数据挖掘任务 第二阶段

姓名：王永锋

学号：16337237

小组：智能分布式系统

这一个阶段，我主要完成了kaggle上的Titanic: Machine Learning from Disaster。

基础特征及其交叉验证结果

在开始完成这个任务前，我先对原始数据做了简单的特征提取，一些较复杂的特征没有处理直接丢弃。

In [108]:
```python
def get_base_feature(dataset):
    dataset = dataset.copy()
    # Sex feature
    dataset['Sex'] = dataset['Sex'].map( {'female':0, 'male':1}).fillna(0).astype(int)

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

    # Fare map
    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'].fillna(2).astype(int)

    #map 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
    dataset['Age'] = dataset['Age'].fillna(2)

    #drop some feature
    drop_elements = ['PassengerId', 'Name', 'Ticket', 'Cabin']
    dataset = dataset.drop(drop_elements, axis=1)
    return dataset
```

In [109]:
```python
get_base_feature(train)
```

Out[109]:

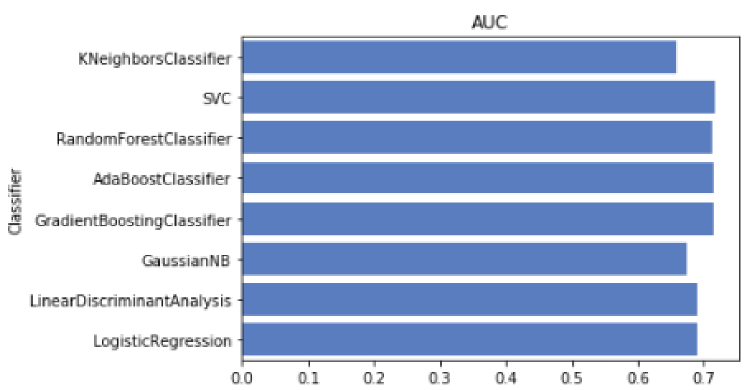
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	1.0	1	0	0	0
1	1	1	0	2.0	1	0	3	1
2	1	3	0	1.0	0	0	1	0
3	1	1	0	2.0	1	0	3	0

在这样的基础特征下，我使用了sklearn的多种分类器进行尝试，按照cross validataion-3folds要求进行计算分类结果的AUC结果，结果可见：

```
                    Classifier       auc
0         KNeighborsClassifier  0.658905
0                          SVC  0.717812
0       RandomForestClassifier  0.713733
0           AdaBoostClassifier  0.714110
0   GradientBoostingClassifier  0.714607
0                   GaussianNB  0.674792
0   LinearDiscriminantAnalysis  0.690482
0           LogisticRegression  0.690317
```

```
D:\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning: Variables are col
  warnings.warn("Variables are collinear.")
D:\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver
ed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma w
om 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning: Variables are col
  warnings.warn("Variables are collinear.")
D:\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver
ed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x175bc6b5128>



可以看到，此时的AUC分数还在0.70左右，仍然较低。

特征工程

在这一个部分，我提取了两个特征，并抛弃了原来的一些特征。

增加孤独特征

在前期数据分析的时候便发现，孤独的人相对而言获救概率更低。

游客类型，具有家庭还是孤独一人

构建新特征：家庭总人数 然后使用家庭总人数进行分组，判断哪一种类型的家庭具有更高的获救概率

In [176]:
```
for dataset in full_data:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
print(train[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False).mean())
```

```
   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
```

In [177]:
```
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())
```

```
   IsAlone  Survived
0        0  0.505650
1        1  0.303538
```

发现孤独一人的游客获救的概率更低

因此，我编写了以下代码提取孤独特征IsAlone，并删去了生成该特征的SibSp和Parch。

增加孤独特征

[110]:
```
def get_alone_feature(dataset):
    dataset = dataset.copy()
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1
    return dataset['IsAlone']
```

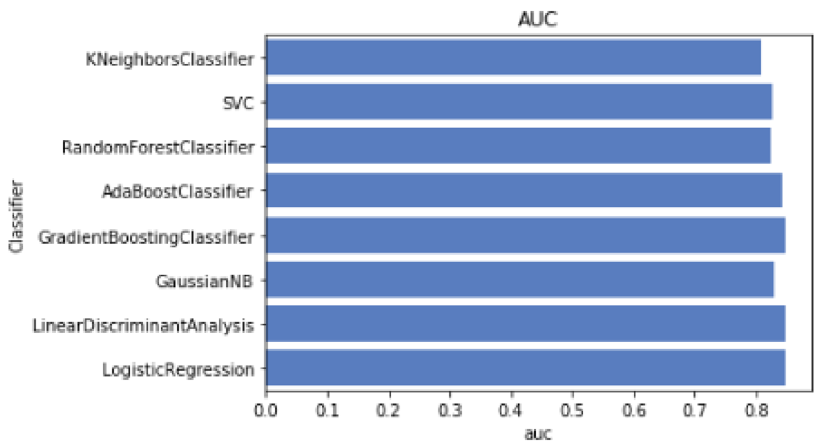
[111]:
```
train_feat = get_base_feature(train)
train_feat['IsAlone'] = get_alone_feature(train)

test_feat = get_base_feature(test)
test_feat['IsAlone'] = get_alone_feature(test)
```

效果是很明显的，平均的AUC分数上升到0.83-0.84左右。

```
                     Classifier       auc
0          KNeighborsClassifier  0.808563
0                           SVC  0.827893
0        RandomForestClassifier  0.824468
0            AdaBoostClassifier  0.843406
0    GradientBoostingClassifier  0.848876
0                    GaussianNB  0.830435
0    LinearDiscriminantAnalysis  0.848217
0            LogisticRegression  0.848341
```

16]: <matplotlib.axes._subplots.AxesSubplot at 0x175bc047f28>



增加名字特征

名字中不同的称呼，可能也能够反映不同的人社会地位的不同，这对获救概率也会有一定的影响。

因此我编写了以下的代码给原数据集增加称呼特征

增加名字特征

[117]:
```
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    if title_search:
        return title_search.group(1)
    return ""

def get_name_feature(dataset):
    dataset = dataset.copy()
    dataset['Title'] = dataset['Name'].apply(get_title)
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess', 'Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', '
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
    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)
    return dataset['Title']
```

[118]:
```
train_feat['Title'] = get_name_feature(train)
test_feat['Title'] = get_name_feature(test)
```

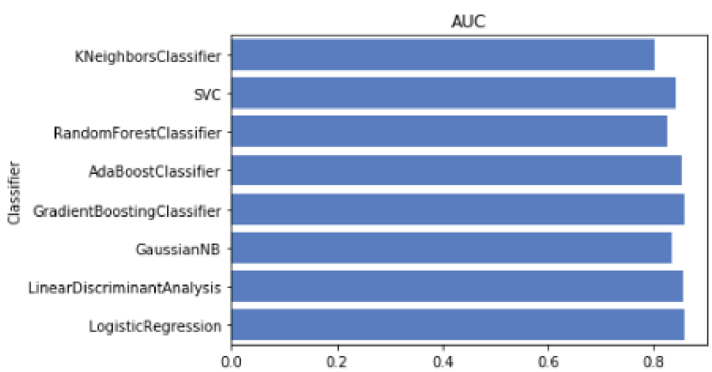
[119]:
```
train_feat
```

该特征对于模型效果的特征也是有一定效果的，AUC分数从0.83~0.84上升到0.84~0.85左右。

```
                    Classifier       auc
0         KNeighborsClassifier  0.801748
0                          SVC  0.842618
0       RandomForestClassifier  0.827816
0           AdaBoostClassifier  0.853279
0   GradientBoostingClassifier  0.859862
0                   GaussianNB  0.833788
0   LinearDiscriminantAnalysis  0.857221
0           LogisticRegression  0.857808
```

```
D:\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be
ed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
D:\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will cha
om 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'aut
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
D:\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be
ed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

120]: <matplotlib.axes._subplots.AxesSubplot at 0x175bbe11d30>



输出特征的重要性程度

重要性程度可见下图

```
et_feature = et.feature_importances(x_train, y_train)
rf_feature = rf.feature_importances(x_train,y_train)
ada_feature = ada.feature_importances(x_train, y_train)
gb_feature = gb.feature_importances(x_train,y_train)
# svc_feature = svc.feature_importances(x_train,y_train)
# gnb_feature = gnb.feature_importances(x_train,y_train)
# lda_feature = lda.feature_importances(x_train,y_train)
# lr_feature = lr.feature_importances(x_train,y_train)
```

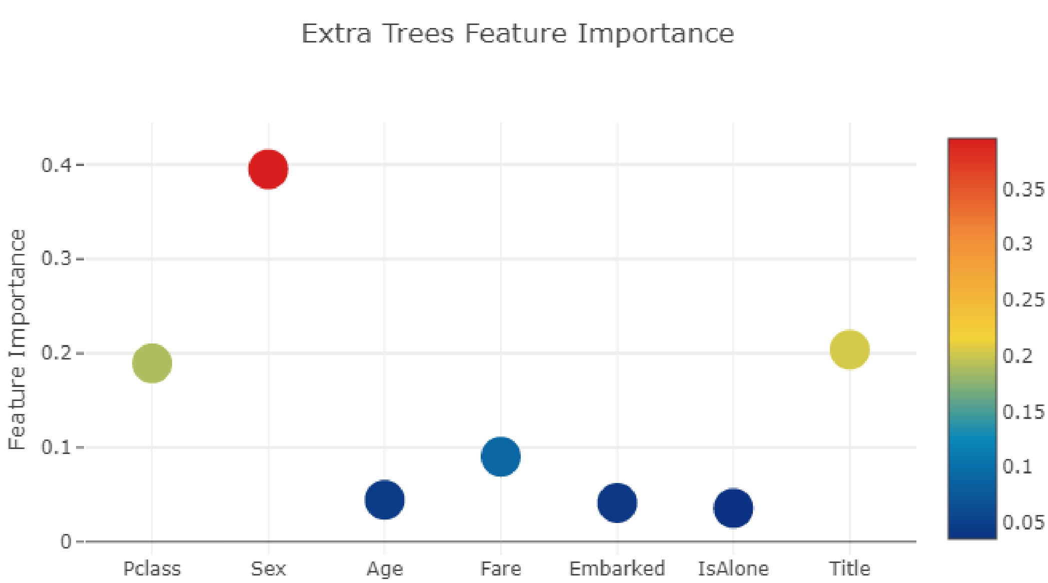
```
[0.19220385 0.39169673 0.05046145 0.08715598 0.04289191 0.03461016
 0.20097995]
[0.21320911 0.24006382 0.0602857  0.11765483 0.04435943 0.03783867
 0.28658844]
```

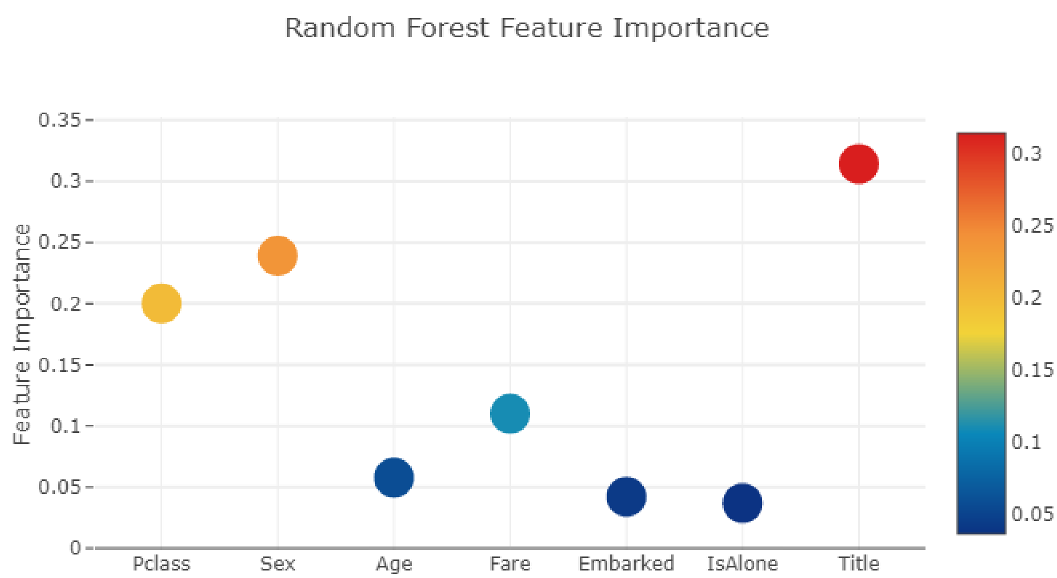
```
D:\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:310: UserWarning:

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

```
[0.022 0.332 0.018 0.042 0.024 0.014 0.548]
[0.17988504 0.02783605 0.06772011 0.10167748 0.03536892 0.01955961
 0.5679528 ]
```

部分结果的可视化可见下图



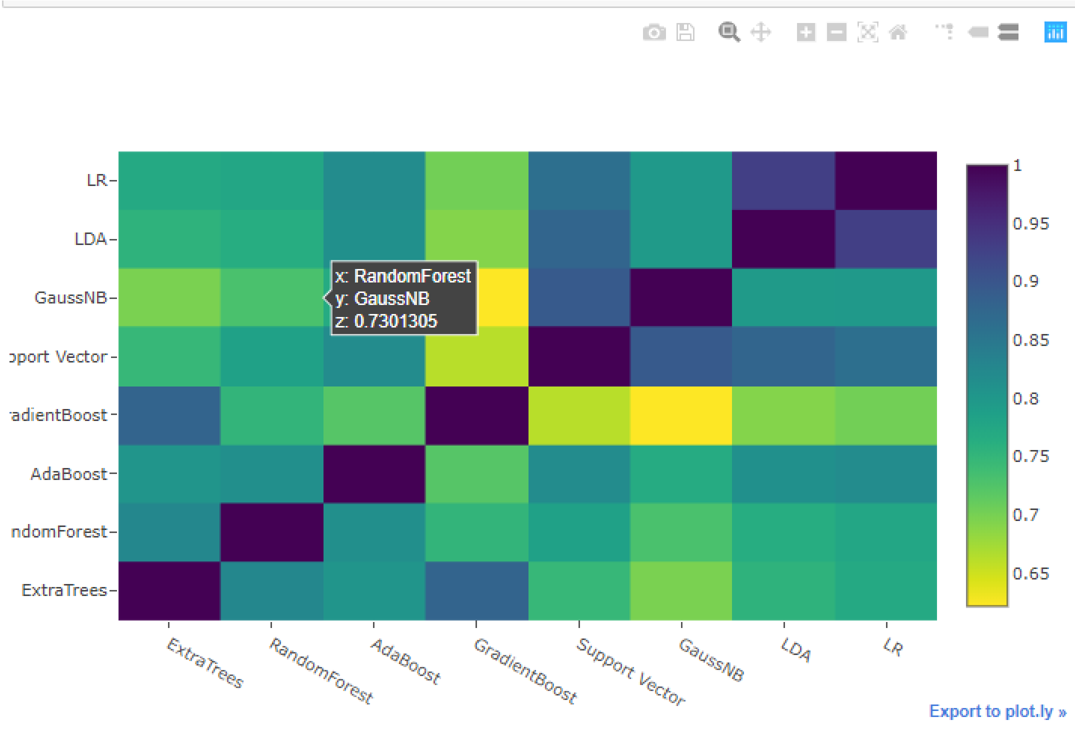


模型融合

这部分我在第一层使用了这些模型作为基模型。

```
# 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)
gnb = SklearnHelper(clf=GaussianNB, seed=SEED, params=gnb_params)
lda = SklearnHelper(clf=LinearDiscriminantAnalysis, seed=SEED, params=lda_params)
lr = SklearnHelper(clf=LogisticRegression, seed=SEED, params=lr_params)
```

这些模型输出的线性相关程度如下图



然后将上面的模型的输出作为输入，使用xgboost模型得到第二层模型的输出。

进行了交叉验证，可以见到这一个二层的容和模型能够将AUC分数稳定在0.85左右
。

但是提升效果并不大，我想可能与之前有些模型的输出结果线性相关程度过高有关
。

```
StackingSubmission = pd.DataFrame({ 'PassengerId': test['PassengerId'].values,
                            'Survived': predictions })
StackingSubmission.to_csv("StackingSubmission2.csv", index=False)
```

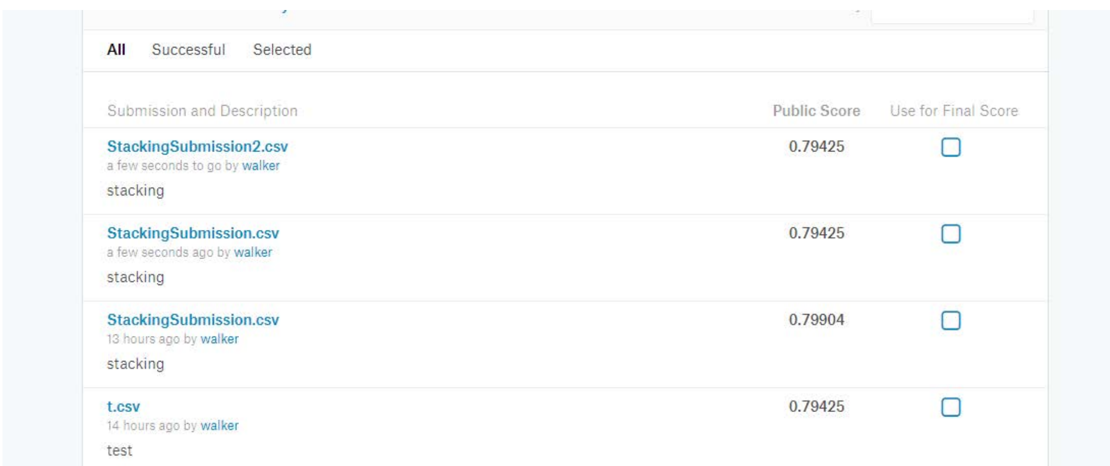
```
n [185]: xg_train = xgb.DMatrix(x_train, label=y_train);
cv = xgb.cv(xgb_params, xg_train, 50000, nfold=3, early_stopping_rounds=early_stopping, verbose_eval=1)
```

```
[0]	train-auc:0.84239+0.00600702	test-auc:0.833937+0.0122053
[1]	train-auc:0.842763+0.00652405	test-auc:0.834863+0.0128916
[2]	train-auc:0.846284+0.00873901	test-auc:0.834228+0.0131382
[3]	train-auc:0.850002+0.0028855	test-auc:0.842314+0.00470324
[4]	train-auc:0.850063+0.00288318	test-auc:0.842971+0.00318569
[5]	train-auc:0.848629+0.00245952	test-auc:0.843822+0.00406912
[6]	train-auc:0.849114+0.00255503	test-auc:0.843718+0.0041324
[7]	train-auc:0.849606+0.00289082	test-auc:0.843513+0.00403889
[8]	train-auc:0.851441+0.00188069	test-auc:0.847619+0.00540695
[9]	train-auc:0.851738+0.00203609	test-auc:0.848509+0.00427682
[10]	train-auc:0.85175+0.00240832	test-auc:0.84844+0.00454328
[11]	train-auc:0.85181+0.00269779	test-auc:0.84934+0.00473601
[12]	train-auc:0.852029+0.00263169	test-auc:0.849159+0.00484067
[13]	train-auc:0.852235+0.0026709	test-auc:0.849243+0.00465093
[14]	train-auc:0.852278+0.00275919	test-auc:0.849209+0.00473419
[15]	train-auc:0.852842+0.00228474	test-auc:0.849284+0.00474364
[16]	train-auc:0.852848+0.00224865	test-auc:0.848249+0.00456743
[17]	train-auc:0.852931+0.00245722	test-auc:0.848395+0.00471119
[18]	train-auc:0.852602+0.00247638	test-auc:0.848083+0.00476656
[19]	train-auc:0.852614+0.00246407	test-auc:0.848147+0.00484692
[20]	train-auc:0.852563+0.00247639	test-auc:0.848215+0.00464752
```

```
n [182]: cv
```

最终结果

在kaggle上提交了多次结果，可以见到这一个二层的融合模型能够将AUC分数稳定在0.85 左右



其中，的确发现，在模型融合中，模型的选择也是很重要的一部分，要选择输出的相关程度较低的，可能能够获得更好的效果。