

Electricity Demand Prediction using Time-Series Algorithms

An Industrial Internship Report

Submitted in partial fulfillment for the award of the degree of

B.Tech.

in

Information Technology

by

SOUBHIK SINHA (19BIT0303)



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

June 2022

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled "**Electricity Demand Prediction using Time-Series Algorithms**" submitted by me to School of Information Technology & Engineering, Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech. (Information Technology)** is a record of bonafide **Industrial Internship – (ITE1902)** work carried out by me. I further declare that the work reported in this **Industrial Internship report** has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.



Place: Vellore

Signature of the Candidate

Date: 06/07/2022

SOUBHIK SINHA



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering [SITE]

CERTIFICATE

This is to certify that the Industrial Internship report entitled “**Electricity Demand Prediction using Time-Series Algorithms**” submitted by **SOUBHIK SINHA (19BIT0303)** to School of Information Technology & Engineering, Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech. (Information Technology)** is a record of bonafide **Industrial Internship – (ITE1902)** work carried out by him/her in **NTPC DADRI (NCPS)**. The **Industrial Internship** project fulfills the requirements as per the regulations of this Institute.

Examiner – Panel In-Charge
(Name and Signature)

Date

**COMPANY CERTIFICATE or EQUIVALENT APPROVED LETTER COPY TO BE ENCLOSED
HERE**



एनटीपीसी लिमिटेड

भारत सरकार का उद्यम

NTPC Limited

(A Govt. of India Enterprise)

**दादरी
DADRI**

संदर्भ संख्या / Ref. No. 08/RLI/22

दिनांक/Date

01.07.22

TO WHOM IT MAY CONCERN

This is to certify that Mr. SOUBHIK SINHA, student of Information Technology(IT), VIT University Vellore has successfully completed one month (30 days) summer training program (from 1st June 2022 to 30th June 2022) in Hybrid Mode at NTPC Dadri(NCPS).

He worked on a project titled, Electricity Demand Prediction using Time-Series Algorithms. This project was aimed to predict power demand for next day considering previous day's power consumption by beneficiary state of UP and Delhi(UT) and power generation, altogether.

During his training, he has been sincere and punctual and demonstrated his skills with an attitude of self-motivation by completing his project on schedule and effectively.

We wish him all the best for his future endeavors.



(Authorized Signatory)

ACKNOWLEDGEMENT

I am extremely thankful to **Narendra Singh sir (Main Guide), DGM (IT Dept.)** and **Manish Rathi Sir (Co-Guide), DGM (IT Dept.) NTPC Dadri**, for giving me an opportunity to work on a genuine industrial issue faced by Power-Plants today & and also guiding me throughout, provoking to learn something new and interesting. Successful completion of a challenging project requires immense help from your team members. I am thankful to my team who gave uninterrupted support. I see this chance as a significant turning point in my professional progress. In order to achieve my intended professional goals, I will make every effort to utilize newly acquired skills and information to the fullest extent feasible and to keep working to enhance them. I look forward to working with every one of you again in the future.

Sincerely,

SOUBHIK SINHA

Place: **Vellore**

Date: **06/07/2022**

TABLE OF CONTENTS

Chapter No.	Contents	Page No.
1	Abstract I. The Company II. Opportunities III. Methodology IV. Benefits to the company through this report	7-8 7 7 8 8
2	Introduction	9-10
3	Background / Pre-requisite Knowledge I. Introduction to Applied Algorithms I.I. Recurrent Neural Network I.II. Long Short-Term Memory Networks I.III. SARIMA I.IV. SARIMAX	11-14 11 11-12 12-13 13 14
4	Technology Used	14-15
5	Explaining Implementation (Codes)	15-31
6	Reflection on the Internship	32
7	Conclusion	33

CHAPTER 1

ABSTRACT

I. The Company:

Known as one of the best Indian public sector companies, **NTPC Limited**, formerly known as **National Thermal Power Corporation Limited**, is involved in the production of electricity and related operations – having headquarters situated at the Indian capital city, **New Delhi**. The primary duties of NTPC in India are the production and distribution of power to State Electricity Boards. The organization also works on consulting and turnkey project contracts for engineering, project management, construction management, and power plant operation and management. The organization has also engaged in coal mining and oil and gas exploration. With a **67,907 MW** electric power producing capacity – thus, comprising approximately **16%** of India's total power capacity and **25%** of Nations' total power generation - it is India's largest power corporation. The company has many power plants under its monitoring and control – which comprises Coal based, Hydro based, Gas based and Solar based plants – laying out across this vast nation in numerous states, and situated in very remote places (due to environmental constraints). One of the famous power plants they have is **NTPC Dadri (National Capital Power Station)**, which caters the needs of electricity the capital city, Delhi, and to other regions located inside Delhi NCR (National Capital Region) and also to other states like Gujarat and one of the most densely populated state, Uttar Pradesh. The **Dadri Station** is a 1820MW Coal based and 817MW Gas based power-plant, comprising of stages, which again comprises 6 units, together. The 2nd Stage was built with the prime purpose to provide uninterrupted electricity to the commonwealth village, on the occasion of Commonwealth Games, held in October 2010. Many foreign delegates and company representatives pay visit for gleaning the exceptional research work in ash management – how ash can be used in agriculture and also for the growth of numerous medicinal plants.

II. Opportunities:

The company has numerous scopes in electrical and mechanical fields. With the recent inculcation of technological advancements and digitization, the IT sector of the company seems booming – especially Data Analytics, Web-site and App development (as the Indian Govt. has made it compulsory to send uninterrupted and regular data of power plants for environmental assessments, consumption and generation rates and ratio, patterns for improvement and research, etc.). The data analytics section is able to analyze numerous patterns in equipment's performance through telemetry data to provide valuable insights.

III. Methodology:

This project is to provide a solution to a problem, faced by most of the power-plants today (if talking as a whole). The project talks about how to use time-series algorithms (here LSTM and RNN) to predict how much electricity has to be generated the next day. A day consists 24hrs, hence consists 96 intervals of 15 mins each. Power-plants record the data every 15 mins, on power generation and consumption of buyers (the states). These intervals are termed as “Time Blocks” or simply “Blocks”. Our task is to generate the expected power to be generated for 10 blocks of the next day. The project is built in Google-Colaboratory using Python programming language and Machine Learning modules like Keras (for importing the models) and Scikit-Learn (for data scaling).

IV. Benefits to the company through this report:

Through this report, the benefits are not at all limited for getting predicted value of power generation for next day. There is a term known as “**Overhauling**” – whose literal meaning is - **take apart (a piece of machinery or equipment) in order to examine it and repair it if necessary**. Every year, all equipments are thoroughly checked and assessed on the basis of their performance and movement. What came out is that, w.r.t fluctuations in power demand, the stations have to decrease the power generation by great factors. This sudden dip sometimes results to great damage to critical components. Using the predicted values of power generation, it will be known beforehand, how much electricity has to be generated and for that, how many components and equipments will have to work together – thus, reducing the cost of maintenance and repairing.

CHAPTER 2

INTRODUCTION

For the ever-increasing demand of power due to exponential industrial growth has led to the establishment of numerous power-plants all around the nation. But power demand today doesn't only comprise reasons with respect to the growing industries. The months of May and June have showed us the highest temperature one can ever imagine in the summers, reaching up to 50°C, that too with intense humidity – feeling like being boiled without water. The best plan to escape this natural torture of sweat and tiredness is to have a cooler or an air conditioner at your home or workplace. Nowadays, shopping malls are equipped with central ACs so that people can move around peacefully. But this is just one of numerous reasons of extreme power demand. Others may denote the power demands in the automobile industries, steel plants, manufacturing units, travel and tourism industry and what not. The demands are increasing at a faster pace than expected. India's highest electricity consumption was last recorded on 6th July, 2022 at 209,809MW. Up till last year, 7th July, 2021, saw the demand peak at 200,570MW. Furthermore, the government predicts that once the monsoon season is through in most areas, the peak power demand might reach 215,000 MW. India is now one of the top-most power consumers in the world – the credits go to booming economy and industrialization. But wait! we just mentioned how we shall be consuming in the initial days of monsoon – that means the power-plants have to produce that much power altogether – problem solved! But the reality is, it won't happen. Every power-plant has their own maximum capacity to generate electricity – they cannot outperform. Also, establishment of a power plant in such a short period of time is something next to impossible.

If we breakdown our power needs on daily basis, we can discover numerous patterns and fluctuations – from minor to major. Let's look into the statistics of some major regions of the nation - The national capital's second-highest ever recorded peak power consumption was reached on 7th July, 2022 when the peak demand for energy reached 7.02 gigawatts (GW). On July 2, 2019, Delhi's peak electricity consumption of 7.40 GW was reported. For the third time, Delhi's electricity consumption has surpassed 7 GW. Peak power consumption in Uttar Pradesh was 25,436 megawatts in May 2022, the same as the previous month. West Bengal has the highest peak power consumption. West Bengal accounted for 68,091 megawatts of the world's 68,091 megawatts of peak electricity consumption as of May 2022. One hundred percent of it, or the top 2, is accounted for by West Bengal. In May 2022, it was calculated that the world's peak power consumption will be 76,996 megawatts. Mumbai's electricity consumption peaked on 1st July, 2022 at a record 3,850 MW, up 10% from the same day last year when it didn't get over 3,100 MW. Chennai, known as the “Detroit of India” for having numerous automobile manufacturing units established, has grabbed a huge value of 3,700 MW. On 4th May, 2022, the metropolitan city reached the max. demand of 3,716 MW and the max. consumption stands at a whopping 81.33 million units (81.33 million KWh). Such diverse power demands and consumption makes it

difficult for power-plants to deliver the buyers (state govt.) needs. In this report, we will be talking specifically about the state of Uttar Pradesh and the National Capital, New Delhi and the Delhi Union Territory (UT).

The power generated, as mentioned earlier, is not constant. If a powerplant has a capacity of, say, 1820 MW – then that is its maximum output, which can be lowered as per the requirement. But what requirement? Every day, the power stations have to send report every 15 mins of a day to the govt., denoting how much power they are generating. The state govt. accordingly plans to request for power demand for the next day. In this way, the data might be getting generated, but there is no analysis and prediction for future day's power generation. There are 96 blocks (or intervals) of 15 mins each in a day – thus that data generated is according to the same. Hence, they can be considered as “Time-Series Dataset”. For its analysis, we will be needing Time-Series Algorithms.

CHAPTER 3

BACKGROUND / PRE-REQUISITE KNOWLEDGE

I. Introduction to applied Algorithms

As the main motive of the project is to find the power generation requirement for the initial 10 blocks (15 mins each) of the next day. The reason for limiting ourselves till the next day and that too for only 10 blocks, is to avoid absurdness in the values predicted. As it will be an analysis of Time-series dataset, we will be using algorithms from the same class – namely, LSTM (Long Short-Term Memory) and RNN (Recurrent Neural Network). NOTE: LSTM is also a type of RNN with little modifications. In addition to the chosen algorithms, I will also would like to give explanation on 2 other algorithms (SARIMA and SARIMAX) on which the company's research & development section have done some assessment and concluded that LSTM and RNN were to be chosen. Without wasting more time, let us dive into the working of all the algorithms (chosen and not chosen both) –

I.I. Recurrent Neural Network

Recurrent neural networks (RNNs) are a form of neural network in which the results of one step are fed into the next phase's computations. Traditional neural networks have inputs and outputs that are independent of one another, but there is a requirement to remember the previous words in situations when it is necessary to anticipate the next word in a phrase. As a result, RNN was developed, which utilized a Hidden Layer to resolve this problem. The Hidden state, which retains some information about a sequence, is the primary and most significant characteristic of RNNs.

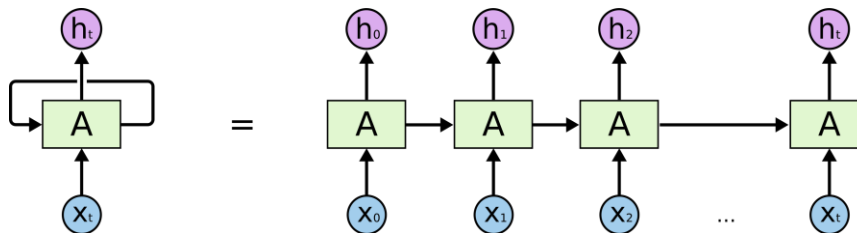


Fig. 1: Diagrammatic working of RNN

RNNs have a "memory" that retains all data related to calculations. It executes the same action on all of the inputs or hidden layers to generate the output, using the same settings for each input. In

contrast to other neural networks, this minimizes the complexity of the parameter set.

Let us have a look how algorithm actually work. RNNs keep all the information needed for computations in a "memory." It uses the same parameters for each input and performs the same action on each input or hidden layer to produce the output. This reduces the complexity of the parameter set compared to other neural networks. By giving all of the layers the identical weights and biases, RNN transforms independent activations into dependent activations, decreasing the complexity of raising parameters and remembers each previous output by using each output as an input to the following hidden layer. Thus, all three layers may be combined into a single recurrent layer such that the weights and bias of all the hidden levels are the identical.

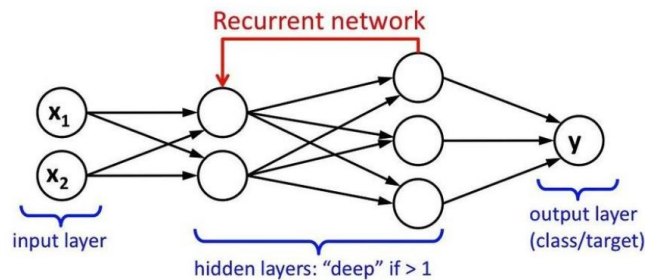


Fig. 2 : The hidden layer of RNN

An RNN retains every piece of knowledge throughout time. Only the ability to recall past inputs makes it helpful for time series prediction. Long Short-Term Memory is the term for this. Convolutional layers and recurrent neural networks are even combined to increase the effective pixel neighborhood.

