****

****

**QG中期考核详细报告书**

**题    目     金融公司预测通过率**

**学   院       自动化学院**

**专 业 物联网工程**

**年级班别 20级(1)班**

**学 号 3120001384**

**学生姓名 陈铿任**

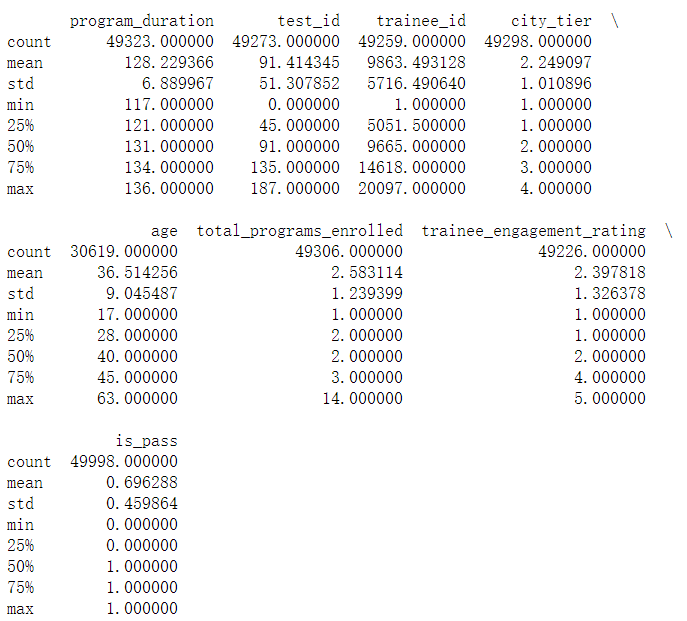
**2021年 4 月16 日**

一.数据集的处理：

1.导入数据集：

MFC = pd.read\_csv("文件路径.csv")。

2.首先我们进行查看数字类型的数据集的数据情况：



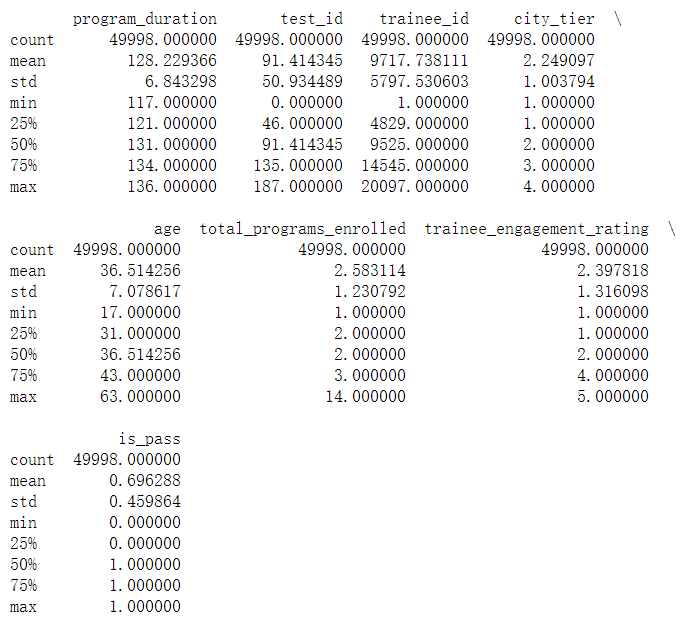
我们可以发现除了通过率外所有数字类型的数据均有缺失值，进行了两种处理方式：

a. 对全部标签进行均值填充：

MFC['label'] = MFC['label'].fillna(MFC[‘label’].mean())

b. 对缺失值较多的age标签切割掉：

MFC = MFC.drop(['age'],axis=1)



3.然后处理字符类型的数据：

方案一：

进行查看指定标签下的值是否缺失print(MFC['label'].unique())；若存在缺 失值(进行指定标签下个各值的统计count = MFC['label'].value\_counts())使 用统计数量最多的数值来填充缺失值然后进行每个字符类型数值的独热编 码。

方案二：

先对标签下各字符数值进行独热编码，再均值填充缺失值。

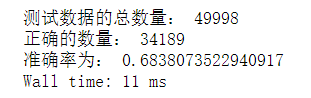
二.对比分析不同模型

(图一为对应数字类型标签缺失均值填充、图二为drop[‘age’]、图三为字符类型标签均值填充)

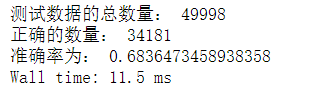
KFold中的kf.get\_n\_splits(MFC)得到交叉验证集的拆分迭代次数为5

1.线性回归模型：

kf = KFold(n\_splits=5,random\_state=1,shuffle=True)



kf = KFold(n\_splits=5,random\_state=42,shuffle=True)



通过两次调参对比准确率和运行时间没有明显的变化，这种模型准确率偏低。

2.逻辑回归模型：

LogisticRegression(random\_state=1,solver='liblinear')



LogisticRegression(random\_state=15,solver='liblinear')



通过调random\_state参数对比，准确率提高了0.01左右时间多了0.6秒

a.数据幅度标准化：

X\_train=StandardScaler.fit\_transform(MFC[predictors])



b.梯度下降：

sgdc=SGDClassifier(random\_state=15,loss='log')



通过对比发现这两种方法效果不是很好，只是在运行时间上变短了。

3.决策树模型：

DecisionTreeClassifier(max\_depth=5)



DecisionTreeClassifier(max\_depth=10)



将max\_depth调为10，准确率一下子上升了0.037。

4.随机森林模型：

RandomForestClassifier(random\_state=1, n\_estimators=50)

kf = model\_selection.KFold(n\_splits=5, random\_state=1,shuffle=True) 

RandomForestClassifier(random\_state=1, n\_estimators=500)

kf = model\_selection.KFold(n\_splits=5, random\_state=1,shuffle=True)



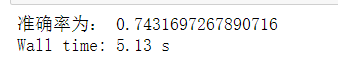
RandomForestClassifier(random\_state=1, n\_estimators=500)

kf = model\_selection.KFold(n\_splits=5, random\_state=42,shuffle=True)



通过调节n\_estimators和random\_state对比，n\_estimators(即构建的子树)越大运行时间越长，而准确率提高了0.002；改变交叉验证的random\_state值，准确率只提高了0.0003。

RandomForestClassifier(min\_samples\_split=4,min\_samples\_leaf=2,oob\_score=True, random\_state=5)



通过增加oob\_score这个参数准确率跳了0.001左右，而且使运行时间降低到了6s内。

新增max\_features='sqrt'/’log2’准确率没有变化，只是运行时间有时比上述少。

5.神经网络模型：

MLPClassifier(max\_iter=200,solver='adam')



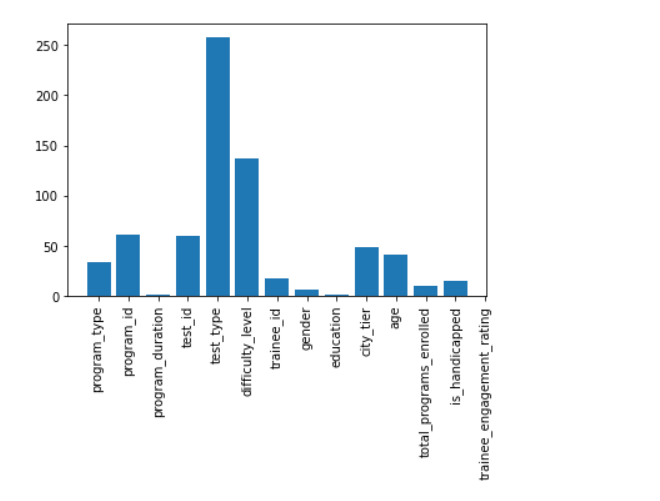
MLPClassifier(hidden\_layer\_sizes=(100,100,100,50),max\_iter=200,solver='adam')



通过对比发现通过调节隐藏层可以提高准确率，但调节不当的话会使运行时间暴增。

6.特征选择：

通过添加噪声，画出直方图。



可以发现

test\_type,difficulty\_level,test\_id,program\_id,city\_tier,program\_type,age,trainee\_id,is\_handicapped所占的权重比较大，选择这8个特征进行模型预测。



7.集成算法：

使用上述选择的特征，集成GradientBoostingClassifier和LogisticRegression来预测模型

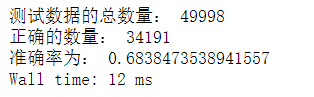


对比发现效果没有想象中的那么好。

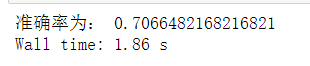
三.对比分析对age的不同处理方式

1.线性回归：(方案b效果略优于方案a)





2.逻辑回归：(方案a在准确率上优于方案b，但在运行时间上方案b占优)



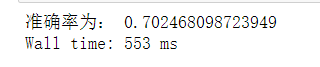
. 

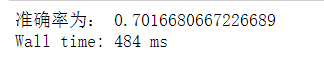
a.数据幅度标准化：(方案a在准确率上优于方案b，但在运行时间上方案b占优)





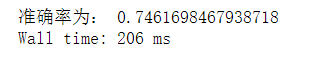
b.梯度下降：(方案a在准确率上优于方案b，但在运行时间上方案b占优)





3.决策树：(方案a效果比较好))



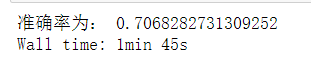


4.随机森林：(方案a效果比方案b好)





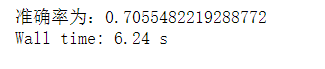
5.神经网络：(方案a效果比方案b好)





6.特征选择：(方案a效果比方案b好)





7.集成多个算法：(方案b准确率略高于方案一，但在运行时间上多了6s左右)



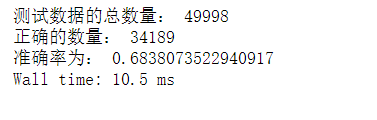


结论：直接去掉age这个标签对于一些模型来说是优化，但效果不是很理想。

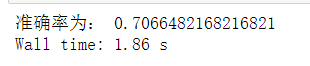
四.对比前面所说的方案一和方案二

1.线性回归：(方案二仅在运行时间上优于方案一)





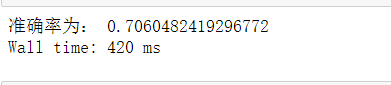
2.逻辑回归：(方案二比方案一效果好一点)



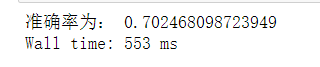
.

a.数据幅度标准化：(方案二仅在运行时间上略优于方案一)





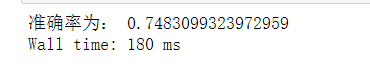
b.梯度下降：(方案二仅在运行时间上略优于方案一)





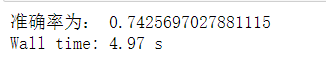
3.决策树：(方案一效果比较好))



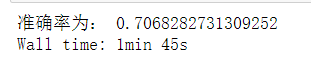


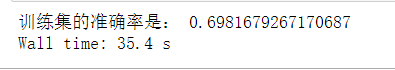
4.随机森林：(方案一准确率优于方案二，运行时间上方案二略胜一筹)





5.神经网络：(方案一在准确率上优于方案二，但在运行时间上则是方案二胜)





6.特征选择：(方案二仅在运行时间上略优于方案一)





7.集成多个算法：(方案二准确率略高于方案一，但在运行时间上多了7s左右)





结论：方案二没有想象中的那么好。