泰坦尼克之灾实验报告

导包

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
import warnings
warnings.simplefilter('ignore')
```

一、数据预处理：缺失值处理+字符串转数字

In [2]:

```
train_set = pd.read_csv('./train.csv')
test_set = pd.read_csv('./test.csv')
train_set.head()
```

Out[2]:

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

In [3]:

```
train_set.info()
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

In [4]:

```
train_set.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:

```
# 数据集合并
train_test = pd.concat([train_set, test_set])
train_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   object 
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object 
 11  Embarked     1307 non-null   object 
dtypes: float64(3), int64(4), object(5)
memory usage: 132.9+ KB
```

In [6]:

```
# 查看每列是否有缺失情况
train_test.isnull().any()
```

Out[6]:

```
PassengerId    False
Survived        True
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare            True
Cabin           True
Embarked        True
dtype: bool
```

In [7]:

```
# 缺失值处理
train_test['Age'].isnull().value_counts()
```

Out[7]:

```
False    1046
True      263
Name: Age, dtype: int64
```

In [8]:

```
# 处理age的缺失,用中位数填充,并修改原数据集
train_test['Age'].fillna(train_test['Age'].median(),inplace = True)
train_test['Age']
```

Out[8]:

```
0       22.0
1       38.0
2       26.0
3       35.0
4       35.0
        ...
413     28.0
414     39.0
415     38.5
416     28.0
417     28.0
Name: Age, Length: 1309, dtype: float64
```

In [9]:

```
train_test['Embarked'].unique()
```

Out[9]:

```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [10]:

```
train_test['Embarked'].value_counts()
```

Out[10]:

```
S    914
C    270
Q    123
Name: Embarked, dtype: int64
```

In [11]:

```
# 处理embarked的缺失，用众数填充,并保存修改
train_test['Embarked'].fillna('S',inplace = True)
```

In [12]:

```
train_test['Embarked'].value_counts()
```

Out[12]:

```
S    916
C    270
Q    123
Name: Embarked, dtype: int64
```

In [13]:

```
# 字符串转数字
train_test['Sex'] = train_test['Sex'].map({'female':0, 'male':1})
train_test['Embarked'] = train_test['Embarked'].map({'S':0, 'C':1, 'Q':2})
```

In [14]:

```
train_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   int64  
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object 
 11  Embarked     1309 non-null   int64  
dtypes: float64(3), int64(6), object(3)
memory usage: 132.9+ KB
```

In [15]:

```
# 缺失值多，去除cabin
train_test.drop('Cabin',axis=1,inplace = True)
```

In [16]:

```
train_test.head()
```

Out[16]:

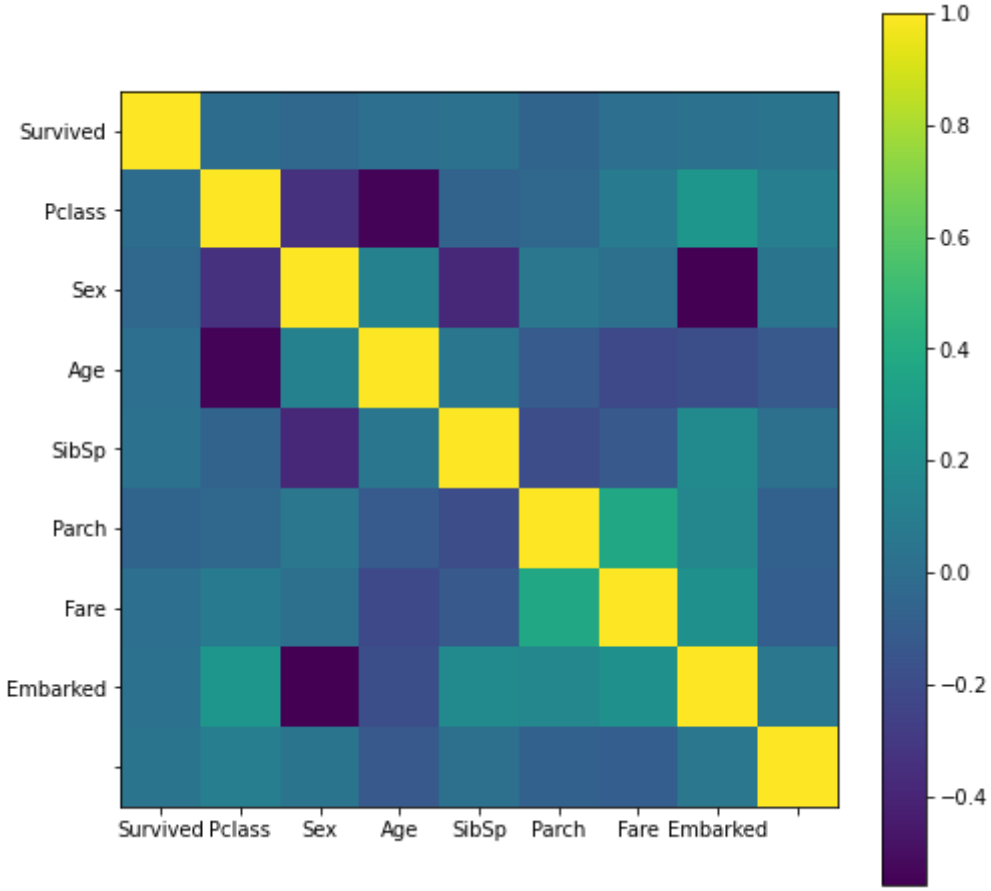
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Em
0	1	0.0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	
2	3	1.0	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	
4	5	0.0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	

In [17]:

```
# 用皮尔逊系数计算各特征相关性，并用热力图可视化
train_test_corr = train_test.corr()
plt.figure(figsize = (8,8))
ax = plt.subplot()
ax.set_xticklabels(train_test_corr.columns)
ax.set_yticklabels(train_test_corr.index)
plt.imshow(train_test_corr)
plt.colorbar()
```

Out[17]:

