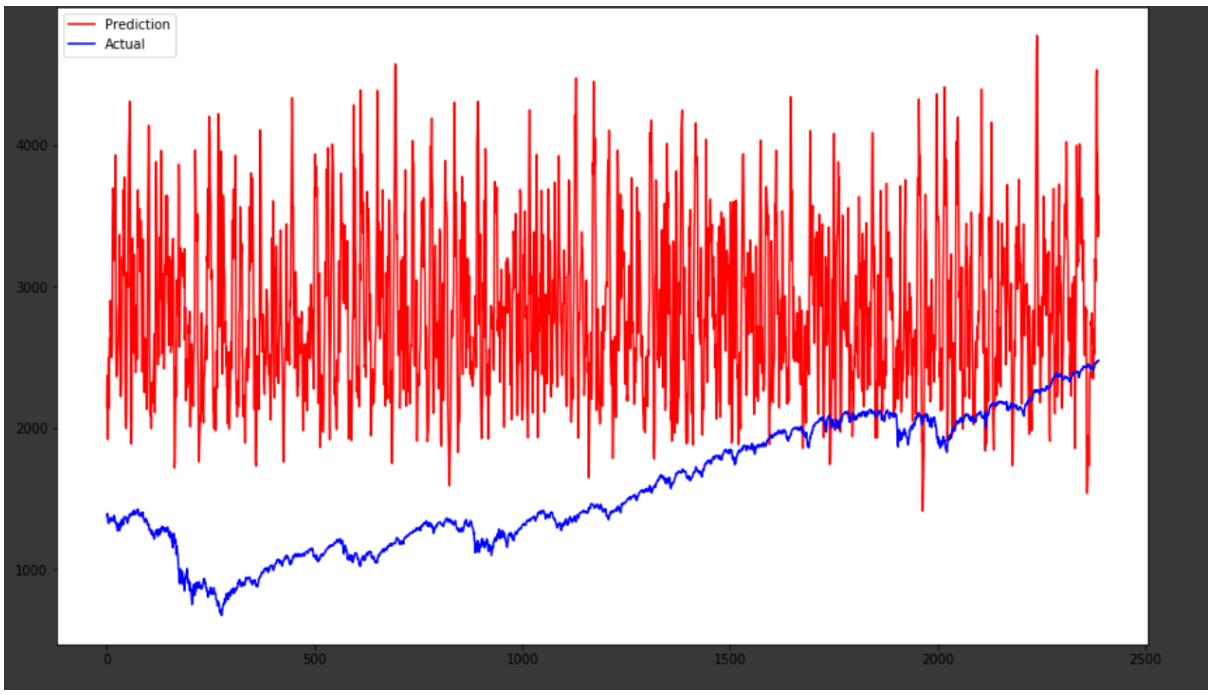


Figure 34: Prevention Module has Synthetic data 1

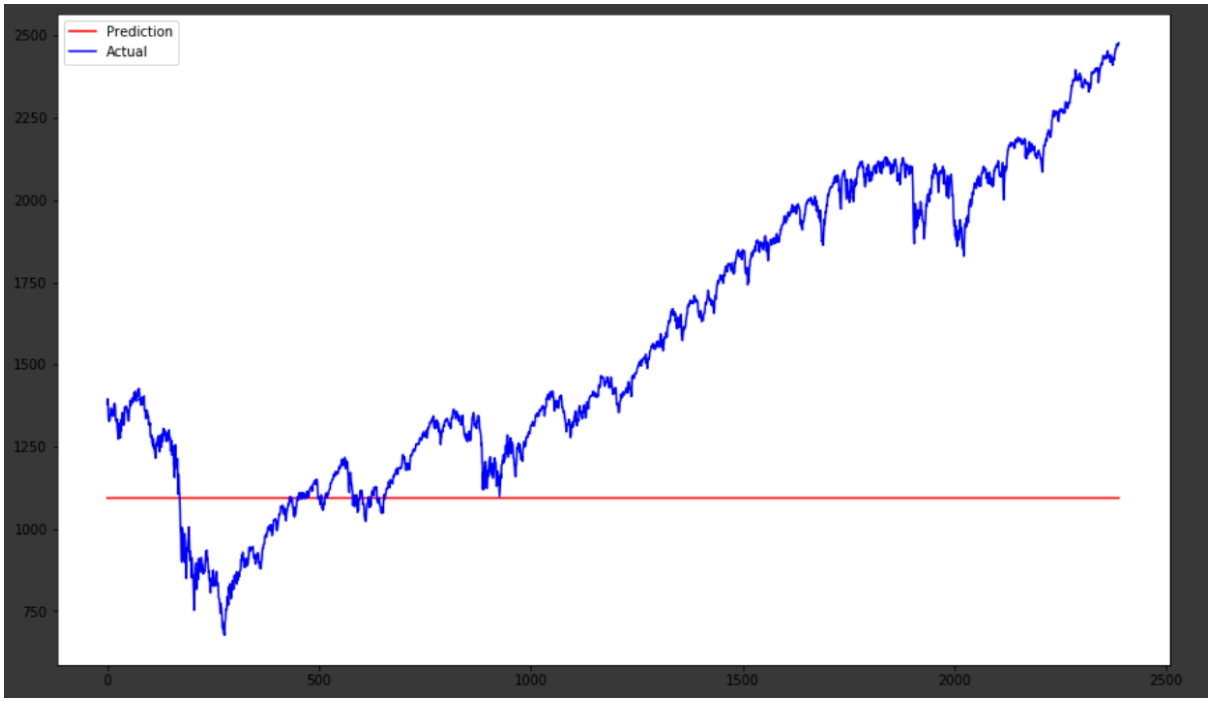


Figure 35: Prevention Module has Synthetic data 2

ModAugNet-c

Prevention Module	Prediction Module	Test MSE
5 stocks	S& P 500	38240.27795906685
Synthetic data 1	S& P 500	38240.27795906685
Synthetic data 2	S& P 500	38240.27795906685
5 stocks	Synthetic data 1	322712.86974726076
Synthetic data 1	Synthetic data 1	322712.86974726076
Synthetic data 2	Synthetic data 1	322712.86974726076
5 stocks	Synthetic data 2	943031.9892485047
Synthetic data 1	Synthetic data 2	943031.9892485047
Synthetic data 2	Synthetic data 2	943031.9892485047

Figure 36: Training results (ModAugNet-c)

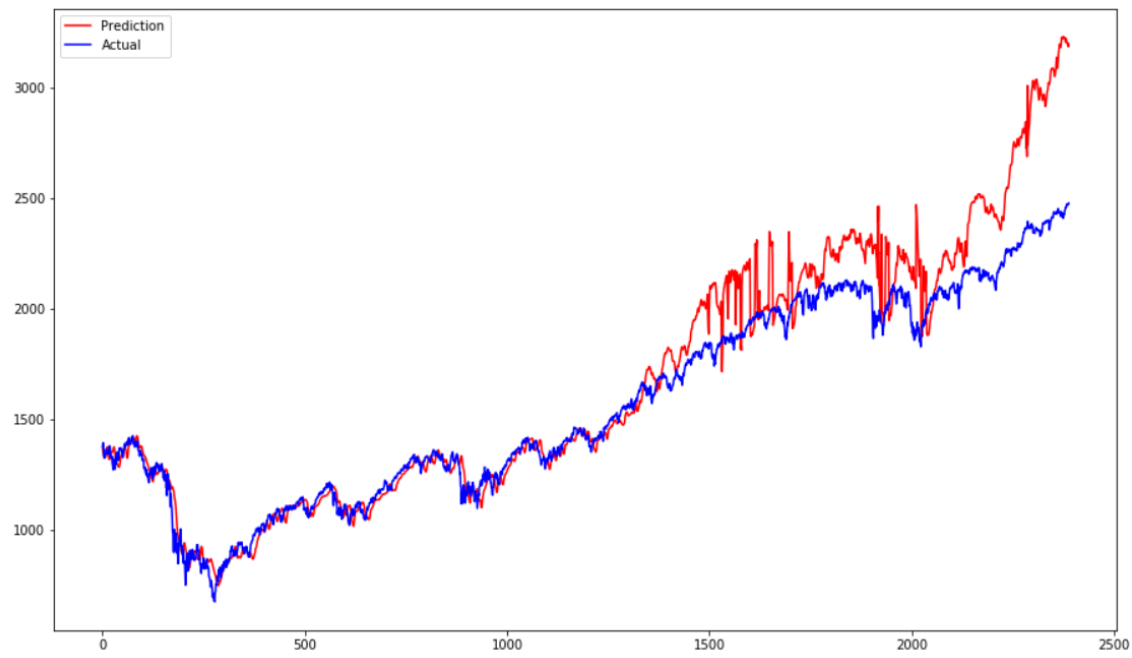


Figure 37: Test results (ModAugNet-c, Prediction Module : Actual Data)