I.II. Long Short-Term Memory Networks

As mentioned earlier, LSTM is a modified & advanced version of RNN. The output from the previous phase is sent into the current step of an RNN as input. It addressed the issue of long-term RNN dependency, in which the RNN can predict words from current data but cannot predict words held in long-term memory. RNN's performance becomes less effective as the gap length rises. By default, LSTM may save the data for a very long time. It is utilized for time-series data processing, forecasting, and classification. Four neural networks, often known as cells, and distinct memory building elements make up the chain structure of the LSTM. Cells and gates both play a role in memory modification and information retention. There are 3 gates:

- A. Forget Gate: The forget gate purges the data that is no longer relevant in the cell state.
- B. Input Gate: The input gate updates the cell state with pertinent information.
- C. Output Gate: The output gate's job is to take meaningful information out of the current cell

state and deliver it as output.

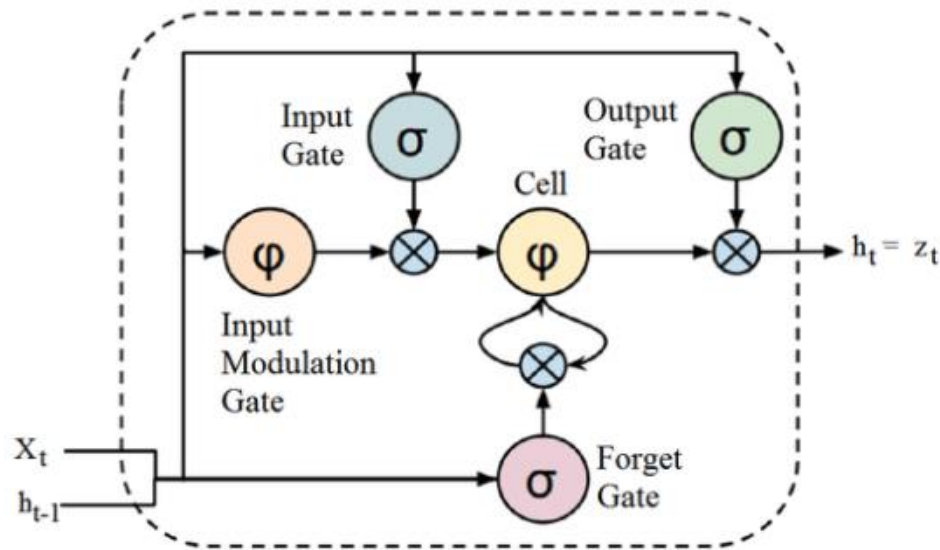


Fig 3: LSTM model

The Internal Cell State is likewise sent forward along with the Hidden State in a Long Short Term Memory Network, which is the only way it differs from a Recurrent Neural Network in terms of fundamental workflow.

I.III. SARIMA (Seasonal Autoregressive Integrated Moving Average)

One of the most popular forecasting techniques for univariate time series data forecasting is Autoregressive Integrated Moving Average, or ARIMA - which can be broken down to 3 components namely – AR (Autoregressive Component), I and MA (Moving Average). The number of lagged series we employ is determined by the p parameter, which represents the autoregressive component of the ARIMA model as $AR(p)$. The Moving Average model is $MA(q)$, where q is the quantity of lag components in the forecasting error.

Although the ARIMA model is excellent, adding seasonality and exogenous factors can have a significant impact. We must employ a different model since the ARIMA model presumes that the time series is steady. SARIMA is an ARIMA extension that enables the direct modelling of the seasonal component of the series. In this, there is an additional set of autoregressive and moving average components. SARIMA models allow for both seasonal frequency and non-seasonal frequency differences in data. Automatic parameter search frameworks can make it simpler to determine which parameters are ideal.

I.IV. SARIMAX (Seasonal Autoregressive Integrated Moving Average with eXogenous factors)

An upgraded version of the ARIMA model is called SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with eXogenous components). We might state that SARIMAX is a model with a seasonal equivalent to SARIMA. It can also manage impacts from outside sources. This aspect of the model is unique compared to other models.

CHAPTER 4

TECHNOLOGY USED

Now that it is decided to work with which algorithms, it's time to discuss the platforms and other technologies, used for creating the project.

1. **Python:** The reason for choosing Python programming language is because of its large community support and numerous libraries for Machine Learning, Deep Learning and Data Analytics. In this project I have used Keras (for importing ML models of LSTM and RNN) and Scikit-Learn (for data scaling using MinMaxScaler). It has a support for connecting external sources like Google Drive – to access files related to analysis directly from the cloud. It offers libraries like yfinance (yahoo finance) for stock market analytics – one of the trending activities in the Data Science. One can even build automation related programs to carry out repetitive stuff.
2. **Python Notebook:** The python programs can be stored in two types of file formats: **.py** and **.ipynb**. The later is considered as a python notebook. There are many platforms which provide support for operating Python notebooks – namely , Visual Studio Code, Spyder , Jupyter Lab. The Notebook provides the feature of Markdown cells , which seems very handy when doing explanation.
3. **Google Colab :** In my opinion this is one of best innovation Google has ever produced especially for AI/ML developers and Data analysts. Although IBM Watson Studio and Microsoft Azure Cloud services came way before it was launched, but they come at a price after a certain point of free use. Google Colab being completely free has provided more than enough features to play with. For a free account it provides 12GB of RAM ,

which is enough to deploy ML models (as intricate as Neural Networks) , with the support of GPUs. There is an option for Colab plus , which includes some premium features like running the notebook in the background even after closing it. Google Colab has an excellent feature of sharing the notebook , either through Open Link or sharing with them as collaborators of the project – something which helped me in this project.

CHAPTER 5

EXPLAINING IMPLEMENTATION (CODE)

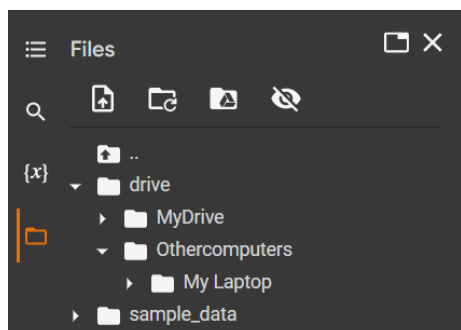
Enough we saw Trailer and explanation , now it's time for some action. The code below will be explained step-by-step in an elaborated and delineating fashion –

1. For importing data , we either save the dataset as a **.csv (comma separated values)** file in our local system or in a Google Drive. Here , I have opted the later option. Hence , I have imported **google.colab** module which directly connects the notebook to the drive (authentication needed for connecting to the drive)

```
[ ] 1 from google.colab import drive
    2 drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.

2. Now that we have connected our drive with the notebook , we will be able to see the contents of the drive from the Colab itself.



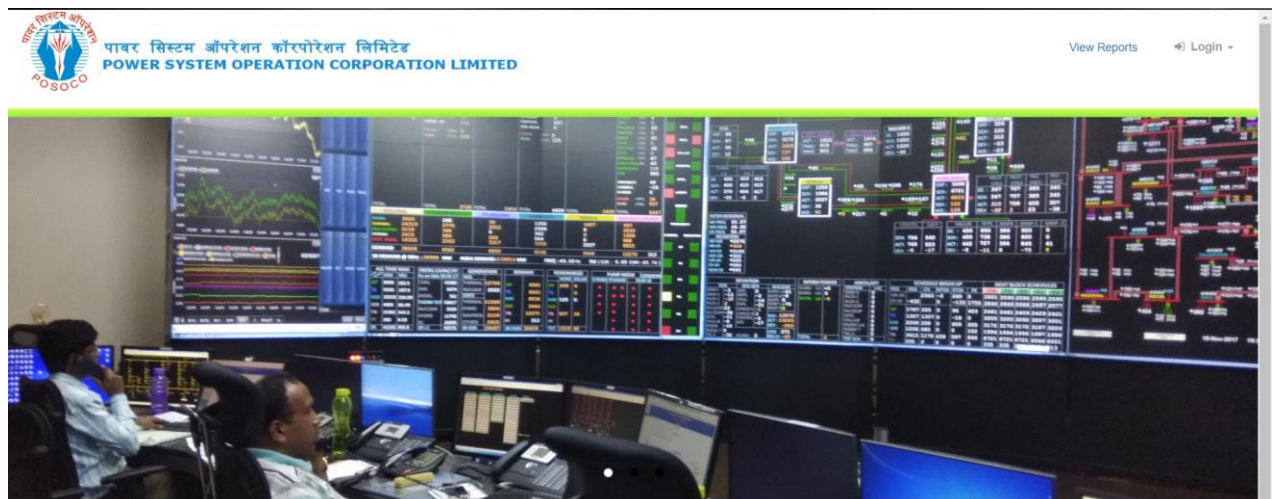
3. It's time to import all the essential modules needed for the analysis. The importing of

other modules like for data scaling and model import will be done later. These initial modules will be necessary for data manipulation and data conversion in various formats – which shall be helpful for models to ingest the data.

```
✓ 0s 1 # Importing Essential Modules / Libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 import seaborn as sns
```

4. Now that we have imported the libraries , it's time to import the dataset. As mentioned earlier , the dataset will be taken from the Google Drive. But wait ! where this data came from ? As NTPC Ltd. is a PSU (Public Sector Undertaking) , the power generation data will be sent to a government body named , **POSOCO Ltd. (Power System Operation Corporation Limited)** , and stored on their servers and is open for accessing to the general public (**LINK to POSOCO Ltd. :**

<https://wbse.srlde.in/Account/Login?ReturnUrl=%2f>). Below is the website snap -



The below snap is the webpage for the source of data

Time Block	Time Desc	ABCREPL_BHDL2	AvSusRJPPL_BKN	NSNTPC_FT61	RSUPL_FT62	Grand Total
29	07:00-07:15	20.51	34.75	12.42	26.64	94.32
30	07:15-07:30	25.60	38.00	24.72	35.63	123.95
31	07:30-07:45	30.69	39.00	35.18	45.36	150.23
32	07:45-08:00	35.78	40.00	47.85	53.09	176.72
33	08:00-08:15	40.79	45.00	80.00	45.00	210.79
34	08:15-08:30	45.79	52.00	90.00	45.00	232.79
35	08:30-08:45	50.80	65.00	100.00	45.00	260.80
36	08:45-09:00	55.81	78.00	125.80	45.00	304.61
37	09:00-09:15	58.89	87.99	130.80	45.00	322.68
38	09:15-09:30	61.97	99.00	140.90	45.00	346.87
39	09:30-09:45	65.04	111.01	150.80	45.00	371.85
40	09:45-10:00	68.12	123.00	160.80	45.00	396.92
41	10:00-10:15	72.84	130.00	170.80	120.00	493.64
42	10:15-10:30	77.56	134.01	180.70	186.39	578.66
43	10:30-10:45	82.28	145.00	213.97	195.21	636.46
44	10:45-11:00	87.00	153.00	225.69	237.65	703.34
45	11:00-11:15	88.50	155.00	244.40	246.29	734.19
46	11:15-11:30	90.01	156.00	244.40	253.96	744.37
47	11:30-11:45	91.57	160.00	244.40	258.74	754.66

- Now that we know the source , the dataset will simply be downloaded and stored in the google drive as a **.csv** file format. Pandas library will be used to import the dataset and will be converted to a Pandas data frame (NOTE : Full path from drive has to be mentioned when importing)

```

1 # Importing Dataset
2
3 df_power = pd.read_csv("drive/Othercomputers/My Laptop/NTPC Internship Project
4 df_power

```

	TIME BLOCK	DATE	TIME DESC	DADRIT2	DELHI_UT	UTTARPRADESH_STATE
0	1	01-03-2022	00:00-00:15	0.0	13.95	184.08
1	2	01-03-2022	00:15-00:30	0.0	13.01	177.06
2	3	01-03-2022	00:30-00:45	0.0	12.34	171.66
3	4	01-03-2022	00:45-01:00	0.0	11.67	168.28
4	5	01-03-2022	01:00-01:15	0.0	10.96	163.82
...
91	92	01-03-2022	22:45-23:00	0.0	17.29	312.96
92	93	01-03-2022	23:00-23:15	0.0	16.67	308.00
93	94	01-03-2022	23:15-23:30	0.0	16.05	296.86
94	95	01-03-2022	23:30-23:45	0.0	15.43	281.36
95	96	01-03-2022	23:45-24:00	0.0	14.80	266.66

96 rows x 6 columns

6. We would now like to see what are the datatype of each of the attributes for the dataset imported.

```
1 # let us see the General information of the dataset
2
3 df_power.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96 entries, 0 to 95
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  -
0   TIME_BLOCK            96 non-null    int64
1   DATE                  96 non-null    object
2   TIME_DESC             96 non-null    object
3   DADRIT2               96 non-null    float64
4   DELHI_UT              96 non-null    float64
5   UTTARPRADESH_STATE    96 non-null    float64
dtypes: float64(3), int64(1), object(2)
memory usage: 4.6+ KB
```

7. Splitting the dataset in testing and training components has to be done for model training and testing. Here , I am splitting the dataset in the ration of 75:25 in training and testing components , respectively.

```
1 # Now Let us try to predict the power to be generated
2 # We create a new dataframe with all the required I/P attributes
3
4 data = df_power.filter(['DADRIT2' , 'DELHI_UT' , 'UTTARPRADESH_STATE'])
5
6 # Dataframe to Numpy Array Conversion
7 dataset = data.values
8
9 # The model should be knowing - how many rows we are taking
10 # for training (Here , the split will be 0.75 and 0.25)
11 training_data_len = int(np.ceil(len(dataset) * (0.75)))
12 training_data_len
```

```
72
```

8. Data has to be scaled to the lowest terms. WHY ? Because models perform much better when ingested with smaller values than the ingestion of larger values. Here , the data is scaled between 0 and 1. The module used here is **sklearn** (Scikit Learn). The scaler used here is **MinMaxScaler**.

```
[ ] 1  # Data Scaling
    2
    3  from sklearn.preprocessing import MinMaxScaler
    4
    5  scaler_x = MinMaxScaler(feature_range = (0,1))
    6  scaler_y = MinMaxScaler(feature_range = (0,1))
    7  scaled_data_x = scaler_x.fit_transform(dataset)
    8
    9  df_power_y = df_power[int(training_data_len) : ]
   10  scaled_data_y = scaler_y.fit_transform(df_power_y.filter(['DADRIT2']).values)
   11
   12  print(scaled_data_x[0:5])
   13  print()
   14  print(scaled_data_y[0:5])
```



```
[[0.      0.17378781 0.40739538]
 [0.      0.1512242  0.38839262]
 [0.      0.13514162 0.37377511]
 [0.      0.11905905 0.36462563]
 [0.      0.10201632 0.35255265]]

[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

9. Hault ! Till now it was good , but here I should tell you something about the algorithms LSTM and RNN. Here , is term called “Time-steps”. We know that LSTM and RNN take inputs , store them in their memory and give the output. Here , the algorithms will be going through numerous such inputs , try to memorize them and will be presented with an output. Thus , I am going to take 10 timesteps , i.e. , after reading through 10 rows of data as input , it will then be presented with the output (which will be 11th row of data) – that’s how they learn. Above that , LSTM takes 3D data – thus , I have to convert (reshape) the data as well. Below is how the training data is created (considering the timesteps).

```

1 # Training dataset creation and split (x_train , y_train)
2
3 train_data = scaled_data_x[0:int(training_data_len), :]
4
5 x_train = []
6 y_train = []
7
8 for i in range(10, len(train_data)):
9     x_train.append(train_data[i-10:i, :])
10    y_train.append(train_data[i,1])
11    if i<=111:
12        print(x_train)
13        print(y_train)
14        print()
15
16 # We have to convert x_train and y_train into Numpy Arrays
17 x_train , y_train = np.array(x_train) , np.array(y_train)

```

```

[0.196      , 0.29692751, 0.08288669],
[0.284      , 0.3106097 , 0.12305777],
[0.356      , 0.32333173, 0.18030967],
[0.42       , 0.33605377, 0.23531482]], array([[0.        , 0.27196351, 0.6
[0.        , 0.27292367, 0.        ],
[0.024      , 0.27868459, 0.00530561],
[0.064      , 0.28468555, 0.01754101],
[0.12       , 0.29068651, 0.04601808],
[0.196      , 0.29692751, 0.08288669],
[0.284      , 0.3106097 , 0.12305777],
[0.356      , 0.32333173, 0.18030967],
[0.42       , 0.33605377, 0.23531482],
[0.536      , 0.349976   , 0.28907477]], array([[0.        , 0.27292367, 0.
[0.024      , 0.27868459, 0.00530561],
[0.064      , 0.28468555, 0.01754101],
[0.12       , 0.29068651, 0.04601808],
[0.196      , 0.29692751, 0.08288669],
[0.284      , 0.3106097 , 0.12305777],
[0.356      , 0.32333173, 0.18030967],
[0.42       , 0.33605377, 0.23531482],
[0.536      , 0.349976   , 0.28907477],
[0.62       , 0.35837734, 0.34272644]], array([[0.024      , 0.27868459, 0.6
[0.064      , 0.28468555, 0.01754101],
[0.12       , 0.29068651, 0.04601808],
[0.196      , 0.29692751, 0.08288669],
[0.284      , 0.3106097 , 0.12305777],
[0.356      , 0.32333173, 0.18030967],
[0.42       , 0.33605377, 0.23531482],
[0.536      , 0.349976   , 0.28907477],
[0.62       , 0.35837734, 0.34272644],
[0.672      , 0.36917907, 0.39540361]], array([[0.064      , 0.28468555, 0.6

```

10. Now I will create the testing data (considering the timesteps). Remember the split – 75:25.

```
1  # Creating Testing data
2  test_data = scaled_data_x[int(training_data_len) - 10 : , :]
3
4  # Creating x_test and y_test
5  x_test = []
6  y_test = dataset[int(training_data_len) : , 1]
7  for i in range(10, len(test_data)):
8      x_test.append(test_data[i-10:i , :])
9
10 # Converting to Numpy Array
11 x_test = np.array(x_test)
```

11. Now it's the time for CLIMAX ! Let us import the models of LSTM and RNN from Keras module. Here , when feeding the training data , we shall have to make sure to reshape the data as if we are ingesting data for 3 attributes (DADRIT2 , DELHI_UT and UTTARPRADESH_STATE) into the model. I am taking 300 epochs (1 epoch means 1 time the complete data will pass through the model , subsequent passing will try to increase the accuracy and precision)

```
1  from keras.models import Sequential
2  from keras.layers import Dense , LSTM
3
4  # Model Building
5  model_lstm = Sequential()
6
7  model_lstm.add(LSTM(64, return_sequences=True, input_shape = (x_train.shape[1] , 3)))
8  model_lstm.add(LSTM(64, return_sequences= False))
9  model_lstm.add(Dense(32))
10 model_lstm.add(Dense(1))
11
12 # Model Compilation
13 model_lstm.compile(optimizer='adam', loss='mean_squared_error')
14
15 # Model Training
16 # Then fit into the model
17 model_lstm.fit(x_train, y_train, batch_size=20, epochs=300)
```

```

Epoch 280/300
4/4 [=====] - 0s 11ms/step - 1 ↑ ↓ ↻ 🗨 ⚙ 📄 🗑 ⋮
Epoch 287/300
4/4 [=====] - 0s 12ms/step - loss: 0.0034
Epoch 288/300
4/4 [=====] - 0s 13ms/step - loss: 0.0026
Epoch 289/300
4/4 [=====] - 0s 12ms/step - loss: 0.0031
Epoch 290/300
4/4 [=====] - 0s 12ms/step - loss: 0.0027
Epoch 291/300
4/4 [=====] - 0s 11ms/step - loss: 0.0027
Epoch 292/300
4/4 [=====] - 0s 12ms/step - loss: 0.0029
Epoch 293/300
4/4 [=====] - 0s 15ms/step - loss: 0.0027
Epoch 294/300
4/4 [=====] - 0s 11ms/step - loss: 0.0025
Epoch 295/300
4/4 [=====] - 0s 15ms/step - loss: 0.0029
Epoch 296/300
4/4 [=====] - 0s 12ms/step - loss: 0.0030
Epoch 297/300
4/4 [=====] - 0s 11ms/step - loss: 0.0029
Epoch 298/300
4/4 [=====] - 0s 12ms/step - loss: 0.0031
Epoch 299/300
4/4 [=====] - 0s 12ms/step - loss: 0.0032
Epoch 300/300
4/4 [=====] - 0s 14ms/step - loss: 0.0030
<keras.callbacks.History at 0x7f9189b11e90>

```

```

1  # NOW LET US CREATE THE RNN MODEL 🌟
2
3  # Importing essential libraries
4  from keras.models import Sequential
5  from keras.layers import Dense
6  from keras.layers import SimpleRNN
7  from keras.layers import Dropout
8
9  # RNN Initialization
10 model_rnn = Sequential()
11
12 # adding first RNN layer and dropout regularization
13 model_rnn.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True, input_shape = (x_train.shape[1], 3)))
14 model_rnn.add(Dropout(0.2))
15
16 # adding second RNN layer and dropout regularization
17 model_rnn.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True))
18 model_rnn.add(Dropout(0.2))
19
20 # adding third RNN layer and dropout regularization
21 model_rnn.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = False))
22 model_rnn.add(Dropout(0.2))
23
24 # adding the output layer
25 model_rnn.add(Dense(units = 1))
26
27 # compiling RNN
28
29 model_rnn.compile(optimizer = "adam", loss = "mean_squared_error")
30
31 # fitting the RNN
32 model_rnn.fit(x_train, y_train, epochs = 300, batch_size = 32)

```

```
Epoch 285/300
2/2 [=====] - 0s 13ms/step - loss: 0.0234
Epoch 286/300
2/2 [=====] - 0s 12ms/step - loss: 0.0190
Epoch 287/300
2/2 [=====] - 0s 12ms/step - loss: 0.0116
Epoch 288/300
2/2 [=====] - 0s 13ms/step - loss: 0.0165
Epoch 289/300
2/2 [=====] - 0s 16ms/step - loss: 0.0190
Epoch 290/300
2/2 [=====] - 0s 13ms/step - loss: 0.0255
Epoch 291/300
2/2 [=====] - 0s 13ms/step - loss: 0.0156
Epoch 292/300
2/2 [=====] - 0s 12ms/step - loss: 0.0200
Epoch 293/300
2/2 [=====] - 0s 13ms/step - loss: 0.0128
Epoch 294/300
2/2 [=====] - 0s 13ms/step - loss: 0.0148
Epoch 295/300
2/2 [=====] - 0s 13ms/step - loss: 0.0181
Epoch 296/300
2/2 [=====] - 0s 22ms/step - loss: 0.0148
Epoch 297/300
2/2 [=====] - 0s 14ms/step - loss: 0.0155
Epoch 298/300
2/2 [=====] - 0s 14ms/step - loss: 0.0180
Epoch 299/300
2/2 [=====] - 0s 14ms/step - loss: 0.0225
Epoch 300/300
2/2 [=====] - 0s 16ms/step - loss: 0.0147
<keras.callbacks.History at 0x7f91861cfed0>
```

12. Now , I will obtain the prediction values w.r.t. the `x_test` values. After that , the values of prediction will be inverted back to their original form of values (just like in DADRIT2 – the target attribute , as LSTM and RNN product only one output – either in a numpy array or a single element)

```
1  # Now let us acquire the predicted close price values
2  # with respect to the models created
3
4  # LSTM
5  predictions_lstm = model_lstm.predict(x_test)
6  predictions_lstm = scaler_y.inverse_transform(predictions_lstm)
7
8
9  # RNN
10 predictions_rnn = model_rnn.predict(x_test)
11 predictions_rnn = scaler_y.inverse_transform(predictions_rnn)
```

<pre>[] 1 predictions_lstm array([[0.64784867], [0.7939145], [0.8760073], [0.91734505], [0.9463377], [0.954635], [0.96433055], [0.92678976], [0.91426885], [0.90908635], [0.9070188], [0.9066521], [0.90344167], [0.82790685], [0.79426867], [0.7728019], [0.77063394], [0.77390385], [0.77684486], [0.64411306], [0.55627453], [0.49868613], [0.45434925], [0.42345616]], dtype=float32)</pre>	<pre>[] 1 predictions_rnn array([[0.6751179], [0.730054], [0.777158], [0.8019067], [0.822321], [0.8362556], [0.8087793], [0.79767203], [0.79372627], [0.7611372], [0.7212592], [0.7126159], [0.7142496], [0.6735288], [0.62909687], [0.6032502], [0.58850163], [0.5675516], [0.5646546], [0.5064136], [0.42392725], [0.37017712], [0.34581697], [0.3045116]], dtype=float32)</pre>
---	--

13. Now comes the evaluation metrics – RMSE (Root Mean Squared Error) and R^2 (R-squared).

```

1  # As now we have stored our predicted close price values
2  # above , it's time for evaluation metrics to come into the
3  # scene
4
5  from sklearn import metrics
6
7  # Get the root mean squared error (RMSE) for LSTM model
8  mse_lstm = metrics.mean_squared_error(y_test, predictions_lstm)
9  rmse_lstm = np.sqrt(mse_lstm)
10 print("LSTM Model RMSE : ", rmse_lstm)
11
12 # Get r2 score (R-squared - Coefficient of Determination) for LSTM Model
13 r2_lstm = metrics.r2_score(y_test, predictions_lstm)
14 print("LSTM Model r2 : ", r2_lstm)
15 print()
16
17 # Get the root mean squared error (RMSE) for RNN Model
18 mse_rnn = metrics.mean_squared_error(y_test, predictions_rnn)
19 rmse_rnn = np.sqrt(mse_rnn)
20 print("RNN Model RMSE: ", rmse_rnn)
21
22 # Get r2 score for RNN Model
23 r2_rnn = metrics.r2_score(y_test, predictions_rnn)
24 print("RNN Model r2: ", r2_rnn)
```



```
LSTM Model RMSE : 35.867505779966415
LSTM Model r2 : -8.26299886101145

RNN Model RMSE: 35.99546344536891
RNN Model r2: -8.329208424072204
```

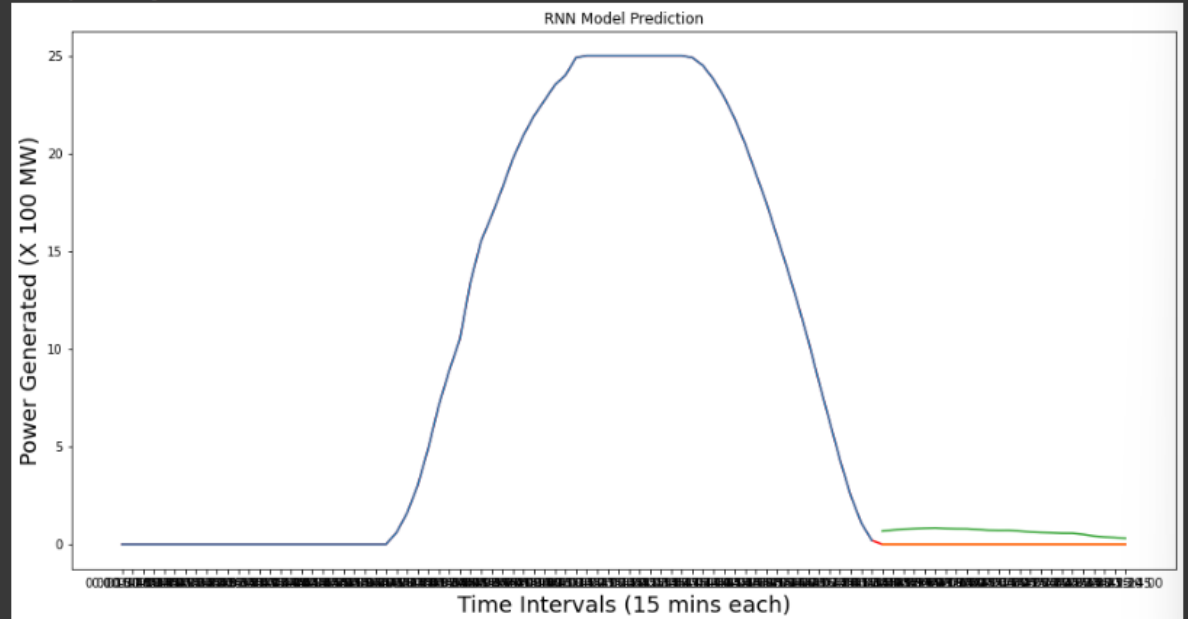
HAULT AGAIN ! Why we are getting R^2 value as negative and why the RMSE value is so high ? This was expected. If talking w.r.t. the powerplant's power generation – even though the plant is not producing any power (Which is an extreme rare case) , it needs some electricity (which might be ignored as being self-consumed) to support its auxiliary equipments to work. Hence , the prediction values will be non-zero at some momements.

