A Report on Forecasting Value of Dengue Cases Using LSTM approach

Course Name: Big Data

Course Code: PM-ASDS19

Submitted to:

Dr. Md. Rezaul Karim

Associate Professor

Department of Statistics, JU

Submitted by:

Mohammad Saiduzzaman Sayed

ID: 20215063

Batch: 5th

Sec: A

Professional Masters in

Applied Statistics and Data Science (PM-ASDS)

JAHANGIRNAGAR UNIVERSITY

Forecasting Value of Dengue Cases Using LSTM approach

(Up To December 2022)

CONTENTS

LIST OF FIGURES

Abstract

We are concerning build a Time Series Forecasting model using the Long Short-Term Memory (LSTM) to forecast Dengue Cases up to December 2022.

We are using secondary data on Dengue cases. Our intention is to forecast dengue cases with the LSTM model. LSTM model is suitable for forecasting the time series model.

Introducing our neural network LSTM is a type of Recurrent Neural Network that can learn long-term dependencies. The model builds with hidden layers, each followed by a nonlinear ReLU activation function. A recent regularization technique, Dropout, is used to reduce overfitting. Before model fit, we analysis components of time series.

This procedure helps us to gather knowledge about how to handle time series, the deep learning method, time series components, LSTM, and steps to reach how to build a model and use it to predict.

Keyword: Dengue Cases, LSTM, Time series forecasting, tensorflow, keras

Chapter 1
Introduction

Dengue fever is a severe disease that has plagued tropical countries for many years. With the growing popularity of machine learning, it is becoming an important topic in research to find ways to use it to aid in the fight against dengue.

Several types of research have been conducted on the topic of dengue outbreak prediction. The recent explosion of various AI applications, particularly deep learning methods, has motivated data mining to new heights. The neural network design helps in capturing data's nonlinear characteristics. As a result, it is increasingly being used in forecasting.

Most of the work related to Dengue Cases involves machine learning approaches. I rarely find someone using LSTM with time series data like dengue cases forecasting. So, in this report, I will try to forecast up to December 2022.

Chapter 2
Methodology

Problem Statement:

To forecast Dengue cases up to December 2022, we indeed create a time series forecasting model using LSTM.

Dataset Details:

Description: The given dataset contains 731 observation and one feature which is Dengue Cases. The time range is 1st Jan 2020 to 31st Dec 2021.

Usage: Dengue_Cases

Flow of process:

This study investigated using deep learning to forecast the number of reported dengue cases. A statistical approach was also used to demonstrate the potential of deep learning models for forecasting, by fitting both non-seasonal and seasonal ARIMA models.

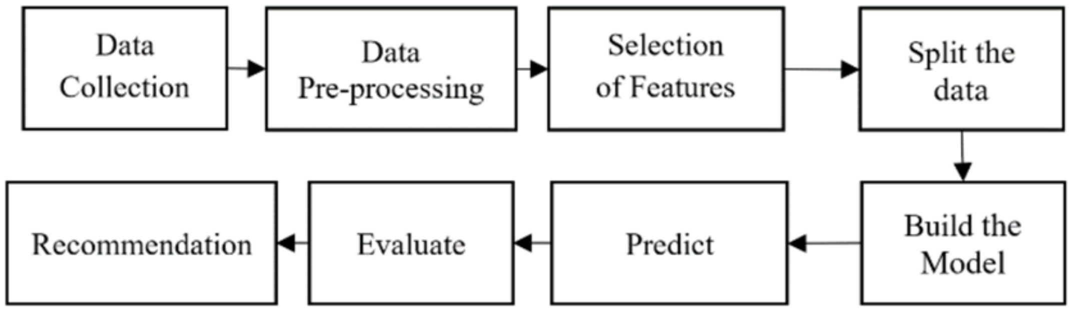


Fig 1: Flow of process for prediction of dengue fever

Data Collection:

Our data is secondary data. It has a time range from 1st January to 31st December 2021 for dengue cases. So, it’s time series data. Hence, there are 731 observations which means its daily data.

Data Pre-processing:

Before it can be used in the model, the dataset is pre-processed. The first procedure deals with missing data in the dataset. And estimate or remove outlier and high influential points. Then normalizing data to become unit free. After scaling, the data must be reshaped into a 3-dimensional matrix for the LSTM model with the appropriate time lag.

Selection of feature and splitting:

Our dataset has one feature which is Dengue Cases. We are using this feature to predict future. Our dataset has one feature which is Dengue Cases. We are using this feature to predict the future. Now, we split our data into the training set and test set. We build the model using training data and check accuracy with test data.

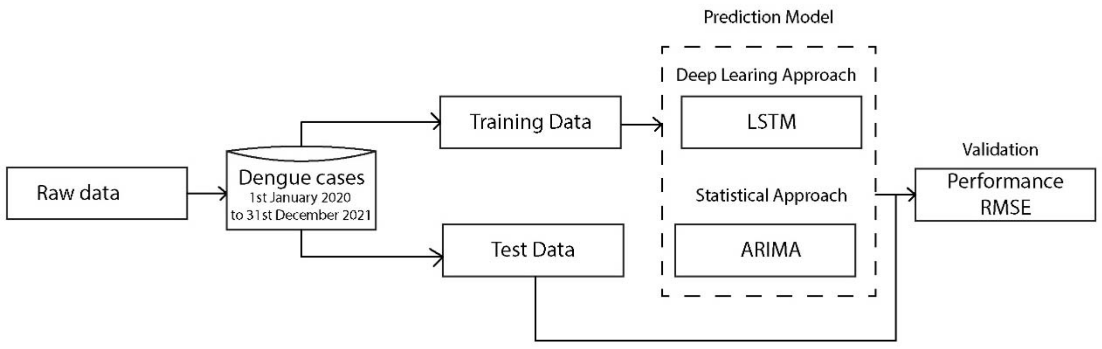
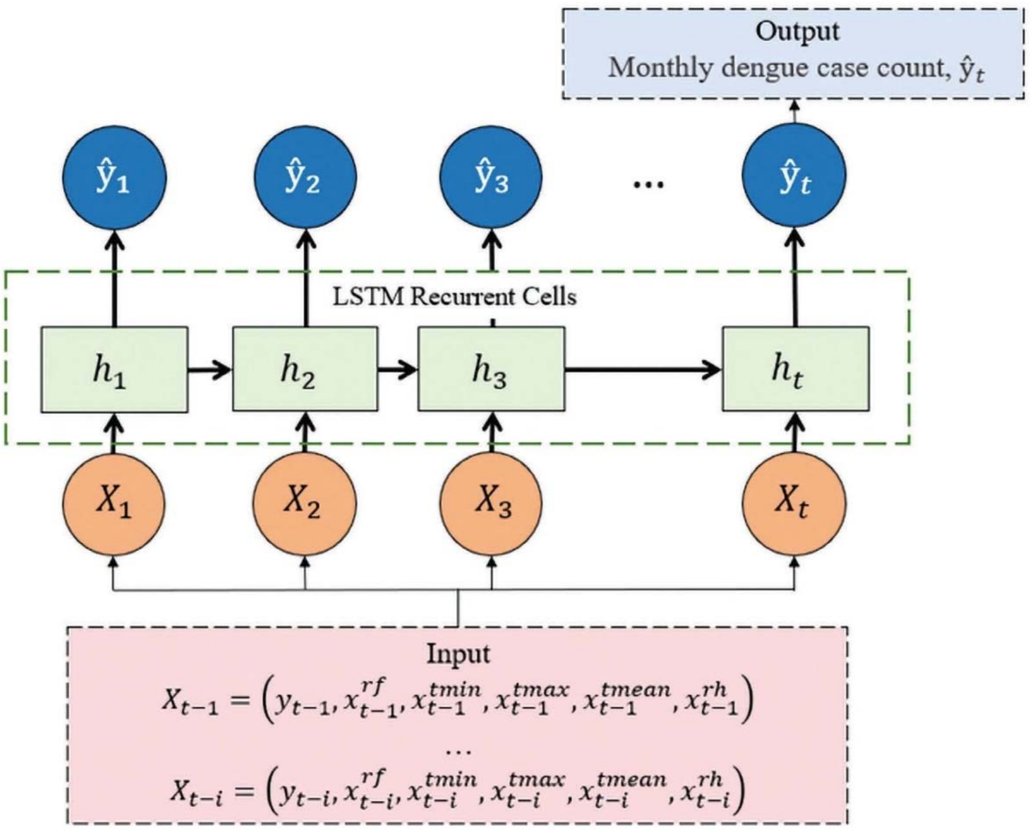


Fig 2: Summarized Workflow

LSTM Model Build and Evaluation:

The LSTM network model is an extension of the RNN architecture designed to have longer reference windows for sequence prediction. An LSTM layer consists of recurrently connected memory blocks that have a bigger capacity than RNNs. This is achieved by including a forget gate in an LSTM cell that allows the model to store relevant previous information selectively.



