

Analysis of financial data on the time series using data
from the stock market

by

Ifad Bhuiyan Shachcha
17201120

Muhammad Ziaus Siam
17201027 and 21341055

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Ifad Bhuiyan Shachcha
17201120



Muhammad Ziaus Siam
17201027 and 21341055

Scanned with CamScanner

Approval

The project titled “Analysis of financial data on the time series using data from the stock market” submitted by

1. Ifad Bhuiyan Shachcha(17201120)
2. Muhammad Ziaus Siam(17201027 and 21341055)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 24, 2022.

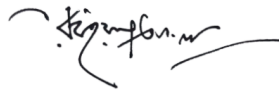
Examining Committee:

Supervisor:
(Member)



Annajiat Alim Rasel
Sr. Lecturer and Undergraduate Coordinator
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Rubayat Ahmed Khan
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:

Md. Golam Rabiul Alam
Assistant Professor

Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Predicting financial data is really important for investors. Often times investors do not have a proper tool to properly assess the market and forecast their predictions. Furthermore, not only investors in modern day civilians are also willing to invest as well and as there is an abundant amount of data available from the financial sector it is of utmost significance to find the optimal algorithm in a general case scenario. This project aims to show a comparison between the results found from some of the popular neural network algorithms. In this project we have employed the help of Dense Neural Network [DNN], Recurrent Neural Network [RNN], Long Short Term Memory unit [LSTM], Convolutional Neural Network [CNN] and a pipeline where we combined LSTM and CNN. We have kept some of the parameters similar and compared the results to determine an algorithm in a general case. This would help people take informed decisions while investing.

Keywords: Stock market; Machine Learning; Finance; Prediction; Dense NN; RNN; LSTM; CNN;

Acknowledgement

Before all we would like to Thank The Almighty Allah to allow us to complete our project successfully without any mishaps. Then, we would like to thank our supervisor Mr. Annajiat Alim Rasel for guiding us throughout the entire process and finally we would like to show our heartfelt gratitude to our parents who made our dreams come true.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	x
Nomenclature	xii
1 Introduction	1
1.1 Thoughts behind the Prediction Model	1
1.2 Research objective	1
2 Related Work	3
3 Models	5
3.1 Time series	5
3.1.1 Time series analysis	5
3.2 Neural Network	11
3.2.1 Classification	11
3.2.2 Clustering	11
3.2.3 Neural network elements	11
3.2.4 Deep Neural Network	12
3.3 Dense Neural Network(DNN)	14
3.3.1 Dense layer	14
3.4 Feedforward network	15
3.5 Recurrent Neural Network	16
3.6 Long Short Term Memory units(LSTM)	18
3.7 Convolutional Neural Network	21
4 Methodologies	23
4.1 Methodologies	23
4.2 Model Structures	24

5	Dataset	26
5.1	Source of dataset	26
5.2	Pre-processing	28
6	Result	29
6.1	Dense Neutral Network	29
6.2	Recurrent Neural Network	31
6.3	LSTM	32
6.4	CNN	34
6.5	Pipeline	35
7	Conclusion	38
	Bibliography	40

List of Figures

3.1	Time series upward trend	6
3.2	Seasonality of data	6
3.3	Trend Seasonality	7
3.4	Noise in time series	8
3.5	Noise added to trend and seasonality	8
3.6	Auto correlation in time series	9
3.7	Auto correlation with noise, trend seasonality	10
3.8	Non stationary data	10
3.9	Neural Network structure	12
3.10	Neural network layers	12
3.11	Deep neural network	14
3.12	Dense neural network	15
3.13	Feedforward NN	16
3.14	Basic Recurrent Neural Network structure	17
3.15	Recurrent Neural Network internal workings	18
3.16	LSTM	19
3.17	simple recurrent neural network vs LSTM	20
3.18	Workflow of gates	21
3.19	Convolution layer	22
3.20	Pooling layer	22
3.21	Full sequence of CNN including the flatten and dense layers	22
4.1	Learning rate tuning	24
4.2	Loss after tuning	24
5.1	SP500 index	26
5.2	First 5 elements of the dataset	27
5.3	Dataset statistics	28
6.1	Dense neural network learning rate tuning	29
6.2	Dense neural network loss after tuning	30
6.3	Dense neural network prediction	30
6.4	Recurrent neural network learning rate tuning	31
6.5	Recurrent neural network loss after tuning	31
6.6	Recurrent neural network prediction	32
6.7	LSTM neural network learning rate tuning	32
6.8	LSTM neural network loss after tuning	33
6.9	LSTM neural network prediction	33
6.10	CNN neural network learning rate tuning	34

6.11 CNN neural network loss after tuning	34
6.12 CNN neural network prediction	35
6.13 Pipeline neural network learning rate tuning	35
6.14 Pipeline neural network loss after tuning	36
6.15 Pipeline neural network prediction	36

List of Tables

4.1	Control parameters for our experiment	23
6.1	MSE and MAE Comparisons of all the models	37

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ϵ Epsilon

v Upsilon

A – VAR Advanced Vector Auto Regression

ABC Artificial Bee Colony

ADF Augmented Dicky-Puller

AIC Akaike information criterion

AR Auto Regression

ARIMA Auto Regressive Integrated Moving Average

ARMA Auto Regressive Moving Average

AWSVR Adaptively weighted Support Vector Regression

BPNN Back Propagation Neural Network

CIDM Computational Intelligence and Data Mining

CNN Convolutional Neural Network

Co – BPNN Connective BPNN

DMAM Double Moving Average Mapping

ELM Extreme Learning Machine

ETS Error Trend and Seasonality

EWT Empirical Wavelet Transform

FFNN Feed Forward Neural Network

GAF Gramian Angular Field

GRU Gated Recurrent Unit

LSTM Long Short-Term Memory

MA Moving Average
MAE Mean Absolute Error
MAM Mean Average Mapping
MLP Multi-Layer Perceptron
MSE Mean Squared Error
RMSE Root Mean Squared Error
RNN Recurrent Neural Network
STL Seasonal Trend Loss
SVM Support Vector Machine
SVR Support Vector Regression

Chapter 1

Introduction

1.1 Thoughts behind the Prediction Model

In this era of information and technology where we are surrounded by vast amount of data. It is paramount to be able to accurately predict events. Reliable and accurate predictions require appropriate techniques and relevant data. Prediction of financial data requires real time data to make an informed decision so that the investor can invest their money. Certain mathematical techniques and Machine Learning Algorithms can make these predictions more efficient and reliable. It is becoming more and more important for organizations to keep track of their transactions and assets. As it will allow them to make future predictions so that proper administrative decisions can be made to adjust. Sets of information related to the financial health of a business is known as financial data. Forecasting financial data is typically performed annually or bi-annually or quarterly. So usually, financial data involves changes across a certain period of time which means financial data are often involved in time series analysis.

Furthermore, as in stock market data the preceding price has much influence over the present one due to economical constraints it would be much beneficial for the parties connected for it to be predicted. In addition to, as with weather forecasting people take precautions based on the prediction in the same way people can do the same with the results that are found from time series stock prediction. As people can take more informed decisions and in a way increasing the economic growth as a whole.

1.2 Research objective

Data forecasting or prediction has become an essential process for academic institutions and business institutions alike. In this fiercely competitive era being able to simulate the effects of current events and predict future events is crucial to having a competitive edge. For businesses this boils down to actively monitoring and recording transactions and assets and then analyzing them to find more efficient methods or to spot discrepancies which might cause problems later down the line [7]. For businesses having proper funds is essential to run smoothly. However, it is not always possible to have enough capital to cover the amount of investment needed

for certain projects and endeavors. This is when companies release shares for their companies. Selling these shares gives companies the funds needed to cover their costs. On the side of investors, they essentially get to have a claim on a portion of the assets of the companies whose shares they hold. Investors may sell their shares back into the market for profit by buying shares when they are low priced and selling them when they are higher priced. Investors also may wish to buy more shares to get a larger portion of dividends from the company. Companies on the other hand have to analyze their stock index history as well as that of their competitors to ascertain the market environment and make decisions.

Be it investors or companies both need to collect and analyze the relevant stock indexes and make predictions. These predictions or forecasting can be done using various mathematical techniques. Depending on the data and its internal distribution some techniques might perform better than the others. Ideally, we would prefer data which have predictable patterns that can then be identified and quantified. This would allow us to use simple linear regression and classification to make accurate predictions. However, real life data is almost always spontaneous and to a certain extent unpredictable. Various factors may influence prices to fluctuate and it is not always clear as to what the cause is or to what extent the effects are. So, we employ different architectures neural networks in some general settings to have a comparisons between them for some insight.

Chapter 2

Related Work

In order to have a better forecasting model we must understand the data better. Thus in the case of stationary data Sun and Xu[8] compared different approaches for forecasting financial data which started by classifying the methods into two groups: Economic analysis-based methods and Data mining-based methods which aimed to compare methods from both categories in light of forecasting financial data. They focused on stationary time series and makes use of algorithms from the two fields. From the economic methods the ARIMA and Advanced VAR (A-VAR) models are chosen and from the data mining field the BPNN, Co-BPNN and RNN models are chosen. The results showed that the A-VAR model has lower RMSE than the ARIMA model with the same order of runtime performance. In the case of the BPNN model it is similar to the ARIMA model but has a longer runtime. The Co-BPNN performs better than the previous methods however it is slower than BPNN by one order of magnitude. The LSTM model performs the best by far while also having the longest runtime. Yu, Jingming, Shumei [11] and Shuping in their paper attempted to present a hybrid model which would work well with both stationary and non-stationary data [2]. They did this by combining the EWT, ABC, ELM and ARIMA models. They first used the EWT model to decompose and denoise the data in order to eliminate the impact of outliers. This makes the data more suitable to be handled by the ARIMA and the ELM model. Then the data is passed onto the two models and the output is generated. The ABC algorithm is used as a swarm intelligence optimizer to handle under-fitted solutions. The Outputs are then passed through a weighted sum processor where the results of both the ARIMA and ELM models are combined. Sai and Sreela[7] have aimed to present a reliable method to predict the fluctuations in stocks caused by seasonal changes. Sai and Sreela had structured their architecture into 3 main components. First comes the pre-processing of data. This involves organizing and sorting the data and then screening, editing and removing erroneous data. Once that is done the data is then classified using Random Forest algorithms. For the comparison experiments 3 algorithms that have the best average training and holdout performance were selected. ID3, C4.5 and CART were chosen for this. Finally, the trend and seasonality for the models are analyzed. Their results showed that about 30% of the data showed signs of following seasonal trends and are good places where investors are more likely to find profits. The remaining 70% being bad stock where even though there are seasonal variations, they are not easily predictable. Jou-Fan and Chen have developed a deep learning framework based on convolutional neural networks (CNN)

to analyze financial time series data[3].They developed two methods for this, the Mean Average Mapping(MAM) and the Double Moving Average Mapping(DMAM) method to transform the time series data into 2D images and also used the Gramian Angular Field (GAF) algorithm. The results showed significant improvement from the expected performance which suggests that this approach allows visualization of patterns and correlations that are not apparent by typical methods. Furthermore, Duraj Agnieszka and Ludwicka Magdalena are detecting the outliers in the financial time series using ARIMA [6].The authors stated that it was visible as an effect of variance grouping, which can indicate necessity to use models that take into account the heterogeneity of the variance over time. Almuammar have proposed a method that uses Gated Recurrent Units (GRU) to forecast multivariate time series and compare it with the simple Multi-Layer Perceptron (MLP) [9]. As GRU models can retain long-term information and perform well with long-term dependencies. The author concluded that even though the results suggest that GRU perform better than statistical methods they are out performed by MLPs in regards to time series forecasting.Tiwari, Bharadwaj and Gupta aimed to apply various analysis techniques and models to compare and contrast between them and determine which approach is most suited for stock price index forecasting[5].It aimed to find the optimum approach to implement the “Buy low, sell high” strategy allowing investors to buy stocks at low prices and sell them at higher prices.The authors had used a multi-layered perceptron and feed forward network with both having sequential models.The FFNN performed the best with the least amount of absolute percentage error with actual data. With seasonal data the Holt-Winters performed best. The fixed parameter ARIMA model did best with polynomial trend data.Furthermore, Zhijie Li1, Yuanxiang Li1, Fei Yu1, Dahai Ge1 aiming to show how AWSVR outperforms the SVR[1]. The result of the experiment shows that the MSE value of the data sets of NASDAQ, SP and FTSE is smaller in AWSVR than in SVR. Fang Wang, Menggang Li b, Yiduo Mei e, and Wenrui Li proposed a method of forecasting using the historical rise and fall probability distribution curve [10]. They used ADF (Augmented Dicky-Puller) to check whether the time series is stable. After that the ARMA model is used which is a combination of AR and MA model. XiangRu Guo, predicted future cash flow based on historical purchases and redemption data to help companies to improve its fund management ability while also minimizing liquidity risks[2].They used ARMA and ADF.Yang yujun, yang yime, Li jianping aimed to study on financial time series forecasting based on the support vector machine by conducting experiments with datasets[4].

Chapter 3

Models

3.1 Time series

Time series is a collection of observations of a particular data point taken at fixed intervals across a period of time. It follows the movement along the selected points of data, such as price of stocks, weather indexes over a specific period of time while this data points being collected at periodic intervals. As there is no fixed set period of time, it allows the data to be gathered in a way that enables the collector to reach their set goal. Time series varies from different amount of time sequences ranging from monthly, trimesters, annual along with weekly, daily, hourly and even biannual or decennial.

3.1.1 Time series analysis

The process of studying a collection of data points over a specific period of time is called time series analysis. Time series analysis involves capturing data at constant intervals over a predetermined length of time instead of collection data randomly or infrequently. Time series data is different from other types of data because it can depict how data points change over time. The main variable that distinguishes time series data from others is the time variable which further adds on to the information that is available to further establish the correlation between other features or variables. A time series can be split into four patterns/components.

