**Temperature Forecast Project using ML**

**1.Project Description**

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 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 (i.e. next-day maximum and minimum air temperatures) in this data. Hindcast validation was conducted for the period from 2015 to 2017.

**Attribute Information:**

**For more information, read [Cho et al, 2020].**

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

You have to build separate models that can predict the minimum temperature for the next day and the maximum temperature for the next day based on the details provided in the dataset.

**Dataset Link-**

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

**2.** **Data Analysis**

**Data Loading:**

The dataset is loaded into a pandas DataFrame using the read\_csv() API, resulting in 7,752 rows and 25 columns. After loading, I used the info() method to get an overview of the dataset, checking data types, non-null counts, and any missing values.

A screenshot of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

**Conversion of Data Types for Analysis**

In the initial stages of data analysis, it's crucial to ensure that each column in a dataset is appropriately typed, as this impacts the accuracy and efficiency of any operations or transformations performed on the data. In our dataset, all columns were of the float data type, except for the **Date** column, which was stored as an object data type. This is typical when dealing with dates that are not parsed into a formal date-time format.

To correct this, the **Date** column was converted into a more appropriate datetime format using the pandas library's to\_datetime() API. This conversion enables us to manipulate and extract temporal features from the date, which could be useful later in time-based analyses such as trends or seasonality.

**Exploratory Data Analysis**

Now we are checking how many exact numbers of null present in each column using df.isnull().sum()

A screenshot of a computer code

Description automatically generated

As the above image clearly illustrates, several columns contain null values that we need to address for a more comprehensive analysis. Filling these null values is essential to ensure the accuracy and reliability of our findings. However, before proceeding with any data imputation or cleaning processes, I’m taking a step back to explore the unique values and their respective counts in each column. This exploration is crucial for understanding the distribution of data and identifying any potential issues. To achieve this, I am utilizing the df.nunique() function, which provides a count of unique entries in each column of the DataFrame.A screenshot of a computer

Description automatically generated

Currently, I am in the process of filling the null values in our dataset using the median method. This approach is particularly effective because the median is less sensitive to outliers compared to the mean, making it a robust choice for imputation. Once all the null values are successfully filled, we will take a closer look at how our data is distributed.

A screenshot of a computer program

Description automatically generated

Understanding the distribution of our data is crucial for identifying patterns, trends, and potential anomalies that could influence our analysis. To achieve this, I will be plotting distribution plots for each column using the Matplotlib library. These visualizations will allow us to gain insights into the data's spread, central tendency, and overall characteristics, helping us make informed decisions as we proceed with our analysis, as illustrated below.

A screenshot of a graph

Description automatically generated

From above image I can see there is skewness

Now we have also need to check for outliers for that I am using boxplot as shown belowA screenshot of a graph

Description automatically generated

We can see there is outliers also present in some columns

**EDA Concluding Remarks**

From the data analysis conducted above, I can conclude that there is some skewness present in the distribution of the data. This skewness indicates that the data may not be symmetrically distributed, which can affect various statistical analyses and models. Additionally, I have identified the presence of outliers in some columns, which could potentially distort our results and lead to misleading conclusions.

To ensure the integrity and accuracy of our analysis, we will address these issues during the preprocessing stage of our data pipeline. This may involve techniques such as transforming skewed distributions to achieve normality and employing methods to handle outliers, such as capping or removing them. By addressing these challenges in the preprocessing phase, we can create a cleaner, more reliable dataset that enhances the quality of our subsequent analysis.

**Pre-processing Pipeline**

One of the key issues we need to address is the problem of skewed data distribution, as skewness can significantly impact the performance of machine learning algorithms. To rectify this, I am implementing a method to remove skewness from the data.

Specifically, I am utilizing the PowerTransformer functionality from the sklearn library. This transformer applies a power transformation to make the data more Gaussian-like, which can improve the performance of various machine learning models. The PowerTransformer can effectively handle both positive and negative skewness, helping to stabilize variance and make the data more amenable to analysis. Below is the implementation of this approach:

I am using powertransform functionality of sklearn as shown below:

A screenshot of a computer

Description automatically generated

By applying this transformation, we can enhance the quality of our dataset, paving the way for more accurate and reliable model training.

Top of Form

Bottom of Form

Now, I will proceed to remove outliers from the dataset to improve the accuracy of the machine learning model. For this purpose, I am using the Z-score functionality from the scipy library. A Z-score, also known as a standard score, is a statistical measure that indicates how many standard deviations a data point is from the mean of the dataset. This method allows us to identify outliers by measuring the distance of each data point from the mean relative to the standard deviation.

