AIR QUALITY INDEX PREDICTION

Sachit Bandhu[1], Siddhant Saxena[2]

[1]*Having 3 Years of Experience as a Data Analyst.*
[2]*Having 2 Years of Experience as a Data Analyst.*

***

1-Abstract: The aim of our project is to provide the air quality index report of major cities of India. The Air Quality Index is an index that is used for reporting daily air quality. With the help of indexes, we know how clean or unhealthy our air is, and what associated health effects might be a concern. The Air Quality Index mainly focuses on the quality of air within our environment, as we can see most of the popular cities in India has the worst quality index just because of the increasing population and pollution, so our motive is to predict the AQI which is based on different factors like temperature pollutions and many more. Majorly AQI is calculated for four major air pollutants regulated by the Clean Air Act: particle pollution, ground-level ozone sulfur dioxide, and carbon monoxide. EPA has established national air quality standards to protect public health for each of these pollutants.

Keywords: AQI (Air Quality Index); EDA (Exploratory data Analysis)

2- INTRODUCTION

In this project we are working on datasets which contains different parameters like temperature winds, storm, snow along with other .These parameters are really important for calculating the PM values. So, for this evaluation we need to extract live data from different website thru python programming language. We have extracted the data from tutiempo.net. The complete project is divided in to different steps and each step contains lifecycle of the project.

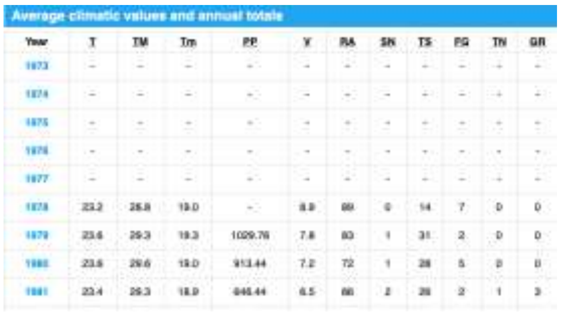
3-Project Working and Explanation

The complete life cycle of this project divided in to 5 challenges. Each of them will be explained in detail in following step.

Step1: Data Extraction from website

So, In the very first step we need to extract data from website which is in html format and we need to convert that data into HTML file so that we can perform EDA process on that file.

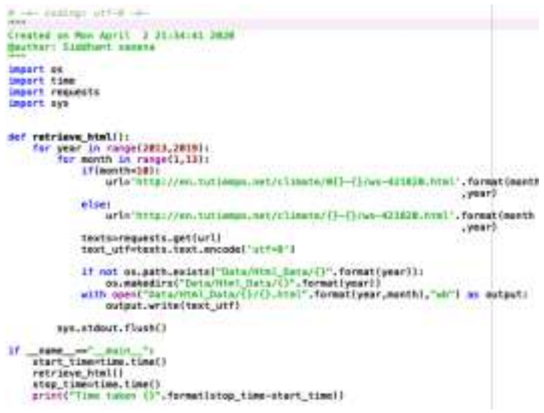
Data, which is showing in this picture contains parameters like temperature, snow level, winds storm, intensity and many more but this data is available on a website.



Average climatic values and annual totals

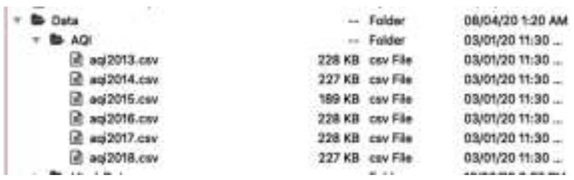
Year	T	TM	Tm	PP	V	RA	SN	TS	FG	TN	GR
1873	-	-	-	-	-	-	-	-	-	-	-
1874	-	-	-	-	-	-	-	-	-	-	-
1875	-	-	-	-	-	-	-	-	-	-	-
1876	-	-	-	-	-	-	-	-	-	-	-
1877	-	-	-	-	-	-	-	-	-	-	-
1878	23.2	28.8	19.0	-	8.8	89	0	14	7	0	0
1879	23.6	29.3	19.3	1029.76	7.8	83	1	31	2	0	0
1880	23.6	29.6	19.0	913.44	7.2	72	1	28	5	0	0
1881	23.4	29.3	18.9	845.44	6.5	86	2	28	2	1	3

So, as we can see that this data is available in table format in www.tutiempo.net but we need this data as in HTML file. This can be happen in two ways. We can use API which can directly fetch the data from particular website and convert that data into desire format or we can use python programming language with the help of Beautifulsoup we can convert HTML data into desire format here is a Python code which is used to fetch data from website.



```
# -*- coding: utf-8 -*-
"""
Created on Mon April  2 21:34:41 2020
@author: Siddhant saxena
"""
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))
```

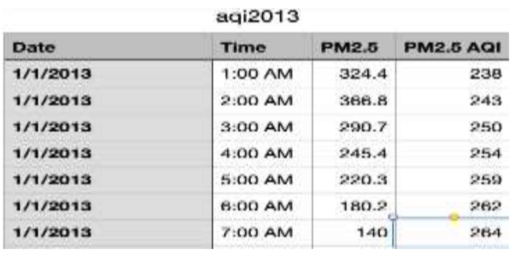
After compile above code, the HTML file will create in Data>AQI Folder now we can easily extract HTML file form that folder and do EDA process to extract useful information which will provide the required parameters.



```
▾ Data                        --      Folder    08/04/20 1:20 AM
  ▾ AQI                       --      Folder    03/01/20 11:30 ...
      aqi2013.csv         228 KB  csv File  03/01/20 11:30 ...
      aqi2014.csv         227 KB  csv File  03/01/20 11:30 ...
      aqi2015.csv         189 KB  csv File  03/01/20 11:30 ...
      aqi2016.csv         228 KB  csv File  03/01/20 11:30 ...
      aqi2017.csv         228 KB  csv File  03/01/20 11:30 ...
      aqi2018.csv         227 KB  csv File  03/01/20 11:30 ...
```

STEP2: Extraction PM2.5

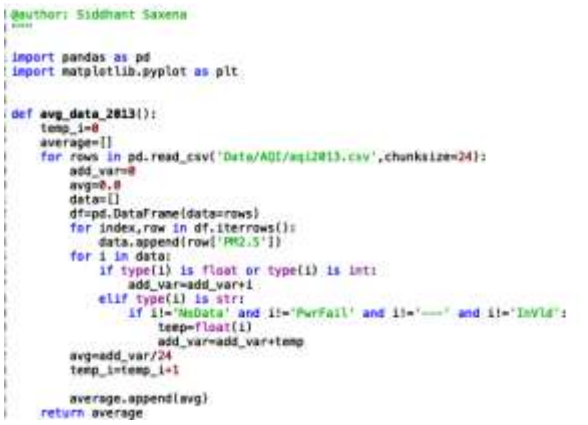
PM2.5 refers to atmospheric particulate matter PM that has a diameter of 3% of a human hair, which is about lesser than 2.5 micrometers. Since they are small and light in weight, fine particles tend to stay longer in the air compared to heavier particles which in turn increases the chances of human beings and animals inhaling them into their bodies. Some particles due to their minute size i.e. smaller than 2.5 micrometers can penetrate deep into the throat and nose and can also penetrate the lungs. In some cases, some may find their way into the circulatory system. The process which we are using here is similar to previous one, we need to fetch PM2.5 values of 2013 to 2018 thru API from a website which gives us this Utility. Data which we get thru website is in CSV format here is a screenshot of data



aqi2013

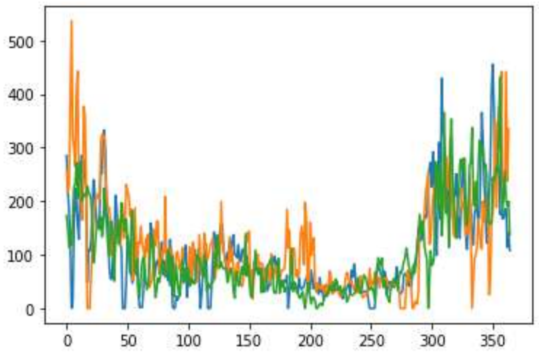
Date	Time	PM2.5	PM2.5 AQI
1/1/2013	1:00 AM	324.4	238
1/1/2013	2:00 AM	366.8	243
1/1/2013	3:00 AM	290.7	250
1/1/2013	4:00 AM	245.4	254
1/1/2013	5:00 AM	220.3	259
1/1/2013	6:00 AM	180.2	262
1/1/2013	7:00 AM	140	264

Here we have PM2.5 value with respect of each hour for 5 year but in the previous file the data is in day format so now we need to convert this data into day format this can be possible by using python programming language. In python code here we are taking the average of hourly dataset for each year.

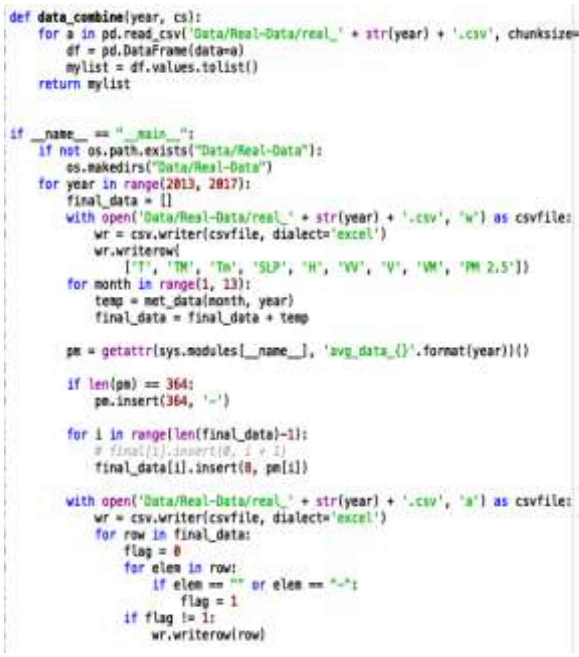


```
@author: Siddhant Saxena
"""

import pandas as pd
import matplotlib.pyplot as plt


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
```

After compile this python file we will get the PM data which is on the basis of per day. Here in output file we have constructed a plot map which shows the average dataset for each year.

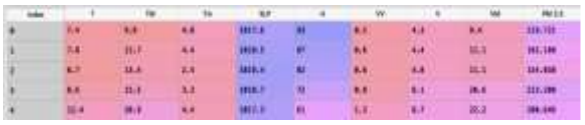


Now we need to concatenate this PM parameter to the previous file where we have required parameters but here is a problem in this dataset, this dataset required data cleaning as there are some unwanted character is present in this set so we need to eliminate that part in the cleaning process.

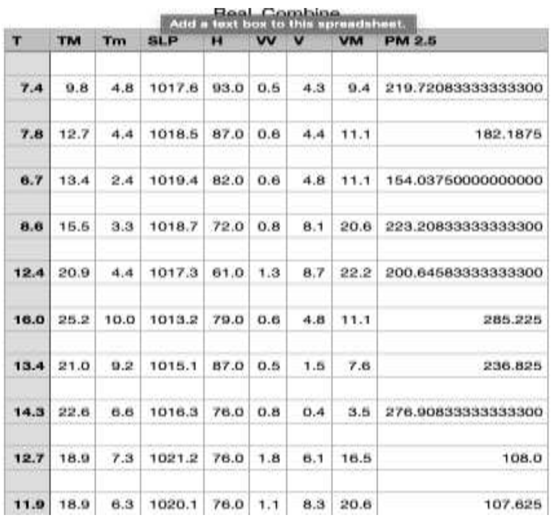


```
def data_combine(year, cs):
    for a in pd.read_csv('Data/Real-Data/real_' + str(year) + '.csv', chunksize=cs):
        df = pd.DataFrame(data=a)
        mylist = df.values.tolist()
    return mylist


if __name__ == "__main__":
    if not os.path.exists("Data/Real-Data"):
        os.makedirs("Data/Real-Data")
    for year in range(2013, 2017):
        final_data = []
        with open('Data/Real-Data/real_' + str(year) + '.csv', 'w') as csvfile:
            wr = csv.writer(csvfile, dialect='excel')
            wr.writerow(
                ['T', 'TM', 'Tm', 'SLP', 'H', 'VV', 'V', 'VM', 'PM 2.5'])
        for month in range(1, 13):
            temp = met_data(month, year)
            final_data = final_data + temp

        pm = getattr(sys.modules[__name__], 'avg_data_{}'.format(year))()

        if len(pm) == 364:
            pm.insert(364, '-')

        for i in range(len(final_data)-1):
            # final[i].insert(0, i + 1)
            final_data[i].insert(0, pm[i])

        with open('Data/Real-Data/real_' + str(year) + '.csv', 'a') as csvfile:
            wr = csv.writer(csvfile, dialect='excel')
            for row in final_data:
                flag = 0
                for elem in row:
                    if elem == "" or elem == "-":
                        flag = 1
                if flag != 1:
                    wr.writerow(row)
```

Here in this code, first we are converting HTML files to CSV format and after that we are combining that file to PM file which is already in CSV Format. By doing this we will get a Single output file in CSV format to apply EDA process this can be happened by importing Beautiful soup library from Python in build library stack. It is very useful method to manipulate html data. Here we are using convert html data in to desire format. At the end of this process we are creating a new CSV file named "**Real_combine.CSV"** which contain all required parameters such as PM values, Temp, Storm and wind velocity for each of the day and we have already removed all unwanted now we can easily apply all EDA process on this file. Here is output of final CSV File named.



index	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
0	7.4	9.8	4.8	1017.6	93	0.5	4.3	9.4	219.721
1	7.8	12.7	4.4	1018.5	87	0.6	4.4	11.1	182.188
2	6.7	13.4	2.4	1019.4	82	0.6	4.8	11.1	154.038
3	8.6	15.5	3.3	1018.7	72	0.8	8.1	20.6	223.208
4	12.4	20.9	4.4	1017.3	61	1.3	8.7	22.2	200.646



Real Combine

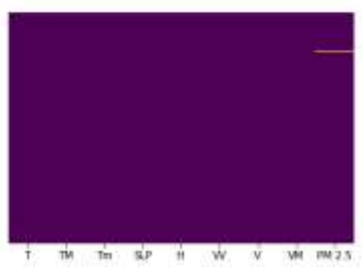
Add a text box to this spreadsheet.

T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
7.4	9.8	4.8	1017.6	93.0	0.5	4.3	9.4	219.72083333333300
7.8	12.7	4.4	1018.5	87.0	0.6	4.4	11.1	182.1875
6.7	13.4	2.4	1019.4	82.0	0.6	4.8	11.1	154.03750000000000
8.6	15.5	3.3	1018.7	72.0	0.8	8.1	20.6	223.20833333333300
12.4	20.9	4.4	1017.3	61.0	1.3	8.7	22.2	200.64583333333300
16.0	25.2	10.0	1013.2	79.0	0.6	4.8	11.1	285.225
13.4	21.0	9.2	1015.1	87.0	0.5	1.5	7.6	236.825
14.3	22.6	6.6	1016.3	76.0	0.8	0.4	3.5	276.90833333333300
12.7	18.9	7.3	1021.2	76.0	1.8	6.1	16.5	108.0
11.9	18.9	6.3	1020.1	76.0	1.1	8.3	20.6	107.625

STEP3: Exploratory Data Analysis

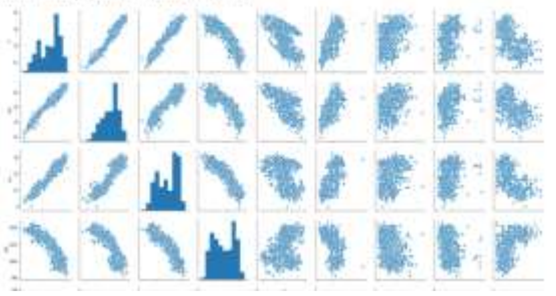
Here we are in out EDA process. As of now we have Data to analyze, first we need to find out the null value in the given data. Since we have a large data set, we can't do this by searching manually so we have created a heat map which shows the negative value in the given data set by applying simple python programming module.

```
## Check for null values

sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
<matplotlib.axes._subplots.AxesSubplot at 0x20446907dd8>
```

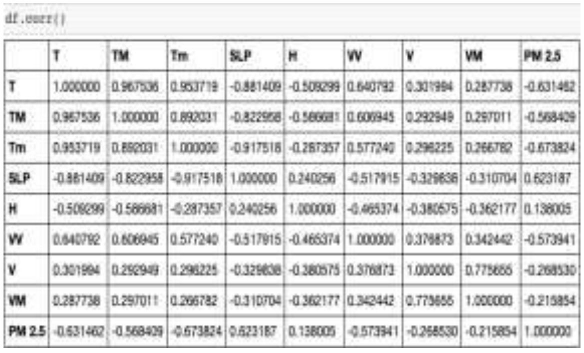


By visualizing the heat-map we can clearly see that there is a null value in PM Column so now we need to eliminate that null value, by using df.dropna() function we can do the same. Now we have done the basic data engineering on dataset, all null values have been eliminated so far and also we have done all data cleaning part now we need to divide our data into dependent and independent feature. After division of data features, we have created pair plot on features so that we can visualize the relation between dependent and independent features.

```
sns.pairplot(df)
<seaborn.axisgrid.PairGrid at 0x29e0829278d>
```



We can also create a correlation graph between the feature, by doing this we will get to know better relationship between them because only then we can apply a suitable machine learning algorithm.



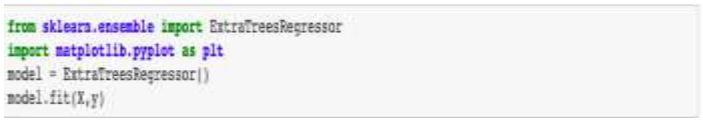
```
df.corr()
```

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
T	1.000000	0.967536	0.953719	-0.881409	-0.509299	0.640792	0.301994	0.287738	-0.631462
TM	0.967536	1.000000	0.892031	-0.822958	-0.586681	0.606945	0.292949	0.297011	-0.568409
Tm	0.953719	0.892031	1.000000	-0.917518	-0.287357	0.577240	0.296225	0.266782	-0.673824
SLP	-0.881409	-0.822958	-0.917518	1.000000	0.240256	-0.517915	-0.329638	-0.310704	0.623187
H	-0.509299	-0.586681	-0.287357	0.240256	1.000000	-0.465374	-0.380575	-0.362177	0.138005
VV	0.640792	0.606945	0.577240	-0.517915	-0.465374	1.000000	0.376873	0.342442	-0.573941
V	0.301994	0.292949	0.296225	-0.329638	-0.380575	0.376873	1.000000	0.775655	-0.268530
VM	0.287738	0.297011	0.266782	-0.310704	-0.362177	0.342442	0.775655	1.000000	-0.215854
PM 2.5	-0.631462	-0.568409	-0.673824	0.623187	0.138005	-0.573941	-0.268530	-0.215854	1.000000

Here this table showing the positive and negative relationship between dependent and independent feature of the dataset basically correlation states how the features are related to each other or the target variable in the given dataset. Correlation can be positive which shows the increase in one value of feature increases the value of the target variable and can be negative which indicates an increase in one principle of feature results to decreases the value of the target variable. Heatmap makes it easy to identify which features are most related to the target variable, we will plot a heatmap of correlated features using the seaborn library.

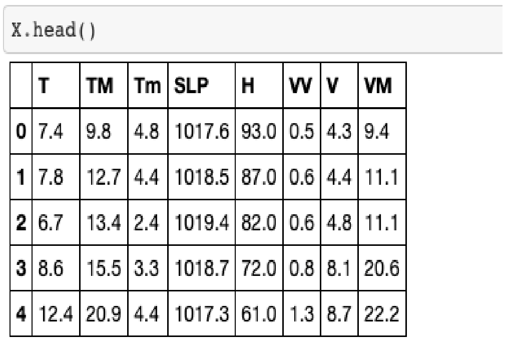
Feature Importance

Feature importance is very important in term of EDA because in current data set we don't have thousands of feature but in real scenario it might be up to thousand so we need to evaluate which features need to be analysis so we have already create a correlation map from where we ca easily extract the required one. We can get the importance of features by using "feature importance property", it gives you score for each feature, the higher the score will get more importance but for that first we need to import the required library.



```
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

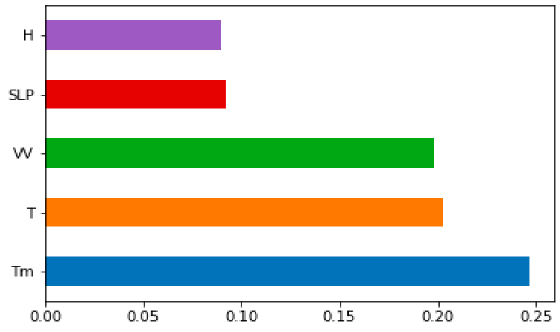
This feature is an in build function in Tree based regression, here we are using the same technique which extract the top 10 features in the given data set.



```
X.head()
```

	T	TM	Tm	SLP	H	VV	V	VM
0	7.4	9.8	4.8	1017.6	93.0	0.5	4.3	9.4
1	7.8	12.7	4.4	1018.5	87.0	0.6	4.4	11.1
2	6.7	13.4	2.4	1019.4	82.0	0.6	4.8	11.1
3	8.6	15.5	3.3	1018.7	72.0	0.8	8.1	20.6
4	12.4	20.9	4.4	1017.3	61.0	1.3	8.7	22.2

So, here is a visitation graph which indicates the top required features of the given data set , here we are taken only 5 features as we need only them but in real world problems we can extract as much we can depending upon the dataset requirement.

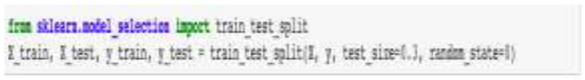


Data Split

Basically, Machine learning algorithm works in two different stages so we usually split the data to apply machine learning algorithm. The whole data is divided in two different parts, which is generally 80 and 20 percent.

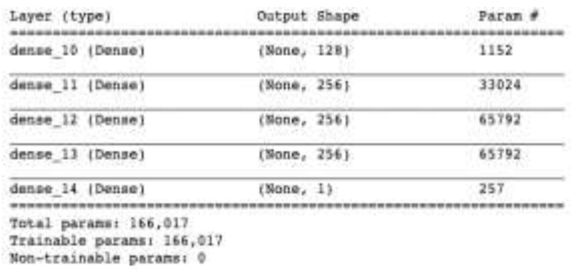
Dataset
Training Set
Test Set

Same approach we are using here by diving out data in Test and Train data set. Here is the python code which splits out data into desire form.



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

By using the train test split here we have split our data set now our task is to do training our data set so that we can achieve our desire output but to train our data set we need to apply machine learning algorithm on our data sets so this is our final and most important task to perform.

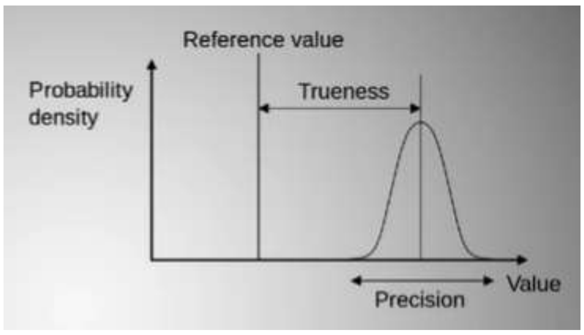


```
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 128)               1152
_________________________________________________________________
dense_11 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_12 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_13 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 257
=================================================================
Total params: 166,017
Trainable params: 166,017
Non-trainable params: 0
```

4-Machine Learning Algorithm

- **Random Forest:**

Random forest or Random Decision forest is a procedure that controlled by constructing multiple decision trees during training phase. The Decision of most of the trees is chosen by random forest as the final decision. They are capable of both Classification and Regression tasks. As the name suggest, this algorithm creates the forest with number of decision trees. In general, the more trees in the forest, the more robust the prediction and this high accuracy. To model multiple decision trees to create the forest you are going to use the same method of constructing the decision with the information gain or Gini index approach amongst other algorithms.



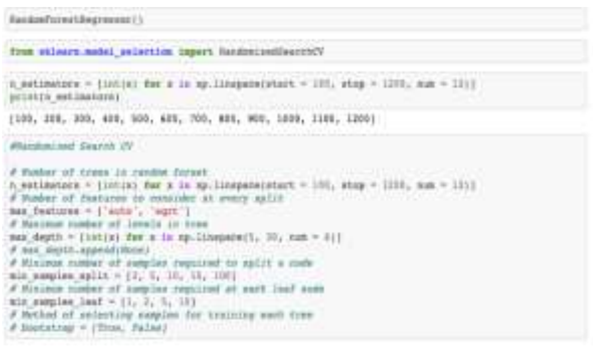
WHY RANDOM FOREST?

It is always important to understand why we use this tool mover the other on? What are the benefits here?

No overfitting: Overfitting means we have fit the data sx o close in the data sample and then we pick up on all the weird parts and instead of predicting the overall data we are predicting the weird stuff. Use of multiple trees reduce the risk of over fitting. Here training time is less.

High accuracy: Runs efficiently on large database. For large data, it predicts highly accurate predictions. In today's world of big data this is very important and this is probably where it really shines. Therefore, Random Forest comes in.

Estimate missing data: In today's world data is messy, so we have Random forest it can maintain accuracy when a large proportion is missing. What that means is when you have data that comes in from five or six different areas. And maybe they took one set of statistics in one area and they took slightly different set of statistics in the other so they have some of the same shared data but one is missing. Then it will build trees and can do very good job of guessing which one fits better even though its missing from the data.

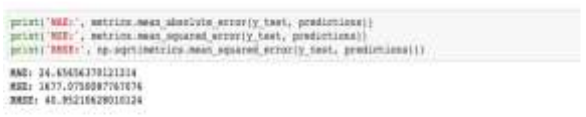


```
RandomForestRegressor()

from sklearn.model_selection import RandomizedSearchCV

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
print(n_estimators)

[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]

#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
# Method of selecting samples for training each tree
# bootstrap = [True, False]
```

So now we have applied random forest algorithm to our given datasets to get the highest accuracy on given true values. Now after applying the Random forest regressor to the given data sets we need to evaluate the mean square error mean Absolute error.

5 Mean Square/Absolute Error:

This is the very last part of any machine learning model to calculate the square mean error or absolute error, by this we can easily evaluate our working model. So, here is the value of our model to proper evaluation on the given paraments.



```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))

MAE: 24.65656370121314
MSE: 1677.0758087767676
RMSE: 40.95215629010124
```

Regression Evaluation Matrix:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors, it is easy to understand because its's an average error.

Mean Squared Error (MSE) is the mean of the squared errors, it is more popular than MAE because MSE "punishes" larger error which leads to be useful in this real world.

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors, it is even more popular than MSE because RMSE is interpretable in the "y" units.

6-Refrences

[1]The Elements of statistics Leaning, Book by Jerome H. Friedman, Robert Tibshirani

[2] Hands on machine leaning with scikit-learning-Book by Aurelien Geron

[3] Machine Learning for Absolute Beginners: A Plain English Introduction, Book by O. Theobald

[4] Python Machine Learning, Book by Sebastian Raschka

[5] Hands-On Python for Finance: A Practical Guide to Implementing Financial Book by Krish Naik

[6] Machine Learning: The Art and Science of Algorithms that Make Sense of Data Book by Peter Flach