<matplotlib.colorbar.Colorbar at 0x1e187848a60>



In [18]:

```
train_test_corr
```

Out[18]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	
PassengerId	1.000000	-0.005007	-0.038354	0.013406	0.025799	-0.055224	0.008942	0.0
Survived	-0.005007	1.000000	-0.338481	-0.543351	-0.064910	-0.035322	0.081629	0.2
Pclass	-0.038354	-0.338481	1.000000	0.124617	-0.377908	0.060832	0.018322	-0.5
Sex	0.013406	-0.543351	0.124617	1.000000	0.053663	-0.109609	-0.213125	-0.1
Age	0.025799	-0.064910	-0.377908	0.053663	1.000000	-0.189972	-0.125851	0.1
SibSp	-0.055224	-0.035322	0.060832	-0.109609	-0.189972	1.000000	0.373587	0.1
Parch	0.008942	0.081629	0.018322	-0.213125	-0.125851	0.373587	1.000000	0.2
Fare	0.031428	0.257307	-0.558629	-0.185523	0.179256	0.160238	0.221539	1.0
Embarked	0.040143	0.106811	0.038875	-0.120423	0.018654	-0.073461	-0.095523	0.0

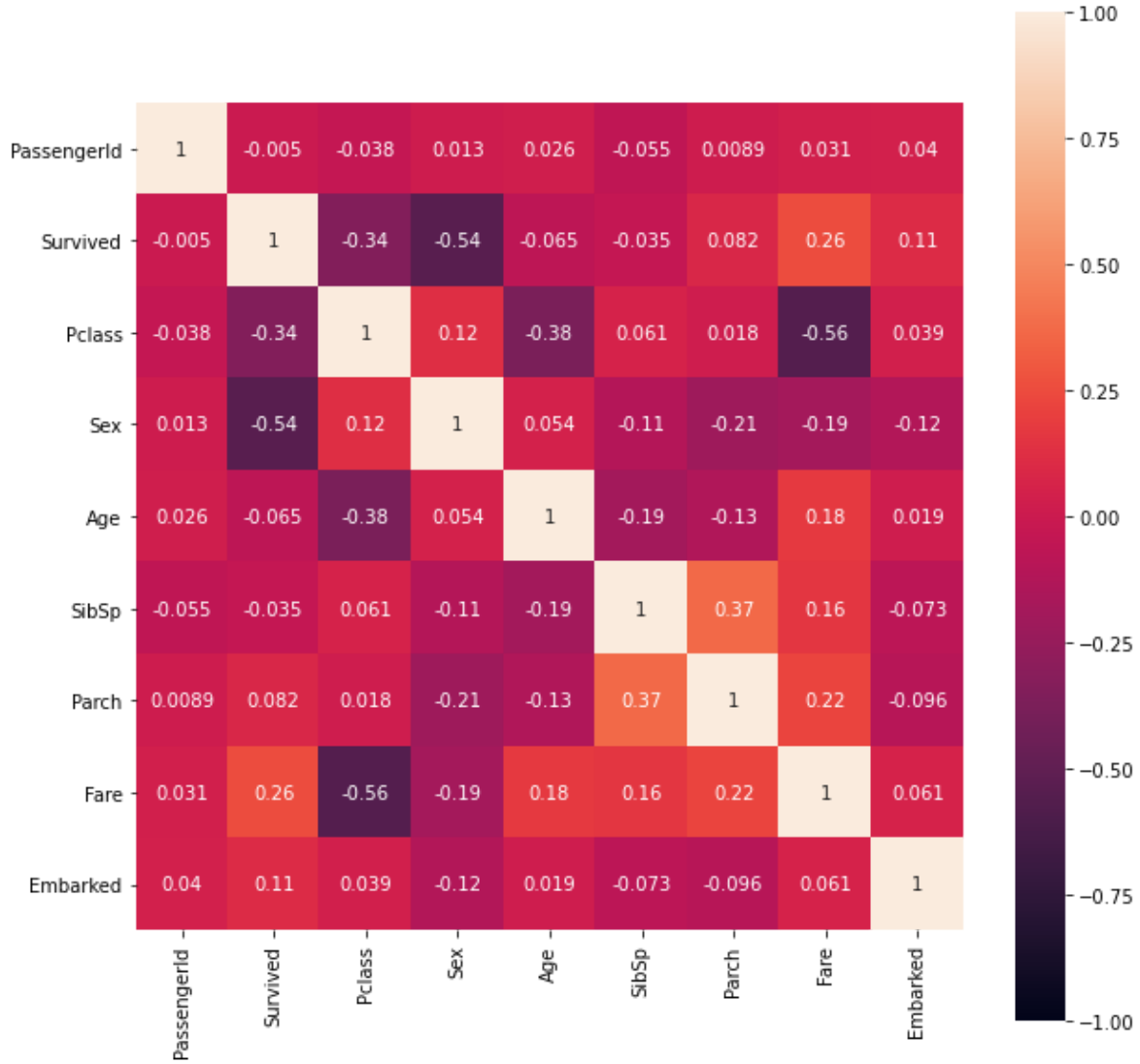
用皮尔逊系数计算各特征相关性，并用热力图可视化

In [19]:

```
plt.subplots(figsize=(10,10))
sns.heatmap(train_test_corr,vmin = -1, annot = True, square = True)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x1e187a2d790>



In [20]:

```
# 处理fare，根据热力图和经验，fare与pclass和embarked有关
train_test.groupby(['Pclass', 'Embarked'])['Fare'].mean()
```

Out[20]:

```
Pclass  Embarked
1       0           72.235825
        1          106.845330
        2           90.000000
2       0           21.206921
        1           23.300593
        2           11.735114
3       0           14.435422
        1           11.021624
        2           10.390820
Name: Fare, dtype: float64
```

In [21]:

```
# 查看缺失值
train_test[train_test['Fare'].isnull()]
```

Out[21]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
152	1044	NaN	3	Storey, Mr. Thomas	1	60.5	0	0	3701	NaN	C

In [22]:

```
# 用对应平均值填充fare缺失值
train_test['Fare'].fillna(14.435422,inplace = True)
```

In [23]:

```
train_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   int64  
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1309 non-null   float64
 10  Embarked     1309 non-null   int64  
dtypes: float64(3), int64(6), object(2)
memory usage: 122.7+ KB
```

In [24]:

```
# 数据预处理完成，保留有用特征，划分数据集
# features = ['Survived', 'Pclass', 'Sex', 'Age', 'SibSp','Parch',  'Fare', 'Embarked']
train_data = train_test[:891]
test_data = train_test[891:]
```

In [25]:

```
train_data.shape
```

Out[25]:

```
(891, 11)
```

In [26]:

```
train_data.columns
```

Out[26]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

In [27]:

```
train_data['Survived']
```

Out[27]:

```
0      0.0
1      1.0
2      1.0
3      1.0
4      0.0
      ...
886    0.0
887    1.0
888    0.0
889    1.0
890    0.0
Name: Survived, Length: 891, dtype: float64
```

In [28]:

```
train_data[['SibSp','Parch']].T
```

Out[28]:

	0	1	2	3	4	5	6	7	8	9	...	881	882	883	884	885	886	887	888	889	890
SibSp	1	1	0	1	0	0	0	3	0	1	...	0	0	0	0	0	0	0	1	0	0
Parch	0	0	0	0	0	0	0	1	2	0	...	0	0	0	0	5	0	0	2	0	0

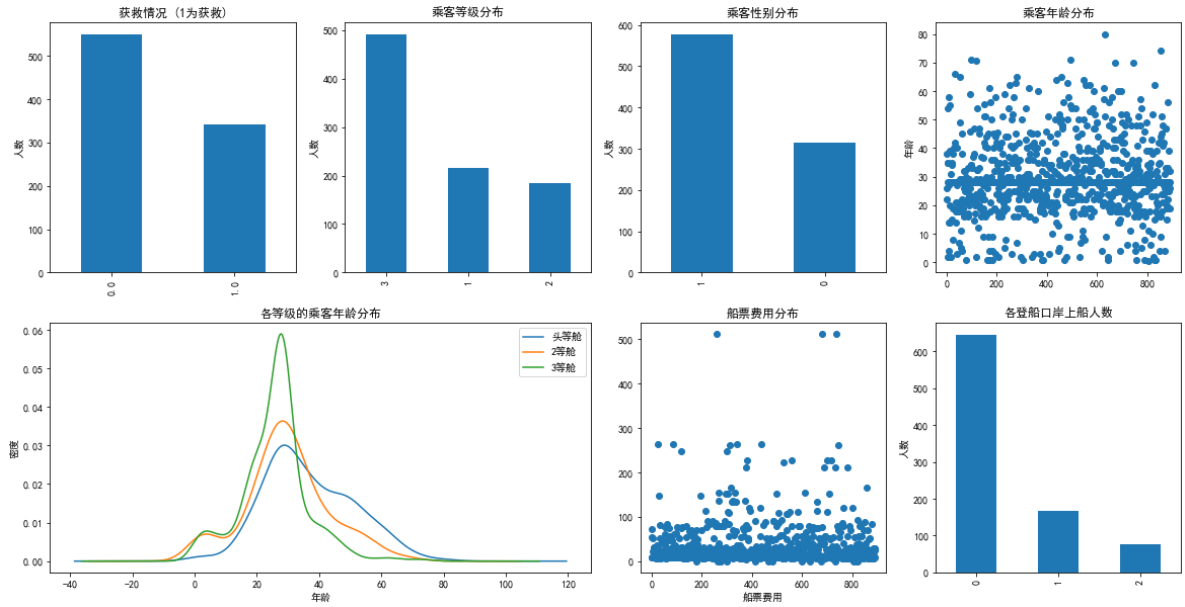
2 rows × 891 columns

features = ['Survived', 'Pclass', 'Sex', 'Age', 'SibSp','Parch', 'Fare', 'Embarked']

二、乘客属性分布

In [29]:

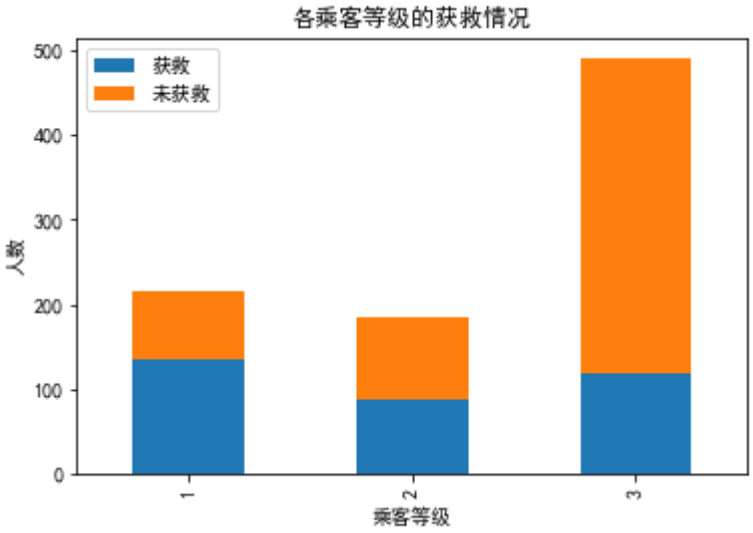
```
#乘客属性分布 绘图
# 设置属性
plt.rcParams['font.sans-serif'] = ['SimHei'] # 正常显示中文标签
plt.rcParams['font.family']='sans-serif'
plt.rcParams['axes.unicode_minus'] = False # 正常显示负号
fig = plt.figure(figsize=(20,10))
fig.set(alpha=0.2) # 设定图表颜色alpha参数
# 获救情况 survived 柱状图
plt.subplot2grid((2,4),(0,0))
train_data['Survived'].value_counts().plot(kind='bar')
plt.title("获救情况 (1为获救)")
plt.ylabel("人数")
# 等级分布 pclass 柱状图
plt.subplot2grid((2,4),(0,1))
train_data.Pclass.value_counts().plot(kind="bar")
plt.ylabel("人数")
plt.title("乘客等级分布")
# 性别分布 sex 柱状图
plt.subplot2grid((2,4),(0,2))
train_data.Sex.value_counts().plot(kind="bar")
plt.ylabel("人数")
plt.title("乘客性别分布")
# 年龄分布 age 散点图
plt.subplot2grid((2,4),(0,3))
x = np.linspace(0,891,891)
plt.scatter(x,train_data['Age'])
plt.ylabel("年龄")
plt.title("乘客年龄分布")
# # 年龄 age 散点图
# plt.subplot2grid((2,4),(0,4))
# plt.scatter(train_data['Survived'], train_data['Age'])
# plt.ylabel("年龄")
# plt.grid(b=True, which='major', axis='y')
# plt.title("按年龄看获救分布 (1为获救)")
# 各等级的乘客年龄分布 密度图
plt.subplot2grid((2,4),(1,0), colspan=2)
train_data.Age[train_data.Pclass == 1].plot(kind='kde')
train_data.Age[train_data.Pclass == 2].plot(kind='kde')
train_data.Age[train_data.Pclass == 3].plot(kind='kde')
plt.xlabel("年龄")
plt.ylabel("密度")
plt.title("各等级的乘客年龄分布")
plt.legend(('头等舱', '2等舱','3等舱'),loc='best')
# 船票分布 fare 散点图
plt.subplot2grid((2,4),(1,2))
plt.scatter(x,train_data['Fare'])
plt.xlabel("船票费用")
plt.title("船票费用分布")
# 各登船口岸上船人数 柱状图
plt.subplot2grid((2,4),(1,3))
train_data.Embarked.value_counts().plot(kind='bar')
plt.title("各登船口岸上船人数")
plt.ylabel("人数")
plt.show()
```



三、按特征分别观察与获救的关系：特征组合+创建新特征

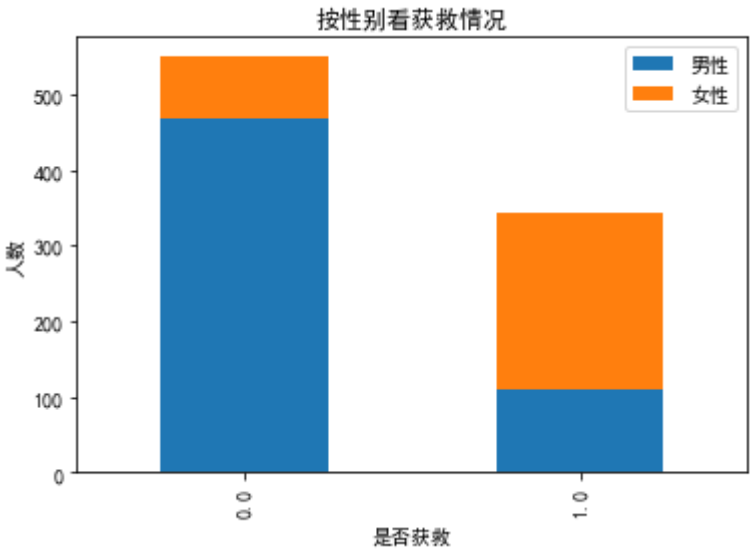
In [30]:

```
# 乘客等级与获救情况
Survived_0 = train_data.Pclass[train_data.Survived == 0].value_counts() # 0 未获救
Survived_1 = train_data.Pclass[train_data.Survived == 1].value_counts() # 1 获救
df = pd.DataFrame({'获救':Survived_1,'未获救':Survived_0})
df.plot(kind = 'bar', stacked = True)
plt.title('各乘客等级的获救情况')
plt.xlabel('乘客等级')
plt.ylabel('人数')
plt.show()
```



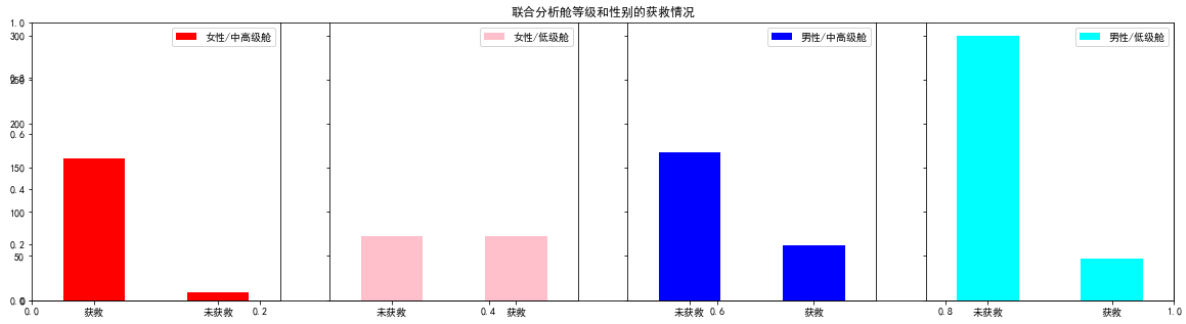
In [31]:

```
# 按性别看获救情况 'female':0, 'male':1
Survived_m = train_data.Survived[train_data.Sex == 1].value_counts() # m 男性
Survived_f = train_data.Survived[train_data.Sex == 0].value_counts()# f 女性
df = pd.DataFrame({'男性':Survived_m,'女性':Survived_f})
df.plot(kind = 'bar', stacked = True)
plt.title('按性别看获救情况')
plt.xlabel('是否获救')
plt.ylabel('人数')
plt.show()
```



In [32]:

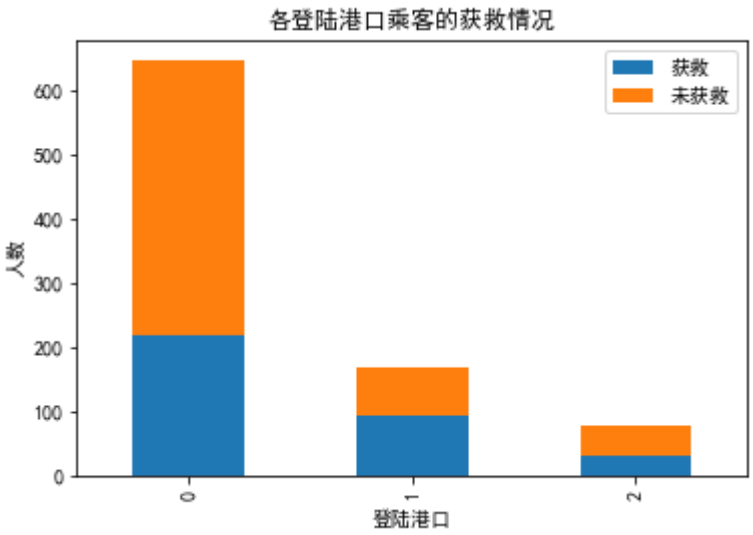
```
#联合分析，等级12与3差异大，男女差异大，根据舱等级（12为中高 3为低）和性别的获救情况
fig = plt.figure(figsize=(20,5))
plt.title('联合分析舱等级和性别的获救情况')
# 女性 0 高级舱
ax1 = fig.add_subplot(141)
train_data.Survived[train_data.Sex == 0][train_data.Pclass != 3].value_counts().plot(kind = 'bar',
ax1.set_xticklabels(['获救','未获救'], rotation = 0)
ax1.legend(['女性/中高级舱'], loc = 'best')

ax2 = fig.add_subplot(142, sharey = ax1)
train_data.Survived[train_data.Sex == 0][train_data.Pclass == 3].value_counts().plot(kind = 'bar',
ax2.set_xticklabels([u"未获救", u"获救"], rotation=0)
plt.legend([u"女性/低级舱"], loc='best')

ax3 = fig.add_subplot(143, sharey = ax1)
train_data.Survived[train_data.Sex == 1][train_data.Pclass != 3].value_counts().plot(kind = 'bar',
ax3.set_xticklabels(['未获救','获救'], rotation = 0)
plt.legend(['男性/中高级舱'], loc = 'best')

ax4 = fig.add_subplot(144, sharey = ax1)
train_data.Survived[train_data.Sex == 1][train_data.Pclass == 3].value_counts().plot(kind = 'bar',
ax4.set_xticklabels(['未获救','获救'], rotation = 0)
plt.legend(['男性/低级舱'], loc = 'best')
plt.show()
```



In [33]:

```
# 各登陆港口乘客的获救情况
fig = plt.figure()
Survived_0 = train_data.Embarked[train_data.Survived == 0].value_counts()
Survived_1 = train_data.Embarked[train_data.Survived == 1].value_counts()
df = pd.DataFrame({'获救':Survived_1,'未获救':Survived_0})
df.plot(kind = 'bar', stacked = True)
plt.title('各登陆港口乘客的获救情况')
plt.xlabel('登陆港口')
plt.ylabel('人数')
plt.show()
```

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



In [34]:

```
# 堂兄妹人数与获救 分组分析
g = train_data.groupby(['SibSp','Survived'])
df = pd.DataFrame(g.count()['PassengerId']).T
df
```

Out[34]:

SibSp	0		1		2		3		4	5	8
Survived	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0
PassengerId	398	210	97	112	15	13	12	4	15	3	5	7

In [35]:

```
# 父母人数与获救
g = train_data.groupby(['Parch','Survived'])
df = pd.DataFrame(g.count()['PassengerId']).T
df
```

Out[35]:

Parch	0		1		2		3		4		5	6
Survived	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0
PassengerId	445	233	53	65	40	40	2	3	4	4	1	1

创建新特征SibSp_Parch，联合分析亲人人数与获救关系

In [36]:

```
# 创建新特征SibSp_Parch，联合分析亲人人数与获救关系
train_test['SibSp_Parch'] = train_test['SibSp'] + train_test['Parch']
g = train_data.groupby(['SibSp_Parch','Survived'])
df = pd.DataFrame(g.count()['PassengerId']).T
df
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-36-05da23d2e223> in <module>
      1 # 创建新特征SibSp_Parch，联合分析亲人人数与获救关系
      2 train_test['SibSp_Parch'] = train_test['SibSp'] + train_test['Parch']
----> 3 g = train_data.groupby(['SibSp_Parch','Survived'])
      4 df = pd.DataFrame(g.count()['PassengerId']).T
      5 df

d:\program files\python 3.8.3\lib\site-packages\pandas\core\frame.py in groupby(self, by, axis, level, as_index, sort, group_keys, squeeze, observed)
   5799         axis = self._get_axis_number(axis)
   5800
-> 5801         return groupby_generic.DataFrameGroupBy(
   5802             obj=self,
   5803             keys=by,

d:\program files\python 3.8.3\lib\site-packages\pandas\core\groupby\groupby.py in __init__(self, obj, keys, axis, level, grouper, exclusions, selection, as_index, sort, group_keys, squeeze, observed, mutated)
    401             from pandas.core.groupby.grouper import get_grouper
    402
--> 403             grouper, exclusions, obj = get_grouper(
    404                 obj,
    405                 keys,

d:\program files\python 3.8.3\lib\site-packages\pandas\core\groupby\grouper.py in get_grouper(obj, key, axis, level, sort, observed, mutated, validate)
    598                 in_axis, name, level, gpr = False, None, gpr, None
    599             else:
--> 600                 raise KeyError(gpr)
    601         elif isinstance(gpr, Grouper) and gpr.key is not None:
    602             # Add key to exclusions

KeyError: 'SibSp_Parch'
```

创建名称（称呼可能体现地位）新特征 NameLength, Title，分类处理

In [178]:

```
train_test['NameLength'] = train_test['Name'].map(lambda name: len(name))
train_test['NameLength'].head()
```

Out[178]:

```
0    23
1    51
2    22
3    44
4    24
Name: NameLength, dtype: int64
```

In [185]:

```
# 正则匹配name,提取称呼，如 'Carlsson, Mr. Frans Olof'
train_test['Title'] = train_test['Name'].map(lambda name: re.search(r', (.*?)\. .*', name).group(1)
train_test['Title'].unique()
```

Out[185]:

```
array(['Mr', 'Mrs', 'Miss', 'Master', 'Don', 'Rev', 'Dr', 'Mme', 'Ms',
       'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'the Countess',
       'Jonkheer', 'Dona'], dtype=object)
```

In [186]:

```
# col表示上校，rev表示牧师，Mlle表示法国小姐，Major陆军少校，Sir爵士，Capt上尉，Countess女伯爵，Jonk
# 把相同的意思的title归成一类.
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Dr": 5, "Rev": 6, "Major": 7, "Col": 7,
train_test['Title'] = train_test['Title'].map(title_mapping)
train_test['Title'].unique()
```

Out[186]:

```
array([ 1,  3,  2,  4,  9,  6,  5,  8,  7, 10], dtype=int64)
```

In [187]:

```
train_test.head()
```

Out[187]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Em
0	1	0.0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	
2	3	1.0	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	
4	5	0.0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	

In [188]:

```
train_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   int64  
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1309 non-null   float64
 10  Embarked     1309 non-null   int64  
 11  SibSp_Parch  1309 non-null   int64  
 12  NameLength   1309 non-null   int64  
 13  Title        1309 non-null   int64  
dtypes: float64(3), int64(9), object(2)
memory usage: 153.4+ KB
```

In [189]:

```
train_test[train_test['Title'].isnull()]
```

Out[189]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Sib

In [ ]:

```

```

In [ ]:

```

```

四、 创建模型，分别使用新老特征预测

In [191]:

```
features_old = ['Survived','Pclass', 'Sex', 'Age', 'SibSp','Parch',  'Fare', 'Embarked']
features_new = ['Survived','Pclass', 'Sex', 'Age','Fare','Embarked', 'SibSp_Parch','NameLength', 'Ti
train_data = train_test[:891]
test_data = train_test[891:]
```

逻辑斯蒂回归模型预测

In [42]:

```
# 逻辑斯蒂回归模型
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [43]:

```
# 老特征
data_old = train_data[features_old].drop(['Survived'],axis = 1)
target_old = train_data[features_old]['Survived']
X_train_old, X_test_old, y_train_old, y_test_old = train_test_split(data_old,target_old,test_size=0.
```

In [44]:

```
logisitc = LogisticRegression(random_state=0)
logisitc.fit(X_train_old, y_train_old)
```

Out[44]:

```
LogisticRegression(random_state=0)
```

In [45]:

```
y_predict = logisitc.predict(X_test_old)
```

In [46]:

```
score = logisitc.score(X_test_old, y_test_old)
score
```

Out[46]:

```
0.8097014925373134
```

In [47]:

```
# 新特征
data_new = train_data[features_new].drop(['Survived'],axis = 1)
target_new = train_data[features_new]['Survived']
X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(data_new,target_new,test_size=0.
```

In [48]:

```
logisitc = LogisticRegression()
logisitc.fit(X_train_new, y_train_new)
logisitc.score(X_test_new, y_test_new)
```

Out[48]:

0.7910447761194029

交叉验证优化 优点:

1：交叉验证用于评估模型的预测性能，尤其是训练好的模型在新数据上的表现，可以在一定程度上减小过拟合。 2：还可以从有限的数据中获取尽可能多的有效信息。 原始采用的train_test_split方法，数据划分具有偶然性；交叉验证通过多次划分，大大降低了这种由一次随机划分带来的偶然性，同时通过多次划分，多次训练，模型也能遇到各种各样的数据，从而提高其泛化能力； 与原始的train_test_split相比，对数据的使用效率更高。 train_test_split，默认训练集、测试集比例为3:1，而对交叉验证来说，如果是5折交叉验证，训练集比测试集为4:1；10折交叉验证训练集比测试集为9:1。数据量越大，模型准确率越高!

In [49]:

```
from sklearn.model_selection import cross_val_score
```

In [50]:

```
#老特征
logisitc = LogisticRegression()
score = cross_val_score(logisitc,data_old, target_old, cv=5).mean()
score
```

Out[50]:

0.7934906785512524

In [51]:

```
# 老特征
logisitc = LogisticRegression()
score = cross_val_score(logisitc,data_old, target_old, cv=10).mean()
score
```

Out[51]:

0.7946192259675405

In [52]:

```
# 新特征
logisitc = LogisticRegression()
score = cross_val_score(logisitc,data_new, target_new, cv=5).mean()
score
```

Out[52]:

0.8092147385600402

In [53]:

```
# 新特征
data = train_data[features_new].drop(['Survived'],axis = 1)
target = train_data[features_new]['Survived']
logisitc = LogisticRegression()
score = cross_val_score(logisitc,data_new, target_new, cv=10).mean()
score
```

Out[53]:

0.8114481897627964

使用随机森林预测

In [54]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [55]:

```
# 老特征
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train_old, y_train_old)
rfc.score(X_test_old, y_test_old)
```

Out[55]:

0.8283582089552238

In [56]:

```
# 新特征
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train_new, y_train_new)
rfc.score(X_test_new, y_test_new)
```

Out[56]:

0.8208955223880597

交叉验证优化

In [57]:

```
# 老特征
rfc = RandomForestClassifier(random_state=0)
score = cross_val_score(rfc,data, target, cv=5).mean()
score
```

Out[57]:

0.8159312033142928

In [58]:

```
# 新特征
rfc = RandomForestClassifier(random_state=0)
score = cross_val_score(rfc,data, target, cv=10).mean()
score
```

Out[58]:

0.8194132334581772

初步小结：新特征优化模型score更高；随机森林分类score高于逻辑斯蒂模型；交叉验证优化计算得出score比train_test_split结果更稳定，train_test_split只进行了一次划分，数据结果具有偶然性，如果在某次划分中，训练集里全是容易学习的数据，测试集里全是复杂的数据，这样就会导致最终的结果不尽如意，所以交叉验证的评价

结果更加有参考意义。一般k为5或者10

网格搜索定义： 网格搜索法是指定参数值的一种穷举搜索方法，通过将估计函数的参数通过交叉验证进行优化来得到最优的学习算法。

步骤： · 将各个参数可能的取值进行排列组合，列出所有可能的组合结果生成“网格”。

· 然后将各组合用于SVM训练，使用交叉验证对表现进行评估。

· 在拟合函数尝试了所有的参数组合后，返回一个合适的分类器，自动调整至最佳（性能度量）参数组合，可以通过clf.best_params_获得参数值

使用交叉验证对模型评估： 如果使用K折交叉验证。将原始数据划分为k份，k-1份作为训练集，1份作为测试集。轮流进行k次。

性能度量： 可以选择accuracy, f1-score, f-beta, precise, recall等 'criterion':("gini","entropy")

随机森林调参优化， 用网格搜索调整参数

In [59]:

```
from sklearn.model_selection import GridSearchCV
```

In [ ]:

```
parameters = {'criterion':("gini","entropy")
,"max_depth":[*range(1,10)]
,"min_samples_split":[*range(1,10)]
,'min_samples_leaf':[*range(1,10)]
}
rfc = RandomForestClassifier(random_state=1)
GS = GridSearchCV(rfc, parameters, cv=10)
GS.fit(X_train_new, y_train_new)
GS.best_params_
GS.best_score_
```

In [61]:

```
parameters = {
"min_samples_split":[*range(1,10)]
,'min_samples_leaf':[*range(1,10)]
}
rfc = RandomForestClassifier(random_state=2)
GS = GridSearchCV(rfc, parameters, cv=10)
GS.fit(X_train_new, y_train_new)
GS.best_params_
GS.best_score_
```

Out[61]:

```
0.8442652329749103
```

In [62]:

```
GS.best_params_
GS.best_score_
```

Out[62]:

```
0.8442652329749103
```

In [63]:

```
GS.best_params_
```

Out[63]:

```
{'min_samples_leaf': 1, 'min_samples_split': 7}
```

In [68]:

```
pd.DataFrame(GS.cv_results_).T
```

Out[68]:

	0	1	2	
mean_fit_time	0.0347071	0.132652	0.123761	0
std_fit_time	0.0056673	0.0198074	0.0104566	0.0
mean_score_time	0	0.00866692	0.010081	0.0
std_score_time	0	0.00110095	0.00389639	0.00
param_min_samples_leaf	1	1	1	
param_min_samples_split	1	2	3	
params	{'min_samples_leaf': 1, 'min_samples_split': 1}	{'min_samples_leaf': 1, 'min_samples_split': 2}	{'min_samples_leaf': 1, 'min_samples_split': 3}	{'min_sampl 'min_sample
split0_test_score	NaN	0.857143	0.825397	
split1_test_score	NaN	0.84127	0.825397	0
split2_test_score	NaN	0.809524	0.809524	
split3_test_score	NaN	0.725806	0.741935	0
split4_test_score	NaN	0.822581	0.822581	
split5_test_score	NaN	0.83871	0.870968	0
split6_test_score	NaN	0.919355	0.903226	0
split7_test_score	NaN	0.774194	0.790323	0
split8_test_score	NaN	0.774194	0.758065	0
split9_test_score	NaN	0.870968	0.854839	0
mean_test_score	NaN	0.823374	0.820225	0
std_test_score	NaN	0.0526522	0.0466079	0.0
rank_test_score	81	71	72	

20 rows × 81 columns

In [69]:

```
parameters = {
"min_samples_split":[*range(1,10)]
,'min_samples_leaf':[*range(1,10)]
}
rfc = RandomForestClassifier(random_state=2)
GS = GridSearchCV(rfc, parameters, cv=10)
GS.fit(data_new, target_new)
GS.best_params_
GS.best_score_
```

Out[69]:

```
0.8440324594257179
```

In [70]:

```
GS.best_params_
```

Out[70]:

```
{'min_samples_leaf': 2, 'min_samples_split': 7}
```

In [71]:

```
pd.DataFrame(GS.cv_results_).T
```

Out[71]:

	0	1	2	
mean_fit_time	0.0807838	0.315551	0.145504	0
std_fit_time	0.0130726	0.0255484	0.0295117	0.0
mean_score_time	0	0.0225392	0.0086832	0.0
std_score_time	0	0.00510859	0.000898009	0.00
param_min_samples_leaf	1	1	1	
param_min_samples_split	1	2	3	
params	{'min_samples_leaf': 1, 'min_samples_split': 1}	{'min_samples_leaf': 1, 'min_samples_split': 2}	{'min_samples_leaf': 1, 'min_samples_split': 3}	{'min_sampl 'min_sample
split0_test_score	NaN	0.777778	0.766667	0
split1_test_score	NaN	0.853933	0.876404	
split2_test_score	NaN	0.764045	0.752809	0
split3_test_score	NaN	0.831461	0.842697	0
split4_test_score	NaN	0.865169	0.876404	0
split5_test_score	NaN	0.831461	0.842697	0
split6_test_score	NaN	0.797753	0.820225	0
split7_test_score	NaN	0.775281	0.797753	0
split8_test_score	NaN	0.865169	0.865169	0
split9_test_score	NaN	0.831461	0.842697	0
mean_test_score	NaN	0.819351	0.828352	
std_test_score	NaN	0.0360972	0.0413582	0.0
rank_test_score	81	72	71	

20 rows × 81 columns

In [72]:

```
parameters = {
"min_samples_split":[*range(1,10)]
,'min_samples_leaf':[*range(1,10)]
}
rfc = RandomForestClassifier(random_state=2)
GS = GridSearchCV(rfc, parameters, cv=10)
GS.fit(data_old, target_old)
GS.best_params_
GS.best_score_
```

Out[72]:

```
0.8361922596754058
```

In [73]:

```
GS.best_params_
```

Out[73]:

```
{'min_samples_leaf': 3, 'min_samples_split': 7}
```

In [139]:

```
rfc = RandomForestClassifier(n_estimators=100, min_samples_split=7, min_samples_leaf=2)
rfc.fit(data_new, target_new)
score = cross_val_score(rfc,data_new, target_new, cv=10).mean()
score
```

Out[139]:

```
0.8406991260923844
```

筛选主要特征预测

In [140]:

```
importances = rfc.feature_importances_
importances
```

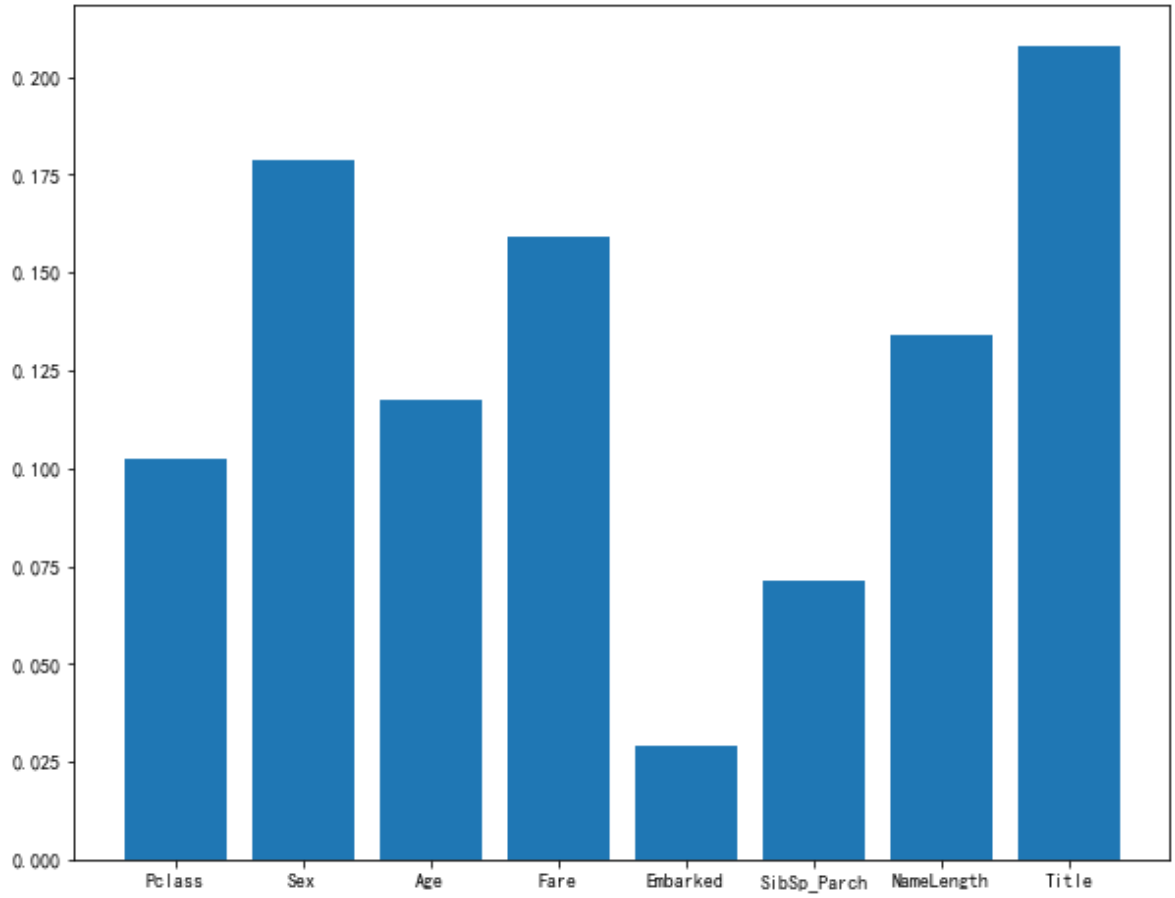
Out[140]:

```
array([0.10253587, 0.17846402, 0.1175261 , 0.1589364 , 0.02924163,
       0.07114541, 0.1341938 , 0.20795677])
```

绘制柱状图比较重要性

In [141]:

```
plt.figure(figsize=(10,8))
plt.bar(np.arange(0,8), importances)
_ = plt.xticks(np.arange(0,8), features_new[1:])
```



小结：通过对随机森林网格优化调参，采取n_estimators=100, min_samples_split=7, min_samples_leaf=2的随机森林，

选取特征为['Pclass', 'Sex', 'Age','Fare','NameLength', 'Title','SibSp_Parch','Embarked']，交叉验证后得到的模型得分最高。

然后对此模型的选取特征的重要性进行评估，得到重要性值并绘制柱状图，发现'SibSp_Parch','Embarked'特征的重要性低，所以采取去掉这两个对模型再次训练。

结果发现，虽然重要性低，仍然对模型得分产生影响，故仍保留这两个特征。该模型得分可到84左右。

In [147]:

```
features=['Pclass', 'Sex', 'Age','Fare','NameLength', 'Title']
rfc = RandomForestClassifier(n_estimators=100, min_samples_split=7, min_samples_leaf=2)
rfc.fit(data_new[features], target_new)
score = cross_val_score(rfc,data_new[features], target_new, cv=10).mean()
score
```

Out[147]:

0.8305867665418228

In [ ]:

```

```

In [148]:

```
features=['Pclass', 'Sex', 'Age','Fare','NameLength', 'Title','Embarked']
rfc = RandomForestClassifier(n_estimators=100, min_samples_split=7, min_samples_leaf=2)
rfc.fit(data_new[features], target_new)
score = cross_val_score(rfc,data_new[features], target_new, cv=10).mean()
score
```

Out[148]:

0.8395380774032459

In [149]:

```
features=['Pclass', 'Sex', 'Age','Fare','NameLength', 'Title','SibSp_Parch']
rfc = RandomForestClassifier(n_estimators=100, min_samples_split=7, min_samples_leaf=2)
rfc.fit(data_new[features], target_new)
score = cross_val_score(rfc,data_new[features], target_new, cv=10).mean()
score
```

Out[149]:

0.8339575530586767

In [151]:

```
features=['Pclass', 'Sex', 'Age','Fare','NameLength', 'Title','SibSp_Parch','Embarked']
rfc = RandomForestClassifier(n_estimators=100, min_samples_split=7, min_samples_leaf=2)
rfc.fit(data_new[features], target_new)
score = cross_val_score(rfc,data_new[features], target_new, cv=10).mean()
score
```

Out[151]:

0.8418227215980025

In [192]:

```
test_data[features].head()
```

Out[192]:

	Pclass	Sex	Age	Fare	NameLength	Title	SibSp_Parch	Embarked
0	3	1	34.5	7.8292	16	1	0	2
1	3	0	47.0	7.0000	32	3	1	0
2	2	1	62.0	9.6875	25	1	0	2
3	3	1	27.0	8.6625	16	1	0	0
4	3	0	22.0	12.2875	44	3	2	0

In [193]:

```
data_test = test_data[features]
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 418 entries, 0 to 417
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Pclass       418 non-null    int64  
 1   Sex          418 non-null    int64  
 2   Age          418 non-null    float64
 3   Fare         418 non-null    float64
 4   NameLength   418 non-null    int64  
 5   Title        418 non-null    int64  
 6   SibSp_Parch  418 non-null    int64  
 7   Embarked     418 non-null    int64  
dtypes: float64(2), int64(6)
memory usage: 29.4 KB
```

In [ ]:

```
```

用模型预测测试集，输出结果

In [195]:

```
target_test = rfc.predict(data_test)
```

In [203]:

```
result = pd.DataFrame({"Survived":target_test},index=test_data["PassengerId"])
```

In [204]:

```
result.head()
```

Out[204]:

PassengerId	Survived
892	0.0
893	0.0
894	0.0
895	0.0
896	1.0

In [205]:

```
result.to_csv('result.csv', header=True, index=True)
```