To begin, I will calculate the Z-scores for all data points in the relevant columns. Typically, values with Z-scores beyond a certain threshold—often set at ±3.0 or ±2.5—are considered outliers. Here’s how I will implement this process:

I am using threshold 3 as shown belowA screenshot of a computer

Description automatically generated

Now I am removing outliers as shown belowA screenshot of a computer code

Description automatically generated

Here we need to make 2 predictions one is . Next\_Tmax and other is Next\_Tmin now at first I am making prediction for Next\_Tmax for that I have to divide my data frame in two parts feature variable x and target variable y as shown below code

#assigning feature and target data

x = df.drop('Next\_Tmax',axis=1)

y = df['Next\_Tmax']

print("feature dimension=",x.shape)

print('label Dimension',y.shape)

After that we need to standardize data for we are using Standardscaler from sklearn library as shown in below image

A screenshot of a computer

Description automatically generated

**Building Machine Learning Models**

Now our target variable is continuous variable so I am going to build regression model for data prediction for that we need to split our training data in training data and test data

So I am splitting in like 25 percent test data and 75 percent training data using train\_test\_split APT

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.25,random\_state = 123)

here we can set any random state according so I am first training this data on LinearRegression and trying multiple number for random state and than which ever give maximum accuracy I am going to use for rest of the training model

so now I am my training my data on liniear regression model which describes the relationship between a dependent variable, y, and one or more independent variables, x.

now after training model I am also calculating:

R2\_Score- which shows how well the data fit the regression model (the goodness of fit).

mean absolute error- is a metric that is used to evaluate the performance of regression models. It's defined as the average of the absolute difference between actual and predicted values

Mean Squared Error- Mean squared error (MSE) measures the amount of error in statistical models. It assesses the average squared difference between the observed and predicted values. When a model has no error, the MSE equals zero. As model error increases, its value increases.

root mean squared error- the residuals' standard deviation, or the average difference between the projected and actual values produced by a statistical model

you can see all that in below image

A screenshot of a computer program

Description automatically generated

After that I used RandomForestRegressor model which is a supervised learning algorithm, meaning that the data on which it operates contains labels or outcomes. It works by creating many decision trees, each built on randomly chosen subsets of the data. The model then aggregates the outputs of all of these decision trees to make an overall prediction for unseen data points. In this way, it can process larger datasets and capture more complex associations than individual decision trees.

After that I used GradientBoostingRegressor model: Gradient Boosting,also called Gradient Boosting Machine (GBM) is a type of [supervised](https://datamapu.com/posts/ml_concepts/supervised_unsupervised/#supervised) Machine Learning algorithm that is based on [ensemble learning](https://datamapu.com/posts/ml_concepts/ensemble/). It consists of a sequential series of models, each one trying to improve the errors of the previous one. It can be used for both regression and classification tasks

After I also used KNN model, DecisionTreeRegressor and XGBRegressor technique and I also calculated R2\_Score, mean absolute error, Mean Squared Error and root mean squared error for all model

After comparing all the value from above algoritham I have found RandomForestRegressor and XGBRegressor giving good accuracy and less error

Then after I did cross validation of values using cross\_val\_score functionality of sklearn library

After calculating difference of R2\_Score and cross\_val\_score I have found RandomForestRegressoris best because it has less difference in cross validation

After that we are building our model with best parameters for selecton of best parameter for training we are using GridSearchCV functionality as shown in below image

A screenshot of a computer

Description automatically generated

After creating model we are saving the model using pickel library

A close up of a computer code

Description automatically generated

We can use it again using load method from pickel anf test our data in future also

Now we have followed same process to pridict  Next\_Tmin only difference we did is we make  Next\_Tmin target variable and other variable as feature variable including Next\_Tmax

And afte following same process I have found random forest regrassor is best for this data to pridict so we ctreated model using Random forest regressor and saved it

**Concluding Remarks**

In conclusion, after experimenting with several regression models, the **RandomForestRegressor** was identified as the most effective algorithm for predicting both **Next\_Tmax** and **Next\_Tmin**. With over 85% accuracy, this model shows great potential for improving weather forecasts, specifically for maximum and minimum temperatures. The project demonstrates the power of machine learning in improving the accuracy of weather predictions and opens avenues for further improvements in meteorological forecasting systems.