廣東工業大學

QG 中期考核详细报告书

题　　目　数据挖掘

学　　院　计算机学院

专　　业　计算机类

年级班别　20 级计算机类 10 班

学　　号　3120005087

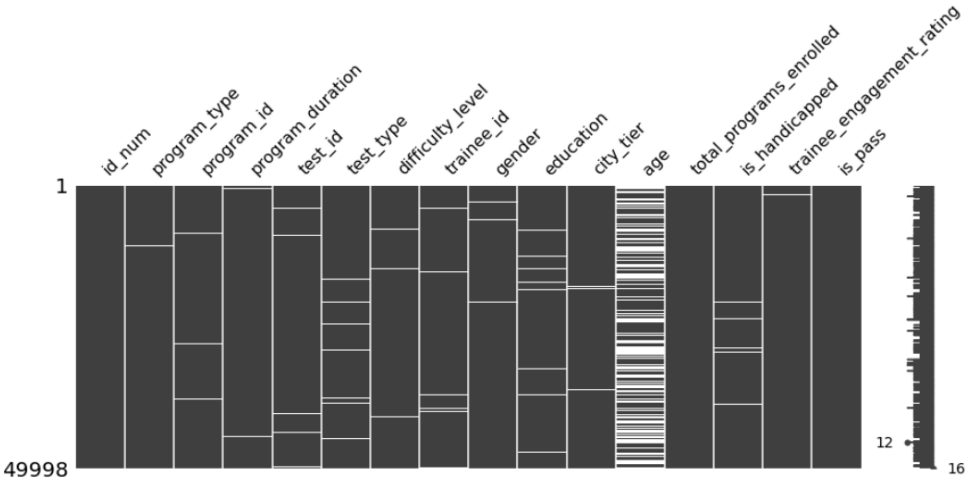
学生姓名　赵朗

2021 年　4 月　17 日

数据清洗：

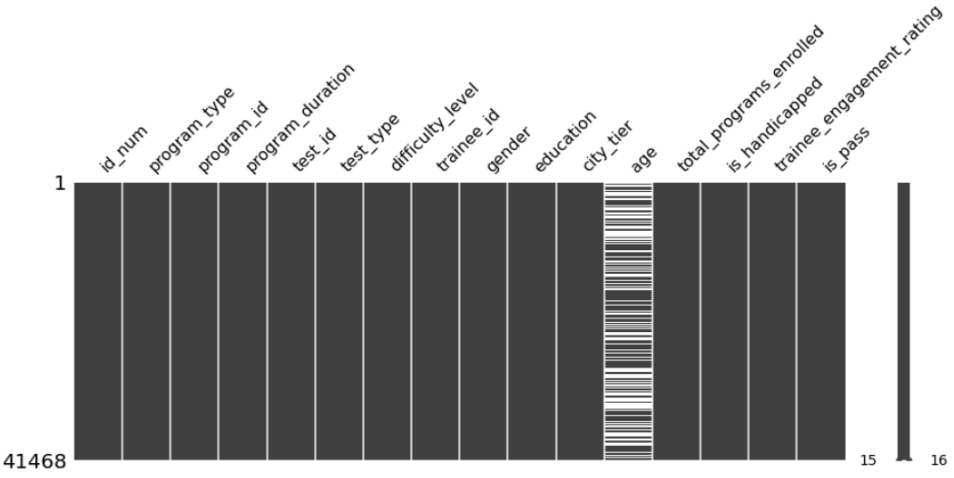
查看缺失值

```
In [3]: msno.matrix(trainee_original, figsize=(15, 5))
Out[3]: <AxesSubplot:>
```



除了 age 列，其余数据皆是零散的缺失，因此将有缺失值的数据直接删去

```
In [4]: #将缺失值较少的数据直接删去
trainee = trainee_original.dropna(axis=0,subset = ["program_type","program_id","program_duration","test_id","test_type","difficulty_level","

In [5]: msno.matrix(trainee, figsize=(15, 5))
Out[5]: <AxesSubplot:>
```



对含有缺失值较多的 age 列进行分析

```
'''
Y = sum(trainee['age'] <= trainee['age'].mean())            #Y:younger than age's mean
O = sum(trainee['age'] > trainee['age'].mean())             #O:older than age's mean
YPass = 0                                                   #Y:younger than age's mean and pass
OPass = 0                                                   #Y:older than age's mean and pass
for(x,y) in zip(trainee['age'],trainee['is_pass']):
    if x<=trainee['age'].mean() and y==1:
        YPass+=1
    elif x>trainee['age'].mean() and y==1:
        OPass+=1
Y,O
YPass,OPass
YPass/Y,OPass/O
trainee['age'].mean()
'''

#通过计算发现年龄大于平均值的人数为13059，通过率为73%
#通过计算发现年龄小于平均值的通过率为12368，通过率为66%
#两者相差不大，因此可以用平均值填充缺失值
```

```
trainee['age'] = trainee['age'].fillna(trainee['age'].mean())
```

