5/11/2016

Demand Forecasting-Deep Learning

Sales Demand Forecasting using Deep Learning – LSTM Neural Network

Bikal Basnet
https://www.linkedin.com/in/bikal-basnet-63ab635b

CONTENTS

FIGURES

TABLES

1 DEFINITION

1.1 PROJECT OVERVIEW

Forecasting is the process of making prediction of the future values. Our major area of concern, in this project is this issue of Forecasting i.e. making accurate future predictions.

Forecasting can be done in various different ways / methods. We will briefly discuss three major forecasting techniques and focus our attention on Non-Parametric Statistical Forecasting method, using the state of the art Long Short Term Memory (LSTM) Deep Neural Network.

1.1.1 Forecasting Techniques

There are three most-commonly used Forecasting techniques. They are [1]

a. Qualitative technique: This forecasting process uses the qualitative data i.e. expert opinion, information about special event and may or may not take the past sales data into consideration [1]. For example, an expert in anticipation of an Apple's 25th anniversary can predict the increase in sales from loyal apple fan base and make such prediction.

This technique is often used when the data is scarce and most commonly, during the first product introduction to market. This process often relies on human judgement and rating schemes. Some of the most popular qualitative methods commonly used today are Delphi, Market Research, Panel Consensus, Visionary Forecast and Historical Analogy. [1]

b. Causal models: This forecasting process uses highly refined model and specific information about the relationship between system elements, to make forecasts. For example, if temperature drops below 15 degree then increase in the sales of the heater can be forecasted. The causal model aims to extract such cause and effect patterns. Some of the most commonly used causal models are Regression models, Econometric Models, Intention-To Buy & Anticipation Surveys, Input- Output Model, Economic Input-Output Model, Diffusion Index, Leading Indicator and Life-Cycle Analysis. [1]

c. Statistical forecasting: This forecasting process is based on time series analysis and projections. It focusses entirely on finding statistical patterns and change in patterns, as observed from the historical data and use it to make the future prediction. This process works with the assumption that the existing patterns in the data will continue and it works towards finding past statistical pattern, to be used later to make forecasts. [1] The process works by developing the mathematical model (formula) out of the past patterns and trends and tests the mathematical model against the held-out test data for reasonableness and confidence. Since the mathematical model is used, it is exactly why this process is known as statistical forecasting.

In this mathematical model formulation process, when the errors are found against the test data, then often such errors are used as the basis for refining the model to yields less error / more confidence. This process is often repeated, until the final forecasting model that generates the least error is formulated. [2]

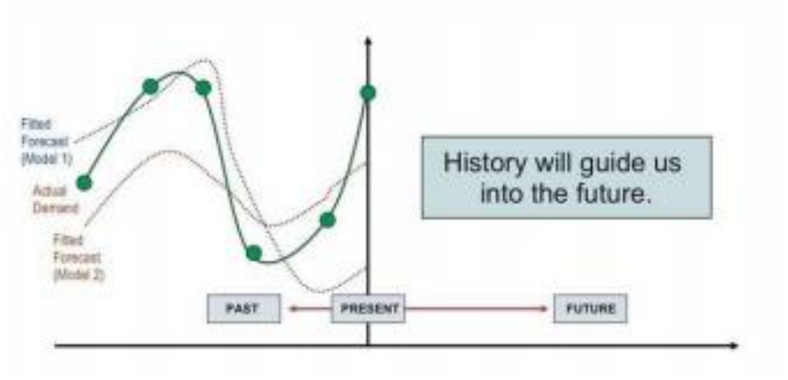
Fitted Forecast (Model 1)
Actual Demand
Fitted Forecast (Model 2)
History will guide us into the future.
PAST
PRESENT
FUTURE

Figure 1 What is Statistical Forecasting Explained [3]

Some of the common Statistical forecasting methods are Moving Average (MA), Exponential Smoothing (ETS), Box- Jenkins (BJ), X-11 and Trend Projections (TP). These techniques in a way or another model the historical data into statistics, either by taking the average of several consecutive data points to find the larger trend (MA), or by averaging with more weight given to new data to older data(ETS) or finding out the mathematical model for the data (BJ) or decomposing the trend into seasonal, trend cycle and irregular elements (X-11) or fitting a trend line to a mathematical equation and projecting(TP).

One of the major challenge often faced by these statistical model is that, the prediction based on the raw data is extremely difficult due to the presence of the all trends (seasonal, longer trends and irregular trends) combined into a single trend. And unfortunately, these existing methods have difficulty dealing with them. Most of them either identify only the seasonal or the trend or the irregularity component and fail to encounter the combined effect of trend and cycles and irregular components effectively, which leads to poor forecast.

In this project, we aim to use the state-of-art deep learning Statistical Forecasting method, which will be able to auto-optimize its forecasting model through iterations, thus ultimately yielding the best forecasting model. Furthermore, we will be more focused on the Retail Domain and will concentrate on Product demand forecasting on the retail domain.

1.1.2 Retail Demand Forecast

Retail demand forecasting is the process of anticipating the future purchasing action by evaluating the past revenue and consumer behavior over the previous months or years to discern patterns [4]. In retail demand forecasting, forecasts are made to predict the demands of products in retail establishments and control the pricing and the inventory. Retail Demand forecasting often faces different challenges, most pressing of them being [5]

- Scale of Problems I.e. Large number of items and stores to forecast
- Intermittent demand (slow and erratic sales of many products)
- Assortment instability (frequent new-item and seasonal assortment changes)

In retail business, often the qualitative forecasting method is used for small scale and mostly for new products. For many items, adopting the qualitative approaches is too labor intensive and infeasible due to the scale problem. Similarly, the causal forecasting approach in the retail also suffer from the problem of scale and is more prominently and often, made infeasible by the lack of features required to develop such models.

On the contrary, one of the most readily and easily available data in the retail domain, is the historic sales data, since every company on the least side store their sales data, if not any other data. With commonly available sales data, the best and most feasible way forward for forecasting is often seen as the statistical forecasting approaches.

The statistical model in the retail industry, not only finds the patterns but also are often implicitly able to encompass the causal models in them, which makes them even more powerful. A simple example being, if let's say decrease in temperature causes sales of some winter products to go up, then such causes and effect will also be implicitly encompassed by the statistical model, in terms of the seasonal trends. And similar will be the case for other special periodic events i.e. New year, Christmas etc.

Furthermore, statistical models are also able to implicitly encompass some of the qualitative aspects of the data i.e. expert opinion into them, which makes them even more powerful. For example, external factors such as local taste, which affect the product sales, often discerned by the local experts, can be learnt by the statistical models, often when the forecast on the regional or store level are made.

1.2 PROJECT ORIGIN:

While historically, the retail demand forecasting has been the subject of manual statistical discovery of the trends and patterns, which not only limited its accuracy, it also made it un-scalable and un-adaptive i.e. unable to respond with the most recent updates. In this version of our Retail demand forecasting, we aim to develop a single scalable, adaptive retail demand forecasting model, that will be applicable across the large number of products and will be able to respond to the current dynamic changes in the trend.

Often the problem faced by any Store manager is to gain optimum usability of their warehouse space and the product availability. For example, if a store manager wrongly anticipates a demand for a product and order it in a large quantity but is not able to sell it, then the product once not sold, will occupy his precious warehouse space. Also, such overstocked products will be subjected to wear and tear and his cashflow will be ultimately wasted since he cannot make the turnover and use it in other more demanding, more profit-making products. Similarly, on the contrary, if a store manager wrongly anticipates the product to be in less demand and orders it in smaller quantity, but that the demand goes high, then the he will not be able to sell it, since he has no stock and will lose profit and more importantly drive his customers to his competition. This makes it clear, the importance of the right/ accurate demand forecasting need, so that a store manager can turn his profitability up with higher customer satisfaction at minimal inventory space and management cost wastage.

And this problem often is complicated by the large number of products, different variety of products with different demand trends, different seasonal trends, combined with numerous factors such as, holiday season, type of holiday season, weather temperature, Economic cycles and many other reasons.

The complication and the complexities severally affect the ability to correctly predict the demand of the products which have grave consequences to an organization.

Some of these adverse consequences are [6]

1. Under forecasting causes the organization to lose sales.
2. Under forecasting creates unnecessary cost often in terms of unnecessary expenses such as air freight, rush delivery services, frantic email and phone calls, overtime and ultimately unhappy customers and fines.
3. In case of products with strictly enforced Minimum advertised price, simply unloading the products with lower price becomes completely, infeasible and with date expiry risk, the product may altogether be a total loss to the organizations.
4. Overstocking costs more in terms of holding working capital, storage cost, wear and tear risk, product expiry date risks, inventory management overhead, extra warehouse space and rent, and the need to borrow additional working capital to compensate for the wasted capital held up with overstocked reserve products.

1.3 Related Dataset:

For our project, we consider the Product sales data of product A for a plastic Manufacturer as provided by the Data Market. [7].

The Input Data was taken from the "Data Market" website, at the following link on September 25,2016.

https://datamarket.com/data/set/22o0/monthly-sales-of-product-a-for-a-plastics-manufacturer#!ds=22o0&display=line&e=-fdjq

The input data "monthly-sales-of-product-A-for-a-plastic-manufacturer.csv" consists of historical sales data for first twelve days of each month, starting from the Jan 1st 2016 to May 10th 2016. The data consists of

- Month-Day: Indicating the date on which the sales of the product was made.
- Monthly sales of product A for a plastic manufacturer: Sales of product A for the given date

1.4 Problem Statement

In our project the main problem we aim to solve is,

- Accurately forecast the demand for the products, to minimize overstocking or understocking

To solve the problem, we will take the Product A's historical sales data, as provided at the Data Market website, information into account. The historical sales data of the product will be then separated into the training data and the test data, with 2/3 allocated as training dataset and the 1/3th allocated as the final held out test set. The Prediction modeler will take the 2/3rd training data and generate prediction modeler which will then finally be tested across the rest of the 1/3rd final test set. The model will can automatically identify any patterns in the data, if exists and will be able to predict the forecast. The forecast will be based on the current product A's trend and if a new product B's demand forecast has to be done, then the product B’s historical information will have to be feed in replacement to the product A’s sales data fed earlier.

Ultimately the demand prediction model, should be able to take historical sales information of a product and generate the prediction for the next any number of days, as required.

1.5 Metrics

To measure the performance of our model we will be using the “Root Mean Squared Error” (RMSE). Technically RMSE of a predictor measures the square root of the average of the squares of the errors or the deviations i.e. the difference between the estimator and the actual estimated value.

Root Mean squared error heavily emphasizes on the outliers and emphasis the error more as the predicted and the actual values gets apart, which in our case of demand forecasting is highly relevant and favored over the Mean absolute error. Mean absolute error is an estimator used to estimate how close the forecast or prediction is to the eventual outcome. It however puts less emphasis on the outlier or the extreme fluctuation in the actual and the predicted demand for those products and hence is not favored over RMSE

2 ANALYSIS

(approximately 2 - 4 pages)

2.1 DATA EXPLORATION

For our project, we explored and found the Product sales data of product A for a plastic Manufacturer at the Data Market. [7] to most closely resemble our retail demand forecasting problem. Hence we have used the data from Data Market.

The data is a time series data consisting of historical sales data for the first twelve days of each month, starting Jan 1st 2016 to May 10th 2016. The data consists of two feature – "time" feature and the "sales amount" feature.

- Month-Day: Indicating the date on which the sales of the product was made.
- Monthly sales of product A for a plastic manufacturer: Sales of product A for the given date

2.1.1 Data Features

In the Product A's Sales time series data, the following primary data features were observed.

Trend: Trend is often described as varying mean over time and is often present in the time series data [8]. In our product sales data, we observed the positive trend up until March 12th and then after March 12th, we observed the negative trend in the sales i.e. we observed that the sales of the product A gradually increased from Jan 1st till Mach 12th and then after 12th March the product A's sales started gradually decreasing. This can also be observed in the Figure 2 below

Seasonality: Seasonality is often described as the variations at specific time frames. For example, people might have tendency to buy more cars month because of pay increment or festivals. [8]

In our particular product A's sales, we observed that the sales of product A appear to follow certain seasonal patterns as shown in the Figure 2 below. The seasonality has been highlighted in 4 red circles, with each circle representing a pattern, in the figure.

Irregularity: In our data, we observed the residual trend i.e. irregularity trend. The irregularity or residual trend in our data does not follow a particular pattern. The irregularity pattern is shown in the yellow circles in the Figure 2 below.

We observed that the residual pattern follows less fluctuations in the early time series period, as depicted in the first yellow circle in the Figure 2. In the mid-time series, as depicted in the second yellow circle the irregular trends are larger in magnitude / variations but for shorter period. At the end time of the time series as represented by the third circle we observed that the residual errors pattern has changed again, with more significant increase in both the intensity and the period length. This irregular trend increase the complexity for making prediction for our data more difficult.

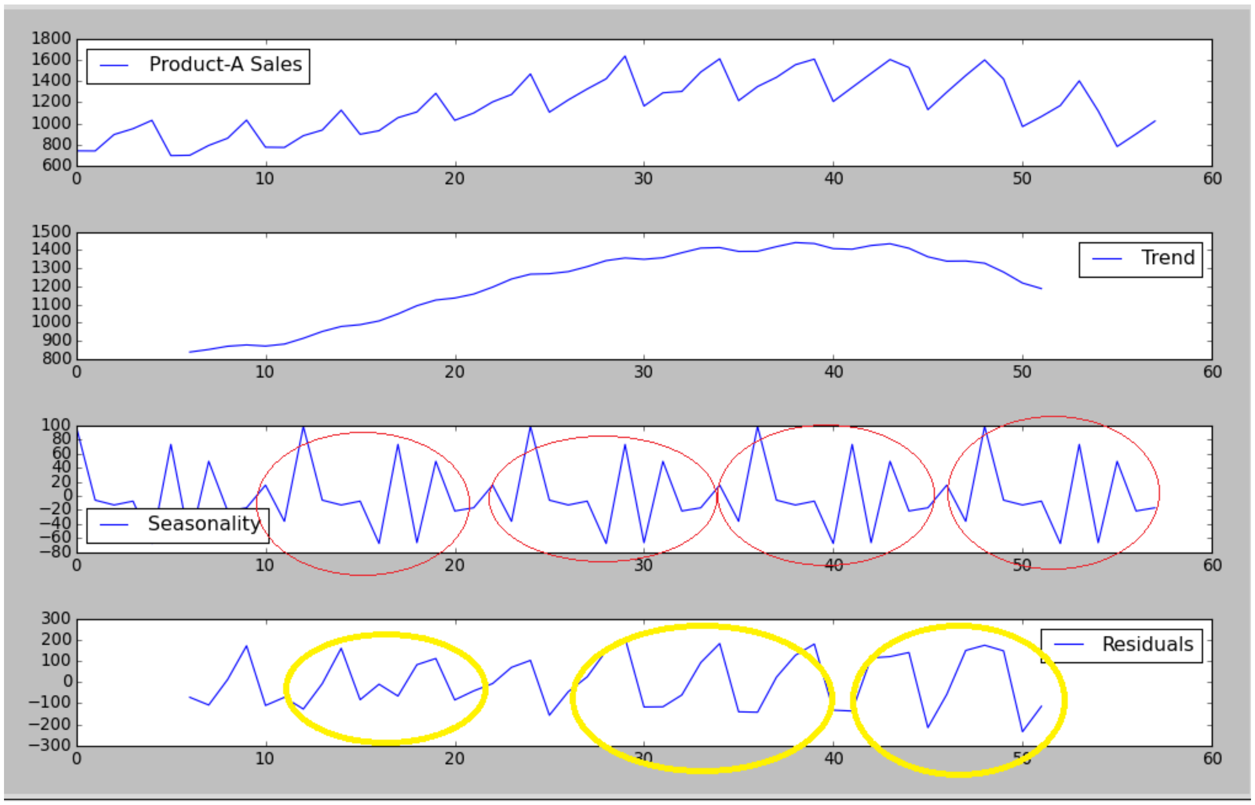

Figure 2 Showing Trend, Seasonality and Residual as observed in the Product A's Sales Data

2.1.2 Statistics about the data:

The following is the Statistics obtained for the data

Start time: Jan 1st 2016

End time: May 10th 2016

Minimum Product Quantity Sales: 697

Maximum Product Sales Quantity: 1637

Average Product Sales Quantity: 1164.1

Standard Deviation of Product Sales Quantity: 267.9

Trend as observed:

nan	nan	nan	Nan
nan	nan	838.70833333	852.875
870.6667	878.0417	871.75	882.5833
914.5417	952.0833	979.625	989.4583
1010.083	1048.75	1093.875	1125.208
1136.292	1158.167	1196.417	1241.083

1267.667	1270.208	1281.792	1309.292
1342.375	1356.917	1349.542	1358.292
1385.833	1411.417	1414.25	1393
1393.5	1419.417	1441.875	1436.458
1408.542	1405.292	1425.625	1435.417
1410.417	1363.5	1339.333	1340.333
1328.167	1279.208	1218.875	1188.25
nan	nan	nan	Nan
nan	nan		

Seasonality:

1.077445	0.997234	0.99728	0.986807	0.949037
1.058319	0.938697	1.037468	0.982317	1.004282
1.003451	0.967663	1.077445	0.997234	0.99728
0.986807	0.949037	1.058319	0.938697	1.037468
0.982317	1.004282	1.003451	0.967663	1.077445
0.997234	0.99728	0.986807	0.949037	1.058319
0.938697	1.037468	0.982317	1.004282	1.003451
0.967663	1.077445	0.997234	0.99728	0.986807
0.949037	1.058319	0.938697	1.037468	0.982317
1.004282	1.003451	0.967663	1.077445	0.997234
0.99728	0.986807	0.949037	1.058319	0.938697
1.037468	0.982317	1.004282		

Irregularity Component:

nan	nan	nan	nan	Nan
nan	0.889122	0.896217	1.006699	1.170331
0.887102	0.906278	0.898141	0.987941	1.152555
0.919701	0.972244	0.950526	1.080036	1.100768
0.922775	0.944867	1.002877	1.060827	1.074796
0.873928	0.956736	1.026301	1.116201	1.139932
0.919632	0.915424	0.957154	1.048354	1.135202
0.902108	0.898483	1.014489	1.081399	1.134385
0.903678	0.901666	1.100708	1.07709	1.10287
0.825946	0.964317	1.120286	1.118078	1.113141
0.798809	0.909111	nan	nan	Nan
nan	nan	nan		

The “nan” values in the trend and the irregularity trend is because we are taking the average of the last 6 value, hence the rolling mean is not defined for the first 6 and the last six values, which are then represented as nan, indicating its unavailability.

2.1.3 Data Quality Issues

In our data, we observed three major issues with the data.

1. Unnormalized scale, between 697 and 1637
2. Missing values of sales data between 13th of each month till the 1st of the next month
3. Outliers.

We will discuss these data abnormalities and how we handle it in the later "Data Preprocessing" section of the report.

2.2 Exploratory Visualization

For our case, we performed exploratory visualization of the product sales amount over time, the only feature we are dealing with.

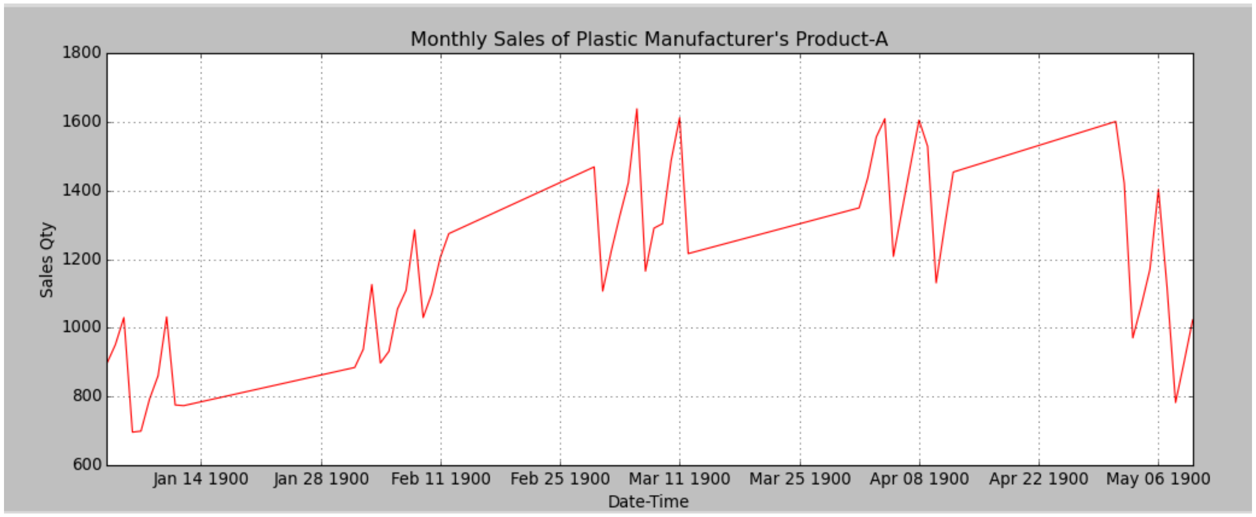

Figure 3Showing Sales of Plastic Manufacturer's Product-A plot

The plot above shows the sales of the product. In the plot the straight lines i.e. between Jan 14 to Jan 30, represent the missing value.

2.3 Algorithms and Techniques

Time Series analysis technique is typically divided into the two approaches. [9]

- Parametric Approach
- Non-Parametric Approach

The parametric approach assumes stationary Stochastic Process i.e. the data has a certain structure which can be described using some parameters i.e. autoregressive or moving averages. The Stationary stochastic process assumes that the mean, variance do not change over time. i.e. a time series data follows a certain trend either upward or downward with a fluctuation occurring at a regular interval e.g. a perfect example of such stochastic process being the gradually increasing orange price with upward fluctuation in the price during the offseason.

On the other hand, in case of the non-parametric approach, no such assumptions regarding the stationarity are made and hence the time series parameters are determined by the training data and not the model. Hence we will have different model for each different training data and the model will be more responsive to the subtle changes in the trend of the data as more data is introduced to the model.

In our case, we will be using the LSTM Long Short Term memory neural network a variant of the Recurrent neural network, originating from the traditional neural network. A traditional neural network takes a range of inputs and connects the input features to some hidden layer, with each separate weight assigned to each input feature and the hidden layer node connection, as visualized in the diagram below. The neural network then ultimately generates the output from the hidden layers mapped to the output features.

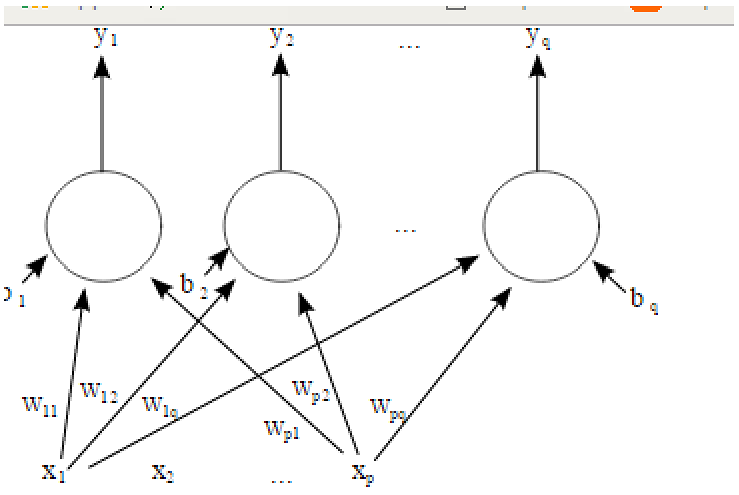

Figure 4 Showing a 1 layer neural network [10]

Explained more elaborately, neural network can be thought of as a complex mathematical function, which takes a vector of values as input and generates a vector of values as output. [11] The neural network is created in such a way that it is easy to alter its parameters. The network works by initially taking the past data and running it through the model where the initial weight vectors are generated randomly. For each input, neural network generates output which may / may not be correct. The algorithm then with the aim of minimizing the error, updates the initial random weights such that the output error is minimized for the next run. This step is then repeated for each iteration, until the error reaches the minimal point. This is how a neural network works. [11]

For example, let's assume two feature data, with x1 = average annual income price of the individual, and x2 = the person's id and let's assume that we were given the responsibility of the predicting the city based upon people's data features. Then with few iteration, the neural network model will learn that the feature x1 i.e. annual income has more prominence in predicting the city to the feature x2 and assign more weight to the feature x1 over feature x2. This is typically how the neural networks train itself to find out the prominence of the features to extract the required output.

In contrast, however, unlike the traditional neural networks in which the data are treated as individual instance with no interrelation between them, a recurrent neural network, treats the data as being interlinked and treats accordingly. For example, while predicting upcoming words in a sentence, a recurrent neural network will take the earlier data points into account and makes the decision e.g. subject + verb+ object. For example, if the earlier two words were consecutively subject and a verb then there is a high chance that the next word will be an object. Recurrent neural network aims to extrapolate such type of cases, which was not handled by general neural networks.

To elaborate more, the central idea behind LSTM architecture is a memory cell that can maintain its state over time, and non- linear gating units which regulate the flow of historical earlier information into and out of the cell. [12]

And although many variants to the LSTM architecture has been proposed and discussed, research have shown that none of the variants improves over the standard significantly and hence we aim to use the standard LSTM architecture, during our implementation, with more focus on forget gates and output activation function the two most critical components of the LSTM. [12]

In our case, the LSTM model will look at the historical product demand trend and then predict the future price based on the historical demand trend.

We will be using the Sequential model, which is a linear stack of layers of the models. [13] We will then add our LSTM model as the part of a layer of the Sequential model.

2.3.1 LSTM Crucial Parameters

As for the parameters, we will need to specify the following which we will elaborate furthermore,

1. *Learning Rate:* It is one of the most important hyper-parameter, always accounting for more than two third of the variance, which makes it very important to set it correctly to achieve good performance. Figure showing the importance and its impact on the error and the time, as observed from a research papers has been attached below to illustrate and emphasize its importance. [12]

   In our case since we will be using the Adam optimizer to find the optimal learning rate and hence we will not have to worry about finding the best learning rate, as it will be done automatically by the optimizer itself.

LSTM: A Search Space Odyssey

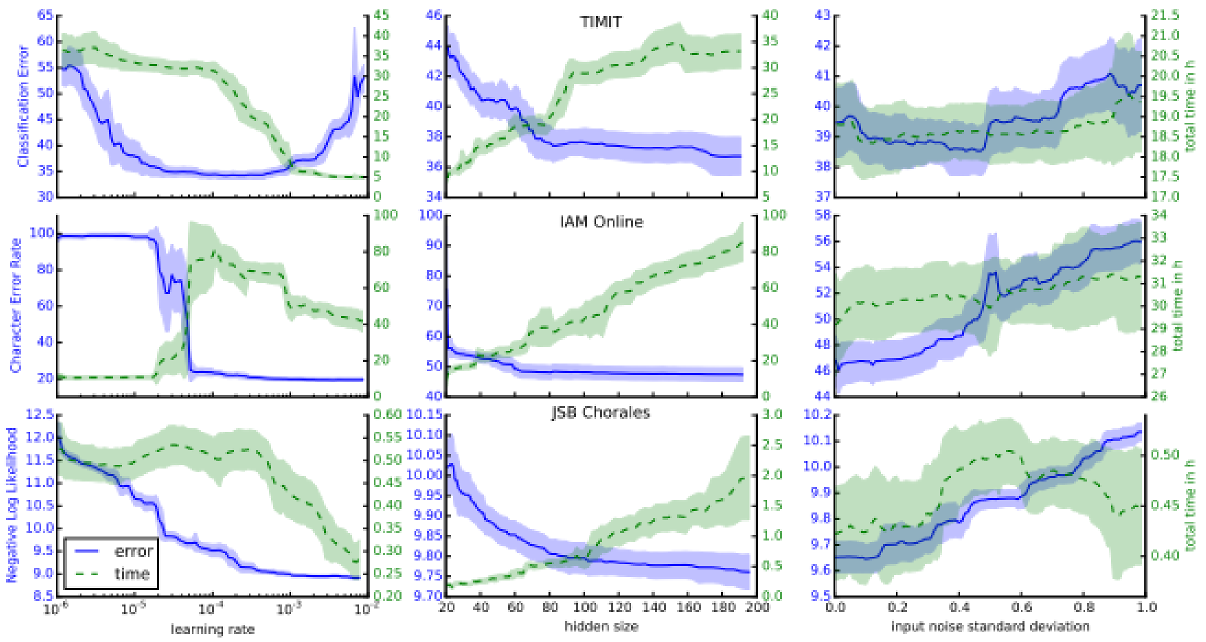

Figure 4. Predicted marginal error (blue) and marginal time for different values of the *learning rate*, *hidden size*, and the *input noise* (columns) for all three datasets (rows). The shaded area indicates the standard deviation between the tree-predicted marginals and thus indicating the reliability of the predicted mean performance. Note that each plot is for the vanilla LSTM but curves for all variants that are not significantly worse look very similar.

Figure 5 Showing the impact of the Learning Rate on Error percentage and Time taken. [12]

2. *Optimizer:* "Adam" optimizer has been chosen as the optimizer of choice. Below we will briefly discuss some basic optimizers available for us and finally discuss on our choice of "Adam" over others
   a. SGD: Stochastic Gradient Descent: SGD also known as incremental gradient descent tries to find minimum or maximum error via iteration. However, one of the major problem it suffers from is that, when the objective function is not convex or pseudo convex, it is almost sure to converge to a local minimum.
   Since SGD has trouble navigating the ravines, where they oscillate across slopes while making hesitant and slow progress towards bottom, we will not be using this optimizer. [14]

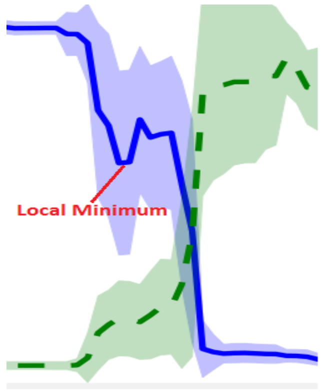

Figure 6Showing the Local Minimum

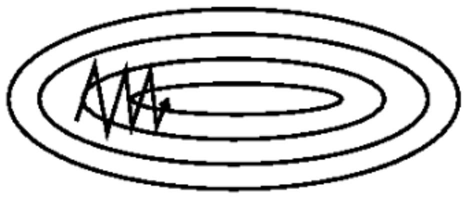

Figure 7 Showing how SGD oscillates when faced with ravine [14]

b. Nesterov accelerated gradient: We can understand Nesterov Accelerated Gradient better with the following example. Let's imagine a ball that rolls down a hill blindly following the slope. However, it will be nice to have a smarter ball that knows where it's going and slows down before the hill slopes up again. The Nesterov accelerated gradient gives this precision for our momentum term. [14].

c. Adagrad: In Nesterov accelerated gradient, while we could adapt our gradient updates to the slope and speed up the SGD, it would be even better if we could adapt our updates to each individual parameter i.e. perform larger updates for infrequent parameters and smaller updates for the frequent parameters. This is achieved with Adagrad. This in turn increases speed, scalability and robustness of SGD and could be used to train large scale neural nets. This was found to be especially suitable for sparse data and was at Google used to recognize cats in YouTube videos. [14]
One of the main benefit of the Adagrad is that it eliminates the need to manually tune the learning rate. However, one of the main disadvantages is its accumulation of squared denominator which keeps growing in training and thus makes the learning rate infinitesimally small, such that no more learning / acquiring of the additional knowledge is possible. [14]

d. AdaDelta: The AdaDelta optimizer is the extension to Adagrad and aims to solve the problem of infinitesimally small learning rate. It does so by ceiling the accumulated past gradient to some fixed window size. As in Adagrad, we do not need to set a default learning rate. [14]

e. RMSProp: RMSProp and AdaDelta have both been developed independently to resolve the Adagrad's diminishing learning rate problem. Unlike in AdaDelta however we need to specify the Gamma and learning rate (n), which is suggested to be set to 0.9 and 0.001 by the RMSProp algorithm developers Hilton. [14]

f. Adam: It is also another method that calculates learning rate for each parameter that is shown by its developers to work well in practice and to compare favorably against other adaptive learning algorithms. The developers also propose the default values for the Adam optimizer parameters as Beta1 – 0.9 Beta2 – 0.999 and Epsilon – 10^-8 [14]

To summarize, RMSProp, AdaDelta and Adam are very similar algorithm and since Adam was found to slightly outperform RMSProp, Adam is generally chosen as the best overall choice. [14] Hence we also will be using the Adam as our optimizer choice

3. *Loss Function:* Loss functions are used to optimize the parameters of the model at the next data run. In the next data run, the weight parameters are tweaked such that cumulative error as computed by the loss function is lesser than in the previous run. In our particular case, mean_squared_error has been used, as we want to have a model that does not generate predictions which deviate largely away from the actual predictions. Large deviations or differences will imply that either we will overstock on the product or understock on the product, which in either of the case is very bad for the business. We prefer slight inconsistencies over large prediction differences, which is exactly what the mean squared errors does as it penalizes heavily, when there are large differences for actual and predicated demand value.

4. *Batch Size:* Batch size determines how many example the model will look at before making a weight update. [15] Larger batch sizes during iteration allows the model to take bigger step-sizes, which means that the optimization algorithm will make progress faster and hence that the final model will be learnt faster. The larger batch size is very attractive computationally in case of deep learning with GPU's i.e. until the memory is filled up, as it allows easy parallelism across processors / machines. However, the batch size cannot be increased infinitely and cannot exceed 1/L where L is Lipchitz constant or smoothness constraint. [16]

   While selecting the optimal batch size, the efficiency of the training and the noisiness of the gradient estimate must also be taken into consideration [16]. For example, let's consider we have a 10,000-training example. If we take the batch size of 1000 then we will have to compute the gradient on a training size of 1000. This gradient is then used to determine whether each of the weights, should be increased or decreased and the magnitude with which it must be done so, to reduce the output error accuracy. However, if we take the batch size of 10, it means that our learning process is going to be 100 times slow, since the gradient computation is almost linear to batch size. The 10-batch size model is going to make 10 parameters update at a time vs the 1000 parameter update done by the 100-batch size model. This means that the lower batch size model is going to be slower with the training time. [16]
   However, on the noisiness part, our 10-size model will be computing the gradient for the 10-training data and hence 10 data size gradients will be a lot noisier than the 1000 data size gradient. Although noise is usually considered bad, in some cases, noise might even be good. For example, if our data contains numerous valleys then with large batch size, our model may get stuck into the first valley it falls into, however in case of the smaller batch size, the noisiness gradient might be enough to push the model out of the first shallower valley or shallow minima to converge into global minima. In practice, moderate mini-batches (10-500) are generally used, combined with decaying learning rate which guarantees long run convergence while having the ability to jump out of shallow minima [16] The increase in minibatch size have also been found to typically decrease the rate of convergence. [17]

5. Epoch Size: Epoch size represents the total number of iterations the data is run through the optimizer [18] Too few epochs, then the model will prematurely stop learning and will not grasp the full knowledge of the data, while with too large epochs, the training time will be longer and the model may train itself futilely without learning any new information.
   In the Figure 8 below, we can clearly observe that, the increase in the number of epochs leads to the increase in model accuracy and the reduction in error (loss) to a certain point. It also must be noted that, infinitely increasing the epoch number does not increase the accuracy infinitely. For example, in case of the green line, representing the high learning rate, it is observed that no matter the increase in the number of epochs after the black circle depicted point, the model accuracy remains the same, however the drawback in the case is that with greater number of epochs and no accuracy increase, the only activity happening is the futile waste of the computation resource and the time.

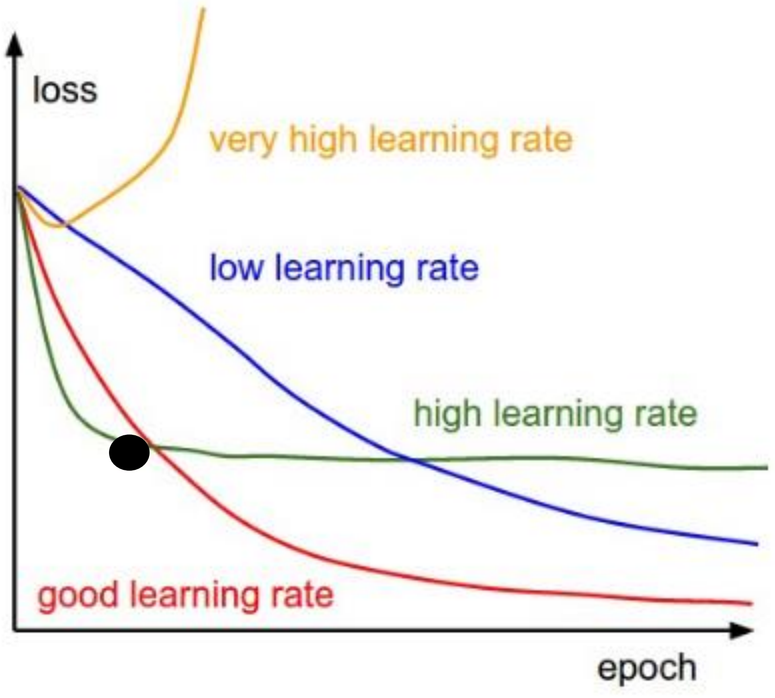

Figure 8 showing the impact of the learning rate and the epoch size on error [19]

2.4 BENCHMARK

For benchmarking, as advocated by the M3 Competition research conclusion, that simple methods such as Gardener's Dampen Trend Exponential Smoothing (ETS) performs as well and even do better in many cases [20] , we used the simple ETS model as the benchmark. We fed the preprocessed data i.e. after removing the extreme outliers and after application of moving average of two time steps, the same data we fed for the LSTM to the benchmarking ETS model.

Since the M3 data [21] and the ETS [22] , both were not available for python, we used the R statistical tool [23] to perform rapid quick benchmarking. We exported both the preprocessed data and the prediction data from the python program into the R program for benchmarking purpose. The R codes for benchmarking has been attached with this report in the Appendix section,

Exponential Trend smoothing	RMSE Train	RMSE Test
AAN	0.081	0.5

Table 1 Benchmark Error Score using Exponential Trend Smoothing

3 METHODOLOGY

3.1 DATA PREPROCESSING

During our data exploration process, we observed three main peculiarities in the data namely data scale, missing value and the outlier.

De-normalized Data: As for the de-normalized data scale between 649 and 1637, since the LSTM algorithms are sensitive to scale especially when the sigmoid or tanh activation functions are used, it is generally recommended to perform Feature scaling to 0-to-1. [24]. Hence we performed Feature scaling and scaled back our product sales quantity between 0-to-1.

Missing Data: As in the case of missing sales data between 13th of each month to the 1st of the next month e.g. missing sales data from 12th- Jan to 1st Feb. Since we do not have the data, we ignore the sales data between those missing date intervals and assume our data as continuous sales data, missing those interval periods data.

Outliers: As for the outliers, we have outliers in the data for example, sold quantity 1637. Also, furthermore, in the time series data, there exists short term fluctuations which might hinder the long-term trends or cycles. To smooth out short term fluctuations and highlight long term trends, we applied the moving average for the training data over the two (selected with experimental testing) subset of the number series. [25] The moving average works by taking the average of the initial fixed subset of the time series data. This created a new subset of numbers which are the ones averaged over the specified number of corresponding subset. We however refrain from applying the moving average to the test data, since we want to evaluate the models true accuracy in the real data rather than over the smoothen data.

Although, moving average was initially thought of as a solution to counter the outlier inconsistencies, during the model robustness test with extreme outlier values injection, it was observed that although the MA was able to moderate it and was able to withstand the extreme outlier values from injecting bias, it nonetheless added significant error to the model's prediction. Hence to furthermore, increase the robustness of our model, we will reject the extreme outliers for example 3000, when the last extreme outlier point prior to the injection was 1600 with average 1164 standard deviation 264. Hence, we will reject the extreme outliers, all-together from consideration from processing data i.e. data that are more than 3 standard deviation away for example 3000 in our case, i.e. any value greater than 1166 +3 *264 = 1958 and less than 1166 – 3 *264 = 374.

Also since the sales demand cannot be a negative value, we also reject any probably data errors i.e. negative values from our dataset. This extreme outlier/ negative value cleaned data was then fed into the moving average to further improve the model robustness and long term trend prediction accuracy.

3.2 Implementation

Based on our background study of the algorithms and techniques, and exploring the advantages of the deep neural network, LSTM to generate more robust models we decided to choose the Long Short Term Memory Recurrent Neural network to generate the prediction model

We used “Keras”, a high- level neural networks library, written in python and capable of running on either TensorFlow or Theano. [13] with python 2.7 on a Linux machine.

The LSTM standard library provided by the Keras was used. The optimizer, parameter for the LSTM was explored over and based on the conceptual background knowledge gained, as discussed in the Algorithm and Technique section of the report earlier, the optimal optimizer algorithms, number of epochs, the batch size and other parameters were selected based on the background study. Thus, selected parameters, were then ran through the experiment to ensure its practical validity.

Initially we downloaded the data from the Dataset. The data in the csv format was then preprocessed and then split into train and test set with 66% data allocated as training data and 33% as testing data.

The training and the testing data was then converted into the LSTM expected input shape by using python – numpy libraries reshape function i.e. 1 column, any rows data format e.g. [[[day1_demand]], [[day2_demand]],…….]. The data was then fed into the LSTM neural network to generate model, which was then later used to evaluate performance against the test data.

An LSTM model has following implementation parameters which we tweaked upon and implemented with. In our case since we will be using the Adam optimizer as discussed earlier.

a. Input Shape (x, y) i.e. The first layer of the model needs to know what input shape it should expect. [26] For our particular case, the input dimension must have 1 column but can accepts any number of data. e.g. [[[day1_demand]], [[day2_demand]],…….] Using our numpy library’s reshape function, we transformed the csv data to the above-mentioned input shape.
b. Input Data: Keras model are trained on the Numpy array [26], which is why we convert the CSV data to such for our case.
c. Batch size: no of samples considered in each batch iteration. When the data is extensively large then processing all the data by the LSTM can be an infeasible solution, in such case for computational reasons, the data is often processed in mini-batches. [14]. However, since in our case, the data is not quite large, we will be using the full dataset at once to compute the model with batch size set as 1.
   e.g. model.fit(data, labels, nb_epoch=10, batch_size=32) means model training is done in batches of 32 samples, iterating 10 times
d. Nb_worker: number of processes to spin up to speed the learning process [18]
e. Dense (64): means fully connected layer with 64 hidden units. We will however be using Dense (1) i.e. 1 hidden unit
f. Multilayer LSTM: (LSTM stacked on top of each other to learn higher- learn temporal representation. We explored stacking multiple LSTM layer on top of each other to explore, if our model will be able to learn higher- level temporal representations, however during runtime inspection, analyzing our results, we found that no significant error reduction i.e. knowledge gain was obtained with addition of each LSTM layers. Hence we reverted back to using single LSTM layer.

3.3 Refinement

We initially started with the default LSTM model, with default parameter settings. The results were poor on the default parameter settings as reported in the table below. The parameters of the LSTM model were then optimized based on the theoretical background study. The conceptually selected optimized parameter settings were then experimented thorough the actual run.

The parameters such as optimizer, loss functions were optimized to “Adam” and “mean squared function”, based on theoretical background study, as mentioned previously in the “Algorithms and Technique” section of the report earlier.

Loss Function Refinement: The rationale for the refinement of the loss function to the “Mean squared Error(MSE)”, is to ensure that the greater difference between the actual and the predicted value is penalized more. The rationale for the selection of the MSE has been more elaborately discussed in the Metrics section of the report. In apart to the theoretical background study based selection, we furthermore also experimented both approaches MAE and MSE and observed MSE to be better, in alignment with the background study.

Optimizer Refinement : As for the rationale for the refinement of the optimizer, we discussed different optimizers available, in the Algorithms and Techniques -> LSTM Crucial Parameters Section of the report earlier. We discussed each optimizer and their pros and cons. And to summarize RMSProp, AdaDelta and Adam were depicted as very similar algorithm with Adam slightly outperforming RMSProp and AdaDelta. Furthermore, it was also generally observed as the best overall choice. [14] Hence we chose Adam as our optimizer. With optimizer choice finalized, we however also tested other optimizers and as had been suggested from the background study, Adam was indeed found to be the best optimizer choice.

Epoch Number Refinement: As for the LSTM epoch number refinement, it was based on our theoretical background study as discussed in the LSTM Crucial Parameters section of the report. As discussed earlier that the increase of number of epoch often tend to increase the model accuracy, we experimented by gradually increasing the epoch size from 10,50,100,200,300 and found the 100 number of epochs to be best parameter value.

LSTM Batch Size Refinement: As for the LSTM batch size, based on the Theoretical background knowledge as discussed in LSTM Crucial Parameters, we limited our search to small mini batches. We performed random search for batch size with 32, 10, 5, 1 and found that as we reduced the batch size the model robustness increased and finally settled in with the 1 batch -size, which generated the most robust model for us.

Similarly stacking of LSTM layers to extract higher order temporal knowledge was experimented, with no significant knowledge gain or the error accuracy loss. And hence LSTM Stacking was rejected.

All of these parameter optimization tests were performed manually, tweaking one parameter at a time. While grid search or random search could have been the other automated faster alternatives to performing the manual refinement, it will have eliminated the possibility to study each parameter’s effect in the model for us. And hence to understand each parameter’s impact better and to understand the model, the concepts/ parameters better, we limited ourselves to the manual parameter refinement approach. Although it was time consuming and laborious for us, it helped us to understand each of the hyperparameter and its effect in detail and strengthened our knowledge in the process.

All of the parameterized test results could not be include here in the Refinement section, hence only the most prominent few configurations have been listed below. As for all the configurations tried during the refinement process, it has been added to the appendix section with the error scores alongside.

Few of the major LSTM experiments with its parameters, and its train and test score as observed have been presented below. The changes in the parameter from one iteration to next, has been highlighted in yellow.

LSTM with parameters as selected	RMSE Train Error score	RMSE Test Error score
Initial Parameter Configuration: LSTM: 1 layer – Loss function: Mean Absolute Error, Optimizer: SGD, epoch =10, batch_size = 32	0.5520	0.6803
LSTM: 1 layer – Loss function: Mean Squared Error, Optimizer: SGD, epoch =10, batch_size = 32	0.3838	0.5303
LSTM: 1 layer – Loss function: Mean Squared Error, Optimizer: SGD, epoch =100, batch_size = 32	0.2592	0.3195
LSTM: 1 layer – Loss function: Mean Squared Error, Optimizer: SGD, epoch =100, batch_size = 1	0.2287	0.2776
LSTM: 1 layer, Dense Layers -6, Loss function: Mean_squared_error, Optimizer: Adam, epoch =100, batch_size = 1	0.0820	0.1201
LSTM 30 Dense Layers -6, Loss function: Meansquared_error, Optimizer: Adam, epoch =100, batch_size = 1	0.0763	0.0806
LSTM: 30 layer, Dense Layers -6, Loss function: Meansquared_error, Optimizer: Adam, epoch =100, batch_size = 1, with moving average -2	0.0357	0.0364
FINAL Config: LSTM: 30 layer, Dense Layers -6, Loss function: Meansquared_error, Optimizer: Adam, epoch =100, batch_size = 1, with moving average -3, outlier > 3 sd away rejected	0.0337	0.0220
LSTM: 1 layer, Dense Layers -6, Loss function: Meansquared_error, Optimizer: Adam, epoch =100, batch_size = 1 with moving average -4	0.0295	0.0250

Table 2 LSTM Parameter Configuration and their respective Error Scores

4 RESULTS

4.1 MODEL EVALUATION AND VALIDATION

The final model has been obtained with extensive theoretical background study based parameter optimization, along with the few experiment based optimization. Most of the theoretical background study based parameter optimization were found valid and indeed increased the accuracy of the model.