14. Let us visualize the above doubt (which is explained) and the prediction values – according to the test data

```
[ ] 1 # Now we shall plot the training data , validation data and Predictions
    2 # of RNN Model
    3
    4 # Data Plotting
    5 train = data[:training_data_len]
    6 valid = data[training_data_len:]
    7 valid['Predictions_(RNN)'] = predictions_rnn
    8
    9 # Data Visualization
   10 plt.figure(figsize=(16,8))
   11 plt.title('RNN Model Prediction')
   12 default_x_ticks = range(len(df_power['TIME DESC']))
   13 plt.plot(default_x_ticks , df_power['DADRIT2'] , color='Red')
   14 plt.xticks(default_x_ticks , df_power['TIME DESC'])
   15 plt.xlabel('Time Intervals (15 mins each)', fontsize=18)
   16 plt.ylabel('Power Generated (X 100 MW)', fontsize=18)
   17 plt.plot(train['DADRIT2'])
   18 plt.plot(valid[['DADRIT2' , 'Predictions_(RNN)']])
   19 plt.show()
```

📄 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/u>
import sys



▶ 1 # Let us see how the predictions goes with the Valid data
2 # for the RNN Model
3 # valid['Date'] = df_INFY[int(training_data_len) :]
4
5 valid['TIME DESC'] = df_power['TIME DESC'][training_data_len:]
6 valid[['TIME DESC' , 'DADRIT2' , 'Predictions_(RNN)']][:5]

📄 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead


See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/u>
"""

	TIME DESC	DADRIT2	Predictions_(RNN)
72	18:00-18:15	0.0	0.675118
73	18:15-18:30	0.0	0.730054
74	18:30-18:45	0.0	0.777158
75	18:45-19:00	0.0	0.801907
76	19:00-19:15	0.0	0.822321

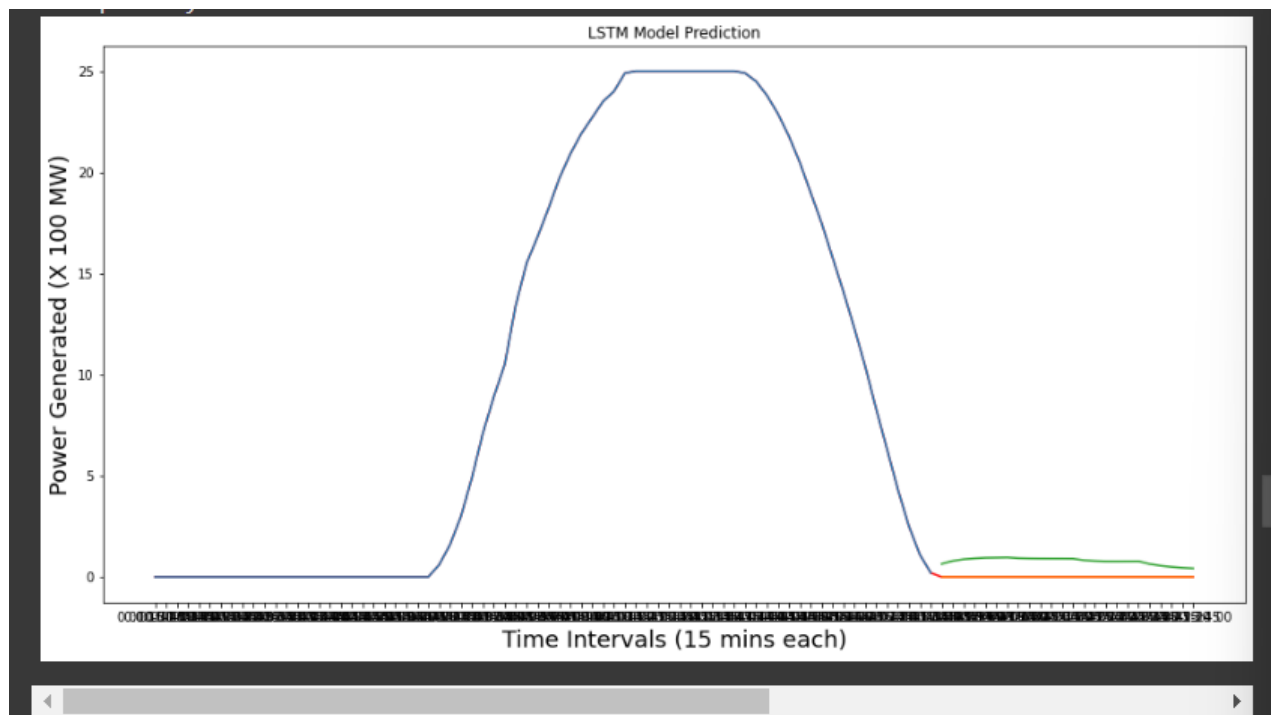
```

1 # Now we shall plot the training data , validation data and Predictions
2 # of LSTM Model
3
4 # Data Plotting
5 train = data[:training_data_len]
6 valid1 = data[training_data_len:]
7 valid1['Predictions_(LSTM)'] = predictions_lstm
8
9 # Data Visualization
10 plt.figure(figsize=(16,8))
11 plt.title('LSTM Model Prediction')
12 default_x_ticks = range(len(df_power['TIME DESC']))
13 plt.plot(default_x_ticks , df_power['DADRIT2'] , color='Red')
14 plt.xticks(default_x_ticks , df_power['TIME DESC'])
15 plt.xlabel('Time Intervals (15 mins each)', fontsize=18)
16 plt.ylabel('Power Generated (X 100 MW)', fontsize=18)
17 plt.plot(train['DADRIT2'])
18 plt.plot(valid1[['DADRIT2' , 'Predictions_(LSTM)']])
19 plt.show()

```

 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead


See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/10min.html#copy-on-write>



```

1  # Let us see how the predictions goes with the Valid data
2  # for the LSTM Model
3  # valid['Date'] = df_INFY[int(training_data_len) : ]
4
5  valid1['TIME DESC'] = df_power['TIME DESC'][training_data_len:]
6  valid1[['TIME DESC' , 'DADRIT2' , 'Predictions_(LSTM)']][:5]

```

 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/u>

	TIME DESC	DADRIT2	Predictions_(LSTM)
72	18:00-18:15	0.0	0.647849
73	18:15-18:30	0.0	0.793914
74	18:30-18:45	0.0	0.876007
75	18:45-19:00	0.0	0.917345
76	19:00-19:15	0.0	0.946338

Now , we shall have a visualized answer as to why the power to be generated (the predicted values) is not ZERO.

15. Now , our main motive of the project to forecast the value of power , to be generated , for initial 10 blocks for the next day. We would like to draw / derive the output for every future block , according to the values of last 30 blocks.

```

[ ] 1 len (test_data)
    34

[ ] 1 predict_test_data = test_data[:,0].reshape((-1))

[ ] 1 def predict(num_prediction , model):
    2     prediction_list = predict_test_data[-30:]
    3
    4     for _ in range(num_prediction-1):
    5         x = prediction_list[-30:]
    6         x = x.reshape((1,10,3))
    7         out = model.predict(x)[0][0]
    8         prediction_list = np.append(prediction_list , out)
    9         prediction_list = prediction_list[30-1:]
    10
    11     return prediction_list
    12
    13 num_prediction = 10 # we need the values of 10 blocks of the next day
    14 forecast_next_day = predict(num_prediction,model_lstm)
    15

```

16. Now that we have found the forecast values , they lack proper show. Thus , modified accordingly.

```
[ ] 1 forecast_next_day

array([0.          , 0.12472469, 0.11905175, 0.15223953, 0.13268505,
       0.1457902 , 0.166721  , 0.16769552, 0.18883625, 0.19931374])

[ ] 1 forecast_next_day = np.reshape(forecast_next_day , (2,5))
    2 forecast_next_day = scaler_y.inverse_transform(forecast_next_day)

[ ] 1 forecast_next_day

array([[0.          , 0.12472469, 0.11905175, 0.15223953, 0.13268505],
       [0.1457902 , 0.166721  , 0.16769552, 0.18883625, 0.19931374]])
    + Code    + Text

[ ] 1 forecast_next_day = np.reshape(forecast_next_day , (10,1))

[ ] 1 forecast_next_day

array([[0.          ],
       [0.12472469],
       [0.11905175],
       [0.15223953],
       [0.13268505],
       [0.1457902 ],
       [0.166721  ],
       [0.16769552],
       [0.18883625],
       [0.19931374]])
```

```
[ ] 1 df = pd.DataFrame(forecast_next_day , columns = ["DADRIT2"] , index = (range(1
    2 df

      DADRIT2
1  0.000000
2  0.124725
3  0.119052
4  0.152240
5  0.132685
6  0.145790
7  0.166721
8  0.167696
9  0.188836
10 0.199314
```

4

```
[ ] 1 new_time_desc = df_power['TIME DESC'][:10]
    2 new_time_desc
```

```
0    00:00-00:15
1    00:15-00:30
2    00:30-00:45
3    00:45-01:00
4    01:00-01:15
5    01:15-01:30
6    01:30-01:45
7    01:45-02:00
8    02:00-02:15
9    02:15-02:30
Name: TIME DESC, dtype: object
```

```
[ ] 1 new_time_desc = np.asarray(new_time_desc)
    2 new_time_desc
```

```
array(['00:00-00:15', '00:15-00:30', '00:30-00:45', '00:45-01:00',
      '01:00-01:15', '01:15-01:30', '01:30-01:45', '01:45-02:00',
      '02:00-02:15', '02:15-02:30'], dtype=object)
```

Power Generation for initial 10 blocks of Next Day

```
[ ] 1 df.insert(0,"TIME DESC",new_time_desc,True)
    2 df
```

	TIME DESC	DADRIT2
1	00:00-00:15	0.000000
2	00:15-00:30	0.124725
3	00:30-00:45	0.119052
4	00:45-01:00	0.152240
5	01:00-01:15	0.132685
6	01:15-01:30	0.145790
7	01:30-01:45	0.166721
8	01:45-02:00	0.167696
9	02:00-02:15	0.188836
10	02:15-02:30	0.199314

The above shows the desired values of power to be generated.

CHAPTER 6

REFLECTION ON THE INTERNSHIP

I consider the internship in this chapter. In relation to my learning objectives, I briefly share my experiences, including if I was successful, any challenges I had, and what I believe I need to do better.

❖ A government organization's operations and working conditions

I had no prior experience working for a PSU in the start. Your reliance on outside organizations and individuals forces you to adopt a flexible outlook. I too felt the reliance when I was there. There was frequently ambiguity about whether and when initiatives could begin. Although the initial dependency and uncertainty bothered me, they drove me to be adaptable and consider my options.

❖ The use of information and abilities acquired at university

It is difficult to identify what information and skills I acquired during my studies that I could put to use during my internship. I had done many projects based on Machine Learning and Data Analysis. I knew how to use Google Colab and Python Notebook. Working in a team-based environment was nothing new to me as I myself have led many team-projects. The pragmatic approach helped me to figure out what all components can be touched for initial run. Apart from that, I also consider myself as good “Report-writer” – hence, I took a genuine interest in writing this report as well.

❖ Knowledge and abilities that might be strengthened in order to perform in a professional setting

There are many areas in which I might still improve, even though working in an organization helps you grow and acquire the essential skills and information. What I could have done in terms of activities to accomplish my learning objectives was not quite evident to me. As a result, I had some challenges during my stay in identifying things that I could do. Prior to my internship, I had a conversation with the company about the project I could work on, but no explicit agreements on my responsibilities were established. My side may have benefited from being more forceful. I will focus more on creating explicit agreements and backup plans for future projects to avoid uncertainty.

Determining an appropriate data collecting and analysis strategy, as well as clearly identifying the study issue, are other factors to which I wish to give particular attention. I

frequently have a propensity to focus more on data collection tasks. I learned throughout my internship that having clear research is crucial since it will direct you throughout the process.

❖ **Organization of Project**

I performed a lot of fieldwork while I was an intern. I've now seen what factors you need to consider while planning a project as a result. In addition, I now understand how to draw up a curriculum and what factors need to be taken into account. To tailor the programme to each group, it is crucial to assess the knowledge already in the room. It is crucial to provide a message that is unbiased, well-supported, and takes into account the opinions of all parties (my teammates actually).

❖ **The impact on prospective career plans**

I had little to no knowledge about what could be the scope of data analysis in today's industry. But , after getting this opportunity to work on a real-industrial problem made me more enthusiastic and opened numerous doors of various options – because data analysis is heavily used in almost every field – say it Medical , Pharmaceutical , Construction , Production and Manufacturing , Education , Space Exploration , etc. Prediction of power generation was one such issue of a field. Hence , looking for more like the aforementioned.

CHAPTER 7

CONCLUSION

Overall, I learned a lot from this internship. In addition to meeting many new individuals, I have learned many new things. I met a few of my learning objectives, but some were not possible due to the circumstances. I got to learn about some new algorithms, especially used for forecasting. Apart from the IT based knowledge that I gained in this internship, I also got to know how a power-plant works. Though I met numerous technical terms which hardly resembles anything to my field of study – but they created a small interest, pulling me to get them know. Additionally, the internship helped me identify my strengths and flaws. This made it easier for me to identify the information and abilities I need to develop going forward. It would be preferable if language proficiency allowed for full participation in initiatives. After finishing my master's, I believe I could begin working.

Finally, this internship has expanded my knowledge and motivated me to work in the fields of data science, machine learning, and, most importantly, artificial intelligence—the superset.