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_score,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]:

	gender	age	bmi	smoking	alcoholuse	Cardiacdisease
count	912.000000	912.000000	912.000000	912.000000	912.000000	912.000000
mean	0.388158	59.612939	25.220066	0.194079	0.140351	0.112939
std	0.487598	10.611613	2.632384	0.395707	0.347541	0.316691
min	0.000000	31.000000	18.400000	0.000000	0.000000	0.000000
25%	0.000000	52.000000	23.600000	0.000000	0.000000	0.000000
50%	0.000000	58.000000	25.400000	0.000000	0.000000	0.000000
75%	1.000000	67.000000	26.600000	0.000000	0.000000	0.000000
max	1.000000	85.000000	32.900000	1.000000	1.000000	1.000000

8 rows × 27 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)
```

```
     gender  age   bmi  smoking  alcoholuse  Cardiacdisease  hypertension  \
206       1   61  26.3        1           0               0             1
731       0   65  26.3        0           0               0             0
283       1   60  27.5        0           0               0             1
79        1   72  29.6        1           0               0             1
902       1   36  22.1        1           1               0             0
..      ...  ...   ...      ...         ...             ...           ...
356       0   57  27.4        0           0               0             1
864       0   67  22.7        0           0               0             0
760       1   51  26.3        0           0               0             0
767       1   74  26.3        0           0               0             0
123       1   69  26.3        1           0               1             1

     diabetesmellitus  cerebrovasculardisease  operationtime  ...    wbc  \
206                 0                       1            164  ...   8.00
731                 0                       0            144  ...   7.50
283                 0                       1            142  ...  13.00
79                  0                       0            151  ...   8.00
902                 0                       0            145  ...   8.27
..                ...                     ...            ...  ...    ...
356                 0                       0            170  ...  13.90
864                 0                       0            141  ...  12.90
760                 1                       0            169  ...  14.30
767                 0                       0            149  ...  11.80
123                 0                       0            157  ...  11.00

        na   na2  bun  crea  preoperativedoseofcbz  durationofcbzyears  \
206  138.4  1384  4.0  83.7                    700                5.69
731  138.9  1389  4.6  77.1                    700                5.26
283  138.4  1384  4.0  81.8                    700                5.60
79   136.8  1368  3.8  79.6                    650                6.66
902  147.6  1476  2.8  18.9                    400                1.60
..     ...   ...  ...   ...                    ...                 ...
356  140.4  1404  4.3  80.0                    600                5.28
864  135.3  1353  4.6  84.8                    600                4.87
760  135.0  1350  3.8  75.6                    700                6.27
767  135.0  1350  3.8  84.8                    650                4.72
123  133.7  1337  4.4  83.4                    750                6.18

     cbzserumlevelbefore  cbzserumlevelafter  cbztherapy
206                 6.11              1.4312           1
731                 4.51              1.1000           1
283                 7.63              1.6700           1
79                  7.89              1.3600           1
902                 4.99              1.3000           1
..                   ...                 ...         ...
356                 4.85              1.2900           1
864                 6.66              1.3100           0
760                 4.79              1.2200           0
767                 4.79              1.5900           1
123                 6.56              1.3800           0

[638 rows x 26 columns]
     delirium
206         1
731         0
283         1
79          1
902         0
..        ...
356         0
```

```
864         0
760         0
767         0
123         1

[638 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
# [DJun, 2018-9-13] 从别处复制粘贴过来这段
X_train = np.array(X_train)
# print(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
# print(y_train)
# print(y_train.shape)
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 )
```

```
[[1.         0.55555556 0.54482759 ... 0.47863636 1.         1.        ]
 [0.         0.62962963 0.54482759 ... 0.10227273 1.         0.        ]
 [1.         0.53703704 0.62758621 ... 0.75       1.         1.        ]
 ...
 [1.         0.37037037 0.54482759 ... 0.23863636 0.         0.        ]
 [1.         0.7962963  0.54482759 ... 0.65909091 1.         0.        ]
 [1.         0.7037037  0.54482759 ... 0.42045455 0.         1.        ]]
```

In [8]:

```
xy_train.shape
```

Out[8]:

(638, 27)

In [9]:

```
lr = LogisticRegression(penalty='l2',tol=0.0001,C=0.1,fit_intercept=True,intercept_scaling=1,class_weight=None,max_iter=100,multi_class='ovr',verbose=0,warm_start=False,n_jobs=1)  # 逻辑回归模型
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_preci_score:%.3f,lr_recall_score:%.3f,lr_f1_score:%.3f,lr_auc:%.3f'
      %(lr_accuracy_score,lr_preci_score,lr_recall_score,lr_f1_score,lr_auc))
```

lr_accuracy_score:0.900,lr_preci_score:0.876,lr_recall_score:0.684,lr_f1_score:0.7
68,lr_auc:0.925

In [10]:

```
tr = DecisionTreeClassifier(splitter='best', max_depth=5, min_samples_split=80,min_samples_leaf=65, min_weight_fraction_leaf=0.01, max_features=None, random_state=None,
max_leaf_nodes=None,class_weight=None, presort=False)  # 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_preci_score:%.3f,tr_recall_score:%.3f,tr_f1_score:%.3f,tr_auc:%.3f'
     %(tr_accuracy_score,tr_preci_score,tr_recall_score,tr_f1_score,tr_auc))
```

tr_accuracy_score:0.861,tr_preci_score:0.817,tr_recall_score:0.548,tr_f1_score:0.6
56,tr_auc:0.902

In [11]:

```
forest=RandomForestClassifier(n_estimators=500,max_features = "auto",min_samples_leaf = 5 ,n_job
s = 100,random_state =41) #  随机森林 forest.fit(X_train,y_train)
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_preci_score:%.3f,forest_recall_score:%.3f,forest_f1_s
core:%.3f,forest_auc:%.3f'
      %
(forest_accuracy_score,forest_preci_score,forest_recall_score,forest_f1_score,forest_auc))
```

```
forest_accuracy_score:0.948,forest_preci_score:0.984,forest_recall_score:0.800,for
est_f1_score:0.883,forest_auc:0.994
```

In [12]:

```
Gbdt=GradientBoostingClassifier(learning_rate=0.2,n_estimators=60,max_depth=2, max_features='aut
o', min_samples_split=20,min_samples_leaf=3,random_state =1)#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_preci_score:%.3f,Gbdt_recall_score:%.3f,Gbdt_f1_score:%.3
f,Gbdt_auc:%.3f'
      %(Gbdt_accuracy_score,Gbdt_preci_score,Gbdt_recall_score,Gbdt_f1_score,Gbdt_auc))
```

```
Gbdt_accuracy_score:0.970,Gbdt_preci_score:0.972,Gbdt_recall_score:0.903,Gbdt_f1_s
core:0.936,Gbdt_auc:0.994
```

In [13]:

```
gbm=lgb.LGBMClassifier(learning_rate=0.08, n_estimators=180, lambda_l1=0.2 ,lambda_l2= 10 ,max_depth=2, bagging_fraction = 0.8,feature_fraction = 0.6) #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_preci_score: %.3f,gbm_recall_score:%.3f,gbm_f1_score:%.3f,gbm_auc:%.3f'
      %(gbm_accuracy_score,gbm_preci_score,gbm_recall_score,gbm_f1_score,gbm_auc))
```

```
gbm_accuracy_score:0.944,gbm_preci_score: 0.934,gbm_recall_score:0.826,gbm_f1_score:0.877,gbm_auc:0.979
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: 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 [14]:

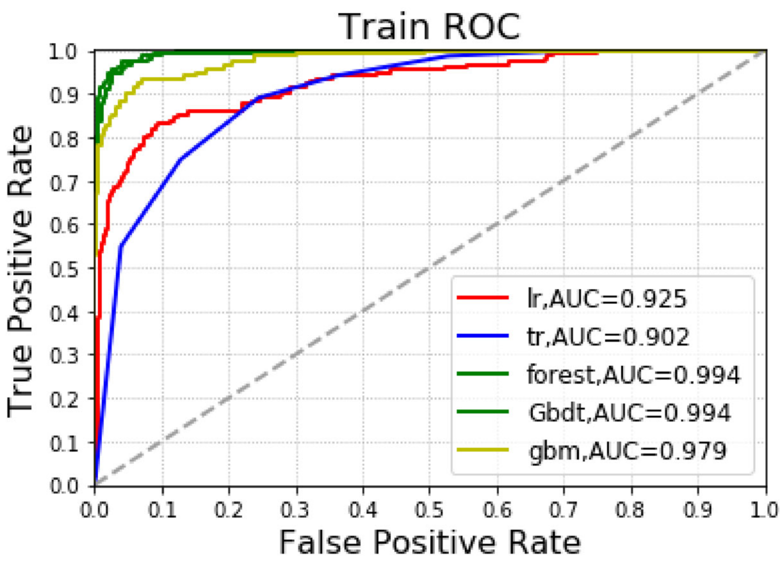
```
lr_fpr,lr_tpr,lr_threasholds=roc_curve(y_train,lr_y_proba[:,1])
tr_fpr,tr_tpr,tr_threasholds=roc_curve(y_train,tr_y_proba[:,1])

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(Xgbc_fpr,Xgbc_tpr,c='g',lw=2,label=u'Xgbc,AUC=%.3f' % svm_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 [15]:

```
lr = LogisticRegression(penalty='l2',tol=0.0001,C=0.1,fit_intercept=True,intercept_scaling=1,cl
ass_weight=None,max_iter=100,multi_class='ovr',verbose=0,warm_start=False,n_jobs=1)   # 逻辑回归
模型
lr.fit(X_train, y_train)
lr_y_proba=lr.predict_proba(X_test)
lr_y_pre=lr.predict(X_test)
```

In [16]:

```
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_preci_score:%.3f,lr_recall_score:%.3f,lr_f1_score:%.3f,lr_au
c:%.3f'
      %(lr_accuracy_score,lr_preci_score,lr_recall_score,lr_f1_score,lr_auc))
```

```
lr_accuracy_score:0.901,lr_preci_score:0.842,lr_recall_score:0.727,lr_f1_score:0.7
80,lr_auc:0.920
```

In [17]:

```
tr = DecisionTreeClassifier(splitter='best', max_depth=5, min_samples_split=80,min_samples_leaf
=65, min_weight_fraction_leaf=0.01, max_features=None, random_state=None,
max_leaf_nodes=None,class_weight=None, presort=False)  # 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 [18]:

```
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_preci_score:%.3f,tr_recall_score:%.3f,tr_f1_score:%.3f,tr_au
c:%.3f'
     %(tr_accuracy_score,tr_preci_score,tr_recall_score,tr_f1_score,tr_auc))
```

```
tr_accuracy_score:0.883,tr_preci_score:0.854,tr_recall_score:0.621,tr_f1_score:0.7
19,tr_auc:0.888
```

In [19]:

```
forest=RandomForestClassifier(n_estimators=500,max_features = "auto",min_samples_leaf = 5 ,n_job
s = 100,random_state =41) #  随机森林 forest.fit(X_train,y_train)
forest.fit(X_train,y_train)
forest_y_pre=forest.predict(X_test)
forest_y_proba=forest.predict_proba(X_test)
```

In [20]:

```
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_preci_score:%.3f,forest_recall_score:%.3f,forest_f1_s
core:%.3f,forest_auc:%.3f'
      %
(forest_accuracy_score,forest_preci_score,forest_recall_score,forest_f1_score,forest_auc))
```

```
forest_accuracy_score:0.909,forest_preci_score:0.887,forest_recall_score:0.712,for
est_f1_score:0.790,forest_auc:0.963
```

In [21]:

```
Gbdt=GradientBoostingClassifier(learning_rate=0.2,n_estimators=60,max_depth=2, max_features='aut
o', min_samples_split=20,min_samples_leaf=3,random_state =1)#CBDT
Gbdt.fit(X_train,y_train)
Gbdt_y_pre=Gbdt.predict(X_test)
Gbdt_y_proba=Gbdt.predict_proba(X_test)
```

In [22]:

```
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_preci_score:%.3f,Gbdt_recall_score:%.3f,Gbdt_f1_score:%.3
f,Gbdt_auc:%.3f'
      %(Gbdt_accuracy_score,Gbdt_preci_score,Gbdt_recall_score,Gbdt_f1_score,Gbdt_auc))
```

```
Gbdt_accuracy_score:0.923,Gbdt_preci_score:0.881,Gbdt_recall_score:0.788,Gbdt_f1_s
core:0.832,Gbdt_auc:0.962
```

In [23]:

```
gbm=lgb.LGBMClassifier(learning_rate=0.08, n_estimators=180, lambda_l1=0.2 ,lambda_l2= 10 ,max_d
epth=2, bagging_fraction = 0.8,feature_fraction = 0.6) #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 [24]:

```
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_preci_score: %.3f,gbm_recall_score:%.3f,gbm_f1_score:%.3f,gbm_auc:%.3f'
      %(gbm_accuracy_score,gbm_preci_score,gbm_recall_score,gbm_f1_score,gbm_auc))
```

```
gbm_accuracy_score:0.920,gbm_preci_score: 0.879,gbm_recall_score:0.773,gbm_f1_scor
e:0.823,gbm_auc:0.956
```