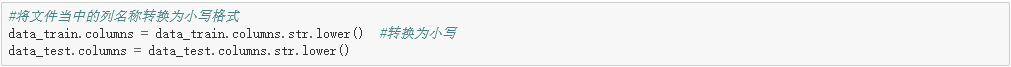
**作业2：分类问题 – 泰坦尼克号 幸存者预测问题**

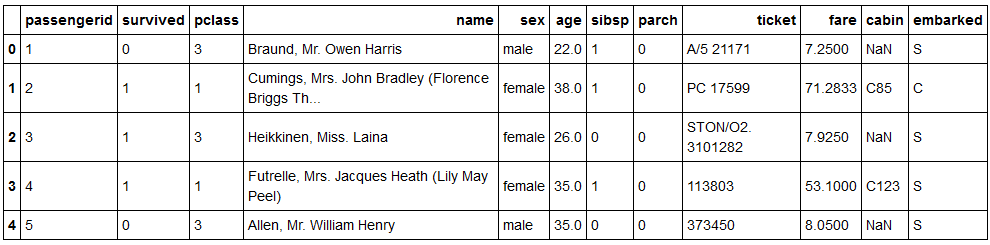
智造1班第5组：陈泽明、黄熙广、邓嘉伟、叶思聪、吴帝可、谢沅佩

1. **数据清洗与分析**

调用Pandas库读取两个csv文件，分别是训练集和测试集，查看它们前5行的数据，发现每个特征的列名称是大写的，为了使得后面的数据处理更加方便，在此将列名称都转换为小写格式：



之后查看训练集中前5行的数据：

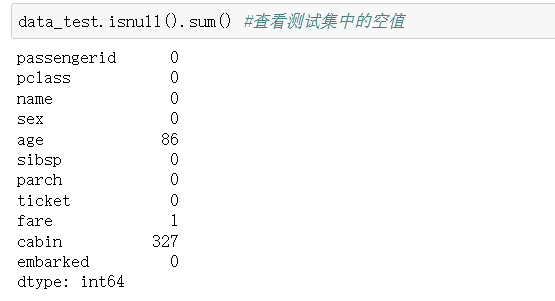
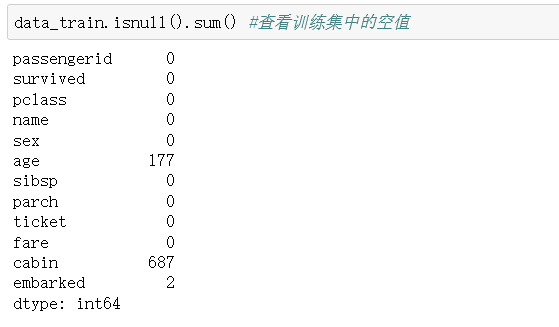


由表：列名称已经全部转换为小写格式，其中有部分数据显示NaN空值，数据集中出现了空值，对数据的使用会造成影响，所以需要对空值进行补全。

在补全空值之前，需要将训练集和测试集合并在一起，这样能够提高数据清洗的效率：

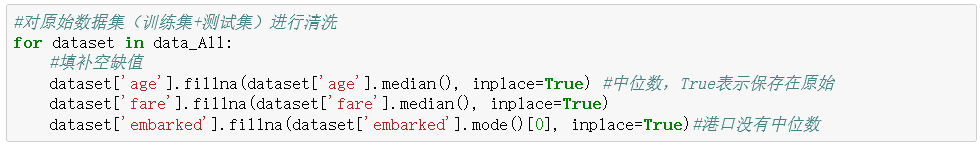


之后查看具体是哪些特征出现了空值，以及出现空值的数量是多少：



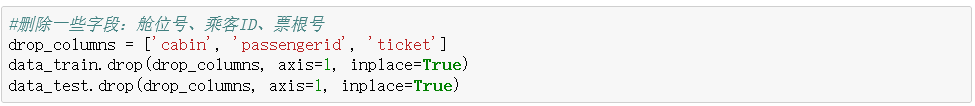
显示的结果表明了在乘客年龄age的那一列数据中，有177个空值，乘客的年龄对于是否获救有着重要的意义，所以这个特征必须补全；同样地，对于船票fare，票值不同，在船上所处的位置就不同，生死的可能也就不同，所以在测试集中有1个fare是空值，那么就需要进行补全；港口号embarked也是如此，不同的港口有不同的经济实力，经济实力可以反应乘客的生活水平，生活水平越好，在船上就有更好的待遇，那么就有更大的可能被获救，所以对于训练集中的2个空缺的embarked需要进行补全；而相对于舱位号cabin，这个特征与其他特征相比就显得不那么重要了，因为无论是训练集还是测试集，舱位号的空值数量太多，以至与无法对其进行利用，所以cabin特征选择忽视掉。

接下来对训练集和测试集的空值进行补全：

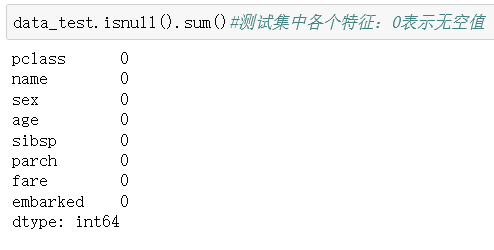
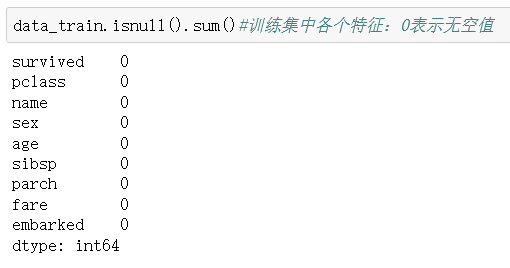


其中乘客年龄age和船票fare采用中位数补全的方法，因为中位数相对于众数或平均数有着更好的说明性，而港口号embarked就使用众数，港口号不是数字，而是字母，所以使用mode()[0]选取数量当中最多的那个字母。

最后删除掉一些没有利用价值的字段及其数据：



在这里删除了舱位号cabin、乘客ID以及票根号ticket，在前面分析过舱位号缺失的数据太多，无利用价值，所以删除掉，乘客ID和票根号对是否获救也无多大影响，所以也选择删除。



经过数据清洗之后，就留下了具有利用价值且不为空的数据。

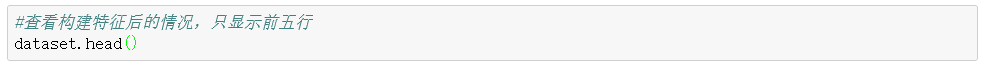
1. **构建新字段以及编码并可视化特征**

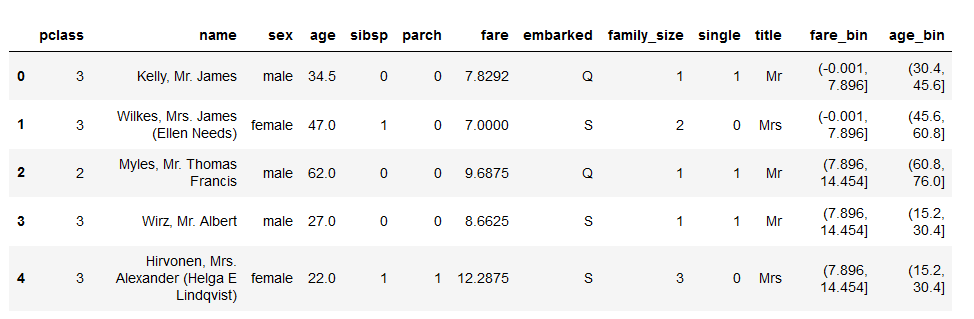
在清洗和分析数据之后，就是构建一些更有用的字段：



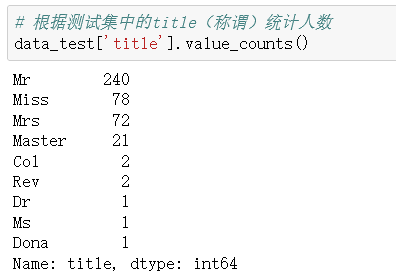
如图中代码显示，我们构建了5个新的字段：家庭规模family\_size、单身情况single、名字称谓title、票价分段fare\_bin以及年龄分段age\_bin。遍历data\_All即训练集和测试集中的数据，家庭规模有sibsp和parch以及本人构成，家庭成员数量对是否获救是很有意义的，所以统计是否单身single也是具有意义的；同样地，对于名字中的称谓title，如Mr、Miss、Mrs等，对于是否获救很巨大意义，因为这些称谓代表着一个人的身份，身份越高贵，越有可能被获救；最后构建的两个新字段分别是，把票价从小到大排列后平均分成4组fare\_bin，把乘客年龄从小到大排列后平均分成5组age\_bin，之所以对其进行分组，是因为这样能够使得船票和年龄更能够与生存survived联系起来，比如10岁以下以及70年以上有更大的获救可能、船票最贵的那个价位更能够容易获救等。

将构建完之后的特征进行显示如下：

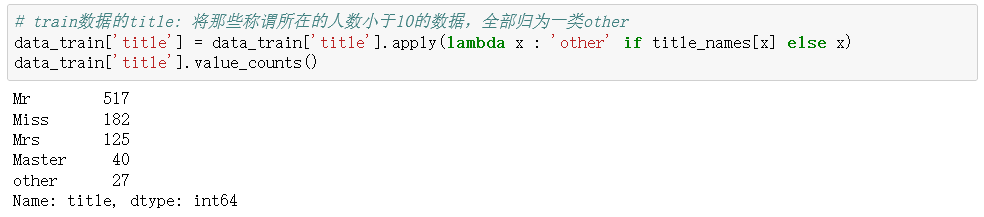


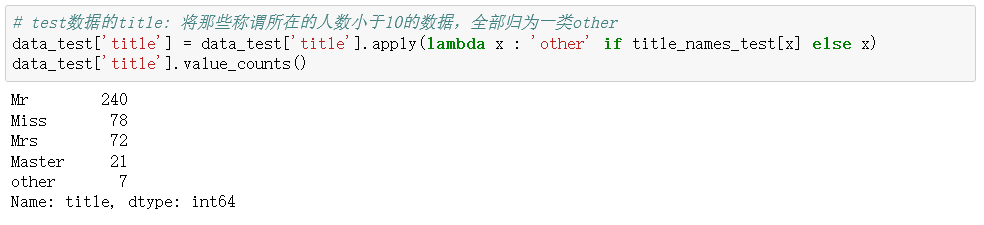


紧接着处理名字称谓title，需要把同一个称谓但是人数少的人归为一类，如下：



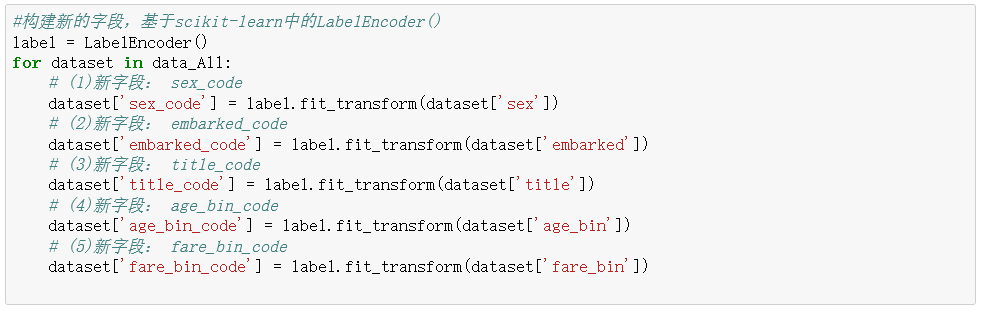
由训练集和测试集中的title可以看出，大部分人处于Mr、Miss、Mrs和Master，而处于其他称谓的人很少，所以需要把其他的称谓归为一类other。

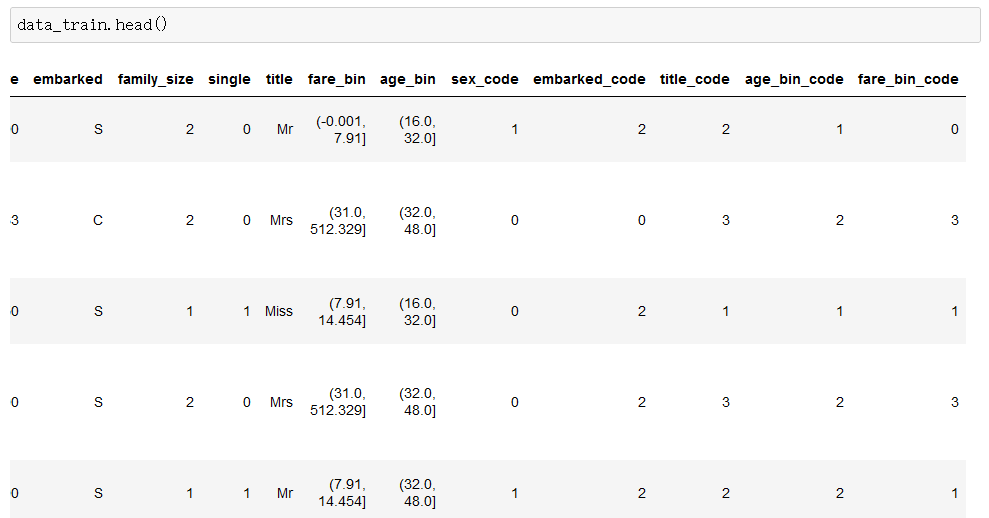




这样做的目的是：使得title在模型训练中就能够更好地与survived联系起来，从而提高准确率。

接下来进行编码，编码的目的是使得有些英文数据能够转变为数字数据，如下，利用sklearn库中的LabelEncoder函数，将某些字段进行编码从而新建新的字段：





如以上数据显示，编码之后的所构建的字段，可以用于模型训练中的特征选择的选取，多一些字段进行训练，可能会提高准确率。然后将数据完整处理后的训练集保存到csv文件中：

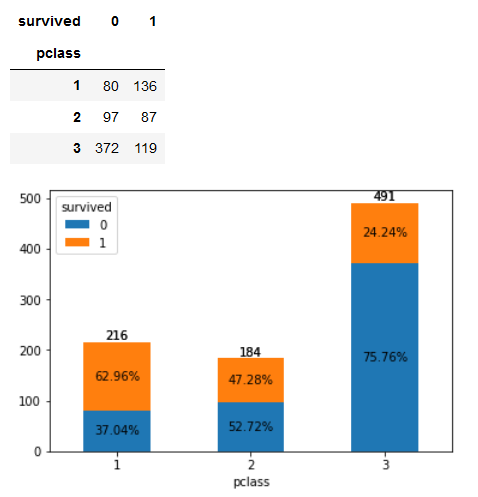
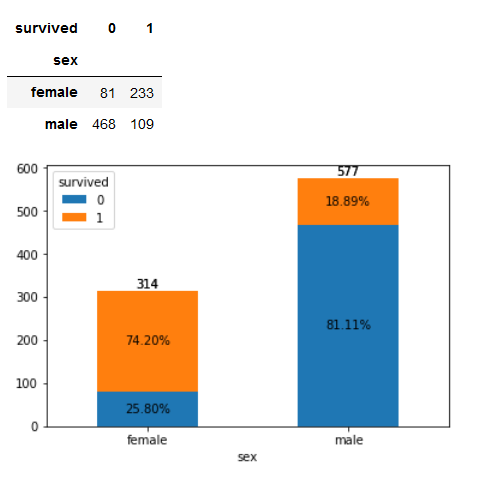


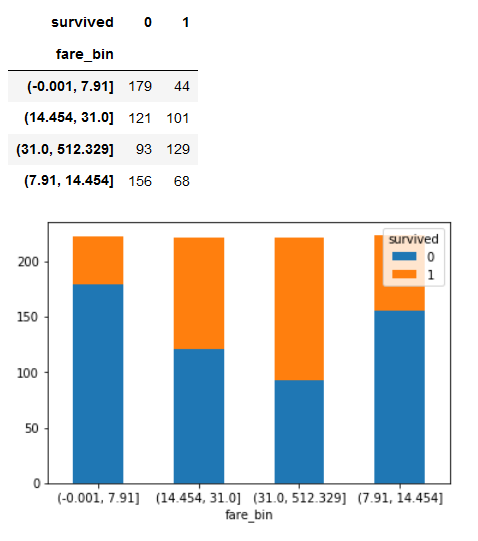
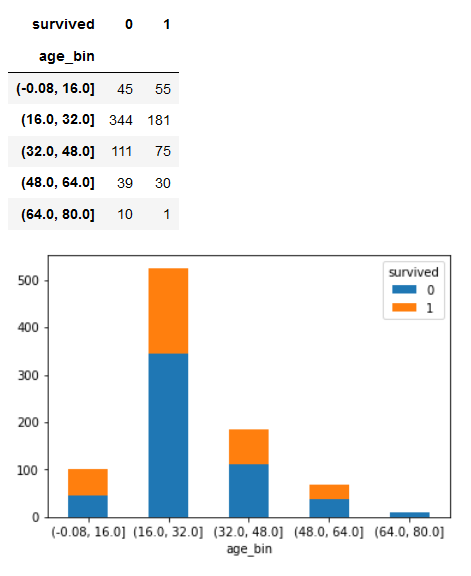
在此，训练集和测试集数据的构建和处理就完成了，最后就是对一些对survived会产生大影响的特征进行可视化（这个可视化过程放在data\_visualization\_of\_Titanic程序中）：

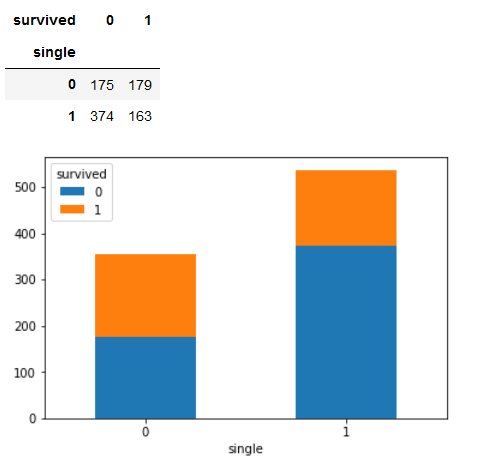
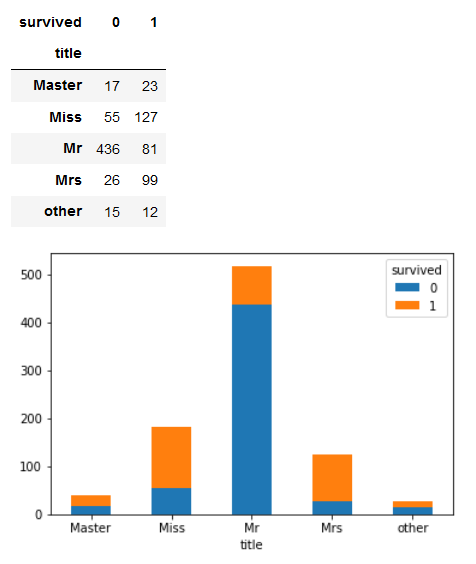
在可视化程序中，我们对八个数据进行可视化：pclass、sex、embarked、family\_size、single、title、fare\_bin以及age\_bin。在读取了data\_train\_new.csv文件后，主要用了如下算法进行每个特征的可视化：

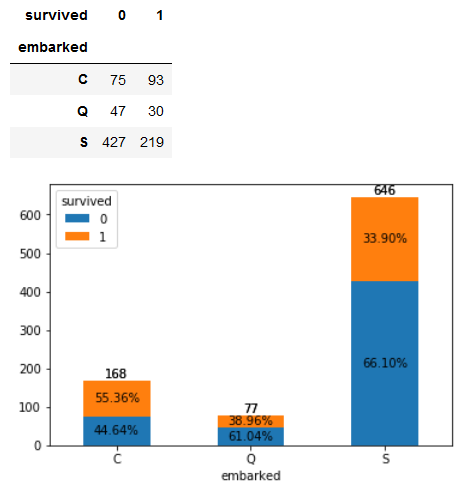
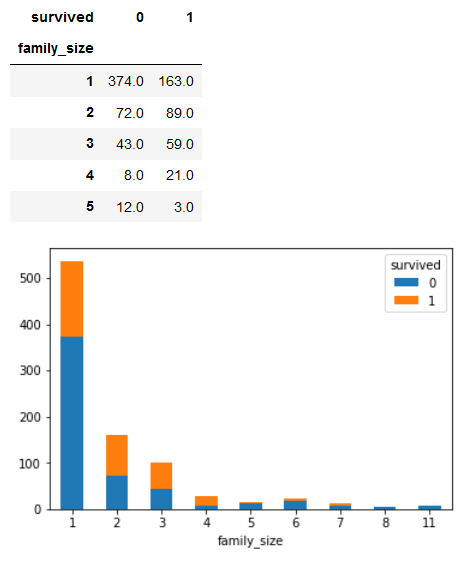


以上图中显示的代码有两个功能：绘制堆积柱状图、在堆积柱状图中部分特征显示百分数。8个特征通过这个算法，可以实现数据的可视化。8个特征的效果图如下：







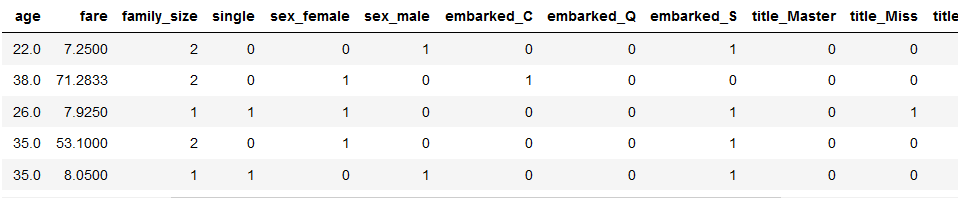
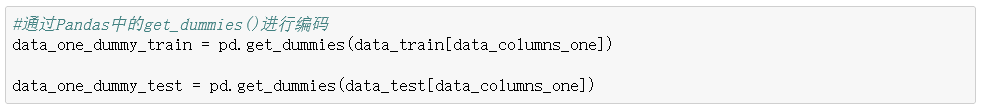


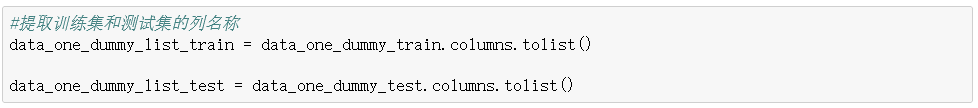
之所以只是将这8个数据可视化出来，目的是这些特征经过前面的数据分析之后，对模型的训练具有非常重大的参考意义，所以这些特征也会被我们用于模型构建中特征的选取。

1. **模型选择与训练结果对比**

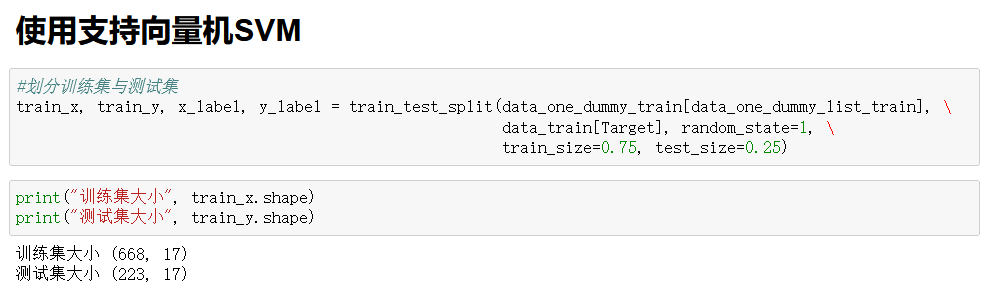
在选择模型进行训练之前，需要先选择特征，我们经过不断地筛选特征和模型准确率的经验，最终我们选择了如下的特征：



之后利用Pandas中的get\_dummies()进行编码，使得列的名称能够与其数据连接起来，如下编码之后，提取训练集和测试集中的列名称：

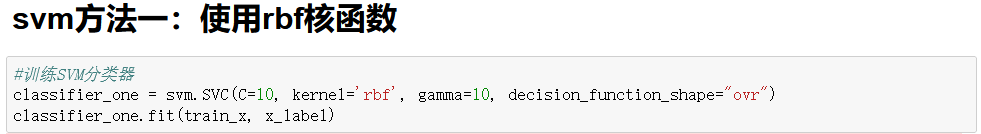
提取的列名称是模型训练中参数的需要，让分类器能够根据列名称去训练和预测数据。

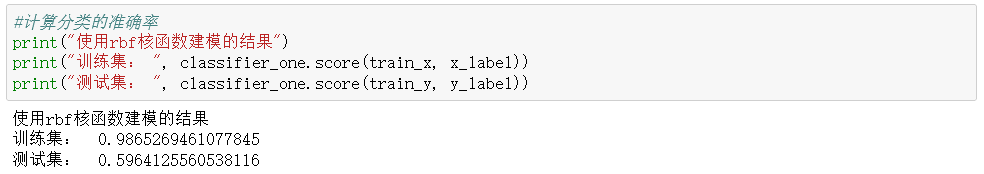
我们选择的是支持向量机SVM进行数据的训练。首先需要将训练集中的数据划分成两个部分，一部分用于训练模型，另一部分用于预测模型的准确率。



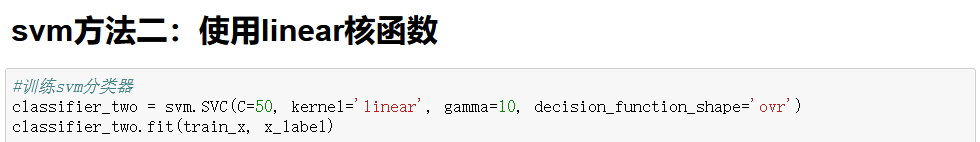
将训练数据集中的数据分成0.75和0.25，分别用于训练和测试，接着打印两个数据集的大小，它们的和就是一个完整训练集。

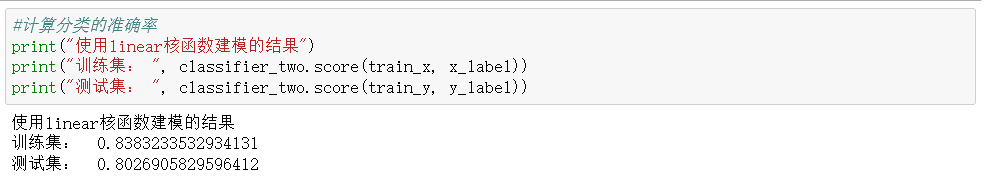
接下来对两个模型的训练结果进行分析和比较，第一个是使用了rbf核函数，第二个是使用了linear核函数。





对于rbf高斯核函数，我们通过不断地调参，完成训练模型之后，最终确认了训练集的准确率高达了98.7%，但是测试集的准确率只有59.6%而已。





对于linear线性核函数，我们也是通过不断地调参，最终确认了C=50，gamma=10，其他的参数默认，所以训练出了最终的结果。**（C是惩罚系数，即对误差的宽容度，我们知道C越大，说明越不能容忍出现误差，也就更容易过拟合。起初我们给参数C=5，但是测试集的准确率太低了，接着我们尝试调高这个C的取值，直到C=50，测试集才出现最大的准确率）。**数据显示，虽然训练集只有83.8%（接近84%）的准确率，但是在测试集中有80.3%的准确率，这个准确率比使用rbf核函数的准确率大得多。

因此，通过这两个核函数以及更改不同参数的效果对比之后，我们最终确认使用了SVM支持向量机的linear核函数的作为分类器。

1. **预测生存情况与分析**

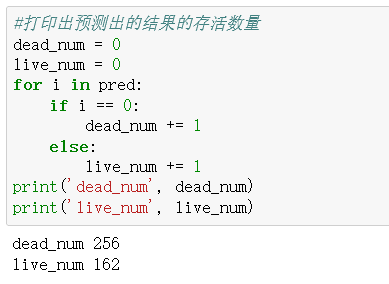
训练好分类器之后，就是使用这个分类器进行测试集test数据的预测：



将预测好的结果pred存放在指定的文件submission.csv文件中：



重新读取test.csv文件，因为这个文件中的PassengerId与submission.csv文件中的PassengerId是相互对应的，然后Survived就对应pred的数据，最后就完成了测试集中survived的填写。

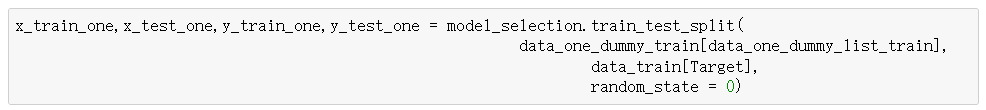


使用训练好的分类器对测试集中的数据进行预测，最终在418个人中，确认了162个人存活，256个人死亡或失踪。

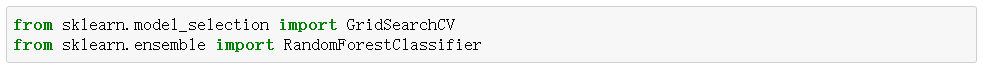
1. **加分项（采用随机森林）**

本作业我们使用的支持向量机SVM，SVM具有很高的冗余度，对数据的分类更为严格，相比之下，我们也尝试过其他新的、老师课堂上没讲过的模型：随机森林random forest，随机森林对于数据就没有像SVM那样严格。随机森林下的每一颗决策树，代表着每个特征，在每颗决策树中随机挑选数据进行训练，然后把每颗决策树的训练结果综合起来，得到最终的分类器。随机森林的“随机”对于前期的数据清洗有着重大的考验，因为如果没有在前期把数据清洗的工作做到最好的话，随机森林有可能选择了例如极端情况的数据进行训练，那么这样的训练结果就会有问题。

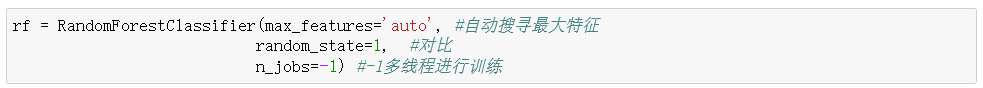
所以我们小组在对数据进行严格清洗之后，尝试着使用随机森林算法，也得到不错的准确率：

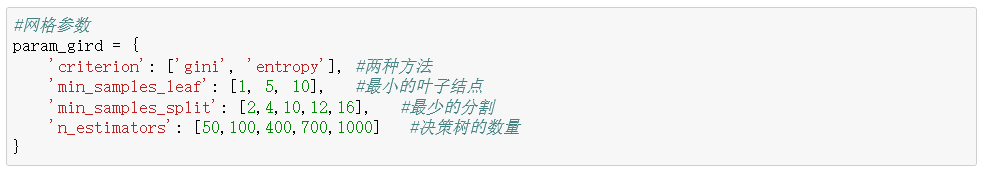


在这里使用了model\_selection函数瓜分数据集和测试集。之后调用随机森林相关的库函数：



接着构建随机森林分类器，采用了网格搜索的方式





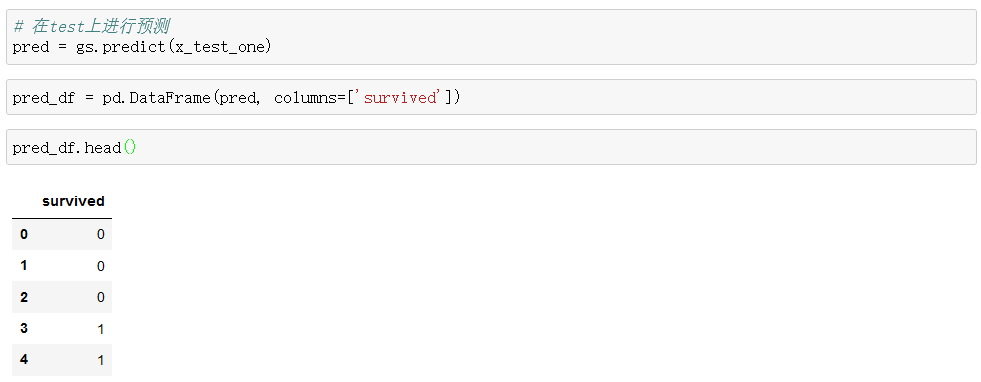


紧接着，使用这个模型进行训练，且训练结果准确率也不错：有83.2%的准确率





最后将训练好的模型用于测试集上的预测：



由这个survived的结果显示，1号和3号与使用了SVM所预测出来的结果不同，我们总结了：不同的分类器有可能会预测出不同的分类结果，如果能够把准确率提得更好，不同的分类器应该就会有相同的分类结果了。到此，对于随机森林算法的运用，我们就尝试到这里。