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_201
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1500 non-null   object 
 1   BEN      314 non-null    float64
 2   CO       628 non-null    float64
 3   EBE      314 non-null    float64
 4   NMHC     188 non-null    float64
 5   NO       1500 non-null   float64
 6   NO_2     1500 non-null   float64
 7   O_3      876 non-null    float64
 8   PM10     749 non-null    float64
 9   PM25     375 non-null    float64
 10  SO_2     628 non-null    float64
 11  TCH      188 non-null    float64
 12  TOL      314 non-null    float64
 13  station  1500 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 164.2+ 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	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	28079008
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28079011
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28079017

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

Out[5]:
```
(1500, 14)
```

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

Out[6]:
```
RangeIndex(start=0, stop=1500, 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	True	False	False	False	False	False	False	True	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1495	False	True	True	True	False	False	False	False	True	True	True	False	True	False
1496	False	True	False	True	True	False	False	False	True	True	False	True	True	False
1497	False	True	False	True	True	False	False	True	False	True	False	True	True	False
1498	False	False	True	False	True	False	False	True	False	False	False	True	False	False
1499	False	True	False	True	True	False	False	False	True	True	True	True	True	False

1500 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	2013-11-01 01:00:00	0.0	0.6	0.0	0.0	135.0	74.0	0.0	0.0	0.0	7.0	0.00	0.0	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	0.0	71.0	83.0	2.0	23.0	16.0	12.0	0.00	8.3	28079008
2	2013-11-01 01:00:00	3.9	0.0	2.8	0.0	49.0	70.0	0.0	0.0	0.0	0.0	0.00	9.0	28079011
3	2013-11-01 01:00:00	0.0	0.5	0.0	0.0	82.0	87.0	3.0	0.0	0.0	0.0	0.00	0.0	28079016
4	2013-11-01 01:00:00	0.0	0.0	0.0	0.0	242.0	111.0	2.0	0.0	0.0	12.0	0.00	0.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1495	2013-11-03 15:00:00	0.0	0.0	0.0	0.2	12.0	27.0	35.0	0.0	0.0	0.0	1.49	0.0	28079027
1496	2013-11-03 15:00:00	0.0	0.3	0.0	0.0	12.0	24.0	37.0	0.0	0.0	5.0	0.00	0.0	28079035
1497	2013-11-03 15:00:00	0.0	0.3	0.0	0.0	11.0	29.0	0.0	5.0	0.0	2.0	0.00	0.0	28079036
1498	2013-11-03 15:00:00	0.3	0.0	0.3	0.0	19.0	28.0	0.0	13.0	6.0	4.0	0.00	1.7	28079038
1499	2013-11-03 15:00:00	0.0	0.3	0.0	0.0	13.0	27.0	33.0	0.0	0.0	0.0	0.00	0.0	28079039

1500 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  \
0     2013-11-01 01:00:00  NaN  0.6  NaN   NaN  135.0   74.0   NaN   NaN
1     2013-11-01 01:00:00  1.5  0.5  1.3   NaN   71.0   83.0   2.0  23.0
2     2013-11-01 01:00:00  3.9  NaN  2.8   NaN   49.0   70.0   NaN   NaN
3     2013-11-01 01:00:00  NaN  0.5  NaN   NaN   82.0   87.0   3.0   NaN
4     2013-11-01 01:00:00  NaN  NaN  NaN   NaN  242.0  111.0   2.0   NaN
...                   ...  ...  ...  ...   ...    ...    ...   ...   ...
1495  2013-11-03 15:00:00  NaN  NaN  NaN   0.2   12.0   27.0  35.0   NaN
1496  2013-11-03 15:00:00  NaN  0.3  NaN   NaN   12.0   24.0  37.0   NaN
1497  2013-11-03 15:00:00  NaN  0.3  NaN   NaN   11.0   29.0   NaN   5.0
1498  2013-11-03 15:00:00  0.3  NaN  0.3   NaN   19.0   28.0   NaN  13.0
1499  2013-11-03 15:00:00  NaN  0.3  NaN   NaN   13.0   27.0  33.0   NaN

      PM25  SO_2   TCH  TOL   station
0      NaN   7.0   NaN  NaN  28079004
1     16.0  12.0   NaN  8.3  28079008
2      NaN   NaN   NaN  9.0  28079011
3      NaN   NaN   NaN  NaN  28079016
4      NaN  12.0   NaN  NaN  28079017
...    ...   ...   ...  ...       ...
1495   NaN   NaN  1.49  NaN  28079027
1496   NaN   5.0   NaN  NaN  28079035
1497   NaN   2.0   NaN  NaN  28079036
1498   6.0   4.0   NaN  1.7  28079038
1499   NaN   NaN   NaN  NaN  28079039

[1500 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")
```

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 0x1d5f8ec0430>
```

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', '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_)
```

```
28079017.20258374
```

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

Out[26]:

	PM10
CO	12.746834
CO	12.746834
O_3	0.125800

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1d58e0084f0>
```

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

Out[28]:
```
0.12146353200048332
```

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]:
```
0.1255896907035967
```

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]:
```
0.00022930226749873217
```

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_)
```

```
[3.1824172  3.18241686 0.07793003]
```

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

```
28079031.78952407
```

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

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

Out[40]:
```
0.0734865241839534
```

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: 13.357190642663173
```

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

```
Mean Absolute square error: 262.11287368574864
```

Evalution Metrics for Ridge

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

```
Mean Absolute error: 13.387716338725554
```

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

```
Mean Absolute square error: 260.8818271050424
```

Evalution for elasticnet

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

```
Mean Absolute error: 14.282653449020453
```

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

```
Mean Absolute square error: 276.4268968910337
```

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_2013.
```

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	2013-11-01 01:00:00	1.0	0.6	4.0	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	28079008
2	2013-11-01 01:00:00	3.9	2.0	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28079011
3	2013-11-01 01:00:00	1.0	0.5	4.0	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2013-11-01 01:00:00	1.0	2.0	4.0	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	1.0	0.4	4.0	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28079056
209876	2013-03-01 00:00:00	1.0	0.4	4.0	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28079057
209877	2013-03-01 00:00:00	1.0	2.0	4.0	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28079058
209878	2013-03-01 00:00:00	1.0	2.0	4.0	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28079059
209879	2013-03-01 00:00:00	1.0	2.0	4.0	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28079060

209880 rows × 14 columns

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

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

Out[55]: (209880, 2)

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

Out[56]: (209880,)

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: Convergenc
eWarning: lbfgs 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/sta
ble/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://s
cikit-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)
```

```
[28079057]
```

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

Out[63]:
```
1.2545406254635368e-17
```

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

Out[64]:
```
4.556134398524688e-218
```

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 [ ]:
```

```