```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as py
        import seaborn as sns
        from sklearn.linear_model import LogisticRegression
```

```
In [2]: df=pd.read_csv(r"D:\New folder\madrid_2018.csv")
        df
```

Out[2]:

| | 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 [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   date     217296 non-null  object
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

```
In [4]: df1=df.fillna(value=0)
        df1
```

Out[4]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2002-04-01 01:00:00 | 0.00 | 1.39 | 0.00 | 0.00 | 0.00 | 145.100006 | 352.100006 | 0.00 | 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 | 0.00 | 0.80 | 0.00 | 0.00 | 0.00 | 103.699997 | 134.000000 | 0.00 | 13.01 | 28.440001 |
| 3 | 2002-04-01 01:00:00 | 0.00 | 1.61 | 0.00 | 0.00 | 0.00 | 97.599998 | 268.000000 | 0.00 | 5.12 | 42.180000 |
| 4 | 2002-04-01 01:00:00 | 0.00 | 1.90 | 0.00 | 0.00 | 0.00 | 92.089996 | 237.199997 | 0.00 | 7.28 | 76.330002 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 217291 | 2002-11-01 00:00:00 | 4.16 | 1.14 | 0.00 | 0.00 | 0.00 | 81.080002 | 265.700012 | 0.00 | 7.21 | 36.750000 |
| 217292 | 2002-11-01 00:00:00 | 3.67 | 1.73 | 2.89 | 0.00 | 0.38 | 113.900002 | 373.100006 | 0.00 | 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 | 0.00 | 149.800003 | 202.199997 | 1.00 | 5.75 | 0.000000 |
| 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 [5]:  df1.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 217296 entries, 0 to 217295
         Data columns (total 16 columns):
          #   Column   Non-Null Count    Dtype
         ---  ------   --------------    -----
          0   date     217296 non-null   object
          1   BEN      217296 non-null   float64
          2   CO       217296 non-null   float64
          3   EBE      217296 non-null   float64
          4   MXY      217296 non-null   float64
          5   NMHC     217296 non-null   float64
          6   NO_2     217296 non-null   float64
          7   NOx      217296 non-null   float64
          8   OXY      217296 non-null   float64
          9   O_3      217296 non-null   float64
          10  PM10     217296 non-null   float64
          11  PXY      217296 non-null   float64
          12  SO_2     217296 non-null   float64
          13  TCH      217296 non-null   float64
          14  TOL      217296 non-null   float64
          15  station  217296 non-null   int64
         dtypes: float64(14), int64(1), object(1)
         memory usage: 26.5+ 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 | PXY |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.00 | 1.39 | 0.00 | 0.00 | 0.00 | 145.100006 | 352.100006 | 0.00 | 6.54 | 41.990002 | 0.00 |
| 1 | 1.93 | 0.71 | 2.33 | 6.20 | 0.15 | 98.150002 | 153.399994 | 2.67 | 6.85 | 20.980000 | 2.53 |
| 2 | 0.00 | 0.80 | 0.00 | 0.00 | 0.00 | 103.699997 | 134.000000 | 0.00 | 13.01 | 28.440001 | 0.00 |
| 3 | 0.00 | 1.61 | 0.00 | 0.00 | 0.00 | 97.599998 | 268.000000 | 0.00 | 5.12 | 42.180000 | 0.00 |
| 4 | 0.00 | 1.90 | 0.00 | 0.00 | 0.00 | 92.089996 | 237.199997 | 0.00 | 7.28 | 76.330002 | 0.00 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 217291 | 4.16 | 1.14 | 0.00 | 0.00 | 0.00 | 81.080002 | 265.700012 | 0.00 | 7.21 | 36.750000 | 0.00 |
| 217292 | 3.67 | 1.73 | 2.89 | 0.00 | 0.38 | 113.900002 | 373.100006 | 0.00 | 5.66 | 63.389999 | 0.00 |
| 217293 | 1.37 | 0.58 | 1.17 | 2.37 | 0.15 | 65.389999 | 107.699997 | 1.30 | 9.11 | 9.640000 | 0.94 |
| 217294 | 4.51 | 0.91 | 4.83 | 10.99 | 0.00 | 149.800003 | 202.199997 | 1.00 | 5.75 | 0.000000 | 5.52 |
| 217295 | 3.11 | 1.17 | 3.00 | 7.77 | 0.26 | 80.110001 | 180.300003 | 2.25 | 7.38 | 29.240000 | 3.35 |

```
In [8]: sns.pairplot(df2)
```

