In [13]:

```python
# import libraries
from sklearn.ensemble import GradientBoostingClassifier
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
#from sklearn.model_selection import GridSearchCV
import matplotlib.pylab as plt
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```python
# define dataset
df = pd.read_csv('PRSA_Data_Aotizhongxin_20130301-20170228.csv' , nrows = 500)

df_new = df.drop(['wd','station'] , axis = 1)
df2 = df_new[np.isfinite(df_new).all(1)]
df2.head()
```

Out[2]:

	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	WSPM
0	1	2013	3	1	0	4	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0	-18.8	0.0	4.4
1	2	2013	3	1	1	8	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2	-18.2	0.0	4.7
2	3	2013	3	1	2	7	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5	-18.2	0.0	5.6
3	4	2013	3	1	3	6	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5	-19.4	0.0	3.1
4	5	2013	3	1	4	3	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	2.0

In [3]:

```python
# Split dataset into test and train data
X_train, X_test, y_train, y_test = train_test_split(df2.drop('PRES' , axis = 1) , df2['PRES'] , test_size = 0.2)
```

In [17]:

```python
# define Gradient Boosting Classifier with hyperarameters
gbc = GradientBoostingClassifier(n_estimators = 500 , learning_rate = 0.05 , random_state = 100 , max_features = 5)
# fit train data to gbc
gbc.fit((X_train).round() , ( y_train).round())
```

Out[17]:

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.05, loss='deviance', max_depth=3,
                           max_features=5, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=500,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=100, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [21]:

```python
prediction = gbc.predict(X_test).round()
```

In [23]:

```python
from sklearn.metrics import classification_report , confusion_matrix
```

In [25]:

```python
from sklearn.metrics import accuracy_score
accuracy_score((y_test).round() , prediction)
```

Out[25]:

```
0.4020618556701031
```

In [32]:

```
# Confusion matrix will give number of correct and incorrect classifications
print(confusion_matrix((y_test).round() , prediction))
```

```
[[3 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 1]
 [0 0 0 ... 0 0 0]]
```

In [30]:

```
# check the classification report
print(classification_report((y_test).round() , prediction))
```

```
              precision    recall  f1-score   support

       996.0       1.00      0.75      0.86         4
       997.0       0.00      0.00      0.00         1
       998.0       0.00      0.00      0.00         0
       999.0       0.00      0.00      0.00         2
      1000.0       0.75      0.43      0.55         7
      1001.0       0.00      0.00      0.00         4
      1002.0       0.00      0.00      0.00         1
      1003.0       0.00      0.00      0.00         0
      1004.0       0.00      0.00      0.00         3
      1005.0       0.00      0.00      0.00         3
      1006.0       0.43      0.38      0.40         8
      1007.0       0.62      0.83      0.71         6
      1008.0       0.40      1.00      0.57         4
      1009.0       0.00      0.00      0.00         3
      1010.0       0.50      0.33      0.40         3
      1011.0       0.00      0.00      0.00         2
      1012.0       0.00      0.00      0.00         2
      1013.0       0.00      0.00      0.00         1
      1014.0       0.38      0.75      0.50         4
      1015.0       0.00      0.00      0.00         6
      1016.0       0.14      0.50      0.22         2
      1017.0       0.57      0.50      0.53         8
      1018.0       0.75      0.43      0.55         7
      1019.0       0.33      1.00      0.50         2
      1020.0       0.00      0.00      0.00         1
      1021.0       0.50      0.50      0.50         2
      1022.0       1.00      0.33      0.50         3
      1023.0       0.00      0.00      0.00         0
      1024.0       1.00      0.75      0.86         4
      1025.0       0.33      1.00      0.50         1
      1026.0       0.00      0.00      0.00         1
      1030.0       1.00      0.50      0.67         2
      1031.0       0.00      0.00      0.00         0

    accuracy                           0.40        97
   macro avg       0.29      0.30      0.27        97
weighted avg       0.43      0.40      0.39        97
```

In [31]:

```
# find the accuracy of model
print("GBC accuracy is %2.2f" % accuracy_score(
    (y_test).round(), gbc.predict(X_test)))
```

```
GBC accuracy is 0.40
```

In [ ]:

```

```