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_2018
```

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      126 non-null    float64
 2   CH4      63 non-null     float64
 3   CO       206 non-null    float64
 4   EBE      126 non-null    float64
 5   NMHC     63 non-null     float64
 6   NO       495 non-null    float64
 7   NO_2     495 non-null    float64
 8   NOx      495 non-null    float64
 9   O_3      286 non-null    float64
 10  PM10     271 non-null    float64
 11  PM25     147 non-null    float64
 12  SO_2     209 non-null    float64
 13  TCH      63 non-null     float64
 14  TOL      126 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	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8	28079008
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	NaN	NaN	28079017

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', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [8]:
```
data.isna()
```
Out[8]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	False	True	True	False	True	True	False	False	False	True	True	True	False	True	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	True	True	False	True	False	False	False	True	True	True	True	True	False	False
3	False	True	True	False	True	True	False	False	False	False	True	True	True	True	True	False
4	False	True	True	True	True	True	False	False	False	False	True	True	False	True	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	False	True	True	True	True	True	False	False	False	False	True	True	True	True	True	False
496	False	True	True	True	True	True	False	False	False	True	False	False	True	True	True	False
497	False	True	True	True	True	True	False	False	False	False	True	True	True	True	True	False
498	False	False	False	True	False	False	False	False	False	True	False	True	True	False	False	False
499	False	True	True	False	True	True	False	False	False	False	False	False	True	True	True	False

500 rows × 16 columns

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

Out[9]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	29.0	31.0	0.0	0.0	0.0	2.0	0.00	0.0	28079004
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8	28079008
2	2018-03-01 01:00:00	0.4	0.00	0.0	0.2	0.00	4.0	41.0	47.0	0.0	0.0	0.0	0.0	0.00	1.1	28079011
3	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	35.0	37.0	54.0	0.0	0.0	0.0	0.00	0.0	28079016
4	2018-03-01 01:00:00	0.0	0.00	0.0	0.0	0.00	1.0	27.0	29.0	49.0	0.0	0.0	3.0	0.00	0.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	2018-03-01 21:00:00	0.0	0.00	0.0	0.0	0.00	1.0	18.0	19.0	66.0	0.0	0.0	0.0	0.00	0.0	28079049
496	2018-03-01 21:00:00	0.0	0.00	0.0	0.0	0.00	30.0	35.0	81.0	0.0	6.0	5.0	0.0	0.00	0.0	28079050
497	2018-03-01 21:00:00	0.0	0.00	0.0	0.0	0.00	1.0	18.0	20.0	67.0	0.0	0.0	0.0	0.00	0.0	28079054
498	2018-03-01 21:00:00	0.4	1.20	0.0	0.1	0.06	2.0	22.0	26.0	0.0	4.0	0.0	0.0	1.25	0.8	28079055
499	2018-03-01 21:00:00	0.0	0.00	0.3	0.0	0.00	19.0	35.0	64.0	65.0	8.0	4.0	0.0	0.00	0.0	28079056

500 rows × 16 columns

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

Out[10]:
```
<bound method DataFrame.isna of                    date  BEN   CH4   CO  EBE  NMHC    NO  N
O_2   NOx   O_3  \
0    2018-03-01 01:00:00  NaN   NaN  0.3  NaN   NaN   1.0  29.0  31.0   NaN
1    2018-03-01 01:00:00  0.5  1.39  0.3  0.2  0.02   6.0  40.0  49.0  52.0
2    2018-03-01 01:00:00  0.4   NaN  NaN  0.2   NaN   4.0  41.0  47.0   NaN
3    2018-03-01 01:00:00  NaN   NaN  0.3  NaN   NaN   1.0  35.0  37.0  54.0
4    2018-03-01 01:00:00  NaN   NaN  NaN  NaN   NaN   1.0  27.0  29.0  49.0
..                   ...  ...   ...  ...  ...   ...   ...   ...   ...   ...
495  2018-03-01 21:00:00  NaN   NaN  NaN  NaN   NaN   1.0  18.0  19.0  66.0
496  2018-03-01 21:00:00  NaN   NaN  NaN  NaN   NaN  30.0  35.0  81.0   NaN
497  2018-03-01 21:00:00  NaN   NaN  NaN  NaN   NaN   1.0  18.0  20.0  67.0
498  2018-03-01 21:00:00  0.4  1.20  NaN  0.1  0.06   2.0  22.0  26.0   NaN
499  2018-03-01 21:00:00  NaN   NaN  0.3  NaN   NaN  19.0  35.0  64.0  65.0

     PM10  PM25  SO_2   TCH  TOL   station
0     NaN   NaN   2.0   NaN  NaN  28079004
1     5.0   4.0   3.0  1.41  0.8  28079008
2     NaN   NaN   NaN   NaN  1.1  28079011
3     NaN   NaN   NaN   NaN  NaN  28079016
4     NaN   NaN   3.0   NaN  NaN  28079017
..    ...   ...   ...   ...  ...       ...
495   NaN   NaN   NaN   NaN  NaN  28079049
496   6.0   5.0   NaN   NaN  NaN  28079050
497   NaN   NaN   NaN   NaN  NaN  28079054
498   4.0   NaN   NaN  1.25  0.8  28079055
499   8.0   4.0   NaN   NaN  NaN  28079056

[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")
```

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 0x28bcafd9880>
```

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_)
```

```
28079022.527471773
```

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

Out[26]:

	PM10
CO	10.310919
CO	10.310919
O_3	0.016526

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x28bd5cd2fa0>
```

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

Out[28]:
```
0.12718946102785744
```

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]:
```
0.1282399636175272
```

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]:
```
-0.004789289789998374
```

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_)
```

```
[ 2.74144059  2.74145716 -0.01452483]
```

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

```
28079034.231329445
```

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

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

Out[40]:
```
0.05936449184919146
```

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: 14.018862525075674

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

Mean Absolute square error: 283.53153379561275
```

Evalution Metrics for Ridge

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

Mean Absolute error: 14.112414263164004

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

Mean Absolute square error: 283.1902791965836
```

Evalution for elasticnet

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

Mean Absolute error: 14.796460316677889

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

Mean Absolute square error: 305.56439967222553
```

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_2018.

In [52]: df.columns

Out[52]: Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', '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	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2018-03-01 01:00:00	1.0	NaN	0.3	4.0	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8	28079008
2	2018-03-01 01:00:00	0.4	NaN	2.0	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3	2018-03-01 01:00:00	1.0	NaN	0.3	4.0	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2018-03-01 01:00:00	1.0	NaN	2.0	4.0	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	1.0	NaN	0.5	4.0	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	NaN	NaN	28079056
69092	2018-02-01 00:00:00	1.0	NaN	0.7	4.0	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	NaN	NaN	28079057
69093	2018-02-01 00:00:00	1.0	NaN	2.0	4.0	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	NaN	NaN	28079058
69094	2018-02-01 00:00:00	1.0	NaN	2.0	4.0	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	NaN	NaN	28079059
69095	2018-02-01 00:00:00	1.0	NaN	2.0	4.0	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	NaN	NaN	28079060

69096 rows × 16 columns

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

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

```
Out[55]: (69096, 2)
```

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

```
Out[56]: (69096,)
```

```
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)
```

```
[28079004]
```

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

Out[63]:
```
0.9389282324945811
```

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

Out[64]:
```
4.486888459491602e-203
```

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 [ ]: