In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import f1_score,precision_score,recall_score,roc_auc_score,accuracy_sco
re,roc_curve,confusion_matrix
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost.sklearn import XGBClassifier
import lightgbm as lgb
from sklearn.preprocessing import MinMaxScaler
import os
import pandas as pd
import lightgbm as lgb
from sklearn.preprocessing import Imputer
import math
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit
%matplotlib inline
from sklearn.preprocessing import Imputer
from sklearn import preprocessing
import numpy as np
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29:
DeprecationWarning: numpy.core.umath_tests is an internal NumPy module and should
 not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:

```
f = 'pocd.csv'
pocd = pd.read_csv(f, encoding = 'gb18030')

pocd.describe()
```

Out[2]:

	SEX	AGE	HEIGHT	WEIGHT	BMI	A
count	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000(
mean	1.223148	63.756481	165.111422	59.136555	21.675307	2.031481
std	0.416550	11.262598	8.067546	10.546948	3.052788	0.390966
min	1.000000	25.000000	91.392661	-12.188420	13.671875	1.000000
25%	1.000000	57.000000	160.000000	52.000000	19.485789	2.000000
50%	1.000000	64.000000	166.000000	59.000000	21.484375	2.000000
75%	1.000000	72.000000	170.000000	65.000000	23.661439	2.000000
max	2.000000	92.000000	183.000000	95.000000	32.006920	3.000000

8 rows × 21 columns

In [3]:

```
X= pocd.iloc[:, 0:-1]
y = pocd.iloc[:, -1:]

X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size=0.3,random_state=41,stratify=y)
print(X_train)
print(y_train)
```

	SEX	AGE	HEIGHT	WEIGHT	BMI	ASA	TMN	TUMOR.SIZE	PLR	\
958	1	75	170.0	72.5	25.086505	2	4	4.159529	165.000000	
229	2	43	151.0	56.5	24.779615	2	3	1.500000	194.000000	
275	1	67	170.0	61.5	21.280277	2	2	3.759810	87.000000	
791	1	56	166.0	60.0	21.773842	2	1	3.000000	123.333333	
559	1	56	167.0	57.0	20.438166	2	3	3.000000	78.235294	
..	...	...	...	...	...	...	...	...	...	
889	2	77	153.0	49.0	20.932120	2	1	5.000000	105.000000	
383	1	51	170.0	58.0	20.069204	2	3	5.500000	150.000000	
981	1	59	163.0	74.0	27.852008	2	3	8.000000	234.615385	
104	1	77	165.0	59.0	21.671258	3	3	4.000000	131.578947	
888	1	66	176.0	58.0	18.724174	2	1	6.000000	61.333333	

	NLR	PREOPERATIVE.HEMOGLOBIN	PLATELET.COUNT	ALBUMIN	\
958	3.125000	147	264	44.2	
229	2.600000	101	194	40.4	
275	1.750000	153	174	43.3	
791	2.476190	120	259	42.8	
559	1.470588	153	133	40.2	
..	...	...	...	...	
889	4.333333	96	126	39.1	
383	1.722222	96	270	38.4	
981	4.538462	91	305	33.3	
104	3.526316	120	250	36.8	
888	2.333333	92	92	38.3	

	NEUTROPHIL.COUNT	LYMPHOCYTE.COUNT	MONOCYTE.COUNT	WBC.COUNT	\
958	5.0	1.6	0.6	7.35	
229	2.6	1.0	0.5	6.10	
275	3.5	2.0	0.5	6.20	
791	5.2	2.1	0.5	7.97	
559	2.5	1.7	0.5	4.80	
..	...	...	...	...	
889	5.2	1.2	0.4	6.93	
383	3.1	1.8	0.2	5.60	
981	5.9	1.3	0.3	7.64	
104	6.7	1.9	0.2	9.10	
888	3.5	1.5	0.6	5.78	

	BORRMANN.TYPES	THE.PATHOLOGICAL.TYPES.GROUP	DEPTH.OF.INVASION
958	2	1	2
229	2	1	2
275	1	0	1
791	2	1	2
559	2	1	2
..	...	...	...
889	2	1	2
383	2	1	2
981	2	1	2
104	2	1	2
888	2	1	2

[756 rows x 20 columns]

	PERITONEAL.METASTASIS
958	0
229	1
275	0
791	0
559	0
..	...
889	0

```
383                     0
981                     0
104                     0
888                     0

[756 rows x 1 columns]
```

In [4]:

```
xy_train = pd.concat([X_train, y_train], axis=1)
xy_test = pd.concat([X_test, y_test], axis=1)
f_xy_train = os.path.splitext(f)[0] + '_train.csv'
f_xy_test = os.path.splitext(f)[0] + '_test.csv'
xy_train.to_csv(f_xy_train, index=False, encoding = 'gb18030')
xy_test.to_csv(f_xy_test, index=False, encoding = 'gb18030')
xy_train = pd.read_csv(f_xy_train, encoding = 'gb18030')
xy_test = pd.read_csv(f_xy_test, encoding = 'gb18030')
```

In [5]:

```
X_train = xy_train.iloc[:, 0:-1]
y_train = xy_train.iloc[:, -1:]
X_test = xy_test.iloc[:, 0:-1]
y_test = xy_test.iloc[:, -1:]
```

In [6]:

```
import numpy as np
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
ya, yb = y_train.shape
y_train = y_train.reshape(ya,)
y_test = np.array(y_test)
ya, yb = y_test.shape
y_test = y_test.reshape(ya,)
```

In [7]:

```
from sklearn.preprocessing import MinMaxScaler
ss = MinMaxScaler()
xy_train = ss.fit_transform(xy_train, y_train)
xy_test = ss.transform(xy_test)
print(xy_train )
```

```
[[0.         0.74626866 0.85663736 ... 1.         1.         0.        ]
 [1.         0.26865672 0.64710736 ... 1.         1.         1.        ]
 [0.         0.62686567 0.85663736 ... 0.         0.         0.        ]
 ...
 [0.         0.50746269 0.7794421  ... 1.         1.         0.        ]
 [0.         0.7761194  0.80149789 ... 1.         1.         0.        ]
 [0.         0.6119403  0.92280473 ... 1.         1.         0.        ]]
```

In [8]:

```
xy_train.shape
```

Out[8]:

```
(756, 21)
```

In [9]:

```
lr = LogisticRegression(penalty='l2',tol=0.0001,C=1,intercept_scaling=1,max_iter=100)
lr.fit(X_train, y_train)
lr_y_proba=lr.predict_proba(X_train)
lr_y_pre=lr.predict(X_train)
lr_score = lr.score(X_train, y_train)
lr_accuracy_score=accuracy_score(y_train,lr_y_pre)
lr_preci_score=precision_score(y_train,lr_y_pre)
lr_recall_score=recall_score(y_train,lr_y_pre)
lr_f1_score=f1_score(y_train,lr_y_pre)
lr_auc=roc_auc_score(y_train,lr_y_proba[:,1])



print('lr_accuracy_score:%.3f,lr_auc:%.3f'
      %(lr_accuracy_score,lr_auc))
```

```
lr_accuracy_score:0.906,lr_auc:0.741
```

In [10]:

```
tr = DecisionTreeClassifier(splitter='best', max_depth=2, min_samples_split=20,min_samples_leaf
=5, min_weight_fraction_leaf=0.1)  # 决策树模型
tr.fit(X_train, y_train)
tr_y_pre=tr.predict(X_train)
tr_y_proba=tr.predict_proba(X_train)
tr_score = tr.score(X_train, y_train)
tr_accuracy_score=accuracy_score(y_train,tr_y_pre)
tr_preci_score=precision_score(y_train,tr_y_pre)
tr_recall_score=recall_score(y_train,tr_y_pre)
tr_f1_score=f1_score(y_train,tr_y_pre)
tr_auc=roc_auc_score(y_train,tr_y_proba[:,1])
print('tr_accuracy_score:%.3f,tr_auc:%.3f'
     %(tr_accuracy_score,tr_auc))
```

```
tr_accuracy_score:0.906,tr_auc:0.712
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no pred
icted samples.
  'precision', 'predicted', average, warn_for)
```

In [11]:

```
forest=RandomForestClassifier(n_estimators=10,max_depth=3,min_samples_split=70, min_samples_leaf
 =6,random_state =41) #  随机森林
forest.fit(X_train,y_train)
forest_y_pre=forest.predict(X_train)
forest_y_proba=forest.predict_proba(X_train)
forest_accuracy_score=accuracy_score(y_train,forest_y_pre)
forest_preci_score=precision_score(y_train,forest_y_pre)
forest_recall_score=recall_score(y_train,forest_y_pre)
forest_f1_score=f1_score(y_train,forest_y_pre)
forest_auc=roc_auc_score(y_train,forest_y_proba[:,1])
print('forest_accuracy_score:%.3f,forest_auc:%.3f'
      %(forest_accuracy_score,forest_auc))
```

```
forest_accuracy_score:0.906,forest_auc:0.796
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no pred
icted samples.
  'precision', 'predicted', average, warn_for)
```

In [12]:

```
Gbdt=GradientBoostingClassifier(learning_rate=0.06,n_estimators=50,max_depth=2, random_state
=41)#CBDT
Gbdt.fit(X_train,y_train)
Gbdt_y_pre=Gbdt.predict(X_train)
Gbdt_y_proba=Gbdt.predict_proba(X_train)
Gbdt_accuracy_score=accuracy_score(y_train,Gbdt_y_pre)
Gbdt_preci_score=precision_score(y_train,Gbdt_y_pre)
Gbdt_recall_score=recall_score(y_train,Gbdt_y_pre)
Gbdt_f1_score=f1_score(y_train,Gbdt_y_pre)
Gbdt_auc=roc_auc_score(y_train,Gbdt_y_proba[:,1])
print('Gbdt_accuracy_score:%.3f,Gbdt_auc:%.3f'
      %(Gbdt_accuracy_score,Gbdt_auc))
```

```
Gbdt_accuracy_score:0.909,Gbdt_auc:0.861
```

In [13]:

```
gbm=lgb.LGBMClassifier(learning_rate=0.1, n_estimators=30,max_depth=3)  #lgb
gbm.fit(X_train,y_train)
gbm_y_pre=gbm.predict(X_train)
gbm_y_proba=gbm.predict_proba(X_train)
gbm_accuracy_score=accuracy_score(y_train,gbm_y_pre)
gbm_preci_score=precision_score(y_train,gbm_y_pre)
gbm_recall_score=recall_score(y_train,gbm_y_pre)
gbm_f1_score=f1_score(y_train,gbm_y_pre)
gbm_auc=roc_auc_score(y_train,gbm_y_proba[:,1])
print('gbm_accuracy_score:%.3f,gbm_auc:%.3f'
      %(gbm_accuracy_score,gbm_auc))
```

```
gbm_accuracy_score:0.909,gbm_auc:0.938

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: Dep
recationWarning: The truth value of an empty array is ambiguous. Returning False,
 but in future this will result in an error. Use `array.size > 0` to check that an
array is not empty.
  if diff:
```

In [15]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import zero_one_loss,log_loss
clf = lr
y_pred = clf.predict(X_train)
y_true = y_train
mse = mean_squared_error(y_true, y_pred)
print(" LR MSE: %.3f" % mse)
def train_zero_one_loss(y_true, y_pred):
    print("zero_one_loss<fraction>:",zero_one_loss(y_true,y_pred,normalize=True))
    print("zero_one_loss<num>:",zero_one_loss(y_true,y_pred,normalize=False))

def train_log_loss(y_true, y_pred):
    print("log_loss<average>:",log_loss(y_true,y_pred,normalize=True))
    print("log_loss<total>:",log_loss(y_true,y_pred,normalize=False))

