Global Radiation Prediction Model using Multi-variant Time Series Forecasting

Report submitted in fulfillment of the requirements for the Undergraduate Project(6th Semester)

Third Year IDD

by

Pranav Dev

20124037

Under the guidance of

Prof. S. K. Pandey

May 2023

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written material from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving the required details in the references.

Place: IIT(BHU) Varanasi)

Date: 3rd May 2023

Pranav Dev

IDD (Part III) Student

Department of Mathematical Sciences

Indian Institute of Technology (BHU)

Varanasi, INDIA 221005

<u>Certificate</u>

*This is to certify that the work contained in this report entitled "**Global radiation prediction model using multi-variant time series forecasting**", being submitted by **Pranav Dev** (Roll no. **20124037**), carried out in the Department of Mathematical Sciences, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT(BHU) Varanasi)
Date: 3rd May 2023

Prof. S. K. Pandey
Professor
Department of Mathematical Sciences
Indian Institute of Technology (BHU)
Varanasi, INDIA 221005

<u>Acknowledgments</u>

I am writing to express my sincere gratitude to my supervising **Professor**, **S.K. Pandey**, for his constant guidance and support during the whole project work.

Place: IIT(BHU) Varanasi)

Date: 3rd May 2023 **Pranav Dev**

Abstract

Nonetheless, it's a little late, people are starting to realize how bad fossil fuels are for the environment. Humans are turning to renewable energy in the hope of a cleaner and less pollutant environment. Thankfully, the sun is a reliable source of clean energy.

Solar energy is the radiant light and heat from the Sun and can be harnessed using a range of technologies. Predicting the attributes associated with solar radiation has become an important element in producing solar energy these days.

In this machine learning model, I will predict the different global radiation parameters such as Clearsky DHI, Clearsky DNI, and Clearsky GHI with the help of multivariate time series forecasting using Long Short Term Memory(LSTM) as Recurrent Neural Network(RNN) layers. Mean Absolute Error and Huber loss will be used to predict the accuracy of the model.

Contents

Chapter 1

Neural Networks

1.1. Brief on Neural Networks

A neural network contains layers of interconnected nodes. Each node is known as a perceptron and is similar to multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

A neural network generally consists of layers, each consisting of multiple nodes. When a neural network has many layers, then that neural network is said to be a deep neural network. Deep neural networks have been able to predict more accurately than the neural network having fewer layers.

The layers of the neural network can be classified into three categories:

- Input Layer: This neural network layer accepts the data and passes it on to the next layer for processing.
- Hidden Layer: This layer of the neural network is responsible for processing the input data received from the input layer, making the predictions using one or many layers, and passing the final predicted values to the next layer, which is the output layer.
- Output Layer: This neural network layer is responsible for passing the predicted values out of the model.

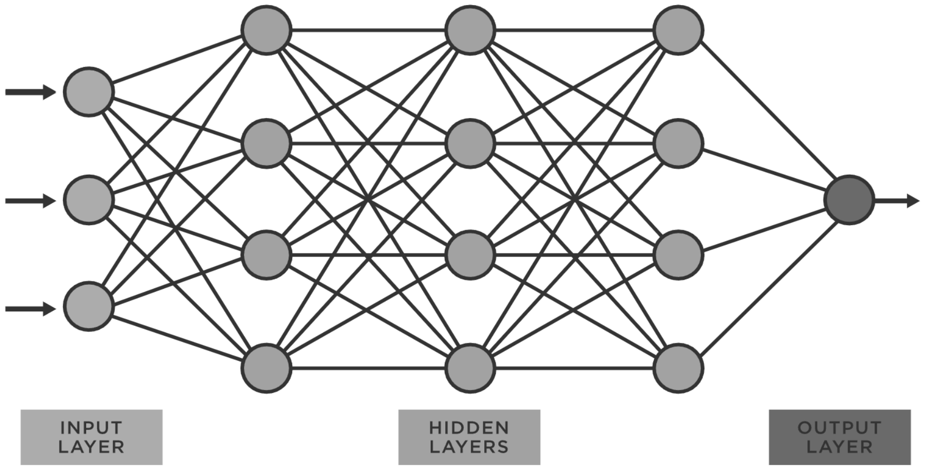


Fig 1.1.1 A simple diagram of a neural network indicating input layer, hidden layers and output layers

1.2. Loss Functions

In machine learning, a loss function is a measure of how accurately a model is able to predict the expected outcome, in other words, the ground truth.

The loss function will take two items as input:

- The output value of the model
- The ground truth expected value

A low value for the loss indicates that the model performed well, whereas a high value for the loss indicates that the model performed poorly. For training the model accurately, selecting a loss function that will be accurate for the model is crucial.

The few of the most common loss functions are:

1.2.1. Mean Squared Error (MSE):

Mean Squared Error (MSE) is the most common and straightforward loss function. For calculating the MSE, one should take the difference between the model's predictions and the ground truth, square it, and average it across the whole dataset.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

1.2.2. Mean Absolute Error (MAE):

For calculating the Mean Absolute Error (MAE), one should take the difference between the model's predictions and the ground truth, apply the absolute value to that difference, and then average it out across the whole dataset.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

1.2.3. Huber Loss:

We saw that the MSE is great for learning outliers while the MAE is great for ignoring them; the Huber loss provides a balance to both errors.
The Huber loss is defined using the following piece-wise function:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

When the loss values are greater than 1, they are magnified; the quadratic function is used to down-weigh them to focus the training on the higher-error data points.

Chapter 2

Sequence Models

Sequence Models are a very powerful tool when we are dealing with time series data; as the predicting of any parameter depends on the data of the previous events, we will extensively use sequence models in our machine learning model.

Sequence Models are machine learning models that input or output a sequence of data. Sequential data may include time series data, text streams, audio clips, video clips, etc.

The sequence models can be classified into:

- Recurrent Neural Network (RNN)
- Gated Recurrent Unit (GRU)
- Long Short Term Memory (LSTM)

2.1. Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is a deep learning algorithm that is specialized for processing sequential data. RNN works on the recurrence principle, in which the internal state at a given time step is dependent on the internal step at the previous time step and the sequence we are working on.

RNNs maintain internal memory; due to this, they are very efficient for machine learning problems that involve sequential data. This feature helps in sharing the weights across time, and RNNs are also able to remember the previous inputs, which the standard neural networks were not able to.

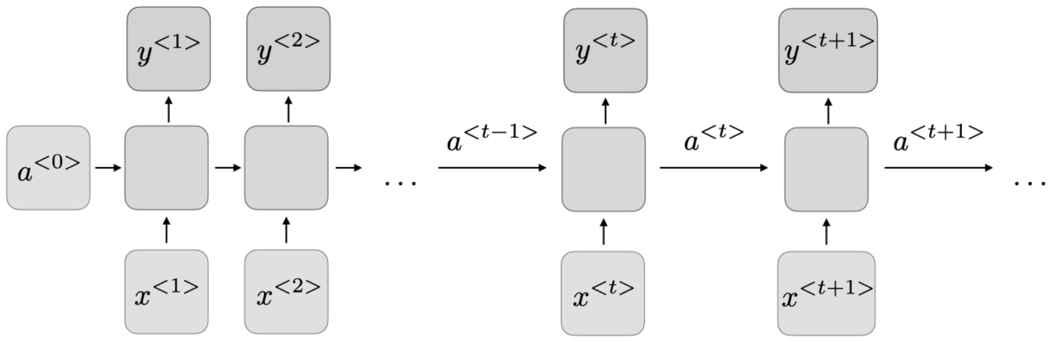


Fig 2.1.1 A simple diagram of a Recurrent neural network (RNN)

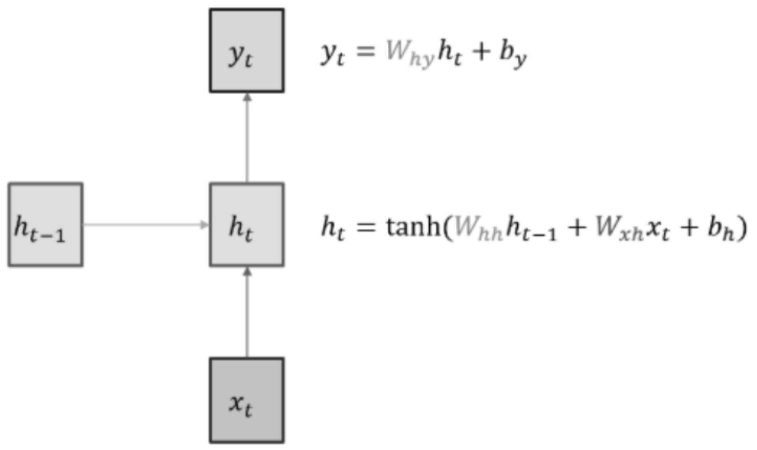


Fig 2.1.2 Working of a simple Recurrent neural network (RNN)

Even though RNNs are able to maintain internal memory, they are short-termed only. One of the drawbacks of RNNs is that they are not able to handle long-term dependencies, and problems such as vanishing gradients or exploding gradients may occur.

2.2. Gated Recurrent Unit (GRU)

To solve problems like vanishing gradients, and long-term dependencies handling, we can use an improved version of RNN, known as Gated Recurrent Unit (GRU).

GRU uses two gates, known as the update gate and the reset gate, two vectors that decide what information should be passed on to the output.

- The *update gate* helps the model determine how much information from the previous time steps needs to be passed on to the current step.
- The *reset gate* helps the model to determine how much of the previous time steps' information should be forgotten.

The vanishing gradient problem is eliminated as the model avoids washing out the new input every single time, keeps the relevant information, and passes it down to the next time steps of the network.

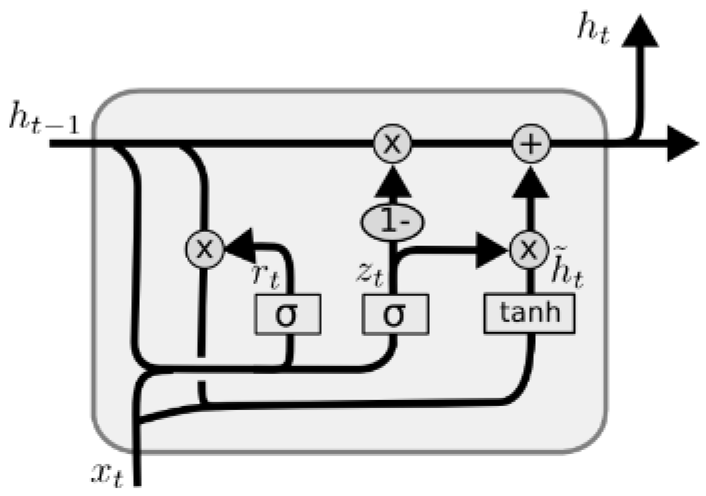


Fig 2.2.1 A simple diagram of a Gated Recurrent Unit (GRU)

The formulas used to process the data in GRUs are:

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

2.3. Long Short-Term Memory (LSTM)

Long Short Term Memory (LSTM) is yet another sequence model which overcome the problems faced by the standard RNNs like short-term dependency, vanishing gradient, etc.

LSTM uses three gates, namely the forget gate, the input gate, and the output gate, to filter out the relevant information that need to be used to process the current time step.

- The *forget gate* decides which units of the current long memory of the network, known as cell state, are useful given both the previous hidden state and new input state.
- The *input gate* decides what new information should be added to the cell state, given both the previous hidden state and new input state.
- The *output gate* decides the new hidden state to go to given the updated cell state.

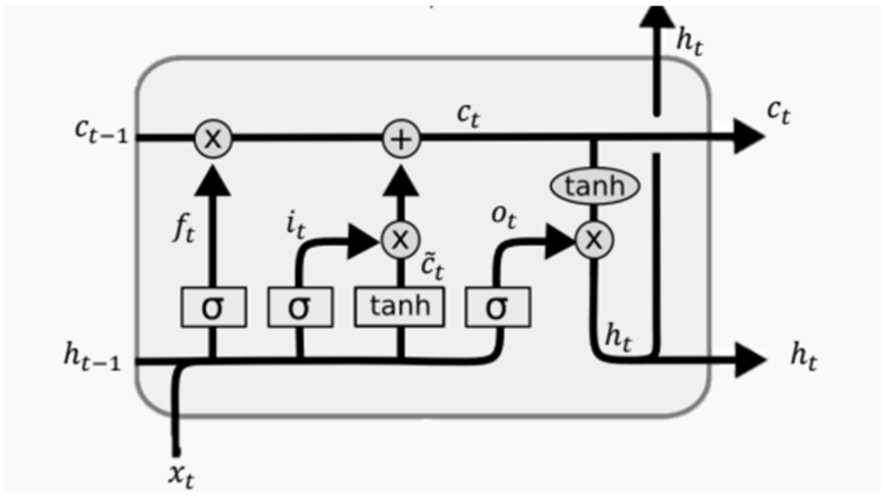


Fig 2.3.1 A simple diagram of a Long Short Term Memory (LSTM)

The formulas used to process the data in LSTMs are:

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}$$

Chapter 3

Time Series Forecasting

A time series is simply a series of data points ordered in time. Time series forecasting refers to the prediction of a time series using the information available of the previous time series.

The basic terminology used when referring to time series are:

- *Seasonality* refers to the distinct periods of time when the data contains consistent irregularities.
- *Trends* indicate whether a variable in the time series will increase or decrease in a given period.
- *Stationarity* refers to a time series whose statistical properties do not change over time. In other words, it has constant mean and variance.

3.1. Modeling a Time Series

Modeling time series refers to the ways to model a time series in order to make predictions. A few of the ways to model a time series are:

- *Moving Average method* is a basic modeling approach to model time series. This model simply states that the next observation is the mean of all past observations.

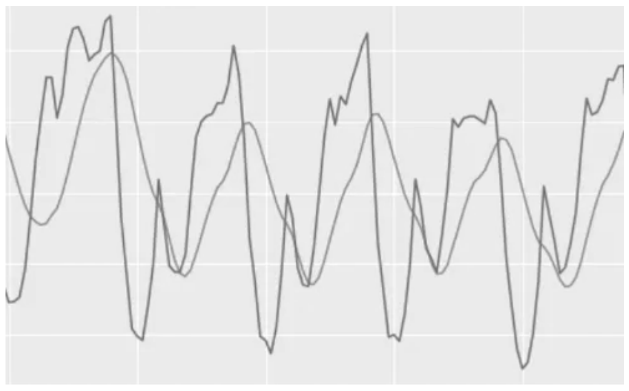


Fig 3.1.1 Moving Average applied with half unit window, the dark lines show the actual values, and the light lines show the rolling mean trend

- *Exponential smoothing* uses the idea of moving average but assigns a different weight to each observation, giving less importance to the observations which are farther to the current time series. Mathematically, it can be expressed as:

$$y = \alpha x_t + (1 - \alpha) y_{t-1}, t > 0$$

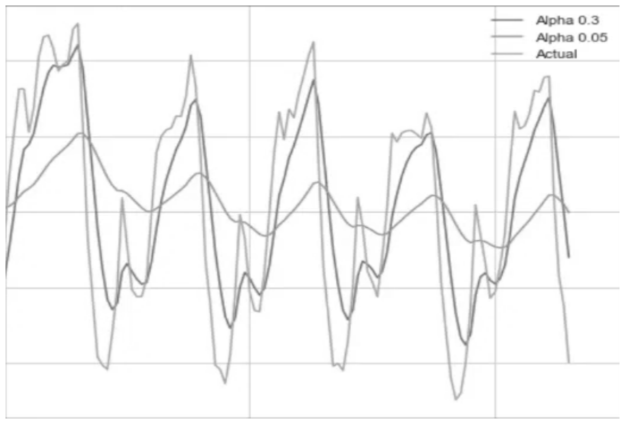


Fig 3.1.2 Example of exponential smoothing using different values of alphas

- Other modeling techniques include double exponential smoothing, triple exponential smoothing, Seasonal autoregressive integrated moving average model (SARIMA), etc.

3.2. Using LSTM for Time Series Forecasting

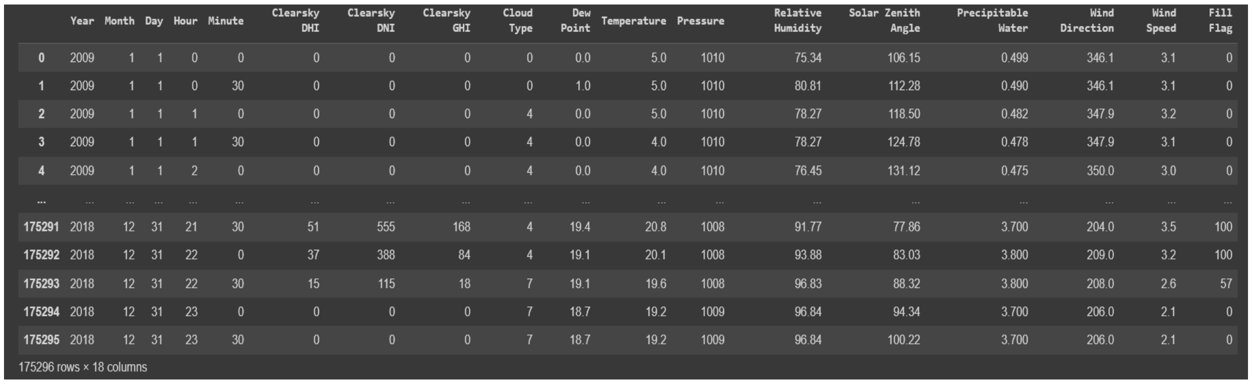
In this neural network model, we will be using Long Short Term Memory (LSTM) layers as the Recurrent Neural Network (RNNs) layers. For time series forecasting, to predict information for a time series, we will be using the previous 400 entries' data. We will pass data in batches of 512 to train the model faster. We will also be running 100 epochs to make sure that our model is trained well to give accurate predictions.

Chapter 4

The Global Radiation Prediction Model

4.1. Analyzing the data

We are provided with the global radiation parameters data of a specific region of every 30-minute interval for ten years, we will be using this data to train our model.



	Year	Month	Day	Hour	Minute	Clearsky DHI	Clearsky DNI	Clearsky GHI	Cloud Type	Dew Point	Temperature	Pressure	Relative Humidity	Solar Zenith Angle	Precipitable Water	Wind Direction	Wind Speed	Fill Flag
0	2009	1	1	0	0	0	0	0	0	0.0	5.0	1010	75.34	106.15	0.499	346.1	3.1	0
1	2009	1	1	0	30	0	0	0	0	1.0	5.0	1010	80.81	112.28	0.490	346.1	3.1	0
2	2009	1	1	1	0	0	0	0	4	0.0	5.0	1010	78.27	118.50	0.482	347.9	3.2	0
3	2009	1	1	1	30	0	0	0	4	0.0	4.0	1010	78.27	124.78	0.478	347.9	3.1	0
4	2009	1	1	2	0	0	0	0	4	0.0	4.0	1010	76.45	131.12	0.475	350.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175291	2018	12	31	21	30	51	555	168	4	19.4	20.8	1008	91.77	77.86	3.700	204.0	3.5	100
175292	2018	12	31	22	0	37	388	84	4	19.1	20.1	1008	93.88	83.03	3.800	209.0	3.2	100
175293	2018	12	31	22	30	15	115	18	7	19.1	19.6	1008	96.83	88.32	3.800	208.0	2.6	57
175294	2018	12	31	23	0	0	0	0	7	18.7	19.2	1009	96.84	94.34	3.700	206.0	2.1	0
175295	2018	12	31	23	30	0	0	0	7	18.7	19.2	1009	96.84	100.22	3.700	206.0	2.1	0

175296 rows × 18 columns

Fig 4.1.1 The global radiation parameters using which we will be training our model

The important parameters of our interest are:

- *Clearsky DHI* (Diffuse Horizontal Irradiance) represents solar radiation that does not arrive on a direct path from the Sun but has been scattered by clouds and particles in the atmosphere and comes equally from all directions.
- *Clearsky DNI* (Direct Normal Irradiance) represents the amount of light coming perpendicular to the surface. The surface here represents ground or something parallel to the ground. This type of irradiance belongs to rays that come in a straight line from the Sun's direction at its current position in the sky.
- *Clearsky GHI* (Global Horizontal Irradiance) represents the total amount of shortwave radiation received from above by a surface that is horizontal (parallel) to the ground.

The visual representation of these parameters is:

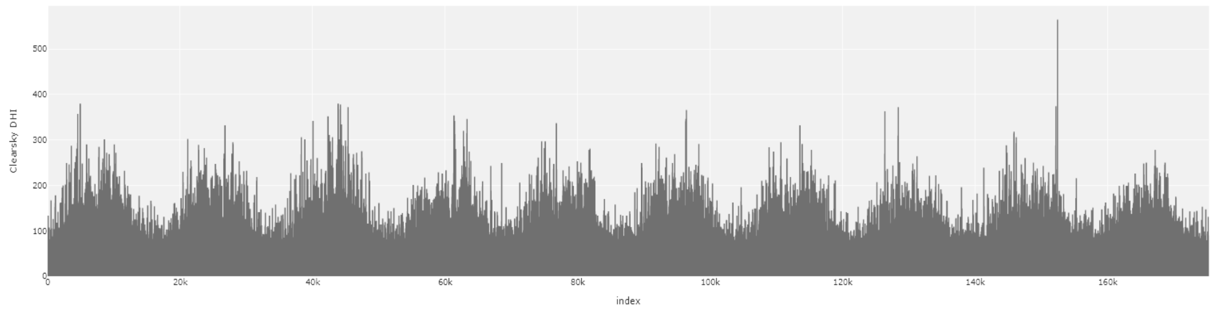


Fig 4.1.2 The visual representation of the parameter Clearsky DHI obtained from the training data

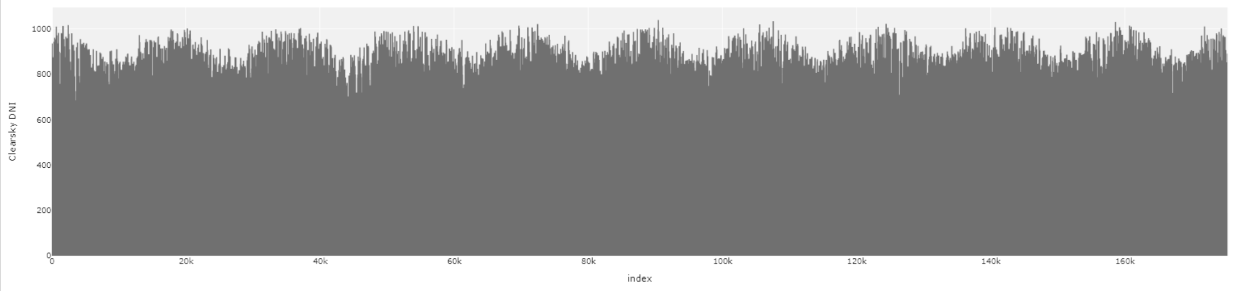


Fig 4.1.3 The visual representation of the parameter Clearsky DNI obtained from the training data

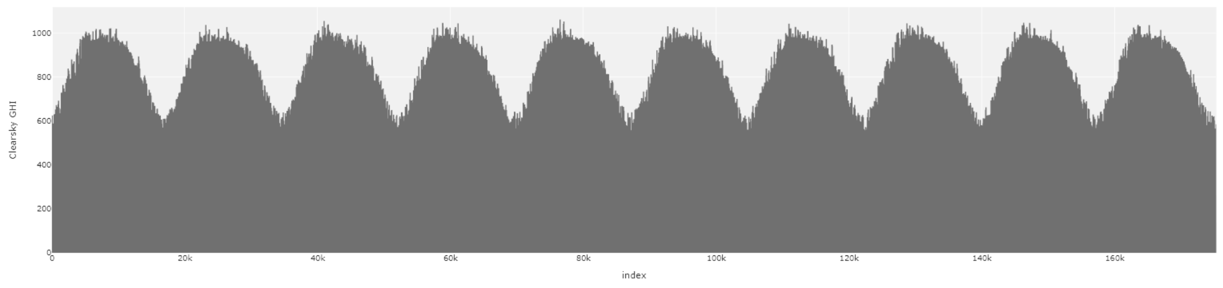
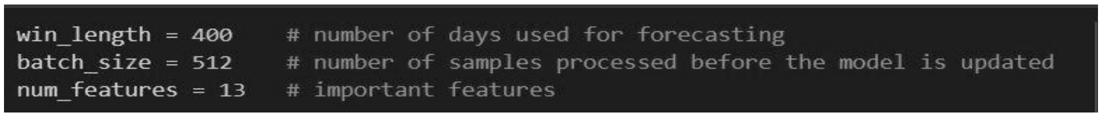


Fig 4.1.4 The visual representation of the parameter Clearsky GHI obtained from the training data

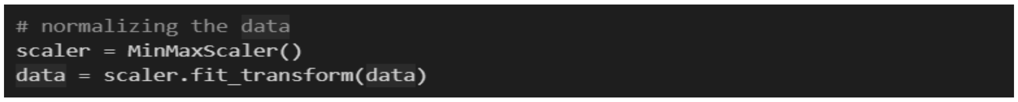
4.2. Creating training data

We will be storing the information in the dataset using LSTMs and other convolution layers of the neural network.



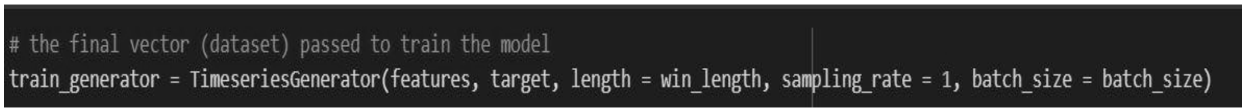
```
win_length = 400    # number of days used for forecasting
batch_size = 512    # number of samples processed before the model is updated
num_features = 13   # important features
```

Fig 4.2.1 Assignng variable to train the model



```
# normalizing the data
scaler = MinMaxScaler()
data = scaler.fit_transform(data)
```

Fig 4.2.2 Applying normalization on the data for better accuracy

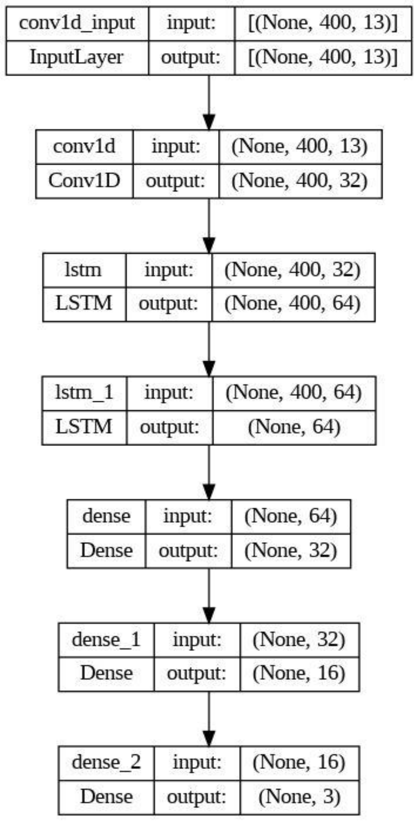


```
# the final vector (dataset) passed to train the model
train_generator = TimeseriesGenerator(features, target, length = win_length, sampling_rate = 1, batch_size = batch_size)
```

Fig 4.2.3 Storing the dataset and other required information in a vector

4.3. Layers Overview

The model has the following layers in the order mentioned.

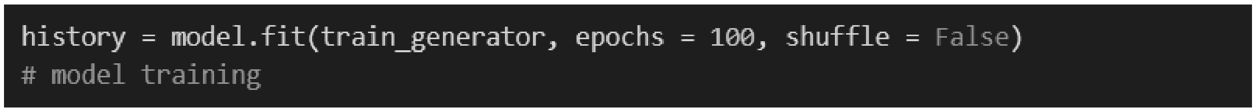


conv1d_input | input: | [(None, 400, 13)]
InputLayer | output: | [(None, 400, 13)]
conv1d | input: | (None, 400, 13)
Conv1D | output: | (None, 400, 32)
lstm | input: | (None, 400, 32)
LSTM | output: | (None, 400, 64)
lstm_1 | input: | (None, 400, 64)
LSTM | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 32)
dense_1 | input: | (None, 32)
Dense | output: | (None, 16)
dense_2 | input: | (None, 16)
Dense | output: | (None, 3)

Fig 4.2.3 The layers of the model in the given order

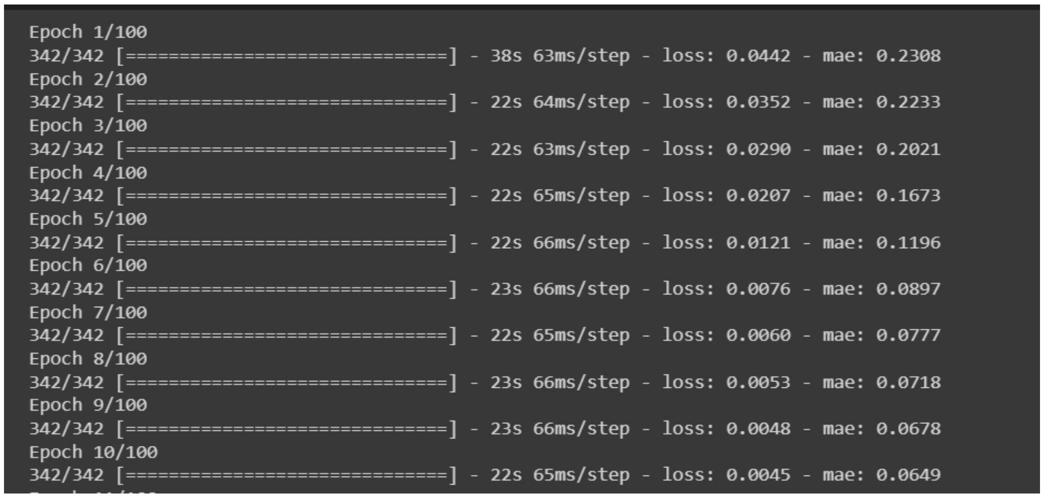
4.4. Training Results

Training the model using the following code mentioned below.



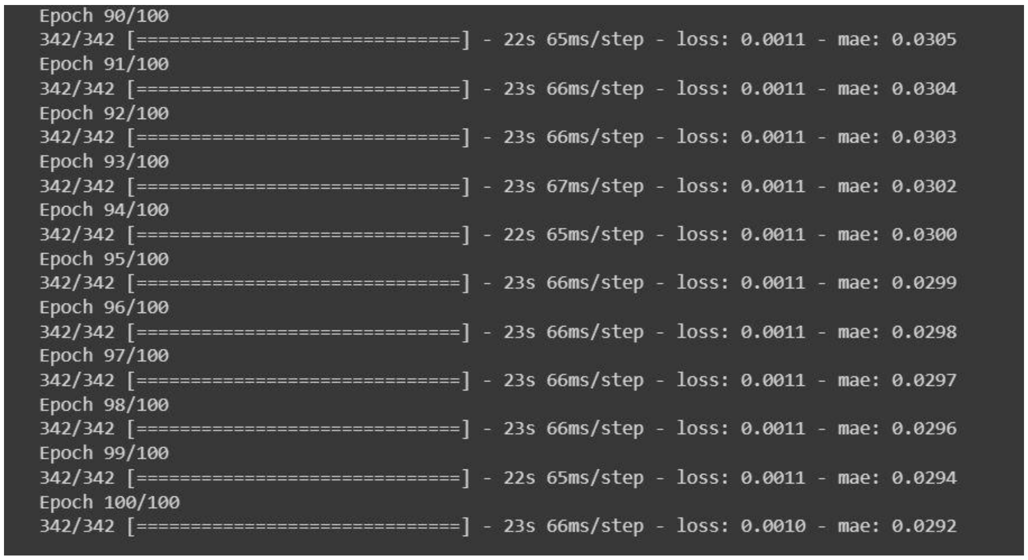
```
history = model.fit(train_generator, epochs = 100, shuffle = False)
# model training
```

Fig 4.4.1 The training function



```
Epoch 1/100
342/342 [==============================] - 38s 63ms/step - loss: 0.0442 - mae: 0.2308
Epoch 2/100
342/342 [==============================] - 22s 64ms/step - loss: 0.0352 - mae: 0.2233
Epoch 3/100
342/342 [==============================] - 22s 63ms/step - loss: 0.0290 - mae: 0.2021
Epoch 4/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0207 - mae: 0.1673
Epoch 5/100
342/342 [==============================] - 22s 66ms/step - loss: 0.0121 - mae: 0.1196
Epoch 6/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0076 - mae: 0.0897
Epoch 7/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0060 - mae: 0.0777
Epoch 8/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0053 - mae: 0.0718
Epoch 9/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0048 - mae: 0.0678
Epoch 10/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0045 - mae: 0.0649
```

Fig 4.4.2 At the start of the training, the values of the Huber loss and the MAE loss



```
Epoch 90/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0011 - mae: 0.0305
Epoch 91/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0304
Epoch 92/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0303
Epoch 93/100
342/342 [==============================] - 23s 67ms/step - loss: 0.0011 - mae: 0.0302
Epoch 94/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0011 - mae: 0.0300
Epoch 95/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0299
Epoch 96/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0298
Epoch 97/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0297
Epoch 98/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0296
Epoch 99/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0011 - mae: 0.0294
Epoch 100/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0010 - mae: 0.0292
```

Fig 4.4.3 At the end of the training, the values of the Huber loss and the MAE loss

The visual representation of some parameters after training the model are:

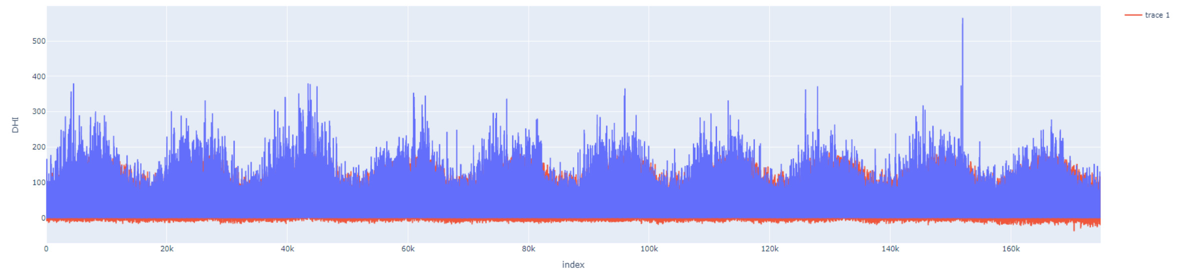


Fig 4.4.4 Predicted Clearsky DHI on the training data, the orange area shows the predicted values, the blue shows the actual values

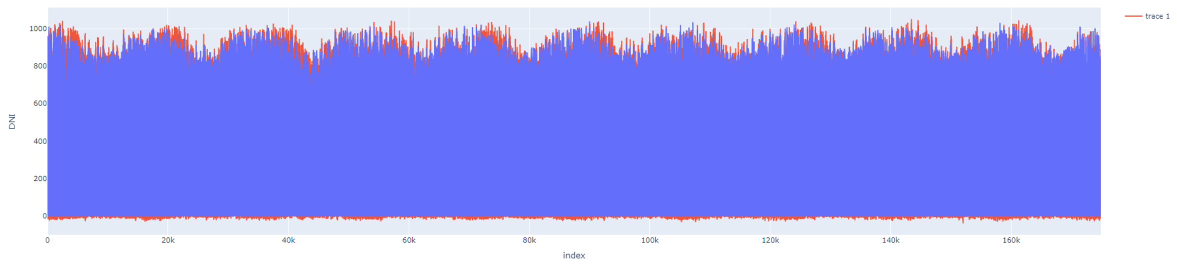


Fig 4.4.5 Predicted Clearsky DNI on the training data, the orange area shows the predicted values, the blue shows the actual values.

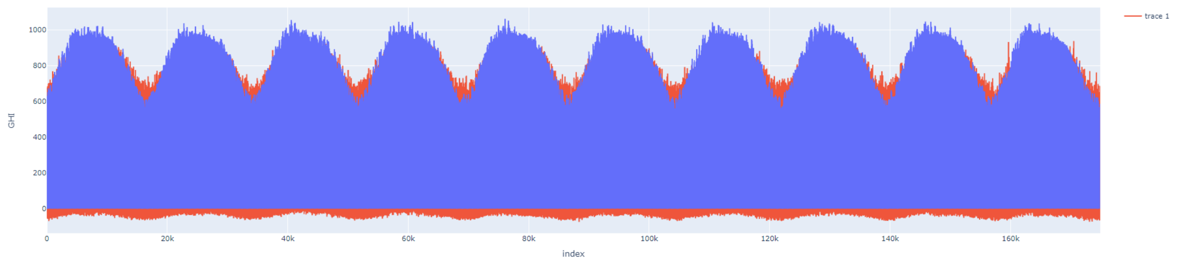


Fig 4.4.6 Predicted Clearsky GHI on the training data, the orange area shows the predicted values, and the blue shows the actual values

4.5. Results on Test Data

The following results were obtained from the test dataset.

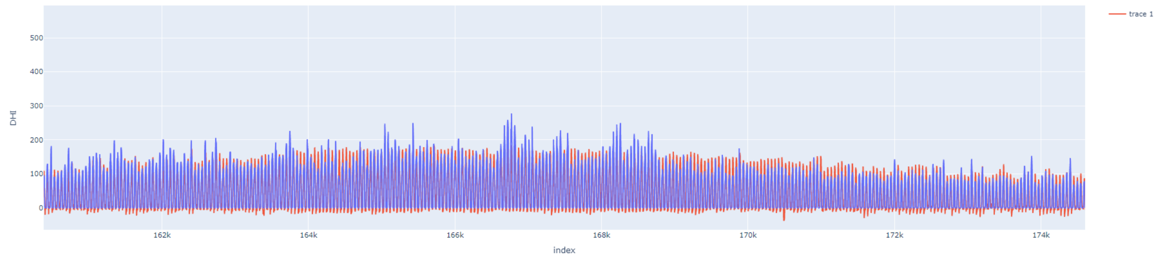


Fig 4.5.1 Predicted Clearsky DHI on the test data, the orange area shows the predicted values, the blue shows the actual values

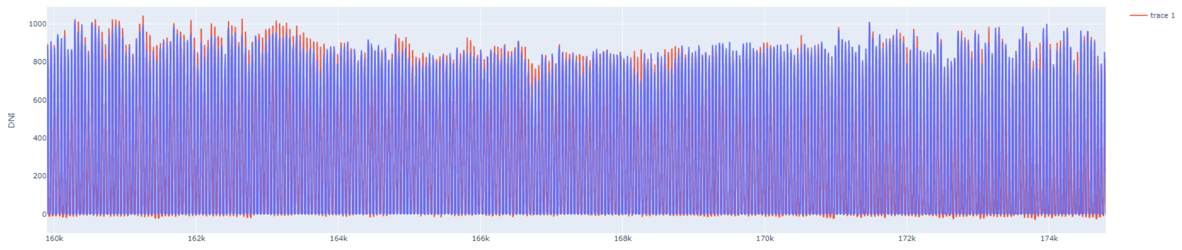


Fig 4.5.2 Predicted Clearsky DNI on the test data, the orange area shows the predicted values, the blue shows the actual values

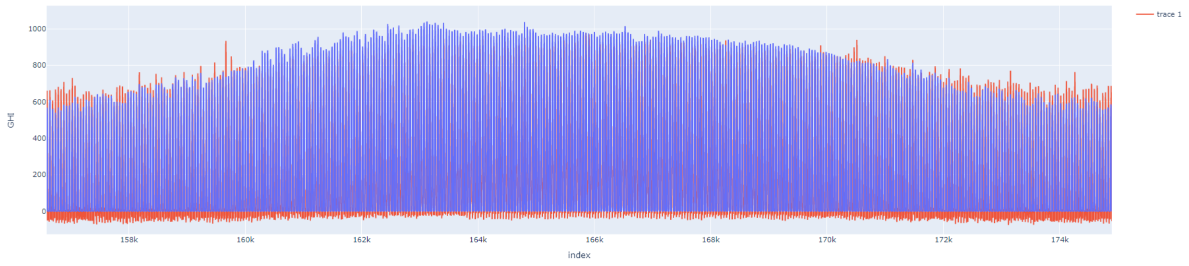


Fig 4.5.3 Predicted Clearsky GHI on the test data, the orange area shows the predicted values, the blue shows the actual values

Conclusion

In building the Global Radiation Prediction Model, sequence models such as Recurrent Neural Network (RNNs), Long Short Term Memory (LSTMs) were implemented with the help of TensorFlow framework library. The multivariate time series forecasting was applied on the same to make the predictions more accurate.

Loss functions such as Huber Loss and Mean Absolute Error (MAE) were used to calculate the model's accuracy.

On the test dataset, Huber loss of 0.0012 and Mean Absolute Error (MAE) of 0.0314 was found, reflecting the model's high accuracy.

Scope of Improvement

The model can be further improved using techniques such as regularization, and optimization, or by using different modelling techniques.

Different regularization methods like L2 regularization, dropout regularization, and early stopping can be implemented to check if these help in improving the accuracy of the model. Different optimization techniques such as better normalization of inputs can be tried to check the further improvement in the accuracy of the model.

Different modeling techniques such as AutoRegressive Integrated Moving Average (ARIMA), and Seasonal AutoRegressive Integrated Moving Average (SARIMA) can be used to model the time series forecasting.

References

The following resources were referred to while completing the project:

- Deep Learning Specialization by *Andrew Ng at coursera.org*
- Understanding LSTM Networks by *colah's blog*
- Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling by *Junyoung Chung*
- The Complete Guide to Time Series Analysis and Forecasting by *Rian Dolphin*
- Time Series Analysis and Forecasting by *Marco Peixeiro*
- *MachineHack* for providing the dataset