廣東工業大學

QG 中期考核详细报告书

题目 数据挖掘竞赛

学 院 计算机学院

专 业 计算机类

年级班别 20 级 7 班

学 号 3220004957

学生姓名 马丽婷

2021 年 4 月 15 日

数据描述

您的客户是一家跨国金融公司，向消费者提供多种产品。尽管主要的贡献来自线下分销渠道，但是有多个渠道可以为消费者提供这些产品。离线渠道通过其代理商网络向消费者出售金融产品，并且根据政府规定，这些代理商必须获得销售金融产品的认证。针对不同类别的金融产品，有多种认证计划。

由于这个离线渠道对公司的总销售额贡献很大，因此公司着重于招聘并认证他们以建立大型代理商网络。在这里，主要的挑战是培训他们获得销售各种类型产品的认证。

在多个程序中，您将获得针对培训课程测试明智的学员绩效数据集。您的任务是根据人口统计信息和培训计划/测试详细信息来预测此类测试的性能。通过找出最重要的因素来提高受训者的参与度和表现，这将使您的客户加强其培训问题。

数据特征

- 变量-说明
- id_num-唯一 ID
- program_id-程序的 ID
- program_type-程序类型
- program_duration-计划持续时间（天）
- test_id-测试 ID
- test_type-测试类型（离线/在线）
- difficulty_level-测试难度级别
- trainee_id-学员的 ID
- gender-受训者性别
- education-学员的教育水平
- city_tier-实习生居住城市的等级
- age-受训者年龄
- total*programs*enrolled(总计划招收)-总课程的学生通过实习
- is_handicapped-受训者是否患有残疾？
- trainee*engagement*rating(学员参与度)-讲师/教学助理为课程提供学员参与度
- is_pass 0-测试失败， 1-测试通过

前言：仔细阅读数据描述，并结合具体的数据特征来看，可以得出以下结论。现有想要成为某公司产品销售代理商的人若干，他们要想获得出售资格，需得通过多个程序接受相关的销售培训并获得相关认证。我们要做的就是，根据给出的包含受训者的基本人口统计信息以及培训详细信息的数据集，通过建立、训练模型，预测出受训者的培训结果，与真实的结果进行比较，来找出影响受训者受训者通过培训的最重要因素，即找出权重相对较大的特征，对受训结果进行二分类的预测。

凭借常识与经验，我先初步挑选出了几个个人认为相对来说更为重要的特征，程序类型、计划持续天数、测试类型、难度级别、受训者性别、教育水平、受训者年龄、学员参与度。

以下为我进行数据挖掘的全过程。

一、数据加载与展示

导入训练集后，我先用 pandas 的 info 和 describe 函数查看了数据集的基本信息。

```
training.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49998 entries, 0 to 49997
Data columns (total 16 columns):
id_num                      49998 non-null object
program_type                49267 non-null object
program_id                  49299 non-null object
program_duration            49323 non-null float64
test_id                     49273 non-null float64
test_type                   49296 non-null object
difficulty_level            49295 non-null object
trainee_id                  49259 non-null float64
gender                      49291 non-null object
education                   49296 non-null object
city_tier                   49298 non-null float64
age                         30619 non-null float64
total_programs_enrolled     49306 non-null float64
is_handicapped              49280 non-null object
trainee_engagement_rating   49226 non-null float64
is_pass                     49998 non-null int64
dtypes: float64(7), int64(1), object(8)
memory usage: 6.1+ MB
```

数据集基本信息

info 函数打印出了训练集的简要概要，近 5 万的数据集，共 16 种数据特征，数据类型有数值型和类别型，具体有 dataframe 的 object 型、float64 和 int64，除唯一的 id_num 外的每一列数据都有不同的缺失。

```
training.describe()
```

	program_duration	test_id	trainee_id	city_tier	age	total_programs_enrolled	trainee_engagement_rating	is_pass
count	49323.000000	49273.000000	49259.000000	49298.000000	30619.000000	49306.000000	49226.000000	49998.000000
mean	128.229366	91.414345	9863.493128	2.249097	36.514256	2.583114	2.397818	0.696288
std	6.889967	51.307852	5716.490640	1.010896	9.045487	1.239399	1.326378	0.459864
min	117.000000	0.000000	1.000000	1.000000	17.000000	1.000000	1.000000	0.000000
25%	121.000000	45.000000	5051.500000	1.000000	28.000000	2.000000	1.000000	0.000000
50%	131.000000	91.000000	9665.000000	2.000000	40.000000	2.000000	2.000000	1.000000
75%	134.000000	135.000000	14618.000000	3.000000	45.000000	3.000000	4.000000	1.000000
max	136.000000	187.000000	20097.000000	4.000000	63.000000	14.000000	5.000000	1.000000

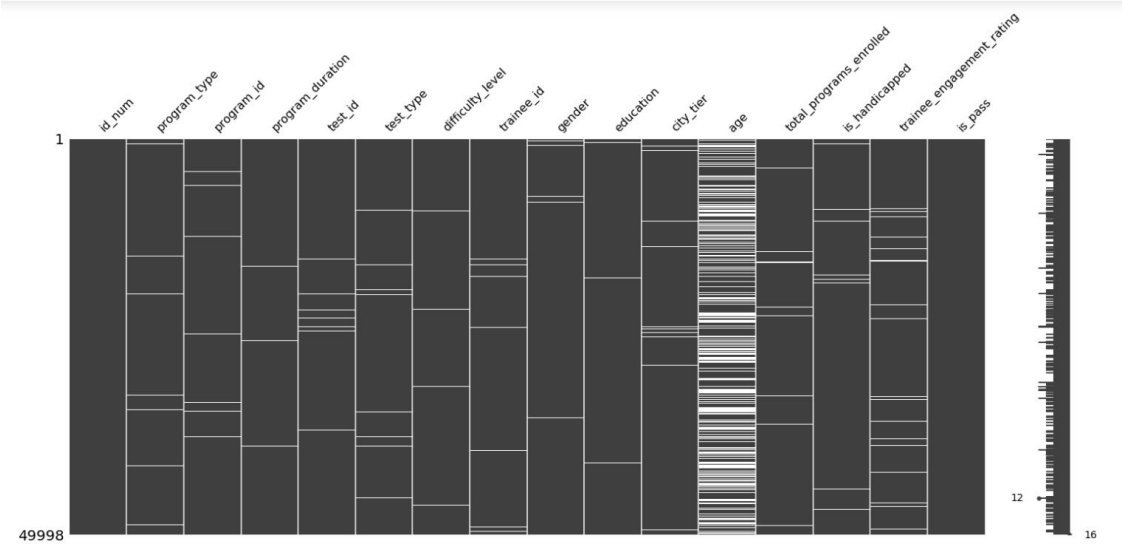
describe 函数生成了仅限于数值型数据的描述性的统计信息，包括数据数量、均值、标准差、最大值、最小值、分位数。可以看到 trainee*id*、*program*duration 均值与其他数据相差很大，这提示我们接下来在数据预处理阶段要对其进行处理。

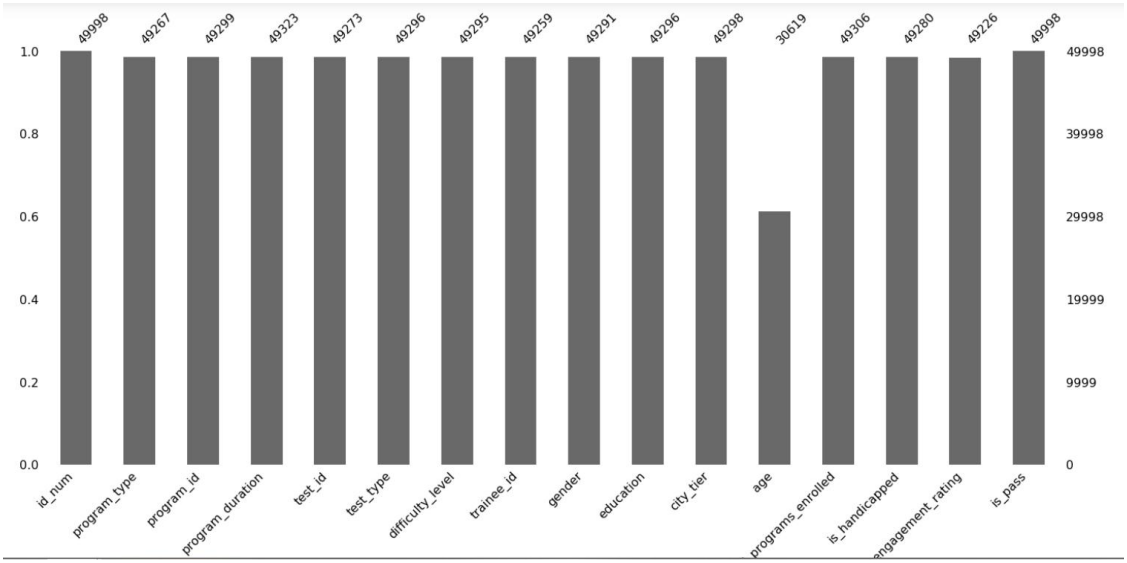
二、数据预处理

在数据预处理阶段，我进行了缺失值处理、下采样、数据编码、标准化。其中，因为我一开始是选用线性回归的二分类进行预测，标准化是在我打算选择逻辑回归再次进行预测时添加的一步，下采样是在我训练逻辑回归模型后，发现过拟合，查看原因与解决办法时发现而后添加的。

1）缺失值处理

使用了 missingno 的 matrix 函数和 bar 函数将缺失值可视化。





图片显示，除 id*num 代表每个人*、*is*pass 作为标签，没有缺失值外，其余特征均有不同程度的缺失值，且与数据集容量相比，仅 age 特征缺失较多，其余有少量缺失值，再看到测试集中也仅 age 特征有大量缺失值，那么我就只填充了 age 特征的缺失值，其余特征的缺失值用

drop 函数删去。对 age 特征，用于填充的是该特征的均值。

2）欠采样

```
1    28889
0    12579
Name: is_pass, dtype: int64
```

is_pass

通过 value_counts 函数可以看到，正样本 1 在总数据集中占到了近 70%，正负样本数不均衡，此时运用机器学习的分类算法预测模型，可能会导致模型更倾向于多数集，无法做出准确预测，又由于数据量还算大。所以可以采用欠采样解决数据不平衡问题。但由于若采用欠采样法随机丢弃正例，可能会丢失一些重要信息，那么可以采用 EasyEnsemble 算法，利用集成学习机制，将正例划分为若干个集合供不同学习器使用，这样对每个学习器来看都进行了欠采样，但在全局来看却不会丢失重要信息。对于类别不平衡的时候采用 CV 进行交叉验证时，由于分类问题在目标分布上表现出很大的不平衡性。如果用 sklearn 库中的函数进行交叉验证的话，建议采用如 StratifiedKFold 和 StratifiedShuffleSplit 中实现的分层抽样方法，确保相对的类别概率在每个训练和验证折叠中大致保留。但是目前我还只是学习了相关算法，还未实用来测试数据。

3）数据编码

一开始我是采用的 pandas 的 get_dummies 函数来对所有字符串型的列进行独热编码的，但由于独热编码是将类别变量转换成了新增的虚拟变量/指示变量，扩充了特征空间，使得数据量很大，出现 memoryerror。因此我直接将特征的类别变量用 0、1 等来编码。

但有一个特殊的特征 program*id*，*类别过多不好进行编码，随便编码可能会导致增添错误信息，因此我没有对特征 program*id 进行编码，而是将它从数据集中删去。

编码如图：

```
# 对difficulty_level进行独热编码
training.difficulty_level.value_counts()
training.loc[training['difficulty_level'] == 'easy', "difficulty_level"] = 0
training.loc[training['difficulty_level'] == 'intermediate', "difficulty_level"] = 1
training.loc[training['difficulty_level'] == 'hard', "difficulty_level"] = 2
training.loc[training['difficulty_level'] == 'vary hard', "difficulty_level"] = 3

# 处理test_type特征
training.test_type.unique()
training.test_type.value_counts()
training.loc[training.test_type == 'offline', "test_type"] = 0
training.loc[training.test_type == 'online', "test_type"] = 1
training.test_type.unique()

# 处理gender特征
training.gender.unique()
training.gender.value_counts()
training.loc[training.gender == 'M', "gender"] = 0
training.loc[training.gender == 'F', "gender"] = 1
training.gender.unique()

# 处理education特征
training.education.unique()
training.education.value_counts()
training.loc[training.education == 'No Qualification', "education"] = 0
training.loc[training.education == 'Matriculation', "education"] = 1
training.loc[training.education == 'High School Diploma', "education"] = 2
training.loc[training.education == 'Bachelors', "education"] = 3
training.loc[training.education == 'Masters', "education"] = 4
```

独热 1

```
# 独热编码program_type特征
training.program_type.value_counts()
training.program_type.describe()
training.loc[training.program_type == 'Y', "program_type"] = 0
training.loc[training.program_type == 'T', "program_type"] = 1
training.loc[training.program_type == 'V', "program_type"] = 2
training.loc[training.program_type == 'U', "program_type"] = 3
training.loc[training.program_type == 'Z', "program_type"] = 4
training.loc[training.program_type == 'X', "program_type"] = 5
training.loc[training.program_type == 'S', "program_type"] = 6

# 独热编码is_handicapped特征
training.is_handicapped.unique()
training.is_handicapped.value_counts()
training.loc[training.is_handicapped == 'Y', "is_handicapped"] = 0
training.loc[training.is_handicapped == 'N', "is_handicapped"] = 1
```

独热 2

4）标准化

	program_type	program_duration	test_id	test_type	difficulty_level	trainee_id	gender	education	city_tier	age
count	41468.000000	41468.000000	41468.000000	41468.000000	41468.000000	41468.000000	41468.000000	41468.000000	41468.000000	41468.000000
mean	1.630052	128.212405	91.400719	0.403516	0.650984	9877.189278	0.466311	1.800135	2.249687	36.530317
std	1.585602	6.889023	51.231736	0.490608	0.823892	5720.148060	0.498870	0.729533	1.011196	7.076440
min	0.000000	117.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.000000	17.000000
25%	0.000000	121.000000	45.000000	0.000000	0.000000	5065.000000	0.000000	1.000000	1.000000	31.000000
50%	1.000000	131.000000	91.000000	0.000000	0.000000	9683.000000	0.000000	2.000000	2.000000	36.514256
75%	3.000000	134.000000	135.000000	1.000000	1.000000	14637.750000	1.000000	2.000000	3.000000	43.000000
max	6.000000	136.000000	187.000000	1.000000	3.000000	20097.000000	1.000000	4.000000	4.000000	63.000000

由于编码后数据特征之间仍存在分布产生的度量偏差，但为了更好的保持样本间距，因此有必要对数据进行标准化处理，基于数据特征服从正态分布这一隐含假设，将这个正态分布调整为均值为 0，方差为 1 的标准正态分布。我采用的是 Z-Score 标准化的标准化方法，先减去均值再除以标准差。标准化后可以更加容易地得出最优参数 w 和 b 以及计算出损失函数 J(w, b)的最小值，从而达到加速收敛的效果。

但需要注意的是，用于区别各受训者的特征 id*num 和作为标签的特征 is*pass，不可以一起进行标准化处理，因此我将 is*pass 单独留了出来，把 id*num 用 drop 函数排除在外。还有一点就是，采用 sklearn 库中的 preprocessing.StandardScaler()进行标准化时，数据集应先转化为数组再传入，标准化完成后再转回 dataframe 类型。

部分数据标准化结果如图：

	program_type	program_duration	test_id	test_type	difficulty_level	trainee_id	gender	education	city_tier	age
count	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04
mean	2.909958e-16	-5.300895e-16	1.020092e-16	-5.246545e-16	-6.286570e-17	1.081683e-16	5.342982e-16	4.845004e-16	1.950119e-16	1.414324e-15
std	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00
min	-1.028046e+00	-1.627595e+00	-1.784086e+00	-8.224906e-01	-7.901425e-01	-1.726583e+00	-9.347469e-01	-2.467546e+00	-1.235865e+00	-2.759940e+00
25%	-1.028046e+00	-1.046954e+00	-9.057136e-01	-8.224906e-01	-7.901425e-01	-8.412802e-01	-9.347469e-01	-1.096790e+00	-1.235865e+00	-7.815205e-01
50%	-3.973630e-01	4.046479e-01	-7.821782e-03	-8.224906e-01	-7.901425e-01	-3.394871e-02	-9.347469e-01	2.739661e-01	-2.469249e-01	-2.269667e-03
75%	8.640025e-01	8.401286e-01	8.510312e-01	1.215819e+00	4.236241e-01	8.322544e-01	1.069808e+00	2.739661e-01	7.420148e-01	9.142677e-01
max	2.756051e+00	1.130449e+00	1.866039e+00	1.215819e+00	2.851157e+00	1.786655e+00	1.069808e+00	3.015478e+00	1.730954e+00	3.740582e+00

三、特征工程

特征选择

当数据预处理完成后，由于特征有 13 个那么多，我们需要选择有意义的特征输入机器学习的算法和模型进行训练，以达到减少特征数量、降维，使模型泛化能力更强，减少过拟合的效果。

我采用的是 Embedded 嵌入法中的使用 SelectFromModel 选择特征，先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。有些机器学习方法本身就具有对特征进行打分的机制，很容易将其运用到特征选择任务中，例如回归模型。我选用了基于 L1 的特征选择，使用 L1 范数作为惩罚项的线性模型会产生稀疏解：大

部分特征对应的系数为 0。当想减少数据维度后再分类时，可以利用 feature*selection.SelectFromModel 选择系数非 0 的特征。对于 SVM 和逻辑回归，参数 C 控制稀疏性：C 越小，被选中的特征越少。还可以使用 feature*selection 库的 SelectFromModel 类结合带 L1 以及 L2 惩罚项的逻辑回归模型进行特征选择，不断调节参数 C，可以得到和基于 L1 的特征选择一样的结果。

通过将参数 C 调到 0.0003，对比前面的统计数据描述可以对比出，选出了以下 5 个特征："test*type", "difficulty*level", "education", "city*tier", "trainee*engagement_rating"。如图：

```
from sklearn.feature_selection import SelectFromModel
X,y = training, is_pass
X.shape

lsvc = LinearSVC(C=0.0003, penalty="l1", dual=False).fit(X, y)
model = SelectFromModel(lsvc, prefit=True)
X_new1 = model.transform(X) # X_new为数组
X_new1.shape
X_new1 = pd.DataFrame(X_new1)
X_new1.describe()
```

```
(41468, 13)
```

```
(41468, 5)
```

	0	1	2	3	4
count	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04	4.146800e+04
mean	-5.246545e-16	-6.286570e-17	4.845004e-16	1.950119e-16	7.723370e-16
std	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00
min	-8.224906e-01	-7.901425e-01	-2.467546e+00	-1.235865e+00	-1.054995e+00
25%	-8.224906e-01	-7.901425e-01	-1.096790e+00	-1.235865e+00	-1.054995e+00
50%	-8.224906e-01	-7.901425e-01	2.739661e-01	-2.469249e-01	-3.000981e-01
75%	1.215819e+00	4.236241e-01	2.739661e-01	7.420148e-01	1.209695e+00
max	1.215819e+00	2.851157e+00	3.015478e+00	1.730954e+00	1.964591e+00

特征选择

四、模型选择与评估

1）线性回归

在编码后，我首先采取的是线性回归的二分类进行分类预测并结合 K 折交叉验证。

从训练数据集中分出一部分做为验证数据，不参与训练，但用于评估模型的训练效果，可以相对客观的评估模型对于训练集之外数据的匹配程度，提高泛化能力。模型在验证数据中的评估常用的是交叉验证，又称循环验证。它将原始数据分成 K 组(K-Fold)，将每个子集数据分别做一次验证集，其余的 K-1 组子集数据作为训练集，这样会得到 K 个模型。这 K 个模

型分别在验证集中评估结果，最后的误差 MSE(Mean Squared Error)加和平均就得到交叉验证误差。交叉验证有效利用了有限的数据，并且评估结果能够尽可能接近模型在测试集上的表现，可以做为模型优化的指标使用。我将训练集划分为 7 块，并且打乱划分，固定随机起点，弱化数据不平衡的作用，使分块中会分到少的反样本。

调用 sklearn 库的 LinearRegression 模型进行训练，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，结果>0.5 标记为 1，结果<=0.5 为 0。

我最终选择的是 "program*type*", "*program*duration", "test*type*", "*difficulty*level", "gender", "education", "age", "trainee*engagement*rating"这八个特征，得到 0.68593 的准确率。在前面选择了特征选择选择出来的五个特征进行训练，准确率比八个特征会降低 0.1 多。

2）逻辑回归

首先还是将原始数据按照比例分割为“测试集”和“训练集”，这里采用的是 sklearn.model*selection 中的 train*test_split 函数。逻辑回归模型选用特征选择出来的五个特征，放入进行训练。超参数 C 为正则化系数λ的倒数，必须为正数，默认为 1，值越小，代表正则化越强。正则化采用 L2，与前面使用的标准化结合，防止过拟合。

因为训练出来的准确率为 1，放入测试集测试，未使用标准化前，准确率仅为 0.4 多，使用后也仅提升了 0.1 多，明显过拟合了。

查找过拟合原因和解决办法时，想到因为特征不止一个，我就想使用多项式回归，结果因为数据量太大，会出现 memoryerror。再然后就是想到了数据不平衡的问题。

五、心得与体会

尽管花了很长时间还是完成得还不是很好，但是还是学习到了很多新的东西，收获到了很多。但仍存在学习过程中太过关注细枝末节、不能很好的识别网络上泛滥的信息、学习效率低等问题。这次中期考核，让我深刻认识到了自己的知识储备量还不够，希望自己能继续努力，一直不断地学习，充实提升自己。