0.1 二次原发肺癌预后预测模型构建

0.1.1 背景介绍

二次原发肺癌是指在同一患者肺内不同位置同时或先后发生两个原发性肺癌.

考虑到部分二次原发肺癌患者肺部已 经有相关手术史和药物治疗史，其手术方式和治疗药物的选择具有较大的局限性,针对二次原发肺癌患者，医生急需一个能帮助其准确判断患者预后的辅助工具。

本工作使用04-18年SEER数据，预测二次原发肺癌患者第二次肺癌的5年生存情况，其中随访结束时间为19年，因此只使用了04-14年的数据进行训练

0.1.2 导入package

In [1]:
```
import os
import time
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from datetime import datetime
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score, roc_curve, mean_squared_error,mean_absolute_error, f1_score
import lightgbm
from lightgbm import Dataset
import xgboost as xgb
from sklearn.ensemble import RandomForestRegressor as rfr
from sklearn.ensemble import ExtraTreesRegressor as etr
from sklearn.linear_model import BayesianRidge as br
from sklearn.ensemble import GradientBoostingRegressor as gbr
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import LinearRegression as lr
from sklearn.linear_model import ElasticNet as en
from sklearn.kernel_ridge import KernelRidge as kr
from sklearn.model_selection import  KFold, StratifiedKFold,GroupKFold, RepeatedKFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
import logging
from numpy import int64,float64
import warnings

warnings.filterwarnings('ignore') #消除warning
```

0.1.3 导入数据集

In [2]:
```
data = pd.read_csv("./year5.csv",encoding='latin-1') #latin-1向下兼容ASCII
data.head
```

Out[2]:
```
<bound method NDFrame.head of       Patient.ID Age.recode.with..1.year.olds.x   Sex.x  Year.of.diagnosis.x  \
0         899968                    70-74 years    Male                 2004
1         916035                    70-74 years    Male                 2004
2         939226                    70-74 years    Male                 2004
3         943384                    65-69 years  Female                 2004
4         950080                    70-74 years    Male                 2004
...          ...                            ...     ...                  ...
4237    75416768                    65-69 years    Male                 2011
4238    75437034                    65-69 years    Male                 2011
4239    75471693                    65-69 years  Female                 2012
4240    75472175                    65-69 years  Female                 2011
4241    75535497                    70-74 years  Female                 2013

     Race.recode..W..B..AI..API..x Origin.recode.NHIA..Hispanic..Non.Hisp..x  \
0        Asian or Pacific Islander              Non-Spanish-Hispanic-Latino
1                            White              Non-Spanish-Hispanic-Latino
2                            White              Non-Spanish-Hispanic-Latino
3                            White              Non-Spanish-Hispanic-Latino
4                            White              Non-Spanish-Hispanic-Latino
```

#去除拒绝手术原因中体现出死亡信息的类别

In [3]:
```
print(data['Reason.no.cancer.directed.surgery.x'].value_counts())
```

```
Surgery performed                                                    2964
Not recommended                                                      1072
Not recommended, contraindicated due to other cond; autopsy only (1973-2002)     112
Recommended but not performed, unknown reason                          54
Recommended but not performed, patient refused                         26
Unknown; death certificate; or autopsy only (2003+)                     8
Recommended, unknown if performed                                       6
Name: Reason.no.cancer.directed.surgery.x, dtype: int64
```

In [4]:
```
data=data[~data['Reason.no.cancer.directed.surgery.x'].isin(['Not recommended, contraindicated due to other cond; autopsy only (1973-2
print(data['Reason.no.cancer.directed.surgery.x'].value_counts())
```

```
Surgery performed                                2964
Not recommended                                  1072
Recommended but not performed, unknown reason      54
Recommended but not performed, patient refused     26
Recommended, unknown if performed                   6
Name: Reason.no.cancer.directed.surgery.x, dtype: int64
```

In [5]:
```
print(data['Reason.no.cancer.directed.surgery.y'].value_counts())
```

```
Not recommended                                                  1959
Surgery performed                                                1868
Not recommended, contraindicated due to other cond; autopsy only (1973-2002)     154
Recommended but not performed, unknown reason                      75
Recommended but not performed, patient refused                     34
Unknown; death certificate; or autopsy only (2003+)                21
Recommended, unknown if performed                                  11
Name: Reason.no.cancer.directed.surgery.y, dtype: int64
```

In [6]:
```
data=data[~data['Reason.no.cancer.directed.surgery.y'].isin(['Not recommended, contraindicated due to other cond; autopsy only (1973-2
print(data['Reason.no.cancer.directed.surgery.y'].value_counts())
```

```
Not recommended                                  1959
Surgery performed                                1868
Recommended but not performed, unknown reason      75
Recommended but not performed, patient refused     34
Recommended, unknown if performed                  11
Name: Reason.no.cancer.directed.surgery.y, dtype: int64
```

In [7]:
```
print(data['Laterality.x'].value_counts())
```

```
Right - origin of primary                                2230
Left - origin of primary                                 1695
Bilateral, single primary                                  11
Paired site, but no information concerning laterality       9
Not a paired  site                                          1
Only one side - side unspecified                            1
Name: Laterality.x, dtype: int64
```

In [8]:
```
data=data[~data['Laterality.x'].isin(['Bilateral, single primary','Paired site, but no information concerning laterality','Not a paire
print(data['Laterality.x'].value_counts())
```

```
Right - origin of primary    2230
Left - origin of primary     1695
Name: Laterality.x, dtype: int64
```

In [9]:
```
print(data['Laterality.y'].value_counts())
```

```
Right - origin of primary                                2146
Left - origin of primary                                 1732
Paired site, but no information concerning laterality      34
Bilateral, single primary                                   7
Not a paired  site                                          3
Only one side - side unspecified                            3
Name: Laterality.y, dtype: int64
```

In [10]:
```
data=data[~data['Laterality.y'].isin(['Bilateral, single primary','Paired site, but no information concerning laterality','Not a paire
print(data['Laterality.y'].value_counts())
```

```
Right - origin of primary    2146
Left - origin of primary     1732
Name: Laterality.y, dtype: int64
```

0.1.4 查看数据的基本信息

In [11]:
```
print(data.label.value_counts()) #数据的基本信息 1代表死亡 0代表存活
```

```
1    2868
0    1010
Name: label, dtype: int64
```