因为年龄大于或小于平均值的人，他们的数量和通过率相差不大，因此用年龄的平均值来进行数据填充

数据分析：

通过观察发现 program_type, program_id, program_duration 是互相对应的，为一类数据
因此根据不同 program_id 的通过率对其进行赋值

In [9]:
```
#program_type, program_id, program_duration 为一组数据，根据不同program_id通过率为其赋值
trainee.loc[trainee['program_id'] == 'Y_1', "program_id"] = 8        #通过率:0.78
trainee.loc[trainee['program_id'] == 'Y_2', "program_id"] = 7        #通过率:0.74
trainee.loc[trainee['program_id'] == 'Y_3', "program_id"] = 8        #通过率:0.75
trainee.loc[trainee['program_id'] == 'Y_4', "program_id"] = 8        #通过率:0.77

trainee.loc[trainee['program_id'] == 'T_1', "program_id"] = 5        #通过率:0.51
trainee.loc[trainee['program_id'] == 'T_2', "program_id"] = 7        #通过率:0.74
trainee.loc[trainee['program_id'] == 'T_3', "program_id"] = 7        #通过率:0.72
trainee.loc[trainee['program_id'] == 'T_4', "program_id"] = 7        #通过率:0.74

trainee.loc[trainee['program_id'] == 'U_1', "program_id"] = 6        #通过率:0.64
trainee.loc[trainee['program_id'] == 'U_2', "program_id"] = 6        #通过率:0.60

trainee.loc[trainee['program_id'] == 'V_1', "program_id"] = 6        #通过率:0.60
trainee.loc[trainee['program_id'] == 'V_2', "program_id"] = 6        #通过率:0.56
trainee.loc[trainee['program_id'] == 'V_3', "program_id"] = 6        #通过率:0.56
trainee.loc[trainee['program_id'] == 'V_4', "program_id"] = 6        #通过率:0.58

trainee.loc[trainee['program_id'] == 'Z_1', "program_id"] = 7        #通过率:0.71
trainee.loc[trainee['program_id'] == 'Z_2', "program_id"] = 7        #通过率:0.72
trainee.loc[trainee['program_id'] == 'Z_3', "program_id"] = 7        #通过率:0.73

trainee.loc[trainee['program_id'] == 'X_1', "program_id"] = 9        #通过率:0.85
trainee.loc[trainee['program_id'] == 'X_2', "program_id"] = 8        #通过率:0.77
trainee.loc[trainee['program_id'] == 'X_3', "program_id"] = 8        #通过率:0.81

trainee.loc[trainee['program_id'] == 'S_1', "program_id"] = 5        #通过率:0.52
trainee.loc[trainee['program_id'] == 'S_2', "program_id"] = 6        #通过率:0.59
```

计算 test_type 特征不同特征值通过率并对其赋值

```
'''
offPass = 0
onPass = 0
for (x,y) in zip(trainee['test_type'],trainee['is_pass']):
    if x=='offline' and y==1:
        offPass+=1
    elif x=='online' and y==1:
        onPass+=1
offPass/sum(trainee['test_type'] == 'offline'),onPass/sum(trainee['test_type'] == 'online')
'''
#通过计算得到online and pass 的概率为0.78，offline and pass 的概率为0.63
```

```
trainee.loc[trainee['test_type'] == 'offline', "test_type"] = 6
trainee.loc[trainee['test_type'] == 'online', "test_type"] = 8
```

计算 difficulty_level 特征不同特征值通过率并对其赋值

```
'''
ePass = 0
iPass = 0
hPass = 0
vPass = 0
for (x,y) in zip(trainee['difficulty_level'],trainee['is_pass']):
    if x=='easy' and y==1:
        ePass+=1
    elif x=='intermediate' and y==1:
        iPass+=1
    elif x=='hard' and y==1:
        hPass+=1
    elif x=='vary hard' and y==1:
        vPass+=1
ePass/sum(trainee['difficulty_level'] == 'easy')            0.73
iPass/sum(trainee['difficulty_level'] == 'intermediate')    0.65
hPass/sum(trainee['difficulty_level'] == 'hard')            0.67
vPass/sum(trainee['difficulty_level'] == 'vary hard')      0.43
'''
```

```
trainee.loc[trainee['difficulty_level'] == 'easy', "difficulty_level"] = 7
trainee.loc[trainee['difficulty_level'] == 'intermediate', "difficulty_level"] = 7
trainee.loc[trainee['difficulty_level'] == 'hard', "difficulty_level"] = 7
trainee.loc[trainee['difficulty_level'] == 'vary hard', "difficulty_level"] = 4
```

计算 education 特征不同特征值通过率并对其赋值

```
'''
HPass = 0
MatPass = 0
BPass = 0
MasPass = 0
NPass = 0
for (x,y) in zip(trainee['education'],trainee['is_pass']):
    if x=='High School Diploma' and y==1:
        HPass+=1
    elif x=='Matriculation' and y==1:
        MatPass+=1
    elif x=='Bachelors' and y==1:
        BPass+=1
    elif x=='Masters' and y==1:
        MasPass+=1
    elif x=='No Qualification' and y==1:
        NPass+=1
HPass/sum(trainee['education'] == 'High School Diploma')     0.71
MatPass/sum(trainee['education'] == 'Matriculation')         0.64
BPass/sum(trainee['education'] == 'Bachelors')               0.74
MasPass/sum(trainee['education'] == 'Masters')               0.86
NPass/sum(trainee['education'] == 'No Qualification')        0.58
'''
```

```
trainee.loc[trainee['education'] == 'High School Diploma', "education"] = 7
trainee.loc[trainee['education'] == 'Matriculation', "education"] = 6
trainee.loc[trainee['education'] == 'Bachelors', "education"] = 7
trainee.loc[trainee['education'] == 'Masters', "education"] = 9
trainee.loc[trainee['education'] == 'No Qualification', "education"] = 6
```

因为 is_handicapper 中残疾的通过率为 0，因此进行赋值

```
trainee.loc[trainee['is_handicapped'] == 'Y', "is_handicapped"] = 0
trainee.loc[trainee['is_handicapped'] == 'N', "is_handicapped"] = 1
```

建立模型：

线性回归:调库并引用，建立线性回归模型，预测结果与实际结果比较获得准确率

```
from sklearn.linear_model import LinearRegression # 导入线性回归的类，采用二分类进行分类预测
from sklearn.model_selection import KFold # K折交叉验证，取平均，调参

predictors = ["age","program_id","test_type","difficulty_level","education","is_handicapped","city_tier","total_programs_enrolled","trainee_
LiR = LinearRegression() # 初始化线性回归类
kf = KFold() # KFold类实例化

predictions = []

LiR.fit(trainee[predictors], trainee["is_pass"]) # 训练模型
predictions.append(LiR.predict(trainee[predictors]))

predictions
```

```
LinearRegression()
```

```
[array([0.52001997, 0.5754585 , 0.76454766, ..., 0.78708482, 0.75237387,
        0.78891639])]
```

```
predictions = np.concatenate(predictions, axis=0) # 转换成数组，才能比较大小

# 使用线性回归得到的结果是在区间[0,1]上的某个值，需要将该值转换成0或1
predictions[predictions >= 0.5] = 1
predictions[predictions < 0.5] = 0

print("测试数据的总数量：", len(predictions))
print("正确的数量：", sum(predictions == trainee["is_pass"]))
accuracy = sum(predictions == trainee["is_pass"]) / len(predictions)
print("准确率为：", accuracy)
```

```
测试数据的总数量： 41468
正确的数量： 29429
准确率为： 0.7096797530626024
```

逻辑回归:调库并引用，建立逻辑回归模型，预测结果与实际结果比较获得准确率

```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression

LoR = LogisticRegression(random_state=1, solver='liblinear') # 初始化逻辑回归类

# 逻辑回归交叉验证
score = model_selection.cross_val_score(LoR, trainee[predictors], trainee["is_pass"], cv=3)
print("准确率为：", score.mean())
```

```
准确率为： 0.7118500363961332
```

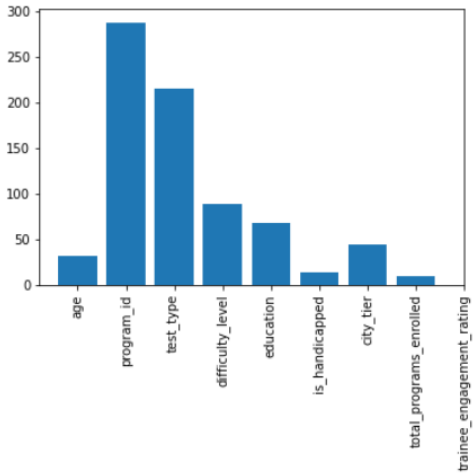
梯度提升:通过调整 max_depth 参数得到最大准确率的模型

```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(random_state=1, n_estimators=50, max_depth=18)

predictions = []

gbc.fit(trainee[predictors], trainee["is_pass"])
predictions.append(gbc.predict(trainee[predictors]))

predictions = np.concatenate(predictions, axis=0) # 转换成数组，才能比较大小

# 使用线性回归得到的结果是在区间[0,1]上的某个值，需要将该值转换成0或1
predictions[predictions >= 0.5] = 1
predictions[predictions < 0.5] = 0

print("测试数据的总数量：", len(predictions))
print("正确的数量：", sum(predictions == trainee["is_pass"]))
accuracy = sum(predictions == trainee["is_pass"]) / len(predictions)
print("准确率为：", accuracy)
```

```
GradientBoostingClassifier(max_depth=17, n_estimators=50, random_state=1)
```

```
测试数据的总数量： 41468
正确的数量： 33577
准确率为： 0.8097086910388733
```

特征选择：




```
from sklearn.ensemble import RandomForestClassifier

predictors = ["program_id","test_type","difficulty_level","education","city_tier","trainee_engagement_rating"]

rfc = RandomForestClassifier()

scores = model_selection.cross_val_score(rfc, trainee[predictors], trainee["is_pass"], cv=kf)
print("随机森林模型的准确率：" + str(scores.mean()))
```

```
随机森林模型的准确率：0.7128628241906163
```

模型选择:

根据不同模型准确率，来将准确率最高的模型应用在测试集中。