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      306 non-null    float64
 2   CO       628 non-null    float64
 3   EBE      306 non-null    float64
 4   NMHC     179 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     747 non-null    float64
 9   PM25     375 non-null    float64
 10  SO_2     624 non-null    float64
 11  TCH      179 non-null    float64
 12  TOL      306 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	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	28079004
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28079008
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	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	False	False	False	False	False	False	False	False	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	True	False	False	False	True	True	True	True	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	2014-06-01 01:00:00	0.0	0.2	0.0	0.00	3.0	10.0	0.0	0.0	0.0	3.0	0.00	0.0	28079004
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28079008
2	2014-06-01 01:00:00	0.3	0.0	0.1	0.00	2.0	6.0	0.0	0.0	0.0	0.0	0.00	1.1	28079011
3	2014-06-01 01:00:00	0.0	0.2	0.0	0.00	1.0	6.0	79.0	0.0	0.0	0.0	0.00	0.0	28079016
4	2014-06-01 01:00:00	0.0	0.0	0.0	0.00	1.0	6.0	75.0	0.0	0.0	4.0	0.00	0.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1495	2014-06-03 15:00:00	0.0	0.0	0.0	0.00	5.0	21.0	110.0	0.0	0.0	0.0	0.00	0.0	28079027
1496	2014-06-03 15:00:00	0.0	0.3	0.0	0.00	6.0	21.0	115.0	0.0	0.0	4.0	0.00	0.0	28079035
1497	2014-06-03 15:00:00	0.0	0.2	0.0	0.00	8.0	27.0	0.0	8.0	0.0	2.0	0.00	0.0	28079036
1498	2014-06-03 15:00:00	0.1	0.0	0.8	0.00	14.0	36.0	0.0	18.0	5.0	3.0	0.00	2.8	28079038
1499	2014-06-03 15:00:00	0.0	0.3	0.0	0.00	7.0	28.0	101.0	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  PM25  \
0     2014-06-01 01:00:00  NaN  0.2  NaN   NaN   3.0  10.0    NaN   NaN   NaN
1     2014-06-01 01:00:00  0.2  0.2  0.1  0.11   3.0  17.0   68.0  10.0   5.0
2     2014-06-01 01:00:00  0.3  NaN  0.1   NaN   2.0   6.0    NaN   NaN   NaN
3     2014-06-01 01:00:00  NaN  0.2  NaN   NaN   1.0   6.0   79.0   NaN   NaN
4     2014-06-01 01:00:00  NaN  NaN  NaN   NaN   1.0   6.0   75.0   NaN   NaN
...                   ...  ...  ...  ...   ...   ...   ...    ...   ...   ...
1495  2014-06-03 15:00:00  NaN  NaN  NaN   NaN   5.0  21.0  110.0   NaN   NaN
1496  2014-06-03 15:00:00  NaN  0.3  NaN   NaN   6.0  21.0  115.0   NaN   NaN
1497  2014-06-03 15:00:00  NaN  0.2  NaN   NaN   8.0  27.0    NaN   8.0   NaN
1498  2014-06-03 15:00:00  0.1  NaN  0.8   NaN  14.0  36.0    NaN  18.0   5.0
1499  2014-06-03 15:00:00  NaN  0.3  NaN   NaN   7.0  28.0  101.0   NaN   NaN

      SO_2   TCH  TOL   station
0      3.0   NaN  NaN  28079004
1      5.0  1.36  1.3  28079008
2      NaN   NaN  1.1  28079011
3      NaN   NaN  NaN  28079016
4      4.0   NaN  NaN  28079017
...    ...   ...  ...       ...
1495   NaN   NaN  NaN  28079027
1496   4.0   NaN  NaN  28079035
1497   2.0   NaN  NaN  28079036
1498   3.0   NaN  2.8  28079038
1499   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")
```

Out[15]:
```
<AxesSubplot:ylabel='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 0x23688d78820>
```

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_)
```

```
28079023.65300028
```

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

Out[26]:

	PM10
CO	9.835674
CO	9.835674
O_3	0.011736

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x23698183e50>
```

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

Out[28]:
```
0.12818354646763808
```

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]:
```
0.12895644721244592
```

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]:
```
-0.0011687075052981832
```

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_)
```

```
[ 2.71502248  2.71502191 -0.00335592]
```

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

```
28079034.254144702
```

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

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

Out[40]:
```
0.06615995694223553
```

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: 13.552289997355805
```

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

```
Mean Absolute square error: 265.3609692400481
```

Evalution Metrics for Ridge

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

```
Mean Absolute error: 13.581106073103017
```

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

```
Mean Absolute square error: 265.12571594798465
```

Evalution for elasticnet

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

```
Mean Absolute error: 14.241044665483965
```

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

```
Mean Absolute square error: 284.23952993424217
```

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_2014.
```

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	2014-06-01 01:00:00	1.0	0.2	4.0	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	28079004
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28079008
2	2014-06-01 01:00:00	0.3	2.0	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3	2014-06-01 01:00:00	1.0	0.2	4.0	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2014-06-01 01:00:00	1.0	2.0	4.0	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210019	2014-09-01 00:00:00	1.0	0.5	4.0	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN	28079056
210020	2014-09-01 00:00:00	1.0	0.3	4.0	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN	28079057
210021	2014-09-01 00:00:00	1.0	2.0	4.0	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN	28079058
210022	2014-09-01 00:00:00	1.0	2.0	4.0	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN	28079059
210023	2014-09-01 00:00:00	1.0	2.0	4.0	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN	28079060

210024 rows × 14 columns

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

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

Out[55]: (210024, 2)

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

Out[56]: (210024,)

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)
```

```
[28079016]
```

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

Out[63]:
```
6.511285321372418e-06
```

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

Out[64]:
```
4.753197380546183e-160
```

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 [ ]:
```
```