In addition, this component expresses a different feature of the evolution of the time series values. Real life time series data typically consists of a combination of one or more of these patterns. These patterns are:

Trend:

Being one of the main components of the time series, it represents the value to go up or down as time progresses. As given a certain amount of time we could see how the graph progresses. An example of with upwards trend is given below:

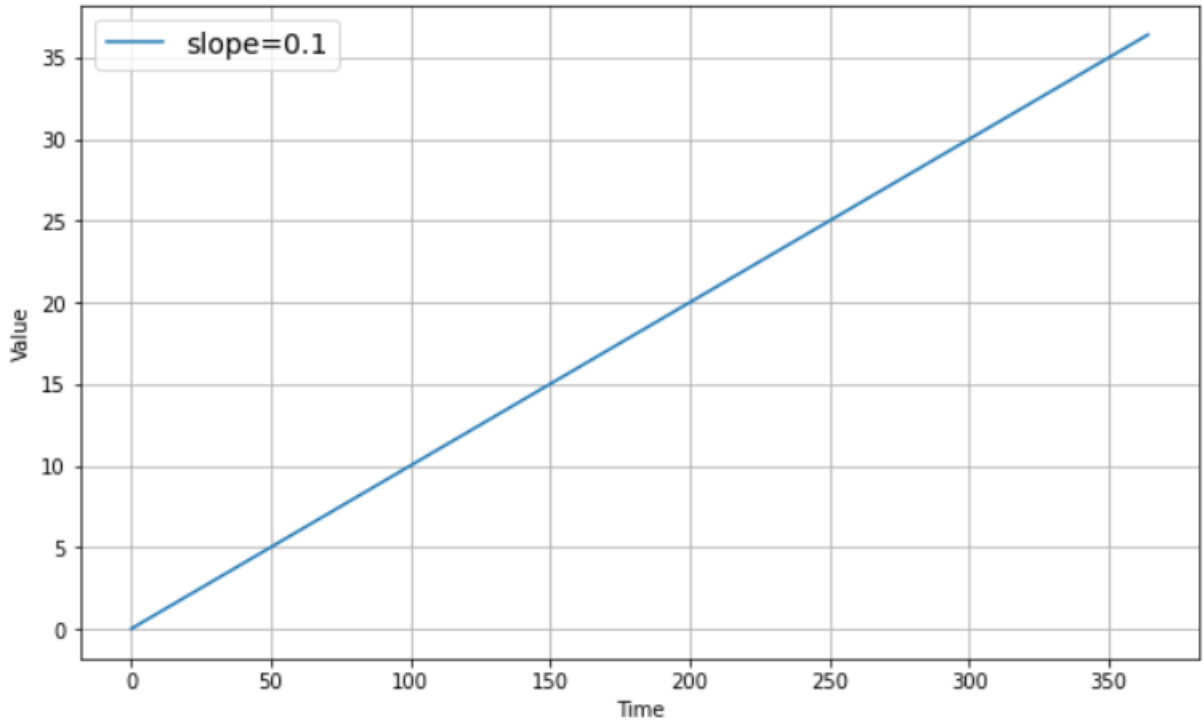


Figure 3.1: Time series upward trend

Seasonality:

Seasonality refers to fluctuations that is constant in a specific time interval

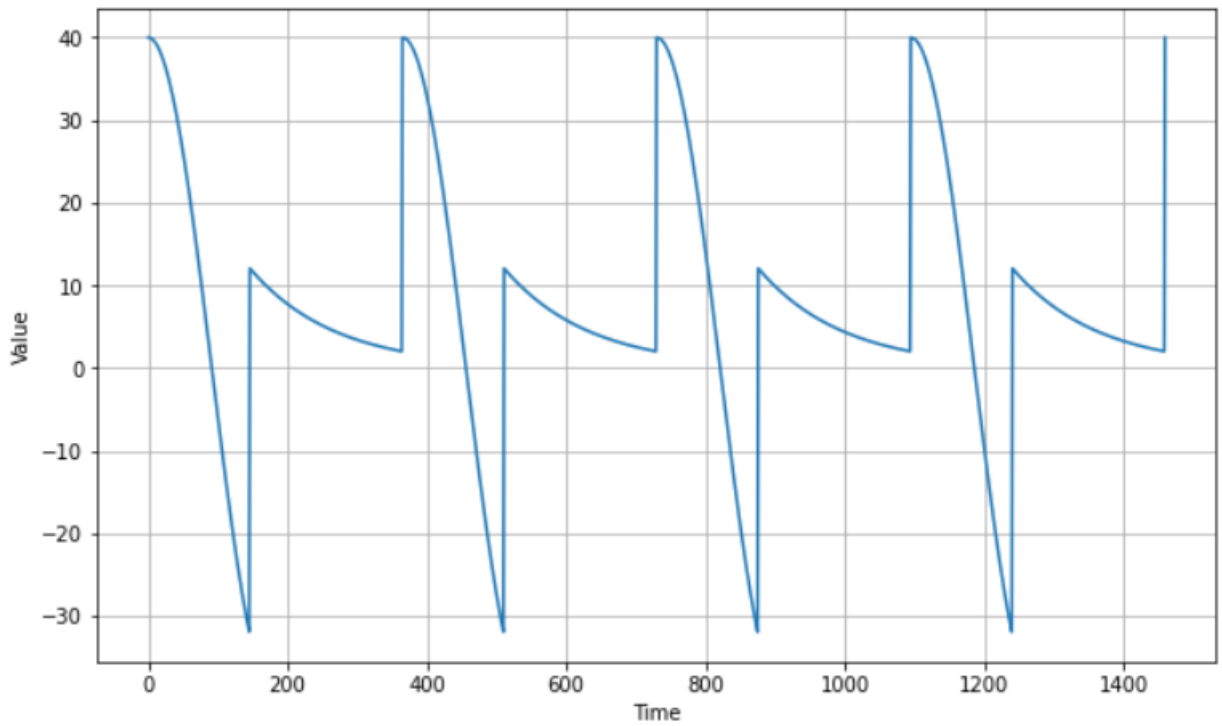


Figure 3.2: Seasonality of data

In the below diagram we can see an example of seasonality with trend.

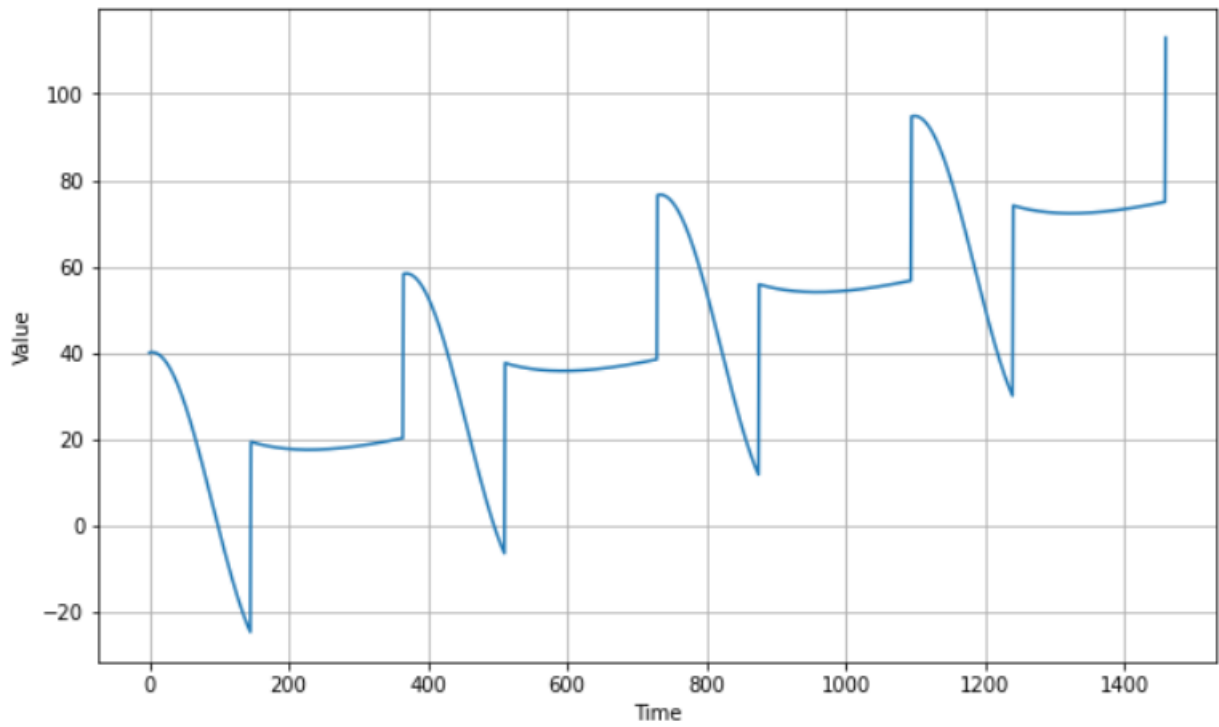


Figure 3.3: Trend Seasonality

Noise: In real life data such smooth data is hardly found. As a result these data usually has some noise riding over it.

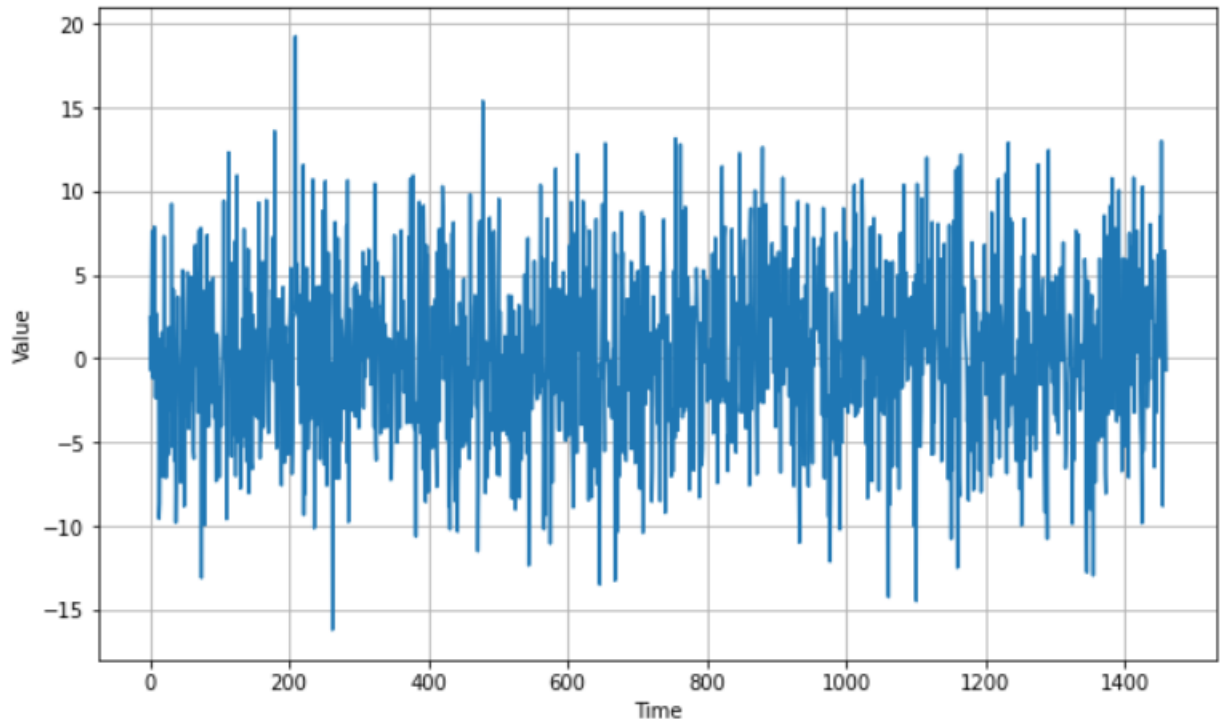


Figure 3.4: Noise in time series

Now if we add this to the figure 3.3 it would yield the below diagram.

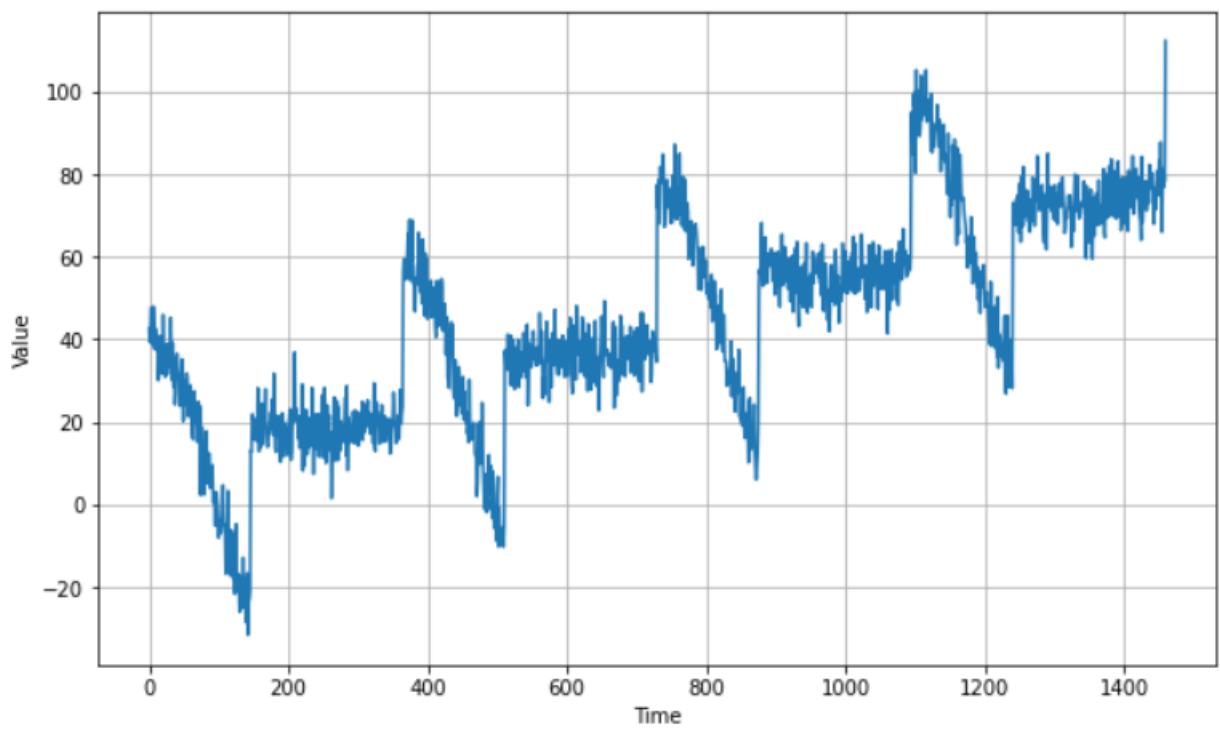


Figure 3.5: Noise added to trend and seasonality

Auto correlation:

Auto correlation which means a measurement of a time step is a derivative of a previous time step. This phenomenon can occur in time series. Below a straight-forward auto correlation is shown which just calculates the value from its preceding time step.

