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_2001.csv
```

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      143 non-null    float64
 2   CO       500 non-null    float64
 3   EBE      123 non-null    float64
 4   MXY      102 non-null    float64
 5   NMHC     217 non-null    float64
 6   NO_2     500 non-null    float64
 7   NOx      500 non-null    float64
 8   OXY      102 non-null    float64
 9   O_3      500 non-null    float64
 10  PM10     480 non-null    float64
 11  PXY      102 non-null    float64
 12  SO_2     489 non-null    float64
 13  TCH      217 non-null    float64
 14  TOL      140 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	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.34	NaN	NaN
1	2001-08-01 01:00:00	1.5	0.34	1.49	4.1	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11	1.24	10.82
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.85	NaN	NaN
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.46	NaN	NaN
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.80	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	False	False	False	True	False	False	True	False	False	True	False
496	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
497	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
498	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
499	False	False	False	True	True	True	False	False	True	False	False	True	False	True	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	T(
0	2001-08-01 01:00:00	0.0	0.37	0.00	0.0	0.00	58.400002	87.150002	0.00	34.529999	105.000000	0.00	6.34	0.00	0.
1	2001-08-01 01:00:00	1.5	0.34	1.49	4.1	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11	1.24	10.
2	2001-08-01 01:00:00	0.0	0.28	0.00	0.0	0.00	50.660000	61.380001	0.00	46.310001	100.099998	0.00	7.85	0.00	0.
3	2001-08-01 01:00:00	0.0	0.47	0.00	0.0	0.00	69.790001	73.449997	0.00	40.650002	69.779999	0.00	6.46	0.00	0.
4	2001-08-01 01:00:00	0.0	0.39	0.00	0.0	0.00	22.830000	24.799999	0.00	66.309998	75.180000	0.00	8.80	0.00	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
495	2001-08-01 21:00:00	0.0	0.42	0.00	0.0	0.15	28.219999	30.760000	0.00	76.300003	9.920000	0.00	7.58	1.36	0.
496	2001-08-01 21:00:00	0.0	0.41	0.00	0.0	0.00	77.129997	81.870003	0.00	65.970001	7.350000	0.00	3.52	0.00	0.
497	2001-08-01 21:00:00	0.0	0.40	0.00	0.0	0.00	60.049999	67.019997	0.00	75.589996	5.000000	0.00	10.04	0.00	0.
498	2001-08-01 21:00:00	0.0	0.71	0.00	0.0	0.00	74.029999	105.300003	0.00	63.349998	4.370000	0.00	12.66	0.00	0.
499	2001-08-01 21:00:00	0.6	0.97	0.00	0.0	0.00	46.639999	58.840000	0.00	85.760002	5.000000	0.00	8.83	0.00	5.

500 rows × 16 columns

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

Out[10]:
```
<bound method DataFrame.isna of                  date  BEN    CO   EBE  MXY  NMHC       NO_2
NOx  \
0    2001-08-01 01:00:00  NaN  0.37   NaN  NaN   NaN  58.400002   87.150002
1    2001-08-01 01:00:00  1.5  0.34  1.49  4.1  0.07  56.250000   75.169998
2    2001-08-01 01:00:00  NaN  0.28   NaN  NaN   NaN  50.660000   61.380001
3    2001-08-01 01:00:00  NaN  0.47   NaN  NaN   NaN  69.790001   73.449997
4    2001-08-01 01:00:00  NaN  0.39   NaN  NaN   NaN  22.830000   24.799999
..                   ...  ...   ...   ...  ...   ...        ...         ...
495  2001-08-01 21:00:00  NaN  0.42   NaN  NaN  0.15  28.219999   30.760000
496  2001-08-01 21:00:00  NaN  0.41   NaN  NaN   NaN  77.129997   81.870003
497  2001-08-01 21:00:00  NaN  0.40   NaN  NaN   NaN  60.049999   67.019997
498  2001-08-01 21:00:00  NaN  0.71   NaN  NaN   NaN  74.029999  105.300003
499  2001-08-01 21:00:00  0.6  0.97   NaN  NaN   NaN  46.639999   58.840000

      OXY        O_3        PM10   PXY   SO_2   TCH    TOL   station
0     NaN  34.529999  105.000000   NaN   6.34   NaN    NaN  28079001
1    2.11  42.160000  100.599998  1.73   8.11  1.24  10.82  28079035
2     NaN  46.310001  100.099998   NaN   7.85   NaN    NaN  28079003
3     NaN  40.650002   69.779999   NaN   6.46   NaN    NaN  28079004
4     NaN  66.309998   75.180000   NaN   8.80   NaN    NaN  28079039
..    ...        ...         ...   ...    ...   ...    ...       ...
495   NaN  76.300003    9.920000   NaN   7.58  1.36    NaN  28079018
496   NaN  65.970001    7.350000   NaN   3.52   NaN    NaN  28079019
497   NaN  75.589996    5.000000   NaN  10.04   NaN    NaN  28079036
498   NaN  63.349998    4.370000   NaN  12.66   NaN    NaN  28079021
499   NaN  85.760002    5.000000   NaN   8.83   NaN   5.93  28079022