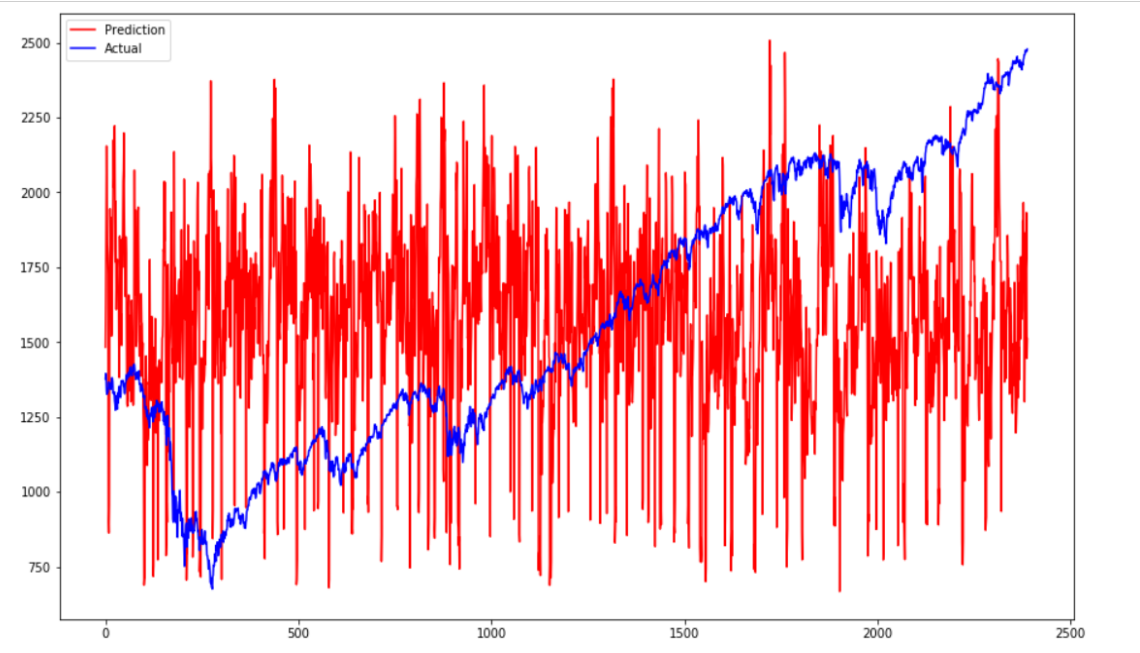


Figure 38: ModugNet-c, Prediction Module : Synthetic Data 1

Figure 39: ModugNet-c, Prediction Module : Synthetic Data 2

Loss for ModAug-c

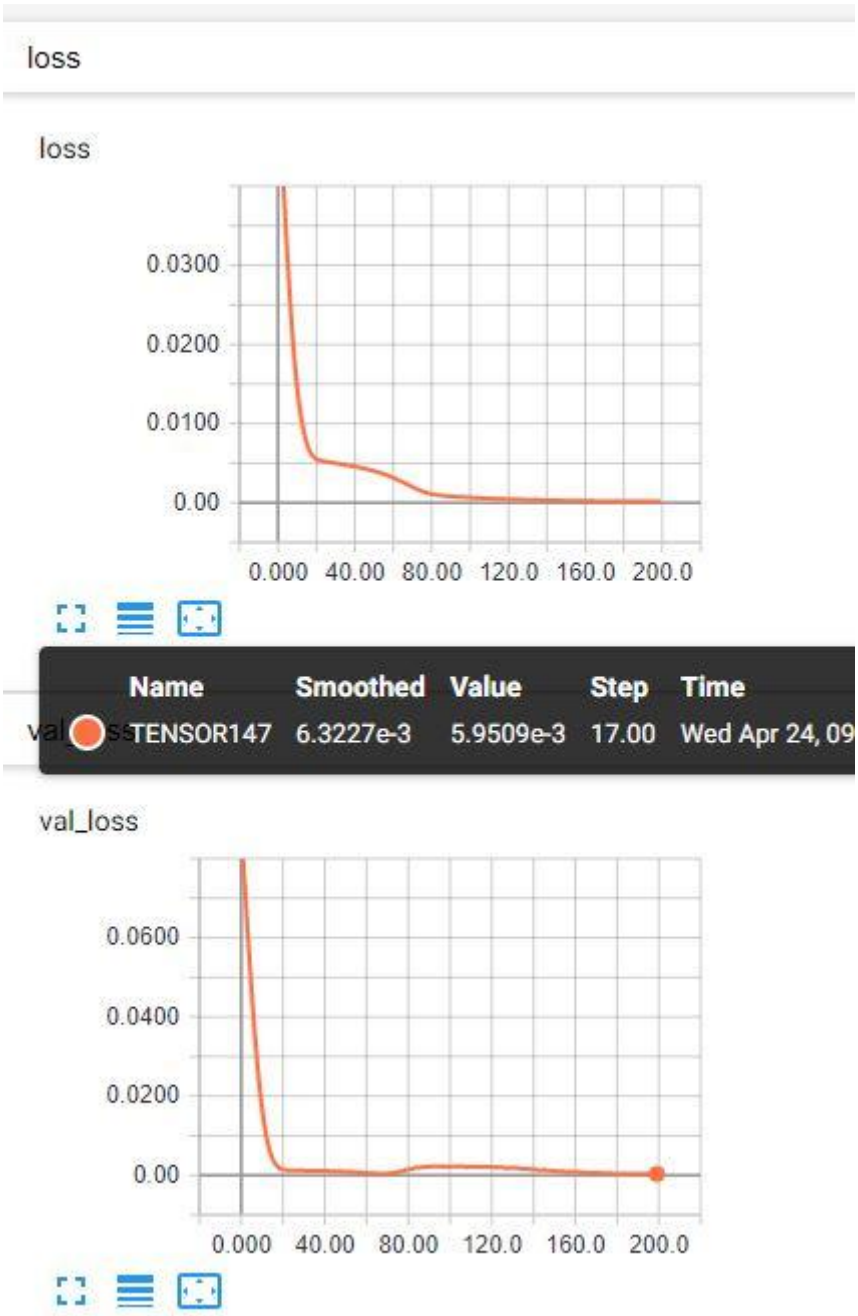


Figure 40: Loss after 252 iterations

Training loss for all iterations

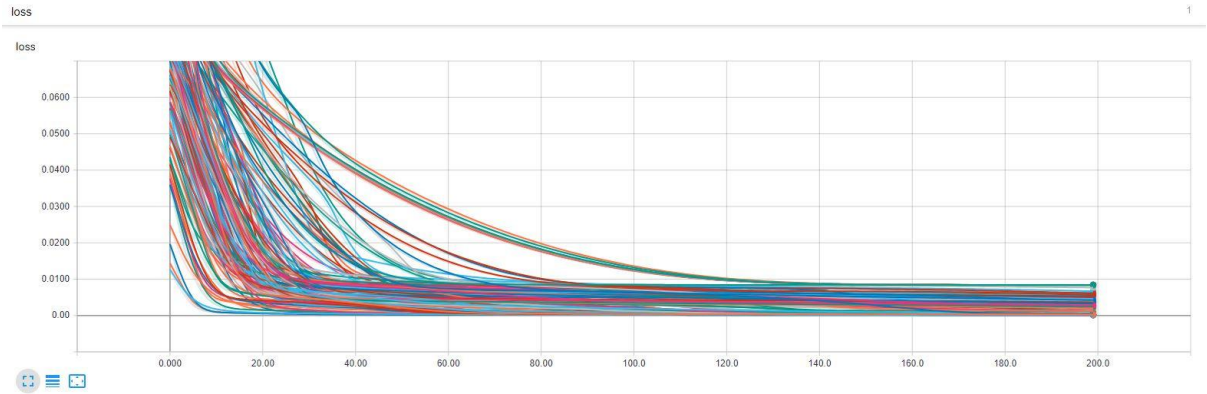


Figure 41

Validation loss for all iterations

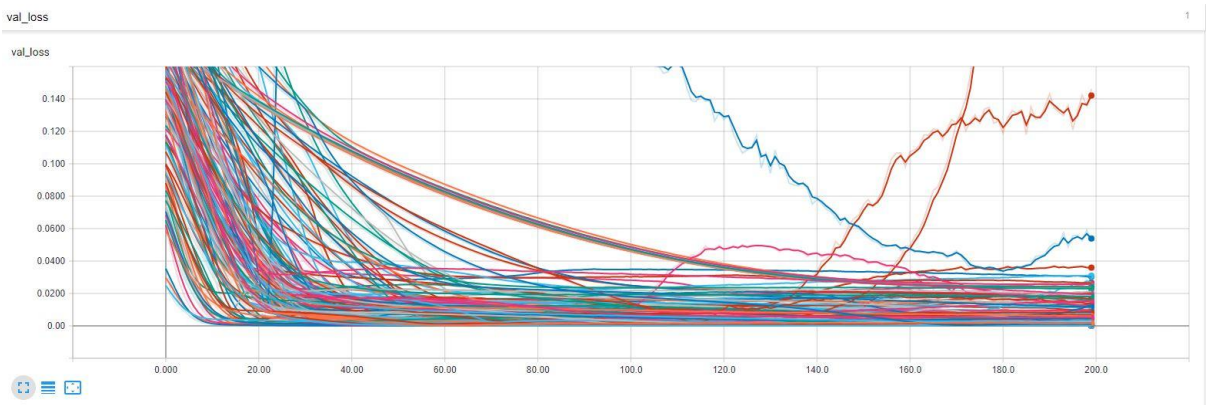


Figure 42

6 Conclusions and Learnings

Forecasting the price of a financial asset is of great importance as it reduces the risk of decision making for investors. The prices of financial assets are dynamic and respond are dependent on a lot of factors which decide its movement. Being able to account every single one of these in hope of making precise predictions is a challenging act. A large number of investment experts and firms have took advantage of this scenario and gained large profits from inexperienced investors. To improve this situation and join the Machine Learning bandwagon a number of studies are now focussed in predicting stock pricing and automating stock trading. These studies have indicated that LSTMs have been the best player in this field and deep learning is the way ahead.

Our work in this project was able to demonstrate the trends in financial data prediction and explore further application of these techniques. The LSTM networks like SingleNet and the Kaggle NYSE dataset prediction model established a firm baseline for our work from where we moved towards more complex and sophisticated networks like Attention network applied in the *Krauss and Fisher*paper. Hypertuning our parameters, applying different normalization techniques, adding Regularization and dropout to eliminate Overfitting and other optimization techniques along with these improved networks helped in achieving lower errors and better predictions. The novel ModAugNet architecture was a giant leap in attempts of reducing overfitting as it applied data-augmentation in an unorthodox way to get more training data and improve the generalization power of the model.

The models in our project employed the S& P500 data predominantly but we also trained these models on the National Stock Exchange (NSE) NIFTY index to compare the results. The model architecture and hyperparameters were all based on optimization of S& P predictions and hence need to be altered for NIFTY predictions.

Apart from the predictions we also prepared Algorithmic Trading strategies and compared a few of them based on their returns. These strategies leveraged portfolios and prices entirely implemented in the python environment. The predictions made by our LSTM models were then plugged in to these trading strategies for backtesting in hopes of bringing it a step closer towards real-time application. The results from different trading strategies were analysed to select the best approach. This gave us great insights about financial analysis along with financial predictions.

In future we would like to add other financial factors like trading signals, investor sentiment, news abstracts etc. to the model input variables in order to further the generalization and reduce overfitting in model. This way we can increase the amount of effective training data provided to the model without having to add previous year stock prices which tend to have lesser impact on current trends. Also we did not consider real-world constraints like transaction costs which should also be included when making trading decisions. We should try to simulate the trading systems as close to the real-world as possible.

All this was as intimidating and untried for us as it was contemporary. This lead to our understanding of Machine Learning and Data Science to increase by leaps and strides. We

also got acquainted with the financial domain of knowledge and expanded our knowledge to new fields. Along with that we got the real time experience of working in an extensive unaccustomed and highly creative Research project. We have hundreds of improvements and additions that we would like to further add to this project in spite of the hundreds we already have but since we were able to get through the basics and fundamentals in a good way through this project, we're immensely motivated to keep learning and gaining more knowledge in this feild.

7 References

1) https://www.researchgate.net/publication/326268581_ModAugNet_A_new_forecasting_framework_for_stock_market_index_value_with_an_overfitting_prevention_LSTM_module_and_a_prediction_LSTM_module

2) https://www.researchgate.net/publication/321630147_Deep_learning_with_long_short-term_memory_networks_for_financial_market_predictions

3) https://www.researchgate.net/publication/273640320_LSTM_A_search_space_odyssey

4) https://www.datacamp.com/community/tutorials/finance-python-trading

5) https://www.kaggle.com/benjibb/lstm-stock-prediction-20170507

6) http://global.krx.co.kr/contents/GLB/05/0502/0502030101/GLB0502030101.jsp?idxCd=1028#_1be3686ec5cb98595cf6179ba66d7a8b=3

7) https://heartbeat.fritz.ai/introduction-to-matplotlib-data-visualization-in-python-d9143287ae39

8) https://www.researchgate.net/publication/327967988_Predicting_Stock_Prices_Using_LSTM