In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as py
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
df=pd.read_csv(r"D:\New folder\madrid_2018.csv")
df
```

Out[5]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	T
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	N
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
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	N
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	N
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	N
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	N
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	N
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	N
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	N

69096 rows × 16 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [4]:

```
df1=df.fillna(value=0)
df1
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	0.00	0.37	0.00	0.00	0.00	58.400002	87.150002	0.00	34.529999	105.0
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	0.00	0.28	0.00	0.00	0.00	50.660000	61.380001	0.00	46.310001	100.0
3	2001-08-01 01:00:00	0.00	0.47	0.00	0.00	0.00	69.790001	73.449997	0.00	40.650002	69.7
4	2001-08-01 01:00:00	0.00	0.39	0.00	0.00	0.00	22.830000	24.799999	0.00	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	0.00	0.00	0.00	73.000000	264.399994	0.00	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	0.00	0.13	71.080002	129.300003	0.00	13.460000	26.8
217869	2001-04-01 00:00:00	0.49	1.09	0.00	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	0.00	80.019997	197.000000	2.58	5.840000	37.8
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.3

217872 rows × 16 columns

In [5]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      217872 non-null  float64
 2   CO       217872 non-null  float64
 3   EBE      217872 non-null  float64
 4   MXY      217872 non-null  float64
 5   NMHC     217872 non-null  float64
 6   NO_2     217872 non-null  float64
 7   NOx      217872 non-null  float64
 8   OXY      217872 non-null  float64
 9   O_3      217872 non-null  float64
 10  PM10     217872 non-null  float64
 11  PXY      217872 non-null  float64
 12  SO_2     217872 non-null  float64
 13  TCH      217872 non-null  float64
 14  TOL      217872 non-null  float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [6]:

```
df1.columns
```

Out[6]:

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

In [7]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
df2
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	0.00	0.37	0.00	0.00	0.00	58.400002	87.150002	0.00	34.529999	105.000000
1	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998
2	0.00	0.28	0.00	0.00	0.00	50.660000	61.380001	0.00	46.310001	100.099998
3	0.00	0.47	0.00	0.00	0.00	69.790001	73.449997	0.00	40.650002	69.779999
4	0.00	0.39	0.00	0.00	0.00	22.830000	24.799999	0.00	66.309998	75.180000
...	...	...	...	...	...	...	...	...	...	...
217867	10.45	1.81	0.00	0.00	0.00	73.000000	264.399994	0.00	5.200000	47.880001
217868	5.20	0.69	4.56	0.00	0.13	71.080002	129.300003	0.00	13.460000	26.809999
217869	0.49	1.09	0.00	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000
217870	5.62	1.01	5.04	11.38	0.00	80.019997	197.000000	2.58	5.840000	37.889999
217871	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999

In [ ]:
```
sns.pairplot(df2)
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2e2bb15a040>
```

In [ ]:
```
sns.distplot(df2['station'])
```

In [ ]:
```
x=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'O_3','PM10', 'PXY',
y=df2['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)
```

linear

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

In [ ]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

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

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

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

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

Ridge

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)
```

Lasso

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

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

elasticnet

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

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

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

In [ ]:
```
print(en.predict(x_test))
```

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

logistic

```
In [ ]: feature_matrix=df2.iloc[:,0:14]
        target_vector=df2.iloc[:,-1]
```

```
In [ ]: feature_matrix=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'O_3','P
        y=df2['station']
```

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

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

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

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

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

```
In [ ]: observation=[[1.4,2.3,5.0,11,12,13,14,15,4,5,7,6,7,13]]
```

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

```
In [ ]: logr.classes_
```

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

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

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

random forest

```
In [ ]: from sklearn.ensemble import RandomForestClassifier
        from sklearn.tree import plot_tree
```

```
In [ ]: x=df2.drop('station',axis=1)
        y=df2['station']
```

```
In [ ]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.70)
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
from sklearn.model_selection import GridSearchCV
```

In [ ]:

```
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='a
grid_search.fit(x_train,y_train)
```

In [ ]:

```
grid_search.best_score_
```

In [ ]:

```
rfc_best =grid_search.best_estimator_
```

In [ ]:

```
py.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

conclusion

The bestfit model is Logistic Regression with score of 0.9102362855254461

In [ ]:

```
```