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      128 non-null    float64
 2   CO       455 non-null    float64
 3   EBE      106 non-null    float64
 4   MXY      72 non-null     float64
 5   NMHC     231 non-null    float64
 6   NO_2     489 non-null    float64
 7   NOx      489 non-null    float64
 8   OXY      78 non-null     float64
 9   O_3      471 non-null    float64
 10  PM10     489 non-null    float64
 11  PXY      71 non-null     float64
 12  SO_2     489 non-null    float64
 13  TCH      231 non-null    float64
 14  TOL      125 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	TC
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	24.299999	Na
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	14.230000	1.5
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	17.879999	Na
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	24.900000	Na
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	18.750000	Na

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	True	False	True	True	False	False	False	False	False	False	True	False	False	True	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	True	False
496	False	True	False	True	True	True	False	False	True	False	False	True	False	True	True	False
497	False	False	False	True	True	True	False	False	True	False	False	True	False	True	False	False
498	False	False	False	False	True	False	False	False	True	False	False	True	False	False	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
0	2003-03-01 01:00:00	0.00	1.72	0.00	0.00	0.00	73.900002	316.299988	0.00	10.550000	55.209999	0.00	24.299999
1	2003-03-01 01:00:00	0.00	1.45	0.00	0.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999	0.00	14.230000
2	2003-03-01 01:00:00	0.00	1.57	0.00	0.00	0.00	80.559998	224.199997	0.00	21.049999	63.240002	0.00	17.879999
3	2003-03-01 01:00:00	0.00	2.45	0.00	0.00	0.00	78.370003	450.399994	0.00	4.220000	67.839996	0.00	24.900000
4	2003-03-01 01:00:00	0.00	3.26	0.00	0.00	0.00	96.250000	479.100006	0.00	8.460000	95.779999	0.00	18.750000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	2003-03-01 18:00:00	0.00	0.43	0.00	0.00	0.00	35.070000	45.700001	0.00	36.939999	18.480000	0.00	10.580000
496	2003-03-01 18:00:00	0.00	0.52	0.00	0.00	0.00	20.570000	53.020000	0.00	37.849998	26.670000	0.00	13.320000
497	2003-03-01 18:00:00	1.29	0.24	0.00	0.00	0.00	33.570000	44.220001	0.00	43.730000	8.510000	0.00	9.790000
498	2003-03-01 18:00:00	0.64	0.49	0.54	0.00	0.11	46.750000	60.240002	0.00	28.889999	17.309999	0.00	9.060000
499	2003-03-01 18:00:00	1.64	0.35	1.59	4.65	0.01	11.550000	14.560000	2.26	55.299999	13.620000	2.33	4.670000

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    2003-03-01 01:00:00   NaN  1.72   NaN   NaN   NaN  73.900002  316.299988
1    2003-03-01 01:00:00   NaN  1.45   NaN   NaN  0.26  72.110001  250.000000
2    2003-03-01 01:00:00   NaN  1.57   NaN   NaN   NaN  80.559998  224.199997
3    2003-03-01 01:00:00   NaN  2.45   NaN   NaN   NaN  78.370003  450.399994
4    2003-03-01 01:00:00   NaN  3.26   NaN   NaN   NaN  96.250000  479.100006
..                   ...   ...   ...   ...   ...   ...        ...         ...
495  2003-03-01 18:00:00   NaN  0.43   NaN   NaN   NaN  35.070000   45.700001
496  2003-03-01 18:00:00   NaN  0.52   NaN   NaN   NaN  20.570000   53.020000
497  2003-03-01 18:00:00  1.29  0.24   NaN   NaN   NaN  33.570000   44.220001
498  2003-03-01 18:00:00  0.64  0.49  0.54   NaN  0.11  46.750000   60.240002
499  2003-03-01 18:00:00  1.64  0.35  1.59  4.65  0.01  11.550000   14.560000

      OXY        O_3       PM10   PXY       SO_2   TCH   TOL   station
0     NaN  10.550000  55.209999   NaN  24.299999   NaN   NaN  28079001
1    0.73   6.720000  52.389999   NaN  14.230000  1.55   NaN  28079035
2     NaN  21.049999  63.240002   NaN  17.879999   NaN   NaN  28079003
3     NaN   4.220000  67.839996   NaN  24.900000   NaN   NaN  28079004
4     NaN   8.460000  95.779999   NaN  18.750000   NaN   NaN  28079039
..    ...        ...        ...   ...        ...   ...   ...       ...
495   NaN  36.939999  18.480000   NaN  10.580000   NaN   NaN  28079036
496   NaN  37.849998  26.670000   NaN  13.320000   NaN   NaN  28079021
497   NaN  43.730000   8.510000   NaN   9.790000   NaN  4.55  28079022
498   NaN  28.889999  17.309999   NaN   9.060000  1.30  2.74  28079023
499  2.26  55.299999  13.620000  2.33   4.670000  1.27  8.79  28079024

[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 0x219bb0b66a0>
```

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_)
```

```
28079025.297791168
```

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

Out[26]:

	PM10
CO	0.295556
CO	0.295556
NOx	-0.020174
O_3	-0.010100

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x219c5f25d90>
```

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

Out[28]:
```
-0.03756098297636257
```

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]:
```
-0.03749465251025619
```

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]:
```
-0.03857121583567302
```

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.         -0.01750114 -0.00942557]
```

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

```
28079025.497416485
```

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

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

Out[40]:
```
-0.03751236211489917
```

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: 11.83939656327168
```

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

```
Mean Absolute square error: 294.29790134152586
```

Evalution Metrics for Ridge

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

```
Mean Absolute error: 11.83573162910839
```

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

```
Mean Absolute square error: 294.2790871057456
```

Evalution for elasticnet

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

```
Mean Absolute error: 11.817074151933193
```

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

```
Mean Absolute square error: 294.28411032805786
```

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_2003.
```

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

Out[52]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', '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	PXY	SO_
0	2003-03-01 01:00:00	1.00	1.72	4.00	5.00	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	24.29999
1	2003-03-01 01:00:00	1.00	1.45	4.00	5.00	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	14.23000
2	2003-03-01 01:00:00	1.00	1.57	4.00	5.00	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	17.87999
3	2003-03-01 01:00:00	1.00	2.45	4.00	5.00	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	24.90000
4	2003-03-01 01:00:00	1.00	3.26	4.00	5.00	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	18.75000
...	...	...	...	...	...	...	...	...	...	...	...	...	.
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.87000
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50	8.36000
243981	2003-10-01 00:00:00	1.00	2.00	4.00	5.00	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN	5.33000
243982	2003-10-01 00:00:00	1.00	2.00	4.00	5.00	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN	6.83000
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.06000

243984 rows × 16 columns

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

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

Out[55]: (243984, 3)

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

Out[56]: (243984,)

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)
```

```
[28079011]
```

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

Out[63]:
```
2.6779756914209437e-06
```

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

Out[64]:
```
0.06581348365824481
```

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 [ ]:
```

```