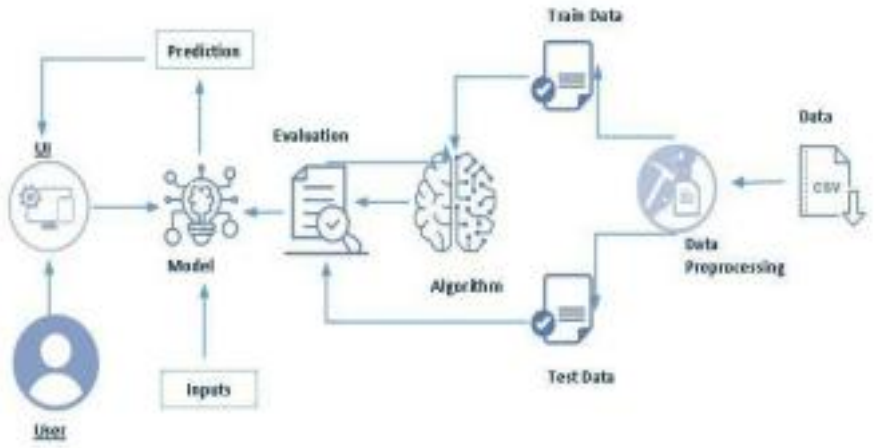
Smart Home – Temperature Prediction

Project Description:

A smart home's devices are connected with each other and can be accessed through one central point like a smartphone, tablet, laptop, or game console. Door locks, televisions, thermostats, home monitors, cameras, lights, and even appliances such as the refrigerator can be controlled through one home automation system. Smart Wi-Fi thermostats have moved well beyond the function they were originally designed for controlling heating and cooling comfort in buildings. They are now also learning from occupant behaviors and permit occupants to control their comfort remotely. Thermal comfort in buildings has been managed for many years by thermostats. At a most basic level, a thermostat allows a resident to set a desired indoor temperature, a means to sense actual temperature within the thermostat housing, and a means to signal the heating and/or cooling devices to turn on or off in order to affect control of the heating, ventilating, and air conditioning (HVAC) system in order to equilibrate the room temperature to the set point temperature. Thermostats use sensors such as thermistors or thermal diodes to measure temperature, they also often include humidity sensors for measuring humidity and microprocessor-based circuitry to control the HVAC system and operate based upon user-defined set point schedules. This project seeks to go beyond this state of the art by utilizing smart Wi-Fi thermostat data in residences to develop dynamic predictive models for room temperature and cooling/heating demand. .While efforts are being made around the world to minimize greenhouse gas emissions and make progress towards a more sustainable society, global energy demand continues to rise. Building energy consumption accounts for 20–40% of the total global energy consumption and Heating, Ventilation, Air Conditioning (HVAC) answer for around 50% of this amount. Therefore, implementing energy efficiency-related strategies and optimization techniques in buildings is a critical step in reducing global energy consumption.

In this project we will just take the data that is generated by the sensors by The University of CEU Cardenal Herrera (CEU-UCH)-Spain. We will preprocess the data and pass it to the Regression algorithms such as Linear Regression, Random forest, LightGDBM, and Xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

Technical Architecture:



Pre requisites:

To complete this project, you must required following software's, concepts and packages

- **Anaconda navigator and pharm:**
  - **o** Refer the link below to download anaconda navigator
  - **o** Link : https://youtu.be/1ra4zH2G4o0
- **Python packages:**
  - o Open anaconda prompt as administrator
  - o Type "pip install numpy" and click enter.
  - o Type "pip install pandas" and click enter.
  - o Type "pip install scikit-learn" and click enter.
  - o Type"pip install matplotlib" and click enter.
  - o Type"pip install scipy" and click enter.
  - o Type"pip install pickle-mixin" and click enter.
  - o Type"pip install seaborn" and click enter.
  - o Type "pip install Flask" and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - o Supervised learning: https://www.javatpoint.com/supervised-machine-learning o Unsupervised learning:
    https://www.javatpoint.com/unsupervised-machine-learning
  - o Regression and classification
  - o Linear Regression:
    https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know about-linear-regression/
  - o Random forest:

https://www.javatpoint.com/machine-learning-random-forest-algorithm

- o Xgboost:

  https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

- o Light GDBM:

  https://www.analyticsvidhya.com/blog/2021/08/complete-guide-on-how-to-use-lightgbm-in-python/

- o Evaluation metrics:

  https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

At the conclusion of this assignment, you will have gained a wide understanding of data and be familiar with the basic principles and techniques used in machine learning.

