Rains in Bharat - II

By

Rajnish Maurya (CSE 18067/378)

Subrata Kumar Biswas (CSE 18094/405)

Subham Pal (CSE 18093/404)

Bachelor Thesis submitted to

Indian Institute of Information Technology Kalyani

for the partial fulfillment of the degree of

Bachelor of Technology

in

Computer Science and Engineering

May, 2021

Certificate

This is to certify that the thesis entitled "Rains in Bharat" being submitted by **Rajnish Maurya (CSE 18067/378)**, **Subham Pal (CSE 18093/404) and Subrata Kumar Biswas (CSE 18094/405)**, undergraduate students, in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India, for the award of Bachelors of Technology in Computer Science Engineering is an original research work carried by them under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulation of Indian Institute of Information Technology Kalyani and in my opinion, has reached the standards needed for submission. The works, techniques and the results presented have not been submitted to any other university or Institute for the award of any other degree or diploma.

Name of the Guide: Dr. Uma Das

Position: Assistant Professor of Physics

Departmental address: IIIT Kalyani, Webel IT Park, Kalyani – 741235 Nadia, West Bengal

Declaration

I hereby declare that the work which being presented in the thesis entitled " Rains of Bharat" is submitted to Indian Institute of Information Technology Kalyani in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering does not contain any classified information during the period from **January, 2021 to May, 2021** under the supervision of Dr. Uma Das, Department of Physics, Indian Institute of Information Technology Kalyani, West Bengal 741235, India.

Name of the Candidates: Rajnish Maurya, Subham Pal & Subrata Kumar Biswas

Reg No/Roll No: CSE 18067/378, CSE 18093/404 & CSE 18094/405 respectively.

Name of the Department: Computer Science Engineering

Institute Name: IIIT Kalyani

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

....

Dr. Uma Das

Assistant Professor of Physics

Department of Physics, IIIT Kalyani, Kalyani – 741235 Nadia, West Bengal

Place:

Date:

Acknowledgments

First of all, I would like to take this opportunity to thank my supervisor Dr. Uma Das for her unwavering support right from selecting the topic to writing the final thesis for this project. She gave us the opportunity to work in a project that we were looking forward to doing and then giving us the creative freedom to design and work on it how we wished to but simultaneously giving us her constructive criticism and advice on what problems we might face along the way specially her advice on how to plan out a timeline or what we could work on really helped us complete this project in a timely manner. This project would not be possible without her constant support throughout. Last but not the least, she provided us with the dataset without which this project would not have been possible.

Name of the Candidates: Rajnish Maurya, Subham Pal & Subrata Kumar Biswas

Reg No/Roll No: CSE 18067/378, CSE 18093/404 & CSE 18094/405 respectively.

Name of the Department: Computer Science Engineering

Institute Name: IIIT Kalyani

Place:

Date:

Table of Contents

Abstract

A climate change project to forecast precipitation rate in the country in each and every coordinate with 0.25 degree resolution by training with data 1998 to 2019. We will attempt to predict the expected rainfall for one more year for a particular Latitude and Longitude by using various models.

Chapter 1: Introduction

We have collected the data containing the rainfall rate in mm/hour from Huffman et al [1]. This daily accumulated precipitation product is generated from the research-quality 3-hourly TRMM Multi-Satellite Precipitation Analysis TMPA (**3B42**). It is produced at the **NASA** GES DISC, as a value added product. Simple summation of valid retrievals in a grid cell is applied for the data day. The TRMM Microwave Imager (TMI) measured microwave **energy** emitted by the Earth and its atmosphere to quantify the water vapor, the cloud water, and the rainfall intensity in the atmosphere. These terminologies are defined in the website of NASA linked in the references [2]. The size of the folder is 1.29GB containing individual records for 8031 days starting from 01-01-1998 with precipitation rates in mm/hour.

Problem Statement

Make a model to forecast precipitation given in mm/hour for one more year after 2019. Data is present from 1998-2019. In order to implement LSTM and other models there is a need to rearrange the data in a proper manner.

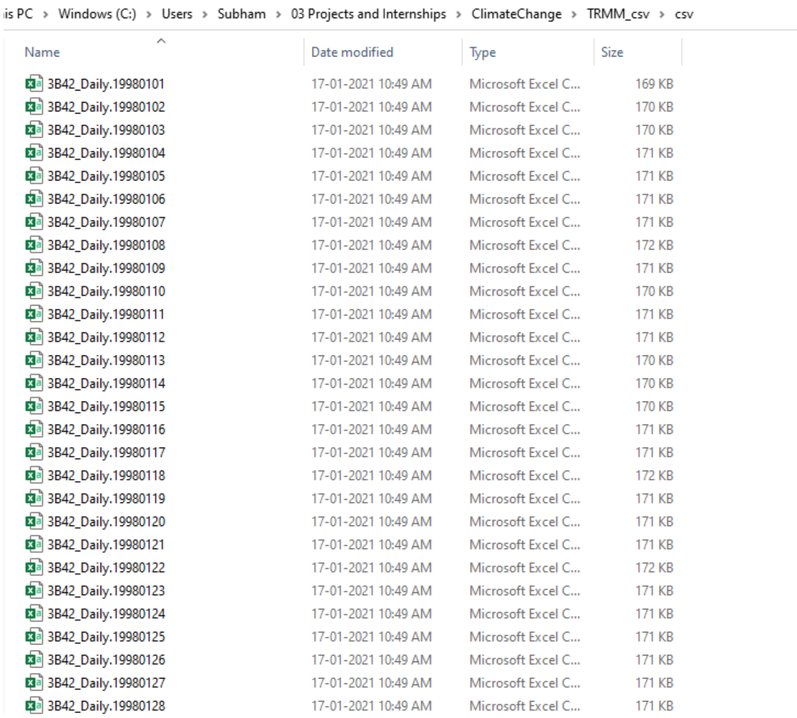
Models Involved

1. Facebook Prophet. (Only for Reference)
2. LSTM
3. ARIMA/SARIMAX

Chapter 2: Data and Methods

Description of Data

Data we received was 3B42 global data in csv format. Each file contains two dimensional arrays of data - 161 x 121 = longitudes and latitudes. For each day there is one individual csv file. Filename contains data in yyyymmdd format.

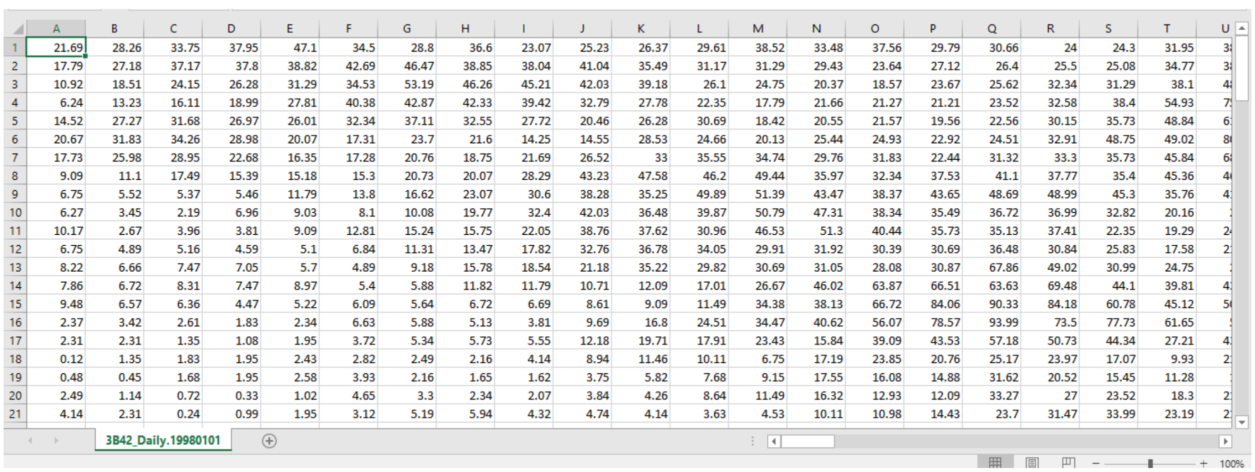


is PC > Windows (C:) > Users > Subham > 03 Projects and Internships > ClimateChange > TRMM_csv > csv

Name	Date modified	Type	Size
3B42_Daily.19980101	17-01-2021 10:49 AM	Microsoft Excel C...	169 KB
3B42_Daily.19980102	17-01-2021 10:49 AM	Microsoft Excel C...	170 KB
3B42_Daily.19980103	17-01-2021 10:49 AM	Microsoft Excel C...	170 KB
3B42_Daily.19980104	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980105	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980106	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980107	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980108	17-01-2021 10:49 AM	Microsoft Excel C...	172 KB
3B42_Daily.19980109	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980110	17-01-2021 10:49 AM	Microsoft Excel C...	170 KB
3B42_Daily.19980111	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980112	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980113	17-01-2021 10:49 AM	Microsoft Excel C...	170 KB
3B42_Daily.19980114	17-01-2021 10:49 AM	Microsoft Excel C...	170 KB
3B42_Daily.19980115	17-01-2021 10:49 AM	Microsoft Excel C...	170 KB
3B42_Daily.19980116	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980117	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980118	17-01-2021 10:49 AM	Microsoft Excel C...	172 KB
3B42_Daily.19980119	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980120	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980121	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980122	17-01-2021 10:49 AM	Microsoft Excel C...	172 KB
3B42_Daily.19980123	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980124	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980125	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980126	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980127	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB
3B42_Daily.19980128	17-01-2021 10:49 AM	Microsoft Excel C...	171 KB

Figure 2.1.

3B42 Daily Data from 1980 to 2019.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	21.69	28.26	33.75	37.95	47.1	34.5	28.8	36.6	23.07	25.23	26.37	29.61	38.52	33.48	37.56	29.79	30.66	24	24.3	31.95
2	17.79	27.18	37.17	37.8	38.82	42.69	46.47	38.85	38.04	41.04	35.49	31.17	31.29	29.43	23.64	27.12	26.4	25.5	25.08	34.77
3	10.92	18.51	24.15	26.28	31.29	34.53	53.19	46.26	45.21	42.03	39.18	26.1	24.75	20.37	18.57	23.67	25.62	32.34	31.29	38.1
4	6.24	13.23	16.11	18.99	27.81	40.38	42.87	42.33	39.42	32.79	27.78	22.35	17.79	21.66	21.27	21.21	23.52	32.58	38.4	54.93
5	14.52	27.27	31.68	26.97	26.01	32.34	37.11	32.55	27.72	20.46	26.28	30.69	18.42	20.55	21.57	19.56	22.56	30.15	35.73	48.84
6	20.67	31.83	34.26	28.98	20.07	17.31	23.7	21.6	14.25	14.55	28.53	24.66	20.13	25.44	24.93	22.92	24.51	32.91	48.75	49.02
7	17.73	25.98	28.95	22.68	16.35	17.28	20.76	18.75	21.69	26.52	33	35.55	34.74	29.76	31.83	22.44	31.32	33.3	35.73	45.84
8	9.09	11.1	17.49	15.39	15.18	15.3	20.73	20.07	28.29	43.23	47.58	46.2	49.44	35.97	32.34	37.53	41.1	37.77	35.4	45.36
9	6.75	5.52	5.37	5.46	11.79	13.8	16.62	23.07	30.6	38.28	35.25	49.89	51.39	43.47	38.37	43.65	48.69	48.99	45.3	35.76
10	6.27	3.45	2.19	6.96	9.03	8.1	10.08	19.77	32.4	42.03	36.48	39.87	50.79	47.31	38.34	35.49	36.72	36.99	32.82	20.16
11	10.17	2.67	3.96	3.81	9.09	12.81	15.24	15.75	22.05	38.76	37.62	30.96	46.53	51.3	40.44	35.73	35.13	37.41	22.35	19.29
12	6.75	4.89	5.16	4.59	5.1	6.84	11.31	13.47	17.82	32.76	36.78	34.05	29.91	31.92	30.39	30.69	36.48	30.84	25.83	17.58
13	8.22	6.66	7.47	7.05	5.7	4.89	9.18	15.78	18.54	21.18	35.22	29.82	30.69	31.05	28.08	30.87	67.86	49.02	30.99	24.75
14	7.86	6.72	8.31	7.47	8.97	5.4	5.88	11.82	11.79	10.71	12.09	17.01	26.67	46.02	63.87	66.51	63.63	69.48	44.1	39.81
15	9.48	6.57	6.36	4.47	5.22	6.09	5.64	6.72	6.69	8.61	9.09	11.49	34.38	38.13	66.72	84.06	90.33	84.18	60.78	45.12
16	2.37	3.42	2.61	1.83	2.34	6.63	5.88	5.13	3.81	9.69	16.8	24.51	34.47	40.62	56.07	78.57	93.99	73.5	77.73	61.65
17	2.31	2.31	1.35	1.08	1.95	3.72	5.34	5.73	5.55	12.18	19.71	17.91	23.43	15.84	39.09	43.53	57.18	50.73	44.34	27.21
18	0.12	1.35	1.83	1.95	2.43	2.82	2.49	2.16	4.14	8.94	11.46	10.11	6.75	17.19	23.85	20.76	25.17	23.97	17.07	9.93
19	0.48	0.45	1.68	1.95	2.58	3.93	2.16	1.65	1.62	3.75	5.82	7.68	9.15	17.55	16.08	14.88	31.62	20.52	15.45	11.28
20	2.49	1.14	0.72	0.33	1.02	4.65	3.3	2.34	2.07	3.84	4.26	8.64	11.49	16.32	12.93	12.09	33.27	27	23.52	18.3
21	4.14	2.31	0.24	0.99	1.95	3.12	5.19	5.94	4.32	4.74	4.14	3.63	4.53	10.11	10.98	14.43	23.7	31.47	33.99	23.19

3B42_Daily.19980101

100%

Figure 2.2.

3B42 Daily Data Sample with 161 x 121 = longitudes x latitudes.

Instructions Provided for Extracting Information

```
IDL> help, lon1
LON1            FLOAT     = Array[161]
IDL> print, lon1
       59.8750      60.1250      60.3750      60.6250      60.8750      61.1250      61.3750      61.6250      61.8750
62.1250
       62.3750      62.6250      62.8750      63.1250      63.3750      63.6250      63.8750      64.1250      64.3750
64.6250
       64.8750      65.1250      65.3750      65.6250      65.8750      66.1250      66.3750      66.6250      66.8750
67.1250
       67.3750      67.6250      67.8750      68.1250      68.3750      68.6250      68.8750      69.1250      69.3750
69.6250
       69.8750      70.1250      70.3750      70.6250      70.8750      71.1250      71.3750      71.6250      71.8750
72.1250
       72.3750      72.6250      72.8750      73.1250      73.3750      73.6250      73.8750      74.1250      74.3750
74.6250
       74.8750      75.1250      75.3750      75.6250      75.8750      76.1250      76.3750      76.6250      76.8750
77.1250
       77.3750      77.6250      77.8750      78.1250      78.3750      78.6250      78.8750      79.1250      79.3750
79.6250
       79.8750      80.1250      80.3750      80.6250      80.8750      81.1250      81.3750      81.6250      81.8750
82.1250
       82.3750      82.6250      82.8750      83.1250      83.3750      83.6250      83.8750      84.1250      84.3750
84.6250
       84.8750      85.1250      85.3750      85.6250      85.8750      86.1250      86.3750      86.6250      86.8750
87.1250
       87.3750      87.6250      87.8750      88.1250      88.3750      88.6250      88.8750      89.1250      89.3750
89.6250
       89.8750      90.1250      90.3750      90.6250      90.8750      91.1250      91.3750      91.6250      91.8750
92.1250
       92.3750      92.6250      92.8750      93.1250      93.3750      93.6250      93.8750      94.1250      94.3750
94.6250
       94.8750      95.1250      95.3750      95.6250      95.8750      96.1250      96.3750      96.6250      96.8750
97.1250
       97.3750      97.6250      97.8750      98.1250      98.37[illegible] [illegible]8.6250  98.8750      99.1250      99.3750
99.6250
       99.8750
```

Figure 2.3.

Rows of the data represent 161 Longitudes starting from 59.8750 to 99.8750 degrees with 0.25 degree resolution. Refer to Figure 3 to understand the context.

```
LAT1            FLOAT     = Array[121]
IDL> print, lat1
       4.87500      5.12500      5.37500      5.62500      5.87500      6.12500      6.37500      6.62500      6.87500
7.12500
       7.37500      7.62500      7.87500      8.12500      8.37500      8.62500      8.87500      9.12500      9.37500
9.62500
       9.87500      10.1250      10.3750      10.6250      10.8750      11.1250      11.3750      11.6250      11.8750
12.1250
       12.3750      12.6250      12.8750      13.1250      13.3750      13.6250      13.8750      14.1250      14.3750
14.6250
       14.8750      15.1250      15.3750      15.6250      15.8750      16.1250      16.3750      16.6250      16.8750
17.1250
       17.3750      17.6250      17.8750      18.1250      18.3750      18.6250      18.8750      19.1250      19.3750
19.6250
       19.8750      20.1250      20.3750      20.6250      20.8750      21.1250      21.3750      21.6250      21.8750
22.1250
       22.3750      22.6250      22.8750      23.1250      23.3750      23.6250      23.8750      24.1250      24.3750
24.6250
       24.8750      25.1250      25.3750      25.6250      25.8750      26.1250      26.3750      26.6250      26.8750
27.1250
       27.3750      27.6250      27.8750      28.1250      28.3750      28.6250      28.8750      29.1250      29.3750
29.6250
       29.8750      30.1250      30.3750      30.6250      30.8750      31.1250      31.3750      31.6250      31.8750
32.1250
       32.3750      32.6250      32.8750      33.1250      33.3750      33.6250      33.8750      34.1250      34.3750
34.6250
       34.8750
IDL> help, data
DATA            FLOAT     = Array[161, 121]
IDL>
```

Figure 2.4.

Columns of the data represent 121 Latitudes starting from 4.8750 to 34.8750 degrees with 0.25 degree resolution. Refer to Figure 3 to understand the context.

Reshaping the daily 3B42

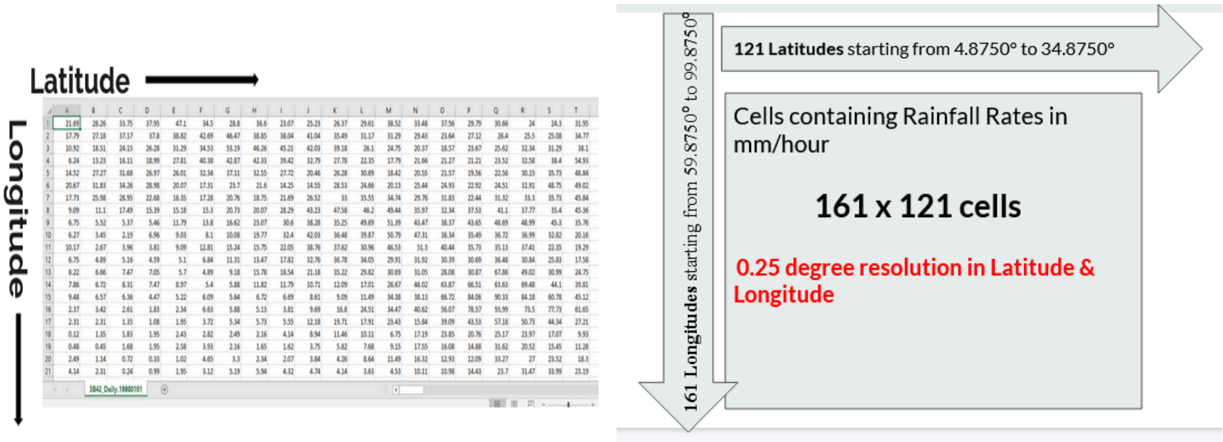


Figure 2.5

Latitude and Longitude is arranged in this manner. Refer to Figure 2.3 and 2.4.

We can see how Latitude and Longitudes are mapped. In order to get precipitation rate for a particular day it is possible to find out from here. But generating an entire time series is still not possible.

In Figure 2.5, we have seen data in nxm matrix format. Now we are going to squeeze the entire thing and make it 1-D.

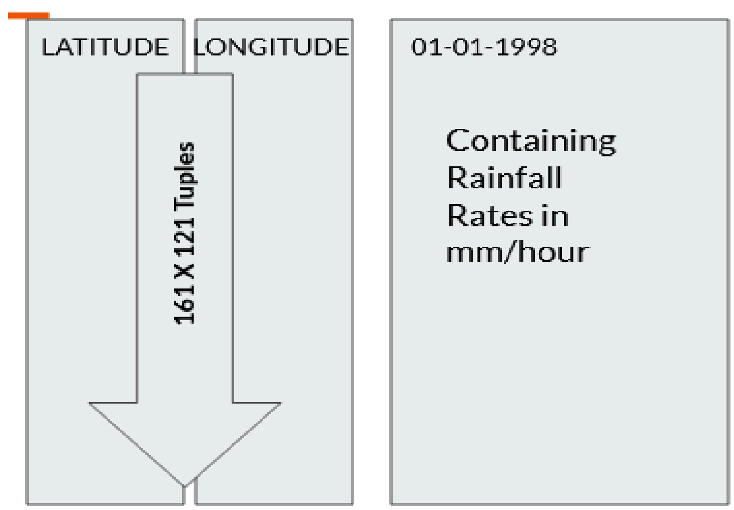


Figure 2.6

Reshaping Data.

Merging CSV Files to a single one

As we know, training a LSTM model takes time, so arranging the data in a proper way is necessary to make a robust model. So far, we can see that data is arranged in separate CSV files for a particular day. In Figure 4 and Figure 5, we received the key to access any coordinate of that particular csv file to give us the rainfall rate for that day. But, in order to get a time series data for a particular location, we need to gather data for that coordinate from all the files. Iterating this process everytime will make the process slow.

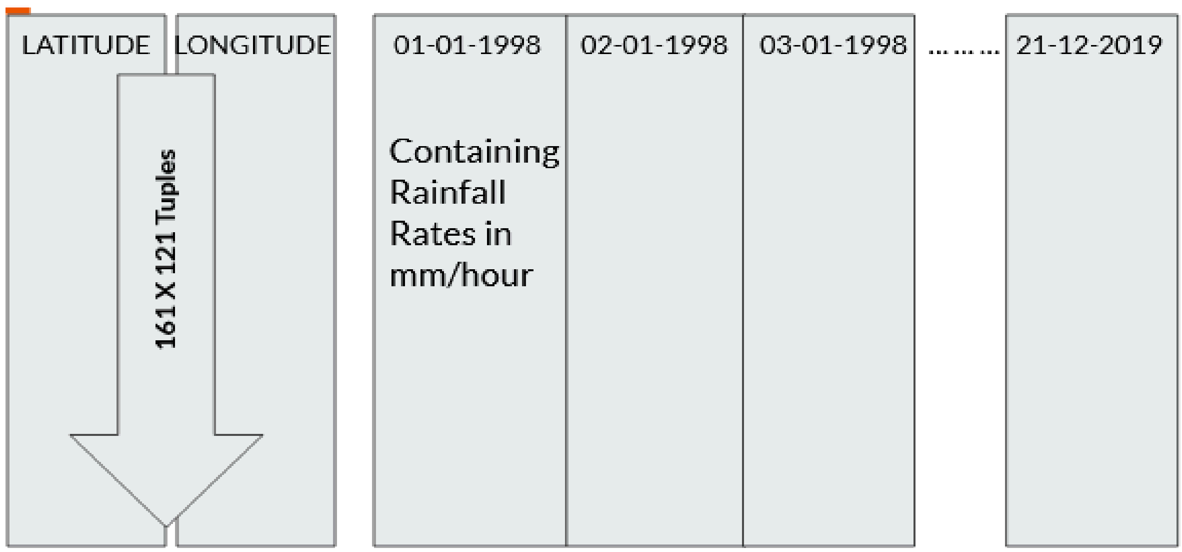


Figure 2.7

Process of making a single Global.csv

We made an attempt to merge all the individual csv files to a single one.

```
lat = [59.875 + 0.25*x for x in range(161)]
lon = [4.875 + 0.25*x for x in range(121)]
```

```
lat_lon = []
for i in lat:
    for j in lon:
        lat_lon.append((i,j))
lat_lon
```

```
[(59.875, 4.875),
 (59.875, 5.125),
 (59.875, 5.375),
 (59.875, 5.625),
 (59.875, 5.875),
```

Figure 2.8.

Generating all possible combinations of Latitude and Longitude.

```
Global = pd.DataFrame(columns=[],
                      index=pd.MultiIndex.from_tuples(lat_lon))
```

```
import glob

path = r'C:\Users\Subham\03 Projects and Internships\ClimateChange\TRMM_csv\csv' # use your path
all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
    df = pd.read_csv(filename, names = range(161))


    date = str(datetime.datetime.strptime(filename.split('.')[-2], "%Y%m%d").date())
    ## Reshaping to 1-D
    ls = []
    for x in range(161):
        ls.extend(list(df[x]))
    Global[date] = ls
```

Figure 2.9.

Creating Global.csv with the latitude-longitude as index as created in Figure 5.

Merging all the individual files to a single csv file.

```
: Global
```

		1998-01-01	1998-01-02	1998-01-03	1998-01-04	1998-01-05	1998-01-06	1998-01-07	1998-01-08	1998-01-09	1998-01-10	...	2019-12-21
59.875	4.875	21.69	2.550000	0.0	2.34	93.72	27.30	1.38	15.00	22.14	53.220000	...	46.05
	5.125	17.79	6.750000	0.0	0.84	64.62	25.41	4.20	7.41	7.35	52.380000	...	39.69
	5.375	10.92	5.880000	0.0	0.00	31.59	24.87	7.08	2.88	4.17	45.300000	...	20.85
	5.625	6.24	9.600000	0.0	0.00	28.95	26.64	10.56	1.44	3.84	31.470000	...	40.38
	5.875	14.52	18.660000	0.0	0.00	24.78	29.46	10.29	0.00	7.44	22.680000	...	34.77
...	...	...	...	...	...	...	...	...	...	...	...	...	...
99.875	33.875	0.00	0.000000	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.000000	...	0.00
	34.125	0.00	0.008728	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.000000	...	0.00
	34.375	0.00	0.000000	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.000000	...	0.00
	34.625	0.00	0.000000	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.000000	...	0.00
	34.875	0.00	0.000000	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.027414	...	0.00

19481 rows × 8031 columns

Figure 2.10

Merged Global csv file. The multi-index gives

all possible coordinates and we get columns for each day.

Now getting time series data for a particular coordinate is very easy, we no longer need to peek into individual csv files as all data is available in the Global.csv. Moreover the data can be retrieved by knowing the coordinates of that place.

```
## Find data for lat-lon
Global.loc[(59.875,4.875)]
```

```
1998-01-01    21.69
1998-01-02     2.55
1998-01-03     0.00
1998-01-04     2.34
1998-01-05    93.72
              ...
2019-12-26     0.00
2019-12-27     0.00
2019-12-28     0.00
2019-12-29     0.00
2019-12-30     0.48
Name: (59.875, 4.875), Length: 8031, dtype: float64
```

Figure 2.11.

Extracting Time Series from Global.csv

Automatic Latitude Longitude Generation System

With the help of the **geopy** module we can get the coordinates of a particular place upon providing the name of the place. In case it fails to recognize the location, then we made it flexible to take manual input.

```
def get_city_coordinates():
  from geopy.geocoders import Nominatim
  geolocator = Nominatim(user_agent="my_user_agent")
  city =input('**HIT ENTER for Manual Mode **OR**\nEnter Location As "City" or "City, Country" format: ')
  country =""
  loc = geolocator.geocode(city+','+ country)
  print("latitude is " ,loc.latitude,"&longtitude is " ,loc.longitude)
  return loc.latitude,loc.longitude
```

Figure 2.12.

Returns coordinate on providing the location name.

Timeseries Generator and Validator

In the previous section, we managed to get the coordinate location. Now we need to find out the nearest location according to our dataset. We designed one algorithm for the same task. If we fail to get the nearest coordinate then it will prompt us to take another location. If we are able to match the coordinates then we generate a time series dataframe.

```
## Findout closest coordinates wrt to available data
def closest(pos):
    latitude = pos[0]
    longitude = pos[1]
    lon = [59.875 + 0.25*x for x in range(161)]
    lat = [4.875 + 0.25*x for x in range(121)]
    LAT = lat[min(range(len(lat)), key = lambda i: abs(lat[i]-latitude))]
    LON = lon[min(range(len(lon)), key = lambda i: abs(lon[i]-longitude))]
    if (abs(latitude-LAT) > 1) or (abs(longitude-LON)>1):
        return 0
    return (LON,LAT)
```

Figure 2.13.

Code to find the closest coordinate.

```
## Run this function to create DataFrame
df = GetLocation()
df.plot(figsize=(25,5))
```

```
**HIT ENTER for Manual Mode **OR**
Enter Location As "City" or "City, Country" format: kolkata
latitude is  22.5414185 &longtitude is  88.35769124388872
Table Created with nearest co-ordinates: (88.375, 22.625)
```

	Rainfall
1998-01-01	0.0000
1998-01-02	0.0000
1998-01-03	0.0000
1998-01-04	4.8600
1998-01-05	0.0000
...	...
2019-12-26	0.0000
2019-12-27	0.0000
2019-12-28	0.0000
2019-12-29	5.0818

2019-12-30	0.0000

8031 rows × 1 columns

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3bbadb6410>
```

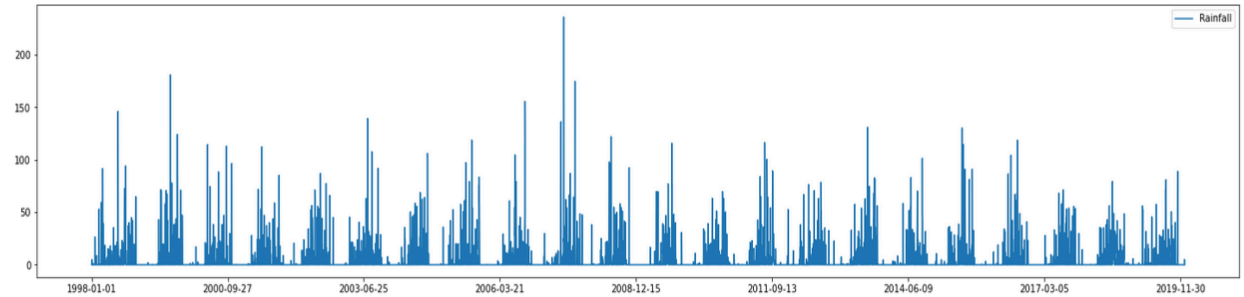


Figure 2.14.

Generation of time series data frame along with plot for location **Kolkata**

All the models are made with respect to Kolkata as its location as given in Figure 14.

Model Using Facebook Prophet

Facebook Prophet is an open source library for univariate time series forecasting. The model takes very less time yet a robust one. We created this model only for reference to LSTM and ARIMAX.

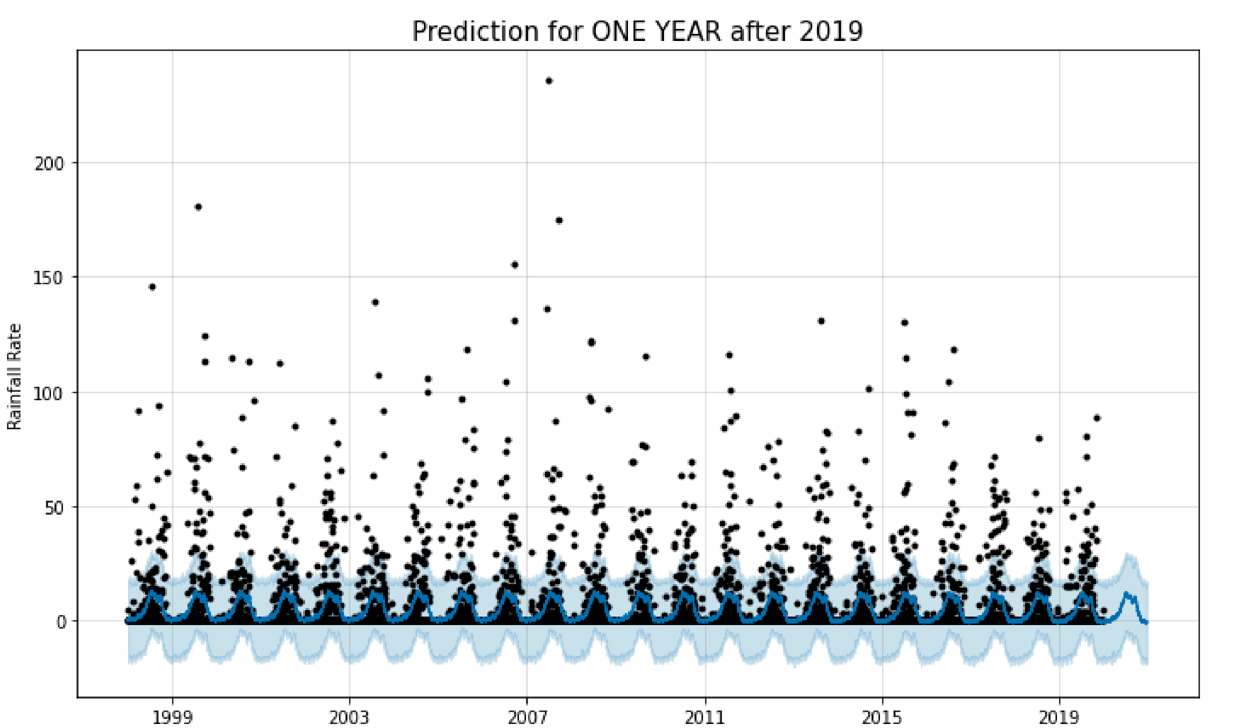


Figure 2.15.

FB Prophet Prediction of Rainfall Rate (mm/hr).

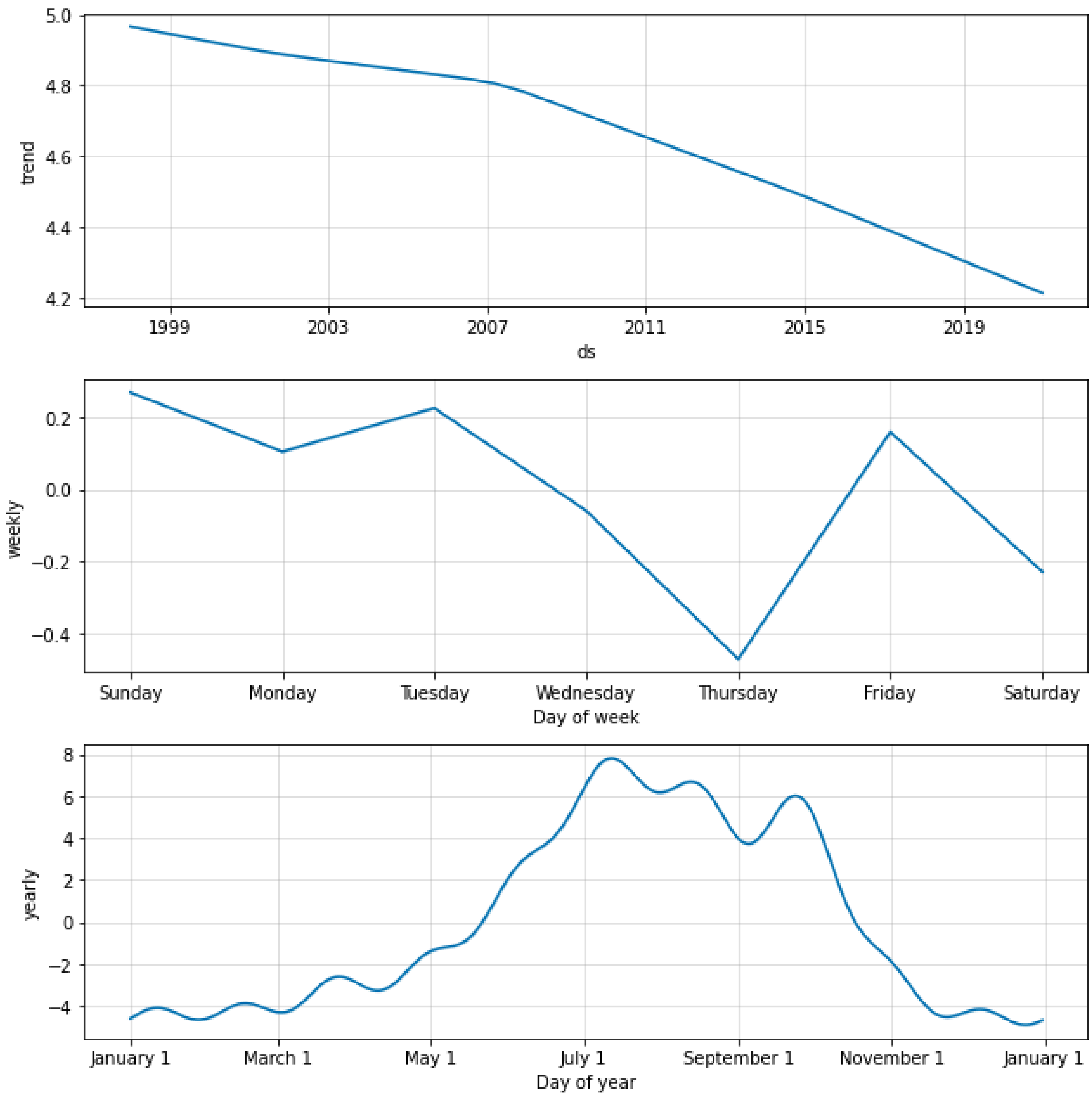


Figure 2.16.

Trend Analysis of Rainfall Rate (mm/hr)

Model Using LSTM

Long short-term memory network

Long short-term memory network(LSTM) is a particular form of recurrent neural network(RNN), which is the general term of a series of neural networks capable of processing sequential data. LSTM is a special type of network structure with three "gates" structures. Three gates in an LSTM unit are input gate , forgetting gate and output gate. While information enters the LSTM network , it can be selected by rules , only the information confirmed to the algorithm will be left, and the information that does not confirm will be forgotten through the forgetting gate.

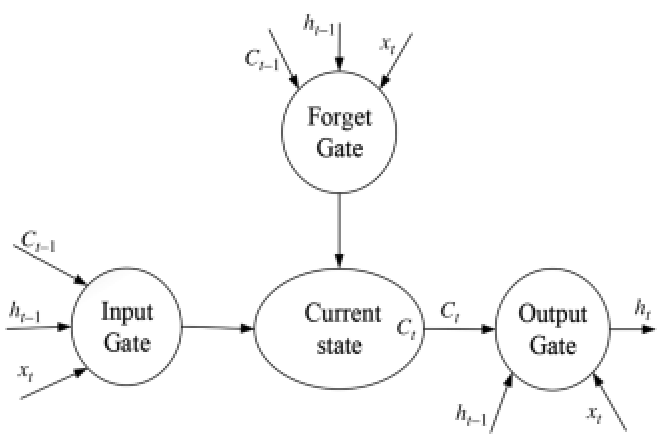


Figure 2.17.

Gates in LSTM

The LSTM can add and delete information for neurons through the gating unit. To determine selectively whether information passes or not, it consists of a sigmoid neural network layer and a pair of multiplication operations. Each element output by the Sigmoid layer is a real number between [0, 1], representing the weight through which the corresponding information .
LSTM unit structure passes. In the LSTM neural network, there is also a layer containing tanh activation function which is used for updating the state of neurons.

σ(x) =(1)/(1 + e^−x)

tanh (x) =(e^x − e^−x)/(e^x + e^−x)

The forgetting gate of the LSTM neural network determines what information needs to be discarded, which reads h_{t-1} and x_t , gives the neuron state C_{t-1} a value of 0–1. The calculation method of forgetting probability is given below:

$f_t=\sigma(W_f \cdot [h_{t-1}, x_t]+b_f)$

where h_{t-1} represents the output of the previous neuron and x_t is the input of the current neuron. σ is the sigmoid function.

The input gate determines how much new information is added to the neuron state. First, the input layer containing the sigmoid activation function determines which information needs to be updated, and then a tanh layer generates candidate vectors c_t, an update is made to the state of the neuron.

$C_t = f_t * C_{t-1} + i_t * c_t$

where the calculation method of i_t and c_t are shown below:

$i_t = \sigma(W_i \cdot [h_{t-1}, x_t]+ b_i)$

$c_t = \tanh(W_c \cdot [h_{t-1}, x_t]+ b_c)$

The output gate is used to control how many current neural united state are filtered and how many controlling units state are filtered which are shown below:

$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

$h_t = o_t * \tanh(C_t)$

LSTM Model Building, training and Evaluation

The first step is to define the model of the LSTM network as a Sequential class. Sequential class is a sequence of layers in a neural network. We create layers and add them in the order that they should be connected. A fully connected layer that often follows LSTM layers and is used for outputting a prediction called Dense. In a layer the first hidden layer in the network must define the number of inputs to expect, e.g the shape of the input layer. Input must be three-dimensional , consisting of samples, time steps and features in that order.

- **Samples** These are the rows in our data. One sample may be one sequence.
- **Time steps** These are the past observation for a feature, such as lag variable.
- **Features** These are columns in our data.

We create our 2D dataset to a 3D dataset using reshape() function in NumPy. The model assumes one or more samples to define the number of time steps and features.Here, Sequential model works as a pipeline in which our raw data fed in at one end and predictions that come at the other end. Sequential layer helps in transforming the data from input to prediction. For example, activation functions that transform a summed signal from each neuron in a layer can be extracted and added to the Sequential as a layer-like object called Activation. The choice of activation function is most important for the output layer as it will define the format that predictions will take.So, we have used logistic activation function(sigmoid-function).

Once we have defined our network, we must compile it because compilation is an efficient step. Compilation requires a number of parameters to be specified, specifically tailored to training the network. The **stochastic gradient descent(sgd)** optimization algorithm was used to train the network to find the model parameters that correspond to the best fit between predicted and actual outputs and the **mean squared error(mse)** loss function is used to evaluate the network that is minimized by the optimization algorithm.

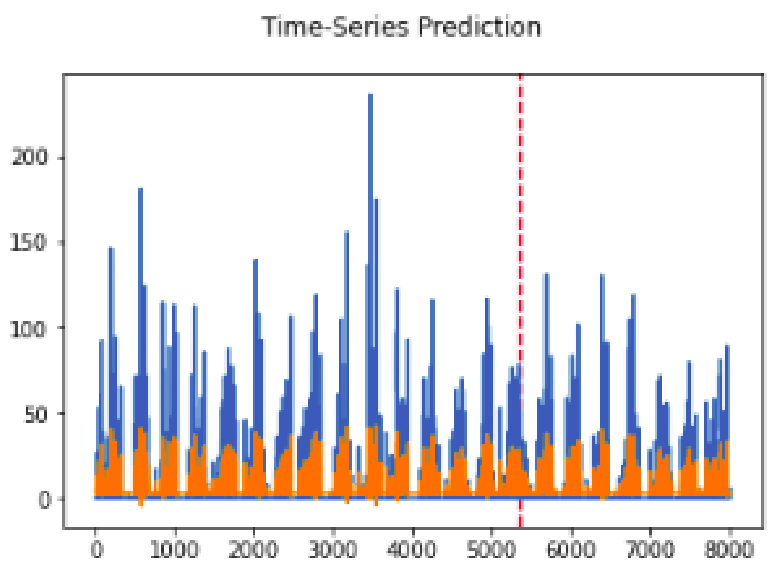


Figure 2.18.

Right side shows the predicted output.

Before the data were used to train the LSTM Network , they were split into a training set and a test set. The data were split by assigning the first 80% to the training set and the last 20% to the testing set. Finally, the training and testing set were split into an input and targeted array. The network is trained using the **Backpropagation Through Time algorithm** and optimized according to the optimization algorithm and the loss function specified when compiling the model.The backpropagation algorithms requires that the network should be trained for a specified number of epochs to all the sequences in the training dataset.Each epoch is partitioned into groups of input-output pattern pairs called batches.

Epoch: One passes through all samples in the training dataset and updates the network weights. LSTMs may be trained for tens, hundreds, or thousands of epochs.

Batch: A pass through a subset of samples in the training dataset after which the network weights are updated. One epoch consists of one or more batches.

Our model is fitted with a 32 batch size and 2000 epochs. The model evaluates the loss function across all of the test patterns.When error values are reduced due to back-propagation, then the deviation of predicted output with actual output is minimized, which in turn is defined as loss. Errors are reduced gradually by achieving stability, and the gradient persists through the proposed model, which leads to minimal losses.The global minimum loss was found to be 0.00334 at around 1800 epochs and remains within 0.00334 until the 1900 epochs, which shows the stability of the proposed model.

Model with ARIMA / SARIMAX

There are two popular models while dealing with time series datasets. One is the Auto-regressive model and the other is the Moving Average model. The basic difference between them is that the moving average (MA) model does not use the past forecasts to predict the future values whereas it uses the errors from the past forecasts. While, the autoregressive model (AR) uses the past forecasts to predict future values.

But, while making accurate predictions, we need both of the models altogether and thus ARIMA was introduced which is abbreviated as Auto-regressive Integrated Moving Average model. It uses both models over a time series dataset. But the problem with ARIMA is that it does not support seasonal data. That is a time series with a repeating cycle. ARIMA expects data that is either not seasonal or has the seasonal component removed.

Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component. (Method name is SARIMAX where 'X' stands for exogenous variable support). It added four seasonal elements to ARIMA i.e P,D,Q & m that stands for Seasonal autoregressive order, Seasonal difference order, Seasonal moving average order and the number of time steps for a single seasonal period respectively.

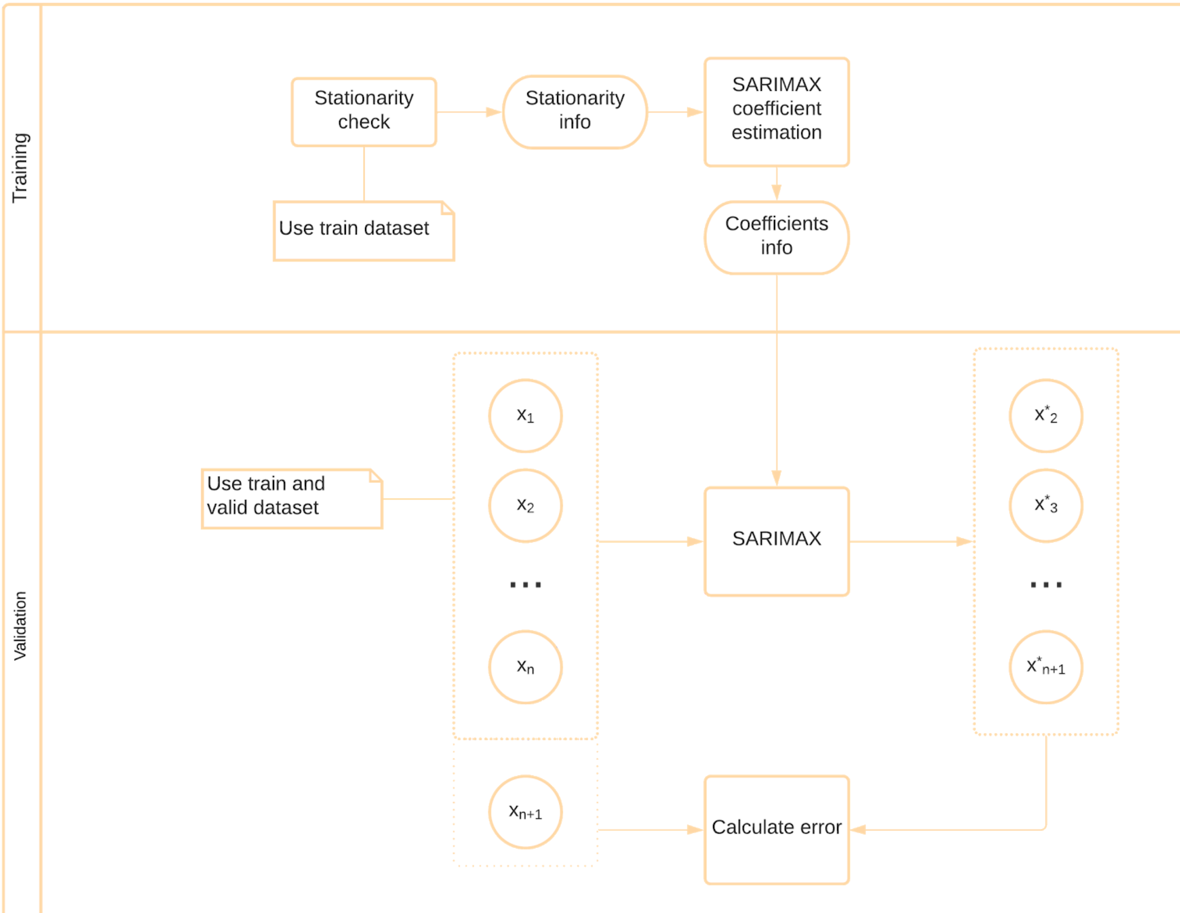


Figure 2.19

Architecture of SARIMAX

Given time series data X_t where t is an integer index and the X_t are real numbers, an ARIMA model output is given by,

$$\left(1-\sum_{i=1}^{p}\varphi_i L^i\right)(1-L)^d X_t = \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t$$

Where L is the lag operator. ε_t is the error term.

Firstly, we create a monthly distribution of our data for seasonality analysis. Visualisation of data helps in tracing seasonality more easily.

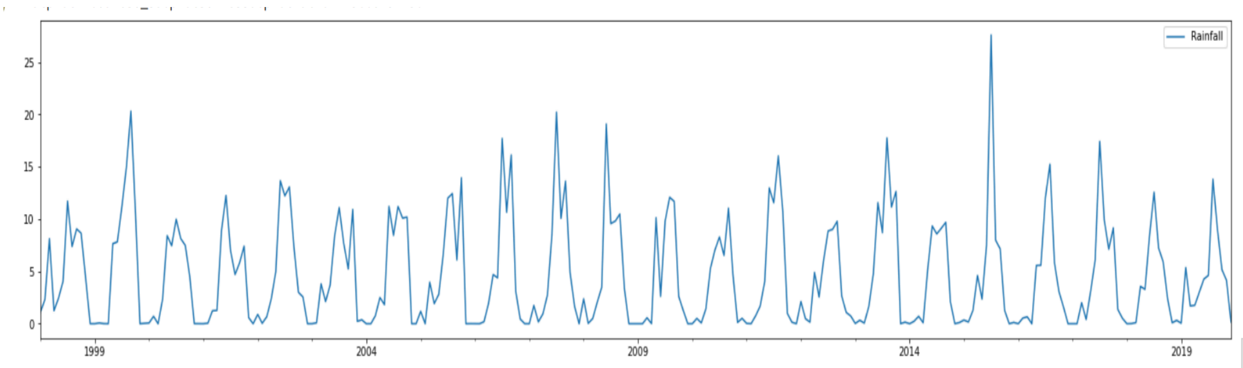


Figure 2.20

Monthly Distribution

Firstly, let's try to make predictions using the ARIMA model.

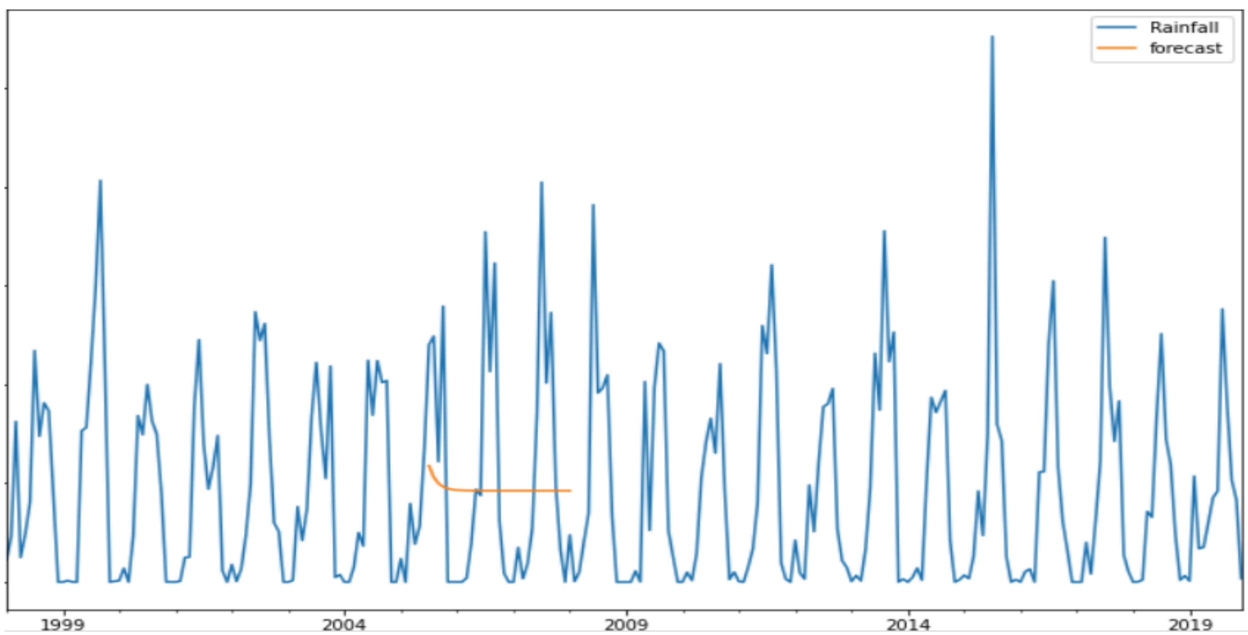


Figure 2.21

ARIMA (train-test split)

Here, it is clearly visible that there is no seasonality in our prediction. The reason is the absence of seasonality factors in our model. Therefore, ARIMA is not recommended for predicting seasonal variations. For the purpose, we will use the SARIMAX model.

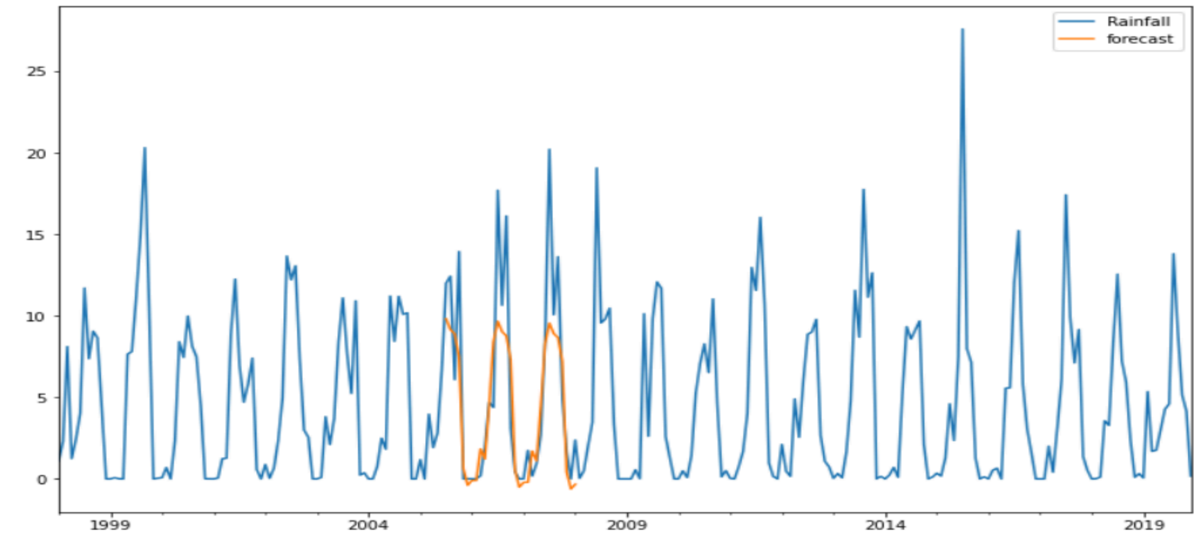


Figure 2.22

SARIMAX - Seasonality Verification (train-test split)

We just verified the presence of seasonality in our predictions by using 'train-test split' of our original data.Therefore, it is verified that the SARIMAX model can make predictions on seasonal datasets. Now, it's time to make our predictions.

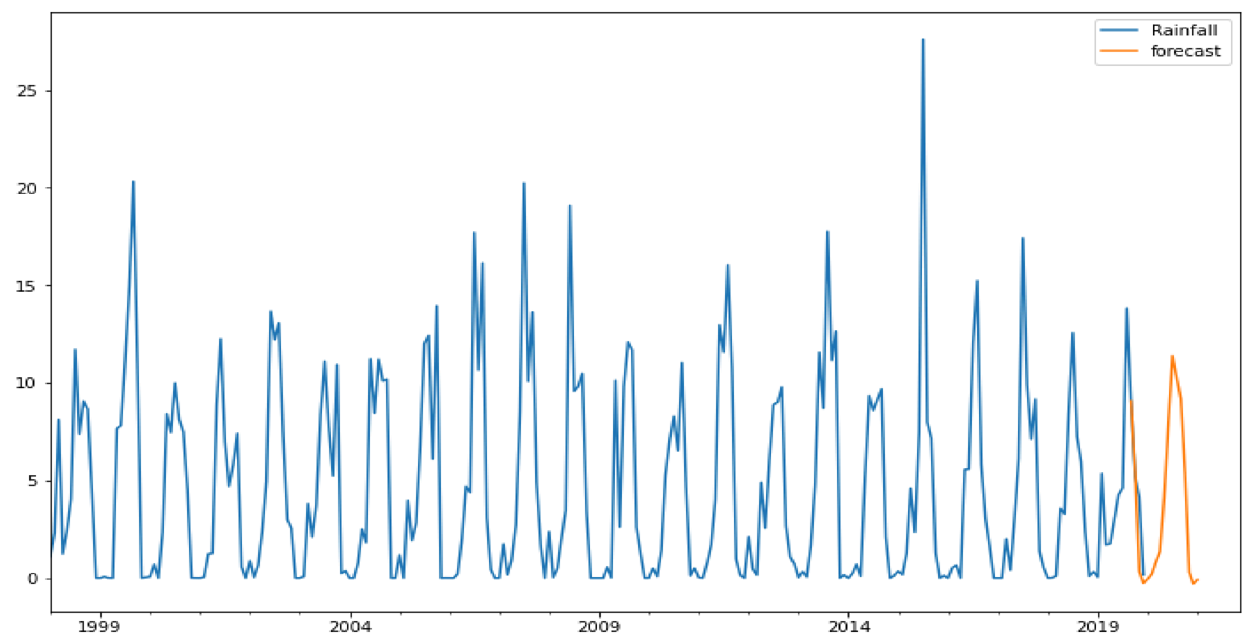


Figure 2.23

SARIMAX Predictions

Chapter 3: Results and Discussion

Model selection is an important factor in predicting satisfactory results. Out of ARIMA and LSTM, LSTM yields better data-fitting and overfitting in long term modelling. In different random training of the model, it was found 84-87% of reduction in error rates while data-fitting as compared to ARIMA model claiming the superiority of LSTM over ARIMA in long term modelling. While predicting, LSTM modelling has the advantage of better error handling when it comes to presence of huge deviation in seasonal dataset. The reason is the possibility of learning noisy and nonlinear relationships with every step and explicit handling of ordered observations by adapting itself.

But, ARIMA/SARIMAX model out-performs machine learning and deep learning methods in both one-step and multi-step forecasting on univariate datasets given that there shouldn't be much errors and deviations other than the seasonality itself.

Chapter 4: Future Work

In the future we will try to put some natural calamities like Cyclones and its effect on the monsoon winds resulting in irregularities. There is also a trend of extreme rainfall in some pockets and drought in other parts of the country. We will try to study the effect of rainfall on the rainfed rivers as most of them are dying. We will deploy the model with FB Prophet in the near future.

References

1. Huffman, G.J., D.T. Bolvin, E.J. Nelkin, and R.F. Adler (2016), TRMM (TMPA) Precipitation L3 1 day 0.25 degree x 0.25 degree V7, Edited by Andrey Savtchenko, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [Data Access Date], 10.5067/TRMM/TMPA/DAY/7
2. NASA, https://gpm.nasa.gov/category/keywords/tmpa

Appendix

Notebook 1: https://nbviewer.jupyter.org/github/subhmm/rainGLOBAL/blob/main/GlobalRainfall.ipynb

Notebook 2:

https://colab.research.google.com/drive/1mieY6EY_DpAiVapjmxw8NtwqlXWq5B6T?usp=sharing