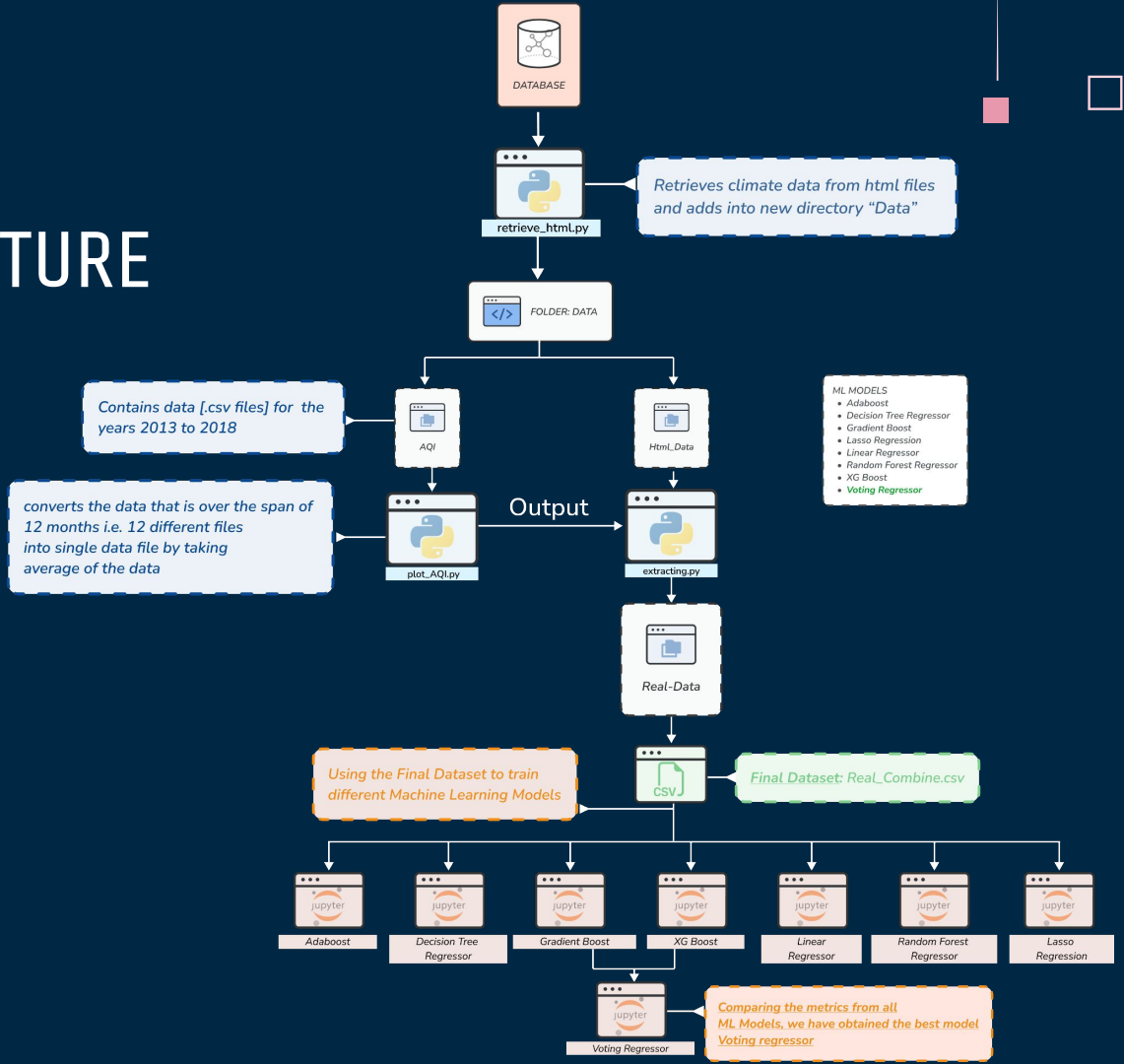
SYSTEM ARCHITECTURE

02

SYSTEM ARCHITECTURE



PROPOSED MODEL

03

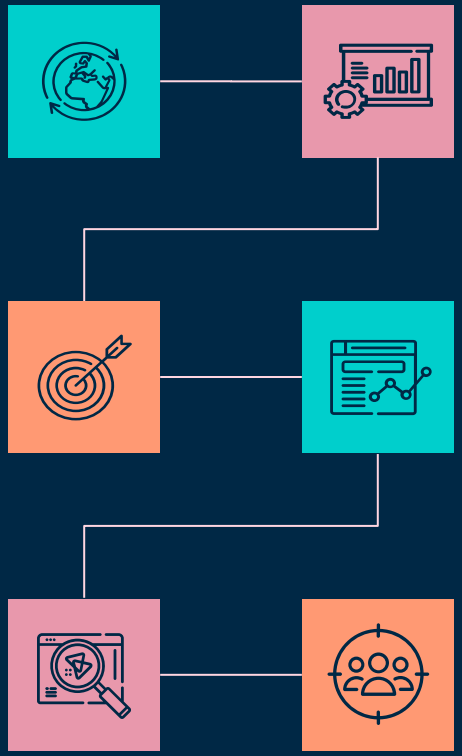
PROPOSED MODEL

DATA COLLECTION

Using Python libraries to parse and extract data from HTML files

DATA PREPROCESSING

Cleaning, transforming, and organizing the data to make it suitable for further analysis or modeling.



FEATURE SELECTION

Remove features that are noisy or irrelevant by dropping the columns and get final dataset

MODEL BUILDING

Build individual ML models and use the final dataset for training the model

MODEL EVALUATION

Using bagging and boosting techniques and evaluating performance metrics

ANALYSIS & IDENTIFICATION

Comparing the performance metrics, identifying and finalising the best ML Model

OUR PROCESS

01

DATABASE

Collect Climate data from the internet using **retrieve_html.py**

02

DATA PREPROCESSING

Use **plot_AQI.py** to combine 12 data files and use its output in **extracting.py** and create final dataset: **Real_Combine.csv**

03

BUILD MODEL

Use Bagging and Boosting Techniques in different ML Model and evaluate the performance metrics

04

FINAL MODEL

Obtain final model by identifying and comparing best **rmse** values

DATA COLLECTION

- Creates all data files, each year has 12 data files representing each month's data

retrieve_html.py

```python
import os
import time
import requests
import sys


def retrieve_html():
    for year in range(2013,2019):
        for month in range(1,13):
            if(month<10):
                url='http://en.tutiempo.net/climate/0{}-{}/ws-421820.html'.format(month
                                                                                  ,year)
            else:
                url='http://en.tutiempo.net/climate/{}-{}/ws-421820.html'.format(month
                                                                                 ,year)
            texts=requests.get(url)
            text_utf=texts.text.encode('utf=8')

            if not os.path.exists("Data/Html_Data/{}".format(year)):
                os.makedirs("Data/Html_Data/{}".format(year))
            with open("Data/Html_Data/{}/{}.html".format(year,month),"wb") as output:
                output.write(text_utf)

        sys.stdout.flush()

if __name__=="__main__":
    start_time=time.time()
    retrieve_html()
    stop_time=time.time()
    print("Time taken {}".format(stop_time-start_time))
```

DATA COLLECTION

- plot_AQI.py does the averaging and converts each year(12 data files) into one single file.
- Each year is averaged to one file and finally we get 4 files from year 2013 to 2016.

plot_AQI.py

```
import ...


def avg_data_2013():
    temp_i=0
    average=[]
    for rows in pd.read_csv('Data/AQI/aqi2013.csv',chunksize=24):
        add_var=0
        avg=0.0
        data=[]
        df=pd.DataFrame(data=rows)
        for index,row in df.iterrows():
            data.append(row['PM2.5'])
        for i in data:
            if type(i) is float or type(i) is int:
                add_var=add_var+i
            elif type(i) is str:
                if i!='NoData' and i!='PwrFail' and i!='---' and i!='InVld':
                    temp=float(i)
                    add_var=add_var+temp
        avg=add_var/24
        temp_i=temp_i+1

        average.append(avg)
    return average

def avg_data_2014():
    temp_i=0
    average=[]
```

DATA COLLECTION

- extracting.py combines all the 4 years data into one single file which is going to be used in various models.
- After combining 4 files, we get one single file, i.e, “Real_combine.csv”.
- Real_Combine.csv file is then used to feed the data into various models like:
  Linear Regression, Decision Tree regressor, Lasso regression, XGBoost, Gradient Boost and AdaBoost

extracting.py

```
import ...

def met_data(month, year):
    file_html = open('Data/html_Data/{}/{}.html'.format(year, month), 'rb')
    plain_text = file_html.read()

    tempD = []
    finalD = []

    soup = BeautifulSoup(plain_text, "lxml")
    for table in soup.findAll('table', {'class':'medias mensuales numspan'}):
        for tbody in table:
            for tr in tbody:
                a= tr.get_text()
                tempD.append(a)
    rows = len(tempD)/15

    for times in range(round(rows)):
        newtempD = []
        for i in range(15):
            newtempD.append(tempD[0])
            tempD.pop(0)
        finalD.append(newtempD)

    length = len(finalD)

    finalD.pop(length-1)
    finalD.pop(0)
```

THANK YOU!!