horizontal line

Comparative Analysis for Energy Forecasting Using an IoT Data

────────────────────────────────────

Tuesday, 16.11.2022

# 

# 

# 

# 

# Report Submitted to -

# Dr. Sonali Agarwal

# Report by

**Abhishek Karmakar (ids2022007) && Sharik Gazi (ids2022009)**

**M.Tech - Indian Institute of Information Technology, Allahabad**

# Abstract

### Energy forecasting for commercial real estate is the process of applying statistics to make predictions about consumption levels and corresponding utility costs in both the short term and long term. Energy forecasting models take into account Global active power, Global inactive power, voltage, and Global intensity, Submetering1, Submetering2 and Submetering3 to make predictions. Forecasting can be both expected price value and probabilistic forecasting. Energy forecasting is a technique to predict future energy needs to achieve demand and supply equilibrium. Here in this project we have created a LSTM model which can predict the future values of all the attributes.

# 

# 

# Table of Contents

## **Introduction**

## **Proposed Methodology**

## **2.1) Visualization and Code Implementation**

**2.2) Elephas Library Implementation**

**2.3) Outcome**

## **Future Work**

# 

# Chapter 1

# Introduction

### The dataset can be downloaded from the link given below. <https://archive.ics.uci.edu/ml/machine-learning-databases/00235/>

### The dataset contains the following attributes: Date, Time, Global\_active\_power, Global\_reactive\_power, Voltage, Global\_intensity, Sub\_metering\_1, Sub\_metering\_2, Sub\_metering\_3.

### This dataset contains 2075259 measurements gathered between December 2006 and November 2010 (47 months).

### Notes:

### 1. (globalactivepower\*1000/60 - submetering1 - submetering2 - submetering3) represents the active energy consumed every minute (in watt hour) in the household by electrical equipment not measured in sub-meterings 1, 2 and 3.

### 2. The dataset contains some missing values in the measurements (nearly 1,25% of the rows). All calendar timestamps are present in the dataset but for some timestamps, the measurement values are missing: a missing value is represented by the absence of value between two consecutive semi-colon attribute separators. For instance, the dataset shows missing values on April 28, 2007.

### **Attribute Information:**

### 1. date: Date in format dd/mm/yyyy

### 2. time: time in format hh:mm:ss

### 3. globalactivepower: household global minute-averaged active power (in kilowatt)

### 4. globalreactivepower: household global minute-averaged reactive power (in kilowatt)

### 5. Voltage: minute-averaged voltage (in volt)

### 6. global\_intensity: household global minute-averaged current intensity (in ampere)

### 7. submetering1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).

### 8. submetering2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.

### 9. submetering3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

### The main objective of this report is to implement a LSTM model for the correct predictions of the all the attributes using PySpark. Well in this report an additional package called as elephas been used to implement the model training. This makes it faster. The keras and the Spark library has been integrated for training, reason behind there has been Deep learning packages embedded in the PySpark library.

# 

# Chapter 2

### **Proposed Methodology**

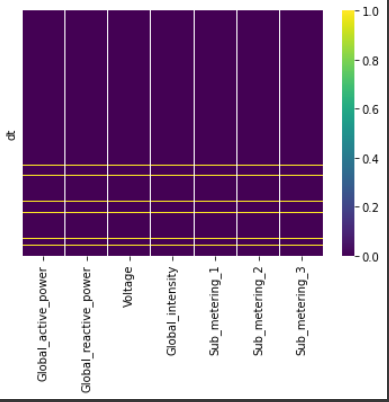
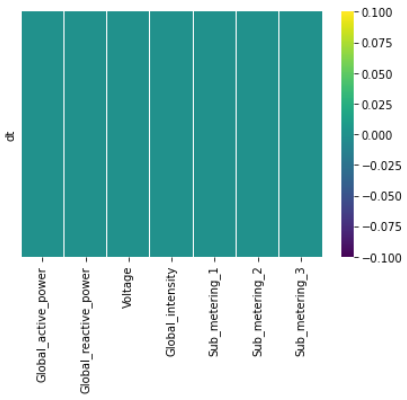
**2.1 Code Implementation**

After the installation of pyspark library a trick has been used. It was seen that when the household prediction dataset was uploaded on the google collab, there was a lot of samples missing. So, in place of that we had tried to pick up the dataset through kaggle.json file. The household prediction dataset has been hosted on kaggle and then that dataset was downloading into google collab using kaggle.json file. The downloaded file is a zip file, so that zipped file has to be extracted and hence then it becomes ready to use. This section of the report consists of visualization (step2) and data cleaning part (part3). Also in part4 we hold the model training phase.

**Step1:** The time and date column has been merged into one and then created as an index. All the samples containing ‘?’ has been converted into NaN. The columns that it contained were all float64 types. This database have 20,75,259 rows and 7 columns.

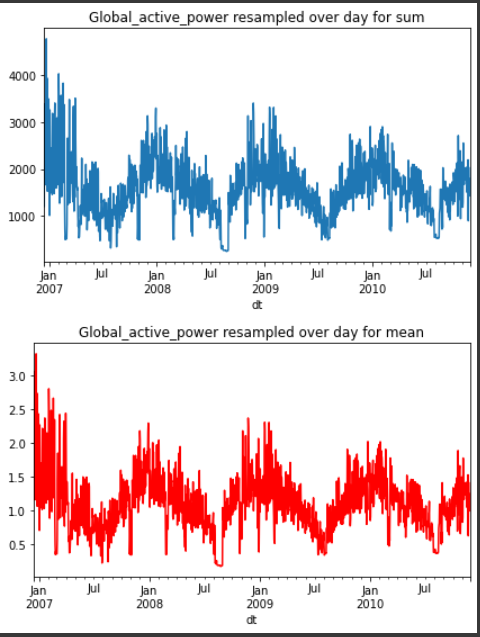
**Step2:** shows the null values contents. So the first thing that has been done to remove all sorts of missing data is to replace it with the mean values.

Left image consisting of the NaN values and right image has been obtained after replaced Nan with mean values.

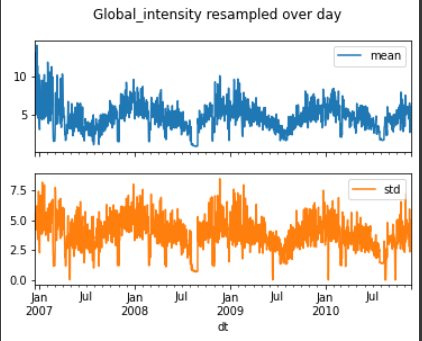
 

**Step3:** Plotting the "just data" has very high granularity. Therefore we have re-sampled the data to make it convenient.

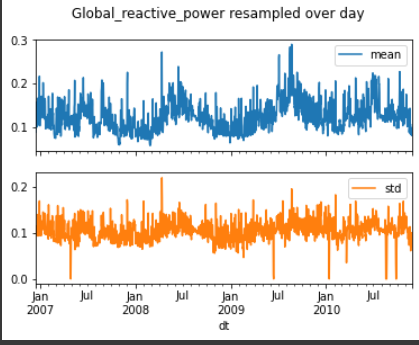
Below we resample over day, and show the sum and mean of Global\_active\_power. It is seen that mean and sum of resampled data set, have similar structure.



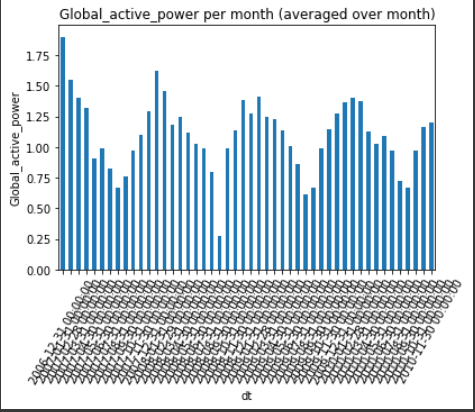
Below we have shown mean and standard Deviation of 'Global\_intensity' resampled over day.



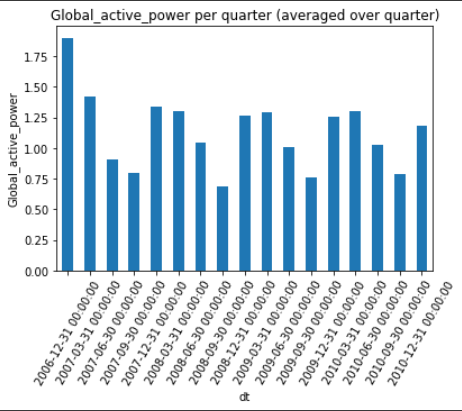
Below we have shown mean and standard Deviation of 'Global\_reactive\_power' resampled over day.



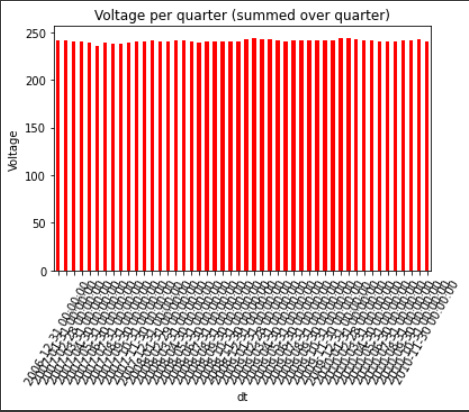
Below again Mean of 'Global\_active\_power' resampled over



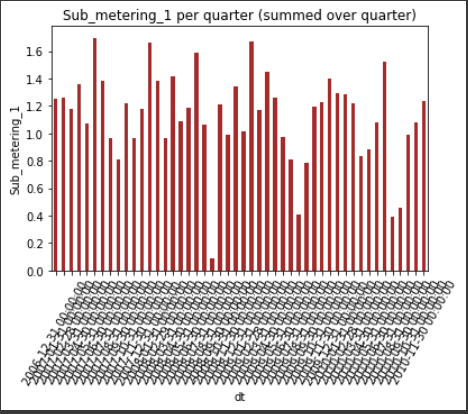
Mean of 'Global\_active\_power' resampled over quarter



It is very important to note from above two plots that resampling over larger time interval will diminish the periodicity of system as we expect. This is important for machine learning feature engineering. So we will be considering the resampling over hourly basis on the later part of the project.

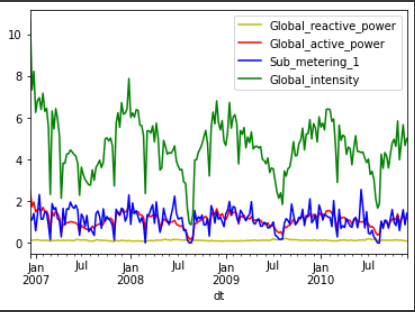


Here is the submetering1 on monthly basis, image attached below.

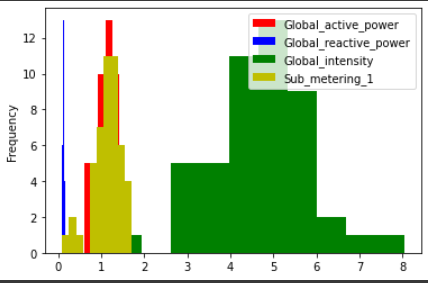


It is seen from the above plots that the mean of 'Voltage' over month is pretty much constant compared to other features. This is important again in feature selection.

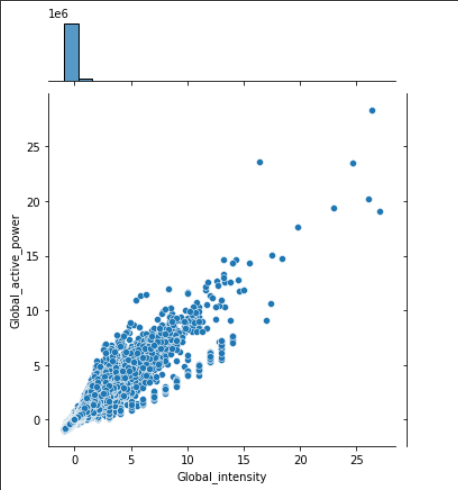
Let’s have a look at resampling over week and computing mean.



Below we have shown histograms of the mean of different feature resampled over month.

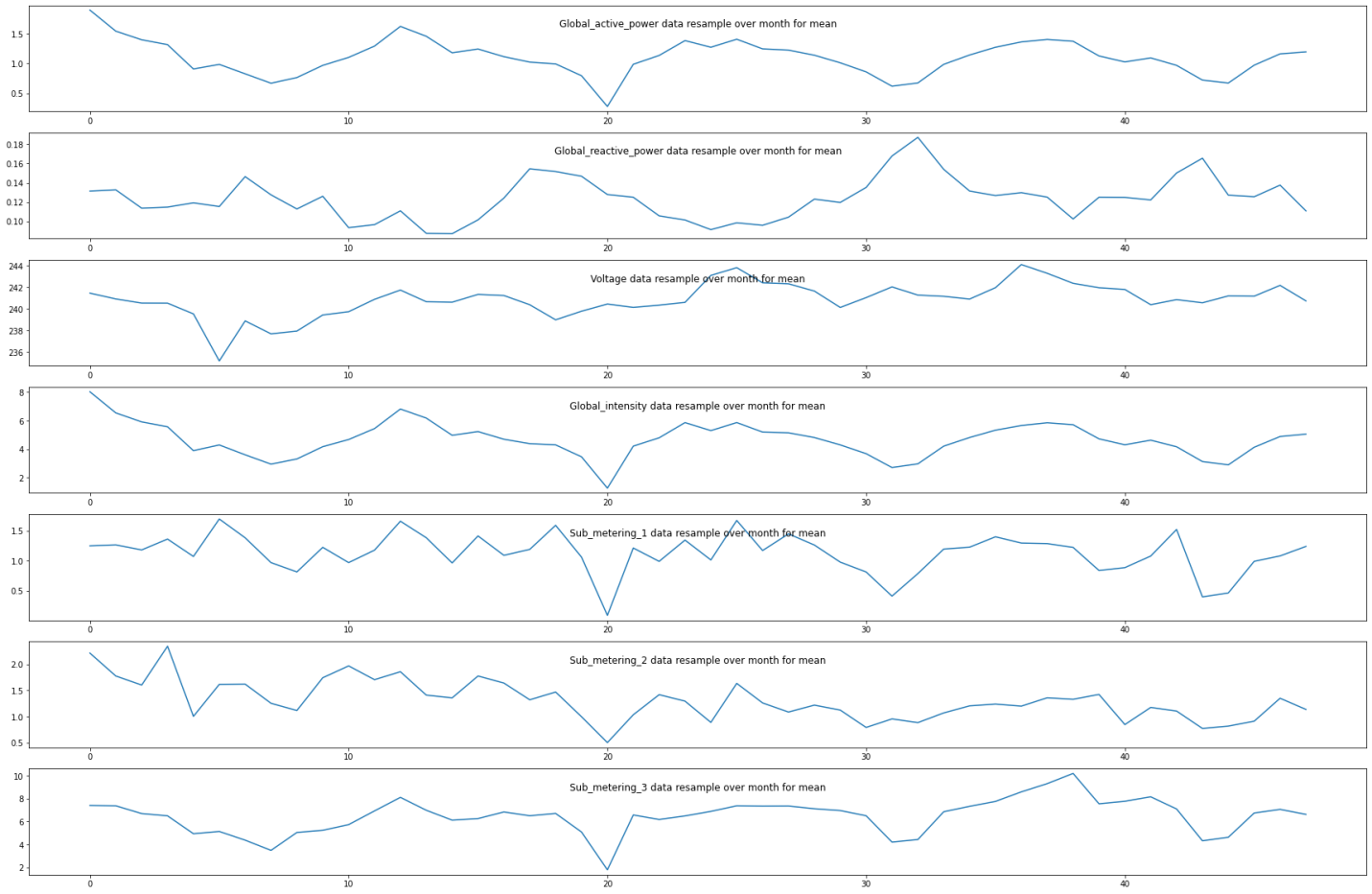


Talking about the correlations between 'Global\_intensity', 'Global\_active\_power', here it’s shown below. Here is how it looks like.

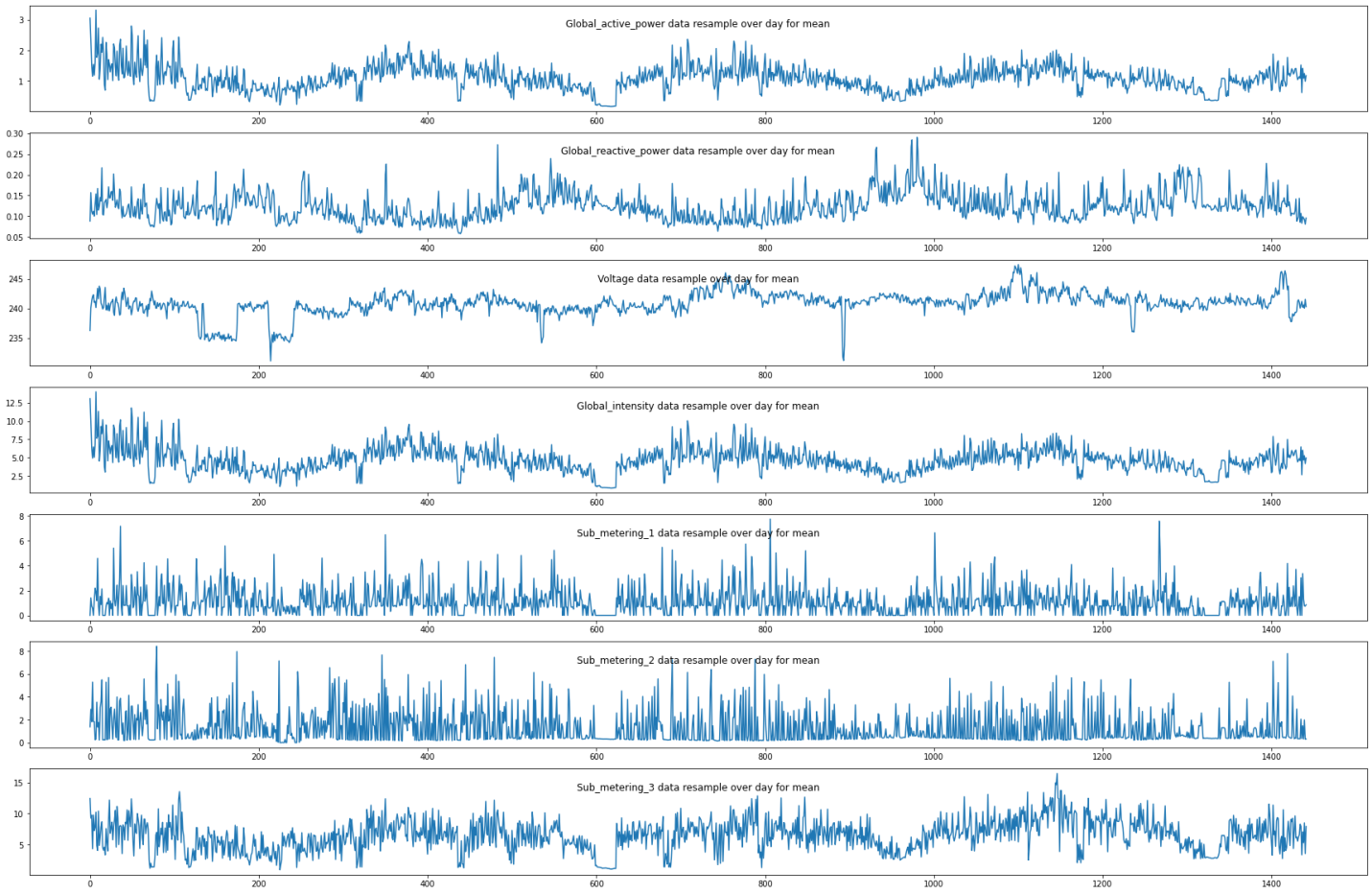


From above two plots it is seen that 'Global\_intensity' and 'Global\_active\_power' correlated. But 'Voltage', 'Global\_active\_power' are less correlated. This is important observation for machine learning purpose.

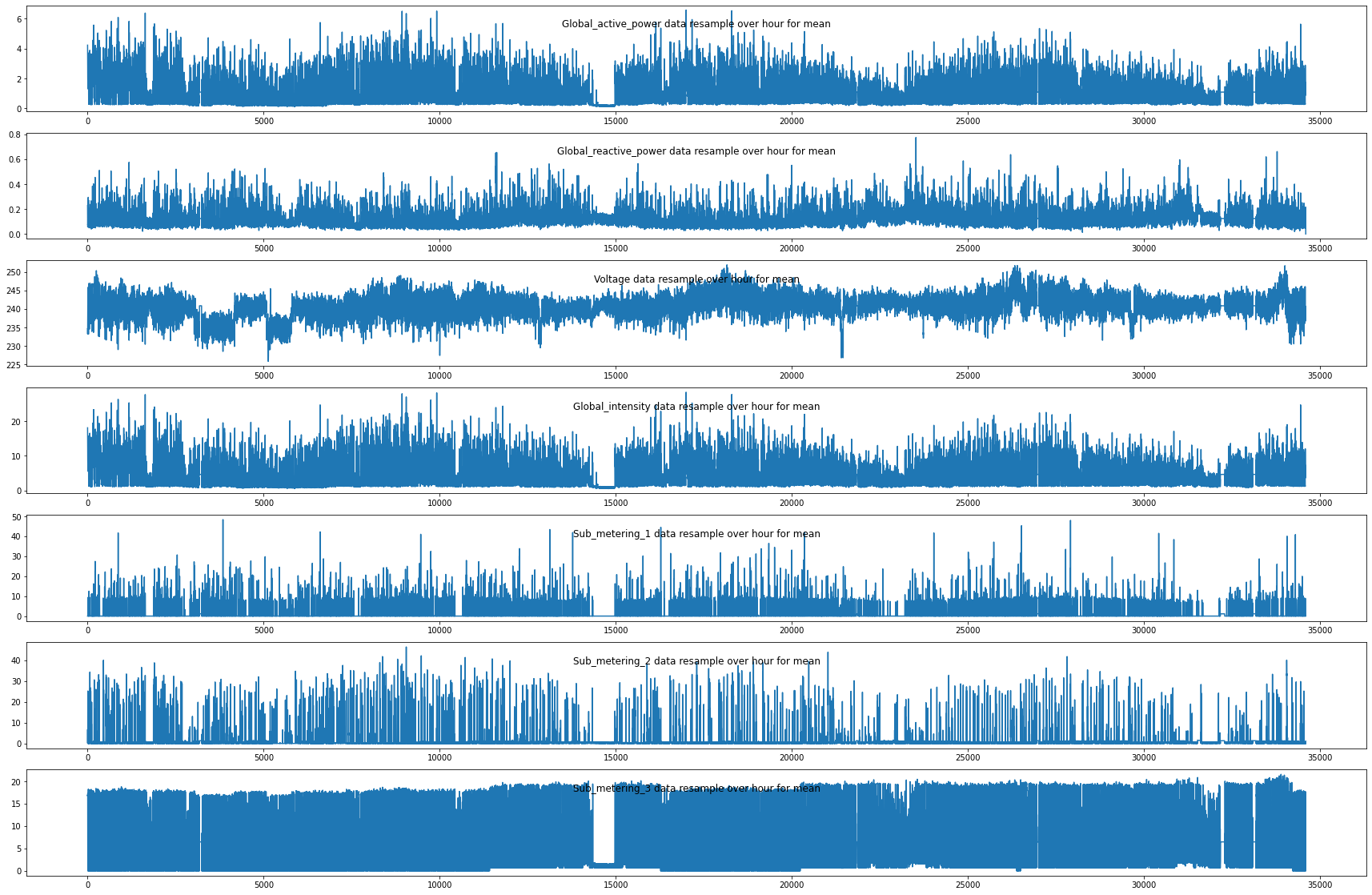
Given Below is the month wise re-sampling output.



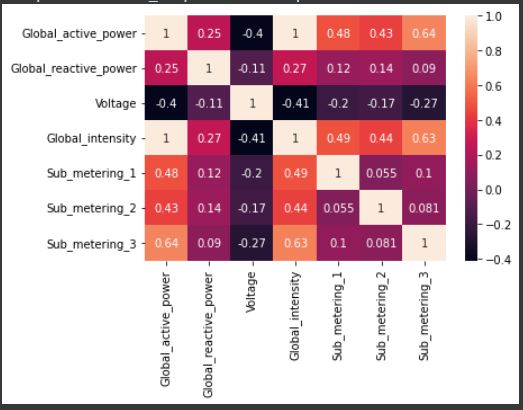
Below shown day wise Re-Sampling where we compared the mean of different features resampled over day.



Shown below image is the hour wise re-sampling output.



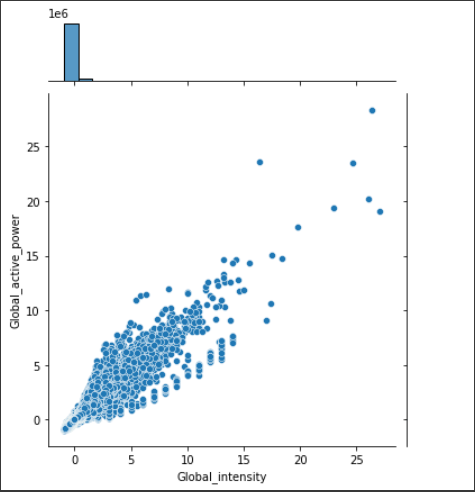
And the correlation matrix is as follows.



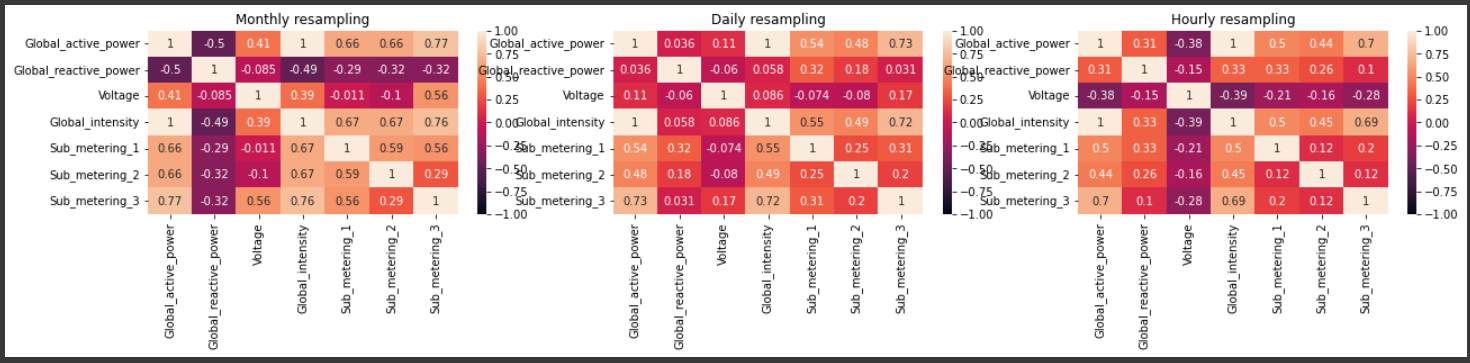
Therefore, if processing all the original data, the runtime will be very costly, but if processing data with large time-scale samples (e.g. monthly), it will affect the model's productivity.

From Correlation Matrix, it is seen that 'Global\_intensity' and 'Global\_active\_power' correlated. But 'Voltage', 'Global\_active\_power' are less correlated.

The correlations between 'Global\_intensity', 'Global\_active\_power' is shown below. Percentage change between the current and a prior element of 'Global\_intensity', 'Global\_active\_power' has been checked through jointplot.



Now, let’s have a look at the monthly, daily and hourly sampling.



It is seen from above that with resampling techniques one can change the correlations among features.

**Step 4:** LSTM Data Preparation and fitting. The LSTM neural network has been selected because: It is best suited for large data, time-series, and sequential problem. In the first step, we will frame the problem to predict the Global\_active\_power.

Now before moving ahead, let’s first remove all the outliers that are present in the dataset. As the outliers always hamper our model accuracy and is no help to us, therefore the first and foremost rule that should be applied before moving forward is removing the outliers.

There is many ways of removing outliers out of which we are selecting the Inter Quartile Range for detecting and removing the outliers.

Each dataset can be divided into quartiles. The first quartile point indicates that 25% of the data points are below that value whereas second quartile is considered as median point of the dataset. The inter quartile method finds the outliers on numerical datasets by following the procedure below

Find the first quartile, Q1.

Find the third quartile, Q3.

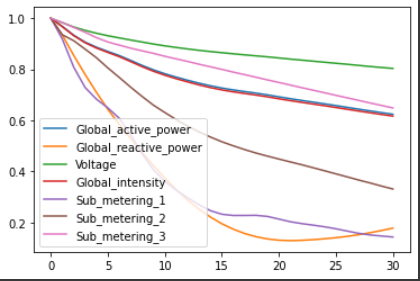
Calculate the IQR. IQR= Q3-Q1.

Define the normal data range with lower limit as Q1–1.5\*IQR and upper limit as Q3+1.5\*IQR.

Any data point outside this range is considered as outlier and should be removed for further analysis.

The concept of quartiles and IQR can best be visualized from the boxplot. It has the minimum and maximum point defined as Q1–1.5\*IQR and Q3+1.5\*IQR respectively. Any point outside this range is outlier.

To fill in missing values, first check for linearity in each column using autocorrelation plots. Below, we get a plot of how autocorrelation of different signals change with lag to better model the imputation. Hence below here is just a graph showing the linearity of data. Seems like autocorrelations start dropping after 5 lags for majority of signals in a much extent. So let’s do linear interpolation but with a limit of 5 to fill in missing values using linear interpolation with limit to 5 using autocorrelation plot.

This part of the implementation has been done for better modelling purpose but unfortunately it didn’t help much so we dropped the idea.

The NaN values have been replaced with the mean of the particular attributes.

From here onwards, the model training phase will begin.

Firstly we will frame the supervised learning problem as predicting the Global\_active\_power at the current time (t) keeping given the Global\_active\_power measurement and other features at the prior time step for training purpose.

In order to reduce the computation time, and also get a quick result to test the model. We have resampled the data over hour to reduce the size of data from 2075259 to 34589 (data are given in minutes).

So, we will have 7 input series variables and the 1 output variable for 'Global\_active\_power' at the current time in hour.

We also were splitting the data into: train and validation sets. Let's select 30000 data over 34,589 data to train; the rest will be used to test the model. We also are splitting the data into train and validation sets. We have split the data into train and test data series. Only 30000 first data points are selected for training purpose, rest 4588 for testing purpose.

For global active power, LSTM model looks like:-

* 100 neurons in the first visible layer
* 30 neuron for Dense Layer and then 1 neuron in the output layer for predicting Global\_active\_power
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient adam version of stochastic gradient descent
* The model will be fit for 25 training epochs with a batch size of 70.

In the final Output the training loss comes to be 0.92% and testing loss comes out to be 0.63%



Mean Absolute Error: 0.376

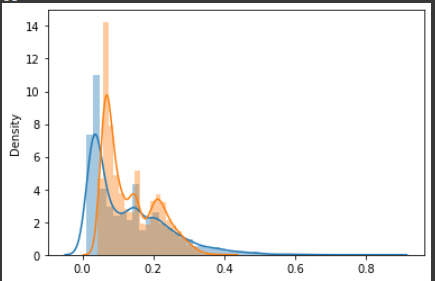
Mean Squared Error: 0.260

Test RMSE: 0.510

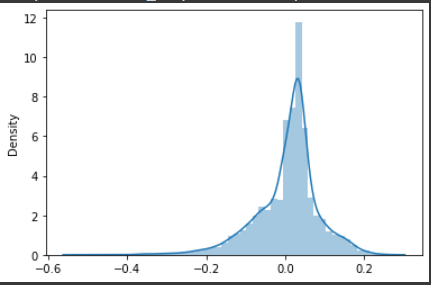
Given below the first 500 hours prediction



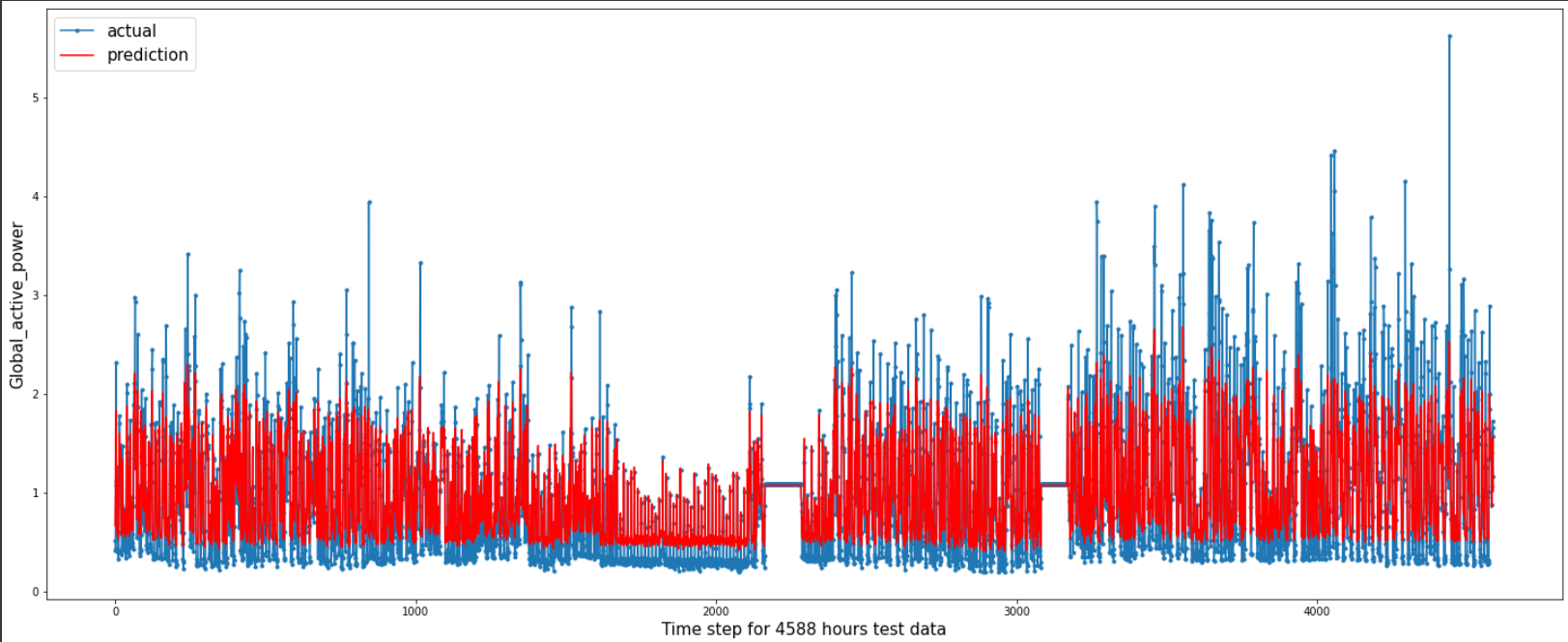
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



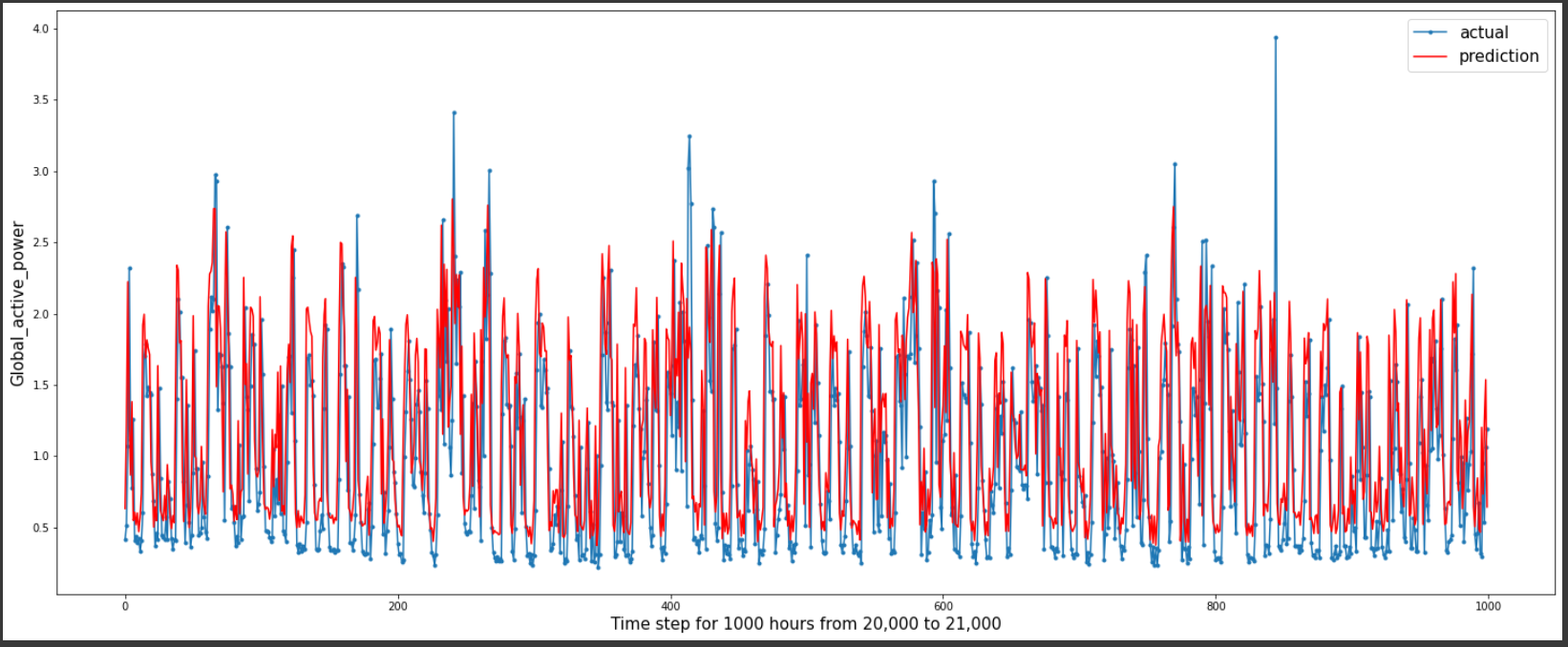
Given below is the distplot of difference of actual and predicted value.



Given below the prediction for the test data.



Given below time step for 1000 hours from 20,000 to 21,000 Prediction.

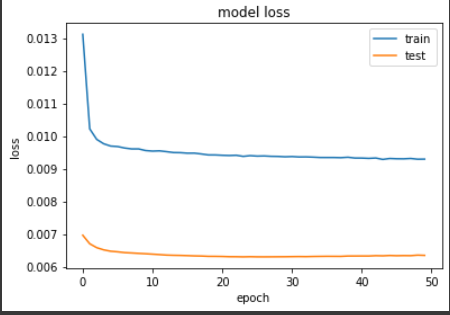


Now, let’s check on for Sub\_metering3. Again the same 30,000 data values have been chosen for training purpose while the remaining for testing purpose. After the min-max normalization, model has been trained.

For Sub\_metering3, model looks like:-

* 100 neurons in the first visible layer
* dropout 10%
* 1 neuron in the output layer for predicting Voltage
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient adam version of stochastic gradient descent
* The model will be fit for 50 training epochs with a batch size of 100.

In the final Output the training loss comes to be 0.93% and testing loss comes out to be 0.63%

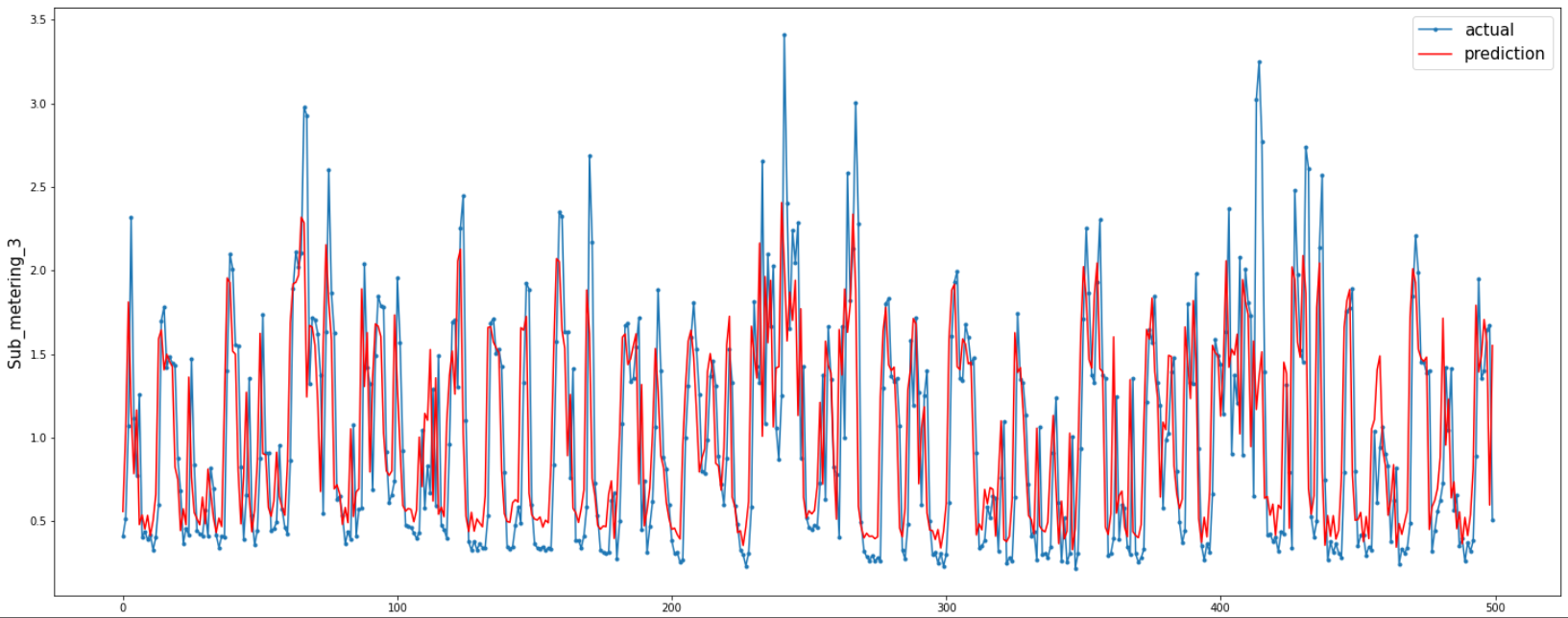


Mean Absolute Error: 0.362

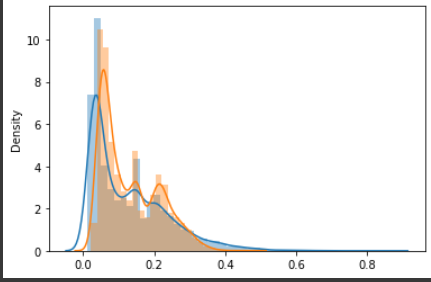
Mean Squared Error: 0.263

Test RMSE: 0.513

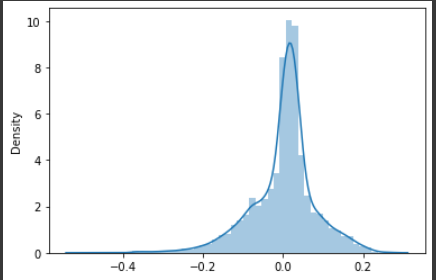
Given below the first 500 hours prediction



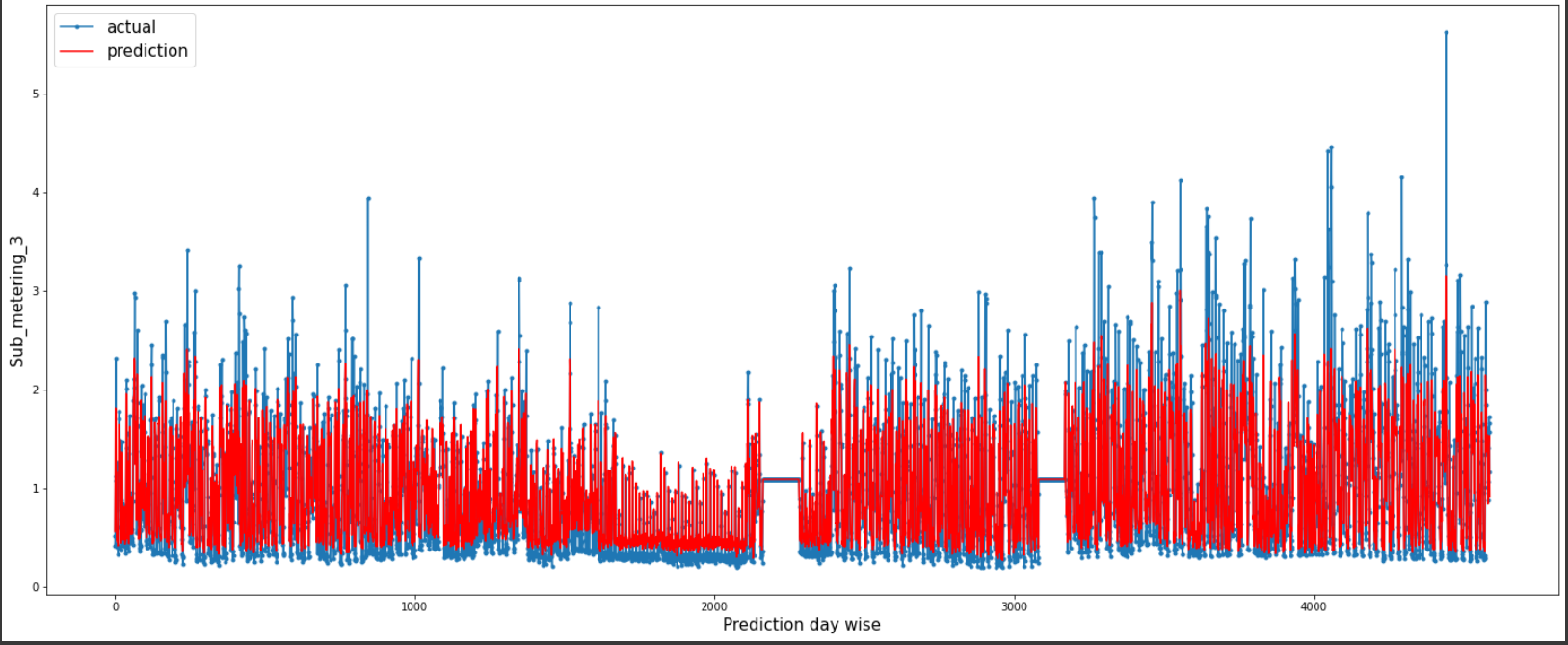
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



Given below is the histogram of difference of actual and predicted value.



Given below the prediction for the test data.

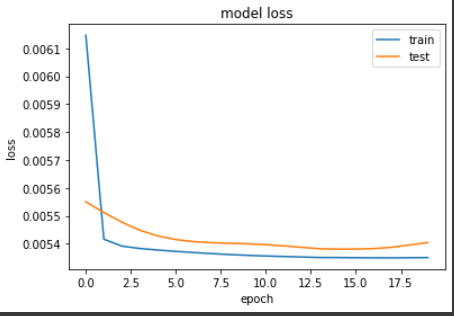


Now, let's try to fit and predict the Global Reactive Power.

Again the same thing which happened above has been repeated. For Global Reactive Power, model looks like:-

* 100 neurons in the first visible layer
* Dense layer of 30 neurons and 1 neuron in the output layer for predicting Voltage
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient adam version of stochastic gradient descent
* The model will be fit for 20 training epochs with a batch size of 100.

In the final Output the training loss comes to be 0.54% and testing loss comes out to be 0.55%

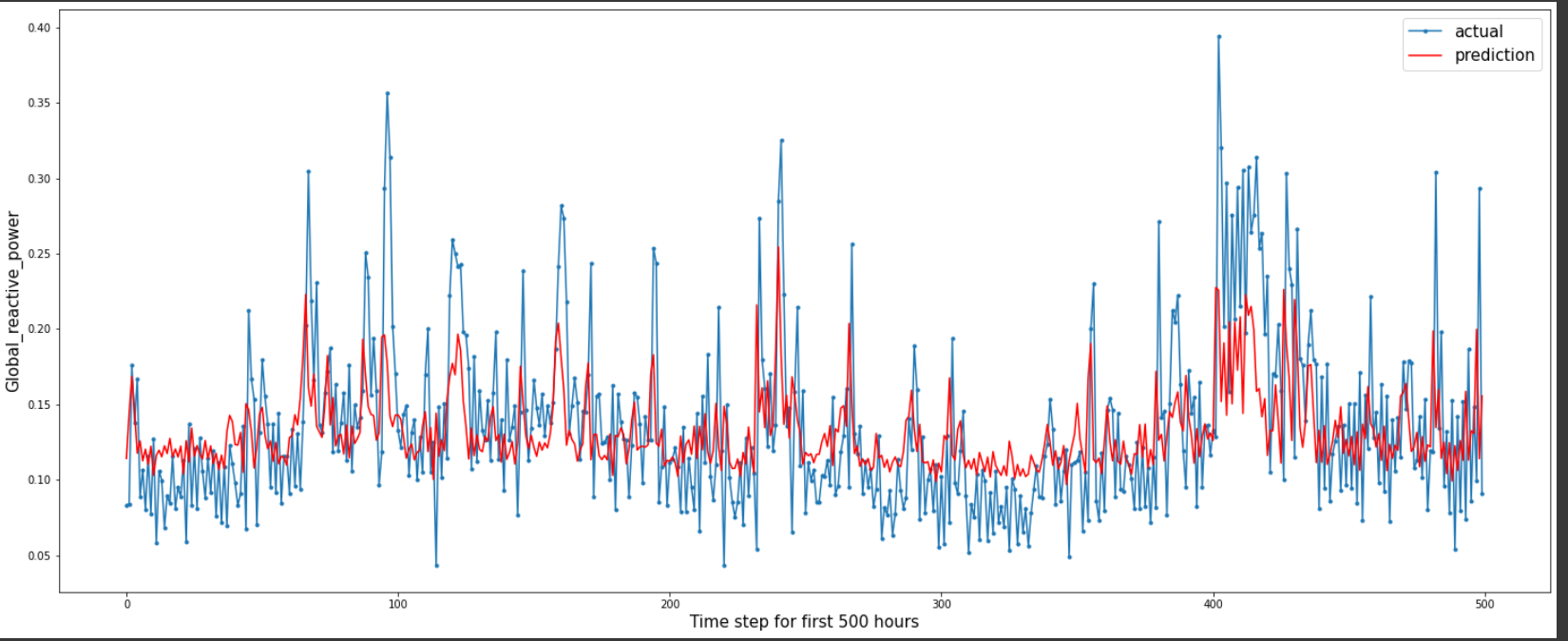


Mean Absolute Error: 0.043

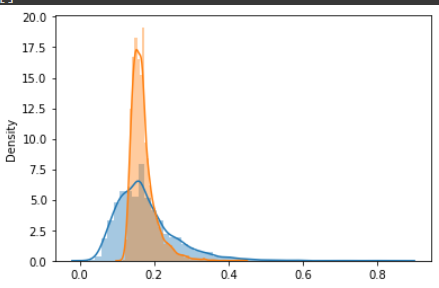
Mean Squared Error: 0.003

Test RMSE: 0.057

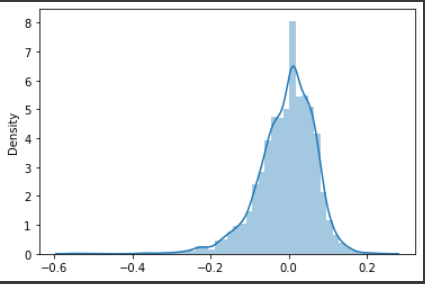
Given below the first 500 hours prediction



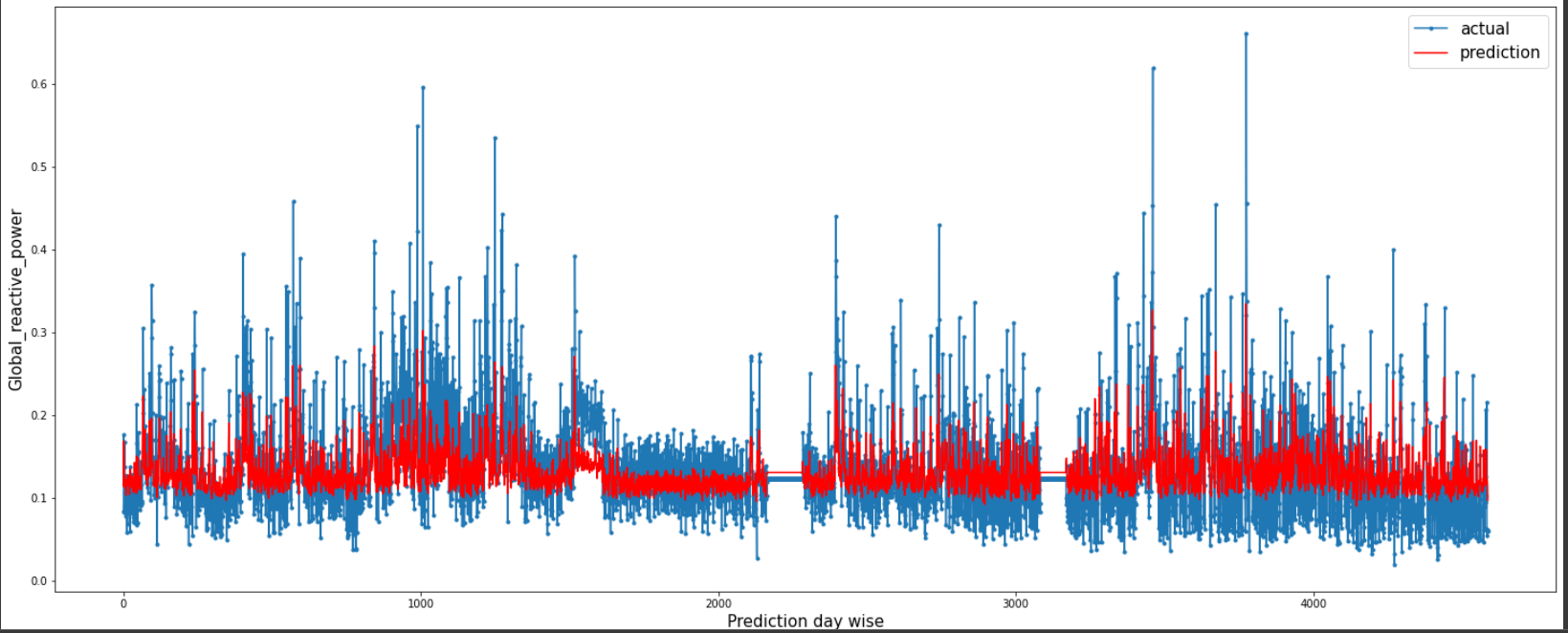
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



Given below is the histogram of the difference of actual and predicted value.



Given below the prediction for the test data.

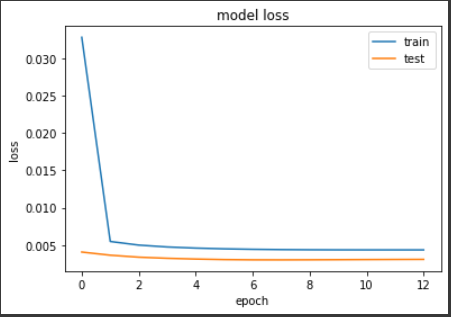


Now, let’s work on the Predict Voltage attribute.

After the min-max scalarization, the Voltage Model looks like:-

* 100 neurons in the first visible layer
* 1 neuron in the output layer for predicting Voltage
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient adam version of stochastic gradient descent
* The model will be fit for 13 training epochs with a batch size of 100.

In the final Output the training loss comes to be 0.43% and testing loss comes out to be 0.31%



Mean Absolute Error: 1.099

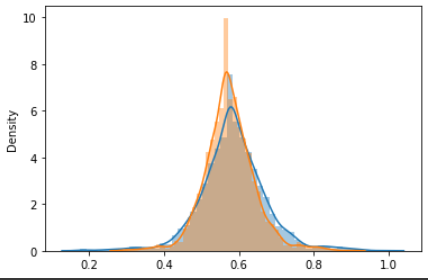
Mean Squared Error: 2.091

Test RMSE: 1.446

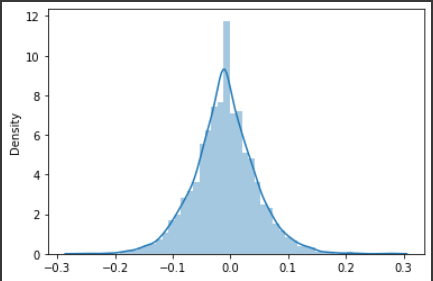
Given below the first 500 hours prediction



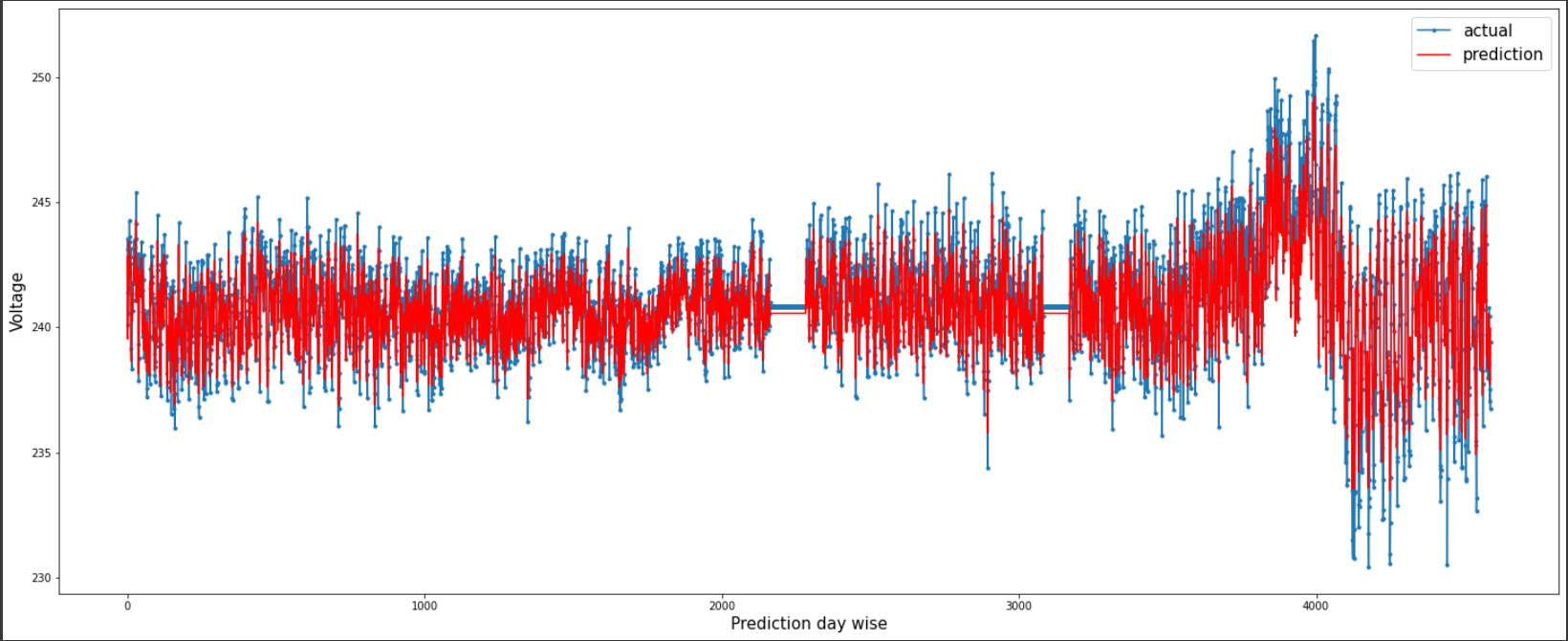
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



Given below is the histogram of the difference of actual and predicted value.



Given below the prediction for the test data.



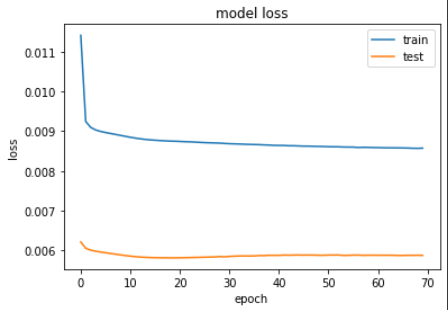
The Model Evaluation for Global Intensity works in the way explained below.

All the normalized data has been passed into the LSTM model training on 30000 data samples and predicting on 4589 data samples.

After the min-max scalarization, the Global Intensity Model looks like:-

* 100 neurons in the first visible layer
* 30 neurons on the Dense Layer
* 1 neuron in the output layer for predicting Global Intensity
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient adam version of stochastic gradient descent
* The model will be fit for 70 training epochs with a batch size of 100.

In the final Output the training loss comes to be 0.86% and testing loss comes out to be 0.59%

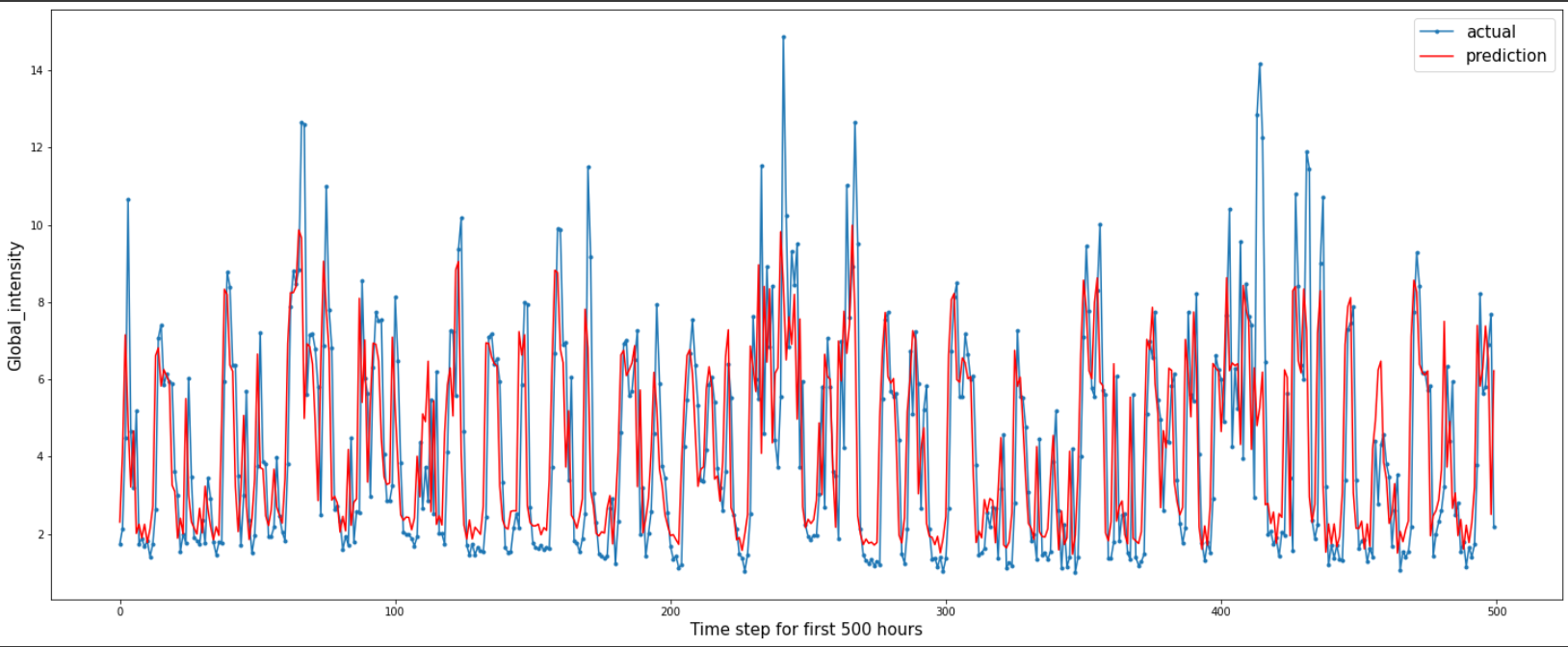


Mean Absolute Error: 1.492

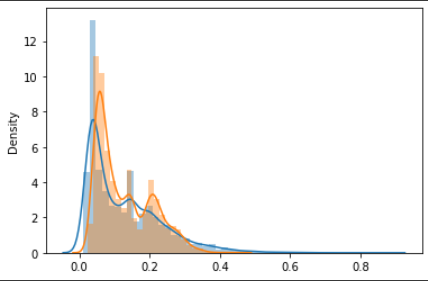
Mean Squared Error: 4.565

Test RMSE: 2.137

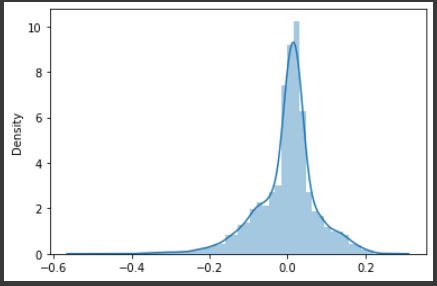
Given below the first 500 hours prediction



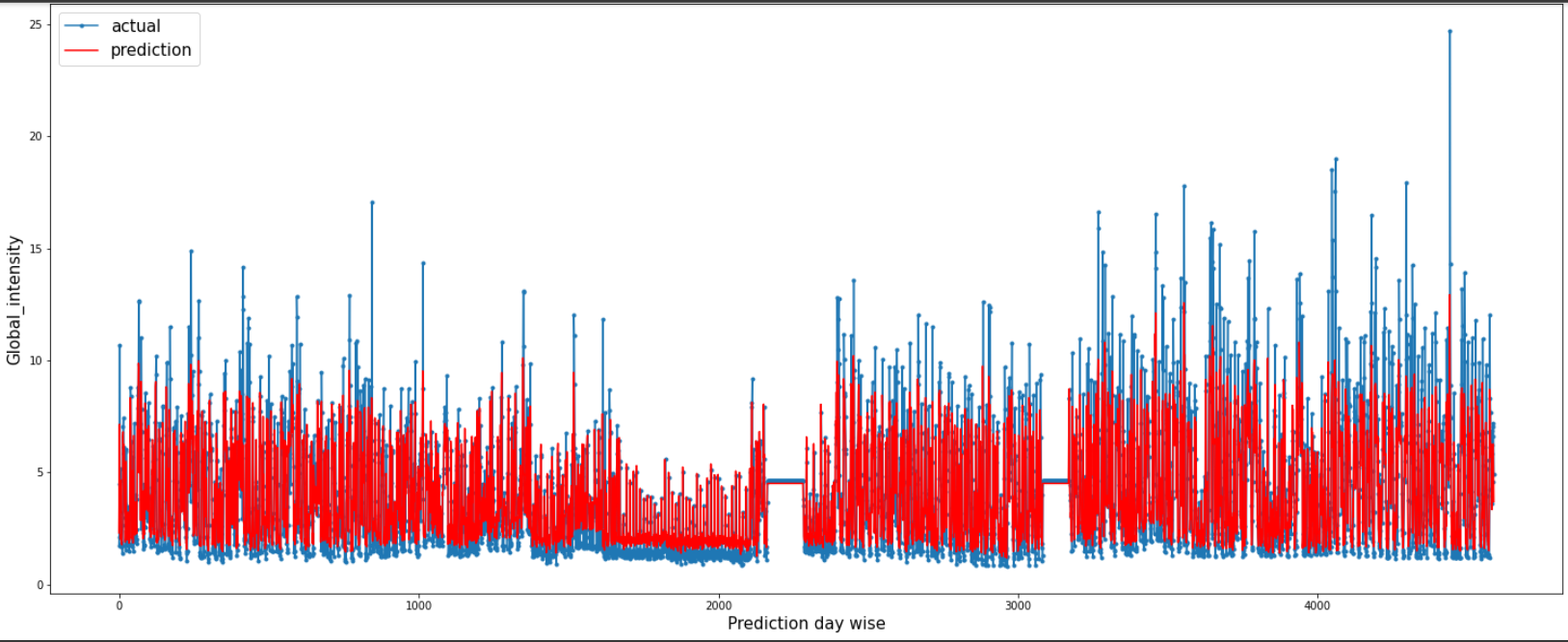
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



Given below is the histogram of the difference of actual and predicted value.



Given below the prediction for the test data.

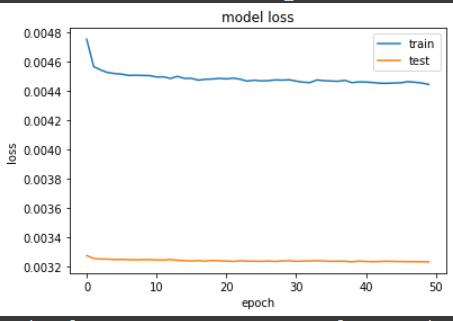


For workflow has now moved towards the Sub\_metering1 attributed. Unlike any other attributes, all the operations remain the same, except for the model training parameters.

After the min-max scalarization, the Sub\_metering1 Model looks like:-

* 200 neurons in the first visible layer
* Dropout of 0.5 %
* 1 neuron in the output layer for predicting Sub\_metering1
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient adam version of stochastic gradient descent
* The model will be fit for 50 training epochs with a batch size of 100.

In the final Output the training loss comes to be 0.44% and testing loss comes out to be 0.32%

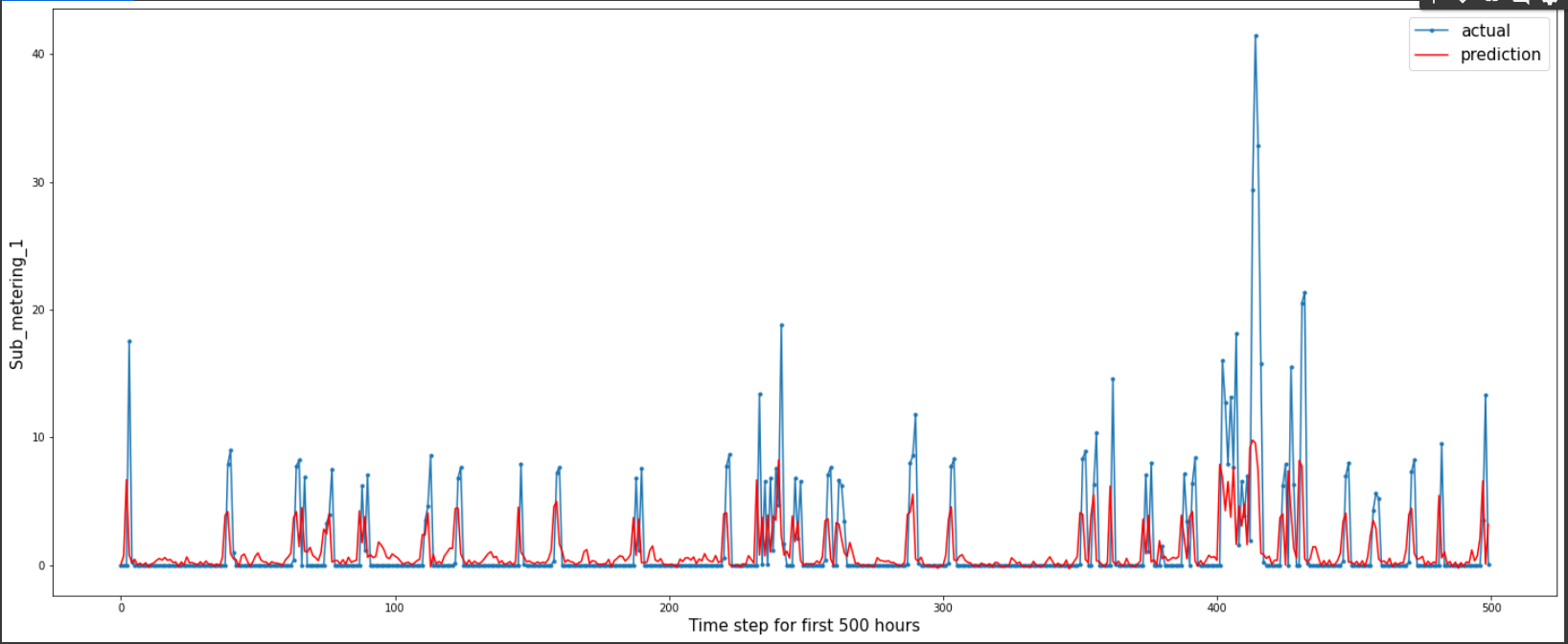


Mean Absolute Error: 1.168

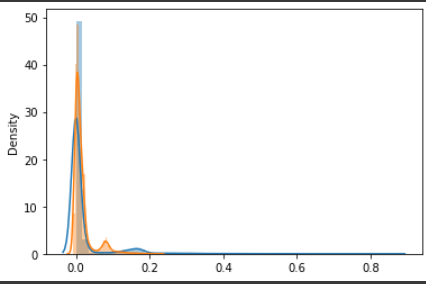
Mean Squared Error: 7.563

Test RMSE: 2.750

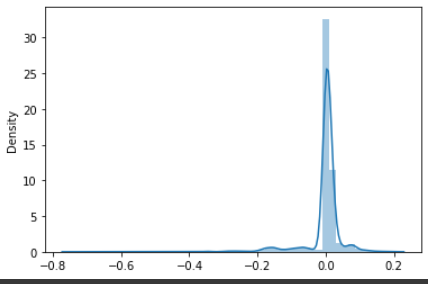
Given below the first 500 hours prediction



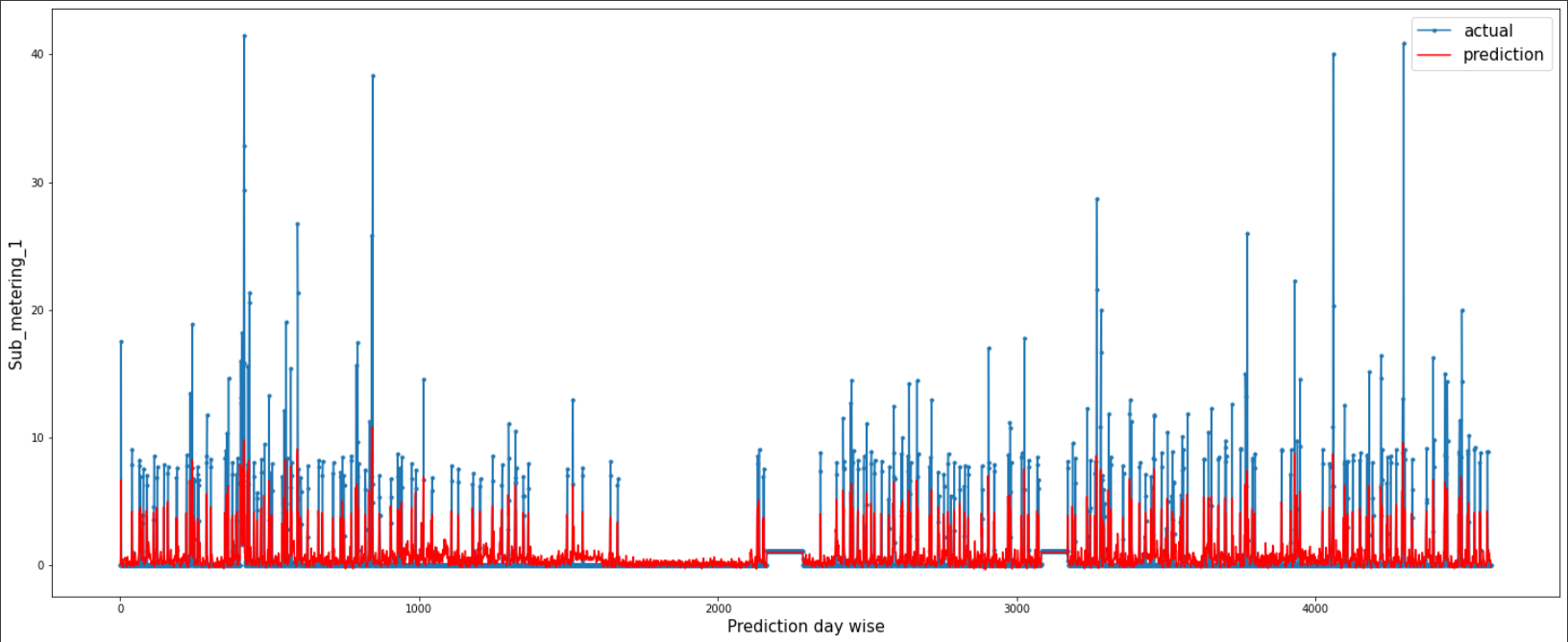
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



Given below is the histogram of the difference of actual and predicted value.



Given below the prediction for the test data.

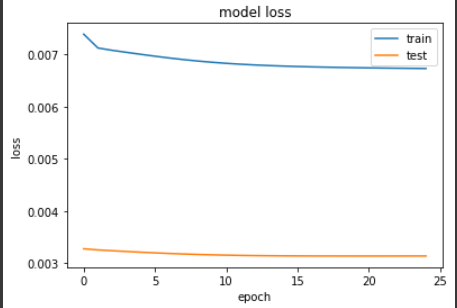


For workflow has now moved towards the Sub\_metering2 attributed. Unlike any other attributes, all the operations remain the same, except for the model training parameters.

After the min-max scalarization, the Sub\_metering2 Model looks like:-

* 100 neurons in the first visible layer
* Dropout of 0.5 %
* 1 neuron in the output layer for predicting Sub\_metering2
* The input shape will be 1 time step with 7 features
* The mean\_squared\_error loss function and the efficient RMSprop version of stochastic gradient descent
* The model will be fit for 50 training epochs with a batch size of 100.

In the final Output the training loss comes to be 0.67% and testing loss comes out to be 0.31%

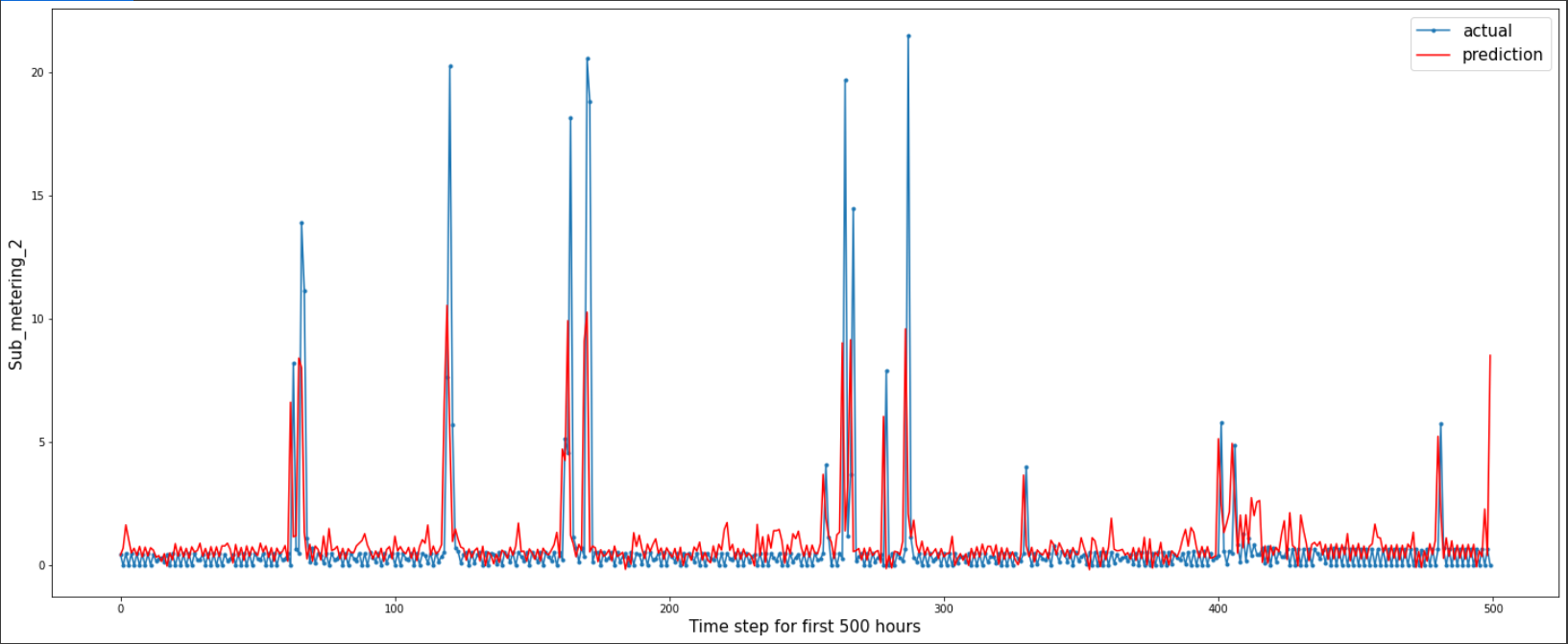


Mean Absolute Error: 1.023

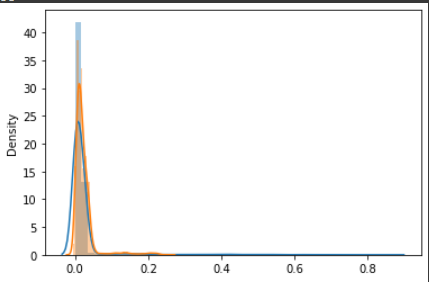
Mean Squared Error: 6.770

Test RMSE: 2.602

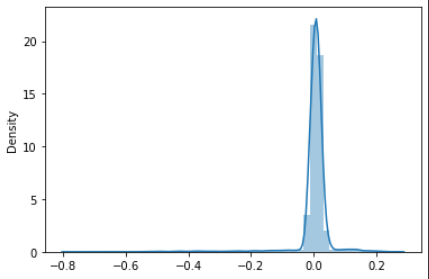
Given below the first 500 hours prediction



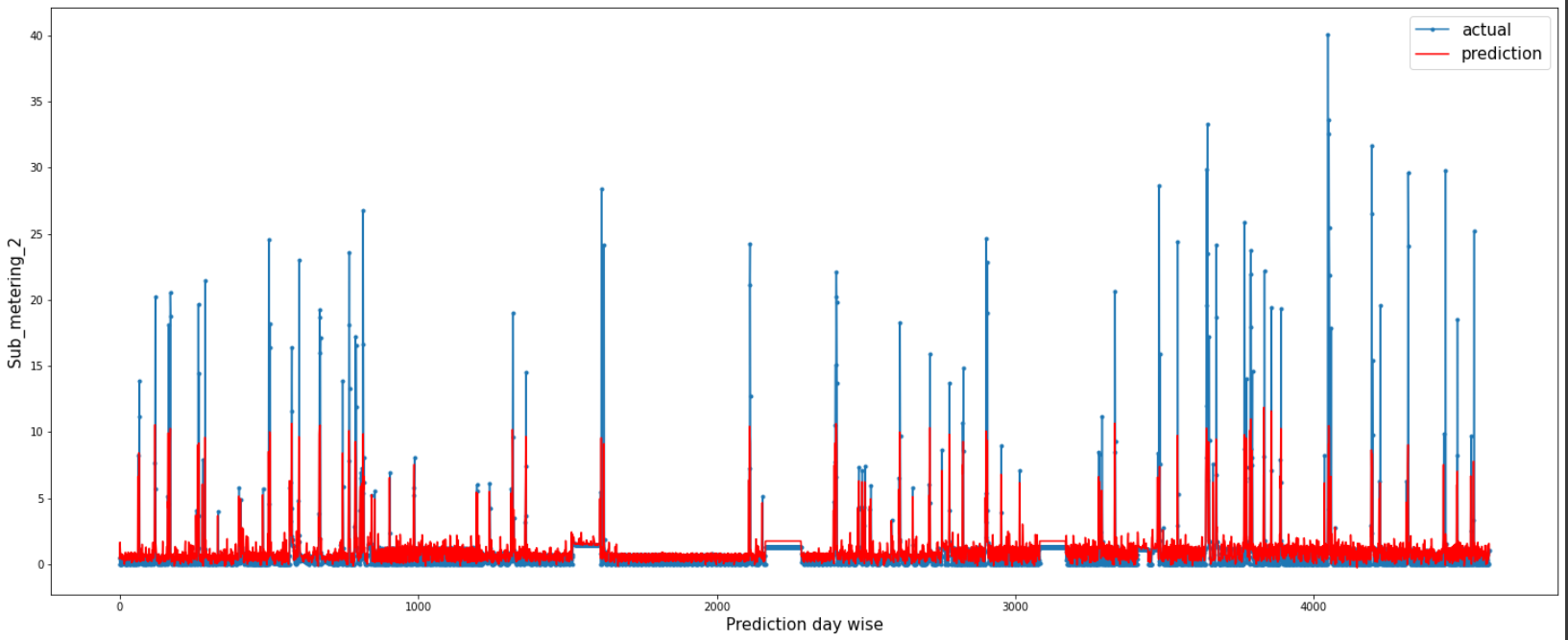
Given below is the actual and predicted distribution of how much similarity there can be for the model correct outcome.



Given below is the histogram of the difference of actual and predicted value.



Given below the prediction for the test data.



**2.3 Elephas**

In this report we have used PySpark, Keras, and Elephas python libraries to build an end-to-end deep learning pipeline that runs on Spark. Spark is an open-source distributed analytics engine that can process large amounts of data with tremendous speed. PySpark is simply the python API for Spark that allows you to use an easy programming language, like python, and leverage the power of Apache Spark.

Elephas brings deep learning with Keras to Spark. Elephas intends to keep the simplicity and high usability of Keras, thereby allowing for fast prototyping of distributed models, which can be run on massive data sets.

Using setAppName() and also set how many workers you want. I’m just running this locally and I set it to a possible 8 workers or cores. For this keras Deep Learning Model just install elephas from PyPI with, Spark will be installed through pyspark for you.

Well after the dataset has been imported with the help of kaggle.json file, we had removed all the outliers using the Interquartile range technique and then filled all the outliers with NaN values. These outliers had been replaced with the mean value before feeding into the model training section.

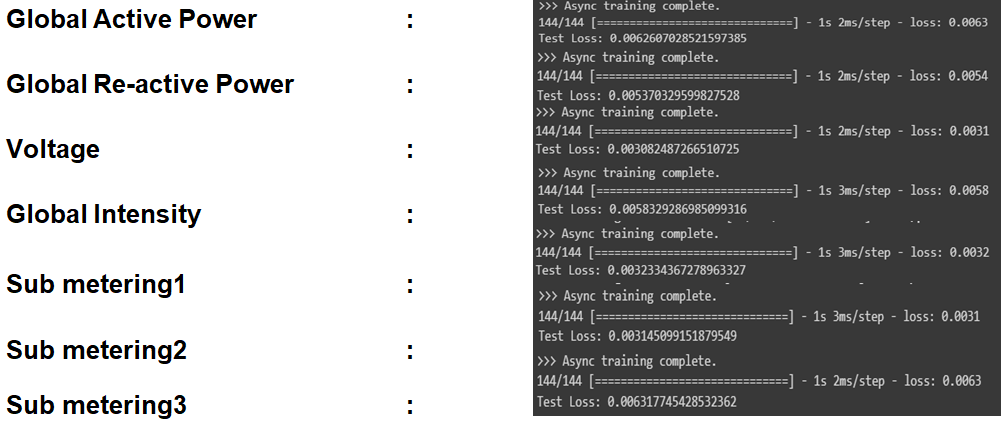
We first did a basic, Basic Spark integration using Elephas.

from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName('Elephas\_App').setMaster('local[8]')

sc = SparkContext(conf=conf)

And then defined the keras model which is supposed to be integrated. The dataset has been converted in a simple RDD and then it has been feed into the Spark Model. And the final evaluation takes through percentage of test data loss.



**2.4 Outcome**

In this Report, we practice to use the LSTM to fit and predict household electric power consumption. Consideration of the amount of input data is important to balance model accuracy and computation cost. For different attributes, different parameter has been used. The database can be used 30000 for fitting purposes and the rest has been used to validate the model. Also using the Elephas library for Big Data, has made it easier and faster to train the model in a significant manner.

|  |  |
| --- | --- |
| **Predictive Attribute** | **Minimum Error** |
| Global Active Power | MSE : 0.260 |
| Global Reactive Power | MSE : 0.003 |
| Voltage | MAE : 1.099 |
| Global Intensity | MAE : 1.492 |
| Sub\_meterring1 | MAE : 1.168 |
| Sub\_metering2 | MAE : 1.023 |
| Sub\_metering3 | MSE : 0.263 |

# Chapter 4

### **Future Work**

### Deep Neural Networks can be used for the perfection of the model. Maybe when in the future some other technology comes which makes it mode faster and efficient way to train the model. An exploration with other models could be used to find out which model tends to proclaim best in terms of accuracy.