# 解题思路

## 框架：

**1.    数据清洗（Data Cleaning）**

**2.    探索性可视化（Exploratory Visualization）**

**3.    特征工程（Feature Engineering）**

**4.    基本建模&评估（Basic Modeling& Evaluation）**

**5.    参数调整（Hyperparameters Tuning）**

**6.   集成方法（EnsembleMethods）**

## 准备工作

引用pandas和numpy包，并读取训练样本文件。

用pandas把csv文件读入成dataframe格式。

import pandas as pd

import numpy as np

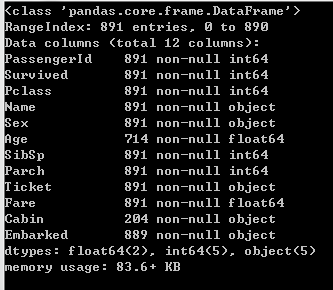
from pandas import Series, DataFrame

data\_train = pd.read\_csv("D:/kaggle/Titanic/data/train.csv")

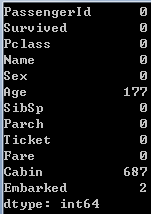
data\_train

## 1. 数据清洗

1. 初步统计特征值缺失情况。

 data\_train.info()

训练样本一共有891名乘客信息，样本量n = 891。



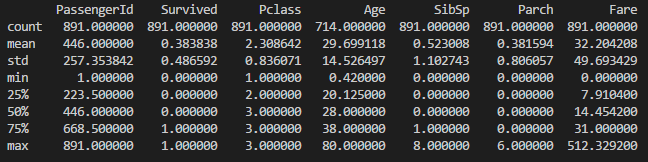
统计缺失值情况：

print(data\_train.isnull().sum())

|  |  |  |  |
| --- | --- | --- | --- |
| 特征名 | 含义 | 非缺失值数量 | 缺失值数量 |
| PassengerId | 乘客ID | 891 | 0 |
| Pclass | 客舱等级(1/2/3等舱位) | 891 | 0 |
| Name | 乘客姓名 | 891 | 0 |
| Sex | 性别 | 891 | 0 |
| Age | 年龄 | 714 | 177 |
| SibSp | 兄弟姐妹数/配偶数 | 891 | 0 |
| Parch | 父母数/子女数 | 891 | 0 |
| Ticket | 船票编号 | 891 | 0 |
| Fare | 船票价格 | 891 | 0 |
| Cabin | 客舱号 | 204 | 687 |
| Embarked | 登船港口 | 889 | 2 |

2. 获取更多信息

print(data\_train.describe())



训练集中大概有0.3838的人获救

3. 针对缺失值较多的Age和Cabin处理

import matplotlib.pyplot as plt

'''#此处引用确保中文不会变成乱码'''

plt.rcParams[ 'font.sans-serif'] = [ 'Microsoft YaHei']

plt.rcParams[ 'axes.unicode\_minus'] = False

'''

缺失值画图分析

'''

fig = plt.figure()

fig.set(alpha=0.2) # alpha参数表示图表颜色透明度，越小越透明

#Survived - Age散点分布图

plt.scatter(data\_train.Survived, data\_train.Age) #scatter散点图，x、y轴分别为Survived和Age值

plt.ylabel(u"年龄") #设定纵坐标名称

''' grid()方法设置网格线。b布尔值表示是否显示网格线。

which取值为'major','minor'，'both'。默认为'major'。

axis: 取值为'both'，'x'，'y'，输入的是哪条轴，则会隐藏哪条轴'''

plt.grid(b=True, which='major', axis='y')

plt.title(u"按年龄看获救分布 (1为获救)")

plt.show()

#各年龄段人数及获救情况 频数直方图

'''

hist()括号里第一个参数就是频率分布直方图横坐标数据，横坐标数据不需要value\_counts()

bin代表直方图条形个数，edgecolor代表条形边界颜色，alpha代表透明度

Survived为0或1同时hist()就会放在一张图中

'''

data\_train.dropna(subset=['Age'], inplace= True) #原数据中Age有缺失值，此条语句去掉缺失值样本。若在缺失值填补后，画直方图则不需要添加此语句

plt.hist(data\_train.Age[data\_train.Survived == 0],bins=30,edgecolor = 'black')

plt.hist(data\_train.Age[data\_train.Survived == 1],bins=30,edgecolor = 'black',alpha = 0.8)

plt.xlabel(u'年龄')

plt.ylabel(u'人数')

plt.title(u'各年龄段人数及获救情况')

plt.legend((u'0', u'1'),loc='best')

plt.show()

#Cabin堆积柱状图

Survived\_cabin = data\_train.Survived[pd.notnull(data\_train.Cabin)].value\_counts()

Survived\_nocabin = data\_train.Survived[pd.isnull(data\_train.Cabin)].value\_counts()

df=pd.DataFrame({u'Exist':Survived\_cabin, u'Null':Survived\_nocabin}).transpose() #transpose更换横纵堆积方向

df.plot(kind='bar', stacked=True) #设置stacked=True即可为DataFrame生成堆积柱状图，这样每行的值就会被堆积在一起

plt.title(u"按Cabin有无看获救情况")

plt.xlabel(u"Cabin有无")

plt.ylabel(u"人数")

plt.show()

#各个Embarked的存活情况

Survived\_0 = data\_train.Embarked[data\_train.Survived == 0].value\_counts()

Survived\_1 = data\_train.Embarked[data\_train.Survived == 1].value\_counts()

df=pd.DataFrame({u'1':Survived\_1, u'0':Survived\_0}) #关心的是P(c|x)，即港口条件下的存活率

df.plot(kind='bar', stacked=True)

plt.title(u"各登录港口乘客的获救情况")

plt.xlabel(u"登录港口")

plt.ylabel(u"人数")

plt.show()

|  |  |
| --- | --- |
| D:\kaggle\Titanic\Fig\Figure_2.png | D:\kaggle\Titanic\Fig\Figure_2-2.png |

图中看出，是否存活的集合中年龄分布是有很大不同的（获救人员年龄分布更平滑，趋于两头），因此对于Age属性中的缺失值选择非删除的填补方法出来。这里用scikit-learn中的RandomForest来拟合一下缺失的年龄数据。（RF是一个用在原始数据中做不同采样，建立多棵DecisionTree，再进行average等等来降低过拟合现象。随机森林算法有很强的抗干扰能力，它包含估计缺失值的算法，如果有一部分的信息遗失，仍可以维持一定的准确度。）

年龄和经济能力（Fare、Pclass）、亲属数量（父母、配偶、子女：Parch、SibSp）等都有相关性。

注：也有人根据乘客Name提取Title判断年龄段，利用Title的中位数对Age插补。

|  |  |
| --- | --- |
| D:\kaggle\Titanic\Fig\Figure_3.png | D:\kaggle\Titanic\Fig\Figure_4.png |

可以看出，Cabin值存在时，存活比例明显高于非存活；Cabin值缺失时，存活比例又明显小于非存活。因此可以采用缺失值填补处理方法中的特殊值填充（Treating Missing Attribute values as Special values），即将Cabin的有无数据作为一个特征。

Embarked缺失值只有2个，数量不多，可以选择直接用出现频率最高的"S"取值来填补。

'''

缺失值填补

'''

from sklearn.ensemble import RandomForestRegressor

### 使用 RandomForestClassifier 填补缺失的年龄属性

def set\_missing\_ages(df):

# 把已有的数值型特征取出来丢进Random Forest Regressor中

age\_df = df[['Age','Fare', 'Parch', 'SibSp', 'Pclass']]

# 乘客分成已知年龄和未知年龄两部分

known\_age = age\_df[age\_df.Age.notnull()].as\_matrix()

unknown\_age = age\_df[age\_df.Age.isnull()].as\_matrix()

# y即目标年龄，所有行的第1列

y = known\_age[:, 0]

# X即特征属性值，所有行的第2列到最后一列

X = known\_age[:, 1:]

# fit到RandomForestRegressor之中

'''

n\_estimators：基学习器决策树的个数，越多越好，但是性能就会越差，至少100左右可以达到可接受的性能和误差率。

n\_jobs：并行job个数。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。

'''

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

rfr.fit(X, y)

# 用得到的模型进行未知年龄结果预测

predictedAges = rfr.predict(unknown\_age[:, 1:])

# 用得到的预测结果填补原缺失数据

df.loc[ (df.Age.isnull()), 'Age' ] = predictedAges

return df, rfr

def set\_Cabin\_type(df):

df.loc[ (df.Cabin.notnull()), 'Cabin' ] = 1

df.loc[ (df.Cabin.isnull()), 'Cabin' ] = 0

return df

def set\_missing\_Embarked(df):

df.loc[ (df.Embarked.isnull()), 'Embarked' ] = 's'

return df

data\_train, rfr = set\_missing\_ages(data\_train)

data\_train = set\_Cabin\_type(data\_train)

data\_train = set\_missing\_Embarked(data\_train)

此时data\_train中的信息：

data\_train.info()



**至此缺失值补全。**

## 2. 探索性可视化

画图分析各个属性和最后的Survived结果之间的关系

'''