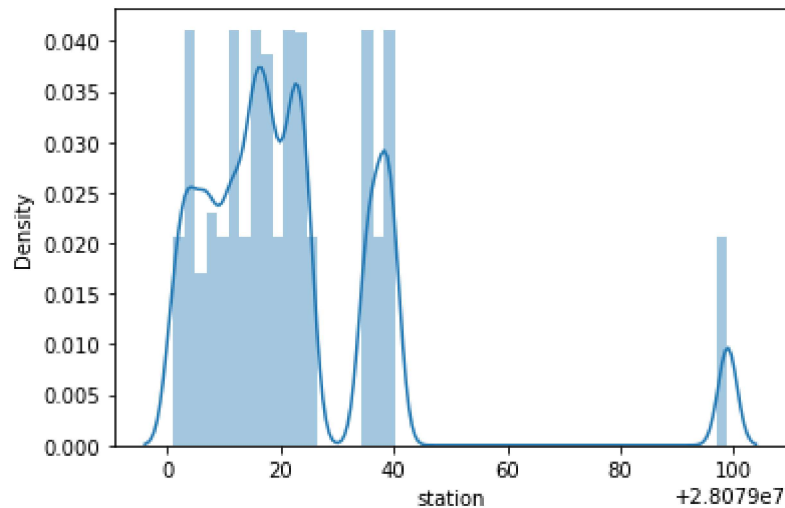Out[8]: <seaborn.axisgrid.PairGrid at 0x20a881eaa90>
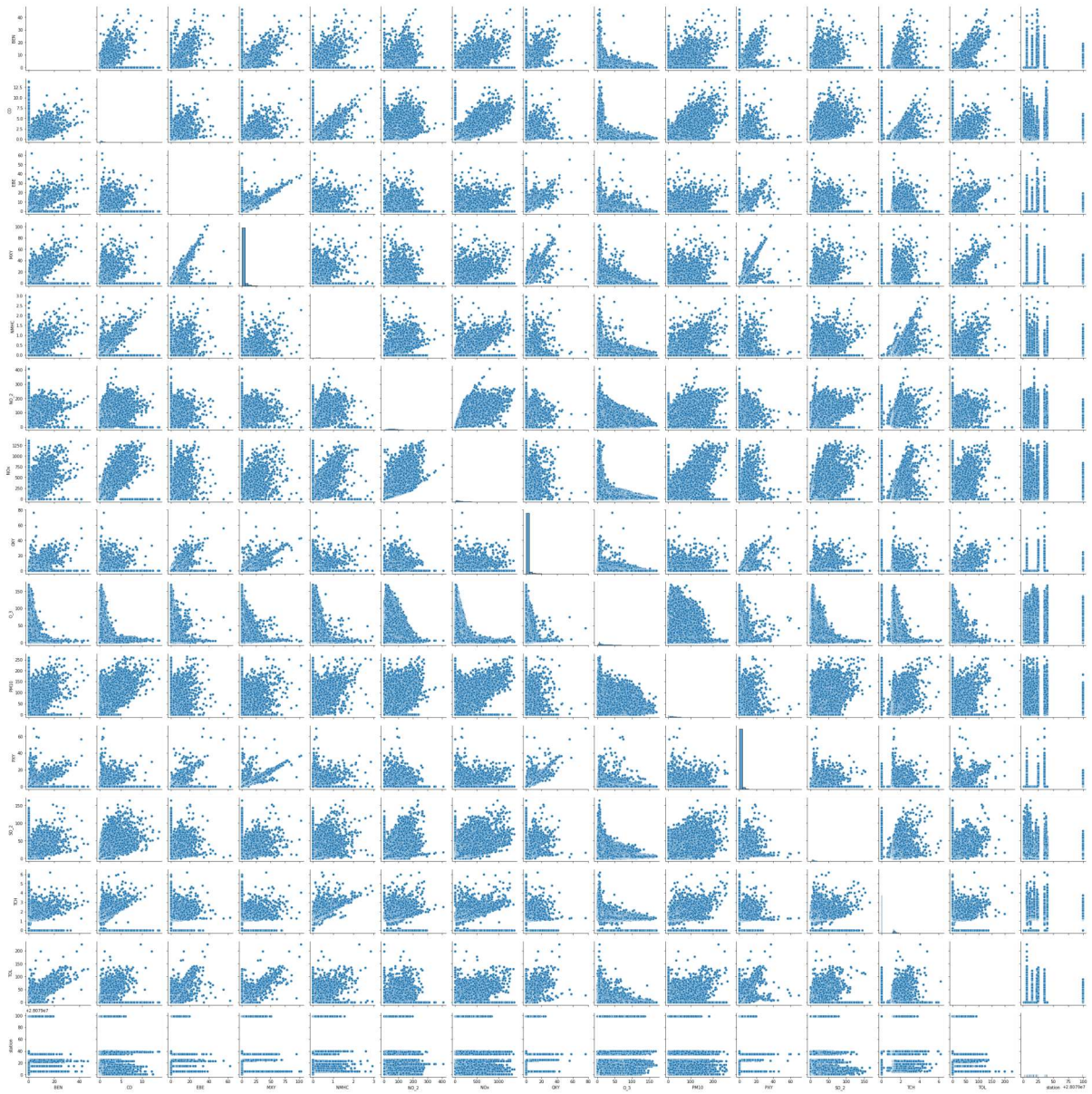
```
In [9]: sns.distplot(df2['station'])

        C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
        ureWarning: `distplot` is a deprecated function and will be removed in a futu
        re version. Please adapt your code to use either `displot` (a figure-level fu
        nction with similar flexibility) or `histplot` (an axes-level function for hi
        stograms).
          warnings.warn(msg, FutureWarning)

Out[9]: <AxesSubplot:xlabel='station', ylabel='Density'>
```



```
In [10]: x=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'O_3','PM10', 'PXY',
         y=df2['station']
```

```
In [24]: 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 [12]: from sklearn.linear_model import LinearRegression
```

```
In [13]: lr=LinearRegression()
         lr.fit(x_train,y_train)
```

```
Out[13]: LinearRegression()
```

```
In [14]: coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
         coeff
```
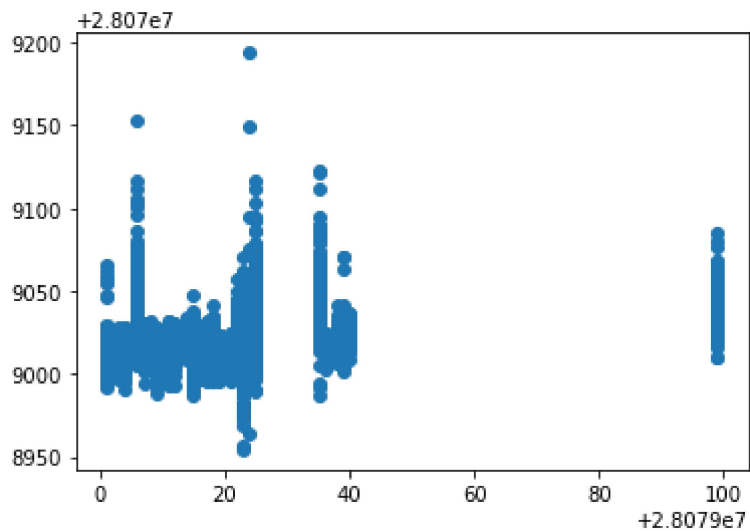
Out[14]:

|       | Co-efficient |
|-------|--------------|
| BEN   | 2.833498     |
| CO    | 4.218625     |
| EBE   | -1.896436    |
| MXY   | 0.996361     |
| NMHC  | -11.596930   |
| NO_2  | -0.044724    |
| NOx   | -0.038944    |
| O_3   | -0.014382    |
| PM10  | 0.032646     |
| PXY   | 4.152622     |
| SO_2  | -0.108335    |
| TCH   | 6.355155     |
| TOL   | -0.418133    |
| OXY   | -2.160064    |

```
In [15]: print(lr.intercept_)
```

28079023.010687977

```
In [16]: prediction =lr.predict(x_test)
         py.scatter(y_test,prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x20aa0591eb0>

```
In [17]:  print(lr.score(x_test,y_test))

          0.13971016103643064

In [18]:  print(lr.score(x_train,y_train))

          0.1377927917601448
```

## Ridge

```
In [19]:  from sklearn.linear_model import Ridge,Lasso

In [20]:  rr=Ridge(alpha=10)
          rr.fit(x_train,y_train)

Out[20]:  Ridge(alpha=10)

In [21]:  rr.score(x_test,y_test)

Out[21]:  0.13971942807063387
```

## Lasso

```
In [22]:  la=Lasso(alpha=10)
          la.fit(x_train,y_train)

Out[22]:  Lasso(alpha=10)

In [23]:  la.score(x_test,y_test)

Out[23]:  0.05961311096197963
```

## elasticnet

```
In [24]:  from sklearn.linear_model import ElasticNet
          en=ElasticNet()
          en.fit(x_train,y_train)

Out[24]:  ElasticNet()

In [25]:  print(en.coef_)

          [ 0.52192221  0.06008224 -0.         0.79420303  0.         -0.05650403
           -0.0175735  -0.01514683  0.04193206  0.86416339 -0.09613152  1.46701895
           -0.02315909  0.         ]
```

```
In [26]: print(en.intercept_)

         28079025.64837437

In [27]: print(en.predict(x_test))

         [28079022.98173612 28079021.18830581 28079020.94239793 ...
          28079040.90115401 28079016.84820627 28079026.37385159]

In [28]: print(en.score(x_test,y_test))

         0.10679846191048736
```

# logistic

```
In [8]: feature_matrix=df2.iloc[:,0:14]
        target_vector=df2.iloc[:,-1]

In [9]: feature_matrix=df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'O_3','PM
        y=df2['station']

In [10]: feature_matrix.shape

Out[10]: (217296, 14)

In [11]: target_vector.shape

Out[11]: (217296,)

In [12]: from sklearn.preprocessing import StandardScaler

In [13]: fs=StandardScaler().fit_transform(feature_matrix)

In [14]: logr =LogisticRegression()
         logr.fit(fs,target_vector)
```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: 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://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
    n_iter_i = _check_optimize_result(

```
Out[14]: LogisticRegression()
```

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

```
In [16]:  prediction=logr.predict(observation)
          print(prediction)
```

```
          [28079099]
```

```
In [17]:  logr.classes_
```

```
Out[17]:  array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
                 28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
                 28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
                 28079024, 28079025, 28079035, 28079036, 28079038, 28079039,
                 28079040, 28079099], dtype=int64)
```

```
In [18]:  logr.score(fs,target_vector)
```

```
Out[18]:  0.9224836168176128
```

```
In [19]:  logr.predict_proba(observation)[0][0]
```

```
Out[19]:  0.0
```

```
In [20]:  logr.predict_proba(observation)[0][1]
```

```
Out[20]:  0.0
```

## random forest

```
In [21]:  from sklearn.ensemble import RandomForestClassifier
          from sklearn.tree import plot_tree
```

```
In [22]:  x=df2.drop('station',axis=1)
          y=df2['station']
```

```
In [25]:  x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.70)
```

```
In [26]:  rfc=RandomForestClassifier()
          rfc.fit(x_train,y_train)
```

```
Out[26]:  RandomForestClassifier()
```

```
In [27]:  parameters = {'max_depth':[1,2,3,4,5],
                        'min_samples_leaf':[5,10,15,20,25],
                        'n_estimators':[10,20,30,40,50]}
```

```
In [28]:  from sklearn.model_selection import GridSearchCV
```

```
In [29]: grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='ac
         grid_search.fit(x_train,y_train)
```

Out[29]: 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 [30]: grid_search.best_score_
```

Out[30]: 0.46224765294225933

```
In [31]: rfc_best =grid_search.best_estimator_
```

```
In [32]: py.figure(figsize=(80,50))
         plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

Out[32]: [Text(2317.846153846154, 2491.5, 'BEN <= 0.08\ngini = 0.96\nsamples = 41186
         \nvalue = [2613, 2648, 2619, 2168, 2497, 320, 2540, 2737, 2575\n2576, 2686,
         2622, 2476, 2383, 2628, 2641, 2616, 2607\n2667, 2543, 2591, 2678, 2659, 272
         0, 2751, 2627]'),
          Text(1373.5384615384614, 2038.5, 'CO <= 0.685\ngini = 0.943\nsamples = 285
         22\nvalue = [2613, 2648, 2619, 16, 2497, 320, 2540, 2737, 2575\n2576, 44, 2
         622, 2476, 2383, 2628, 2641, 155, 10, 179\n41, 72, 2678, 2659, 2720, 2751,
         25]'),
          Text(686.7692307692307, 1585.5, 'NMHC <= 0.005\ngini = 0.939\nsamples = 17
         000\nvalue = [778, 1645, 1812, 16, 1245, 83, 935, 2041, 1386, 767\n32, 200
         8, 1895, 1726, 1675, 1960, 106, 10, 146, 37\n57, 1910, 930, 1801, 1893, 1
         9]'),
          Text(343.38461538461536, 1132.5, 'CO <= 0.205\ngini = 0.919\nsamples = 125
         54\nvalue = [778, 1645, 1812, 16, 70, 2, 935, 10, 1269, 767, 18\n2008, 189
         5, 56, 1675, 1960, 106, 9, 9, 37, 24\n1910, 930, 1801, 90, 4]'),
          Text(171.69230769230768, 679.5, 'TCH <= 1.205\ngini = 0.886\nsamples = 222
         7\nvalue = [59, 137, 428, 16, 20, 2, 22, 10, 508, 77, 2, 291\n575, 53, 58,
         658, 20, 9, 3, 12, 14, 134, 177\n223, 25, 2]'),
          Text(85.84615384615384, 226.5, 'gini = 0.884\nsamples = 2169\nvalue = [59,
```

# conclusion

**The bestfit model is Logistic Regression with score of 0.9224836168176128**

```
In [ ]:
```