一、对数据的整体浏览

pd.read\_csv() ---返回一个DataFrame

DataFrame.info()---返回DataFrame的简要信息

DataFrame.describe()---返回数据的汇总统计

891行、12列的个人信息，包括：

PassengerId => 乘客ID、

Pclass => 乘客等级(1/2/3等舱位)、

Name => 乘客姓名、

Sex => 性别、

Age => 年龄、

SibSp => 堂兄弟/妹个数、

Parch => 父母与小孩个数、

Ticket => 船票信息、

Fare => 票价、

Cabin => 客舱、

Embarked => 登船港口

存在一定的缺失，有的是数字型，有的是文本型。

只有0.38的人存活了

二、初步数据分析

1.可视化各个属性的分布

plt.figure() 创建‘对象’

DataFrame.FEATURE.value\_counts() 统计某一个特征的个数,返回series

Series.plot(kind=” ”)

‘line’ : line plot (default)

‘bar’ : vertical bar plot

‘barh’ : horizontal bar plot

‘hist’ : histogram

‘box’ : boxplot

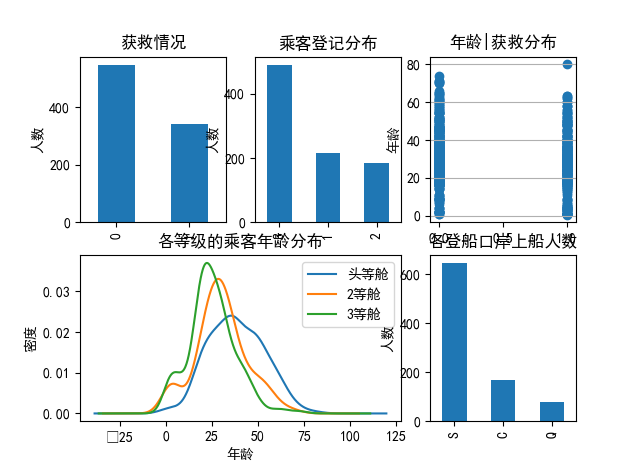
‘kde’ : Kernel Density Estimation plot

‘density’ : same as ‘kde’

‘area’ : area plot

‘pie’ : pie plot

plt.grid(b = “True”,axis = ‘y’) 沿着y轴设置



猜想：

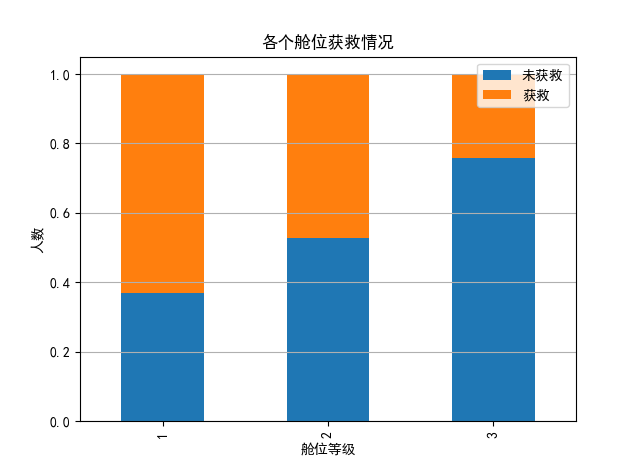
舱位等级与金钱、社会地位有关---影响最后获救的几率

性别、年龄----“女士和小孩先走”

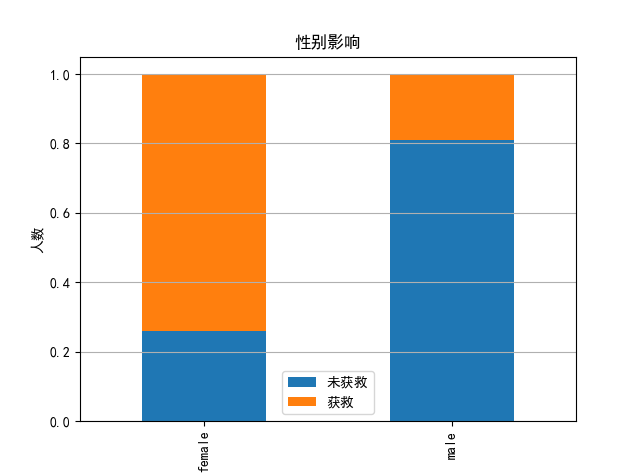
登船口---体现社会地位？

2.可视化各个属性与获救结果的关系：

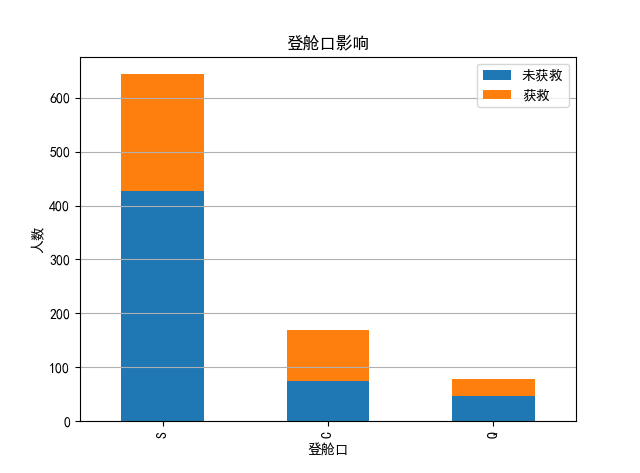
舱位等级：



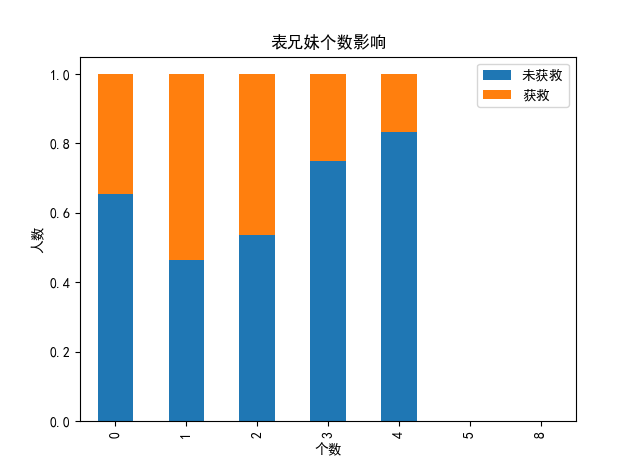
性别因素：



登舱口因素：



表兄妹个数影响：



三、特征工程

通常遇到缺值的情况，我们会有几种常见的处理方式：

1. 如果缺值的样本占总数比例极高，我们可能就直接舍弃了，作为特征加入的话，可能反倒带入noise，影响最后的结果了
2. 如果缺值的样本适中，而该属性非连续值特征属性(比如说类目属性)，那就把NaN作为一个新类别，加到类别特征中
3. 如果缺值的样本适中，而该属性为连续值特征属性，有时候我们会考虑给定一个step(比如这里的age，我们可以考虑每隔2/3岁为一个步长)，然后把它离散化，之后把NaN作为一个type加到属性类目中。
4. 有些情况下，缺失的值个数并不是特别多，那我们也可以试着根据已有的值，拟合一下数据，补充上。

RandomForestRegressor(random\_state=0, n\_estimators=2000, n\_jobs=-1)

# random\_state--->int--->随机数生成seed

随机数是由随机种子根据一定的计算方法计算出来的数值。所以，只要计算方法一定，随机种子一定，那么产生的随机数就不会变。

# random\_state--->instance--->随机数生成器  
# n\_estimators--->想要建立子树的数量。  
# n\_jpbs--->这个参数告诉引擎有多少处理器是它可以使用。 “-1”意味着没有限制，而“1”值意味着它只能使用一个处理器。

regressor.fit(X,y)

The training input samples. Internally, its dtype will be converted to dtype=np.float32. If a sparse matrix is provided, it will be converted into a sparse csc\_matrix.

regressor.preditct(X’)

X同上

df.loc[ condition ,’feature’] 🡪在feature列找到满足condition的值

Age和Fare两个属性，乘客的数值幅度变化，也忒大了吧！！ 甚至不收敛！…所以先用scikit-learn里面的preprocessing模块对其做一个scaling，所谓scaling，其实就是将一些变化幅度较大的特征化到[-1,1]之内。

StandardScaler(*copy=True*, *with\_mean=True*, *with\_std=True*)

# scaler = preprocessing.StandardScaler()

fit(X, y=None)

# age\_scale\_param = scaler.fit(df['Age'])

计算均值和方差，为后面的归一

fit\_transform(X, y=None, \*\*fit\_params)

先fit，再trans

# df['Fare\_scaled'] = scaler.fit\_transform(df['Fare'], fare\_scale\_param)

决策树：

分类决策树的核心思想就是在一个数据集中找到一个最优特征，然后从这个特征的选值中找一个最优候选值，根据这个最优候选值将数据集分为两个子数据集，然后递归上述操作，直到满足指定条件为止

1.最优特征怎么找？：常用的方法是更具信息增益或者信息增益率来寻找最优特征

2.为什么还要找最优特征的最优候选值？：如果我们一下子从父节点直接分了多个子节点，那么我们的数据肯定会对这个控制很敏感，敏感就会导致出错。建议对多分类问题找最优候选值来转化为二分类问题

ID3算法就是对各个feature信息计算信息增益，然后选择信息增益最大的feature作为决策点将数据分成两部分。然后再对这两部分分别生成决策树。

C4.5与ID3相比其实就是用信息增益比代替信息增益，应为信息增益有一个缺点：信息增益选择属性时偏向选择取值多的属性

CART(classification and regression tree)的算法整体过程和上面的差异不大，然是CART的决策是二叉树的，每一个决策只能是“是”和“否”，换句话说，即使一个feature有多个可能取值，也只选择其中一个而把数据分成两部分而不是多个，它用到的是基尼指数。

随机森林

同一批数据，用同样的算法只能产生一棵树，这时Bagging策略可以帮助我们产生不同的数据集。Bagging策略来源于bootstrap aggregation：从样本集（假设样本集N个数据点）中重采样选出Nb个样本（有放回的采样，样本数据点个数仍然不变为N），在所有样本上，对这n个样本建立分类器（ID3\C4.5\CART\SVM\L-OGISTIC），重复以上两步m次，获得m个分类器，最后根据这m个分类器的投票结果，决定数据属于哪一类。

随机森林在bagging的基础上更进一步：

1.  样本的随机：从样本集中用Bootstrap随机选取n个样本

2.  特征的随机：从所有属性中随机选取K个属性，选择最佳分割属性作为节点建立CART决策树（泛化的理解，这里面也可以是其他类型的分类器，比如SVM、Logistics）

3.  重复以上两步m次，即建立了m棵CART决策树

4.  这m个CART形成随机森林，通过投票表决结果，决定数据属于哪一类（投票机制有一票否决制、少数服从多数、加权多数）

遇到的问题：

1.matplotlib图表无法显示中文：

添加语句：引入中文字体

from pylab import \*  
mpl.rcParams['font.sans-serif'] = ['SimHei']

2.