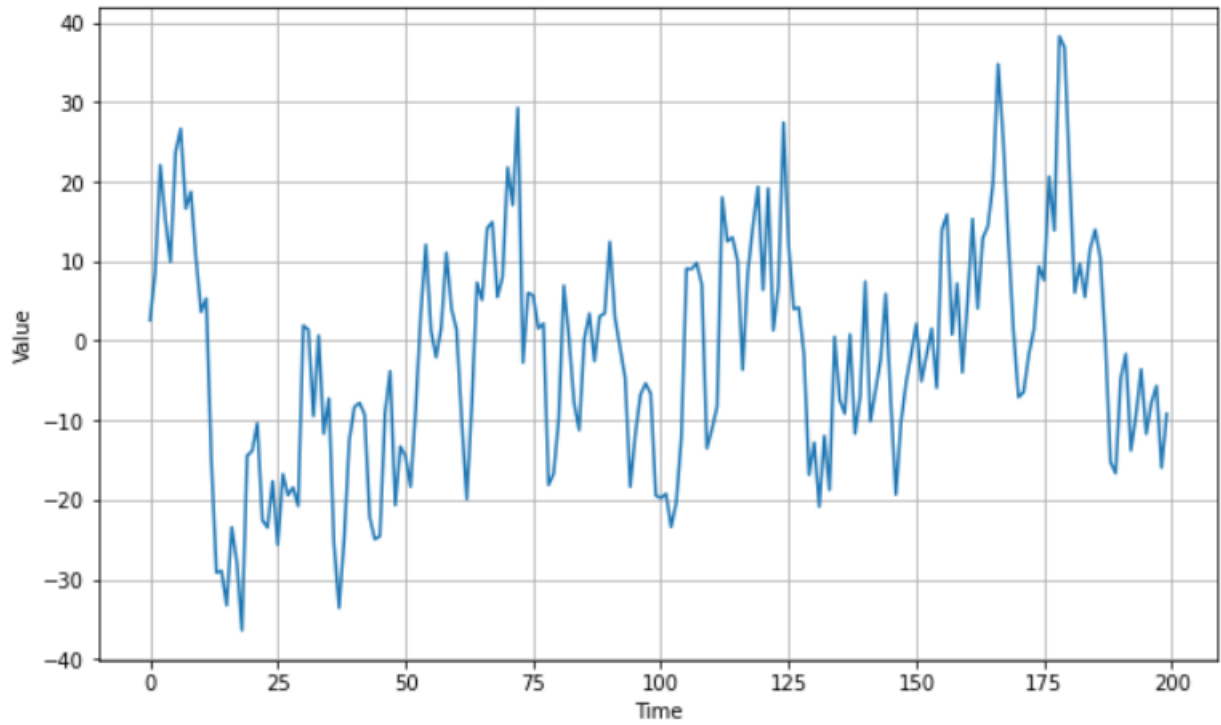


Figure 3.6: Auto correlation in time series

The below diagram adds auto correlation to figure 3.4

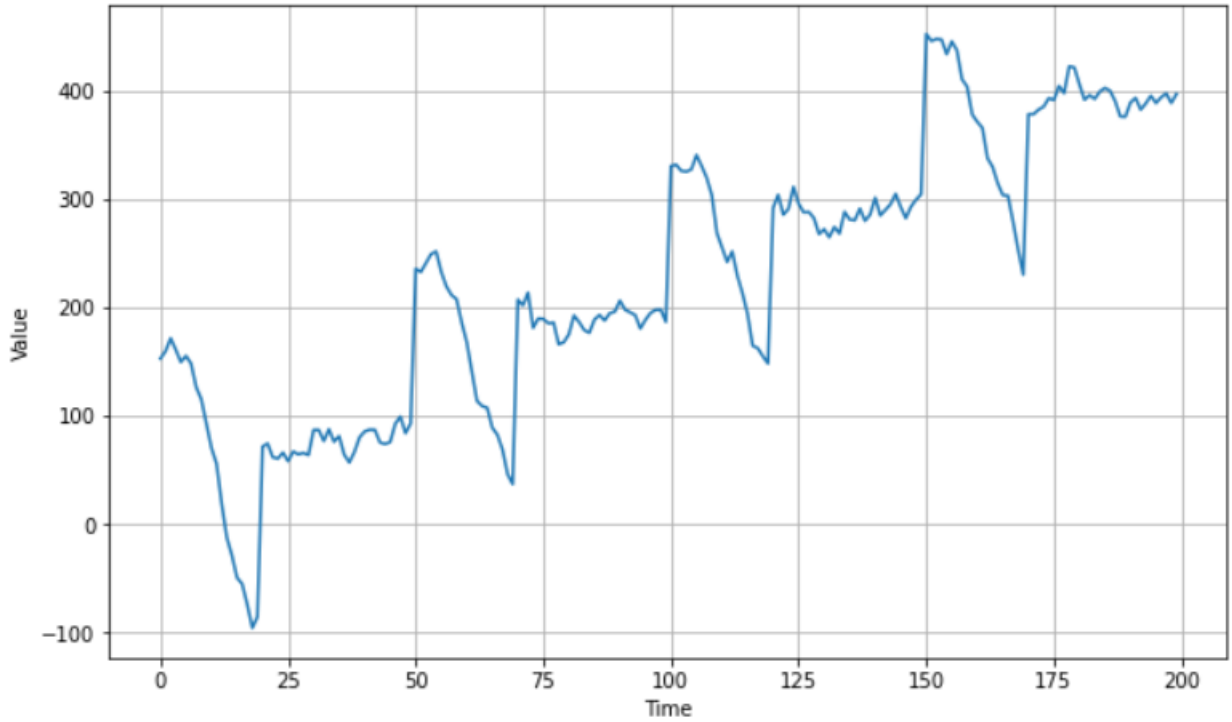


Figure 3.7: Auto correlation with noise, trend seasonality

Non stationary data:

Real life data doesn't always follow a certain pattern. Big events can alter the traversal line. It might decline or go upwards at points.

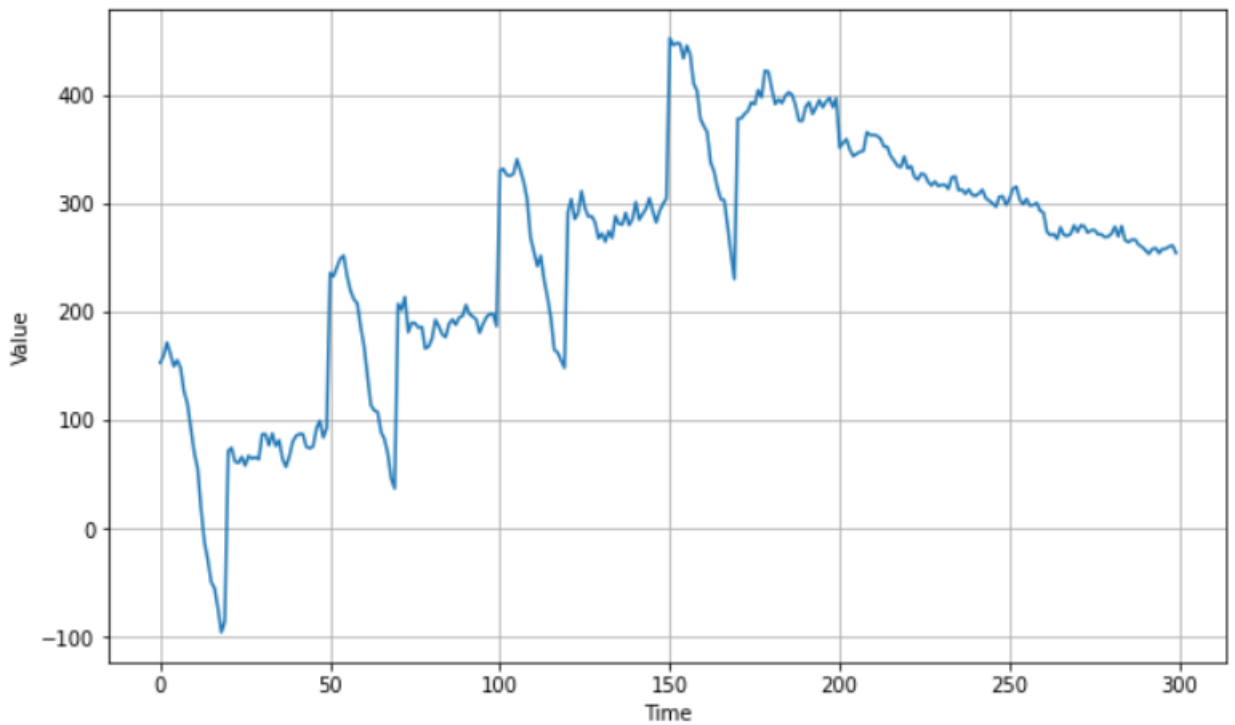


Figure 3.8: Non stationary data

all the examples above are given to have a better understanding of graphs in time series and to show attributes of it.

3.2 Neural Network

A collection of algorithms used to identify patterns which are roughly based after the human brain. By using machine perception, it categorizes or groups raw input and this process is used by the neural networks to analyze sensory data. Neural networks recognize numerical patterns which are enshrined in vectors. Thus, all real-world data weather images, sound, text or time series must be converted to it. Warren S. McCulloch and Walter Pitts conducted a study which tries to comprehend how complex patterns via linked brain cells or neurons might generate from the human brain. The comparison of neurons with a binary threshold to Boolean logic was one of the main ideas. We might use neural networks to cluster and categorize data.

3.2.1 Classification

Labelled dataset is the key component for all the classification task. As any real-life problem first must be transformed into a dataset for the neural network to learn the association between labels and data. Supervised learning refers to this process. This classification ranges from face detection, expression detection, recognizing gestures in a video, detecting voice or identifying speakers, to categorize email as spam to finding patterns in time series. In short, it is possible to train a neural network with a labelled dataset and when the desired outputs correlates to that dataset.

3.2.2 Clustering

Firstly, in contrast with the supervised leaning learning without labels is called unsupervised learning. Now clustering refers to detecting similarities and as in deep learning labels aren't required to detect similarities it is much easier to find data. Because in the real-world most data are unlabeled.

3.2.3 Neural network elements

Networks composed of several layers are known as “stacked neural networks”. Layers are made of nodes and these nodes are just location where computing occurs. This schematic is roughly modelled after neurons that are situated in the human brain.

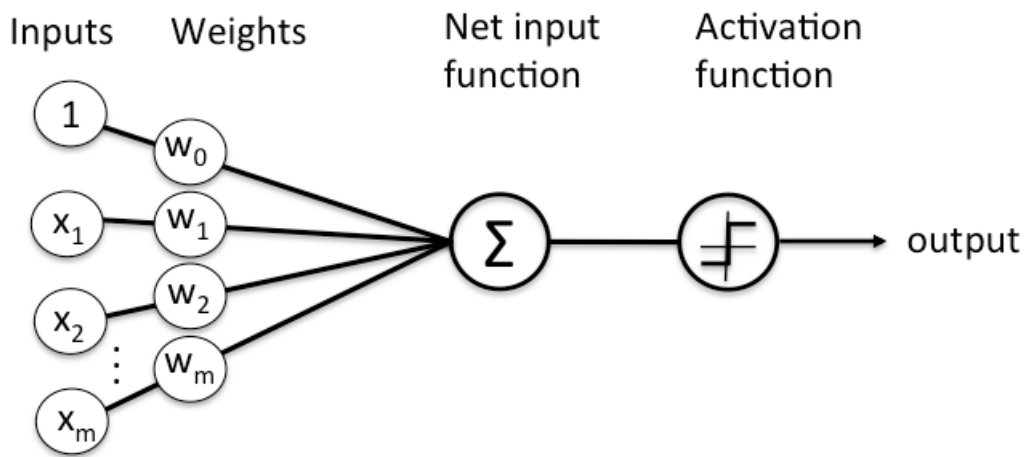


Figure 3.9: Neural Network structure

As data flows through the network, a layer of neurons get activated and deactivated. The first layer takes in the input data processes it and then sends an output to the following layer. The models configurable weights and biases are then associated with our input data and output labels.

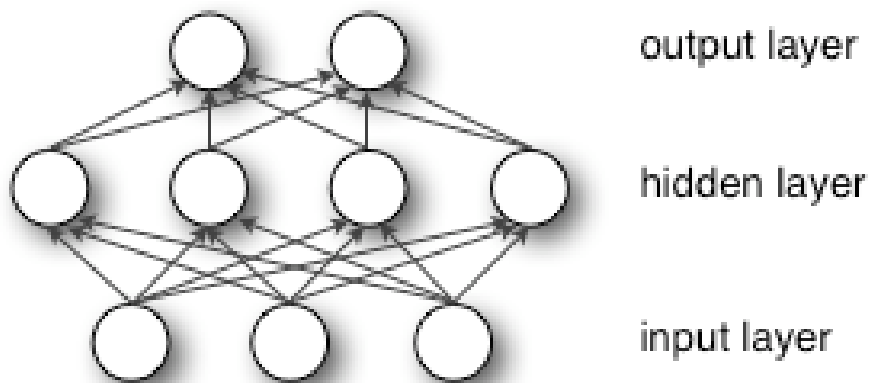


Figure 3.10: Neural network layers

3.2.4 Deep Neural Network

What differentiates deep neural networks from single hidden layer networks is the number of layers that the information need to go through during the propagation along the network. The earliest version of the neural networks were perceptrons which were shallow since they consisted of one input layer, one output layer and atmost one hidden layer in between them. A deep neural network must have atleast four layers including the input and output layers. That is to say they must have atleast two hidden layers. Each layer in a deep learning neural network analyzes various types of pattern depending on the input from the layer before it. Since neurons accumulate and merge the information from the layer before it, the deeper we go into the network the more complex the features that our nodes can detect.

This phenomenon is called feature hierarchy or a rising complexity and abstraction hierarchy. This means that deep learning networks can handle extremely large non linear datasets with high dimensionality and billions of parameters.

The vast majority of the data in the world are unlabeled or unsaturated data or raw media and neural networks have the capability to detect latent patterns within them. Neural networks as a result are widely used in handling images, video and text. In case of time series data, the data may be structured to highlight normal behaviour or abnormal behaviour. Deep neural networks do not need human intervention and can extract features and patterns from data automatically in the hidden layers unlike other more traditional machine learning algorithms. Deep learning allows for extracting complex features easily that would otherwise require a team of specialised data scientists years to achieve. While learning from unlabelled data the nodes in the hidden layers automatically extract features and associate them with the input data from which they created their samples. The network constantly tries to minimize the difference between its predictions and that of the input data.

This process allows deep neural networks to learn about the correlations between particular aspects. That is to say they associate the distinct patterns in the data with their labels. The fact that deep learning networks can process large amounts of unlabelled data and learn from them give it an edge over prior algorithms. The output layer assigns the likelihood to a certain even or label and is also terminates the network.

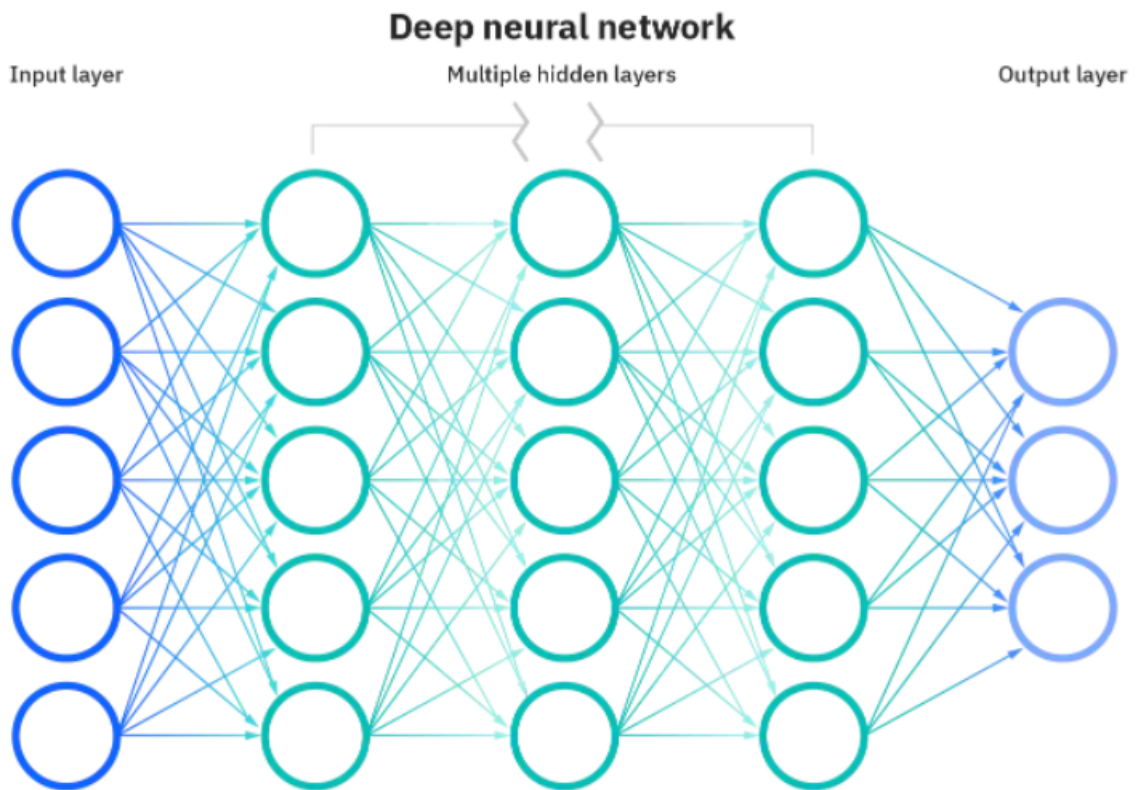


Figure 3.11: Deep neural network

3.3 Dense Neural Network(DNN)

The architecture of a deep learning model may be thought of as its layers. Different sorts of layers can be utilized in the models. A layer that is fully connected is referred to as a dense layer which is usually used in the final stages of the neural network. In order to establish the relationship between the values of the data the network is working with, this layer aids in adjusting the dimensionality of the output from the previous layer.

3.3.1 Dense layer

A dense layer is a layer which is intimately coupled with its previous layer which implies that the every neuron of its preceding layer is connected to the neurons of that layer. It is considered to be the most popular layer in artificial neural network.

In a dense layer, each neuron receives input from every neuron in the preceding layer. The neurons then perform matrix multiplication where the size of the row vector of the previous layers output is equal to the size of the input column vector of the neuron.

The structure of a dense neural network is given below

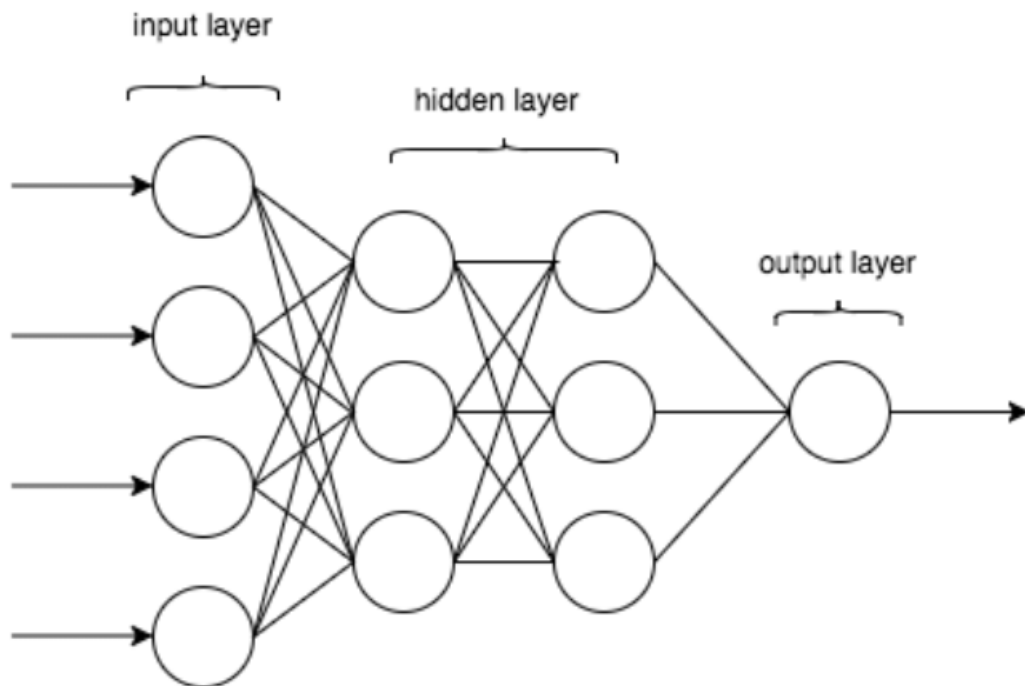


Figure 3.12: Dense neural network

3.4 Feedforward network

In feedforward networks, supervised learning is used to turn input samples into outputs. The output is a label which is associated to the input data that was used to get it. That is to say, raw data is mapped to the extracted features and patterns. The network classifies data uses a set of parameters that are trained for classifying the input data.

Furthermore, a feedforward network doesn't have any concept of temporal order and the only input it takes into account is the most recent one. Below a basic feedforward network diagram is given:

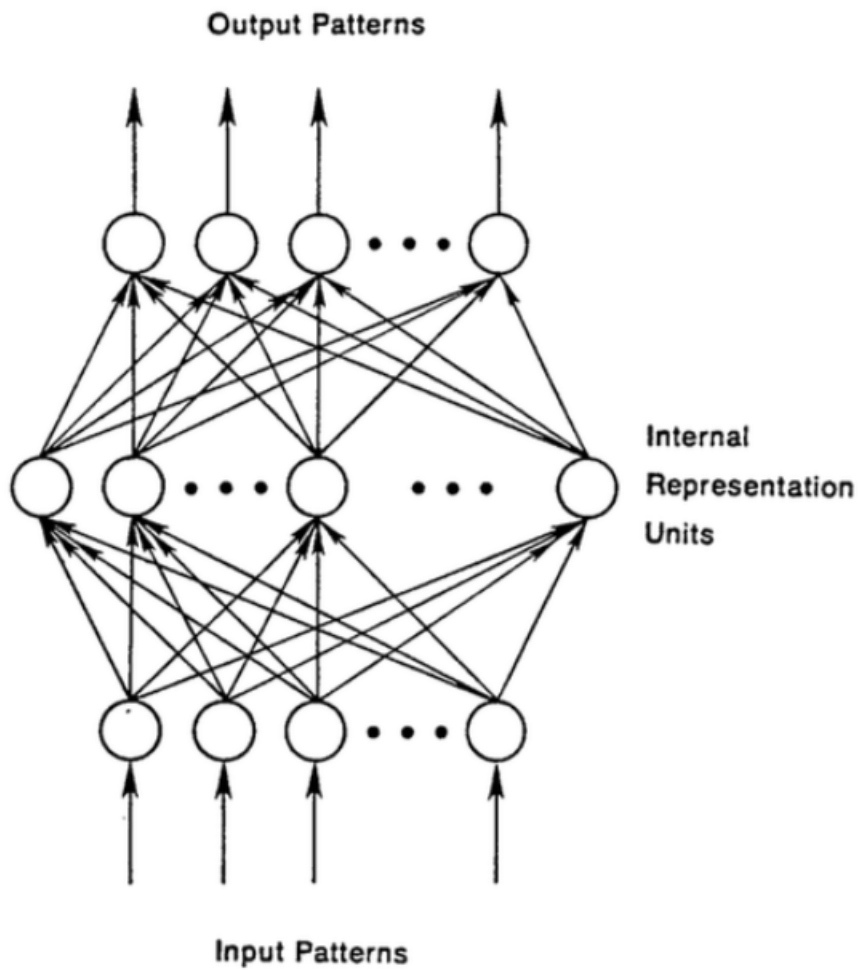


Figure 3.13: Feedforward NN

3.5 Recurrent Neural Network

Unlike the feedforward network RNN doesn't only use just the current input sample but also the previous inputs that has been exposed to them till that time. The diagram below explains this phenomenon.

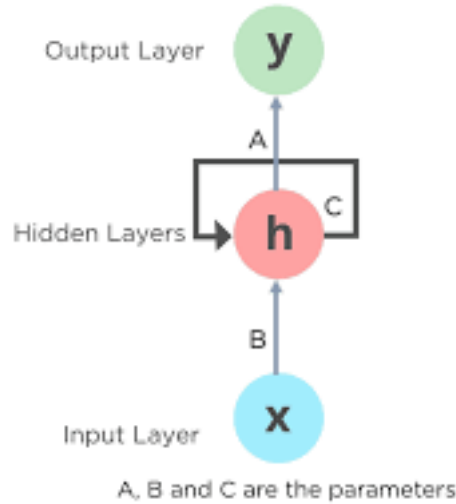


Figure 3.14: Basic Recurrent Neural Network structure

The result a recurrent net reaches at time step t is actually derivative from the result that it reaches at time step $t-1$. Thus, recurrent networks has two flows of inputs one of which is the recent past and the other one is the present which then merge to influence to its response to incoming data which kind of mimic real life human interaction.

Recurrent Neural networks have a feedback loop which connects to the previous output which is where it is different from feedforward networks. That is to say, the outputs are fed back into the recurrent layer. This means Recurrent Neural networks retain information or memory of past iterations. This allows it to perform predictions that Feedforward networks are unable to match. This sequential information is preserved through multiple time steps affecting each subsequent sample. Sequential data like time series data have long term dependencies which means related events maybe separated by several time steps.

$$h_t = \phi(Wx_t + Uh_{t-1}) \quad (3.1)$$

h_t is the hidden state at time step t which is also a function of the input at the same time step x_t altered by a weight matrix W . Furthermore, it was then added with the hidden state to hidden state matrix U which is also known as the transition matrix multiplied by the hidden state of the previous time step h_{t-1} . The amount of weight that is to be given to the current input as well as the preceding hidden state is decide by the weight matrices. Finally, the error which is produced will be sent back via backpropagation and it is used to alter their weighs until the error can no longer be reduced. The function ϕ is usually a logistic sigmoid or tanh function. Each hidden state has traces of all the hidden states before it as long as the corresponding memory is still available.

Below is given a Recurrent Neural Network data flow:

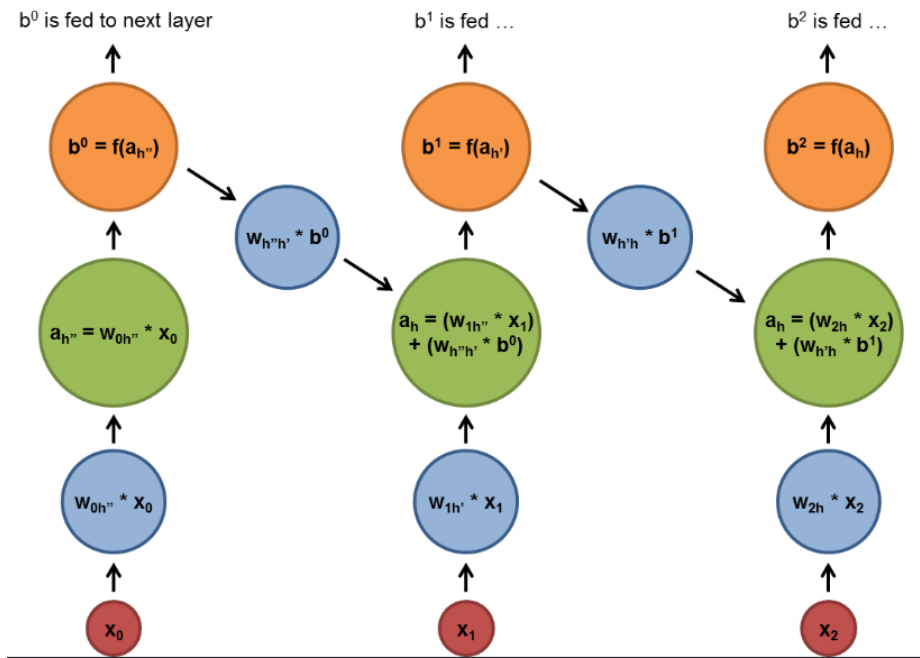


Figure 3.15: Recurrent Neural Network internal workings

3.6 Long Short Term Memory units(LSTM)

In the mid-90s, Hochreiter and Schmidhuber presented a variant of the recurrent neural network model. They called it Long Shot-Term Memory units or LSTMs. This variation was better at preserving errors that can be backpropagated through time and layers. LSTMs use a gated cell to store information separately from the main flow of the network. The gates of this cell determine which information to retain or remove. These analog gates are implemented by performing element wise multiplication of the data with sigmoid functions which produce outputs in the range of 0 to 1. Since the gates are analog they are more suitable than digital gates for backpropagation because they are differentiable.

The gates operate on the data that is passed onto them just like nodes in a neural network. They have their own weights and biases that get tuned alongside the nodes in the main neural network during the learning process. The weights and biases get tuned each iteration of the backpropagation. The weights get adjusted via gradient descent based on the backpropagating errors. Over time they learn to select the appropriate information to store, erase or pass onto the main network.

The diagram below shows data flow through memory cell and the control over them by the gates:

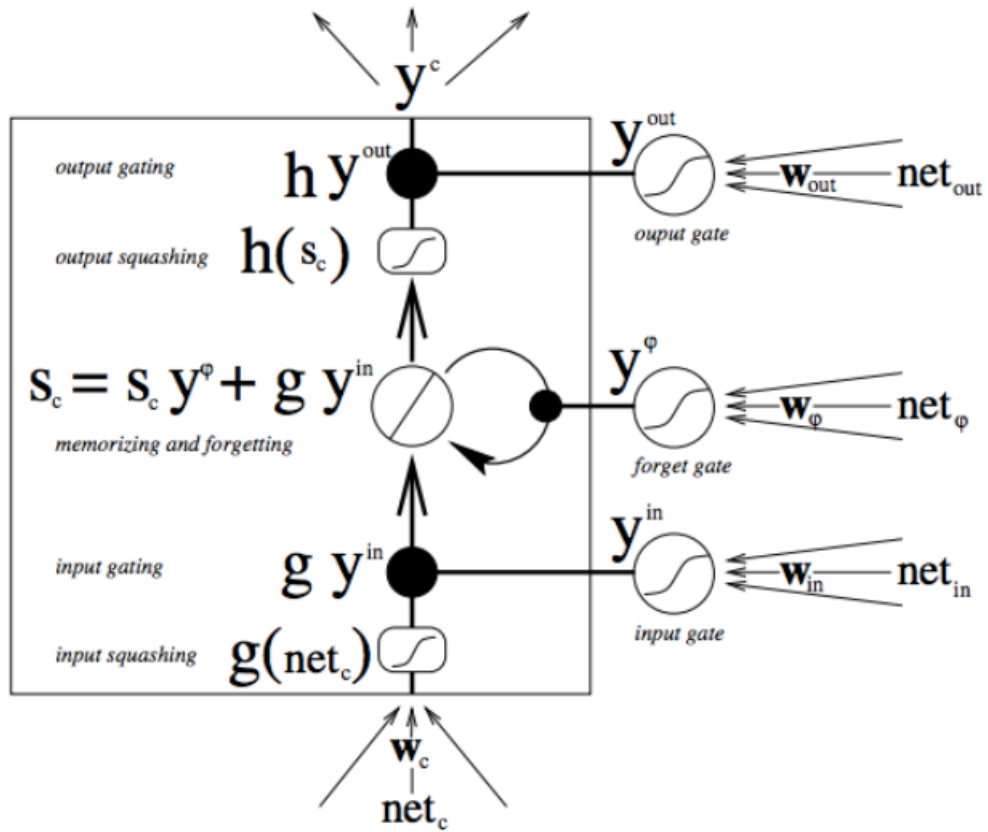


Figure 3.16: LSTM

The information enters the cell from the bottom. This information is also given to each of the three gates of the cell. The information gets passed through a sigmoid function in all four locations. The gates are represented by the black dots which then receive the transformed information and after processing it decide how the information is to be handled, whether the new input should be integrated with the current cell state or which portion of the current cell state should be erased or how much the current cell state should affect the output of the main neural network.

Here is another diagram for comparison with a simple recurrent neural network against a LSTM.

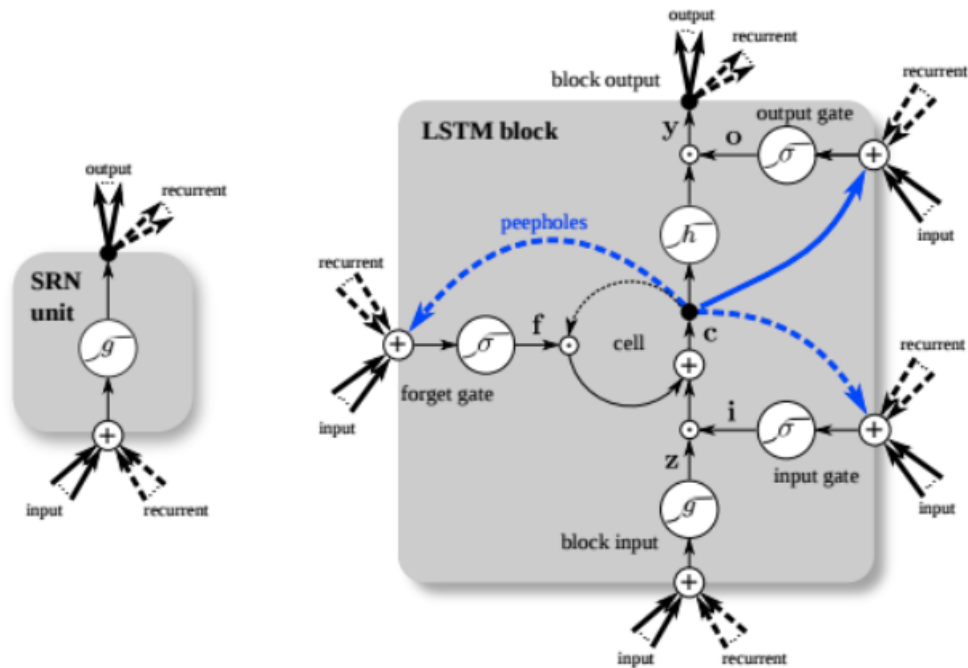


Figure 3.17: simple recurrent neural network vs LSTM

While transforming the input in LSTM memory cells, the addition and multiplication play different roles. The addition in the middle in both pictures are what makes LSTM so special. When the backpropagation is very deep, this adjustment helps to maintain a constant error despite how simple and straightforward it may seem. The difference lies in the fact that instead of just multiplying the current state to the incoming data to obtain the new cell state, the new data and old cell state are summed.

Filtering for input, output and forgetting are done by different sets of weights. Since when the forget gate is open, the current state of the memory cell is just multiplied by one and then propagated forward to the next time step, it is modelled as a linear identity function.

The gates are shown in action in the diagram below, with closed gates represented by straight lines and open gates represented by blank circles. The forget gates are the lines and circles that go horizontally along the hidden layer.

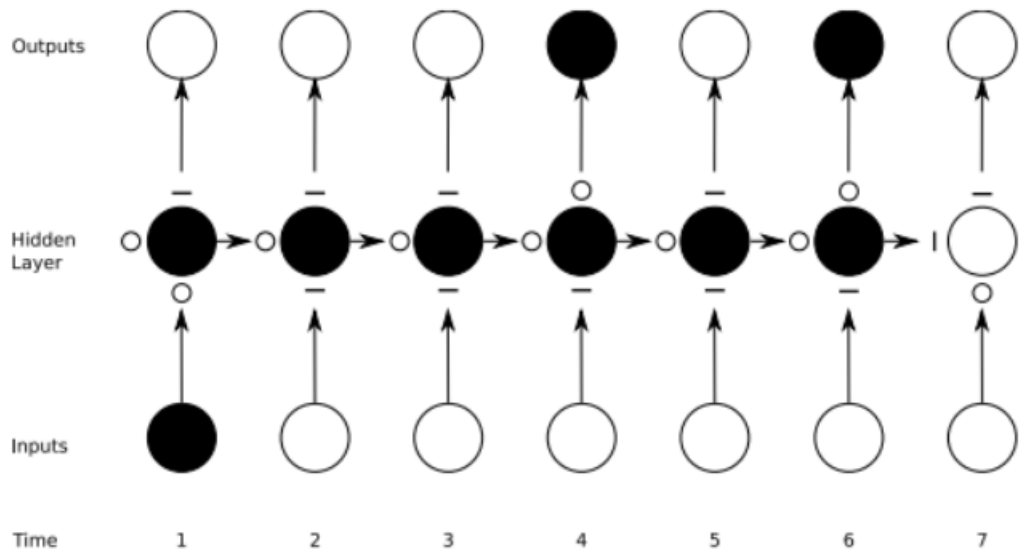


Figure 3.18: Workflow of gates

3.7 Convolutional Neural Network

Typically, CNNs are used to classify images. Images in CNN are represented as a matrix of pixel values with each pixel ranging from a value of 0 to 255. Images tend to have one or more channels representing the primary colors. Grayscale pictures have only one channel where colorful pictures have 3 for RED, GREEN, BLUE. An image is stored as a matrix of pixel values with shape [image width, image length, number of channels]. The primary layers responsible for performing the convolutions are the convolutional layer, the pooling layer and the dense layer.

The Convolutional layer extracts important distinguishing features from subsections of the image. This is done by filtering the subsections through a feature matrix. The feature matrix is multiplying with the subsections and the resultant matrix contains the extracted feature from the image. For example, a layer might learn to spot shoelaces from pictures of shoes. These features are then associated with the labels of the images. So, in our example whenever the network detects a shoelace it will try to associate the image with that of a shoe. An image may have multiple features extracted from it and we specify that number when we design the model. For time series data, we can apply the same concept. Time series data can simply be thought of as a one-dimensional matrix. Here too we can find patterns that may be extracted by a convolutional layer. Where for images we use a 2D convolutional layer, for time series we can simply use a 1D convolutional layer.

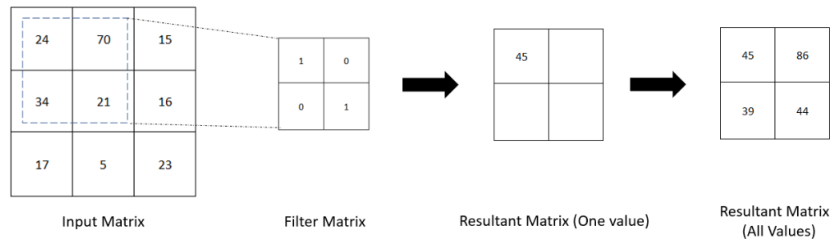


Figure 3.19: Convolution layer

A convolutional layer is typically followed by a pooling layer. Here the resultant matrix is multiplied with a pooling matrix which compresses the feature to consist of only its important features. This is done by selecting the average or maximum value from each small subsection of the data matrix. Image data requires 2D Pooling layers but for time series we can simply apply the same principals using 1D Pooling layers.

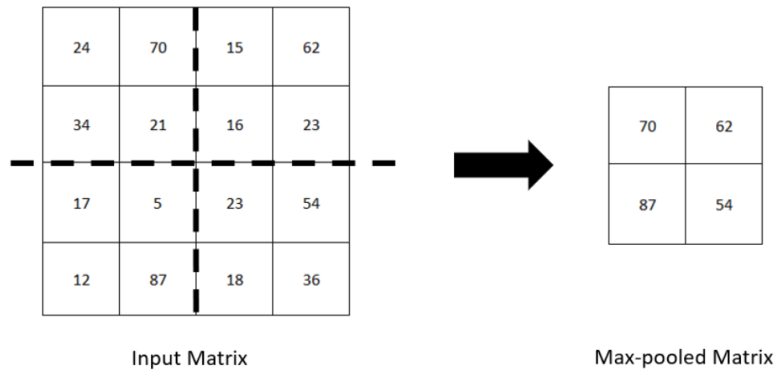


Figure 3.20: Pooling layer

After this comes a flattening layer followed by a dense layer which essentially converts our 1D time series data into a 3D matrix. Hence we can now apply the same concepts for our time series data that we typically do with images.

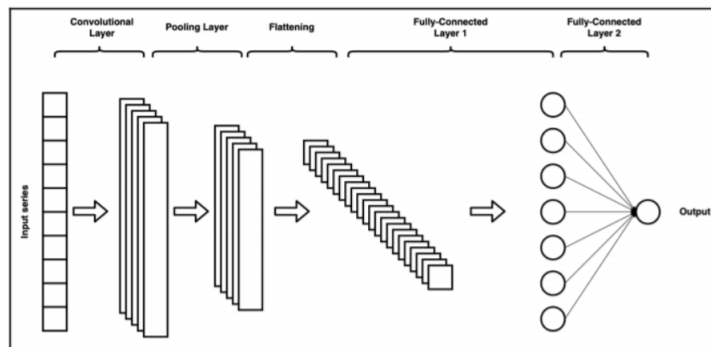


Figure 3.21: Full sequence of CNN including the flatten and dense layers

Chapter 4

Methodologies

4.1 Methodologies

We have seen the overwhelming majority of the time series analysis were done using statistical models such as ARIMA, SVM. So, We decided to implemented five different Neural Network models and analyzed their performance comparative to each other. To ensure fairness we have kept as many variables/parameters constant throughout the models as possible. The following parameters have been maintained whenever possible.

Activation functions of hidden layers	ReLu
Number of layers of each type	2
Output layer	single-unit Dense layer
Output layer activation function	None
Model Type	Sequential
Optimizer	Adam
Loss metric	Mean squared error (MSE) Mean absolute error (MAE)
Number of epochs	100

Table 4.1: Control parameters for our experiment

We tuned our learning rates for the optimizers by first training the model using a “callback” function that gradually changes the learning rate every epoch. After this training we plot the learning rate against the loss to determine the most suitable minima. The learning rate corresponding to this minima is then selected as the learning rate for that particular model. The model is then retrained using this learning rate to obtain the fully trained model which is then used to predict/validate results.

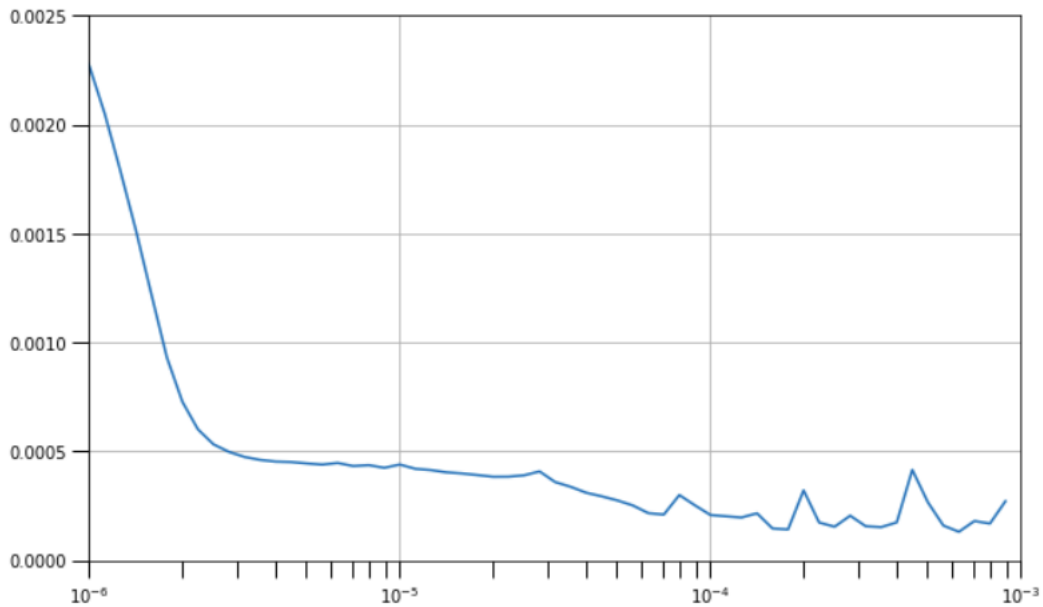


Figure 4.1: Learning rate tuning

For this figure above, we would typically choose a 10^{-5} as the learning rate for our final model. Training with this parameter would result in the graph that looks like the figure below.

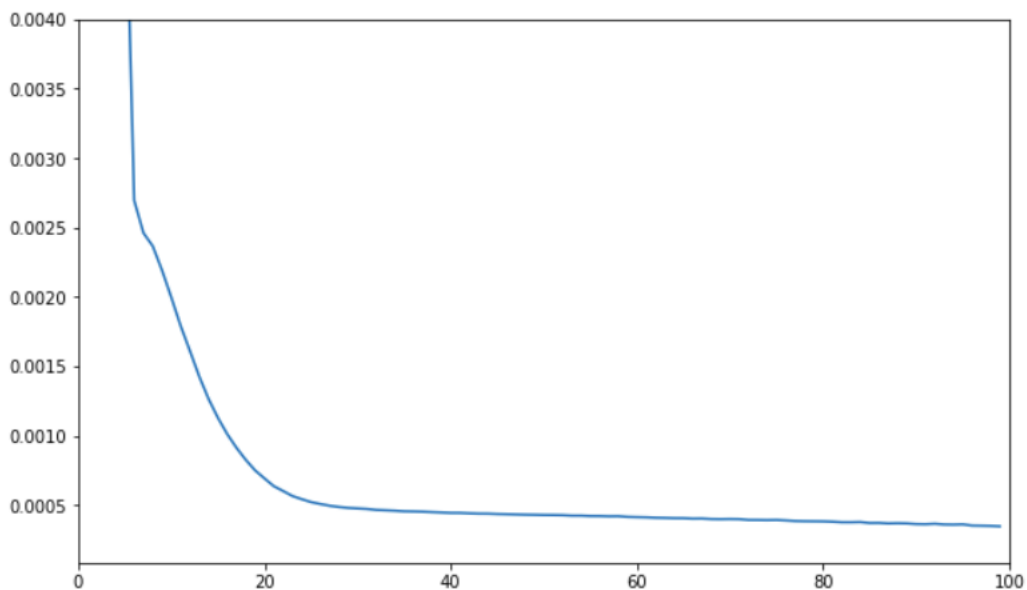


Figure 4.2: Loss after tuning

4.2 Model Structures

Dense Neural Network (DNN) - Two Dense layers with 10 units each were used as hidden layers.

Recurrent Neural Network (RNN) - We used a lambda layer to expand our input data dimensions to adjust for our RNN layers. This is followed by 2 SimpleRNN layers with 10 units each as hidden layers.

LSTM - We used a lambda layer to expand our input data dimensions to adjust for our LSTM layers. This is followed by 2 Bi-directional LSTM layers with 10 units each as hidden layers.

Convolutional Neural Network (CNN) - We used two 1-D Convolutional layers with a Maxpooling layer following each one. The convolution layers have 16 filters and kernel size of 2. The Maxpooling layers have a pooling size of 2. This sequence of layers is followed by a Flattening layer and a Dense Layer with 10 units.

CNN + LSTM - For this model we used a 1-D Convolutional layer with 64 filters and kernel size of 3 followed by two LSTM layers with 64 nodes each as hidden layers.

Chapter 5

Dataset

5.1 Source of dataset

For our dataset we chose the SP500 index. This is a widely used index which traces the performance of 500 large companies across the world. The market is matured and well maintained. We took the dataset from kaggle's repository for the SP500 index which updated daily.

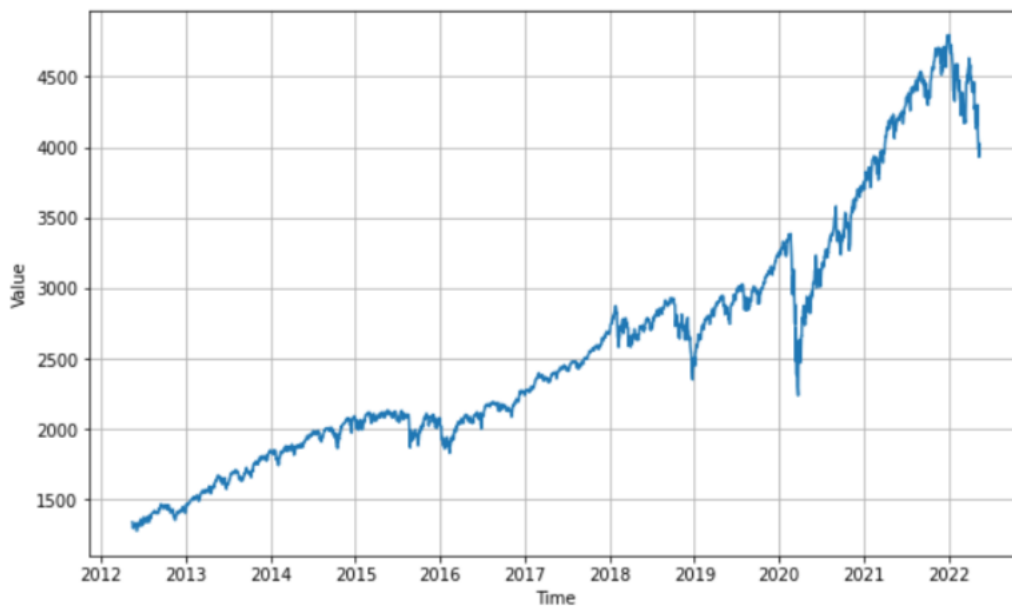


Figure 5.1: SP500 index

	Date	S&P500
0	2012-05-14	1338.35
1	2012-05-15	1330.66
2	2012-05-16	1324.80
3	2012-05-17	1304.86
4	2012-05-18	1295.22

Figure 5.2: First 5 elements of the dataset

S&P500

count	2518.000000
mean	2582.951152
std	873.213678
min	1278.040000
25%	1969.545000
50%	2392.595000
75%	2975.987500
max	4796.560000

Figure 5.3: Dataset statistics

5.2 Pre-processing

We took the dataset and normalized it. Following that we divided the data into batches with a batch size of 32 with each batch containing 30 days of data. The label of each day was the closing price of the following day. Following that we split our dataset into training and testing sets keeping the last year as our testing duration.

Chapter 6

Result

6.1 Dense Neutral Network

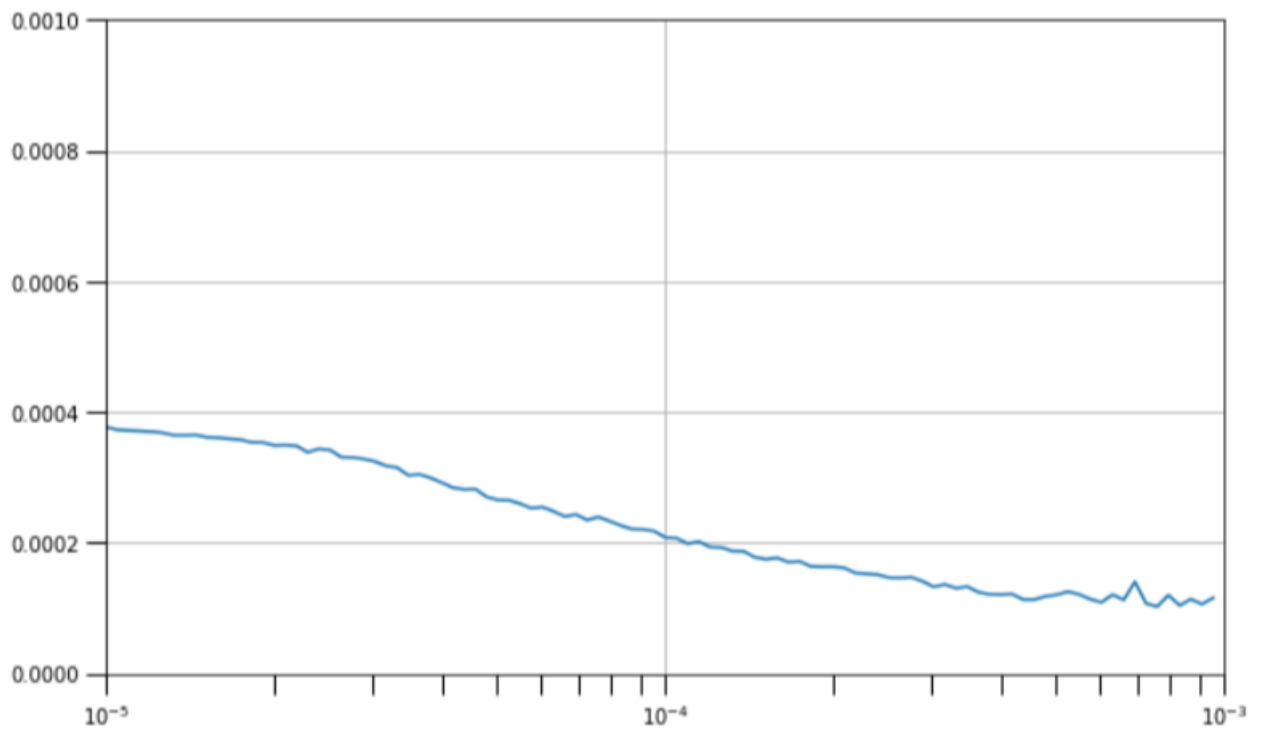


Figure 6.1: Dense neural network learning rate tuning

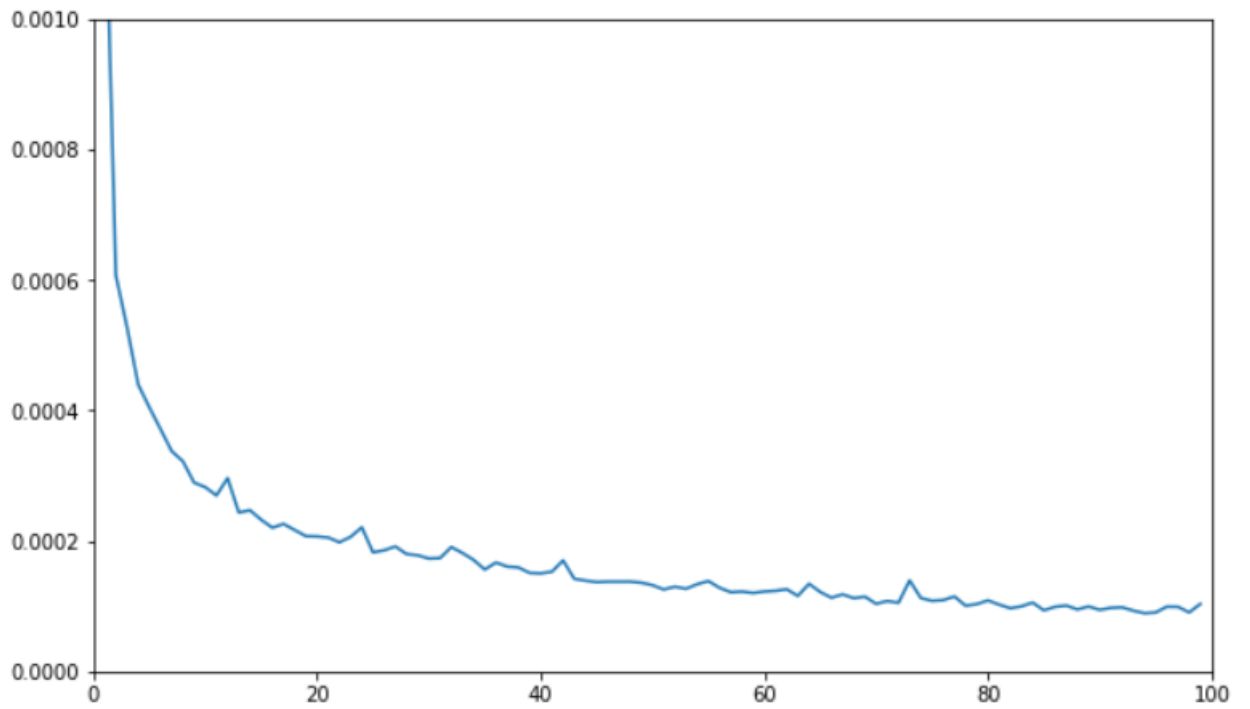


Figure 6.2: Dense neural network loss after tuning

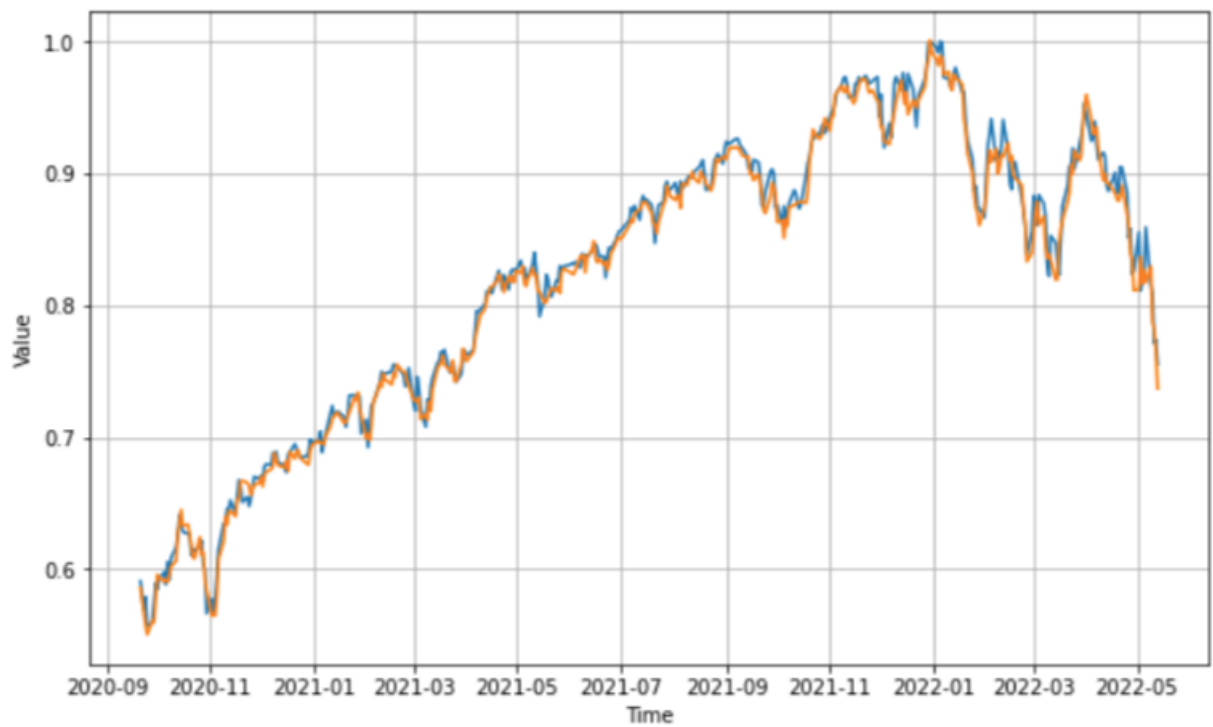


Figure 6.3: Dense neural network prediction

6.2 Recurrent Neural Network

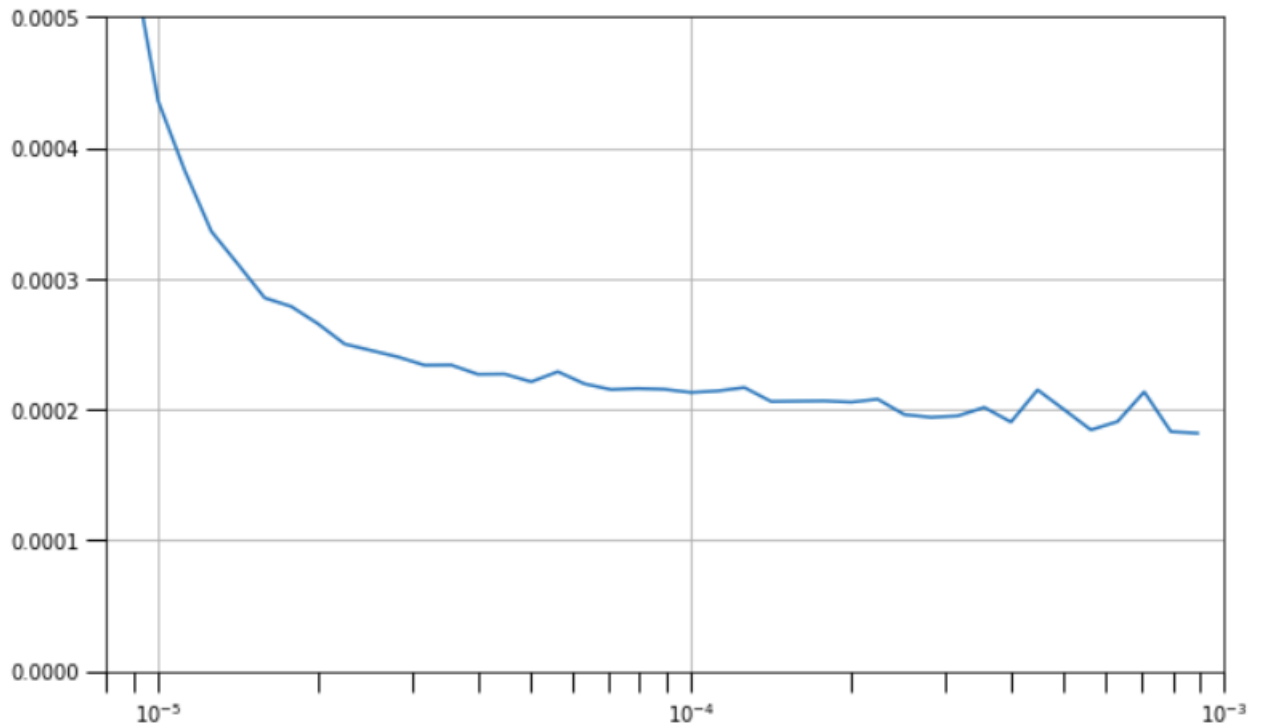


Figure 6.4: Recurrent neural network learning rate tuning

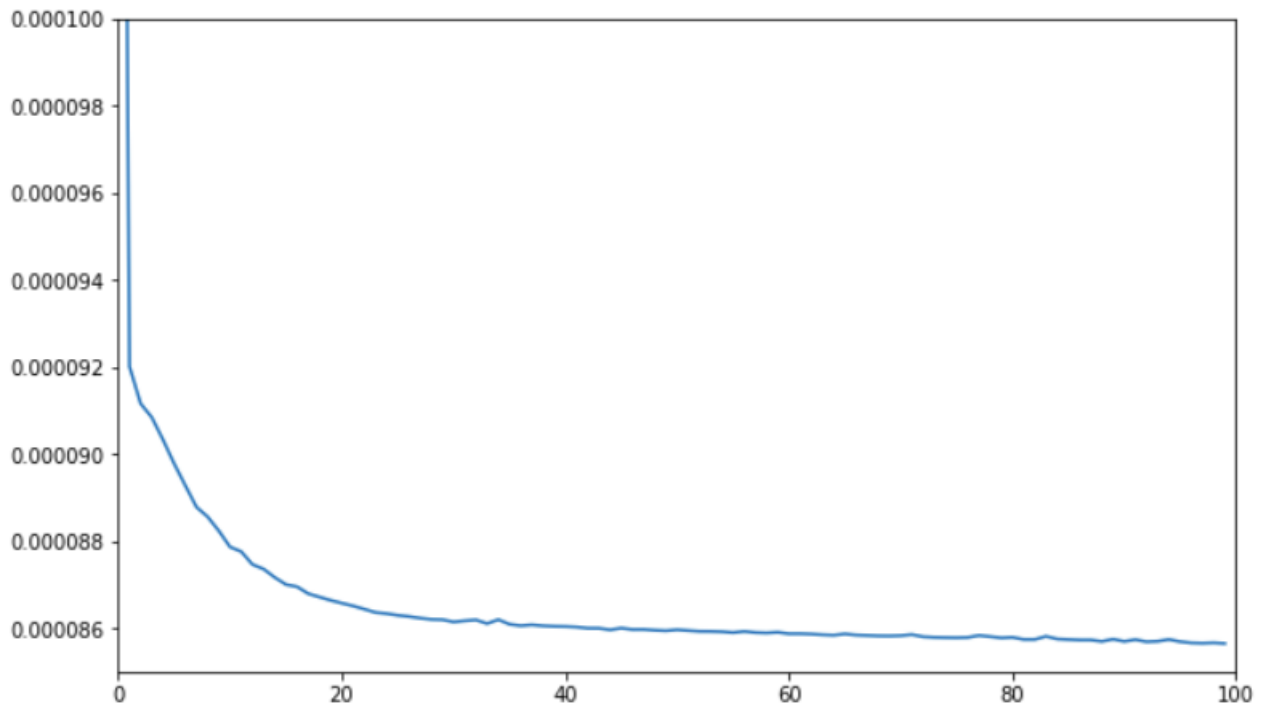


Figure 6.5: Recurrent neural network loss after tuning

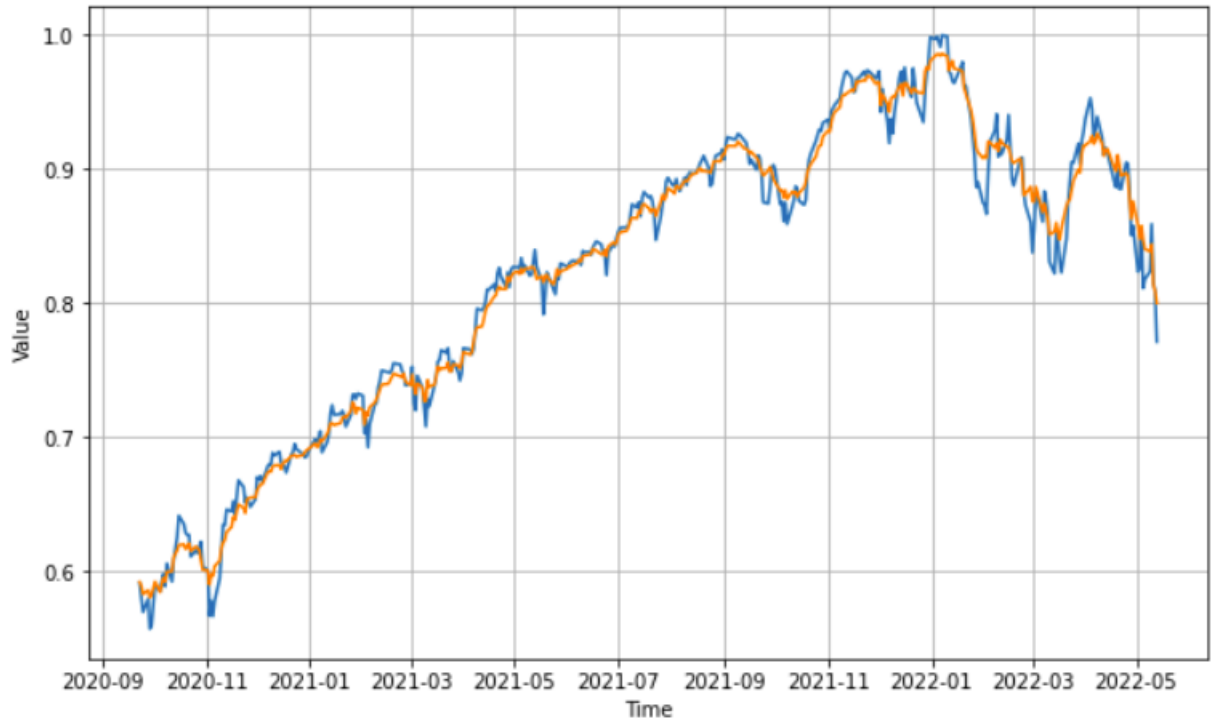


Figure 6.6: Recurrent neural network prediction

6.3 LSTM

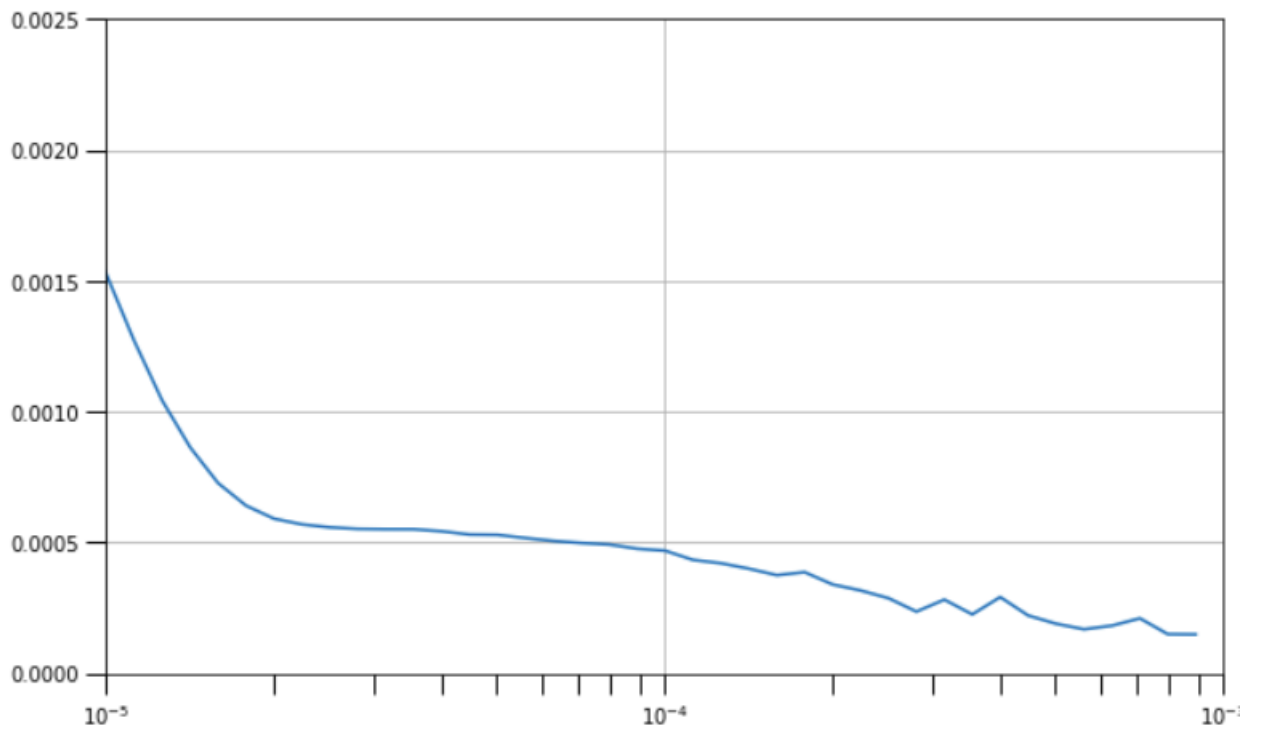


Figure 6.7: LSTM neural network learning rate tuning

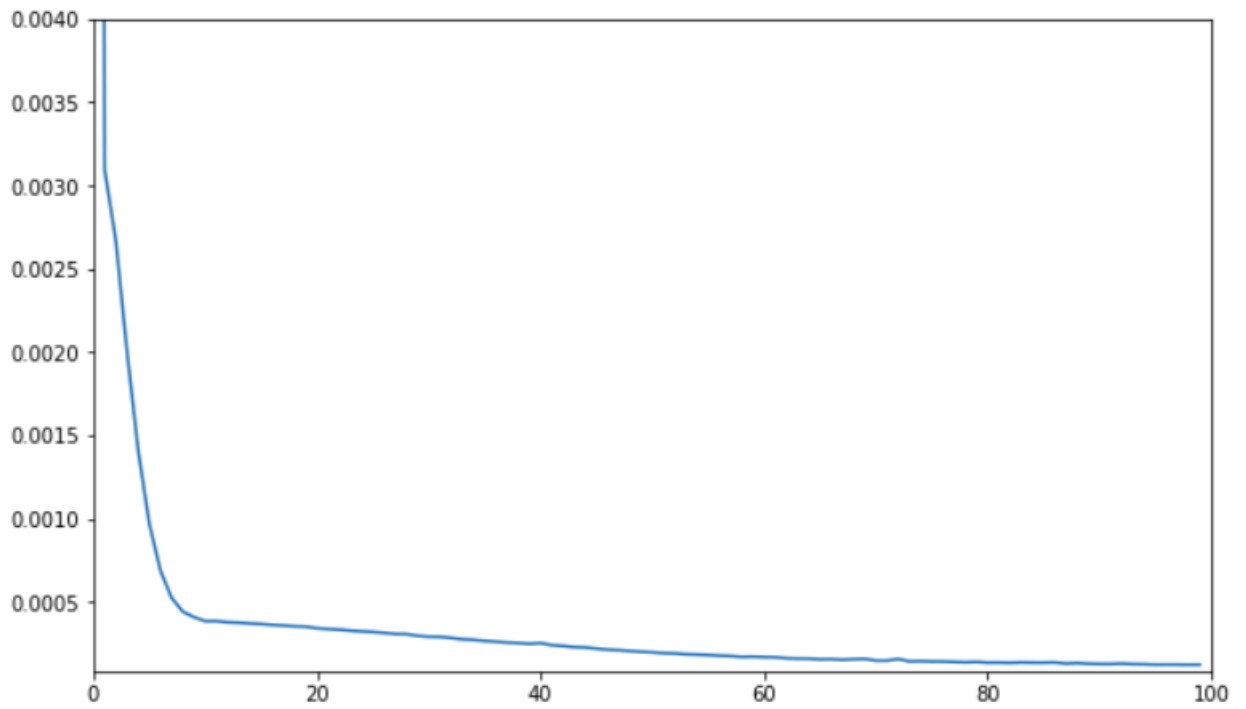


Figure 6.8: LSTM neural network loss after tuning

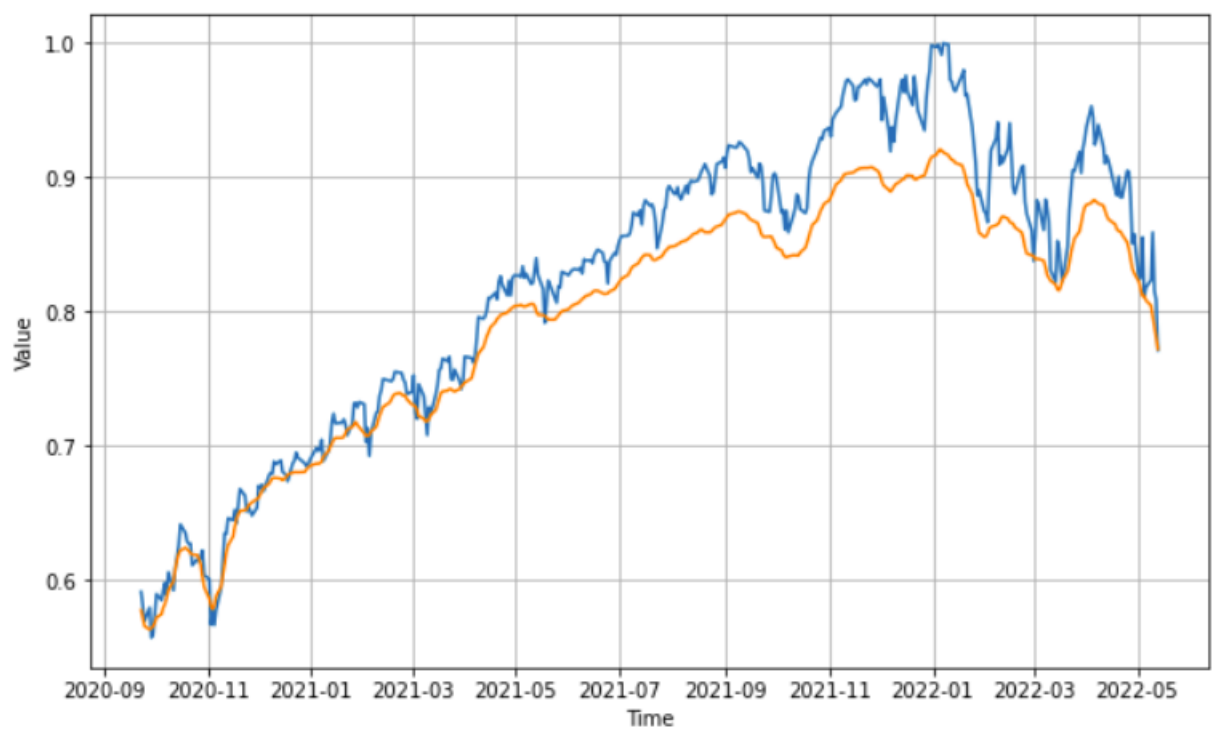


Figure 6.9: LSTM neural network prediction

6.4 CNN

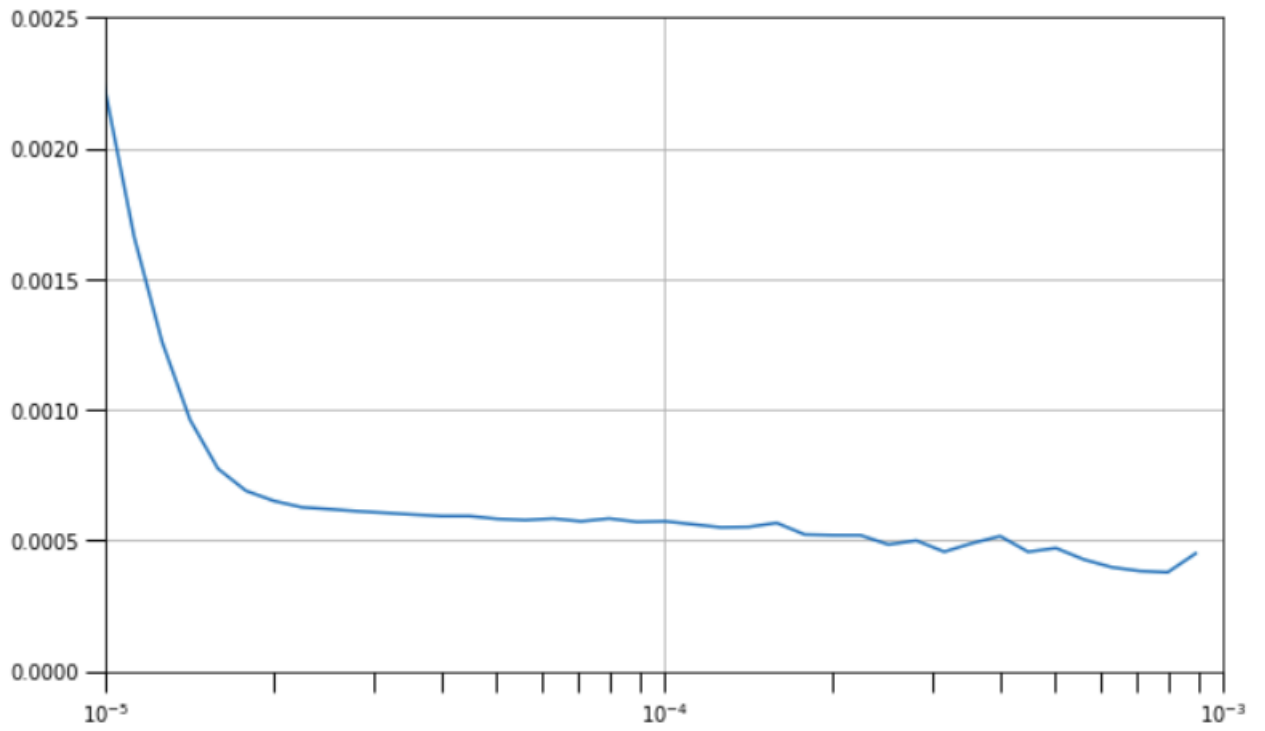


Figure 6.10: CNN neural network learning rate tuning

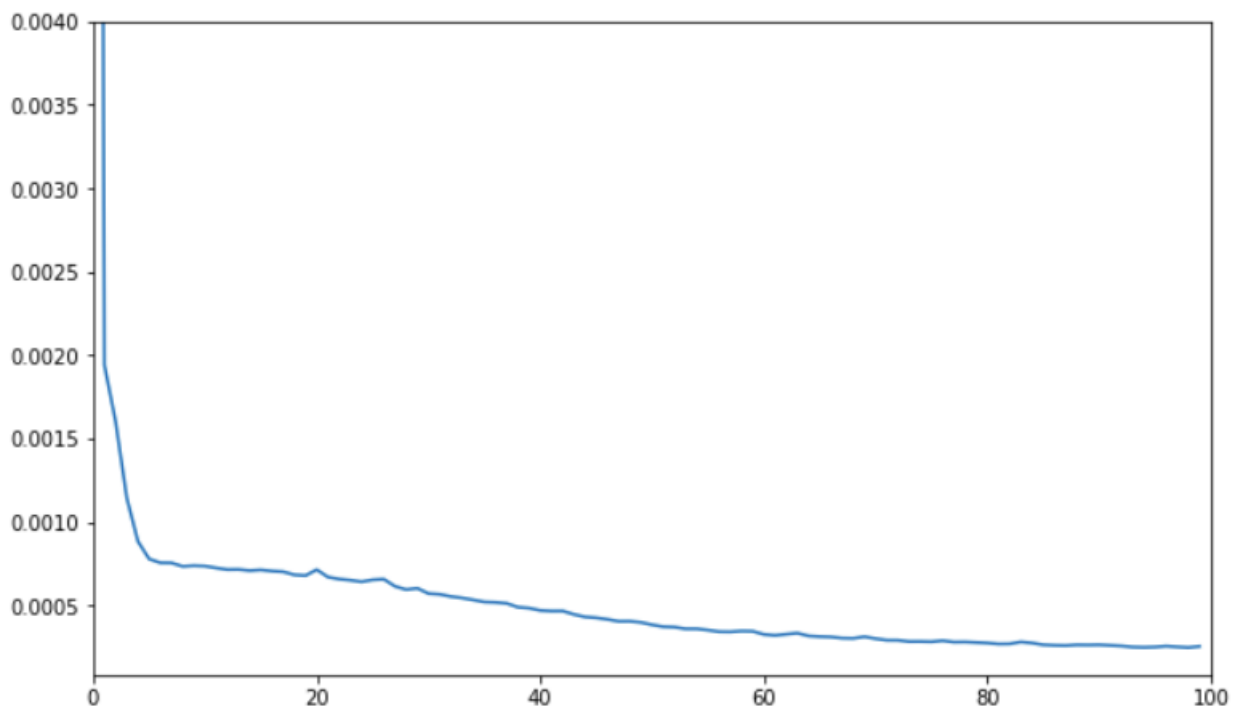


Figure 6.11: CNN neural network loss after tuning

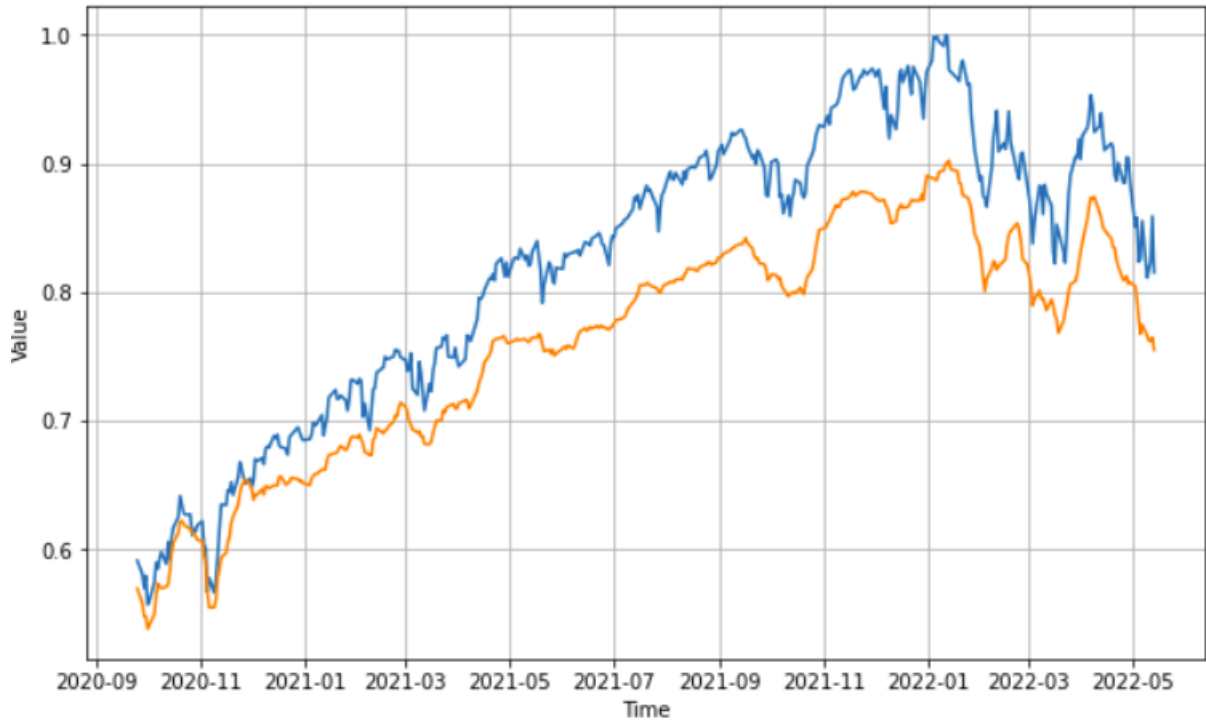


Figure 6.12: CNN neural network prediction

6.5 Pipeline

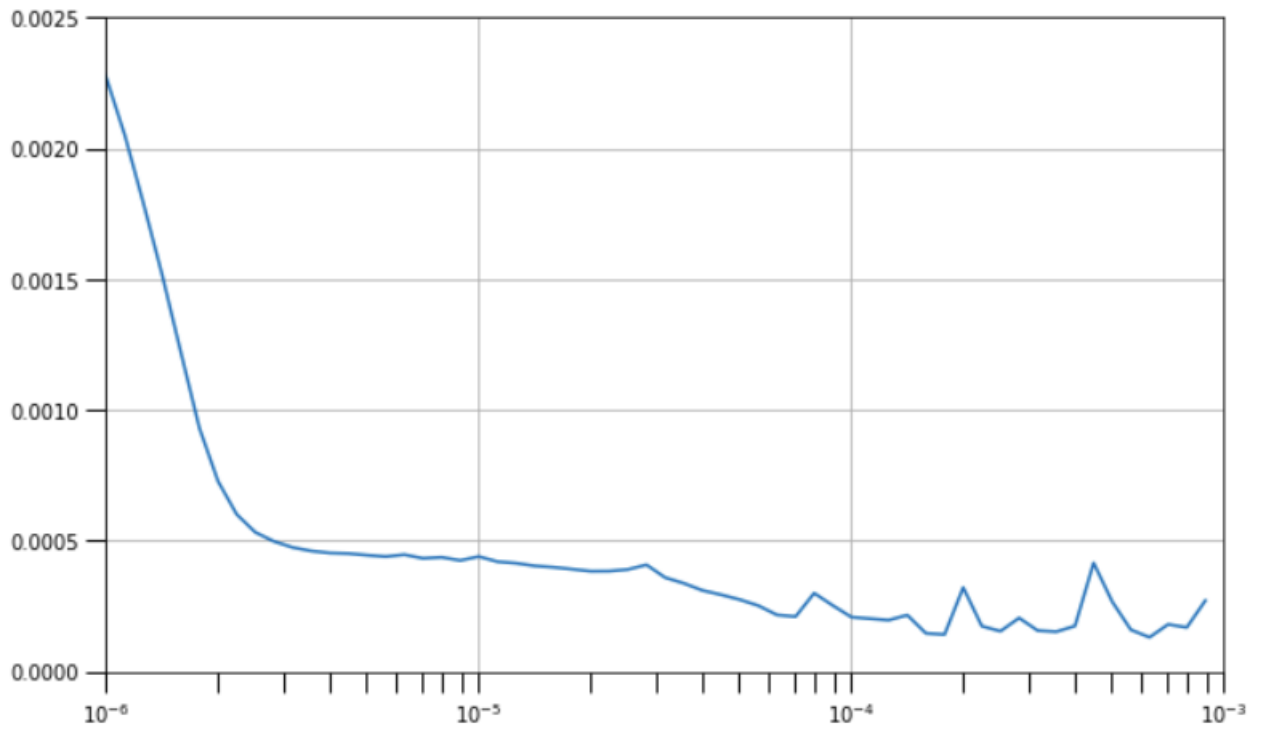


Figure 6.13: Pipeline neural network learning rate tuning

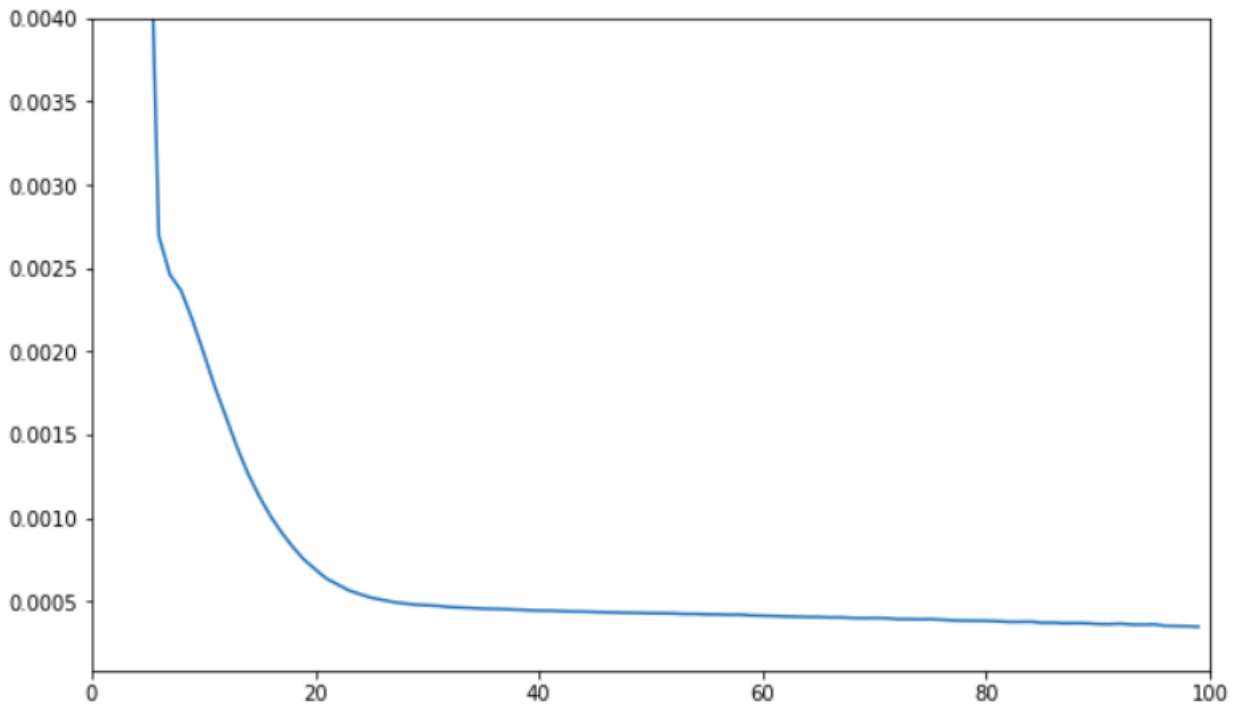


Figure 6.14: Pipeline neural network loss after tuning

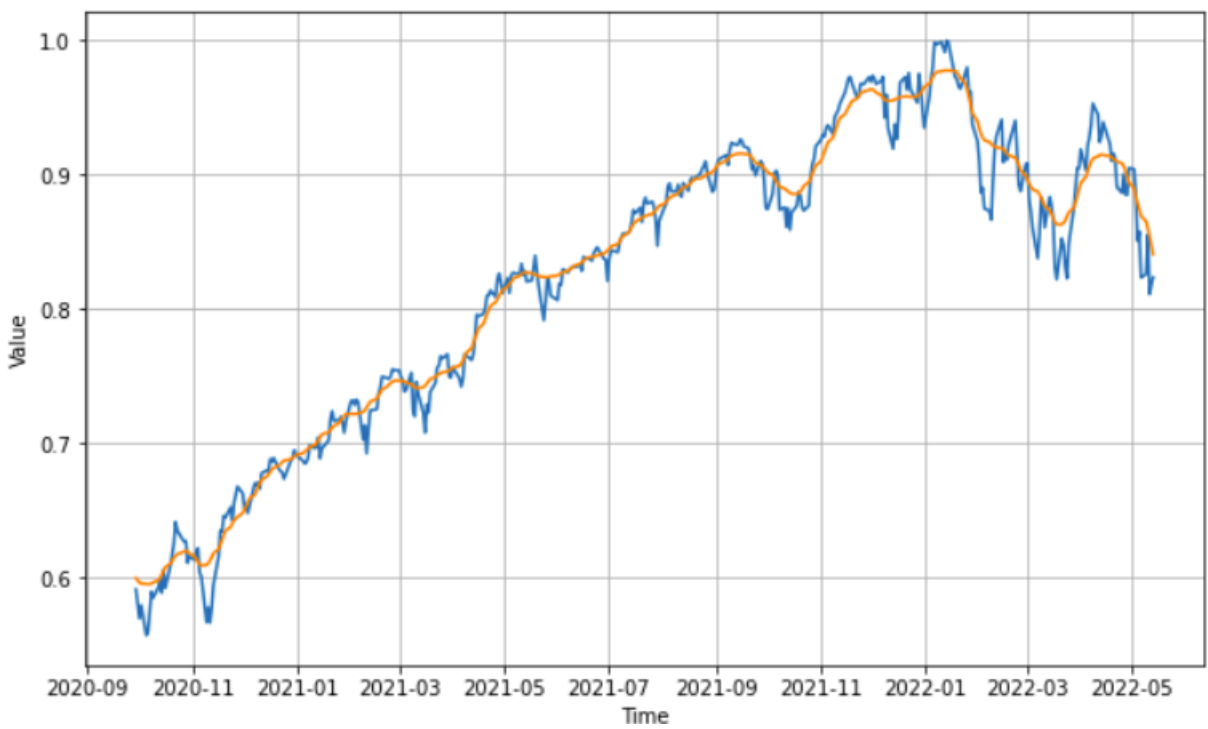


Figure 6.15: Pipeline neural network prediction

Neural Networks	MSE	MAE	Time taken per epoch(sec)
DNN	0.000110926	0.008160368	0.35
RNN	0.000127108	0.008575701	3
LSTM	0.001304831	0.029767652	3
CNN	0.004349234	0.060925997	1
Pipeline	0.000233072	0.011257544	1

Table 6.1: MSE and MAE Comparisons of all the models