Some of the parameters, such as LSTM stacking although conceptually supposed to capture the higher order temporal representation in the data and increase the model accuracy, failed to do so in reality and implementation, most probably due to the lack of higher order temporal data presence in our data.

The final model thus finally obtained was found to be reasonable and not overfitting, generic enough, aligning with the solutions expectation, as observed in the plot below. In the figure below, the blue lines represent the actual data, while the green colored lines represent the prediction made by the final model on the training data as plotted against the actual data. Similarly, the red colored lines represent the prediction made by the model on the test data.

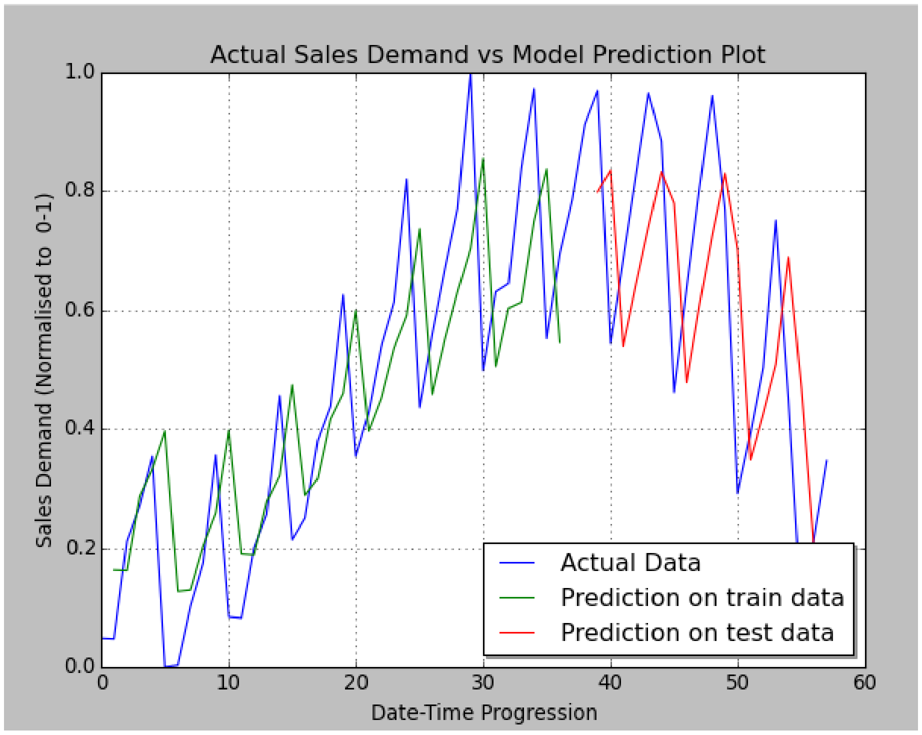

Figure 9 Showing Original Product Sales Demand Data vs Model Prediction on train data – green. On test data - red

As for the model evaluation and validation, all the tests including the data variations and different data sets were done based on the same parameter configuration as that of our original best model. Special care was taken to ensure so, as violation of the assumption will imply that the model validation will be performed on a different model with different configuration to that of the best model, we had developed.

We started the data validation process with trying our model on a different data set from a different domain. The model was tested against air passenger – travel demand data – (Jan 49 –Dec 60) [27], particularly in “Air Passengers – travel demand” time series data. The model was observed to found perform very well with RMSE – score of 0.0208, even far better than in the case of our product sales demand prediction case. The sales demand forecast, plot for the air passengers has been attached in the appendix section of the report.

The model was observed to be robust enough, as it was able to predict the demand forecast across multiple data sets, more accurately i.e. air passengers demand and the sales demand.

We also tried small perturbation in the data, such as replacing the outliers with moderate values in some of the time series data and tested our model for its robustness with outlier moderated using “Winsorising” i.e. replacing the outlier with the nearest non-suspect value (its earlier day value in our case) and in the second case, injecting extremely high outlier value. And as expected, we observed that the model was robust enough to sustain the correct prediction.

For the first case of winsorising the following values were replaced in the data, as

Original Data	Changed to
“1-10”,1032	“1-10”,700
“2-03”,1126	“2-03”,800

In this case, the introduction of the winsorisation of the outlier values lead to the change in the RMSE score on the test data from 0.0220 to 0.236 i.e. very few RMSE score increase in the error, thus proving the robustness of the model.

In another outlier case of extremely high outlier value injection, as in the case of the following data, the RMSE score changed from 0.0220 to 0.0160, as the outlier values was removed. Although replacing some of some of the training data values with extreme values lead to increase in the model accuracy, in another instance (Iteration 2) when another set of outlier values were changed, the error increased from 0.0220 to 0.245, which indicates that the first instance of decreased model accuracy was a chance encounter. However more importantly, we can observe from both instances, that our model is robust when faced with outliers, able to deal with multiple extreme values.

Test Iteration 1:

Original Data	Changed to
“1-10”,1032	“1-10”,3000
“2-03”,1126	“2-03”,2500

Test Iteration 2:

Original Data	Changed to
1-11, 776	1-11, 2600
2-01, 885	2-01, 2800
2-03, 1126	2-03, 2500
2-12, 1274	2-12, 3000
3-07, 1165	3-07, 2000

Based on this observation, it can be concluded that our model is robust enough to sustain few incorrect outlier misrepresentation / inconsistencies.

We will further more test our model with missing data by in fact replacing the following data with blank as follows. We observed that the model performed robustly enough scoring from RMSE test score of 0.0220 to 0.0294 for Iteration 1 and 0.0140 for Iteration 2, thus proving the robustness of the model against the missing values.

Test Iteration 1:

Original Data	Changed to
"1-07",700	1-07,
"1-08", 793	1-08,
3-12, 1216	3-12,

Test Iteration 2:

Original Data	Changed to
"1-07",700	1-07,
"1-08", 793	1-08,
2-05,932	2-05,
2-12,1274	2-12,
3-12, 1216	3-12,

Furthermore, we also verified the model's prediction accuracy, not only based on the RMSE score, but by plotting the predictions against the original data and re-confirmed the models' validity and trust. The figure showing the plot has been presented in the earlier section of this "Result", Fig. 1.

4.2 JUSTIFICATION

After having developed the final model via numerous iteration and tuning, we finally obtained the RMSE score of 0.03 on the training data and 0.02 on the test data. The RMSE score when compared with the benchmark is significantly better and improved. The results found is significantly stronger then the benchmark result reported earlier. Both the benchmark model, its score and the final model 's RMSE score has been presented in the below table.

Model	RMSE Train	RMSE Test
Exponential Time smoothing AAN (Benchmark model)	0.081	0.5
LSTM: 30 layer, Dense Layers -6, Loss function: Meansquared_error, Optimizer: Adam, epoch =100, batch_size = 1, with moving average -3, outlier > 3 sd away rejected	0.0337	0.0220

Table 3 Showing Error score of the Benchmark vs the LSTM model

The final model has been thoroughly analyzed and discussed mainly, in the algorithm and refinement section of the report. Furthermore, based on the RMSE score and the data plot, which is also presented in the Free – Form visualization section later, we can be confident that the model thus developed has solved our problem significantly.

5 CONCLUSION

5.1 FREE-FORM VISUALIZATION

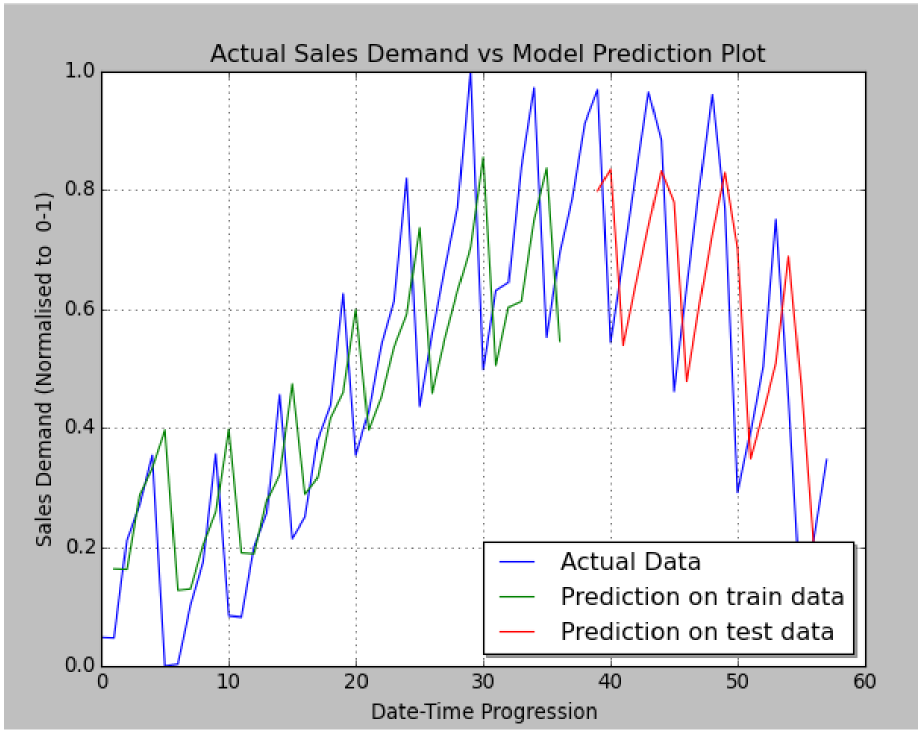

Figure 10 Showing the Actual Data vs Model Prediction

As presented in the figure above, the time series data consists of data points with seasonal patterns, moving average i.e. increasing or decreasing sales on average, some random errors such as increase or decrease in sales, which make it a complex problem to tackle.

