In [1]:
```python
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
data=pd.read_csv(r"C:\Users\user\Desktop\DINESH\C10_air\madrid_2004.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389

245496 rows × 17 columns

In [3]:
```
data.head(10)
```

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110001
6	2004-08-01 01:00:00	NaN	0.43	NaN	NaN	0.17	54.270000	64.279999	NaN	66.589996	54.270000
7	2004-08-01 01:00:00	1.41	0.47	2.35	NaN	0.02	71.730003	87.519997	NaN	53.270000	45.180000
8	2004-08-01 01:00:00	NaN	1.28	NaN	NaN	NaN	147.699997	202.500000	NaN	10.280000	52.430000
9	2004-08-01 01:00:00	NaN	0.43	NaN	NaN	0.27	54.290001	68.099998	NaN	66.709999	54.700001

In [4]:

```
data.tail(20)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
245476	2004-06-01 00:00:00	NaN	1.09	NaN	NaN	NaN	97.199997	130.300003	NaN	5.100000	24.520
245477	2004-06-01 00:00:00	NaN	0.60	NaN	NaN	NaN	82.959999	109.099998	NaN	31.730000	I
245478	2004-06-01 00:00:00	NaN	0.64	NaN	NaN	0.07	96.010002	148.399994	NaN	5.580000	13.590
245479	2004-06-01 00:00:00	NaN	0.53	NaN	NaN	NaN	84.010002	96.470001	NaN	13.110000	5.030
245480	2004-06-01 00:00:00	NaN	0.52	NaN	NaN	0.15	95.650002	116.400002	NaN	8.750000	21.959
245481	2004-06-01 00:00:00	NaN	0.96	NaN	NaN	NaN	89.629997	162.800003	NaN	9.710000	31.590
245482	2004-06-01 00:00:00	5.90	0.78	4.18	NaN	0.21	99.489998	181.399994	NaN	9.670000	24.059
245483	2004-06-01 00:00:00	NaN	0.29	NaN	NaN	NaN	89.970001	115.099998	NaN	12.730000	19.049
245484	2004-06-01 00:00:00	NaN	0.62	NaN	NaN	NaN	94.419998	141.100006	NaN	5.490000	36.720
245485	2004-06-01 00:00:00	NaN	0.07	NaN	NaN	0.75	71.010002	81.650002	NaN	18.910000	38.970
245486	2004-06-01 00:00:00	NaN	0.74	NaN	NaN	NaN	104.400002	199.100006	NaN	3.370000	45.700
245487	2004-06-01 00:00:00	NaN	0.73	NaN	NaN	NaN	103.000000	132.699997	NaN	3.960000	7.770
245488	2004-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	79.470001	99.080002	NaN	15.180000	5.160
245489	2004-06-01 00:00:00	4.80	0.51	NaN	NaN	NaN	64.680000	126.199997	NaN	6.610000	46.759
245490	2004-06-01 00:00:00	NaN	0.71	NaN	NaN	0.29	105.800003	165.100006	NaN	5.770000	23.280
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389

In [5]:
```
data.describe()
```

Out[5]:

	BEN	CO	EBE	MXY	NMHC	NO_2
count	65158.000000	226043.000000	56781.000000	39867.000000	107630.000000	243280.000000
mean	2.126076	0.654113	2.754981	5.241563	0.167904	60.757049
std	2.479568	0.610924	3.547181	5.544696	0.168483	33.765691
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.660000	0.290000	1.000000	1.610000	0.060000	35.290001
50%	1.390000	0.490000	1.840000	3.600000	0.120000	56.459999
75%	2.750000	0.820000	3.300000	6.970000	0.220000	80.410004
max	46.180000	12.000000	81.860001	99.320000	4.810000	398.500000

In [6]:
```
np.shape(data)
```

Out[6]: (245496, 17)

In [7]:
```
np.size(data)
```

Out[7]: 4173432

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

Out[8]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
0	False	True	False	True	True	True	False	False	True	False	False	False	True
1	False	False	False	False	False	False	False	False	False	False	False	True	False
2	False	True	False	True	True	True	False	False	True	False	False	True	True
3	False	True	False	True	True	True	False	False	True	False	False	True	True
4	False	True	False	True	True	True	False	False	True	False	False	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
245491	False	False	False	False	False	False	False	False	False	False	False	False	False
245492	False	False	False	False	False	True	False	False	False	False	False	True	False
245493	False	True	True	True	True	False	False	False	True	False	False	False	True
245494	False	True	True	True	True	False	False	False	True	True	False	True	True
245495	False	False	False	False	False	False	False	False	False	False	False	False	False

245496 rows × 17 columns

In [9]:
```
data.dropna()
```

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.549
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.630
32	2004-08-01 02:00:00	1.94	0.67	3.14	4.91	0.06	113.500000	165.800003	2.56	26.980000	86.930
49	2004-08-01 02:00:00	0.29	0.30	0.47	0.76	0.07	33.919998	34.840000	0.46	75.570000	48.959
...	...	...	...	...	...	...	...	...	...	...	
245463	2004-05-31 23:00:00	0.62	0.08	0.54	0.70	0.04	44.360001	45.450001	0.42	43.419998	19.290
245467	2004-05-31 23:00:00	2.39	0.67	2.49	3.92	0.20	89.809998	132.800003	2.09	14.740000	31.809
245473	2004-06-01 00:00:00	3.72	1.12	4.33	8.79	0.24	113.900002	253.600006	4.51	9.380000	21.219
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389

19397 rows × 17 columns

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

Out[10]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [11]:
```
sd=data[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
```

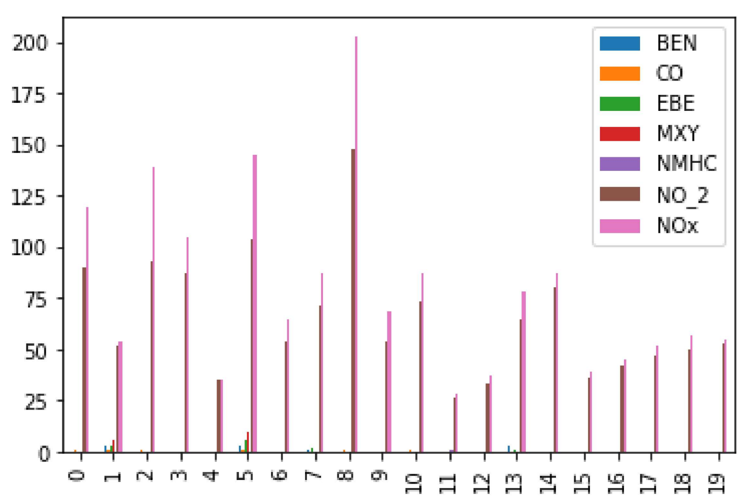
In [12]:
```
dd=sd.head(20)
dd
```

Out[12]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
0	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002
1	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001
2	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006
3	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000
4	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998
5	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003
6	NaN	0.43	NaN	NaN	0.17	54.270000	64.279999
7	1.41	0.47	2.35	NaN	0.02	71.730003	87.519997
8	NaN	1.28	NaN	NaN	NaN	147.699997	202.500000
9	NaN	0.43	NaN	NaN	0.27	54.290001	68.099998
10	NaN	0.60	NaN	NaN	NaN	73.410004	87.059998
11	NaN	0.22	NaN	NaN	1.11	26.730000	28.510000
12	NaN	0.37	NaN	NaN	NaN	33.570000	37.590000
13	2.93	0.40	1.36	NaN	0.02	64.830002	78.709999
14	NaN	0.21	NaN	NaN	NaN	80.660004	87.050003
15	NaN	0.37	NaN	NaN	NaN	36.279999	38.810001
16	NaN	0.23	NaN	NaN	0.37	42.150002	44.810001
17	NaN	0.41	NaN	NaN	NaN	47.110001	51.950001
18	NaN	0.46	NaN	NaN	NaN	50.250000	56.279999
19	NaN	0.30	NaN	NaN	NaN	52.410000	54.599998

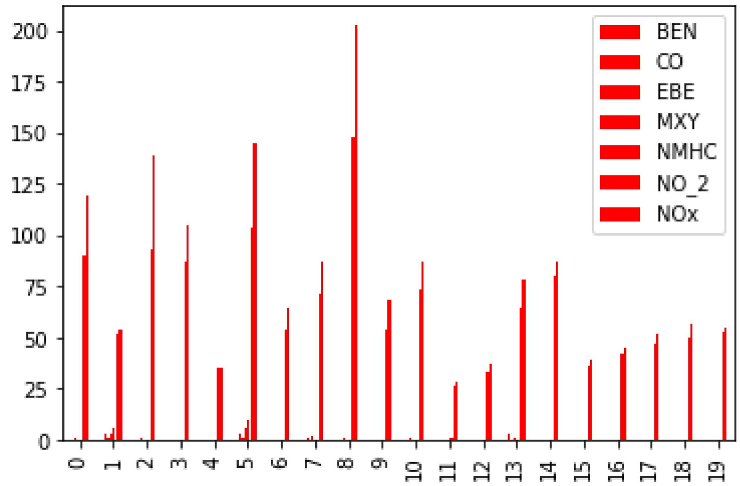
In [13]:
```
dd.plot.bar()
```

Out[13]: <AxesSubplot:>



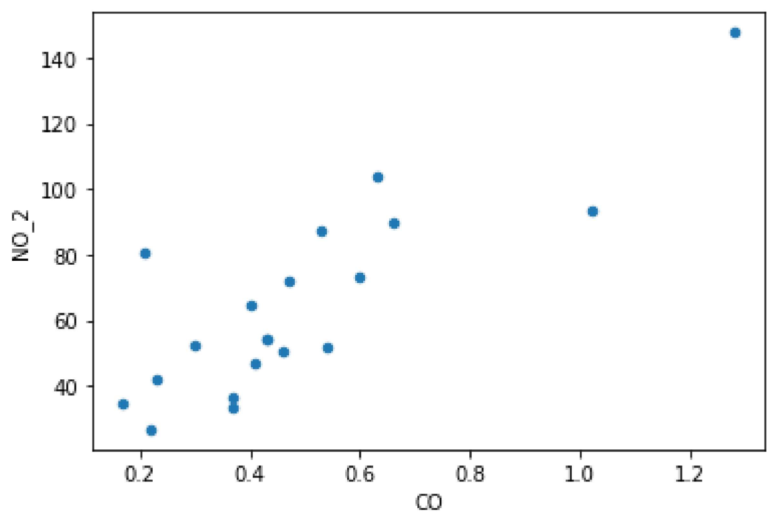
In [14]:
```
dd.plot.bar(color='r')
```

Out[14]: <AxesSubplot:>



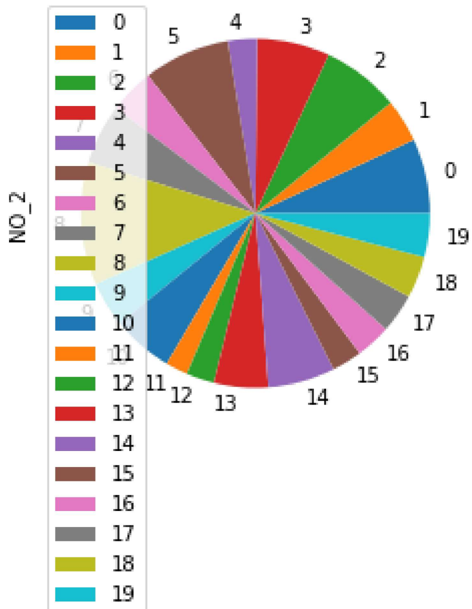
In [15]:
```
dd.plot.scatter(x='CO',y='NO_2')
```

Out[15]: <AxesSubplot:xlabel='CO', ylabel='NO_2'>



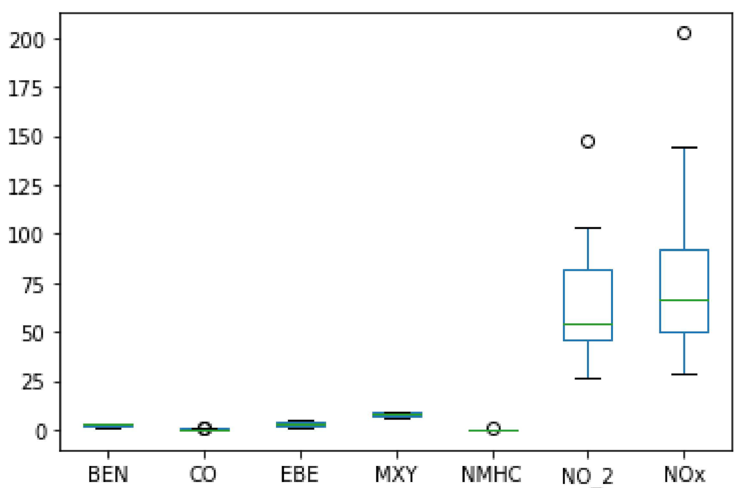
In [16]:
```
dd.plot.pie(y='NO_2')
```

Out[16]: <AxesSubplot:ylabel='NO_2'>



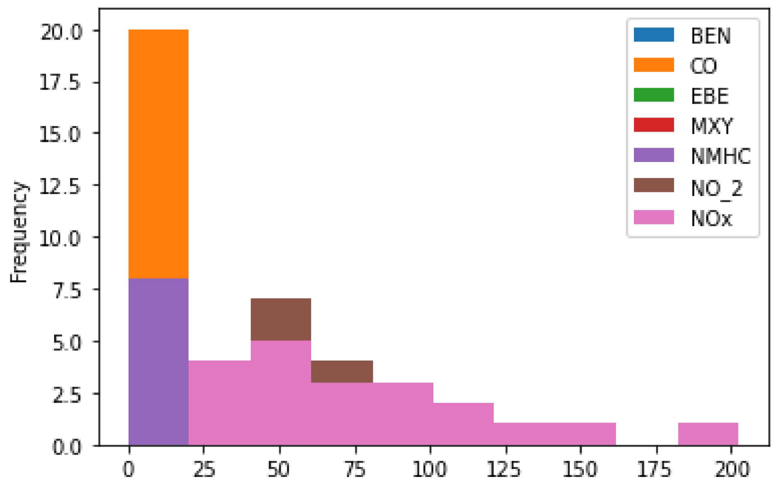
In [17]:
```
dd.plot.box()
```

Out[17]: <AxesSubplot:>



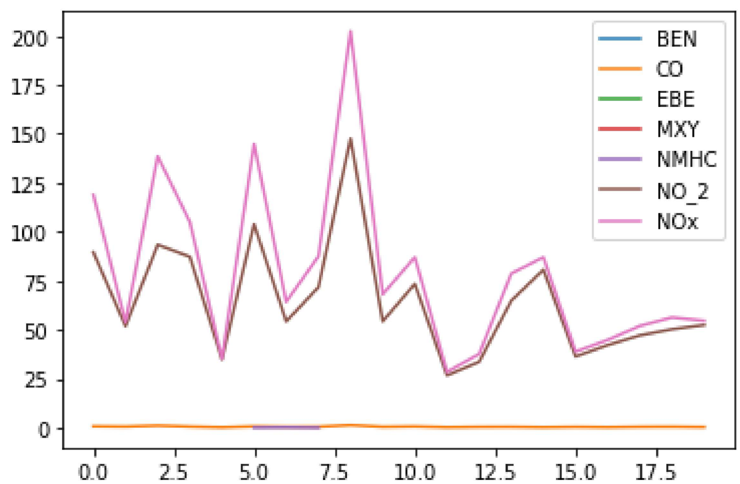
In [18]:
```
dd.plot.hist()
```

Out[18]: <AxesSubplot:ylabel='Frequency'>

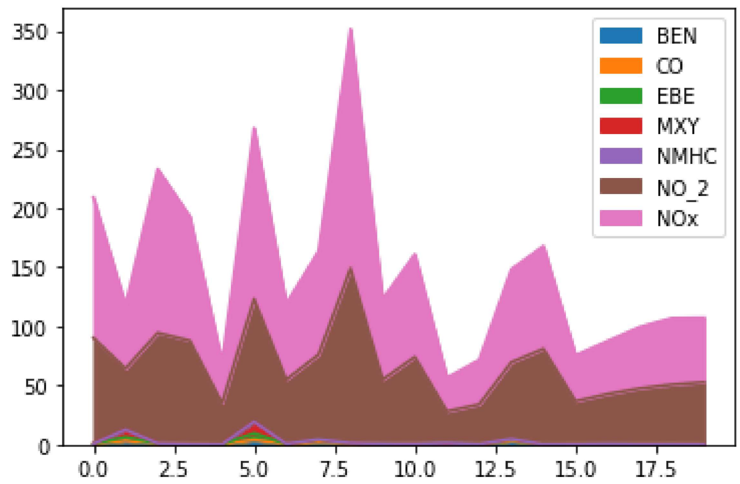


In [19]:
```
dd.plot.line()
```

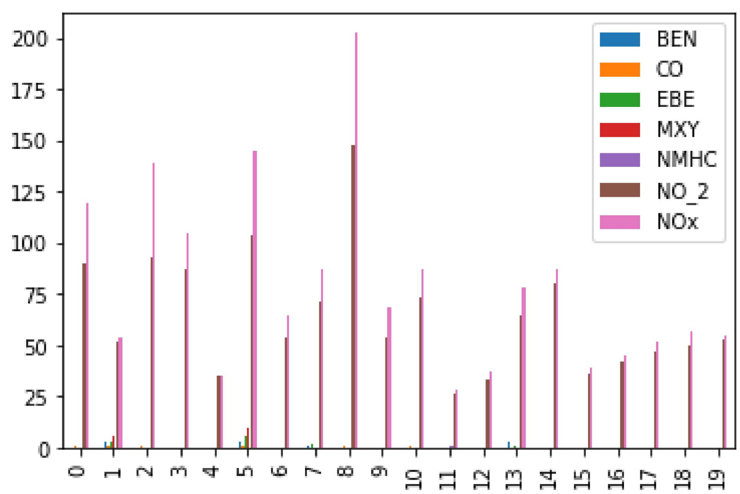
Out[19]: <AxesSubplot:>



In [20]:
```
dd.plot.area()
```
Out[20]: <AxesSubplot:>

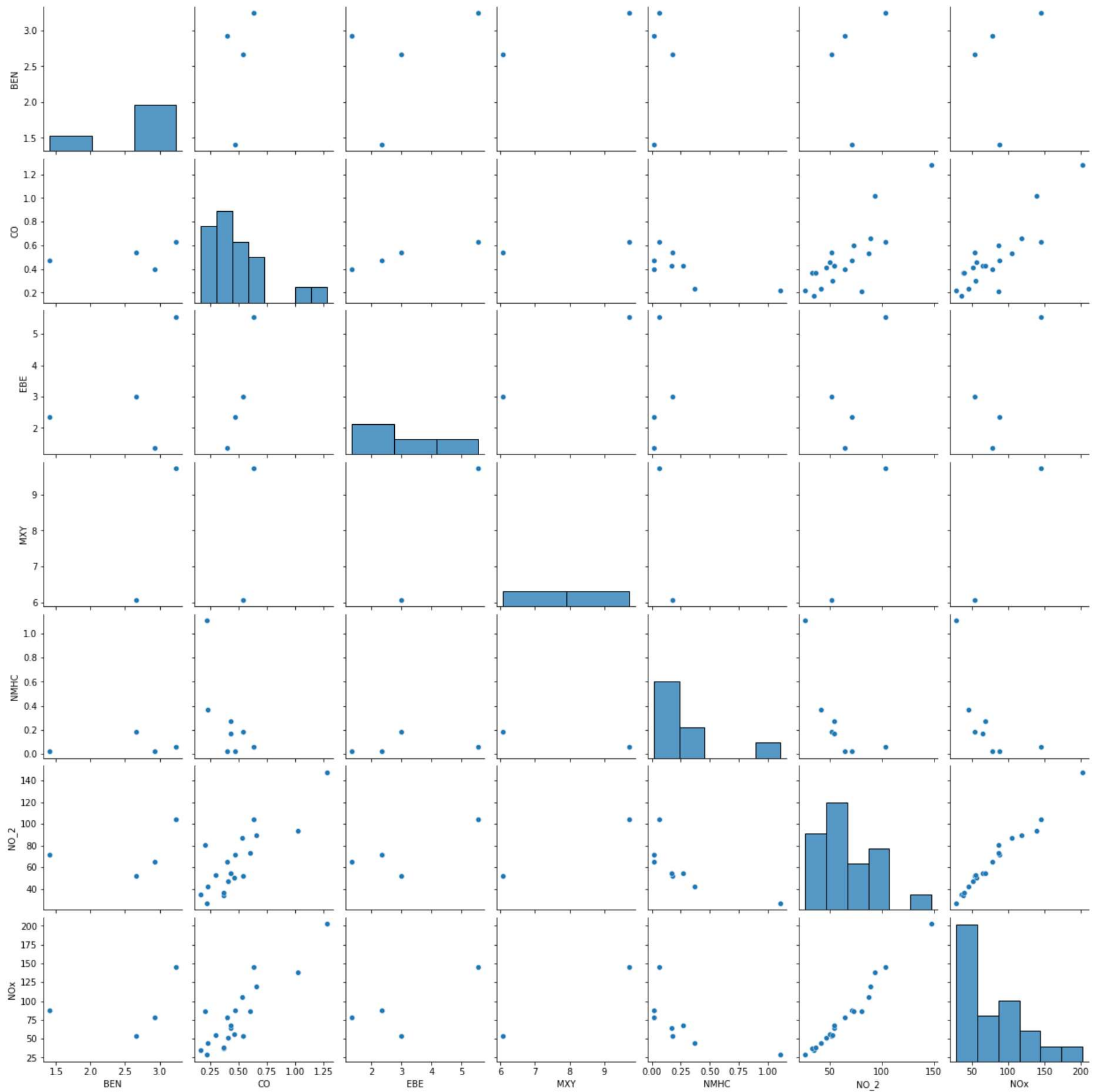


In [21]:
```
dd.plot.bar()
```
Out[21]: <AxesSubplot:>



In [22]:
```
sns.pairplot(dd)
```

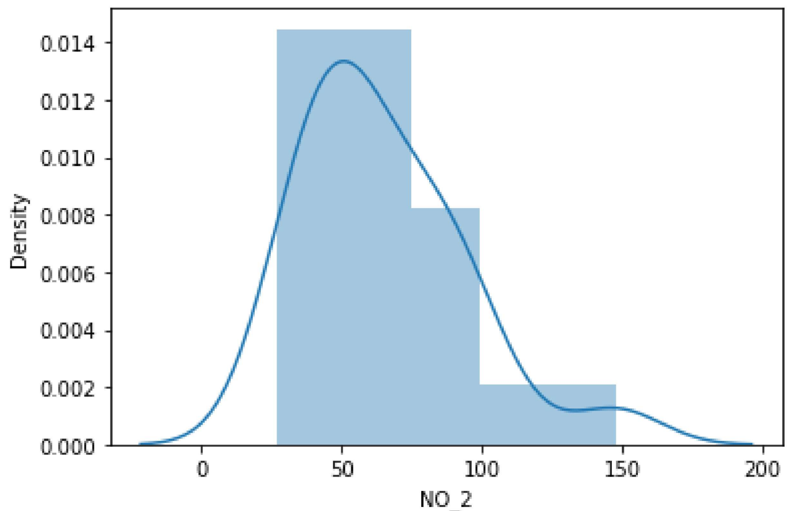
Out[22]: <seaborn.axisgrid.PairGrid at 0x20e04f434c0>



In [23]:
```
sns.distplot(dd['NO_2'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` 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[23]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



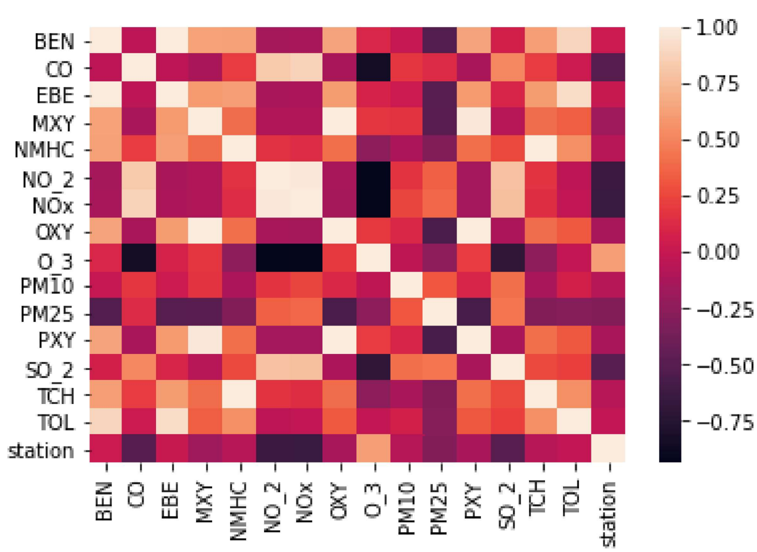
In [24]:
```
ds=data.fillna(20)
```

In [25]:
```
ssd=ds.head(20)
```

In [26]:
```
sd1=ssd[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
```

In [27]:
```
sns.heatmap(ssd.corr())
```

Out[27]: <AxesSubplot:>



LinearRegression()

In [28]:
```
x= ssd[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
y=ssd['station']
```

In [29]:
```
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 [30]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[30]:
```
LinearRegression()
```

In [31]:
```
print(lr.intercept_)
```

```
28079021.311735038
```

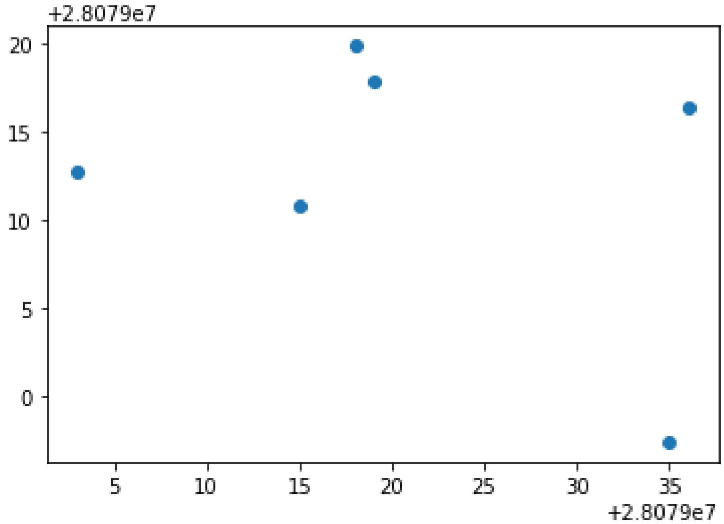
In [32]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[32]:

	Co-efficient
BEN	0.177250
CO	-27.148781
EBE	0.104523
MXY	0.448197
NMHC	0.146942
NO_2	-0.863192
NOx	0.593036

In [33]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[33]: <matplotlib.collections.PathCollection at 0x20e0c6a1940>



In [34]:
```
print(lr.score(x_test,y_test))
```

-1.4128224526907096

In [35]:
```
lr.score(x_test,y_test)
```

Out[35]: -1.4128224526907096

In [36]:
```
lr.score(x_train,y_train)
```

Out[36]: 0.4262556086742496

Ridge,Lasso

In [37]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [38]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[38]: Ridge(alpha=10)

In [39]:
```
dr.score(x_test,y_test)
```

Out[39]: 0.19335512136394362

In [40]:
```
dr.score(x_train,y_train)
```

Out[40]: 0.40126442001487095

In [41]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[41]:
```
Lasso(alpha=10)
```

In [42]:
```
la.score(x_test,y_test)
```

Out[42]:
```
0.26341352476396396
```

In [43]:
```
la.score(x_train,y_train)
```

Out[43]:
```
0.3762533769377412
```

ElasticNet

In [44]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[44]:
```
ElasticNet()
```

In [45]:
```
print(en.coef_)
```

```
[ 0.05686951 -0.          0.17718851 -0.23188718  0.06212572 -0.20299096
 -0.00942888]
```

In [46]:
```
print(en.intercept_)
```

```
28079028.679425504
```

In [47]:
```
prediction=en.predict(x_test)
```

In [48]:
```
print(en.score(x_test,y_test))
```

```
0.14866260315832935
```

In [49]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [50]:
```
from sklearn.linear_model import LogisticRegression
```

In [51]:
```
feature_matrix = ssd[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
target_vector=ssd['station']
```

In [52]:
```
feature_matrix.shape
```
Out[52]:
```
(20, 7)
```
In [53]:
```
target_vector.shape
```
Out[53]:
```
(20,)
```
In [54]:
```
from sklearn.preprocessing import StandardScaler
```
In [55]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```
In [56]:
```
logr= LogisticRegression()
logr.fit(fs,target_vector)
```
Out[56]:
```
LogisticRegression()
```
In [57]:
```
observation =[[1.2,2.3,3.3,4.3,5.3,6.3,7.3]]
```
In [58]:
```
prediction=logr.predict(observation)
print(prediction)
```
```
[28079009]
```
In [59]:
```
logr.classes_
```
Out[59]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079035, 28079036,
       28079039, 28079040], dtype=int64)
```
In [60]:
```
logr.predict_proba(observation)[0][0]
```
Out[60]:
```
0.026985460953367686
```
In [61]:
```
ged=data[['BEN','CO','EBE','MXY','NMHC','NO_2','NOx','OXY','O_3','PM10','PXY','
```
In [62]:
```
d=ged.fillna(20)
```
In [63]:
```
dg=d.head(100)
```
In [64]:
```
x=dg[['BEN','CO','EBE','MXY','NMHC','NO_2','NOx','OXY','O_3','PM10','PXY','SO_2
y=dg['station']
```
In [65]:
```
print(len(x))
print(len(y))
```
```
100
100
```

In [66]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [67]:
```python
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[67]:
```
RandomForestClassifier()
```

In [68]:
```python
paramets = {'max_depth':[1,2,3,4,5,6,7],
              'min_samples_leaf':[5,10,15,20,25,30,35],
              'n_estimators':[10,20,30,40,50,60,70]}
```

In [69]:
```python
from sklearn.model_selection import GridSearchCV
grid_search= GridSearchCV(estimator = rfc,param_grid=paramets,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:
666: UserWarning: The least populated class in y has only 1 members, which is
less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[69]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7],
                         'min_samples_leaf': [5, 10, 15, 20, 25, 30, 35],
                         'n_estimators': [10, 20, 30, 40, 50, 60, 70]},
             scoring='accuracy')
```

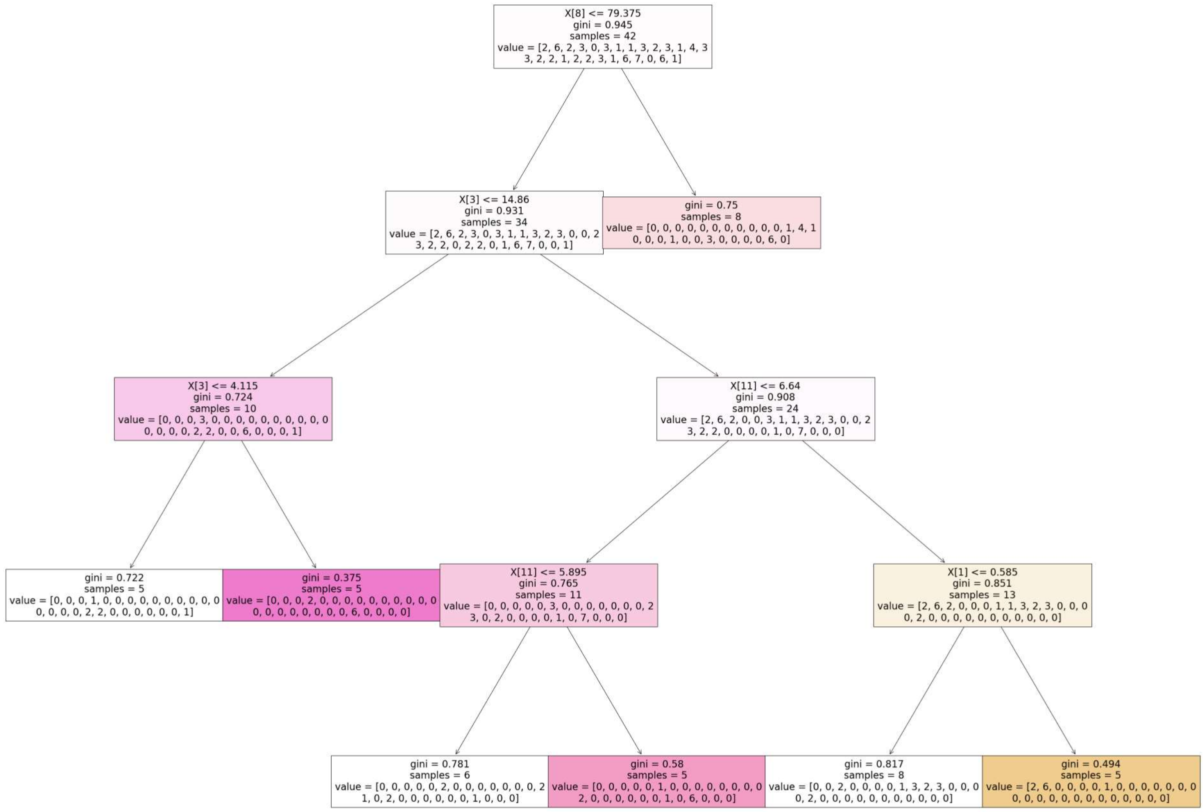
In [70]:
```python
grid_search.best_score_
```

Out[70]:
```
0.4285714285714286
```

In [71]:
```python
rfc_best=grid_search.best_estimator_
```

In [73]:
```python
from sklearn.tree import plot_tree
plt.figure(figsize=(50,40))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[73]:
```
[Text(1395.0, 1956.96, 'X[8] <= 79.375\ngini = 0.945\nsamples = 42\nvalue = 
[2, 6, 2, 3, 0, 3, 1, 1, 3, 2, 3, 1, 4, 3\n3, 2, 2, 1, 2, 2, 3, 1, 6, 7, 0, 
6, 1]'),
 Text(1141.3636363636363, 1522.0800000000002, 'X[3] <= 14.86\ngini = 0.931\ns
amples = 34\nvalue = [2, 6, 2, 3, 0, 3, 1, 1, 3, 2, 3, 0, 0, 2\n3, 2, 2, 0, 
2, 2, 0, 1, 6, 7, 0, 0, 1]'),
 Text(507.27272727272725, 1087.2, 'X[3] <= 4.115\ngini = 0.724\nsamples = 10
\nvalue = [0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2, 2, 0, 0, 
6, 0, 0, 0, 1]'),
 Text(253.63636363636363, 652.3200000000002, 'gini = 0.722\nsamples = 5\nvalu
e = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 
0, 0, 1]'),
 Text(760.9090909090909, 652.3200000000002, 'gini = 0.375\nsamples = 5\nvalue
= [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 
0, 0]'),
 Text(1775.4545454545455, 1087.2, 'X[11] <= 6.64\ngini = 0.908\nsamples = 24
\nvalue = [2, 6, 2, 0, 0, 3, 1, 1, 3, 2, 3, 0, 0, 2\n3, 2, 2, 0, 0, 0, 0, 1, 
0, 7, 0, 0, 0]'),
 Text(1268.181818181818, 652.3200000000002, 'X[11] <= 5.895\ngini = 0.765\nsa
mples = 11\nvalue = [0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 2\n3, 0, 2, 0, 0, 
0, 0, 1, 0, 7, 0, 0, 0]'),
 Text(1014.5454545454545, 217.44000000000005, 'gini = 0.781\nsamples = 6\nval
ue = [0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2\n1, 0, 2, 0, 0, 0, 0, 0, 0, 1, 
0, 0, 0]'),
 Text(1521.8181818181818, 217.44000000000005, 'gini = 0.58\nsamples = 5\nvalu
e = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\n2, 0, 0, 0, 0, 0, 0, 1, 0, 6, 
0, 0, 0]'),
 Text(2282.7272727272725, 652.3200000000002, 'X[1] <= 0.585\ngini = 0.851\nsa
mples = 13\nvalue = [2, 6, 2, 0, 0, 0, 1, 1, 3, 2, 3, 0, 0, 0\n0, 2, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2029.090909090909, 217.44000000000005, 'gini = 0.817\nsamples = 8\nvalu
e = [0, 0, 2, 0, 0, 0, 0, 1, 3, 2, 3, 0, 0, 0\n0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0]'),
 Text(2536.363636363636, 217.44000000000005, 'gini = 0.494\nsamples = 5\nvalu
e = [2, 6, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0]'),
 Text(1648.6363636363635, 1522.0800000000002, 'gini = 0.75\nsamples = 8\nvalu
e = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 1\n0, 0, 0, 1, 0, 0, 3, 0, 0, 0, 
0, 6, 0]')]
```



Conclusion : LinearRegression() 0.4262556086742496 HIGH RANGE

In [ ]: