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_200
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     500 non-null    object 
 1   BEN      147 non-null    float64
 2   CO       464 non-null    float64
 3   EBE      130 non-null    float64
 4   MXY      92 non-null     float64
 5   NMHC     241 non-null    float64
 6   NO_2     500 non-null    float64
 7   NOx      500 non-null    float64
 8   OXY      92 non-null     float64
 9   O_3      479 non-null    float64
 10  PM10     482 non-null    float64
 11  PM25     129 non-null    float64
 12  PXY      92 non-null     float64
 13  SO_2     500 non-null    float64
 14  TCH      241 non-null    float64
 15  TOL      147 non-null    float64
 16  station  500 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 66.5+ KB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.860001	NaN	12
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	NaN	3.38	6
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	NaN	NaN	8
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	NaN	NaN	8
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	NaN	NaN	8

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

Out[5]:
```
(500, 17)
```

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

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

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

Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

Out[8]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	TOL	s
0	False	True	False	True	True	True	False	False	True	False	False	False	True	False	True	True	
1	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False	
2	False	True	False	True	True	True	False	False	True	False	False	True	True	False	True	True	
3	False	True	False	True	True	True	False	False	True	False	False	True	True	False	True	True	
4	False	True	False	True	True	True	False	False	True	False	False	True	True	False	True	True	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	False	True	False	True	True	False	False	False	True	False	False	True	True	False	False	True	
496	False	True	False	True	True	True	False	False	True	False	False	True	True	False	True	True	
497	False	True	False	True	True	False	False	False	True	False	False	True	True	False	False	True	
498	False	True	False	True	True	True	False	False	True	False	False	True	True	False	True	True	
499	False	False	False	False	True	False	False	False	True	False	False	False	True	False	False	False	

500 rows × 17 columns

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

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
0	2004-08-01 01:00:00	0.00	0.66	0.00	0.00	0.00	89.550003	118.900002	0.00	40.020000	39.990002	25.860001	0.00
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	0.000000	3.38
2	2004-08-01 01:00:00	0.00	1.02	0.00	0.00	0.00	93.389999	138.600006	0.00	20.860001	49.480000	0.000000	0.00
3	2004-08-01 01:00:00	0.00	0.53	0.00	0.00	0.00	87.290001	105.000000	0.00	36.730000	31.070000	0.000000	0.00
4	2004-08-01 01:00:00	0.00	0.17	0.00	0.00	0.00	34.910000	35.349998	0.00	86.269997	54.080002	0.000000	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	2004-08-01 19:00:00	0.00	0.25	0.00	0.00	0.26	32.529999	46.299999	0.00	103.099998	20.100000	0.000000	0.00
496	2004-08-01 19:00:00	0.00	1.15	0.00	0.00	0.00	65.250000	92.500000	0.00	71.769997	41.130001	0.000000	0.00
497	2004-08-01 19:00:00	0.00	0.25	0.00	0.00	0.82	13.230000	14.670000	0.00	113.800003	34.619999	0.000000	0.00
498	2004-08-01 19:00:00	0.00	0.26	0.00	0.00	0.00	24.690001	29.760000	0.00	97.300003	50.400002	0.000000	0.00
499	2004-08-01 19:00:00	2.94	0.34	1.01	0.00	0.04	47.040001	64.209999	0.00	95.959999	25.100000	12.970000	0.00

500 rows × 17 columns

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

Out[10]:
```
<bound method DataFrame.isna of                     date   BEN    CO   EBE   MXY  NMHC
NO_2         NOx  \
0    2004-08-01 01:00:00   NaN  0.66   NaN   NaN   NaN  89.550003  118.900002
1    2004-08-01 01:00:00  2.66  0.54  2.99  6.08  0.18  51.799999   53.860001
2    2004-08-01 01:00:00   NaN  1.02   NaN   NaN   NaN  93.389999  138.600006
3    2004-08-01 01:00:00   NaN  0.53   NaN   NaN   NaN  87.290001  105.000000
4    2004-08-01 01:00:00   NaN  0.17   NaN   NaN   NaN  34.910000   35.349998
..                   ...   ...   ...   ...   ...   ...        ...         ...
495  2004-08-01 19:00:00   NaN  0.25   NaN   NaN  0.26  32.529999   46.299999
496  2004-08-01 19:00:00   NaN  1.15   NaN   NaN   NaN  65.250000   92.500000
497  2004-08-01 19:00:00   NaN  0.25   NaN   NaN  0.82  13.230000   14.670000
498  2004-08-01 19:00:00   NaN  0.26   NaN   NaN   NaN  24.690001   29.760000
499  2004-08-01 19:00:00  2.94  0.34  1.01   NaN  0.04  47.040001   64.209999

      OXY         O_3       PM10       PM25   PXY   SO_2   TCH    TOL  \
0     NaN   40.020000  39.990002  25.860001   NaN  12.20   NaN    NaN
1    3.28   51.689999  22.950001        NaN  3.38   6.12  1.57  11.37
2     NaN   20.860001  49.480000        NaN   NaN   8.99   NaN    NaN
3     NaN   36.730000  31.070000        NaN   NaN   8.82   NaN    NaN
4     NaN   86.269997  54.080002        NaN   NaN   8.71   NaN    NaN
..    ...         ...        ...        ...   ...    ...   ...    ...
495   NaN  103.099998  20.100000        NaN   NaN   7.98  1.34    NaN
496   NaN   71.769997  41.130001        NaN   NaN   9.93   NaN    NaN
497   NaN  113.800003  34.619999        NaN   NaN   4.78  1.46    NaN
498   NaN   97.300003  50.400002        NaN   NaN   7.67   NaN    NaN
499   NaN   95.959999  25.100000  12.970000   NaN   6.91  1.31   2.46

      station
0    28079001
1    28079035
2    28079003
3    28079004
4    28079039
..        ...
495  28079011
496  28079012
497  28079040
498  28079014
499  28079015

[500 rows x 17 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")
```

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 0x23f3760c4f0>
```

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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `di
stplot` is a deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility) or `histplot
` (an axes-level function for histograms).
  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', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2']]
```

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

In [22]:
```
x=data2[['CO','CO','NOx','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_)
```

```
28079033.260229
```

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

Out[26]:

	PM10
CO	-0.189598
CO	-0.189598
NOx	-0.153066
O_3	-0.046669

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x23f448ea340>
```

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

Out[28]:
```
0.047882027807252814
```

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]:
```
0.04789416263573887
```

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]:
```
0.04848784477294399
```

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_)
```

```
[-0.         -0.         -0.15335091 -0.04529529]
```

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

```
28079033.03659595
```

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

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

Out[40]:
```
0.04799813140185971
```

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: 10.838858732879162
```

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

```
Mean Absolute square error: 300.8227735395107
```

Evalution Metrics for Ridge

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

```
Mean Absolute error: 10.836204633141557
```

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

```
Mean Absolute square error: 300.8189395264281
```

Evalution for elasticnet

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

```
Mean Absolute error: 10.825047250986099
```

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

```
Mean Absolute square error: 300.7860904746305
```

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_2004.
```

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

Out[52]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [53]:
```
new_df=df.fillna({'BEN':1,'CO':2,'EBE':4,'MXY':5})
new_df
```

Out[53]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PX
0	2004-08-01 01:00:00	1.00	0.66	4.00	5.00	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.860001	Na
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	NaN	3.
2	2004-08-01 01:00:00	1.00	1.02	4.00	5.00	NaN	93.389999	138.600006	NaN	20.860001	49.480000	NaN	Na
3	2004-08-01 01:00:00	1.00	0.53	4.00	5.00	NaN	87.290001	105.000000	NaN	36.730000	31.070000	NaN	Na
4	2004-08-01 01:00:00	1.00	0.17	4.00	5.00	NaN	34.910000	35.349998	NaN	86.269997	54.080002	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.860000	0.
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	NaN	2.
245493	2004-06-01 00:00:00	1.00	2.00	4.00	5.00	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.040000	Na
245494	2004-06-01 00:00:00	1.00	2.00	4.00	5.00	0.09	82.599998	102.599998	NaN	NaN	45.630001	NaN	Na
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.959999	2.

245496 rows × 17 columns

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

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

Out[55]: (245496, 3)

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

Out[56]: (245496,)

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)
```

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

In [61]:
```
observation =[[3,90,5]]
```

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

```
[28079023]
```

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

Out[63]:
```
8.437662868313907e-28
```

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

Out[64]:
```
1.108346610295162e-21
```

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 [ ]:
```

```