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: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1000 non-null   object 
 1   BEN      292 non-null    float64
 2   CO       459 non-null    float64
 3   EBE      292 non-null    float64
 4   MXY      83 non-null     float64
 5   NMHC     249 non-null    float64
 6   NO_2     1000 non-null   float64
 7   NOx      1000 non-null   float64
 8   OXY      83 non-null     float64
 9   O_3      618 non-null    float64
 10  PM10     500 non-null    float64
 11  PM25     251 non-null    float64
 12  PXY      83 non-null     float64
 13  SO_2     460 non-null    float64
 14  TCH      249 non-null    float64
 15  TOL      292 non-null    float64
 16  station  1000 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 132.9+ KB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	T
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	NaN	NaN	NaN	10.15	N
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	NaN	NaN	NaN	12.24	N
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	NaN	NaN	NaN	NaN	N
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.41	7.870000	NaN	10.06	1
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.67	22.030001	NaN	10.68	N

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

Out[5]:
```
(1000, 17)
```

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

Out[6]:
```
RangeIndex(start=0, stop=1000, step=1)
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', '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	PM25	PXY	SO_2	TCH	TOL	s
0	False	True	False	True	True	True	False	False	True	False	True	True	True	False	True	True	
1	False	True	False	True	True	True	False	False	True	True	True	True	True	False	True	True	
2	False	True	False	True	True	True	False	False	True	False	True	True	True	True	True	True	
3	False	False	False	False	True	False	False	False	True	False	False	False	True	False	False	False	
4	False	False	True	False	True	True	False	False	True	True	False	False	True	False	True	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
996	False	True	True	True	True	False	False	False	True	False	True	True	True	True	False	True	
997	False	True	True	True	True	True	False	False	True	True	False	False	True	True	True	True	
998	False	True	True	True	True	True	False	False	True	False	True	True	True	True	True	True	
999	False	True	True	True	True	True	False	False	True	True	False	False	True	True	True	True	

1000 rows × 17 columns

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

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	S
0	2010-03-01 01:00:00	0.00	0.29	0.00	0.00	0.00	25.090000	29.219999	0.00	68.930000	0.000000	0.000000	0.00	1
1	2010-03-01 01:00:00	0.00	0.27	0.00	0.00	0.00	24.879999	30.040001	0.00	0.000000	0.000000	0.000000	0.00	1
2	2010-03-01 01:00:00	0.00	0.28	0.00	0.00	0.00	17.410000	20.540001	0.00	72.120003	0.000000	0.000000	0.00	
3	2010-03-01 01:00:00	0.38	0.24	1.74	0.00	0.05	15.610000	21.080000	0.00	72.970001	19.410000	7.870000	0.00	1
4	2010-03-01 01:00:00	0.79	0.00	1.32	0.00	0.00	21.430000	26.070000	0.00	0.000000	24.670000	22.030001	0.00	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	2010-03-02 18:00:00	0.51	0.20	0.91	1.27	0.39	20.330000	22.940001	1.42	86.410004	14.280000	6.830000	0.98	
996	2010-03-02 18:00:00	0.00	0.00	0.00	0.00	0.13	28.370001	40.669998	0.00	73.480003	0.000000	0.000000	0.00	
997	2010-03-02 18:00:00	0.00	0.00	0.00	0.00	0.00	44.029999	50.509998	0.00	0.000000	22.049999	7.100000	0.00	
998	2010-03-02 18:00:00	0.00	0.00	0.00	0.00	0.00	31.770000	37.040001	0.00	85.040001	0.000000	0.000000	0.00	
999	2010-03-02 18:00:00	0.00	0.00	0.00	0.00	0.00	32.500000	39.279999	0.00	0.000000	16.350000	7.220000	0.00	

1000 rows × 17 columns

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

Out[10]:
```
<bound method DataFrame.isna of                     date   BEN    CO   EBE   MXY  NMHC
NO_2        NOx  \
0    2010-03-01 01:00:00   NaN  0.29   NaN   NaN   NaN  25.090000  29.219999
1    2010-03-01 01:00:00   NaN  0.27   NaN   NaN   NaN  24.879999  30.040001
2    2010-03-01 01:00:00   NaN  0.28   NaN   NaN   NaN  17.410000  20.540001
3    2010-03-01 01:00:00  0.38  0.24  1.74   NaN  0.05  15.610000  21.080000
4    2010-03-01 01:00:00  0.79   NaN  1.32   NaN   NaN  21.430000  26.070000
..                   ...   ...   ...   ...   ...   ...        ...        ...
995  2010-03-02 18:00:00  0.51  0.20  0.91  1.27  0.39  20.330000  22.940001
996  2010-03-02 18:00:00   NaN   NaN   NaN   NaN  0.13  28.370001  40.669998
997  2010-03-02 18:00:00   NaN   NaN   NaN   NaN   NaN  44.029999  50.509998
998  2010-03-02 18:00:00   NaN   NaN   NaN   NaN   NaN  31.770000  37.040001
999  2010-03-02 18:00:00   NaN   NaN   NaN   NaN   NaN  32.500000  39.279999

      OXY        O_3       PM10       PM25   PXY   SO_2   TCH   TOL   station
0     NaN  68.930000        NaN        NaN   NaN  10.15   NaN   NaN  28079003
1     NaN        NaN        NaN        NaN   NaN  12.24   NaN   NaN  28079004
2     NaN  72.120003        NaN        NaN   NaN    NaN   NaN   NaN  28079039
3     NaN  72.970001  19.410000   7.870000   NaN  10.06  1.52  1.49  28079008
4     NaN        NaN  24.670000  22.030001   NaN  10.68   NaN  2.88  28079038
..    ...        ...        ...        ...   ...    ...   ...   ...       ...
995  1.42  86.410004  14.280000   6.830000  0.98   7.79  1.63  1.25  28079024
996   NaN  73.480003        NaN        NaN   NaN    NaN  1.33   NaN  28079027
997   NaN        NaN  22.049999   7.100000   NaN    NaN   NaN   NaN  28079047
998   NaN  85.040001        NaN        NaN   NaN    NaN   NaN   NaN  28079049
999   NaN        NaN  16.350000   7.220000   NaN    NaN   NaN   NaN  28079050

[1000 rows x 17 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 0x1abd62340d0>
```

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_)
```

```
28079022.262776714
```

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

Out[26]:

	PM10
CO	8.002475
CO	8.002475
NOx	0.015476
O_3	0.132265

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1abe4b7cc70>
```

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

Out[28]:
```
0.0705715235124541
```

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]:
```
0.0705352413801531
```

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]:
```
0.0127649478782349
```

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_)
```

```
[1.89738239 1.89736525 0.00473216 0.10628334]
```

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

```
28079032.072577298
```

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

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

Out[40]:
```
0.037039169467973476
```

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: 16.869182174503802
```

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

```
Mean Absolute square error: 462.492646010391
```

Evalution Metrics for Ridge

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

```
Mean Absolute error: 16.923351790395877
```

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

```
Mean Absolute square error: 462.5107003521669
```

Evalution for elasticnet

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

```
Mean Absolute error: 17.948945077831546
```

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

```
Mean Absolute square error: 479.1786713919222
```

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_2010.
```

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

Out[52]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', '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	PM25	PX
0	2010-03-01 01:00:00	1.00	0.29	4.00	5.0	NaN	25.090000	29.219999	NaN	68.930000	NaN	NaN	Na
1	2010-03-01 01:00:00	1.00	0.27	4.00	5.0	NaN	24.879999	30.040001	NaN	NaN	NaN	NaN	Na
2	2010-03-01 01:00:00	1.00	0.28	4.00	5.0	NaN	17.410000	20.540001	NaN	72.120003	NaN	NaN	Na
3	2010-03-01 01:00:00	0.38	0.24	1.74	5.0	0.05	15.610000	21.080000	NaN	72.970001	19.410000	7.870000	Na
4	2010-03-01 01:00:00	0.79	2.00	1.32	5.0	NaN	21.430000	26.070000	NaN	NaN	24.670000	22.030001	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	1.00	0.55	4.00	5.0	NaN	125.000000	219.899994	NaN	25.379999	NaN	NaN	Na
209444	2010-08-01 00:00:00	1.00	0.27	4.00	5.0	NaN	45.709999	47.410000	NaN	NaN	51.259998	NaN	Na
209445	2010-08-01 00:00:00	1.00	2.00	4.00	5.0	0.24	46.560001	49.040001	NaN	46.250000	NaN	NaN	Na
209446	2010-08-01 00:00:00	1.00	2.00	4.00	5.0	NaN	46.770000	50.119999	NaN	77.709999	NaN	NaN	Na
209447	2010-08-01 00:00:00	0.92	0.43	0.71	5.0	0.25	76.330002	88.190002	NaN	52.259998	47.150002	26.860001	Na

209448 rows × 17 columns

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

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

Out[55]: (209448, 3)

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

Out[56]: (209448,)

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 =[[3,90,5]]
```

In [62]:
```
prediction5 =logr.predict(observation)
print(prediction5)
```

```
[28079057]
```

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

Out[63]:
```
1.515722885966314e-15
```

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

Out[64]:
```
1.0849567985746407e-17
```

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 [ ]:
```

```