We can see that for relatively low number of epochs in our case 100 DNN performs the best with RNN coming in a close second. However, DNN took approximately 350ms per epoch to train whereas RNN took 3 sec on average per epoch. This is because our dataset was univariate as such our data was fairly uncomplicated and it is known that models such as RNN, LSTM and CNN will perform better with multi-variate data with increasing number of features.

Chapter 7

Conclusion

Nowadays the prediction of financial data is a must which enable most economic institutions to make informed decisions for the maximum profit. Every algorithm has a use-case where it performs better than others. Thus, It is important to know the dataset intimately before selecting the proper algorithm. It also helps to keep in my mind that some algorithms can predict farther into the future more accurately than others. Our aim here was to provide a side by side comparison of these models while keeping their configurations and parameters as close as possible to each other. Even though such models have been used for time series analysis before, we scarcely any resources for a direct comparison between their performances for an extensive list of models. As such we decided to see to it ourselves. For our use-case we were simply predicting one day into the future and our data was univariate and trained to only 100 epochs. So, our simplistic DNN model came out on top. Further study is needed with more complicated data with more features and increasing the range of number of days ahead to be forecasted.

Bibliography

- [1] Z. Li, Y. Li, F. Yu, and D. Ge, “Adaptively weighted support vector regression for financial time series prediction,” in *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 3062–3065. DOI: 10.1109/IJCNN.2014.6889426.
- [2] X. Guo, “The research of forecasting cash inflow and outflow based on time series analysis,” in *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1, 2015, pp. 68–71. DOI: 10.1109/ISCID.2015.289.
- [3] J.-F. Chen, W.-L. Chen, C.-P. Huang, S.-H. Huang, and A.-P. Chen, “Financial time-series data analysis using deep convolutional neural networks,” in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, 2016, pp. 87–92. DOI: 10.1109/CCBD.2016.027.
- [4] Y. Yujun, Y. Yimei, and L. Jianping, “Research on financial time series forecasting based on svm,” in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2016, pp. 346–349. DOI: 10.1109/ICCWAMTIP.2016.8079870.
- [5] S. Tiwari, A. Bharadwaj, and S. Gupta, “Stock price prediction using data analytics,” in *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, 2017, pp. 1–5. DOI: 10.1109/ICAC3.2017.8318783.
- [6] D. Agnieszka and L. Magdalena, “Detection of outliers in the financial time series using arima models,” in *2018 Applications of Electromagnetics in Modern Techniques and Medicine (PTZE)*, 2018, pp. 49–52. DOI: 10.1109/PTZE.2018.8503260.
- [7] S. S. Ratakonda and S. Sasi, “Seasonal trend analysis on multi-variate time series data,” in *2018 International Conference on Data Science and Engineering (ICDSE)*, 2018, pp. 1–6. DOI: 10.1109/ICDSE.2018.8527804.
- [8] H. Sun and J. Xu, “Improved approaches for financial market forecasting based on stationary time series analysis,” in *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, 2018, pp. 334–339. DOI: 10.1109/ICBDA.2018.8367703.
- [9] M. Almuammar and M. Fasli, “Deep learning for non-stationary multivariate time series forecasting,” in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 2097–2106. DOI: 10.1109/BigData47090.2019.9006192.

- [10] F. Wang, M. Li, Y. Mei, and W. Li, “Time series data mining: A case study with big data analytics approach,” *IEEE Access*, vol. 8, pp. 14 322–14 328, 2020. DOI: 10.1109/ACCESS.2020.2966553.
- [11] H. Yu, L. J. Ming, R. Sumei, and Z. Shuping, “A hybrid model for financial time series forecasting—integration of ewt, arima with the improved abc optimized elm,” *IEEE Access*, vol. 8, pp. 84 501–84 518, 2020. DOI: 10.1109/ACCESS.2020.2987547.