3.08.2023(P2)

In [79]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [80]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[80]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002
...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000

217296 rows × 16 columns

In [81]:
```
df=df.dropna()
```

In [82]:
```
df=df.head(20)
```

In [83]:
```
df.columns
```

Out[83]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 1 to 124
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     20 non-null     object 
 1   BEN      20 non-null     float64
 2   CO       20 non-null     float64
 3   EBE      20 non-null     float64
 4   MXY      20 non-null     float64
 5   NMHC     20 non-null     float64
 6   NO_2     20 non-null     float64
 7   NOx      20 non-null     float64
 8   OXY      20 non-null     float64
 9   O_3      20 non-null     float64
 10  PM10     20 non-null     float64
 11  PXY      20 non-null     float64
 12  SO_2     20 non-null     float64
 13  TCH      20 non-null     float64
 14  TOL      20 non-null     float64
 15  station  20 non-null     int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 2.7+ KB
```

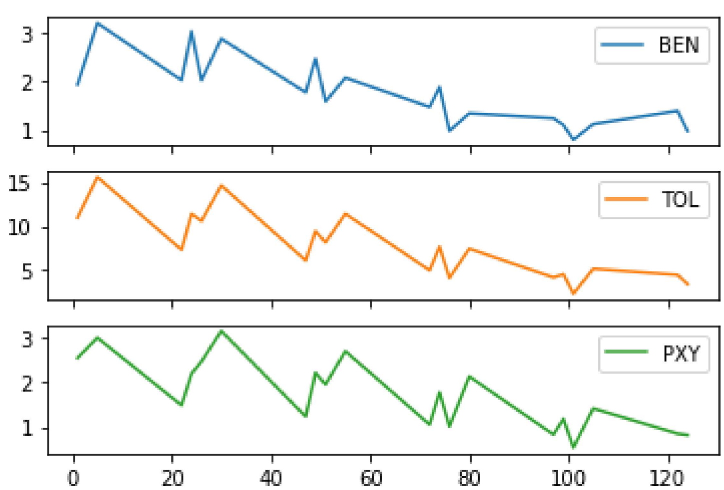
In [85]:
```
data=df[['BEN', 'TOL', 'PXY']]
data
```

Out[85]:

	BEN	TOL	PXY
1	1.93	10.98	2.53
5	3.19	15.60	2.98
22	2.02	7.32	1.48
24	3.02	11.42	2.18
26	2.02	10.60	2.45
30	2.87	14.65	3.13
47	1.77	6.11	1.23
49	2.46	9.44	2.20
51	1.58	8.16	1.94
55	2.07	11.42	2.68
72	1.47	4.99	1.05
74	1.88	7.67	1.77
76	0.98	4.13	1.01
80	1.34	7.45	2.12
97	1.24	4.19	0.83
99	1.10	4.56	1.18
101	0.80	2.33	0.54
105	1.12	5.17	1.41
122	1.39	4.48	0.85
124	0.98	3.44	0.82

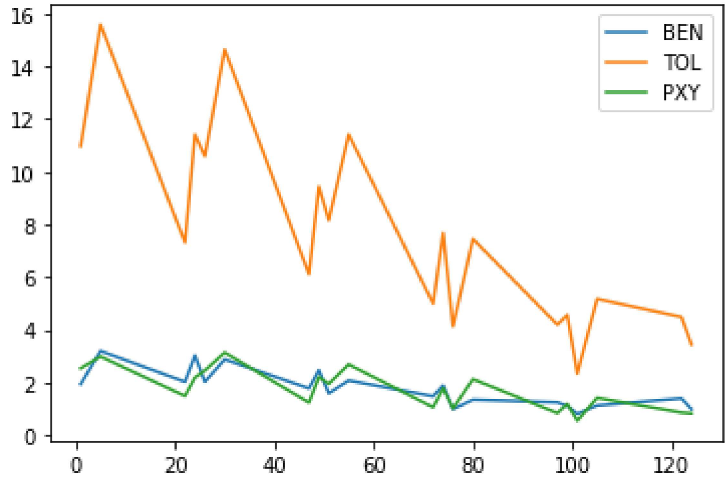
In [86]:
```
data.plot.line(subplots=True)
```

Out[86]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [87]:
```
data.plot.line()
```

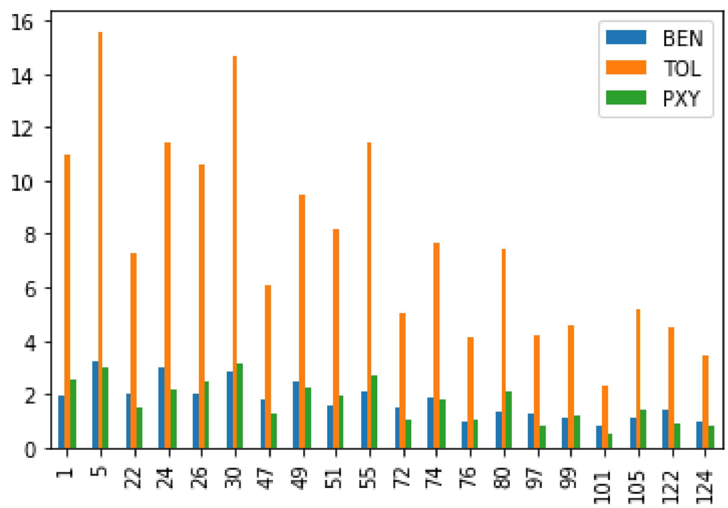
Out[87]: <AxesSubplot:>



In [88]:
```
b=data[0:50]
```

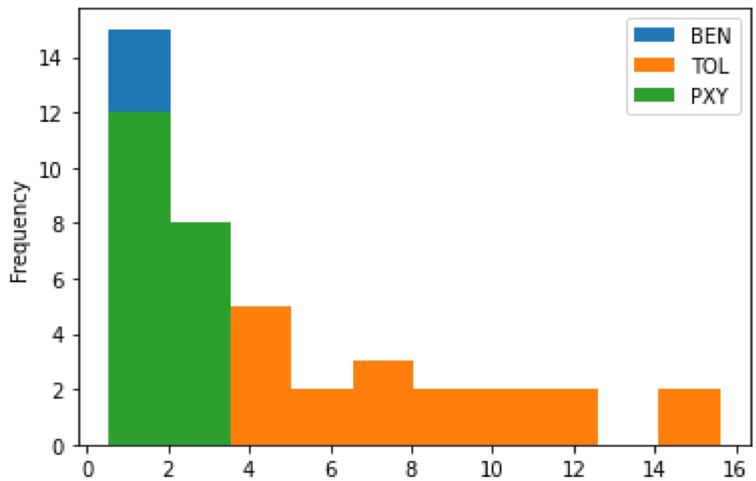
In [89]:
```
b.plot.bar()
```

Out[89]: <AxesSubplot:>



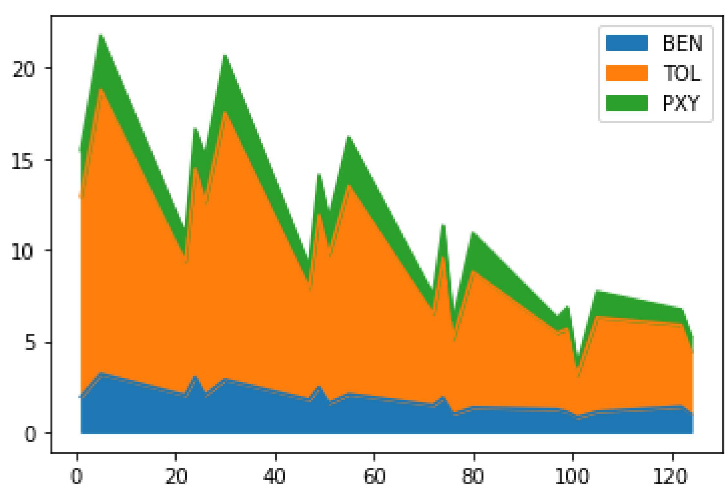
In [90]:
```
data.plot.hist()
```

Out[90]: <AxesSubplot:ylabel='Frequency'>



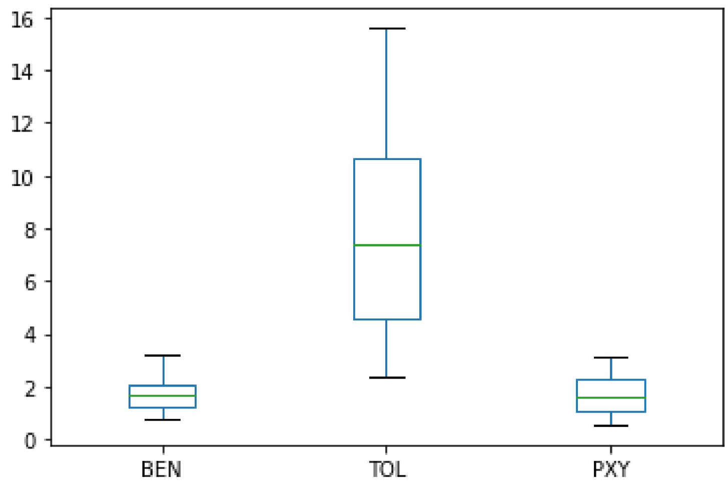
In [91]:
```
data.plot.area()
```

Out[91]: <AxesSubplot:>



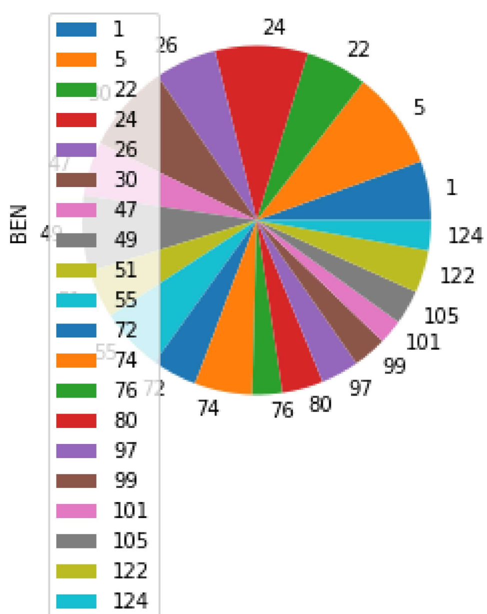
In [92]:
```
data.plot.box()
```

Out[92]: <AxesSubplot:>



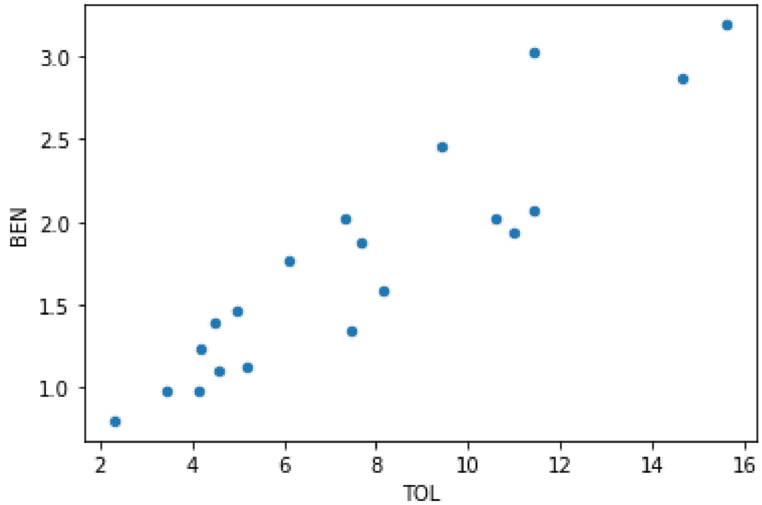
In [93]:
```
b.plot.pie(y='BEN' )
```

Out[93]: <AxesSubplot:ylabel='BEN'>



In [94]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[94]: <AxesSubplot:xlabel='TOL', ylabel='BEN'>



In [95]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 1 to 124
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     20 non-null     object 
 1   BEN      20 non-null     float64
 2   CO       20 non-null     float64
 3   EBE      20 non-null     float64
 4   MXY      20 non-null     float64
 5   NMHC     20 non-null     float64
 6   NO_2     20 non-null     float64
 7   NOx      20 non-null     float64
 8   OXY      20 non-null     float64
 9   O_3      20 non-null     float64
 10  PM10     20 non-null     float64
 11  PXY      20 non-null     float64
 12  SO_2     20 non-null     float64
 13  TCH      20 non-null     float64
 14  TOL      20 non-null     float64
 15  station  20 non-null     int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 2.7+ KB
```

In [96]:
```
df.describe()
```

Out[96]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
count	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
mean	1.761500	0.508000	1.781500	4.235500	0.094500	71.100000	96.450999	1.912500
std	0.700115	0.273546	0.829358	2.019235	0.057809	26.411393	50.134432	0.892659
min	0.800000	0.120000	0.490000	1.240000	0.000000	22.080000	24.139999	0.460000
25%	1.210000	0.272500	1.115000	2.477500	0.045000	44.017499	54.509998	1.257500
50%	1.675000	0.525000	1.660000	3.985000	0.115000	75.925003	88.530003	1.745000
75%	2.032500	0.712500	2.342500	5.557500	0.130000	92.164999	136.874996	2.580000
max	3.190000	1.040000	3.450000	7.920000	0.210000	113.699997	195.399994	3.730000

In [97]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [*]:
```
sns.pairplot(df1[0:50])
```

Out[98]: <seaborn.axisgrid.PairGrid at 0x19113de65b0>

In [*]:
```
sns.distplot(df1['station'])
```

In [*]:
```
sns.heatmap(df1.corr())
```

In [*]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [*]:
```
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 [*]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

In [*]:
```
lr.intercept_
```

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

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

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

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

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

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

In [*]:
```
rr.score(x_test,y_test)
```

In [*]:
```
rr.score(x_train,y_train)
```

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

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

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

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

In [*]:
```
en.coef_
```

In [*]:
```
en.intercept_
```

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

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

In [*]:
```
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)))
```

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

In [*]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

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

Out[56]:
```
(20,)
```

In [57]:
```
from sklearn.preprocessing import StandardScaler
```

In [58]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [59]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[59]:
```
LogisticRegression(max_iter=10000)
```

In [60]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [61]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [62]:
```
logr.score(fs,target_vector)
```

Out[62]:
```
1.0
```

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

Out[63]:
```
0.021267025366369423
```

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

Out[64]:
```
array([[2.12670254e-02, 7.21980940e-16, 4.47700636e-08, 9.78732930e-01]])
```

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [66]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[66]:
```
RandomForestClassifier()
```

In [67]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [68]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[68]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

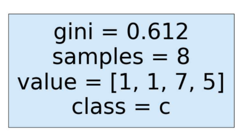
In [69]:
```
grid_search.best_score_
```

Out[69]:
```
0.3571428571428571
```

In [70]:
```
rfc_best=grid_search.best_estimator_
```

In [74]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,5))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b',
```

Out[74]:
```
[Text(1395.0, 135.9, 'gini = 0.612\nsamples = 8\nvalue = [1, 1, 7, 5]\nclass
= c')]
```



Conclusion

Linear Regression =1.0

Ridge Regression =0.6834711491207041

Lasso Regression =0.6132115096399764

ElasticNet Regression =-2.6934722951520897

Logistic Regression =-2.6934722951520897

Randomforest =-2.69347229515208971

Logistic Regression is suitable for this dataset

In [ ]:

```

```