Time-Series Modelling for COVID-19 Prediction

Presented By
Md Ahmad Jami

INTRODUCTION

- COVID-19 is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2).
- The disease was first identified in December 2019 in Wuhan.
- The first case of COVID 19 in India is reported in 30 January 2020 in Kerala.
- **Symptoms**: fever, cough, fatigue, shortness of breath and loss of smell.

DATASET

Data has been extracted from various website till November 2021 and stored in

GitHub and kaggle

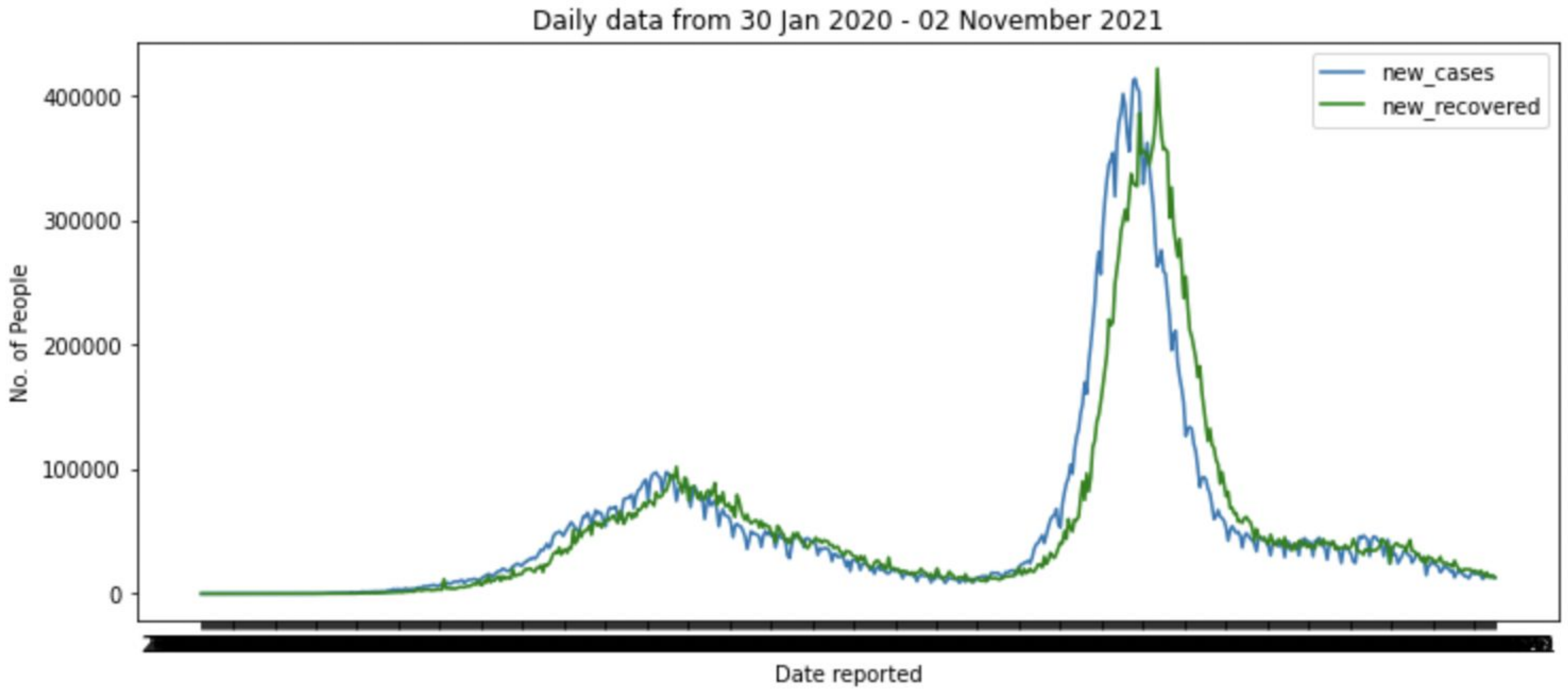
- Johns Hopkins University
- World Health Organization (WHO): https://www.who.int/
- Government of India: https://www.mygov.in/covid-19, Ministry of Health and Family Welfare: https://www.mohfw.gov.in/
- Our World in Data: https://ourworldindata.org/covid-vaccinations?country=~IND

Data Set

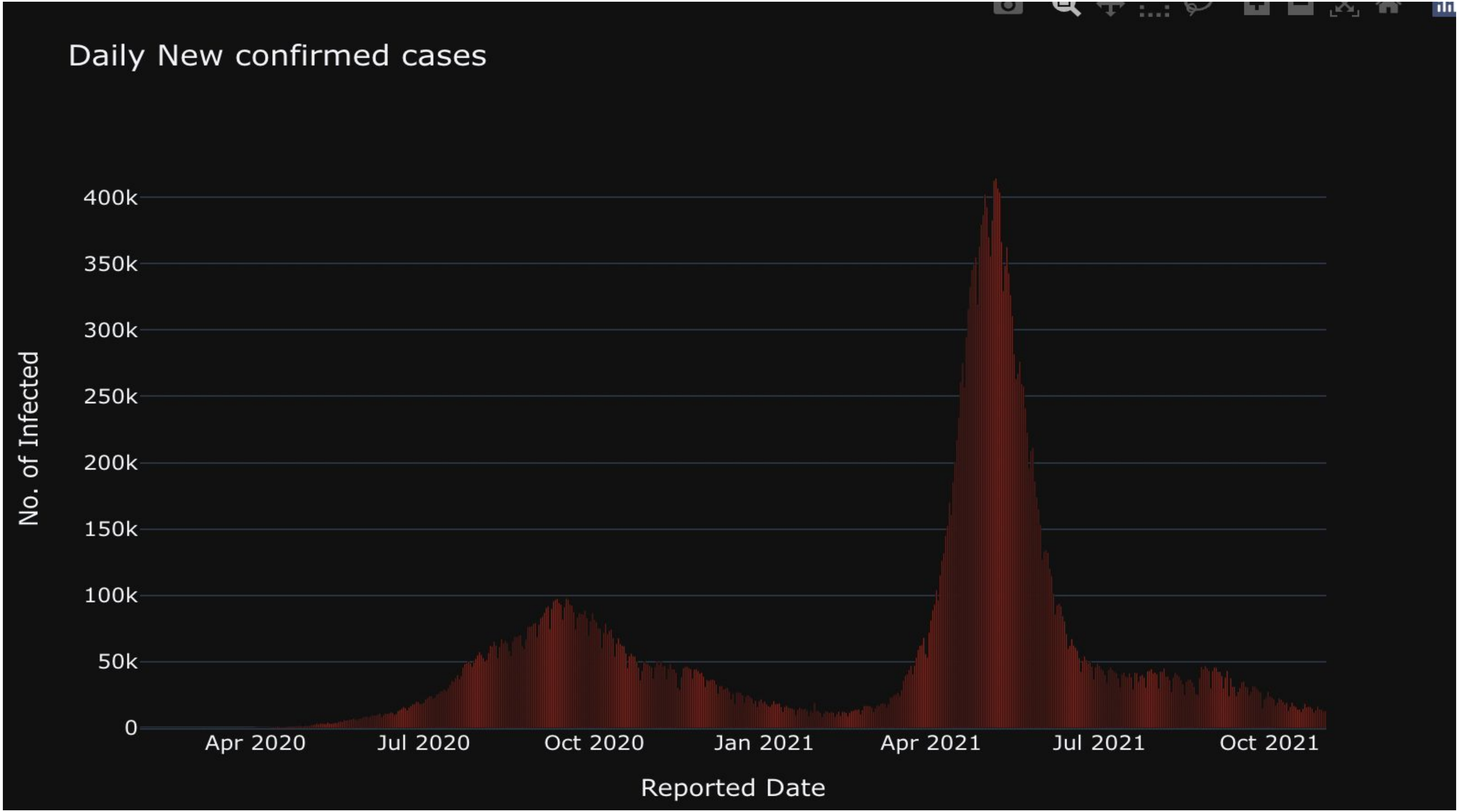
	Date_reported	new_cases	cum_cases	new_death	cum_death	cum_recovered	cum_active_cases
0	2020-01-30	1	1	0	0	0.0	1.0
1	2020-01-31	0	1	0	0	0.0	1.0
2	2020-02-01	0	1	0	0	0.0	1.0
3	2020-02-02	1	2	0	0	0.0	2.0
4	2020-02-03	1	3	0	0	0.0	3.0

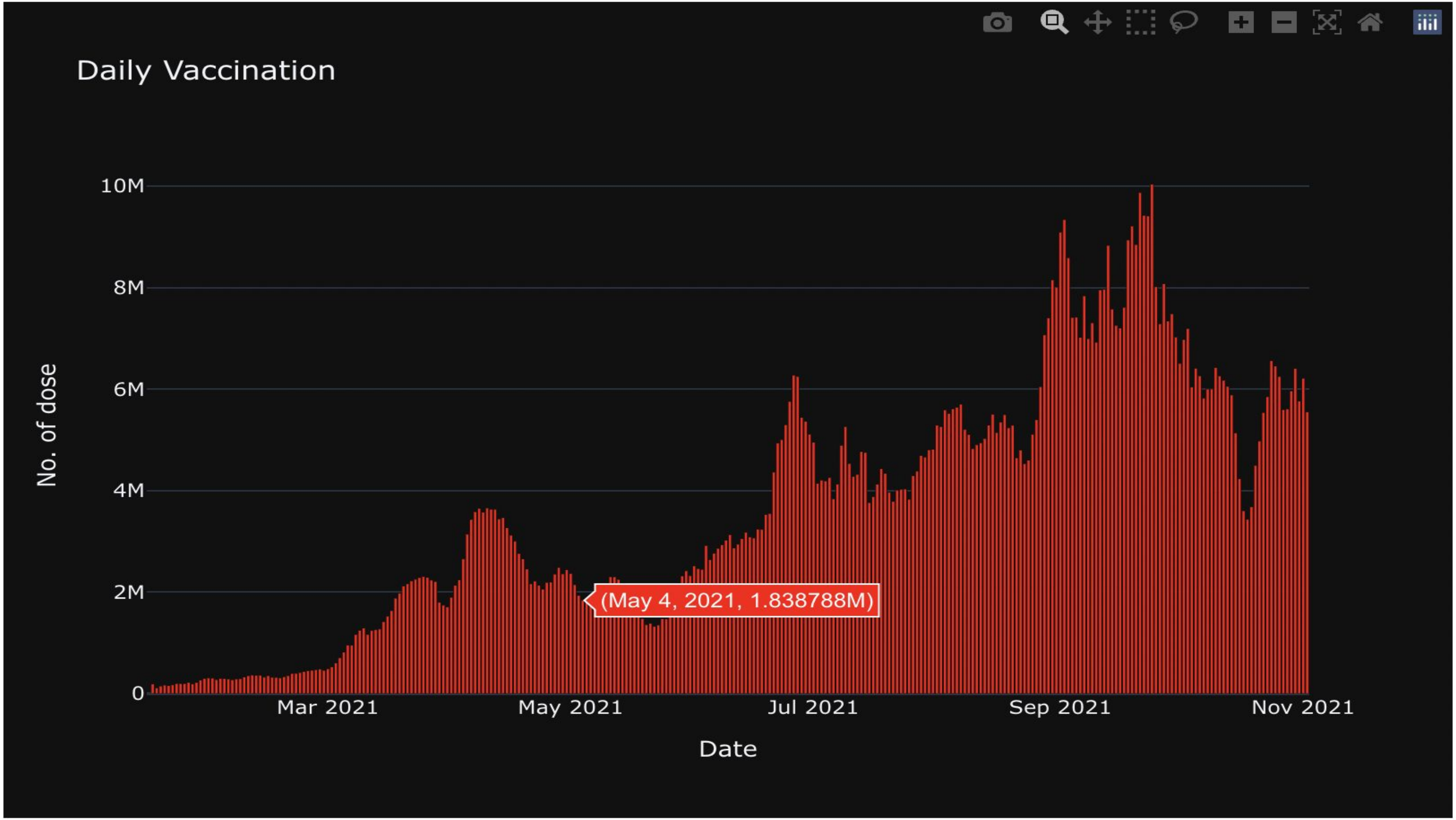
Vaccination DataSet

	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinated_per_hundred
0	2021-01-15	0.0	0.0	NaN	NaN	NaN	0.00	0.00
1	2021-01-16	191181.0	191181.0	NaN	191181.0	191181.0	0.01	0.01
2	2021-01-17	224301.0	224301.0	NaN	33120.0	112150.0	0.02	0.02
3	2021-01-18	454049.0	454049.0	NaN	229748.0	151350.0	0.03	0.03
4	2021-01-19	674835.0	674835.0	NaN	220786.0	168709.0	0.05	0.05



Daily data from 30 Jan 2020 - 02 November 2021
new_cases
new_recovered
400000
300000
200000
100000
0
No. of People
Date reported


Daily New confirmed cases
No. of Infected
400k
350k
300k
250k
200k
150k
100k
50k
0
Apr 2020
Jul 2020
Oct 2020
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Reported Date



Daily Vaccination
10M
8M
6M
4M
2M
0
No. of dose
Mar 2021
May 2021
Jul 2021
Sep 2021
Nov 2021
Date
(May 4, 2021, 1.838788M)

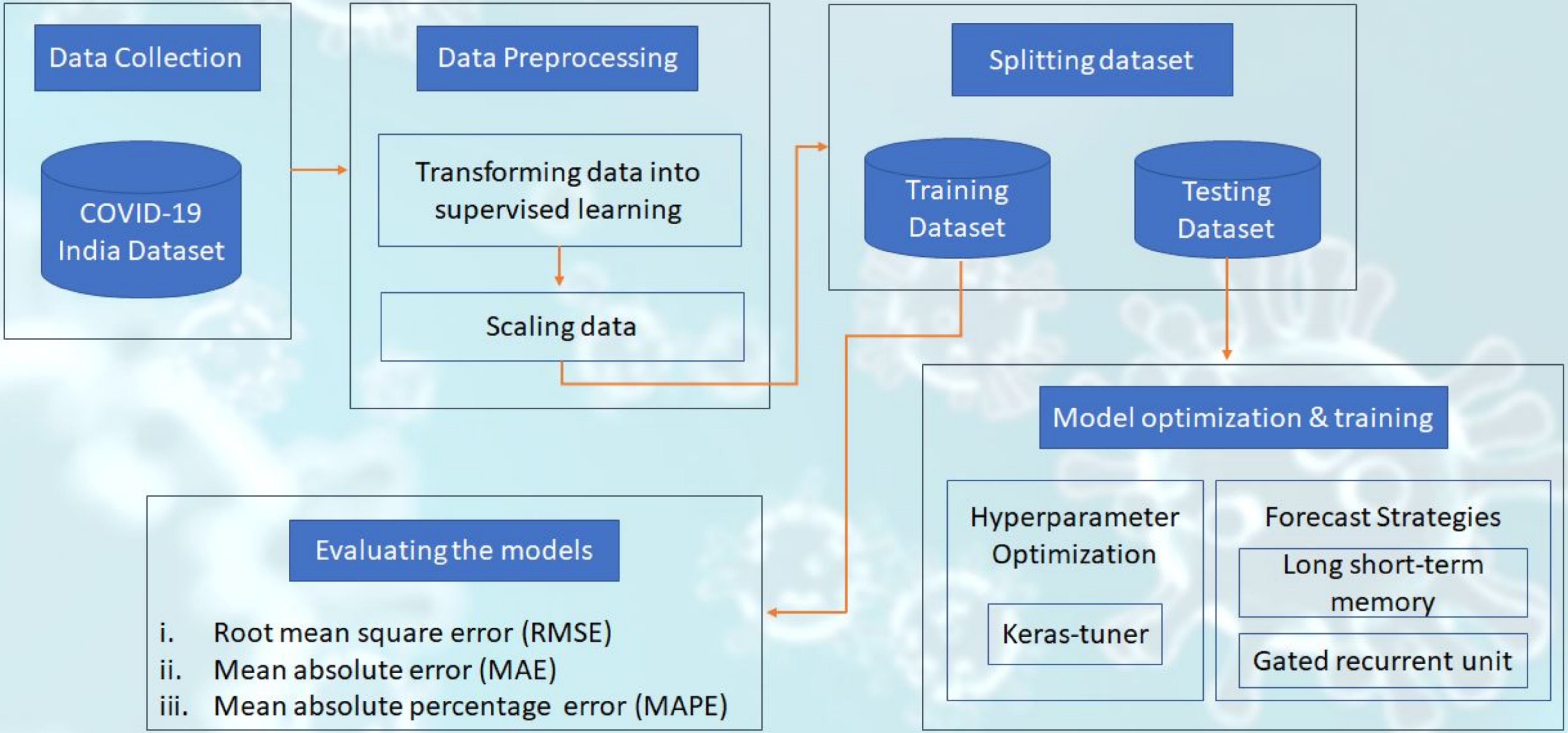
Objective or Problem Statement

To predict the 3rd Wave of Covid 19 using variant of Recurrent Neural Network: Long Short Term Memory

METHODOLOGY

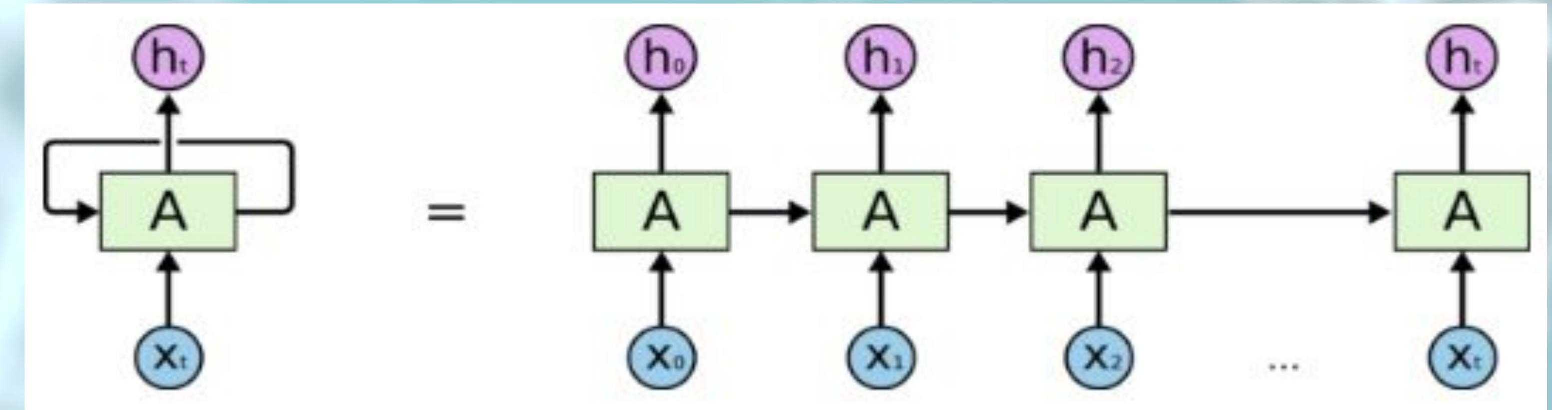
- Exploratory Data Analysis:
  - Summing up the daily cases
  - Normalizing the dataset
  - PCA
  - Make sequential data
- Seprate the data into train (90%) and test set (10%)
- Feeding data into Long Short Term Memory.
- Train the model
- Predict the value from trained model.
- Evaluating the model

FRAMEWORK


Data Collection
COVID-19 India Dataset
Data Preprocessing
Transforming data into supervised learning
Scaling data
Splitting dataset
Training Dataset
Testing Dataset
Model optimization & training
Hyperparameter Optimization
Keras-tuner
Forecast Strategies
Long short-term memory
Gated recurrent unit
Evaluating the models
i. Root mean square error (RMSE)
ii. Mean absolute error (MAE)
iii. Mean absolute percentage error (MAPE)

RECURRENT NEURAL NETWORK (RNN)

- **Recurrent Neural Network (RNN)** is a neural network model proposed in 80's for time series.
- The structure of the network is similar to feed forward neural network., with the distinction that it allows a recurrent hidden state whose activation at each time is dependent on that of the previous time (cycle).



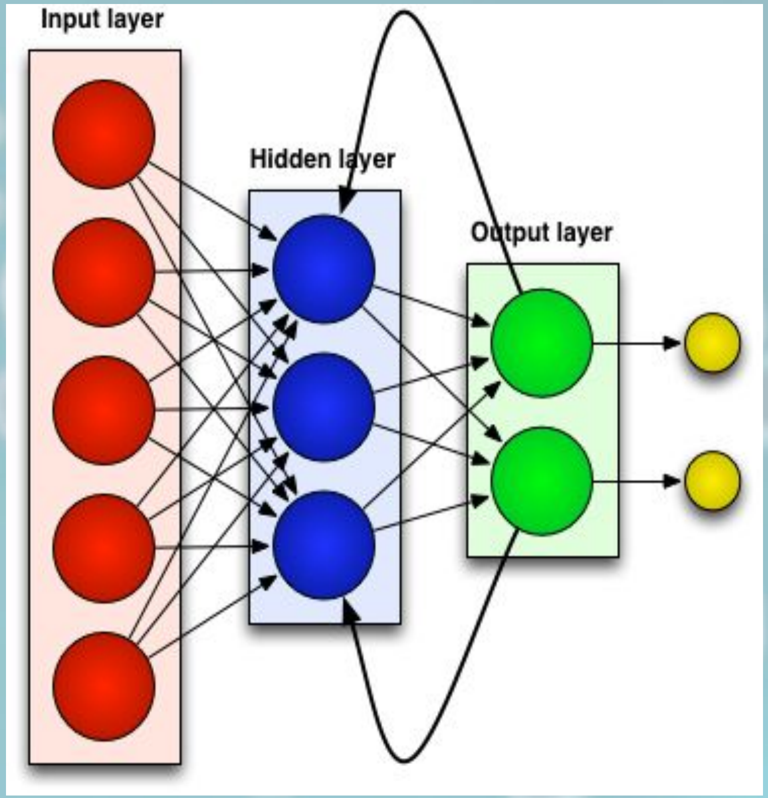
RECURRENT NEURAL NETWORK (RNN)

PROS:

1. **RNN** can model a sequence of data so that each sample can be assumed to be dependent on previous state**.**
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighbourhood.

CONS:

1. Gradient vanishing and exploding problems.
2. Cannot process long sequences.



LONG SHORT TERM MEMORY (LSTM)

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

LSTM use a series of 'gates' which control how the information in a sequence of data comes into, **is stored in and leaves the network**. There are three gates in a typical LSTM; forget gate, input gate and output gate. These gates can be thought of as filters and are each their own neural network.

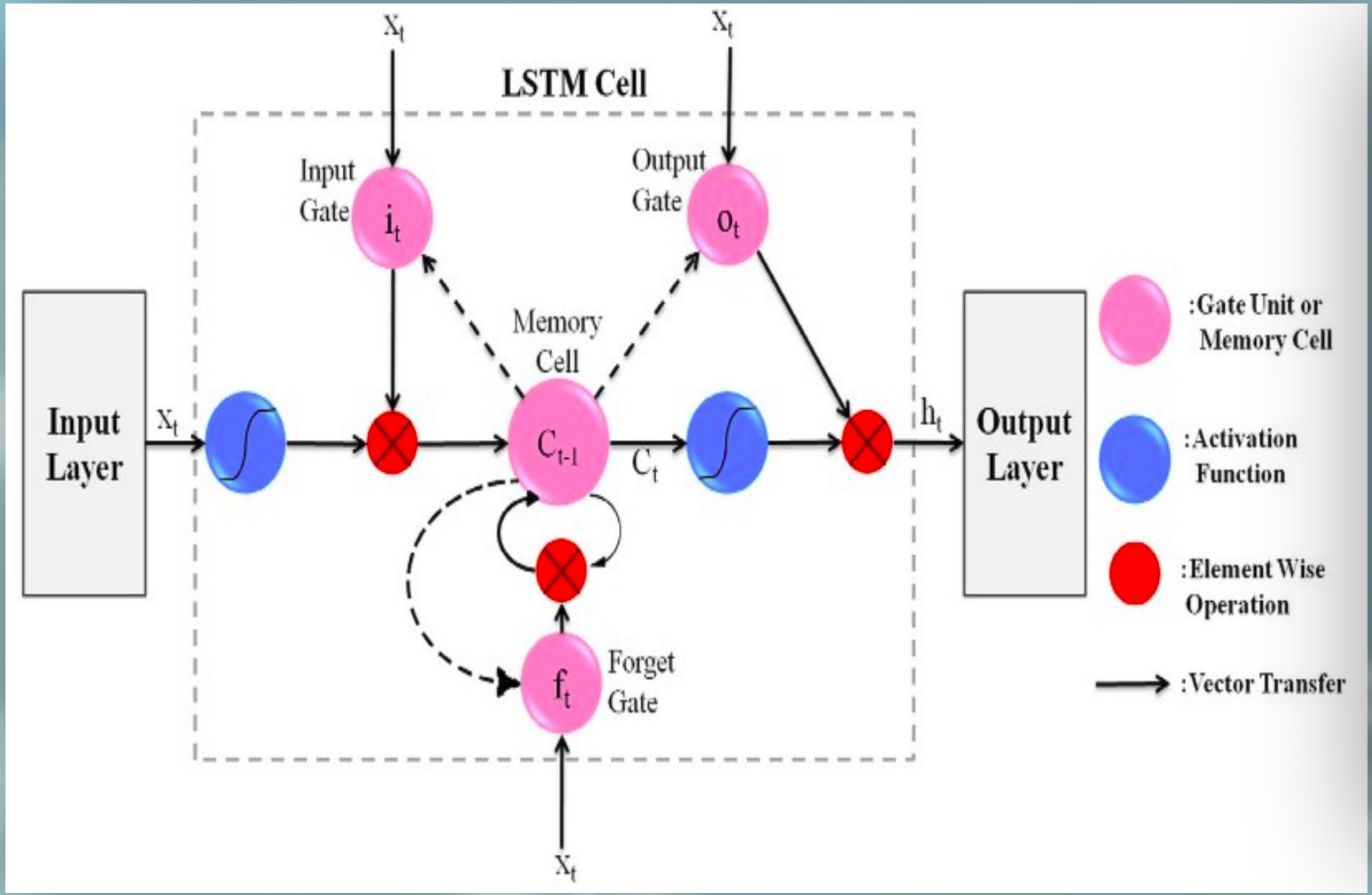
PROS :

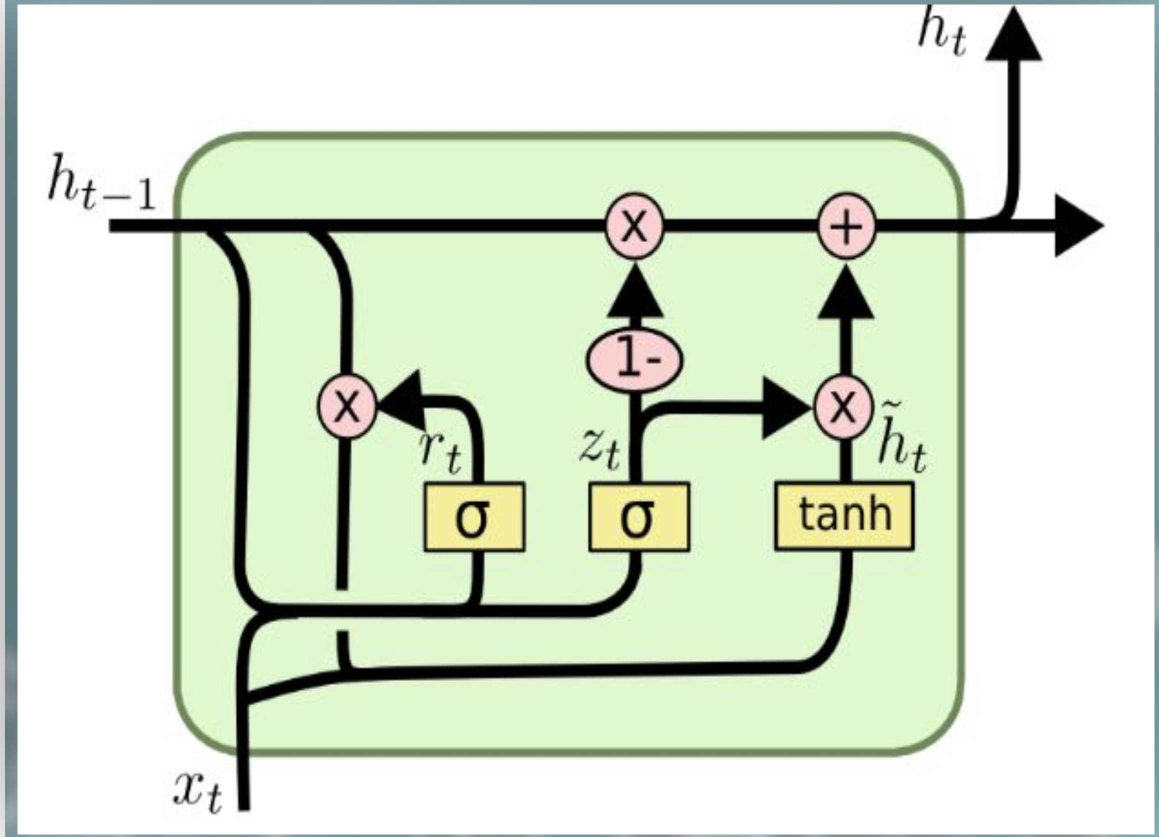
LSTM networks are **well-suited to classifying, processing and making predictions based on time series data**, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

CONS :

LSTM are **prone to overfitting and it is difficult to apply the dropout algorithm** to curb this issue. Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network.

LONG SHORT TERM MEMORY (LSTM)





LSTM Architecture

Mathematical Equation for LSTM

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

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

$$\overline{c}_t = \tanh(W_C \, . \, [h_{t-1} , x_t] + b_C)$$

$$C_t = f_t * C_{t-1} \ + i_t * \overline{c}_t$$

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

$$h_t = o_t * \tanh (C_t)$$

CONTINUED….

$$f_t = (W_f . [C_{t-1} , h_{t-1} , x_t] + b_f)$$

$$i_t = (W_i . [C_{t-1} , h_{t-1} , x_t] + b_i)$$

$$o_t = (W_o . [C_t , h_{t-1} , x_t] + b_o)$$

$$C_t = f_t * C_{t-1} + (1 - f_t) * \bar{c}_t$$

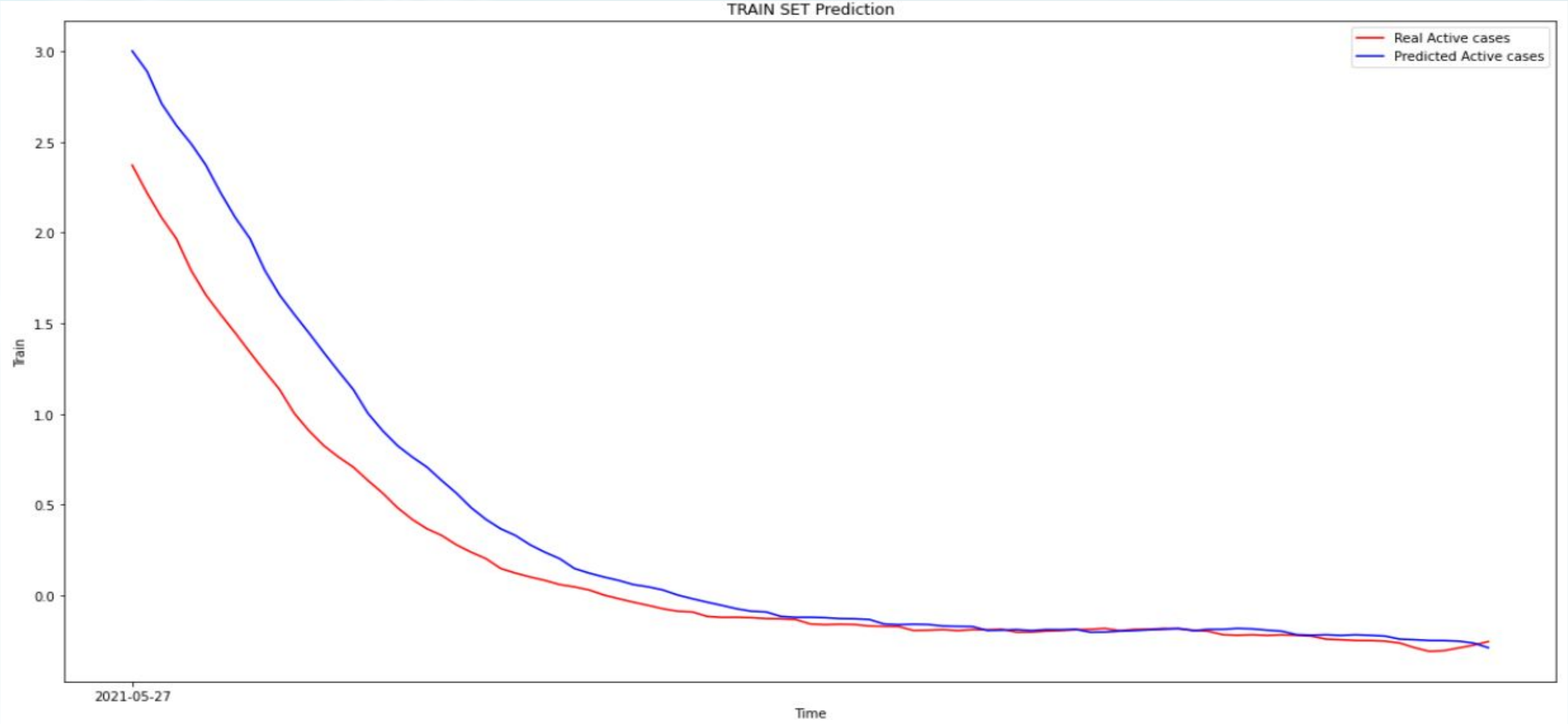
$$z_t = \sigma (Wz . [h_{t-1} , x_t])$$

$$r_t = \sigma (Wr . [h_{t-1} , x_t])$$

$$\tilde{h}_t = \tanh(W . [r_t * h_{t-1} , x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

RESULT



TRAIN SET Prediction
Real Active cases
Predicted Active cases
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Train
2021-05-27
Time

MODULES USED

Worked on: Kaggle Kernels

Modules to be used are:

1. numpy
2. Pandas
3. seaborn
4. Matplotlib
5. sklearn
6. Keras, sequential, Dense,LSTM
7. Keras, Dropout, Activation, Flatten

Implementation

Dataset

[3]:

```
df.tail(10)
```

[3]:

	date	new_cases	cum_cases	new_death	cum_death	new_recovered	cum_recovered	cum_active_cases
631	2021-10-22	16327	34158595	666	453152	17636	33524312	181131
632	2021-10-23	16079	34174674	559	453711	16509	33540821	180142
633	2021-10-24	14654	34189328	442	454153	18608	33559429	175746
634	2021-10-25	11852	34201180	357	454510	16102	33575531	171139
635	2021-10-26	13499	34214679	584	455094	14012	33589543	170042
636	2021-10-27	16351	34231030	734	455828	17077	33606620	168582
637	2021-10-28	14307	34245337	805	456633	13189	33619809	168895
638	2021-10-29	14215	34259552	551	457184	13549	33633358	169010
639	2021-10-30	12940	34272492	445	457629	14672	33648030	166833
640	2021-10-31	12907	34285399	251	457880	13152	33661182	166337

Normalizing the dataset

```
scaler = StandardScaler()
X = scaler.fit_transform(X)
dfx = pd.DataFrame(data=X,columns=df.columns[1:5])
dfx
```

	new_cases	cum_cases	new_death	cum_death
0	-0.682469	-1.033750	-0.730209	-1.079702
1	-0.682482	-1.033750	-0.730209	-1.079702
2	-0.682482	-1.033750	-0.730209	-1.079702
3	-0.682469	-1.033750	-0.730209	-1.079702
4	-0.682469	-1.033750	-0.730209	-1.079702
...	...	...	...	...
636	-0.473848	1.736234	0.020116	1.797852
637	-0.499929	1.737392	0.092696	1.802934
638	-0.501103	1.738542	-0.166954	1.806412
639	-0.517372	1.739590	-0.275312	1.809221
640	-0.517793	1.740634	-0.473627	1.810806

PCA Implementation:-

```
pd.DataFrame(pca.components_, columns = data.columns)
```

	Cum_Cases	Cum_Active Cases	Cum_Cured	Cum_Death	total_vaccinations	people_vaccinated	people_fully_vaccinated
0	0.406390	0.044745	0.408662	0.409468	0.409721	0.411045	0.40168
1	-0.407751	-0.336539	-0.374314	-0.350191	0.385698	0.378528	0.40705
2	-0.061240	0.931979	-0.176923	-0.227781	0.120486	0.118544	0.12613

```
n_pcs= pca.n_components_ # get number of component
# get the index of the most important feature on EACH component
most_important = [np.abs(pca.components_[i]).argmax() for i in range(n_pcs)]
initial_feature_names = data.columns
# get the most important feature names
most_important_names = [initial_feature_names[most_important[i]] for i in range(n_pcs)]
```

```
most_important_names
```

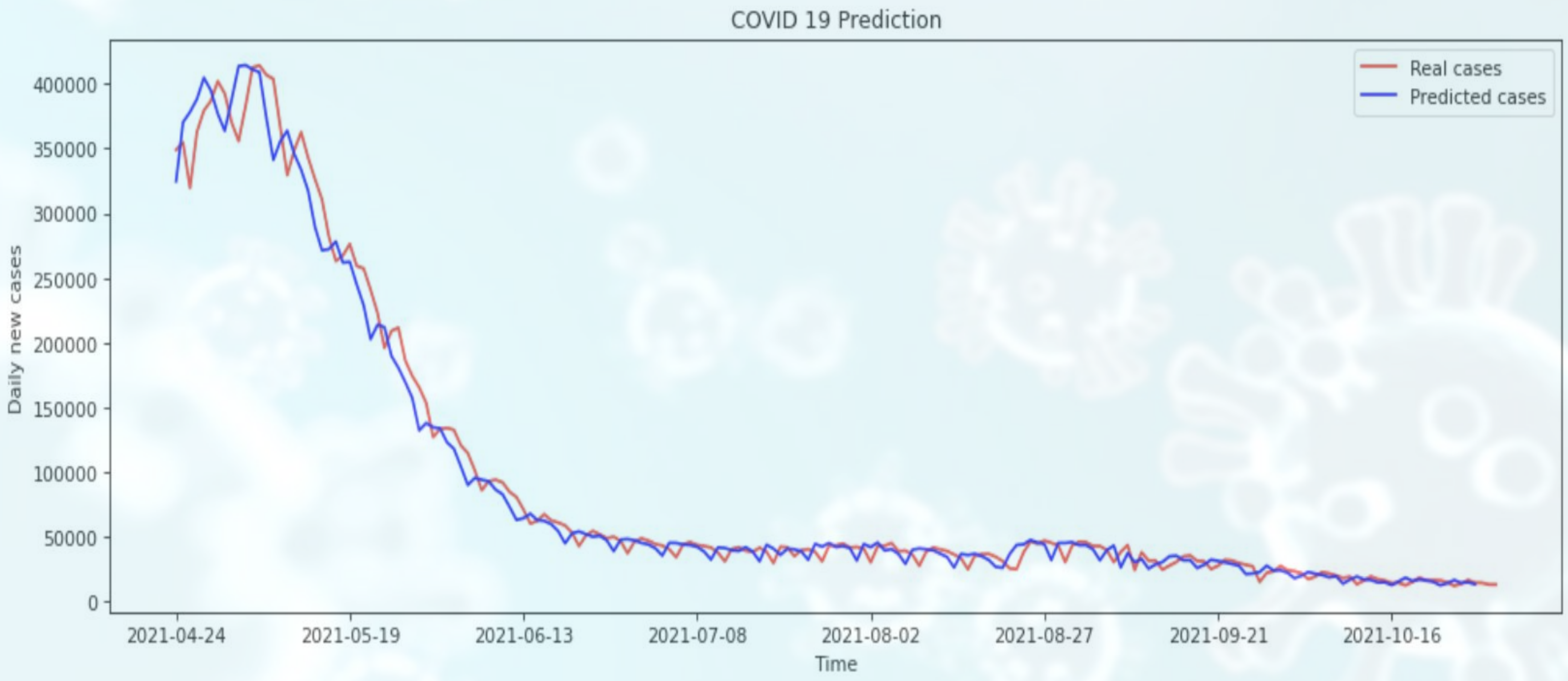
```
['people_vaccinated', 'Cum_Cases', 'Cum_Active Cases']
```

CODE

We have executed coding in Kaggle Platform kaggle

Kaggle Notebook: https://www.kaggle.com/mdahmadjami/time-series/notebook

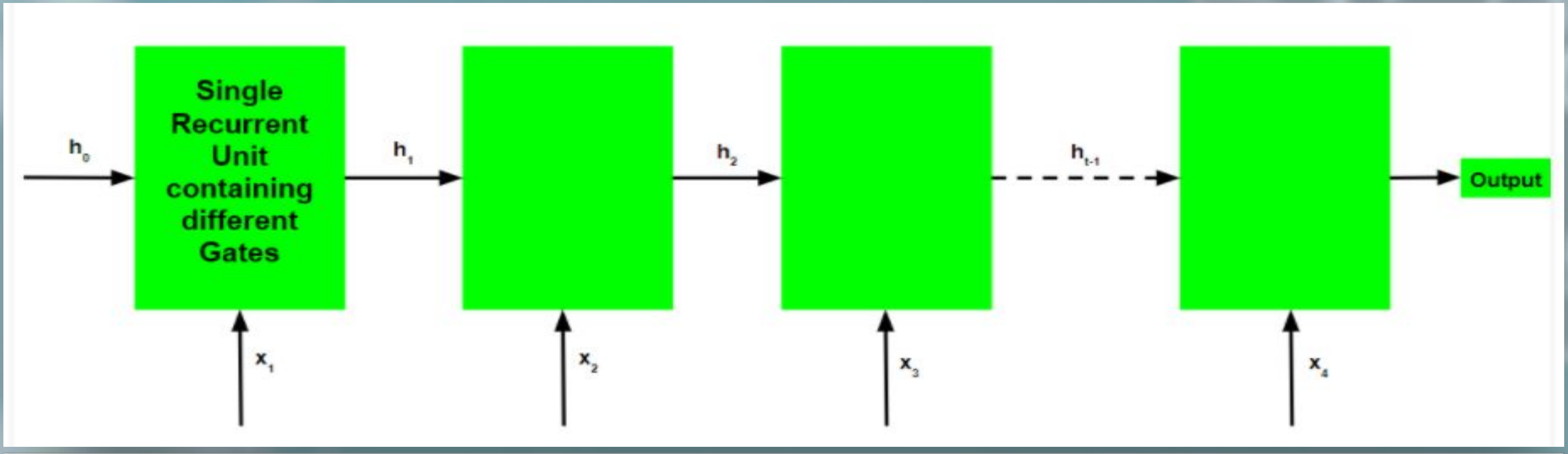
RESULT


COVID 19 Prediction
Real cases
Predicted cases
Daily new cases
400000
350000
300000
250000
200000
150000
100000
50000
0
2021-04-24
2021-05-19
2021-06-13
2021-07-08
2021-08-02
2021-08-27
2021-09-21
2021-10-16
Time

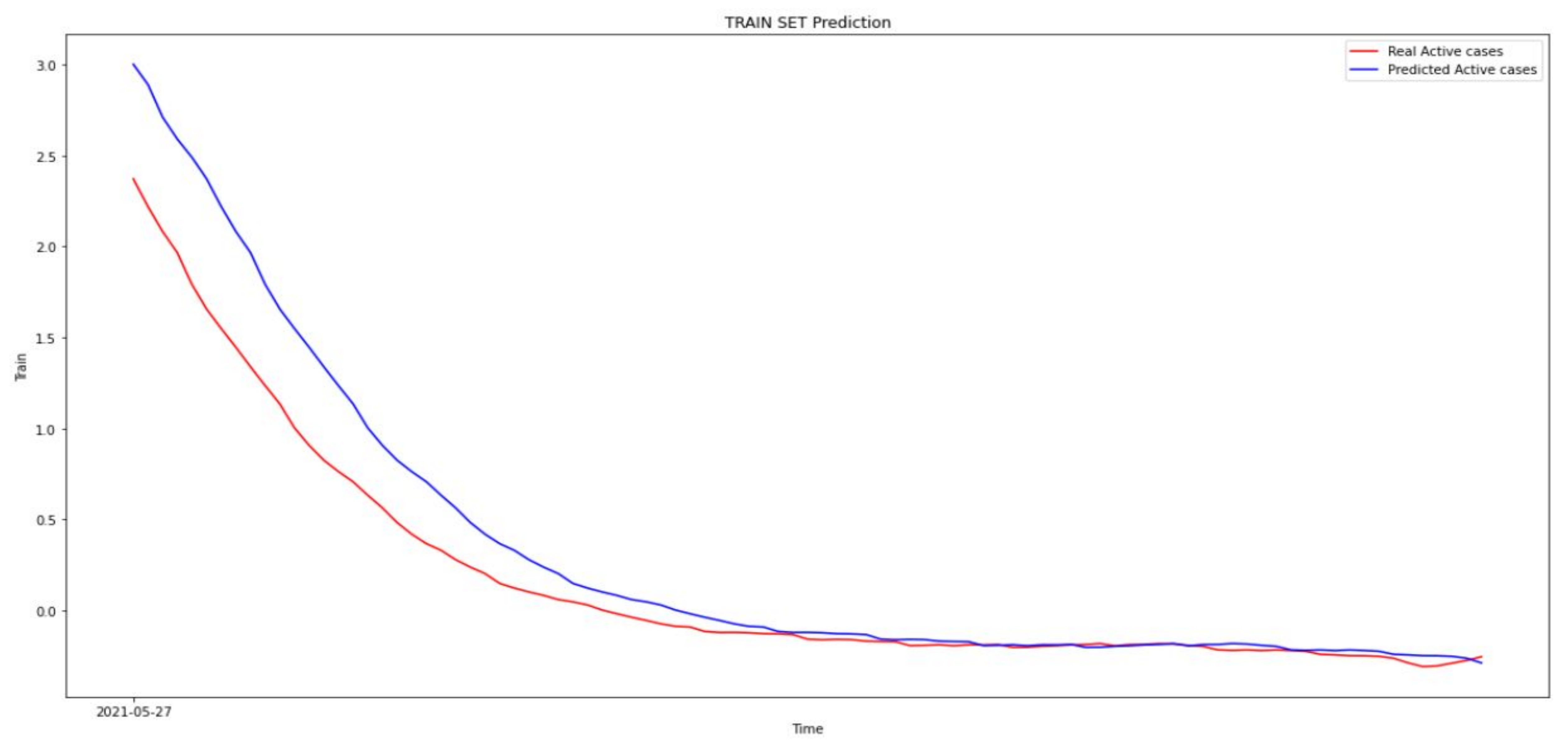
GATED RECURRENT UNIT (GRU)

A gated recurrent unit (GRU) is **part of a specific model of recurrent neural network that intends to use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering**, for instance, in speech recognition.

The basic workflow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist **of gates which modulate the current input and the previous.**



RESULT



TRAIN SET Prediction
Real Active cases
Predicted Active cases
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Train
2021-05-27
Time

COMPARISON

After Implementation of both the model, the **Error Values** are:

For LSTM = 0.27241103

FOR GRU = 0.28456110

The performance differences of these two deep learning models, involving two dimensions: dataset size for training, long/short text, and quantitative evaluation on five indicators including running speed, accuracy, recall, F1 value, and AUC. The corpus uses the datasets officially released by Yelp Inc. In terms of model training speed, GRU is 29.29% faster than LSTM for processing the same dataset; and in terms of performance, GRU performance will surpass LSTM in the scenario of long text and small dataset, and inferior to LSTM in other scenarios. Considering the two dimensions of both performance and computing power cost, the performance-cost ratio of GRU is higher than that of LSTM, which is 23.45%, 27.69%, and 26.95% higher in accuracy ratio, recall ratio, and F1 ratio respectively.

CONCLUSION

So, we can conclude that there was no spike in the predicted graph till the month November.

Thanks