In [1]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (0.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from sklearn) (1.0.2)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->sklearn)
(1.21.5)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->sklearn)
(1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->skl
earn) (3.1.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->sklearn)
(1.4.1)
```

In [19]:
```
# 导入需要的库
import os
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
```

In [3]:
```
# 导入数据
os.chdir('/content/drive/MyDrive/Colab Notebooks/data/titanic')
```

In [4]:
```
data = pd.read_csv('./data.csv', index_col=0)
data.head()
```

Out[4]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

In [6]:
```
# 数据预处理
## 删除缺失值过多的列，和观察判断来说和预测的y没关系的列
data.drop(['Cabin', 'Name', 'Ticket'], inplace=True, axis=1)
```

In [7]:
```
## 处理缺失值，对缺失值较多的列进行填补
data['Age'] = data['Age'].fillna(data['Age'].mean())
## 有一些特征只确实一两个值，可以采取直接删除记录的方法
data = data.dropna()  # Embarked
```

In [8]:
```
## 将分类型变量转换为数值型变量
### 二分类
data['Sex'] = (data['Sex']=='male').astype('int')
```

In [9]:
```
### 三分类
labels = data['Embarked'].unique().tolist()
data['Embarked'] = data['Embarked'].apply(lambda x: labels.index(x))
```

In [10]:
```
data.head()
```

Out[10]:

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	0	3	1	22.0	1	0	7.2500	0
2	1	1	0	38.0	1	0	71.2833	1
3	1	3	0	26.0	0	0	7.9250	0
4	1	1	0	35.0	1	0	53.1000	0
5	0	3	1	35.0	0	0	8.0500	0

In [11]:
```
# 提取特征矩阵和标签，拆分训练集和测试集
X = data.iloc[:, data.columns != 'Survived']
y = data.iloc[:, data.columns == 'Survived']
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.3)
```

In [12]:
```
Xtrain.head()
```

Out[12]:

PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
408	2	1	3.0	1	1	18.7500	0
872	1	0	47.0	1	1	52.5542	0
439	1	1	64.0	1	4	263.0000	0
457	1	1	65.0	0	0	26.5500	0
753	3	1	33.0	0	0	9.5000	0

In [13]:
```
# 修正训练集和测试集的索引
for i in [Xtrain, Xtest, Ytrain, Ytest]:
  i.index = range(i.shape[0])
```

In [15]:
```
# 导入模型，粗略跑一下查看结果
clf = DecisionTreeClassifier(random_state=25)
clf = clf.fit(Xtrain, Ytrain)
score_ = clf.score(Xtest, Ytest)
print(score_)
score = cross_val_score(clf, X, y, cv=10).mean()
print(score)
```

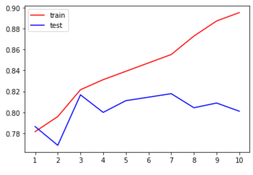
```
0.7752808988764045
0.7739274770173645
```

In [16]:
```
# 在不同 max_depth 下观察模型的拟合状态
tr = []
te = []
for i in range(10):
  clf = DecisionTreeClassifier(random_state=25
                               ,max_depth=i+1
                               ,criterion='entropy')
  clf = clf.fit(Xtrain, Ytrain)
  score_tr = clf.score(Xtrain, Ytrain)
  score_te = cross_val_score(clf, X, y, cv=10).mean()
  tr.append(score_tr)
  te.append(score_te)

print(max(te))

plt.plot(range(1,11), tr, color='red', label='train')
plt.plot(range(1,11), te, color='blue', label='test')
plt.xticks(range(1,11))
plt.legend()
plt.show()
```

```
0.8177860061287026
```



In [26]:
```
# 用网格搜索调整参数
parameters = {
    'splitter': ('best', 'random')
    ,'criterion': ('gini', 'entropy')
    ,'max_depth': [*range(1,10)]
    ,'min_samples_leaf': [*range(1,50,5)]
    ,'min_impurity_decrease': [*np.linspace(0, 0.5, 20)]
}

clf = DecisionTreeClassifier(random_state=25)
GS = GridSearchCV(clf, parameters, cv=10)
GS.fit(Xtrain, Ytrain)
```

Out[26]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=25),
             param_grid={'criterion': ('gini', 'entropy'),
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                         'min_impurity_decrease': [0.0, 0.02631578947368421,
                                                   0.05263157894736842,
                                                   0.07894736842105263,
                                                   0.10526315789473684,
                                                   0.13157894736842105,
                                                   0.15789473684210525,
                                                   0.18421052631578946,
                                                   0.21052631578947367,
                                                   0.23684210526315788,
                                                   0.2631578947368421,
                                                   0.2894736842105263,
                                                   0.3157894736842105,
                                                   0.3421052631578947,
                                                   0.3684210526315789,
                                                   0.39473684210526316,
                                                   0.42105263157894735,
                                                   0.4473684210526315,
                                                   0.4736842105263576, 0.5],
                         'min_samples_leaf': [1, 6, 11, 16, 21, 26, 31, 36, 41,
                                              46],
                         'splitter': ('best', 'random')})
```

In [27]:
```
GS.best_params_
```

Out[27]:
```
{'criterion': 'entropy',
 'max_depth': 4,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'splitter': 'best'}
```

In [28]:
```
GS.best_score_
```

Out[28]:
```
0.8247567844342039
```