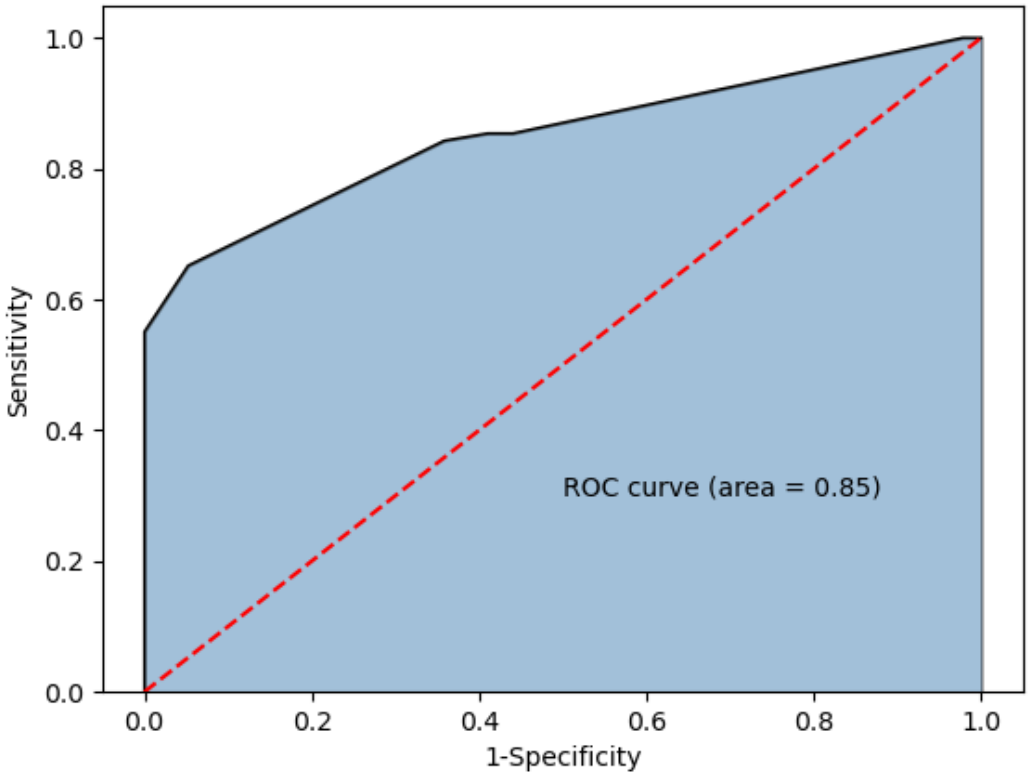
决策树实例

运行结果




```
模型在测试集的预测准确率:
0.8295964125560538
```

代码解释

代码涉及了数据预处理、决策树分类模型的建立、模型评估和绘制ROC曲线等步骤。

```
Titanic.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
Titanic.isnull().sum(axis=0)
```

删除Titanic数据集中的'PassengerId'、'Name'、'Ticket'和'Cabin'列。然后，使用isnull().sum()计算剩余自变量中的缺失值数量。

```
fillna_Titanic = []
for i in Titanic.Sex.unique():
    update = Titanic.loc[Titanic.Sex == i,].fillna(value={'Age':
Titanic.Age[Titanic.Sex == i].mean()})
    fillna_Titanic.append(update)
Titanic = pd.concat(fillna_Titanic)
```

对'Sex'列进行分组，然后分别计算每个组中乘客年龄的平均值，并将缺失值用各组的平均年龄进行填充。

```
Titanic.fillna(value={'Embarked': Titanic.Embarked.mode()[0]}, inplace=True)
Titanic.head()
```

使用Embarked变量的众数（出现频率最高的值）来填充Embarked列中的缺失值。

```
Titanic.Pclass = Titanic.Pclass.astype('category')
dummy = pd.get_dummies(Titanic[['Sex', 'Embarked', 'Pclass']])
Titanic = pd.concat([Titanic, dummy], axis=1)
Titanic.drop(['Sex', 'Embarked', 'Pclass'], inplace=True, axis=1)
Titanic.head()
```

将Pclass列的数值型变量转换为类别型变量，并使用get_dummies函数
对'Sex'、'Embarked'和'Pclass'进行哑变量处理。然后将哑变量的数据集与原始数据集Titanic水平合并，并删除原始的'Sex'、'Embarked'和'Pclass'列。

```
predictors = Titanic.columns[1:]
X_train, X_test, y

_train, y_test = model_selection.train_test_split(Titanic[predictors],
Titanic.Survived,

 test_size=0.25, random_state=1234)
```

将Titanic数据集拆分为输入特征X和目标变量y。使用train_test_split函数将数据集划分为训练集和测试集，其中测试集占比为0.25。

```
max_depth = [2, 3, 4, 5, 6]
min_samples_split = [2, 4, 6, 8]
min_samples_leaf = [2, 4, 8, 10, 12]
parameters = {'max_depth': max_depth, 'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf}
grid_dtcateg = GridSearchCV(estimator=tree.DecisionTreeClassifier(),
param_grid=parameters, cv=10)
grid_dtcateg.fit(X_train, y_train)
grid_dtcateg.best_params_
```

使用GridSearchCV函数进行网格搜索，以选择决策树分类模型中的最佳参数组合。参数max_depth、min_samples_split和min_samples_leaf分别代表树的最大深度、内部节点分裂所需的最小样本数和叶节点所需的最小样本数。使用cv=10进行10折交叉验证，找到最佳参数组合后返回。

```
CART_Class = tree.DecisionTreeClassifier(max_depth=3, min_samples_leaf=4,
min_samples_split=2)
decision_tree = CART_Class.fit(X_train, y_train)
pred = CART_Class.predict(X_test)
```

根据最佳参数值构建决策树分类模型，并使用fit函数对训练集进行拟合。然后使用predict函数对测试集进行预测，将预测结果存储在pred中。

```
print('模型在测试集的预测准确率: \n', metrics.accuracy_score(y_test, pred))
```

使用accuracy_score函数计算模型在测试集上的预测准确率，并将结果打印出来。

```
y_score = CART_Class.predict_proba(X_test)[:, 1]
fpr, tpr, threshold = metrics.roc_curve(y_test, y_score)
roc_auc = metrics.auc(fpr, tpr)

plt.stackplot(fpr, tpr, color='steelblue', alpha=0.5, edgecolor='black')
plt.plot(fpr, tpr, color='black', lw=1)
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.text(0.5, 0.3, 'ROC curve (area = %0.2f)' % roc_auc)
plt.xlabel('1-Specificity')
plt.ylabel('Sensitivity')
plt.show()
```

使用predict_proba函数获取决策树模型在测试集上预测为正例的概率，并根据真实标签和预测概率计算出ROC曲线的假正例率(fpr)和真正例率(tpr)。然后使用stackplot函数绘制ROC曲线的面积图，使用plot函数绘制边际线和对角线，使用text函数添加曲线下面积（AUC）的文本信息，使用xlabel和ylabel函数添加坐标轴标签，最后使用show函数显示图形。