REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

MEHENNI Amel Hakima et MOUCER Nessrine

Sur le thème

Long Short-Term Memory Networks for Time Series Forecasting

Soutenu publiquement le 28 / 06 / 2022 à Tiaret devant le jury composé de :

Mr. KOUADRIA Abderahmane	MCB	Université Ibn-Khaldoun	Président
Mr. BOUDAA Boudjemaa	MCA	Université Ibn-Khaldoun	Encadrant
Mr. BERBER El-Mehdi	MAA	Université Ibn-Khaldoun	Examinateur

2021-2022

Abstract

Time series forecasting occurs when you make scientific predictions based on historical time stamped data. It involves building models through historical analysis and using them to make observations and drive future Strategic decision-making. Their main specificities compared to the most common areas of machine learning are their dependence over time and their seasonal behaviors that can appear in their evolution. In the literature, statistical models are widely used for time series forecasting. However, there are many complex models or approaches that can be very useful in some cases. Generalized Autoregressive Conditional Heteroskedasticity (GARCH), Bayesian models and ARIMA vectors (VAR) are just a few examples. There are also even time series models borrowed from deep learning. Recently, many fields including computer vision have paid a lot of attention to deep learning techniques like recurrent neural networks and long short term memory (LSTM), convolutional neural networks (CNN) or GRU cells can be successfully employed for time series forecasting issues. Through this work, we aim to develop a RNN model with LSTM (Long Short-Term Memory) cell for time series forecasting. The proposed model will be experimented and evaluated on real-world datasets with the known metrics in this field.

Key-words: Time Series Forecasting, RNN, LSTM, Single-step, Multi-step, Univariate, Multivariate.

Résumé

La prévision de séries chronologiques est une méthode de prédiction des événements futurs basée sur des données passées. L'apprentissage automatique est un processus d'utilisation de modèles pour faire des prédictions sur des événements futurs. Certaines des principales différences entre l'apprentissage automatique et d'autres formes d'analyse de données sont que les modèles d'apprentissage automatique dépendent de données passées et peuvent parfois présenter un comportement saisonnier. Dans la littérature, les modèles statistiques sont très répandus utilisé pour prévoir les séries chronologiques. Il existe de nombreux modèles complexes différents qui peuvent être très utiles dans des cas spécifiques. Quelques exemples incluent l'hétéroskédasticité conditionnelle autorégressive généralisée (GARCH), les modèles bayésiens et les vecteurs ARIMA (VAR). Il existe également des modèles d'apprentissage profond qui sont empruntés à l'analyse de séries chronologiques. Les cellules GRU peuvent être un outil utile pour prévoir les problèmes de séries chronologiques. Grâce à ce travail, nous objectif de développer un modèle RNN cellulaire LSTM (Long Term Short Term Memory) pour les prévisions des séries chronologiques. Le modèle proposé sera testé et évalué sur la base de données réelles, avec les mesures connues dans ce domaine.

Mots-clés: Prévision de séries chronologiques, RNN, LSTM, Single-step, Multi-step, Univariée, Multivariée.

Acknowledgement

We would like to thank God, for letting us through all the difficulties. We have experienced your guidance day by day, you are the one who let us finish our degree. We will keep on trusting you for our future.

We would like to acknowledge and give our warmest thanks to our supervisor ***Mr Boudaa Boudjemaa*** *who made this work possible. His guidance and advice carried us through all the stages of writing our project. We would also like to thank our committee members for letting our defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you. Lastly, we appreciate all the support we received from our family who have been our source of inspiration.*

Thank you all.

Dedications

I dedicate this thesis to God Almighty my creator, my strong pillar, my source of inspiration, knowledge and understanding, he has been the source of my strength throughout this program and on his wings only have I soared. I also dedicate this work to my father, who has been nicely my supporter until my research was fully finished. And my mother who, for months past has encouraged me attentively with her fullest and truest attention to accomplish my work with truthful self-confidence. Also I would like to dedicate my best brother Abdelkader for his helpful to complete my project. My friends and cousins Imen, Sarra, Narimene,, Yassmine, Saoussen, Aicha and Fatima.

Mehenni Amel hakima.

This thesis is dedicated to the sake of Allah, my Creator and my Master, My great teacher and messenger, Mohammed (May Allah bless and grant him), who taught us the purpose of life. My great parents, who never stop giving of themselves in countless ways, for teaching us that even the largest task can be accomplished if it is done one step as a time. My beloved brothers and sisters: Redha ,Ilyes, Rania and Lydia. All the people in my life who touch my heart like Imen, Sara , Yassmine, Saoussen and Narimene.

Moucer Nesrine.

Contents

List of Figures

List of Tables

List of Abbreviations

LSTM: Long Short Term Memory
AI: Artificial intelligence
AV: Autonomous Vehicles
ML: Machine Learning
NNs: Neural Networks
ARIMA: Auto Regressive Integrated Moving Average
SES: Single Exponential Smoothing
CNNs: Convolutional Neural Networks
RNNs: Recurrent Neural Networks
ANN: Artificial Neural Network
GANs: Generative Adversarial Networks
RBFNs: Radial Basis Function Networks
ADFT: Augmented Dickey-Fuller Test
KPSS: Kwiatkowski Phillips Schmidt Shin
KNN: K-Nearest Neighbor
DTW: Dynamic Time Warping
DBSCAN: Density - Based Spatial Clustering of Applications with Noise
AR: Auto Regressive
MA: Moving Average
SARIMAX: Seasonal Auto Regressive Integrated Moving Average with exogenous factors
SARIMA: Seasonal Auto Regressive Integrated Moving Average
DES: Double Exponential Smoothing
TES: Triple Exponential Smoothing
SVM: Support Vector Machine
SVR: Support Vector Regression
CART: Classification and Regression Tree
MLP: Multi-Layer Perceptron
NLP: Natural Language Processing
ESN: Echo State Network
RC: Reservoir Computing
GRU: Gated Recurrent Units networks
ReLU: Rectified Linear Unit
MSE: Mean Squared Error
MAE: Mean Absolute Error
MSD: Mean Squared Deviation
RMSP: Root Mean Squared Propagation or RMSProp
RMSE: Root Mean Square Error
BPTT: Backpropagation Through Time
ERNN: Elman Recurrent Neural Networks
API: Application Programming Interface
RF: Random Forest
NB: Naive bayes

General introduction

1. Background

Forecasting in business context is about future customer demand, it is an essential business process for any firm whatever the product it might sell and the key question is what will customer demand be at some future point in the time?, a future point of time could be tomorrow (shorter term operational decisions or short term forecast) or even next week or next month or next year (longer term operational decisions or long term forecast).

There are two main ways to estimate future customer demand; the first is *qualitative* approaches that based on subjective assessment. Qualitative forecasting methods include the *grass roots* approach that is future customer demand can be measured by talking to salespeople, their knowledge of the market and customer trends and preferences may be helpful to estimate future sales. Market research is another method of qualitative approach this can include surveying customers, testing product performance or having focus groups to discuss a product and how consumers might receive it, then there is Delphi method which is about enlisting the opinion of trusted advisors and consumers regarding what is going to happen based on their consumer behavior, compiling this data, and presenting it to decision-makers. The second one is *quantitative* approach, there are two main groups of quantitative forecasting methods, the least used is casual forecasting (associative model) that understand the casual drivers of demand is the goal for example we might expect demand for a product to increase as population and income levels in market area arise, the effect of many other factors on customer demand can be estimated using regression analysis, this allow to predict future demand based on the expected future state of the predictor variables, the other method of quantitative forecasting and the most important is *time series forecasting* it is simply based on prior demand or sales data and compared to casual method, time series forecasting is much simpler and easier to implement and it is widely used in managerial practice.

Artificial Intelligence (AI) has gained considerable prominence over the last decade fueled by a number of high profile applications in Autonomous Vehicles (AV), intelligent robots, image and speech recognition, automatic translations, medical and law usage as well as beating champions in games like chess, Jeopardy, GO and poker, and so AI found applications in the field of forecasting and a considerable amount of research has been conducted on how a special class of it, utilizing Machine Learning methods (ML) and especially Neural Networks (NNs), can be exploited to improve time series predictions.

2. Problem statement

Classical machine learning models are widely used for time series forecasting, such as exponential smoothing and ARIMA. Exponential smoothing has been around since 1950s, then it extended many times giving a total of fifteen methods like the simple exponential smoothing (or SES) method, Holt's linear method, additive Holt-Winters' method. Auto-Regressive Integrated Moving Average are developed in 1930s-1940s by an electrical engineer Norbert Wiener et al, then statisticians George Box and Gwilym Jenkins developed systematic methods for applying them to business data in the 1970s. These two methods are the most popular in the traditional machine learning models but there are much more. However these methods have limitations. First the performance of the methods can be affected by the missing values. Second, in the data, they are not able to recognize complex patterns. Finally, they don't work well in long forecast. These limits have been solved by the appearance of deep learning in 1943s when Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain, also the appearance of Backpropagation model in 1960s. The fast development led this two innovation to give birth to several approaches like: Convolutional Neural Networks (CNNs), Long Short Term Memory Networks (LSTMs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), Radial Basis Function Networks (RBFNs).

In the light of the above, a question must be asked: *How deep learning can be used for time series forecasting?*

3. Approach

To answer the previous question, an RNN model with LSTM (Long Short-Term Memory) cell for time series forecasting is developed. The proposed model will be experimented and evaluated on real-world datasets with the known metrics in this field.

4. Outline

This thesis is divided into three chapters and a general introduction, which includes the following sections: background, problem statement, approach, and outline. Chapters one and two are theoretical; chapter three will be the application section, in which the steps of implementation and the results will be presented; and finally, we will draw a broad conclusion based on the results and discussion from chapter three. Each chapter's structure is as follows:

- *Chapter one: Time series forecasting*

This chapter gives a gentle introduction to time series generally and time series forecasting specially, with different technics used for forecasting and concludes with real world study cases that used time series forecasting.

- *Chapter two: Long-Short Term Memory*

This chapter introduce deep learning with neuron networks and how *Long-Short Term Memory* gives solutions for recurrent neural networks in order to solve problems.

- *Chapter three: LSTM-based RNN Model Development for Time Series Forecasting*

This chapter is the practical part of this thesis in which the overview and architecture of the LSTM- based RNN model development for time series forecasting are presented, as well as datasets and baselines used. At the last, the baselines results are compared to the proposed LSTM-based RNN model with a discussion of the obtained results.

Chapter 1

Time series forecasting

1.1 Introduction

Time series data is everywhere since the increasing use of advanced devices and software in the world. Time series data is gathered, stored, visualized and analyzed for various purposes across various domains, in data mining, pattern recognition and machine learning, time series analysis is used for clustering, classification and query by content, anomaly detection and forecasting. Time series data is used in time series analysis and time series forecasting to detect and predict patterns essentially looking at change over time.

1.2 Time series

1.2.1 Definition

A time series is a sequence or series of numerical data point fixed at certain chronological time order. In a time series, time is often the independent variable and the goal is usually to make a forecast for the future.

1.2.1 Time series data and components

There are four basic components of time series:

1.2.2.1 Trend

A trend exists when there is a long-term increase or decrease in the data. Trend usually happens for some time and then disappears. It is known that the trend does not repeat [1]. A trend could

be:

- Uptrend/Downtrend: Time Series Analysis shows a general pattern that is upward then it is Uptrend. If it shows a pattern that is downward then it is downtrend (See Figure 1.1).

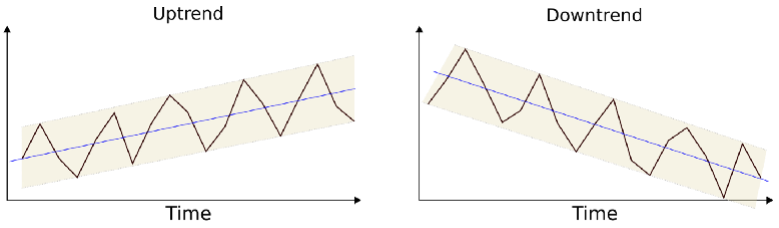

Figure 1.1: Up/Down-trend

- Horizontal or Stationary trend: If no pattern observed then it is called a horizontal or stationary trend (See Figure 1.2).

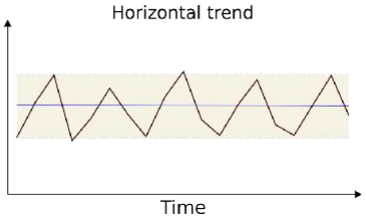

Figure 1.2: Horizontal trend

1.2.2.2 Seasonal

Seasonality refers to predictable changes that occur over a one-year period. It can be used to help analyze stocks and economic trends also it can determine certain business decision such as inventories and staffing for companies. By seasonality, we mean periodic fluctuations. For example, retail sales tend to peak for the Christmas season and then decline after the holidays. So time series of retail sales will typically show increasing sales from September through December and declining sales in January and February. A time series data can be explored with many graphical representations [2] to detect seasonality such as:

- **Box plots**

Box plots can be used as an alternative to the seasonal subseries plot to detect seasonality. The following (Figure 1.3) is showing what a box plot look like, all details about this example is on [3].

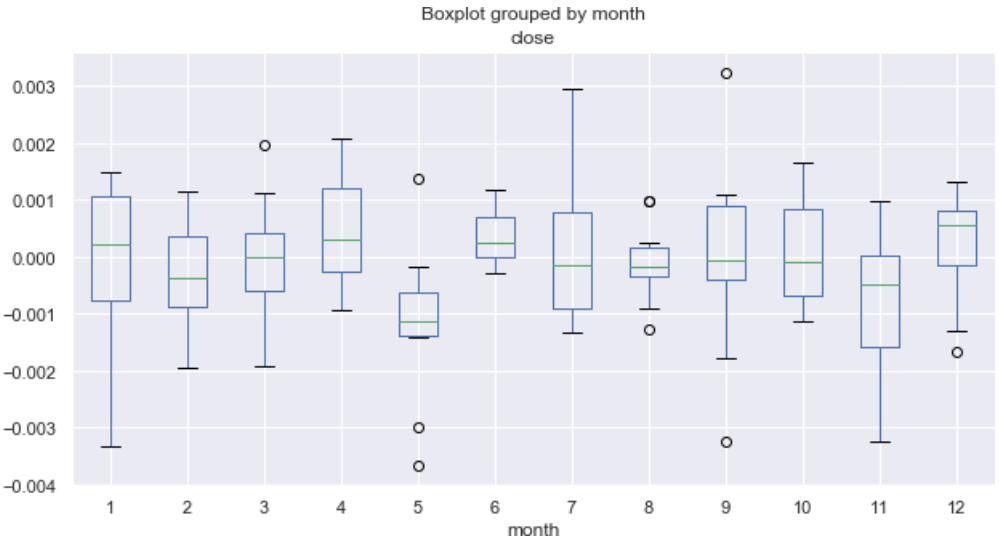

Figure 1.3: Box plot example

- **Seasonal subseries plot**

Seasonal subseries plot is a specialized technique for showing seasonality. The seasonal subseries plot shows the seasonal pattern more clearly. In the following (Figure 1.4) shows an example of seasonal subseries plot which illustrate CO2 concentrations. The CO2 concentrations are at a minimum in September and October. From there, steadily the concentrations increase until June and then begin declining until September [2].

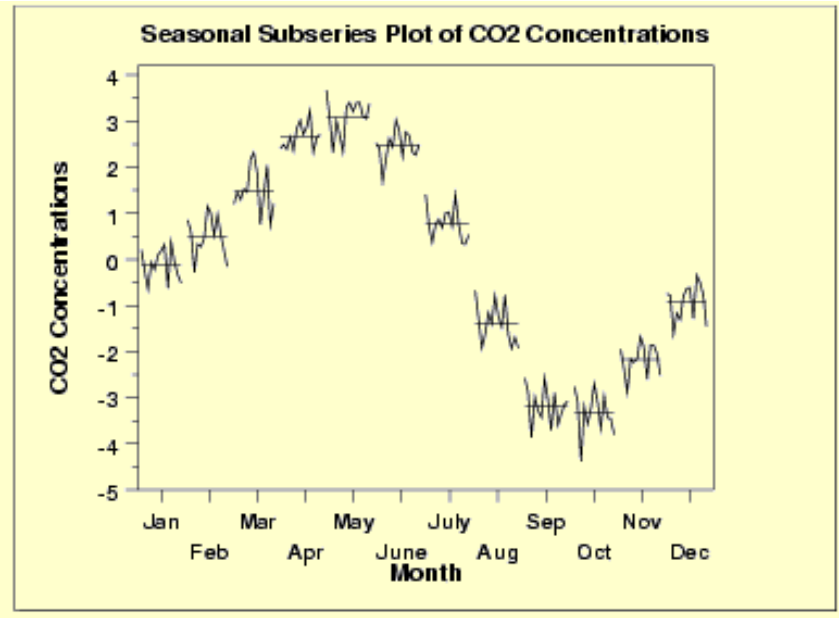

Figure 1.4: Seasonal subseries plot of CO2 concentrations

1.2.2.3 Cyclical

A cyclic pattern exists when data exhibit rises and falls that are not of fixed period. A seasonal behavior can be confused with cyclical one, but there is some characteristic that helps make difference. First of all if the fluctuations are not of a fixed frequency then they are cyclic, if the frequency is unchanging and associated with some aspect of the calendar, then it is seasonal. In general, the average length of cycles is longer than the length of a seasonal pattern, and the magnitudes of cycles tend to be more variable than the magnitudes of seasonal patterns [4].

The following (Figure 1.5) is an example of cyclical time series of monthly housing sales.

It shows some strong cyclic behavior with a period of about 6–10 years [4].

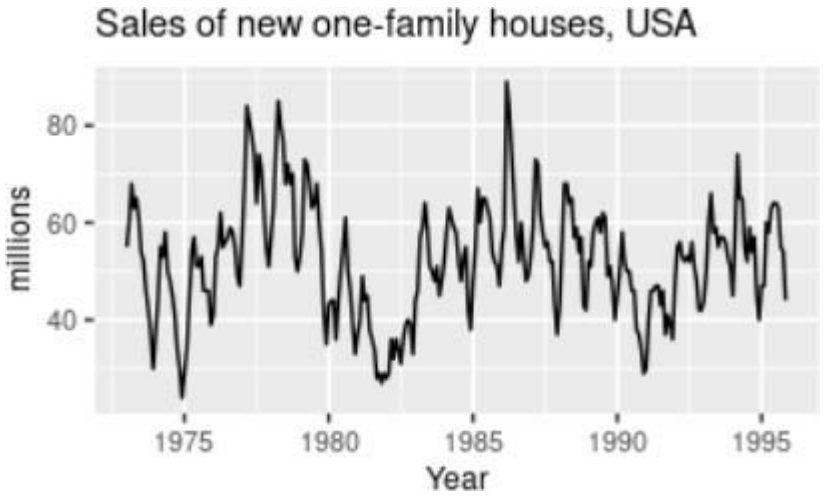

Figure 1.5: Cyclical time series

1.2.2.4 Irregular

Irregular variations correspond to the movements that appear irregularly and generally during short periods. In practice, all the components of time series that cannot be attributed to the influence of cyclic fluctuations or seasonal variations or tendency are classed as irregular [5]. The following (Figure 1.6) is a highly irregular time series. A highly irregular time series have fluctuations that can dominate movements, which will mask the trend and seasonality.

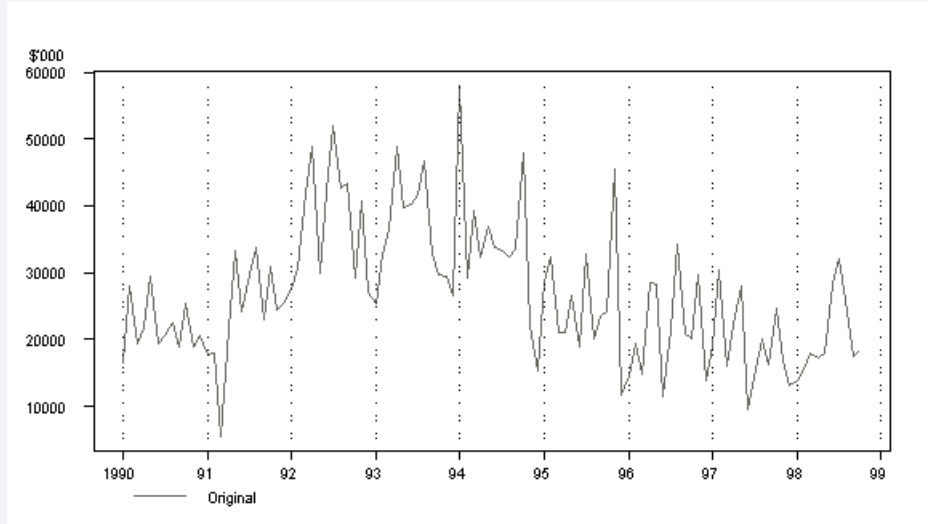

Figure 1.6: Irregular time series

1.2.3 Understanding time series

A time series is usually represented in a graph which models every data point on the x-axis and y-axis. If the plot has highest or lowest in a long time period that means the time series data is a trend time series. If there is recurrence over time measured by years it called seasonal time series and if the visualization shows changes many time in a years so that is a cyclical time series data. The following (Figure1.7) shows an example of what component of a raw time series look like and what is a time series look likes after removing trends, seasonal and cyclical, in other words what a residual (remainder) data is:

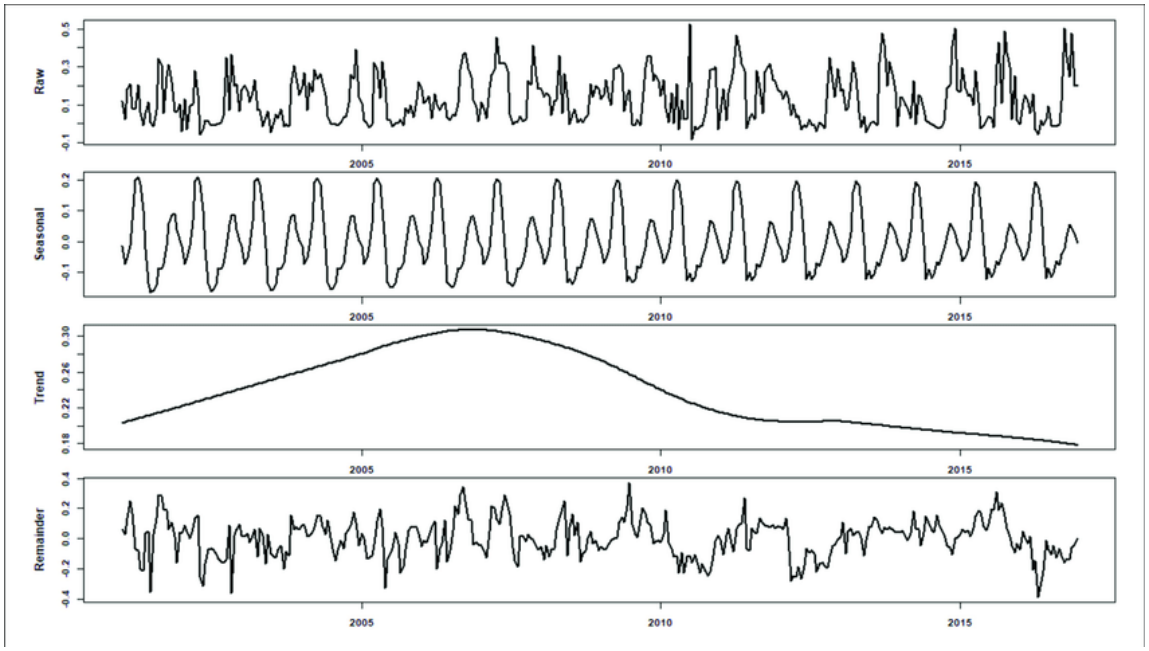

Figure 1.7: Component of time series

1.2.4 Time series types

1.2.4.1 Univariate time series

A univariate time series refers to time series that has only one observation recorded sequentially over equal time increments. Generally a univariate time series data set is given with one column numbers. (Figure1.8) [6] is a simple example of temperature values with time measured by each hour.

1.2.4.2 Multivariate time series

A multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables. This dependency is used for forecasting future values [7]. The following (Figure1.9) shows example for a multivariate time series:

Time	Temperature
5:00 am	59 °F
6:00 am	59 °F
7:00 am	58 °F
8:00 am	58 °F
9:00 am	60 °F
10:00 am	62 °F
11:00 am	64 °F
12:00 pm	66 °F

Table 1.1: Univariate time series (temperature values)

Date	Open	High	Low	Close	Volume	Market Cap
Aug 07, 2017	3212.78	3397.68	3180.89	3378.94	1,482,280,000	52,987,300,000
Aug 06, 2017	3257.61	3293.29	3155.6	3213.94	1,105,030,000	53,720,900,000
Aug 05, 2017	2897.63	3290.01	2874.83	3252.91	1,945,700,000	47,778,200,000
Aug 04, 2017	2806.93	2899.33	2743.72	2895.89	1,002,120,000	46,276,200,000
Aug 03, 2017	2709.56	2813.31	2685.14	2804.73	804,797,000	44,666,400,000
Aug 02, 2017	2727.13	2762.53	2668.59	2710.67	1,094,950,000	44,950,800,000
Aug 01, 2017	2871.3	2921.35	2685.61	2718.26	1,324,670,000	47,321,800,000

Table 1.2: Multivariate time series (Bitcoin price prediction)

1.2.5 Stationary time series

If the properties of a time series such as, variance[1] and autocorrelation[2], are all constant over time, this time series is called a stationary time series [8]. It does not mean that the series does not change over time, just that the way it changes does not itself change over time [9], for example in (Figure 1.10) the strong cycles in series might appear to make it non-stationary. But these cycles are aperiodic, they are caused when the lynx population becomes too large for the available feed, so that they stop breeding and the population falls to low numbers, then

[1]Mesure how far each number in the set is from the mean and thus from every number in the set
[2] Represent the degree of similarity between the original forms and once legged one or more time period.

the regeneration of their food sources allows the population to grow again, and so on. In the long-term, the timing of these cycles is not predictable. Hence the series is stationary.

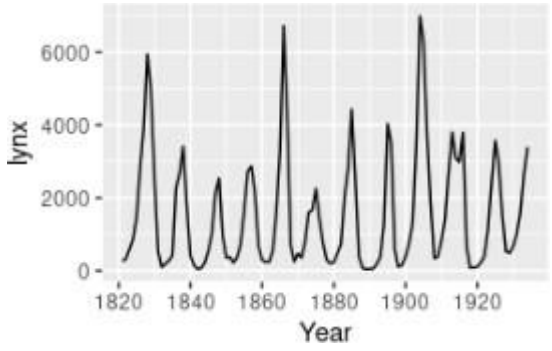

Figure 1.8: Annual total of lynx trapped in the McKenzie River district of north-west Canada

Stationarity is widely used for time series analysis, making the ability to understand, detect and model it necessary for the application of many prominent tools and procedures in time series analysis, and this is the major importance of stationary time series. When the data is no-stationary it is important to transform it into stationary one to make it easier to analyze there is an example of stationary and non-stationary data (Figure1.11).

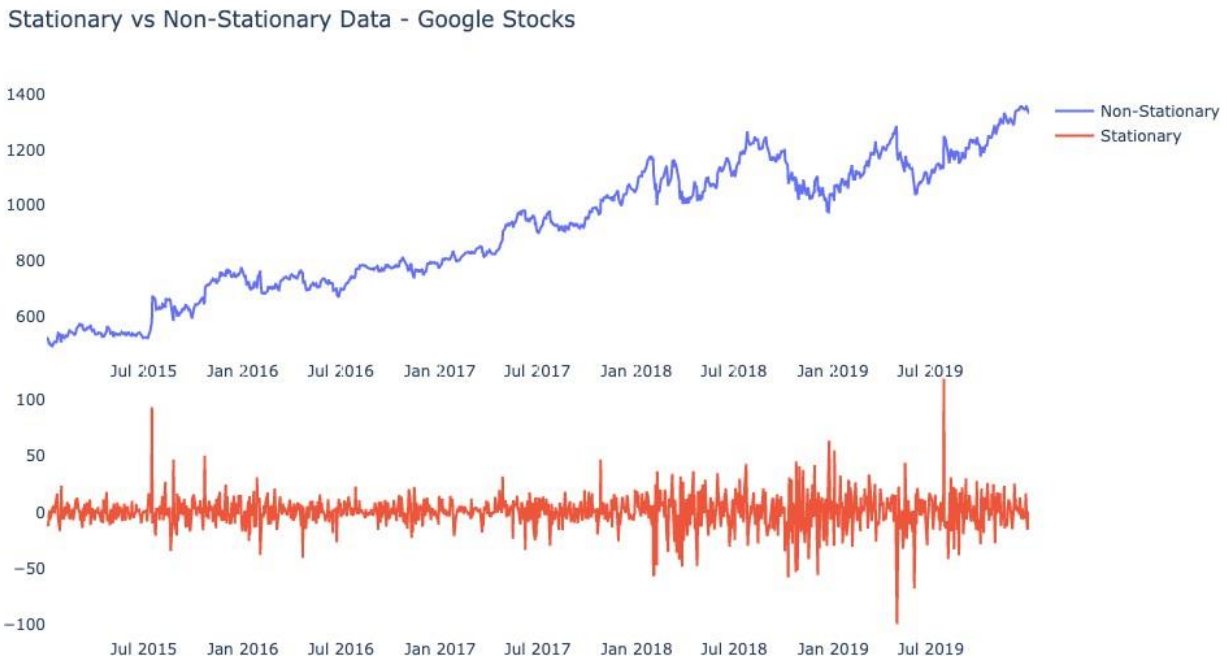

Figure 1.9: Stationary and non-stationary time series

There is some statistical method to check stationarity such as:

- **Augmented Dickey-Fuller Test** is a stationary unit root test. Unit root test is a feature in time series that make it non-stationary time series, in other words the presence of unit root means the time series is non-stationary [10]. Because of type I error rate[3]. The ADFT is used with caution. Hypothesis tests are as following:
    - Null hypothesis (H0): The time series data is non-stationary.
    - Alternate hypothesis (H1): The time series is stationary (or trend-stationary).

The Augmented Dickey-Fuller Test (ADFT) is a modifiable version of the Dickey-Fuller test; however the (ADFT) can manipulate more complicated models.

- **Kwiatkowski Phillips Schmidt Shin (KPSS) test** has the objective of checking stationarity all over the mean, linear trend or if it is non-stationary due to unit root. Hypothesis tests are as following:
    - Null hypothesis (H0): The data is stationary.
    - Alternate hypothesis (H1): The data is not stationary.

The most common algorithm to transform a non-stationary data into stationary one are:

- **The difference** the objective is to create a new series data that is one less point that the original data, but it is possible to difference the data more than once. The new series $Y_i$is represented in terms of the old one Z_i as :

$$Y_i = Z_i - Z_{i-1}$$

- **Logarithmic transformation** as the name indicate the function *log()* is used to transform the data into stationary series. First of all if the values of the data are negative it must be transform into positive one, and this may be considered as a limit.

Stationarity in time series has three famous types [11]:

- **Strong stationarity** if the distribution of the aleatory value produced is exactly the same in measure of probability a long time that means the stationary is strong.
- **Weak stationarity** the main characteristic of the weak stationarity is it has a constant mean

[3]is the incorrect rejection of a true null hypothesis. Null hypothesis or H0 is the commonly accepted fact that researchers work to reject.

and covariance and the correlation never change over time.

- **N-th order stationarity** when there is many moment invariant to time :

- 1-order stationarity: constant mean, the other statistics can change like variance.
- 2-order stationarity: constant variance, mean, autocovariance.
- 3-order stationarity: constant skew (deviation).
- 4-order stationarity: constant kurtosis.

1.2.6 Time series analysis

Time series analysis is a way to analyze a time series. Behind analyzing a specific time series there is many objectives such as understand how this time series work, and what factors are affecting a certain variable(s) at different points of time [12], as a result the organizations understand the underlying causes of trends and why these trends occur, thus putting them in a position to better predict the future. There are many models to analyze a time series and these models include:

1.2.6.1 Classification

Classification is to identify data and assign categories to data collections for more detailed analysis. For classification there are many technique to perform a time series the most widely known is the distance-based approach, the main idea is to classify the most similar items together by measuring the distance between items, more the distance is smaller, more items are identical. There are many algorithm based on distance approach like the famous KNN, dynamic time warping (DTW) is also another algorithm that is used for calculate the remoteness between two time series. The association between DTW and 1-NN has been the "gold standard" for time series classification used as a comparative algorithm [13].

1.2.6.2 Curve fitting

Curve fitting is a kind of representation of a data spread by a "best fitting" function (curve) along the entire range [14]. This will allow studying perfectly the relationships between variables and then make a better prediction in the future. When the distance between the line and the data point is minimum, the line is said to be the best fit. The goodness of a line is identifying by:

- Sum of square of errors: is the summation squared between the raw data point and the prediction data point.

- R-square: it is defined how successful the fit is in explaining the variation of the data. The value of R-square is defined how better the fit is and it range between 0 (the worst possible fit for a data set) and 1 (the best possible fit for a data set). This is a simple example (Figure 1.12).

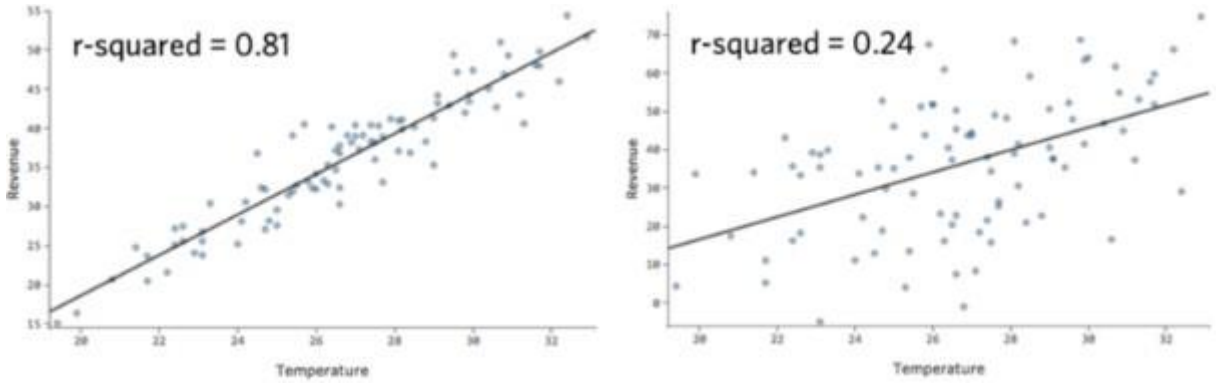

Figure 1.10: R-squared

- Adjusted R-square: as the name signifies the adjusted R-square is a modified version of the R-square. It allows knowing how the model fit a curve or a line due to the increase and decrease of the R-square, if there increasingly unnecessary use of variables the R-square will decrease, and more necessary variables are appended to the model, more R-square increase [15]. The following (Figure1.13) showing an example about the adjusted R- square.

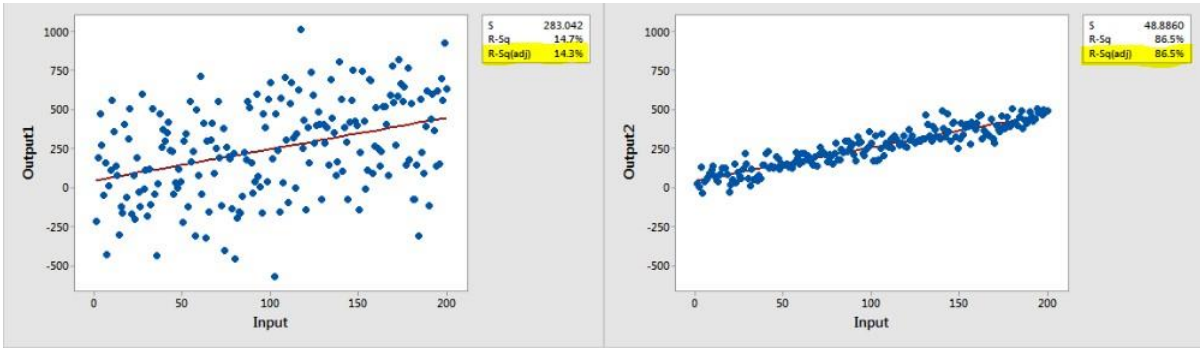

Figure 1.11: The adjusted R-square

- Root mean square error: is the standard deviation[4] of the prediction errors, the root

[4] The standard deviation shows how much the data is spread out around the mean or average: specifically, it shows if the scores are close or are lots of scores way above (or way below) the average score.

mean square error or RMSE shows how concentrated the data is around the line of best fit [16]. RMSE is considered as the best error metric for numerical predictions. This is the formula of an RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{n}^{i=1}(S_i - O_i)^2} \quad (1.1)$$

[17] where:

- S_i predicted values of a variable.
- O_i are the observations.
- n is the number of observations available for analysis.

1.2.6.3 Segmentation

Segmentation is to split the data into groups or clusters; as a result the complexity of the data is reduced. That allows conducting an analysis of the data stored database easier. There are many segmentation techniques in machine learning [18] such as:

1. K-means: in K-means algorithm a number of cluster must be initialized and so the centroids of each cluster. Each cluster has similar examples and the similarity is measured by the distance between them.

2. DBscan: is a type of density based clustering. The main difference between DBscan and K-means is that is easier to deal with it. It is not necessary to initialize the number of clusters to use it, instead of that the distance between values must be calculated by a function. It is also famous that the results obtained by DBscan algorithm are more reasonable that K-means results [19].

3. Support vector machines: in support vector machines, kernel functions are used to solve non-linear problems. Gaussian kernel gives a good linear separation in higher dimension; after finding smallest sphere that encloses the image of the data. This sphere is then mapped back to data space where it forms a set of contours that enclose the data points. The data points are then interpreted as the cluster boundaries. As the width parameter of the Gaussian kernel is decreased, the number of disconnected contours in data space increases, leading to an increasing number of clusters, and further segmentation.

1.2.6.4 Descriptive analysis

Descriptive analysis is one of analysis of data's type that gives a general conclusion about the data point and it's performing further statistical analysis. In descriptive analysis there are methods that are useful for a single variable at a time and others for data on multiple variables.

The methods that analyze a data with single variable are:

- Measures of Frequency: the main objective is to give a percent in order to know how frequently a certain event or response is probable to happen.
- Measures of Central Tendency: is defined as statistical measures that represent a single

value of the data, the central tendency can found by mean, median and mode [20].

- Measures of Dispersion: measures of dispersion describe the propagation of score in a distribution [21]. Range, variance, standard deviation, skewness and IQR are examples of measures of dispersion that help understanding the data better [22].
- Measures of Position: it is about determine the position of a value and its reaction to other

values [23].

The methods that are for multivariate are the same as bivariate analysis:

- Contingency table: analyst calls it a "two-way-table" it is used to understand the relationship between categorical variables [24].
- Scatter plots: is a chart that makes the ability to see the relationship between variables.

1.3 Time series forecasting

Time series forecasting is based on time series analysis, because forecasting is one of the objectives of time series analysis. Time series forecasting use past information that are composed of many items, those data are collected during a period of time, then a different technics and algorithms are used in order to give a general idea of what could happen in the future, based on the idea that what has happened in the past is in some way similar to what could happen in future. Those technics and algorithms are summed up in a concept named modeling fit on historical data. A model is generally represented as graph which contains the ancient values as input compared to time. The plot of time series forecasting like any plot of any time series shows the trend, seasonality and irregularity, those information are used to generally choose which algorithm to use for predictive modeling [25].

1.3.1 Models of time series forecasting

There are three types of models that can model a time series forecasting:

1.3.1.1 Statistical model

Statistical models are based on mathematics and statistical supposition in order to create prediction from data, the most popular statistical models are:

- **Auto-Regressive Integrated Moving Average**

ARIMA is an ARMA model with a preprocessing step included in the model, this

preprocessing step is differencing (integrating)[26]. ARIMA also known as a well-suited

baseline for time series, it is a model combined of two others models Auto Regressive (AR) and Moving Average (MA) with the additive Integrated (I). The integrated represent the number of transformation needed to become a stationary time series. The first application of auto regressive models back to 1920s and 1930s because of the work of Yule and J. Walker [27], auto regressive models are for forecasting variable of interest using a linear combination of past values of the variable. The term auto-regression indicates that it is a regression of the variable against itself [4]. In other hand in moving average the forecasts correspond to a linear combination of past forecast errors, the moving average model is always stationary contrary to the auto regressive model.

ARIMA has evolved to become more appropriate with stationarity and to use external data in our forecast that called exogenous variables. Those new models are named SARIMA and SARIMAX, SARIMA model is about making the possibility to differencing data by seasonal frequency, or by non-seasonal differencing, SARIMAX made the model much quicker by the addition of seasonal effects and exogenous variables into account for preparing the actual forecast [28].

- **Exponential smoothing**

Exponential smoothing has been around since 1950s [4], this model is widely use in forecasting. However exponential smoothing is based on weights and old time series data that means the higher association weights are due to the past observations. Exponential smoothing average has many types such as single exponential smoothing average (SES), double and triples exponential smoothing (DES) and (TES). Single exponential smoothing or simple exponential smoothing is suitable for forecasting data with no trend or seasonal pattern that are clearly observed.

In [4], in exponential smoothing section, simple exponential smoothing subsection there is a clearly example of how a time series without clear trend or seasonal look like. Dou- ble exponential smoothing models are weighted averages trend and levels without seasonality, the (Figure1.14) bellow shows how the plot of a time series with DES look like, also how the trends and levels are noticeable the figure is from a research in industry domain sited as [29]. Triple exponential smoothing also called Holt Winters method for short-term predictions from a sample of periodic historical data. A triple exponential smoothing model subsumes single and double exponential smoothing by the configuration of the nature of the trend and of the seasonality, as well as any dampening of the trend [30].

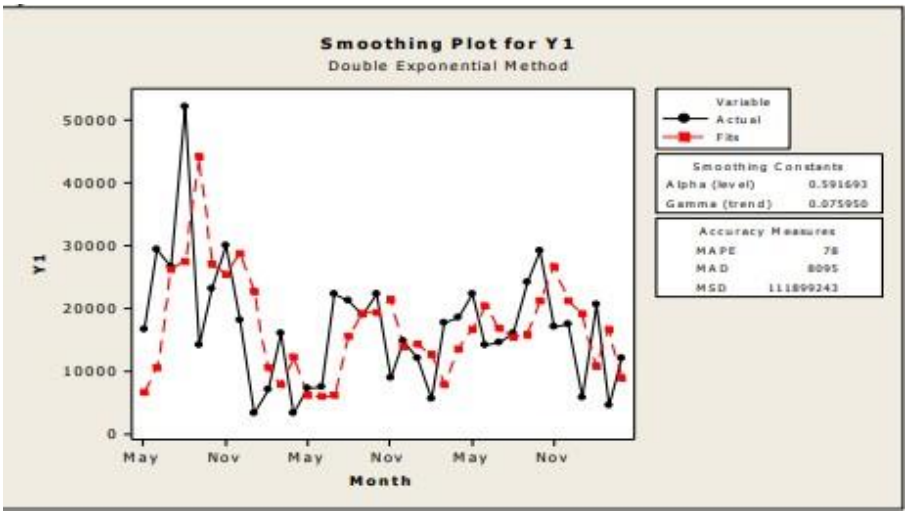

Figure 1.12: Double exponential method

1.3.1.2 Machine learning models

Machine learning or automatic learning is an AI technique that allows computers to detect trends in a knowledge history, to derive a prediction model for the future [31]. Machine learning algorithms autonomously learn to perform a task or make predictions from data and improve their performance over time; there are a variety of machine learning's algorithms with different purpose. Regression algorithms allow understanding the relationships between data; there are also algorithms for classification and for making decisions. The following are few examples of machine learning models:

- **Classification and Regression Tree (CART)**

Classification and regression tree is a very important algorithm in machine learning. A decision tree is a technique used for prediction, it is employed to progress from observations about an item that is represented by branches and finally concludes at the item's target value, which is represented in the leaves [32].

The general structure of the decision tree is generally composed of root node, internal node and leaf node. First, the root node is the first node on which the data is trained. Second, the internal node is the point where sub-group is split to a new sub-group or leaf nodes; this is called a decision node as well because this is where node splits further based on the best attribute of the sub-group. Finally, leaf node is the final node from any internal node that holds the decision. (Figure 1.14) bellow shows the structure of the decision tree:

CART algorithm can be used for both classification and regression. Classification is for finding the class into which the target variable belongs, when the target variable is continuous.

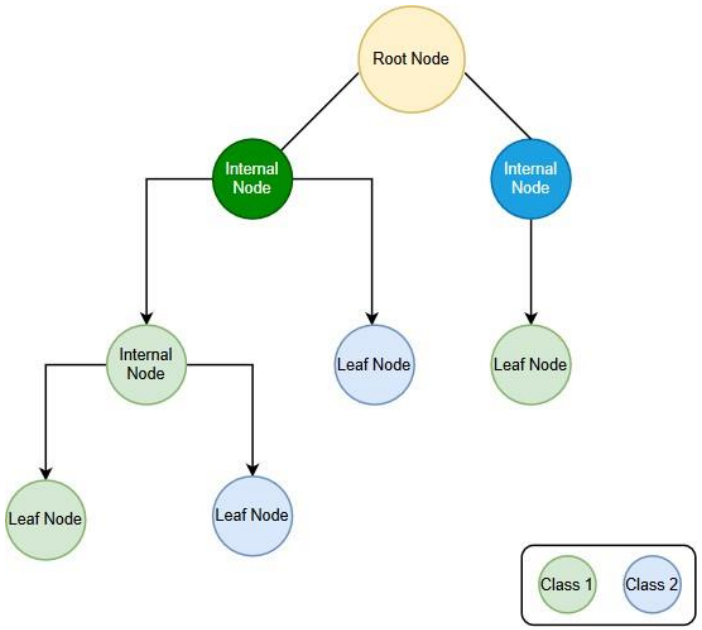

Figure 1.13: Structure of decision tree

Regression is used to forecast the value of a continuous variable [32]. The process is to split recursively from the root node which contains the data set, the root node is split in two, then split the subsets then sub-subsets, it finds further splitting will not give any pure sub-nodes or maximum number of leaves in a growing tree or termed it as a Tree pruning [33].

- **Support Vector Regression**

Support vector machine provides better support for forecasting time series from nonlinear systems. SVM is a machine-learning technique based on statistics; SVM performs well in forecasting time series. Support Vector Regression (SVR) is a regression technique based on SVM [34]. The principle of support vector machine is finding a line or hyperplane that separates the different classes in the plot. Then it classifies the new point depending on whether it lies on the positive or negative side of the hyperplane depending on the classes to predict. In other words there is what it called decision boundary, a decision boundary can be thought of as a demarcation line that has two sides, positive side and negative side, which the examples are classified as either positive or negative. The objective is, to basically consider the points that are within the decision boundary line. The fit line is the hyperplane that has a maximum number of points [35]. The following (Figure 1.16) is a demonstration of this explication:

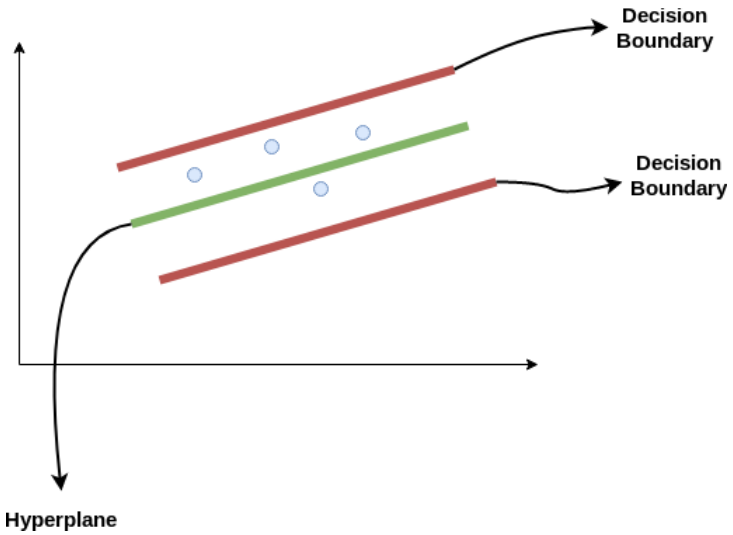

Figure 1.14: Demonstration of decision boundary

1.3.1.3 Deep learning method

The very famous methods in deep learning for time series forecasting are those based on Artificial Neural Networks (ANN), for their capacity and flexibility to map non-linear relationships from data given their deep structure [36]. Another advantage of ANNs is that they reduce preprocessing effort because they can extract temporal patterns automatically without any theoretical assumptions on the data distribution [37]. Based on ANN researchers have developed many architecture beginning with Multi-Layer Perceptron (MLP), then the more sophisticated methods, situated bellow:

- **Multi-Layer Perceptron (MLP)**

Multi-Layer Perceptron is a type of artificial neural network where the architecture is such that all the nodes, or neurons, in one layer are connected to the neurons in the next layer that what it called a fully connected neural network. MLP is structured on three principle block, one input layer, one or more hidden layers of perceptron and one input layer. The input layer simply distributes the input features to the first hidden layer. The first hidden layer receives as inputs the features distributed by the input layer. The other hidden layers receive as inputs the output of each perceptron from the previous layer. The output layer of perceptron, receive as inputs the output of each perceptron of the last hidden layer [38]. Even that Multi-Layer Perceptron is a very basic model it is widely chosen for studies, and the following (Table 1.1) represent the results of few recent studies using MLP.

Refs	Year	Technique	Outperforms	Application	Description
[39]	2016	Ensemble MLP	Statistical methods	NN3 and NN5 competitions	Two-layers ensemble. The first layer finds an appropriate lag, and the second layer employs the obtained lag for forecasting.
[40]	2018	Parallelized MLPs	Linear Regression, Gradient-Boosted Trees, Random Forest	Electricity demand	Split the problem into several forecasting sub problems to predict many samples simulate
[41]	2020	MLP/MLP-WOA/MLP-GA	Comparison	predicting wind speed	The results demonstrated that the hybrid MLP-WOA model had superior capabilities in increasing the accuracy of standalone MLP models.

Table 1.3: Results of some studies with MLP

- **Convolutional network**

Convolutional neural network CNN are developed in1980 [42]. CNNs have solved many classification tasks such as object recognition, speech recognition and natural language processing [37]. The layers of a CNN consist of an input layer, an output layer, and a hidden layer which includes several convolutional layers, grouping layers, fully connected layers, and normalization layers. Removing limitations and increasing efficiency for image processing results in a much more efficient, easier to train, and specialized system for image processing and natural language processing.

- **Echo state network (ESN)**

Echo state network are a kind of recurrent neural network (RNN). Recurrent neural network will be explained with details in chapter two. ESN are based on Reservoir Computing (RC), which simplifies the training procedure of traditional RNNs. Reservoir Computing's input signal is connected to a non-trainable and random dynamical system (the reservoir), thus creating a higher dimension representation (embedding). This embedding is then connected to the desired output via trainable units [43]. (Figure 1.17) is a modelization of the ESN architecture. Echo state network is a very powerful neural network for time series forecasting comparing to MLP and statistical methods when modeling chaotic time series data and many studies improve that [37].

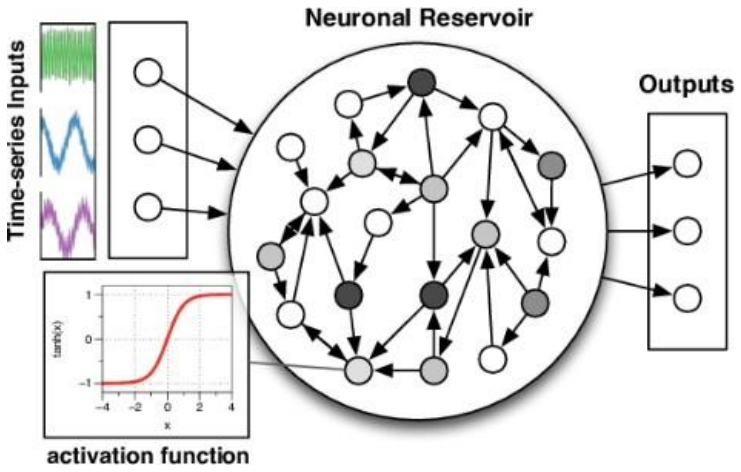

Figure 1.15: Architecture of ESN

- **Gated recurrent units network**

Gated recurrent units network (GRU) is another kind of RNNs, it was introduced by Kyunghyun Cho et al in 2014. GRU is a simplification of the LSTMs which will be explained in chapter two. The majority distinction is that GRU is composed of two gates that are reset and update while LSTM has three gates that are input, output and forget gate. GRU is less complex than LSTM because it has less number of gates, so it is faster in training [44].

1.3.2 Taxonomy of time series forecasting problems

When there is a time series forecasting problem, the perfect choice is to take the time necessary to solve it, because every decision have an impact on the results. The following framework is proposed by [45] which is very useful for finding the type of time series data problem and even help choosing an algorithm for solving the problem. The framework is about finding the answers of the following question:

1.3.2.1 What are the inputs and outputs for a forecast?

The input data is not the data used to train the model. It is the data used to make one forecast, for example the last seven days of sales data to forecast the next one day of sales data. When the expected result is forecasting, generally the input data is a collection of past observations so the inputs will be the historical data provided to the model in order to make a single forecast, and the outputs are prediction or forecast for a future time step beyond the data provided as input.

1.3.2.2 What are the endogenous and exogenous variables?

The main goal of this question is to identify easily overlooked exogenous data or even engineered features that may improve the model. The endogenous variables are input variables that are influenced by other variables in the system and on which the output variable depends. Exogenous variables are input variables that are not influenced by other variables in the system and on which the output variable depends. Some of the exogenous variables characteristic is that are fixed when they enter the model, they influence endogenous variables in the model and they are not determined or explained by the model. Endogenous variables can be explained easily with an example like in agriculture; the amount of crop yields is endogenous because it is dependent on many other variables, such as the weather, soil fertility, water availability, pests, and diseases.

1.3.2.1 Are the time series variables unstructured or structured?

When the time series data is represented in a plot, we can identify if the time series variables have a pattern or not. After taking a look to the plot, if there is no obvious systematic time-dependent pattern the data is considered unstructured, but if there is systematic time-dependent patterns that is structured data. In other word a time series that is a time series may have obvious patterns, such as a trend or seasonal cycles is considered as structured, if not it is unstructured data.

1.3.2.2 Is it a regression or classification predictive modeling problem? What are some alternate ways to frame the time series forecasting problem?

A time series forecasting problem that composed of one or more future countable values is a regression type predictive modeling problem. Regression predictive modeling problems are those where a quantity is predicted. A quantity is a countable value; for example a price, a count, a volume, and so on. Classification predictive modeling problems are those where a category is predicted. A category is a label from a small well-defined set of labels; for example hot, cold, up, down, and buy, sell are categories. A time series forecasting problem in which classify input time series data is the goal that it is a classification type predictive modeling problem.

Some problems, like predicting an ordinal value, can be framed as either classification or regression. There is such flexibility between classification and regression, a classification problem can be a regression problem and a regression problem can also be a classification problem. It is possible to simplify a time series forecasting problem with a reframing.

1.3.2.3 Is the problem a univariate or multivariate time series ?

A univariate and multivariate time series data is well defined previously. However the inputs and outputs may be not symmetrical. For example, the data can have multiple variables as input to the model and only be interested in predicting one of the variables as output. In this case, there is an assumption in the model that the multiple input variables aid and are required in predicting the single output variable. To sum up in a time series problem it can happen that what is needed as input is a univariate and a multivariate output or multivariate input and univariate output.

1.3.2.4 Does the problem require a single-step or a multi-step forecast?

A forecast problem that requires a prediction of the next time step, is called a one-step forecast model, and a forecast problem that requires a prediction of more than one time step is called a multi-step forecast model for example considering an observed temperature data over the last 7 days, if predicting the day 8 is request, that means it is a one or single-step forecasting. Whereas if the prediction of two or next five days is requested, that means that it is a multi-step forecasting.

1.3.2.1 Does the problem require a static or a dynamically updated model?

There are two cases when developing a model. The first is that the model can be use repeatedly to make predictions. Given that the model is not updated or changed between forecasts, and this is what it called a statically model. The second one is that the data may receive new observations prior to making a subsequent forecast that could be used to create a new model or update the existing model, and simply that is called dynamic updated model. In other words a static model is a forecast model is fit once and used to make predictions, and dynamic model is a forecast model is fit on newly available data prior to each prediction.

1.3.2.2 Are the observations contiguous or discontiguous?

A time series contiguous is composed of observations that are made uniform over time. Many time series problems have contiguous observations, such as one observation each hour, day, month or year. Discontiguous time series is composed of observations that are not uniform over time. The lack of uniformity of the observations may be caused by missing or corrupt values.

Answering this question gives an idea of the fundamental difficulty of the forecasting problem, thus make the ability to try to solve this problem with more awareness about the state of difficulty of the forecasting problem

1.3.3 Some real world use cases of time series forecasting

1.3.3.1 Veritas

Vertias is a plateform that uses artificial intelligence (AI) and machine learning (ML) to deliver services. This company is operating in fifty eight countries, with more than eighty thousand business and enterprise customers located all around the world. Veritas needed to automate storage forecasting in its SaaS platform.

Previously, Veritas measured problems through their auto-support capabilities. They had years of Veritas AutoSupport information and hundreds of millions of telemetry data points from more than ten thousand Veritas appliances. But they didn't have any analytics for forecasting to enable preventing problems from happening.

A huge time series data is collected from Veritas ' Auto Support capabilities, and so that permit to make forecasting for a multitude of use cases from application performance optimization to workload anomaly detection. Yet the challenge was to automate a historically manual process handcrafted for the analysis of a single data series of just tens of data points to large-scale processing of thousands of time series and millions of data points. For the full use case study see [46].

1.3.3.2 Playtech

Playtech is a company that is a gaming software company founded in 1999. Playtech deliver many online games and it is listed on the London Stock Exchange and is a constituent of the FTSE 250 Index[5]. Playtech has five thousand employees, offices in seventeen countries, one hundred forty global licensees, and operates in twenty regulated jurisdictions.

Playtech needed to build a predictive monitoring and alerting system on time series data, for early detection of outages and security breaches. Their existing Hewlett Packard Service Health Analyzer solution was not delivering accurate and early predictions. Their existing Hewlett Packard Service Health Analyzer solution was not delivering accurate and early predictions. They wanted a monitoring and anomaly detection solution designed for distributed systems. Monitoring distributed systems, as they found by analyzing successful and not-so-successful monitoring projects, is not a trivial task. It involves non-obvious obstacles and picking the right solution for various monitoring tasks.

Playtech decided to build yet Another Alert System using InfluxDB, Alerta, and Grafana. For the full case study see [47].

[5] The FTSE 250 is a stock market index calculated by the FTSE Group. Complementing the main FTSE 100 index, it is made up of the 250 British companies whose capitalization is between 101^{e} and 350^{e} place.

1.4 Conclusion

This chapter gives a fully introduction about time series. It begin with a simple definition of time series then go deeply in details, after that it shows the importance of a time series analysis and how time series forecasting is based on time series analysis. In time series forecasting section there are the important things to consider like the models that deal win time series forecasting and how to understand then solve the time series forecasting problem. Finally, a few real world examples that use time series data to solve their problems are presented. The next chapter will be about recurrent neural networks and LSTM.

Chapter 2

Long Short-Term Memory (LSTM)

2.1 Introduction

Neural networks are inspired by how the neuron works in human brain. In 1943, Walter Pitts a mathematician, and Warren McCulloch, a neurophysiologist, and Donald Hebb, a Canadian psychologist, published a paper suggesting neural networks as a way to imitate human brains [48]. In 1969, the Perceptrons book was published by Minsky and Papert, this book shows the limitations of a single layer neural network, which were the type of artificial neural networks being used at that time. In 1986, the Parallel Distributed Processing texts was published, which contain the backpropagation learning algorithm, which enabled multiple layers of perceptrons to be trained, and thus introduced the hidden layer(s) to artificial neural networks, and was the birth of MLPs (multiple layered perceptrons) that was introduced in chapter one. Since then, many research and development efforts in artificial neural networks took place [49].

2.2 Neural Network

A neural network is made up of vertically stacked components called Layers; those layers are composed of small individual units called neurons. As explained before a neuron network is an imitation of biological neurons that receive signal from other neurons, make some processing, and produces an output [50]. Neural networks are composed of three layers: input layer, output layer and a hidden layer[6]. Neural networks are basically working like follows: when a neuron receives signals as inputs from the preceding layers of the model, it adds up those signals and multiplied it by the corresponding weights and passes them on to an activation function [51].

[6] The input layer is the first layers in the neural network, the output layer is the last layer and the hidden layer is all the layers between them [48].

Weights in Neuron network represent the connection between two neurons and decide how much influence the input will have on the output, if the weights are near to zero that means that changing input values don't change the output, if the weights are negative that means increasing this input will decrease the output [52]. Activation function decide which neuron will be relevented in the process of prediction [53], without an activation function the neural network is just a simple linear regression model, a non-linear activation function make transformations to the input making it capable to learn and perform more complex tasks [54], in the following more details about that will be presented.

2.3 Training neural network

Deep learning is a part of machine learning, however the word *learning* in machine worlds means mapping inputs to target. It is known that machine learning have a simple structure such as a decision tree that is mentioned in chapter one, further more deep learning is based on more complex structure that is artificial neural networks. Now, the very first neuron network *the perceptron* have only one input, one output and one hidden layer, but after that if the neuron have more than three hidden layers is qualified as deep neuron network. Deep neuron network's models has a lots of hidden layers and a very complicated architecture in order to making sense and extract knowledge from complicated data. When training a deep neural network, each layer trains from the features outputs of the previous layer and more going deeply in the network the more neurons identify better the features since they aggregate and recombine features from the previous layer, that why deep learning networks are able to handle very large data sets [55].

Like sited before weights are very essential to know the importance of the inputs. First, a neural network is trained on the training set, it is initialized with a set of weights. These weights are then optimized during the training period, when talking about optimizing weights in a network, it is really about adjusting the weights on synapses, and synapses are like roads in a neural network, they connect inputs to neurons, neurons to neurons, and neurons to outputs. After that, those weights are transmitted between neurons, the weights are applied to the inputs along with an additional value "the bias". The bias is simply a constant which controls when to activate the activation function, adding the bias, reduces the variance and hence introduce flexibility and better generalization to the network [56].

In neural network, there is a need to know how the prediction of the model, deviate from the actual prediction and this is what the loss function do. The loss function is used in both back-propagation and gradient descent in order to optimize the model, this will be more explained next.

To resume, the process of training is :

1. Multiply the data to weights and add bias
2. Pass the result to a non- linear activation function which calculate a specific output, generally is between 0 and 1.
3. The output produced with the activation function is then compared with expected output, with the cost function.
4. After receiving information about how far the produced output from the expected one, the backpropagation process is applied in order to adjust weight and bias
5. Repeat the process with the updated weights and biases until the produced output is reasonably close to the expected output.

With deep learning, the intervention of human is less than in machine learning methods, deep learning however needs more data point in order to provide high level results. This characteristic allows neural networks generate the unstructured data with more performance and can with some technique the data can be adapted to changes, and that what is known as adaptation in neural network. There is three strategies used for adaptation [57]:

- Structural adaptation: the goal is to find the optimal network architecture, for example a robot that is not able to classify obstacles in the path. That robot will constantly keep on colliding with obstacles while moving around an environment. And, that's where adaptive neural networks who can adapt to structure comes into handy. They can help robots in the real-time classification of obstacles and prevent collisions from them.
- Functional adaptation: this adaptation is for adapting the activation function to reduce errors of outputs, it keeps on adapting and changing the slope of activation functions of learning models until it reaches the optimal slope, the most known algorithm is the gradient descent.
- Parameter adaptation: parameter adaptation is adapting to changing input data functions, like weights and biases while training. Neural networks can get knowledge from new weight inputs without losing knowledge gained from previous inputs with minimum loss in accuracy. The most known parameter adaptation algorithm is the backpropagation.

2.3.1 Backpropagation and gradient descent

2.3.1.1 Definition

The gradient descent is a method used to minimize the cost function. It is process that occurs in the backpropagation phase where the goal is to continuously resample the gradient of the model's parameter in the opposite direction based on the weight, updating consistently until we reach the global minimum [58], when said the opposite direction that means the opposite of a normal feed-forward direction which is from the input to output like shows the (Figure 2.1). In other word, the backpropagation and gradient descent are two different methods that form a powerful combination in the learning process of neural networks.

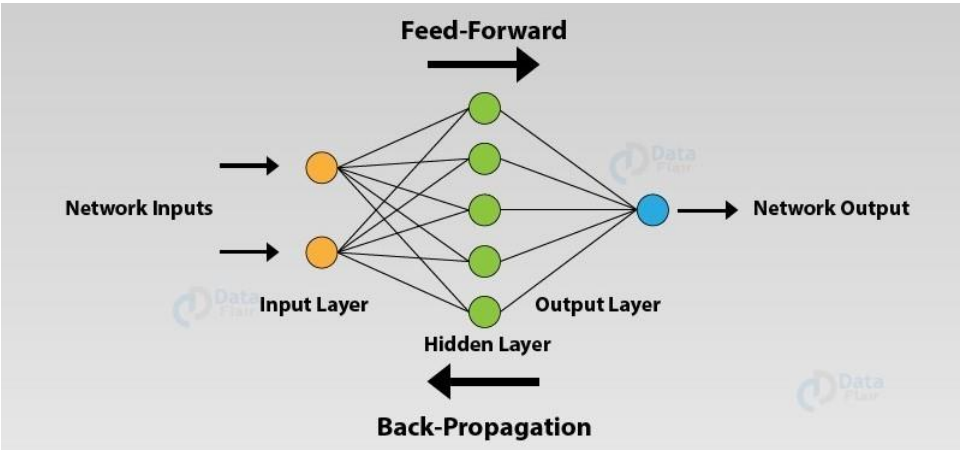

Figure 2.1: Backpropagation

Note that the gradient is the derivative of the Error with respect to the weight, in order to op- timize weight to minimize error, because moving the weights more towards the positive x-axis, involve optimizing the loss function and achieve minimum value, the mathematical representation is :

$$Gradient = \frac{dE}{dW} \qquad (2.1)$$

Where E is the error and w is the weight.

Now the negative of the Gradient shows the directions along which the weights should be moved in order to optimize the loss function. However, if the loss increases with an increase in weight so Gradient will be positive that where is the point B showing in the (Figure 2.2), and if the loss decreases with an increase in weight so gradient will be negative, that where is the point A. So according to that from A we need to go move towards positive x-axis and the gradient is negative. From point B, we need to move towards negative x-axis but the gradient is positive.

The gradient descent use the learning rate or what is known as step size or the Alpha, and that is the size of the steps that are taken to reach the minimum, after knowing which direction to take it is important to know how much moving the weights, with taking steps carefully, if the steps are too large it would overstep the optimal value (the optimal value can be zero or very close to zero). If it is very low it takes tiny steps and takes a lot of steps to optimize [59]. Step distance and the update weight formula are:

$$dx = alpha * |\frac{dE}{dW}| \quad (2.2)$$

Where dx is the distance and alpha is the learning rate.

$$\hat{W} = w - alpha * |\frac{dE}{dW}| \quad (2.3)$$

Where $\hat{W}$ is the updated weight.

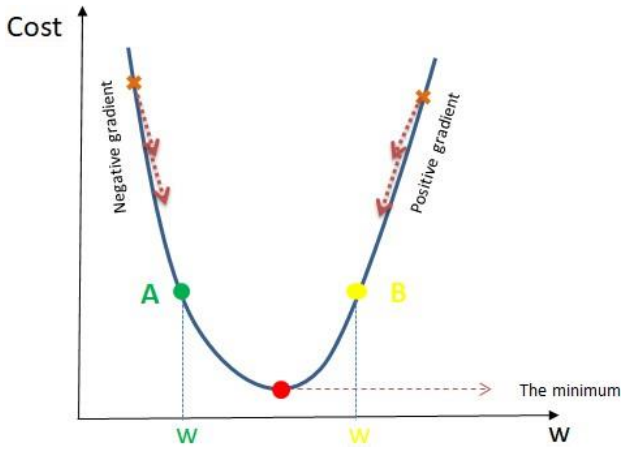

Figure 2.2: Gradient descent

2.3.1.2 Challenges with backpropagation and gradient descent

The gradient descent has some challenges that include [60]:

- **Local minima and saddle points**

When the slope of the cost function is at or close to zero, the model stops learning. A few scenarios beyond the global minimum can also yield this slope, which are local minima and saddle points. Local minima mimic the shape of a global minimum, where the slope of the cost function increases on either side of the current point. However, with saddle points, the negative gradient only exists on one side of the point, reaching a local maximum on one side and a local minimum on the other. Its name inspired by that of a horse's saddle (see Figure 2.3).

- **Vanishing and Exploding Gradients**

The vanishing and exploding gradient descent are two problems that occur in recurrent neural network. Vanishing gradient occurs when the gradient is too small. As we move backwardsduring backpropagation, the gradient continues to become smaller, causing the earlier layers in the network to learn more slowly than later layers. When this happens, the weight parame- ters update until they become insignificant. Exploding gradient is the opposite of the vanishing gradient, this happens when the gradient is too large, creating an unstable model. In this case, the model weights will grow too large. this will be explained in *Problems of recurrent neural network* section.

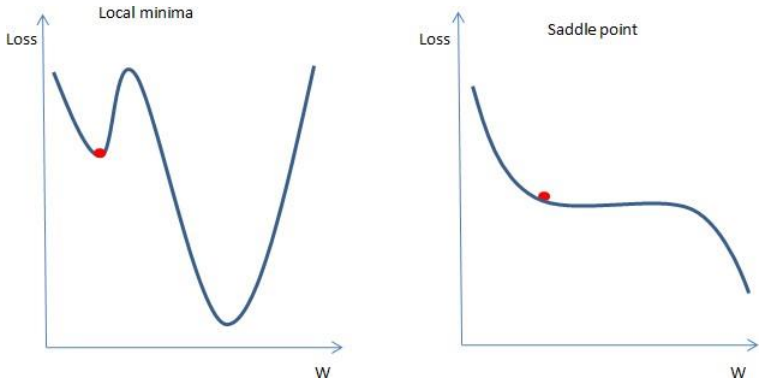

Figure 2.3: Local minima and saddle points

2.3.2 Activation function

The activation function is very important part of artificial neuron networks. As the name signify the activation function decide which neuron will be *activated* to the next layer, in other word it convert input signal into output signal. An artificial neuron network without activation function has limits since the output signal will simply be a linear function. A linear function is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data. Without activation function the model can't learn and model other sophisticated data like images, videos and audio. That why as sited before it is important to use the non-linear activation function which have a degree more than one and they have a curvature, thus perform more complex tasks [61] [54].

The most popular non-linear functions are:

2.3.2.1 Sigmoid function

Sigmoid function also named logistic function, it takes any value as input but outputs are in the range of 0 to 1. If the inputs are small (more negative) the output will be near 0. The large input (positive ones), the more output are near to 1. The sigmoid/logistic activation function is used

widely in probability tasks, due to its range because as we know the probability is between 0 and 1, and this is the major reason for choosing the sigmoid function in such tasks.

The graph of sigmoid function is like shows the (Figure 2.4). This function can be represented mathematically as:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

However, the logistic function has some limitations such as vanishing gradient due to its small derivative [62].

2.3.2.2 Tangent function

The Tangent activation function also named Tangent hyperbolic function is similar to the sigmoid function; the only distinction is that the range of Tangent function is from -1 to 1 as shows the (Figure 2.4). Tangent activation function makes the ability to map the output value like strongly negative, positive or neutral. Like the sigmoid function the tangent function also have the vanishing gradient problem. It can be represented mathematically [54] like :

$$f(x) = \frac{2}{(1+e^{-2x})} - 1 \tag{2.5}$$

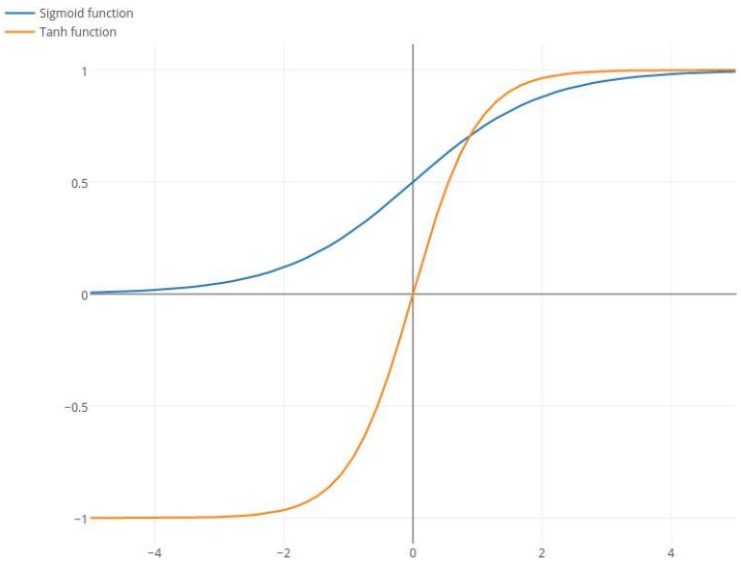

Figure 2.4: Sigmoid and Tangent activation function

2.3.2.3 Rectifier function 'ReLU'

The Rectified Linear Unit return 0 if it receives any negative input, and returns any positive value back. According to [63]this activation function is better than the sigmoid and the tangent activation functions, in other words the ReLU learn much faster then sigmoid and tangent function because it involves simpler mathematical operations [54], it doesn't have the vanishing gradient problem, but the exploding gradient, also there is dying ReLU problem and this occur when it's stuck in the negative side and always outputs 0, like represented (in the left side) in (Figure 2.5) . The equation of ReLU activation function is :

$$f(x) = max(0,x) \tag{2.6}$$

.

As a result of the dying ReLU, the network ends up with large useless inputs. This is why the leaky ReLU is a solution of the dying ReLU, the leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so. (Figure 2.5) also shows leaky ReLU (in the right side).

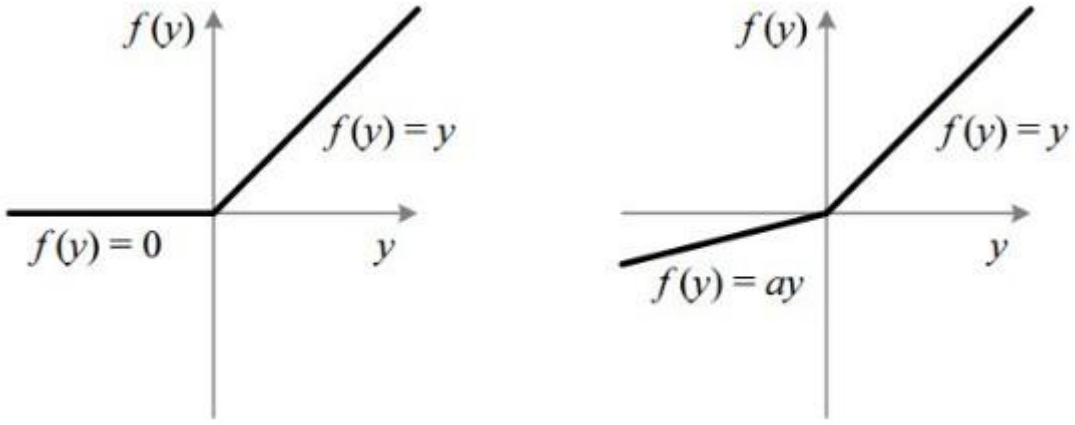

Figure 2.5: ReLU and leaky ReLU

2.3.3 Loss function

As mentioned before, decreasing the errors is very important when training a neuron network. There is some loss function mostly used :

- Mean Squared Error loss function

Mean squared error (MSE) also known as mean squared deviation (MSD) is the best choice when dealing with a regression problem. Finding the best line fit is when the mean squared error is smaller, when is equal to zero that means the model have no errors.

Mean squared error is represented mathematically:

$$MSE = \frac{\sum(y_i - \hat{y})^2}{N} \tag{2.7}$$

Where:

y_i is the i^{th} observed value.

$\hat{y}_i$ is the corresponding predicted value.

N is the number of observations.

The MSE is great for ensuring that our trained model has no outlier predictions with huge errors, since the MSE puts larger weight on these errors due to the squaring part of the function.

- Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is only slightly different in definition from the MSE, instead of take the difference between your model's predictions and the ground truth, square it, and average it out across the whole dataset like the MSE, the MAE take the difference between your model's predictions and the ground truth, apply the absolute value to that difference, and average it out across the whole dataset.

The MAE is formally defined by the following equation:

$$MAE = \frac{\sum|y_i - \hat{y}_i|}{N} \tag{2.8}$$

Where:

y_i is the i^{th} observed value.

$\hat{y}_i$ is the corresponding predicted value.

N is the number of observations.

The beauty of the MAE is that due to taking the absolute value, all of the errors will be weighted on the same linear scale. Thus, unlike the MSE, the weights won't be putting too much on the outliers and the loss function provides a generic and even measure of how well the model is performing.

- ADAM

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data . It

requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm. The momentum algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Root mean square prop or RMSprop is an adaptive learning algorithm that takes 'exponential moving average' instead of taking the cumulative sum of squared gradients.

2.4 Recurrent neural networks (RNN)

The concept of Recurrent neural networks was brought up in 1986 [64]. It is the famous gender of neural networks, the following is an overview about this neurons.

2.4.1 Definition

Recurrent neural networks is a variant of artificial neural networks, designed for time dependent data. RNN has a recurrent hidden state whose activation at each time is dependent on that of the previous time, that why RNNs can handle time series [65]. RNN also can be used for forecasting, due to the network that sees one observation at a time and can learn information about the previous observations and how relevant the observation is to forecasting [37]. RNNs are used in many application for forecasting, such as forecasting cloud datascenter [66], stock market forecasting [67], speech recognition [68], sequence study of the DNA [69], human action recognition [70]. Also recently with the past virus wave (COVID19), RNNs has been used to achieve many purposes like prediction [71], detection system [72] and analyze positive cases [73].

2.4.2 Architecture

In all neural networks the inputs and outputs are independent of one another [74]. But in some cases there are a need to remember the previous state like in the Natural Language Processing (NLP) for example in content categorization which is a linguistic-based document summary, including search and indexing, content alerts and duplication detection, the prior state must remembered in order to make the ability to detect duplications[75]. Therefore RNN has been developed to solve this problem, with a layer call hidden layer, this hidden state remember specific information about a sequence [74]. RNNs present the idea of recurrent connections, in other words it reconnects the output of a neuron in the hidden layer as a feed stream to the same

hidden layer neuron [76]. The structure of a vanilla RNN[7] has a circular shape as shows the following (Figure2.6) [77]:

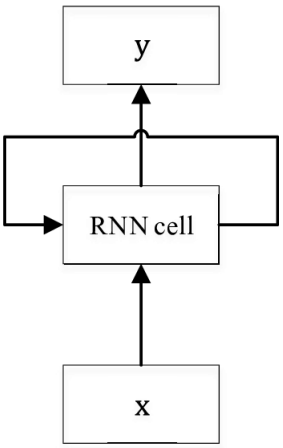

Figure 2.6: Architecture of vanilla RNN

Going deeply this (Figure2.7) is the architecture of RNN in which, the $a^{<t>}$ represent the activation, $y^{<t>}$ represent the output and $x^{<t>}$ is the input at each time step.

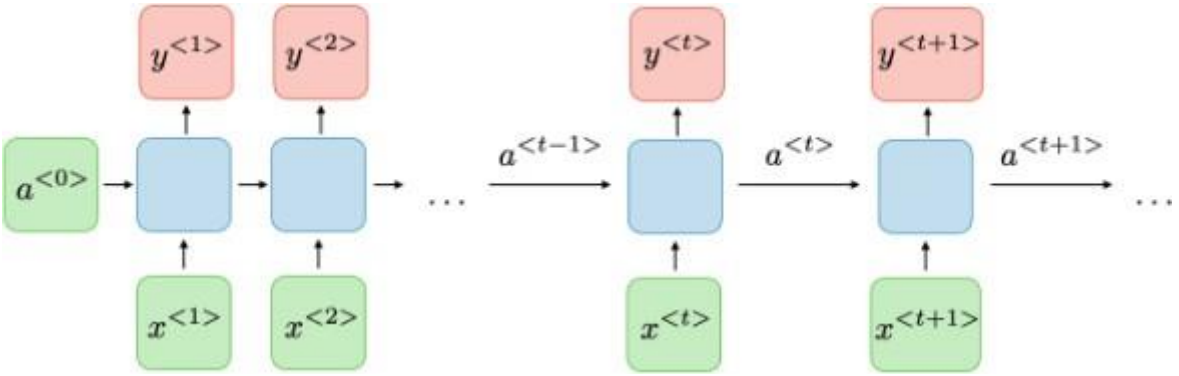

Figure 2.7: Architecture of RNNs

The RNNs really work as following, first the input layer *x* receives and processes the neural network's input, then it passing it on to the middle layer. Many hidden layers can be found in the middle layer *h* (the blue cell), each with its own activation functions, weights, and biases, those parameters are standardized by RNN in order to make sure that each hidden layer have the same characteristics. RNN also create only one hidden layer and loop over it as many times

[7] the state consists of a single hidden vector, [77]

as necessary, instead of constructing numerous hidden layers [74]. The activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as following [74]:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}.x^{<t>} + b_a) \tag{2.9}$$

$$y^{<t>} = g_2\ (W_{ya}a^{<t>} + b_y) \tag{2.10}$$

In which: W_{aa}, W_{ax}, W_{ya} are coefficients that are shared temporally, and g_1, g_2 are activation function. The following (Figure2.8) shows how all this parameters ordered in RNN cell:

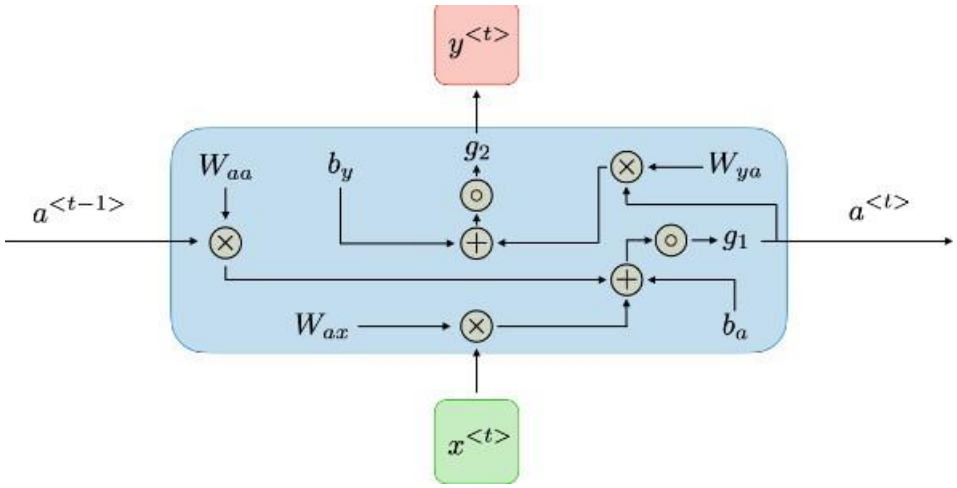

Figure 2.8: RNN cell

2.4.3 Types of RNN

Recurrent neuron network has four types:

- One-to-One: is the very basic one which has one input and one output, It has fixed input and output sizes and acts as a traditional neural network (see Figure 2.9). The One-to-One application can be found in Image Classification.

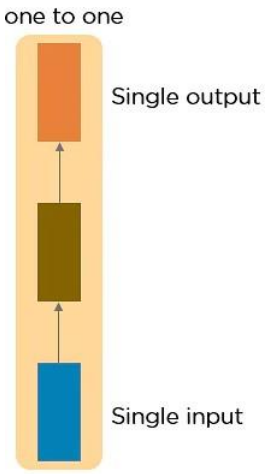

Figure 2.9: One to one

- One-to-Many: is a type of RNN that gives multiple outputs when given a single input. It takes a fixed input size and gives a sequence of data outputs (see Figure 2.10). The One-to-Many application can be found in image captioning, text generation.

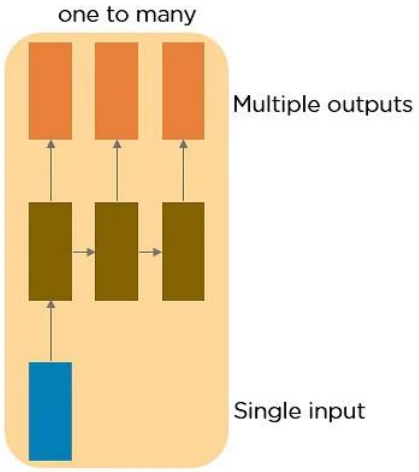

Figure 2.10: One-to-Many

- Many-to-One: this one is about giving a multiple inputs in order to have just one output (see Figure 2.11). Sentiment analysis is a good example of this kind of network where a given sentence can be classified as expressing positive or negative sentiments.

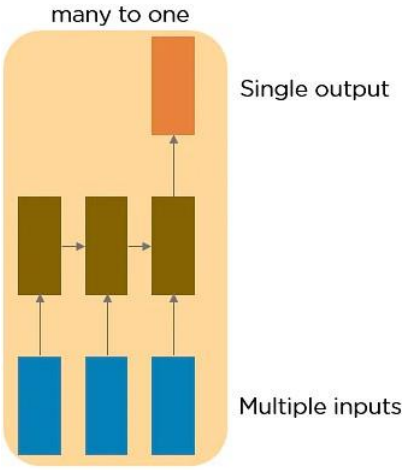

Figure 2.11: Many-to-One

- Many-to-Many: this time gives multiple inputs to get multiple outputs (see Figure 2.12), like in machine translation.

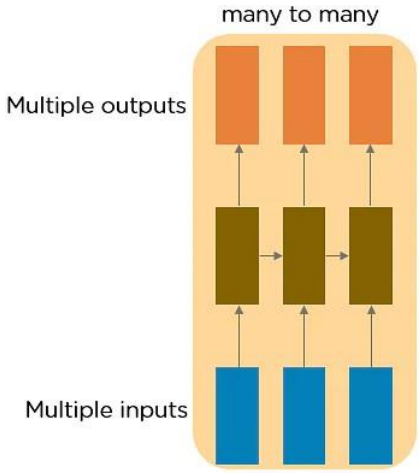

Figure 2.12: Many-to-Many

2.4.4 Problems of recurrent neural networks

In the training process, RNN use the backpropagation to update weights in order to correct the errors, and since the RNN deals with sequential data and every time we go back it's like going back in time towards the past, the process of backpropagation in neural network is called Backpropagation Through Time (BPTT). Now going deeply to understand how the (BPTT)

works the following (Figure 2.13) represent the (BPTT) process with 4 layers. If the (BPTT) begin at time t=3, then the derivative of E_3 with respect to that of S_3 is considered, x_3 is also connected to S_3. So, its derivative is also considered. Now S_3 is connected to S_2 so S_3 is depending on the value from S_2 and here derivative of S_3 with respect to S_2 is also considered. This acts as a chain rule and we accumulate all the dependency with their derivatives and use it for error calculation.

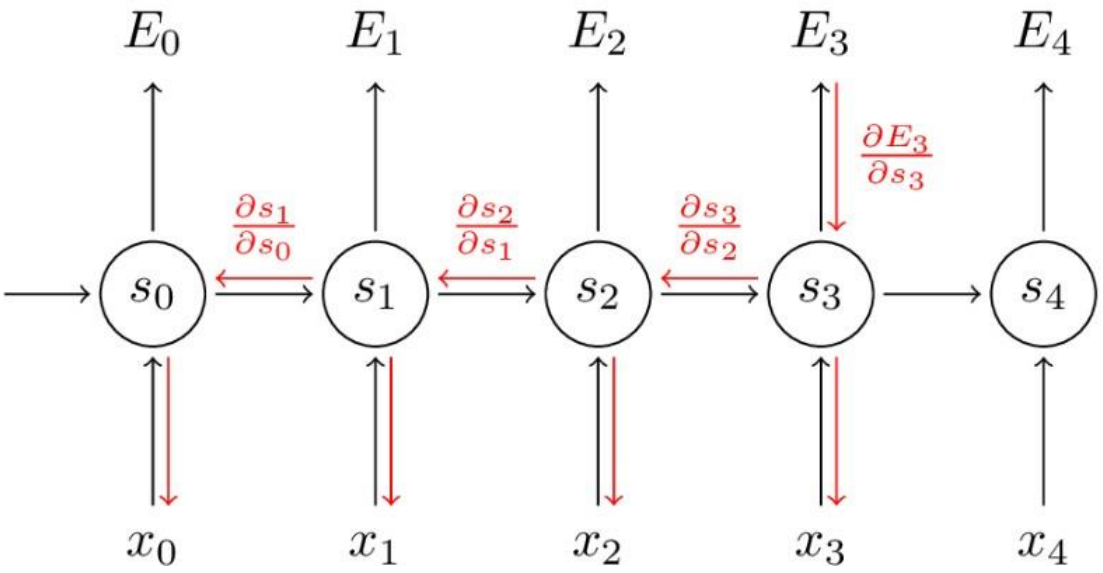

Figure 2.13: Backpropagation Through Time

The equation of gradient in E_3 is from S_3 , also S_2 is associated with S_3 and S_1 is associated with S_2, so all S_1, S_2, $S_3$are having impact on E_3. Accumulating everything gives the following equation:

$$\frac{dE_3}{dW_s} = \frac{dE_3}{dy_3}\frac{dy_3}{dS_3}\frac{dS_3}{dW_s} + \frac{dE_3}{dy_3}\frac{dy_3}{dS_3}\frac{dS_3}{dS_2}\frac{dS_2}{dW_s} + \frac{dE_3}{dy_3}\frac{dy_3}{dS_3}\frac{dS_3}{dS_2}\frac{dS_2}{dS_1}\frac{dS_1}{dW_s} \tag{2.11}$$

This is the general equation for adjusting weights in (BPTT):

$$\frac{dE_N}{dW_x} = \frac{dE_N}{dy_N}\frac{dy_N}{dS_i}\frac{dS_i}{dW_x} \tag{2.12}$$

Note that the y represents the estimated output.

Until now, this is how the Backpropagation Trough Time works, this example was just about four layers but in training process there is much more layers could even be hundreds or more, so the time of training can take t=100 or more, when calculate in such large range it ends up

with very small value such that it may end up being useless to correct the error. And this is the *vanishing gradient problem.*

In the other hand, the gradient value becomes very big and this often occurs when we initialize larger weights, If the model suffered from this issue it cannot update the weights at all. And this is the *exploding gradient problem [78].*

2.4.5 Elman recurrent neural networks ERNN

Elman recurrent network is a kind of recurrent neural network proposed by Elman in 1991[79]. ERNN is used for time series prediction [80]. This neural network is composed of four layers: the input layer, the output layer, the context layer and the hidden layer. An Elman RNN is a network which in principle is set up as a regular feedforward network, this means that all neurons in one layer are connected with all neurons in the next layer, not like other RNNs this ERNN has an additional layer called the context layer. The output of the hidden layer are input of the context layer, which store output hidden layer values of the previous time, in other words the output of each hidden neuron from the hidden neuron layer is copied into a specific neuron in the context layer, an extra input signal for all the neurons in the hidden layer came from the value of the context neuron. Therefore, the Elman network has an explicit memory of one time lag as show the (Figure2.14) [81] [82].

Elman recurrent neural network also has two problems: exploding and vanishing gradient problems.

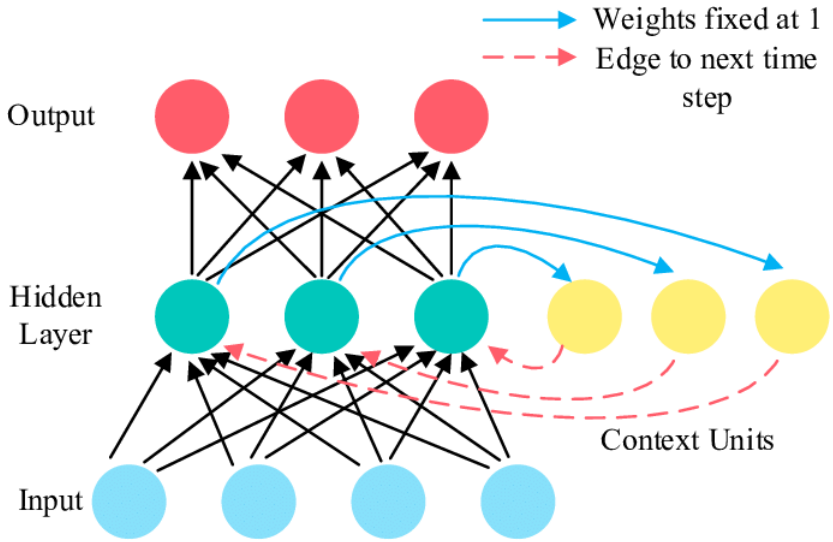

Figure 2.14: Structure of Elman RNN

2.5 Long-Short Term Memory networks (LSTM)

Long-Short Term Memory first appearance was in 1997, two man Sepp Hochreiter and Juergen Schmidhuber published a paper about long-short term memory [83] and since then it been used in many field.

2.5.1 Definition

Long-Short Term Memory neural network was developed as a solution to solve the problems of Elman RNN [84], which are vanishing and exploding gradient. LSTM is explicitly a subfield of RNN architecture, which is more stable and efficient in dealing with both long-term, as well as short-term dependency problems. It is very useful when the gap between the past and the required future values are substantial [85]. LSTM has feedback connections so that it's useful for different handle types of time series data [86]. In addition, the LSTM is used in healthcare domain for example: to control robot for heart surgery to tie knots[87], also to predict behaviors like predicting airport passenger behavior [88], even in music; the LSTM can learn to reproduce a musical chord structure [89].

In order to understand more the capabilities of the LSTM against the other RNN extension, the following example may be helpful [90]:

Let's say we have a network generating text based on some input given to us. At the start of the text, it is mentioned that the author has a "dog named Cliff". After a few other sentences where there is no mention of a pet or dog, the author brings up his pet again, and the model has to generate the next word to "However, Cliff, my pet_". As the word pet appeared right before the blank, a RNN can deduce that the next word will likely be an animal that can be kept as a pet. Due to the short memory the RNN will be able to just remember a few sentences right before "However, Cliff, my pet__", which are useless, and the sentence "dog named Cliff" has already lost. However, due to the long-memory of the LSTM, can retain the earlier information that the author has a pet dog, and this will aid the model in choosing "the dog" in blank of "However, Cliff, my pet_". We can illustrate this situation in (Figure 2.15).

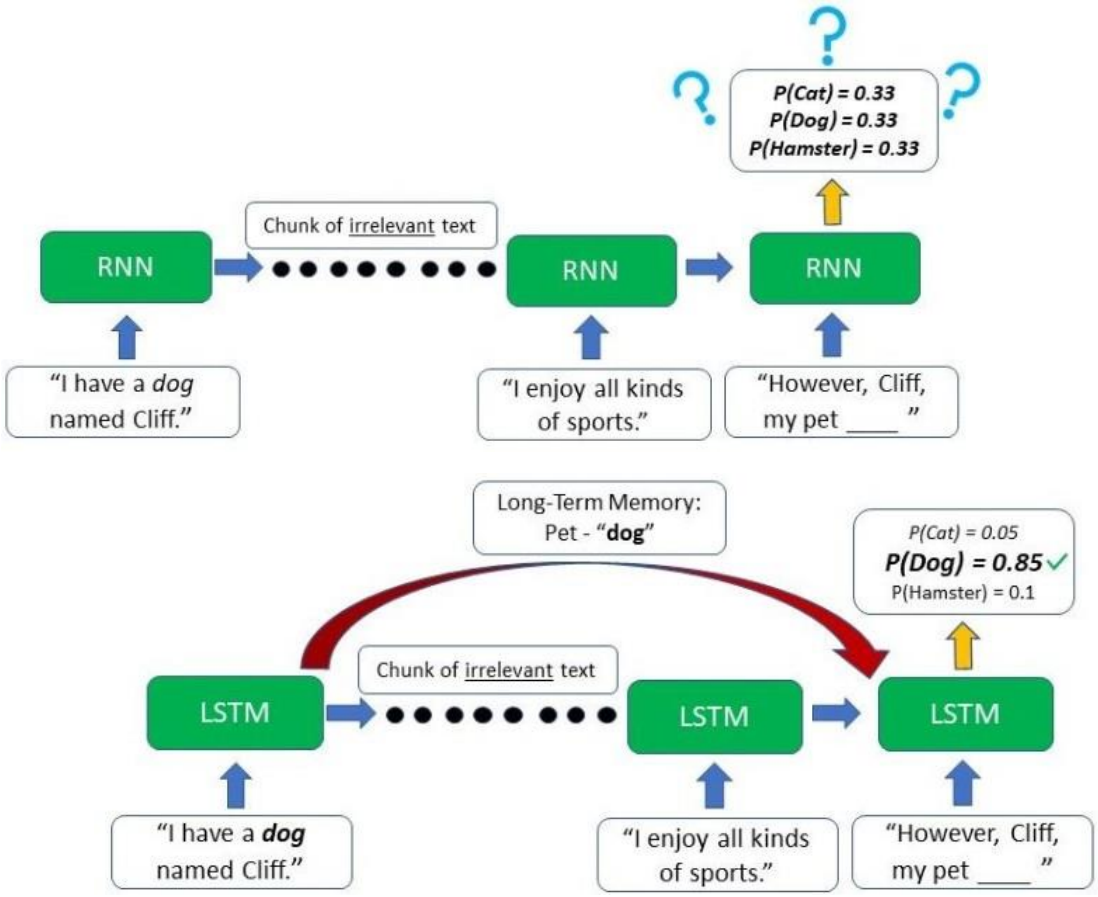

Figure 2.15: LSTM vs RNN

2.5.2 Inner working and architecture of the LSTM

While the RNN use a simple activation function (generally the tanh), the LSTM cell has more sophisticated mechanism under the cell. The output from the LSTM cell depends on three things:

- The current long-term memory of the network, known as the cell state.
- The output at the previous point in time (short term memory), known as the previous hidden state.
- The input data at the current time step.

Furthermore, LSTM use gates to decide which information can be kept or discarded, a cell contain three gates which are: the input gate, forget gate and output gate. The inner working will be explained by steps given in each step the architecture that corresponds:

2.5.2.1 Step one

First step is for the forget gate, that chooses whether the information coming from the long-term memory is to be remembered or is irrelevant and can be forgotten. The forget gate decide if the information is important or not by using the sigmoid activation function in which the short term memory (the previous hidden state) and the new inputs are passed through it, given a result between 0 and 1, if the result is 0 or near to it the information must be forgotten and if it is 1 or closer than the information must be kept. In other word, the parts of inputs which are forgotten have less weight (see Figure).

Equation that represent this process:

$$f = (H_{t-1} * W_{forget} + X_t * W_{forget} + bias_{forget}) \quad (2.13)$$

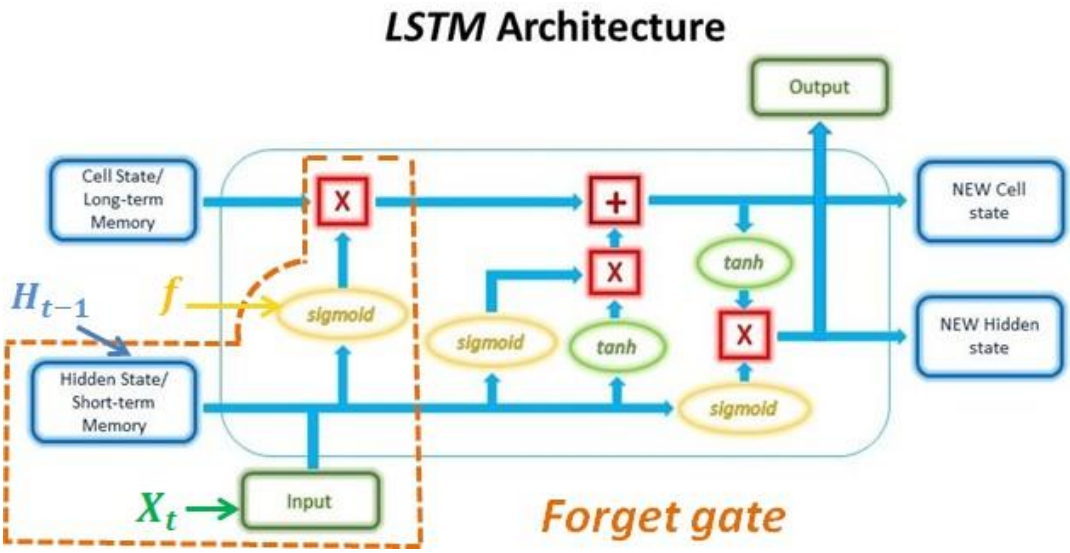

Figure 2.16: Forget gate architecture

2.5.2.2 Step two

Second step is for the input gate in which it decides what new information will be stored in the long-term memory. When the input gate receives outputs from the previous hidden state and the new input data, the process begin. It divides into two layers, the first layer is about the filtering operation in which, the information can be selected to pass or to discard. The outputs from the hidden state and the new data are passed throw a sigmoid function who return the value in rang 0 to 1, in order to make it easy to select which information will be useful, if it 0 or near to it, the information is unimportant, if it is 1 or closely to 1 this mean that the information is important and must be remembered. The output from the first layer passes to the second layer which applies the tangent function in order to regulate the network.

- Equation of the first layer:

$$i_1 = sigmoid(H_{t-1} * W_h + X_t * W_x + bias) \tag{2.14}$$

Where:

- H_{t-1} represents the hidden state (short-term memory).
- W_h represents the weight associated with the hidden state.
- X_t represents the input in the current time t.
- W_x represents the weight associated with the input.
- i_1 is the output like shows the (Figure 2.17).

- Equation of the second layer:

$$i_2 = tangent(H_{t-1} * W_h + X_t * W_x + bias) \tag{2.15}$$

The operands of the second equation are similar to the first one, since the tangent function take the the short term memory and current input as well. After that the two result of this equation from the two layers are multiplied then pass into the step 2.

$$i_{input} = i_1 * i_2 \tag{2.16}$$

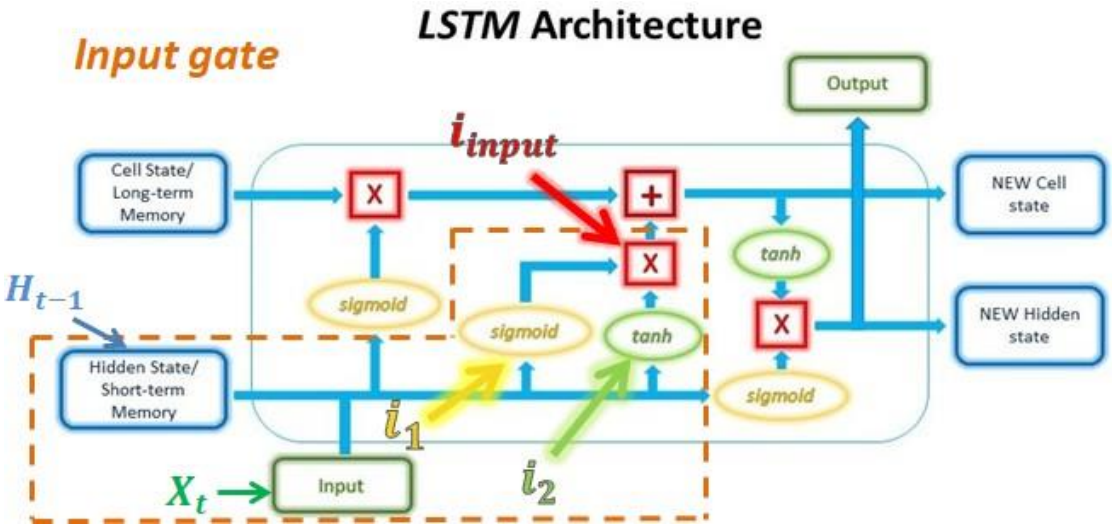

Figure 2.17: Input gate architecture

After that the result will be made up of 0s and 1s and will be multiplied with the long-term memory to choose which parts of the long-term memory to retain, and then produce the new version of the long term memory. In other word, after the two steps there is enough information to calculate the cell state, (see Figure 2.18).

- Equation that represent this process:

$$C_t = C_{t-1} * f + i_{input} \tag{2.17}$$

Where the $C_{t-1}$is the long term memory.

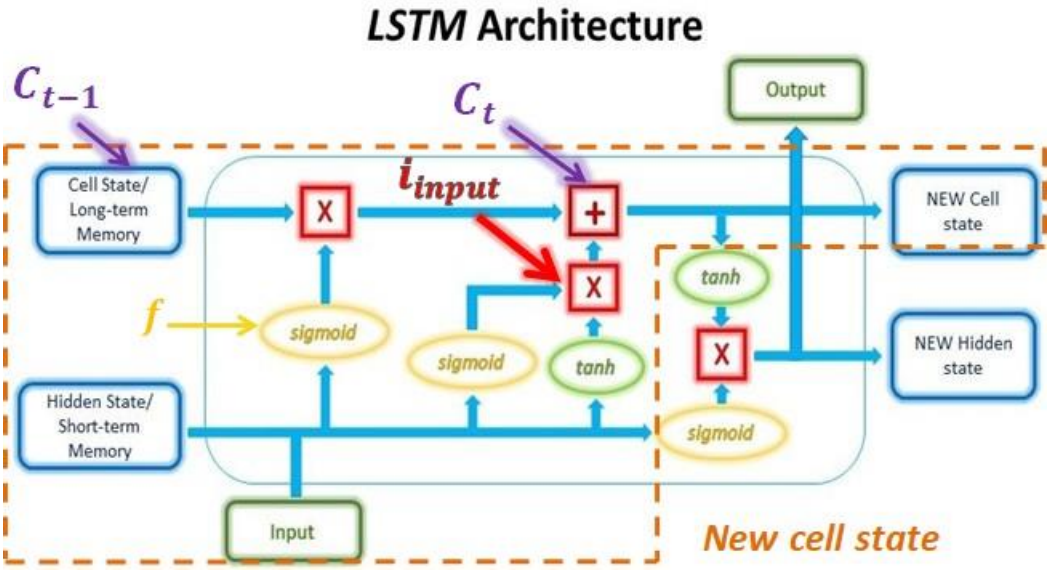

Figure 2.18: New cell state

2.5.2.3 Step three

After updating the long term memory it is time to decide what the next hidden state should be, and that what output gates do. The output gates produce the short-term memory which will be passed on to the cell in the next time step, it takes the current input, the previous short-term memory, and the newly computed long-term memory as parameters. First, the previous hidden state and the current input are passed into a sigmoid function. Then the newly modified cell state is passed to the tanh function. Finally, tanh output will be multiplied to the sigmoid output to decide what information the hidden state should carry. The output of the current time step can also be drawn from this hidden state, (see Figure 2.19).

- Equation that represent this process:

$$O_1 = sigmoid(H_{t-1} * W_{output1} + X_t * W_{output1} + bias_{output1}) \quad (2.18)$$

$$O_2 = tangente(C_t * W_{output2} + bias_{output2}) \quad (2.19)$$

$$H_t, O_t = O_1 * O_2 \quad (2.20)$$

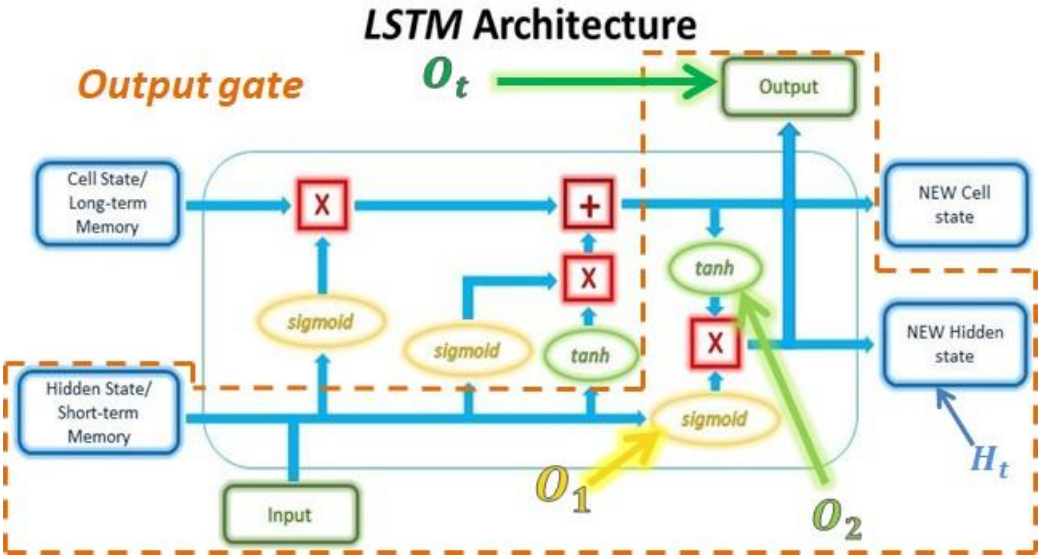

Figure 2.19: Output gate

2.6 Conclusion

This chapter was about Neural Networks. It starts with an introduction about the beginning of neural network, then a simple definition with an explication of how the neural network imitate the human brain working in section training neural network. The Recurrent Neural Networks is known as the most powerful extension of Artificial neural networks, in this chapter everything about RNN from the definition to the architecture and types is presented, also a subsection about Elman recurrent network which is an extension of RNN and its problems solved by the Long-Short-Term Memory. Finally, this chapter gives a gentle introduction about the LSTM. In the next chapter, an LSTM model will be used to make predictions on time series data.

Chapter 3

LSTM-based RNN Model Development for Time Series Forecasting

3.1 Introduction

With forecasting, companies can detect risks and then react to reduce them, in order to give the best results and achieve customer demand. Forecasting gives the capability to make informed business decisions and develop data-driven strategies for businesses. Time series data analysis is one of the quantitative methods for business forecasting that is concerned with measurable data such as historical data. Deep learning, on the other hand, offers advanced technics for business forecasting, thanks to the neural network which has the advantage that it can approximate nonlinear functions. In other words, neural networks can handle more complex and big data. Furthermore, Recurrent neural networks evoluate in order to battle amnesia, it is famous that RNNs have a sense of memory which helps them in keeping track of what happened earlier in the sequential data (time series). In this chapter, we will apply LSTM to time series forecasting and compare it to several baselines models.

3.2 Proposed time series forecasting model

For this thesis, the proposed model solution for time series forecasting is Long Short-Term Memory networks. LSTMs are able to model temporal dependencies in larger horizons without forgetting the short-term patterns. LSTM networks differ from ERNN in the hidden layer, also known as LSTM memory cell, there is many types of LSTM models that can be used on time series forecasting as Vanilla LSTM that we were applicated for. In this section, we will discover

how we can develop it for univariate and multivariate time series data. Thus, we have choosen two kind of time series datasets in order to realize this purpose. We gave an overview as well as the architecture of proposed model.

3.2.1 Functioning process overview

In order to achieve our goal which is forecasting, the following process is applicate; First, there is a need to have collected historical data because time series forecasting occurs when making predictions based on historical time-stamped data. These data points consist of sequential measurements made from the same source over a time interval and are used to track change over time. After that, we need to applicate a forecasting model to make observations and drive future strategic decision-making or have a forecast as a result. For the model, Vanilla LSTM model was applicated on collected time series data to make the forecast as shows Figure 3.1.

Figure 3.1: Overview of the studied LSTM for time series forecasting

3.2.2 Architecture of the proposed model

The following Figure 3.2, is the architecture of the proposed Vanilla LSTM model. For this study two datasets are presented: one multivariate time series data and the other is univariate time series data, both are first passed into pre-processing step: for univariate data the values was converted between range 0 and 1, for multivariate data the values was transformed into datetime. After that the desired features are extracted, and then the dataset is divided into two sub-datasets: one for training and the second for testing which is applicate to the trained Vanilla LSTM model to have as a result a forecasting.

The Vanilla LSTM model has a single hidden layer of LSTM units and an output layer to make a prediction. The architecture of Vanilla LSTM for univariate dataset is as showing in Figure 3.2, with one hundred LSTM units and one input at one time, in the other hand the architecture of Vanilla LSTM for multivariate dataset is the same except that there is no *reshape* and the model treat multiple inputs at one time, we used fifteen LSTM unit for this data.

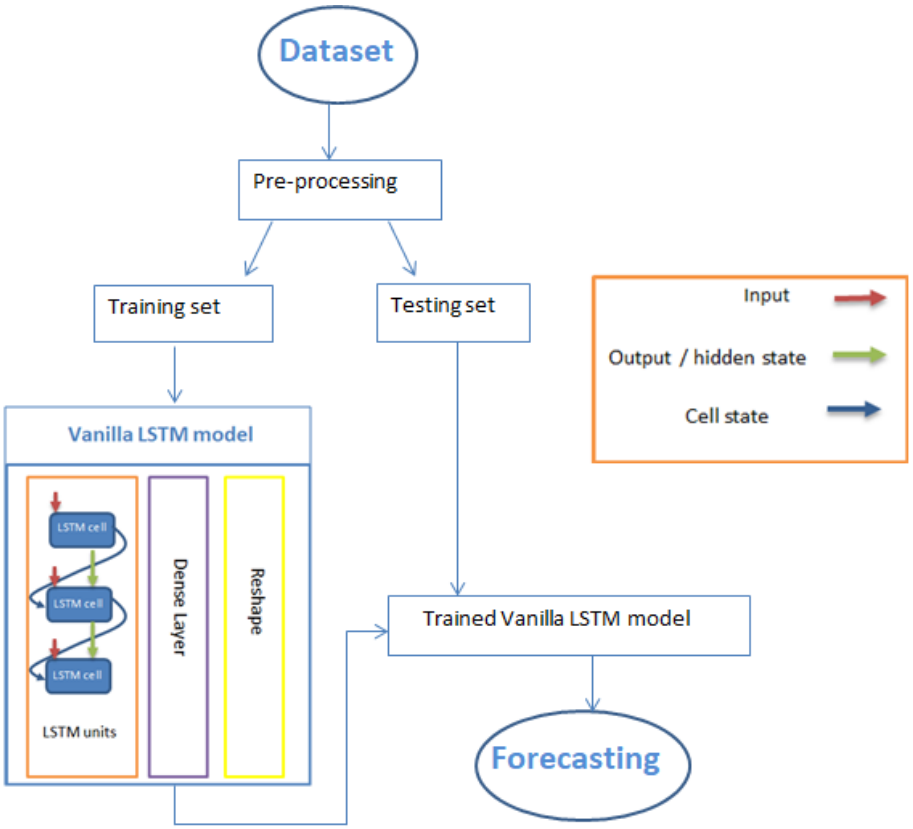

Figure 3.2: Proposed model architecture

3.3 Implementation tools

The implementation was done with *python* language and *keras* (see Figure 3.3), the machine used is i5-7200U CPU with 2.50GHz, 8G RAM and 256 SSD hard drive.

Figure 3.3: Python and Keras

3.3.1 Python

Python was created by Guido van Rossum, and released in 1991. It is the most popular language because of its simple syntax which similar to the English language. Python is the

perfect selection for machine learning projects and its benefits include:

- Consistency and simplicity: Python is easy to understand and read, which allows developers to focus on the machine learning problems and not the language technicalities. The great advantage of Python is that it has different libraries and can perform many complex machine learning tasks.
- Platform independency: Python works on different platforms : Windows, Mac, Linux..
- Good visualization options: Python offer such good libraries to visualize the data like Matplotlib.
- Versality: Python works very well in backend of web development, and also for machine learning project.

3.3.2 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform *Tensorflow.* It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. Keras is used by organizations and companies including NASA and YouTube. However, with over one million individual users as of late 2021, Keras has strong adoption across both the industry and the research community. In 2019, Keras was ranked as #1 for deep learning both among primary frameworks and among all frameworks used by top-5 teams in *Kaggle competition.* The following (Figure 3.4) shows statistics.[8]

[8] [1]Those statistics are from: https://keras.io/why_keras/

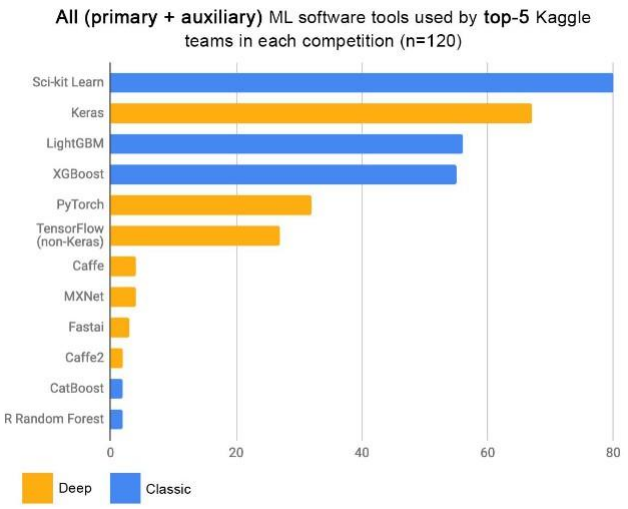

Figure 3.4: Keras ranking in kaggle competitions

3.3.3 Libraries

It is known that libraries provide developers with pre-defined and optimized function to make their work easier; the most advantage is that libraries can lessen the time to project development, this is the most libraries that we used in this work:

- NumPy: was created in 2005 by Travis Oliphant, it is a Python library used for working with arrays. NumPy provides an array object that is up to 50x faster than traditional Python lists.
- Matplotlib: is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter.
- Pandas: started by Wes McKinney in 2008, it allows to access many of matplotlib's and NumPy's methods with less code.
- Datetime: Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

3.4 Experiments

Experiments section is about the datasets used in this work and how the solution is implemented given details about it, the different baselines and evaluation metrics used.

3.4.1 Datasets

Of the many different datasets, we have chosen two datasets from *kaggle*:

- The first data is a univariate time series dataset for Milk production, which is collected from 1962 to 1975 and composed of two features (Date and Production) as shown in Figure 3.5. The goal of this data is to forecast milk production in the next 33 months.

Date	Production
1962-01-01	589
1962-02-01	561
1962-03-01	640
1962-04-01	656
1962-05-01	727

Figure 3.5: Milk production dataset

- The second data is Netflix multivariate time series dataset, Netflix is an American subscription streaming service and Production Company, launched on 1997; it offers a film and television series library through distribution deals as well as its own productions, known as Netflix Originals. Netflix dataset was published in late May 2022, for this one we aim to forecast the open price of the stock in next three months. It is collected from 2002 to 2022 like shows Figure 3.6and it has six features (Date, Open, High, Low, Close, Adj Close, and Volume).

	Date	Open	High	Low	Close	Adj Close	Volume
0	2002-05-23	1.156429	1.242857	1.145714	1.196429	1.196429	104790000
1	2002-05-24	1.214286	1.225000	1.197143	1.210000	1.210000	11104800
2	2002-05-28	1.213571	1.232143	1.157143	1.157143	1.157143	6609400
3	2002-05-29	1.164286	1.164286	1.085714	1.103571	1.103571	6757800
4	2002-05-30	1.107857	1.107857	1.071429	1.071429	1.071429	10154200

Figure 3.6: Netflix dataset

3.4.2 Implementation details

In order to give more details about our work, the following are global implementation steps that give a clear image of the code organization for the univariate and multivariate datasets.

3.4.2.1 Univariate data

Figure 3.7 shows the implementation process for univariate time series data which will be explained next with code screenshots illustration.

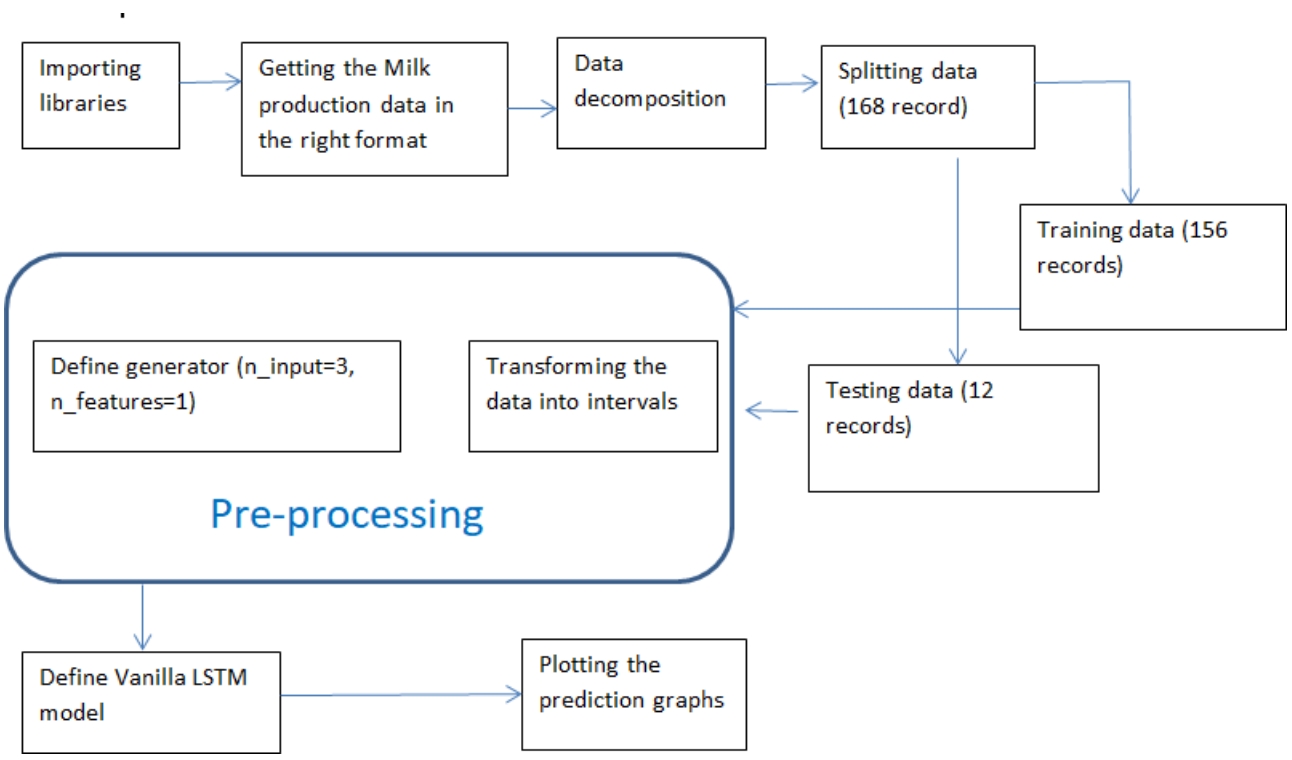

Figure 3.7: Univariate dataset implementation process

In this program we will see how to implement a univariate data for time series. Respecting time, let us predict production sales using date as index feature. We are trying to forecast if the number of prediction increase or decrease, starting by importing libraries (see Figure 3.8).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Figure 3.8: Importing libraries

After that, getting data ready in the right format (Figure 3.9):

```
df = pd.read_csv('monthly_milk_production.csv',index_col='Date',parse_dates=True)
df.index.freq='MS'
```

Figure 3.9: Getting the data ready

Then we were plotted data (see Figure 3.10):

```
Entrée [6]: df.plot(figsize=(12,6))

Out[6]: <AxesSubplot:xlabel='Date'>
```

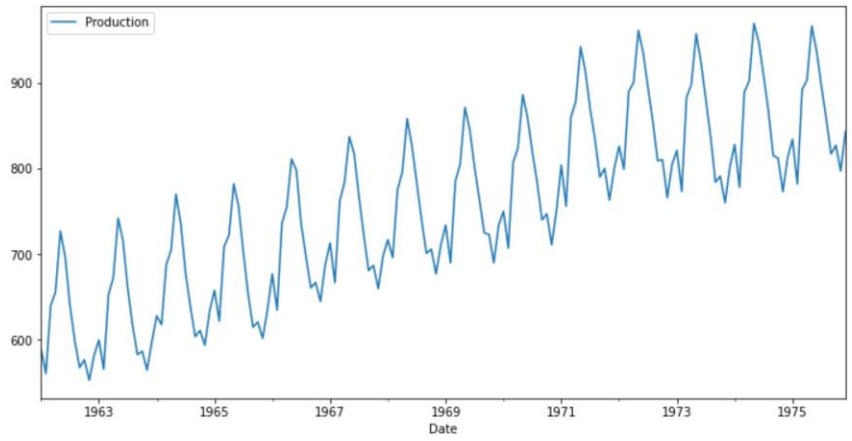

Figure 3.10: Data plot

From the beginning we can see that there is a kind of seansonality. Repetitive shema and a general trend that increase over time.

Then, if we need to decompose and see the season for example and trend we use *seasonal_decompose* wich decomposes the different parts of time series (Figure 3.11):

```
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
results = seasonal_decompose(df['Production'])
results.plot();
```

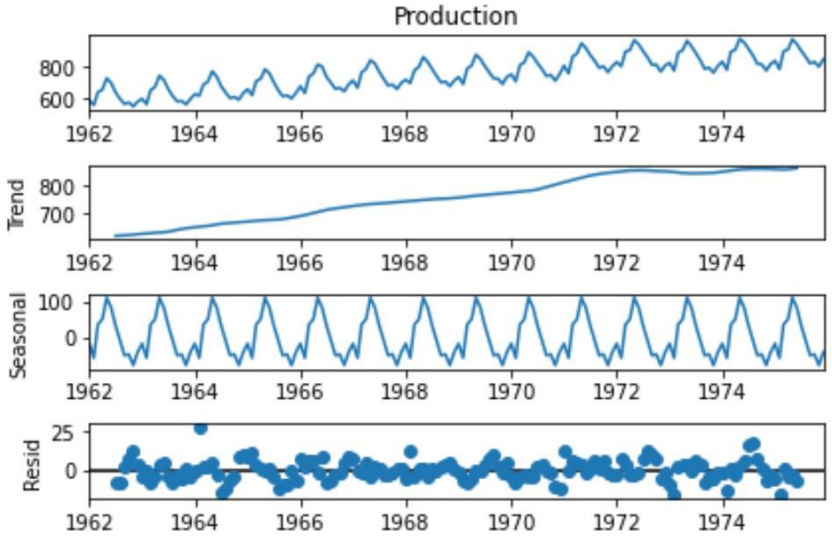

Figure 3.11: Time series decomposition

Now, we divide the dataset into a training part and test. Here we took all the values except the last 12 months we took them as a forecast (see Figure 3.12):

```
train = df.iloc[:156]
test = df.iloc[156:]
```

Figure 3.12: Dividing data

For the normalization of the data we will use Min Max Scaler to convert the data in interval 0 and 1 like shows (Figure 3.13) :

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

Figure 3.13: Normalization

Here, the machine was trained then we formatted it. After that we used time series generator to convert the data (see Figure 3.14):

```
from keras.preprocessing.sequence import TimeseriesGenerator
```

```
# define generator
n_input = 3
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_size=1)
```

```
X,y = generator[0]
print(f'Given the Array: \n{X.flatten()}')
print(f'Predict this y: \n {y}')
```

```
Given the Array:
[0.08653846 0.01923077 0.20913462]
Predict this y:
 [[0.24759615]]
```

Figure 3.14: Pre-processing implementation

Then defining the model as shown (Figure3.15):

Figure 3.15: Vanilla LSTM model

```
# define model
model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse',metrics=['mse','mae']))
```

When we use LSTM we always need to reshape data in 2 or 3 dimensions. First parameter is number of records, second is the shape of this particular, how many numbers of line steps are there and one should be something like features (Figure 3.16):

```
test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):

    # get the prediction value for the first batch
    current_pred = model.predict(current_batch)[0]

    # append the prediction into the array
    test_predictions.append(current_pred)

    # use the prediction to update the batch and remove the first value
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

Figure 3.16: Reshaping the data

Finally, we plotted the graph of our production of milk and the predictions of it, more details will be presented in Results and discussion section.

3.4.2.2 Multivariate data

As in univariate dataset we will give the implementation process in Figure 3.17 and then those steps will be explained next.

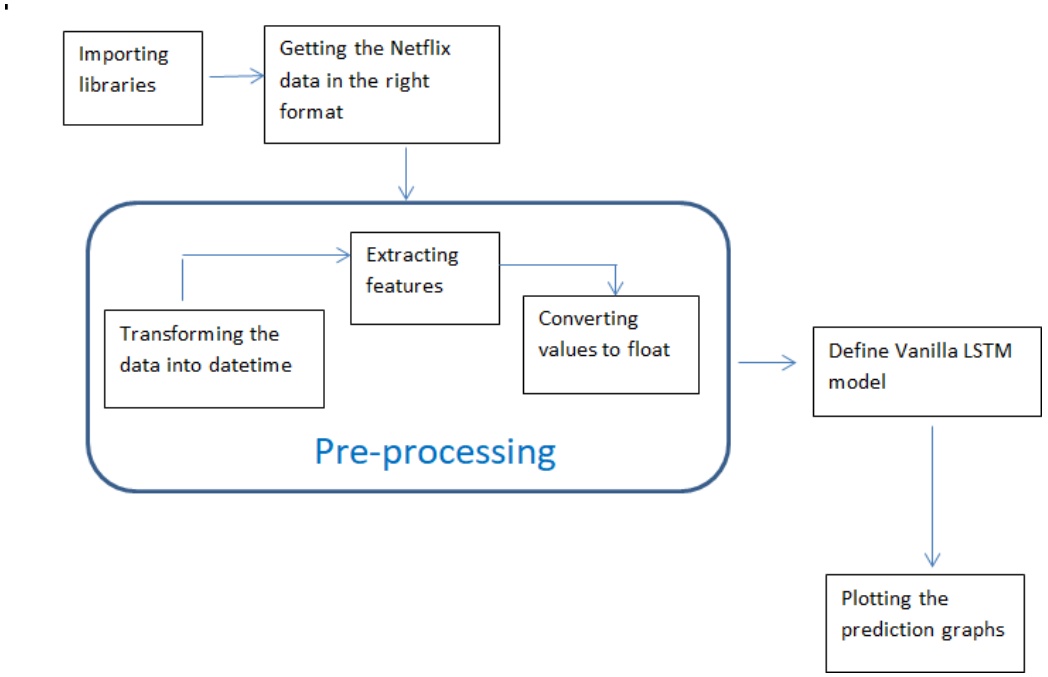

Figure 3.17: Multivariate data implementation process

In the following we will se how a multivariate time series data is implemented to do forecasting, we are trying to predict if the open value is decrease or increase. In this data, we have five variables counting the one to be forecasted. First of all, we start by importing libraries (see Figure 3.18):

```
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import datetime as datetime
```

Figure 3.18: Importing libraries used for multivariate time series

Like in univariate data we need to get the data in the right format and for that the same code is applicate. After that, we transformed the data which is as an object text into datetime for plotting (Figure 3.19).

```
train_dates = pd.to_datetime(df['Date'])
print(train_dates.tail(15))
```

Figure 3.19: Transform the data into datetime

The following step is to extract the training dates into a separate series, which means extracting the column that we would like to use as variables.

After that, we converted all the values to float because when we actually do normalization we don’t lose any information (see Figure 3.20):

```
cols = list(df)[1:6]
print(cols)
```

```
df_for_training = df[cols].astype(float)
```

Figure3.20:Convertingvalues

In the next step (see Figure 3.21):

```
scaler = StandardScaler()
scaler = scaler.fit(df_for_training)
df_for_training_scaled = scaler.transform(df_for_training)
```

Figure 3.21: Pre-processing of multivariate data After

that defining model like shows(Figure 3.22):

```
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(trainX.shape[1], trainX.shape[2])
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse',metrics=(['mse', 'mae']))
```

Figure 3.22: Defining Vanilla LSTM

Finally, do the prediction and plot it. For the result, like sited before it will be presented in the Rsults and discussion section.

3.4.3 Baselines

Baseline models are considered as reference in a machine learning project. In other words, a baseline is a point of reference for all other modeling techniques on the problem. The following are baselines that we have chosen for this study:

3.4.3.1 Random Forest

The Random Forest algorithm is a type of supervised learning that builds decision tree on difficult samples and takes their majority vote for classification and average in case of regression[91]. Here is the 4 steps way of the Random Forest:

- Selection of Random samples from a given dataset.
- Construction of decision tree for each sample.
- Arrangement of a prediction result for each produced result.
- Selection of the final result, here the prediction result with the most votes is selected.
- Plot the result.

3.4.3.2 Naïve Bayes

Naïve Bayes Classifier is a probabilistic classifier and is based on Bayes Theorem. In Machine learning, a classification problem represents the selection of the Best Hypothesis given the data. Given a new data point, we try to classify which class label this new data instance belongs to. The prior knowledge about the past data helps us in classifying the new data point [92]. Here is the 4 steps way of naive bayes:

- Transform the dataset into supervised learning.
- Establish the train and test dataset.
- Define the persistence model.
- Make forecast and establish a baseline performance.
- Plot the result.

3.4.3.3 ARIMA

ARIMA is the famous baseline used, as sited in chapter one ARIMA is composed of Auto-regression, Integrated (differencing), Moving Average. Thus the steps to build an ARIMA are [93]:

- Auto-regression which refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- Integrated (differencing) that represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
- Moving Average: incorporates the dependency between an observation and a residual error from a moving average model applied to lag observations.

3.4.4 Evaluation metrics

In addition of baselines, evaluation metrics are very essential part of this study to make comparison between our proposed Vanilla LSTM model and other baselines. The evaluation metrics used are Mean Squared Error and Mean Absolute Error:

- Mean Squared Error (MSE): the MSE measures the average squared difference between forecasted and true values [94].
- Mean Absolute Error (MAE): this metric tell us how accurate our predictions are and, what is the amount of deviation from the actual values [95].

However this two metrics are well defined in chapter two:

Training neural network section (2.3), loss function sub-section (2.3.3).

3.5 Results and discussion

The results of our work are presented in this section, we first gives graphs provided from the implementation and then compared the results of our model with baselines to finally conclude with discussion.

3.5.1 Results

In univariate time series data, after taking the first look to the graph in Figure 3.23 apparently the prediction graph produced by the implementation is nearly similar to the original Milk production plot. We can observe that as the production , prediction increase highly in May month measured with more than 900 production, then it decrease and re increase slowly in October with approximately 800 production so there is fluctuation between ascending and descendingof prediction mean.

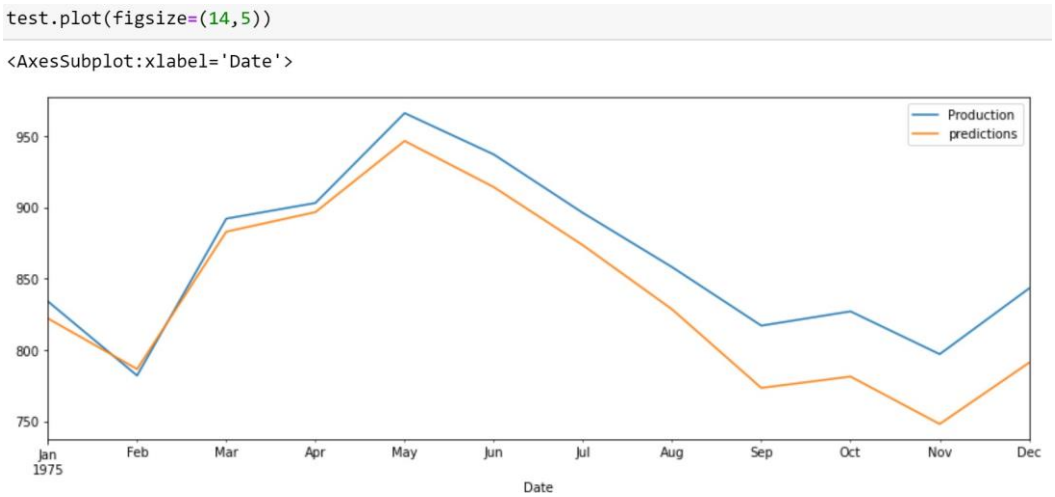

Figure 3.23: Milk production and prediction graph

Passing to our multivariate time series dataset, we plot the original data (Figure 3.24) and as sited before the purpose is to forecast the Netflix opening price for the next three months, since the data ends in April that means that we want to predict for May Jun and Jully months, the prediction of this work as presented in Figure 3.25 indicate an observed increase between day 01/06/2022 and 15/06/2022.

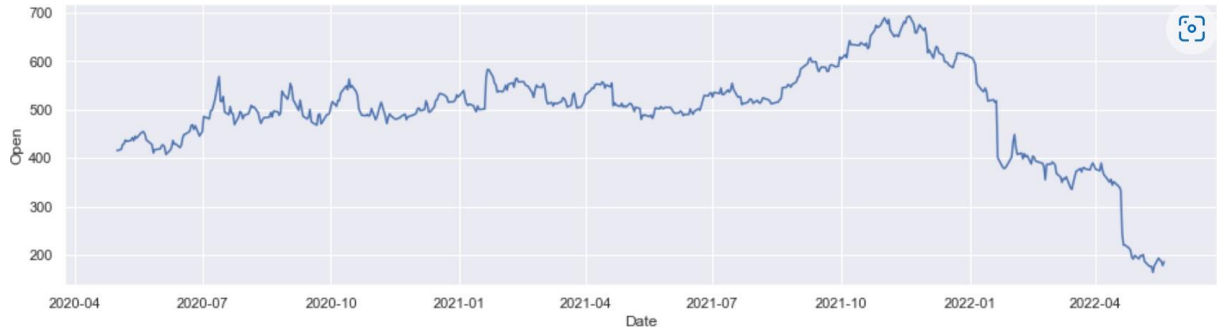

Figure 3.24: Plot of the Netflix open price stock

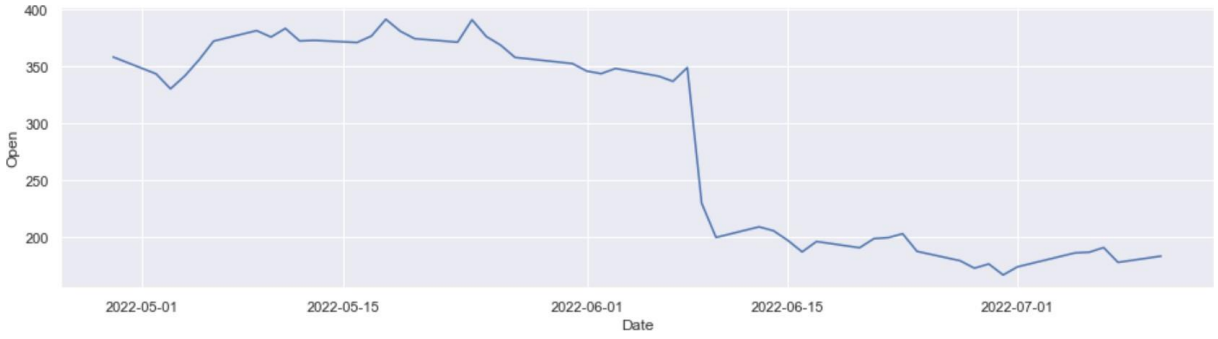

Figure 3.25: Plot of the Netflix opening price prediction

3.5.2 Comparison

The following is the results of the two datasets with evaluation metrics and baselines. The result of evaluation metrics of our Vanilla LSTM model are presented for the Milk production dataset with graphs like indicate Figure 3.26, the left one is for MAE and the right one is MSE, as well as for Netflix dataset in Figure 3.27.

Then we compared those results with the three baselines sited before (Random forest (RF), Naïve Bayes (NB) and ARIMA) and all results are organized in Table 3.1.

	Milk dataset				Netflix dataset			
Baselines	RF	NB	ARIMA	Vanilla LSTM	RF	NB	ARIMA	Vanilla LSTM
MSE	4.10	2302.2	**$7.83e^{-5}$**	$1.42e^{-4}$	0.48	48.69	0.023	**$1.60e^{-4}$**
MAE	1.49	40.84	**0.0072**	0.0213	0.19	4.63	0.122	**0.0051**

Table 3.1: Evaluation metrics with baseline methods

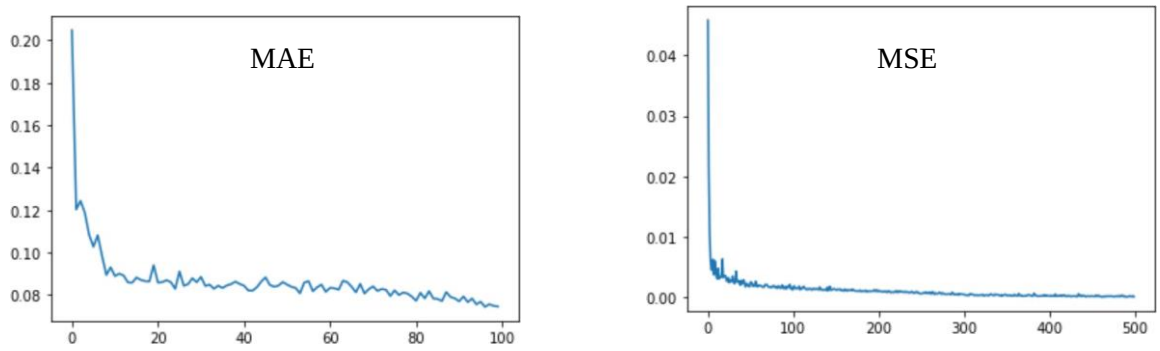

Figure 3.26: MAE and MSE for univariate dataset

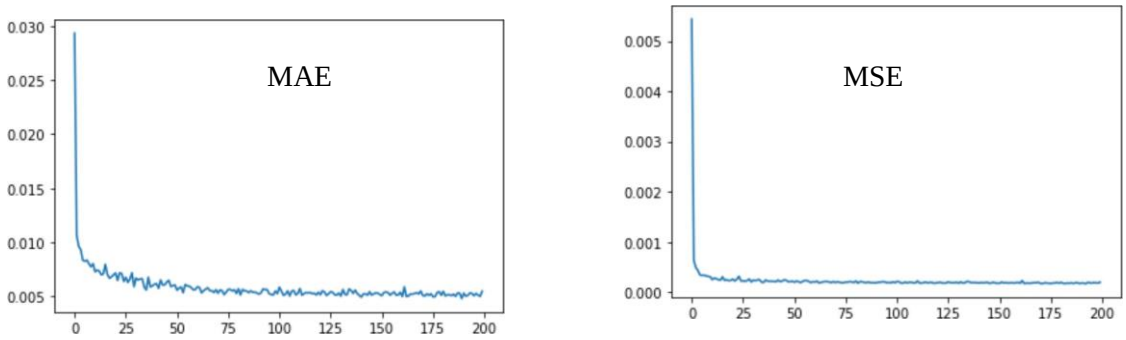

Figure 3.27: MAE and MSE for multivariate dataset

After comparing the results of prediction between our model and other baselines, we can clearly see that our model performed very well in both of the datasets, especially in ARIMA. If we analyze the result we conclude that there is a big difference between baselines and LSTM model except for ARIMA, but also we can observe that the results of our LSTM model are very nearby to the results of ARIMA model that making it more precise for prediction in time series.

3.5.3 Discussion

As shown in Figure 3.25, in the range between [15-05-2022 and 01-06-2022] there is an uptrend, Netflix has really gone a long way ahead of its competition because of its more successful Tv shows and movies that have garnered attention and a high number of views. This has helped escalate the rate of subscriptions. Netflix has been more successful in identifying the true interest of customers or audiences. After the date (01-06-2022) there is an observed downtrending, the catalog of Netflix changes by the country for example in USA, France or Belgique users will not have access to the same broadcasts, also according to leadres of Netflix, the more the platform will be successful in terms of the number of views, the more it will increase in value and therefore this growth justifies the increase in the price of the subscription.

3.6 Conclusion

In this chapter, we proposed a Vanilla LSTM model for time series forecasting and we gave the architecture of this model, as well as the different technics used for the realization of this solution. We presented also a detailed implementation process and its results, then we compared them using baselines. At the last, we can say that the LSTM cell adds long-term memory in an even more performant way because it allows even more parameters to be learned. This makes it the most powerful Recurrent Neural Networks to do forecasting, especially when we have a longer-term trend in the data. LSTMs are one of the state-of-the-art models for forecasting at the moment and we saw that it is true according to our Vanilla LSTM model.

General conclusion

In chapter one, we gave a definition about time series data and the ability to applicate statistics on this data wich named time series analysis, on the other side we clarify that time series forecasting is one of time series analysis goals. In order to forecast there is several machine learning methods but with some limitations, however deep learning provided better resutls in trainnig process in the last decades. When talking about deep learnig, Neural networks are the states of the art especially Reccurent neural networks wich was well defined in chapter two with the diffrent extensions to solve some problems of RNNs. The purpose of these 2 chapters is to give an idea for what are time series forecasting and about how we can answer the previous question. This question was:

" How deep learning can be used for time series forecasting?"

Long Short-Term Memory is presented as a solution to forecast time series data. The proposed model was Vanilla LSTM, this model was applicated on some real datasets , an univariate time series data for Milk production and multivariate time series data for Netflix. After that the results was compared with baselines. Consequently, we can say that Vanilla LSTM model provide satifying results that make it more precise in time series forecasting. Then, we say if deep learning arrive to this particular opportunities it can be more better and why not the best for forecasting.

In future work, deep learning models combined with LSTM will be used to forecast time series data for multilayers model hyper-parameters, as well as these models will be evaluated on stacked and bidirectionnal models various benchmark time series datasets.

Bibliography

[1] Tejalkadam18m, "What is a trend in time series?," *geeksforgeeks.org*, 2021.

[2] "Nist/sematech e-handbook of statistical methods, https://www.itl.nist.gov/ div898/handbook/pmc/sect , november2003."

[3] M. Dimitrievsky, "Exploring seasonal patterns of financial time series with boxplot," *mql5.com*, 2020.

[4] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles Practice*. OTexts with bookdown, 8 may 2018.

[5] *Irregular Variation*, pp. 269–270. New York, NY: Springer New York, 2008.

[6] F. Ndiritu, "Univariate time series using facebook prophet," *section web site*, March 18, 2022.

[7] A. Singh, "A multivariate time series guide to forecasting and modeling (with python codes)," *analyticsvidhya*, September 27, 2018.

[8] R. Nau, "Statistical forecasting:notes on regression and time series analysis," *Duke University*, August 18, 2020.

[9] S. palachay, "Stationarity in time series analysis," *Towards data science*, Apr 8, 2019.

[10] S. Prabhakaran, "Augmented dickey fuller test (adf test) â must read guide," *machinelearningplus.com*, november 2, 2019.

[11] L. Smigel, "What is stationarity? a visual guide," *Analyzingalpha.com*, April 27, 2022.

[12] S. Pandian, "A comprehensive guide to time series analysis," *Analytics Vidhya web site*, October 23, 2021.

[13] T. R. D. Y. chi Chang Raju Pavuluri Shankar Subramanian, "What is time series classification?," *developer.ibm.com*, January 25, 2022.

[14] S. Glen, "Curve fitting," *StatisticsHowTo.com: Elementary Statistics for the rest of us!*, August 20, 2018.

[15] stephanie glen, "Adjusted r2 / adjusted r-squared: What is it used for?," *statistic-showto.com*, 2021.

[16] stepahnie glen, "Rmse: Root mean square error," *statisticalhowto.com*, 2021.

[17] M. R. H. Simon P. Neill, *Fundamentals of Ocean Renewable Energy*. 2018.

[18] V. Kurama, "An introduction to segmentation, correlation and time series modeling," *builtin.com*, July 13, 2021.

[19] N. S. Chauhan, "Dbscan clustering algorithm in machine learning," *kdnuggets.com*, April 4, 2022.

[20] zach, "Measures of central tendency: Definition examples," *statology.org*, SEPTEMBER 18, 2018.

[21] A. Beyer, *Introduction to Statistics for Psychology*. August 1, 2021.

[22] A. Sugandhi, "Measures of dispersion: All you need to know," *knowledgehut.com*, 04th Apr, 2022.

[23] A. S. Rawat, "An overview of descriptive analysis," *analyticssteps.com*, Mar 31, 2021.

[24] J. Frost, "Contingency table: Definition, examples interpreting," *statisticsbyjim.com*.

[25] A. Dotis-Georgiou, "An introduction to time series forecasting," *infoworld.com*, JUL 22, 2021.

[26] B. Artley, "Time series forecasting with arima , sarima and sarimax," *towardsdata-science.com*, apr,16.

[27] L. Zoubir, "A brief history of time series analysis," *Stokholm University*, October 31, 2017.

[28] J. K. Nikolas Ulrich Moroff, Ersin Kurt, "Machine learning and statistics: A study for assessing innovative demand frprecasting models," *Elsevier*, 2021.

[29] Okolie Paul Chukwulozie Azaka Onyemazuwa Andrew Okoli Ndubuisi Celestine Sinebe Jude Ebieladoh. "Analysis and forecasting of the production quantity in amanufacturing industry using historical data," *The International Journal Of Engineering And Science (IJES)*, 2015.

[30] J. Brownlee, "How to grid search triple exponential smoothing for time series forecasting in python," *Machine learning mastery*, October22,2018 update on August 28, 2020.

[31] A. Crochet-Damais, "Machine learning : definition, modele, algorithme et langage,"*journaldunet.fr*, 2022.

[32] B. Dutta, "A classification and regression tree (cart) algorithm," *analyticssteps.com*, Jul 27, 2021.

[33] Deepankar, "Decision tree with cart algorithm," *Geek Culture*, Apr19,2021.

[34] Ribeiro Pedro Rafael Andrea Maria do Carmo Iago Richard Rodrigues Djamel Sadok Theo Lynn Patricia Takako Endo, "Short-term firm level energy consumptionforecasting for energy intensive manufacturing:a comparison of machine learning and deeplearning models,"*al- gorithms*, 30 October 2020.

[35] A. Sethi, "Support vector regression tutorial for machine learning," *analyticsvidhya.com*, March 27, 2020.

[36] A. K. Muhammad Qamar Raza, "Un avis sur artifiprevision de la demande de charge basee sur l intelligence nominaletechniques pour les reseaux intelligents et les batiments," 2015.

[37] Pedro Lara-Benıtez Manuel Carranza-Garc iajos e. Riquelme, "An experimental review on deep learning architecturesfor time series forecasting," *International Journal of Neural Systems*, 8 Apr 2021.

[38] Tim Menzies Leandro Minku Burak Turhan Ekrem Kocaguneli Fayola Peters, *Sharing Data and Models in Software Engineering*. Elsevier, 2015.

[39] Md. Mustafizur Rahman et Md. Monirul Islam et Kazuyuki Murase et Xin Yao, "Layered ensemble archi- tecture for time series forecasting," *IEEE Transactions on Cybernetics*, vol. 46, pp. 270– 283, 2016.

[40] F. T. et Antonio Galicia et Alicia Troncoso Lora et Francisco MartInez-Alvarez, "Une approche evolutive basee sur lâapprentissage profond pour la prevision des series chronologiques big data," *Integr. Comput. Aided Eng.*, vol. 25, pp. 335–348, 2018.

[41] S. Shamshirband, "Wind speed prediction using a hybrid model of the multi-layerperceptron and whale optimization algorithm," *Elsiever.com*, 2020.

[42] M. Mandal, "introduction to convolutional neural network (cnn),"*Analysticsvidhiya.com*, 2021.

[43] M. Ciortan, "Gentle introduction to echo state networks," *Towards Data Science*, 2019.

[44] Z. SOUHILA, "Session-based recommender systems using recurrent neural networks," Master's thesis, UNIVERSITE IBN KHALDOUN - TIARET, 2019-2020.

[45] J. Brownlee, *Deep Learning for Time Series Forecasting*. Machine learning mastery, 2018.

[46] M. Tomasini, "How veritas technologies uses influxdb to enable time series forecasting at scale," *influxdata.com*, may 2019.

[47] A. Tavgen, "How playtech is using influxdb to monitor their distributed system," *influx-data.com*, july 2018.

[48] K. D. Foote, "A brief history of neural networks," *dataversity.net*, November 9, 2021.

[49] S. Walczak, "Artificial neural networks," in *Encyclopedia of Information Science and Technology, Fourth Edition*, pp. 120–131, IGI Global, 2018.

[50] H. Singh, "Deep learning 101: Beginners guide to neural network," *analyticsvidhya*, March 1, 2021.

[51] N. McCullum, "Deep learning neural networks explained in plain english," *freecode-camp.org*, June 28, 2020.

[52] K. Ahirwar, "Everything you need to know about neural networks," *hackernoon.com*, May 27, 2022.

[53] J. Bhattacharyya, "Activation functions in neural networks: An overview,"*analyticsindia-mag.com*, November 6,2020.

[54] S. Tiwari, "Activation functions in neural networks," *geeksforgeeks.org*, 18 May,2022.

[55] A. Wolfewicz, "Deep learning vs. machine learning , what is the difference?," *levity.ai*, April 21, 2022.

[56] F. Malik, "Neural networks bias and weights," *medium.com*, May 18,2019.

[57] N. Joshi, "Everything you need to know about adaptive neural networks," *allerin.com*, February 25, 2020.

[58] H. Mahmood, "Gradient descent," *towardsdatascience.com*, Jan 2,2019.

[59] A. Roy, "An introduction to gradient descent and backpropagation," *Towardesdata-science.com*, Jun 14, 2020.

[60] I. C. Education, "Gradient descent," *ibm.com*, 27 October 2020.

[61] A. S. Walia, "Activation functions used in neural networks-which is better?," *medium.com*, October 31,2022.

[62] P. Baheti, "12 types of neural network activation functions: How to choose?," *v7labs.com*, May 26, 2022.

[63] G. L. Team, "What is rectified linear unit (relu)? | introduction to relu activation function," *mygreatlearning.com*, Aug 29,2020.

[64] C. Yanhui, "A battle against amnesia: A brief history and introduction of recurrent neural networks," *towardsdatascience.com*, Mar 8, 2021.

[65] F. Cheng and J. Zhao, "A novel process monitoring approach based on feature points distance dynamic autoencoder," in *29th European Symposium on Computer Aided Process Engineering* (A. A. Kiss, E. Zondervan, R. Lakerveld, and L. ozkan, eds.), vol. 46 of *Computer Aided Chemical Engineering*, pp. 757–762, Elsevier, 2019.

[66] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018.

[67] A. Moghar and M. Hamiche, "Stock market prediction using lstm recurrent neural network," *Procedia Computer Science*, vol. 170, pp. 1168–1173, 2020.

[68] A. Shewalkar, "Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 235–245, 2019.

[69] Q. Liu, L. Fang, G. Yu, D. Wang, C.-L. Xiao, and K. Wang, "Detection of dna base modifications by deep recurrent neural network on oxford nanopore sequencing data," *Nature communications*, vol. 10, no. 1, pp. 1–11, 2019.

[70] Z. Zhang, Z. Lv, C. Gan, and Q. Zhu, "Human action recognition using convolutional lstm and fully-connected lstm with different attentions," *Neurocomputing*, vol. 410, pp. 304–316, 2020.

[71] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm," *Chaos, Solitons & Fractals*, vol. 140, p. 110212, 2020.

[72] A. Hassan, I. Shahin, and M. B. Alsabek, "Covid-19 detection system using recurrent neural networks," in *2020 International conference on communications, computing, cybersecurity, and informatics (CCCI)*, pp. 1–5, IEEE, 2020.

[73] P. Arora, H. Kumar, and B. K. Panigrahi, "Prediction and analysis of covid-19 positive cases using deep learning models: A descriptive case study of india," *Chaos, Solitons & Fractals*, vol. 139, p. 110017, 2020.

[74] D. Kalita, "A brief overview of recurrent neural networks (rnn)," *analyticsvidhiya.com*, March11,2022.

[75] "What is natural language processing nlp." published on sas.com, 2022.

[76] E. Singh, N. Kuzhagaliyeva, and S. M. Sarathy, "Chapter 9 - using deep learning to diagnose preignition in turbocharged spark-ignited engines," in *Artificial Intelligence and Data Driven Optimization of Internal Combustion Engines* (J. Badra, P. Pal, Y. Pei, and S. Som, eds.), pp. 213–237, Elsevier, 2022.

[77] J. Yang and J. Kim, "An accident diagnosis algorithm using long short-term memory.," *Nuclear Engineering and Technology*, vol. 50, 05 2018.

[78] S. M, "Letâs understand the problems with recurrent neural networks," *analyticsvidhiya.com*, july 10,2021.

[79] S. Kremer, "On the computational power of elman-style recurrent networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 1000–1004, 1995.

[80] R. Chandra, "Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 12, pp. 3123–3136, 2015.

[81] E. D. Übeyli and M. Übeyli, *Case studies for applications of elman recurrent neural networks*. IntechOpen London, UK, 2008.

[82] G. Ren, Y. Cao, S. Wen, T. Huang, and Z. Zeng, "A modified elman neural network with a new learning rate scheme," *Neurocomputing*, vol. 286, 04 2018.

[83] A. Katte, "Meet juergen schmidhuber â the father of lstm, an outsider in the world of deep learning," *analyticsindiamag.com*, SEPTEMBER 17, 2018.

[84] A. Rehmer and A. Kroll, "On the vanishing and exploding gradient problem in gated recurrent units," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1243–1248, 2020. 21st IFAC World Congress.

[85] T. L. Firuz Ahamed Nahid Weerakorn Ongsakul Nimal Madhu M, *Hybrid Neural Networks for Renewable Energy Forecasting: Solar and Wind Energy Forecasting Using LSTM and RNN, Research Advancements in Smart Technology, Optimization, and Renewable Energy*. ig-global, 2021.

[86] J. S. Fan Wu, *Predictions For COVID-19 With Deep Learning Models of Long Short-Term Memory (LSTM),Biomedical and Business Applications Using Artificial Neural Networks and Machine Learning*. ig-global, 2022.

[87] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Advanced Robotics*, vol. 22, no. 13-14, pp. 1521–1537, 2008.

[88] F. Orsini, M. Gastaldi, L. Mantecchini, and R. Rossi, "Neural networks trained with wifi traces to predict airport passenger behavior," in *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 1–7, 2019.

[89] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, p. 48, 2002.

[90] G. loye, "Long short-term memory: From zero to hero with pytorch," *floydhub*, jun 15, 2019.

[91] B. Alam, "Implementation of random forest algorithm using python," *hands-on.cloud*, January 22 2022.

[92] "naive bayes classifier step by step." holypython.com, 2021.

[93] A. Hayes, "Autoregressive integrated moving average (arima)," *investopedia.com*, October 12 2022.

[94] C. Conol, "Benchmarking methods for time series forecast," *Analytics Vidhya*, April 15 2020.

[95] S. Acharya, "What are rmse and mae?," *towardsdatascience.com*, May 14 2021.