- Be familiar with outlier transformation methods, data pre-processing, and basic visualization principles.

Project Flow:

- The user input is entered via interacting with the UI.
- The integrated model analyzes the entered data.
- The prediction appears on the user interface once the model has

analyzed the input.

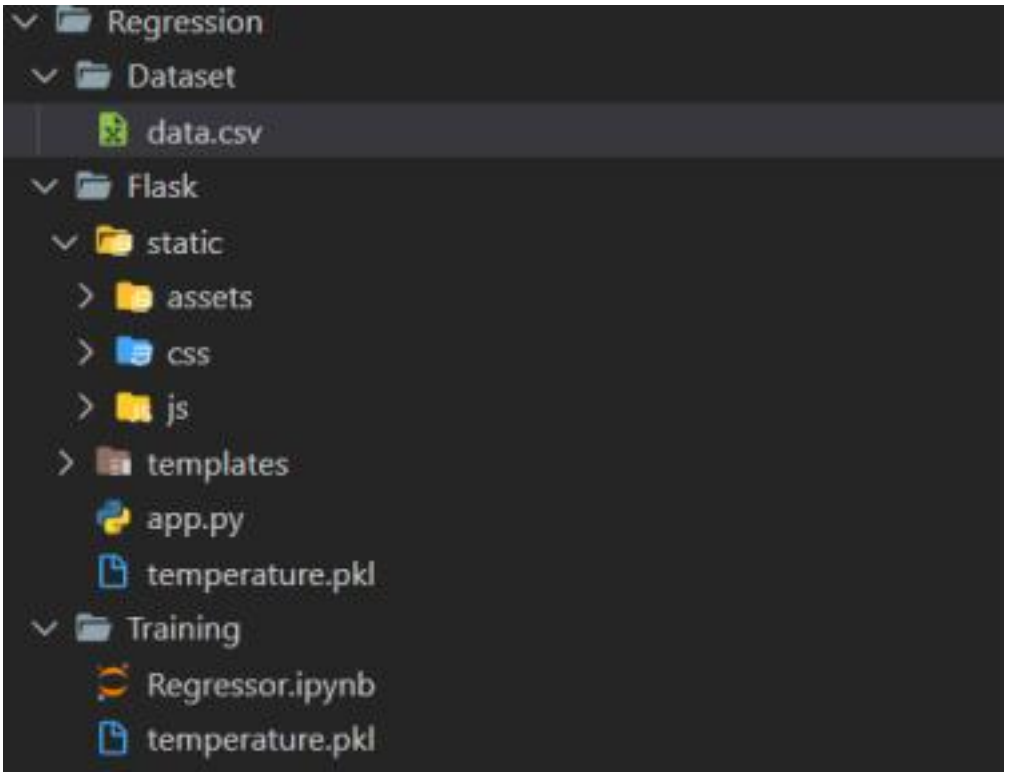
In order to do this, we must finish all of the tasks mentioned below.

- Data collection
  - o Collect the dataset or create the dataset
- Visualizing and analyzing data
  - o Univariate analysis
  - o Bivariate analysis
  - o Multivariate analysis
  - o Descriptive analysis
- Data pre-processing
  - o Checking for null values
  - o Handling outlier
  - o Handling categorical data
  - o Splitting data into train and test
- Model building
  - o Import the model building libraries
  - o Initializing the model
  - o Training and testing the model
  - o Evaluating performance of model
  - o Save the model
- Application Building

o Create an HTML file
o Build python code

Project Structure:

Make the Project folder with the files listed below in it.



- HTML files kept in the templates folder and a Python script app.py for scripting are required for our Flask application.
- Temperature.pkl is the model we have stored. We'll utilize this approach going forward for flask integration.
- Model training files are located in the Training folder, and IBM deployment files are located in the Training_ibm folder.

Milestone 1: Data Collection

ML depends heavily on data, it is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository,colab etc.

Link:

https://www.kaggle.com/competitions/smart-homes-temperature-time-series-forecasting/data

Milestone 2: Visualizing and analyzing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image.

```
[ ]  #importing libraries
     import pandas as pd
     import numpy as np
     import sklearn
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.ensemble import RandomForestRegressor
     import xgboost as xgb
     import lightgbm as lgb
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv () to read the dataset. As a parameter we have to give the directory of csv file.

```
[ ]  #Load the dataset
     df = pd.read_csv("/content/train.csv")
```

```
[ ]  #displaying first 5 rows
     df.head()
```

	Id	Date	Time	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Ligh
0	0	13/03/2012	11:45	216.560	221.920	39.9125	42.4150	81.6650	
1	1	13/03/2012	12:00	219.947	220.363	39.9267	42.2453	81.7413	
2	2	13/03/2012	12:15	219.403	218.933	39.7720	42.2267	81.4240	
3	3	13/03/2012	12:30	218.613	217.045	39.7760	42.0987	81.5013	
4	4	13/03/2012	12:45	217.714	216.080	39.7757	42.0686	81.4657	

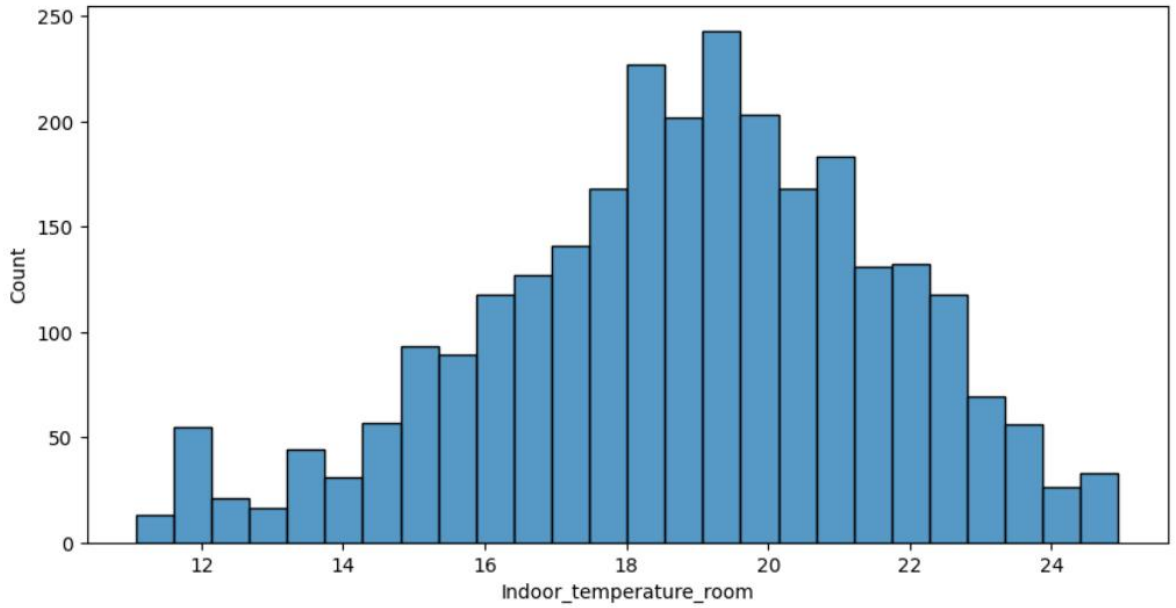
Activity 3: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

- Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
#UNIVARIATE ANALYSIS
plt.figure(figsize=(10,5))
sns.histplot(data=df,x='Indoor_temperature_room',)
```

```
<Axes: xlabel='Indoor_temperature_room', ylabel='Count'>
```



- From the plot we came to know, Indoor_temperature_room column, which is our output column it follows normal distribution.

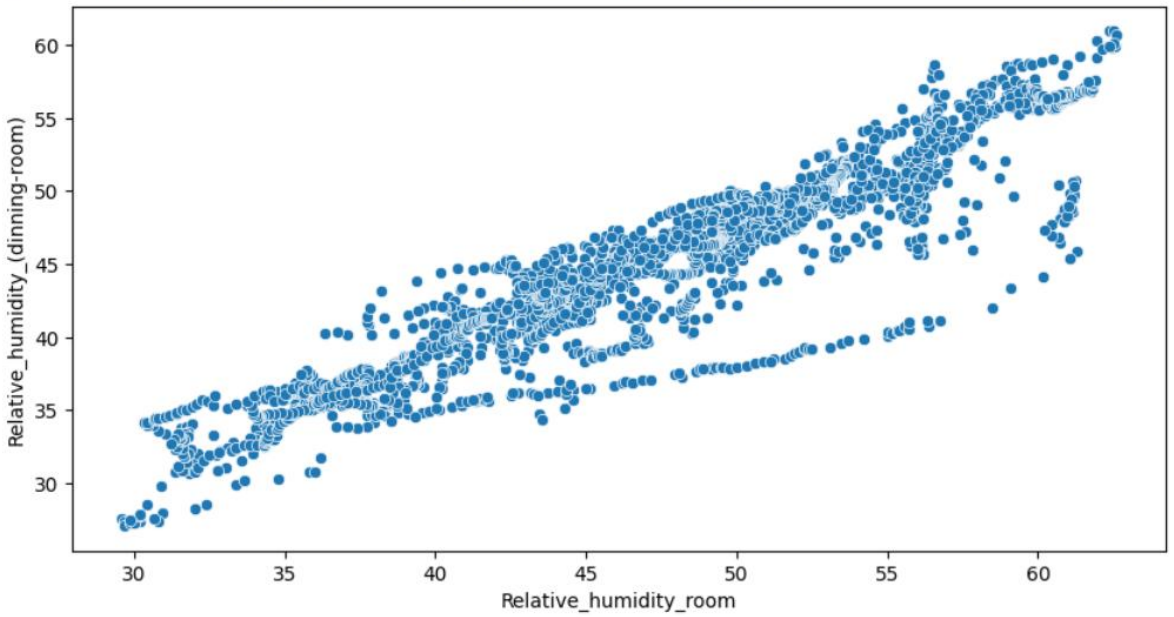
Activity 4: Bivariate analysis

Scatter Plot():

Scatter plots are the graphs that present the relationship between two variables in a data-set. It represents data points on a two-dimensional plane or on a Cartesian system. The independent variable or attribute is plotted on the X-axis, while the dependent variable is plotted on the Y-axis.

```
[ ]  #BIVARIATE ANALYSIS
     plt.figure(figsize=(10,5))
     sns.scatterplot(data=df,x='Relative_humidity_room',y='Relative_humidity_(dinning-room)')
```

```
<Axes: xlabel='Relative_humidity_room', ylabel='Relative_humidity_(dinning-room)'>
```

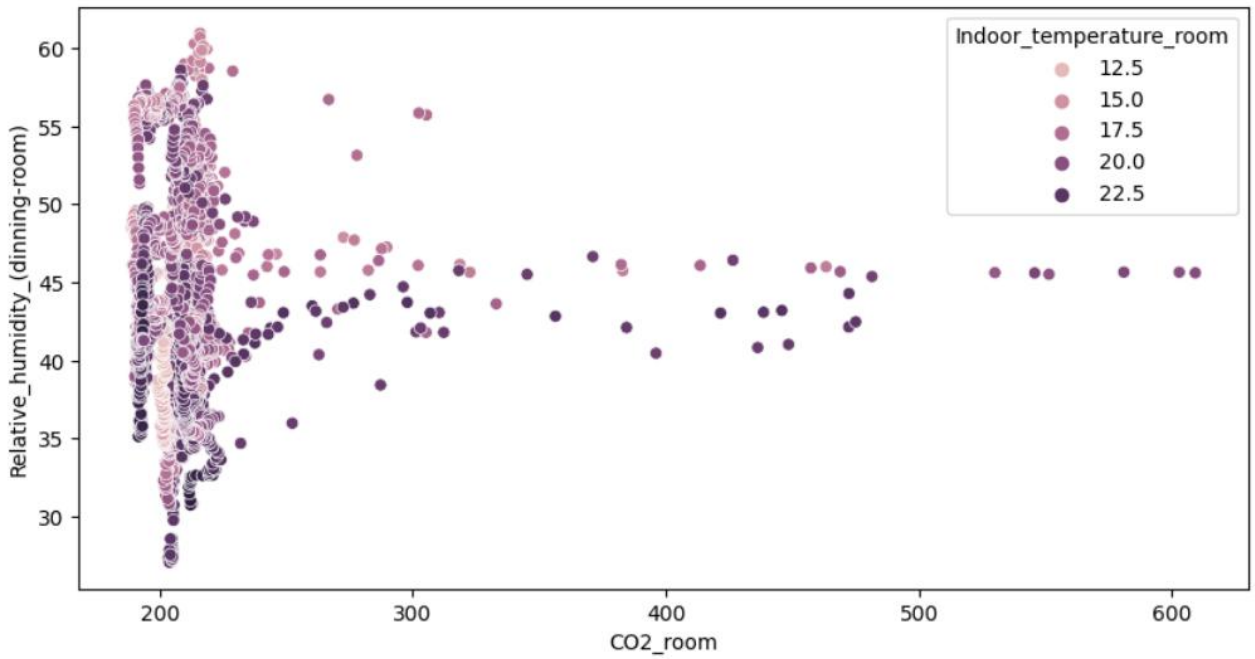


Activity 5: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used swarm plot from Seaborn package.

```
[ ]  #MULTIVARIATE ANALYSIS
     plt.figure(figsize=(10,5))
     sns.scatterplot(data = df,x='CO2_room',y='Relative_humidity_(dinning-room)',hue='Indoor_temperature_room')
```

```
<Axes: xlabel='CO2_room', ylabel='Relative_humidity_(dinning-room)'>
```



Activity 6: Descriptive analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

	Id	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Mete
count	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	
mean	1381.500000	208.479123	211.065844	44.878420	47.321220	26.745381	40.732571	0.047033	
std	798.042397	27.032686	28.469144	6.587440	7.557795	23.298441	42.326087	0.206705	
min	0.000000	187.339000	188.907000	27.084000	29.594700	10.740000	11.328000	0.000000	
25%	690.750000	200.893250	202.682750	40.351975	42.531325	11.588700	13.265300	0.000000	
50%	1381.500000	207.045500	209.408000	45.434650	47.534700	11.801300	17.690000	0.000000	
75%	2072.250000	211.245500	213.218750	49.352675	52.685975	31.224000	52.057350	0.000000	
max	2763.000000	594.389000	609.237000	60.957300	62.594700	110.693000	162.965000	1.000000	

Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 1: Checking for null values

- Let's find the shape of our dataset first, to find the shape of our data, df.shape method is used. To find the data type, df.info() function is used.

```
[ ]  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Id                                  2764 non-null   int64
 1   Date                                2764 non-null   object
 2   Time                                2764 non-null   object
 3   CO2_(dinning-room)                  2764 non-null   float64
 4   CO2_room                            2764 non-null   float64
 5   Relative_humidity_(dinning-room)    2764 non-null   float64
 6   Relative_humidity_room              2764 non-null   float64
 7   Lighting_(dinning-room)             2764 non-null   float64
 8   Lighting_room                       2764 non-null   float64
 9   Meteo_Rain                          2764 non-null   float64
 10  Meteo_Sun_dusk                      2764 non-null   float64
 11  Meteo_Wind                          2764 non-null   float64
 12  Meteo_Sun_light_in_west_facade      2764 non-null   float64
 13  Meteo_Sun_light_in_east_facade      2764 non-null   float64
 14  Meteo_Sun_light_in_south_facade     2764 non-null   float64
 15  Meteo_Sun_irradiance                2764 non-null   float64
 16  Outdoor_relative_humidity_Sensor    2764 non-null   float64
 17  Day_of_the_week                     2764 non-null   float64
 18  Indoor_temperature_room             2764 non-null   float64
dtypes: float64(16), int64(1), object(2)
memory usage: 410.4+ KB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.

```
[ ]  df.isnull().sum()
```

```
Id                                  0
Date                                0
Time                                0
CO2_(dinning-room)                  0
CO2_room                            0
Relative_humidity_(dinning-room)    0
Relative_humidity_room              0
Lighting_(dinning-room)             0
Lighting_room                       0
Meteo_Rain                          0
Meteo_Sun_dusk                      0
Meteo_Wind                          0
Meteo_Sun_light_in_west_facade      0
Meteo_Sun_light_in_east_facade      0
Meteo_Sun_light_in_south_facade     0
Meteo_Sun_irradiance                0
Outdoor_relative_humidity_Sensor    0
Day_of_the_week                     0
Indoor_temperature_room             0
dtype: int64
```

From the above code of analysis, we can infer that columns Do not have any Null Values, so we don't perfrom numll values operations on this dataset.

Activity 2: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

- In our dataset, we don't have any categorical values and most of the values we have are float so, we don't perform any encoding techniques.

```
[ ]  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Id                                  2764 non-null   int64
 1   Date                                2764 non-null   object
 2   Time                                2764 non-null   object
 3   CO2_(dinning-room)                  2764 non-null   float64
 4   CO2_room                            2764 non-null   float64
 5   Relative_humidity_(dinning-room)    2764 non-null   float64
 6   Relative_humidity_room              2764 non-null   float64
 7   Lighting_(dinning-room)             2764 non-null   float64
 8   Lighting_room                       2764 non-null   float64
 9   Meteo_Rain                          2764 non-null   float64
 10  Meteo_Sun_dusk                      2764 non-null   float64
 11  Meteo_Wind                          2764 non-null   float64
 12  Meteo_Sun_light_in_west_facade      2764 non-null   float64
 13  Meteo_Sun_light_in_east_facade      2764 non-null   float64
 14  Meteo_Sun_light_in_south_facade     2764 non-null   float64
 15  Meteo_Sun_irradiance                2764 non-null   float64
 16  Outdoor_relative_humidity_Sensor    2764 non-null   float64
 17  Day_of_the_week                     2764 non-null   float64
 18  Indoor_temperature_room             2764 non-null   float64
dtypes: float64(16), int64(1), object(2)
memory usage: 410.4+ KB
```

Activity 3: Scaling the Data

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction

Models such as linear regression need scaled data, as they follow distance based method and Gradient Descent and Tree concept no need of scaling.

```
0s
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
sc.fit(x_train)
x_test_scaled=sc.transform(x_test)
x_train_scaled =sc.fit_transform(x_train)
```

We will only perform scaling on the input values and in this project scaling is performed for only linear regression

Activity 4: Splitting data into train and test

Now let's split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set
Here x and y variables are created. On x variable, df is passed with dropping the target

variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
X=df.drop(['Indoor_temperature_room','Id','Date','Time'],axis=1)
Y=df['Indoor_temperature_room']
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3,random_state=40)
```

Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1: Liner Regression model

A function named Linear Regression is created and train and test data are passed as the parameters. Inside the function, Linear Regression algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, used r2 score

```
from sklearn.linear_model import LinearRegression
lir = LinearRegression()
lir.fit(x_train_scaled, y_train)
y_pred = lir.predict(x_test_scaled)
```

```
[136] from sklearn.metrics import r2_score
      r2_score(y_pred,y_test)

      0.1890930750206835
```

Activity 2: Random forest model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestRegressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, used r2 score

```
[137] from sklearn.ensemble import RandomForestRegressor
      rf = RandomForestRegressor()
      rf.fit(x_train,y_train)
```

```
RandomForestRegressor
RandomForestRegressor()
```

```
pred=rf.predict(x_test)
```

```
[140] from sklearn.metrics import r2_score
      r2_score(y_test,pred)
```

```
0.9452322805321266
```

Activity 3: Light Gradient Boost model

A function named lg is created and train and test data are passed as the parameters. Inside the function, LGBM Regressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model used r2 score.

```
[144] from lightgbm import LGBMRegressor
      lgbm = LGBMRegressor()
      lgbm.fit(x_train,y_train)
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000361 seco
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 3309
[LightGBM] [Info] Number of data points in the train set: 1934, number of used features: 15
[LightGBM] [Info] Start training from score 18.893650
LGBMRegressor
LGBMRegressor()
```

```
[146] pred=lgbm.predict(x_test)
```

```
from sklearn.metrics import r2_score
r2_score(y_test,pred)
```

```
0.9573017164935795
```

