import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Import dataset

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Desktop\vicky\C10_air\csvs_per_year\csvs_per_year\madrid_2015.csv")[0:500
```

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     500 non-null    object 
 1   BEN      126 non-null    float64
 2   CO       208 non-null    float64
 3   EBE      126 non-null    float64
 4   NMHC     63 non-null     float64
 5   NO       491 non-null    float64
 6   NO_2     491 non-null    float64
 7   O_3      288 non-null    float64
 8   PM10     247 non-null    float64
 9   PM25     126 non-null    float64
 10  SO_2     197 non-null    float64
 11  TCH      63 non-null     float64
 12  TOL      123 non-null    float64
 13  station  500 non-null    int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 54.8+ KB
```

In [4]:
```
data.head()
```

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28079004
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28079008
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28079011
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28079017

In [5]:
```
data.shape
```

Out[5]:
```
(500, 14)
```

In [6]:
```
data.index
```

Out[6]:
```
RangeIndex(start=0, stop=500, step=1)
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [8]:
```
data.isna()
```

Out[8]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	False	True	False	True	True	False	False	True	True	True	False	True	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	True	False	True	False	False	True	True	True	True	True	False	False
3	False	True	False	True	True	False	False	False	True	True	True	True	True	False
4	False	True	True	True	True	False	False	False	True	True	False	True	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	False	True	True	True	True	False	False	False	True	True	True	True	True	False
496	False	True	True	True	True	False	False	True	False	False	True	True	True	False
497	False	True	True	True	True	False	False	False	True	True	True	True	True	False
498	False	False	True	False	False	False	False	True	False	True	True	False	False	False
499	False	True	False	True	True	False	False	False	True	True	True	True	True	False

500 rows × 14 columns

In [9]:
```
data.fillna(value=0)
```

Out[9]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2015-10-01 01:00:00	0.0	0.8	0.0	0.00	90.0	82.0	0.0	0.0	0.0	10.0	0.00	0.0	28079004
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28079008
2	2015-10-01 01:00:00	3.1	0.0	1.8	0.00	29.0	97.0	0.0	0.0	0.0	0.0	0.00	7.1	28079011
3	2015-10-01 01:00:00	0.0	0.6	0.0	0.00	30.0	103.0	2.0	0.0	0.0	0.0	0.00	0.0	28079016
4	2015-10-01 01:00:00	0.0	0.0	0.0	0.00	95.0	96.0	2.0	0.0	0.0	9.0	0.00	0.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	2015-10-01 21:00:00	0.0	0.0	0.0	0.00	1.0	44.0	56.0	0.0	0.0	0.0	0.00	0.0	28079049
496	2015-10-01 21:00:00	0.0	0.0	0.0	0.00	52.0	82.0	0.0	21.0	10.0	0.0	0.00	0.0	28079050
497	2015-10-01 21:00:00	0.0	0.0	0.0	0.00	1.0	51.0	63.0	0.0	0.0	0.0	0.00	0.0	28079054
498	2015-10-01 21:00:00	0.1	0.0	0.1	0.29	2.0	35.0	0.0	21.0	0.0	0.0	1.62	0.8	28079055
499	2015-10-01 21:00:00	0.0	0.2	0.0	0.00	8.0	43.0	68.0	0.0	0.0	0.0	0.00	0.0	28079056

500 rows × 14 columns

In [10]:
```
data.isna
```

Out[10]:
```
<bound method DataFrame.isna of                    date  BEN   CO  EBE  NMHC    NO   NO_2   O_3  PM10
PM25  \
0    2015-10-01 01:00:00  NaN  0.8  NaN   NaN  90.0   82.0   NaN   NaN   NaN
1    2015-10-01 01:00:00  2.0  0.8  1.6  0.33  40.0   95.0   4.0  37.0  24.0
2    2015-10-01 01:00:00  3.1  NaN  1.8   NaN  29.0   97.0   NaN   NaN   NaN
3    2015-10-01 01:00:00  NaN  0.6  NaN   NaN  30.0  103.0   2.0   NaN   NaN
4    2015-10-01 01:00:00  NaN  NaN  NaN   NaN  95.0   96.0   2.0   NaN   NaN
..                   ...  ...  ...  ...   ...   ...    ...   ...   ...   ...
495  2015-10-01 21:00:00  NaN  NaN  NaN   NaN   1.0   44.0  56.0   NaN   NaN
496  2015-10-01 21:00:00  NaN  NaN  NaN   NaN  52.0   82.0   NaN  21.0  10.0
497  2015-10-01 21:00:00  NaN  NaN  NaN   NaN   1.0   51.0  63.0   NaN   NaN
498  2015-10-01 21:00:00  0.1  NaN  0.1  0.29   2.0   35.0   NaN  21.0   NaN
499  2015-10-01 21:00:00  NaN  0.2  NaN   NaN   8.0   43.0  68.0   NaN   NaN

     SO_2   TCH  TOL   station
0    10.0   NaN  NaN  28079004
1    12.0  1.83  8.3  28079008
2     NaN   NaN  7.1  28079011
3     NaN   NaN  NaN  28079016
4     9.0   NaN  NaN  28079017
..    ...   ...  ...       ...
495   NaN   NaN  NaN  28079049
496   NaN   NaN  NaN  28079050
497   NaN   NaN  NaN  28079054
498   NaN  1.62  0.8  28079055
499   NaN   NaN  NaN  28079056

[500 rows x 14 columns]>
```

Plotting using various method

In [11]:
```
data.plot.line()
```

Out[11]:
```
<AxesSubplot:>
```

In [12]:
```
data.plot.bar()
```

Out[12]:
```
<AxesSubplot:>
```

In [13]:
```
data.plot.area()
```

Out[13]:
```
<AxesSubplot:>
```

In [14]:
```
data.plot.hist()
```

Out[14]:
```
<AxesSubplot:ylabel='Frequency'>
```

In [15]:
```
data.plot.pie(y="BEN")
```

Out[15]:
```
<AxesSubplot:ylabel='BEN'>
```

In [16]:
```
data.plot.scatter(x="NO_2",y='O_3')
```

Out[16]:
```
<AxesSubplot:xlabel='NO_2', ylabel='O_3'>
```

seaborn Visualize

In [17]:
```
sns.pairplot(data)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x2c375a71730>
```

In [18]:
```
sns.distplot(data['BEN'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either `displ
ot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogr
ams).
  warnings.warn(msg, FutureWarning)
```

Out[18]: <AxesSubplot:xlabel='BEN', ylabel='Density'>

In [19]:
```
sns.heatmap(data.corr())
```

Out[19]: <AxesSubplot:>

In [20]:
```
data1=data[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2','O_3',
       'PM10', 'SO_2']]
```

In [21]:
```
data2=data1.fillna(value=1)
```

In [22]:
```
x=data2[['CO','CO','O_3']]
y=data['station']
```

Linear Regression

In [23]:
```
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)
```

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]: LinearRegression()

In [25]:
```
print(lr.intercept_)
```

```
28079023.793197438
```

In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['PM10'])
coeff
```

Out[26]:

	PM10
CO	8.467415
CO	8.467415
O_3	0.070230

In [27]:
```
prediction1=lr.predict(x_train)
plt.scatter(y_train,prediction1)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x2c37f8ce6a0>
```

In [28]:
```
lr.score(x_test,y_test)
```

Out[28]:
```
0.1455715730681788
```

In [29]:
```
prediction1=lr.predict(x_test)
```

Ridge

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[30]:
```
Ridge(alpha=10)
```

In [31]:
```
rr.score(x_test,y_test)
```

Out[31]:
```
0.1386871748922941
```

In [32]:
```
prediction2=rr.predict(x_test)
```

Lasso

In [33]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[33]:
```
Lasso(alpha=10)
```

In [34]:
```
la.score(x_test,y_test)
```

Out[34]:
```
-0.008177553523821546
```

In [35]:
```
prediction3=la.score(x_test,y_test)
```

Elastic Net

In [36]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[36]:
```
ElasticNet()
```

In [37]:
```
print(en.coef_)
```

```
[1.87242781 1.87242726 0.05591794]
```

In [38]:
```
print(en.intercept_)
```

```
28079034.03224754
```

In [39]:
```
prediction4=en.predict(x_test)
```

In [40]:
```
en.score(x_test,y_test)
```

Out[40]:
```
0.04328535014316459
```

Evalution Metrics for linear

In [41]:
```
from sklearn import metrics
```

In [42]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction1))
```

```
Mean Absolute error: 13.801668424531817
```

In [43]:
```
print("Mean Absolute square error:",metrics.mean_squared_error(y_test,prediction1))
```

```
Mean Absolute square error: 274.774916299666
```

Evalution Metrics for Ridge

In [44]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction2))
```

```
Mean Absolute error: 13.943412126551072
```

In [45]:
```
print("Mean Absolute square error:",metrics.mean_squared_error(y_test,prediction2))
```

```
Mean Absolute square error: 276.98886409555695
```

Evalution for elasticnet

In [46]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction4))
```

```
Mean Absolute error: 15.206427322427432
```

In [47]:
```
print("Mean Absolute square error:",metrics.mean_squared_error(y_test,prediction4))
```

```
Mean Absolute square error: 307.6690563550886
```

Feature matrix

In [48]:
```
from sklearn.preprocessing import StandardScaler
```

In [49]:
```
from sklearn import utils
```

In [50]:
```
from sklearn.linear_model import LogisticRegression
```

In [51]:
```
df=pd.read_csv(r"C:\Users\user\Desktop\vicky\C10_air\csvs_per_year\csvs_per_year\madrid_2015.csv")
```

In [52]:
```
df.columns
```

Out[52]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [53]:
```
new_df=df.fillna({'BEN':1,'CO':2,'EBE':4})
new_df
```

Out[53]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2015-10-01 01:00:00	1.0	0.8	4.0	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28079004
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28079008
2	2015-10-01 01:00:00	3.1	2.0	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28079011
3	2015-10-01 01:00:00	1.0	0.6	4.0	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2015-10-01 01:00:00	1.0	2.0	4.0	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210091	2015-08-01 00:00:00	1.0	0.2	4.0	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28079056
210092	2015-08-01 00:00:00	1.0	0.2	4.0	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28079057
210093	2015-08-01 00:00:00	1.0	2.0	4.0	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28079058
210094	2015-08-01 00:00:00	1.0	2.0	4.0	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28079059
210095	2015-08-01 00:00:00	1.0	2.0	4.0	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28079060

210096 rows × 14 columns

In [54]:
```
feature_matrix = new_df[['CO','EBE']]
target_vector = new_df['station']
```

In [55]:
```
feature_matrix.shape
```

Out[55]:
```
(210096, 2)
```

In [56]:
```
target_vector.shape
```

Out[56]:
```
(210096,)
```

In [57]:
```
from sklearn.preprocessing import StandardScaler
```

In [58]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [59]:
```
logr=LogisticRegression()
```

In [60]:
```
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lb
fgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/
preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.
org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[60]:
```
LogisticRegression()
```

In [61]:
```
observation =[[3,90]]
```

In [62]:
```
prediction5 =logr.predict(observation)
print(prediction5)
```

```
[28079016]
```

In [63]:
```
logr.predict_proba(observation)[0][0]
```

Out[63]:
```
7.842232395141757e-05
```

In [64]:
```
logr.predict_proba(observation)[0][1]
```

Out[64]:
```
8.968843640373328e-172
```

import pickle

In [65]:
```
import pickle
```

In [66]:
```
filename1="prediction1"
```

In [67]:
```
filename2="prediction2"
```

In [68]:
```
filename3="prediction3"
```

In [69]:
```
filename4="prediction4"
```

In [70]:
```
filename5="prediction5"
```

In [71]:
```
pickle.dump(lr,open(filename1,'wb'))
```

In [72]:
```
pickle.dump(lr,open(filename2,'wb'))
```

In [73]:
```
pickle.dump(lr,open(filename3,'wb'))
```

In [74]:
```
pickle.dump(lr,open(filename4,'wb'))
```

In [75]:
```
pickle.dump(lr,open(filename5,'wb'))
```

In [ ]: