AIR QUALITY

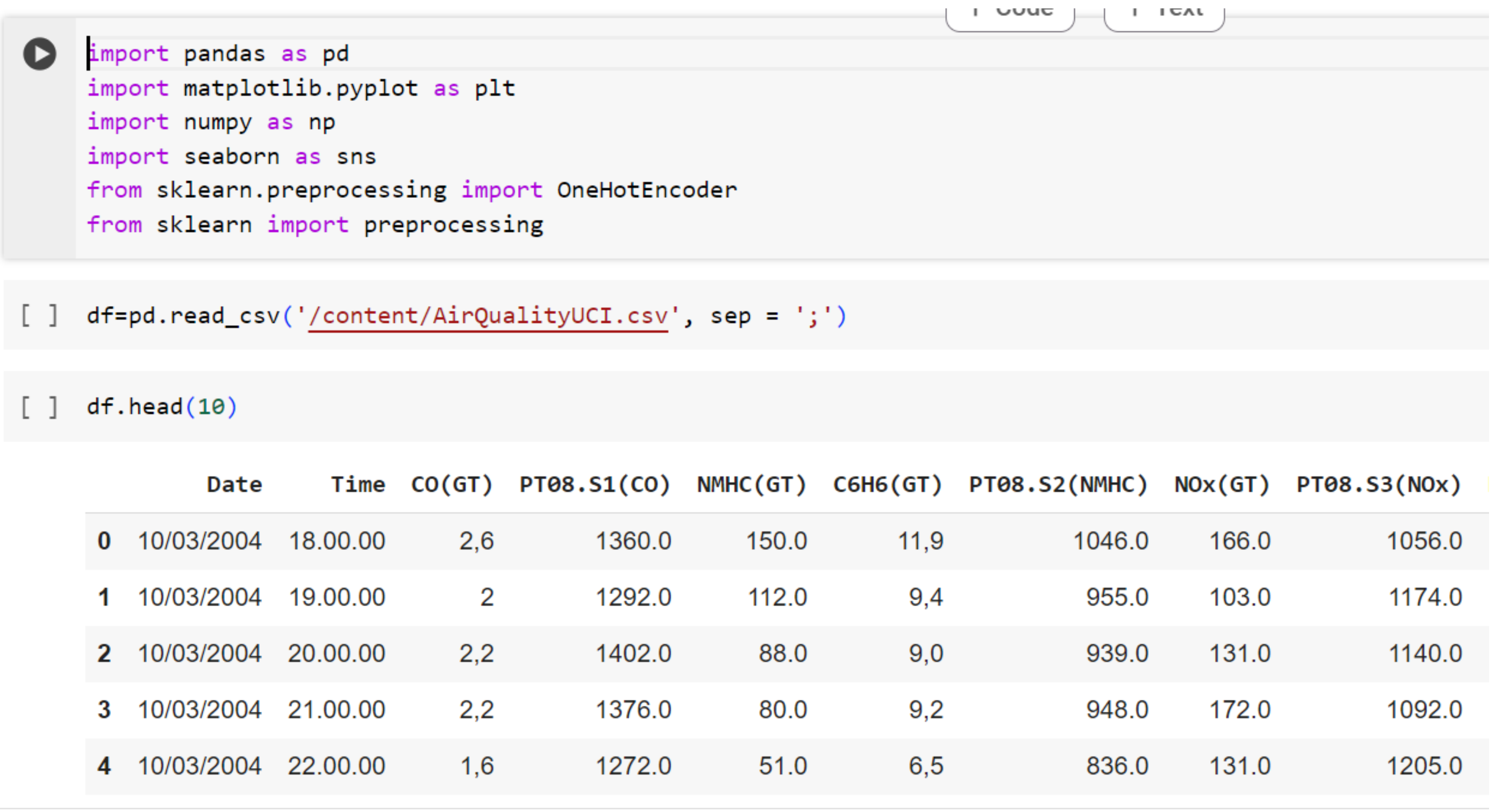
SUBMITTED BY :SOURAV SHARMA 894097

ABOUT THE DATA SET

This dataset contains the responses of a gas multisensor device deployed on the field in an Italian city. Hourly responses averages are recorded along with gas concentrations references from a certified analyzer.

The dataset contains 9358 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level,within an Italian city. Data were recorded from March 2004 to February 2005 (one year)representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses. Ground Truth hourly averaged concentrations for CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) and were provided by a co-located reference certified analyzer

IMPORTING FILE AND SEE FIRST 10 ROWS



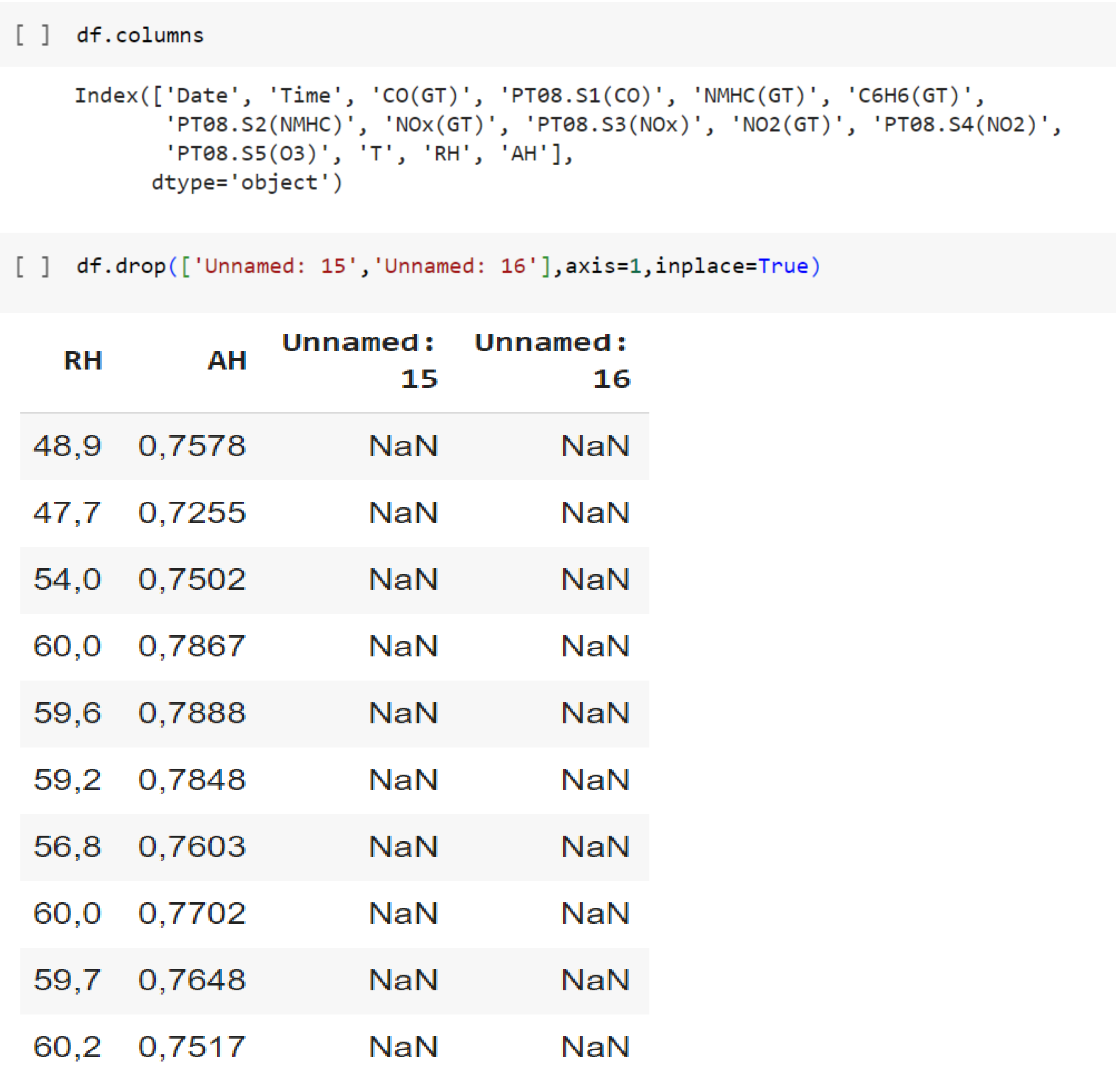
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn import preprocessing
```

```
[ ]  df=pd.read_csv('/content/AirQualityUCI.csv', sep = ';')
```

```
[ ]  df.head(10)
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)
0	10/03/2004	18.00.00	2,6	1360.0	150.0	11,9	1046.0	166.0	1056.0
1	10/03/2004	19.00.00	2	1292.0	112.0	9,4	955.0	103.0	1174.0
2	10/03/2004	20.00.00	2,2	1402.0	88.0	9,0	939.0	131.0	1140.0
3	10/03/2004	21.00.00	2,2	1376.0	80.0	9,2	948.0	172.0	1092.0
4	10/03/2004	22.00.00	1,6	1272.0	51.0	6,5	836.0	131.0	1205.0

REMOVING THE BLANK COLUMNS FROM THE DATA SET



```
[ ]  df.columns
```

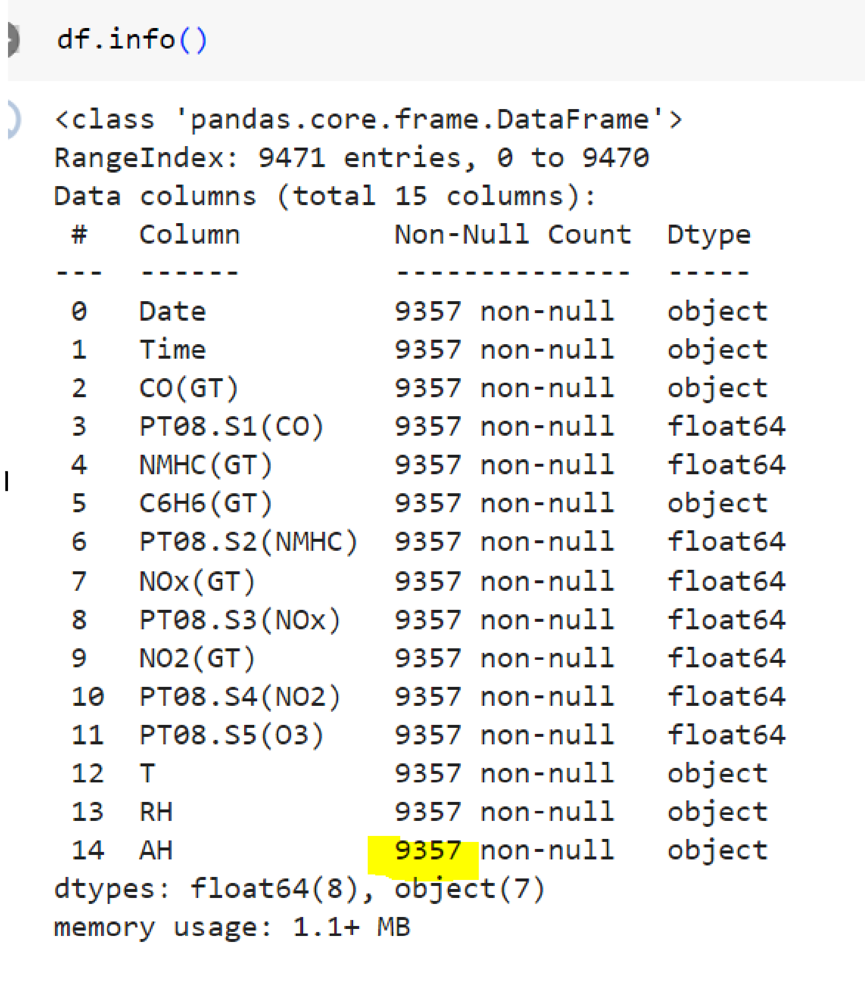
```
Index(['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(GT)',
       'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)',
       'PT08.S5(O3)', 'T', 'RH', 'AH'],
      dtype='object')
```

```
[ ]  df.drop(['Unnamed: 15','Unnamed: 16'],axis=1,inplace=True)
```

RH	AH	Unnamed: 15	Unnamed: 16
48,9	0,7578	NaN	NaN
47,7	0,7255	NaN	NaN
54,0	0,7502	NaN	NaN
60,0	0,7867	NaN	NaN
59,6	0,7888	NaN	NaN
59,2	0,7848	NaN	NaN
56,8	0,7603	NaN	NaN
60,0	0,7702	NaN	NaN
59,7	0,7648	NaN	NaN
60,2	0,7517	NaN	NaN

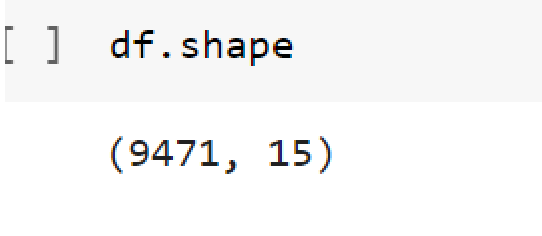
LET'S TAKE A CLOSER LOOK AT THE NULL DATA

- As we can see there are 9357 non null vales rest ar null values. So we can remove them.



```
df.info()
```

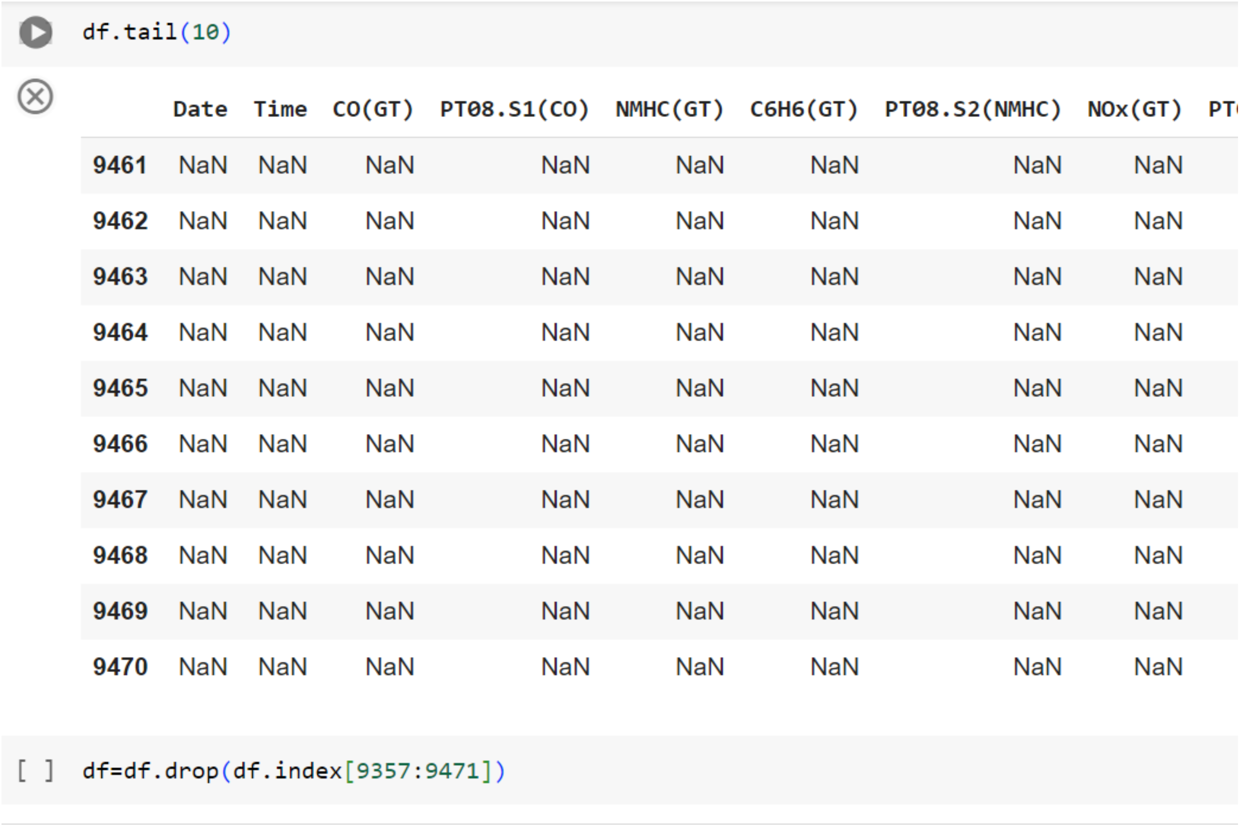
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9471 entries, 0 to 9470
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           9357 non-null   object 
 1   Time           9357 non-null   object 
 2   CO(GT)         9357 non-null   object 
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   float64
 5   C6H6(GT)       9357 non-null   object 
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
 12  T              9357 non-null   object 
 13  RH             9357 non-null   object 
 14  AH             9357 non-null   object 
dtypes: float64(8), object(7)
memory usage: 1.1+ MB
```



```
[ ]  df.shape
```

```
(9471, 15)
```

REMOVING NULL VALUES



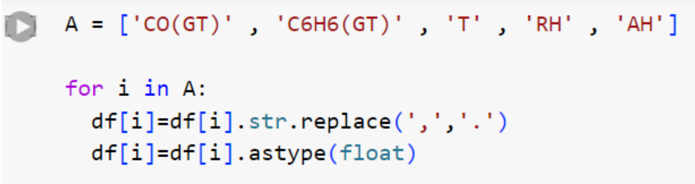
```
df.tail(10)
```

```
      Date  Time  CO(GT)  PT08.S1(CO)  NMHC(GT)  C6H6(GT)  PT08.S2(NMHC)  NOx(GT)  PT
9461   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9462   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9463   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9464   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9465   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9466   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9467   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9468   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9469   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
9470   NaN   NaN     NaN          NaN       NaN       NaN            NaN      NaN
```

```
[ ]  df=df.drop(df.index[9357:9471])
```

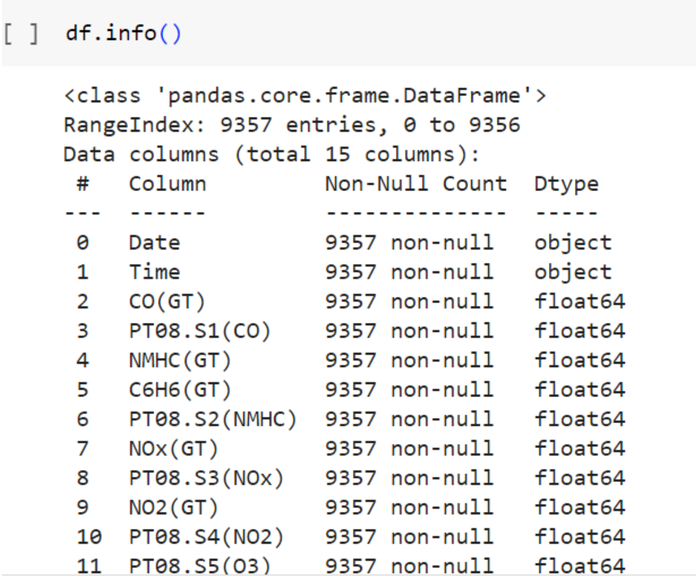
DATA CLEANING

- We can see that some columns are not integer. We should change the string columns to an integer which we can use:



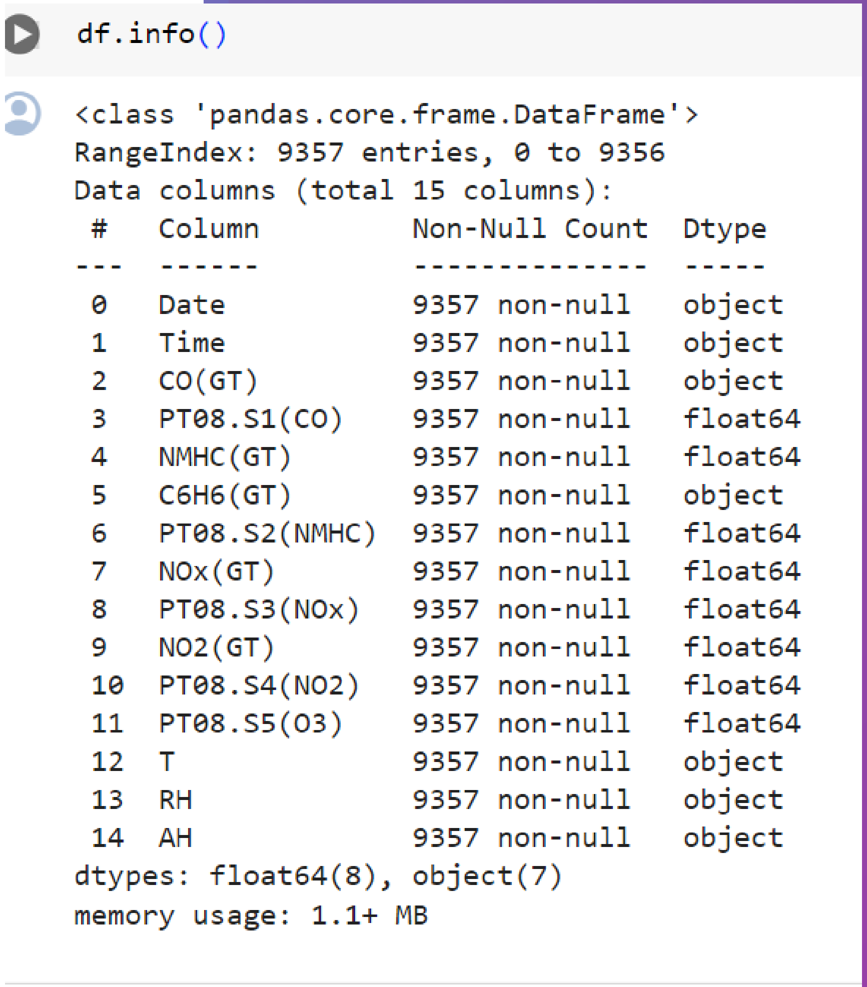
```
A = ['CO(GT)' , 'C6H6(GT)' , 'T' , 'RH' , 'AH']

for i in A:
  df[i]=df[i].str.replace(',','.')
  df[i]=df[i].astype(float)
```



```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           9357 non-null   object
 1   Time           9357 non-null   object
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   float64
 5   C6H6(GT)       9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
```

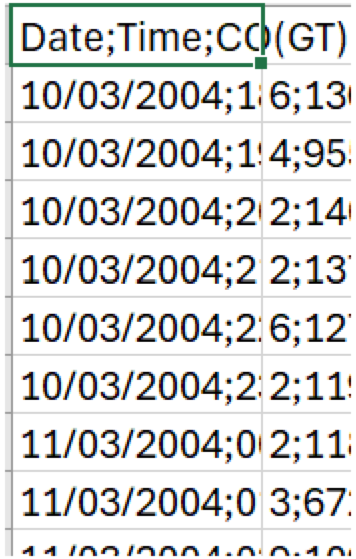


```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           9357 non-null   object
 1   Time           9357 non-null   object
 2   CO(GT)         9357 non-null   object
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   float64
 5   C6H6(GT)       9357 non-null   object
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
 12  T              9357 non-null   object
 13  RH             9357 non-null   object
 14  AH             9357 non-null   object
dtypes: float64(8), object(7)
memory usage: 1.1+ MB
```

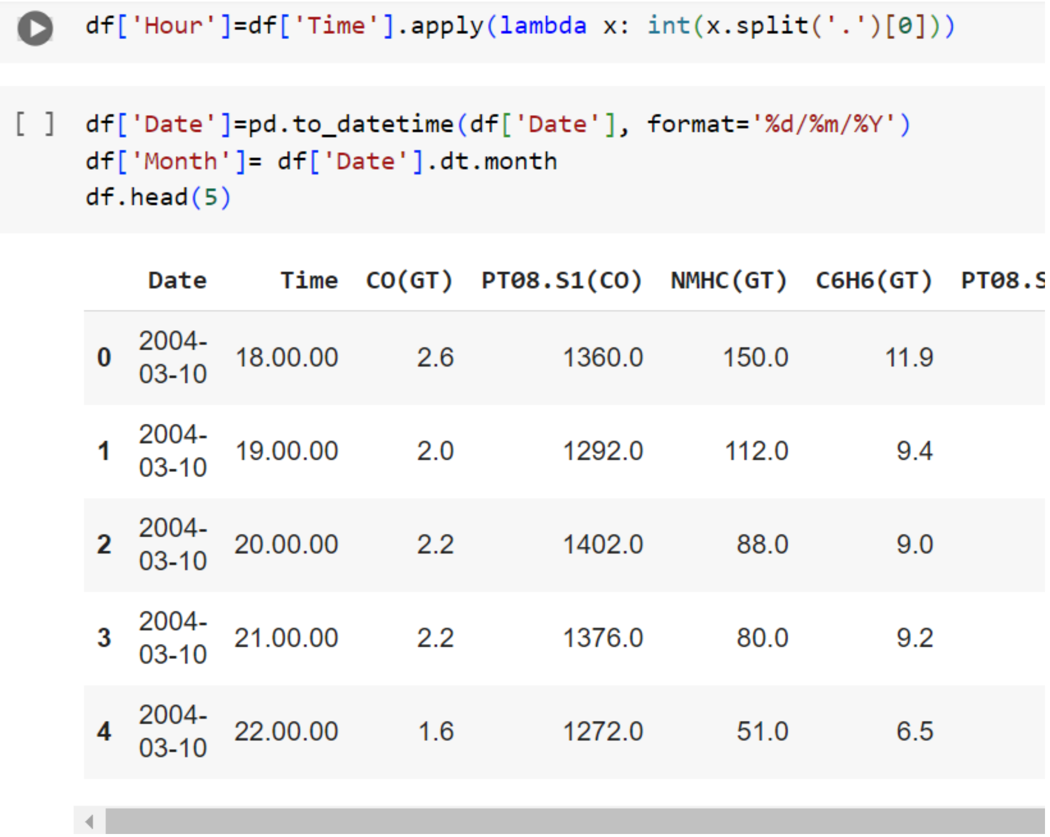
SETTING THE DATE FORMAT

- From below to format to YYYY-MM-DD



Date;Time;CO	(GT)
10/03/2004;1	6;13
10/03/2004;1	4;95
10/03/2004;2	2;14
10/03/2004;2	2;13
10/03/2004;2	6;12
10/03/2004;2	2;11
11/03/2004;0	2;11
11/03/2004;0	3;67

SETTING UP THE DATE



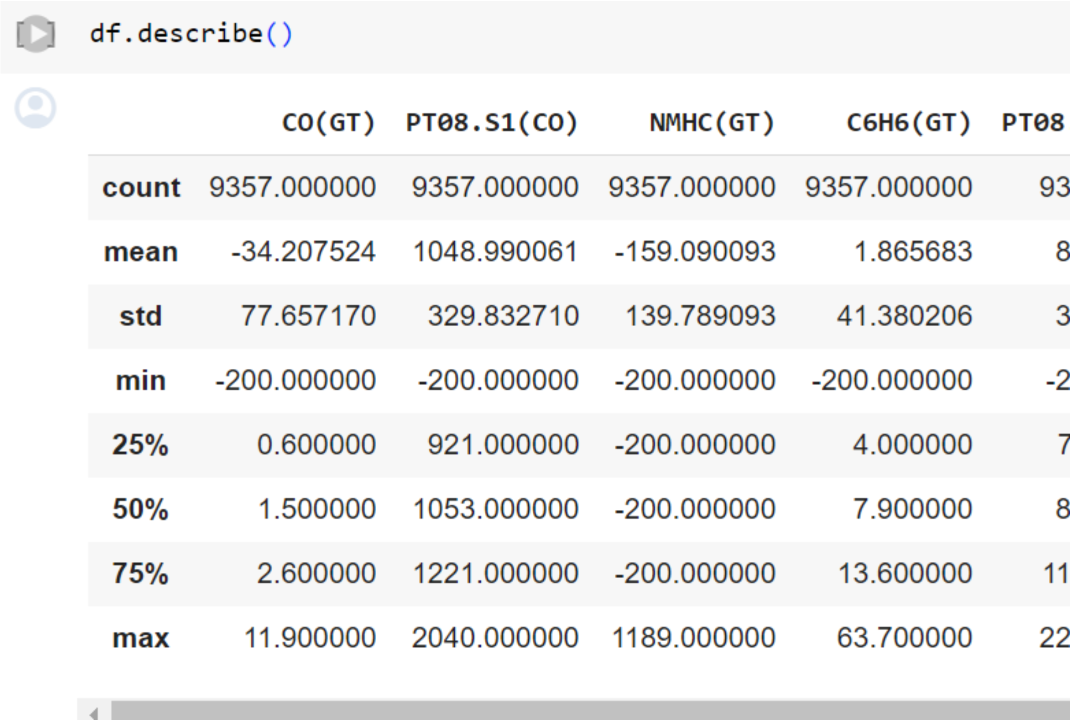
```
df['Hour']=df['Time'].apply(lambda x: int(x.split('.')[0]))
```

```
[ ] df['Date']=pd.to_datetime(df['Date'], format='%d/%m/%Y')
    df['Month']= df['Date'].dt.month
    df.head(5)
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S
0	2004-03-10	18.00.00	2.6	1360.0	150.0	11.9	
1	2004-03-10	19.00.00	2.0	1292.0	112.0	9.4	
2	2004-03-10	20.00.00	2.2	1402.0	88.0	9.0	
3	2004-03-10	21.00.00	2.2	1376.0	80.0	9.2	
4	2004-03-10	22.00.00	1.6	1272.0	51.0	6.5	

EXPLORING DATA SET USING DESCRIBE()

- Describe() give output a summary of the numerical columns in your DataFrame, including count, mean, standard deviation, minimum, maximum, and various percentiles.

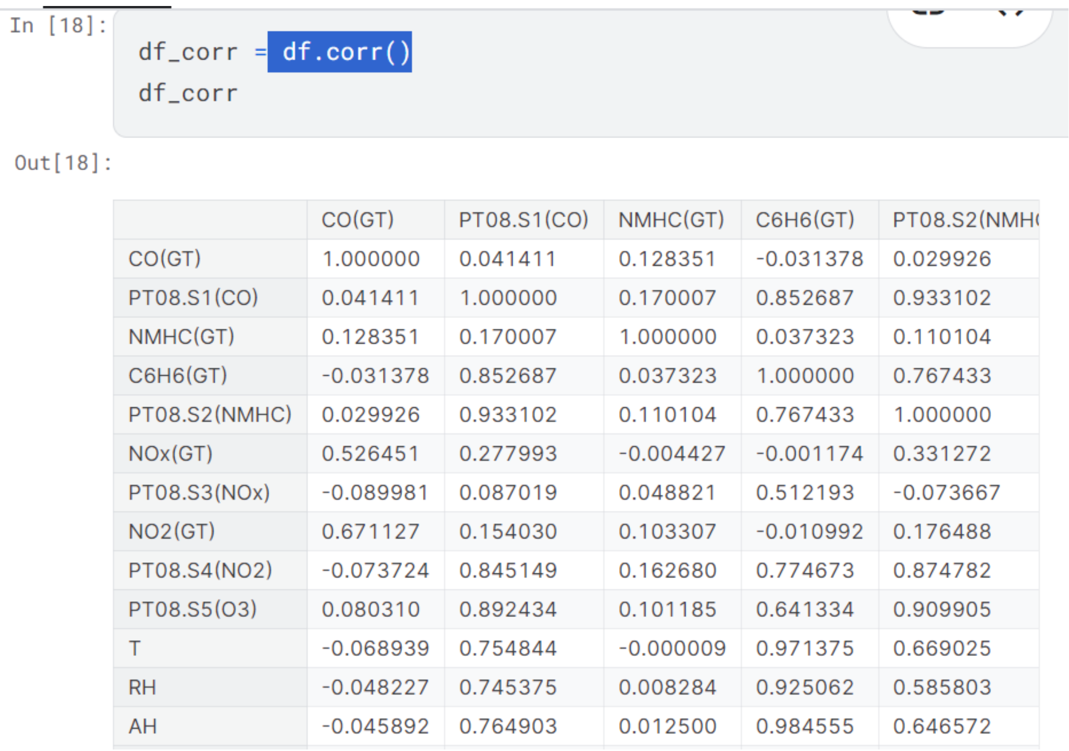


```
df.describe()
```

	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08
count	9357.000000	9357.000000	9357.000000	9357.000000	93
mean	-34.207524	1048.990061	-159.090093	1.865683	8
std	77.657170	329.832710	139.789093	41.380206	3
min	-200.000000	-200.000000	-200.000000	-200.000000	-2
25%	0.600000	921.000000	-200.000000	4.000000	7
50%	1.500000	1053.000000	-200.000000	7.900000	8
75%	2.600000	1221.000000	-200.000000	13.600000	11
max	11.900000	2040.000000	1189.000000	63.700000	22

USING CORR()

- This will display the correlation coefficients between all pairs of numerical columns in your DataFrame. The values will range from -1 to 1, where:
- 1 indicates a perfect positive correlation,
- -1 indicates a perfect negative correlation, and
- 0 indicates no correlation.
- Positive values indicate a positive correlation (as one variable increases, the other tends to increase as well), while negative values indicate a negative correlation (as one variable increases, the other tends to decrease).

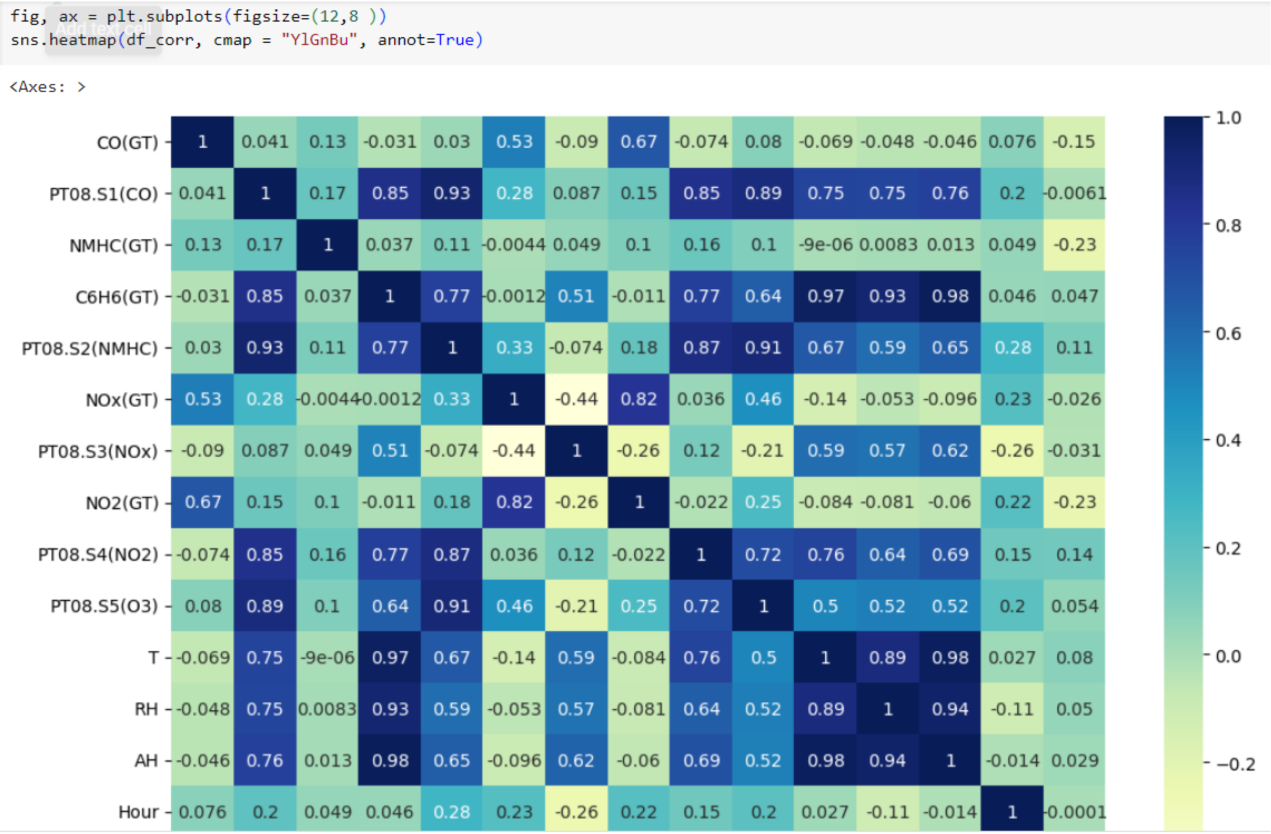


```
In [18]:
df_corr = df.corr()
df_corr
```

Out[18]:

	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMH
CO(GT)	1.000000	0.041411	0.128351	-0.031378	0.029926
PT08.S1(CO)	0.041411	1.000000	0.170007	0.852687	0.933102
NMHC(GT)	0.128351	0.170007	1.000000	0.037323	0.110104
C6H6(GT)	-0.031378	0.852687	0.037323	1.000000	0.767433
PT08.S2(NMHC)	0.029926	0.933102	0.110104	0.767433	1.000000
NOx(GT)	0.526451	0.277993	-0.004427	-0.001174	0.331272
PT08.S3(NOx)	-0.089981	0.087019	0.048821	0.512193	-0.073667
NO2(GT)	0.671127	0.154030	0.103307	-0.010992	0.176488
PT08.S4(NO2)	-0.073724	0.845149	0.162680	0.774673	0.874782
PT08.S5(O3)	0.080310	0.892434	0.101185	0.641334	0.909905
T	-0.068939	0.754844	-0.000009	0.971375	0.669025
RH	-0.048227	0.745375	0.008284	0.925062	0.585803
AH	-0.045892	0.764903	0.012500	0.984555	0.646572

HEAT MAP

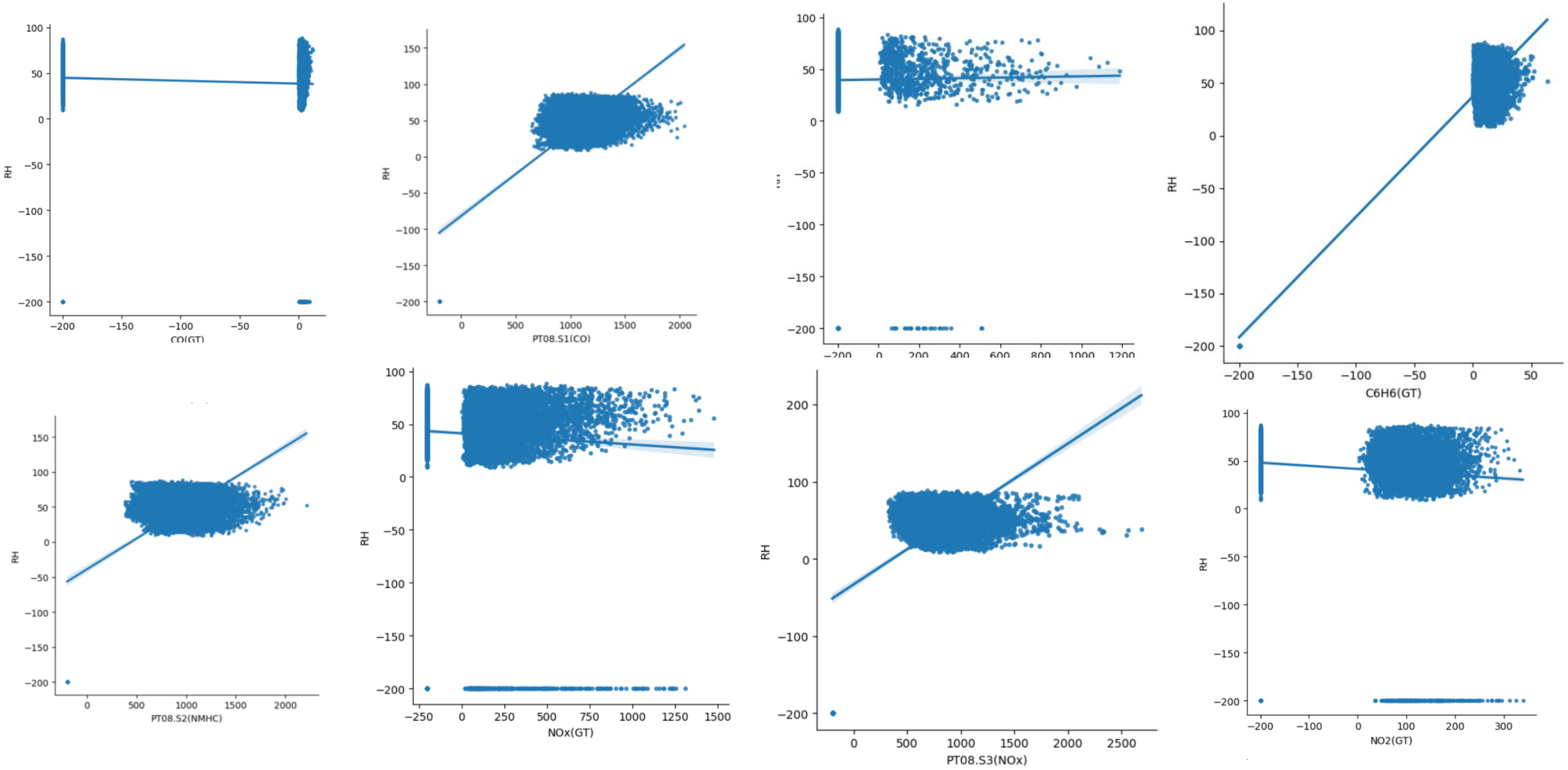


```
fig, ax = plt.subplots(figsize=(12,8 ))
sns.heatmap(df_corr, cmap = "YlGnBu", annot=True)
```

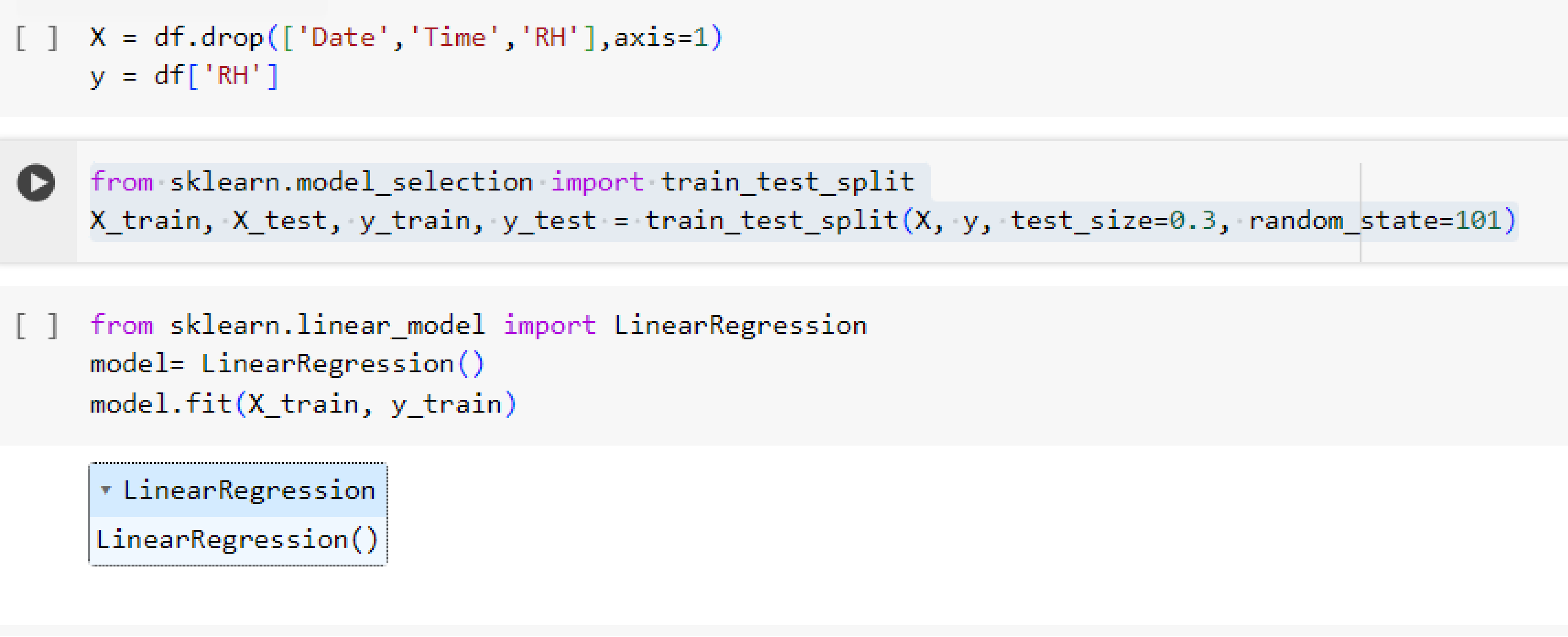
```
<Axes: >
```

CO(GT)	1	0.041	0.13	-0.031	0.03	0.53	-0.09	0.67	-0.074	0.08	-0.069	-0.048	-0.046	0.076	-0.15
PT08.S1(CO)	0.041	1	0.17	0.85	0.93	0.28	0.087	0.15	0.85	0.89	0.75	0.75	0.76	0.2	-0.0061
NMHC(GT)	0.13	0.17	1	0.037	0.11	-0.0044	0.049	0.1	0.16	0.1	-9e-06	0.0083	0.013	0.049	-0.23
C6H6(GT)	-0.031	0.85	0.037	1	0.77	-0.0012	0.51	-0.011	0.77	0.64	0.97	0.93	0.98	0.046	0.047
PT08.S2(NMHC)	0.03	0.93	0.11	0.77	1	0.33	-0.074	0.18	0.87	0.91	0.67	0.59	0.65	0.28	0.11
NOx(GT)	0.53	0.28	-0.0044	-0.0012	0.33	1	-0.44	0.82	0.036	0.46	-0.14	-0.053	-0.096	0.23	-0.026
PT08.S3(NOx)	-0.09	0.087	0.049	0.51	-0.074	-0.44	1	-0.26	0.12	-0.21	0.59	0.57	0.62	-0.26	-0.031
NO2(GT)	0.67	0.15	0.1	-0.011	0.18	0.82	-0.26	1	-0.022	0.25	-0.084	-0.081	-0.06	0.22	-0.23
PT08.S4(NO2)	-0.074	0.85	0.16	0.77	0.87	0.036	0.12	-0.022	1	0.72	0.76	0.64	0.69	0.15	0.14
PT08.S5(O3)	0.08	0.89	0.1	0.64	0.91	0.46	-0.21	0.25	0.72	1	0.5	0.52	0.52	0.2	0.054
T	-0.069	0.75	-9e-06	0.97	0.67	-0.14	0.59	-0.084	0.76	0.5	1	0.89	0.98	0.027	0.08
RH	-0.048	0.75	0.0083	0.93	0.59	-0.053	0.57	-0.081	0.64	0.52	0.89	1	0.94	-0.11	0.05
AH	-0.046	0.76	0.013	0.98	0.65	-0.096	0.62	-0.06	0.69	0.52	0.98	0.94	1	-0.014	0.029
Hour	0.076	0.2	0.049	0.046	0.28	0.23	-0.26	0.22	0.15	0.2	0.027	-0.11	-0.014	1	-0.0001

SCATTER PLOT TO FOR EACH NUMERICAL COLUMN AGAINST 'RH'



BUILDING MODEL WITH 30% TEST 70% TRAIN DATA



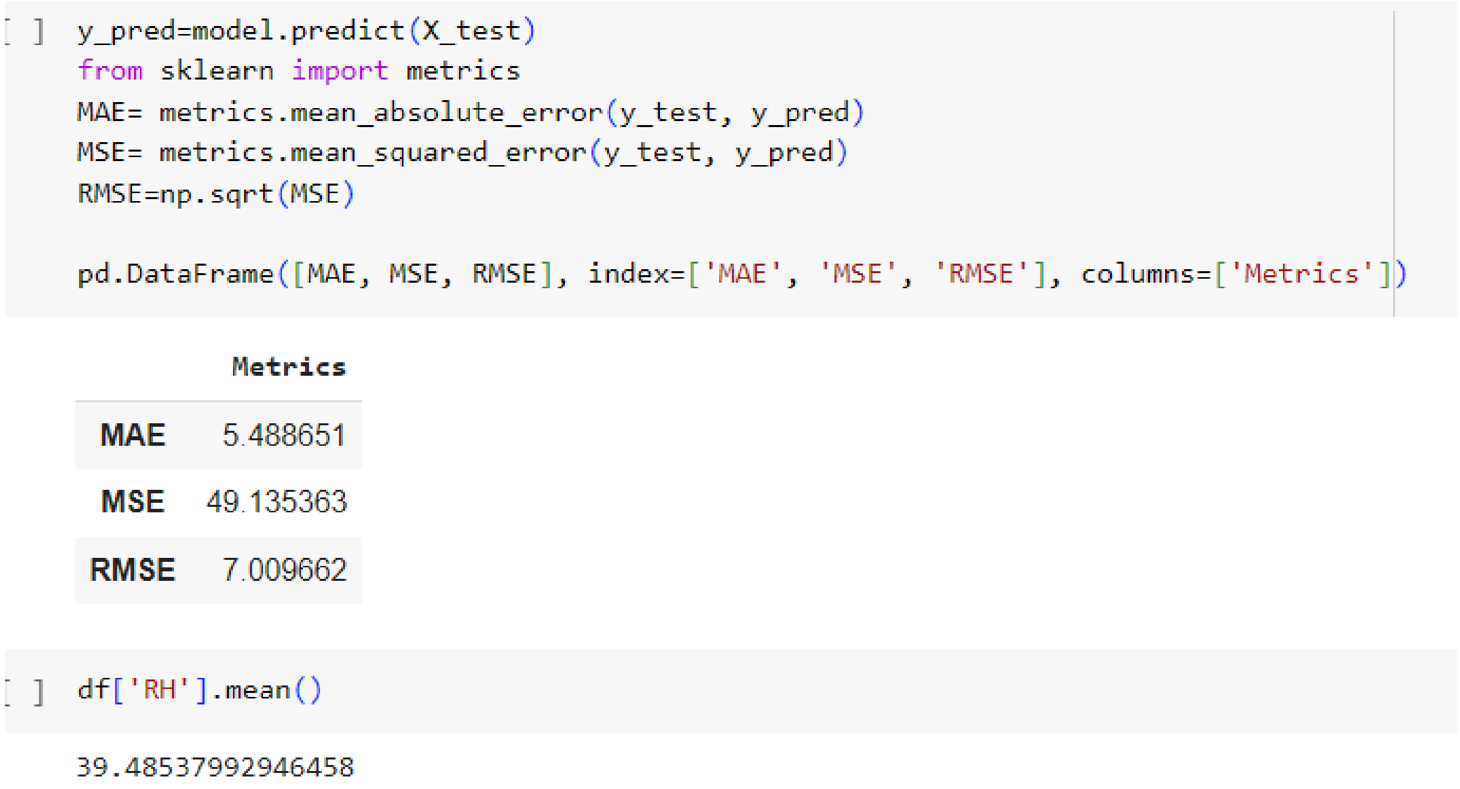
```
[ ]  X = df.drop(['Date','Time','RH'],axis=1)
     y = df['RH']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
[ ]  from sklearn.linear_model import LinearRegression
     model= LinearRegression()
     model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

MAE,MSE,RMSE,MEAN



```python
y_pred=model.predict(X_test)
from sklearn import metrics
MAE= metrics.mean_absolute_error(y_test, y_pred)
MSE= metrics.mean_squared_error(y_test, y_pred)
RMSE=np.sqrt(MSE)

pd.DataFrame([MAE, MSE, RMSE], index=['MAE', 'MSE', 'RMSE'], columns=['Metrics'])
```

```
       Metrics
MAE   5.488651
MSE  49.135363
RMSE  7.009662
```

```python
df['RH'].mean()
```

```
39.48537992946458
```

FILS FOR YOUR REFERENCE

Jupyter Source
File

Microsoft Excel
ma Separated Valu

THANK YOU