Output
Monthly dengue case count, $\hat{y}_t$
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
...
$\hat{y}_t$
LSTM Recurrent Cells
h_1
h_2
h_3
h_t
X_1
X_2
X_3
X_t
Input
$X_{t-1} = \left(y_{t-1}, x_{t-1}^{rf}, x_{t-1}^{tmin}, x_{t-1}^{tmax}, x_{t-1}^{tmean}, x_{t-1}^{rh}\right)$
...
$X_{t-i} = \left(y_{t-i}, x_{t-i}^{rf}, x_{t-i}^{tmin}, x_{t-i}^{tmax}, x_{t-i}^{tmean}, x_{t-i}^{rh}\right)$

Fig 3: The architecture for forecasting using the LSTM network

One way to evaluate forecasting model by calculating the commonly used forecasting accuracy measurement is Root Mean Squared Error (RMSE). The equation of RMSE shown below, where n is the total of data, y_t is actual value, and y_t' is predicted value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_t - y_t')^2}{n}}$$

Compactional Tools:

This whole project computing is done in Google Collaboratory and using python 3.9. The packages we used in the project are TensorFlow, Keras, Matplot, and Numpy.

Chapter 3
Data Analysis and Result

The LSTM-RNN-based architecture for forecasting dengue cases and deaths is formed of a hidden layer called the LSTM. The layer contains 64 memory cells for computing results, and the values obtained are transferred to the next iteration. Since the data is available for a period of one year, the time step of LSTM is set to a value of 12. The activation function used in the proposed methodology is Rectified Linear Unit (ReLU). The performance of ReLU is comparatively good that the other activation functions like sigmoid and tanh.

Visualizations:

Time plot:

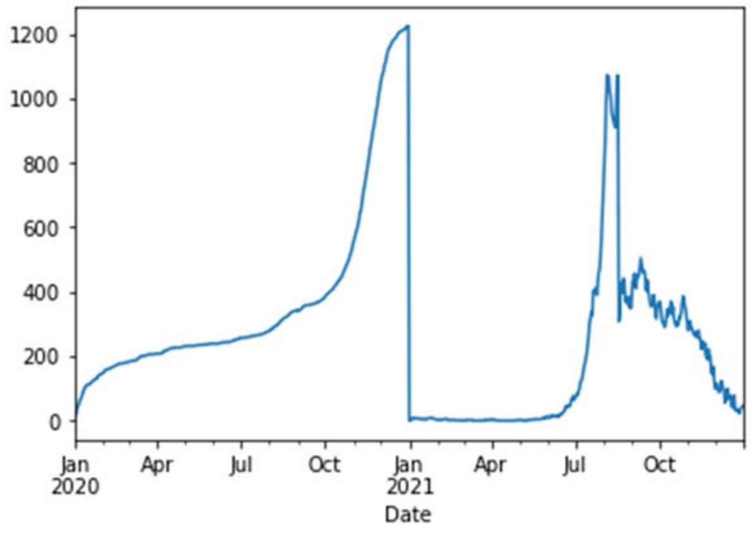


Fig 4: Time Plot

The above time plot shows that there were outliers and sharp changes in our data. This is the daily data, so we assume that the seasonality is present. And we see the upward trend and downward trend in the data.

Decomposition:

The below decomposition plot shows that clear trend and seasonality in our data. And we see the fluctuation in the observed data.

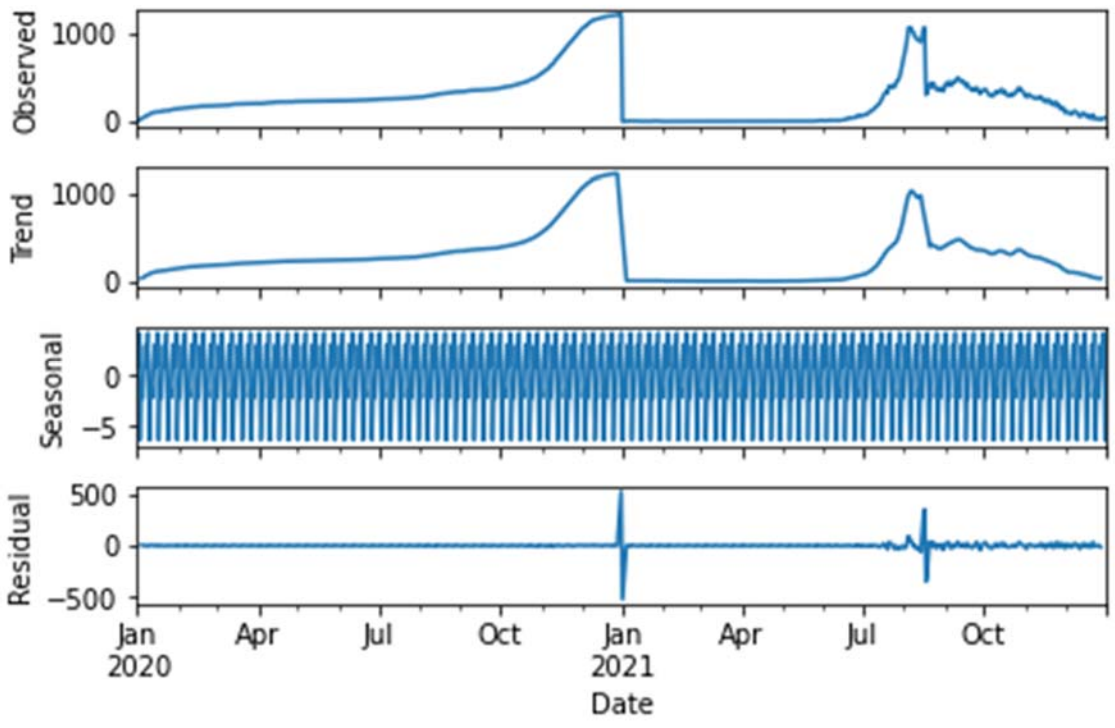


Fig 5: Decomposition Plot

Summary of Model:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)               Output Shape              Param #
=================================================================
lstm_4 (LSTM)              (None, 12, 150)           91200
lstm_5 (LSTM)              (None, 64)                55040
dense_3 (Dense)            (None, 64)                4160
dropout (Dropout)          (None, 64)                0
dense_4 (Dense)            (None, 1)                 65
=================================================================
Total params: 150,465
Trainable params: 150,465
Non-trainable params: 0
_________________________________________________________________
```

Layers details:

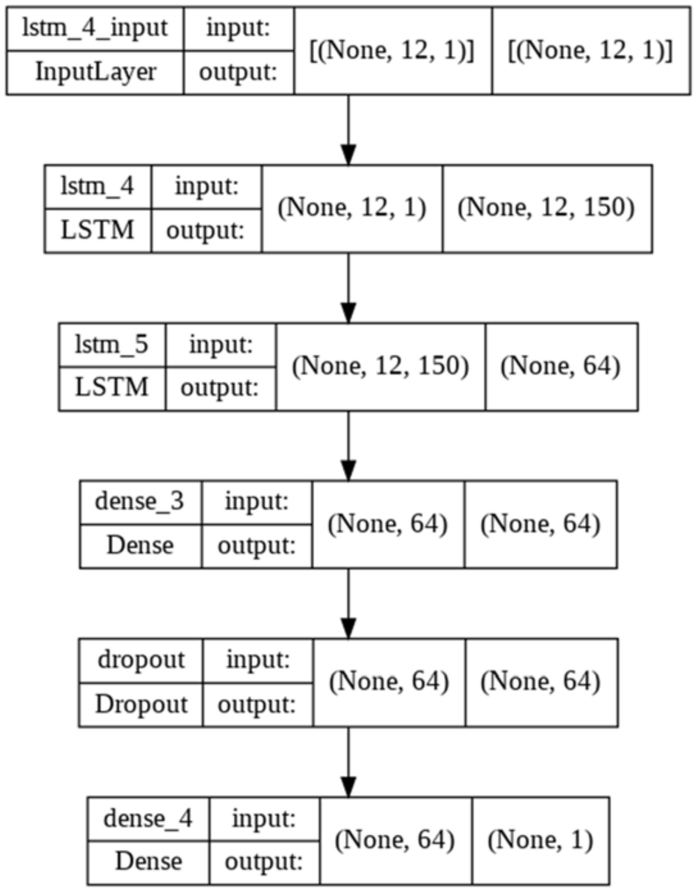

lstm_4_input | input: | [(None, 12, 1)] | [(None, 12, 1)]
InputLayer | output:
lstm_4 | input: | (None, 12, 1) | (None, 12, 150)
LSTM | output:
lstm_5 | input: | (None, 12, 150) | (None, 64)
LSTM | output:
dense_3 | input: | (None, 64) | (None, 64)
Dense | output:
dropout | input: | (None, 64) | (None, 64)
Dropout | output:
dense_4 | input: | (None, 64) | (None, 1)
Dense | output:

Fig 6: Layers

Prediction:

The below plot shows the original cases vs predictions with test data.

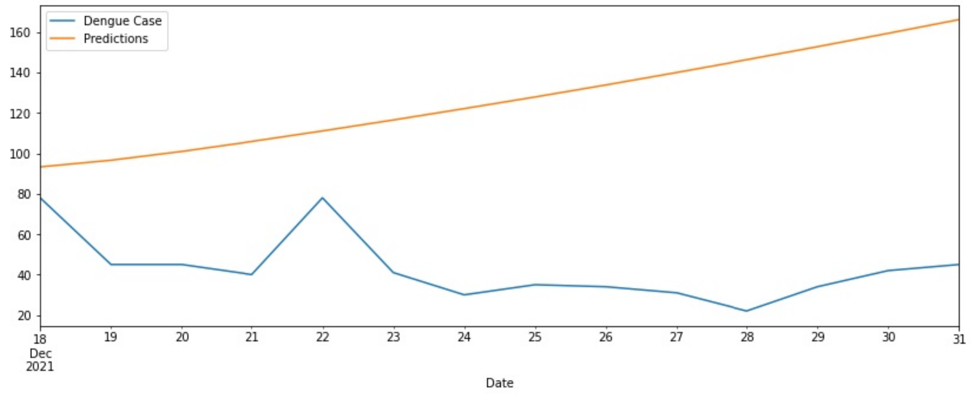


Fig 7: Original vs Predictions

Forecasting:

The below plot shows forecasting 1 year ahead of dengue cases using previous records.

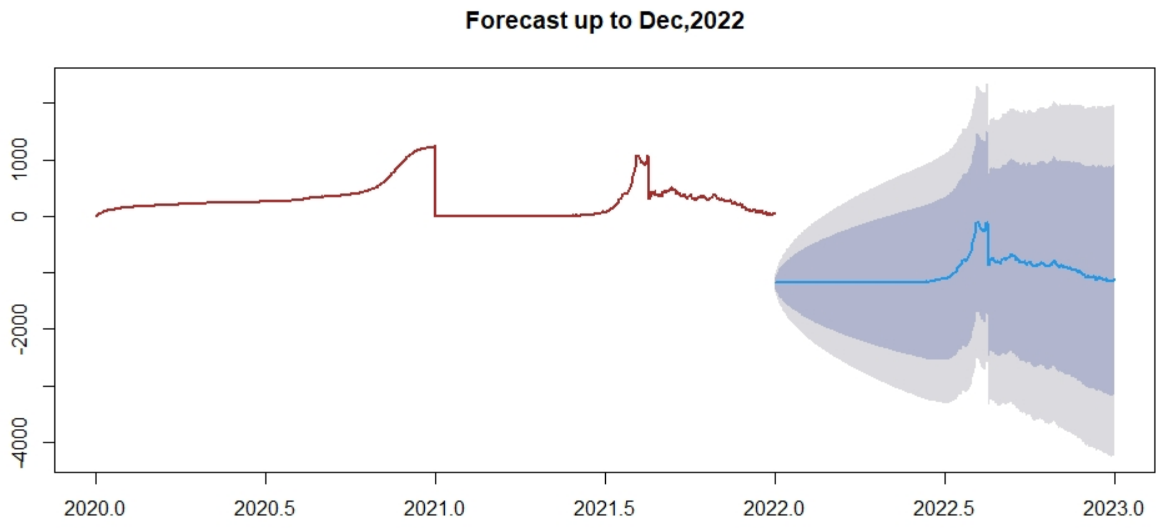


Fig 8: Forecasting Dengue Cases up to Dec,2022

Chapter 4

Conclusions

This study used a deep learning approach to forecast the total number of reported dengue cases for the following year. The ability of the LSTM model to retain information from a long sequence of data has been illustrated.

The study found that using the deep learning approach resulted in a significant increase in forecast accuracy; However, there is still much space for improvement.

Other model hyperparameters, such as increasing the number of epochs, changing the learning rate, and considering other optimizers, could be used to further investigate the LSTM model architecture. Other sequence model networks, such as gated recurrent units, another RNN network extension, or transformer models, could also be investigated. On the basis of an optimized model, the use of transfer learning can be investigated to improve forecast accuracy in other regions with fewer dengue cases.

References

Olah, C. (2015). Understanding LSTM Networks. Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs

Lestari,A.N. Tyasnurita,R., Vinarti,A.R., Smola,J.A., Anggraeni, W. (2022). Long Short-Term Memory forecasting model for dengue fever cases in Malang regency, Indonesia.
Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050921023553?ref=pdf_download&fr=RR-2&rr=72d5371c9f5878bb

Nadda,W., Boonchien,W., Boonchieng,E. (2022). Influenza, dengue and common cold detection using LSTM with fully connected neural network and keywords selection. Retrieved from https://biodatamining.biomedcentral.com/articles/10.1186/s13040-022-00288-9

Appendix

Python Code of LSTM approach

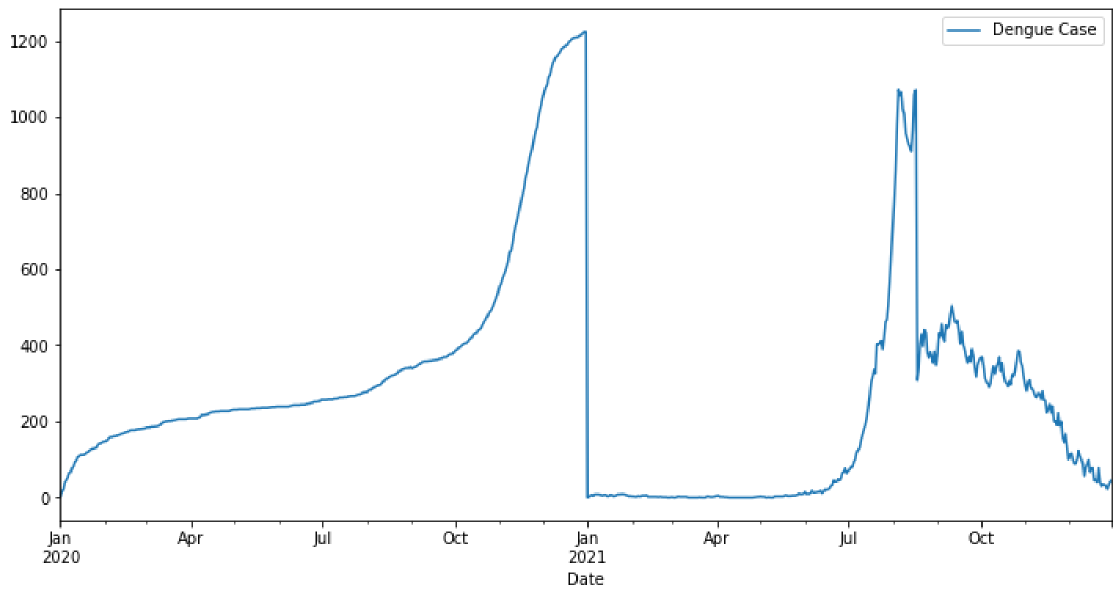
```
[1]: # Library
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[164]: # import dataset
       df=pd.read_csv("/content/Dengue_case.csv", index_col='Date', parse_dates=True)
       df
```

```
[164]:             Dengue Case
       Date
       2020-01-01            3
       2020-01-02           17
       2020-01-03           20
       2020-01-04           39
       2020-01-05           45
       …                  …
       2021-12-27           31
       2021-12-28           22
       2021-12-29           34
       2021-12-30           42
       2021-12-31           45

       [731 rows x 1 columns]
```

```
[166]: # plot
       df.plot(figsize=(12,6))
```

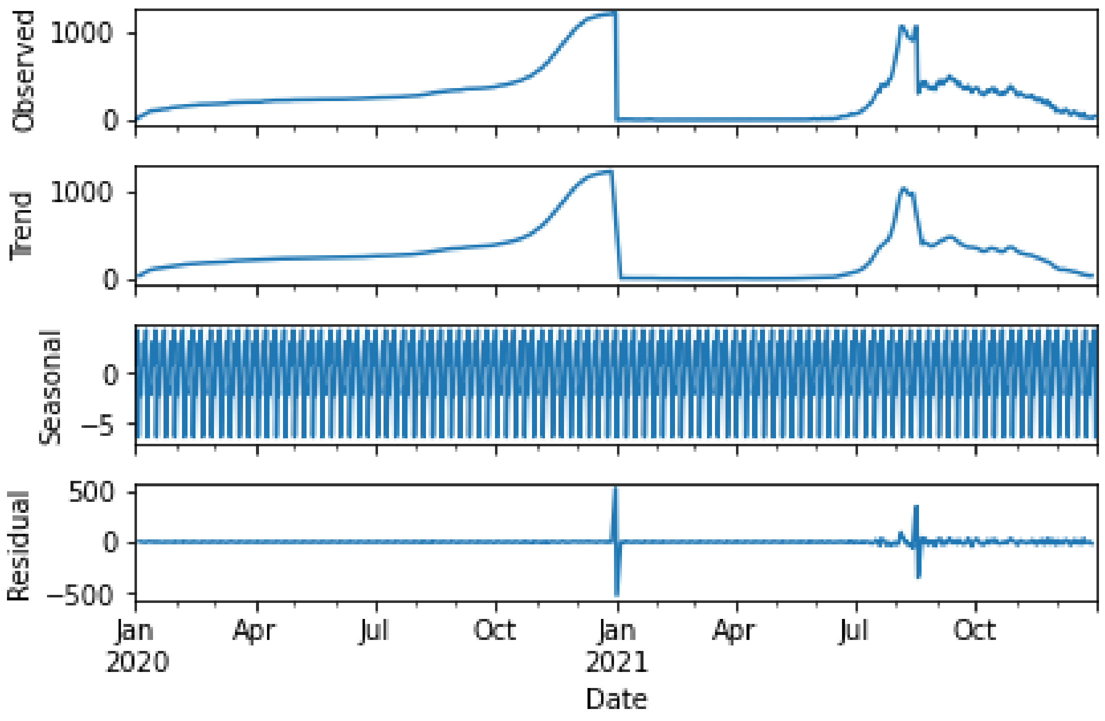
```
[166]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f8bee4ad0>
```



```
[167]: from statsmodels.tsa.seasonal import seasonal_decompose
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
[171]: # Decomposition
       results = seasonal_decompose(df['Dengue Case'])
       results.plot();
```



```
[174]: # splitting
       train = df.iloc[:-14]
       test = df.iloc[-14:]
```

```
[175]: # scalling
       from sklearn.preprocessing import MinMaxScaler
       scaler = MinMaxScaler()
       scaler.fit(train)
       scaled_train = scaler.transform(train)
       scaled_test = scaler.transform(test)
```

```
[179]: # define generator
       from keras.preprocessing.sequence import TimeseriesGenerator
       n_input = 3
       n_features = 1
       generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,␣
        ↪batch_size=1)
       X,y = generator[0]
       print(f'Given the Array: \n{X.flatten()}')
       print(f'Predict this y: \n {y}')
```

```
[181]: # We do the same thing, but now instead for 12 months
       n_input = 12
```

```
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,␣
↪batch_size=1)
```

[182]:
```
# Build Model
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

#Define Model
model = Sequential()
model.add(LSTM(150, activation='relu', return_sequences=True,␣
↪input_shape=(n_input, n_features)))
model.add(LSTM(64, activation='relu'))
model.add(Dense(64))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=["accuracy"])

model.summary()
```

```
Model: "sequential_5"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_8 (LSTM)               (None, 12, 150)           91200

 lstm_9 (LSTM)               (None, 64)                55040

 dense_8 (Dense)             (None, 64)                4160

 dense_9 (Dense)             (None, 1)                 65

=================================================================
Total params: 150,465
Trainable params: 150,465
Non-trainable params: 0
_________________________________________________________________
```

[183]:
```
model.fit_generator(generator,epochs=5)
```

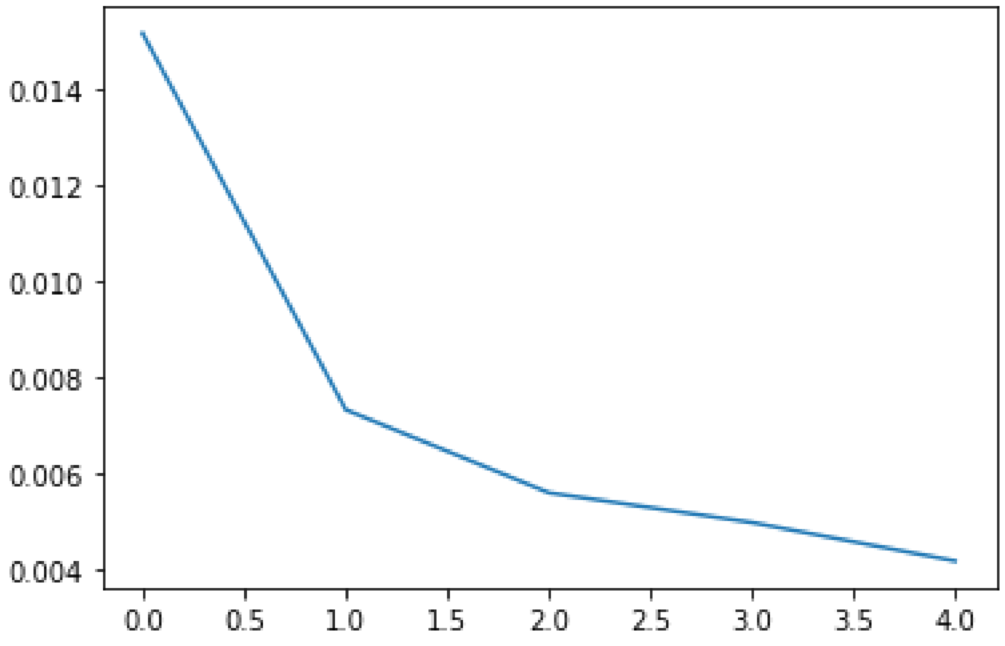
```
Epoch 1/5

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.

705/705 [==============================] - 13s 14ms/step - loss: 0.0152 -
accuracy: 0.0582
```

```
Epoch 2/5
705/705 [==============================] - 12s 16ms/step - loss: 0.0073 -
accuracy: 0.0582
Epoch 3/5
705/705 [==============================] - 10s 14ms/step - loss: 0.0056 -
accuracy: 0.0582
Epoch 4/5
705/705 [==============================] - 10s 14ms/step - loss: 0.0050 -
accuracy: 0.0582
Epoch 5/5
705/705 [==============================] - 10s 14ms/step - loss: 0.0042 -
accuracy: 0.0582
```

```
[183]: <keras.callbacks.History at 0x7f1f8ee2c2d0>
```

```
[184]: loss_per_epoch = model.history.history['loss']
       plt.plot(range(len(loss_per_epoch)),loss_per_epoch)
```

```
[184]: [<matplotlib.lines.Line2D at 0x7f1f8e9579d0>]
```



```
[185]: last_train_batch = scaled_train[-12:]
       last_train_batch = last_train_batch.reshape((1, n_input, n_features))
       model.predict(last_train_batch)
       scaled_test[0]
```

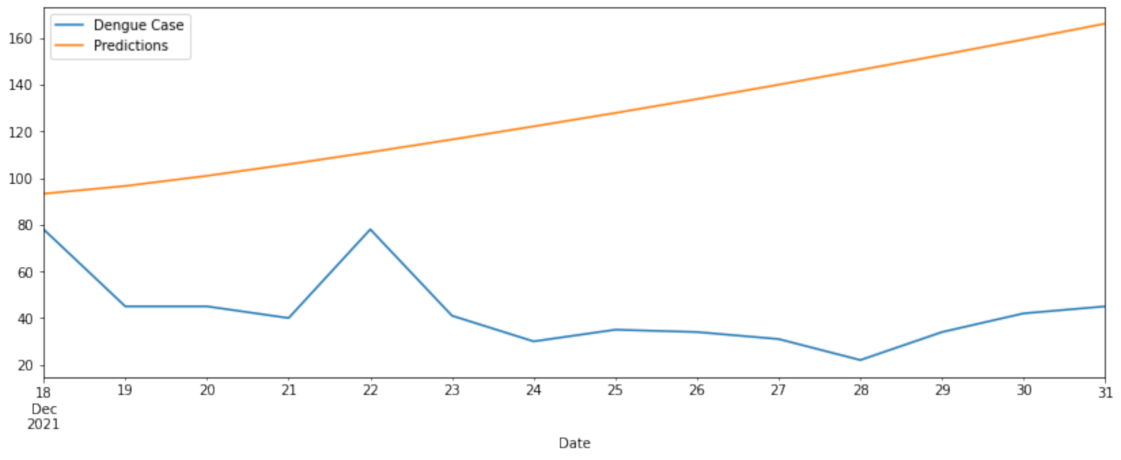
```
[185]: array([0.06372549])
```

[186]:
```python
test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):

    # get the prediction value for the first batch
    current_pred = model.predict(current_batch)[0]

    # append the prediction into the array
    test_predictions.append(current_pred)

    # use the prediction to update the batch and remove the first value
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

[187]:
```python
true_predictions = scaler.inverse_transform(test_predictions)
test['Predictions'] = true_predictions
test.plot(figsize=(14,5))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
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/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[187]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1f8f1a9a50>
```



[195]:
```python
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse=sqrt(mean_squared_error(test['Dengue Case'],test['Predictions']))
print("Root Mean Square Error", rmse)
```

```
Root Mean Square Error 90.285209904657
```

THE END