Temperature Forecast: Korea Meteorological Administration

1. Problem Definition.

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].
station - used weather station number: 1 to 25
Date - Present day: yyyy-mm-dd ('2013-06-30' to '2017-08-30')
Present_Tmax - Maximum air temperature between 0 and 21 h on the present day (Â°C): 20 to 37.6
Present_Tmin - Minimum air temperature between 0 and 21 h on the present day (Â°C): 11.3 to 29.9
LDAPS_RHmin - LDAPS model forecast of next-day minimum relative humidity (%): 19.8 to 98.5
LDAPS_RHmax - LDAPS model forecast of next-day maximum relative humidity (%): 58.9 to 100
LDAPS_Tmax_lapse - LDAPS model forecast of next-day maximum air temperature applied lapse rate (Â°C): 17.6 to 38.5
LDAPS_Tmin_lapse - LDAPS model forecast of next-day minimum air temperature applied lapse rate (Â°C): 14.3 to 29.6
LDAPS_WS - LDAPS model forecast of next-day average wind speed (m/s): 2.9 to 21.9
LDAPS_LH - LDAPS model forecast of next-day average latent heat flux (W/m2): -13.6 to 213.4
LDAPS_CC1 - LDAPS model forecast of next-day 1st 6-hour split average cloud cover (0-5 h) (%): 0 to 0.97
LDAPS_CC2 - LDAPS model forecast of next-day 2nd 6-hour split average cloud cover (6-11 h) (%): 0 to 0.97
LDAPS_CC3 - LDAPS model forecast of next-day 3rd 6-hour split average cloud cover (12-17 h) (%): 0 to 0.98
LDAPS_CC4 - LDAPS model forecast of next-day 4th 6-hour split average cloud cover (18-23 h) (%): 0 to 0.97
LDAPS_PPT1 - LDAPS model forecast of next-day 1st 6-hour split average precipitation (0-5 h) (%): 0 to 23.7
LDAPS_PPT2 - LDAPS model forecast of next-day 2nd 6-hour split average precipitation (6-11 h) (%): 0 to 21.6
LDAPS_PPT3 - LDAPS model forecast of next-day 3rd 6-hour split average precipitation (12-17 h) (%): 0 to 15.8

LDAPS_PPT4 - LDAPS model forecast of next-day 4th 6-hour split average precipitation (18-23 h) (%): 0 to 16.7
lat - Latitude (Â°): 37.456 to 37.645
lon - Longitude (Â°): 126.826 to 127.135
DEM - Elevation (m): 12.4 to 212.3
Slope - Slope (Â°): 0.1 to 5.2
Solar radiation - Daily incoming solar radiation (wh/m2): 4329.5 to 5992.9
Next_Tmax - The next-day maximum air temperature (Â°C): 17.4 to 38.9
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

2. Data Analysis.

The dataset consists of 7752 records and 25 features including a target variable “Price”.

Since we have 27 missing values in target columns "Next_Tmax" and "Next_Tmin" out of a total 7750 records, we will drop the entire row correspondent to these column missing values.

```
df.dropna(axis=0, how='any', subset=['Next_Tmax', 'Next_Tmin'], inplace=True)
```

Now we have a total of 7725 records as 27 rows with missing target values are deleted.

We have multiple missing values and since all these features are numerical, missing value are filled with corresponding Mean values.

```
df["Present_Tmax"] = df["Present_Tmax"].fillna(df["Present_Tmax"].mean())
df["Present_Tmin"] = df["Present_Tmin"].fillna(df["Present_Tmin"].mean())
df["LDAPS_RHmin"] = df["LDAPS_RHmin"].fillna(df["LDAPS_RHmin"].mean())
df["LDAPS_RHmax"] = df["LDAPS_RHmax"].fillna(df["LDAPS_RHmax"].mean())
df["LDAPS_Tmax_lapse"] = df["LDAPS_Tmax_lapse"].fillna(df["LDAPS_Tmax_lapse"].mean())
df["LDAPS_Tmin_lapse"] = df["LDAPS_Tmin_lapse"].fillna(df["LDAPS_Tmin_lapse"].mean())
df["LDAPS_WS"] = df["LDAPS_WS"].fillna(df["LDAPS_WS"].mean())
df["LDAPS_LH"] = df["LDAPS_LH"].fillna(df["LDAPS_LH"].mean())
df["LDAPS_CC1"] = df["LDAPS_CC1"].fillna(df["LDAPS_CC1"].mean())
df["LDAPS_CC2"] = df["LDAPS_CC2"].fillna(df["LDAPS_CC2"].mean())
df["LDAPS_CC3"] = df["LDAPS_CC3"].fillna(df["LDAPS_CC3"].mean())
df["LDAPS_CC4"] = df["LDAPS_CC4"].fillna(df["LDAPS_CC4"].mean())
df["LDAPS_PPT1"] = df["LDAPS_PPT1"].fillna(df["LDAPS_PPT1"].mean())
df["LDAPS_PPT2"] = df["LDAPS_PPT2"].fillna(df["LDAPS_PPT2"].mean())
df["LDAPS_PPT3"] = df["LDAPS_PPT3"].fillna(df["LDAPS_PPT3"].mean())
df["LDAPS_PPT4"] = df["LDAPS_PPT4"].fillna(df["LDAPS_PPT4"].mean())
```

Apart from the above, I have deliberately avoided features "station" and "Date" as "station" is an entity out of 25 such locations and filling the missing value may have altered the results. "Date" is given on a daily basis and temperature is consistent, hence filling date value would alter the results.

```
|: df.drop(columns=["station","Date"], inplace=True)
```

I have split the date column into “Month” and “Year” for better analysis.

```
: df['Year'] = pd.DatetimeIndex(df['Date']).year
  df['Month'] = pd.DatetimeIndex(df['Date']).month
```

Also, the dataset has some outliers or skewness as the distribution is not normal.

I have considered columns 'Present_Tmax', 'LDAPS_RHmin', 'LDAPS_RHmax', 'LDAPS_Tmin_lapse', 'LDAPS_WS', 'LDAPS_LH', 'LDAPS_CC1', 'LDAPS_CC2', 'LDAPS_CC3', 'LDAPS_CC4', 'LDAPS_PPT1', 'LDAPS_PPT2', 'LDAPS_PPT3', 'LDAPS_PPT4', 'DEM', 'Slope' and 'Solar radiation' as these have more skewness. Hence I used Z-Score to remove outliers if any as follows:

```
In [34]: from scipy.stats import zscore

         z_score = zscore(df[['Present_Tmax', 'LDAPS_RHmin', 'LDAPS_RHmax',
                              'LDAPS_Tmin_lapse', 'LDAPS_WS', 'LDAPS_LH',
                              'LDAPS_CC1', 'LDAPS_CC2', 'LDAPS_CC3', 'LDAPS_CC4', 'LDAPS_PPT1',
                              'LDAPS_PPT2', 'LDAPS_PPT3', 'LDAPS_PPT4', 'DEM', 'Slope',
                              'Solar radiation']])

         abs_zscore = np.abs(z_score)

         filtering_entry = (abs_zscore < 3).all(axis=1)

         df = df[filtering_entry]
```

In this case we are losing about 10.62 % of data after applying z-score and by dropping these data I have received better accuracy. Ideally, it's not a good choice to drop such huge amount of data but in this case by not dropping the outliers, the model did not give better results, hence these outlier data was removed completely

```
In [35]: # Percentage data loss:

         loss_percent = (7725-6904)/7725*100
         print(loss_percent)

         10.627831715210355
```

Since we have 2 target variables, both are dropped from x and are stored in y as follows:

```
In [38]: x = df.drop(columns = ["Next_Tmax", "Next_Tmin"], axis=1)
         y = df[["Next_Tmax", "Next_Tmin"]]
```

Since we have skewness in all the features, I will consider applying one of the skewness reduction techniques. But before that StandardScaler is used for further scaling up of data

```
In [42]: from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()
         x_scaled = scaler.fit_transform(x)
         x_scaled
```

```
Out[42]: array([[-0.42579235, -0.79484082,  0.19771653, ...,  1.54744573,
                 -1.42636481, -0.48612358],
                [ 0.68961297, -0.71012279, -0.23552485, ...,  1.25486307,
                 -1.42636481, -0.48612358],
                [ 0.58504372,  0.0099805 , -0.49384609, ...,  1.24123382,
                 -1.42636481, -0.48612358],
                ...,
                [-2.30803884, -2.61627855, -2.08076657, ..., -2.12117383,
                  1.43278381,  0.40588896],
                [-2.30803884, -2.36212445, -2.27602489, ..., -2.13287025,
                  1.43278381,  0.40588896],
                [-2.34289526, -2.4892015 , -2.35610561, ..., -2.10215993,
                  1.43278381,  0.40588896]])
```

The following features are considered for skewness removal and are stored in a varaibale called “feat_Skew”

'Present_Tmax', 'Present_Tmin', 'LDAPS_RHmin', 'LDAPS_RHmax', 'LDAPS_Tmax_lapse', 'LDAPS_Tmin_lapse', 'LDAPS_WS', 'LDAPS_LH', 'LDAPS_CC1', 'LDAPS_CC2', 'LDAPS_CC3', 'LDAPS_CC4', 'LDAPS_PPT1', 'LDAPS_PPT2', 'LDAPS_PPT3', 'LDAPS_PPT4', 'DEM', 'Slope' and 'Solar radiation'

I have not considered “Month” and “Year” as it makes no sense in trying to remove skewness from this. I have also avoided using “lat” and “lon” as these are latitude and longitudes and they are specific coordinates.

```
In [41]: feat_skew = ['Present_Tmax', 'Present_Tmin', 'LDAPS_RHmin', 'LDAPS_RHmax',
                      'LDAPS_Tmax_lapse', 'LDAPS_Tmin_lapse', 'LDAPS_WS', 'LDAPS_LH',
                      'LDAPS_CC1', 'LDAPS_CC2', 'LDAPS_CC3', 'LDAPS_CC4', 'LDAPS_PPT1',
                      'LDAPS_PPT2', 'LDAPS_PPT3', 'LDAPS_PPT4', 'DEM', 'Slope',
                      'Solar radiation']
```

To reduce the skewness to some extent, I have used a Power Transformer technique. Since we have both positive and negative values in skewness, I have used "Yeo-Johnson" technique. The data is now further standardized and skewness is further reduced which further resulted in the normal distribution of the dataset to some extent.

```
]: from sklearn.preprocessing import PowerTransformer

   scaler = PowerTransformer(method="yeo-johnson")

   x[feat_skew] = scaler.fit_transform(x[feat_skew].values)

   x[feat_skew]
```

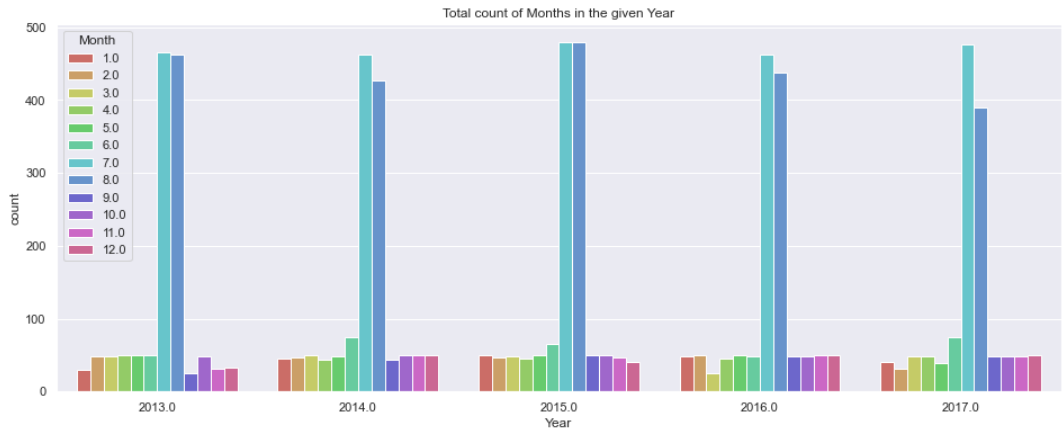
Since there was a possibility of multicollinearity, I used Variation Inflation Factor to identify such features and dropped " LDAPS_Tmin_lapse" and "LDAPS_Tmax_lapse" as it had high collinearity and its correlation with target features was very less compared to other features.

```
In [46]: from statsmodels.stats.outliers_influence import variance_inflation_factor

         vif = pd.DataFrame()
         vif["vif"] = [variance_inflation_factor( x_scaled, i) for i in range(x_scaled.shape[1])]
         vif["Features"] = x.columns
         vif
```

3. EDA Concluding Remark.

From the below plot we can observe that the dataset consists of information from the years 2013 to 2017 and we have "July" and "August" months data more than the other months recorded data. Hence this dataset may not be that effective as temperature control related issues are highly dynamic and having an updated and live data is very much important.



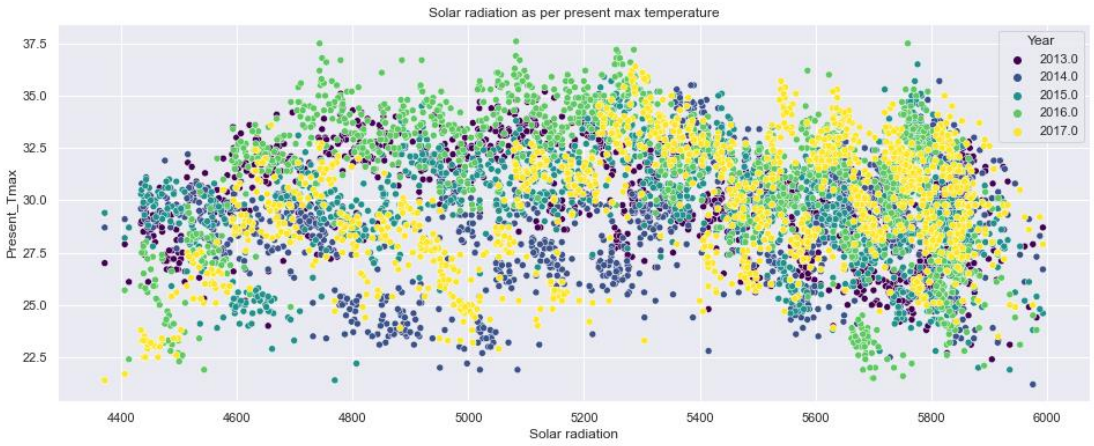
Let's observe if there is any relationship between the present-day temperature and solar radiation.

The central concept of LDAP is the information model, which deals with the kind of information stored in directories and the structuring of information. The information model revolves around an entry, which is a collection of attributes with type and value.

LDAPS (Lightweight Directory Access Protocol) is probably under Government jurisdiction of South Korea and we can see from the below graph Solar radiation plays an important role in creating fluctuations in present temperature.

From the plot we can Also see that there is rise in solar radiation and is highest for the year 2017 followed by 2016

We can see temperatures have usually been around 25 C to 36 C when solar radiation is high and temperatures are low between 17 C to 25 C approximately



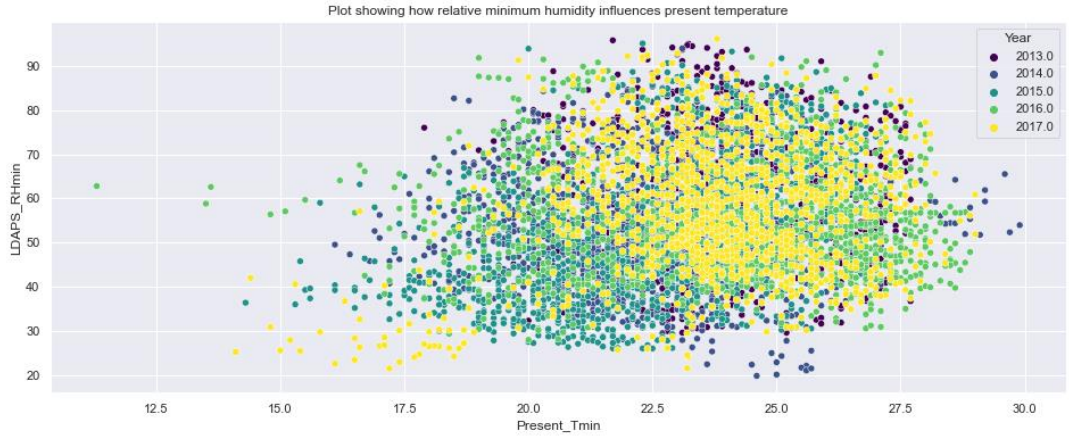
Let's observe if there is any relationship between the approximate / relative minimum humidity and present temperature.

We can see that when the temperature is to its minimum level, humidity is not necessarily minimum and is evenly scattered.

But we can also see that after 17.5 C and up to 28 C humidity has been somewhat consistent.

But the dataset appears to record more details for the year 2017. We can say that power plants dealing with natural resources as such need a lot of sophisticated technology to record every single data point and it could either be a situation when South Korea had major geological changes in 2017 which seemed to have increased gradually over the period of years steadily.

When the temperature was high the humidity was also very high especially in the year 2015 and 2017.

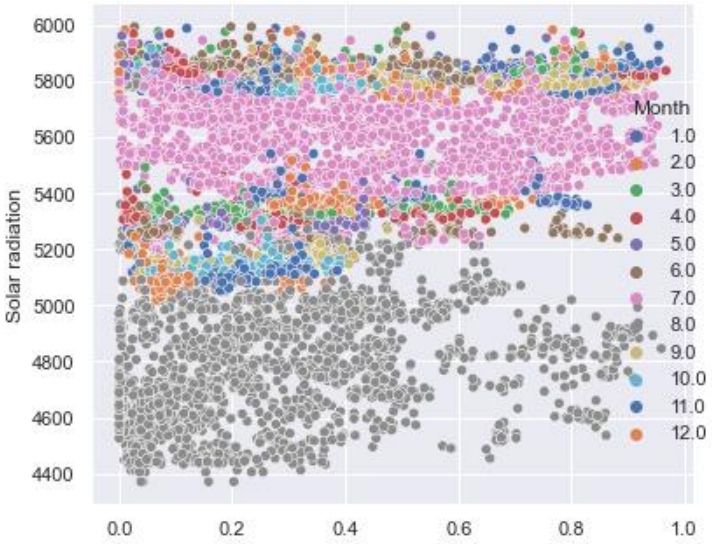


From the below 4 replots, we can see that Solar radiation is high consistently in the month of July and then in the month of August

We know at present; South Korea is having a semi-Rainy and Summer season, especially in the areas / cities shown in the previous plots

The average cloud cover split seems to be same across all the 6-hour shift and the highest solar radiation is recorded at about 6000 but the number of months is very less in this range. This could mean it's not constant and weather doesn't seem to help much in this case.

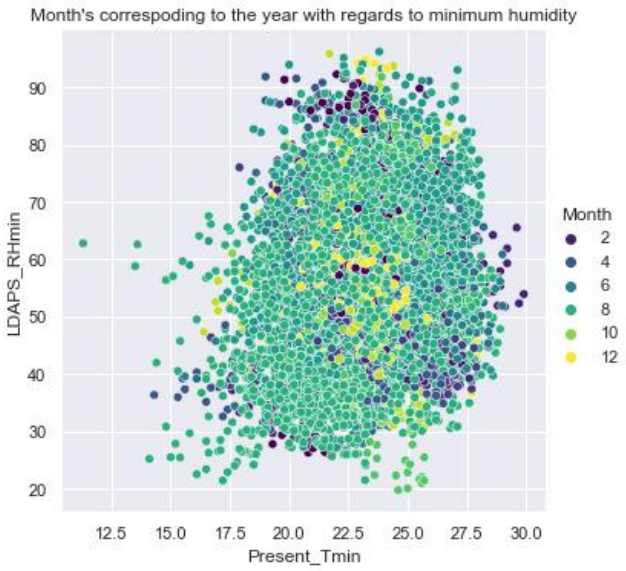
Since it's a Meteorogical department, there could be seismic or digging activities contributing to this radiation.



From the below plot we can see that "June" and "August" seem to have both relative minimum and maximum humidity.

If you observe both the graphs below there is no much difference when it comes to humidity and it seems consistent across the months and I have used "relplot" for the same

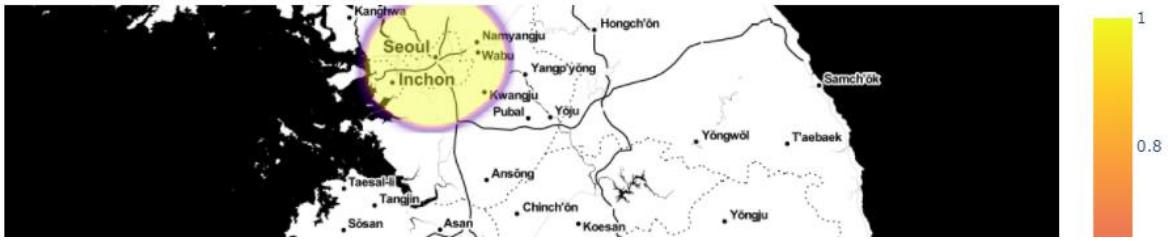
Humidity is high as "July" and "August" is summer season in South Korea and also coastal areas are generally humid and this could be the reasons for increase or decrease in temperature and humidity.



Let's observe the geographic location of South Korea. We know any organization, plants or a funded group is setup near the availability of resources rather than in some random remote locations.

We can see that such Temperature plants are located near to Sea / Coastal side areas with proper transport connectivity. It appears cities like Inchon, Seoul, Wabu, Yangpyong and Pubai have such geographic conditions for which these plants are suitable to be setup.

I have used Plotly's "density_mapbox" graph by making use of available latitudes and longitudes, by coinciding with the country's geo location itself,



4. Pre-Processing Pipeline and Building Machine Learning Models.

Since it's a Multioutput Regression problem I have used a total of 4 compatible Algorithms i.e. DecisionTreeRegressor, RandomForestRegressor, KNeighborsRegressor, and ExtraTreeRegressor to build this model that are acceptable by multioutput regressor and select the best suitable model by applying hyperparameter tuning.

In this case, **RandomForestRegressor** gave better results with an accuracy of approximately 98.24% as shown below.

Also, the difference between the adjusted R2 score and cross-validation score is very less for **RandomForestRegressor** compared to other algorithms.

In [51]:
```
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor()
rf_reg.fit(x_train,y_train)

y_pred = rf_reg.predict(x_test)

print("Adjusted R2 squared : ",rf_reg.score(x_train,y_train))
print("Mean Absolute Error (MAE): ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE): ",mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE): ",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Adjusted R2 squared :  0.982943367107951
Mean Absolute Error (MAE):  0.6712579633204632
Mean Squared Error (MSE):  0.8812780726351345
Root Mean Squared Error (RMSE):  0.9387641198060003
```

```
In [54]: from sklearn.model_selection import cross_val_score
```

```
In [56]: scr = cross_val_score(rf_reg, x, y, cv=5)
         print("Cross Validation score of RandomForestRegressor model is:", scr.mean())

         Cross Validation score of RandomForestRegressor model is: 0.5380642848838927
```

I then used approximately 4 parameters i.e criterion, max_features, obb_score and random_state for Hyperparameter tuning to check if the score could be improved to some extent if possible.

```
In [65]: parameters = {"criterion":["mse", "mae"],
                      "max_features":["auto", "sqrt", "log2"],
                      "oob_score":[True, False],
                      "random_state":[30, 50, 70, 100, 120]
                      }
```

The predictions are then re-run using RandomizedSearchCV to find the best possible parameters by passing them to “best_params_” as follows:

```
In [67]: from sklearn.model_selection import RandomizedSearchCV
         RCV = RandomizedSearchCV(RandomForestRegressor(), parameters, cv=5, n_iter=10)
```

```
In [68]: RCV.fit(x_train, y_train)
```

```
Out[68]: RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(),
                            param_distributions={'criterion': ['mse', 'mae'],
                                                 'max_features': ['auto', 'sqrt',
                                                                  'log2'],
                                                 'oob_score': [True, False],
                                                 'random_state': [30, 50, 70, 100, 120]})
```

```
In [69]: RCV.best_params_
```

```
Out[69]: {'random_state': 30,
          'oob_score': False,
          'max_features': 'sqrt',
          'criterion': 'mse'}
```

Rebuild the model using the appropriate params we received from best_params_ by applying it to MultiOutputRegressor. It’s observed that the model accuracy is still the same as before and there have been no further improvements post hyper parameter tuning.

```
In [70]: mod_final = MultiOutputRegressor(RandomForestRegressor(random_state = 30, oob_score =True, max_features= "sqrt", criterion= "mse

         mod_final.fit(x_train,y_train)
         pred = mod_final.predict(x_test)
```

```
In [71]: mod_final.score(x_train,y_train)
```

```
Out[71]: 0.9832558746144471
```

6. Concluding Remarks.

- There have been multiple and diverse investments and efforts made to achieve shared growth by responding to the global crisis of the climate. The (KMA) Korea Meteorological Administration has implemented various cooperation projects and provide training programs and quality education for international development to support its neighboring and partner countries exposed to disaster hazards, where Asian countries are highly considered.
- The weather of the Korean peninsula is much like that of the east coast of the continent. The winters on this area are extraordinarily bloodless and dry, while the summers are extraordinarily warm and humid. Temperatures can attain 40°C or better at some stage in the year.
- Temperatures in Seoul frequently exceed 30° (86°F) for the duration of the day and infrequently fall beneath 25° (77°F) at night. In July and August, temps can attain 36° (97°F) for numerous days at a time, ensuing in maximum of the insufferable warmness remains and the wind stays calm.
- In August, the Korean peninsula within reason humid – possibly damper than California or South Europe, however now no longer through much. This shows that humid climate with temperatures above 35°C can power people insane.
- Furthermore, the months with the very best temperatures above 30°C are from the center of June to past due June, and from the center of July to the center of September. The wet season falls among those times, it reaches temperatures of over 35°C from past due July to the center to past due August.
- As a part of the effort, the KMA might be keeping Global Forum on International Development Cooperation for Climate Change Response in Asia, to percentage improvement cooperation sports withinside the discipline of climate and climate, in addition to discover approaches to create synergy results thru cooperation and collaboration, via way of means of inviting home and distant places professionals from worldwide improvement cooperation agencies.