**AI**

**期 中 报 告**

**学生姓名** 吴语港

**学生学号** SA19225404

**实验日期** 2019/11/3

**实验名称**  Titanic: Machine

Learning from Disaster

**学生学号** SA19225404

**实验地点** 思贤楼302实验室

**索引**

**0.了解Kaggle：**

**1．观察大局：**

**2.获得数据：**

**3.从数据探索和可视化中获得洞见：**

**4.机器学习算法的数据准备：**

**5.选择和训练模型：**

**6.微调模型：**

**0.了解Kaggle：**

**Kaggle**成立于2010年，是一个进行数据发掘和预测竞赛的在线平台。从公司的角度来讲，可以提供一些数据，进而提出一个实际需要解决的问题；从参赛者的角度来讲，他们将组队参与项目，针对其中一个问题提出解决方案，最终由公司选出的最佳方案可以获得5K-10K美金的奖金。

除此之外，Kaggle官方每年还会举办一次大规模的竞赛，奖金高达一百万美金，吸引了广大的数据科学爱好者参与其中。从某种角度来讲，可以把它理解为一个众包平台，类似国内的猪八戒。但是不同于传统的低层次劳动力需求，Kaggle一直致力于解决业界难题，因此也创造了一种全新的劳动力市场——不再以学历和工作经验作为唯一的人才评判标准，而是着眼于个人技能，为顶尖人才和公司之间搭建了一座桥梁。

输入<https://www.kaggle.com/>即可进入Kaggle主页，网站有这么几个版块：

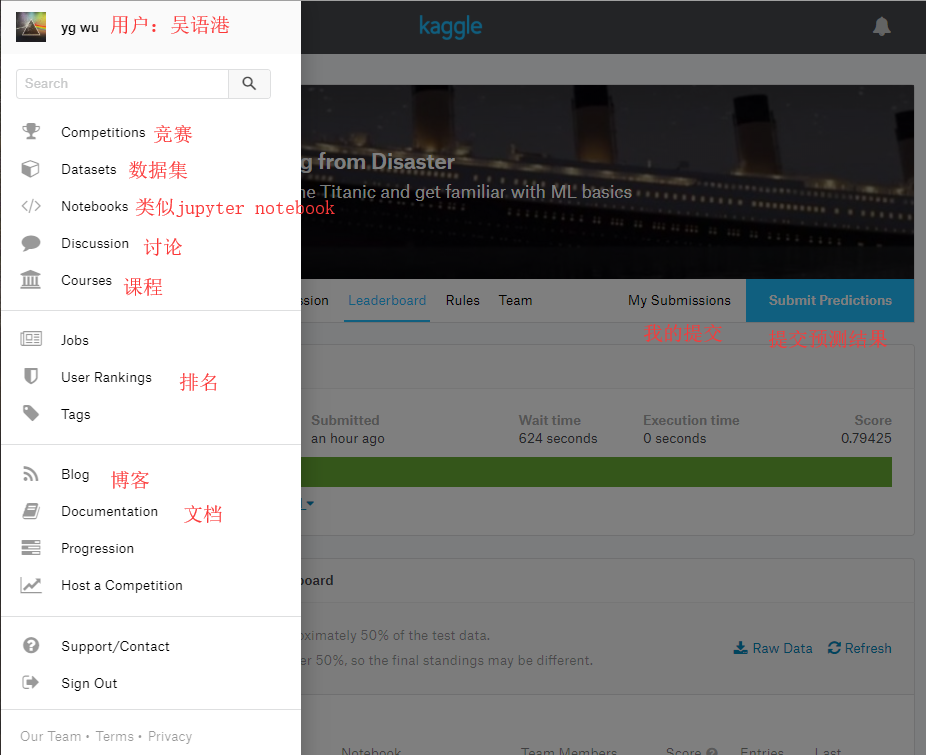
**竞赛competitions**

**数据datasets**

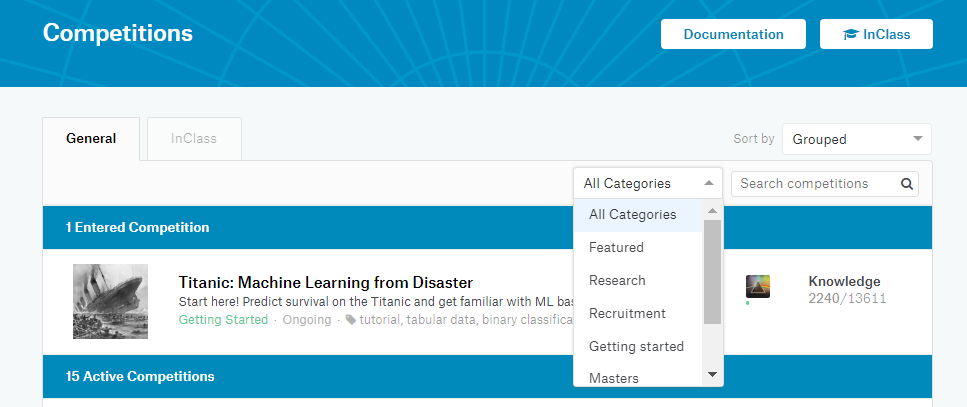
**代码kernels**

**讨论区 Discussion**

**在线课程学习learn**



**Kaggle竞赛分类：**

Kaggle上的竞赛有各种分类，例如奖金极高竞争激烈的的 “Featured”，相对平民化的 “Research”等等。但整体的项目模式是一样的，就是通过出题方给予的训练集建立模型，再利用测试集算出结果用来评比。同时，每个进行中的竞赛项目都会显示剩余时间、参与的队伍数量以及奖金金额，并且会实时更新选手排位。在截止日期之前，所有队伍都可以自由加入竞赛，或者对已经提交的方案进行完善，因此排名也会不断变动，不到最后一刻谁都不知道花落谁家。

**推荐比赛Featured**：瞄准商业问题带有奖金的公开竞赛。如果有幸赢得比赛，不但可以获得奖金，模型也可能会被竞赛赞助商应用到商业实践中呢。

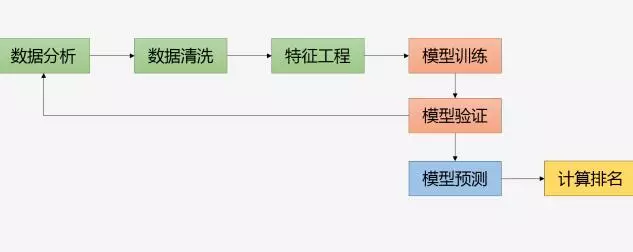
**人才征募Recruitment**：赞助企业寻求数据科学家、算法设计人才的渠道。只允许个人参赛，不接受团队报名。  
研究型Research竞赛通常是机器学习前沿技术或者公益性质的题目。竞赛奖励可能是现金，也有一部分以会议邀请、发表论文的形式奖励。

**游乐场Playground**：题目以有趣为主，比如猫狗照片分类的问题。现在这个分类下的题目不算多，但是热度很高。

**入门比赛Getting Started**：给萌新们一个试水的机会，没有奖金，但有非常多的前辈经验可供学习。很久以前Kaggle这个栏目名称是101的时候，比赛题目还很多，但是现在只保留经典的入门竞赛：手写数字识别、沉船事故幸存估计、脸部识别。

**课业比赛In Class**：是学校教授机器学习的老师留作业的地方，这里的竞赛有些会向public开放参赛，也有些仅仅是学校内部教学使用。

比赛流程



**Kaggle竞赛的排名机制**

在比赛结束之前，参赛者每天最多可以提交5次测试集的预测结果。每一次提交结果都会获得最新的临时排名成绩，直至比赛结束获得最终排名。在比赛过程中，Kaggle将参赛者每次提交的结果取出25%-33%，并依照准确率进行临时排名。在比赛结束时，参赛者可以指定几个已经提交的结果，Kaggle从中去除之前用于临时排名的部分，用剩余数据的准确率综合得到最终排名。所以，比赛过程中用于最终排名的那部分数据，参赛者是始终得不到关于准确率的反馈的。这样一定程度避免参赛模型的过拟合，保证评选出兼顾准确率和泛化能力的模型。

**数据Datasets版块**

每一个竞赛题目都有一个数据入口，描述数据相关的信息，与主页上的Datasets选择一个数据其实指向同一个地方。在这里可以下载到提交结果的示范、测试集、训练集。Kaggle的数据以CSV格式最常见，提交的结果也要求是CSV格式。

**代码Kernels板块**

这是Kaggle最棒的功能！在这里可看到其他参赛者自愿公开的模型代码，是学习和交流的最佳所在！取名为kernels意味支持线上调试和运行代码，目前支持Python、R。对那些暂时缺少硬件资源的参赛者，相当于Kaggle提供了一个“云计算”平台，可以作为一个备选的计算资源。其中Python语言中绝大部分是使用Jupyter Notebook完成的。

**1．观察大局：**

Kaggle是一个数据分析建模的应用竞赛平台，有点类似KDD-CUP（国际知识发现和数据挖掘竞赛），企业或者研究者可以将问题背景、数据、期望指标等发布到Kaggle上，以竞赛的形式向广大的数据科学家征集解决方案。

**故事背景**

泰坦尼克号的沉没是历史上最臭名昭著的海难之一。

1912年4月15日，在她的处女航中，泰坦尼克号与冰山相撞后沉没，2224名乘客和船员中有1502人遇难。

这一耸人听闻的悲剧震惊了国际社会，并为船只带来了更好的安全监管。

造成这样的损失的原因之一是没有足够的救生艇给乘客和船员。

尽管在沉船事故中幸存下来的运气有一些因素，但一些群体比其他群体更有可能存活下来，比如妇女、儿童和上层社会。

在这个挑战中，要求完成对可能存活下来的人的分析。

特别地，要求应用机器学习的工具来预测哪些乘客在这场悲剧中幸存了下来。

Titanic生存预测是Kaggle上参赛人数最多的竞赛之一。它要求参赛选手通过训练数据集分析出什么类型的人更可能幸存，并预测出测试数据集中的所有乘客是否生还。



**常用数据集**

学习机器学习最好使用真实数据进行实验， 而不仅仅是人工数据

集。 我们有成千上万覆盖了各个领域的开放数据集可以选择。 以下是

一些可以获得数据的地方：

·流行的开放数据存储库：

·UC Irvine Machine Learning

Repository（http://archive.ics.uci.edu/ml/）

·Kaggle datasets（https://www.kaggle.com/datasets）

·Amazon’s AWS datasets（http://aws.amazon.com/fr/datasets/）

·元门户站点（它们会列出开放的数据存储库） ：

·http://dataportals.org/

·http://opendatamonitor.eu/

·http://quandl.com/

·其他一些列出许多流行的开放数据存储库的页面：

·Wikipedia’s list of Machine Learning

datasets（https://goo.gl/SJHN2k）

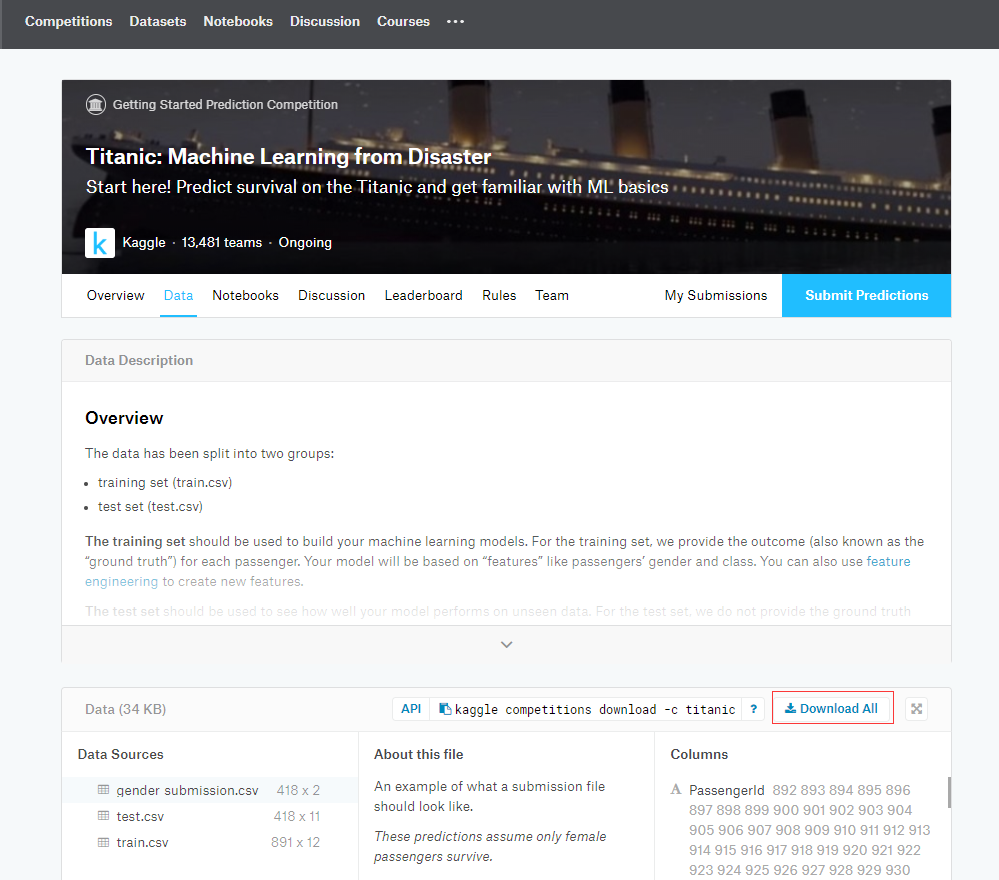
·Quora.com question（http://goo.gl/zDR78y）

·Datasets subreddit（<https://www.reddit.com/r/datasets>）

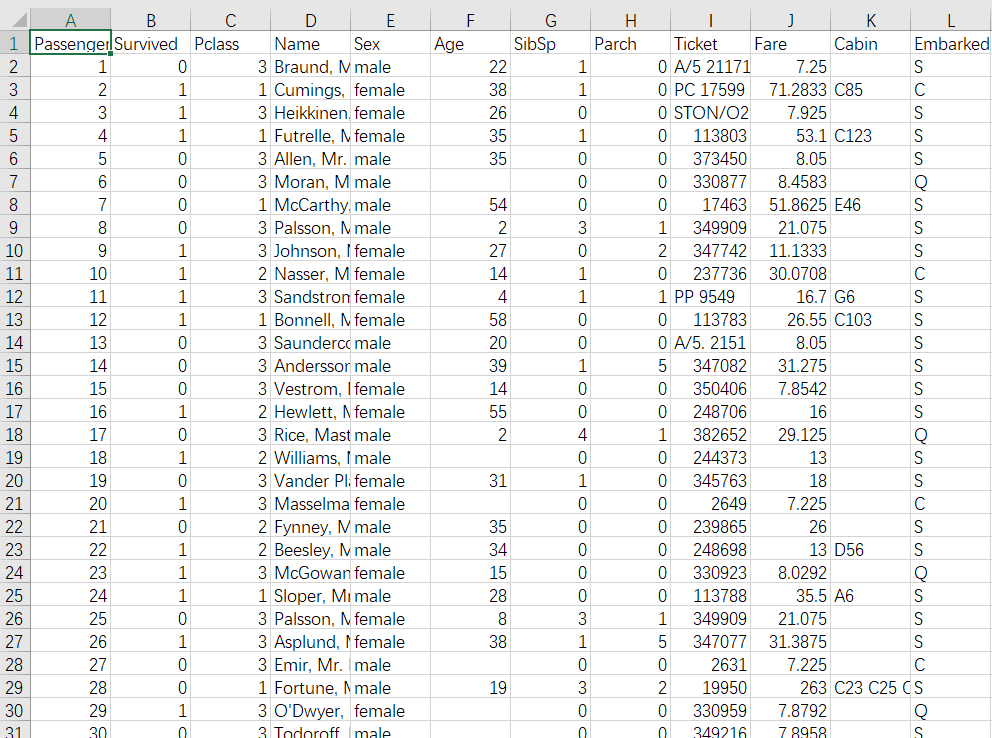
**2.获得数据：**

Titanic生存预测中提供了两组数据：train.csv 和test.csv，分别是训练集和测试集。本实验为练手项目，所以数据就直接给了，不过有缺失的数据，需要我们来处理一下。

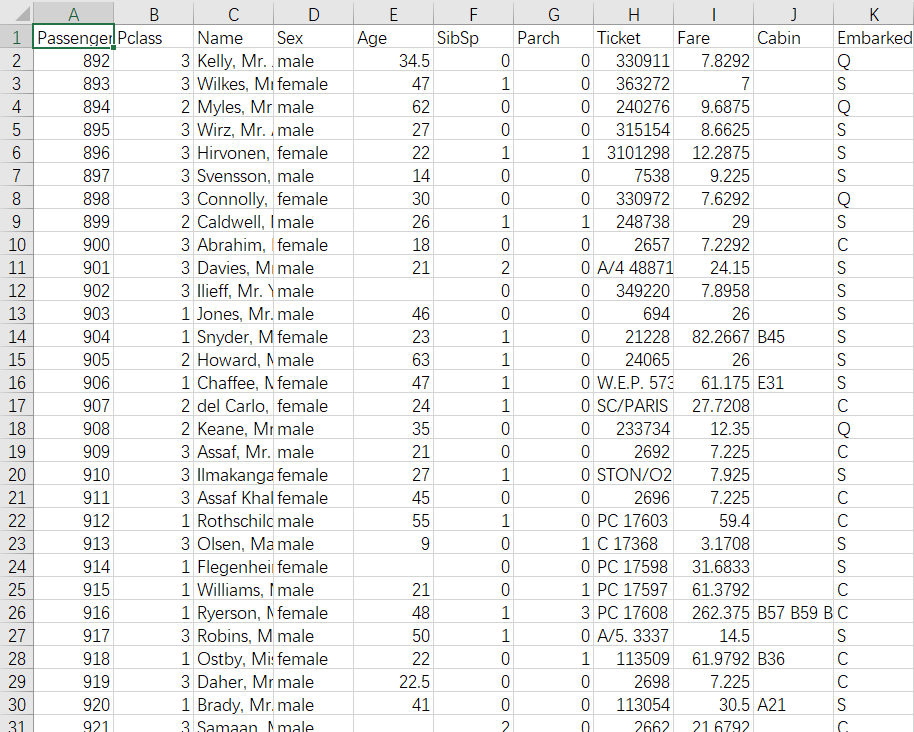
打开kaggle对应项目的数据栏



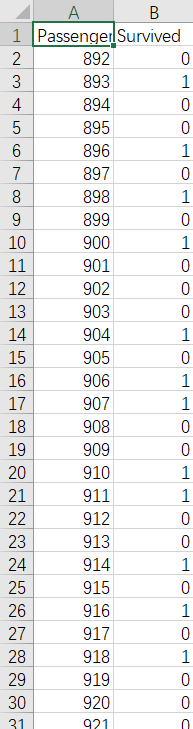
下载数据



Train.csv



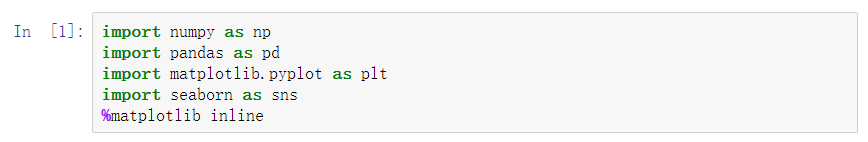
Test.csv



gender\_submission.csv

导入numpy，pandas，matplotlib，seaborn包

注：Seaborn是基于matplotlib的图形可视化python包。它提供了一种高度交互式界面,便于用户能够做出各种有吸引力的统计图表



从当前目录导入train.csv和test.csv，并显示两个数据文件的相关信息





**了解数据特征含义：**

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

PassengerId：乘客编号，唯一；

Survived：是否获救 0未获救 1获救；

Pclass：客舱等级；

Name：乘客姓名；

Sex：性别；

Age：年龄；

SibSp：在船上的兄弟姐妹或者配偶数量；

Parch：在船上的父母或儿女数量；

Ticket：船票编号；

Fare：船票价格；

Cabin：客舱号；

Embarked：登船港口

其中PassengerId为预测唯一序号，Survived为预测目标值。

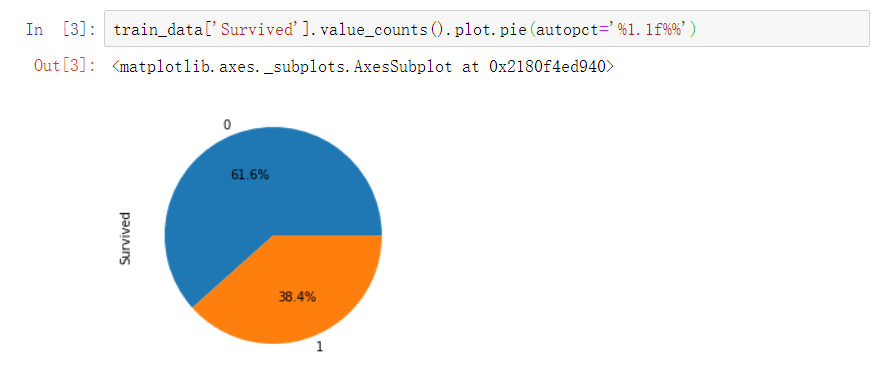
**了解数据基本情况：**

可以发现，训练数据集一共891个样本，其中Age、Cabin、Embarked属性有缺失值。我们需要对缺失值进行处理，Age字段缺失值可以用预测年龄来代替，Cabin字段由于Cabin项缺失太多，只能将有无Cain作为特征值进行建模，Embarked字段由于数据仅有两个样本缺值，可以选择随机填充该属性特征例值，或者删除这两个样本。

**查看各特征值有无异常**

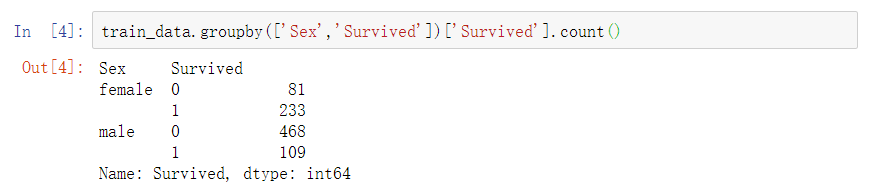
Survived属性包含0、1两种值无异常，其中549死亡，342存活；Pclass包含1、2、3三种值无异常；Name属性包含891个值，无重复无异常；Sex包含male 577位，female 314位；SibSp 包含0、1、2、3、4、5、8几种值，其中属性为n的人数，应该为n+1的倍数，即若某人有8个SIbSp则至少有9个人互为SibSp，该数据出现的原因是因为训练数据只包含部分船上的人员，因此该特征暂未发现异常；Parch特征类似，暂未发现异常；Ticket字段包含681个不同值，该值为每个票据的基本特征，与生存与否关系不大；特征Fare为船票价格，未发现异常，可能与Pclass属性有关，也可以划分区间作为预测是否存活的特征；特征Cabin存在缺失值可单独处理，另外我们发现客舱编号以A、B、C、D、E开头，可能与船舱位置有关；特征Embarked包含三个值，船上乘客来源于三个地方。

通过matplotlib的饼图模块看到了存活比例，大概1/3多一些

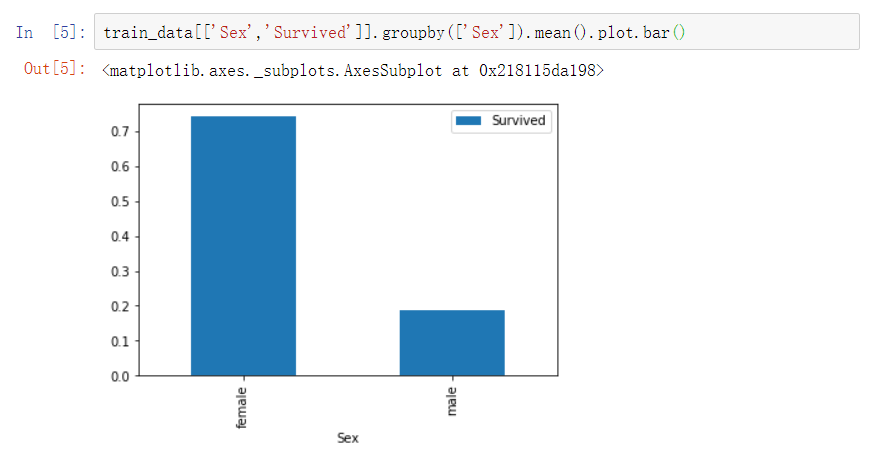
**3.从数据探索和可视化中获得洞见：**

**（1）性别与生存的关系：**

通过表格查看女性存活233/314，男性存活109/577



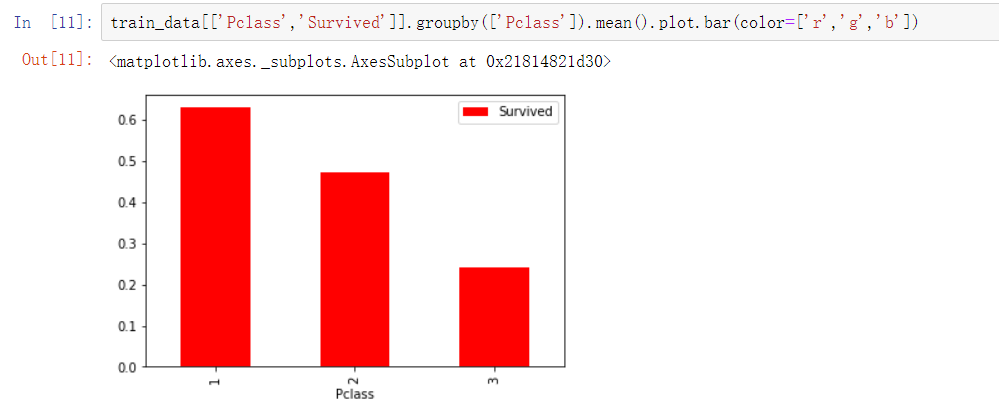
通过Matplotlib的条形统计图bar查看



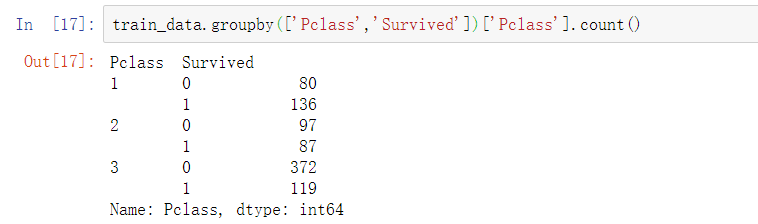
女性的存活比例远大于男性，符合妇女先走的特点

**（2）船舱等级与生存的关系**

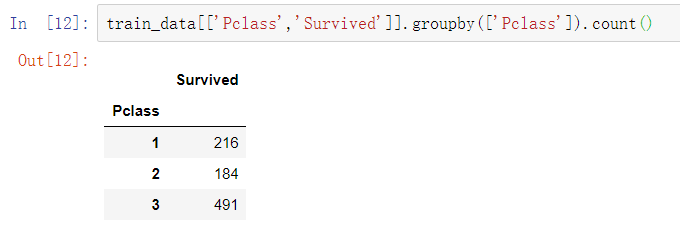
船舱等级与生存的关系，图示：



表格计数：

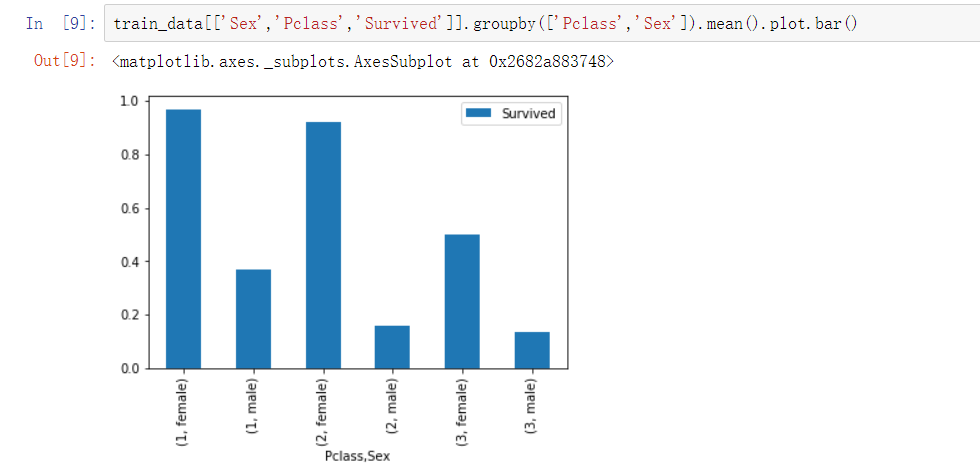


对比两种语句的不同表现形式，上面分组参考了两个属性，下面的只参考了一个属性，上面的展示的更好

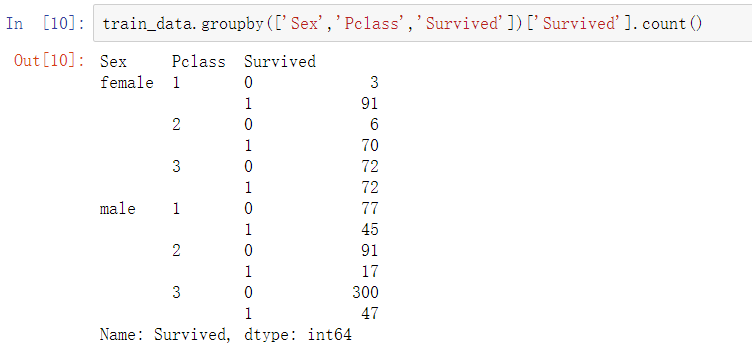


看来船舱等级越高，生存率越高，下面将性别，船舱等级和存活情况放一起比较

如下图所示：



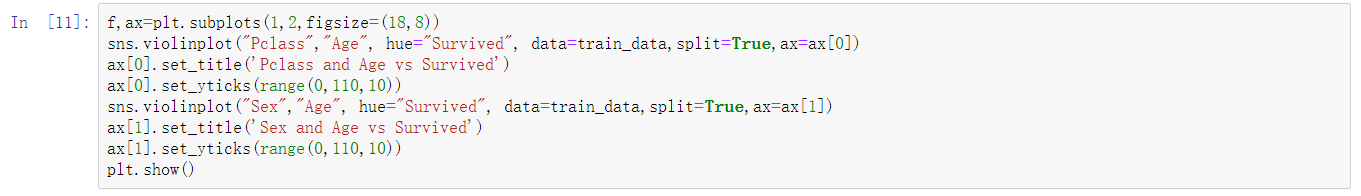
表格计数：



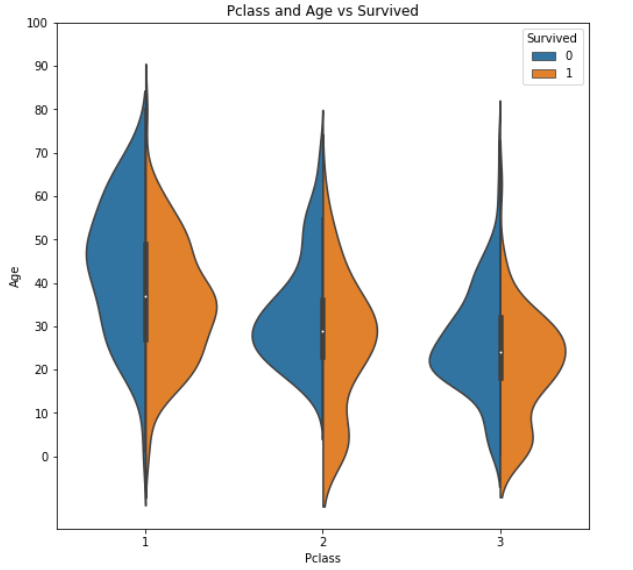
从上图和表中明显可以看到，虽然泰坦尼克号逃生总体符合妇女优先，但是对各个等级船舱还是有区别的，而且一等舱中的男子凭借自身的社会地位强行混入了救生艇。如白星航运公司主席伊斯梅（他否决了配备48艘救生艇的想法，认为少点也没关系）则抛下他的乘客、他的船员、他的船，在最后一刻跳进可折叠式救生艇C（共有39名乘客）。

**（3）年龄与存活的关系**

下面的图使用了matplot的sns模块，表现力更强



船舱等级和年龄与存活的关系：



船舱等级越高，男性年龄越大，看来社会地位和经济实力的提升需要男性的长时间奋斗，但是在存活的人当中，特别是头等舱年轻的人更容易存活，看来是要保存年轻的生产力，还要为祖国好好的做贡献。年长的人就准备牺牲自我了。

性别和年龄与存活的关系：



存活的人当中，女性比男性更年长不知道是不是女性平均寿命比男性长的缘故。

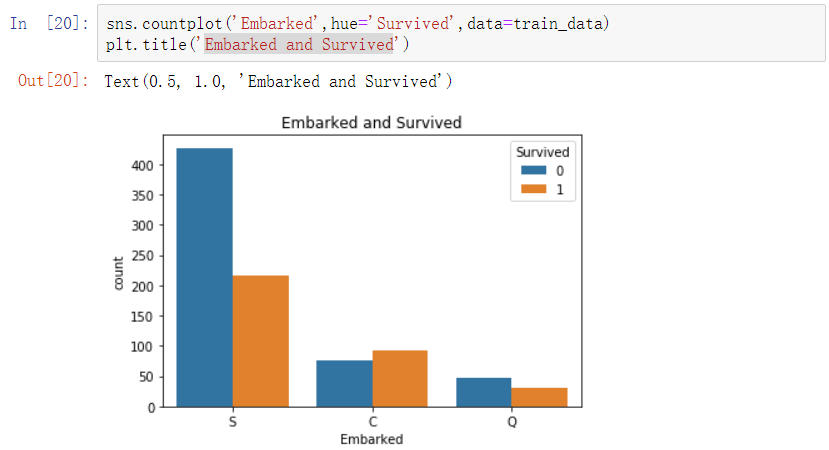
**（4）称呼与存活关系**

在数据的Name项中包含了对该乘客的称呼，如Mr、Miss、Mrs等，这些信息包含了乘客的年龄、性别、也有可能包含社会地位，如Dr、Lady、Major、Master等称呼。

这一项不方便用图表展示，但是在特征工程中，我们会将其加入到特征中。

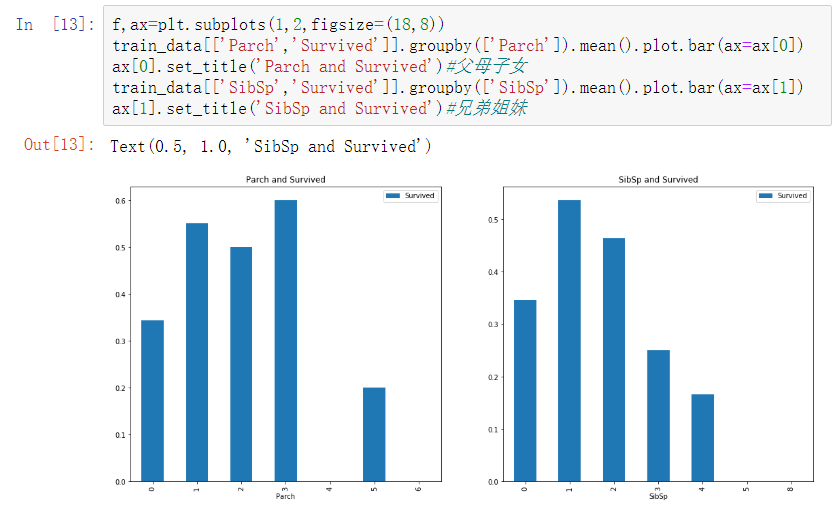
**（5）登船港口与存活关系**

泰坦尼克号从英国的南安普顿港出发，途径法国瑟堡和爱尔兰昆士敦，一部分在瑟堡或昆士敦下船的人逃过了一劫。



**（6）船上亲友人数与存活关系**

从图中可以看到，孤身一人存活率很低，但是如果亲友太多，难以估计周全，也很危险



**（7）其他因素**

剩余因素还有船票价格、船舱号和船票号，这三个因素都可能会影响乘客在船中的位置从而影响逃生顺序，但是因为这三个因素与生存之间看不出明显规律，所以在后期模型融合时，将这些因素交给模型来决定其重要性。

**（8）总结**

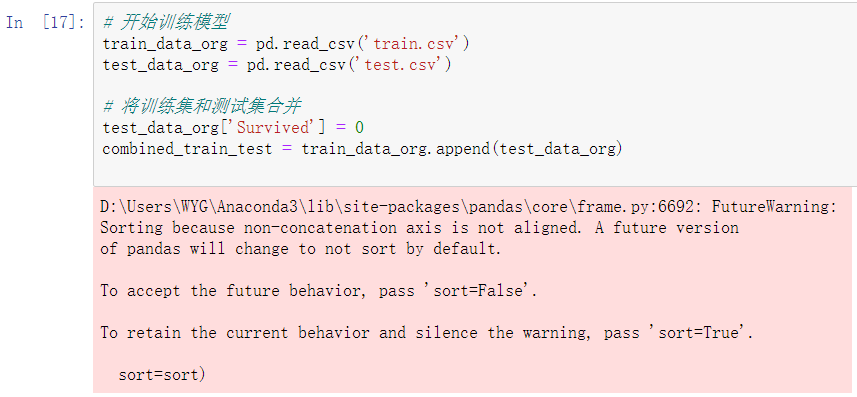
通过分析，我们发现当时的历史背景下，所属阶层、经济水平处于高位的人更容易获救，并且由于当时的急救策略，女人、孩子以及有家庭成员存在的人更容易获救。并且能够发现，当时上传的三个港口经济发展情况。

通过，集中模型，我们对乘客生存进行预测，发现对于预测结果并不是特别满意，想要获得一个理想对模型，需要对数据进行进一步的特征化，同时可以利用voting对不同模型进行融合，调整参数，增加预测准确率

其实我有个想法，在救生艇容量一定的情况下妇女和小孩的体重比较轻，能救出更多的生命，可能是一个原因。

**4.机器学习算法的数据准备：**

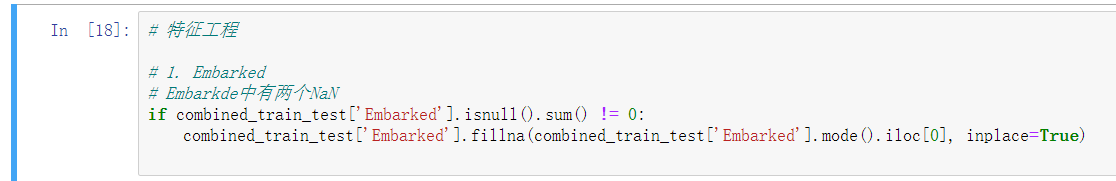
首先将train和test合并一起进行特征工程处理：



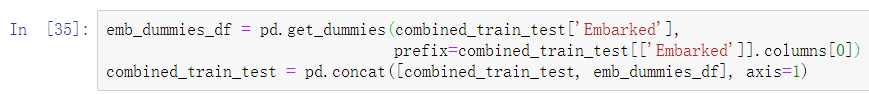
特征工程即从各项参数中提取出可能影响到最终结果的特征，作为模型的预测依据。特征工程一般应先从含有缺失值即NaN的项开始。

**（1）Embarked（登船口）**

先填充缺失值，对缺失的Embarked以众数来填补

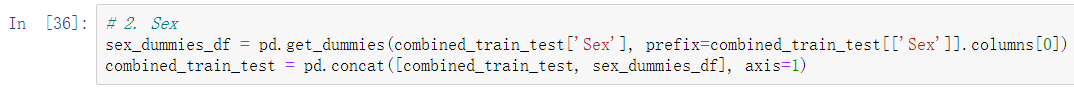


再将Embarked的三个上船港口分为3列，每一列均只包含0和1两个值



**（2）Sex (性别)**

无缺失值，直接分列



**（3）Name(名字)**

**名字当中包含了一些称谓，可能包含一些社会地位信息，需要提取出一些特征**

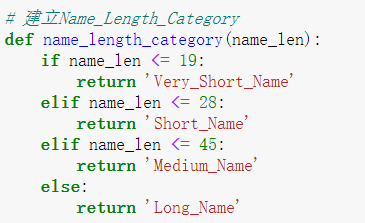
从名字中提取出称呼：

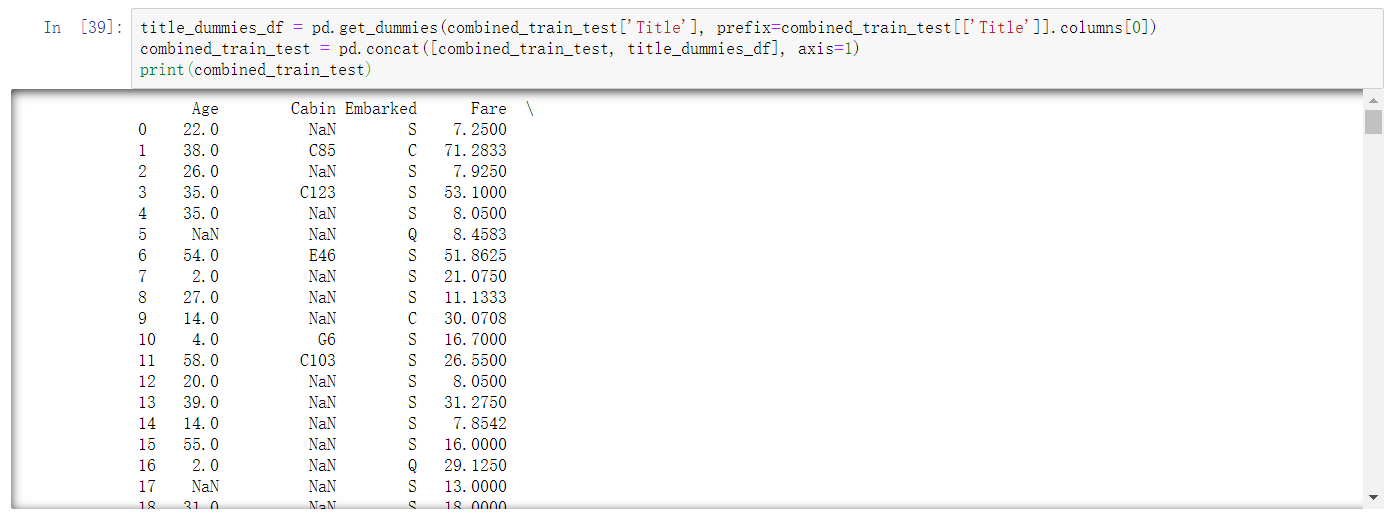


将各式称呼统一：

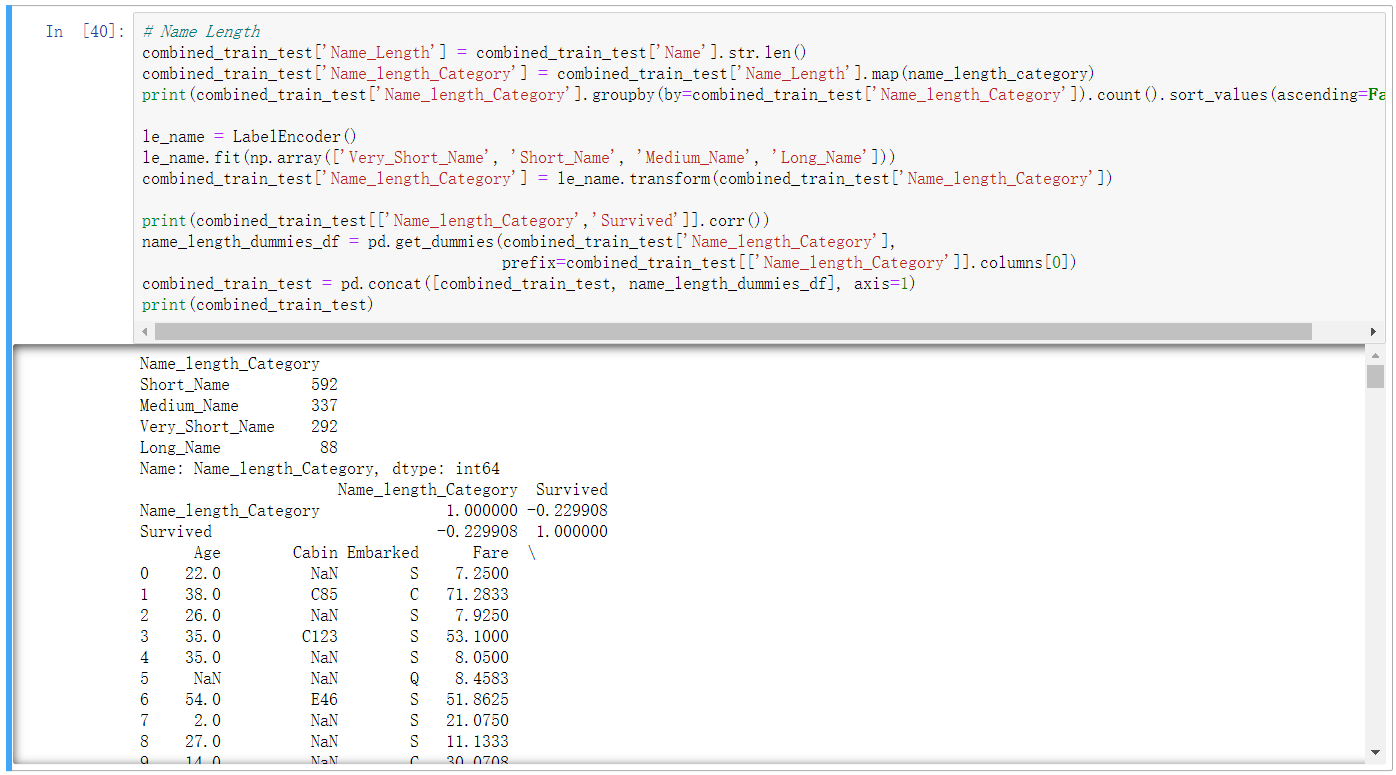


对名字长短进行分类

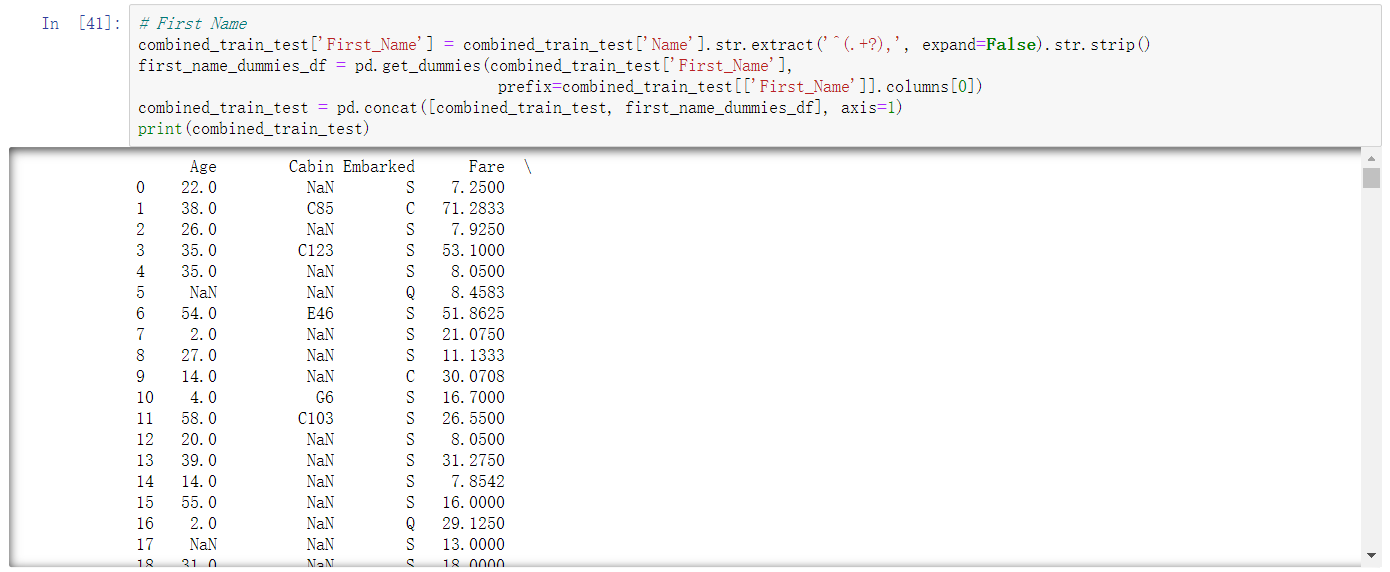




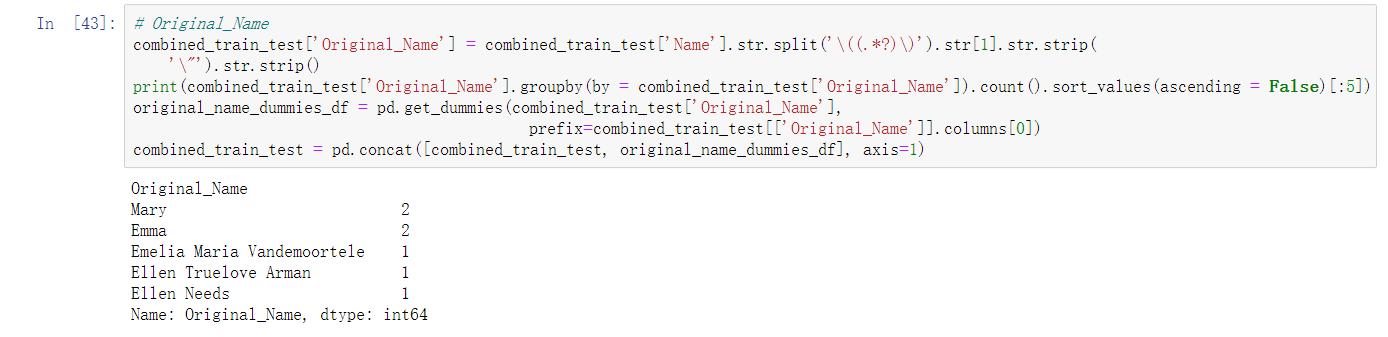
名字的长度



对“名”进行处理

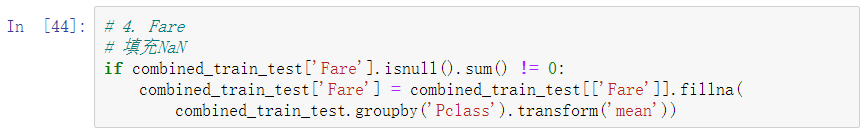


对“姓”进行处理

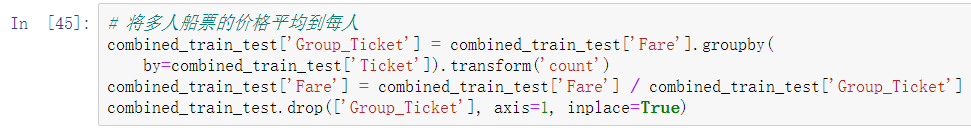


**（4）Fare(票价)**

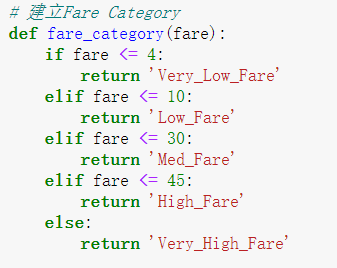
填充NaN，按一二三等舱各自的均价来填充。



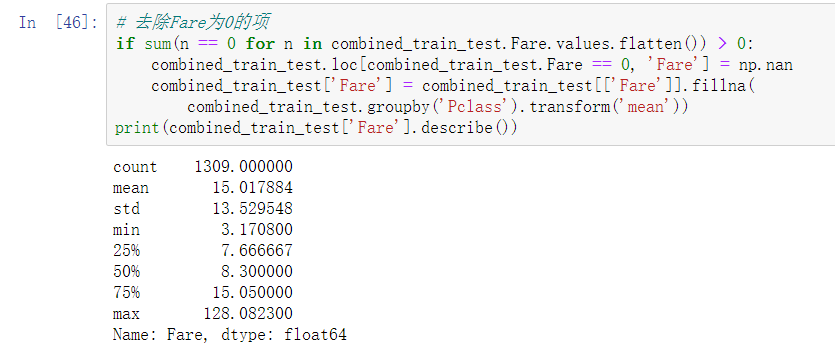
泰坦尼克号中有家庭团体票（分析Ticket号可以得到），所以需要将团体票分到每个人。

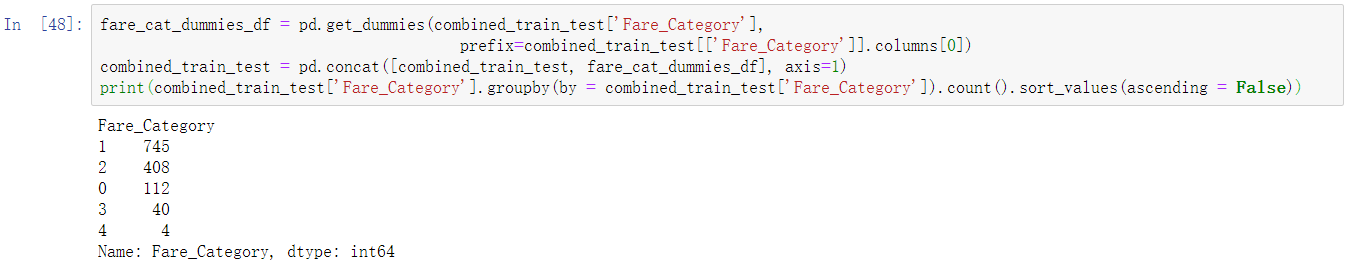
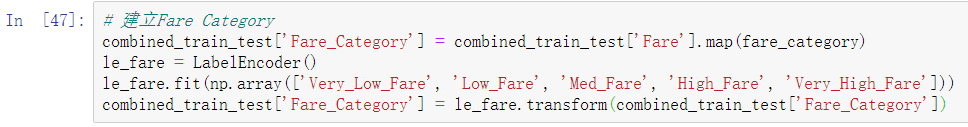


票价分级



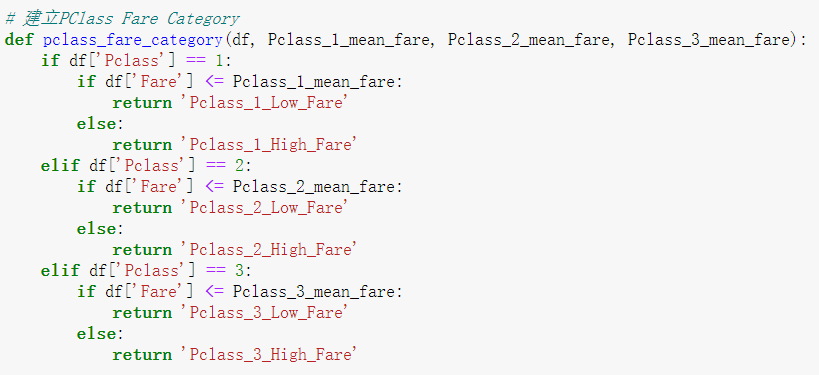
分列（这一项分列与不分列均可）

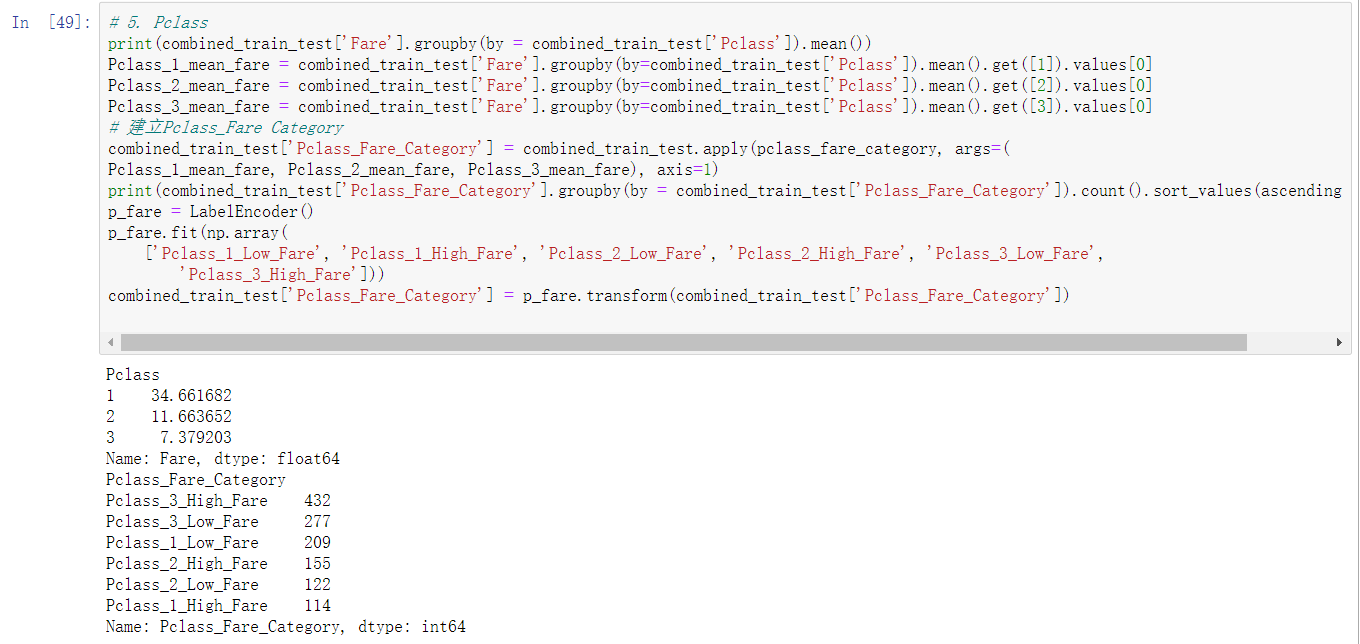




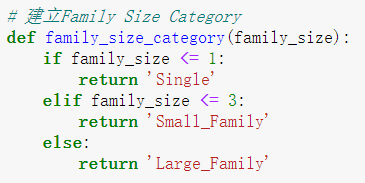
**（5）Pclass(船舱等级)**

Pclass项本身已经不需要处理，为了更好地利用这一项，我们假设一二三等舱各自内部的票价也与逃生方式相关，从而分出高价一等舱、低价一等舱……这样的分类。

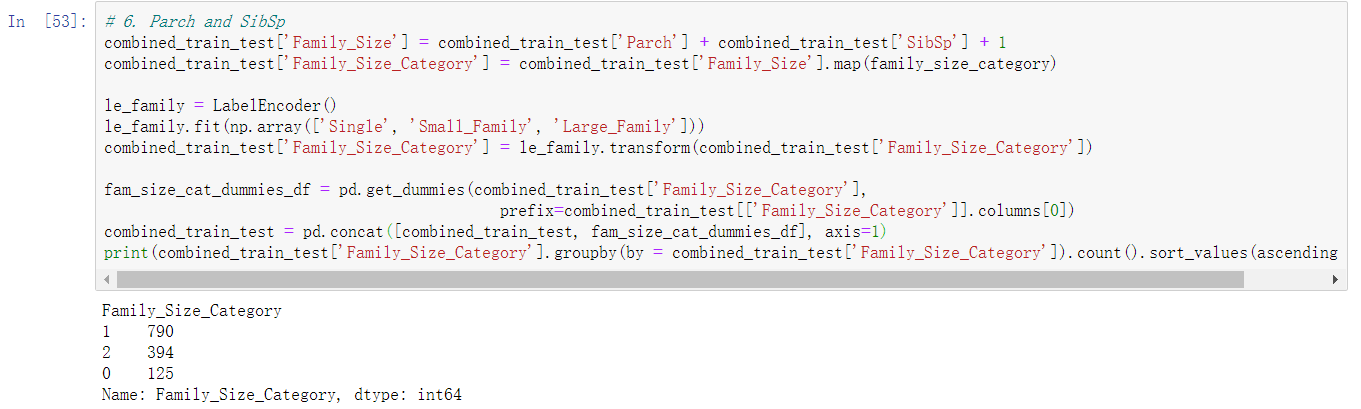




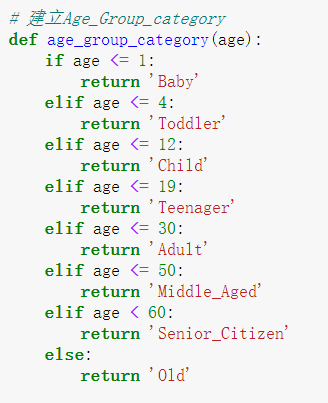
**（6）Parch and SibSp(亲属：父母子女+兄弟姐妹)**



这两组数据都能显著影响到Survived，但是影响方式不完全相同，所以将这两项合并成FamilySize组的同时保留这两项。



**（7）Age(年龄)**



因为Age项缺失较多，所以不能直接将其填充为众数或者平均数。常见有两种填充法，一是根据Title项中的Mr、Master、Miss等称呼的平均年龄填充，或者综合几项（Sex、Title、Pclass）的Age均值。二是利用其他组特征量，采用机器学习算法来预测Age，本例采用的是第二种方法。

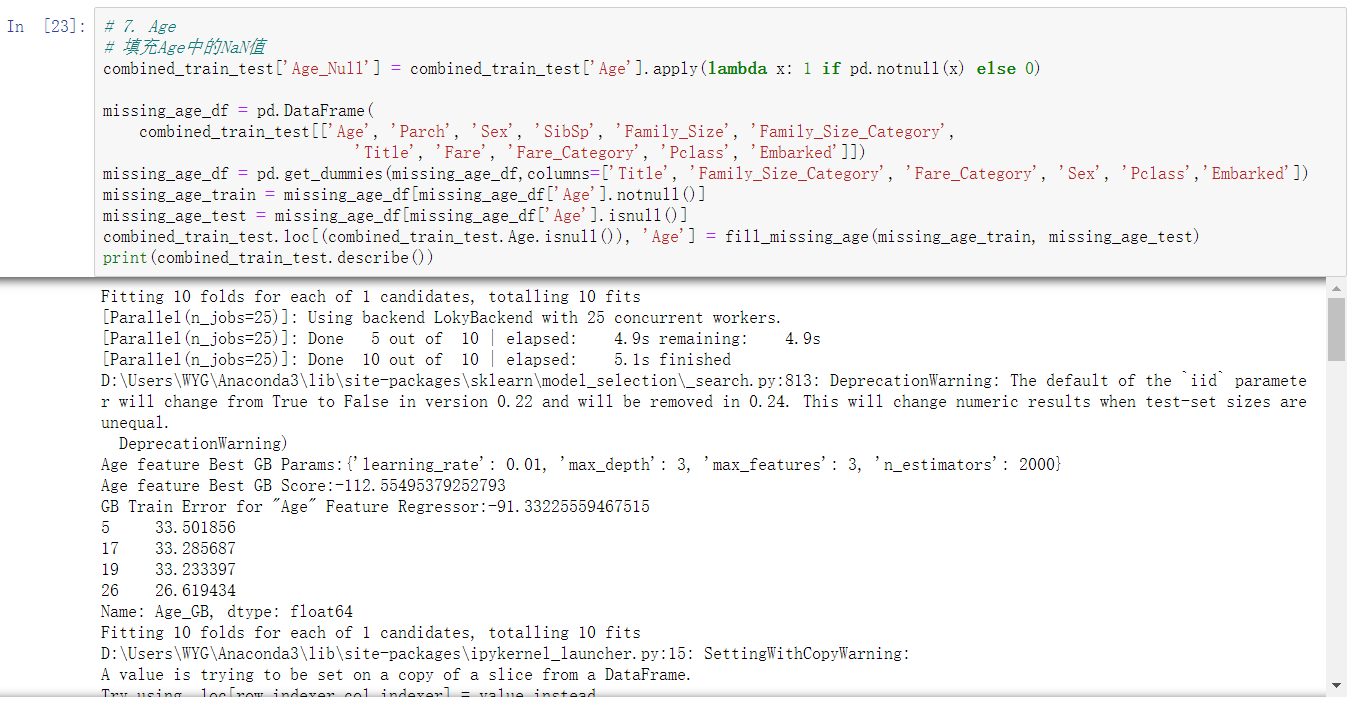
将Age完整的项作为训练集、将Age缺失的项作为测试集。

建立融合模型

使用梯度提升和线性回归来填充age值



填充Age

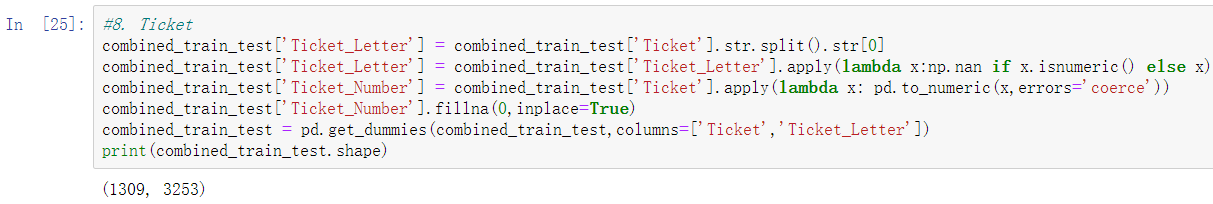


检查异常值的情况



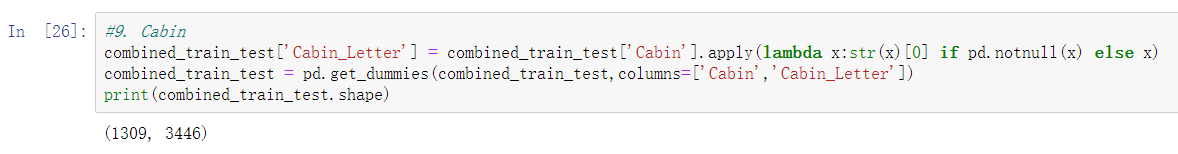
**（8）Ticket：**

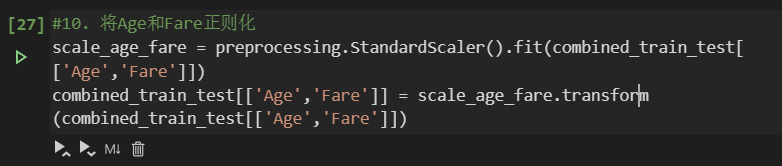
将Ticket中的字母与数字分开，分为Ticket\_Letter和Ticket\_Number两项。



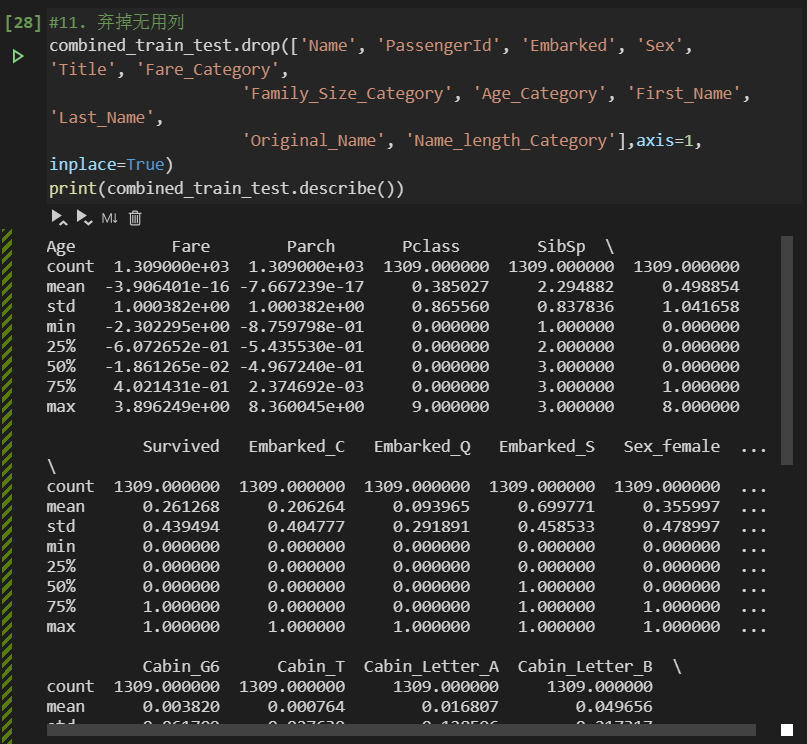
**（9）Cabin：**

Cabin项缺失太多，只能将有无Cain作为特征值进行建模

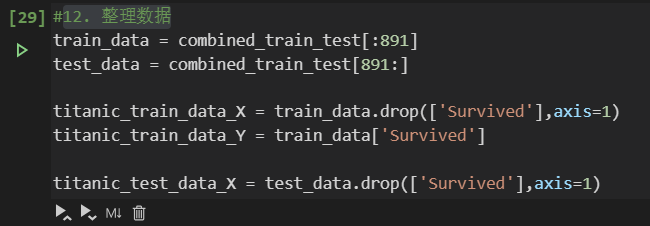
**（10）将Age和Fare正则化：**



**（11）弃掉无用列：**



**（12）整理数据**



将训练集与测试集分离

将输入属性和输出属性分离

准备开始训练模型

**5.选择和训练模型：**

**（1）用几个模型筛选出较为重要的特征：**

# 筛选重要特征

def get\_top\_n\_features(titanic\_train\_data\_X, titanic\_train\_data\_Y, top\_n\_features):

# 随机森林randomforest

rf\_est = RandomForestClassifier(random\_state=42)

rf\_param\_grid = {'n\_estimators': [500], 'min\_samples\_split': [2, 3], 'max\_depth': [20]}

rf\_grid = model\_selection.GridSearchCV(rf\_est, rf\_param\_grid, n\_jobs=25, cv=10, verbose=1)

rf\_grid.fit(titanic\_train\_data\_X,titanic\_train\_data\_Y)

print('Top N Features Best RF Params:' + str(rf\_grid.best\_params\_))

print('Top N Features Best RF Score:' + str(rf\_grid.best\_score\_))

print('Top N Features RF Train Error:' + str(rf\_grid.score(titanic\_train\_data\_X, titanic\_train\_data\_Y)))

feature\_imp\_sorted\_rf = pd.DataFrame({'feature': list(titanic\_train\_data\_X),

'importance': rf\_grid.best\_estimator\_.feature\_importances\_}).sort\_values('importance', ascending=False)

features\_top\_n\_rf = feature\_imp\_sorted\_rf.head(top\_n\_features)['feature']

print('Sample 25 Features from RF Classifier')

print(str(features\_top\_n\_rf[:25]))

# AdaBoost算法

ada\_est = ensemble.AdaBoostClassifier(random\_state=42)

ada\_param\_grid = {'n\_estimators': [500], 'learning\_rate': [0.5, 0.6]}

ada\_grid = model\_selection.GridSearchCV(ada\_est, ada\_param\_grid, n\_jobs=25, cv=10, verbose=1)

ada\_grid.fit(titanic\_train\_data\_X, titanic\_train\_data\_Y)

print('Top N Features Best Ada Params:' + str(ada\_grid.best\_params\_))

print('Top N Features Best Ada Score:' + str(ada\_grid.best\_score\_))

print('Top N Features Ada Train Error:' + str(ada\_grid.score(titanic\_train\_data\_X, titanic\_train\_data\_Y)))

feature\_imp\_sorted\_ada = pd.DataFrame({'feature': list(titanic\_train\_data\_X),

'importance': ada\_grid.best\_estimator\_.feature\_importances\_}).sort\_values('importance', ascending=False)

features\_top\_n\_ada = feature\_imp\_sorted\_ada.head(top\_n\_features)['feature']

print('Sample 25 Feature from Ada Classifier:')

print(str(features\_top\_n\_ada[:25]))

# 极限树ExtraTree

et\_est = ensemble.ExtraTreesClassifier(random\_state=42)

et\_param\_grid = {'n\_estimators': [500], 'min\_samples\_split': [3, 4], 'max\_depth': [15]}

et\_grid = model\_selection.GridSearchCV(et\_est, et\_param\_grid, n\_jobs=25, cv=10, verbose=1)

et\_grid.fit(titanic\_train\_data\_X, titanic\_train\_data\_Y)

print('Top N Features Best ET Params:' + str(et\_grid.best\_params\_))

print('Top N Features Best ET Score:' + str(et\_grid.best\_score\_))

print('Top N Features ET Train Error:' + str(et\_grid.score(titanic\_train\_data\_X, titanic\_train\_data\_Y)))

feature\_imp\_sorted\_et = pd.DataFrame({'feature': list(titanic\_train\_data\_X),

'importance': et\_grid.best\_estimator\_.feature\_importances\_}).sort\_values('importance', ascending=False)

features\_top\_n\_et = feature\_imp\_sorted\_et.head(top\_n\_features)['feature']

print('Sample 25 Features from ET Classifier:')

print(str(features\_top\_n\_et[:25]))

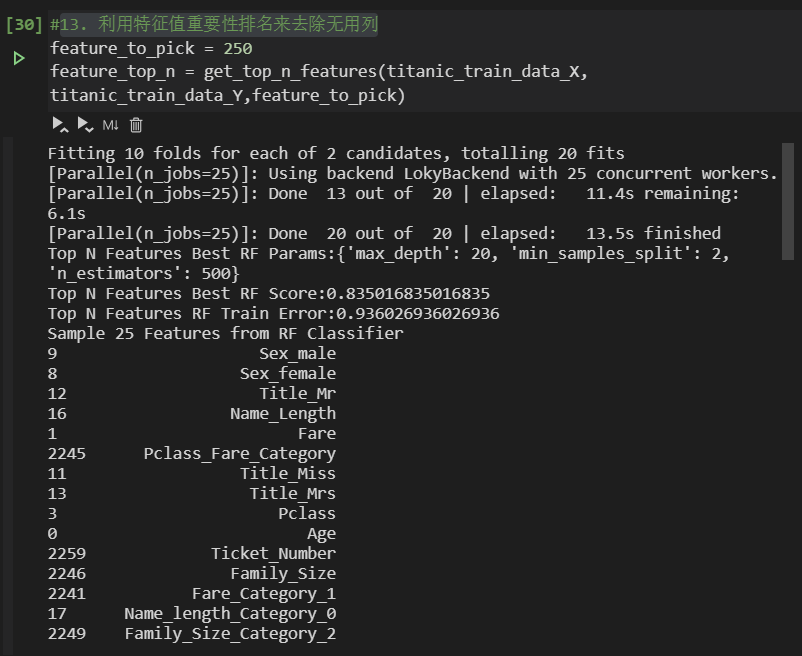
# 融合以上三个模型

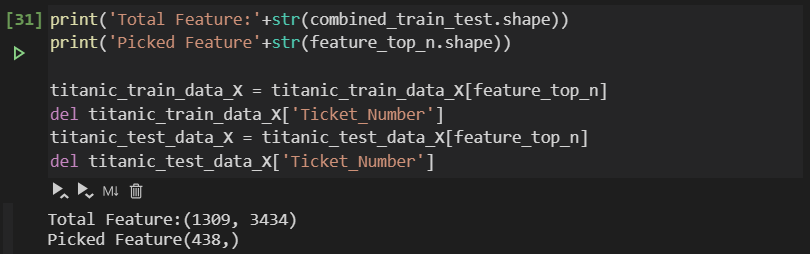
features\_top\_n = pd.concat([features\_top\_n\_rf, features\_top\_n\_ada, features\_top\_n\_et],ignore\_index=True).drop\_duplicates()

return features\_top\_n

**（2）根据筛选出的特征值挑选训练集和测试集：**

利用特征值重要性排名来去除无用列





**（3）利用votingClassifer建立最终预测模型：**

#14.建立模型

rf\_est = ensemble.RandomForestClassifier(n\_estimators = 750, criterion = 'gini', max\_features = 'sqrt',

max\_depth = 3, min\_samples\_split = 4, min\_samples\_leaf = 2,

n\_jobs = 50, random\_state = 42, verbose = 1)

gbm\_est = ensemble.GradientBoostingClassifier(n\_estimators=900, learning\_rate=0.0008, loss='exponential',

min\_samples\_split=3, min\_samples\_leaf=2, max\_features='sqrt',

max\_depth=3, random\_state=42, verbose=1)

et\_est = ensemble.ExtraTreesClassifier(n\_estimators=750, max\_features='sqrt', max\_depth=35, n\_jobs=50,

criterion='entropy', random\_state=42, verbose=1)

voting\_est = ensemble.VotingClassifier(estimators = [('rf', rf\_est),('gbm', gbm\_est),('et', et\_est)],

voting = 'soft', weights = [3,5,2],

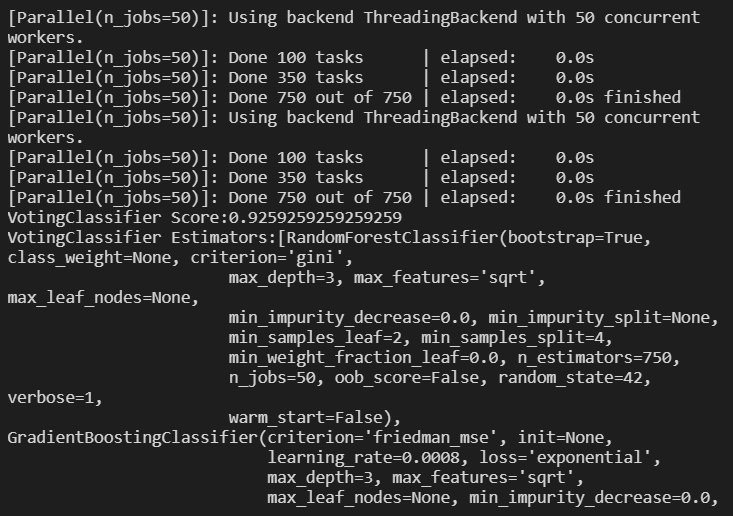
n\_jobs = 50)

voting\_est.fit(titanic\_train\_data\_X,titanic\_train\_data\_Y)

print('VotingClassifier Score:' + str(voting\_est.score(titanic\_train\_data\_X,titanic\_train\_data\_Y)))

print('VotingClassifier Estimators:' + str(voting\_est.estimators\_))

如果不用VotingClassifier的也可以自己根据这几个模型的测试准确率给几个模型的结果自定义权重，将最终的加权平均值作为预测结果。



**（4）预测及生成提交文件**

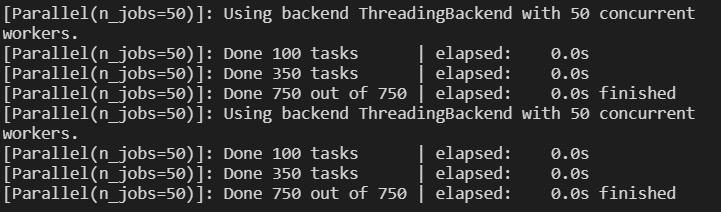
#预测

titanic\_test\_data\_X['Survived'] = voting\_est.predict(titanic\_test\_data\_X)

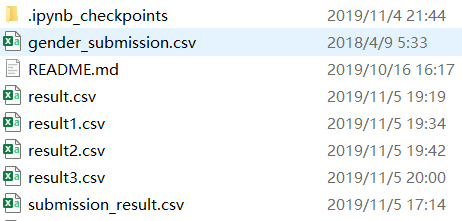
submission = pd.DataFrame({'PassengerId':test\_data\_org.loc[:,'PassengerId'],

'Survived':titanic\_test\_data\_X.loc[:,'Survived']})

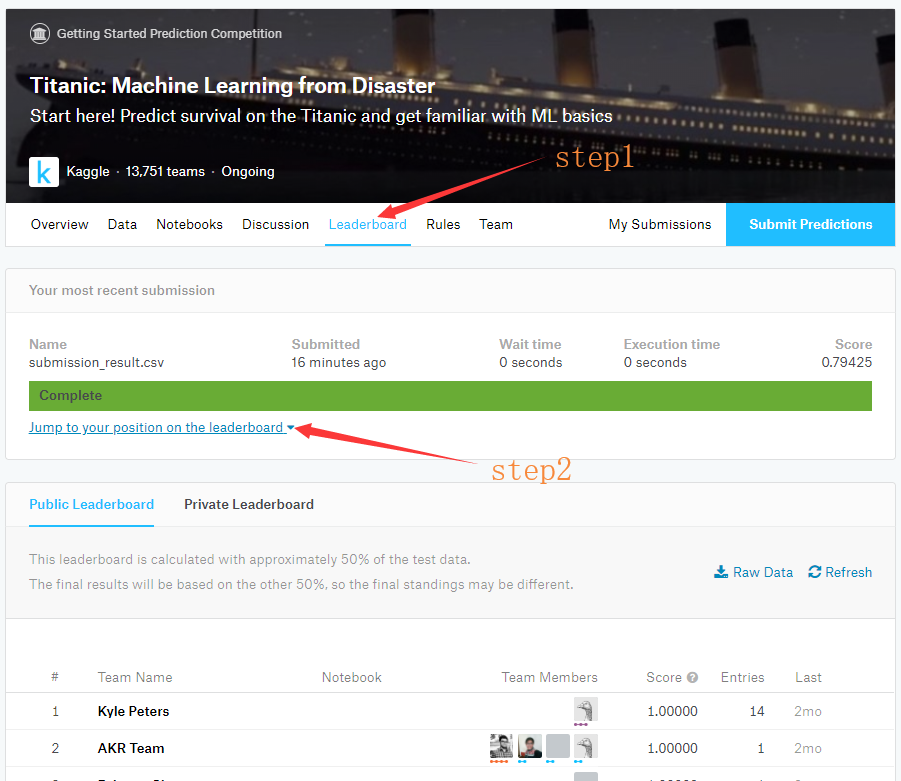
submission.to\_csv('result.csv',index=False,sep=',')



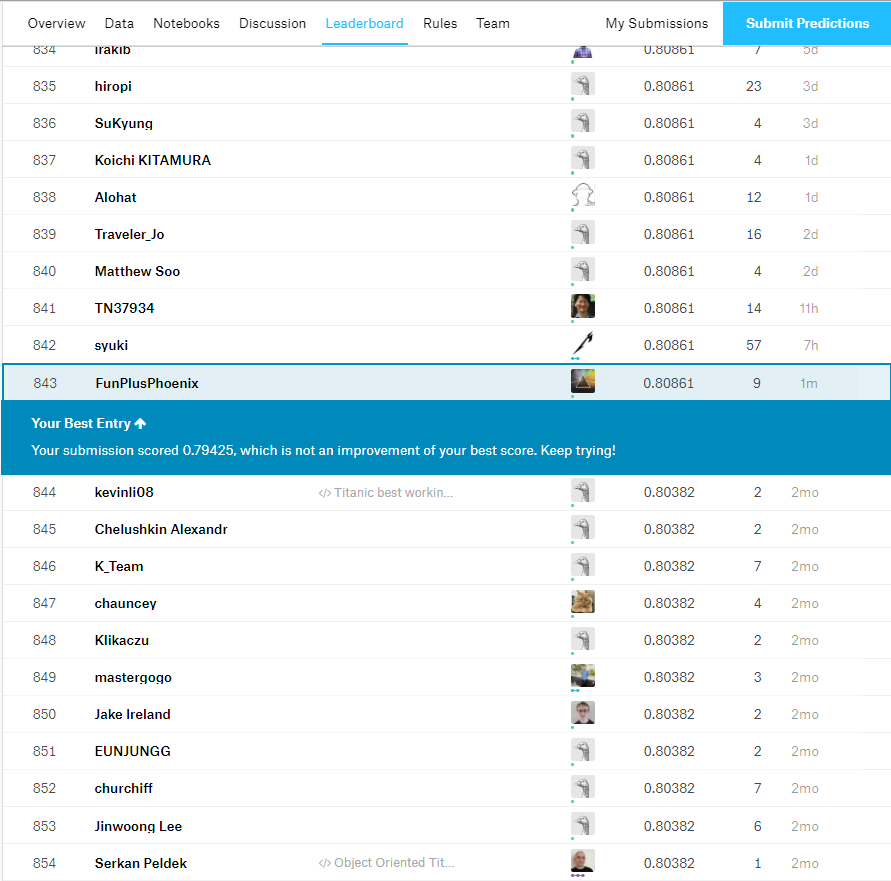
输出文件已经生成，提交到kaggle查看排名



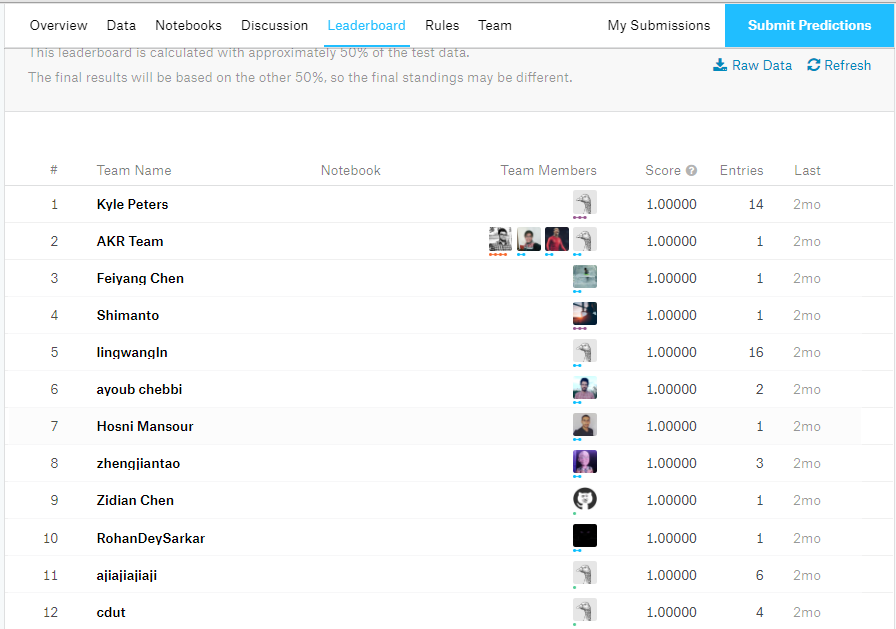
查看排名的步骤



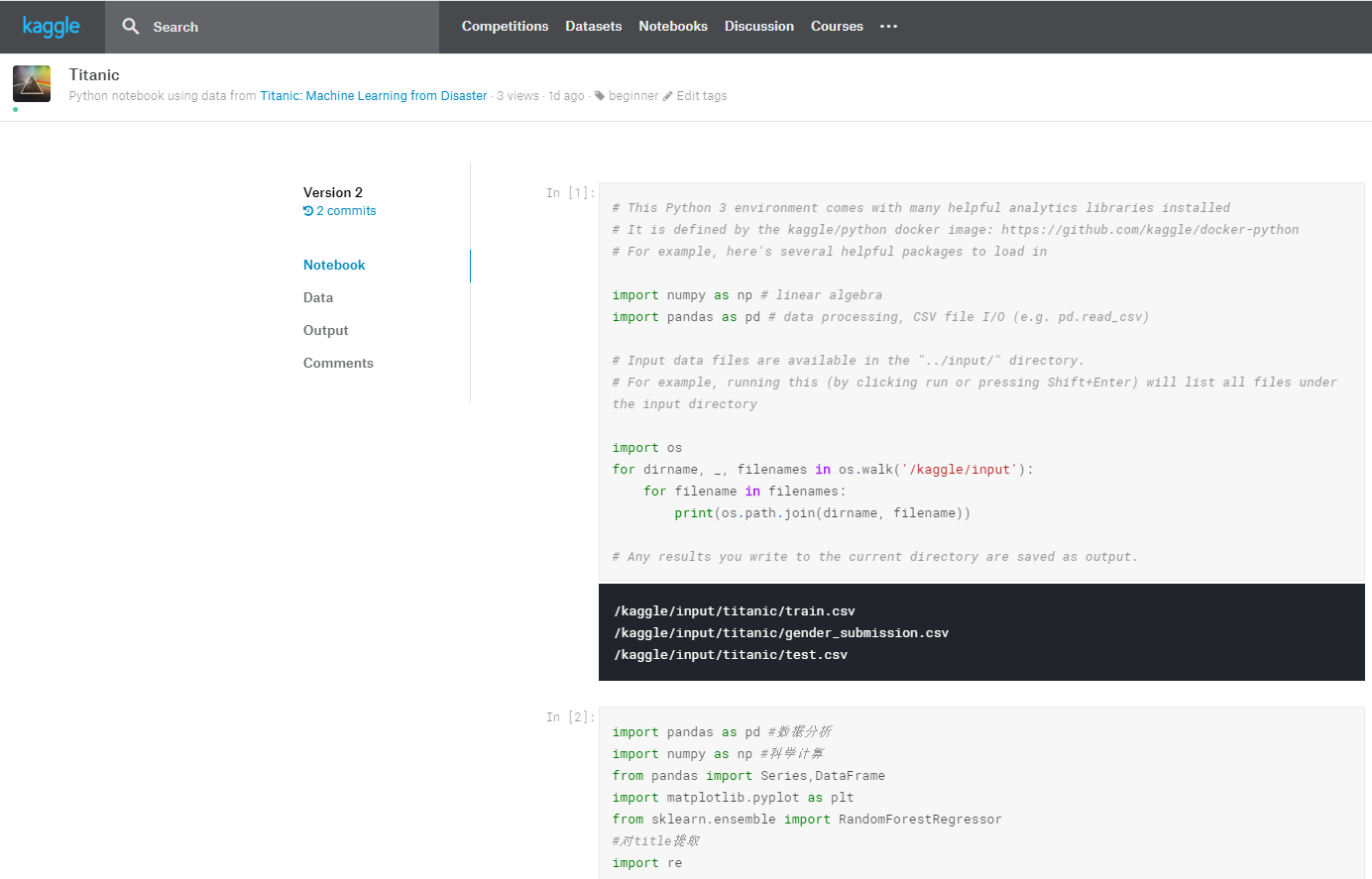
看到了自己的排名是843/13751（当然，这是后面模型微调后提交的最好成绩）



0.79在一万三千人中排两千名，0.808在一万三千人中排八百名，看来这个分段大家咬的很紧呀。



看了看排最前面的人准确率居然是百分百，感觉就是把测试集答案给交上去了，正常情况也就是80%+的准确率，正规比赛应该不会出现这种情况。

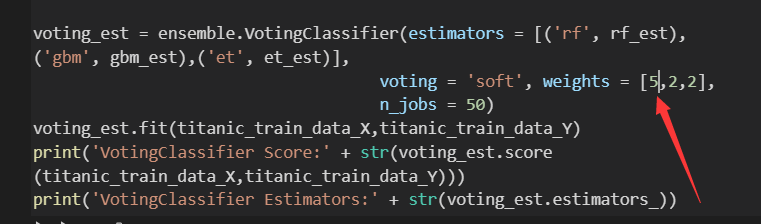


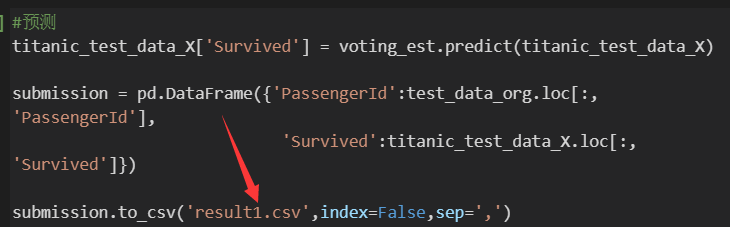
看到了kaggle提供的在线notebook，还是挺不错的，为大家学习大数据竞赛提供了良好的条件。

**6.微调模型：**

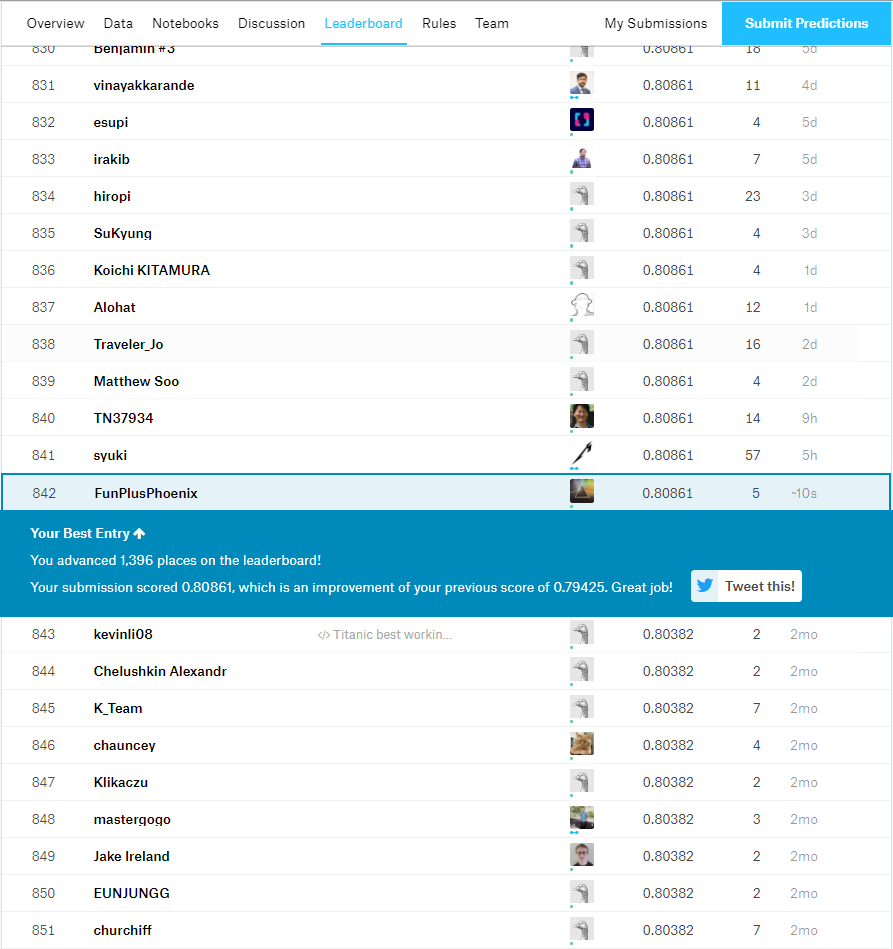
**准备通过调整不同基础模型的权重来寻找最佳权重：**

**（1）修改voting权重为5 2 2 输出result1.csv并提交kaggle**



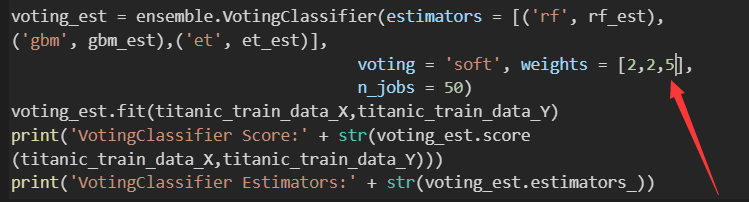


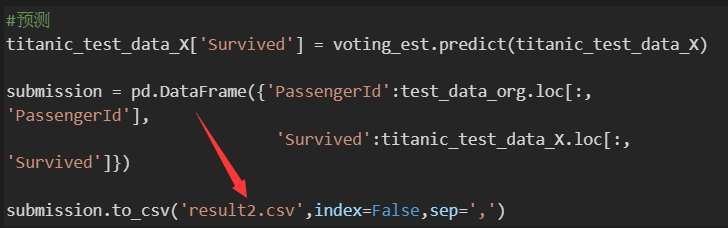
提交到kaggle查看排名

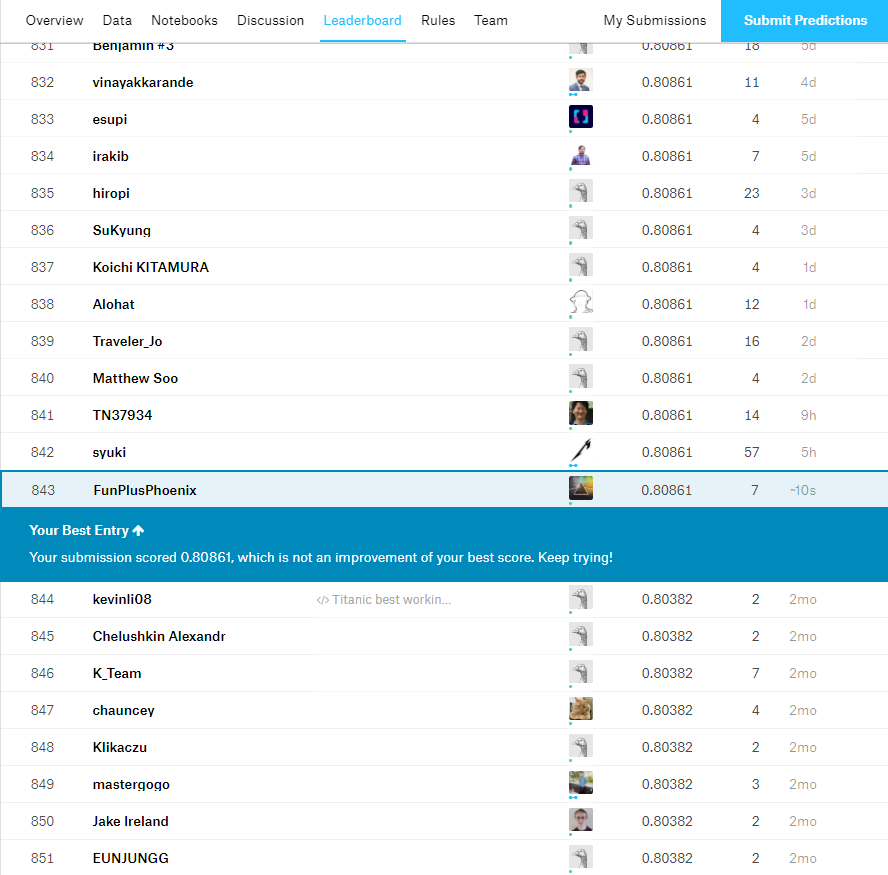


得分提高了0.014，但是名次提高了1396名，真的是不可思议，提高一分干掉千人

**修改voting权重为2 2 5 输出result2.csv并提交kaggle**

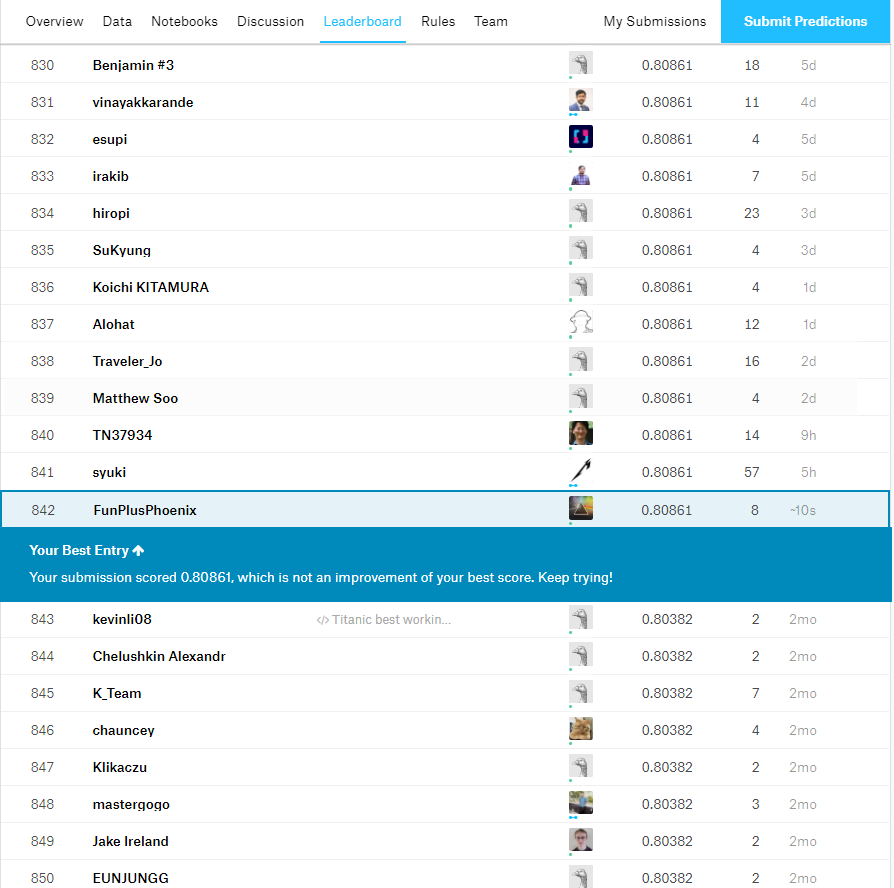






名次没什么变化，得分还是原来的样子。

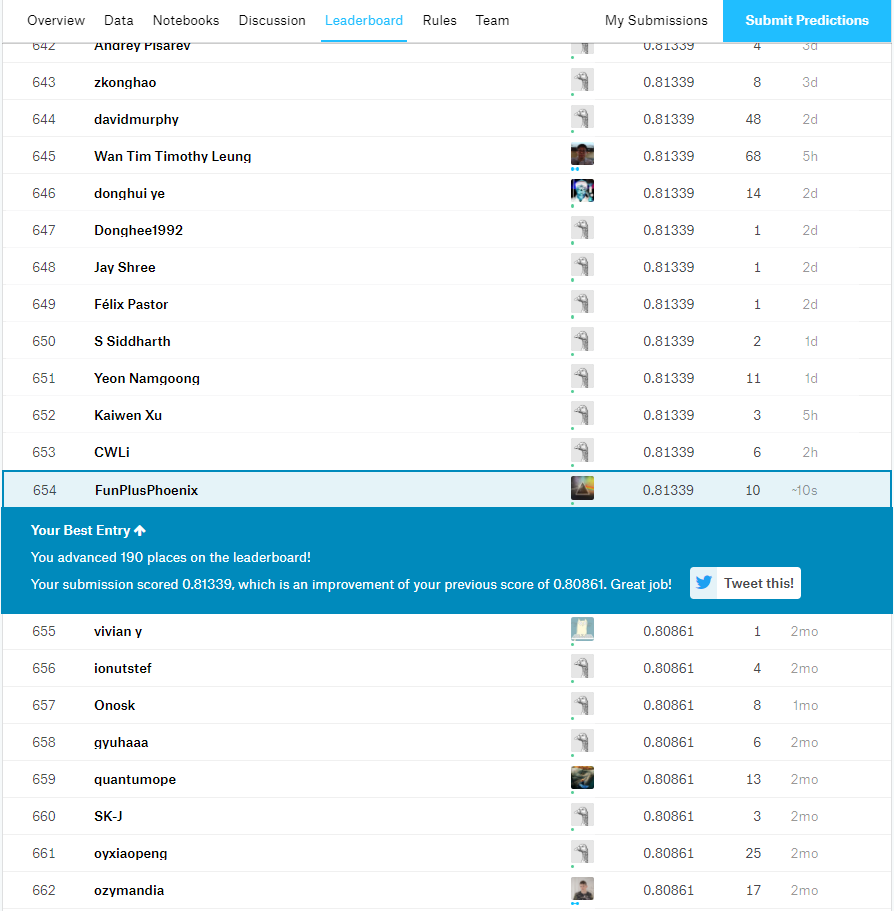
**修改voting权重为0 4 4 输出result3.csv并提交kaggle**



**没有变化，提升困难**

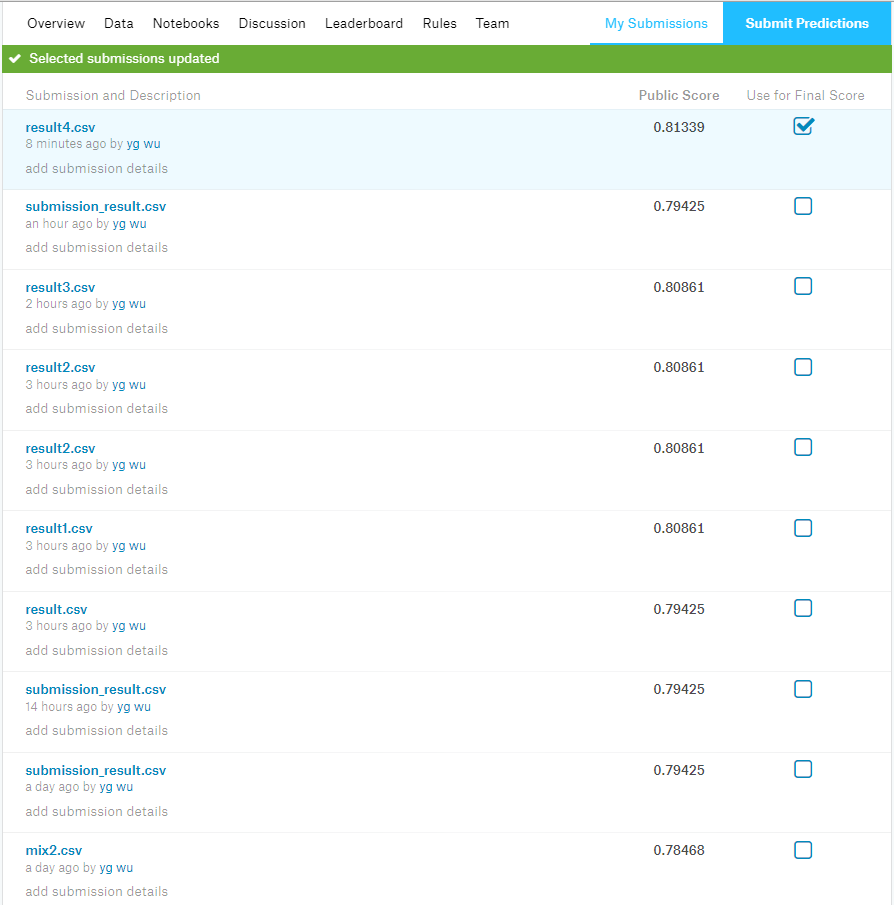
**修改voting权重为4 0 4 输出result3.csv并提交kaggle**

（因为我发现随机森林和极限树比较给力，于是就多给一些权重，刚好我学号尾数也是404）



名次又上升了190名到了654名，百尺竿头更进一步，great！感觉非常棒，那么模型微调就到此结束了。

**我的得分提升过程：**



**还有一个问题就是kaggle提交预测结果.csv需要上外网**

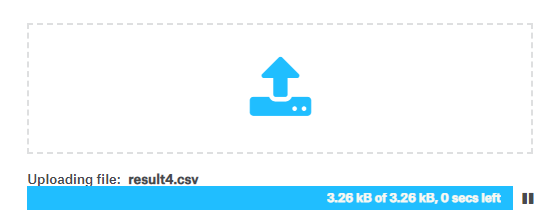
如果没有上外网：



上了外网后：



然后：



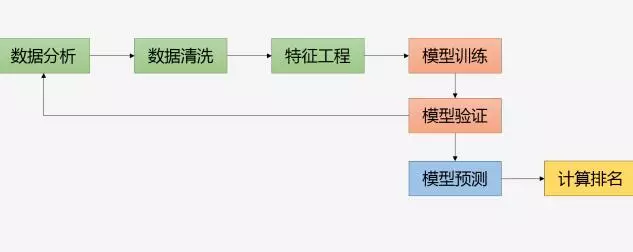
等到处理完成



点击提交



**到这里，我的第一次kaggle练手项目就完成了**，学到了很多东西，再回顾一下一般的流程就是



其中特征工程的部分内容感觉还需要以后的学习再接再厉，大数据竞赛还是不容易的，如果能多人组队分工的话也挺好的，希望大家再以后的kaggle竞赛中越战越勇。