Activity 4: Xgboost model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model used r2score

```
[151] from xgboost import XGBRegressor
      xg=XGBRegressor()
      xg.fit(x_train,y_train)
```

```
▸ XGBRegressor
```

```
[153] pred=xg.predict(x_test)
```

```
from sklearn.metrics import r2_score
r2_score(y_test,pred)
```

```
0.9516587395812729
```

Now let's see the performance of all the models and save the best model

Activity 5: Evaluating performance of the model and saving the model

From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name). Our model is performing well. So, we are saving the model by pickle.dump().

```
[155] import pickle
      pickle.dump(rf,open('temperature.pkl','wb'))
```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

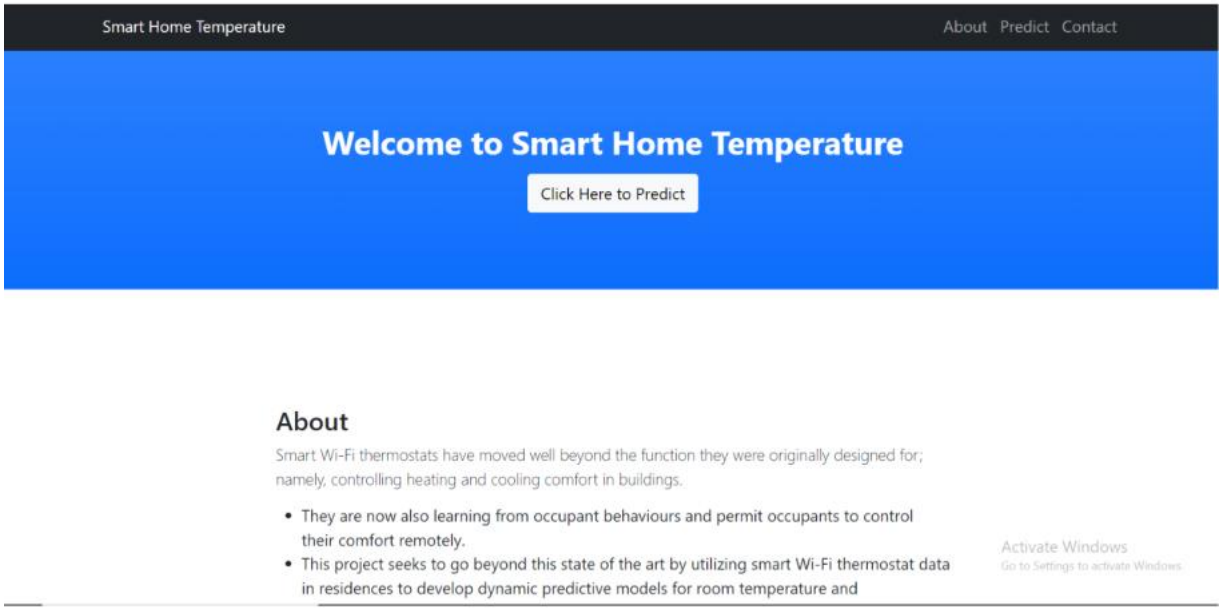
- Building HTML Pages
- Building server side script

Activity1: Building Html Pages:

For this project create three HTML files namely

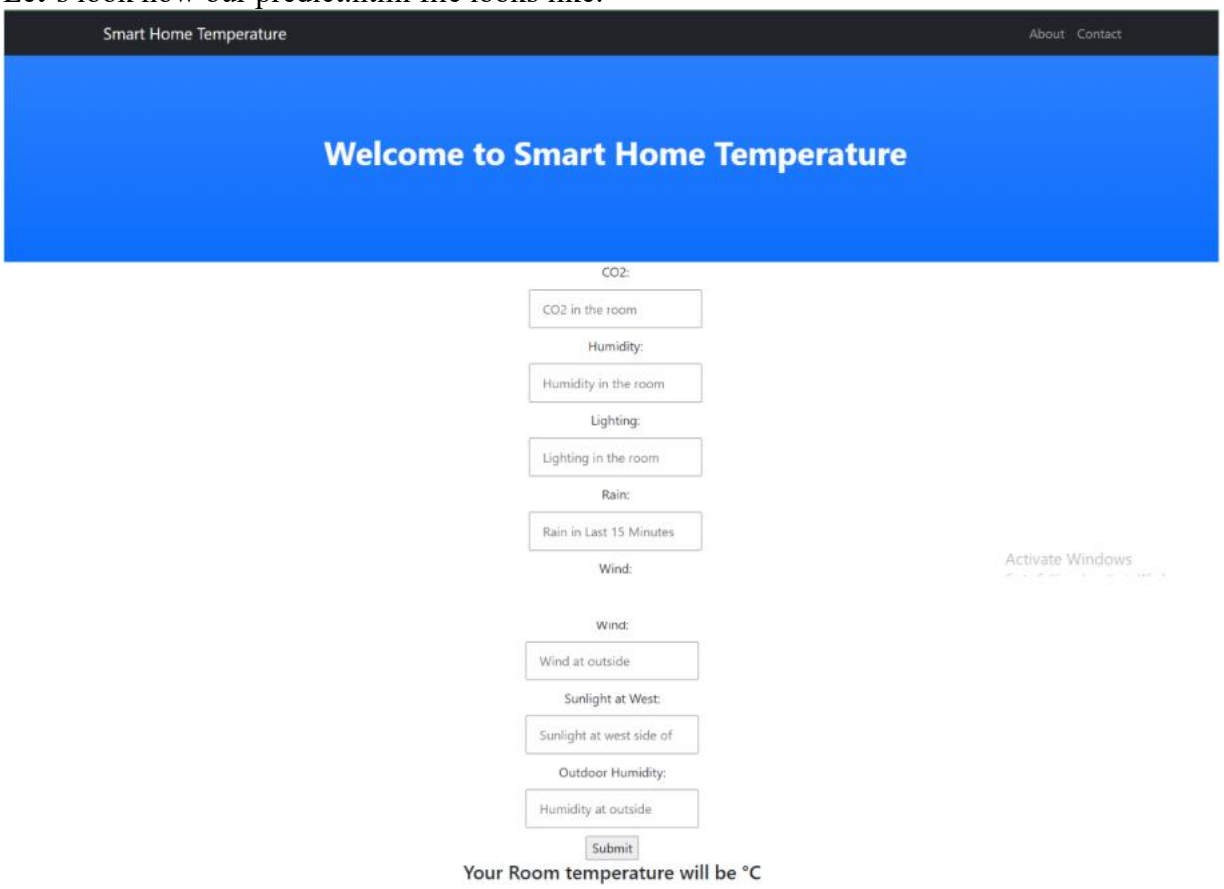
- index.html
- predict.html and save them in templates folder.

Let's see how our index.html page looks like:



Now when you click on predict button from top right corner you will get redirected to predict.html

Let's look how our predict.html file looks like:



Activity 2: Build Python code:

Import the libraries

```
from flask import Flask,request,render_template
import pickle
import numpy as np
import pandas as pd
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
model=pickle.load(open("C:\Users\pangu\Downloads\AIML PROJ"))
app=Flask(__name__)
```

Render HTML page:

```
@app.route("/")
def home():
  return render_template("index.html")

@app.route("/predict")
def predict():
  return render_template("predict.html")
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/output',methods=['post','get'])
def output():
  input_feature=[float(x) for x in request.form.values()]
  input_feature=[np.array(input_feature)]
  print(input_feature)
  names=['CO2_room','Relative_humdity_room','Lighting_room','Meteo_Rain','Meteo_wind','Outdoor_rela
  print(names)
  data=pd.DataFrame(input_feature,columns=names)
  print(data)
  prediction=model.predict(data)
  print(prediction)
  return render_template('predict.html',prediction=prediction[0])
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
if __name__=='__main__':
    app.run(debug=true)
```

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
Restarting with watchdog (windowsapi)
Debugger is active!
Debugger PIN: 857-463-000
Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Prediction: