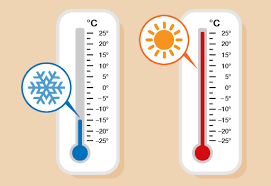
**PROJECT: TEMPERATURE FORECAST**



NAME : SHUBHAM SHARMA

EMAIL ID : SHUBHAM040297@GMAIL.COM

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **PARTICULARS** | **PAGE NO.** |
| Problem Introduction | 3 |
| ML Workflow | 4 |
| Data Analysis | 5 |
| EDA Concluding Remarks | 8 |
| Pre-processing Pipeline | 9 |
| Building Machine Learning Models | 15 |
| Concluding Remarks | 19 |

**PROBLEM INTRODUCTION**

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. It consists of summer data from 2013 to 2017.

The input data is largely composed of the LDAPS model's next-day forecast data, in-situ maximum and minimum temperatures of present-day, and geographic auxiliary variables. There are two outputs in this data i.e., next-day maximum and minimum air temperatures. Hindcast validation was conducted for the period from 2015 to 2017.

It is supervised because we have both the features and the target that we want to predict. The target makes this a regression task because it is continuous. During training, we will give multiple regression models for both the features and targets, and it must learn how to map the data to a prediction. Another reason for it being a regression task is that the target value is continuous as opposed to discrete classes in classification.

That sums up the background of the project so let’s begin!

**ML WORKFLOW**

Before we jump right into programming, I will outline the objectives of the project. The following steps are structured on my machine learning workflow. Keeping in mind our problem and our model the steps are:

* State the question and determine the required data (completed)
* Acquire the data
* Identify and correct missing data points/anomalies
* Prepare the data for the machine learning model by cleaning/wrangling
* Establish a baseline model
* Train the model on the training data
* Make predictions on the test data
* Compare predictions to the known test set targets and calculate performance metrics
* If performance is not satisfactory, adjust the model, acquire more data, or try a different modeling technique
* Interpret model and report results visually and numerically

**DATA ANALYSIS**

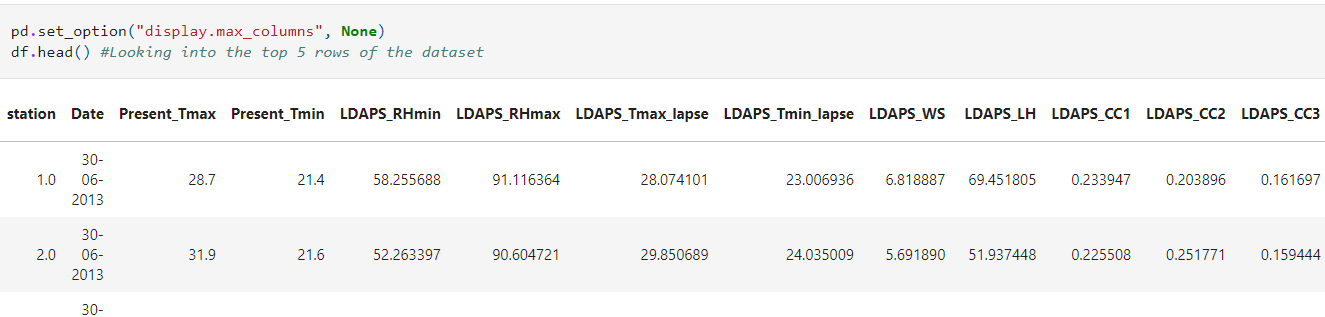
The first step is to acquire the data for the project. The data for the project has been acquired through GitHub.

The link for the data has been attached below.

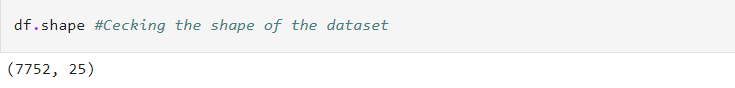
Link: <https://github.com/dsrscientist/Dataset2/blob/main/temperature.csv>

After importing some important libraries and modules, the code below loads in the CSV data which I store into a variable that will be used later.

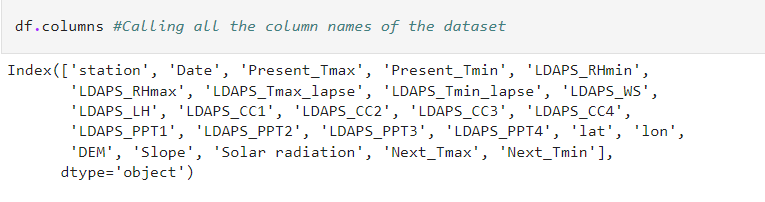
Since the data has been acquired, our first step will be to take a look at the general structure of the dataset and the columns in order to understand the features effectively.



Checking the shape of the data set



Now, we look into the columns of the data set

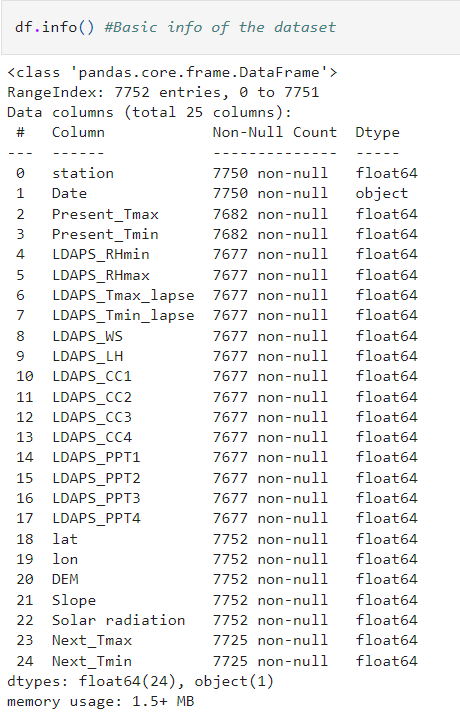


The attribute information is:  
1. station - used weather station number: 1 to 25  
2. Date - Present day: yyyy-mm-dd ('2013-06-30' to '2017-08-30')  
3. Present\_Tmax - Maximum air temperature between 0 and 21 h on the present day (Â°C): 20 to 37.6  
4. Present\_Tmin - Minimum air temperature between 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

**EDA CONCLUDING REMARKS**

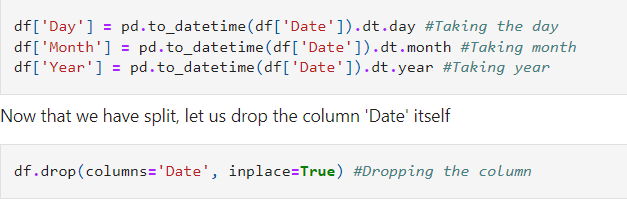
We see from the data that there are 7752 rows of data with 25 columns. The features all being float type with only ‘Date’ being object type.

Looking through the data (shown above), I noticed several missing data points, which is a great reminder that data collected in the real world will never be perfect. Missing data can impact analysis immensely, as can incorrect data or outliers. To identify anomalies, we can quickly find missing using the info() method on our Data-Frame.

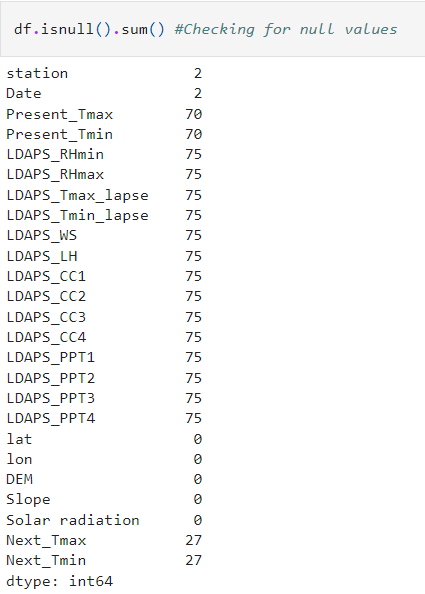


**PRE-PROCESSING PIPELINE**

We shall convert ‘Date’ with Day, Month and Year separate thereby converting the column to float type. Then we drop the ‘Date’ column as we have already created different float type columns for the date.

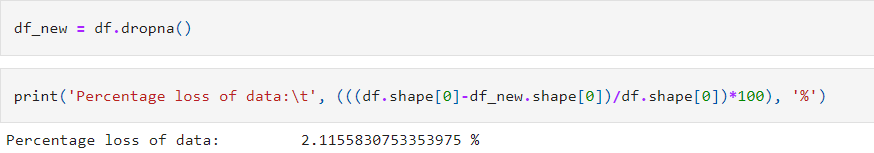


Now, we treat for the null values in the data set. Also, we can use the “.isnull()” and “.sum()” methods directly on our data-frame to find the total amount of missing values in each column. We observe that there are 75 rows of data with null values.

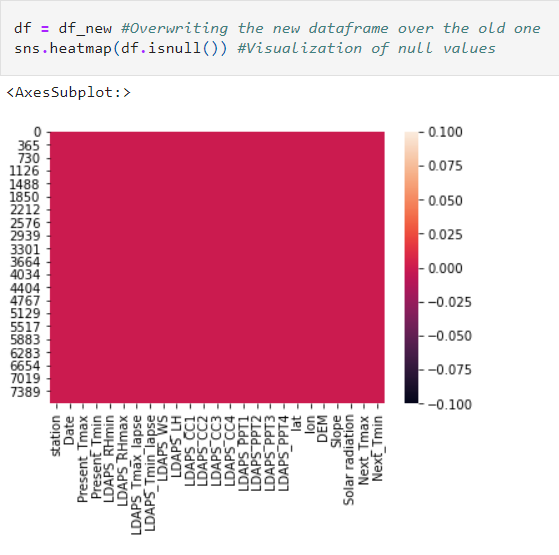


Since we have more than 7000 rows of data, we can just drop the rows with null values, so we attain a complete data set for the modelling. This causes data loss of only 2% which can be considered acceptable.

Now if you remember we also had missing values which we saw earlier in our dataset. So I think we will call the dropna(), just in case there are any other missing values in our dataset:

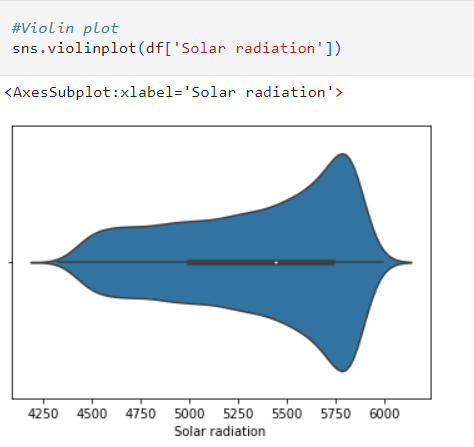
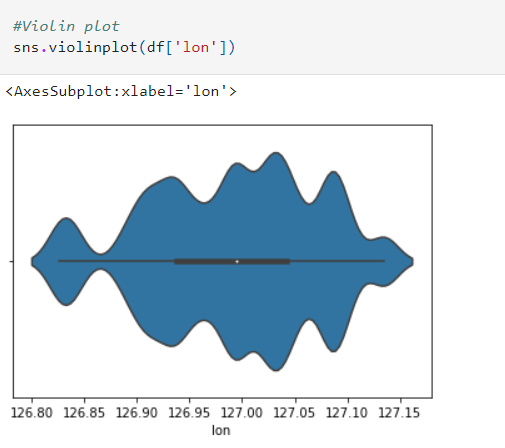


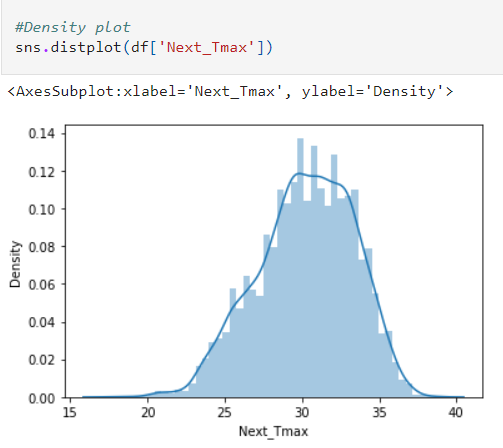
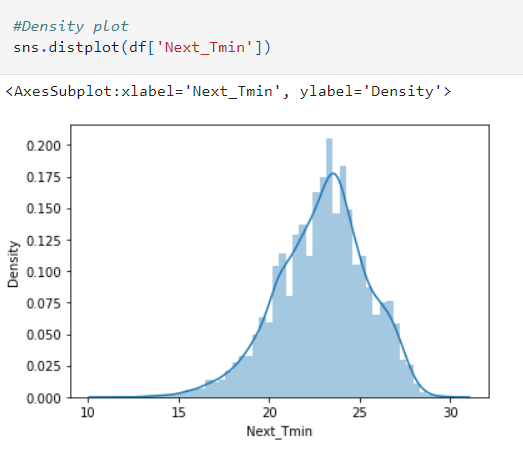
We can check the dataset again to confirm that there are no more null values present. Better is if we use heatmap visualization for better clarity of the presence of null values present in the data set.



With the completion of the above steps, we have converted the data into a no null data set with all columns being numerical type.

The next step is to visualize the columns independently to better understand the distribution of the data. This will be done using the violin plot.

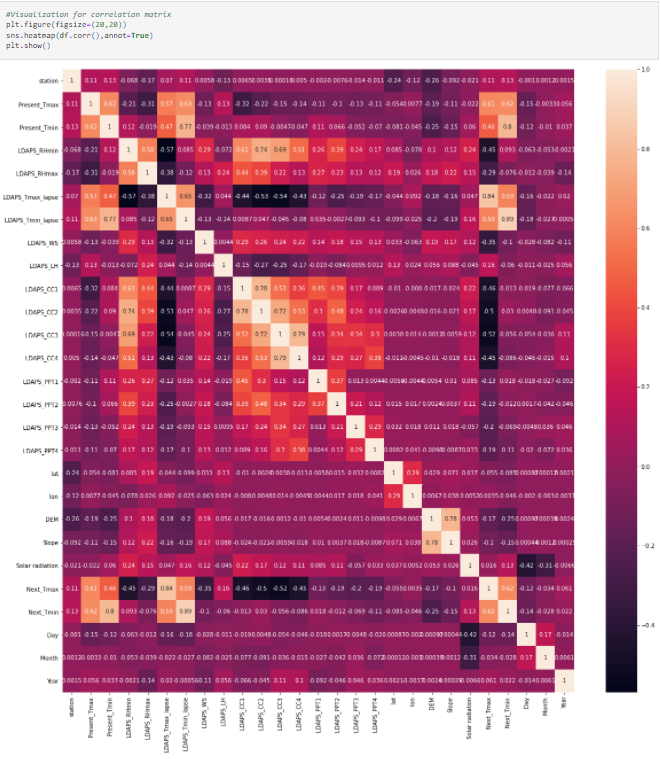
 

It looks like we are ready for the next steps, setting up our target and features, train/test split, and establishing our baseline.

Something I like to do while working with regression problems is to look at the cleaned data frame and to see if we can truly use one column as our target and the others as our features.

One method of determining that is by plotting a correlation matrix, just to get an understanding of how related each column is to each other:

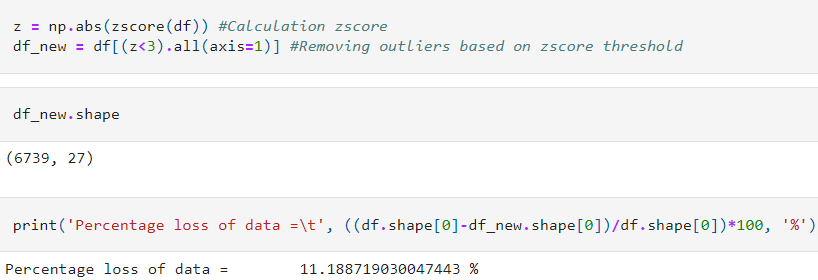


As we can see the columns are highly correlated to one another. So, we should have pretty strong & positive predictions just from glancing at this plot.

However, we still need to remove outliers and skewness as we have discussed earlier that their presence might negatively impact the model immensely.

With the steps, we find that the data is complete and all numerical. However, there is the presence of outliers and skewness. Therefore, we have to treat the data for it.

* We use the z-score method to treat for outliers. The Z-score method to remove outliers is one where the data is scored on a scale. In this particular dataset, we keep a range of +/-3. Using this method, we can see that some data has been eliminated from the dataset. To see the amount of data that has been eliminated, we produce a percentage difference. This produces a data loss of 11%.



* We split the data into target and feature variables before we treat it for skewness keeping in mind there are two target variables in this data set. The target, also known as ‘y’, is the value we want to predict, in this case, the next day’s minimum and maximum temperature, and the features are all the columns (minus our target) the model uses to make a prediction

We assign ‘x’ -> feature variable

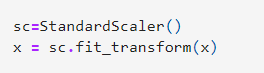
‘y’ -> target variable



* The best distribution of data to produce an efficient model is when it is uniformly distributed. We use the power transform to treat data for skewness.



Since the data has been treated for outliers and skewness too, we move forward to the modelling of the data. We reach this state after scaling the feature variables of the data by using the Standard Scaler method.



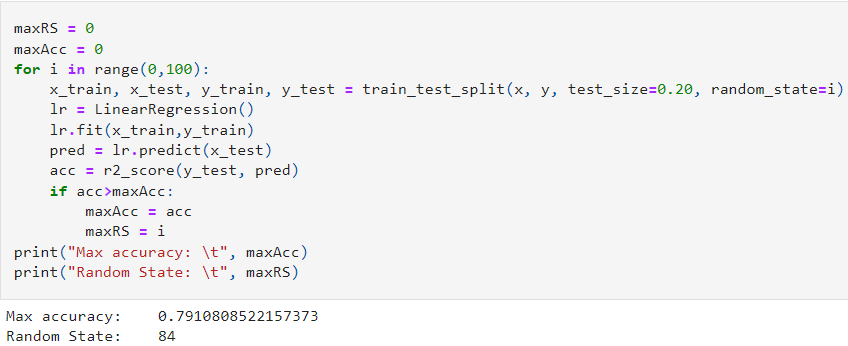
**Building Machine Learning Models**

We are now on the final step of the data preparation part of our ML workflow: **Splitting data into training and testing sets.**

During training, we let the model ‘see’ the answers, in this case, the actual temperature, so it can learn how to predict the temperature from the features. There is a relationship between all the features and the target value, and the model’s job is to learn this relationship during training. Then, as it comes time to evaluate the model, we ask it to make predictions on a testing set where it only has access to the features (not the target)!

Generally, when training a regression model, we randomly split the data into training and testing sets to get a representation of all data points.

Before we start with the model preparation, we check for the best random state using the Linear Regression scoring the best r2\_score method. We do this twice, one for each target variable respectively.



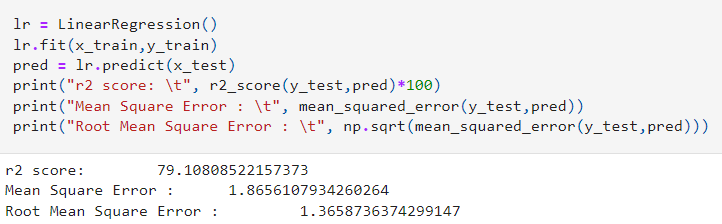
Using this random state, we split the data set using train-test split for the random state that we have just calculated.



After all the work of data preparation, creating, and training the model can be carried out easily using scikit-learn. For this problem, we could try a multitude of models, and in this situation, we are going to use six different models:

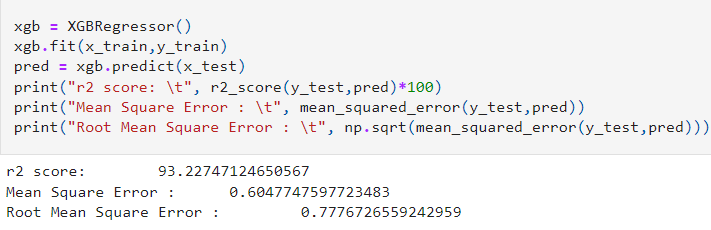
* **Linear Regression**

Linear regression is a statistical approach that models the relationship between input features and output. Our goal here is to predict the value of the output based on the input features.



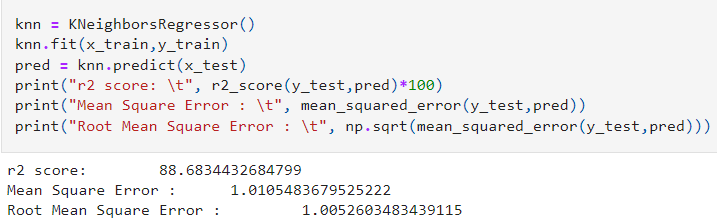
* **XG Boost Regression**

**XGBoost** is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible**, and **portable**. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.



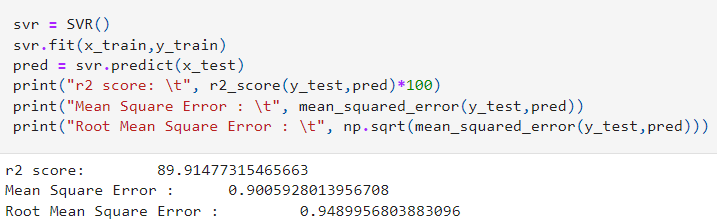
* **K-Nearest Neighbour Regression**

KNN calculates the distance from all points in the proximity of the unknown data and filters out the ones with the shortest distances to it. As a result, it’s often referred to as a distance-based algorithm.



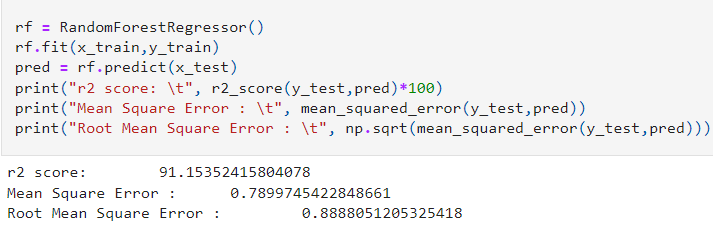
* **Support Vector Machines**

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.



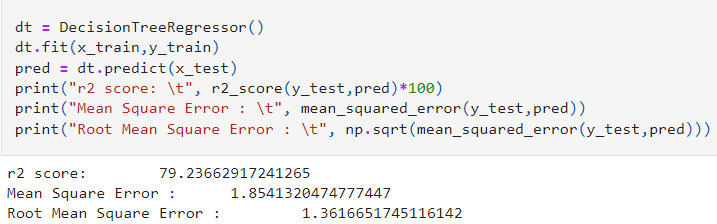
* **Random Forest Regressor**

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique commonly known as **bagging**.



* **Decision Tree Regressor**

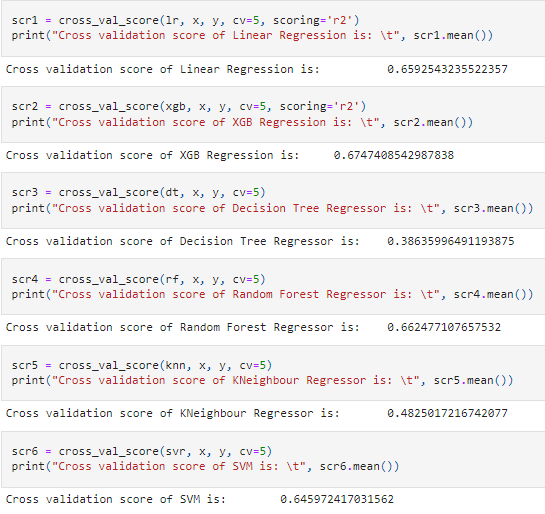
It is run completely through the entire tree by answering True/False questions till it reaches the leaf node. The final prediction is the average of the value of the dependent variable in that particular leaf node. Through multiple iterations, the Tree is able to predict a proper value for the data point.



We note that the XGB Regressor produces the maximum r2\_score of 93.22% with the least mean square error value which is good.

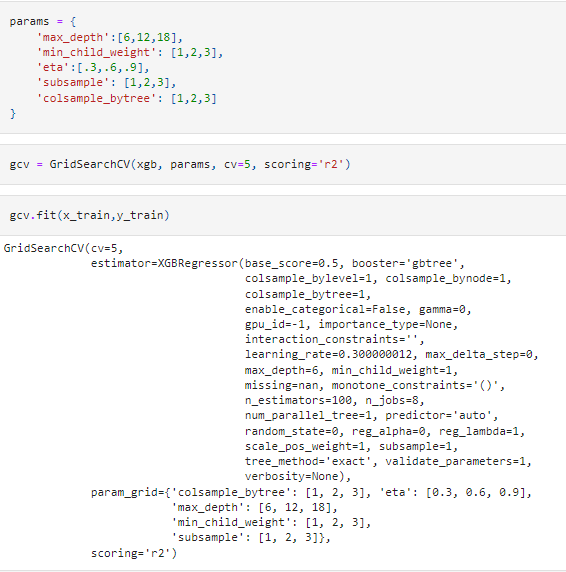
**Concluding Remarks**

We can conclude that we now have the highest accuracy model. However in machine learning, we couldn’t fit the model on the training data and say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the cross-validation technique. We can do a cross-validation check to better understand the model accuracy.



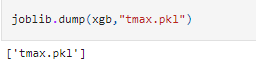
With XGB Regressor performing the best in cross-validation, we see that the model works best for the dataset among others.

However, we can still use various parameters and algorithms to tune the best model to marginally increase its efficiency. This method is known as hyperparameter tuning. Two methods to do the tuning are the Grid Search method and Randomized CV method. It is a hit and trial method and so we try adding more parameters to the Grid Search CV to find the best tuned model.



Now, we try this method to calculate the r2 score to check if the efficiency has increased.

In the end, we save the model with the highest efficiency using a pickle.



With this, we conclude our project that calculates the next day’s minimum and maximum temperature based on the dataset with an efficiency of **93.22%.**