# **Importing Libraries**

### In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

# **Importing Datasets**

### In [2]:

df=pd.read\_csv(r"C:\Users\user\Downloads\C10\_air\csvs\_per\_year\csvs(Dataset)\madrid\_2003.
df

### Out[2]:

<b>0</b> 0	2003- 03-01 01:00:00	NaN									
			1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55
<b>1</b> 0	2003- 03-01 1:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.
<b>2</b> 0	2003- 03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.:
<b>3</b> 0	2003- 03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.
<b>4</b> 0	2003- 03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.
<b>243979</b> 0	2003- 10-01 0:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.:
<b>243980</b> 0	2003- 10-01 0:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.
<b>243981</b> 0	2003- 10-01 0:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.
<b>243982</b> 0	2003- 10-01 0:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.
<b>243983</b> 0	2003- 10-01 0:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.
243984 ro	243984 rows × 16 columns										
4								•			

# **Data Cleaning and Data Preprocessing**

### In [3]:

df=df.dropna()

#### In [4]:

```
df.columns
```

```
Out[4]:
```

### In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33010 entries, 5 to 243983
Data columns (total 16 columns):
             Non-Null Count Dtype
    Column
    -----
             -----
---
0
    date
             33010 non-null object
 1
    BEN
             33010 non-null float64
 2
    CO
             33010 non-null float64
 3
    EBE
             33010 non-null float64
 4
    MXY
             33010 non-null float64
 5
             33010 non-null float64
    NMHC
 6
    NO_2
             33010 non-null float64
 7
    NOx
             33010 non-null float64
 8
    OXY
             33010 non-null float64
 9
    0 3
             33010 non-null float64
 10
    PM10
             33010 non-null float64
 11
    PXY
             33010 non-null float64
 12
    SO_2
             33010 non-null float64
 13
    TCH
             33010 non-null float64
 14
             33010 non-null float64
    TOL
15 station 33010 non-null int64
dtypes: float64(14), int64(1), object(1)
memory usage: 4.3+ MB
```

### In [7]:

```
data=df[['EBE', 'MXY', 'PXY']]
data
```

### Out[7]:

	EBE	MXY	PXY
5	9.83	21.49	7.94
23	3.43	7.08	2.62
27	5.75	10.88	4.24
33	10.63	24.73	8.93
51	3.20	7.08	2.70
243955	3.07	9.38	3.48
243957	3.88	10.86	3.89
243961	4.53	10.88	4.13
243979	2.01	3.17	1.20
243983	2.15	6.41	2.43

33010 rows × 3 columns

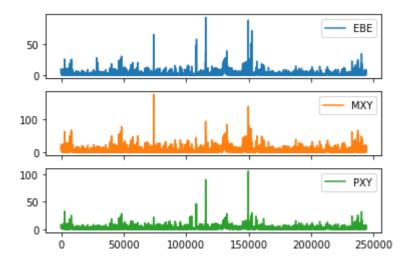
## Line chart

### In [8]:

```
data.plot.line(subplots=True)
```

### Out[8]:

array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



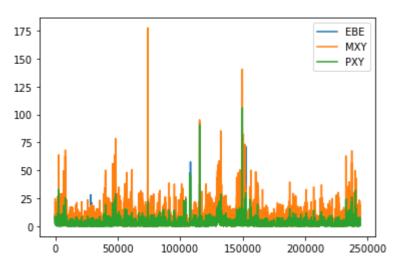
## Line chart

### In [9]:

data.plot.line()

### Out[9]:

### <AxesSubplot:>



## **Bar chart**

### In [10]:

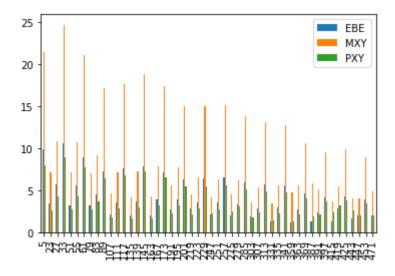
b=data[0:50]

### In [11]:

b.plot.bar()

### Out[11]:

### <AxesSubplot:>



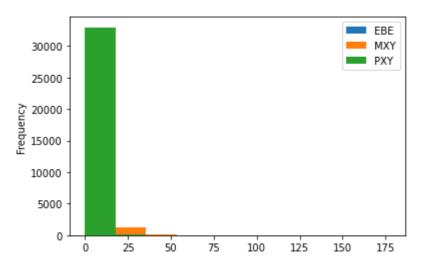
# Histogram

### In [12]:

data.plot.hist()

### Out[12]:

<AxesSubplot:ylabel='Frequency'>



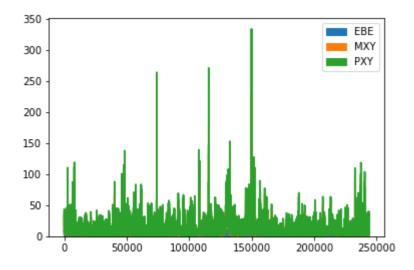
## Area chart

### In [13]:

data.plot.area()

### Out[13]:

<AxesSubplot:>



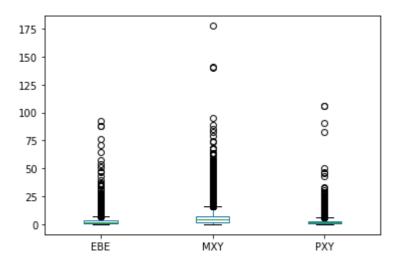
### **Box chart**

### In [14]:

data.plot.box()

### Out[14]:

### <AxesSubplot:>



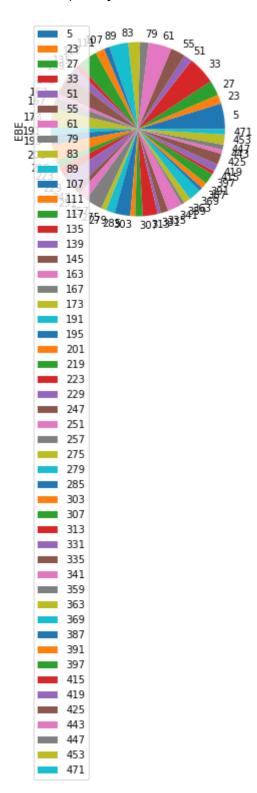
# Pie chart

```
In [16]:
```

```
b.plot.pie(y='EBE' )
```

### Out[16]:

<AxesSubplot:ylabel='EBE'>



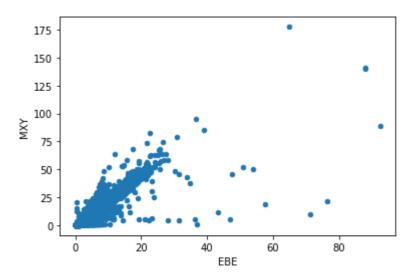
## **Scatter chart**

#### In [17]:

```
data.plot.scatter(x='EBE' ,y='MXY')
```

#### Out[17]:

<AxesSubplot:xlabel='EBE', ylabel='MXY'>



### In [18]:

```
df.info()
```

11

12

13

14

PXY

TCH

TOL

SO 2

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33010 entries, 5 to 243983
Data columns (total 16 columns):
#
     Column
              Non-Null Count Dtype
     _____
              ______
                              object
 0
     date
              33010 non-null
 1
     BEN
              33010 non-null
                              float64
 2
     CO
              33010 non-null
                              float64
 3
     EBE
              33010 non-null
                              float64
 4
     MXY
              33010 non-null
                              float64
 5
     NMHC
              33010 non-null
                              float64
 6
     NO 2
              33010 non-null
                              float64
 7
     NOx
              33010 non-null
                              float64
 8
     0XY
              33010 non-null
                              float64
 9
     0_3
              33010 non-null
                              float64
 10
     PM10
              33010 non-null
                              float64
```

station 33010 non-null dtypes: float64(14), int64(1), object(1)

33010 non-null

33010 non-null

33010 non-null

33010 non-null

float64 float64

float64

float64

int64

memory usage: 4.3+ MB

```
In [19]:
```

```
df.describe()
```

### Out[19]:

	BEN	СО	EBE	MXY	NMHC	NO_2
count	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000
mean	2.192633	0.759868	2.639726	5.838414	0.137177	57.328049
std	2.064160	0.545999	2.825194	6.267296	0.127863	31.811082
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.900000	0.430000	1.010000	1.880000	0.060000	34.529999
50%	1.610000	0.620000	1.890000	4.070000	0.110000	55.105000
75%	2.810000	0.930000	3.300000	7.530000	0.170000	76.160004
max	66.389999	7.920000	92.589996	177.600006	2.180000	342.700012
4						<b>&gt;</b>

### In [20]:

```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3', 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

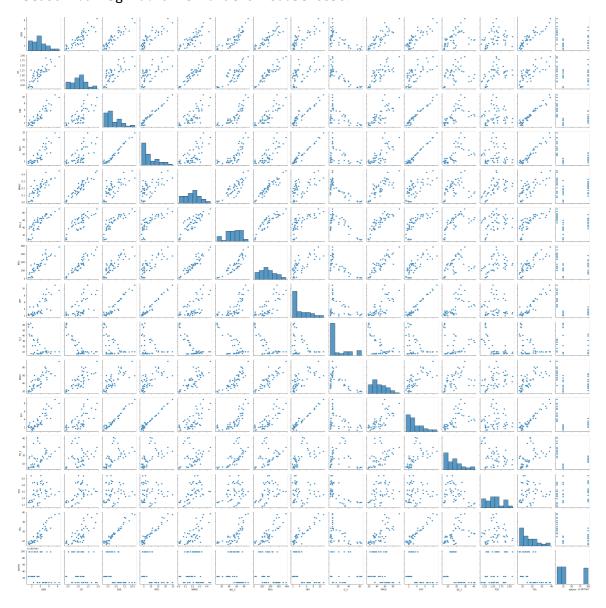
## **EDA AND VISUALIZATION**

### In [21]:

sns.pairplot(df1[0:50])

### Out[21]:

<seaborn.axisgrid.PairGrid at 0x18dd3ef6b50>



#### In [22]:

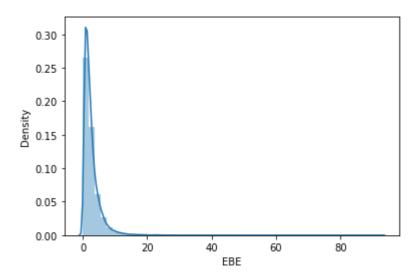
```
sns.distplot(df1['EBE'])
```

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[22]:

<AxesSubplot:xlabel='EBE', ylabel='Density'>

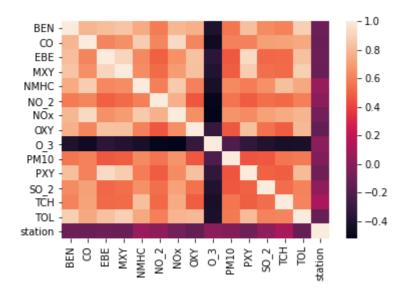


#### In [23]:

sns.heatmap(df1.corr())

#### Out[23]:

### <AxesSubplot:>



## TO TRAIN THE MODEL AND MODEL BULDING

```
In [24]:
```

### In [25]:

```
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)
```

## **Linear Regression**

### In [26]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

### Out[26]:

LinearRegression()

### In [27]:

```
lr.intercept_
```

#### Out[27]:

28079000.88287165

### In [28]:

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

### Out[28]:

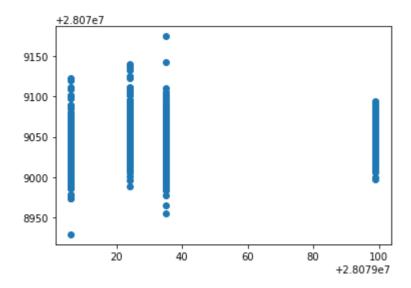
	Co-efficient		
BEN	1.692702		
со	-38.977600		
EBE	-1.867924		
MXY	0.174105		
NMHC	152.722698		
NO_2	0.163994		
NOx	-0.071702		
OXY	-1.299032		
O_3	-0.014141		
PM10	-0.050933		
PXY	2.076299		
SO_2	0.865766		
тсн	35.557182		
TOL	-0.922173		

### In [29]:

```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

### Out[29]:

<matplotlib.collections.PathCollection at 0x18de1c99c70>



## **ACCURACY**

```
8/3/23, 10:47 AM
                                             madrid 2003 - Jupyter Notebook
 In [30]:
 lr.score(x_test,y_test)
 Out[30]:
 0.1724179923348299
 In [31]:
 lr.score(x_train,y_train)
 Out[31]:
 0.1775053138791619
 Ridge and Lasso
 In [32]:
 from sklearn.linear_model import Ridge,Lasso
 In [33]:
 rr=Ridge(alpha=10)
 rr.fit(x_train,y_train)
 Out[33]:
 Ridge(alpha=10)
 Accuracy(Ridge)
 In [34]:
 rr.score(x_test,y_test)
 Out[34]:
 0.17104202048880912
```

```
In [35]:
rr.score(x_train,y_train)
Out[35]:
0.17647021880331226
In [36]:
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[36]:

Lasso(alpha=10)

```
In [37]:
```

```
la.score(x_train,y_train)
```

### Out[37]:

0.03743296904759286

# **Accuracy(Lasso)**

```
In [38]:
la.score(x_test,y_test)
Out[38]:
0.03248687905869285
```

## **Accuracy(Elastic Net)**

```
In [40]:
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
Out[40]:
ElasticNet()
In [41]:
en.coef_
Out[41]:
                                           , -0.01685342, 0.14109596,
array([ 0.
                 , -0.22243934, 0.
        0.15774441, -0.07187445, -1.12341392, -0.04543365,
                                                            0.08531711,
        0.33830415, 0.7394519, 1.57593375, -0.47188519)
In [42]:
en.intercept_
Out[42]:
28079037.19291707
In [43]:
```

prediction=en.predict(x\_test)

```
In [44]:
en.score(x_test,y_test)
Out[44]:
0.04581467200955813
```

### **Evaluation Metrics**

```
In [45]:
```

34.41706889545367

```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))

29.119161570718372
1184.5346313544042
```

# **Logistic Regression**

```
In [46]:
from sklearn.linear_model import LogisticRegression

In [47]:
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOX', 'OXY', 'O_3', 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']

In [48]:
feature_matrix.shape

Out[48]:
(33010, 14)

In [49]:
target_vector.shape

Out[49]:
(33010,)
In [50]:
```

from sklearn.preprocessing import StandardScaler

```
In [51]:
fs=StandardScaler().fit_transform(feature_matrix)
In [52]:
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
Out[52]:
LogisticRegression(max_iter=10000)
In [53]:
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
In [54]:
prediction=logr.predict(observation)
print(prediction)
[28079035]
In [55]:
logr.classes_
Out[55]:
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
In [56]:
logr.score(fs,target_vector)
Out[56]:
0.7584974250227204
In [57]:
logr.predict_proba(observation)[0][0]
Out[57]:
2.3306153265290618e-23
In [58]:
logr.predict_proba(observation)
Out[58]:
```

array([[2.33061533e-23, 1.44436075e-55, 1.00000000e+00, 6.68457491e-16]])

### **Random Forest**

```
In [59]:
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
In [60]:
```

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

#### Out[60]:

RandomForestClassifier()

### In [61]:

### In [62]:

```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

### Out[62]:

#### In [63]:

```
grid_search.best_score_
```

### Out[63]:

0.7254074069545836

#### In [64]:

```
rfc_best=grid_search.best_estimator_
```

### In [65]:

```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],f
```

Out[65]:

```
[Text(2222.7000000000003, 1993.2, 'NOx <= 38.185\ngini = 0.749\nsamples =
14550\nvalue = [5307, 5726, 5973, 6101]\nclass = d'),
 Text(1171.8000000000002, 1630.8000000000002, 'CO <= 0.385 \setminus ini = 0.488 \setminus ini = 0.48
samples = 2668\nvalue = [198, 2886, 639, 467]\nclass = b'),
 Text(595.2, 1268.4, 'TOL <= 1.385\ngini = 0.679\nsamples = 1400\nvalue =
[174, 987, 596, 443]\nclass = b'),
 Text(297.6, 906.0, 'NMHC <= 0.035\ngini = 0.301\nsamples = 379\nvalue =
[4, 491, 84, 17]\nclass = b'),
 Text(148.8, 543.599999999999, 'SO 2 <= 5.48\ngini = 0.545\nsamples = 78
\nvalue = [4, 39, 72, 7]\nclass = c'),
 9, 0, 0]\nclass = b'),
 Text(223.2000000000000, 181.199999999982, 'gini = 0.202\nsamples = 51
\nvalue = [2, 0, 72, 7] \setminus class = c'),
 amples = 301\nvalue = [0, 452, 12, 10]\nclass = b'),
 Text(372.0, 181.199999999999, 'gini = 0.005\nsamples = 280\nvalue = [0,
436, 0, 1]\nclass = b'),
 Text(520.800000000001, 181.19999999999982, 'gini = 0.649 \nsamples = 21 \n
value = [0, 16, 12, 9] \setminus class = b'),
 Text(892.800000000001, 906.0, 'NO_2 <= 17.925 | mgini = 0.721 | msamples = 1
021\nvalue = [170, 496, 512, 426]\nclass = c'),
 Text(744.0, 543.59999999999, 'SO_2 <= 6.335\ngini = 0.602\nsamples = 33
2\nvalue = [63, 311, 93, 68]\nclass = b'),
 Text(669.6, 181.1999999999982, 'gini = 0.314\nsamples = 199\nvalue = [2
7, 254, 25, 4]\nclass = b'),
 Text(818.4000000000001, 181.1999999999982, 'gini = 0.738\nsamples = 133
\nvalue = [36, 57, 68, 64]\nclass = c'),
 samples = 689\nvalue = [107, 185, 419, 358]\nclass = c'),
 Text(967.2, 181.1999999999982, 'gini = 0.377 \nsamples = 97 \nvalue = [29, 18]
109, 5, 0]\nclass = b'),
 Text(1116.0, 181.199999999999, 'gini = 0.637\nsamples = 592\nvalue = [7
8, 76, 414, 358]\nclass = c'),
 Text(1748.4, 1268.4, 'SO_2 <= 8.595\ngini = 0.089\nsamples = 1268\nvalue
= [24, 1899, 43, 24]\nclass = b'),
 Text(1488.0, 906.0, 'PXY <= 1.015\ngini = 0.028\nsamples = 1197\nvalue =
[2, 1850, 8, 17]\nclass = b'),
 Text(1339.2, 543.59999999999, 'NO_2 <= 34.87\ngini = 0.018\nsamples = 1
136\nvalue = [1, 1773, 5, 10]\nclass = b'),
 Text(1264.8000000000002, 181.1999999999982, 'gini = 0.016\nsamples = 112
6\nvalue = [1, 1762, 3, 10]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.26\nsamples = 10\n
value = [0, 11, 2, 0] \setminus class = b'),
 Text(1636.8000000000002, 543.59999999999, 'MXY <= 2.51\ngini = 0.227\ns
amples = 61\nvalue = [1, 77, 3, 7]\nclass = b'),
 11, 0, 7]\nclass = b'),
 66, 3, 0]\nclass = b'),
 Text(2008.800000000000, 906.0, 'MXY <= 2.34\ngini = 0.674\nsamples = 71
\nvalue = [22, 49, 35, 7]\nclass = b'),
 Text(1934.4, 543.59999999999, 'EBE <= 0.755\ngini = 0.563\nsamples = 54
\nvalue = [0, 43, 30, 7] \setminus class = b'),
 Text(1860.000000000000, 181.199999999982, 'gini = 0.506\nsamples = 32
\nvalue = [0, 12, 29, 4] \setminus class = c'),
 Text(2008.800000000000, 181.199999999982, 'gini = 0.207\nsamples = 22
\nvalue = [0, 31, 1, 3] \setminus class = b'),
 Text(2083.200000000003, 543.59999999999, 'gini = 0.5\nsamples = 17\nva
lue = [22, 6, 5, 0] \setminus ass = a'),
 Text(3273.6000000000004, 1630.8000000000002, 'SO 2 <= 6.625 \cdot injini = 0.736
```

```
\nsamples = 11882\nvalue = [5109, 2840, 5334, 5634]\nclass = d'),
 Text(2678.4, 1268.4, 'NMHC <= 0.065\ngini = 0.64\nsamples = 1802\nvalue =
[956, 1352, 512, 71]\nclass = b'),
 Text(2380.8, 906.0, 'MXY <= 2.085\ngini = 0.587\nsamples = 774\nvalue =
[719, 185, 317, 37] \setminus (ass = a'),
 Text(2232.0, 543.599999999999, 'CO <= 0.425\ngini = 0.645\nsamples = 178
\nvalue = [52, 114, 117, 5]\nclass = c'),
 Text(2157.600000000004, 181.1999999999982, 'gini = 0.541\nsamples = 105
\nvalue = 44, 15, 103, 5]\n ss = c'),
 99, 14, 0]\nclass = b'),
 \text{Tex}(2529.600 - 000000004, 543.599 - 99999 - 99, 'TCH = 1.265 \cdot igini = 0.478 \cdot n
samples = 596\nvalue = [667, 71, 200, 32]\nclass = a'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.081 \nsamples = 356
\nvalue = [570, 0, 25, 0]\nclass = a'),
 Text(2604.0, 181.19999999999999, 'gini = 0.672\nsamples = 240\nvalue = /[9
Text(2976.0, 906.0, 'OXY <= 2.27\ngini = 0.454\nsamples = 1028\nvalue =
[237, 1167, 195, 34]\nclass = b'),
 amples = 795\nvalue = [17, 1085, 126, 29]\nclass = b'),
6, 1080, 70, 27]\nclass = b'),
 Text(2901.600000000004, 181.199999999982, 'gini = 0.227\nsamples = 39
Accuracy, 543.599999999999, 'CO <= 0.615\ngini = 0.576\nsamples = 233
\nvalue = [220, 82, 69, 5]\nclass = a'),
L\bar{l}_{near}^{+} Regression.0.1773033138794619 gini = 0.603\nsamples = 71\nvalue = [9,
42, 55, 5]\nclass = c'),
 Text(3199.2000000000003,
                                            181.1999999999982, 'gini = 0.34\nsamples = 162
Ridge Regression: 4003743296904759386 a'),
 Text(3868.8, 1268.4, 'EBE <= 3.635\ngini = 0.713\nsamples = 10080\nvalue
£as$5Reg1e88ioA.8283248687905869285 d'),
 Text(3571.2000000000003, 906.0, 'NO_2 \le 25.495 \mid = 0.688 \mid = 0.6
7050\nvalue = [1696, 1219, 3671, 4644]\nclass = d'),
Elastic Net Regression 0.04581467200955813 = 1.015\ngini = 0.151\nsamples = 13
5\nvalue = [203, 15, 0, 3]\nclass = a'),
Lbeystie 348g People 0.758497423022720299999982, 'gini = 0.564\nsamples = 18
\nvalue = [17, 11, 0, 3] \setminus ass = a'),
samples = 6915\nvalue = [1493, 1204, 3671, 4641]\nclass = d'),
Logistic Regression is suitable for this 2 dataset = 0.381 \nsamples = 352
\nvalue = [6, 67, 418, 51]\nclass = c'),
 Text(3794.4, 181.199999999999, 'gini = 0.679\nsamples = 6563\nvalue =
[1487, 1137, 3253, 4590]\nclass = d'),
 Text(4166.400000000001, 906.0, 'PXY <= 3.435\ngini = 0.64\nsamples = 3030
\nvalue = [2457, 269, 1151, 919]\nclass = a'),
 Text(4017.600000000004, 543.599999999999, 'BEN <= 2.345\ngini = 0.697\n
samples = 374\nvalue = [169, 129, 65, 264]\nclass = d'),
 Text(3943.2000000000003. 181.199999999982. 'gini = 0.711\nsamples = 98
```