**Temperature Prediction Project**

**Problem Statement:**

This data is for the purpose of bias correction of next-day maximum and minimum air temperatures forecast of the LDAPS model operated by the Korea Meteorological Administration over Seoul, South Korea. This data consists of summer data from 2013 to 2017. The task is to predict the Temperature using the provided dataset.

**Attribute Information:**  
1. station- used weather station number: 1 to 25  
2. Date- Present day: yyyy-mm-dd ('2013-06-30' to '2017-08-30')  
3. Present\_Tmax- Maximum air temperature between 0 and 21 h on the present day (Â°C): 20 to 37.6  
4. Present\_Tmin- Minimum air temperature between 0 and 21 h on the present day (Â°C): 11.3 to 29.9  
5. LDAPS\_RHmin- LDAPS model forecast of next-day minimum relative humidity (%): 19.8 to 98.5  
6. LDAPS\_RHmax- LDAPS model forecast of next-day maximum relative humidity (%): 58.9 to 100  
7. LDAPS\_Tmax\_lapse- LDAPS model forecast of next-day maximum air temperature applied lapse rate (Â°C): 17.6 to 38.5  
8. LDAPS\_Tmin\_lapse- LDAPS model forecast of next-day minimum air temperature applied lapse rate (Â°C): 14.3 to 29.6  
9. LDAPS\_WS- LDAPS model forecast of next-day average wind speed (m/s): 2.9 to 21.9  
10. LDAPS\_LH- LDAPS model forecast of next-day average latent heat flux (W/m2): -13.6 to 213.4  
11. LDAPS\_CC1- LDAPS model forecast of next-day 1st 6-hour split average cloud cover (0-5 h) (%): 0 to 0.97  
12. LDAPS\_CC2- LDAPS model forecast of next-day 2nd 6-hour split average cloud cover (6-11 h) (%): 0 to 0.97  
13. LDAPS\_CC3- LDAPS model forecast of next-day 3rd 6-hour split average cloud cover (12-17 h) (%): 0 to 0.98  
14. LDAPS\_CC4- LDAPS model forecast of next-day 4th 6-hour split average cloud cover (18-23 h) (%): 0 to 0.97  
15. LDAPS\_PPT1- LDAPS model forecast of next-day 1st 6-hour split average precipitation (0-5 h) (%): 0 to 23.7  
16. LDAPS\_PPT2- LDAPS model forecast of next-day 2nd 6-hour split average precipitation (6-11 h) (%): 0 to 21.6  
17. LDAPS\_PPT3- LDAPS model forecast of next-day 3rd 6-hour split average precipitation (12-17 h) (%): 0 to 15.8  
18. LDAPS\_PPT4- LDAPS model forecast of next-day 4th 6-hour split average precipitation (18-23 h) (%): 0 to 16.7  
19. lat- Latitude (Â°): 37.456 to 37.645  
20. lon- Longitude (Â°): 126.826 to 127.135  
21. DEM- Elevation (m): 12.4 to 212.3  
22. Slope- Slope (Â°): 0.1 to 5.2  
23. Solar radiation- Daily incoming solar radiation (wh/m2): 4329.5 to 5992.9  
24. Next\_Tmax- The next-day maximum air temperature (Â°C): 17.4 to 38.9  
25. Next\_Tmin- The next-day minimum air temperature (Â°C): 11.3 to 29.8T

**Please note that there are two target variables here:**

1) Next\_Tmax: Next day maximum temperature

2) Next\_Tmin: Next day minimum temperature

**Data Analysis:**

First, we will import few basic libraries that will be required for this project:



Now we will import the dataset using pandas -



Here, df will be our data frame in which we have imported the dataset.

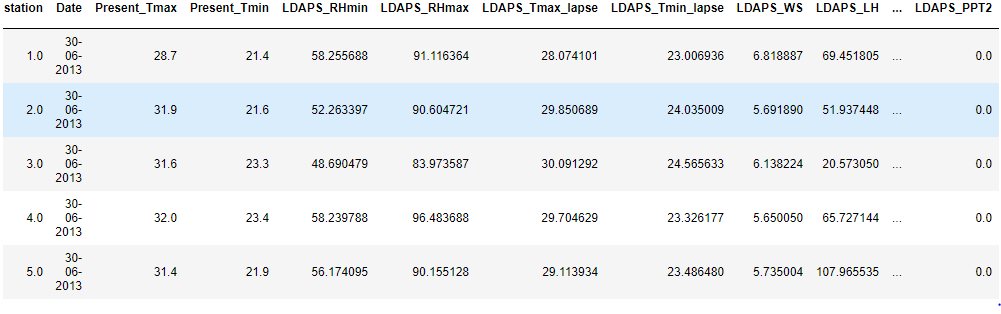
Let’s start exploring our dataset for the Temperature forecasting problem.

The first step for analysing the data is checking whether we have the right type of data in the columns or not. We can test this by checking the head of the dataset. To check the head of the dataset we can use ‘head()’ function available in pandas.

Code:



Output:



We have got the head of the dataset. But the question is what does it means to have right type of data in the columns?

We can explore one example from the above head: For ‘Present\_Tmax’ column, we can infer that numeric, i.e., Integer values should be present in this object. But further if we check the data type of the ‘Present\_Tmax’ column using dtypes() function of pandas and if it turns out to be object type, this means there is a miss match between the object type that is present and object type that is required. In that case, we will be required to convert the type and correct it. But for now, let’s get back to our current dataset and its scenarios.

Further, we will check what is the shape of the dataset. We can do this by using shape attribute in pandas.

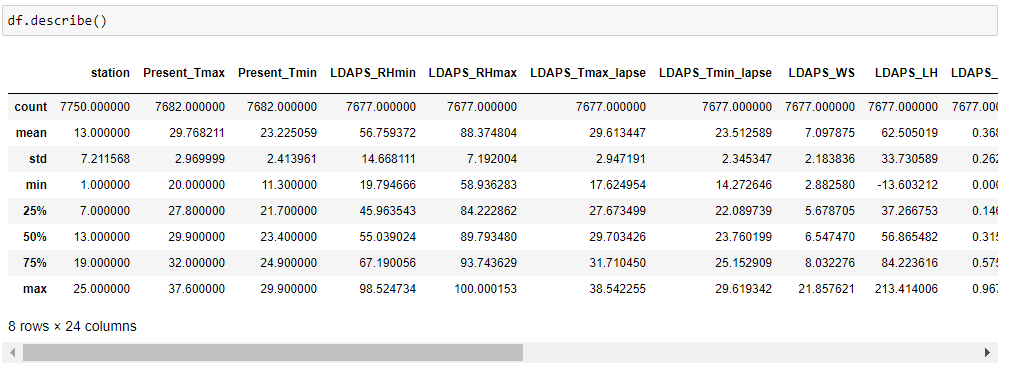
Code/Output:



So, we have 7752 rows and 25 columns in our dataset.

We can check the general information like Minimum value, Maximum value, Standard deviation, mean and other parameters of the dataset using describe function of pandas

Code/Output:



Now, we will find whether the dataset contains incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data.

Q) What can we do to check all of the above in our dataset?

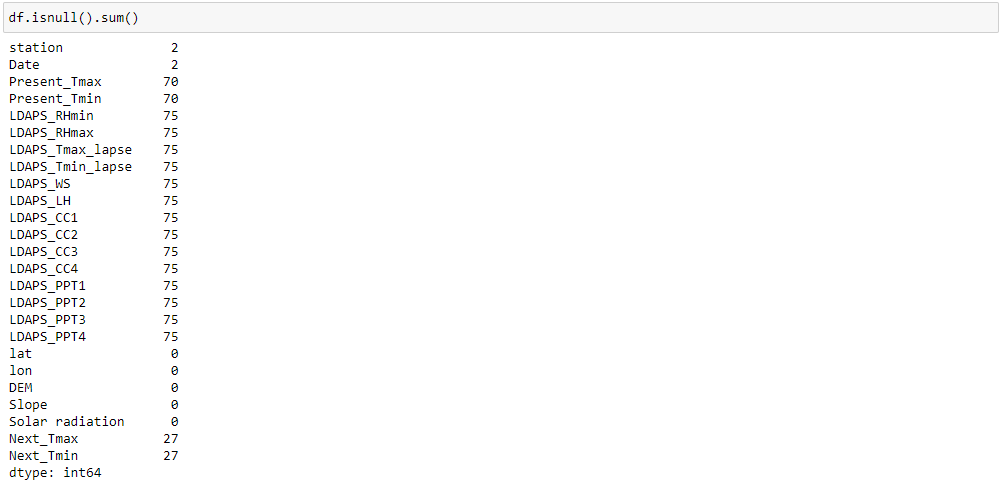
A) There are various techniques to explore our dataset for the anomalies. Few of them are checking null values, duplicate values, outliers, skewness and checking whether the data is normally distributed or not.

**Lets discuss them one by one:**

1. **Checking null values:**

We can use the pandas inbuilt function to check the null values in the dataset:

Code/Output:



We can see that there are null values in the dataset.

1. **Duplicate Rows:**

To check the duplicate rows in the dataset we will use ‘duplicated’ function available in pandas



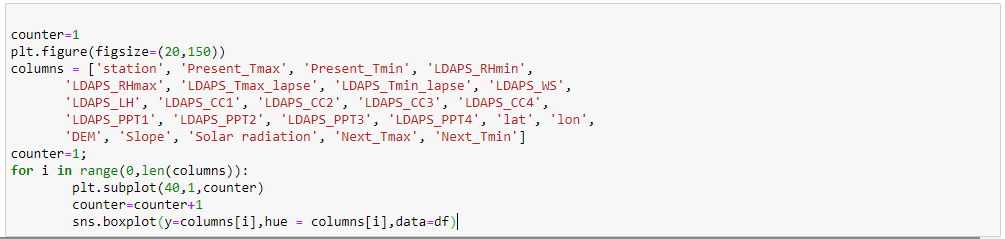
We have 0 duplicated rows in the dataset.

1. **Outliers:**

We will check the outliers in the Integer/Float type columns. Various methods are available to check outliers in the dataset and we will use one of those methods and i.e., by plotting Box-Plot to check the outliers in the dataset.

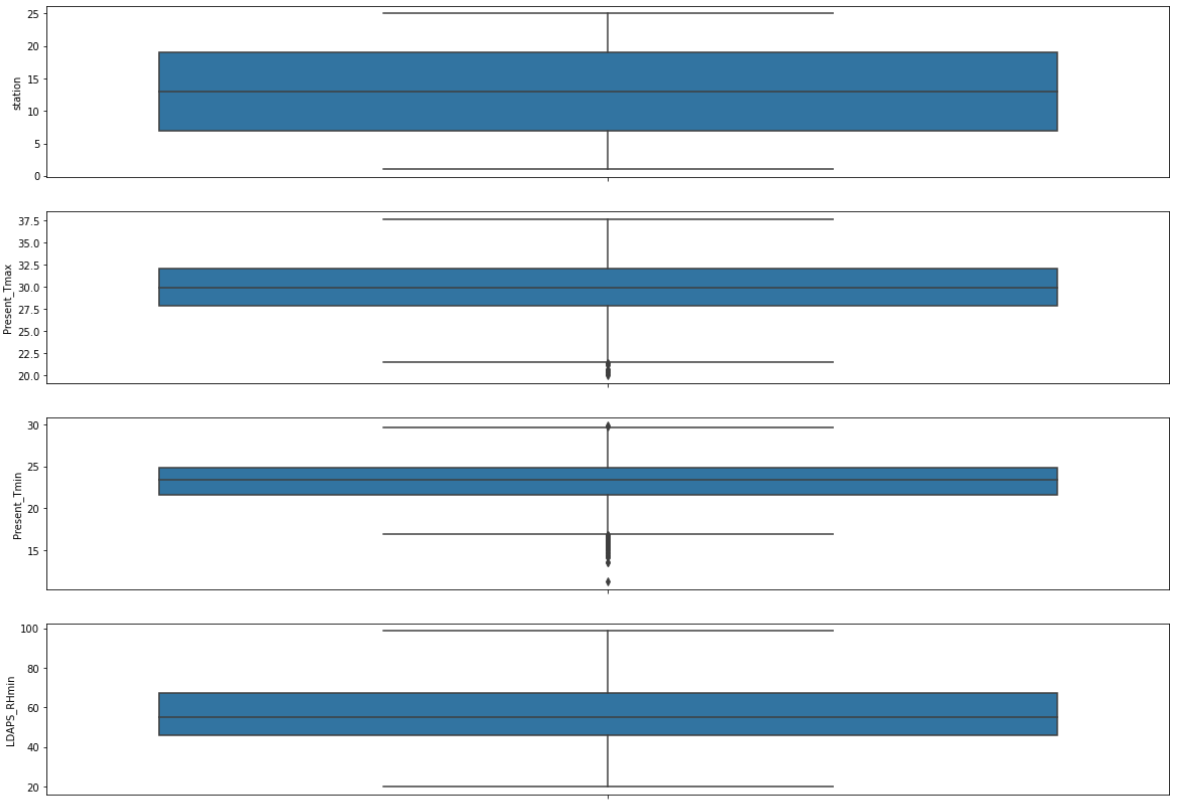
We will import the seaborn library to plot the boxplot.

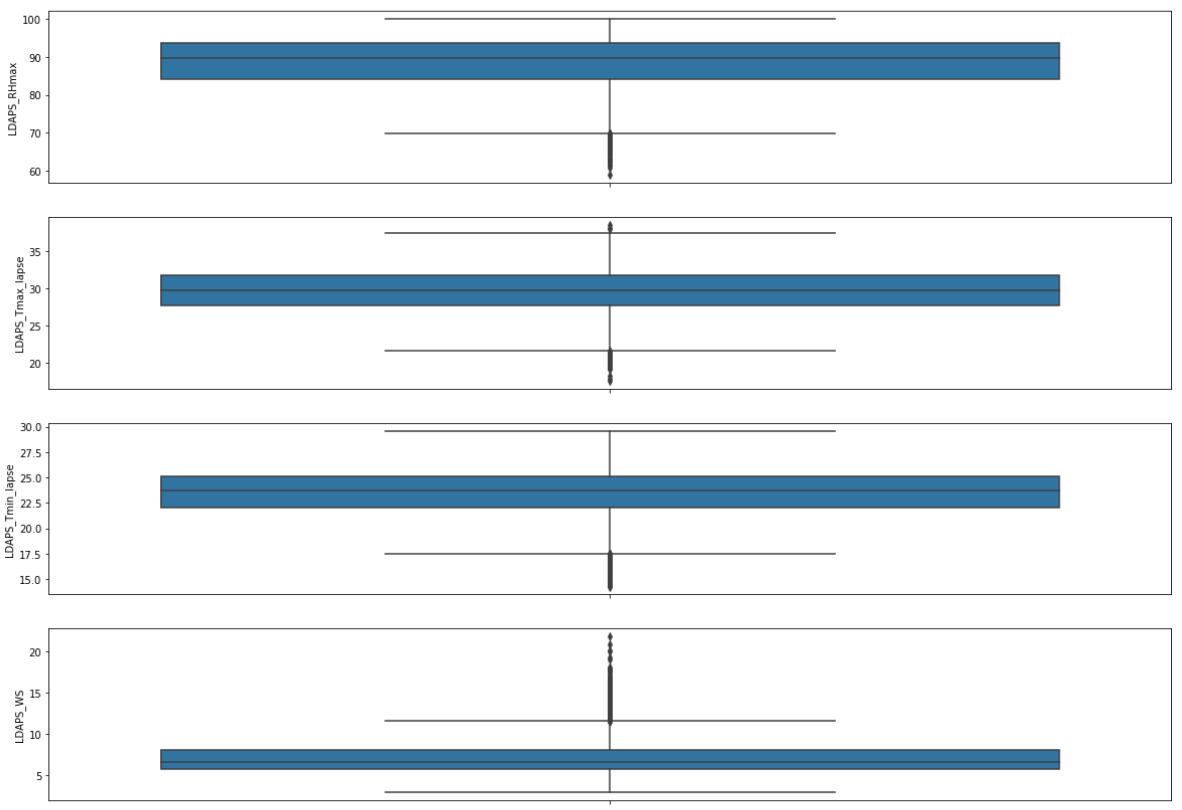
The code for checking the outliers for the numeric columns of our dataset:

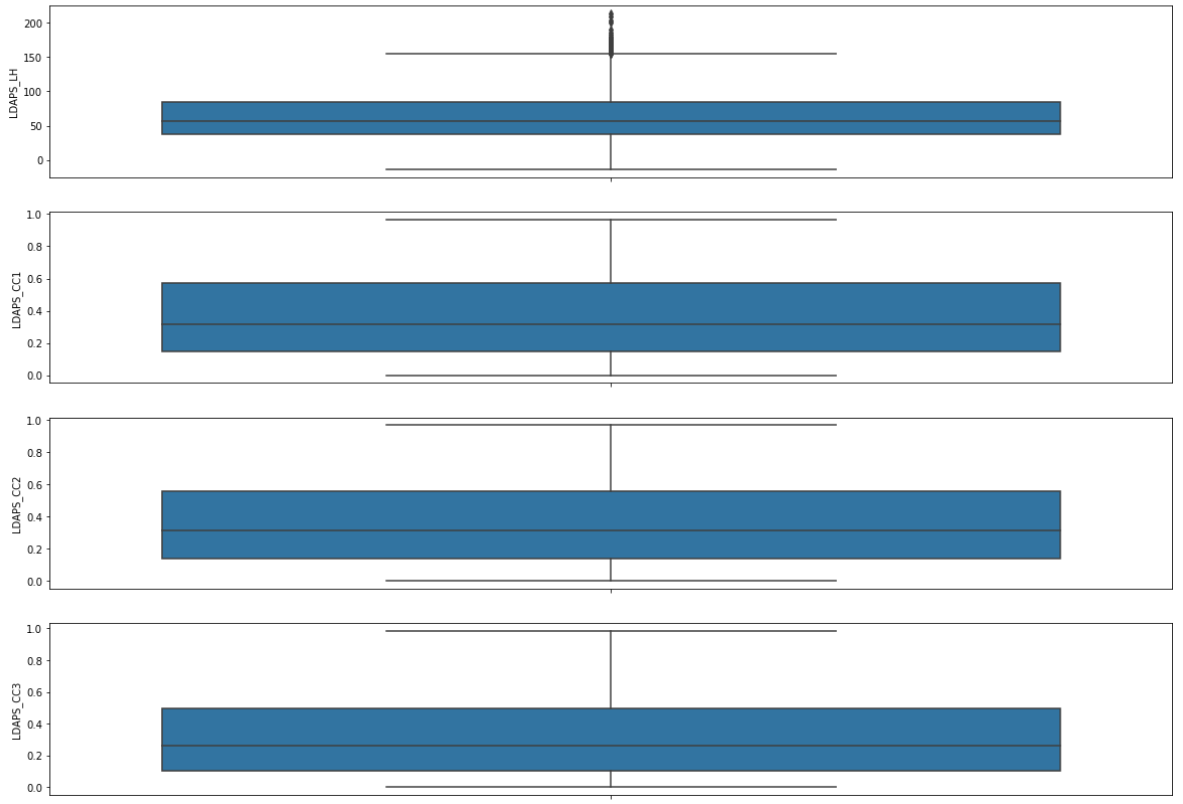


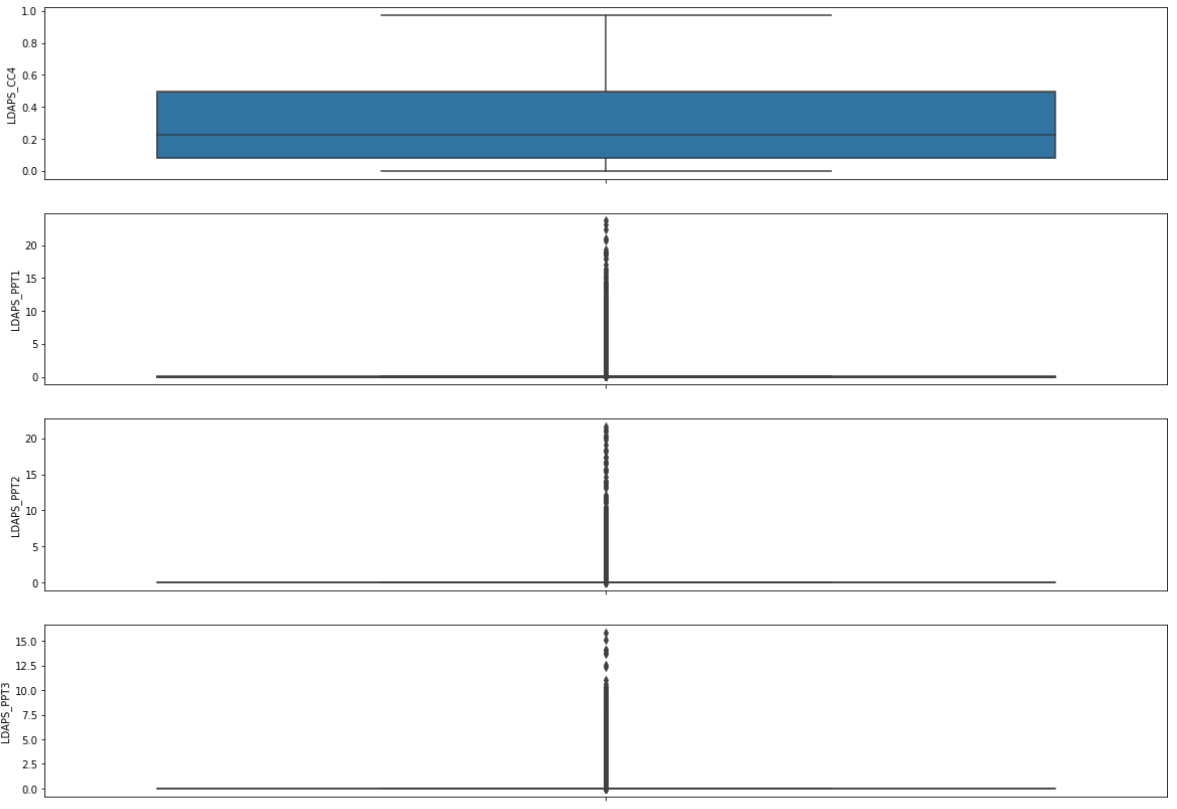
In the above code, we have stored the Numeric/Integer/Float columns name in a list to plot their box-plot one by one.

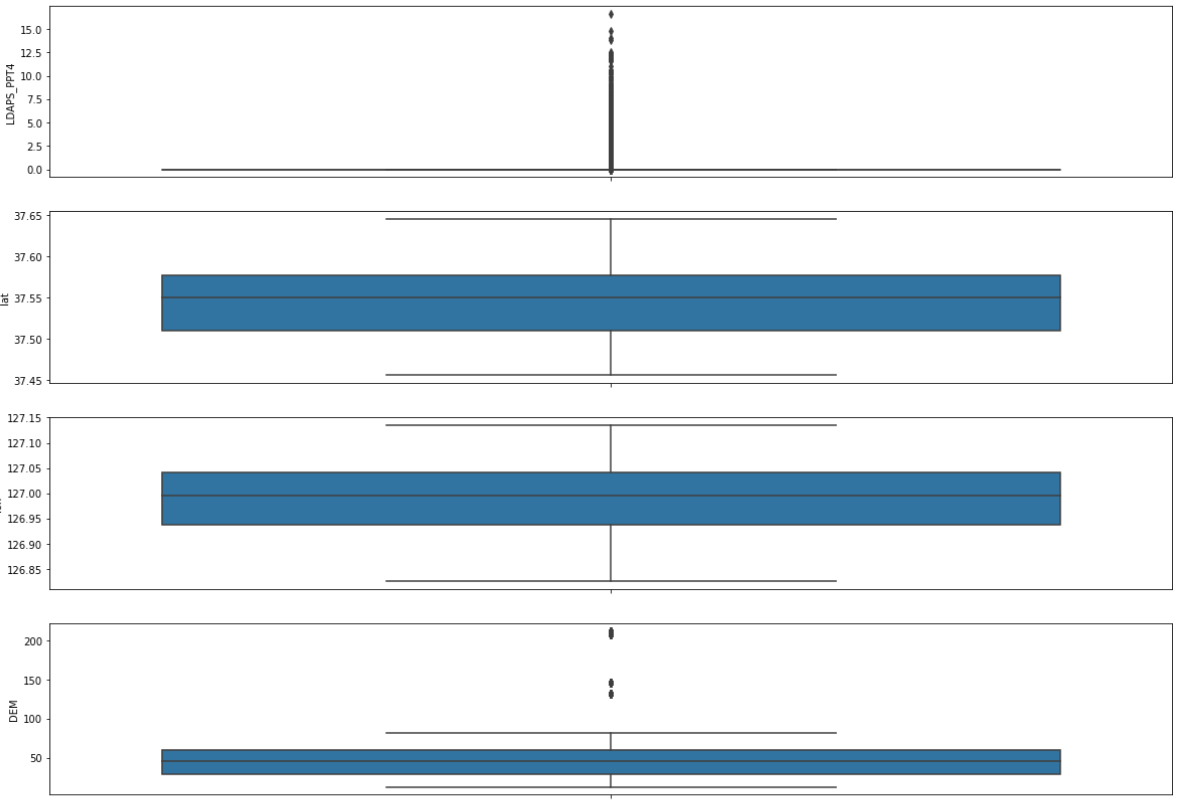
Output:

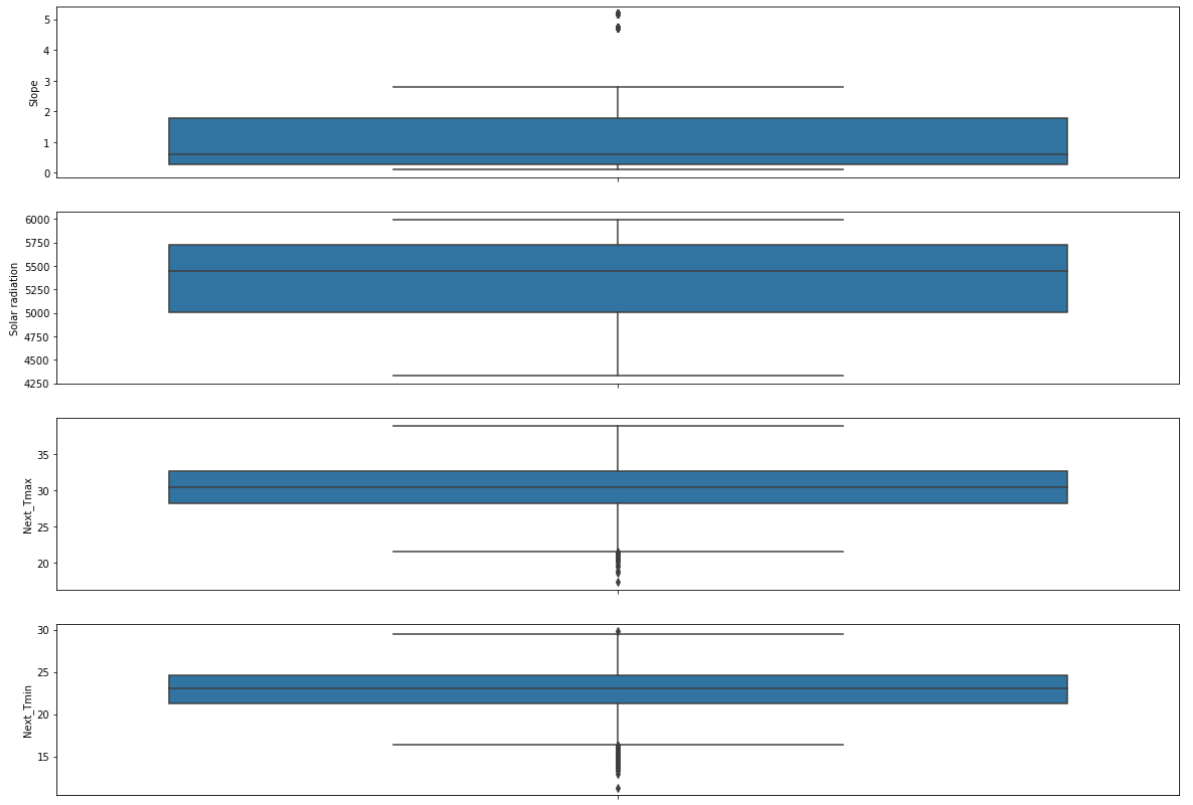












We can observe that outliers are present in Present\_Tmax, Present\_Tmin, LDAPS\_RHmax, LDAPS\_Tmax\_lapse, LDAPS\_Tmin\_lapse, LDAPS\_WS, LDAPS\_LH, LDAPS\_PPT1. LDAPS\_PPT2, LDAPS\_PPT3, LDAPS\_PPT4, DEM, SLOPE, Next\_Tmax, Next\_Tmin columns.

How did we got to know from the above plots that outliers are present? Well, if there are any values present in the plot outside of the upper limit and lower limit of the box plot, then they are considered as outliers.

If we need to calculate the upper/lower limit we can do it using below formula:

IQR = df[i].quantile(0.75)-df[i].quantile(0.25)

Upper\_limit = df[i].quantile(0.75) + 1.5\*IQR

Lowerl\_imit = df[i].quantile(0.25) - 1.5\*IQR

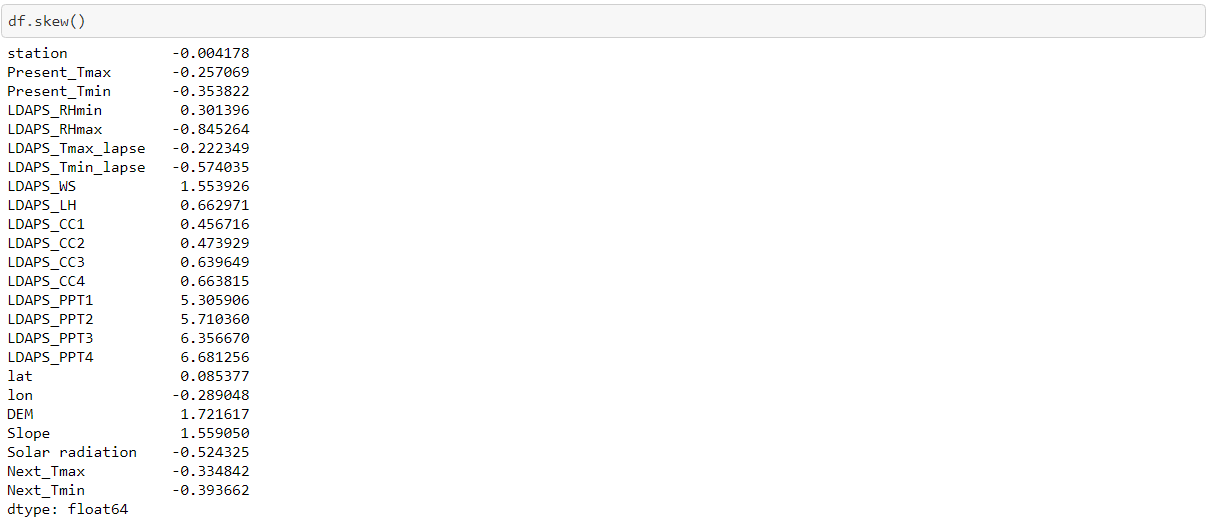
(Here ‘i’ denotes the column name)

Any values below the lower limit and any values above the upper limit will be identified as outliers.

1. **Skewness:**

Let’s check the skewness of the dataset using pandas function skew()

Code/ Output:



As we know that the skewness outside the range of -0.5 to +0.5 is not good for model building and in our dataset skewness for few columns is very high.

We can see that few columns are highly right skewed.

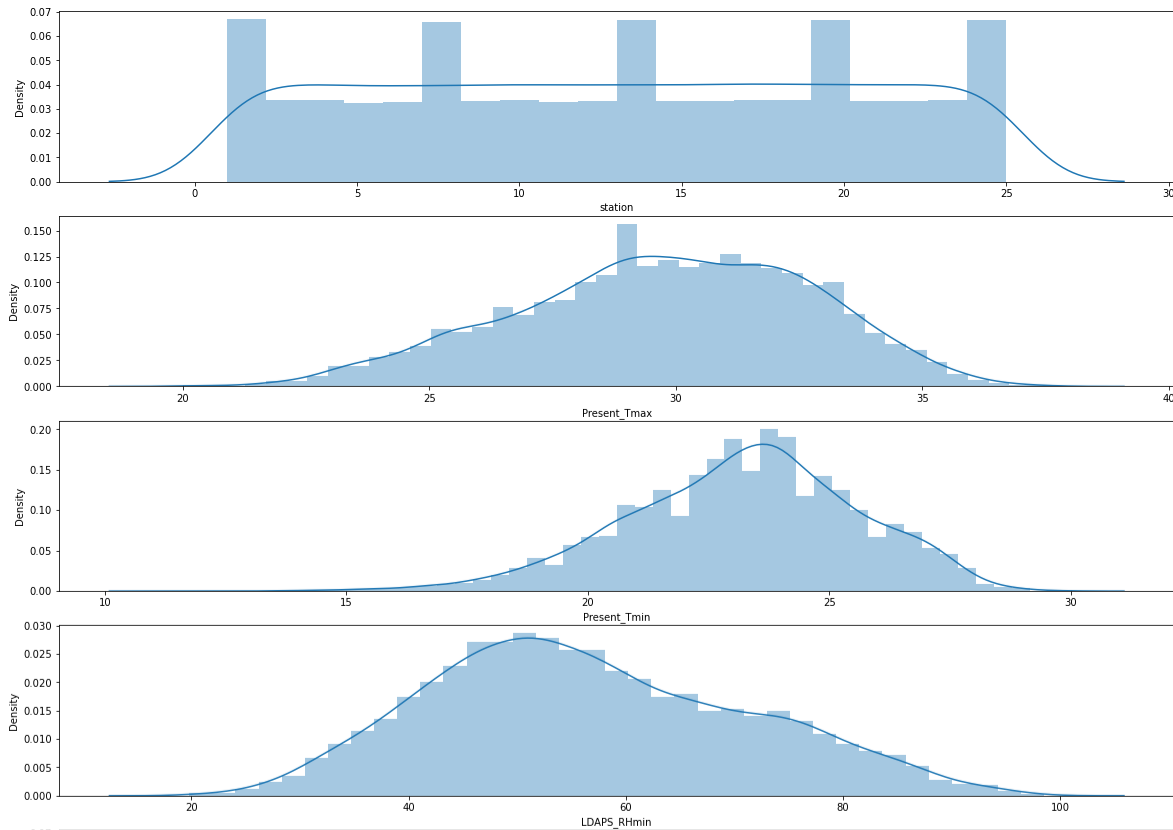
1. **Distribution of the data in columns:**

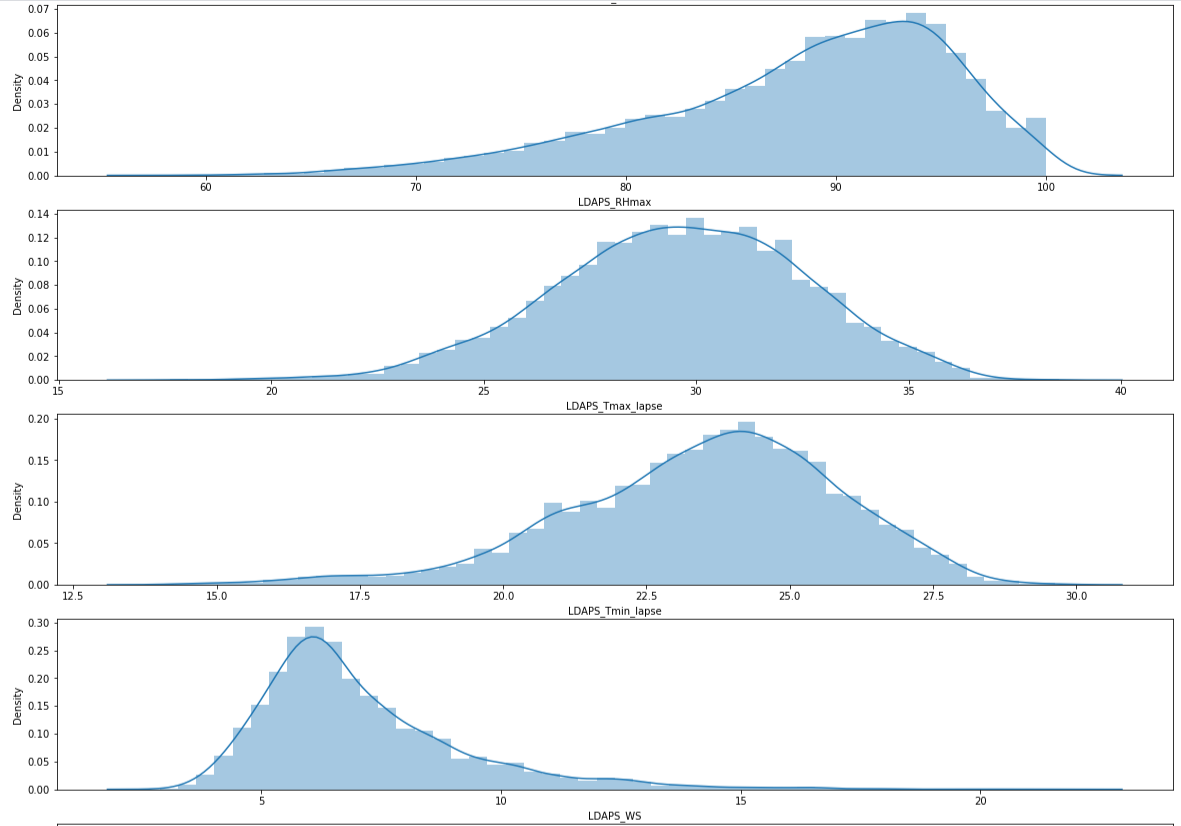
We can check whether the data is normally distributed or not using distance plot. We will use same sea born library to plot this graph

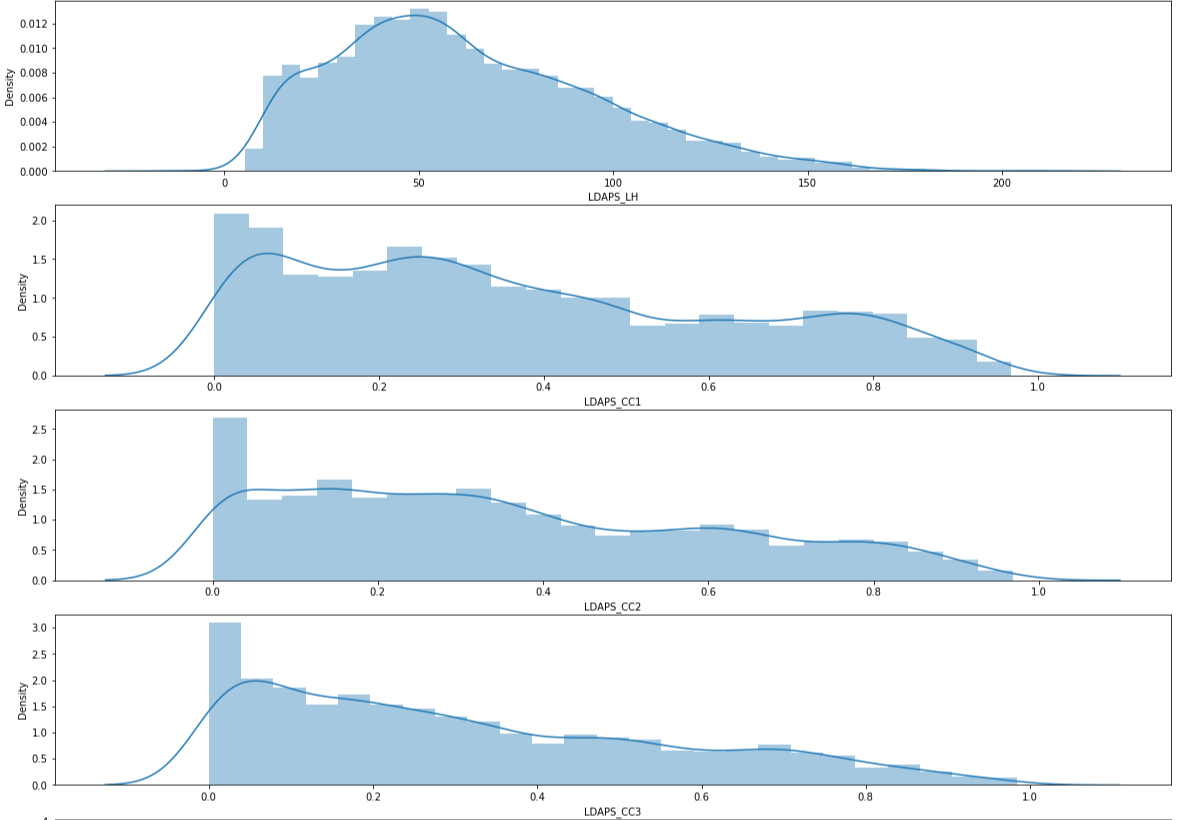
Code:

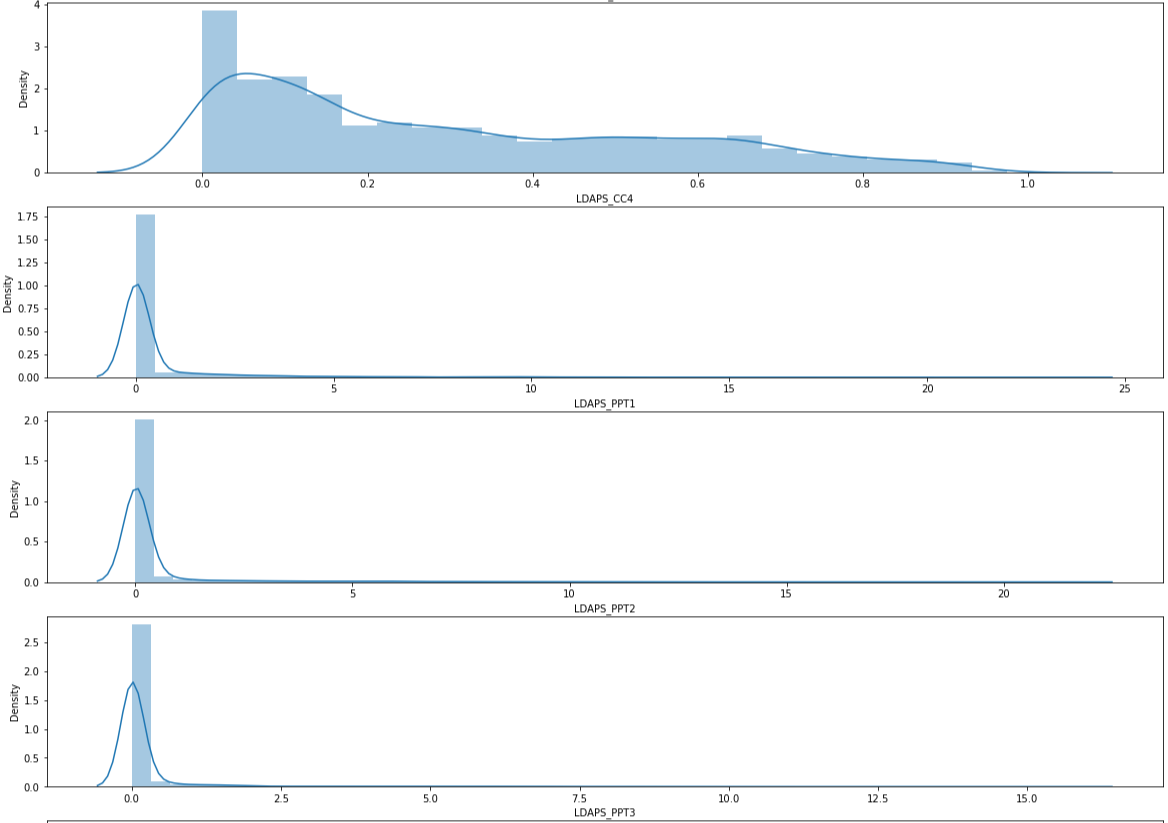


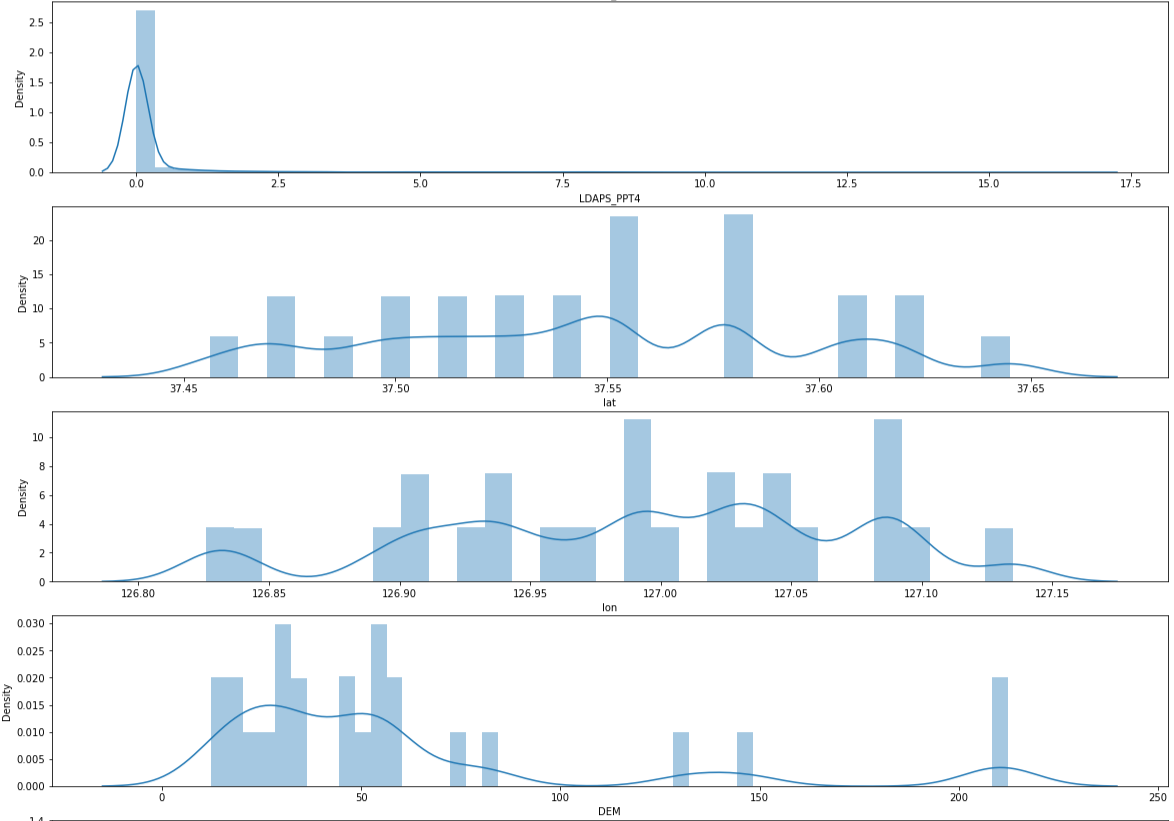
Output:

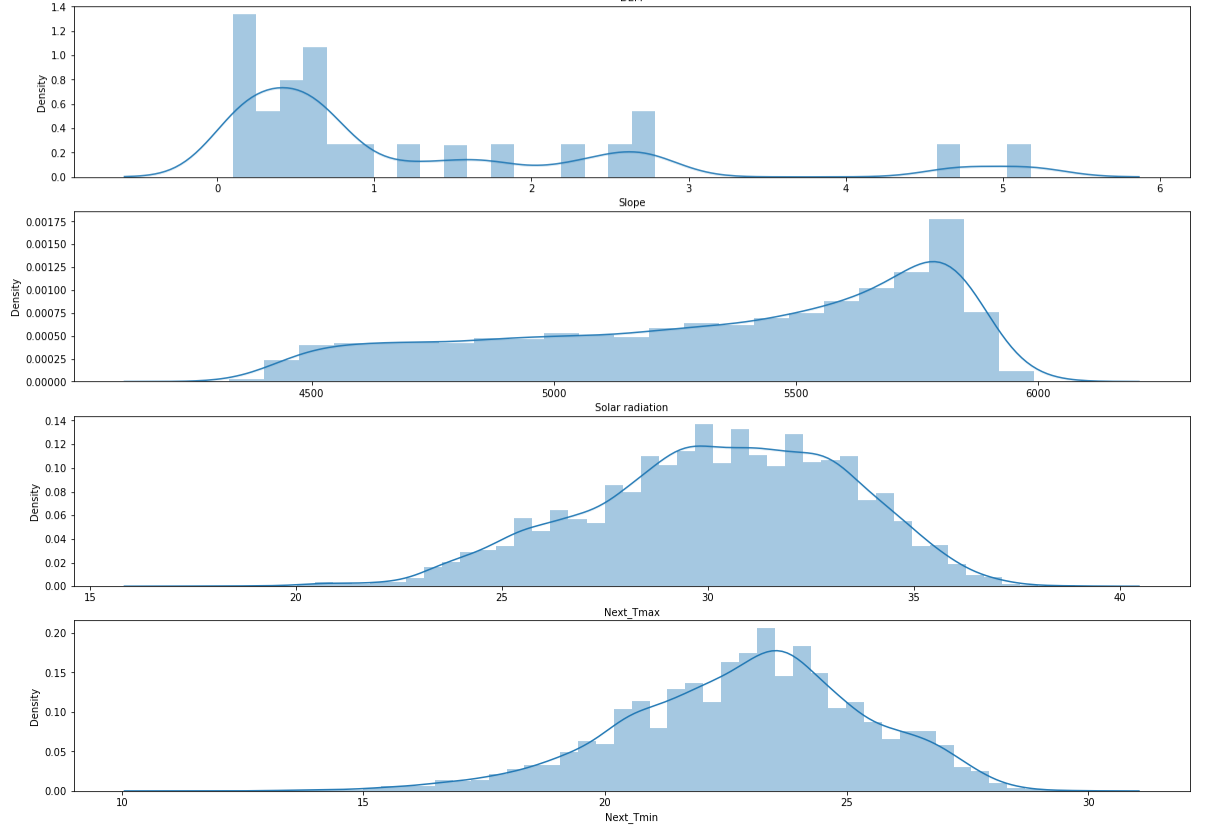












From above plots we can see for LDAPS\_CC1, LDAPS\_CC2, lat, lon, DEM, Slope columns have multiple peeks in the graph which means that the data is not normally distributed.

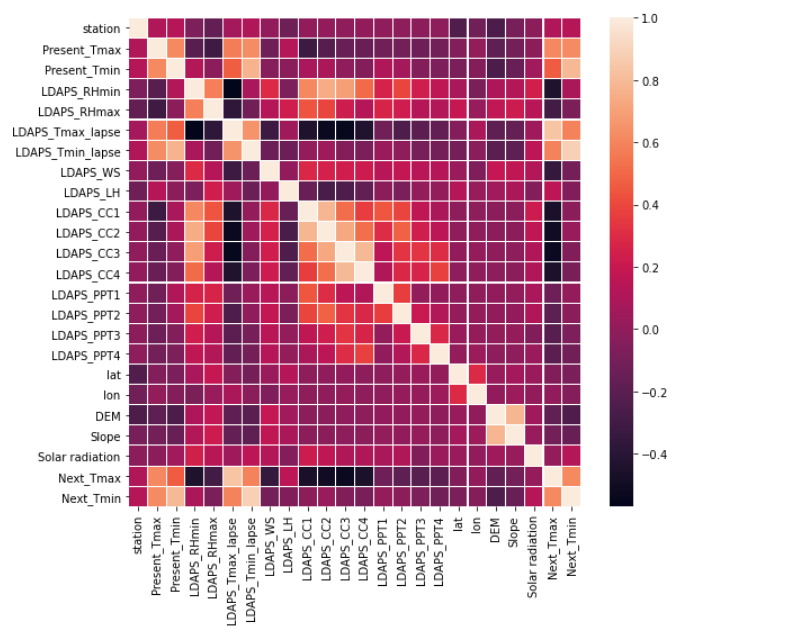
Now, after checking anomalies in our dataset, let’s check whether there is any correlation between the columns.

We will use ‘corr()’ method to find correlation and ‘heatmap’ plot to visualize the same.

Code:



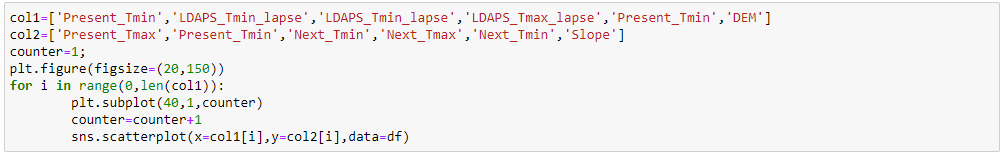
Output:



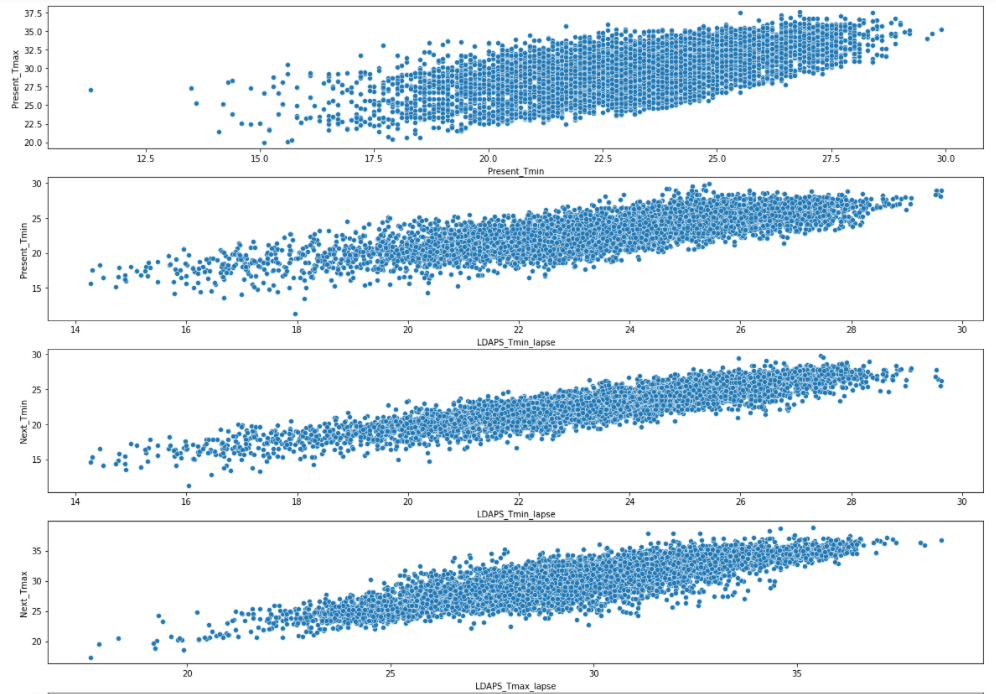
We can see that there is correlation between the Numeric columns Present\_Tmin and Present\_Tmax, LDAPS\_Tmin\_lapse and Present\_Tmin, LDAPS\_Tmin\_lapse and Next\_Tmin, LDAPS\_Tmax\_lapse and Next\_Tmax, Present\_Tmin and Next\_Tmin, DEM and Slope.

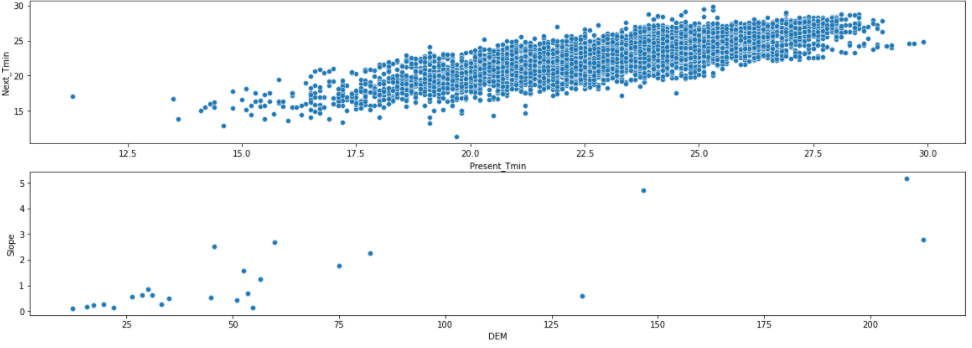
Lets plot scatter-plots between the correlated columns to get more clear view of the data.

Code:



Output:





We can observe linear relationship between LDAPS\_Tmin\_lapse and Present\_Tmin, LDAPS\_Tmin\_lapse and Next\_Tmin, LDAPS\_Tmax\_lapse and Next\_Tmax, Present\_Tmin and Next\_Tmin.

But the data in the above graph does not have a strong correlation between them. If the columns contained strong correlation, we would have dropped one of them as they would represent the same relationship and will not affect the model building.

**EDA Concluding Remark:**

We found that in our dataset we have null values/missing values, after that we also checked that our dataset does not contain duplicate rows. For numeric values, we have lots of outliers which also contributes in making our dataset skewed. Then we found that few columns are not normally distributed. We will solve all of these to make our dataset clean and increase the model efficiency in the next section. Please note, the more we analyse/clean the dataset the better the predicting models will perform.

**Pre-Processing Pipeline:**

Let’s start processing our data to move one step closer for model building. Previously, we found that there are many anomalies which are required to be removed or resolved in the dataset. We will process them one by one.

**Removing Null values:**

There are several methods for solving Null values issue. We can remove them or replace them with Mean/Median/Mode depending upon the data present in the column.

We will choose the remove method. let’s assume that data is not costly and we can afford the data loss. So, best method will be removing the rows where null or garbage values are present.

Code:



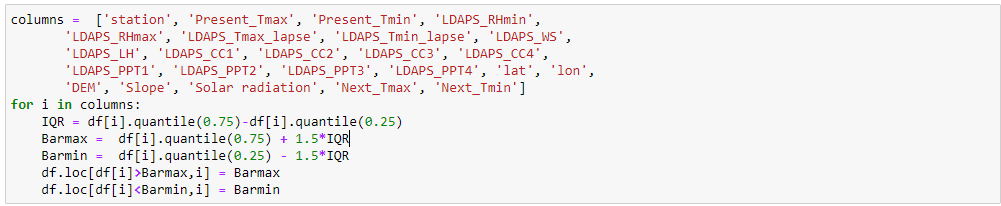
Here, we have used dropna() method to remove the null value rows.

**Outlier treatment:**

We found that in numeric columns there were huge numbers of outliers. Treating them is important for better prediction. There are several ways to treat outliers. Either we can remove them from the dataset or replace them with appropriate values.

But as we have discussed before we have to keep the data loss as low as possible. So, we will use the replacement method.

Code:

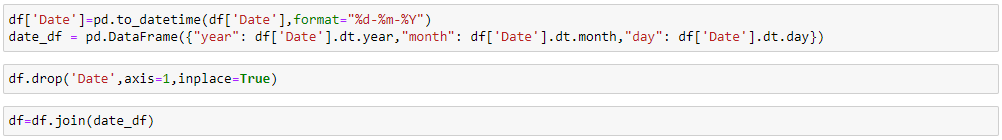


In the above code, we have replaced the outliers with upper limit and lower limit as per the Bar-plot.

**Converting the Date type column:**

Previously we observed that we have Date type column, now we will convert it into date, month, year columns for better data quality and better model building

Code:



We have dropped the original Date column from the dataset as now we have converted the column.

**Scaling the data:**

While visualising the dataset, we observed that few columns are not normally distributed. We can fix this issue by scaling the data. There are several ways to scale the data like using Standard scaler, Min-Max scaler etc. We will use Min-Max scaler in this case:

The Min-Max scaler is available in ‘sklearn’ library.

Code:



Did you notice that first we dropped few columns? And here the question arises why did we dropped them? Answer to this question is very simple, if we observe that we dropped only Target columns because we should avoid scaling the target variable because it can affect our model building efficiency and predictions.

**Removing Skewness:**

The skewness can be removed using several different methods. We can use log, sqrt, cbrt, binning etc methods for this. But here we will use binning method for our dataset. Here, we will again use python library to use the power\_tranform the dataset.

The ‘power\_transform’ is available in ‘sklearn.preprocessing’.

Code:



Now, we are ready to move onto the next phase, i.e., model building.

**Building Machine Learning Models:**

We will be building Regression Model for finding the temperature.

Did you notice that there are 2 target variables in the dataset? Generally, the predicting models are designed to work on only one target variable. So, what should we do?

There are multiple ways to solve this issue and one of them is using MultiOutputRegressor. We can import this from sklearn library as below:

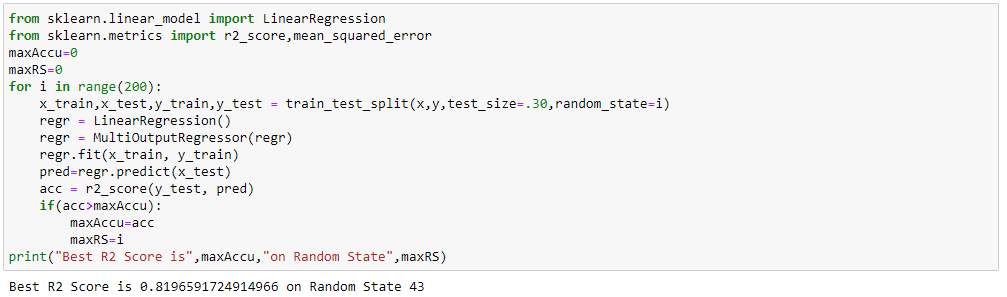
from sklearn.multioutput import MultiOutputRegressor

Let’s move ahead-

First, we need to split the dataset into training set and test set.



Here, we are using a model to test on which random state the accuracy will be best, so that we can do the train-test slit on that random state. Below we got random state as 43, so we will perform the train-test split at random state 43.



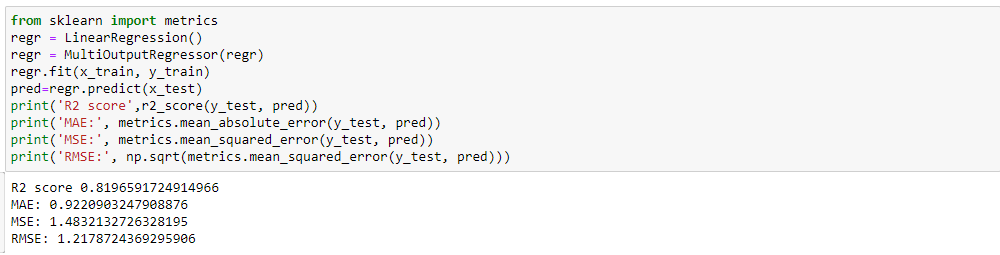
Above, we have kept the 70% of data as training data and we will perform testing on the remaining 30% of data.

Code:

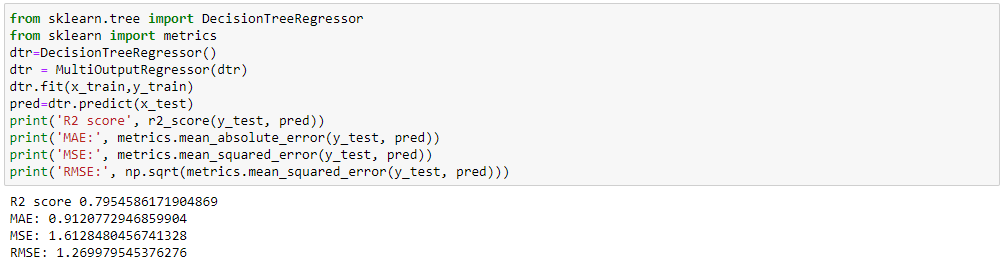


We will build many different models. We will not discuss how the model internally works but we will compare different parameters to conclude which model is predicting the best and then perform hyper tuning on the selected model.

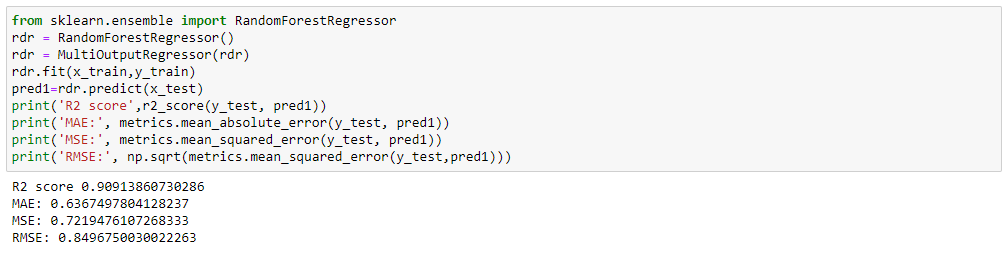
**Linear regression:**



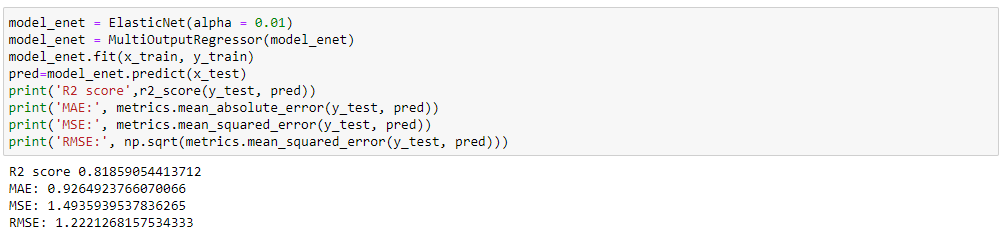
**Decision Tree Regressor:**



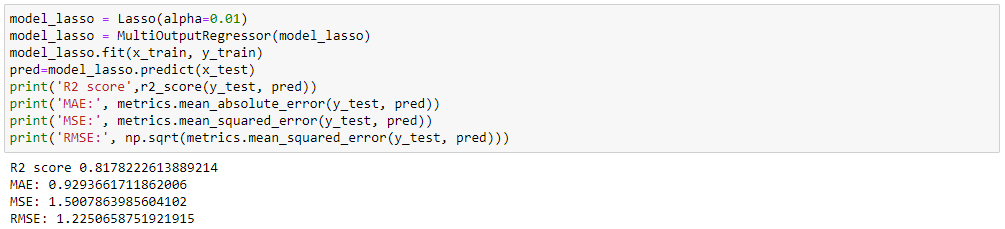
**Random Forest Regressor:**



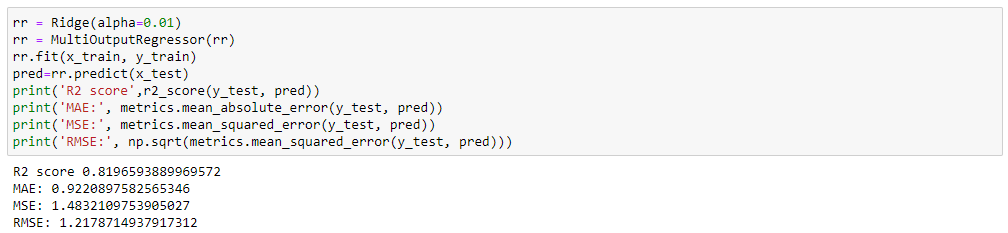
**Elastic Net:**



**Lasso:**



**Ridge:**



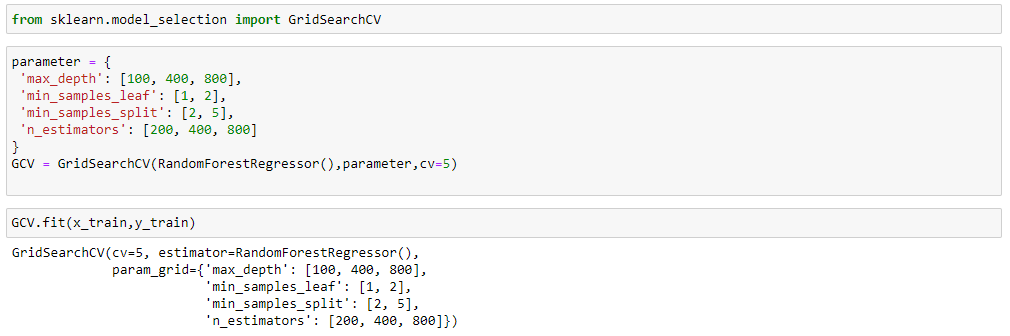
Let’s summarize R2 Score for the above models in a table to get more clear view and compare the different models.

|  |  |
| --- | --- |
| **Models** | **R2-score** |
| **Linear Regression** | 0.81 |
| **Decision Tree** | 0.79 |
| **Random Forest** | 0.90 |
| **Elastic Net** | 0.81 |
| **Ridge** | 0.81 |
| **Lasso** | 0.81 |

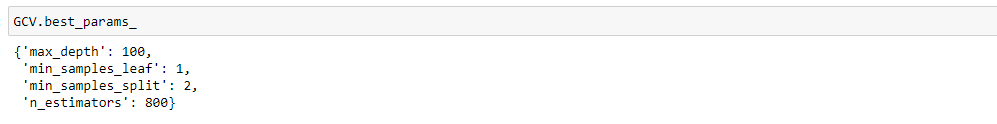
If we observe the above table carefully, we can see that Random Forest model is performing the best by having the highest R2 score compared to other models.Hence, let’s Hyper tune the Random Forest model further.

We will use ‘GridSearchCV’ from ‘sklearn’ library for the Hyper tuning, It tries different combinations of the parameters that we will pass and check which parameter suits the model best:

Code:

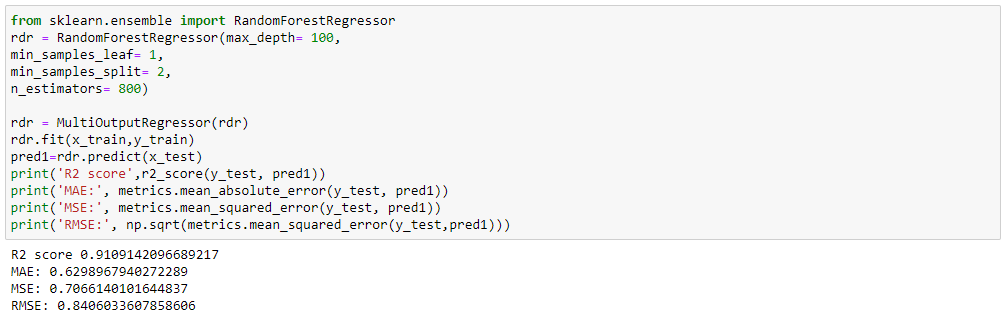


Now let’s check the best parameters for the model:



We have the best parameters now, let’s build our final model i.e., Random Forest model using these parameters:

Code:



**Concluding Remarks**

After several iterations of exploring and conditioning on the data, we have built a useful algorithm for predicting the Temperature. The technique applied in this project is an implementation of a simple machine learning model, the Random Forest.

We can conclude that Random Forest algorithm is performing best among all the algorithms.