



MAKERERE UNIVERSITY

DESIGN OF A MACHINE LEARNING BASED SYSTEM FOR PHARMACEUTICAL PURCHASES

Abraham Jerry Kakooza

Reg No.: 16/U/327

Department of Electrical and Computer Engineering

School of Engineering

College of Engineering, Design, Art and Technology

Supervisor: **Dr Andrew Katumba**

Co-supervisor: **Prof. Eng. Dr. Peter Lating, iLabs Principal Investigator**

December 7, 2020

A Report submitted in partial fulfillment of the requirements for the
Degree of Bachelor of Science in Telecommunications Engineering
at Makerere University.

Declaration

ABRAHAM KAKOOZA JERRY

I declare that this research was completed with support from Soteria Pharmacy in Luweero District for the period from 1st August 2019 to 4th December 2020.

Academic Integrity Pledge:

I have abided by the Makerere University academic integrity policy on this assignment. I also confirm that this report is only prepared for my academic requirement, not for any other purpose.

Signature _____ *Date* _____

Approval

This report has been submitted with approval and under the supervision of the following supervisors:

Main Supervisor

Lecturer

Department of Electrical and Computer Engineering,
Makerere University

Dr. Andrew Katumba _____ *Date* _____

Co-Supervisor

Lecturer

Department of Electrical and Computer Engineering,
Makerere University

Prof. Peter Lating _____ *Date* _____

Dedication

I , Kakooza Abraham Jerry , dedicate this to my parents who have been there for me in this entire study period both emotionally and financially, my supervisor Dr. Andrew Katumba, my mentors Tibabwetiza Joel, my partner Kevin Upyem, and lastly the entire ilabs team Mark Phillip and Ronald Ogwang .

Acknowledgments

I would like to thank God for enabling me to complete my Final Year Project successfully. I also extend my gratitude to the ilabs at Mak Project for the opportunity to work with them which has given me a wealth of skills and experience to build my engineering career.

I highly acknowledge my main supervisor Dr. Andrew Katumba and my co-supervisor Prof. Peter Lating for the support, guidance and knowledge provided at each step of this project. I highly acknowledge my mentors David Tusubira and Tibabwetiza Joel for the guidance and support.

Lastly I would like to thank the Soteria Pharmacy community, Ms. Brenda , Ms. Lydia Ssebunya and the ilabs team Mark Phillip and Ogwang Ronnie for the collective support.

Abstract

To obtain inherent laws from vast amounts of pharmaceutical sales data and to provide valuable information to pharmacy managers, this work validates different methods and approaches to perform a sales forecast. Part of the data is used to train a neural network algorithm, with backpropagation for some methods, step by step, where shallow nets face selected scenarios, with different space-time data considerations.

In each method, by using a sum of square differences, and a peak search procedure, a reasonable quality in the obtained abstract representations is pursued. First, an auto-encoder is trained to develop in its hidden layer neural data abstractions about a random-moving window. Thereafter by using the abstraction of the net plus recently captured information, a second shallow net is trained to produce its own one-day ahead estimates, using new timing and data procedures. After training, the whole stacked system's performance is compared with the naive forecast scenario's mean square error and if it's a better value, the method is used to produce stable daily forecasting for assorted products and periods. The system has been tested in real-time with real data.

Contents

1	Introduction	1
1.1	Project Background	1
1.2	Problem Statement	2
1.3	Aims and Objectives	3
1.4	Contributions of the Project	3
1.5	Organization of the Report	3
2	Literature Review	5
2.1	Introduction	5
2.2	Literature Review	5
2.3	Data Collection	6
2.4	Machine Learning	6
2.4.1	Introduction	6
2.5	Time Series Forecasting	7
2.5.1	Introduction	7
2.5.2	ARIMA	8
2.5.3	Forecasting with LSTM methods	14
2.5.4	Website Application Programming Interface(API)	18
2.6	Summary	18

3	Methodology	20
3.1	Materials and Methods	20
3.2	Data and Results	26
3.2.1	Naïve forecasting method	26
3.2.2	ARIMA method	27
3.2.3	Forecasting with LSTM methods	28
3.3	Web API	36
4	Discussion	37
5	Conclusions and Future Works	38
5.1	Conclusions	38
5.1.1	Summary of findings	38
5.1.2	Limitations of Study	38
5.1.3	Recommendations	39
5.1.4	Acknowledgements	39
5.2	Ideas for Future Work	39
A	Appendix	42

List of Figures

2.1	AutoCorrelation	9
2.2	Partial Correlation	10
2.3	Lag Factor Concept	10
2.4	AR equation	11
2.5	MA equation	12
2.6	Combined Equation	12
2.7	Transformation	13
2.8	Feed forward RNN	15
2.9	Back Propagation Through time feed forward RNN	16
2.10	The framework considered while carrying out the project	19
3.1	Box plot of the Amoxycilin Caps drug	20
3.2	Box plot of Ampicloxa drug	21
3.3	Box plot of Ceftriaxone Injections	21
3.4	Box plot of Ciprofloxacin drug	22
3.5	Box plot of Cotrimoxazole drug	22
3.6	Illustration plot of the trend and rolling mean for Amoxycilin Caps.	23
3.7	Illustration plot of the trend and rolling mean for Ceftriaxone Injections. . .	23
3.8	Illustration plot of the trend and rolling mean for Cotrimoxazole drugs. . . .	24
3.9	Illustration plot of the trend and rolling mean for Ampicloxa drugs.	24

3.10	Illustration plot of the trend and rolling mean for Ampicloxa drugs.	25
3.11	Summary workflow chart.	25
3.12	Illustration plot of the Amoxycilin naïve forecasting results.	26
3.13	Illustration plot of the Ceftriaxone naïve forecasting results.	26
3.14	Illustration plot of the Cotrimoxazole naïve forecasting results.	27
3.15	Illustration plot of the Ampicloxa naïve forecasting results.	27
3.16	Illustration plot of the Ciprofloxacin naïve forecasting results.	27
3.17	Installing the environment Variables.	28
3.18	Vanilla LSTM model Graphical Layout.	29
3.19	Vanilla LSTM model Summary.	29
3.20	Vanilla LSTM model results for Amoxycilin Caps.	29
3.21	Vanilla LSTM model results for Ceftriaxone Injections.	30
3.22	Vanilla LSTM model results for Cotrimoxazole Tabs.	30
3.23	Vanilla LSTM model results for Ampicloxa Tabs.	30
3.24	Vanilla LSTM model results for Ampicloxa Tabs.	31
3.25	Stacked LSTM model Graphical Layout.	31
3.26	Stacked LSTM model Summary.	32
3.27	Stacked LSTM model results for Amoxycilin Tabs.	32
3.28	Stacked LSTM model results for Ceftriaxone Tabs.	32
3.29	Stacked LSTM model results for Cotrimoxazole Tabs.	33
3.30	Stacked LSTM model results for Ampicloxa Tabs.	33
3.31	Stacked LSTM model results for Ciprofloxacin Tabs.	33
3.32	Bidirectional LSTM model Graphical Layout.	34
3.33	Bidirectional LSTM model Summary.	34
3.34	Bidirectional LSTM model results for Amoxycilin Tabs.	34

3.35	Bidirectional LSTM model results for Ceftriaxone Injections.	35
3.36	Bidirectional LSTM model results for Cotrimoxazole Tabs.	35
3.37	Bidirectional LSTM model results for Ampicloxa Tabs.	35
3.38	Bidirectional LSTM model results for Ciprofloxacin Tabs.	36
3.39	Web App API.	36
4.1	MSE overall results.	37
A.1	My Naive forecast code Base on Google Colab	42
A.2	My Vanilla LSTM forecast code Base on Google Colab	43
A.3	My Stacked LSTM forecast code Base on Google Colab	43
A.4	Results of the machine learning forecast	44

List of Tables

List of Abbreviations

ARIMA	Auto Regression Integrated Moving Average
LSTM	Long Short Term Memory
MSE	Mean Square Error
KPI	Key Performance Indicator
API	Application Programming Interface
GPU	Graphical Processing Unit
BPTT	Back Propagation Through Time
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Network

Chapter 1

Introduction

This chapter gives a brief explanation of the project background, problem statement, justification, objectives, and a summary of the methodology.

1.1 Project Background

Soteria Pharmacy is a licensed Pharmacy outlet located in Luweero District and has been in operation for 4 years now. It is directly opposite the Luweero National referral hospital and close to the main road for quick access by the residents of the area. In March 2018, Kampala and Mbale had 1029 private pharmacies with valid operating licenses from Uganda's MRA, of which 622 were retail (licensed to dispense small quantities to patients only), 175 were wholesale (licensed to sell in bulk to retailers only) and 232 were dual (licensed for both wholesale and retail supply). [14] One of the responsibilities of pharmacies in Uganda is to have a minimum stock of medicines. This ensures patients can have it when prescribed. The adoption of technological tools to help in inventory tracking by pharmacies has enabled us collect vast amounts of data but all this information has not been put to full use as it only presents reports for past occurrences.

In addition, pharmacies need to get a good forecast of the medication needs due to the short term validity of many medicines and the need to control stock levels. This avoids excessive costs and loss of customers due to stock outages.

A good sales forecast is usually associated with striking a balance between stock costs and adequate satisfaction of customer demand. People act on the basis of forecasting models whether they are on paper or in their heads. You are better off quantifying these estimations so you can discuss them rationally as opposed to making them based on intuition.

To specific case of pharmacies in Uganda, the problem is of particular importance due to the short cycle life of many products and the importance of quality which is in turn strongly linked to public health.

During our research study of Soteria Pharmacy procurement process with an interview of Ms.Brenda the incharge of this, we realised she uses personal judgement of current stock levels athand and the rate at which people come in to ask for a ceratin drug then determine how much more stock should be purchased.Given her difficulties in accurately predicting the future sales,this report explores the product sales forecast at the indiviual level of this Pharmacy.I made a forecast for 5 sold drugs.The forecast was based on analysis of historical data for a period of 24 months and future results analyses of 50 days determined.ARIMA and LSTM methods were used in the sales forecast of which the best model was recommended including a conclusive way of improving our results.

People act on the basis of forecasting models whether they are on paper or in their heads. One is better off quantifying these estimations so as to discuss them rationally as opposed to making them based on intuition. For pharmaceutical distribution companies, it is essential to get good estimates of drugs, due to the short shelf life of many medicines and the need to control stock levels, to avoid excessive inventory costs while guaranteeing customer demand satisfaction, and thus decreasing the possibility of a loss of customers due to stock outages. Stock management, transportations, and financial spends contain a high percentage of total pharmaceutical companies' expenses. As a rule, companies pay immediately when buy medications from manufacturers, and then sales compensate these spend gradually. This gap is a danger of unplanned expenses to occur. Therefore, the majority of distributors in this industry look for modern and precise forecasting methods of future sales to decrease purchase and storage costs and to increase profit by meeting clients' needs timely. Common existing forecasting methods are ineffective for pharmaceutical companies because these methods require a large dataset of each medication sales. In its turn, medications are constantly replaced by analogs or refreshed to enhance pharmacological effects or reduce collateral effects.

1.2 Problem Statement

Our solution is addressing the high volumes of expiring drugs before purchase in pharmacies. We anticipate that with an insight into the future sales, you can make the right purchases which in turn minimises the inventory expenditure to incarcerate the expired medicines. Currently pharmacies are using intuition from domain experts to identify the amount of

stock to purchase based on their current time of work in a particular place who may not be able to process large amounts of information to cater for promotional offer sales, decision buys from wholesalers, rotational nature of pharmacy personnel especially in large pharmacies with like first pharmacy or even in case of a new entrant, may find a steep start up adjustment curve from wrong estimations.

1.3 Aims and Objectives

The main aim of the research project was to precisely predict sales of drugs and medical supplies.

The specific objectives were:

- To collect datasets of previous sales and purchases from Soteria Pharmacy.
- To train and validate datasets with ARIMA and LSTM models.
- To optimize the best model algorithm for accurate performance.
- To develop a web API for pharmacies.

1.4 Contributions of the Project

The following major contributions have been accomplished during the course of this research:

1. Performed a data cleaning task, analysis and training with the various Machine learning models and came up with a graphical output.
2. Developed a robust neural network model that could accurately predict future sales for a period of 50 days using a 2 year period worth of sales and purchases information.
3. Proposed a multivariate input approach with other characteristics such as precipitational weather information and promotional sales.

1.5 Organization of the Report

This report has 3 chapters.

- Introduction

- Analysis and Findings which characterizes essential concepts, an overview of the categories and description of the main methods associated with the several techniques of time series analysis.
- Conclusion presents some final considerations and future work proposals.

Chapter 2

Literature Review

2.1 Introduction

Health forecasting is a novel area of forecasting, and a valuable tool for predicting future health events or situations such as demands for health services and healthcare needs. It facilitates preventive medicine and health care intervention strategies, by pre-informing health service providers to take appropriate mitigating actions to minimize risks and manage demand.

2.2 Literature Review

Consequently, it is required to build an accurate forecasting model of pharmaceutical preparations sales using one of the machine learning methods taking into account constant medications refreshment and a lack of previous sales data [1].

Before digital technology dominated the world, the forecasting process was done manually by experienced individuals in the domain. This intuition required a lot of experience and was prone to error. Due to this reason, they started realizing the need for automating the pharmacy sales forecasting process. Thus, research and experiments were carried out with statistical, machine learning, deep learning, and ensemble techniques to achieve more accurate sales forecasts. [2] Algorithms to illustrate inherent laws in large amounts of data and to forecast future data patterns have been researched since 1920. However no breakthroughs till 1980. [3] In the deep learning world, state-of-the-art performance has gained a good reputation in fields like object recognition, [4] speech recognition [5], natural language processing [6], physiological effect modeling [7], and many others. More recently papers on time-series prediction or classification with deep neural networks have been reported. [8–11]

Aimed at obtaining inherent laws of historical data series in pharmaceutical sales, and forecasting the demand, controlling the inventory, reducing the costs, and improving the service level, this paper designs research on the data records from a pharmacy and presents different forecasting algorithms. The testing results provide support for the fact that these algorithms greatly improve the forecast accuracy from the naive forecasting method.

2.3 Data Collection

Data mining is the search of relationships, patterns, or models that are implied on data stored in large databases. It can be seen as the process of exploiting large amounts of data for the purpose of identification of consistent standards, such as rules of association or time sequences to detect systematic relationships between variables. Uses algorithms to discover rules, identify factors and key trends, discover hidden patterns and relationships in large databases; this information after being interpreted, is used to support decision making organizational.

2.4 Machine Learning

2.4.1 Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to

navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes. [12]

2.5 Time Series Forecasting

2.5.1 Introduction

A time series is a sequence where a metric is recorded over regular time intervals. Depending on the frequency, a time series can be of yearly (ex: annual budget), quarterly (ex: expenses), monthly (ex: air traffic), weekly (ex: sales qty), daily (ex: weather), hourly (ex: stocks price), minutes (ex: inbound calls in a call center) and even seconds wise (ex: web traffic).

Forecasting is the next step where you want to predict the future values the series is going to take.

Now forecasting a time series can be broadly divided into two types.

If you use only the previous values of the time series to predict its future values, it is called Univariate Time Series Forecasting.

And if you use predictors other than the series (a.k.a exogenous variables) to forecast it is called Multi Variate Time Series Forecasting

A time series is usually modelled through a stochastic process $Y(t)$, i.e. a sequence of random variables. In a forecasting setting we find ourselves at time t and we are interested in estimating $Y(t+h)$, using only information available at time t . [13] Anything that is observed sequentially over time is a time series. When forecasting time series data, the aim is to estimate how the sequence of observations will continue into the future. Forecasting involves taking models fit on historical data and using them to predict future observations.

Descriptive models can borrow for the future (i.e. to smooth or remove noise), they only seek to best describe the data. The skill of a time series forecasting model is determined by its performance at predicting the future. This is often at the expense of being able to explain why a specific prediction was made, confidence intervals and even better understanding the underlying causes behind the problem.

Components of Time Series

Time series analysis provides a body of techniques to better understand a dataset.

Perhaps the most useful of these is the decomposition of a time series into 4 constituent parts:

1. Level. The baseline value for the series if it were a straight line.
2. Trend. The optional and often linear increasing or decreasing behavior of the series over time.
3. Seasonality. The optional repeating patterns or cycles of behavior over time.
4. Noise. The optional variability in the observations that cannot be explained by the model. All time series have a level, most have noise, and the trend and seasonality are optional.

2.5.2 ARIMA

ARIMA, short for ‘Auto Regressive Integrated Moving Average’ is actually a class of models that ‘explains’ a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

An ARIMA model is characterized by 3 terms: p , d , q

where,

p is the order of the Auto Regression (AR) term

q is the order of the Moving Average (MA) term

d is the number of differencing required to make the time series stationary

A pure Auto Regressive (AR only) model is one where Y_t depends only on its own lags. That is, Y_t is a function of the ‘lags of Y_t ’.

Likewise a pure Moving Average (MA only) model is one where Y_t depends only on the

lagged forecast errors.

ARIMA, short for ‘AutoRegressive Integrated Moving Average’, is a forecasting algorithm based on the idea that the information in the past values of the time series can alone be used to predict the future values. Autoregressive models often make sense for business data. They express the fact that where you go depends partly on where you are and partly on what happens to you along the way (as expressed by the random noise component). Forecasting with an autoregressive process is done with predicted values from the estimated regression equation after going forward one unit in time.

Correlation and Lag Factor

Let’s consider an example of predicting the price of gas at any particular day, say Sunday. It’s obvious and easy to think of, that Sunday’s price will be dependent on Saturday’s price. Now let’s consider the effect of Friday’s price on Sunday’s price.

S_t = Price of Gas on Sunday
 S_{t-1} = Price of Gas on Saturday
 S_{t-2} = Price of Gas on Friday

As Sunday’s price is dependent on Saturday’s (2), which in-turn is dependent on Friday’s (1), there could be an indirect transitive relation between price on Friday and price on Sunday. Aforementioned relations are visualised in the diagram below.

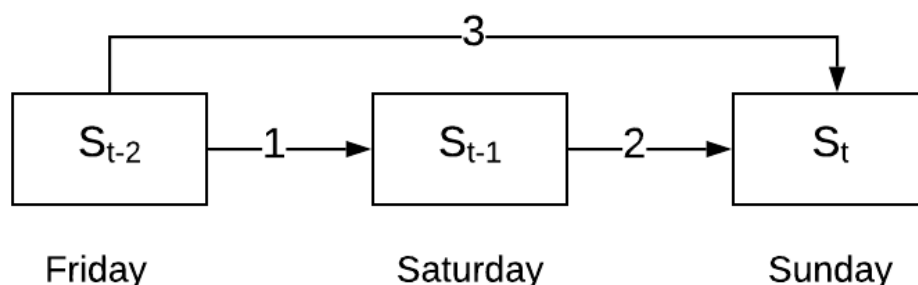


Figure 2.1: AutoCorrelation

Apart from these, there exists one more relation which might be difficult to imagine at first i.e. the direct effect of Friday's price on Sunday's price (3). This mutual relationship between day prices here, is called correlation. If values of two variables increase (or decrease) together then they are said to have a positive correlation. If the value of one variable increases and the other decreases (or vice versa) then they have a negative correlation.

Correlation is an important concept of time series and it has 2 flavours, Auto-correlation, which considers both direct and indirect effects (illustrated in the diagram above) and,

Partial Auto-correlation, which considers only the direct effects like in the figure below

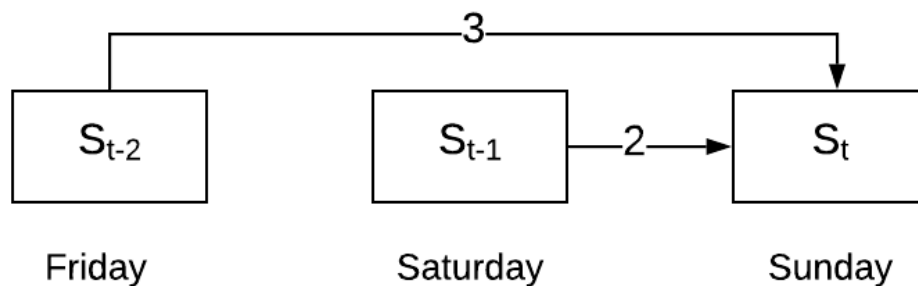


Figure 2.2: Partial Correlation

Lag Factor

As we have considered the effect on Sunday's price based on the price 2 days prior(i.e. Friday)... the lag factor here is 2.

Correlation(Sunday,Saturday) => lag factor of 1
 Correlation(Sunday, Friday) => lag factor of 2
 Correlation(Sunday, Thursday) => lag factor of 3
 ...

Figure 2.3: Lag Factor Concept

Auto-correlation and Partial Auto-correlation are important to understand before moving on to ARIMA as they are crucial for selecting the right parameters for your model.

AutoRegression(AR)

Auto Regressive (AR) model is a specific type of regression model where, the dependent variable depends on past values of itself. This necessarily means that the current values are correlated with values in the previous time-steps. And more specifically, the type of correlation here is partial auto-correlation. The equation for the AR model is shown below

$$Y_t = \beta_1 + \Phi_1 Y_{t-1} + \Phi_2 Y_{t-2} + \dots + \Phi_p Y_{t-p}$$

Figure 2.4: AR equation

The respective weights of the corresponding lagged observations are decided by the correlation between... that lagged observation and the current observation. If the correlation is more, the weight corresponding to that lagged observation is high (and vice-versa).

Notice the (p) in the equation...

This (p) is called the lag order. It represents the number of prior lag observations we include in the model i.e. the number of lags which have a significant correlation with the current observation.

Moving Average (MA)

Moving Average (MA) model works by analysing how wrong you were in predicting values for the previous time-periods to make a better estimate for the current time-period.

Basically, this model factors in errors from the lagged observations. The effects of these previous(lagged) observation errors, on the current observation, depends on the auto-correlation between them. This is similar in sense as the AR model which considers the partial auto-correlations.

$$Y_t = \beta_2 + \omega_1 \epsilon_{t-1} + \omega_2 \epsilon_{t-2} + \dots + \omega_q \epsilon_{t-q} + \epsilon_t$$

Figure 2.5: MA equation

The epsilon terms represent the errors observed at respective lags and the weights are calculated statistically depending on the correlations. Notice the (q) in the equation. . . (q) represents the size of the moving window i.e. the number of lag observation errors which have a significant impact on the current observation. Its similar to the lag order(p), but it considers errors instead of the observations themselves.

MA model supplements the AR model by taking into considerations, the errors from the previous time-periods thereby helping to get a better estimate. When we combine the AR and MA equation we get

$$Y_t = (\beta_1 + \beta_2) + (\Phi_1 Y_{t-1} + \dots + \Phi_p Y_{t-p}) + (\omega_1 \epsilon_{t-1} + \dots + \omega_q \epsilon_{t-q} + \epsilon_t)$$

Figure 2.6: Combined Equation

Stationary

The models we've discussed so far (AR and MA) assume that the series is stationary. This is also means that "stationarity" is a necessary condition to take advantage of these models for any time series.

Basically, for a time series to be stationary, it should satisfy the following 3 conditions. . .

Mean is constant

Standard Deviation is constant

Seasonality doesn't exist

In most cases, these conditions can be visually analysed by studying the plot against time.

You will come across a lot of series which are clearly not stationary.

Does this mean that forecasting cannot be applied in these cases ?

Well... this is where the sub-acronym I comes in...

Integration

Lets say, for example, you come across a series with a “non-constant” mean. Its clearly observable that the mean increases over time i.e. the series is not stationary.

If we could somehow eliminate this upward trend we'll be good to go.

One way to do this is to consider the differences between consecutive timesteps. This is equivalent to performing a transformation of the form...

$$Z_t = Y_{t+1} - Y_t$$

Figure 2.7: Transformation

Applying this transformation we get the following trend with an observable constant mean. The standard deviation is constant as well and seasonality doesn't exist i.e. the series is now stationary.

I stands for Integrated (though it has nothing to do with integration). It just means that, instead of predicting the time series itself we will predict the differences of the series from one time step to the next time step. Notice that here we have taken the first order difference i.e. a single phase of differencing of consecutive terms. This is could done multiple times to make the series stationary. [15]

2.5.3 Forecasting with LSTM methods

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

Recurrent neural networks are different from traditional feed-forward neural networks.

This difference in the addition of complexity comes with the promise of new behaviors that the traditional methods cannot achieve.

Long-term forecasting validation has been done with three LSTM configurations: Vanilla LSTM, Stacked LSTM and Bi-directional LSTM. Relu activation function was used, optimizer was Adam and loss function was Mean Squared Error. The best results were achieved with training the model in 400 epochs. Before fitting, all data was standardized (rescaled in interval -1,1) and transformed to data for supervised problem.

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones. [16]

Recurrent neural networks (RNN) are a class of neural networks that are helpful in modeling sequence data. Derived from feedforward networks, RNNs exhibit similar behavior to how human brains function. Simply put: recurrent neural networks produce predictive results in sequential data that other algorithms can't. RNN's and feed-forward neural networks get their names from the way they channel information.

In a feed-forward neural network, the information only moves in one direction — from the input layer, through the hidden layers, to the output layer. The information moves straight through the network and never touches a node twice.

Feed-forward neural networks have no memory of the input they receive and are bad at predicting what's coming next. Because a feed-forward network only considers the current input, it has no notion of order in time. It simply can't remember anything about what happened in the past except its training.

A recurrent neural network, however, is able to remember those characters because of its

internal memory. It produces output, copies that output and loops it back into the network. Simply put: recurrent neural networks add the immediate past to the present.

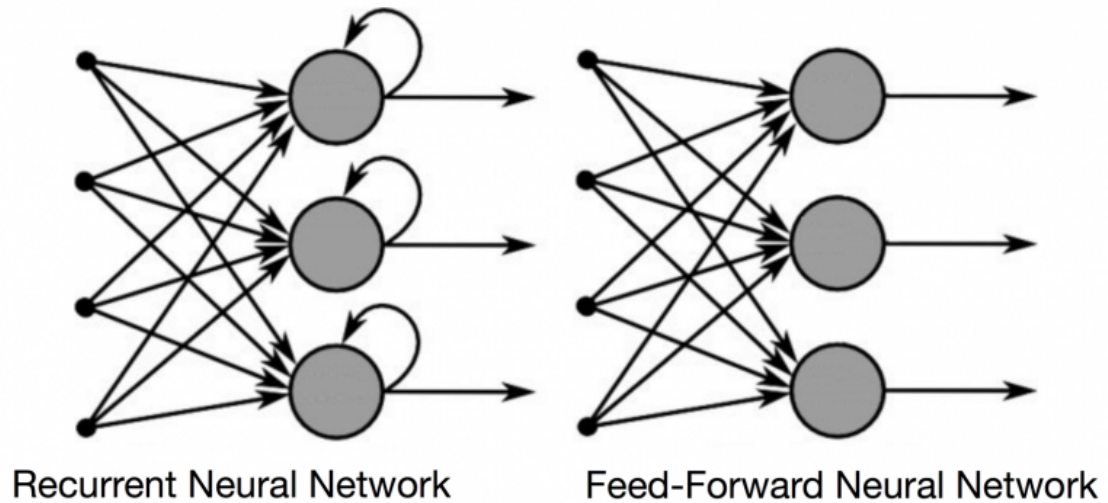


Figure 2.8: Feed forward RNN

A feed-forward neural network assigns, like all other deep learning algorithms, a weight matrix to its inputs and then produces the output. Note that RNNs apply weights to the current and also to the previous input. Furthermore, a recurrent neural network will also tweak the weights for both through gradient descent and backpropagation through time (BPTT).

Back Propagation through time(BPTT)

In neural networks, you basically do forward-propagation to get the output of your model and check if this output is correct or incorrect, to get the error. Backpropagation is nothing but going backwards through your neural network to find the partial derivatives of the error with respect to the weights, which enables you to subtract this value from the weights.

Those derivatives are then used by gradient descent, an algorithm that can iteratively minimize a given function. Then it adjusts the weights up or down, depending on which decreases the error. That is exactly how a neural network learns during the training process.

The image below illustrates the concept of forward propagation and backpropagation in a feed-forward neural network:

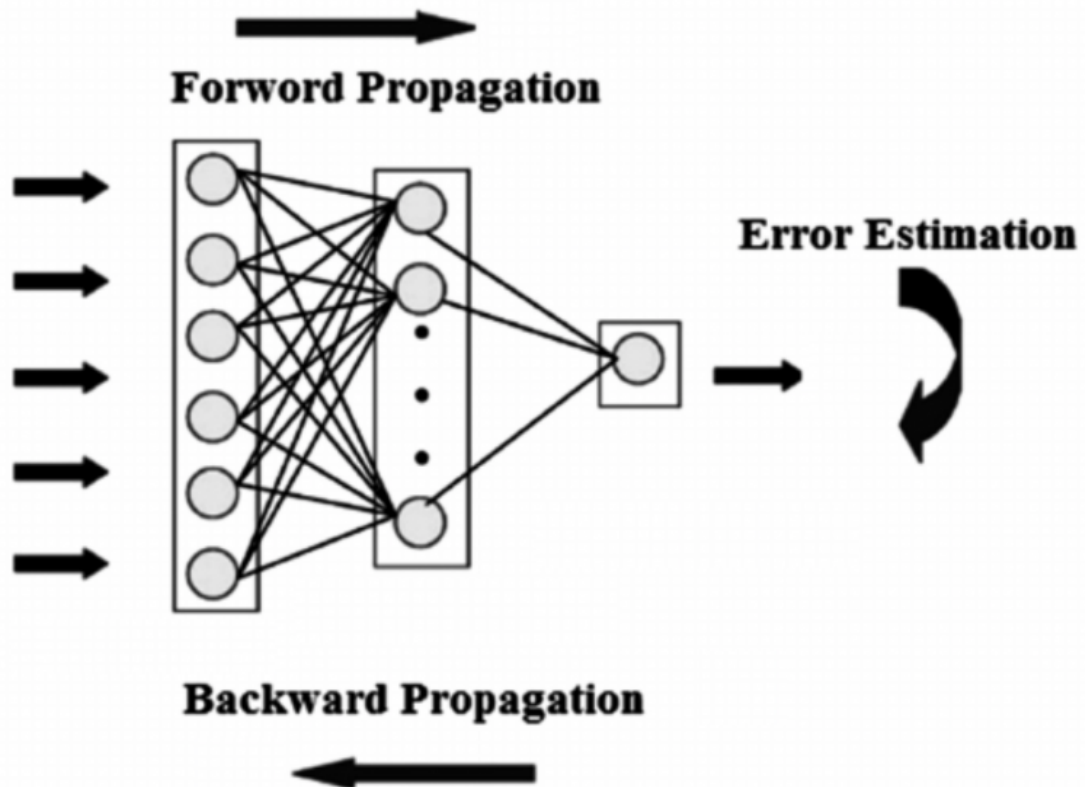


Figure 2.9: Back Propagation Through time feed forward RNN

LSTM Networks

Long Short Term Memory networks usually just called LSTMs are a special kind of RNN, capable of learning long term dependencies.

They were introduced by Hochreiter and Schmidhuber (1997), and were refined and popularized by many people in following work.

1.They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

Architecture of LSTMs

The functioning of LSTM can be visualized by understanding the functioning of a news channel's team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take either of the three steps.

Let's say, we were assuming that the murder was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what do you do? You immediately forget the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You input this information into your news feed, right?

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and output the relevant things to your audience. Maybe in the form of "XYZ turns out to be the prime suspect." [17]

A typical LSTM network is comprised of different memory blocks called cells (the rectangles that we see in the image). There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates. Each of them is being discussed below.

Vanilla LSTM

A Vanilla LSTM is an LSTM model that has a single hidden layer of LSTM units, and an output layer used to make a prediction. [18]

Stacked LSTM

A Stacked LSTM architecture can be defined as an LSTM model comprised of multiple LSTM layers. An LSTM layer above provides a sequence output rather than a single value output to the LSTM layer below. Specifically, one output per input time step, rather than one output time step for all input time steps.

Bidirectional LSTM

In bidirectional RNNs, future states can also influence the present state and the past states by allowing information to flow backward. Past outputs are updated as needed depending on the new information received.

2.5.4 Website Application Programming Interface(API)

An application programming interface (API) is a computing interface that defines interactions between multiple software intermediaries.

StreamLit API

Streamlit is an open-source app framework for Machine Learning and Data Science teams. Create beautiful data apps . [19] It is a python framework and works with Python version 3.6 - 3.8 .

2.6 Summary

The results from the various models are to be verified using the mean square error estimator.(MSEE) The MSE is the average of the squared differences in the values between the actual real outcome and the predicted outcome using our model. I use the MSE as a key performance indicator in order to judge if my model is performing better than those already in existence.

This project was conducted through a framework for health forecasting. The key processes involved in developing a general health forecasting service, are illustrated below

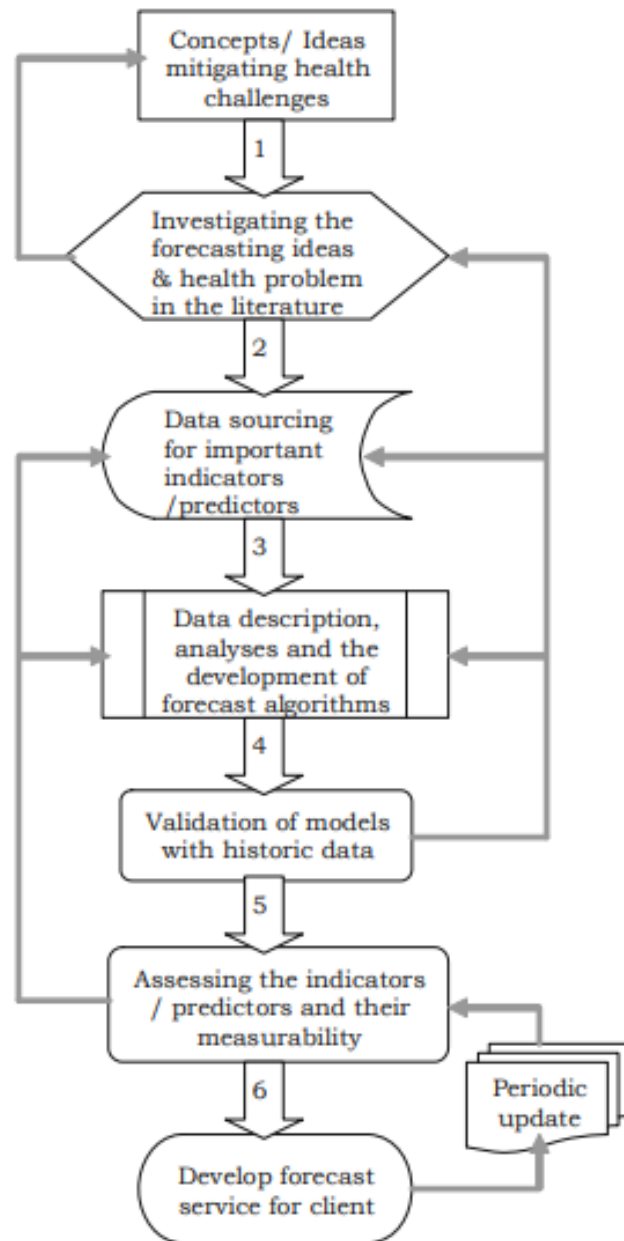


Figure 2.10: The framework considered while carrying out the project

Chapter 3

Methodology

3.1 Materials and Methods

We analyzed daily sales data for a period of 2 years by arranging it in a proper format to easily perform predictions with the same data file using Microsoft excel. It was imported into the google colab jupyter notebook by a clone from GitHub, then using pandas passed into a variable for further processing. From box-plots, Cotrimoxazole had more outliers than the rest of the drugs which made it harder to predict future sales.

We also checked for presence of seasonality and trend with a 30 day and 365 day rolling mean graph.

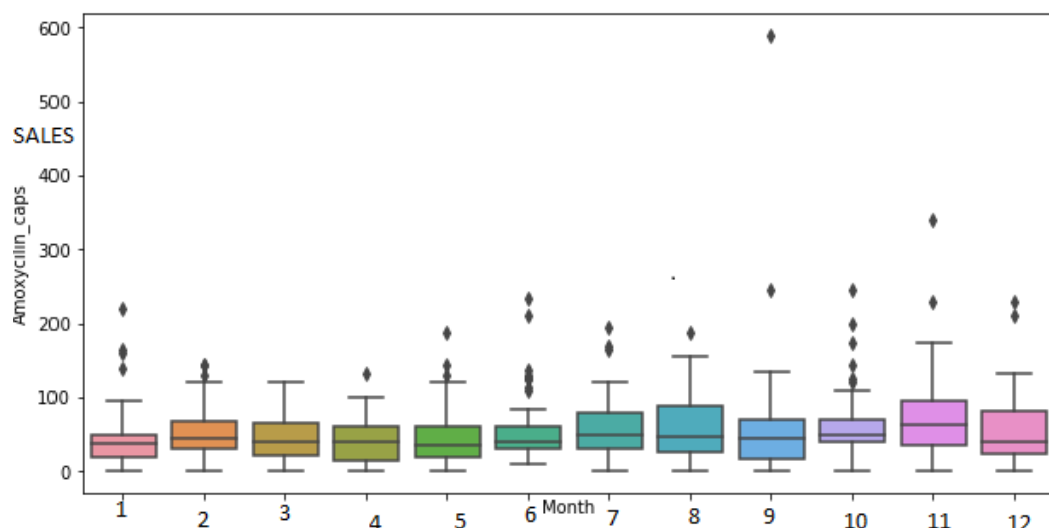


Figure 3.1: Box plot of the Amoxycilin Caps drug

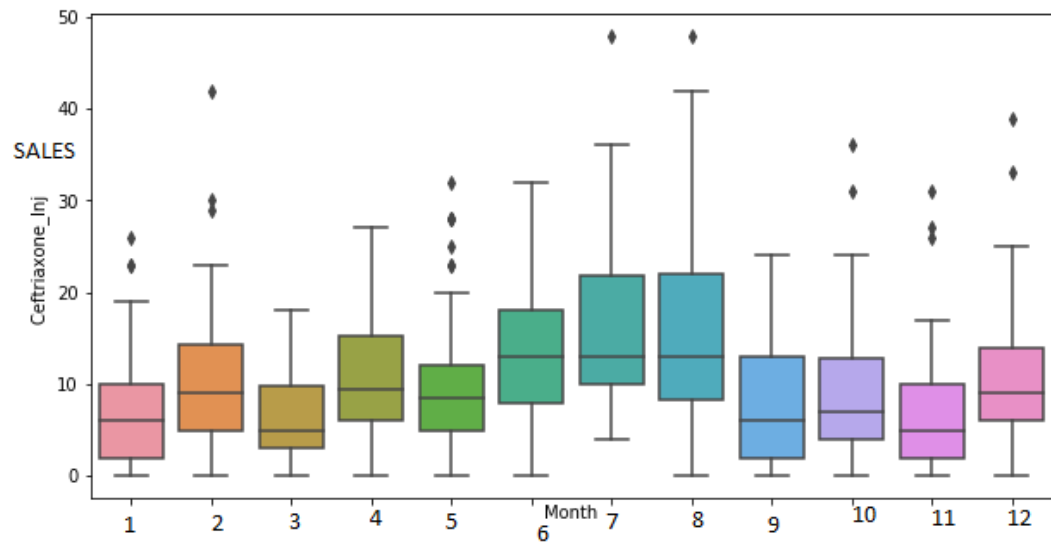


Figure 3.2: Box plot of Ampicloxa drug

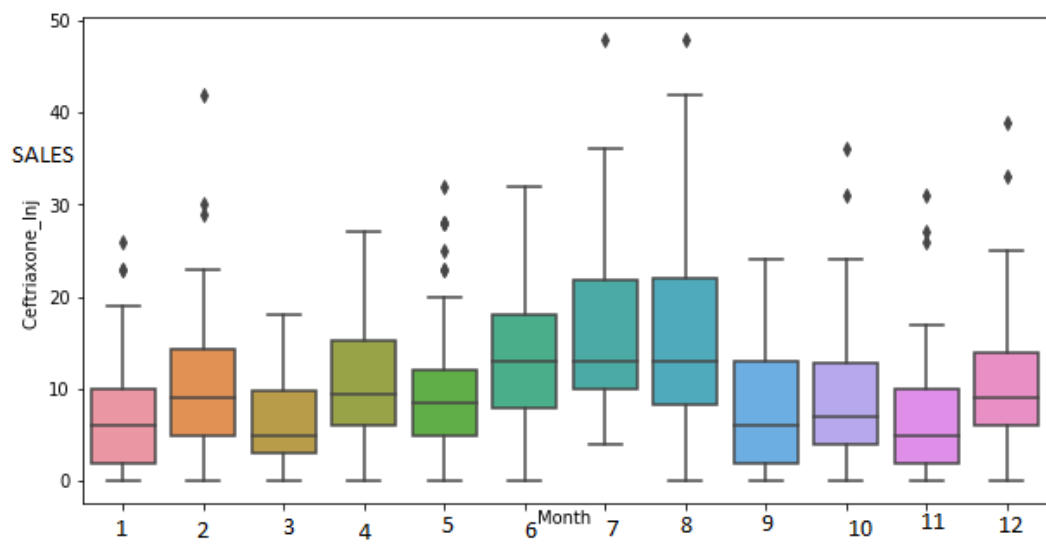


Figure 3.3: Box plot of Ceftriaxone Injections

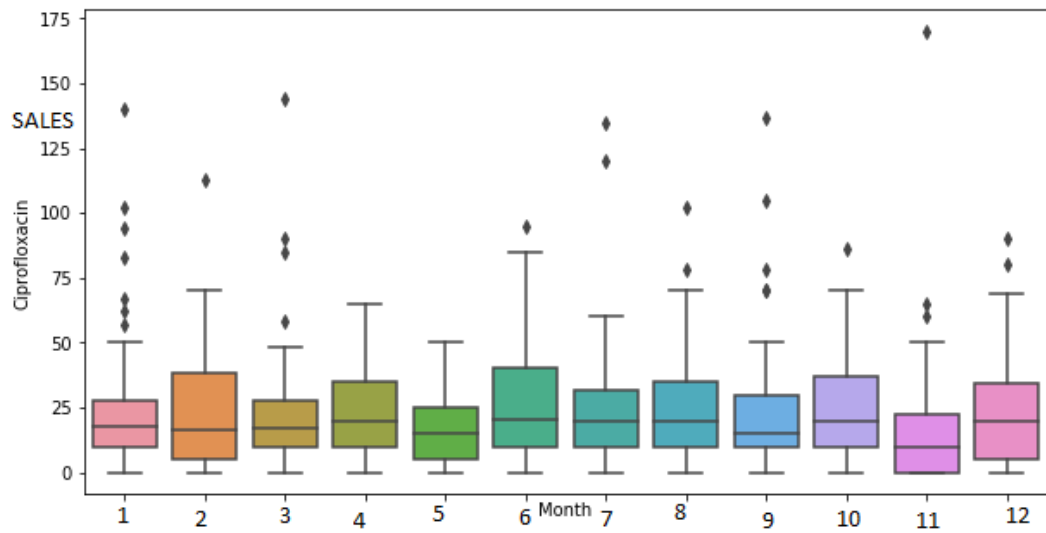


Figure 3.4: Box plot of Ciprofloxacin drug

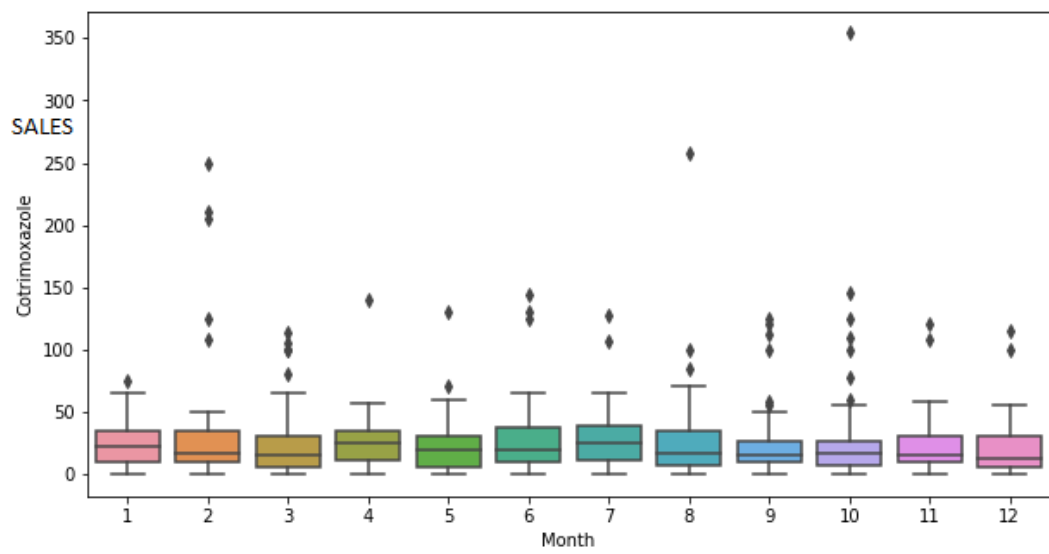


Figure 3.5: Box plot of Cotrimoxazole drug

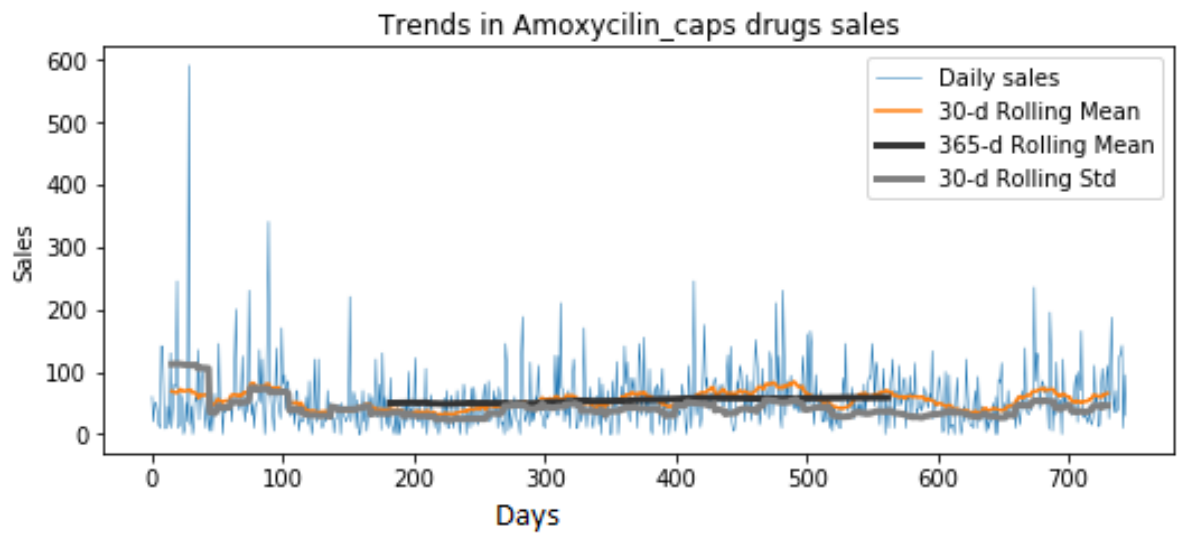


Figure 3.6: Illustration plot of the trend and rolling mean for Amoxycilin Caps.

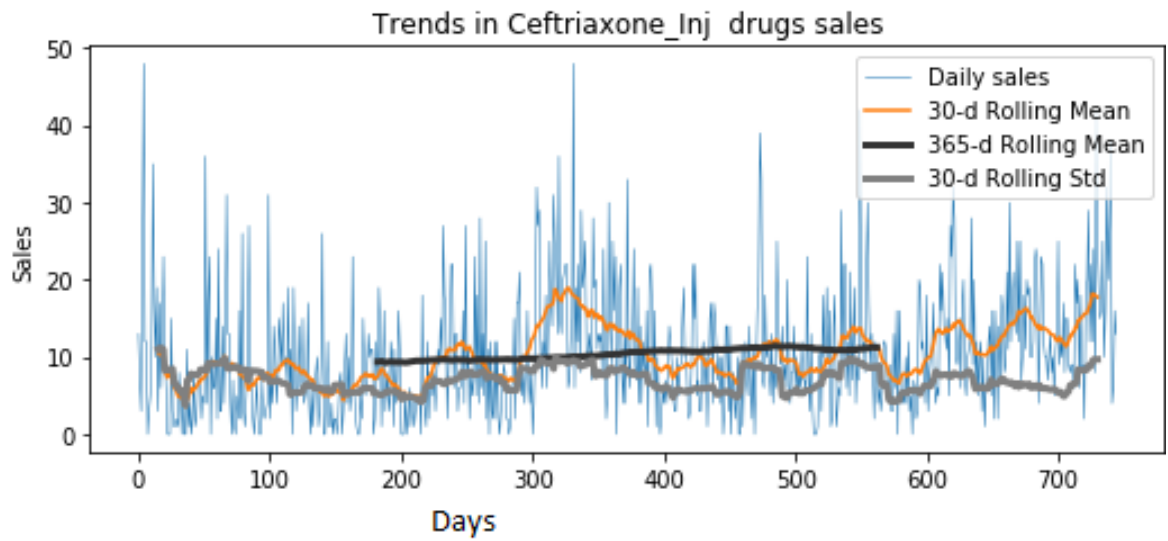


Figure 3.7: Illustration plot of the trend and rolling mean for Ceftriaxone Injections.

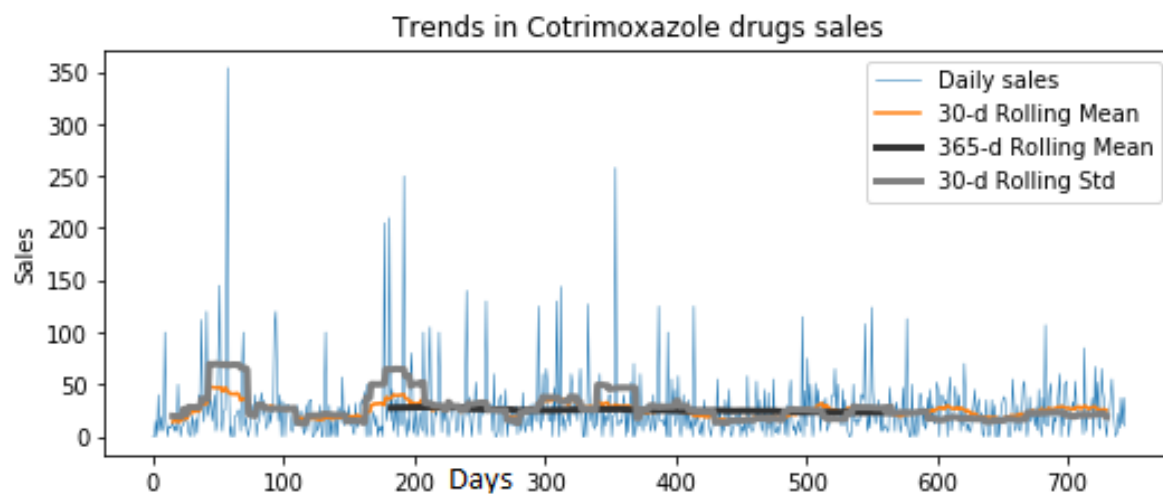


Figure 3.8: Illustration plot of the trend and rolling mean for Cotrimoxazole drugs.

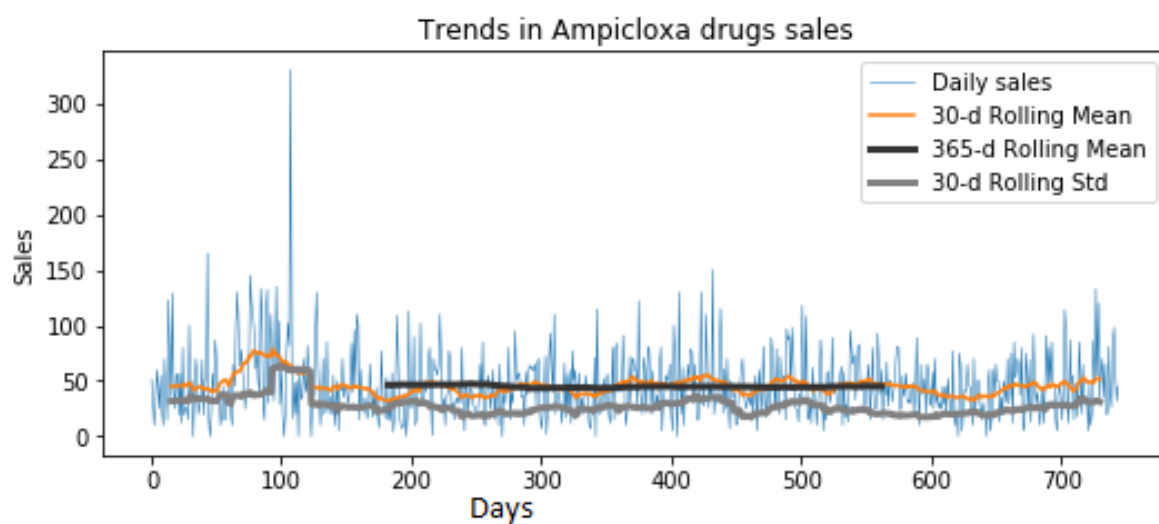


Figure 3.9: Illustration plot of the trend and rolling mean for Ampicloxa drugs.

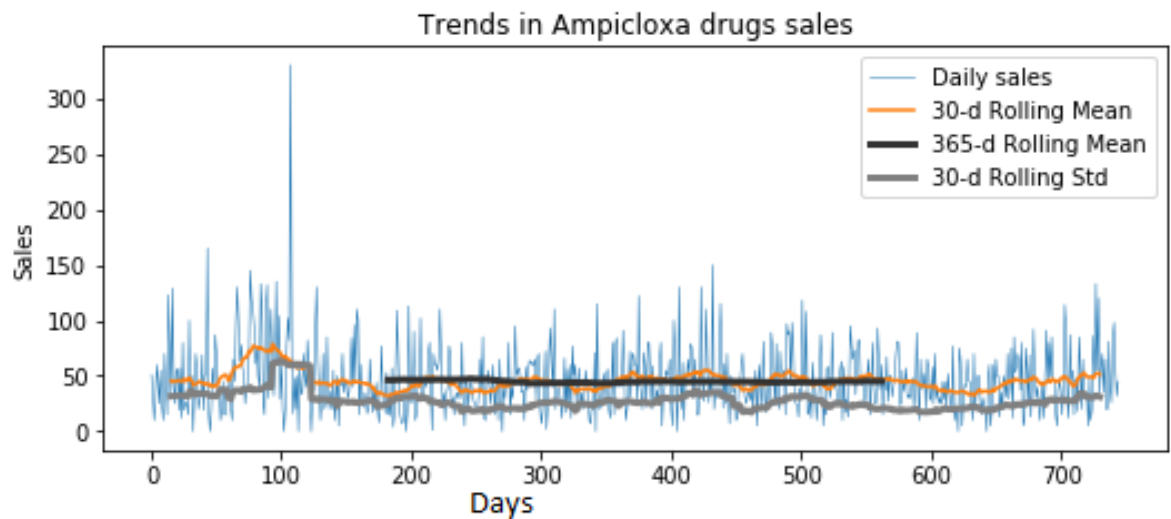


Figure 3.10: Illustration plot of the trend and rolling mean for Ampicloxa drugs.

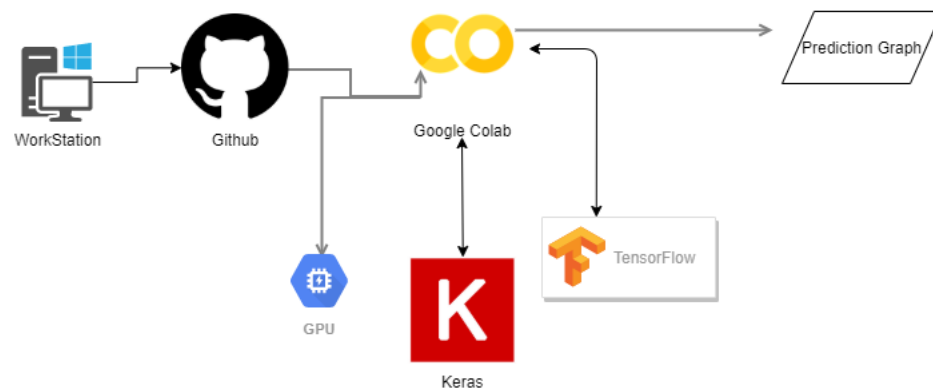


Figure 3.11: Summary workflow chart.

Included Components

1. Keras: The Python Deep Learning library.
2. Tensorflow: An open-source software library for Machine Intelligence.
3. Github: An open-source platform to store code.
4. Graphical Processing Unit (GPU): A processing unit for computationally intense algorithms.

3.2 Data and Results

3.2.1 Naïve forecasting method

This was the baseline method since it is what most pharmacists use during the normal day to day operation of their business for drug replenishments. Below are the results in graphical format. Our forecast values were based on the last 50 days of our 2 year period data sample. This uses a day time shift of the previous sales to predict future sales. As a case study for soteria pharmacy in luweero district, they normally use intuition similar to this whereby the next day purchases are made basic on the previous day sales and hoping the same trend. With our time day shift results , we modelled a naive forecasting scenario and were able to assume this as the result of what is currently being used considering most pharmacy procurement personnel use intuition. Figure 3.4 shows the plots obtained from the naive forecast scenario.

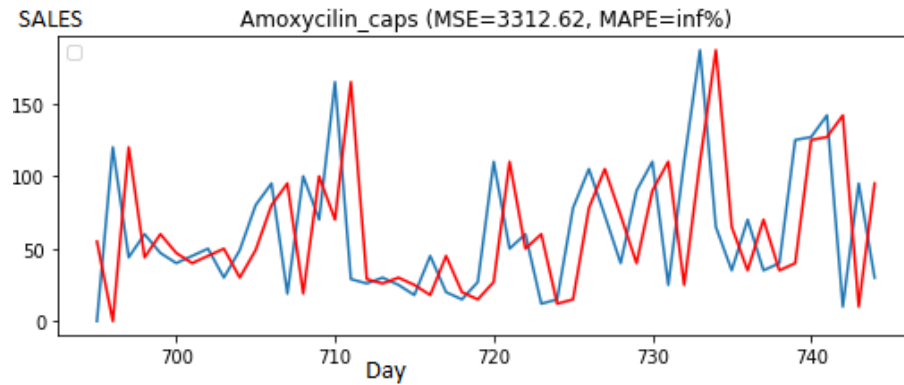


Figure 3.12: Illustration plot of the Amoxycilin naïve forecasting results.

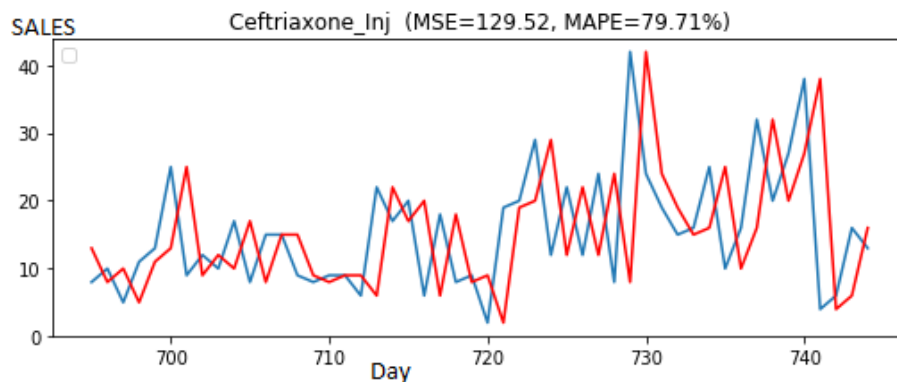


Figure 3.13: Illustration plot of the Ceftriaxone naïve forecasting results.

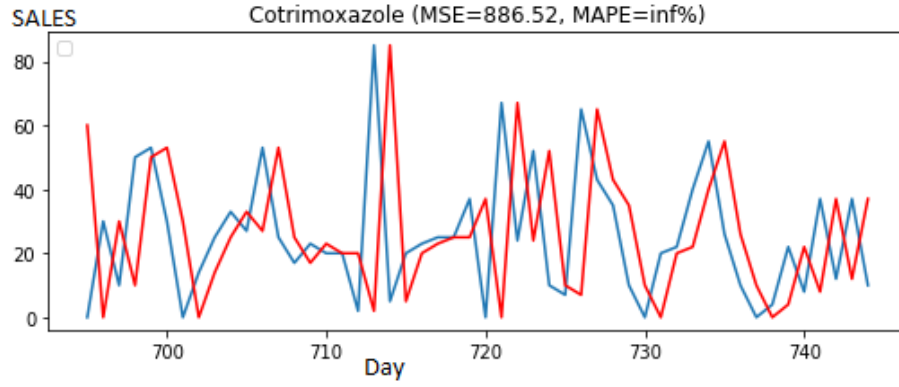


Figure 3.14: Illustration plot of the Cotrimoxazole naïve forecasting results.

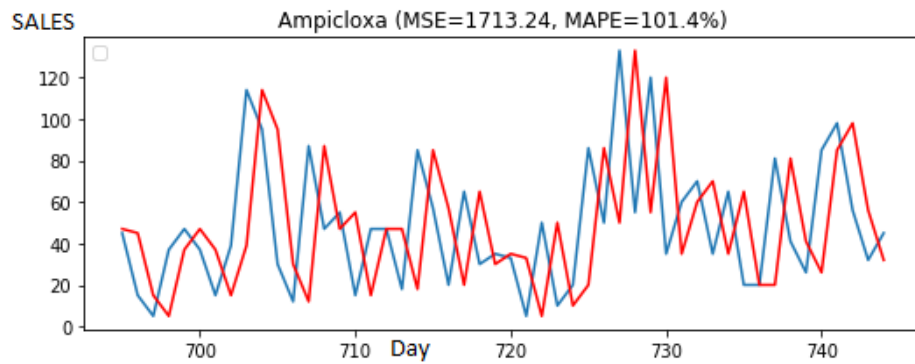


Figure 3.15: Illustration plot of the Ampicloxa naïve forecasting results.

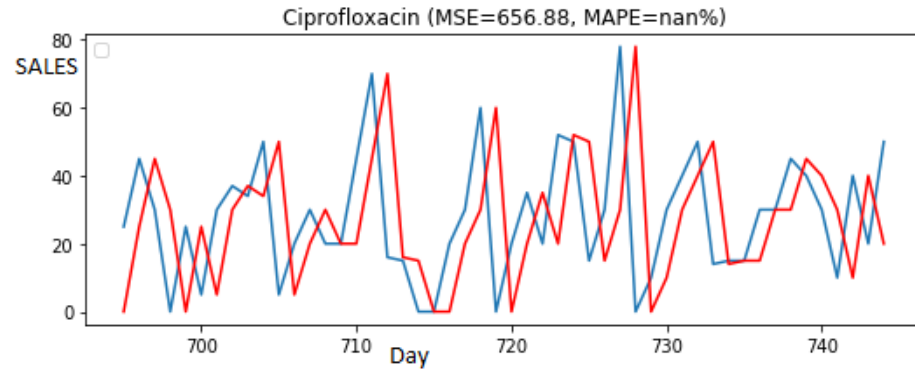


Figure 3.16: Illustration plot of the Ciprofloxacin naïve forecasting results.

3.2.2 ARIMA method

First, method `arma_order_select_ic` was used to determine initial p and q parameters. The method computes Akaike's Information Criterion (AIC) for many ARIMA models and chooses the best configuration. However, AIC is not used to score accuracy of the fore-

casting methods in this research. Mean squared error is used instead. For that, reason, grid search optimization method was applied, where different combinations of the hyper-parameters were used to calculate MSE and then, the combination producing the least MSE was chosen as optimal. Grid search optimization for rolling forecast produced the following best combinations of the hyper-parameters:

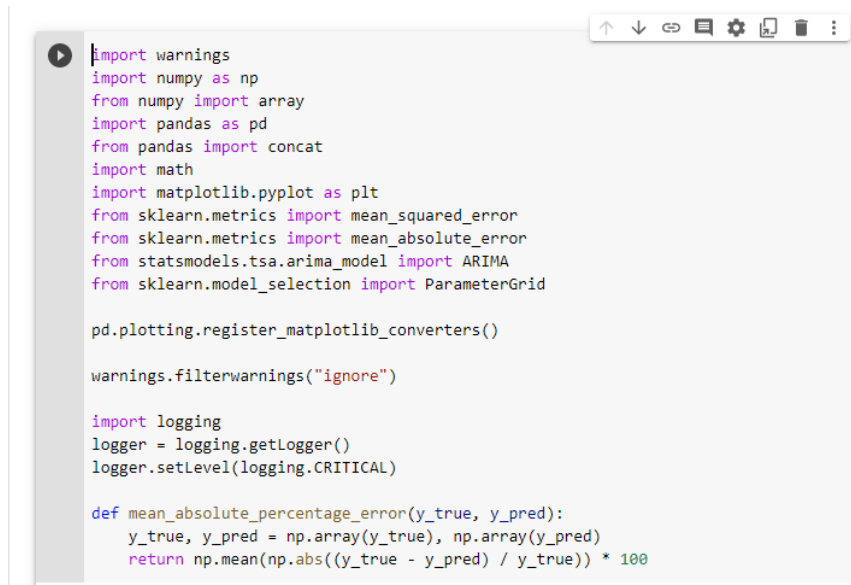
Amoxyclin_caps – Best ARIMA(1,0,0) MSE=1524.745

3.2.3 Forecasting with LSTM methods

My number of past observations tested in input sequences was 5.

I used the library Keras, which is a high-level API for neural networks and works on top of TensorFlow.

I imported the required dependencies as shown in the figure below.

A screenshot of a code editor window with a light gray background. The code is written in Python and includes imports for warnings, numpy, pandas, math, matplotlib, sklearn metrics, statsmodels, and sklearn model selection. It also shows the registration of matplotlib converters, filtering of warnings, and logging setup. A custom function 'mean_absolute_percentage_error' is defined at the bottom. The code is as follows:

```
import warnings
import numpy as np
from numpy import array
import pandas as pd
from pandas import concat
import math
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from statsmodels.tsa.arima_model import ARIMA
from sklearn.model_selection import ParameterGrid

pd.plotting.register_matplotlib_converters()

warnings.filterwarnings("ignore")

import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

Figure 3.17: Installing the environment Variables.

Vanilla LSTM

In order to get reproducible results in forecasting with LSTM, following values are fixed: seed value, 'PYTHONHASHSEED' environment variable, Python's, numpy's and Tensorflow's built-in pseudo-random generators. A new global Tensorflow session is configured. Below are the results from the vanilla LSTM model. This model is unidirectional in the sense that the current output is only influenced by the past states and the current input.

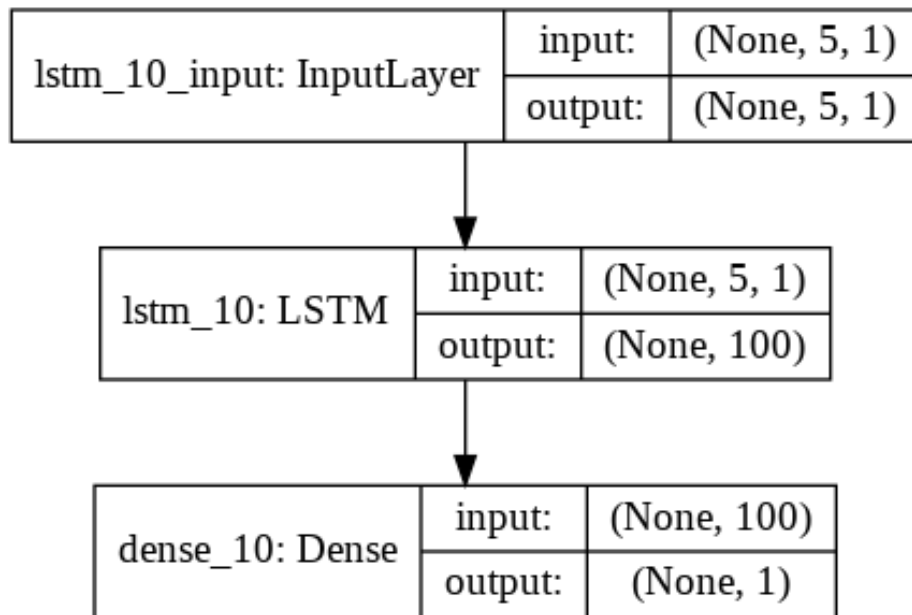


Figure 3.18: Vanilla LSTM model Graphical Layout.

Model: "sequential_10"

Layer (type)	Output Shape	Param #
lstm_10 (LSTM)	(None, 100)	40800
dense_10 (Dense)	(None, 1)	101
Total params: 40,901		
Trainable params: 40,901		
Non-trainable params: 0		
None		

Figure 3.19: Vanilla LSTM model Summary.

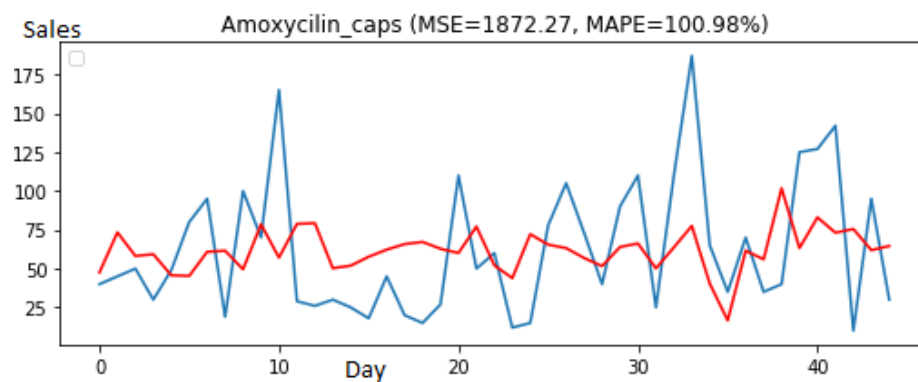


Figure 3.20: Vanilla LSTM model results for Amoxicilin Caps.

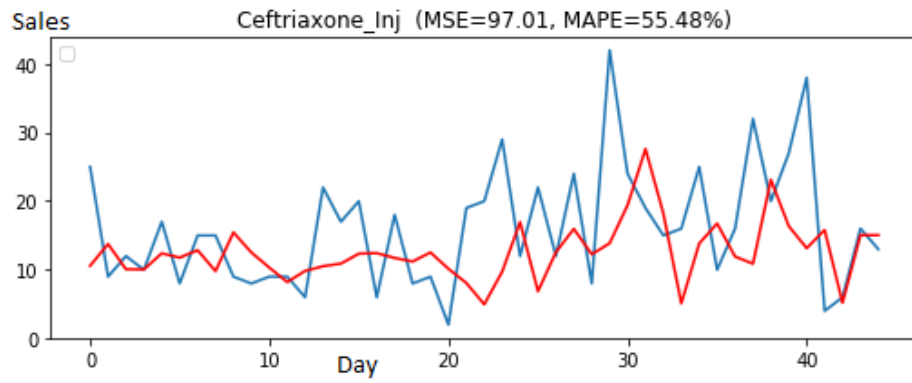


Figure 3.21: Vanilla LSTM model results for Ceftriaxone Injections.

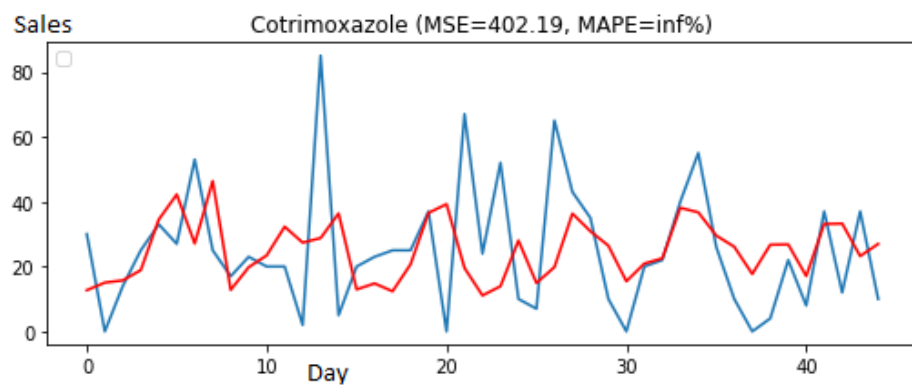


Figure 3.22: Vanilla LSTM model results for Cotrimoxazole Tabs.

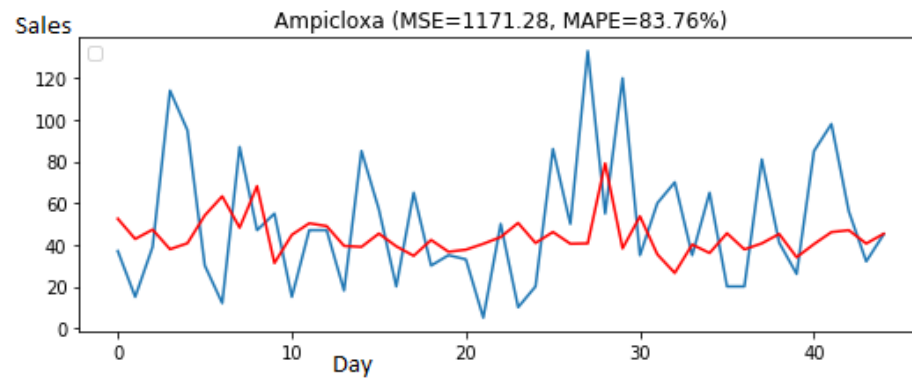


Figure 3.23: Vanilla LSTM model results for Ampicloxa Tabs.

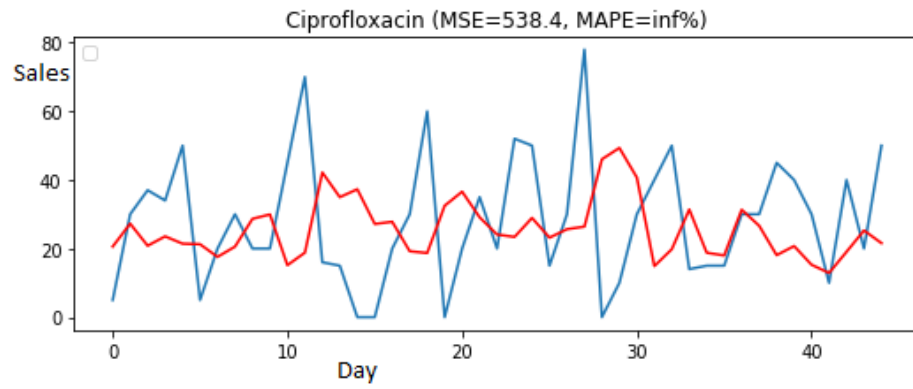


Figure 3.24: Vanilla LSTM model results for Ampicloxa Tabs.

Stacked LSTM method

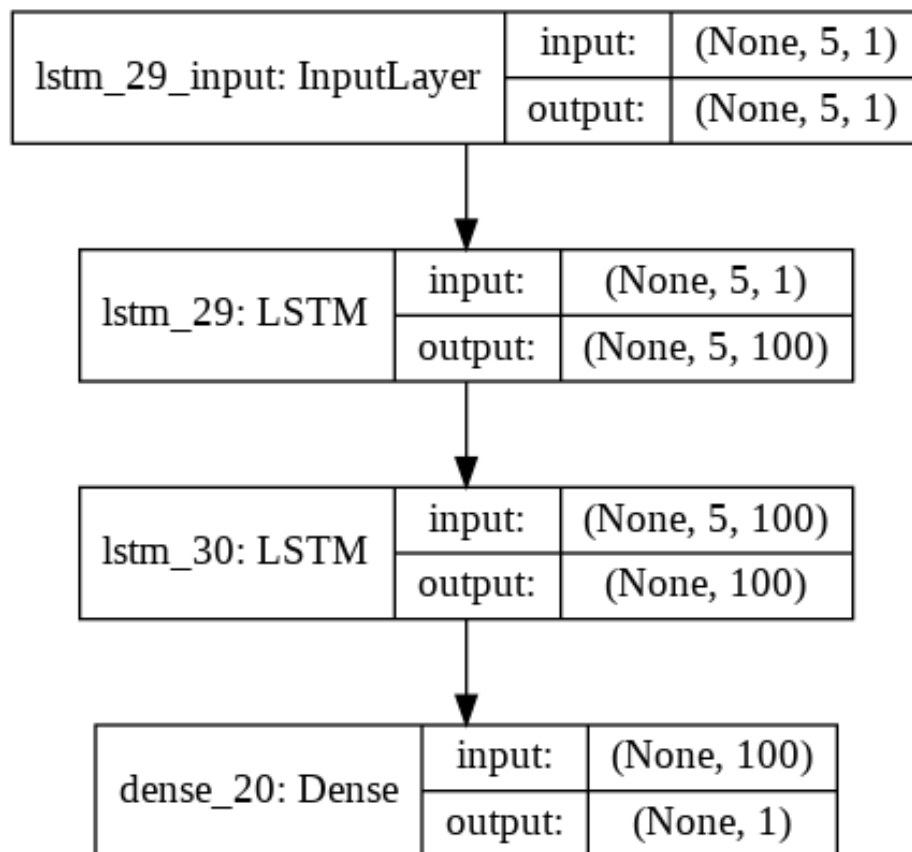


Figure 3.25: Stacked LSTM model Graphical Layout.

Model: "sequential_20"

Layer (type)	Output Shape	Param #
lstm_29 (LSTM)	(None, 5, 100)	40800
lstm_30 (LSTM)	(None, 100)	80400
dense_20 (Dense)	(None, 1)	101
Total params: 121,301		
Trainable params: 121,301		
Non-trainable params: 0		

None

Figure 3.26: Stacked LSTM model Summary.

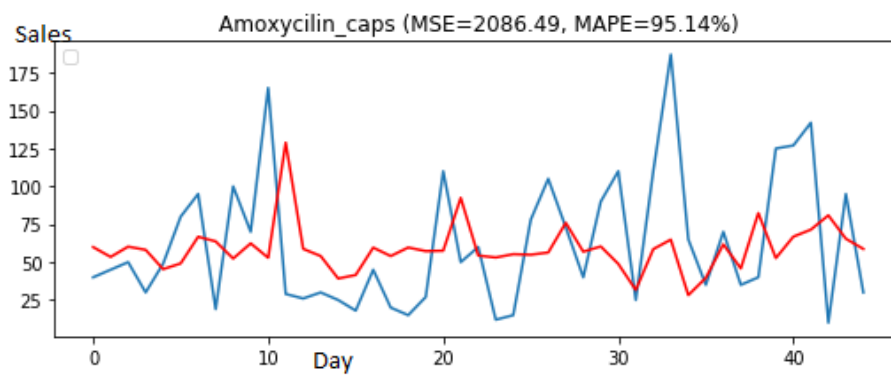


Figure 3.27: Stacked LSTM model results for Amoxycilin Tabs.

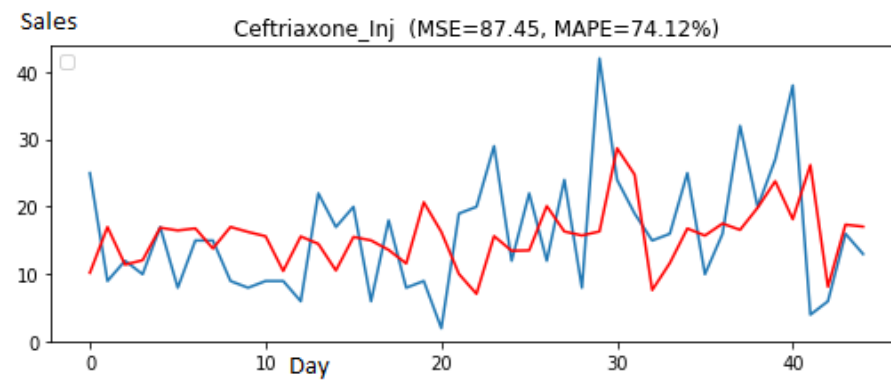


Figure 3.28: Stacked LSTM model results for Ceftriaxone Tabs.

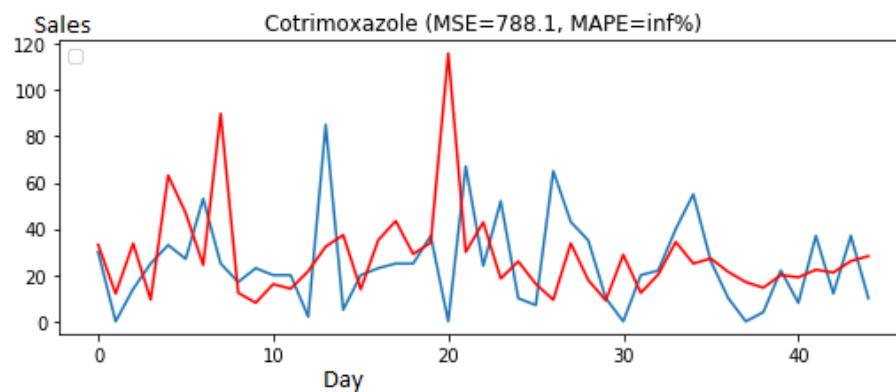


Figure 3.29: Stacked LSTM model results for Cotrimoxazole Tabs.

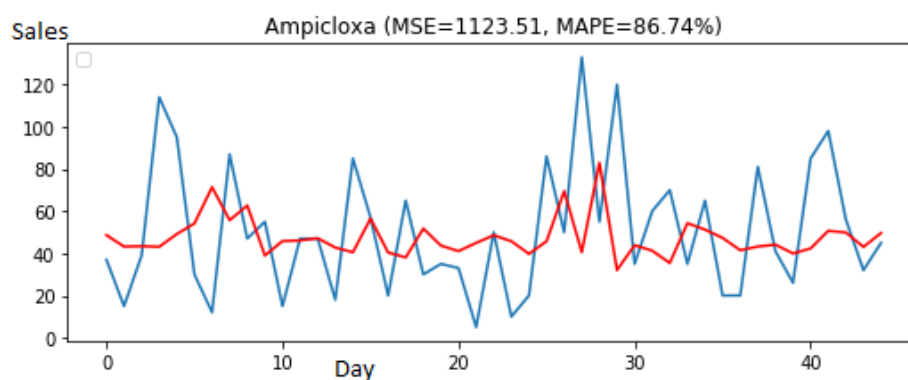


Figure 3.30: Stacked LSTM model results for Ampicloxa Tabs.

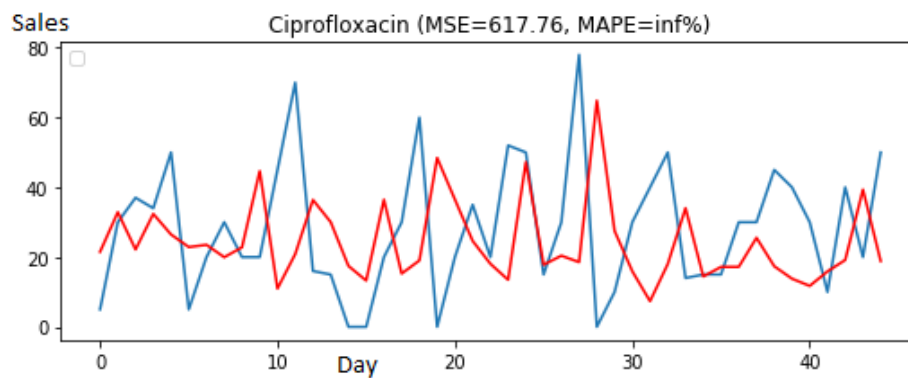


Figure 3.31: Stacked LSTM model results for Ciprofloxacin Tabs.

Bidirectional LSTM

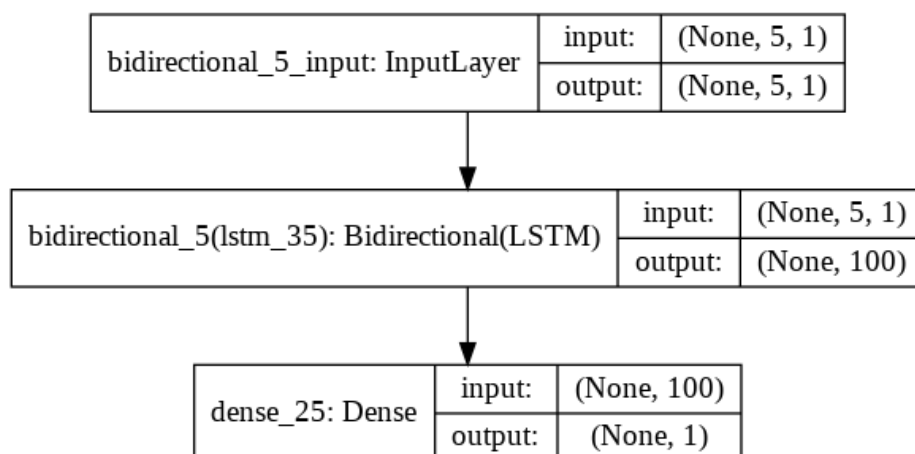


Figure 3.32: Bidirectional LSTM model Graphical Layout.

Model: "sequential_25"

Layer (type)	Output Shape	Param #
bidirectional_5 (Bidirection	(None, 100)	20800
dense_25 (Dense)	(None, 1)	101
Total params: 20,901		
Trainable params: 20,901		
Non-trainable params: 0		
None		

Figure 3.33: Bidirectional LSTM model Summary.

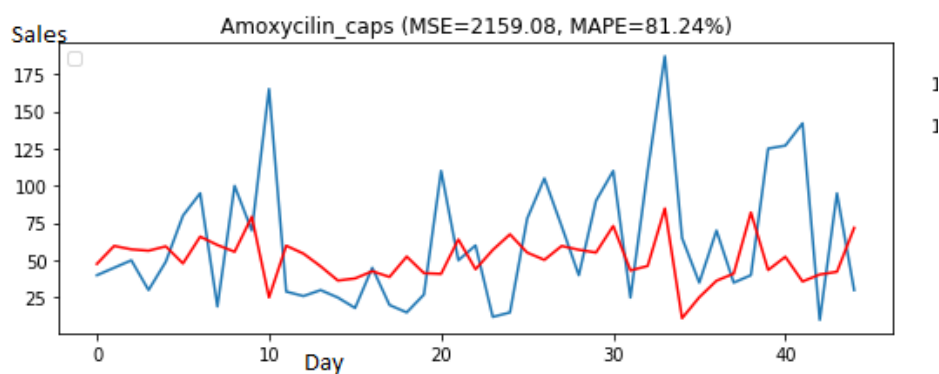


Figure 3.34: Bidirectional LSTM model results for Amoxicilin Tabs.

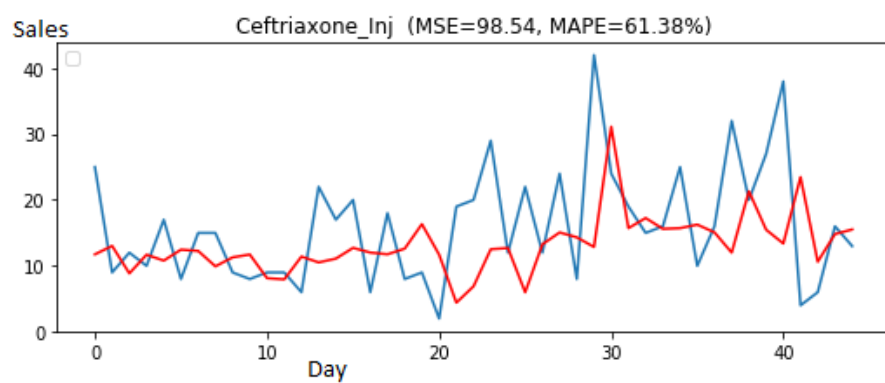


Figure 3.35: Bidirectional LSTM model results for Ceftriaxone Injections.

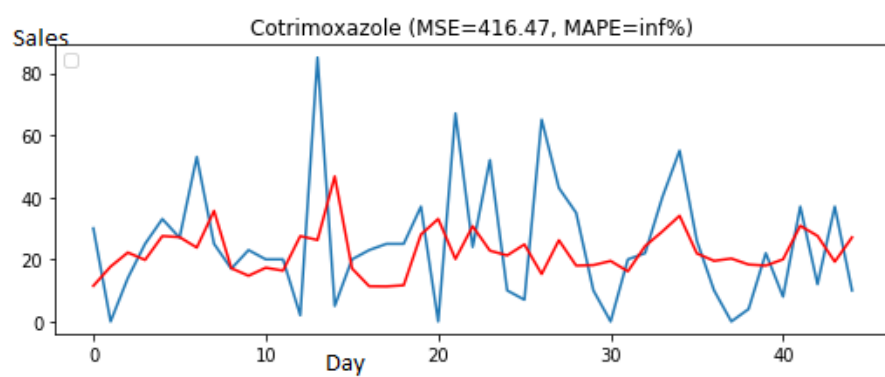


Figure 3.36: Bidirectional LSTM model results for Cotrimoxazole Tabs.

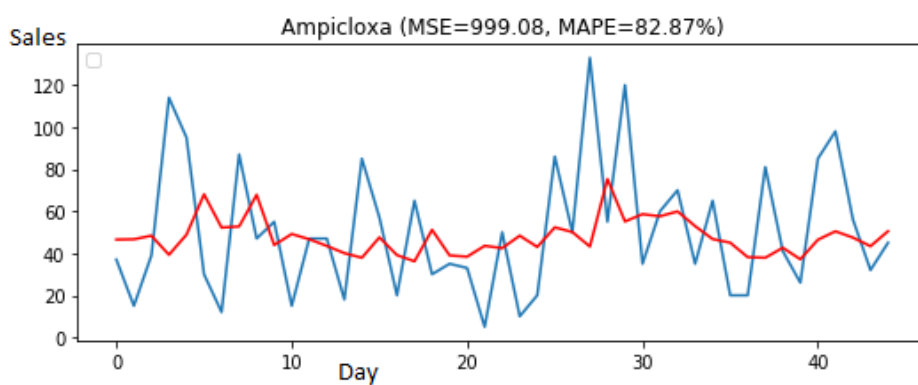


Figure 3.37: Bidirectional LSTM model results for Ampicloxa Tabs.

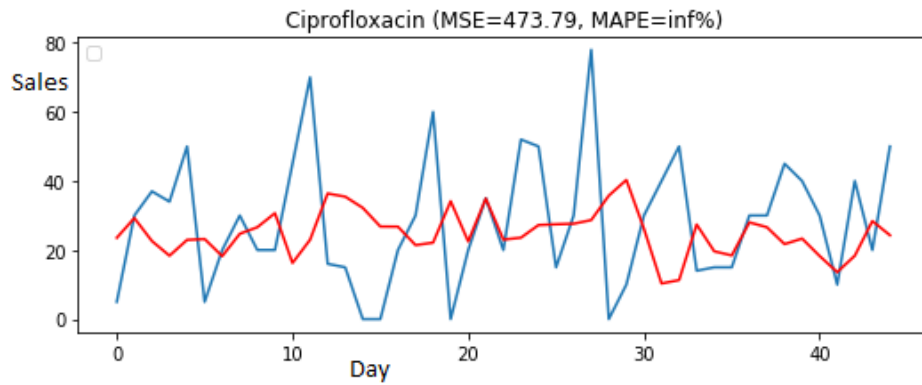


Figure 3.38: Bidirectional LSTM model results for Ciprofloxacin Tabs.

3.3 Web API

I then built a web API that can be used by pharmacy managers using the streamlit python framework. One is able to select one of the various machine learning models in a drop down selection menu as shown below. When you select the model to run , Sales data is loaded from an excel sheet and converted into the pandas data structure .This is then run and trained with the model. Results of the model are saved and used to provide an output which is the actual forecast of the model.

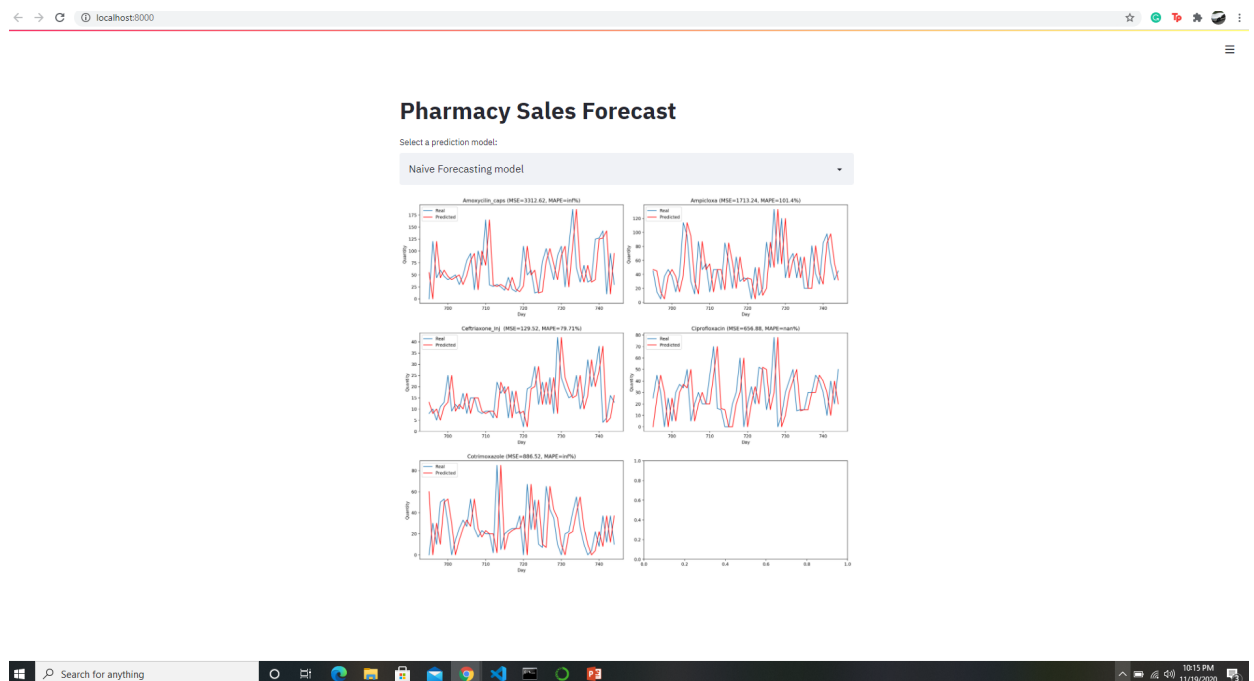


Figure 3.39: Web App API.

Chapter 4

Discussion

Figure 4.1 shows that all the methods used have a less MSE than the currently used Naive forecasting methods.

The results from running my data with all the models are shown below.

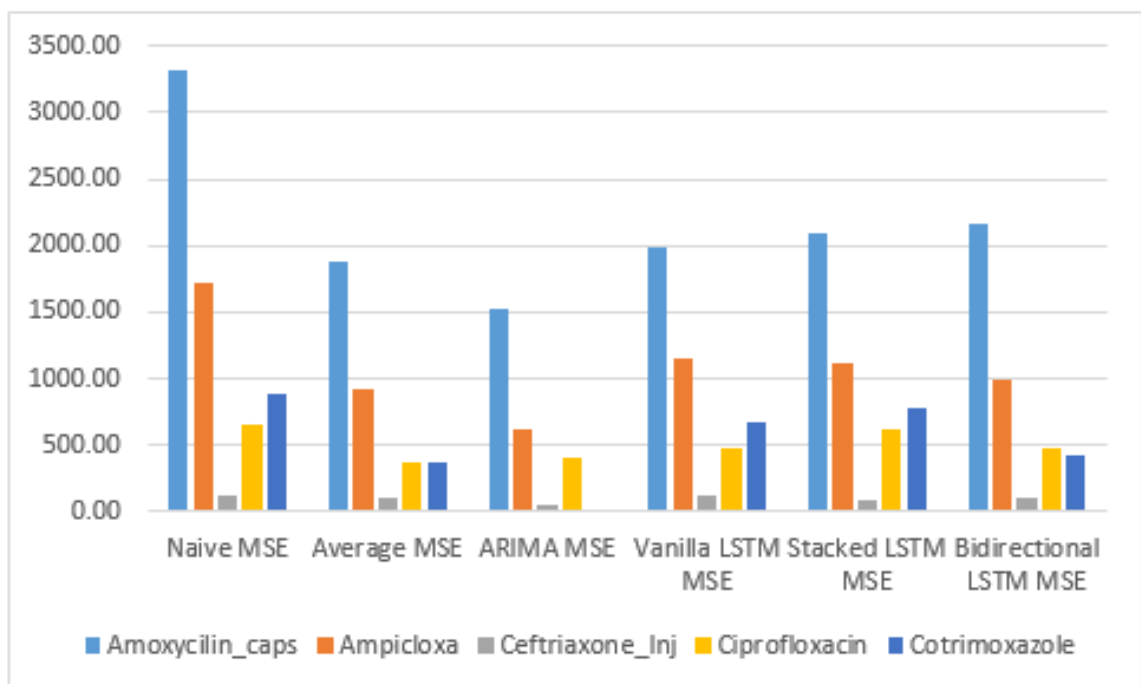


Figure 4.1: MSE overall results.

However with more data of about 5 years or more, better results can be obtained for the LSTM methods .

Chapter 5

Conclusions and Future Works

This chapter contains concluding remarks of the project, that is, the conclusions, challenges met and recommendations.

5.1 Conclusions

5.1.1 Summary of findings

From the work done in this project, it shows that the ARIMA model gives the lowest MSE compared to the rest of the models. Future sales were forecast and viewed within a Website API by the pharmacy owners in order to make informed decisions. Data Collection was done using an inventory management system in place at Soteria Pharmacy. The various machine learning methods applied have been discussed in the report and implemented with the Mean square error as a performance metric to determine how they perform in accordance to the already existing method which is modeled as the Naive forecast in this project.

5.1.2 Limitations of Study

Among the shortcomings of our research are the reluctance of most pharmacies especially those able to provide us with the long time-span data to avail us with this information considering it is company sensitive and only known to a specific set of employees. Also the innavailability of weather data from open platform sources hence requiring us to purchase the precipitation data from the Uganda National Metrological Association (UNMA). It was also hard to quantize the promotional sales and factor them to have multivariate variables contribute to the input data of the Recurrent Neural Network.

5.1.3 Recommendations

We suggest that more data sets are used with a longer lifetime span of about 5 years worth data. We also suggest the incorporation of other secondary dependent features that affect the sales such as weather patterns, promotional sales, marketing elements in order to get more accurate results.

5.1.4 Acknowledgements

I thank Dr. Andrew Katumba for the support offered at every stage of this project from day 1, he is a key component to enabling us see this idea to fruition. We also acknowledge the management of Soteria pharmacy in Luweero district. Specifically Ms. Brenda that availed us with the required information and data plus explaining the entire workflow of the pharmacy purchasing process to us.

5.2 Ideas for Future Work

Incorporating a multivariate model that makes use of various input data to forecast the results such as weather data ,promotion sales data e.t.c

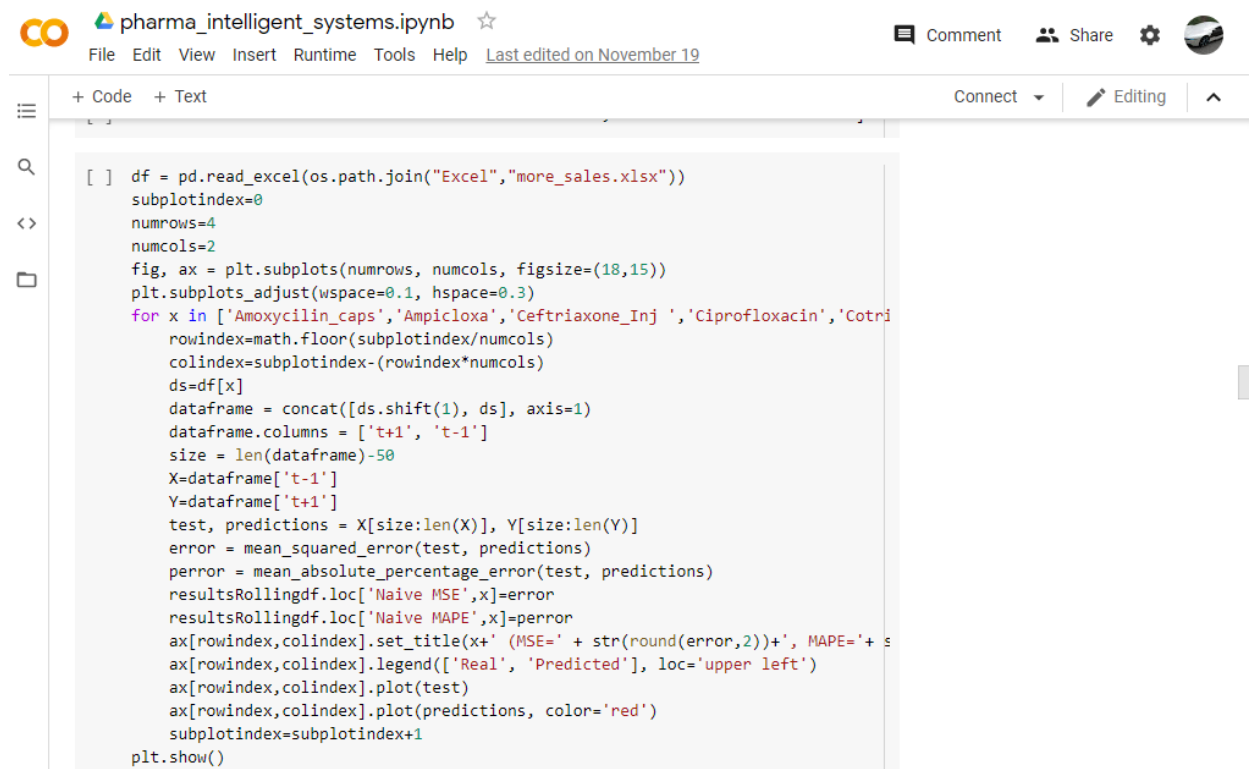
Bibliography

- [1] S. H. Perera, N. Jihan, M. Jayasinghe and Srinath "Sales forecasting using multivariate long short term memory network models," *PeerJ PrePrints*, vol. 7, pp. e27712, 2019.
- [2] Zadeh Khalil, N. Sepehri, M. M., Farvaresh "Intelligent sales prediction for pharmaceutical distribution companies, A data mining based approach," *Mathematical Problems in Engineering*, 2014.
- [3] Lian-hong Cai, "Rough Set in Neural Network," *Computer Engineering*, Vol 27, No.5 pp65-69, 2001.
- [4] Alex Krizhevsky, I Sutskever, and GE Hinton, "Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems," *object recognition*, pages 1–9, 2012
- [5] O Abdel- Hamid and A Mohamed. "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," *Acoustics, Speech, and Signal Processing*, 2012
- [6] R. Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning," *Proceedings of the 25th international conference on Machine learning*, 2008
- [7] HP Martinez. "Learning deep physiological models of affect." *Computational Intelligence Magazine*, (April):20–33, 2013. (physiological affect modelling)
- [8] Atiya, Amir F. , Gayar, Neamat El and El- Shishiny, Hisham (2010) "An Empirical Comparison of Machine Learning Models for Time Series Forecasting." *Econometric Reviews*, 29: 5, 594-621
- [9] S. Prasad, P. Prasad. "Deep Recurrent Neural Networks for Time Series Prediction"
- [10] "DeepLearningforEvent-DrivenStockPrediction Mladen Dalto 'Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting" *2015 IEEE International Conference on Industrial Technology (ICIT)*

- [11] XiaoDing, YueZhang, TingLiu, JunwenDuan, Mladen Dalto, "Deep neural networks for time series prediction with applications in ultrashort-term wind forecasting time series I" *University of Zagreb, Faculty of Electrical Engineering and Computing*
- [12] Lisa Tagliaferri" An Introduction to Machine Learning"
<https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>
- [13] Davide Burba" An overview of time series forecasting models"
<https://towardsdatascience.com/an-overview-of-time-series-forecasting-models-a2fa7a358fcb>
- [14] "National Drug Authority. NDA licensed outlets as at 12th March, 2018. Kampala: National Drug Authority; 2018."
- [15] Accessed on December 12th 2020 "https://towardsdatascience.com/arima-simplified-b63315f27cbc"
- [16] Accessed on December 12th 2020 "https://colah.github.io/posts/2015-08-Understanding-LSTMs/"
- [17] Accessed on December 18th 2020, *analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/*
- [18] Accessed on December 18th 2020, *<https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>*
- [19] Accessed on December 18th 2020, *<https://docs.streamlit.io/en/stable/>*

Appendix A

Appendix



```
[ ] df = pd.read_excel(os.path.join("Excel", "more_sales.xlsx"))
subplotindex=0
numrows=4
numcols=2
fig, ax = plt.subplots(numrows, numcols, figsize=(18,15))
plt.subplots_adjust(wspace=0.1, hspace=0.3)
for x in ['Amoxycilin_caps', 'Ampicloxa', 'Ceftriaxone_Inj ', 'Ciprofloxacin', 'Cotri
rowindex=math.floor(subplotindex/numcols)
colindex=subplotindex-(rowindex*numcols)
ds=df[x]
dataframe = concat([ds.shift(1), ds], axis=1)
dataframe.columns = ['t+1', 't-1']
size = len(dataframe)-50
X=dataframe['t-1']
Y=dataframe['t+1']
test, predictions = X[size:len(X)], Y[size:len(Y)]
error = mean_squared_error(test, predictions)
perror = mean_absolute_percentage_error(test, predictions)
resultsRollingdf.loc['Naive MSE',x]=error
resultsRollingdf.loc['Naive MAPE',x]=perror
ax[rowindex,colindex].set_title(x+' (MSE= ' + str(round(error,2))+', MAPE='+ s
ax[rowindex,colindex].legend(['Real', 'Predicted'], loc='upper left')
ax[rowindex,colindex].plot(test)
ax[rowindex,colindex].plot(predictions, color='red')
subplotindex=subplotindex+1
plt.show()
```

Figure A.1: My Naive forecast code Base on Google Colab

```
[ ] r=['Amoxicillin_caps']
for x in r:
    rowindex=math.floor(subplotindex/numcols)
    colindex=subplotindex-(rowindex*numcols)
    X=df[x].values
    scaler = MinMaxScaler(feature_range = (0, 1))
    X=scaler.fit_transform(X.reshape(-1, 1))
    X_train,y_train=split_sequence(X[0:size], n_steps)
    X_test,y_test=split_sequence(X[size:len(df)], n_steps)
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], n_features))

    model = Sequential()
    model.add(LSTM(1000, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
    model.add(LSTM(1000, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    history = model.fit(X_train, y_train, epochs=100, verbose=1, validation_data=(X_test, y_test))
    train[str(sd)] = history.history['loss']
    val[str(sd)] = history.history['val_loss']
    az[rowindex,colindex].set_title(x)
    az[rowindex,colindex].plot(train[str(sd)])
    az[rowindex,colindex].plot(val[str(sd)], color='red')
    sd=sd+1

X_test = X_test.reshape((len(X_test), n_steps, n_features))
predictions = model.predict(X_test, verbose=0)
y_test=scaler.inverse_transform(y_test)
predictions = scaler.inverse_transform(predictions)
error = mean_squared_error(y_test, predictions)
```

Figure A.2: My Vanilla LSTM forecast code Base on Google Colab

```
warnings.filterwarnings('ignore')
[ ] #r=['Amoxicillin_caps', 'Ampicloxa', 'Ceftriaxone_Inj ', 'Ciprofloxacin', 'cotrimoxazole']
r=['Amoxicillin_caps']
for x in r:
    rowindex=math.floor(subplotindex/numcols)
    colindex=subplotindex-(rowindex*numcols)
    X=df[x].values
    scaler = MinMaxScaler(feature_range = (0, 1))
    X=scaler.fit_transform(X.reshape(-1, 1))
    X_train,y_train=split_sequence(X[0:size], n_steps)
    X_test,y_test=split_sequence(X[size:len(df)], n_steps)
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], n_features))

    model = Sequential()
    model.add(Bidirectional(LSTM(1000, activation='relu', input_shape=(n_steps, n_features))))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
    cb.logs=[]
    history = model.fit(X_train, y_train, epochs=100, verbose=1, callbacks=[cb], validation_data=(X_test, y_test))
    train[str(sd)] = history.history['loss']
    val[str(sd)] = history.history['val_loss']
    az[rowindex,colindex].set_title(x)
    az[rowindex,colindex].plot(train[str(sd)])
    az[rowindex,colindex].plot(val[str(sd)], color='red')
    sd=sd+1
    #loss, acc = model.evaluate(X_test, y_test)
    #print(sum(cb.logs))
    #print(acc*100.0)
    X_test = X_test.reshape((len(X_test), n_steps, n_features))
    predictions = model.predict(X_test, verbose=0)
    y_test=scaler.inverse_transform(y_test)
    predictions = scaler.inverse_transform(predictions)
    error = mean_squared_error(y_test, predictions)
    perror = mean_absolute_percentage_error(y_test, predictions)
    resultsLongtermdf.loc['Bidirectional LSTM MSE',x]=error
    resultsLongtermdf.loc['Bidirectional LSTM MAPE',x]=perror
    ax[rowindex,colindex].set_title(x+' (MSE=' + str(round(error,2))+', MAPE='+ str(round(perror,2)) + '%)')
```

Figure A.3: My Stacked LSTM forecast code Base on Google Colab

	Amoxycilin_caps	Ampicloxa	Ceftriaxone_Inj	Ciprofloxacin	Cotrimoxazole
Average MSE	1882.194144	927.985467	102.020312	363.422187	373.897875
Average MAPE	inf	94.288002	48.285348	inf	inf
ARIMA MSE	0.000000	0.000000	0.000000	0.000000	0.000000
ARIMA MAPE	0.000000	0.000000	0.000000	0.000000	0.000000
AutoARIMA MSE	0.000000	0.000000	0.000000	0.000000	0.000000
AutoARIMA MAPE	0.000000	0.000000	0.000000	0.000000	0.000000
Prophet MSE	0.000000	0.000000	0.000000	0.000000	0.000000
Prophet MAPE	0.000000	0.000000	0.000000	0.000000	0.000000
Vanilla LSTM MSE	1992.988497	1152.440552	114.375701	477.080848	678.311004
Vanilla LSTM MAPE	89.127919	87.236741	69.777434	inf	inf
Stacked LSTM MSE	2086.485384	1123.511486	87.453323	617.756463	788.095327
Stacked LSTM MAPE	95.139783	86.743554	74.122829	inf	inf
Bidirectional LSTM MSE	2159.075832	999.079763	98.544152	473.785420	416.467764
Bidirectional LSTM MAPE	81.239085	82.872384	61.377742	inf	inf

Figure A.4: Results of the machine learning forecast