探索性可视化

'''

import matplotlib.pyplot as plt

'''#此处引用确保中文不会变成乱码'''

plt.rcParams[ 'font.sans-serif'] = [ 'Microsoft YaHei']

plt.rcParams[ 'axes.unicode\_minus'] = False

fig = plt.figure()

fig.set(alpha=0.2) # alpha参数表示图表颜色透明度，越小越透明

'''subplot2grid()方法：在一张大图里分列几个小图：(2,3)表示大图总尺寸为两行三列，(0,0)表示当前图的起始下标'''

#（1）Survived人数

plt.subplot2grid((2,2),(0,0))

data\_train.Survived.value\_counts().plot(kind='bar') # kind = bar 柱状图

plt.title(u"获救情况 (1为获救)") # 标题

plt.ylabel(u"人数")

#（2）Pclass人数

plt.subplot2grid((2,2),(0,1))

data\_train.Pclass.value\_counts().plot(kind="bar")

plt.title(u"乘客等级分布")

plt.ylabel(u"人数")

#（3）不同Pclass的Age密度分布图

plt.subplot2grid((2,2),(1,0), colspan=2) #在大图中从(1,0)位置开始，占两个列宽

data\_train.Age[data\_train.Pclass == 1].plot(kind='kde') #kde表示密度图

data\_train.Age[data\_train.Pclass == 2].plot(kind='kde')

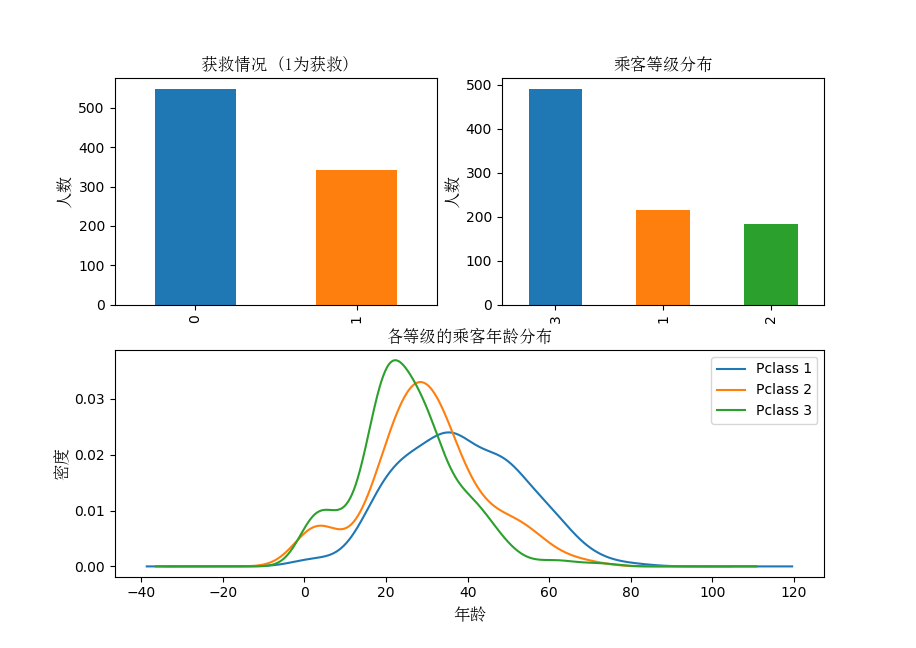
data\_train.Age[data\_train.Pclass == 3].plot(kind='kde')

plt.xlabel(u"年龄")# plots an axis lable

plt.ylabel(u"密度")

plt.title(u"各等级的乘客年龄分布")

plt.legend((u'Pclass 1', u'Pclass 2',u'Pclass 3'),loc='best') # 图例写法！！

plt.show()

Pclass

根据Pclass各个等级频数，将1、2划分为高等舱，3划分为低等舱。

#不同Pclass获救情况

Survived\_0 = data\_train.Pclass[data\_train.Survived == 0].value\_counts()

Survived\_1 = data\_train.Pclass[data\_train.Survived == 1].value\_counts()

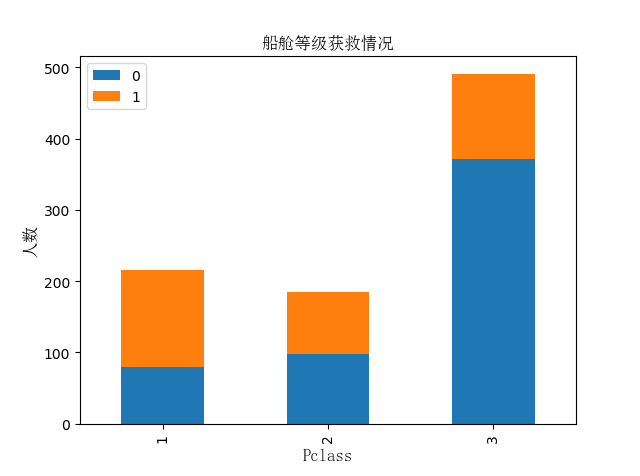
df=pd.DataFrame({u'0':Survived\_0, u'1':Survived\_1}) #此处只能两段数据堆积柱状图，若要三段及以上需transpose()方法

df.plot(kind='bar', stacked=True)

plt.title(u"船舱等级获救情况",fontproperties=font\_set)

plt.xlabel(u"Pclass",fontproperties=font\_set)

plt.ylabel(u"人数",fontproperties=font\_set)

plt.show()

#不同Sex获救情况

Survived\_m = data\_train.Survived[data\_train.Sex == 'male'].value\_counts()

Survived\_f = data\_train.Survived[data\_train.Sex == 'female'].value\_counts()

df=pd.DataFrame({u'male':Survived\_m, u'female':Survived\_f}).transpose()

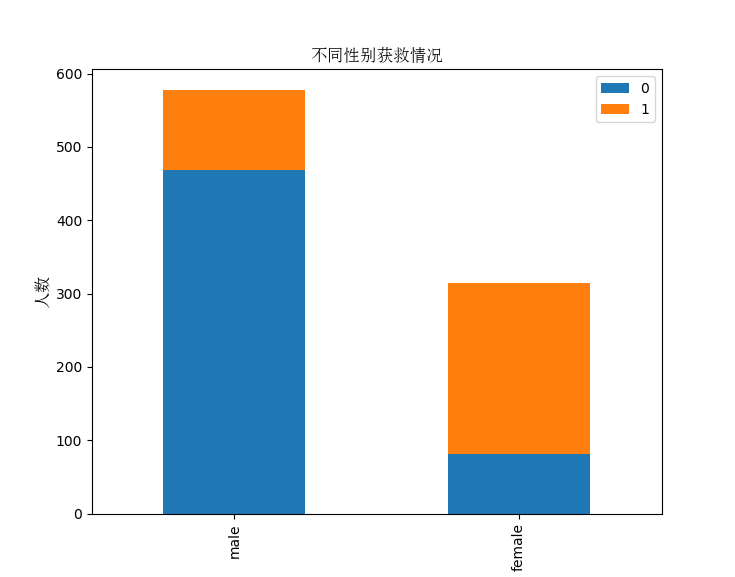
df.plot(kind='bar', stacked=True)

plt.title(u"不同性别获救情况",fontproperties=font\_set)

plt.xlabel(u"性别",fontproperties=font\_set)

plt.ylabel(u"人数",fontproperties=font\_set)

plt.show()



将Sex和Pclass合起来看：

'''画子图：pyplot的方式中plt.subplot()参数和面向对象中的add\_subplot()参数和含义都相同'''

fig=plt.figure()

fig.set(alpha=0.65)

plt.title(u"根据舱等级和性别的获救情况",fontproperties=font\_set)

ax1=fig.add\_subplot(141) #141代表：总共1行、4列，当前子图处于第1个位置，之后的子图位置标记按每一行逐列再逐行的顺序往下排布

data\_train.Survived[data\_train.Sex == 'female'][data\_train.Pclass != 3].value\_counts().plot(kind='bar', label="female, high class", color='#FA2479')

ax1.set\_xticklabels([u"1", u"0"], rotation=0)       #1和0的顺序：Pclass是1,2的female从上到下先遇见Survived是1

ax1.legend([u"female/Pclass = 1,2"], loc='best')

ax2=fig.add\_subplot(142, sharey=ax1)

data\_train.Survived[data\_train.Sex == 'female'][data\_train.Pclass == 3].value\_counts().plot(kind='bar', label='female, low class', color='pink')

ax2.set\_xticklabels([u"1", u"0"], rotation=0)

plt.legend([u"female/Pclass = 3"], loc='best')

ax3=fig.add\_subplot(143, sharey=ax1)

data\_train.Survived[data\_train.Sex == 'male'][data\_train.Pclass != 3].value\_counts().plot(kind='bar', label='male, high class',color='lightblue')

ax3.set\_xticklabels([u"0", u"1"], rotation=0)

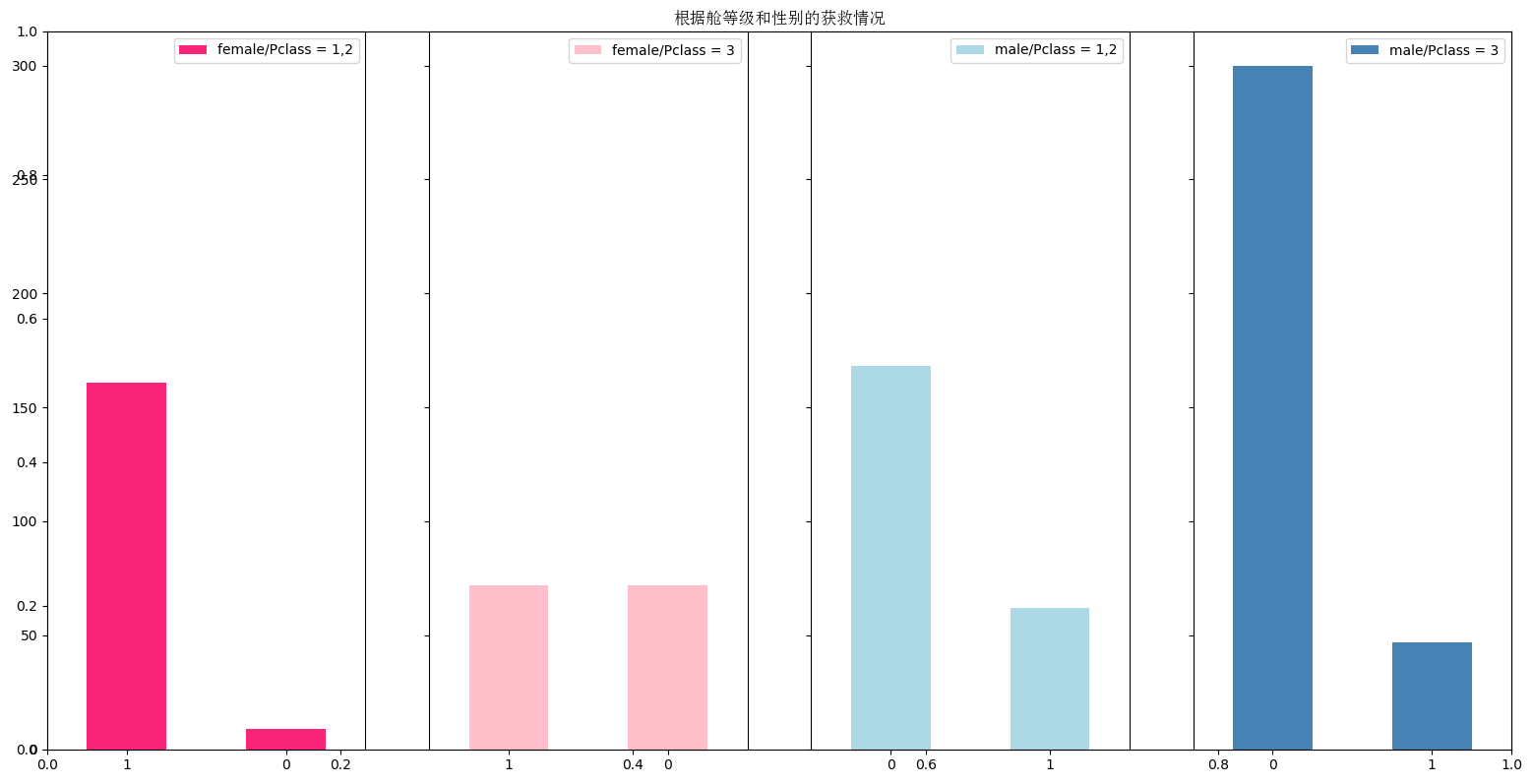
plt.legend([u"male/Pclass = 1,2"], loc='best')

ax4=fig.add\_subplot(144, sharey=ax1)

data\_train.Survived[data\_train.Sex == 'male'][data\_train.Pclass == 3].value\_counts().plot(kind='bar', label='male, low class', color='steelblue')

ax4.set\_xticklabels([u"0", u"1"], rotation=0)

plt.legend([u"male/Pclass = 3"], loc='best')

plt.show()

## 初步的属性之间相关性分析：

'''

数据清洗前的相关性分析

'''

colormap = plt.cm.viridis

plt.figure(figsize=(14,12))

sns.heatmap(data\_train[data\_train.columns[1:]].corr(),linewidths=0.1,vmax=1.0, square=True, cmap=colormap, linecolor='white', annot=True)

plt.title('Pearson Correlation of Features', y=1.05, size=15)

plt.show()

## D:\kaggle\Titanic\Exploration\Fig\corr.png