train_zero_one_loss(y_true, y_pred)
train_log_loss(y_true, y_pred)
def train_confusion_matrix(y_true, y_pred):
    print('Confusion Matrix:\n',confusion_matrix(y_true,y_pred,labels=[0,1]))
train_confusion_matrix(y_true, y_pred)

clf = tr
y_pred = clf.predict(X_train)
y_true = y_train
mse = mean_squared_error(y_true, y_pred)
print(" TR MSE: %.3f" % mse)
def train_zero_one_loss(y_true, y_pred):
    print("zero_one_loss<fraction>:",zero_one_loss(y_true,y_pred,normalize=True))
    print("zero_one_loss<num>:",zero_one_loss(y_true,y_pred,normalize=False))

def train_log_loss(y_true, y_pred):
    print("log_loss<average>:",log_loss(y_true,y_pred,normalize=True))
    print("log_loss<total>:",log_loss(y_true,y_pred,normalize=False))



train_zero_one_loss(y_true, y_pred)
clf = forest
y_pred = clf.predict(X_train)
y_true = y_train
mse = mean_squared_error(y_true, y_pred)
print(" forest MSE: %.3f" % mse)
clf =Gbdt
y_pred = clf.predict(X_train)
y_true = y_train
mse = mean_squared_error(y_true, y_pred)
print(" Gbdt MSE: %.3f" % mse)

clf = gbm
y_pred = clf.predict(X_train)
y_true = y_train
mse = mean_squared_error(y_true, y_pred)
print(" gbm MSE: %.3f" % mse)
```

```
 LR MSE: 0.094
zero_one_loss<fraction>: 0.09391534391534395
zero_one_loss<num>: 71
log_loss<average>: 3.243722121211758
log_loss<total>: 2452.253923636089
Confusion Matrix:
 [[684   1]
 [ 70   1]]
 TR MSE: 0.094
zero_one_loss<fraction>: 0.09391534391534395
zero_one_loss<num>: 71
 forest MSE: 0.094
 Gbdt MSE: 0.091
 gbm MSE: 0.091

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: Dep
recationWarning: The truth value of an empty array is ambiguous. Returning False,
 but in future this will result in an error. Use `array.size > 0` to check that an
array is not empty.
  if diff:
```

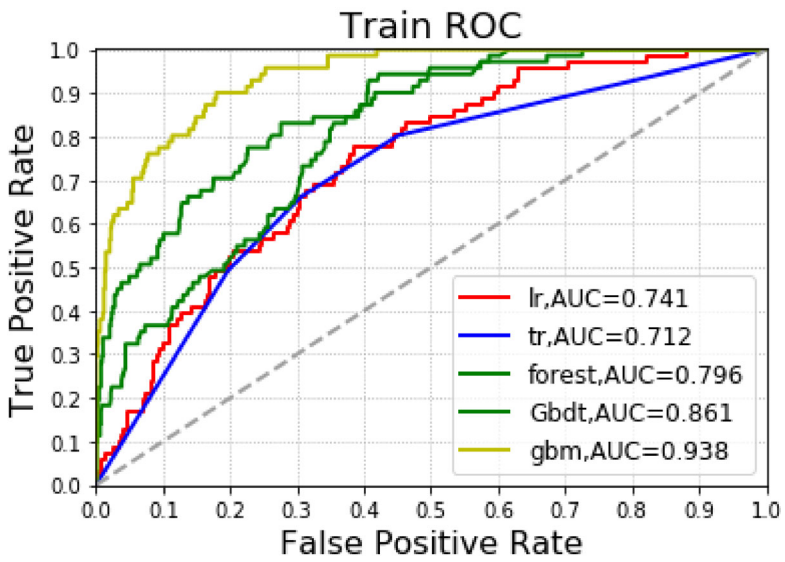
In [16]:

```
lr_fpr,lr_tpr,lr_threasholds=roc_curve(y_train,lr_y_proba[:,1])
tr_fpr,tr_tpr,tr_threasholds=roc_curve(y_train,tr_y_proba[:,1])
forest_fpr,forest_tpr,forest_threasholds=roc_curve(y_train,forest_y_proba[:,1])
Gbdt_fpr,Gbdt_tpr,Gbdt_threasholds=roc_curve(y_train,Gbdt_y_proba[:,1])
gbm_fpr,gbm_tpr,gbm_threasholds=roc_curve(y_train,gbm_y_proba[:,1])

plt.plot(lr_fpr,lr_tpr,c='r',lw=2,label=u'lr,AUC=%.3f'% lr_auc)
plt.plot(tr_fpr,tr_tpr,c='b',lw=2,label=u'tr,AUC=%.3f' % tr_auc)
plt.plot(forest_fpr,forest_tpr,c='g',lw=2,label=u'forest,AUC=%.3f' % forest_auc)
plt.plot(Gbdt_fpr,Gbdt_tpr,c='g',lw=2,label=u'Gbdt,AUC=%.3f' % Gbdt_auc)

plt.plot(gbm_fpr,gbm_tpr,c='y',lw=2,label=u'gbm,AUC=%.3f' % gbm_auc)

plt.plot((0,1),(0,1),c='#a0a0a0',lw=2,ls='--')
plt.xlim(-0.001, 1.001)
plt.ylim(-0.001, 1.001)
plt.xticks(np.arange(0, 1.1, 0.1))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.xlabel('False Positive Rate', fontsize=16)
plt.ylabel('True Positive Rate', fontsize=16)
plt.grid(b=True, ls=':')
plt.legend(loc='lower right', fancybox=True, framealpha=0.8, fontsize=12)
plt.title(u'Train ROC', fontsize=18)
fig = plt.figure(figsize=(8, 15), dpi= 600)
plt.show()
```



<Figure size 4800x9000 with 0 Axes>

In [17]:

```
lr = LogisticRegression(penalty='l2',tol=0.0001,C=1,intercept_scaling=1,max_iter=100)
lr.fit(X_train, y_train)
lr_y_proba=lr.predict_proba(X_test)
lr_y_pre=lr.predict(X_test)
```

In [18]:

```
lr_score = lr.score(X_test, y_test)
lr_accuracy_score=accuracy_score(y_test,lr_y_pre)
lr_preci_score=precision_score(y_test,lr_y_pre)
lr_recall_score=recall_score(y_test,lr_y_pre)
lr_f1_score=f1_score(y_test,lr_y_pre)
lr_auc=roc_auc_score(y_test,lr_y_proba[:,1])
print('lr_accuracy_score:%.3f,lr_auc:%.3f'
      %(lr_accuracy_score,lr_auc))
```

```
lr_accuracy_score:0.904,lr_auc:0.680
```

In [19]:

```
tr = DecisionTreeClassifier(splitter='best', max_depth=2, min_samples_split=20,min_samples_leaf
=5, min_weight_fraction_leaf=0.1)  # 决策树模型
tr.fit(X_train, y_train)
tr_y_pre=tr.predict(X_test)
tr_y_proba=tr.predict_proba(X_test)
```

In [20]:

```
tr_score = tr.score(X_test, y_test)
tr_accuracy_score=accuracy_score(y_test,tr_y_pre)
tr_preci_score=precision_score(y_test,tr_y_pre)
tr_recall_score=recall_score(y_test,tr_y_pre)
tr_f1_score=f1_score(y_test,tr_y_pre)
tr_auc=roc_auc_score(y_test,tr_y_proba[:,1])
print('tr_accuracy_score:%.3f,tr_auc:%.3f'
     %(tr_accuracy_score,tr_auc))
```

```
tr_accuracy_score:0.907,tr_auc:0.657

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no pred
icted samples.
  'precision', 'predicted', average, warn_for)
```

In [21]:

```
forest=RandomForestClassifier(n_estimators=10,max_depth=3,min_samples_split=70, min_samples_leaf
 =6,random_state =41) #  随机森林
forest.fit(X_train,y_train)
forest_y_pre=forest.predict(X_test)
forest_y_proba=forest.predict_proba(X_test)
```

In [22]:

```
forest_accuracy_score=accuracy_score(y_test,forest_y_pre)
forest_preci_score=precision_score(y_test,forest_y_pre)
forest_recall_score=recall_score(y_test,forest_y_pre)
forest_f1_score=f1_score(y_test,forest_y_pre)
forest_auc=roc_auc_score(y_test,forest_y_proba[:,1])
print('forest_accuracy_score:%.3f,forest_auc:%.3f'
      %(forest_accuracy_score,forest_auc))
```

```
forest_accuracy_score:0.907,forest_auc:0.696
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no pred
icted samples.
  'precision', 'predicted', average, warn_for)
```

In [23]:

```
Gbdt=GradientBoostingClassifier(learning_rate=0.06,n_estimators=50,max_depth=2, random_state
=41)#CBDT
Gbdt.fit(X_train,y_train)
Gbdt_y_pre=Gbdt.predict(X_test)
Gbdt_y_proba=Gbdt.predict_proba(X_test)
```

In [24]:

```
Gbdt_accuracy_score=accuracy_score(y_test,Gbdt_y_pre)
Gbdt_preci_score=precision_score(y_test,Gbdt_y_pre)
Gbdt_recall_score=recall_score(y_test,Gbdt_y_pre)
Gbdt_f1_score=f1_score(y_test,Gbdt_y_pre)
Gbdt_auc=roc_auc_score(y_test,Gbdt_y_proba[:,1])
print('Gbdt_accuracy_score:%.3f,Gbdt_auc:%.3f'
      %(Gbdt_accuracy_score,Gbdt_auc))
```

```
Gbdt_accuracy_score:0.904,Gbdt_auc:0.725
```

In [25]:

```
gbm=lgb.LGBMClassifier(learning_rate=0.1, n_estimators=30,max_depth=3)  #lgb
gbm.fit(X_train,y_train)
gbm_y_pre=gbm.predict(X_test)
gbm_y_proba=gbm.predict_proba(X_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: Dep
recationWarning: The truth value of an empty array is ambiguous. Returning False,
 but in future this will result in an error. Use `array.size > 0` to check that an
array is not empty.
  if diff:
```

In [26]:

```
gbm_accuracy_score=accuracy_score(y_test,gbm_y_pre)
gbm_preci_score=precision_score(y_test,gbm_y_pre)
gbm_recall_score=recall_score(y_test,gbm_y_pre)
gbm_f1_score=f1_score(y_test,gbm_y_pre)
gbm_auc=roc_auc_score(y_test,gbm_y_proba[:,1])
print('gbm_accuracy_score:%.3f,gbm_auc:%.3f'
      %(gbm_accuracy_score,gbm_auc))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135:
UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no pred
icted samples.
  'precision', 'predicted', average, warn_for)
```

```
gbm_accuracy_score:0.907,gbm_auc:0.745
```

In [27]:

```
clf = lr
y_pred = clf.predict(X_test)
y_true = y_test
mse = mean_squared_error(y_true, y_pred)
print(" LR MSE: %.3f" % mse)
clf = tr
y_pred = clf.predict(X_test)
y_true = y_test
mse = mean_squared_error(y_true, y_pred)
print(" TR MSE: %.3f" % mse)

clf = forest
y_pred = clf.predict(X_test)
y_true = y_test
mse = mean_squared_error(y_true, y_pred)
print(" forest MSE: %.3f" % mse)
clf =Gbdt
y_pred = clf.predict(X_test)
y_true = y_test
mse = mean_squared_error(y_true, y_pred)
print(" Gbdt MSE: %.3f" % mse)

clf = gbm
y_pred = clf.predict(X_test)
y_true = y_test
mse = mean_squared_error(y_true, y_pred)
print(" gbm MSE: %.3f" % mse)
```

```
 LR MSE: 0.096
 TR MSE: 0.093
 forest MSE: 0.093
 Gbdt MSE: 0.096
 gbm MSE: 0.093
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: Dep
recationWarning: The truth value of an empty array is ambiguous. Returning False,
 but in future this will result in an error. Use `array.size > 0` to check that an
array is not empty.
  if diff:
```