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 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     500 non-null    object 
 1   BEN      158 non-null    float64
 2   CO       499 non-null    float64
 3   EBE      139 non-null    float64
 4   MXY      100 non-null    float64
 5   NMHC     184 non-null    float64
 6   NO_2     495 non-null    float64
 7   NOx      495 non-null    float64
 8   OXY      100 non-null    float64
 9   O_3      499 non-null    float64
 10  PM10     475 non-null    float64
 11  PXY      100 non-null    float64
 12  SO_2     498 non-null    float64
 13  TCH      184 non-null    float64
 14  TOL      158 non-null    float64
 15  station  500 non-null    int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 62.6+ KB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.32	NaN	NaN
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.2	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.66	1.82	10.9
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.67	NaN	NaN
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.99	NaN	NaN
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.26	NaN	NaN

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

Out[5]:
```
(500, 16)
```

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', '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	PXY	SO_2	TCH	TOL	station
0	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
3	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
4	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	False	True	False	True	True	True	False	False	True	False	False	True	False	True	False	False
496	False	False	False	False	True	False	False	False	True	False	False	True	False	False	False	False
497	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
498	False	False	False	False	False	True	False	False	False	False	True	False	False	True	False	False
499	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

500 rows × 16 columns

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

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH
0	2002-04-01 01:00:00	0.00	1.39	0.00	0.00	0.00	145.100006	352.100006	0.00	6.540000	41.990002	0.00	21.32	0.0(
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000	20.980000	2.53	11.66	1.8(
2	2002-04-01 01:00:00	0.00	0.80	0.00	0.00	0.00	103.699997	134.000000	0.00	13.010000	28.440001	0.00	13.67	0.0(
3	2002-04-01 01:00:00	0.00	1.61	0.00	0.00	0.00	97.599998	268.000000	0.00	5.120000	42.180000	0.00	16.99	0.0(
4	2002-04-01 01:00:00	0.00	1.90	0.00	0.00	0.00	92.089996	237.199997	0.00	7.280000	76.330002	0.00	15.26	0.0(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
495	2002-04-01 20:00:00	0.00	0.50	0.00	0.00	0.00	78.959999	103.900002	0.00	46.090000	15.040000	0.00	9.10	0.0(
496	2002-04-01 20:00:00	0.57	0.51	0.91	0.00	0.15	58.060001	72.809998	0.00	52.220001	8.150000	0.00	7.01	1.3(
497	2002-04-01 20:00:00	0.51	0.51	0.39	0.70	0.10	25.420000	26.780001	0.59	68.930000	22.129999	0.65	1.75	1.3(
498	2002-04-01 20:00:00	1.37	0.61	0.85	1.85	0.00	102.800003	198.100006	0.76	22.490000	0.000000	0.78	20.15	0.0(
499	2002-04-01 20:00:00	2.22	0.67	2.02	4.93	0.17	67.699997	97.940002	2.18	55.910000	21.180000	1.96	10.71	1.4(

500 rows × 16 columns

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

Out[10]:
```
<bound method DataFrame.isna of                     date   BEN    CO   EBE   MXY  NMHC
NO_2  \
0    2002-04-01 01:00:00   NaN  1.39   NaN   NaN   NaN  145.100006
1    2002-04-01 01:00:00  1.93  0.71  2.33  6.20  0.15   98.150002
2    2002-04-01 01:00:00   NaN  0.80   NaN   NaN   NaN  103.699997
3    2002-04-01 01:00:00   NaN  1.61   NaN   NaN   NaN   97.599998
4    2002-04-01 01:00:00   NaN  1.90   NaN   NaN   NaN   92.089996
..                   ...   ...   ...   ...   ...   ...         ...
495  2002-04-01 20:00:00   NaN  0.50   NaN   NaN   NaN   78.959999
496  2002-04-01 20:00:00  0.57  0.51  0.91   NaN  0.15   58.060001
497  2002-04-01 20:00:00  0.51  0.51  0.39  0.70  0.10   25.420000
498  2002-04-01 20:00:00  1.37  0.61  0.85  1.85   NaN  102.800003
499  2002-04-01 20:00:00  2.22  0.67  2.02  4.93  0.17   67.699997

            NOx   OXY        O_3       PM10   PXY   SO_2   TCH     TOL  \
0    352.100006   NaN   6.540000  41.990002   NaN  21.32   NaN     NaN
1    153.399994  2.67   6.850000  20.980000  2.53  11.66  1.82   10.98
2    134.000000   NaN  13.010000  28.440001   NaN  13.67   NaN     NaN
3    268.000000   NaN   5.120000  42.180000   NaN  16.99   NaN     NaN
4    237.199997   NaN   7.280000  76.330002   NaN  15.26   NaN     NaN
..          ...   ...        ...        ...   ...    ...   ...     ...
495  103.900002   NaN  46.090000  15.040000   NaN   9.10   NaN    3.86
496   72.809998   NaN  52.220001   8.150000   NaN   7.01  1.33    4.53
497   26.780001  0.59  68.930000  22.129999  0.65   1.75  1.34    1.74
498  198.100006  0.76  22.490000        NaN  0.78  20.15   NaN    3.84
499   97.940002  2.18  55.910000  21.180000  1.96  10.71  1.40    8.52

      station
0    28079001
1    28079035
2    28079003
3    28079004
4    28079039
..        ...
495  28079022
496  28079023
497  28079024
498  28079025
499  28079099

[500 rows x 16 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 0x19d742878b0>
```

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 [24]:
```
data2=data1.fillna(value=1)
```

In [25]:
```
x=data2[['CO','CO','NOx','O_3']]
y=data['station']
```

Linear Regression

In [26]:
```
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 [27]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
print(lr.intercept_)
```

```
28079023.567117084
```

In [29]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['PM10'])
coeff
```

Out[29]:

	PM10
CO	2.146801
CO	2.146801
NOx	-0.037215
O_3	0.014465

In [30]:
```
prediction1=lr.predict(x_train)
plt.scatter(y_train,prediction1)
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x19d001aaeb0>
```

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

Out[31]:
```
0.016185460013266706
```

In [32]:
```
prediction1=lr.predict(x_test)
```

Ridge

In [33]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

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

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

Out[34]:
```
0.015634314132378924
```

In [35]:
```
prediction2=rr.predict(x_test)
```

Lasso

In [36]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[36]:
```
Lasso(alpha=10)
```

In [37]:
```
la.score(x_test,y_test)
```

Out[37]:
```
0.009038190529810475
```

In [38]:
```
prediction3=la.score(x_test,y_test)
```

Elastic Net

In [39]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[39]:
```
ElasticNet()
```

In [40]:
```
print(en.coef_)
```

```
[ 0.          0.         -0.01708491  0.01853226]
```

In [41]:
```
print(en.intercept_)
```

```
28079023.77456478
```

In [42]:
```
prediction4=en.predict(x_test)
```

In [43]:
```
en.score(x_test,y_test)
```

Out[43]:
```
0.008414038578976268
```

Evalution Metrics for linear

In [44]:
```
from sklearn import metrics
```

In [45]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction1))
```

```
Mean Absolute error: 12.245161265358329
```

In [46]:
```
print("Mean Absolute square error:",metrics.mean_squared_error(y_test,prediction1))
```

```
Mean Absolute square error: 328.0863643278451
```

Evalution Metrics for Ridge

In [47]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction2))
```

```
Mean Absolute error: 12.245569568872451
```

In [48]:
```
print("Mean Absolute square error:",metrics.mean_squared_error(y_test,prediction2))
```

```
Mean Absolute square error: 328.27016263629173
```

Evalution for elasticnet

In [49]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction4))
```

```
Mean Absolute error: 12.257236495564381
```

In [50]:
```
print("Mean Absolute square error:",metrics.mean_squared_error(y_test,prediction4))
```

```
Mean Absolute square error: 330.67800868804153
```

Feature matrix

In [51]:
```
from sklearn.preprocessing import StandardScaler
```

In [52]:
```
from sklearn import utils
```

In [53]:
```
from sklearn.linear_model import LogisticRegression
```

In [54]:
```
df=pd.read_csv(r"C:\Users\user\Desktop\vicky\C10_air\csvs_per_year\csvs_per_year\madrid_2002.
```

In [55]:
```
df.columns
```

Out[55]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [56]:
```
new_df=df.fillna({'BEN':1,'CO':2,'EBE':4,'MXY':5})
new_df
```

Out[56]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2002-04-01 01:00:00	1.00	1.39	4.00	5.00	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.320000
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.660000
2	2002-04-01 01:00:00	1.00	0.80	4.00	5.00	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.670000
3	2002-04-01 01:00:00	1.00	1.61	4.00	5.00	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.990000
4	2002-04-01 01:00:00	1.00	1.90	4.00	5.00	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.260000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	4.00	5.00	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.210000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	5.00	0.38	113.900002	373.100006	NaN	5.66	63.389999	NaN	15.640000
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.620000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN	5.52	24.219999
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.910000

217296 rows × 16 columns

In [57]:
```
feature_matrix = new_df[['CO','EBE','MXY']]
target_vector = new_df['station']
```

In [58]:
```
feature_matrix.shape
```

Out[58]: (217296, 3)

In [59]:
```
target_vector.shape
```

Out[59]: (217296,)

In [60]:
```
from sklearn.preprocessing import StandardScaler
```

In [61]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [62]:
```
logr=LogisticRegression()
```

In [63]:
```
logr.fit(fs,target_vector)
```

Out[63]:
```
LogisticRegression()
```

In [64]:
```
observation =[[3,90,5]]
```

In [65]:
```
prediction5 =logr.predict(observation)
print(prediction5)
```

```
[28079021]
```

In [66]:
```
logr.predict_proba(observation)[0][0]
```

Out[66]:
```
6.699436900384852e-11
```

In [67]:
```
logr.predict_proba(observation)[0][1]
```

Out[67]:
```
0.00032566222001853063
```

import pickle

In [68]:
```
import pickle
```

In [69]:
```
filename1="prediction1"
```

In [70]:
```
filename2="prediction2"
```

In [71]:
```
filename3="prediction3"
```

In [72]:
```
filename4="prediction4"
```

In [73]:
```
filename5="prediction5"
```

In [74]:
```
pickle.dump(lr,open(filename1,'wb'))
```

In [75]:
```
pickle.dump(lr,open(filename2,'wb'))
```

In [76]:
```
pickle.dump(lr,open(filename3,'wb'))
```

In [77]:
```
pickle.dump(lr,open(filename4,'wb'))
```

In [78]:
```
pickle.dump(lr,open(filename5,'wb'))
```

In [ ]:
```

```