In [12]:
```
print(data.columns)
```

```
Index(['Patient.ID', 'Age.recode.with..1.year.olds.x', 'Sex.x',
       'Year.of.diagnosis.x', 'Race.recode..W..B..AI..API..x',
       'Origin.recode.NHIA..Hispanic..Non.Hisp..x', 'Primary.Site...labeled.x',
       'Grade..thru.2017..x', 'Laterality.x', 'Diagnostic.Confirmation.x',
       'SEER.Combined.Summary.Stage.2000..2004.2017..x',
       'RX.Summ..Surg.Prim.Site..1998...x',
       'RX.Summ..Scope.Reg.LN.Sur..2003...x',
       'RX.Summ..Surg.Oth.Reg.Dis..2003...x', 'RX.Summ..Surg.Rad.Seq.x',
       'Reason.no.cancer.directed.surgery.x', 'Radiation.recode.x',
       'Chemotherapy.recode..yes..no.unk..x', 'RX.Summ..Systemic.Sur.Seq.x',
       'Regional.nodes.examined..1988...x',
       'Regional.nodes.positive..1988...x', 'Type.of.Reporting.Source.x',
       'Marital.status.at.diagnosis.x', 'Histology.classification.x',
       'radiotherapy.x', 'tumor_size.x', 'surgery_method.x', 'lymph_node.x',
       'AJCC8TH_N.x', 'AJCC8TH_M.x', 'AJCC8TH_T.x', 'AJCC8TH_STAGE.x',
       'Age.recode.with..1.year.olds.y', 'Year.of.diagnosis.y',
       'Primary.Site...labeled.y', 'Grade..thru.2017..y', 'Laterality.y',
       'Diagnostic.Confirmation.y',
       'SEER.Combined.Summary.Stage.2000..2004.2017..y',
       'RX.Summ..Surg.Prim.Site..1998...y',
       'RX.Summ..Scope.Reg.LN.Sur..2003...y',
       'RX.Summ..Surg.Oth.Reg.Dis..2003...y', 'RX.Summ..Surg.Rad.Seq.y',
       'Reason.no.cancer.directed.surgery.y', 'Radiation.recode.y',
       'Chemotherapy.recode..yes..no.unk..y', 'RX.Summ..Systemic.Sur.Seq.y',
       'Regional.nodes.examined..1988...y',
       'Regional.nodes.positive..1988...y', 'Primary.by.international.rules.y',
       'Type.of.Reporting.Source.y', 'Marital.status.at.diagnosis.y',
       'Histology.classification.y', 'radiotherapy.y', 'tumor_size.y',
       'surgery_method.y', 'lymph_node.y', 'AJCC8TH_N.y', 'AJCC8TH_M.y',
       'AJCC8TH_T.y', 'AJCC8TH_STAGE.y', 'latency_month', 'label'],
      dtype='object')
```

In [13]:
```
data.head(10)#.x代表第一次诊断 .y代表第二次诊断
```

Out[13]:

	Patient.ID	Age.recode.with..1.year.olds.x	Sex.x	Year.of.diagnosis.x	Race.recode..W..B..AI..API..x	Origin.recode.NHIA..Hispanic..Non.Hisp..x	Primary.
0	899968	70-74 years	Male	2004	Asian or Pacific Islander	Non-Spanish-Hispanic-Latino	C34.1-L
1	916035	70-74 years	Male	2004	White	Non-Spanish-Hispanic-Latino	C34.3-L
2	939226	70-74 years	Male	2004	White	Non-Spanish-Hispanic-Latino	C34.1-L
3	943384	65-69 years	Female	2004	White	Non-Spanish-Hispanic-Latino	C3
4	950080	70-74 years	Male	2004	White	Non-Spanish-Hispanic-Latino	C34.1-L
5	955185	60-64 years	Male	2004	Asian or Pacific Islander	Non-Spanish-Hispanic-Latino	C34.1-L
6	958600	65-69 years	Male	2004	Asian or Pacific Islander	Non-Spanish-Hispanic-Latino	C3
7	960531	55-59 years	Female	2004	Black	Non-Spanish-Hispanic-Latino	C34.1-L
8	961167	60-64 years	Male	2004	White	Spanish-Hispanic-Latino	C34.3-L
9	962230	75-79 years	Female	2004	Black	Non-Spanish-Hispanic-Latino	C34.3-L

10 rows × 63 columns

0.1.5 数据预处理

In [14]:
```
#淋巴结检测个数
data['Regional.nodes.examined..1988...x']=data['Regional.nodes.examined..1988...x'].replace([95,96,97,98,99],None)
data['Regional.nodes.examined..1988...y']=data['Regional.nodes.examined..1988...y'].replace([95,96,97,98,99],None)
data['Regional.nodes.positive..1988...x']=data['Regional.nodes.positive..1988...x'].replace([95,96,97,98,99],None)
data['Regional.nodes.positive..1988...y']=data['Regional.nodes.positive..1988...y'].replace([95,96,97,98,99],None)
```

In [15]:
```
#肿瘤大小
data['tumor_size.x']=data['tumor_size.x'].replace('Unknown',None)
data['tumor_size.y']=data['tumor_size.y'].replace('Unknown',None)
```

In [16]:
```
#添加特征 两次肿瘤是否同侧
data['same_lat']=None
data.loc[data['Laterality.x']==data['Laterality.y'],'same_lat']=1
data.loc[data['Laterality.x']!=data['Laterality.y'],'same_lat']=0
data['same_lat'].value_counts()
```

Out[16]:
```
0    2445
1    1433
Name: same_lat, dtype: int64
```

In [17]:
```
#将变量分为数值型、有序变量和无序变量
noneed_sort_label=['Sex.x','Race.recode..W..B..AI..API..x','Origin.recode.NHIA..Hispanic..Non.Hisp..x','Primary.Site...labeled.x','Lat
                   'RX.Summ..Surg.Rad.Seq.x','Reason.no.cancer.directed.surgery.x','Radiation.recode.x','Chemotherapy.recode..yes..no.unk..
                   'Marital.status.at.diagnosis.x','Histology.classification.x', 'radiotherapy.x','surgery_method.x', 'lymph_node.x',#.x
                   'Primary.Site...labeled.y','Laterality.y','Diagnostic.Confirmation.y','RX.Summ..Scope.Reg.LN.Sur..2003...y','RX.Summ..Su
                   'RX.Summ..Surg.Rad.Seq.y','Reason.no.cancer.directed.surgery.y','Radiation.recode.y','Chemotherapy.recode..yes..no.unk..
                   'Marital.status.at.diagnosis.y','Histology.classification.y', 'radiotherapy.y','surgery_method.y', 'lymph_node.y','same_
needsort_label=['Age.recode.with..1.year.olds.x','Grade..thru.2017..x','SEER.Combined.Summary.Stage.2000..2004.2017..x', 'RX.Summ..Sur
               'Age.recode.with..1.year.olds.y','Grade..thru.2017..y','SEER.Combined.Summary.Stage.2000..2004.2017..y', 'RX.Summ..Surg.Pr
numeric_label=['Year.of.diagnosis.x','Regional.nodes.examined..1988...x','Regional.nodes.positive..1988...x','tumor_size.x','Year.of.d
     'Regional.nodes.examined..1988...y','Regional.nodes.positive..1988...y','tumor_size.y','latency_month']
```

In [18]:
```
#有序变量则需按顺序标签
data['Age.recode.with..1.year.olds.x']=data['Age.recode.with..1.year.olds.x'].apply(lambda x: ['01-04 years','20-24 years','25-29 yea
data['Age.recode.with..1.year.olds.y']=data['Age.recode.with..1.year.olds.y'].apply(lambda x: ['01-04 years','20-24 years','25-29 yea
data['Grade..thru.2017..x']=data['Grade..thru.2017..x'].apply(lambda x: ['Well differentiated; Grade I','Moderately differentiated; G
data['Grade..thru.2017..y']=data['Grade..thru.2017..y'].apply(lambda x: ['Well differentiated; Grade I','Moderately differentiated; G
data['SEER.Combined.Summary.Stage.2000..2004.2017..x']=data['SEER.Combined.Summary.Stage.2000..2004.2017..x'].apply(lambda x: ['Local
data['SEER.Combined.Summary.Stage.2000..2004.2017..y']=data['SEER.Combined.Summary.Stage.2000..2004.2017..y'].apply(lambda x: ['Local
data['AJCC8TH_T.x']=data['AJCC8TH_T.x'].apply(lambda x: ['T0','T1a','T1b','T1NOS','T1c','T2a','T2NOS','T2b','T3','T3/T4','TX','T4'].i
data['AJCC8TH_T.y']=data['AJCC8TH_T.y'].apply(lambda x: ['T0','T1a','T1b','T1NOS','T1c','T2a','T2NOS','T2b','T3','T3/T4','TX','T4'].i
data['AJCC8TH_N.x']=data['AJCC8TH_N.x'].apply(lambda x: ['N0','N1','N2','NX','N3'].index(x))
data['AJCC8TH_N.y']=data['AJCC8TH_N.y'].apply(lambda x: ['N0','N1','N2','NX','N3'].index(x))
data['AJCC8TH_M.x']=data['AJCC8TH_M.x'].apply(lambda x: ['M0','MX','M1'].index(x))
data['AJCC8TH_M.y']=data['AJCC8TH_M.y'].apply(lambda x: ['M0','MX','M1'].index(x))
data['AJCC8TH_STAGE.x']=data['AJCC8TH_STAGE.x'].apply(lambda x: ['IA1','IA2','IA3','INOS','IB','IIA','IIB','IIIA','IIIA/B','IIIB','II
data['AJCC8TH_STAGE.y']=data['AJCC8TH_STAGE.y'].apply(lambda x: ['IA1','IA2','IA3','INOS','IB','IIA','IIB','IIIA','IIIA/B','IIIB','II
```

In [19]:
```
#对无序变量进行labelencoder标签
for i in noneed_sort_label:
    data[i]=preprocessing.LabelEncoder().fit_transform(data[i])
```

In [20]:
```
#并类型转换,数值型变量填补缺失值
data['tumor_size.x']=data['tumor_size.x'].astype(float)
data['tumor_size.y']=data['tumor_size.y'].astype(float)
for i in numeric_label:
  # data[i]=data[i].fillna(data[i].mean())
  data[i]=data[i].fillna(data[i].median())
```

#重要特征选择(可跳过)

In [21]:
```
imp_fea0 = ['Patient.ID','Reason.no.cancer.directed.surgery.x','Histology.classification.x','AJCC8TH_N.x','AJCC8TH_STAGE.x',
            'Grade..thru.2017..y','SEER.Combined.Summary.Stage.2000..2004.2017..y','Reason.no.cancer.directed.surgery.y','Histology.cl
            'AJCC8TH_N.y','AJCC8TH_M.y','AJCC8TH_STAGE.y' ,'Sex.x','label']
imp_fea1 = ['Patient.ID','Sex.x','Year.of.diagnosis.x','Primary.Site...labeled.x','Grade..thru.2017..x',
        'Laterality.x','SEER.Combined.Summary.Stage.2000..2004.2017..x','RX.Summ..Scope.Reg.LN.Sur..2003...x',
        'Reason.no.cancer.directed.surgery.x','Radiation.recode.x','Chemotherapy.recode..yes..no.unk..y',
        'RX.Summ..Systemic.Sur.Seq.x','Histology.classification.x','tumor_size.x','lymph_node.x',
        'AJCC8TH_N.x','Grade..thru.2017..y','Laterality.y','Diagnostic.Confirmation.y','SEER.Combined.Summary.Stage.2000..2004.2017..y
        'RX.Summ..Scope.Reg.LN.Sur..2003...y','Reason.no.cancer.directed.surgery.y','Radiation.recode.y',
        'Primary.by.international.rules.y','Histology.classification.y','tumor_size.y',
        'lymph_node.y','AJCC8TH_N.y','AJCC8TH_M.y','AJCC8TH_T.y','AJCC8TH_STAGE.y','latency_month','label']
imp_fea2 =['Patient.ID','Age.recode.with..1.year.olds.x','Sex.x','Race.recode..W..B..AI..API..x','Grade..thru.2017..x',
'SEER.Combined.Summary.Stage.2000..2004.2017..x','RX.Summ..Surg.Prim.Site..1998...x','RX.Summ..Scope.Reg.LN.Sur..2003...x',
'RX.Summ..Surg.Rad.Seq.x','Chemotherapy.recode..yes..no.unk..x','RX.Summ..Systemic.Sur.Seq.x','Type.of.Reporting.Source.x',
'Histology.classification.x','Year.of.diagnosis.y','Grade..thru.2017..y',
'SEER.Combined.Summary.Stage.2000..2004.2017..y','RX.Summ..Scope.Reg.LN.Sur..2003...y','Reason.no.cancer.directed.surgery.y','Chemothe
'Primary.by.international.rules.y','Marital.status.at.diagnosis.y','Histology.classification.y','surgery_method.y','AJCC8TH_STAGE.y','
#imp_fea分别问lasso、cox和逐步回归法得到的最优特征子集
data= data[:data.shape[0]][imp_fea0]
data.shape #最重要的n个特征
```

Out[21]: (3878, 15)

In [22]:
```
#拆分训练测试集
train_data , test_data = train_test_split(data, test_size = 0.25,random_state=3)
id, label = 'Patient.ID', 'label'
```

In [23]:
```
#对数值型变量进行标准化 （标准化后决策树有点问题）
# for i in numeric_label:
#     train_data[i]=preprocessing.scale(train_data[i])
```

In [24]:
```
X_train=train_data.drop(columns=[id,label]).values
target=y_train=train_data[label].values

X_test=test_data.drop(columns=[id,label]).values
y_test = test_data[label].values
use_feature=train_data.drop(columns=[id,label]).columns
print(X_train.shape)
#train_shape = X_train.shape[0]
```

```
(2908, 13)
```

#定义函数

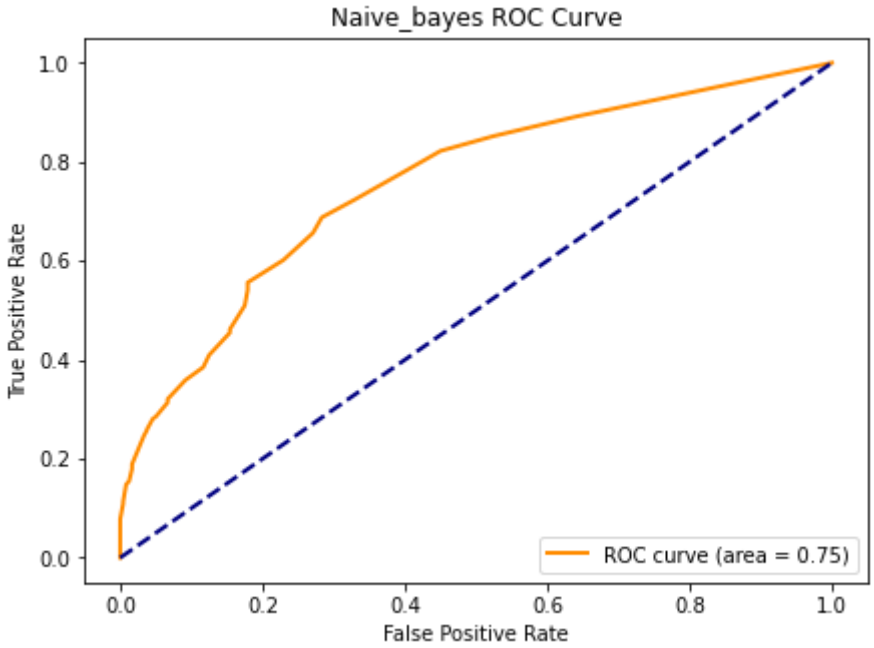
In [25]:
```
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

def show_scores(y_pred1,y_pred2):
    print("accuracy_score is:",metrics.accuracy_score(y_test,y_pred1))
    print("roc_auc_score is:",metrics.roc_auc_score(y_test,y_pred2))
    print("precision_score is:",metrics.precision_score(y_test,y_pred1))
    print("recall_score is:",metrics.recall_score(y_test,y_pred1))
    print("f1_score is:",metrics.f1_score(y_test,y_pred1))
    print("classification_report:")
    print(metrics.classification_report(y_test,y_pred1))


def plot_roc(y_pred2,name):
    fpr, tpr, thersholds = roc_curve(y_test, y_pred2, pos_label=1)
    fpr,tpr,threshold = roc_curve(y_test, y_pred2) ###计算真阳率和假阳率
    roc_auc = auc(fpr,tpr) ###计算auc的值
    plt.figure()
    lw = 2
    plt.figure(figsize=(7,5))
    plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc) ###假阳率为横坐标，真阳率为纵坐标做曲线
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    # for i, value in enumerate(thersholds):
    #     print("%f %f %f" % (fpr[i], tpr[i], value))
    plt.xlim([-0.05, 1.05])  # 设置x、y轴的上下限，以免和边缘重合，更好的观察图像的整体
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')  # 可以使用中文，但需要导入一些库即字体
    plt.title(name)
    plt.legend(loc="lower right")
    plt.show()
    #plt.savefig('./naive_bayes.jpg')#保存图片


#混淆矩阵
def show_confusion_matrix(model,y_pred1):
    y_pred = model.predict(X_test)
    confmat = confusion_matrix(y_true=y_test,y_pred=y_pred1)
    fig,ax = plt.subplots(figsize=(2.5,2.5))
```

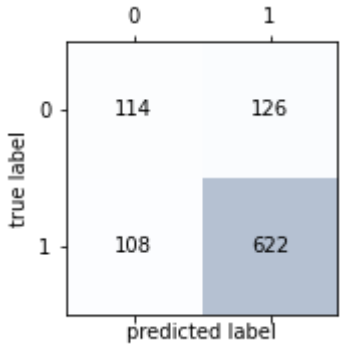
```
    ax.matshow(confmat, cmap=plt.cm.Blues,alpha=0.3)
    for i in range(confmat.shape[0]):
        for j in range(confmat.shape[1]):
            ax.text(x=j,y=i,s=confmat[i,j],va='center',ha='center')
    plt.xlabel('predicted label')
    plt.ylabel('true label')
    plt.show()
```

朴素贝叶斯

In [26]:
```
# 朴素贝叶斯
from sklearn.naive_bayes import GaussianNB,CategoricalNB,MultinomialNB,ComplementNB,BernoulliNB
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer

parameters={'binarize':[*np.arange(0.0,10,0.1)]}
GS=GridSearchCV(BernoulliNB(),parameters,cv=10,n_jobs=-1)
GS.fit(X_train,y_train)
# print(GS.best_params_)

y_pred2=GS.predict_proba(X_test)[:,1]#预测概率
y_pred1=GS.predict(X_test)#预测标签
print(GS.best_params_)
# NB = BernoulliNB ()
# NB=NB.fit(X_train,y_train)
# y_pred2=NB.predict_proba(X_test)[:,1]
# y_pred1=NB.predict(X_test)
# roc_auc_score (y_test, y_pred)
show_scores(y_pred1,y_pred2)
plot_roc(y_pred2,'Naive_bayes ROC Curve')
show_confusion_matrix(GS,y_pred1)
```

```
{'binarize': 5.800000000000001}
accuracy_score is: 0.7587628865979381
roc_auc_score is: 0.7540239726027398
precision_score is: 0.8315508021390374
recall_score is: 0.852054794520548
f1_score is: 0.8416779431664413
classification_report:
              precision    recall  f1-score   support

           0       0.51      0.47      0.49       240
           1       0.83      0.85      0.84       730

    accuracy                           0.76       970
   macro avg       0.67      0.66      0.67       970
weighted avg       0.75      0.76      0.76       970


<Figure size 432x288 with 0 Axes>
```


Naive_bayes ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)
0.0
0.2
0.4
0.6
0.8
1.0


0
1
114
126
108
622
true label
predicted label

决策树

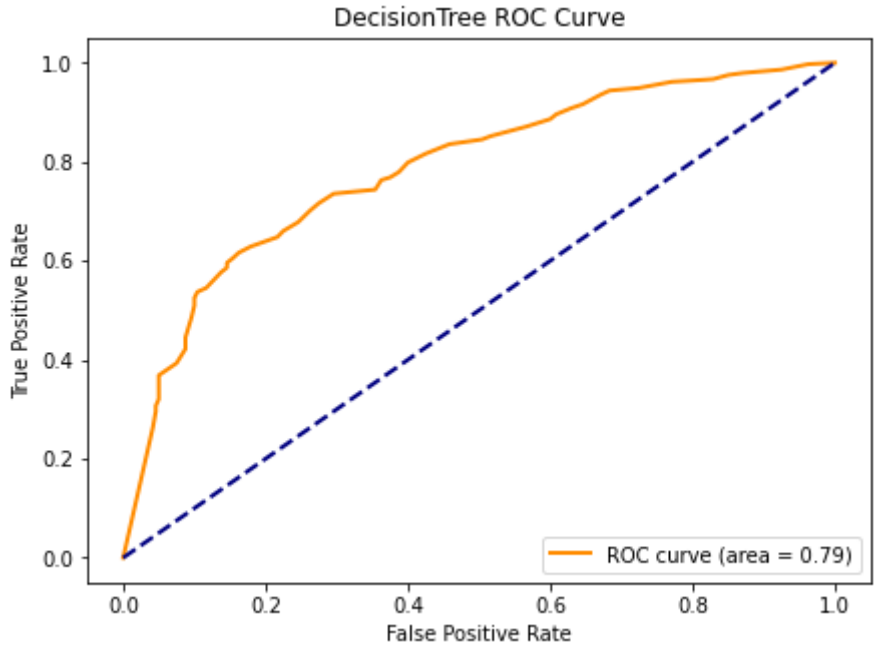
In [27]:
```
###决策树网格搜索
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer
from sklearn.tree import DecisionTreeClassifier
parameters = {'splitter':('best','random')
              ,'criterion':("gini","entropy")
              ,"max_depth":[*range(1,50)]
              ,'min_samples_leaf':[*range(1,51,2)]
}
tree = DecisionTreeClassifier(random_state=25)
GS = GridSearchCV(tree, parameters, cv=10) # cv交叉验证
GS.fit(X_train,y_train)
print(GS.best_params_)

#{'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 33, 'splitter': 'best'}

# print(GS.best_score_)  # 0.79
# tree = DecisionTreeClassifier(random_state=25,max_depth=10,min_samples_leaf=31,splitter='best',criterion='gini')
# tree=tree.fit(X_train,y_train)
y_pred=GS.predict_proba(X_test)[:,1]
y_pred
# metrics.accuracy_score(y_test,y_pred)
```

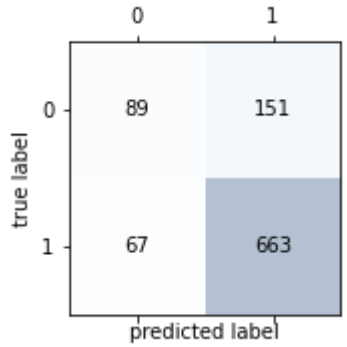
```
{'criterion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 29, 'splitter': 'random'}
```

Out[27]: array([0.52173913, 0.3744186 , 1.   , 0.52173913, 0.61038961,
0.61038961, 0.46376812, 0.77192982, 0.3744186 , 0.97674419,

In [28]:
```
tree = DecisionTreeClassifier(random_state=25,max_depth=10,min_samples_leaf=33,splitter='best',criterion='gini')
tree=tree.fit(X_train,y_train)
# y_pred=clf.predict_proba(X_test)[:,1]
y_pred2=tree.predict_proba(X_test)[:,1]#预测概率
y_pred1=tree.predict(X_test)#预测标签
show_scores(y_pred1,y_pred2)
plot_roc(y_pred2,'DecisionTree ROC Curve')
show_confusion_matrix(tree,y_pred1)
```

```
accuracy_score is: 0.7752577319587629
roc_auc_score is: 0.7853510273972603
precision_score is: 0.8144963144963145
recall_score is: 0.9082191780821918
f1_score is: 0.8588082901554404
classification_report:
              precision    recall  f1-score   support

           0       0.57      0.37      0.45       240
           1       0.81      0.91      0.86       730

    accuracy                           0.78       970
   macro avg       0.69      0.64      0.65       970
weighted avg       0.75      0.78      0.76       970


<Figure size 432x288 with 0 Axes>
```


DecisionTree ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.79)
0.0
0.2
0.4
0.6
0.8
1.0


0
1
89
151
67
663
true label
predicted label

In [29]:
```
#展示树结构
import graphviz,sklearn.tree
dot_data = sklearn.tree.export_graphviz(tree, out_file=None)
graph = graphviz.Source(dot_data)
dot_data = sklearn.tree.export_graphviz(tree, out_file=None,
                        feature_names=train_data.drop(columns=[id,label]).columns,
                        class_names='label',
                        filled=True, rounded=True,
                        special_characters=True)
graph = graphviz.Source(dot_data)
```

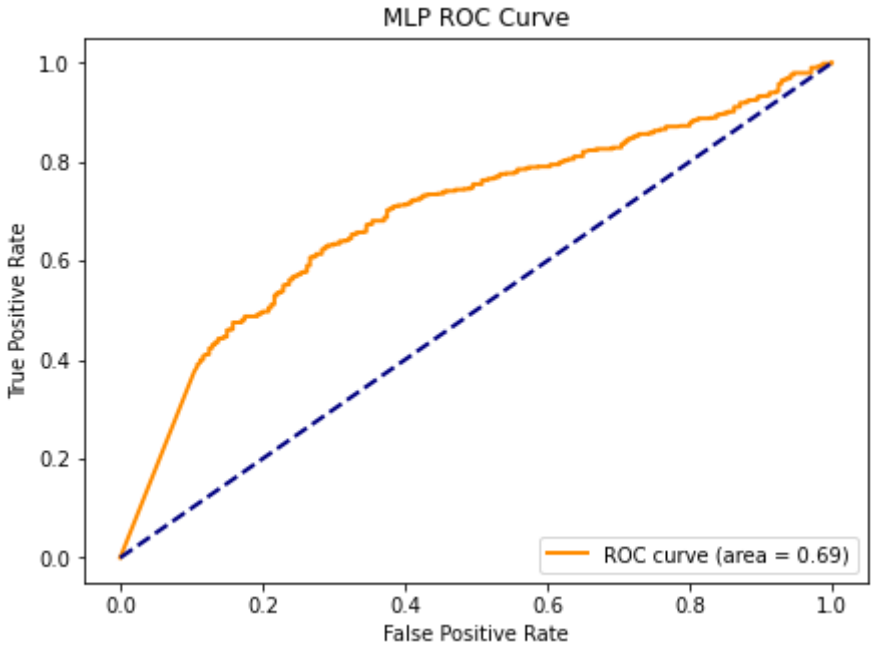
#多层感知机

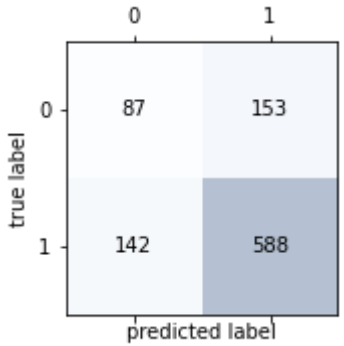
In [30]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
param_grid = {"hidden_layer_sizes": [(62,), (62,124,30),(62,124,50,10),(62,248,124,30)],
              "solver": ['adam', 'sgd', 'lbfgs'],
              "max_iter": [100,500,1000,2000],'learning_rate_init':[0.001, 0.01, 0.1, 0.2]
                                }
# param_grid = [{'svc__C':param_range,'svc__kernel':['linear','rbf'],'svc__gamma':param_range}]
gs = RandomizedSearchCV(estimator=MLPClassifier(), param_distributions=param_grid,scoring='accuracy',cv=10,n_jobs=-1)
gs = gs.fit(X_train,y_train)
print(gs.best_score_)
print(gs.best_params_)
```

```
0.7771762057115772
{'solver': 'adam', 'max_iter': 1000, 'learning_rate_init': 0.1, 'hidden_layer_sizes': (62,)}
```

In [31]:
```
mlp = MLPClassifier(hidden_layer_sizes=(62, 124, 50,10),solver='lbfgs',max_iter=1000,learning_rate_init=0.01)
mlp=mlp.fit(X_train,y_train)
# y_pred=clf.predict_proba(X_test)[:,1]
y_pred2=mlp.predict_proba(X_test)[:,1]#预测概率
y_pred1=mlp.predict(X_test)#预测标签
show_scores(y_pred1,y_pred2)
plot_roc(y_pred2,'MLP ROC Curve')
show_confusion_matrix(mlp,y_pred1)
```

```
accuracy_score is: 0.6958762886597938
roc_auc_score is: 0.6928995433789955
precision_score is: 0.7935222672064778
recall_score is: 0.8054794520547945
f1_score is: 0.7994561522773624
classification_report:
              precision    recall  f1-score   support

           0       0.38      0.36      0.37       240
           1       0.79      0.81      0.80       730

    accuracy                           0.70       970
   macro avg       0.59      0.58      0.59       970
weighted avg       0.69      0.70      0.69       970


<Figure size 432x288 with 0 Axes>
```


MLP ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.69)
0.0
0.2
0.4
0.6
0.8
1.0


0
1
true label
predicted label
87
153
142
588

支持向量机

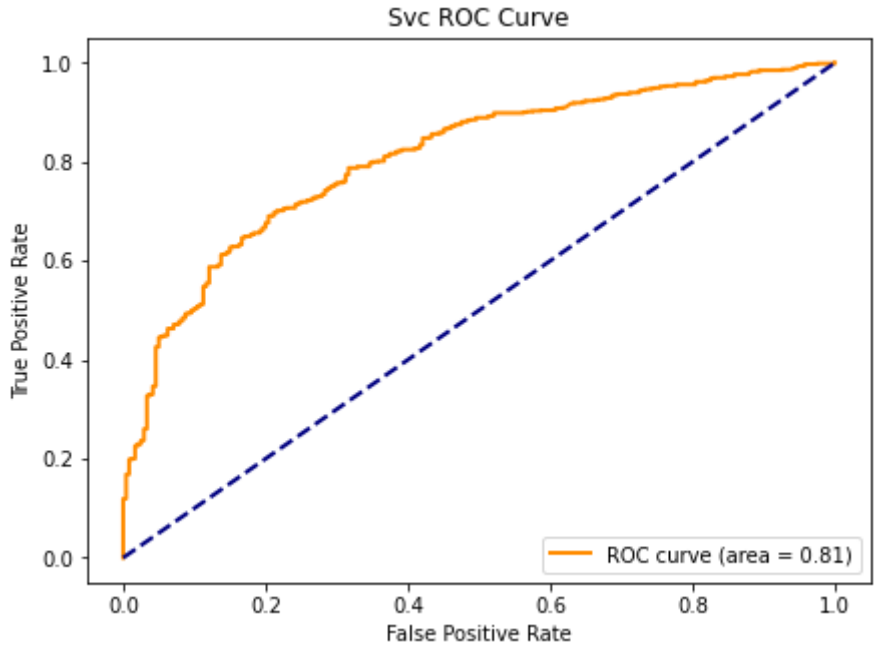
In [32]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
'''
C:正则化参数。正则化的强度与C成反比。必须严格为正。惩罚是平方的l2惩罚。
kernel:{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, 默认='rbf'
degree:多项式和的阶数
gamma:“ rbf”, “ poly”和“ Sigmoid”的内核系数。
shrinking:是否软间隔分类，默认true
'''
from sklearn.model_selection import RandomizedSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
pipe_svc = make_pipeline(StandardScaler(),SVC(random_state=1))
param_range = [0.0001,0.001,0.01,0.1,1.0,10.0,100.0]
param_grid = [{'svc__C':param_range,'svc__kernel':['linear']},{'svc__C':param_range,'svc__gamma':param_range,'svc__kernel':['rbf']}]
# param_grid = [{'svc__C':param_range,'svc__kernel':['linear','rbf'],'svc__gamma':param_range}]
gs = RandomizedSearchCV(estimator=pipe_svc, param_distributions=param_grid,scoring='accuracy',cv=10,n_jobs=-1)
gs = gs.fit(X_train,y_train)
print(gs.best_score_)
print(gs.best_params_)
```

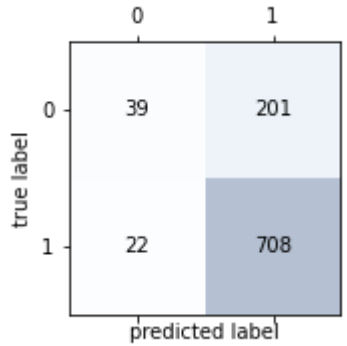
```
0.7696077734328711
{'svc__kernel': 'rbf', 'svc__gamma': 0.1, 'svc__C': 0.1}
```

In [33]:

```
svc = make_pipeline(StandardScaler(), SVC(kernel='rbf',C=10,gamma=0.001,probability=True))
svc=svc.fit(X_train,y_train)
y_pred1=svc.predict(X_test)
y_pred2=svc.predict_proba(X_test)[:,1]
show_scores(y_pred1,y_pred2)
plot_roc(y_pred2,'Svc ROC Curve')
show_confusion_matrix(svc,y_pred1)
```

```
accuracy_score is: 0.7701030927835052
roc_auc_score is: 0.8077111872146119
precision_score is: 0.7788778877887789
recall_score is: 0.9698630136986301
f1_score is: 0.8639414276998171
classification_report:
              precision    recall  f1-score   support

           0       0.64      0.16      0.26       240
           1       0.78      0.97      0.86       730

    accuracy                           0.77       970
   macro avg       0.71      0.57      0.56       970
weighted avg       0.74      0.77      0.71       970


<Figure size 432x288 with 0 Axes>
```





1 集成学习模型

1.lightGBM

In [34]:

```
##### lightgbm #
#lightGBM决策树
lightgbm_param = {
'num_leaves': 7,
'min_data_in_leaf': 20, #叶子可能具有的最小记录数
'objective':'binary',
'max_depth': -1,
'learning_rate': 0.003,
"boosting": "gbdt", #用gbdt算法
"feature_fraction": 0.18, #例如 0.18时，意味着在每次迭代中随机选择18%的参数来建树
"bagging_freq": 1,
"bagging_fraction": 0.55, #每次迭代时用的数据比例
"bagging_seed": 14,
"metric": 'auc',
"lambda_l1": 0.1,
"lambda_l2": 0.2,
"verbosity": -1}
folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=4)   #交叉切分：5
oof_lightgbm = np.zeros(len(X_train))
predictions_lightgbm = np.zeros(len(X_test))

for fold_, (trn_idx, val_idx) in enumerate(folds.split(X_train, y_train)):
    import lightgbm
    from lightgbm import Dataset
    print("fold n° {}".format(fold_+1))
    trn_data = lightgbm.Dataset(X_train[trn_idx], y_train[trn_idx])
    val_data = lightgbm.Dataset(X_train[val_idx], y_train[val_idx])#train:val=4:1

    num_round = 10000
    lightgbm = lightgbm.train(lightgbm_param, trn_data, num_round, valid_sets = [trn_data, val_data], verbose_eval=500, early_stopping
    oof_lightgbm[val_idx] = lightgbm.predict(X_train[val_idx], num_iteration=lightgbm.best_iteration)
    predictions_lightgbm += lightgbm.predict(X_test, num_iteration=lightgbm.best_iteration) / folds.n_splits
    y_pred = (predictions_lightgbm >= 0.5)*1
print("CV score: {:<8.8f}".format(mean_squared_error(oof_lightgbm, target)))
```

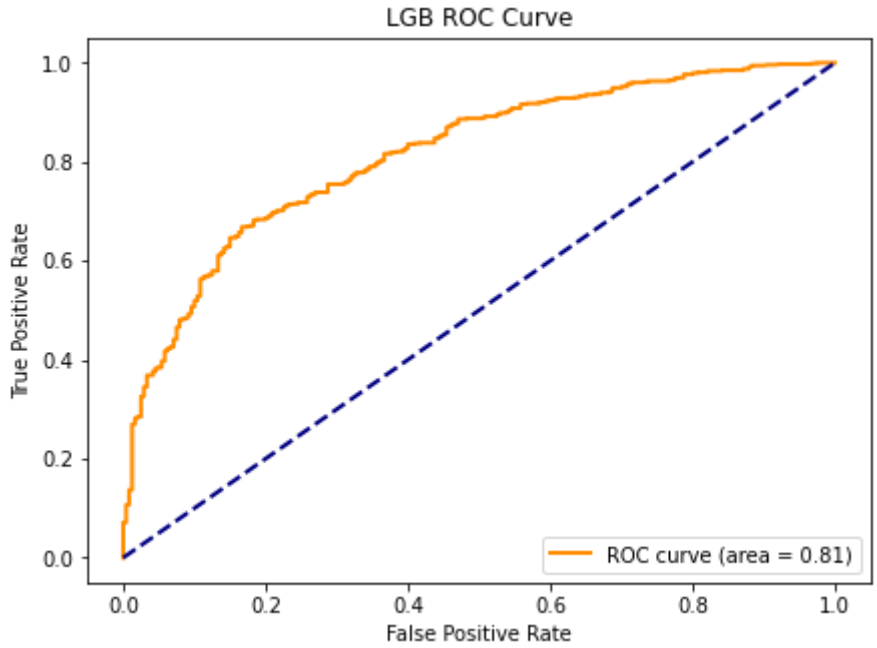
```
fold n° 1
Training until validation scores don't improve for 800 rounds
[500]   training's auc: 0.820416        valid_1's auc: 0.812371
[1000]  training's auc: 0.826682        valid_1's auc: 0.813661
[1500]  training's auc: 0.831615        valid_1's auc: 0.814404
```

```
[2000]  training's auc: 0.836076        valid_1's auc: 0.814434
[2500]  training's auc: 0.840255        valid_1's auc: 0.814116
[3000]  training's auc: 0.843783        valid_1's auc: 0.813964
Early stopping, best iteration is:
[2254]  training's auc: 0.838464        valid_1's auc: 0.814601
fold n° 2
Training until validation scores don't improve for 800 rounds
[500]   training's auc: 0.818843        valid_1's auc: 0.819653
[1000]  training's auc: 0.825289        valid_1's auc: 0.820822
[1500]  training's auc: 0.831109        valid_1's auc: 0.820822
[2000]  training's auc: 0.836007        valid_1's auc: 0.821292
[2500]  training's auc: 0.840242        valid_1's auc: 0.821049
[3000]  training's auc: 0.844006        valid_1's auc: 0.819699
Early stopping, best iteration is:
[2241]  training's auc: 0.838148        valid_1's auc: 0.821838
fold n° 3
Training until validation scores don't improve for 800 rounds
[500]   training's auc: 0.825069        valid_1's auc: 0.775178
Early stopping, best iteration is:
[27]    training's auc: 0.821195        valid_1's auc: 0.777044
fold n° 4
Training until validation scores don't improve for 800 rounds
[500]   training's auc: 0.819796        valid_1's auc: 0.807712
Early stopping, best iteration is:
[27]    training's auc: 0.814012        valid_1's auc: 0.813042
fold n° 5
Training until validation scores don't improve for 800 rounds
[500]   training's auc: 0.816378        valid_1's auc: 0.813354
[1000]  training's auc: 0.821761        valid_1's auc: 0.817307
[1500]  training's auc: 0.826324        valid_1's auc: 0.821413
[2000]  training's auc: 0.830476        valid_1's auc: 0.8252
[2500]  training's auc: 0.833758        valid_1's auc: 0.827238
[3000]  training's auc: 0.837127        valid_1's auc: 0.828728
[3500]  training's auc: 0.839877        valid_1's auc: 0.829367
[4000]  training's auc: 0.842436        valid_1's auc: 0.830492
[4500]  training's auc: 0.84495 valid_1's auc: 0.831739
[5000]  training's auc: 0.847286        valid_1's auc: 0.831526
[5500]  training's auc: 0.849522        valid_1's auc: 0.832241
[6000]  training's auc: 0.851619        valid_1's auc: 0.832971
[6500]  training's auc: 0.853681        valid_1's auc: 0.83329
[7000]  training's auc: 0.855386        valid_1's auc: 0.83364
[7500]  training's auc: 0.857027        valid_1's auc: 0.834051
```

```
[8000]  training's auc: 0.858659        valid_1's auc: 0.83402
[8500]  training's auc: 0.860366        valid_1's auc: 0.834051
[9000]  training's auc: 0.861843        valid_1's auc: 0.833807
Early stopping, best iteration is:
[8366]  training's auc: 0.859833        valid_1's auc: 0.834431
CV score: 0.16307050
```

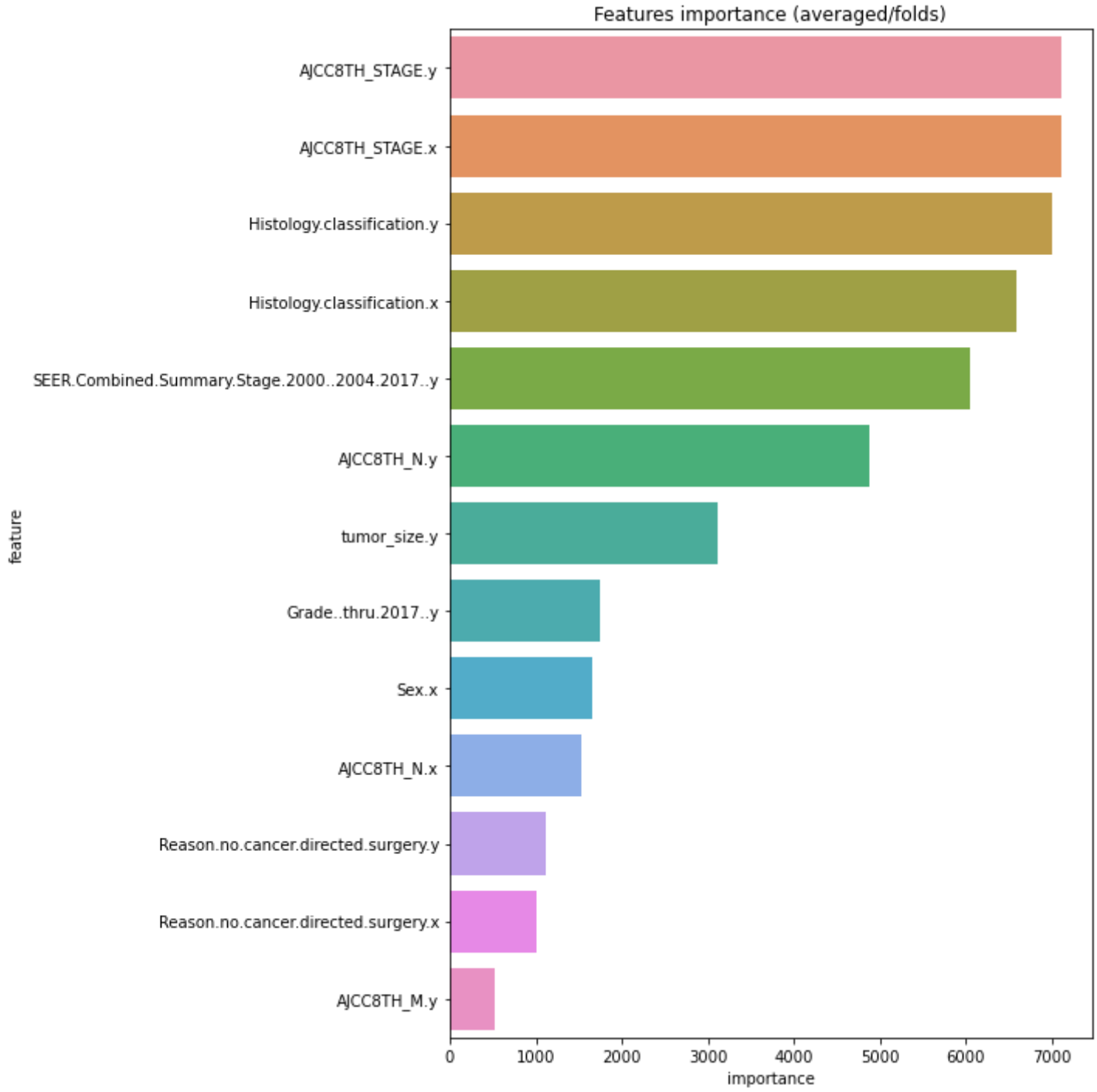
In [35]:
```
# from sklearn.metrics import roc_curve, auc
# import matplotlib.pyplot as plt
# fpr, tpr, thersholds = roc_curve(y_test,predictions_lightgbm, pos_label=1)
show_scores(y_pred,predictions_lightgbm)
plot_roc(predictions_lightgbm,'LGB ROC Curve')
```

```
accuracy_score is: 0.7752577319587629
roc_auc_score is: 0.814925799086758
precision_score is: 0.7764578833693304
recall_score is: 0.9849315068493151
f1_score is: 0.8683574879227053
classification_report:
              precision    recall  f1-score   support

           0       0.75      0.14      0.23       240
           1       0.78      0.98      0.87       730

    accuracy                           0.78       970
   macro avg       0.76      0.56      0.55       970
weighted avg       0.77      0.78      0.71       970


<Figure size 432x288 with 0 Axes>
```


LGB ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.81)
0.0
0.2
0.4
0.6
0.8
1.0

In [36]:

```
#---------------特征重要性
pd.set_option('display.max_columns', None)
#显示所有行
pd.set_option('display.max_rows', None)
#设置value的显示长度为100，默认为50
pd.set_option('max_colwidth',100)
df = pd.DataFrame(data[use_feature].columns.tolist(), columns=['feature'])
df['importance']=list(lightgbm.feature_importance())
df = df.sort_values(by='importance',ascending=False)
plt.figure(figsize=(10,10))
sns.barplot(x="importance", y="feature", data=df.head(15))
plt.title('Features importance (averaged/folds)')
plt.tight_layout()
# plt.savefig('./featurelgb.jpg')
```


Features importance (averaged/folds)
AJCC8TH_STAGE.y
AJCC8TH_STAGE.x
Histology.classification.y
Histology.classification.x
SEER.Combined.Summary.Stage.2000..2004.2017..y
AJCC8TH_N.y
tumor_size.y
Grade..thru.2017..y
Sex.x
AJCC8TH_N.x
Reason.no.cancer.directed.surgery.y
Reason.no.cancer.directed.surgery.x
AJCC8TH_M.y
feature
0
1000
2000
3000
4000
5000
6000
7000
importance

2.xgboost

In [37]:

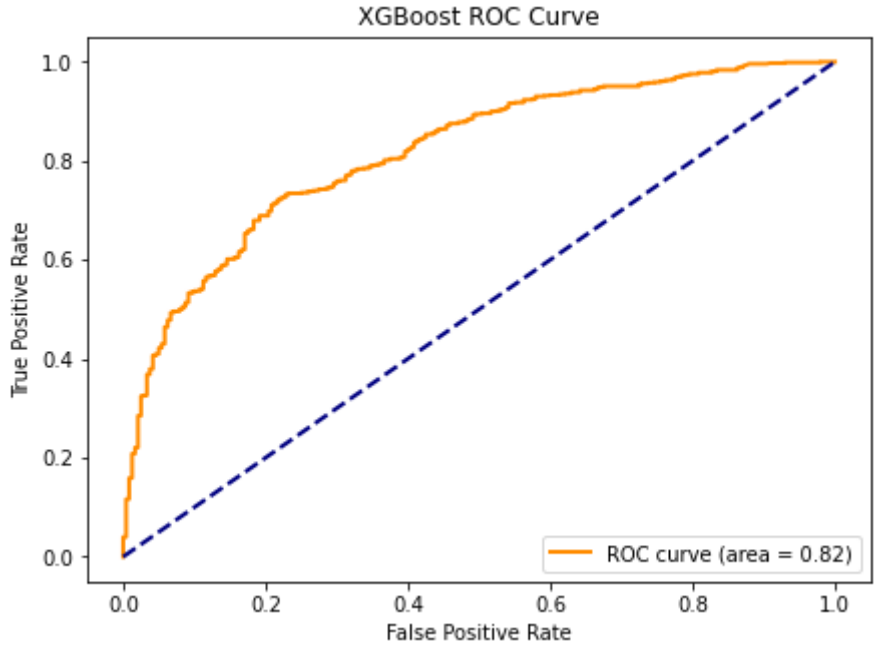
```
##### xgboost
#xgboostoost
xgboost_params = {'eta': 0.02,  #lr
                  'max_depth': 6,
                  'min_child_weight':3,#最小叶子节点样本权重和
                  'gamma':0, #指定节点分裂所需的最小损失函数下降值。
                  'subsample': 0.7,  #控制对于每棵树，随机采样的比例
                  'colsample_bytree': 0.3,  #用来控制每棵随机采样的列数的占比 (每一列是一个特征)。
                  'lambda':2,
                  'objective': 'binary:logitraw',
                  'eval_metric': 'auc',
                  'silent': True,
                  'nthread': -1}


folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=2019)
oof_xgboost = np.zeros(len(X_train))
predictions_xgboost = np.zeros(len(X_test))

for fold_, (trn_idx, val_idx) in enumerate(folds.split(X_train, y_train)):
    import xgboost
    from xgboost import DMatrix
    print("fold n° {}".format(fold_+1))
    trn_data = xgboost.DMatrix(X_train[trn_idx], y_train[trn_idx])
    val_data = xgboost.DMatrix(X_train[val_idx], y_train[val_idx])
    watchlist = [(trn_data, 'train'), (val_data, 'valid_data')]
    mod = xgboost.train(dtrain=trn_data, num_boost_round=3000, evals=watchlist, early_stopping_rounds=600, verbose_eval=500, params=x
    import xgboost
    from xgboost import *
    oof_xgboost[val_idx] = mod.predict(xgboost.DMatrix(X_train[val_idx]), ntree_limit=mod.best_ntree_limit)
    predictions_xgboost += mod.predict(xgboost.DMatrix(X_test), ntree_limit=mod.best_ntree_limit) / folds.n_splits

print("CV score: {:<8.8f}".format(mean_squared_error(oof_xgboost, target)))
y_pred = (predictions_xgboost >= 0.5)*1
print("auc is:",metrics.roc_auc_score(y_test,y_pred))
```

```
fold n° 1
[15:47:56] WARNING: C:/Users/administrator/workspace/xgboost-win64_release_1.6.0/src/learner.cc:627:
Parameters: { "silent" } might not be used.
```

```
  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[0]     train-auc:0.75983       valid_data-auc:0.74173
[500]   train-auc:0.87953       valid_data-auc:0.81328
[681]   train-auc:0.89102       valid_data-auc:0.80973
fold n° 2
[15:47:57] WARNING: C:/Users/administrator/workspace/xgboost-win64_release_1.6.0/src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

In [38]:
```
show_scores(y_pred,predictions_xgboost)
plot_roc(predictions_xgboost,'XGBoost ROC Curve')
```

```
accuracy_score is: 0.7793814432989691
roc_auc_score is: 0.8170433789954339
precision_score is: 0.8603351955307262
recall_score is: 0.8438356164383561
f1_score is: 0.8520055325034579
classification_report:
              precision    recall  f1-score   support

           0       0.55      0.58      0.57       240
           1       0.86      0.84      0.85       730

    accuracy                           0.78       970
   macro avg       0.71      0.71      0.71       970
weighted avg       0.78      0.78      0.78       970


<Figure size 432x288 with 0 Axes>
```



3.RandomForestRegressor随机森林

#模型集成

In [53]:
```
train_stack2 = np.vstack([oof_xgboost,oof_lightgbm,oof_rfr]).transpose()
# transpose()函数的作用就是调换x,y,z的位置,也就是数组的索引值
test_stack2 = np.vstack([predictions_xgboost,predictions_lightgbm,predictions_rfr]).transpose()

#交叉验证:5折，重复2次
folds_stack = RepeatedKFold(n_splits=5, n_repeats=2, random_state=7)
oof_stack2 = np.zeros(train_stack2.shape[0])
predictions_lr2 = np.zeros(test_stack2.shape[0])

for fold_, (trn_idx, val_idx) in enumerate(folds_stack.split(train_stack2,target)):
    print("fold {}".format(fold_))
    trn_data, trn_y = train_stack2[trn_idx], target[trn_idx]
    val_data, val_y = train_stack2[val_idx], target[val_idx]
    #Kernel Ridge Regression
    lr2 = kr()
    lr2.fit(trn_data, trn_y)

    oof_stack2[val_idx] = lr2.predict(val_data)
    predictions_lr2 += lr2.predict(test_stack2) / 10
metrics.roc_auc_score(target, oof_stack2)
```

```
fold 0
fold 1
fold 2
fold 3
fold 4
fold 5
fold 6
fold 7
fold 8
fold 9
```

Out[53]: 0.802988288605688

Grid Search 调参方法存在的共性弊端就是：耗时；参数越多，候选值越多，耗费时间越长 所以，一般情况下，先定一个大范围，然后再细化。

In [ ]:

In [ ]: