

Time Series Analysis



INDIAN INSTITUTE OF TECHNOLOGY JODHPUR

Semester 7 Summer 2021

**Guided By
Dr. Gaurav Harit**

Submitted By :

**Aditya Kumar
B18CSE002**

**Kartik Vyas
B18CSE020**



Table of Contents

1 Overview	3
2 Motivation	3
3 Goals and Problem Statement	4
4 Datasets	5
5 Development Plans and Execution	7
6 Future Prospects	24
7 Techstack	24
8 Codebase Link	25
9 Conclusion	25
10 Acknowledgments	26
11 References	26

1 Overview

Time can be easily said to one of the most important aspects of our lives and perhaps of this world. Time is a factor that governs each and everything. Governments, organisations, institutions, individuals, all are bound together by time. While we cannot travel to the future and know it, we can definitely try to predict it.

Time Series Forecasting refers to a possible prediction of values that are going to occur in the future. Time Series forecasting provides solutions to a vast ocean of questions and curiosities. Time series analysis involves working on models that focus on capturing or describing an observed time series in order to understand the underlying causes. This field of study seeks the “why” behind a time series dataset.


We have also worked on anomaly detection; anomaly can be briefly explained as “An observation which deviates so much from other observations as to hint suspicions that it was generated from a different mechanism.” It is really important to detect these anomalies as they could help us in identifying the unexpected behaviours in time series, which could benefit several use cases.

In this report for our B.Tech Project, we have made a thorough analysis of time series and worked around the concepts of time series prediction and anomaly detection. We propose several tools and techniques, and also discuss our shortcomings, challenges that we faced along the way and the major outcomes that we have concluded.

2 Motivation

Time Series analysis has the most wide range of applications in several research fields and industries. It has tons of practical applications including:

- ❖ Weather Forecasting
- ❖ Climate Forecasting
- ❖ Economic Forecasting

- 
- ❖ Healthcare Forecasting
 - ❖ Engineering Forecasting
 - ❖ Finance Forecasting
 - ❖ Retail Forecasting
 - ❖ Business Forecasting
 - ❖ Environmental Studies Forecasting
 - ❖ Social Studies Forecasting and more

We are focusing on time series prediction and anomaly detection. We have leveraged usage of several datasets and different architectures for deep learning models to in depth analyse and make solutions for the real world problems.

For ease of understanding, we have focused mainly on medicinal datasets to answer several questions related to hospitals and pharmaceutical houses. We got motivated to follow this path due to the recent pandemic situation which the entire world is facing. The motivation was to work on real datasets and the code is written in such a robust way that it can easily be extended to any other problem and dataset.

3 Goals and Problem Statement

Our goal is to perform time-series analysis on different datasets using different deep learning architectures which can help in solving real-world problems. We aim to perform single-step time series forecasting, multistep time series forecasting, and anomaly detection for univariate and multivariate datasets. In some cases we have used multiple architectures which helped us in choosing the best architecture for that problem. Following are the goals and the architectures that we focussed on during the course of the project.

- ❖ Anomaly Detection
 - LSTM Autoencoders
- ❖ Single-Step Time Series Prediction
 - LSTM
- ❖ Multi-Step Time Series Prediction
 - LSTM

- CNN LSTM Encoder - Decoder
- LSTM Encoder - Decoder

We worked extensively keeping in mind one thought that we work on problems that have real world applications. We then worked majorly on healthcare problems owing to the damage done by pandemic Covid. Some of the problems that we have focused on in healthcare domain are :

- ❖ Prediction of number of patients incoming in a hospital
- ❖ Prediction of sales of medicines
- ❖ Anomaly of patients incoming in hospital

Solutions to these problems would also hint at an outbreak of a disease and would also aid healthcare centers to maintain medicinal inventory and slot management of healthcare officials. This being said, we have also worked on a predictive model for a service outlet where we predict the upcoming sales trend. The models are in such a way that they can be easily accounted for newer problems using different datasets.

4 Datasets

4.1 London Bike Sharing Dataset

This dataset contains data for the number of bike shares on an hourly basis. The dataset spans for a time period of 2 years. This multivariate dataset has several features in the dataset namely:

- timestamp - timestamp field for grouping the data
- cnt - the count of a new bike shares
- t1 - real temperature in C
- t2 - temperature in C “feels like”
- hum - humidity in percentage
- wind_speed - wind speed in km/h
- weather_code - category of the weather
- is_holiday - boolean field - 1 holiday / 0 non holiday
- is_weekend - boolean field - 1 if the day is weekend

- season - category field meteorological seasons: 0-spring ; 1-summer; 2-fall; 3-winter.

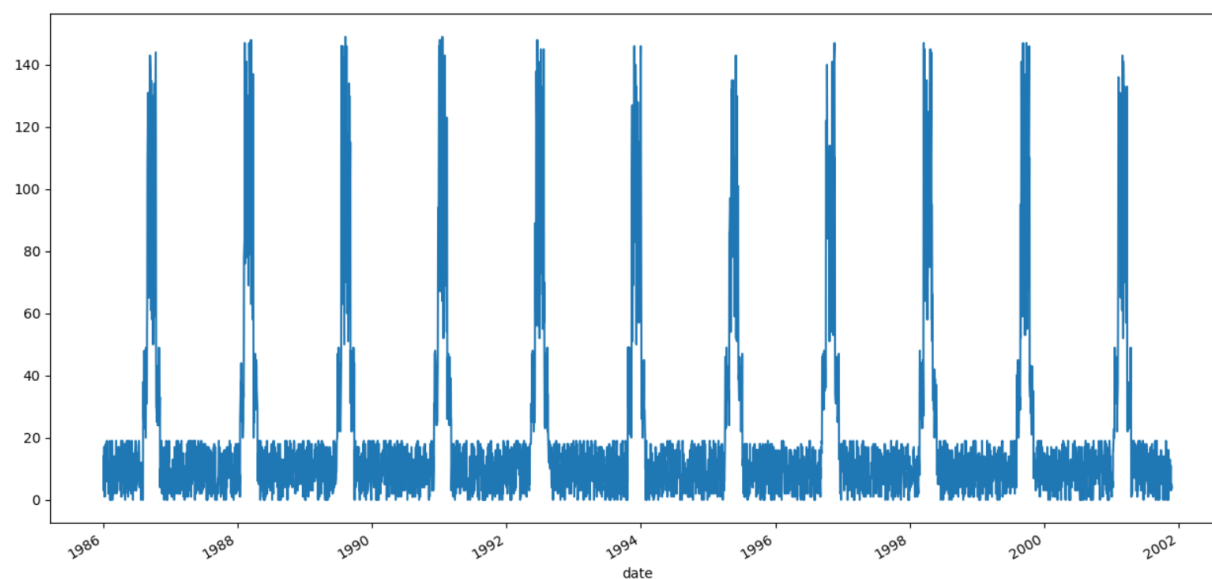
Below is an image of what the dataset looks like.

A	B	C	D	E	F	G	H	I	J
timestamp	cnt	t1	t2	hum	wind_speed	weather_c	is_holiday	is_weekend	season
04-01-2015 00:00	182	3	2	93	6	3	0	1	3
04-01-2015 01:00	138	3	2.5	93	5	1	0	1	3
04-01-2015 02:00	134	2.5	2.5	96.5	0	1	0	1	3
04-01-2015 03:00	72	2	2	100	0	1	0	1	3
04-01-2015 04:00	47	2	0	93	6.5	1	0	1	3
04-01-2015 05:00	46	2	2	93	4	1	0	1	3
04-01-2015 06:00	51	1	-1	100	7	4	0	1	3
04-01-2015 07:00	75	1	-1	100	7	4	0	1	3
04-01-2015 08:00	131	1.5	-1	96.5	8	4	0	1	3
04-01-2015 09:00	301	2	-0.5	100	9	3	0	1	3
04-01-2015 10:00	528	3	-0.5	93	12	3	0	1	3
04-01-2015 11:00	727	2	-1.5	100	12	3	0	1	3

This dataset is really beneficial as there are several parameters that really make sense and the data is enough in quantity for training of the model to take place. The data is indeed a direct image of what the real world data looks like. In fact, using this dataset hints towards the thought that the same model could be trained for newer datasets of business for businesses to get a much better insight into what would be the future number of sales.

4.2 Patients Dataset [Generated]

A dataset of much relevance could not be found that revolves around how many patients are visiting a hospital. Thus we created a dataset using C++ that spans for a period of eleven years. This dataset is made for the purpose of anomaly detection. In each year, there is a time period of a month or so, where the value is really high, indicating a possible outbreak of a disease. The dataset has been made keeping in mind that the dataset resonates with real world datasets. Here is what the dataset looks like in a graphical format :



4.3 Pharma Sales Dataset

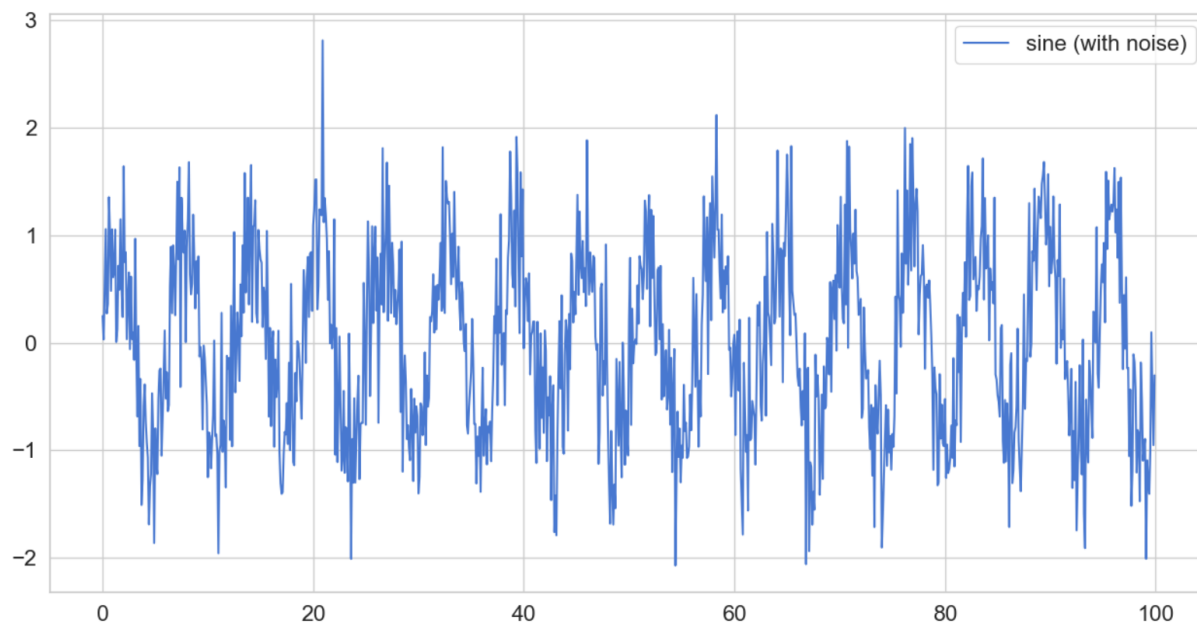
This dataset has been built from an initial dataset which included 600000 transactions collected over a period of six years (period 2014-2019). The data includes the date and time of sale, pharmaceutical drug brand name, quantity sold. All this data has been recorded from a single Pharmacy outlet. Sales data are resampled to hourly, daily, weekly, and monthly basis. The selected group of drugs from the dataset (57 drugs) is classified into the following Anatomical Therapeutic Chemical (ATC) Classification System categories : M01AB, M01AE, N02BA, N02BE/B, N05B, N05C, R03, R06

5 Development Plans and Execution

5.1 Single Step Univariate Time Series Prediction

5.1.1 Data

The dataset has been randomly generated. Normal Distribution has been used to generate random values that are added to the sine curve. This generates randomly distributed values over time that makes the job for the model non trivial. Thousand values are generated The graphical representation of the dataset seems like as shown below.



5.1.2 Data Preprocessing

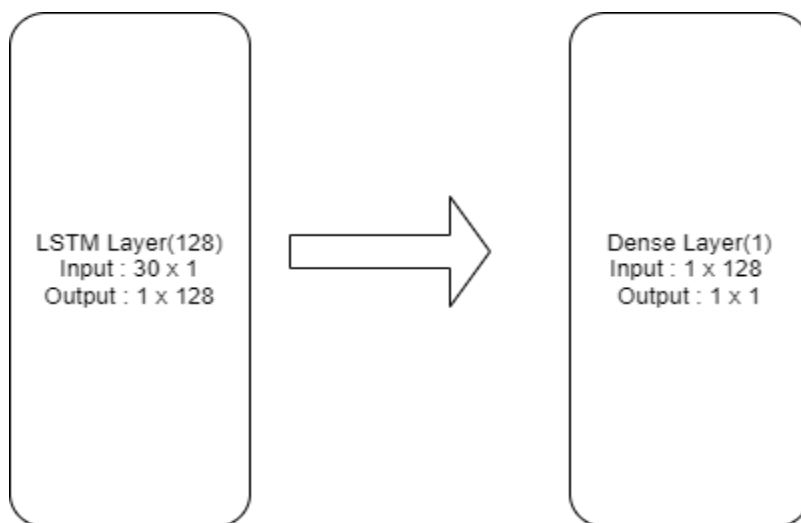
The data was divided in an 80:20 ratio. The model was trained on eighty percent of data and twenty percent of data was used for testing of the model. Next sequences were created from data. Initially the time step for sequence was set at 10. It was observed that as the number of time steps was increased there was an increase in accuracy of the model.

5.1.3 Model

We need something more robust than the classical RNN models, the reason being errors arising due to gradient calculations. The issue can be either due to vanishing gradient problems in which weights become very small or exploding gradient problems in which weights become very large. Another problem is with the memory, the older data points are forgotten and do not play much role in forecasting the predicted values. This problem is solved with the aid of gated RNNs. Therefore, we shall use LSTM[Long Short Term Memory] Neural Network for time series prediction.

We are using Keras for the ease in implementing the LSTM model. The architecture for the model is shown below. The input is the data of a particular

number of previous days and the output is the predicted value. Let us now understand the architecture of the model with the help of a self explanatory diagram.



LSTM Layer : The intuition behind LSTM develops due to the gates that it has. The cell state is based on the interactions and it can basically remove or add the information. Information thus can basically pass on without getting changed, thus the model can focus on the old values as well.

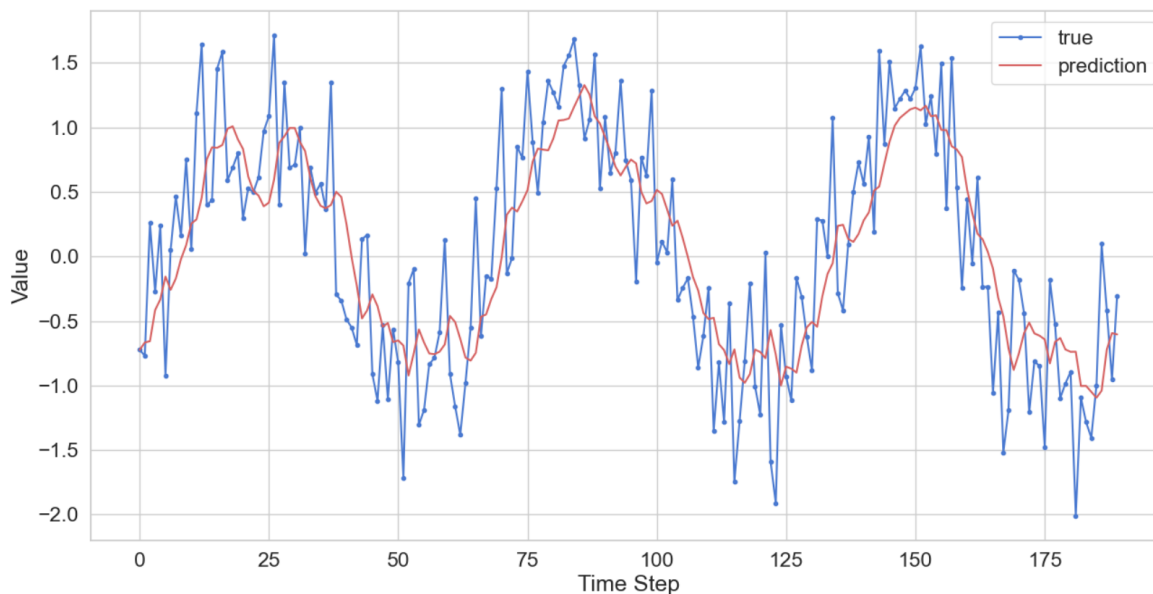
Dense Layer : One of the most commonly used layers in the neural networks. It is connected to every neuron in the previous layer in the neural network. In the background, it basically performs a matrix vector multiplication. The parameters are updated by backpropagation. It is mostly used to change the dimensions of the vector.

Adam was used as the optimizer. It is an optimization algorithm and can be used instead of stochastic gradient descent to update network weights on the basis of training data. The name Adam is derived from adaptive moment association. Adam is different from classical stochastic gradient descent(SGD). In the case of stochastic gradient descent, it maintains a single learning rate for all weight updates and maintains the same learning rate throughout the training. While in the case of Adam learning rate is maintained for each network weight and is adapted separately as the model learns. Adam can be defined as combining the advantages of two other extensions of stochastic

gradient descent: adaptive gradient algorithm and root mean square propagation .

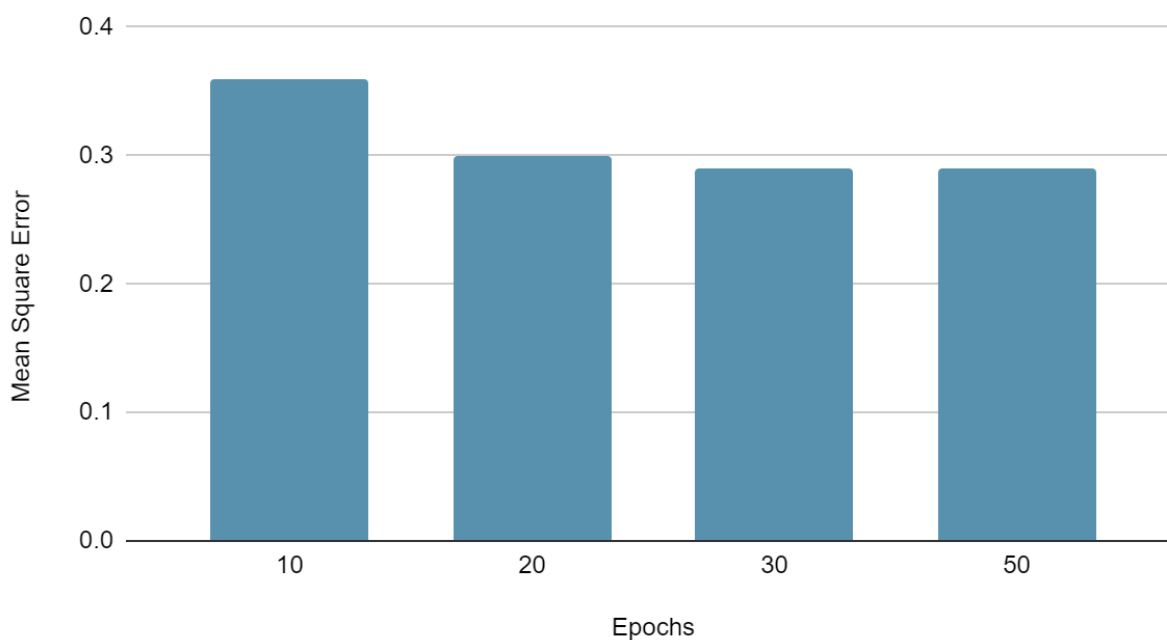
Mean Square Error has been used to calculate the error. Mean squared error is computed by taking the average of the squared differences between the predicted and actual value. The obtained result is always positive irrespective of the sign of predicted and actual value. Mean square works really well in cases of values forecasting. We have preferred mean square error because it rather punishes the higher values of error and we want the errors that are high in magnitude to reach lower errors. The intuition behind this is simply that the model gets more punished for larger mistakes.

Below is the plotted graph of predicted and true values and as it could be seen the model is doing a fairly good job. We have extensively experimented with several different combinations of the model to get the best result. The variation of Mean Square Error along with the number of epochs and number of time steps is shown as well below in tabular and graphical form as well.



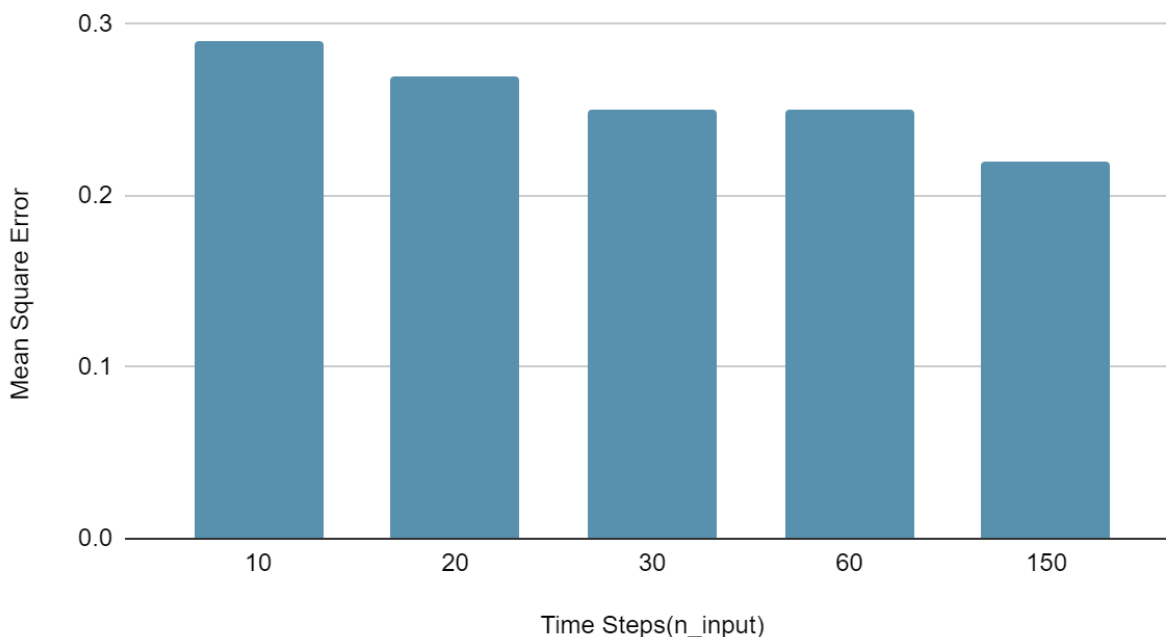
Epochs	Mean Square Error
10	0.36
20	0.30
30	0.29
50	0.29

Mean Square Error vs. Epochs



Time Steps(n_input)	Mean Square Error
10	0.29
20	0.27
30	0.25
60	0.25
150	0.22

Mean Square Error vs. Time Steps(n_input)



The crucial factor is not to overtrain the model and at the same time make a model which understands the context in a reasonable way, thus we selected the number of epochs as 30. As we increase the timesteps, since the prediction is based on more data, it is bound to improve the accuracy. However, that would make the model more complicated and difficult to train; thus there is a tradeoff between the two. As a pivot point, we have selected the timesteps as 30. Other than these, we experimented with the number of layers, the number of nodes, adding dropout, and different activation functions; after selecting the most efficient model, we got the MSE as 0.25.

5.2 Single Step Multivariate Time Series Prediction

5.2.1 Data

The data that has been used for multivariate parameters is a standard dataset that is maintained on an hourly basis. The data values are the number of cycles rented by the outlet and the data spans for a period of two years. Using the

data was really beneficial as there are several parameters that really make sense and the data is enough in quantity for training of the model to take place. The data is indeed a direct image of what the real world data looks like. In fact, using this dataset hints towards the thought that the same model could be trained for newer datasets of business for businesses to get a much better insight into what would be the future number of sales.

5.2.2 Data Preprocessing

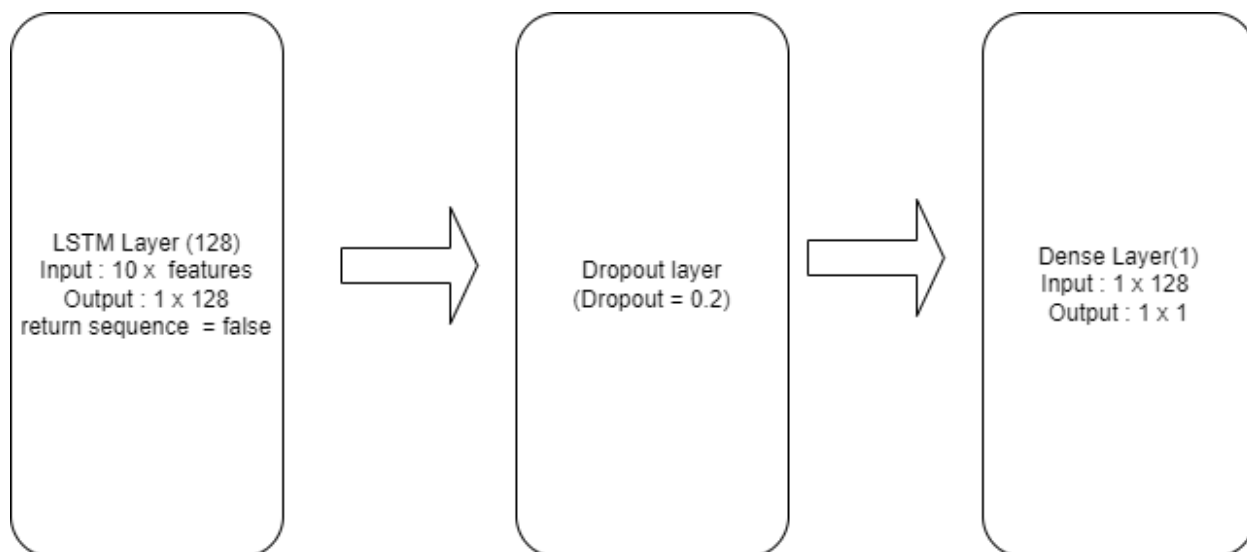
The data was divided in an 80:20 ratio. The model was trained on 80 percent of data and 20 percent of data was used for testing of the model. Next some of the features were scaled. Robust Scaler is being used for scaling the features. We are not using standard scaler here because in case of outliers being present in data it does not guarantee balanced feature scaling. Using Robust scaler outliers can be removed. Robust scaler uses statistics that are robust to outliers to scale features. It removes median and scales data in between 1st quartile and 3rd quartile. This range is also known as interquartile range. Use of interquartile range makes it robust to outliers. Formula for robust scaler is as follows

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

Next sequences were created from data. Initially the time step for sequence was set at 10. It was observed that as the number of time steps was increased there was an increase in accuracy of the model.

5.2.3 Model

We shall again use LSTM[Long Short Term Memory] Neural Network for multivariate time series prediction for the advantages already stated above. The architecture for the model is shown below with the help of a diagram.

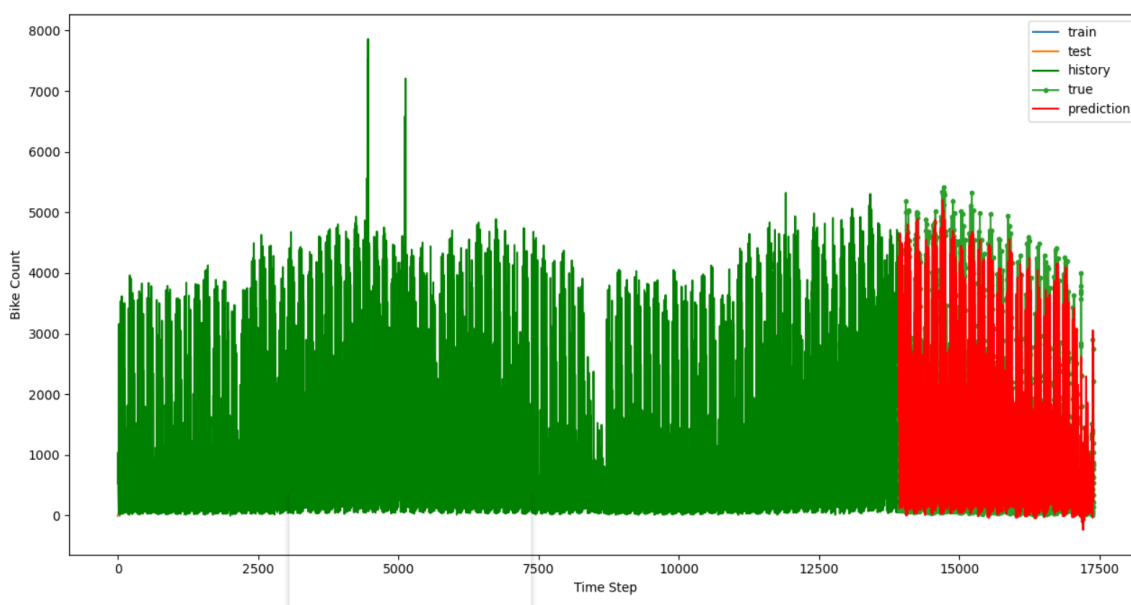


LSTM Layer : We are using a LSTM layer to gain a further deep understanding of the data.

Dropout Layer : In order to avoid overfitting of the model, we have set the dropout rate to 0.2 after carefully considering the results by experimentation.

Adam was used as the optimizer and MSE as the loss function.

After thoroughly experimenting with the layers and fine tuning the model, the least MSE value was 0.02. The predicted and true values are plotted in a graph as shown below.



5.3 Anomaly Detection

Anomaly detection is the task of identifying rare events in data being analyzed. Some of the uses of anomaly detection are bank fraud detection, tumor detection in medical imaging, etc. We will be using anomaly detection to detect abnormal events in the hospital dataset. This problem would in turn highlight the cases of anomalies in case there are any suspicious events due to which there are sudden changes in the number of patients visiting in a hospital.

5.3.1 Dataset

Hospital dataset is data about the number of patients visiting the city hospital daily over the years. Anomaly detection may help us in identifying some disease outbreak or can also help in warning the hospital to be ready with the precautions.

5.3.2 Data Preprocessing

Hospital dataset : 90 percent of data was used to train the model and remaining 10 percent to test the trained model. Next data was rescaled using standard scaler. Standard scaler makes mean equal to zero and scales data to unit variance because it follows standard normal distribution . Next sequences were created from data. Initially the time step for the sequence was set at 20.

5.3.3 LSTM Autoencoder

Autoencoders try to learn efficient representation of input using fewer parameters. The encoding is done in such a way that the dimensions of input and output are essentially the same. We have used LSTM Autoencoder for the given task. The reason being that we need to reconstruct the data and then see the difference between the predicted and actual value, and in case the difference is really large, then consider that as a case of anomaly. The steps that a basic autoencoder follow can be summarized as follows :

- ❖ Train the autoencoder model on normal training data
- ❖ Run the model on the test data and recreate the data

- ❖ The data point is termed as an anomaly if reconstruction error(difference between the actual and the predicted value) is above a certain threshold.

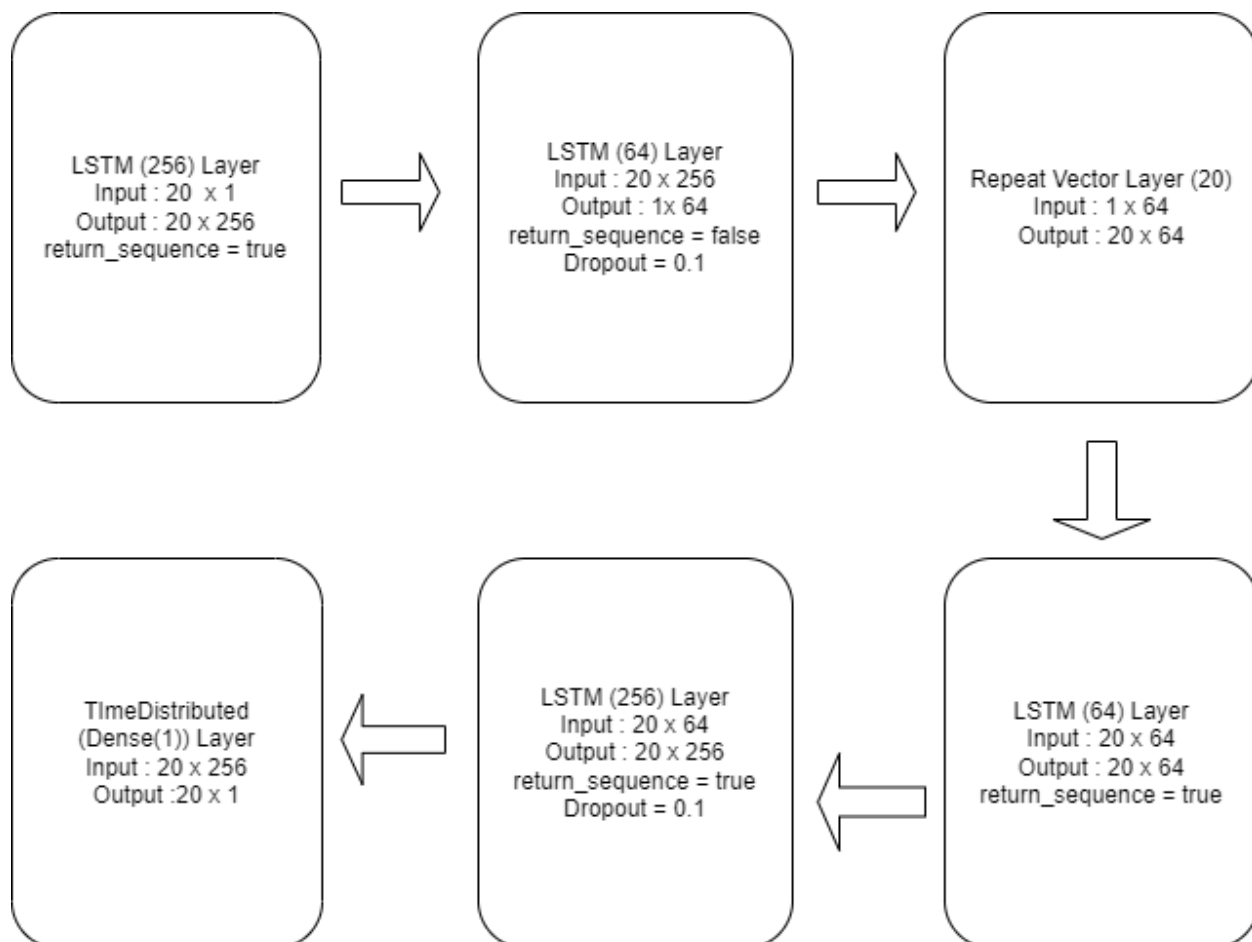
5.3.3.1 Model

The Autoencoder architecture is divided into two modules, encoder and decoder. Encoder and decoder parts consist of LSTM layers. Two Dimensional matrix is fed as input into the LSTM network. The number of cells in the LSTM layer is equal to the number of timesteps. Setting return subsequences as true in the LSTM layer makes each cell per timestep of the LSTM layer to emit signals. There is direct connection between respective timestep cells of consecutive LSTM layers in the encoder and decoder part. We are using a repeat vector layer between encoder and decoder module to duplicate the encoded vector obtained from encoder for the LSTM layer of the decoder module, thus basically this repeat vector layer acts as a bridge between the encoder and decoder part.

The Autoencoder module consists of two LSTM layers. First LSTM layer outputs N_1 features with t timesteps as return sequence was set as true while second layer outputs encoded vector of size $N_2 \times 1$ as return sequence was set as false. The repeat vector layer repeats the encoded vector timesteps and prepares a 2D array for the LSTM layer of the decoder. The task of the decoder layer is to unfold the decoding thus the layers of decoder are stacked in the reverse order to that of encoder. First LSTM layer in decoder is mirror image of second layer in encoder, similarly second LSTM layer in decoder is mirror image of first layer in encoder.

In the end, a time distributed layer is added to get the final desired output. Time distributed layer creates a vector of length equal to number of features outputted from the previous layer. As the number of features outputted by the second lstm layer of decoder is N_1 therefore the Time distributed layer creates a vector of length N_1 . The matrix multiplication between the output of the second layer of decoder and output of the Time distributed layer gives us the output of the same dimension as the input array basically [Number of timesteps] x [number of features].

The architecture of the model is described as below with the help of a diagram.



Adam was used as the optimizer while training the model.

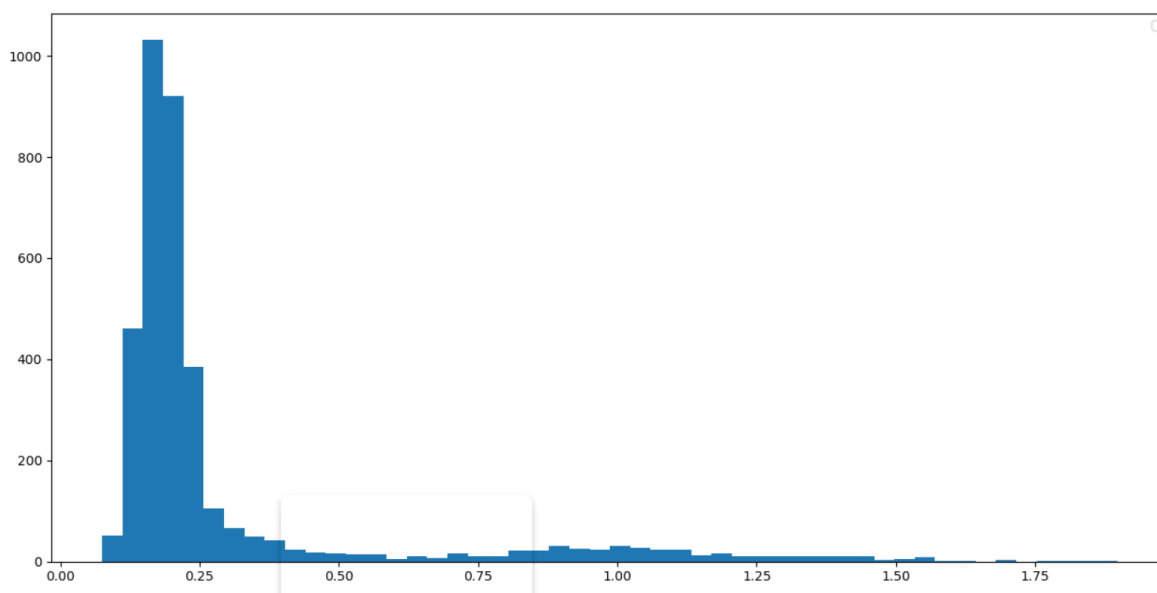
Mean absolute error was used as the loss function while training the model.

Rectified Linear Unit(ReLU) is being used as the activation function. It is a piecewise linear function that will output the input directly if positive else it will output zero.

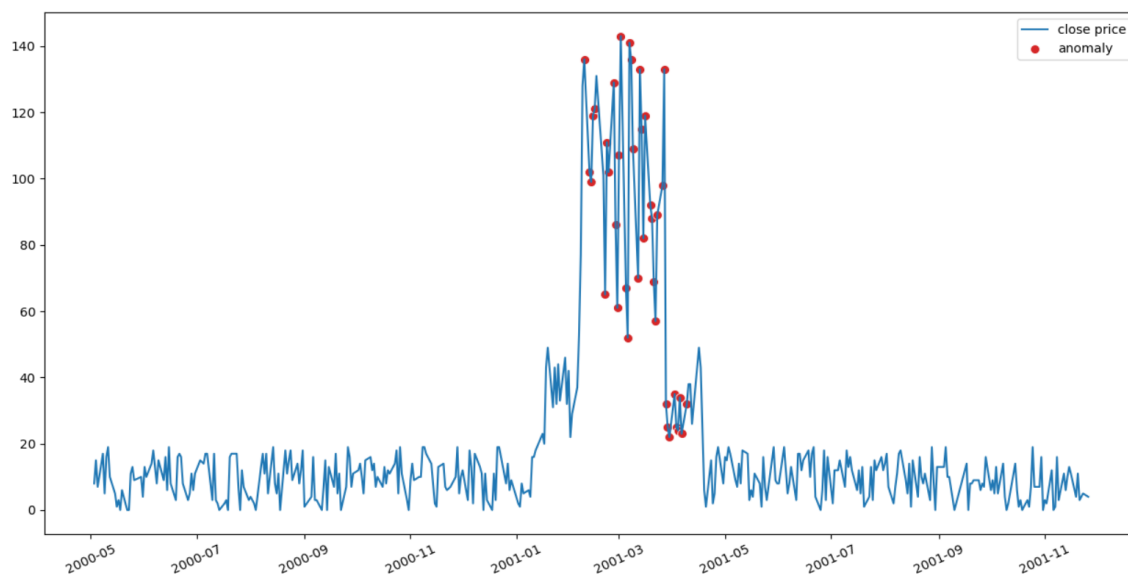
5.3.3.2 Evaluation


First of all, we calculate mean absolute error on the training data by making predictions using a trained model and plot the spread of error. After analyzing the plot we decide a certain threshold value. Next we make predictions on test data using a trained model and we calculate mean absolute error for predicted values. If the mean absolute error is more than threshold value then it is

declared as anomaly. Below is the graphical representation of the spread of error on the trained data, from here we select the threshold value; in case the reconstruction error for the test data points is more than this threshold, then the data point is an anomaly. We select the threshold value as 0.7.



Let us now take a look at the final curve with anomalies highlighted in red colour. The number of epochs for this model is 30 and the timesteps are 20.





We can easily infer from the graph that the points where there are sudden changes are all highlighted, which give an indication of anomalies that are a change in patterns of the number of patients coming to the hospital.

5.4 Multi Step Time Series Prediction

We aim to predict the sales of drugs in a pharmacy using historical data. This could help the healthcare centers in many ways such as better management of medicinal inventory, prediction of rush days in a week, better slot management of healthcare officials, possible highlights of a disease outbreak. There are multiple ways multi-step time series prediction can be done, we would first analyse different ways to do that and then analyse models that serve the purpose. We would predict the values of how many medicines would be sold in the duration of the next seven days.

5.4.1 Dataset

We are using pharma sales dataset for training and testing of models. This dataset consists of daily sales of drugs over a period of 6 years. More details have already been mentioned in the Datasets section above.

5.4.2 Data preprocessing

Pharma Sales dataset : Model was trained on ninety percent of the data and rest ten percent was used for testing of trained model. Next sequences were created from data. Initially the time step for sequence was set at fourteen.

5.4.3 Models

The models can be one of the four types :

- ❖ Direct Multi-step Forecast Strategy: Developing a separate model for each forecast time step
- ❖ Recursive Multi-step Forecast: One-step model multiple times where the prediction for the prior time step is used as an input for the future predictions
- ❖ Direct-Recursive Hybrid Strategies: This type of model can be called a

hybrid model of the above two. A different model is constructed for each time step to be predicted, and each model uses the predictions made by models for earlier values.

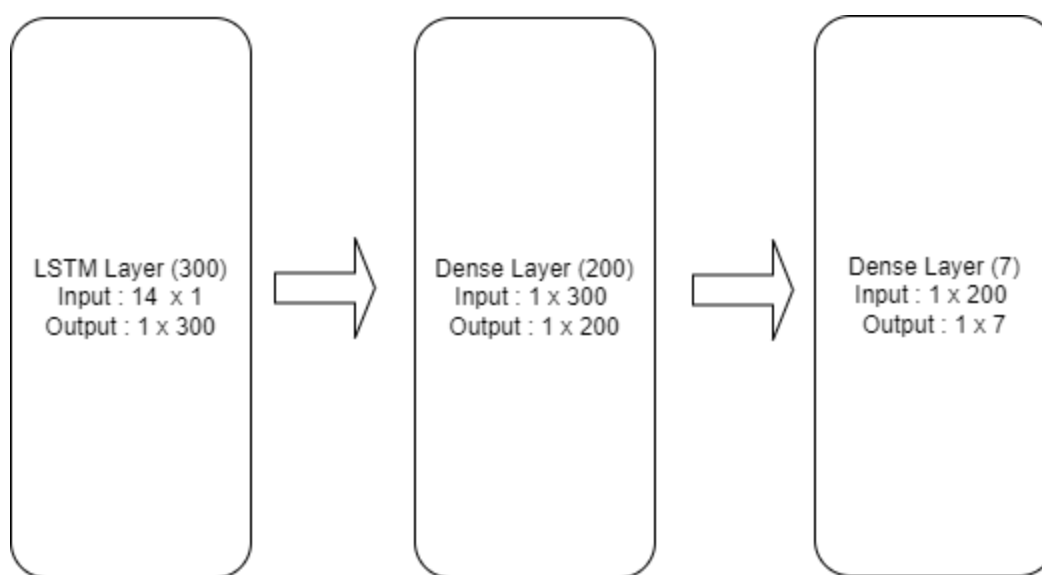
- ❖ **Multiple Output Strategy:** Involves developing one model that is capable of predicting the entire forecast sequence

We have chosen the last approach, that is, Multiple Output Strategy, so as a single model is made saving upon the computation and as we do not use the predicted values to further predict the values, there are no aggregating errors. We have worked on three models, details for which are mentioned below.

5.4.3.1 LSTM

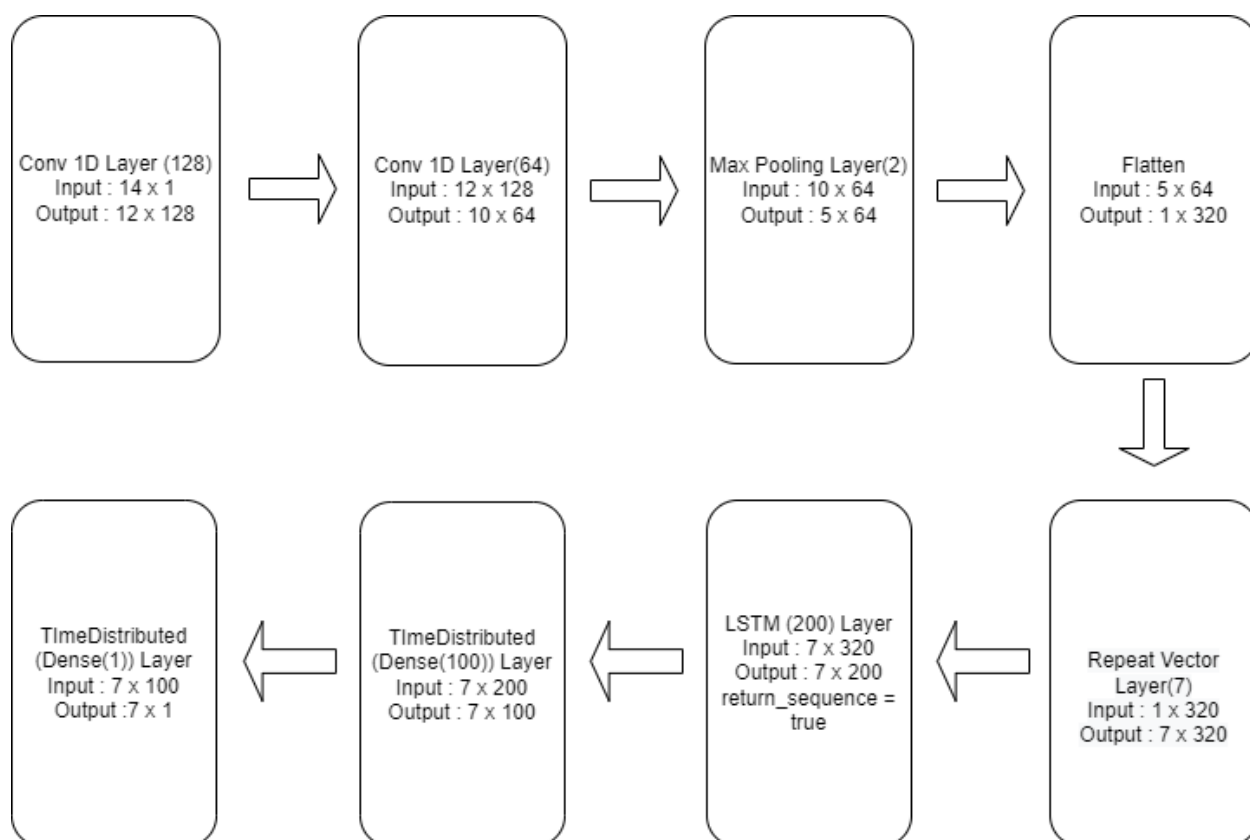
We have already discussed in much depth the benefits of LSTM. Here we are using the LSTM layer, followed by a dense layer of 100 units. Then further a dense layer which outputs a linear vector of 7 days, which are basically the prediction for the entire next week.

Mean Square Error has been used and Adam optimizer has been used. Below is a diagram that explains the architecture of the model that we have used.



5.4.3.2 CNN-LSTM

The model consists of two modules, encoder and decoder. The architecture of the model is shown below. The encoder module consists of two convolution layers and a max pooling layer. The first convolution layer reads the input and projects the result into feature maps. The second convolution layer further tries to amplify any salient feature existing in feature maps obtained from the first layer. The input sequences in the convolutional layer are read with a kernel size of three time steps. Next, the feature maps are simplified by pooling layer by keeping half of the values with the max signal. Next, this distilled feature map is flattened into vectors and is fed into a repeat vector layer. Next, the repeat vector layer repeats the obtained vector for each time step in the output sequence.



Further, this vector is fed into the LSTM layer of the decoder. Each unit of this layer outputs a value for each day of sequence as the return sequence was set to true and forms the basis for prediction of each day in the output sequence. Next a dense layer is being used to interpret each time step in the output

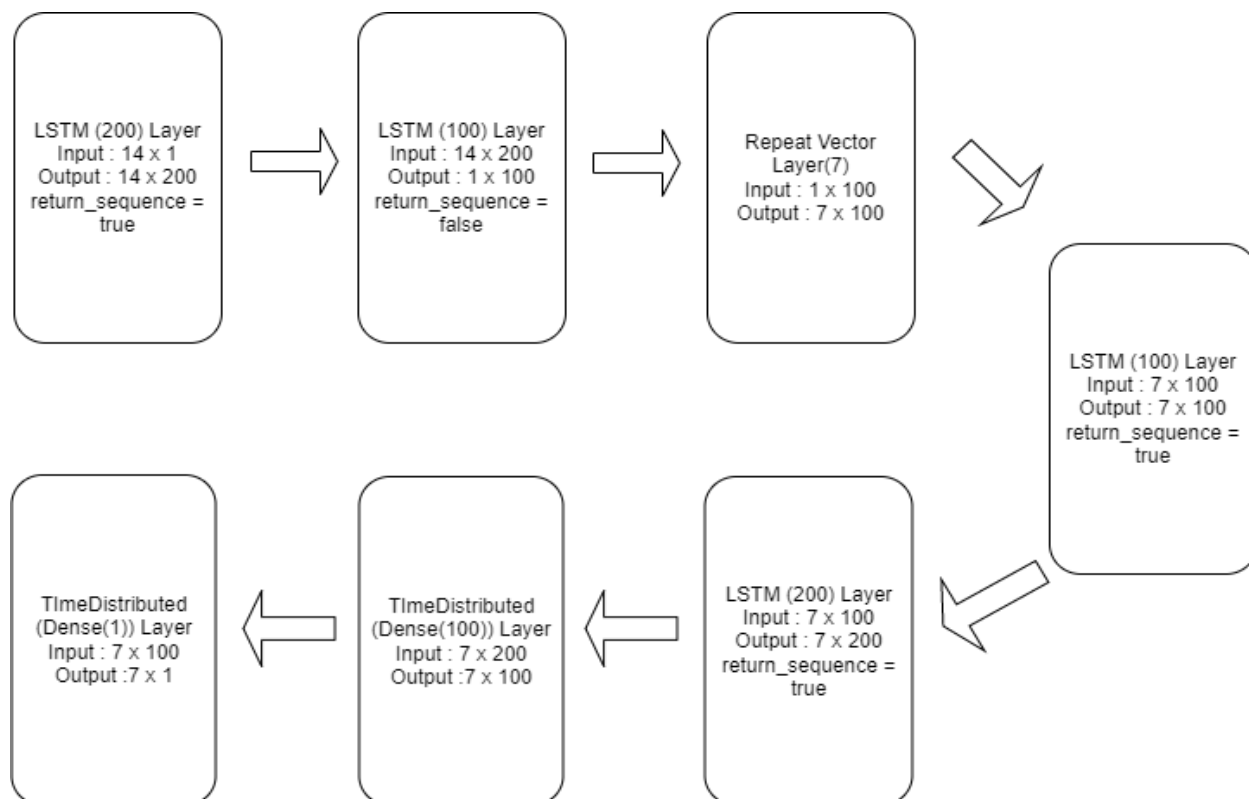
sequence. Next an output layer is used which outputs a single step in the sequence at a time. Both fully connected layer and output layer are wrapped in Time Distributed Wrapper because they will be used to process each time step provided by the decoder.

Mean Squared Error was used as loss function

Adam was used as the optimizer

5.4.3.3 LSTM Encoder-Decoder

The model consists of two modules, encoder, and decoder. The architecture of the model is shown below. Both encoder and decoder modules are composed of LSTM layers while the repeat vector layer acts as a bridge between encoder and decoder. LSTM model in the decoder is beneficial because it knows what was predicted for a prior day in the sequence and accumulates internal states while outputting the sequence. Encoder layer consisting of an LSTM layer captures features from input sequences. It will output a vector of dimension equal to the number of units in the layer. The repeat vector layer repeats the obtained vector for each time step in the output sequence.



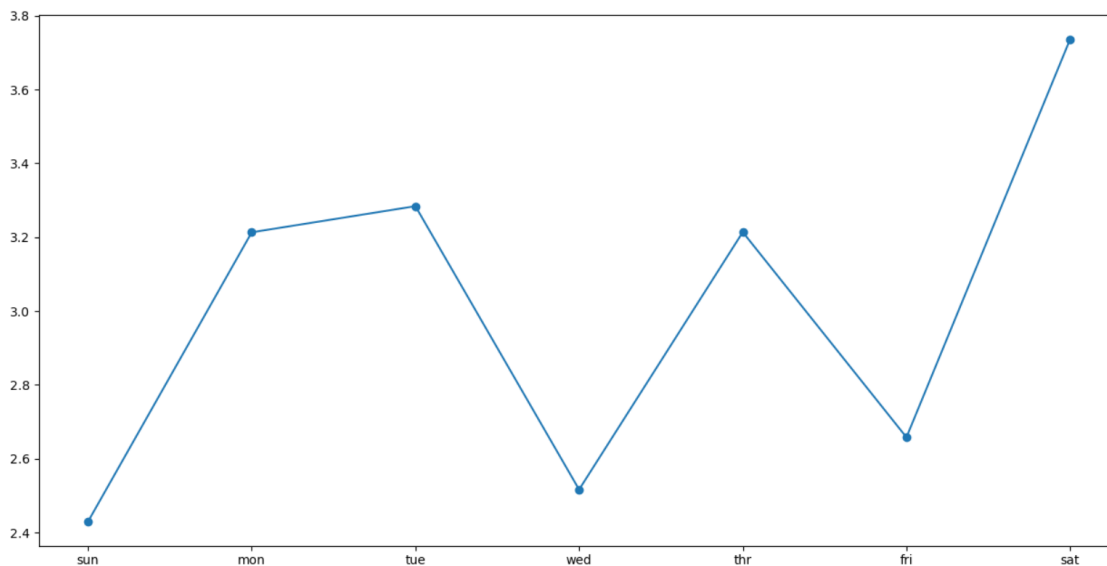
Further, this vector is fed into the LSTM layer of the decoder. Then, a dense layer is being used to interpret each time step in the output sequence. Next, an output layer is used which outputs a single step in the sequence at a time. Both fully connected layer and output layer are wrapped in Time Distributed Wrapper because they will be used to process each time step provided by the decoder.

Mean Squared Error was used as loss function

Adam was used as the optimizer

5.4.4 Evaluation

The model is evaluated on the basis of MSE calculated for each forecast day as well as overall MSE calculated. MSE is being used here because it is more punishing in case of larger mistakes. Below is the MSE value plotted over the week days, knowing the MSE trend for the week days also hints at the days for which we are better able to predict the upcoming sales.



We have duly experimented with the models on the basis of fine tuning several parameters. As a final word, we have chosen the epochs as 20 and the number of timesteps on the basis of which the prediction is made as 14.

Epochs	n_input	LSTM	LSTM + CNN	LSTM + Encoder-Decoder
10	7	2.887	2.932	2.948
10	14	2.918	2.881	3.080
10	21	2.963	2.941	2.908
20	7	2.880	2.897	2.864
20	14	2.907	2.877	2.863
20	21	2.984	2.933	2.939
30	7	2.858	2.914	2.887
30	14	2.887	2.888	2.861
30	21	2.906	2.903	2.919

We can see from the table that for the selected epochs and n_input, LSTM Encoder-Decoder works the best followed closely by the LSTM+CNN layer. The reason for this can be due to the additional layers that are present in the architecture and perhaps first the encoding and decoding understands the context in a better way.

6 Future Prospects

- ❖ The foundation has already been laid for multivariate multi step time series prediction, they can be explored in a much better depth.
- ❖ GNNs can be experimented upon after due research on the topic.
- ❖ Motif Detection

7 Techstack

- ❖ Programming languages : Python, C++

- ❖ DL libraries : Tensorflow, Keras

8 Codebase Link

Github Codebase Link :

<https://github.com/vyaskartik20/TimeSeriesPrediction>

9 Conclusion

Different types of time series including univariate and multivariate time series have been analyzed using different techniques. Anomaly detection was performed on a dataset focusing on the number of patients using LSTM autoencoder which can help hospitals to be ready in cases of anomaly detected and prevent the damage that could be caused due to sudden disease outbreak. Single-step time series prediction was performed on the pharma sales dataset using the LSTM model which could help hospitals and chemist shops better manage the inventory and number of healthcare officials required and can also help them in maintaining.

Multivariate data of the bike-sharing service of London has been used to predict the future number of sales with the help of an LSTM model. Further, multistep time series forecasting was performed on the pharma sales dataset using three different models. The models used in multistep forecasting were the LSTM model, LSTM encoder-decoder model, CNN-LSTM encoder-decoder model. All the models were duly fine-tuned according to the datasets and have been made in such a way they can be easily extrapolated to new problems.

10 Acknowledgments

We are heartily thankful to our instructor, Dr. Gaurav Harit, for providing us the necessary guidance, the needed constant support, and helping us throughout the course of the project via continuous interaction and evaluation of the course at regular intervals.

11 References

- ❖ https://www.tensorflow.org/tutorials/structured_data/time_series
- ❖ <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- ❖ <https://www.frontiersin.org/articles/10.3389/fdata.2020.00004/full>
- ❖ <https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset>
- ❖ <https://towardsdatascience.com/step-by-step-understanding-lstm-autoencoder-layers-ffab055b6352>
- ❖ <https://www.kaggle.com/milanzdravkovic/pharma-sales-data>
- ❖ <https://machinelearningmastery.com/multi-step-time-series-forecasting/>
- ❖ <https://www.hindawi.com/journals/complexity/2020/6622927/>