[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 0x22f0154cd60>
```

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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distpl
ot` 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-l
evel 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]:
```
x=data[['CO','CO','NOx','O_3']]
y=data['station']
```

Linear Regression

In [22]:
```
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 [23]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[23]:
```
LinearRegression()
```

In [24]:
```
print(lr.intercept_)
```

```
28079021.463279463
```

In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['PM10'])
coeff
```

Out[25]:

	PM10
CO	0.035089
CO	0.035089
NOx	-0.029160
O_3	0.057595

In [26]:
```
prediction1=lr.predict(x_train)
plt.scatter(y_train,prediction1)
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x22f0d27d400>
```

In [27]:
```
lr.score(x_test,y_test)
```

Out[27]:
```
-0.027222798650798685
```

In [28]:
```
prediction1=lr.predict(x_test)
```

Ridge

In [29]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[29]:
```
Ridge(alpha=10)
```

In [30]:
```
rr.score(x_test,y_test)
```

Out[30]:
```
-0.027240739871605024
```

In [31]:
```
prediction2=rr.predict(x_test)
```

Lasso

In [32]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[32]:
```
Lasso(alpha=10)
```

In [33]:
```
la.score(x_test,y_test)
```

Out[33]:
```
-0.02387722514894941
```

In [34]:
```
prediction3=la.score(x_test,y_test)
```

Elastic Net

In [35]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[35]:
```
ElasticNet()
```

In [36]:
```
print(en.coef_)
```

```
[ 0.          0.         -0.02905885  0.05639175]
```

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

```
28079021.554001026
```

In [38]:
```
prediction4=en.predict(x_test)
```

In [39]:
```
en.score(x_test,y_test)
```

Out[39]:
```
-0.027099379607107288
```

Evalution Metrics for linear

In [40]:
```
from sklearn import metrics
```

In [41]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction1))
```

```
Mean Absolute error: 13.845179902762174
```

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

```
Mean Absolute square error: 460.29355975164265
```

Evalution Metrics for Ridge

In [43]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction2))
```

```
Mean Absolute error: 13.845265374456842
```

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

```
Mean Absolute square error: 460.30159912577074
```

Evalution for elasticnet

In [45]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction4))
```

```
Mean Absolute error: 13.84042699366808
```

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

```
Mean Absolute square error: 460.2382562760612
```

Feature matrix

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
from sklearn import utils
```

In [51]:
```
from sklearn.linear_model import LogisticRegression
```

In [52]:
```
df=pd.read_csv(r"C:\Users\user\Desktop\vicky\C10_air\csvs_per_year\csvs_per_year\madrid_2001.csv"
```

In [53]:
```
df.columns
```

Out[53]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [54]:
```
new_df=df.fillna({'BEN':1,'CO':2,'EBE':4,'MXY':5})
new_df
```

Out[54]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2001-08-01 01:00:00	1.00	0.37	4.00	5.00	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.340000
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.110000
2	2001-08-01 01:00:00	1.00	0.28	4.00	5.00	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.850000
3	2001-08-01 01:00:00	1.00	0.47	4.00	5.00	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.460000
4	2001-08-01 01:00:00	1.00	0.39	4.00	5.00	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.800000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
217867	2001-04-01 00:00:00	10.45	1.81	4.00	5.00	NaN	73.000000	264.399994	NaN	5.200000	47.880001	NaN	39.910000
217868	2001-04-01 00:00:00	5.20	0.69	4.56	5.00	0.13	71.080002	129.300003	NaN	13.460000	26.809999	NaN	13.450000
217869	2001-04-01 00:00:00	0.49	1.09	4.00	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14.700000
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39.919998
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95	27.340000

217872 rows × 16 columns

In [55]:
```
feature_matrix = new_df[['CO','EBE','MXY']]
target_vector = new_df['station']
```

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

Out[56]: (217872, 3)

In [57]:
```
target_vector.shape
```

Out[57]: (217872,)

In [58]:
```
from sklearn.preprocessing import StandardScaler
```

In [59]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [60]:
```
logr=LogisticRegression()
```

In [61]:
```
logr.fit(fs,target_vector)
```

Out[61]:
```
LogisticRegression()
```

In [62]:
```
observation =[[3,90,5]]
```

In [63]:
```
prediction5 =logr.predict(observation)
print(prediction5)
```

```
[28079040]
```

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

Out[64]:
```
1.2158082226832085e-07
```

In [65]:
```
logr.predict_proba(observation)[0][1]
```

Out[65]:
```
0.08827983300058923
```

import pickle

In [66]:
```
import pickle
```

In [67]:
```
filename1="prediction1"
```

In [68]:
```
filename2="prediction2"
```

In [69]:
```
filename3="prediction3"
```

In [70]:
```
filename4="prediction4"
```

In [71]:
```
filename5="prediction5"
```

In [72]:
```
pickle.dump(lr,open(filename1,'wb'))
```

In [73]:
```
pickle.dump(lr,open(filename2,'wb'))
```

In [74]:
```
pickle.dump(lr,open(filename3,'wb'))
```

In [75]:
```
pickle.dump(lr,open(filename4,'wb'))
```

In [76]:
```
pickle.dump(lr,open(filename5,'wb'))
```