数据挖掘比赛入门

1. 实验环境熟悉

培训目标：

（1） 熟悉 python 安装及使用。

（2） 熟悉 python package 的安装及使用。

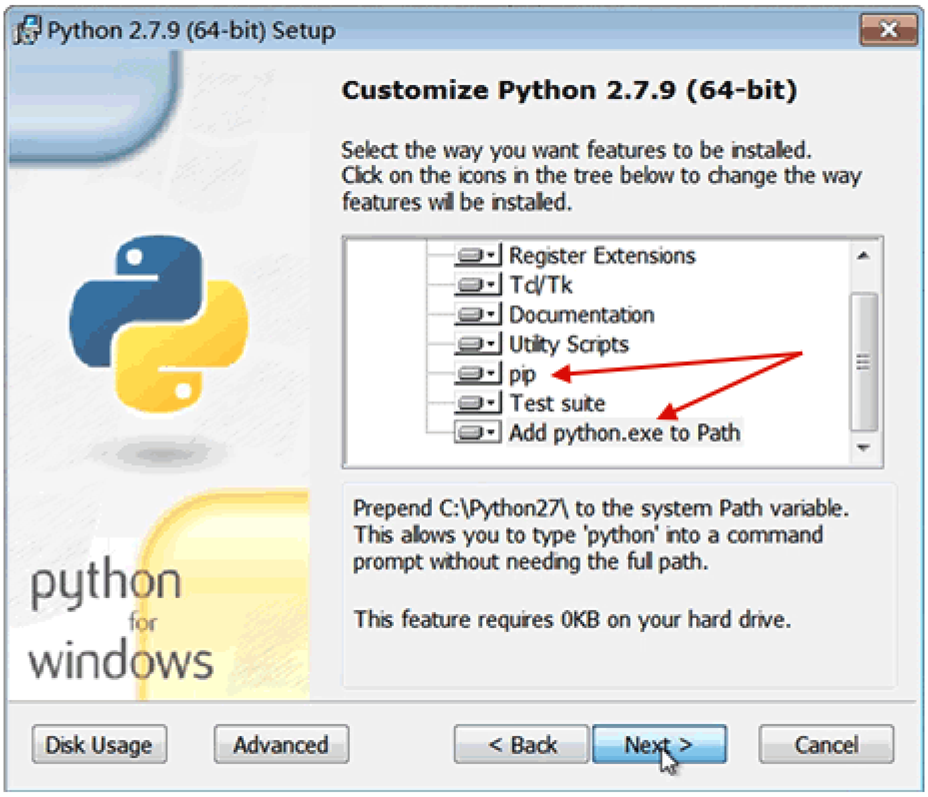
（3） 深入了解 pandas、numpy 等数据处理包的使用。

（4） 简单了解 xgboost、sklearn、keras 的安装及使用。

培训资料：

a. Python 安装教程

首先，从 Python 的官方网站 python.org 下载最新的 2.7 版本，网速慢的同学请移步国内镜像。然后，运行下载的 MSI 安装包，在选择安装组件的一步时，勾上所有的组件：



Python 2.7.9 (64-bit) Setup

Customize Python 2.7.9 (64-bit)

Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.

- Register Extensions
- Tcl/Tk
- Documentation
- Utility Scripts
- pip
- Test suite
- Add python.exe to Path

Prepend C:\Python27\ to the system Path variable. This allows you to type 'python' into a command prompt without needing the full path.

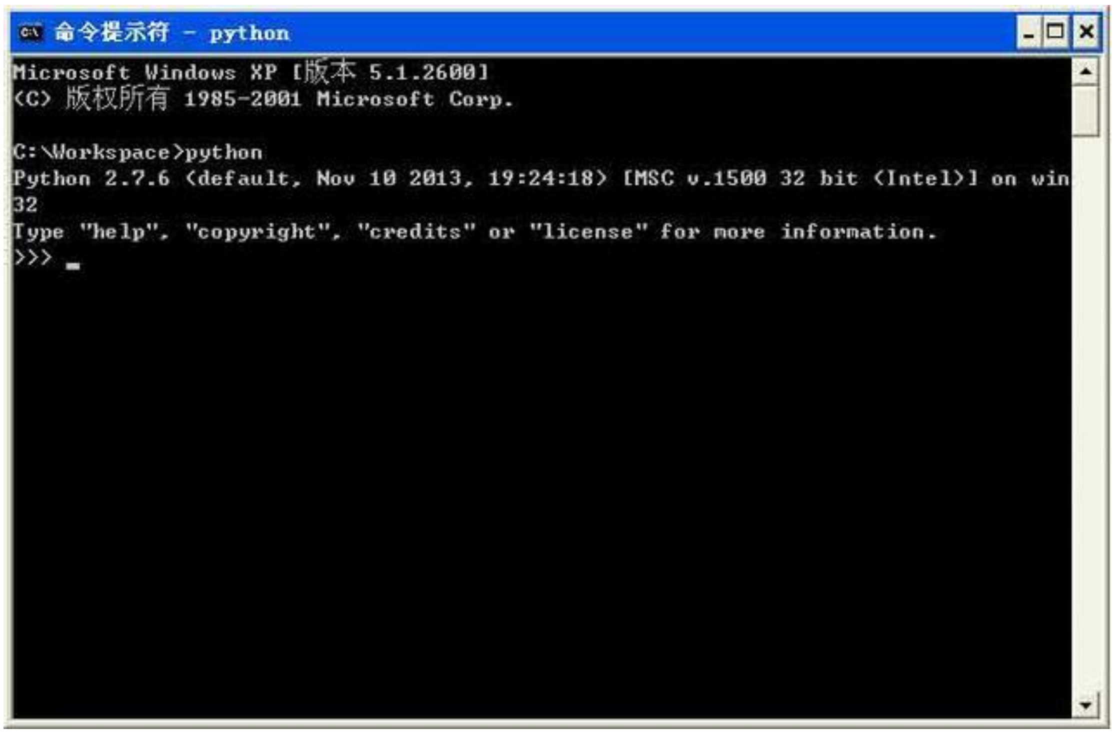
This feature requires 0KB on your hard drive.

特别要注意选上 `pip` 和 `Add python.exe to Path`，然后一路点“Next”即可完成安装。

安装成功后，默认会安装到 C:\Python27 目录下，然后打开命令提示符窗

口，敲入 python 后，会出现两种情况：

情况一：



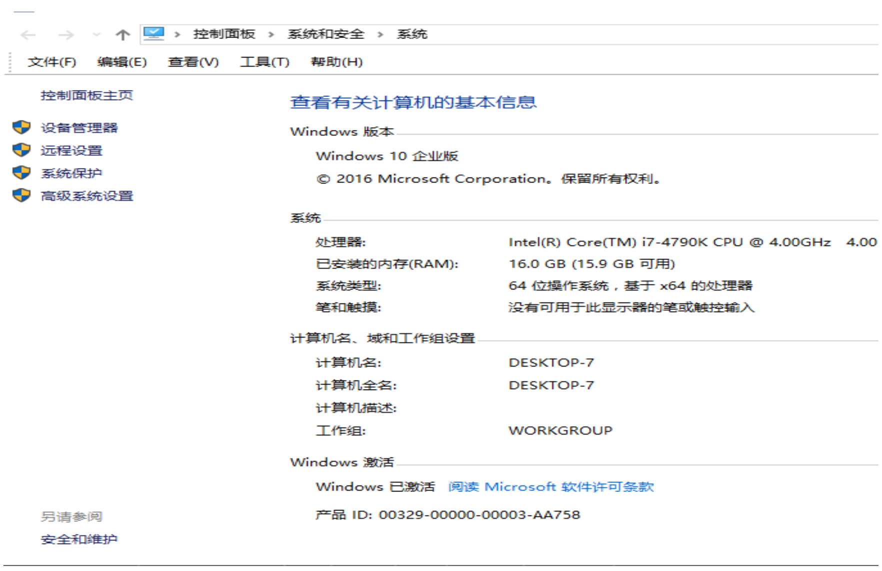
```
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Workspace>python
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

出现上述画面说明安装成功了。

情况二：

‘python’不是内部或外部命令，也不是可运行的程序或批处理文件。

出现此种情况后，说明 python 路径没有加入环境变量中。在我的电脑右键属性后出现下述画面：



查看有关计算机的基本信息

Windows 版本

Windows 10 企业版

© 2016 Microsoft Corporation。保留所有权利。

系统

处理器:	Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz 4.00
已安装的内存(RAM):	16.0 GB (15.9 GB 可用)
系统类型:	64 位操作系统，基于 x64 的处理器
笔和触摸:	没有可用于此显示器的笔或触控输入

计算机名、域和工作组设置

计算机名:	DESKTOP-7
计算机全名:	DESKTOP-7
计算机描述:	
工作组:	WORKGROUP

Windows 激活

Windows 已激活 阅读 Microsoft 软件许可条款

产品 ID: 00329-00000-00003-AA758

点击高级系统设置->环境变量：

在系统变量里的 Path 路径里（如果没有则新建 Path 路径）添加 Python 的安装路径。

完成上述操作后再在命令提示符中输入 Python 检查安装是否成功。

b. Pandas 及 numpy 的安装简单使用

windows 安装 pip 后可以直接利用 pip 来安装所需要的 package。如果没有安装 pip，可以按照下面这个链接安装 pip：

http://pip-cn.readthedocs.io/en/latest/installing.html

pip install --user numpy scipy pandas

键入上述命令后系统会帮忙自动下载所需要的的 packge 和依赖 package，然后还会自动给你安装上去。

如果发现上述下载源的速度比较慢，推荐使用国内的源进行下载：

pip install matplotlib -i http://pypi.douban.com/simple --trusted-host pypi.douban.com

pip install numpy -i http://pypi.douban.com/simple --trusted-host pypi.douban.com

pip install pandas -i http://pypi.douban.com/simple --trusted-host pypi.douban.com

pip install seaborn scipy -i http://pypi.douban.com/simple --trusted-host pypi.douban.com

在 pandas 的使用上，推荐一个教程 10 分钟学会 pandas：

http://pandas.pydata.org/pandas-docs/stable/10min.html

在 numpy 的使用上，推荐一个官方的学习文档：

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Pandas 和 numpy 是处理数据的利器，在做好数据挖掘的同时必须要掌握这两个利器。

c. Xgboost 及 Sklearn 的安装及使用

在做以下培训要求之前，请确保已了解过 python，知道下列环境中各个安装包的主要用途。

对于从未接触过 python 及数据挖掘相关的小白，推荐 Python2 教程：

http://www.runoob.com/python/python-tutorial.html

Python3 教程：http://www.runoob.com/python3/python3-tutorial.html

两个版本的 python 内容差别不是特别大，语法规则有差异；培训要求中的环境是 python2 的环境。所以，可以先按 python2 去学。Python3 适当了解。

Python 初学者，重点关注，字符串(str)，列表(list)，元组(tuple),字典(dict)。

简要介绍下，环境中各包的作用。

numpy, scipy 是线性带数处理包，大型矩阵等相关运算(详细介绍百度 or google)

pandas, sklearn,是对格式化数据进行特征工程时的强大工具，初期学习，多数主要依靠这两个包进行特征工程。

matplotlib 绘图可视化，辅助特征工程。

xgboost, 一个牛叉的模型分类器，后续各种分类算法中会学到。

keras 深度学习框架，主要是各种神经网络的工具包。由于 keras 是基于 tensorflow 实现的，windows 下基于 python2 不支持 tensorflow，win 下 python2 会安装失败。 所以对于环境搭配，建议转战 linux 系统(ubuntu)。

genism 自然语言处理相关，word2vec，doc2vec 之类的。

Ubuntu 本地虚拟机安装：

Vmware 下载：

https://my.vmware.com/web/vmware/details?downloadGroup=WKST-1257-WIN&productId=524&rPId=17067 (激活码 *5A02H-AU243-TZJ49-GTC7K-3C61N* 或百度)

Vmware 安装 ubuntu: 先下载 ubuntu(64-bit), 自行下载系统

安装教程：http://blog.csdn.net/u013142781/article/details/50529030

Ubuntu 下(ubuntu 自带 python2.7 环境)，各依赖包的安装：

[1] Ubuntu 安装 numpy, scipy, matplotlib

http://blog.csdn.net/shomy_liu/article/details/48543449

[2]安装 pandas

https://morvanzhou.github.io/tutorials/data-manipulation/np-pd/1-2-install/

[3] Sklearn 安装

https://morvanzhou.github.io/tutorials/machine-learning/sklearn/1-2-install/

[4] Linux 下安装 xgBoost

http://blog.csdn.net/mmc2015/article/details/44623301

[5] Keras 安装和配置指南(Linux)

https://keras-cn.readthedocs.io/en/latest/for_beginners/keras_linux/

[6]genism 安装

http://www.linuxdiyf.com/linux/18639.html

培训要求：

（1） 在自己电脑上安装 python 环境，包含 numpy、matplotlib、scipy、pandas、sklearn、xgboost、keras、gensim 环境。

2. 数据挖掘入门

培训目标：

（1） 对数据挖掘概念熟悉掌握

（2） 对机器学习算法中的树模型、线性模型使用熟悉掌握

（3） 对机器学习算法的内容深入了解。

（4） 对整套数据挖掘流程清晰掌握。

对于未接触过 kaggle，或者刚接触有困难的，推荐一个 kaggle 入门：

https://www.zhihu.com/question/23987009

目标：至少知道 kaggle 中每个 title 下各模块的作用，比如说，查看 kernels,排名等。 去习惯浏览英文网站。

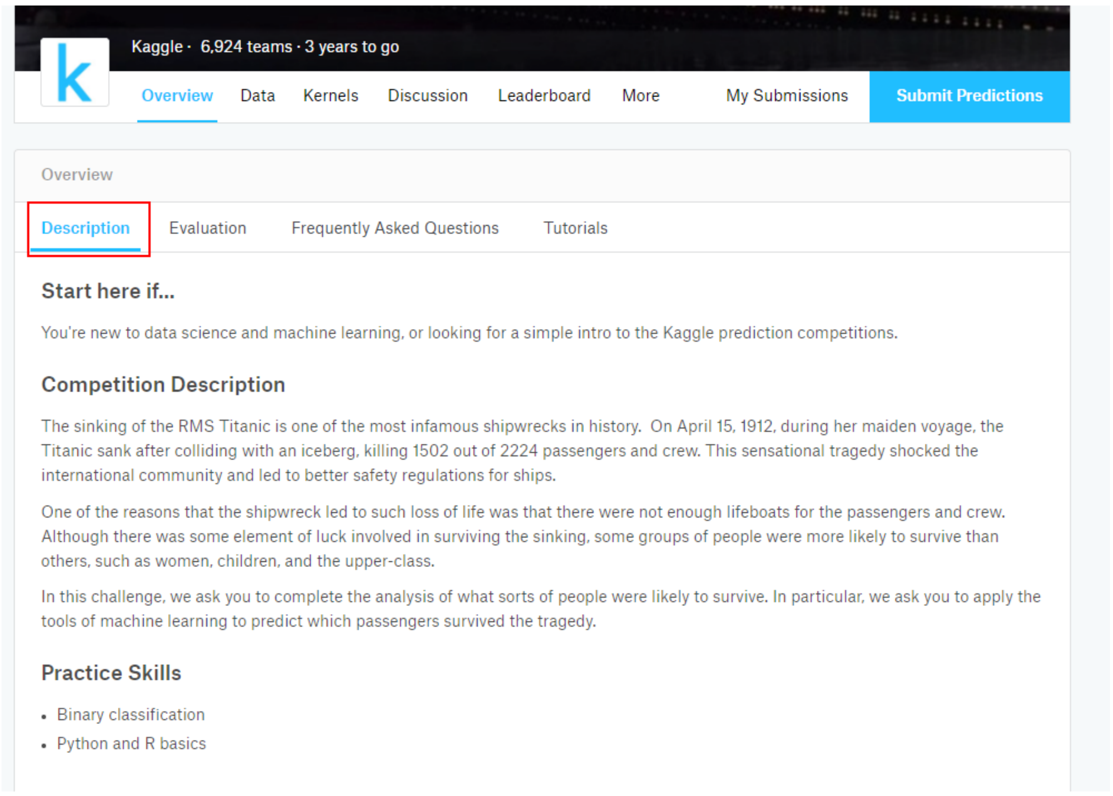
培训资料：

（1） https://www.kaggle.com/arthurtok/titanic/introduction-to-ensembling-stacking-in-python 数据挖掘 kaggle 流程

数据挖掘 kaggle 流程介绍

a. 题目 https://www.kaggle.com/c/titanic

总体描述



Start here if...

You're new to data science and machine learning, or looking for a simple intro to the Kaggle prediction competitions.

Competition Description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

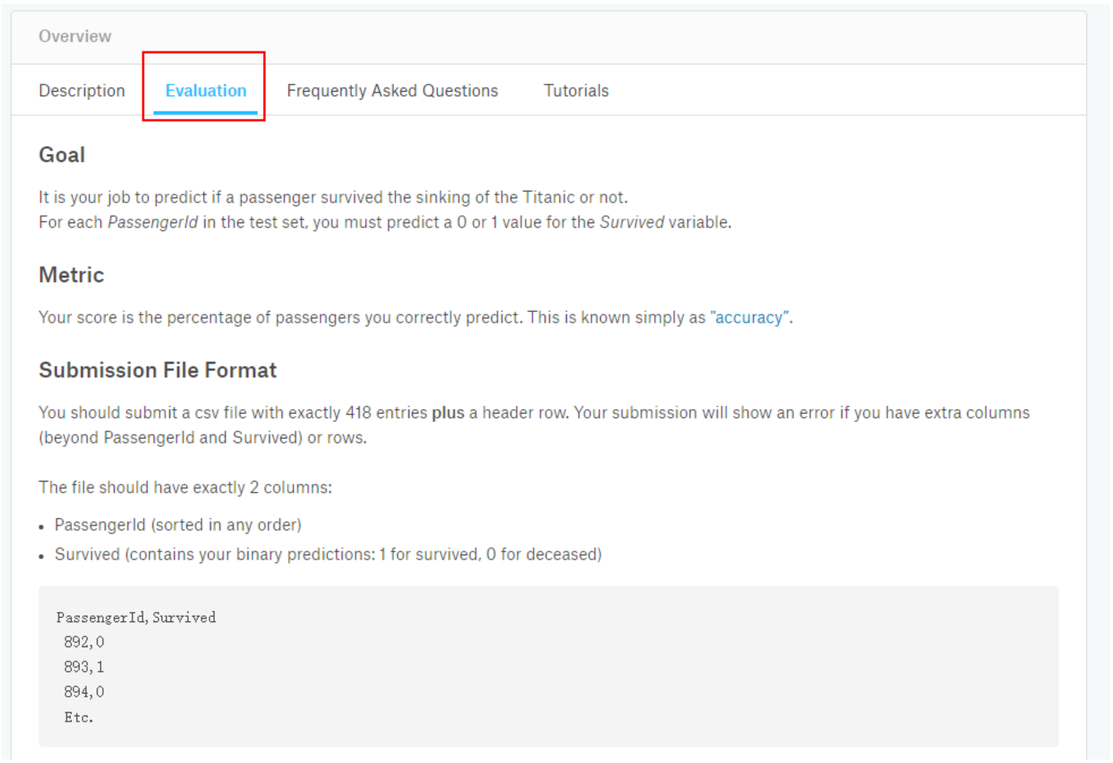
One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

Practice Skills

- Binary classification
- Python and R basics

评价标准和提交样例



Goal

It is your job to predict if a passenger survived the sinking of the Titanic or not.
For each *PassengerId* in the test set, you must predict a 0 or 1 value for the *Survived* variable.

Metric

Your score is the percentage of passengers you correctly predict. This is known simply as "accuracy".

Submission File Format

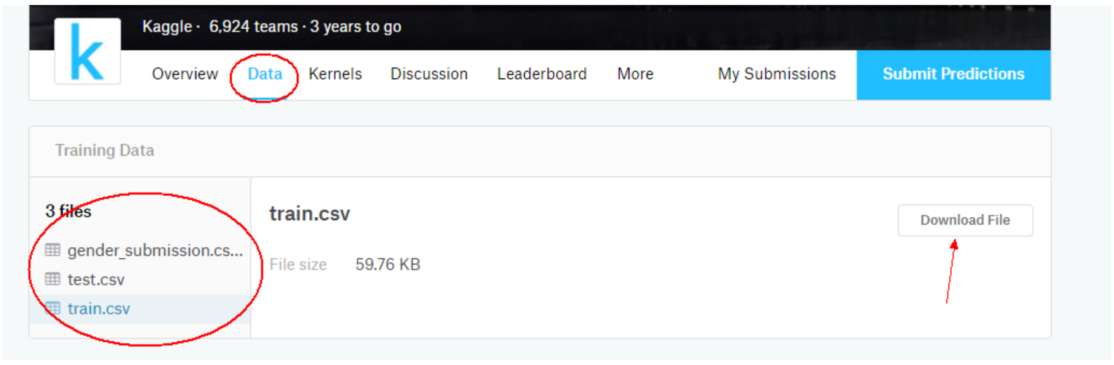
You should submit a csv file with exactly 418 entries **plus** a header row. Your submission will show an error if you have extra columns (beyond PassengerId and Survived) or rows.

The file should have exactly 2 columns:

- PassengerId (sorted in any order)
- Survived (contains your binary predictions: 1 for survived, 0 for deceased)

```
PassengerId,Survived
 892,0
 893,1
 894,0
 Etc.
```

数据下载



b. 初探数据

```
%matplotlib inline
import numpy as np
import pandas as pd
import re as re

train = pd.read_csv('../input/train.csv', header = 0, dtype={'Age': np.float64})
test  = pd.read_csv('../input/test.csv' , header = 0, dtype={'Age': np.float64})
full_data = [train, test]

print (train.info())
```

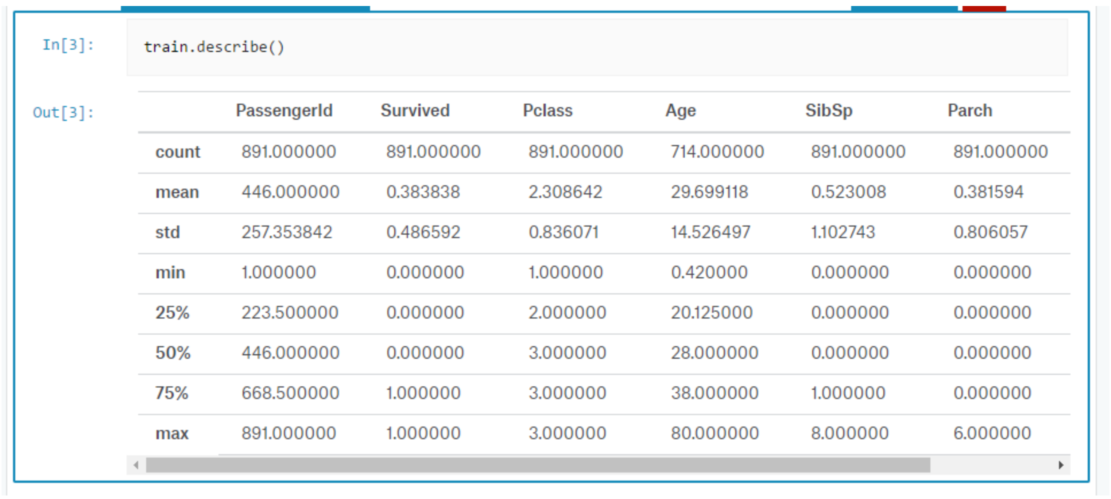
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
None
```

数据对应的含义

- 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
- Survived => 是否获救 => label

更详细的



```
In[3]: train.describe()
```

Out[3]:

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

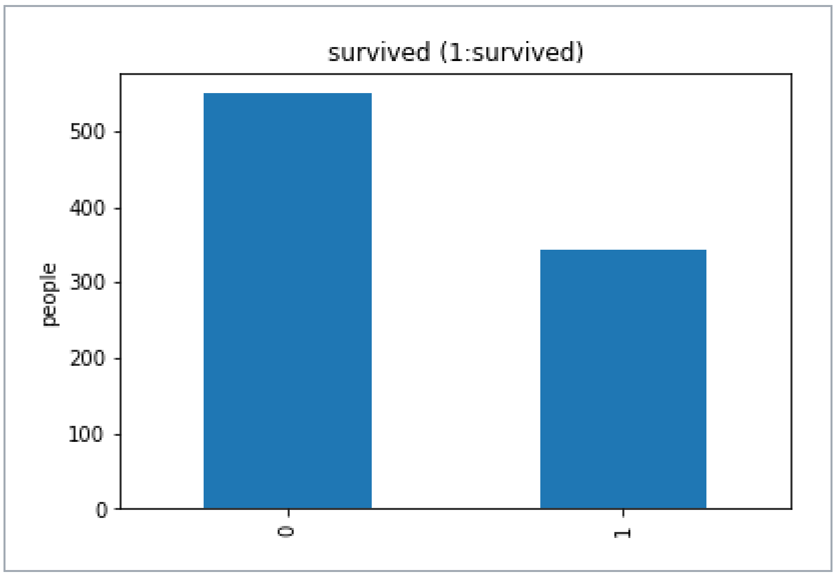
mean 字段告诉我们，大概 0.383838 的人最后获救了，2/3 等舱的人数比 1 等舱要多，平均乘客年龄大概是 29.7 岁等

c. 可视化探索

获救情况

```
train.Survived.value_counts().plot(kind='bar')# 柱状图
plt.title(u"survived (1:survived)") # 标题
plt.ylabel(u"people")
```

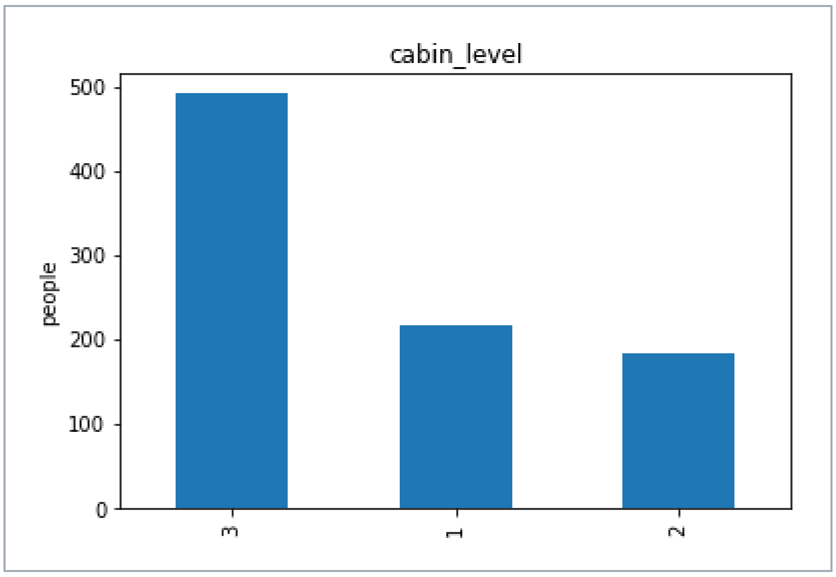
```
<matplotlib.text.Text at 0x7f236a0ac438>
```



乘客等级分布

```
train.Pclass.value_counts().plot(kind="bar")
plt.ylabel(u"people")
plt.title(u"cabin_level")
```

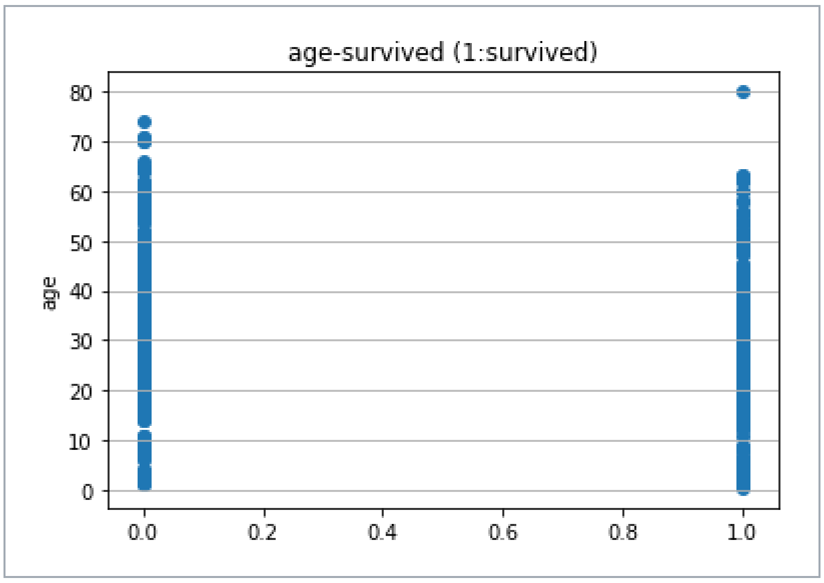
```
<matplotlib.text.Text at 0x7f2366c96dd8>
```



按年龄看获救分布

```
plt.scatter(train.Survived, train.Age)
plt.ylabel(u"age")                         # 设定纵坐标名称
plt.grid(b=True, which='major', axis='y')
plt.title(u"age-survived (1:survived)")
```

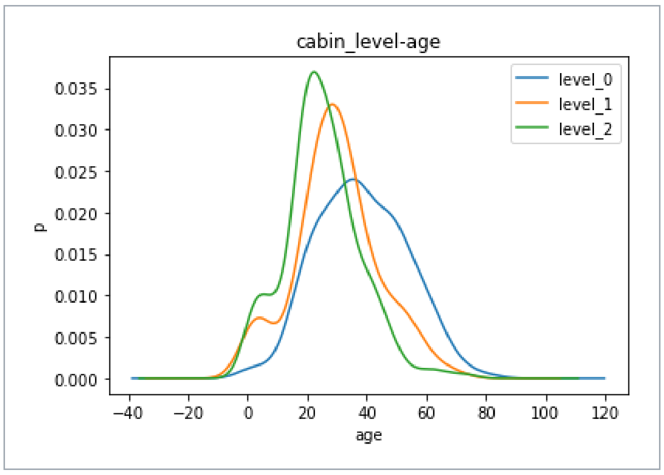
```
<matplotlib.text.Text at 0x7f2366bb8c18>
```



各等级的乘客年龄分布

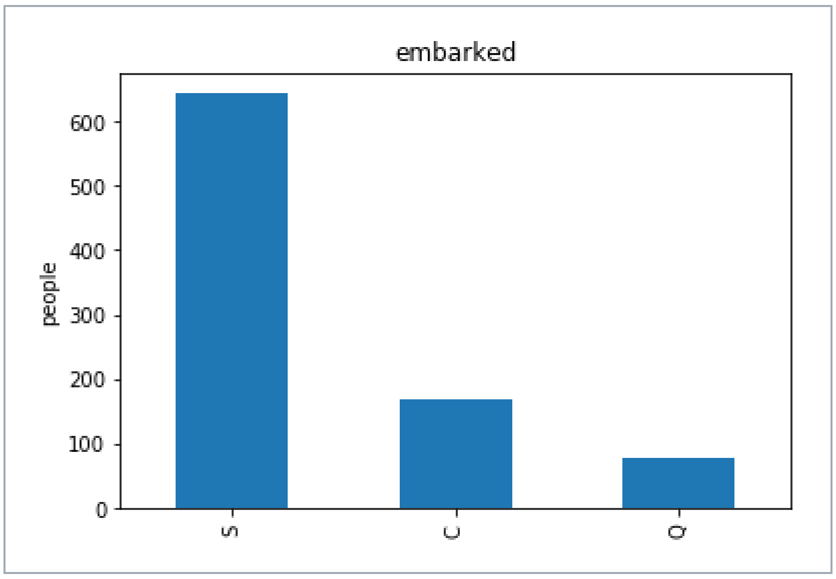
```
train.Age[train.Pclass == 1].plot(kind='kde')
train.Age[train.Pclass == 2].plot(kind='kde')
train.Age[train.Pclass == 3].plot(kind='kde')
plt.xlabel(u"age")# plots an axis lable
plt.ylabel(u"p")
plt.title(u"cabin_level-age")
plt.legend((u'level_0', u'level_1',u'level_2'),loc='best') # sets our legend for our graph
```

```
<matplotlib.legend.Legend at 0x7f2366b11be0>
```



各登船口岸上船人数

```
train.Embarked.value_counts().plot(kind='bar')
plt.title(u"embarked")
plt.ylabel(u"people")
plt.show()
```



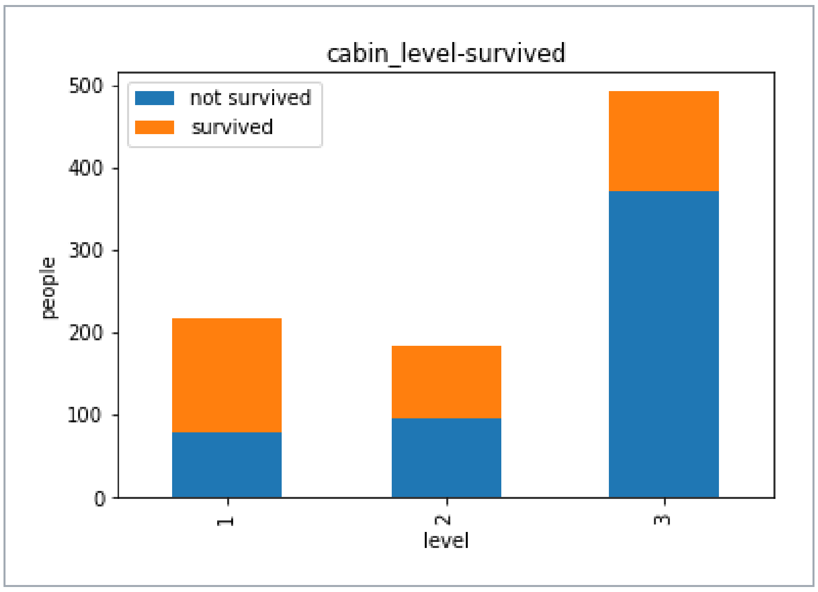
此时的一些想法

- 不同舱位/乘客等级可能和财富/地位有关系，最后获救概率可能会不一样
- 年龄对获救概率也一定是有影响的
- 和登船港口是不是有关系呢？也许登船港口不同，人的出身地位不同？

继续验证猜想

各乘客等级获救情况

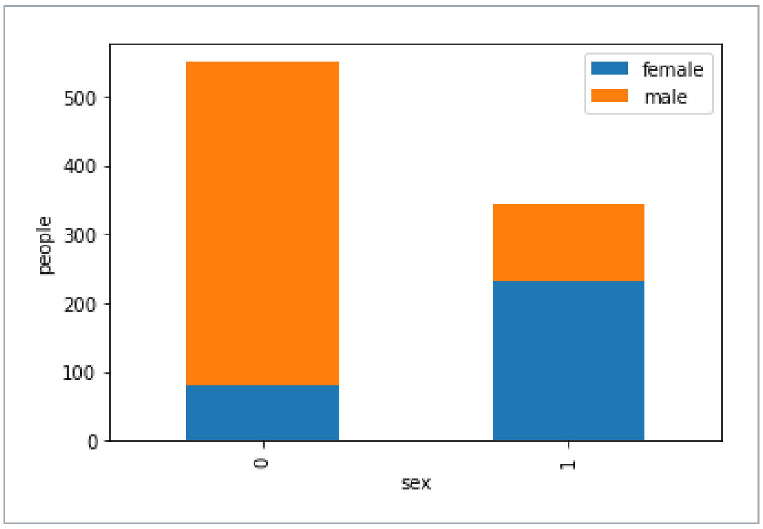
```
Survived_0 = train.Pclass[train.Survived == 0].value_counts()
Survived_1 = train.Pclass[train.Survived == 1].value_counts()
df=pd.DataFrame({u'survived':Survived_1, u'not survived':Survived_0})
df.plot(kind='bar', stacked=True)
plt.title(u"cabin_level-survived")
plt.xlabel(u"level")
plt.ylabel(u"people")
plt.show()
```



明显等级为 1 的乘客，获救的概率高很多

各性别获救情况

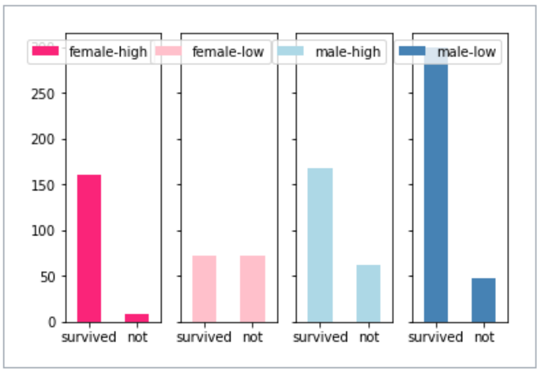
```
Survived_m = train.Survived[train.Sex == 'male'].value_counts()
Survived_f = train.Survived[train.Sex == 'female'].value_counts()
df=pd.DataFrame({u'male':Survived_m, u'female':Survived_f})
df.plot(kind='bar', stacked=True)
plt.xlabel(u"sex")
plt.ylabel(u"people")
plt.show()
```



女性获救率更高

各种舱级别情况下各性别的获救情况

```
fig=plt.figure()

ax1=fig.add_subplot(141)
train.Survived[train.Sex == 'female'][train.Pclass != 3].value_counts().plot(kind='bar', label="female hi
ax1.set_xticklabels([u"survived", u"not"], rotation=0)
ax1.legend([u"female-high"], loc='best')

ax2=fig.add_subplot(142, sharey=ax1)
train.Survived[train.Sex == 'female'][train.Pclass == 3].value_counts().plot(kind='bar', label='female, l
ax2.set_xticklabels([u"survived", u"not"], rotation=0)
plt.legend([u"female-low"], loc='best')

ax3=fig.add_subplot(143, sharey=ax1)
train.Survived[train.Sex == 'male'][train.Pclass != 3].value_counts().plot(kind='bar', label='male, high
ax3.set_xticklabels([u"survived", u"not"], rotation=0)
plt.legend([u"male-high"], loc='best')

ax4=fig.add_subplot(144, sharey=ax1)
train.Survived[train.Sex == 'male'][train.Pclass == 3].value_counts().plot(kind='bar', label='male low cl
ax4.set_xticklabels([u"survived", u"not"], rotation=0)
plt.legend([u"male-low"], loc='best')

plt.show()
```



验证了之前的判断

d. 特征工程

3. SibSp and Parch

With the number of siblings/spouse and the number of children/parents we can create new feature called Family Size.

In[28]:
```
for dataset in full_data:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
print (train[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False).mean())
```

Out[28]:
```
   FamilySize  Survived
0           1  0.303538
1           2  0.552795
2           3  0.578431
3           4  0.724138
4           5  0.200000
5           6  0.136364
6           7  0.333333
7           8  0.000000
8          11  0.000000
```

it seems has a good effect on our prediction but let's go further and categorize people to check whether they are alone in this ship or not.

In[29]:
```
for dataset in full_data:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1
print (train[['IsAlone', 'Survived']].groupby(['IsAlone'], as_index=False).mean())
```

Out[29]:
```
   IsAlone  Survived
0        0  0.505650
1        1  0.303538
```

good! the impact is considerable.

4. Embarked

the embarked feature has some missing value. and we try to fill those with the most occurred value ('S').

```
for dataset in full_data:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')
print (train[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean())
```

```
  Embarked  Survived
0        C  0.553571
1        Q  0.389610
2        S  0.339009
```

5. Fare

Fare also has some missing value and we will replace it with the median. then we categorize it into 4 ranges.

```
for dataset in full_data:
    dataset['Fare'] = dataset['Fare'].fillna(train['Fare'].median())
train['CategoricalFare'] = pd.qcut(train['Fare'], 4)
print (train[['CategoricalFare', 'Survived']].groupby(['CategoricalFare'], as_index=False).mean())
```

```
  CategoricalFare  Survived
0       [0, 7.91]  0.197309
1  (7.91, 14.454]  0.303571
2    (14.454, 31]  0.454955
3   (31, 512.329]  0.581081
```

6. Age

we have plenty of missing values in this feature. # generate random numbers between (mean - std) and (mean + std). then we categorize age into 5 range.

```
for dataset in full_data:
    age_avg        = dataset['Age'].mean()
    age_std        = dataset['Age'].std()
    age_null_count = dataset['Age'].isnull().sum()

    age_null_random_list = np.random.randint(age_avg - age_std, age_avg + age_std, size=age_null_count)
    dataset['Age'][np.isnan(dataset['Age'])] = age_null_random_list
    dataset['Age'] = dataset['Age'].astype(int)

train['CategoricalAge'] = pd.cut(train['Age'], 5)

print (train[['CategoricalAge', 'Survived']].groupby(['CategoricalAge'], as_index=False).mean())
```

```
  CategoricalAge  Survived
0  (-0.08, 16]  0.521739
1     (16, 32]  0.356044
2     (32, 48]  0.369295
3     (48, 64]  0.434783
4     (64, 80]  0.090909
```

7. Name

inside this feature we can find the title of people.

```
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    # If the title exists, extract and return it.
    if title_search:
        return title_search.group(1)
    return ""

for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)

print(pd.crosstab(train['Title'], train['Sex']))
```

```
Sex       female  male
Title
Capt           0     1
Col            0     2
Countess       1     0
Don            0     1
Dr             1     6
Jonkheer       0     1
Lady           1     0
Major          0     2
Master         0    40
Miss         182     0
Mlle           2     0
Mme            1     0
Mr             0   517
Mrs          125     0
Ms             1     0
Rev            0     6
Sir            0     1
```

```
for dataset in full_data:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col',\
    'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')

    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

print (train[['Title', 'Survived']].groupby(['Title'], as_index=False).mean())
```

```
    Title  Survived
0  Master  0.575000
1    Miss  0.702703
2      Mr  0.156673
3     Mrs  0.793651
4    Rare  0.347826
```

```
for dataset in full_data:
    # Mapping Sex
    dataset['Sex'] = dataset['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

    # Mapping titles
    title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

    # Mapping Embarked
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)

    # Mapping Fare
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare']                               = 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   = 2
    dataset.loc[ dataset['Fare'] > 31, 'Fare']                                  = 3
    dataset['Fare'] = dataset['Fare'].astype(int)

    # Mapping Age
    dataset.loc[ dataset['Age'] <= 16, 'Age']                          = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age']                           = 4

# Feature Selection
drop_elements = ['PassengerId', 'Name', 'Ticket', 'Cabin', 'SibSp',\
                 'Parch', 'FamilySize']
train = train.drop(drop_elements, axis = 1)
train = train.drop(['CategoricalAge', 'CategoricalFare'], axis = 1)

test  = test.drop(drop_elements, axis = 1)

print (train.head(10))

train = train.values
test  = test.values
```

```
   Survived  Pclass  Sex  Age  Fare  Embarked  IsAlone  Title
0         0       3    1    1     0         0        0      1
1         1       1    0    2     3         1        0      3
2         1       3    0    1     1         0        1      2
3         1       1    0    2     3         0        0      3
4         0       3    1    2     1         0        1      1
5         0       3    1    1     1         2        1      1
6         0       1    1    3     3         0        1      1
7         0       3    1    0     2         0        0      4
8         1       3    0    1     1         0        0      3
9         1       2    0    0     2         1        0      3
```

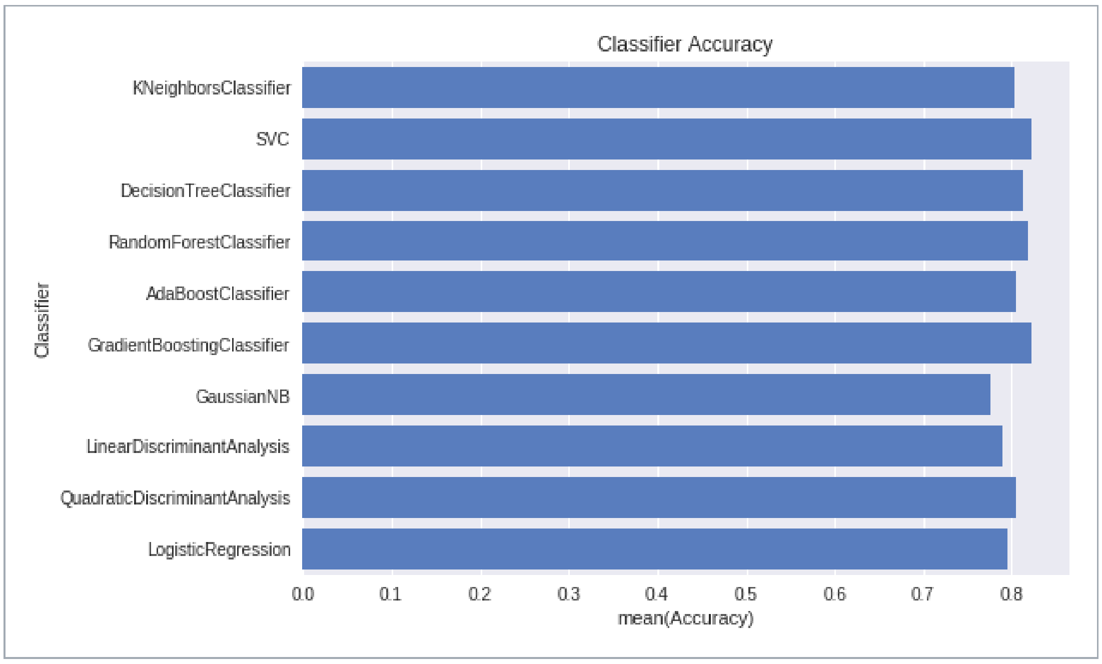
good! now we have a clean dataset and ready to predict. let's find which classifier works better on this dataset.

e. 分类模型训练

Classifier Comparison

```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.metrics import accuracy_score, log_loss
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression

classifiers = [
    KNeighborsClassifier(3),
    SVC(probability=True),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    GaussianNB(),
    LinearDiscriminantAnalysis(),
    QuadraticDiscriminantAnalysis(),
    LogisticRegression()]

log_cols = ["Classifier", "Accuracy"]
log      = pd.DataFrame(columns=log_cols)

sss = StratifiedShuffleSplit(n_splits=10, test_size=0.1, random_state=0)

X = train[0::, 1::]
y = train[0::, 0]

acc_dict = {}

for train_index, test_index in sss.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    for clf in classifiers:
        name = clf.__class__.__name__
        clf.fit(X_train, y_train)
        train_predictions = clf.predict(X_test)
        acc = accuracy_score(y_test, train_predictions)
        if name in acc_dict:
            acc_dict[name] += acc
        else:
            acc_dict[name] = acc

for clf in acc_dict:
    acc_dict[clf] = acc_dict[clf] / 10.0
    log_entry = pd.DataFrame([[clf, acc_dict[clf]]], columns=log_cols)
    log = log.append(log_entry)

plt.xlabel('Accuracy')
plt.title('Classifier Accuracy')

sns.set_color_codes("muted")
sns.barplot(x='Accuracy', y='Classifier', data=log, color="b")
```

模型结果比较



选择最优的单模型，预测得出结果

now we can use SVC classifier to predict our data.

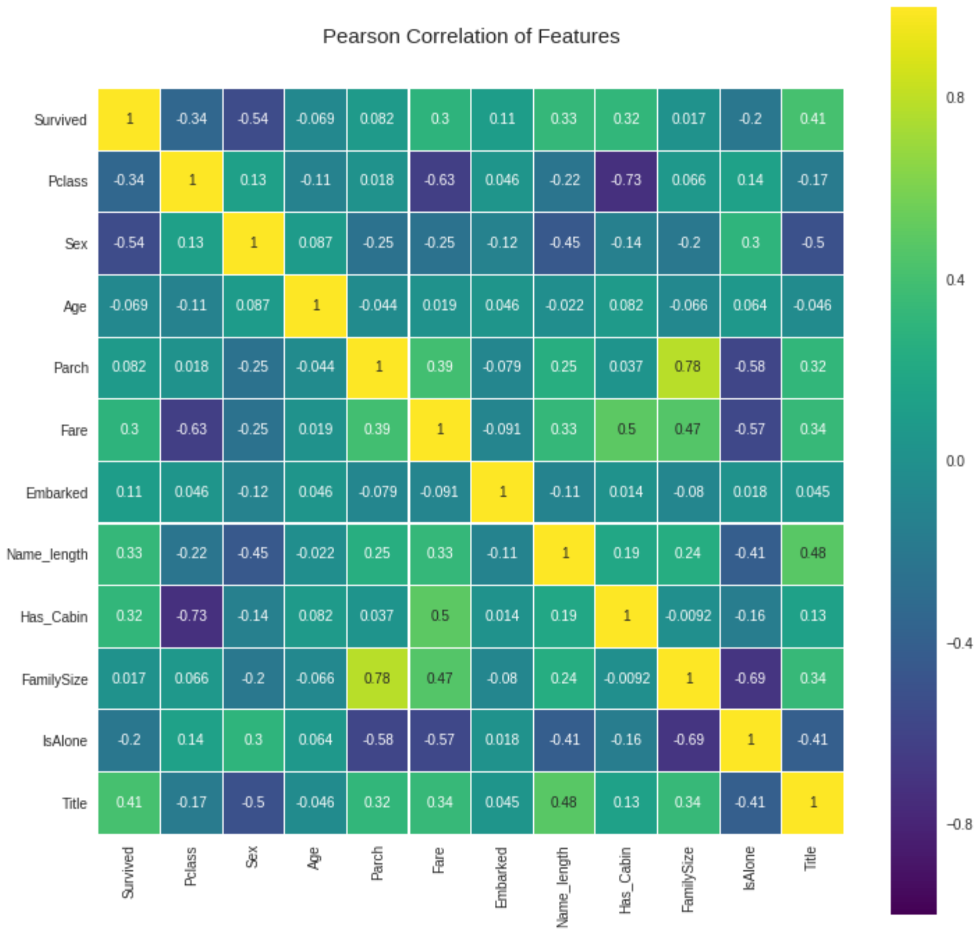
```
candidate_classifier = SVC()
candidate_classifier.fit(train[0::, 1::], train[0::, 0])
result = candidate_classifier.predict(test)
```

f.　模型融合

查看特征相关性

```
colormap = plt.cm.viridis
plt.figure(figsize=(12,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(train.astype(float).corr(),linewidths=0.1,vmax=1.0, square=True, cmap=colormap,
linecolor='white', annot=True)
```

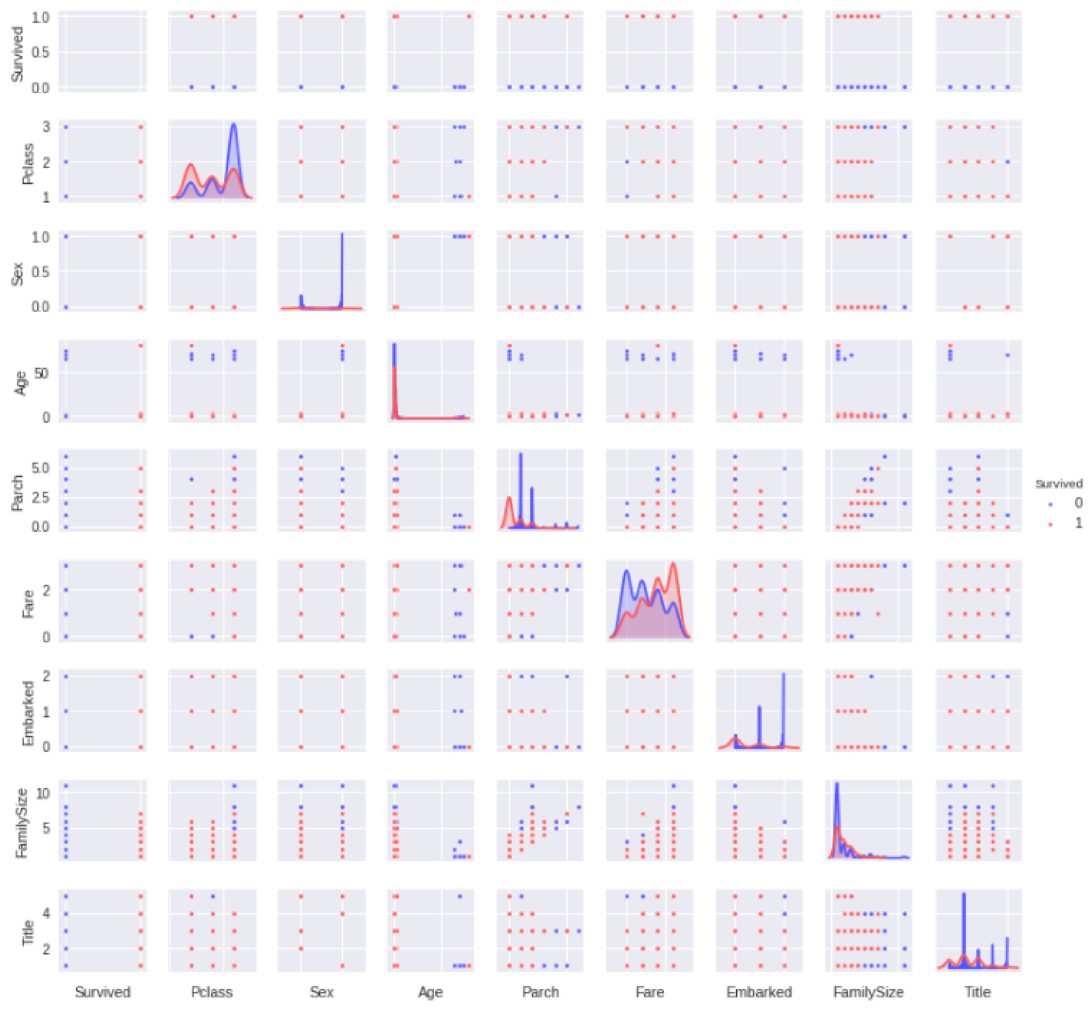
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f18e9058ac8>
```



```
g = sns.pairplot(train[[u'Survived', u'Pclass', u'Sex', u'Age', u'Parch', u'Fare', u'Embarked',
       u'FamilySize', u'Title']], hue='Survived', palette = 'seismic',size=1.2,diag_kind =
'kde',diag_kws=dict(shade=True),plot_kws=dict(s=10) )
g.set(xticklabels=[])
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/nonparametric/kde.py:494: RuntimeWarning:

invalid value encountered in true_divide

/opt/conda/lib/python3.6/site-packages/statsmodels/nonparametric/kdetools.py:34: RuntimeWarning:

invalid value encountered in double_scalars
```

```
<seaborn.axisgrid.PairGrid at 0x7f18e575e048>
```



对 sklearn 的模型进行封装，方便多次调用

```python
# Some useful parameters which will come in handy later on
ntrain = train.shape[0]
ntest = test.shape[0]
SEED = 0 # for reproducibility
NFOLDS = 5 # set folds for out-of-fold prediction
kf = KFold(ntrain, n_folds= NFOLDS, random_state=SEED)

# Class to extend the Sklearn classifier
class SklearnHelper(object):
    def __init__(self, clf, seed=0, params=None):
        params['random_state'] = seed
        self.clf = clf(**params)

    def train(self, x_train, y_train):
        self.clf.fit(x_train, y_train)

    def predict(self, x):
        return self.clf.predict(x)

    def fit(self,x,y):
        return self.clf.fit(x,y)

    def feature_importances(self,x,y):
        print(self.clf.fit(x,y).feature_importances_)

# Class to extend XGboost classifer
```

设置模型参数

```python
# Put in our parameters for said classifiers
# Random Forest parameters
rf_params = {
    'n_jobs': -1,
    'n_estimators': 500,
     'warm_start': True,
     #'max_features': 0.2,
    'max_depth': 6,
    'min_samples_leaf': 2,
    'max_features' : 'sqrt',
    'verbose': 0
}

# Extra Trees Parameters
et_params = {
    'n_jobs': -1,
    'n_estimators':500,
    #'max_features': 0.5,
    'max_depth': 8,
    'min_samples_leaf': 2,
    'verbose': 0
}

# AdaBoost parameters
ada_params = {
    'n_estimators': 500,
    'learning_rate' : 0.75
}

# Gradient Boosting parameters
gb_params = {
    'n_estimators': 500,
     #'max_features': 0.2,
    'max_depth': 5,
    'min_samples_leaf': 2,
    'verbose': 0
}

# Support Vector Classifier parameters
svc_params = {
    'kernel' : 'linear',
    'C' : 0.025
    }
```

实例化模型

```
# Create 5 objects that represent our 4 models
rf = SklearnHelper(clf=RandomForestClassifier, seed=SEED, params=rf_params)
et = SklearnHelper(clf=ExtraTreesClassifier, seed=SEED, params=et_params)
ada = SklearnHelper(clf=AdaBoostClassifier, seed=SEED, params=ada_params)
gb = SklearnHelper(clf=GradientBoostingClassifier, seed=SEED, params=gb_params)
svc = SklearnHelper(clf=SVC, seed=SEED, params=svc_params)
```

准备训练测试集

```
# Create Numpy arrays of train, test and target ( Survived) dataframes to feed into our models
y_train = train['Survived'].ravel()
train = train.drop(['Survived'], axis=1)
x_train = train.values # Creates an array of the train data
x_test = test.values # Creats an array of the test data
```

防止过拟合，划分训练测试集

```
def get_oof(clf, x_train, y_train, x_test):
    oof_train = np.zeros((ntrain,))
    oof_test = np.zeros((ntest,))
    oof_test_skf = np.empty((NFOLDS, ntest))

    for i, (train_index, test_index) in enumerate(kf):
        x_tr = x_train[train_index]
        y_tr = y_train[train_index]
        x_te = x_train[test_index]

        clf.train(x_tr, y_tr)

        oof_train[test_index] = clf.predict(x_te)
        oof_test_skf[i, :] = clf.predict(x_test)

    oof_test[:] = oof_test_skf.mean(axis=0)
    return oof_train.reshape(-1, 1), oof_test.reshape(-1, 1)
```

```
# Create our OOF train and test predictions. These base results will be used as new features
et_oof_train, et_oof_test = get_oof(et, x_train, y_train, x_test) # Extra Trees
rf_oof_train, rf_oof_test = get_oof(rf,x_train, y_train, x_test) # Random Forest
ada_oof_train, ada_oof_test = get_oof(ada, x_train, y_train, x_test) # AdaBoost
gb_oof_train, gb_oof_test = get_oof(gb,x_train, y_train, x_test) # Gradient Boost
svc_oof_train, svc_oof_test = get_oof(svc,x_train, y_train, x_test) # Support Vector Classifier

print("Training is complete")
```

得出特征重要性评估

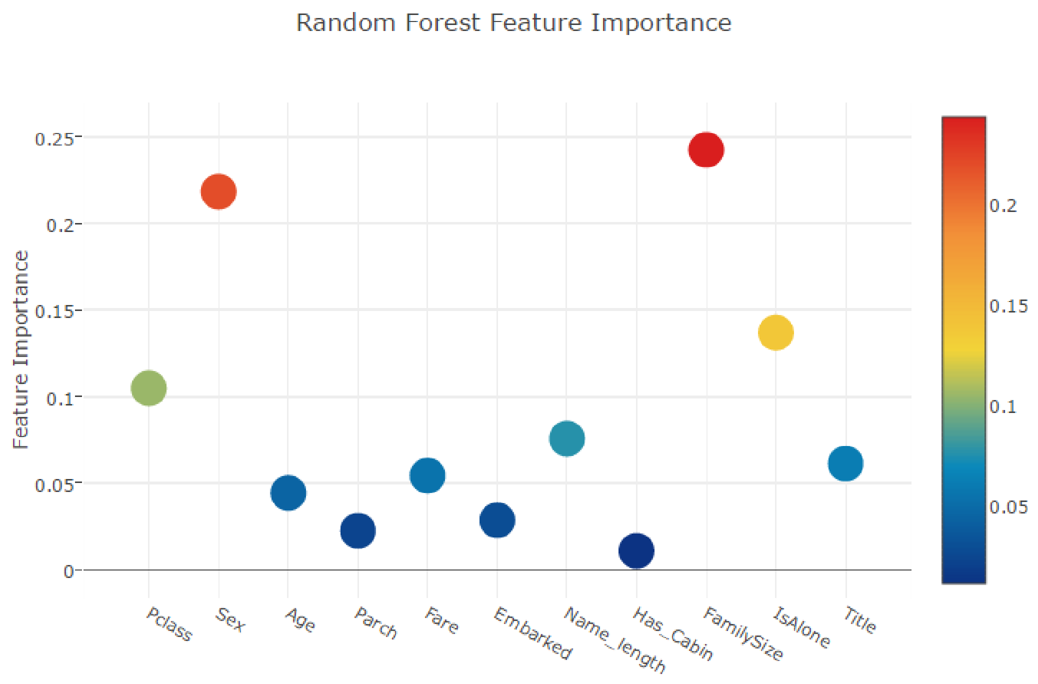
```
rf_feature = rf.feature_importances(x_train,y_train)
et_feature = et.feature_importances(x_train, y_train)
ada_feature = ada.feature_importances(x_train, y_train)
gb_feature = gb.feature_importances(x_train,y_train)
```

```
[ 0.12606282  0.20356521  0.0328714   0.02072402  0.07173324  0.02357351
  0.10810543  0.06457925  0.06807713  0.01350545  0.26720254]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/ensemble/forest.py:304: UserWarning:

Warm-start fitting without increasing n_estimators does not fit new trees.
```

```
[ 0.12071086  0.38397434  0.02246782  0.01693796  0.05648391  0.02685072
  0.04723326  0.08434829  0.04458866  0.02177096  0.17463323]
[ 0.034   0.012   0.02    0.066   0.038   0.01    0.68    0.016   0.052   0.002
  0.07 ]
[ 0.067636    0.04196342  0.11637161  0.03255574  0.08184844  0.05576079
  0.40231058  0.0207198   0.07008327  0.02294378  0.08780656]
```

对特征重要性可视化（只给出了随机森林的结果）



第二层模型

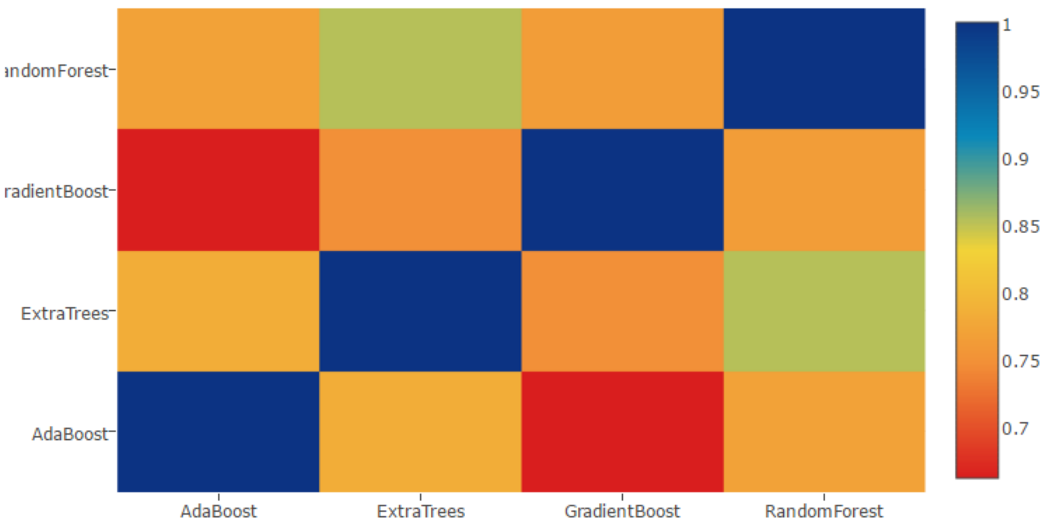
将第一层模型的输出作为特征输入

```
base_predictions_train = pd.DataFrame( {'RandomForest': rf_oof_train.ravel(),
     'ExtraTrees': et_oof_train.ravel(),
     'AdaBoost': ada_oof_train.ravel(),
      'GradientBoost': gb_oof_train.ravel()
    })
base_predictions_train.head()
```

	AdaBoost	ExtraTrees	GradientBoost	RandomForest
0	0.0	0.0	0.0	0.0
1	1.0	1.0	1.0	1.0
2	1.0	0.0	1.0	1.0
3	1.0	1.0	1.0	1.0
4	0.0	0.0	0.0	0.0

可视化第一层模型的相关程度

```
data = [
    go.Heatmap(
        z= base_predictions_train.astype(float).corr().values ,
        x=base_predictions_train.columns.values,
        y= base_predictions_train.columns.values,
          colorscale='Portland',
            showscale=True,
            reversescale = True
    )
]
py.iplot(data, filename='labelled-heatmap')
```



生成第二层模型的训练测试集

```
x_train = np.concatenate(( et_oof_train, rf_oof_train, ada_oof_train, gb_oof_train, svc_oof_train), axis=1)
x_test = np.concatenate(( et_oof_test, rf_oof_test, ada_oof_test, gb_oof_test, svc_oof_test), axis=1)
```

第二层模型使用 xgb 训练

```
gbm = xgb.XGBClassifier(
    #learning_rate = 0.02,
 n_estimators= 2000,
 max_depth= 4,
 min_child_weight= 2,
 #gamma=1,
 gamma=0.9,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread= -1,
 scale_pos_weight=1).fit(x_train, y_train)
predictions = gbm.predict(x_test)
```

得出预测结果

```
# Generate Submission File
StackingSubmission = pd.DataFrame({ 'PassengerId': PassengerId,
                            'Survived': predictions })
StackingSubmission.to_csv("StackingSubmission.csv", index=False)
```

（2） https://www.kaggle.com/benhamner/d/uciml/iris/python-data-visualizations 数据可视化基础教程

培训要求：

参加 DataCastle 上的大学生助学金精准资助预测

http://www.dcjingsai.com/common/cmpt/%E5%A4%A7%E5%AD%A6%E7%94%9F%E5%8A%A9%E5%AD%A6%E9%87%91%E7%B2%BE%E5%87%86%E8%B5%84%E5%8A%A9%E9%A2%84%E6%B5%8B_%E7%AB%9E%E8%B5%9B%E4%BF%A1%E6%81%AF.html

a. 按照 cross validation-3folds 要求计算分类的 F1 结果。
b. 在基础特征的基础上进行特征工程加工，使得 F1 的结果得到提升。
c. 在基础特征的基础上进行可视化来辅助自己特征处理。
d. 模型训练后输出特征的重要性程度。
e. 学习掌握多层模型机器学习流程。