The blue line in the figure above represents the actual sales demand data as plotted across time and is used for the model training purpose, while the green line represents prediction made by the model for the training data. We observe that the model is able to significantly correctly predict for the train data. Moreover, we are interested in how the model will perform for the test data to check for, if the model has indeed over trained itself, overfitting and thus being non-usable for the test and the general data. The red color line in the above figure, is the prediction for the test data, which shows that indeed it is befitting very well for the test data as well.

5.2 REFLECTION

We started with the problem of making the accurate forecasting for the sales data for each of the Product sold, to prevent both the under stocking and over stocking situation and hence ultimately optimize our inventory flow to increase the profit margins of a company.

In our particular case, we took the data of a Product A of a plastic manufacturer, then preprocessed it for scale and moving average in order to be able to train our model better.

For the model, we used the Long-short term memory recurrent neural network, which can remember or forgot the historical sales data, as is needed to generate the best prediction model. The model was fed with 66% of our data while the rest 34% were held out for validation.

We then iterated and optimized our model across various parameters, primarily the choice of right optimizer, right error calculating loss function, right number of iterations. And finally, the final robust model was obtained.

The final LSTM model was then compared with our simple Exponential Trend smoothing (ETS) model, with the finding that our model outclassed it in terms of more accurate prediction.

Of particular interest were the model evaluation section where the idea of the need of extreme outlier rejection, moving average once rejected were found to be significant. These preprocessing handles were then introduced later, as we went on observing it. This seemed like overlooking of the data preprocessing in our part, especially to create a robust model, which not only performs better for our data case but also across generic time series data set.

One of the major difficulty, we encountered was understanding the LSTM model and more significantly its optimization parameter, to generate the best model. This can be most probably due to the novelty of its introduction of the deep neural network i.e. LSTM, in our particular case.

Furthermore, we ultimately were able to obtain a model that was significantly able to make accurate prediction on time series data. Although we did not check the model against a number of data samples, when checked against numerous variations i.e. extreme outliers, missing data and different data set as in air travel demand, the model performed robustly and hence can be concluded that the model can be used in the general setting to solve this type of time series prediction problems.

5.3 Improvement

We tried simple linear LSTM stacking implementation, which in fact in our case resulted in decrease in the model robustness and hence was rejected. One of the areas to be explored further will be staking the LSTM cells with different stacking configuration to explore the models further robustness. Trying those different configurations to explore if the model's accuracy can be further increased is one of the areas to explore further.

Furthermore, testing the model across many other different sales domain data set i.e. cloth sales domain, electronic sales domain etc., would be an interesting area to explore further to ensure the model robustness.

6 REFERENCES

[1] "Harvard Business Review Article : How to Choose the Right Forecasting Technique," [Online]. Available: https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique.

[2] "Statistical Forecasting, John Galt, the forecast experts," [Online]. Available: http://johngalt.com/galt_university/learning-resources/business-forecasting-glossary/statistical-forecasting/.

[3] "Statistical Forecasting, Introduction: Eyeon Forecasting Experts," [Online]. Available: http://www.eyeon.nl/documenten/Pid11/pid11_10_masterclass_statistical_forecasting.pdf.

[4] "How to explain Forecasting in Retail: Small Business," [Online]. Available: http://smallbusiness.chron.com/explain-forecasting-retail-37966.html.

[5] "Challenges Retailers Face in forecasting," [Online]. Available: http://www.slideshare.net/sankarshanjoshi/retail-forecasting.

[6] [Online]. Available: http://www.tompkinsinc.com/ten-business-reasons-demand-forecasts-matter/.

[7] [Online]. Available: https://datamarket.com/.

[8] "Analytics Vidhya : Time Seires with python," [Online]. Available: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/.

[9] "Parametric and Non-Parametric Time Series Analysis," [Online]. Available: https://arxiv.org/pdf/0801.1599.pdf.

[10] "Single Layer Knn," [Online]. Available: https://en.wikipedia.org/wiki/File:Single_layer_ann.svg.

[11] "Neural Network in simple words," [Online]. Available: http://programmers.stackexchange.com/questions/72093/what-is-a-neural-network-in-simple-words.

[12] "LSTM : A search Space odessey Journal paper," [Online]. Available: https://arxiv.org/pdf/1503.04069v1.pdf.

[13] "Keras," [Online]. Available: https://keras.io/getting-started/sequential-model-guide/.

[14] "Gradient Descent optimisation algorithms," [Online]. Available: http://sebastianruder.com/optimizing-gradient-descent/index.html.

[15] "Stackexchange : How large should the batch size be for stochastic gradient descent?," [Online]. Available: http://stats.stackexchange.com/questions/140811/how-large-should-the-batch-size-be-for-stochastic-gradient-descent.

[16] "Quora : How does batch size affect performance?," [Online]. Available: https://www.quora.com/Intuitively-how-does-mini-batch-size-affect-the-performance-of-stochastic-gradient-descent.

[17] "Efficient Mini-batch Training for Stochastic Optimisation," [Online]. Available: http://www.cs.cmu.edu/~muli/file/minibatch_sgd.pdf.

[18] "Keras Sequential model API," [Online]. Available: https://keras.io/models/sequential/.

[19] "CS231n Convolutional Neural Netowrk," [Online]. Available: http://cs231n.github.io/neural-networks-3/.

[20] "Makridakia-The M3 Competition, International Journal of Forecasting," [Online]. Available: http://www.forecastingprinciples.com/paperpdf/Makridakia-The%20M3%20Competition.pdf.

[21] "Package Mcomp for R," [Online]. Available: https://cran.r-project.org/web/packages/Mcomp/Mcomp.pdf.

[22] "Package Forecast for R," [Online]. Available: https://cran.r-project.org/web/packages/forecast/forecast.pdf.

[23] " R Statistical Computing Tool," [Online]. Available: https://www.r-project.org/.

[24] "Time Series Prediction with LSTM Recurrent Networks," [Online]. Available: http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

[25] "What is Moving average os Smoothing TEchnique ? Nist.gov," [Online]. Available: http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc42.htm.

[26] "Guide to Sequential model," [Online]. Available: https://keras.io/getting-started/sequential-model-guide/.

[27] "DAtamarket : International Air passengers," [Online]. Available: https://datamarket.com/data/set/22u3/international-airline-passengers-monthly-totals-in-thousands-jan-49-dec-60#!ds=22u3&display=line.

[28] [Online]. Available: https://www.walmart.com/.

[29] "Keras architecture," [Online]. Available: https://github.com/fchollet/keras/issues/2496.

[30] "SAS Retail Forecasting Fact Sheet," [Online]. Available: http://www.sas.com/content/dam/SAS/en_us/doc/factsheet/sas-retail-forecasting-105158.pdf.

[31] "Demand Assortment and Optimisation Ch 2," [Online]. Available: http://repository.upenn.edu/cgi/viewcontent.cgi?article=1569&context=edissertations.

7 APPENDIX:

7.1 PREDICTION PLOT ON THE AIRLINES PASSENGERS DATA

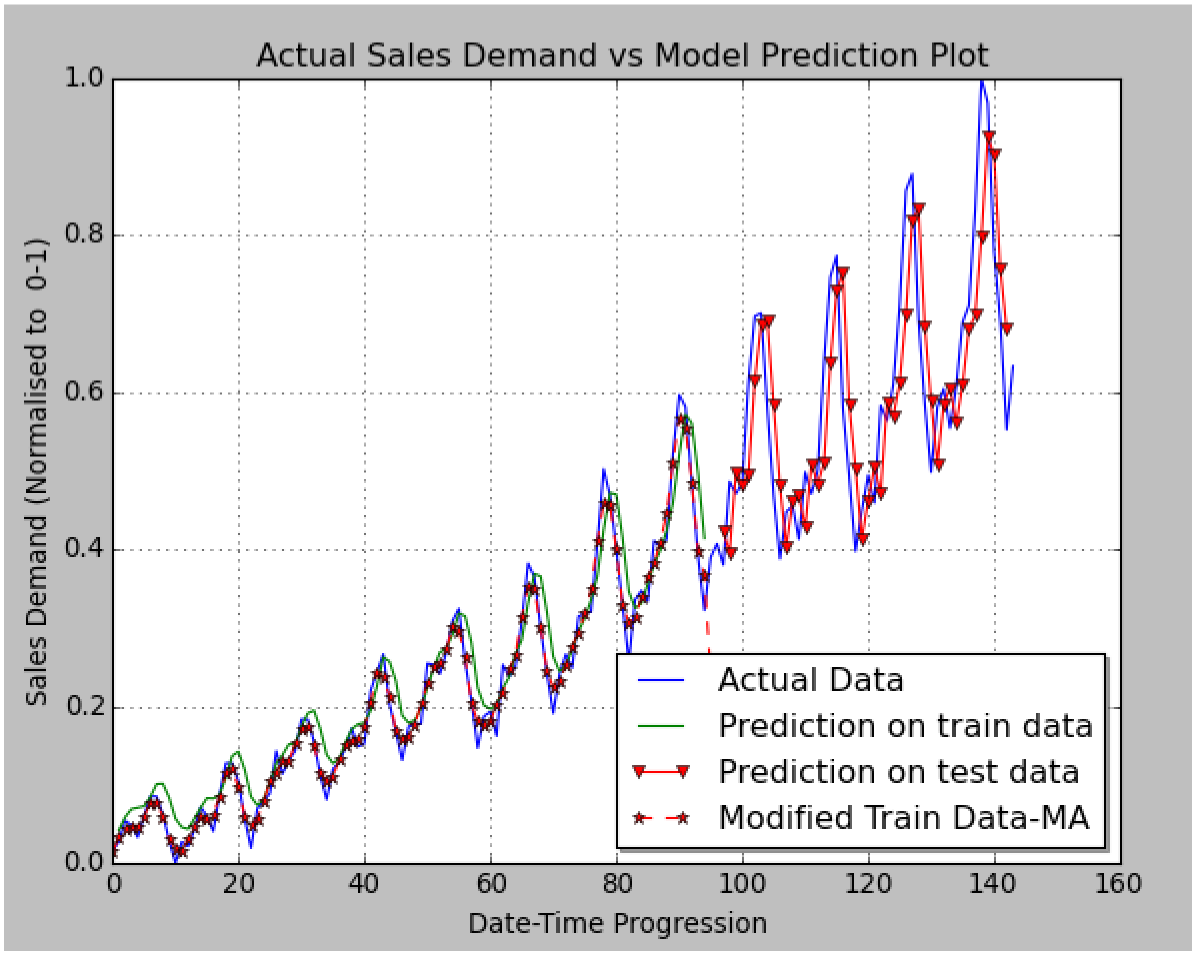

Fig Showing Actual travel demand vs the prediction made by our model on the train data and the test data

7.2 DATA USED TO TEST FOR MODEL ROBUSTNESS AGAINST FEW OUTLIERS MISREPRESENTATION.

"Month","Monthly sales of product A for a plastics manufacturer"
"1-01",742
"1-02",741
"1-03",896
"1-04",951
"1-05",1030
"1-06",697
"1-07",700
"1-08",793
"1-09",861
"1-10",861
"1-11",776
"1-12",774
"2-01",885
"2-02",938
"2-03",938
"2-04",898
"2-05",932
"2-06",1055
"2-07",1109
"2-08",1285
"2-09",1030
"2-10",1099

"2-11",1204
"2-12",1274
"3-01",1468
"3-02",1107
"3-03",1223
"3-04",1326
"3-05",1422
"3-06",1637
"3-07",1165
"3-08",1290
"3-09",1303
"3-10",1486
"3-11",1611
"3-12",1216
"4-01",1349
"4-02",1436
"4-03",1555
"4-04",1608
"4-05",1208
"4-06",1341
"4-07",1473
"4-08",1604
"4-09",1528
"4-10",1131

"4-11",1296
"4-12",1453
"5-01",1600
"5-02",1420
"5-03",971
"5-04",1066
"5-05",1170
"5-06",1403
"5-07",1119
"5-08",783
"5-09",901
"5-10",1023

7.3 R- CODES USED FOR BENCHMARKING:

Exponential Time smoothing performed on our Sales demand forecast data

library(Mcomp)

library(forecast)

product_a_sales_train_preprocessed_ma_2step_data = c(0.03156026,0.10212763,0.1762411 ,0.27872336,0.20815599,0.1762411 ,0.17695032,0.2049645 ,0.17411345,0.12198581,0.17943261,0.30425529,0.30886521,0.35035459,0.34432622,0.48156025,0.47269497,0.46914889,0.4404255 ,0.59574467,0.59858153,0.60531912,0.55496448,0.66666661,0.81347511,0.75638292,0.70957444,0.59113473,0.7049645 ,0.81879427,0.83510635,0.81737588,0.79751772,0.88936168,0.62730495)

product_A_ts_data <- ts(product_a_sales_train_preprocessed_ma_2step_data, start=c(1950), frequency=1)

product_A_ts_model = ets(product_A_ts_data, model = "AAN")

summary(product_A_ts_model)

```
# Generate 20  future predictions

pred_for_product_A = forecast(product_A_ts_model, 20)


#Calculate rmse error on the held out test data

pred_for_product_A_vectorised = (unclass( pred_for_product_A)$mean)

actual_future_data_for_product_A_from_data_scaled = c(0.91276592,0.96914893,0.54361695,0.68510634,0.8255319
,0.96489352,0.88404256,0.46170205,0.63723403,0.80425531,0.96063823,0.76914889,0.2914893
,0.39255315,0.50319141,0.75106376,0.44893616,0.09148937,0.21702129,0.34680849)

rmse(pred_for_product_A_vectorised, actual_future_data_for_product_A_from_data_scaled)  # 0.5
```

7.4 LSTM PERFORMANCE ACROSS NUMEROUS PARAMETER OPTIMIZATION TEST

LSTM with parameters as selected	RMSE Train Error score	RMSE Test Error score
Adagrad – LSTM 1 layer epoch – 100, batch_size = 1	0.18	0.23
Sgd	0.18	0.22
Adamax	0.1718	0.2176
Nadam	0.1698	0.2170
Adam	0.1720	0.2179
Rmsprop	0.1718	0.2176
Adadelta	0.19	0.23
Adam – 100 -1	0.1736	0.2182
Adam – 10 -1	Bad	Bad
Adam – 1000 – 1	0.1668	0.2164
Adam – 200 -1	0.1697	0.2170
Adam – 50 -1	0.1759	0.2187
Adam – with no dense	0.1720	0.2271
Adam – with dense	0.1720	0.2179
Adam – with LSTM 10	0.1744	0.2193
Adam with LSTM 1	0.1666	0.2168
Adam with LSTM 1 no dense	0.1720	0.2271
Iter2	0.1720	0.2271
Iter3	0.1720	0.2271
Adam with LSTM 1 dense 1 epoch -100 batchsize 1	0.1666	0.2168
Adam with LSTM 2 (stacked on top of LSTM)	0.1654	0.2152

Adam with LSTM 1 dense 1 epoch -100 batchsize 3	0.1783	0.2229
Adam with LSTM 1 dense 1 epoch -300 batchsize 1	0.1649	0.2153
Adam with LSTM 1 dense 1 epoch -300 batchsize 3	0.1654	0.2154
Adam with LSTM 20 , 2 more hidden LSTM no dense	0.089	0.13
Adam with LSTM 20 , 2 more hidden LSTM no dense	0.1046	0.1701
Adam with LSTM 20 dense 1, hidden 1 LSTM		
Adam with LSTM 20 dense 1, hidden 1 LSTM	0.0863	0.1354
Adam with LSTM 20 dense 1- 12 times, no hidden LSTM	0.0762	0.0964
Adam with LSTM 20 dense 1- 3 times, no hidden LSTM	0.0751	0.0836
Adam with LSTM 20 dense 1- 2 times, no hidden LSTM	0.0768	0.0897
Adam with LSTM 20 dense 1- 6 times, no hidden LSTM	0.0763	0.0806
Adam with LSTM 20 dense 1, no hidden LSTM		
Adam with LSTM 20 dense 1, no hidden LSTM	0.0768	0.0905
Adam with LSTM 10 dense 1, no hidden LSTM	0.0763	0.0952
Adam with LSTM 5 dense 1	0.0778	0.1131
Adam with LSTM 5 dense 1	0.1699	0.2176
Adam epoch 100 batch_size 2	0.17	0.2179
Adam epoch 100 batch_size 3	0.1746	0.2197
Adam epoch 100 batch_size 4	0.1820	0.2258
Adam epoch 100 batch_size 5	0.1909	0.2356
Adam epoch 300 batch_size 3	0.1686	0.2166
LSTM: 1 layer – Loss function: Poisson, Optimiser: adam, epoch =100, batch_size = 1	0.0837	0.1265
LSTM: 1 layer – Loss function: Mean_squared_error, Optimiser: adam, epoch =100, batch_size = 1	0.0820	0.1201
LSTM: 1 layer – Loss function: Mean_absolute_error, Optimiser: SGD, epoch =100, batch_size = 1	0.1947	0.2633

7.5 Sample Input Data Set

```
"Month","Monthly sales of product A for a plastics manufacturer"
"1-01",742
"1-02",741
"1-03",896
"1-04",951
"1-05",1030
"1-06",697
"1-07",700
"1-08",793
"1-09",861
"1-10",1032
"1-11",776
"1-12",774
"2-01",885
"2-02",938
"2-03",1126
"2-04",898
"2-05",932
"2-06",1055
"2-07",1109
"2-08",1285
"2-09",1030
"2-10",1099
"2-11",1204
"2-12",1274
"3-01",1468
"3-02",1107
"3-03",1223
"3-04",1326
"3-05",1422
"3-06",1637
"3-07",1165
"3-08",1290
"3-09",1303
```

```
"3-10",1486

"3-11",1611

"3-12",1216

"4-01",1349

"4-02",1436

"4-03",1555

"4-04",1608

"4-05",1208

"4-06",1341

"4-07",1473

"4-08",1604

"4-09",1528

"4-10",1131

"4-11",1296

"4-12",1453

"5-01",1600

"5-02",1420

"5-03",971

"5-04",1066

"5-05",1170

"5-06",1403

"5-07",1119

"5-08",783

"5-09",901

"5-10",1023
```