

**华东理工大学**

**模式识别大作业**

**题目 信用卡申请比赛**

**院系 信息科学与工程学院**

**专业 信息与通信工程**

**组员**  刘渝

**指导老师**  赵海涛

**信用卡申请比赛**

组员：刘渝

**一、信用卡申请比赛简介**

商业问题:银行收到如此多的信用卡申请，手动遍历每个请求非常耗时，也容易出现人为错误。但是，如果可以使用历史数据来构建一个模型，该模型可以列出候选对象，从而节省时间。

提供的训练集包含信用卡的过去的申请。数据集中的每一行都包含申请人的信息(性别、年龄、债务、婚姻等)，输出是决策(1:批准，0:拒绝)在这个挑战中，要求完成对什么样的人可能批准申请得到信用卡的分析。

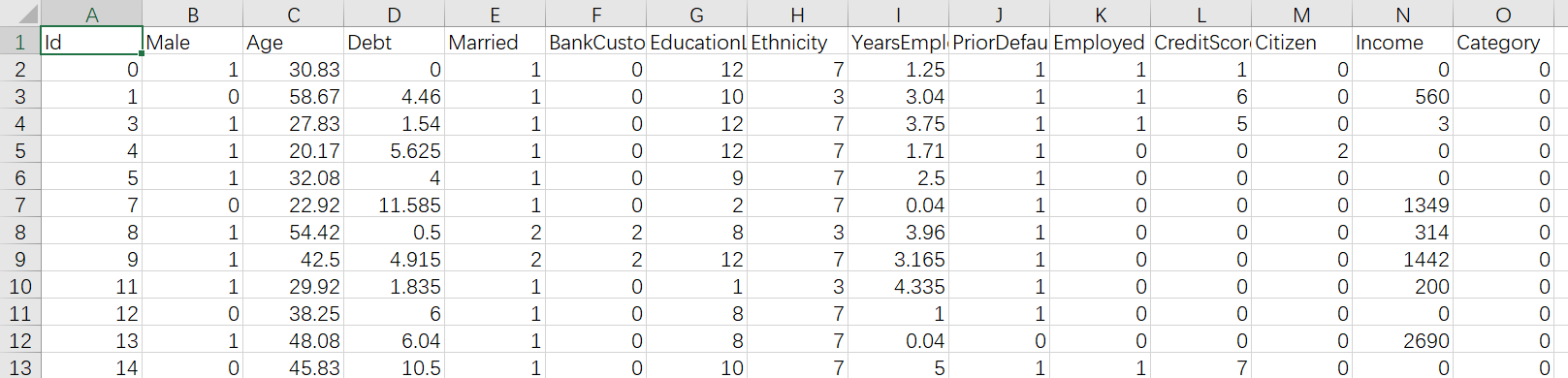
来源kaggle: <https://www.kaggle.com/c/creditcardcompetitioncs6501/overview>

**二、实验方案**

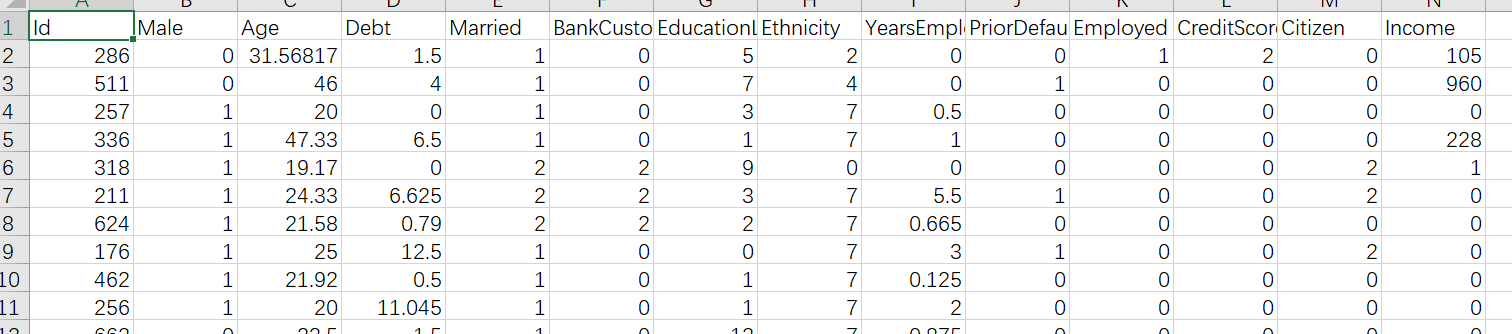
将某人申请信用卡成功与否抽象为一个0-1二分类问题，这里本人考虑到可以建立一个通用的训练模型构函数求最优模型参数，而后将各个模型和相应的参数范围输入到函数中，通过求解比较模型训练后的准确率来最终选择模型，本次设计的语言为：python。因为得益于python中大量的第三方机器学习模块，这项工作的数据处理以及模型构建，参数求最优化将会变得简单，一个真正好的程序员应该用最简单的代码实现复杂的要求，这也是奉行奥卡姆剃刀法则。

* 1. **数据读入**

数据可从Kaggle上的信用卡比赛中下载。数据包括训练数据train.csv，和测试数据test.csv。内容如图所示：



**图1训练集信息**



**图2 测试集信息**

其中测试数据一共有540组，训练数据有150组。数据所表示的意思如下所示：

* Id
* Male
* Age
* Debt
* Married
* BankCustomer
* EducationLevel
* Ethnicity
* YearsEmployed
* PriorDefault
* Employed
* CreditScore
* Citizen
* Income

可以看出。每个申请者的有14个特征，最后的Category是其标签。

引入实验可能用到的模块：

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from collections import defaultdict

from sklearn.model\_selection import train\_test\_split

import numpy as np

from sklearn import tree

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer

from sklearn.metrics import accuracy\_score,roc\_auc\_score

from time import time

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.neighbors import KNeighborsClassifier 。

数据读入：

df=pd.read\_csv('train.csv')

**2.2 数据分析与处理**

df=df.drop('Id',axis=1)

x=df.drop('Category',axis=1)

y=df.Category

#转换成数值型数据

d = defaultdict(LabelEncoder)

x\_trains = x.apply(lambda x: d[x.name].fit\_transform(x))

最简单的方式就是将申请人的Id名丢掉，然后将目标输出定为y。本次的训练数据集没有缺失值，所以不需要处理。

**2.3 统一的训练模型构建**

def fit\_model(alg,parameters):

#由于数据较少，使用全部数据进行网格搜索

scorer=make\_scorer(roc\_auc\_score)

#使用roc\_auc\_score作为评分标准

grid = GridSearchCV(alg,parameters,scoring=scorer,cv=5)

#使用网格搜索，出入参数

start=time() #计时

grid=grid.fit(x,y) #模型训练

end=time()

t=round(end-start,3)

print (grid.best\_params\_)

#输出最佳参数

print ('searching time for {} is {} s'.format(alg.\_\_class\_\_.\_\_name\_\_,t))

#输出搜索时间

print("train score:", grid.score(x, y))

# print("test score:", grid.score(x\_test, y\_test))

return grid

* 该模型函数有两个变量：alg和parameters,前者是模型，后者是参数。

大杀器：通常算法不够好，需要调试参数时必不可少。比如SVM的惩罚因子C，核函数kernel，gamma参数等，对于不同的数据使用不同的参数，结果可能差1-5个点，sklearn为我们提供专门调试参数的函数GridSearchCV，它存在的意义就是自动调参，只要把参数输进去，就能给出最优化的结果和参数。但是这个方法适合于小数据集，一旦数据的量级上去了，很难得出结果。这个时候就是需要动脑筋了。数据量比较大的时候可以使用一个快速调优的方法——坐标下降。它其实是一种贪心算法：拿当前对模型影响最大的参数调优，直到最优化；再拿下一个影响最大的参数调优，如此下去，直到所有的参数调整完毕。这个方法的缺点就是可能会调到局部最优而不是全局最优，但是省时间省力，巨大的优势面前，还是试一试吧，后续可再优化。

* 参数说明

（1）estimator

选择使用的分类器，并且传入除需要确定最佳的参数之外的其他参数。一个分类器都需要一个scoring参数，或者score方法：estimator=RandomForestClassifier(min\_samples\_split=100,min\_samples\_leaf=20,max\_depth=8,max\_features='sqrt',random\_state=10),

（2） param\_grid

需要最优化的参数的取值，值为字典或者列表，例如：param\_grid =param\_test1，param\_test1 = {'n\_estimators':range(10,71,10)}。

（3） scoring=None

模型评价标准，默认None,这时需要使用score函数；或者如scoring='roc\_auc'，根据所选模型不同，评价准则不同。字符串（函数名），或是可调用对象，需要其函数签名形如：scorer(estimator, X, y)；本次的评价选择roc\_auc

（4）fit\_params=None

（5）n\_jobs=1

n\_jobs: 并行数，int：个数,-1：跟CPU核数一致, 1:默认值

（6）iid=True

iid:默认True,为True时，默认为各个样本fold概率分布一致，误差估计为所有样本之和，而非各个fold的平均。

（7）refit=True

默认为True,程序将会以交叉验证训练集得到的最佳参数，重新对所有可用的训练集与开发集进行，作为最终用于性能评估的最佳模型参数。即在搜索参数结束后，用最佳参数结果再次fit一遍全部数据集。

（8）cv=None

交叉验证参数，默认None，使用三折交叉验证。指定fold数量，默认为3，也可以是yield训练/测试数据的生成器。本次使用5折。

（9）verbose=0, scoring=None

verbose：日志冗长度，int：冗长度，0：不输出训练过程，1：偶尔输出，>1：对每个子模型都输出。

（10）pre\_dispatch=‘2\*n\_jobs’

指定总共分发的并行任务数。当n\_jobs大于1时，数据将在每个运行点进行复制，这可能导致OOM，而设置pre\_dispatch参数，则可以预先划分总共的job数量，使数据最多被复制pre\_dispatch次

（11）error\_score=’raise’

（12）return\_train\_score=’warn’

如果“False”，cv\_results\_属性将不包括训练分数

回到sklearn里面的GridSearchCV，GridSearchCV用于系统地遍历多种参数组合，通过交叉验证确定最佳效果参数。

**2.4 分类模型以及参数范围**

alg1=DecisionTreeClassifier(random\_state=29)

alg2=SVC(probability=True,random\_state=29) #由于使用roc\_auc\_score作为评分标准，需将SVC中的probability参数设置为True

alg3=RandomForestClassifier(random\_state=29)

alg4=AdaBoostClassifier(random\_state=29)

alg5=KNeighborsClassifier(n\_jobs=-1)

#列出需要调整的参数范围

parameters1={'max\_depth':range(1,10),'min\_samples\_split':range(2,10)}

parameters2 = {"C":range(1,20), "gamma": [0.05,0.1,0.15,0.2,0.25]}

parameters3\_1 = {'n\_estimators':range(10,200,10)}

parameters3\_2 ={'max\_depth':range(1,10),'min\_samples\_split':range(2,10)}

#搜索空间太大，分两次调整参数

parameters4 = {'n\_estimators':range(10,200,10),'learning\_rate':[i/10.0 for i in range(5,15)]}

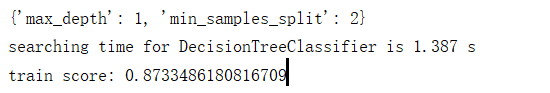
parameters5 = {'n\_neighbors':range(2,10),'leaf\_size':range(10,80,20) }

需要考虑的算法是：决策树，SVM，随机森林，Adaboost，KNN。

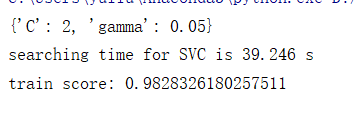
分别对应的是alg2,alg2,alg3,alg4,alg5。接下来要做的就是把这五个算法，分别导入到之前设定的最优参数的统一模型。

**2.5 最优参数以及训练集准确率**

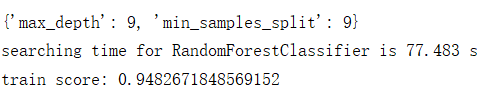
决策树最佳参数以及准确率：



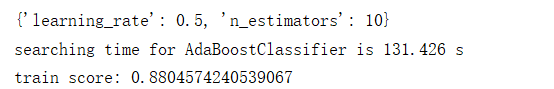
SVM最佳参数以及准确率：



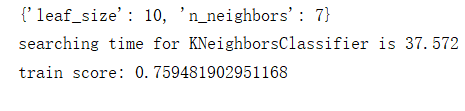
随机森林最佳参数以及准确率：



AdaBoostClassifier最佳参数以及准确率：



KNN最佳参数以及准确率：



可以看出来随机森林和SVM分类器的准确率比较高。

**2.6 模型预测输出**

test=pd.read\_csv('test.csv')

Id=test.Id

test=test.drop('Id',axis=1)

clf1=fit\_model(alg1,parameters1)

clf2=fit\_model(alg2,parameters2)

clf3\_m1=fit\_model(alg3,parameters3\_1)

alg3=RandomForestClassifier(random\_state=29,n\_estimators=180)

clf3=fit\_model(alg3,parameters3\_2)

clf4=fit\_model(alg4,parameters4)

clf5=fit\_model(alg5,parameters5)

def save(clf,i):

category=clf.predict(test)

sub=pd.DataFrame({ 'Id': Id, 'Category': category })

sub.to\_csv("res\_tan\_{}.csv".format(i), index=False)

i=1

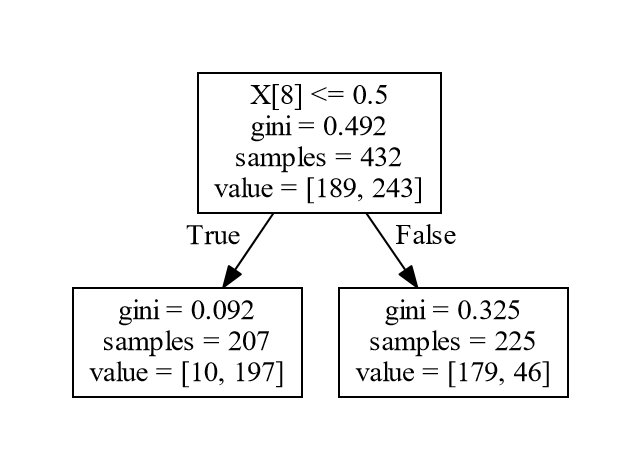
for clf in [clf1,clf2,clf3,clf4,clf5]:

save(clf,i)

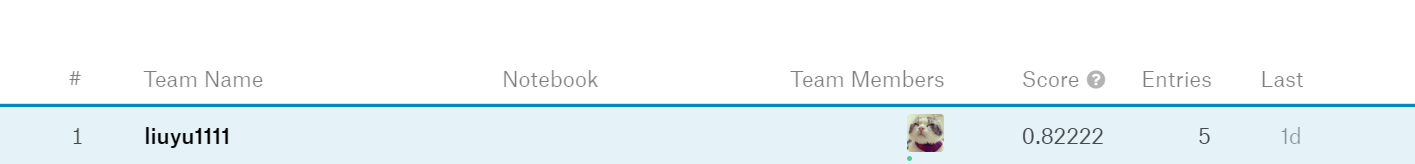
i=i+1

**三、实验结论**

对自己实现的模型，进行可视化，例如决策树，它的最佳深度是1。



**图3 决策时可视化**



**图4 Kaggle结果**

将自己训练的五个模型，传入Kaggle，得分0.82222,排名第一。由此可见在做二分类的预测问题时传统的随机森林和SVM更加有效。