廣東工業大學

QG 中期考核详细报告书

题　　目 QG 中期考核详细报告书

学　　院 计算机学院

专　　业 计算机类

年级班别 20 级计算机类 12 班

学　　号 3220005188

学生姓名 聂芳琪

2020 年 4 月 15 日

前言

在接到这个任务后的第二晚，我开始了对数据挖掘竞赛的了解，Datawhale 上的大佬有一大堆的总结，说了好多，但是我在脑海里只记住了两句话，（这里不是说大佬讲的太多，纯粹是我没记住）：

1. 代码可以复制的就不自己敲

2. 解决问题远比证明自己更加优秀

听起来真的真的真的好高级，尤其是第一点！（不过由于自己编程太菜，我还是手动输入吧，自觉）于是乎，我就上 kaggle，找了一篇预测挺高分的泰坦尼克号的新手文章，打算用一个上午的时间研读一番，但是，由于自己的英文水平，以及 python 的水平远远不及大佬，超级菜的聂某人，非常自觉地离开了这篇文章。转战知乎，找了另一篇数据挖掘实战分享笔记，是中文版。

这里附上一份链接：《Kaggle Titanic 生存预测 -- 详细流程吐血梳理》https://zhuanlan.zhihu.com/p/31743196

我真的非常非常非常感谢这位大佬的“吐血”梳理以及感谢 Datawhale 上大佬的无私奉献，让我对数据挖掘竞赛的流程有了基本的认识，以及大概的操作。而我也正式从一名萌新从入门到入坑。当我看到通过自己的处理数据，调整参数，设置特征……使得预测值不断上升的时候，开心！这两个字足以形容，这一次 QG 训练营之后，我想继续在 kaggle 玩儿，然后呢，还要提高自己的英文水平（残忍现实）

还有一点，一定要经常在 QG 群里看看，在做完大部分工作之后才知道原来继元师兄已经提供给我们范例了，新手友好款。

前言大概差不多是这些了，感谢 QG 工作室的师兄非常有耐心的看完，万分感谢。

一、赛题分析

数据介绍简要概括

数据集

针对培训课程测试明智的学员绩效数据集

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。

目的

通过找出最重要的因素来提高受训者的参与度和表现，这将使该公司加强其培训问题

描述性特征

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 总课程的学生通过实习

is_handicapped 受训者是否患有残疾?

trainee_engagement_rating 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

二、流程

第一阶段：专注于数据预处理和探索

第二阶段：模型的建立，验证和预测

三、第一阶段

数据探索

对比，对比训练集不同样本之间的特征分布

分组，到按类别标签、某个离散变量的不同取值 groupby 后的 sum、unique。

抓大放小，就是对于那些特征重要性较高的变量，要做重点分析。因为这些变量对你模型预测能力的影响是较大的。

可视化，图形给人的冲击力往往是要大于数字本身的。

1、读取并复制数据

可能后来还会再用来分析，不过我可能不会。

2、数据整体认知

研究训练集、测试集、记录数、变量数、变量类型、变量属性值、标签等内容。

In [193]:
```
train.dtypes.to_frame().rename(columns={0:'Column type'})
```

Out[193]:

	Column type
id_num	object
program_type	int64
program_id	object
program_duration	float64
test_id	float64
test_type	int64
difficulty_level	object
trainee_id	float64
gender	int64
education	int64
city_tier	float64
age	float64
total_programs_enrolled	float64
is_handicapped	int64
trainee_engagement_rating	float64
is_pass	int64

```
train.info()
print('_'*40)
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 46504 entries, 0 to 49997
Data columns (total 16 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   id_num                     46504 non-null  object 
 1   program_type               46504 non-null  int64  
 2   program_id                 46504 non-null  object 
 3   program_duration           46504 non-null  float64
 4   test_id                    46504 non-null  float64
 5   test_type                  46504 non-null  int64  
 6   difficulty_level           46504 non-null  object 
 7   trainee_id                 46504 non-null  float64
 8   gender                     46504 non-null  int64  
 9   education                  46504 non-null  int64  
 10  city_tier                  46504 non-null  float64
 11  age                        46504 non-null  float64
 12  total_programs_enrolled    46504 non-null  float64
 13  is_handicapped             46504 non-null  int64  
 14  trainee_engagement_rating  46504 non-null  float64
 15  is_pass                    46504 non-null  int64  
dtypes: float64(7), int64(6), object(3)
memory usage: 6.0+ MB
```

```
_____________________________________
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11684 entries, 0 to 11683
Data columns (total 16 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id_num                      11684 non-null  object
 1   program_type                11684 non-null  int64
 2   program_id                  11684 non-null  object
 3   program_duration            11684 non-null  int64
 4   test_id                     11684 non-null  int64
 5   test_type                   11684 non-null  int64
 6   difficulty_level            11684 non-null  object
 7   trainee_id                  11684 non-null  int64
 8   gender                      11684 non-null  int64
 9   education                   11684 non-null  int64
 10  city_tier                   11684 non-null  int64
 11  age                         11684 non-null  float64
 12  total_programs_enrolled     11684 non-null  int64
 13  is_handicapped              11684 non-null  int64
 14  trainee_engagement_rating   11684 non-null  float64
 15  is_pass                     0 non-null      float64
dtypes: float64(3), int64(10), object(3)
memory usage: 1.4+ MB
```

3、数据质量分析

分析数据的缺失值,异常值等特性。

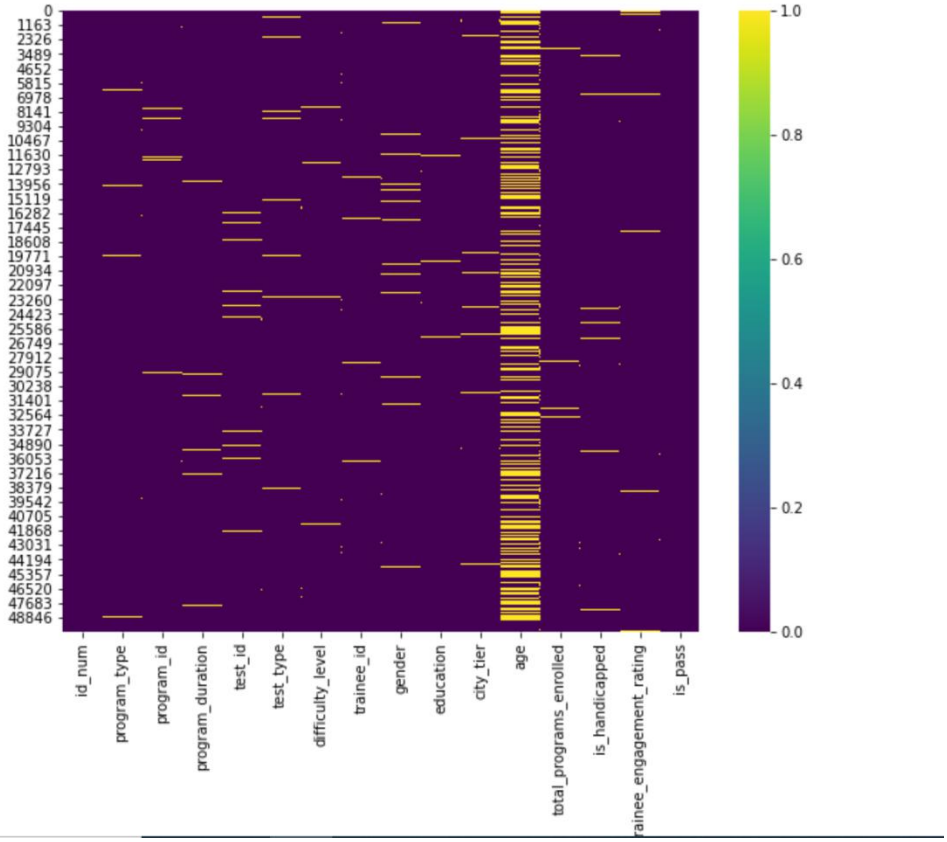
In [204]:
```
#检查是否有缺失值
train.isnull().sum().to_frame().rename(columns={0:'Null values'})
```

Out[204]:

	Null values
id_num	0
program_type	731
program_id	699
program_duration	675
test_id	725
test_type	702
difficulty_level	703
trainee_id	739
gender	707
education	702
city_tier	700
age	19379
total_programs_enrolled	692
is_handicapped	718
trainee_engagement_rating	772
is_pass	0

In [62]:
```
#可视化所有的缺失值
plt.figure(figsize=(10,8));
sns.heatmap(train.isnull(), cmap='viridis')
```

Out[62]: <AxesSubplot:>



检查有无异常值

In [206]:
```
train.describe()
```

Out[206]:

	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

In [207]:
```
test.describe()
```

Out[207]:

	program_duration	test_id	trainee_id	city_tier	age	total_programs_enrolled	trainee_engagement_rating	is_pass
count	11684.000000	11684.000000	11684.000000	11684.000000	7284.000000	11684.000000	11673.000000	0.0
mean	128.147723	90.689404	9784.998203	2.250171	36.561642	2.588069	2.394500	NaN
std	6.888674	51.016147	5690.862283	1.015069	9.075222	1.249099	1.330828	NaN
min	117.000000	0.000000	3.000000	1.000000	17.000000	1.000000	1.000000	NaN
25%	121.000000	45.000000	4987.500000	1.000000	28.000000	2.000000	1.000000	NaN
50%	131.000000	89.000000	9586.000000	2.000000	40.000000	2.000000	2.000000	NaN
75%	134.000000	135.000000	14428.000000	3.000000	45.000000	3.000000	4.000000	NaN
max	136.000000	187.000000	20098.000000	4.000000	63.000000	12.000000	5.000000	NaN

可见没有异常值

数据清洗

4、缺失值填补与删除

经过大约计算在 50000 份数据里 800 份数据大概占比 0.016，所以 difficulty_level 、 education 、 city_tier 、 test_type 、 program_duration、total_programs_enrolled、is_handicapped、trainee_engagement_rating 这些列的缺失值用众数填充；

age 缺失值较多，用平均值填充

其他不太好暴力填充的缺失值，则直接删除。

测试集采用相同的办法。

查看一下是否还有缺失值

5、数据清洗小结

- 清洗数据：
  - 缺失值
  - 重复值
  - 异常值
- 清洗结果：
  - 清洗结果：以上三种数据均进行了删除处理，填补处理

- 清洗数据占比：占比约 7%

清洗前后数据量变化小于 1.6%（指删除处理部分），因此可以忽略对结果的影响。

	Null values
id_num	0
program_type	0
program_id	0
program_duration	0
test_id	0
test_type	0
difficulty_level	0
trainee_id	0
gender	0
education	0
city_tier	0
age	0
total_programs_enrolled	0
is_handicapped	0
trainee_engagement_rating	0
is_pass	0

反思：这里的数据处理过于简单，比如 age 是缺失值最多的，所以保证一定的缺失值填充准确率是非常重要的，对结果也会产生较大的影响。如果我能使用回归，随机森林模型预测缺失值则会很大地提高填充数地准确度，下一次做数据处理的时候，希望我可以运用到。

6、数据统计量分析

第一，对单个变量的统计分析

考察单个变量的均值、中位数、众数、分位数、方差、变异系数等。

In [206]:
```
train.describe()
```

Out[206]:

	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

In [207]:
```
test.describe()
```

Out[207]:

	program_duration	test_id	trainee_id	city_tier	age	total_programs_enrolled	trainee_engagement_rating	is_pass
count	11684.000000	11684.000000	11684.000000	11684.000000	7284.000000	11684.000000	11673.000000	0.0
mean	128.147723	90.689404	9784.998203	2.250171	36.561642	2.588069	2.394500	NaN
std	6.888674	51.016147	5690.862283	1.015069	9.075222	1.249099	1.330828	NaN
min	117.000000	0.000000	3.000000	1.000000	17.000000	1.000000	1.000000	NaN
25%	121.000000	45.000000	4987.500000	1.000000	28.000000	2.000000	1.000000	NaN
50%	131.000000	89.000000	9586.000000	2.000000	40.000000	2.000000	2.000000	NaN
75%	134.000000	135.000000	14428.000000	3.000000	45.000000	3.000000	4.000000	NaN
max	136.000000	187.000000	20098.000000	4.000000	63.000000	12.000000	5.000000	NaN

第二，对两个变量的统计分析

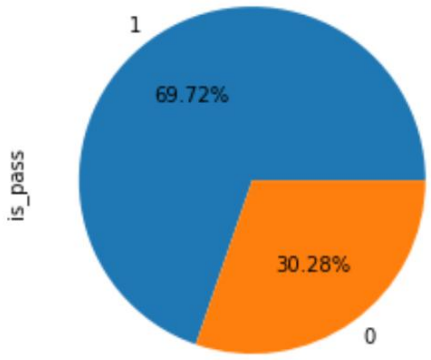
两个变量统计分布之间的关系。

先看看训练集中地通过率

In [229]:
```
#在可视化中使用所有数据
df = pd.concat([train,test], axis=0)

train['is_pass'].value_counts().plot.pie(autopct = '%1.2f%%')
```

Out[229]: <AxesSubplot:ylabel='is_pass'>

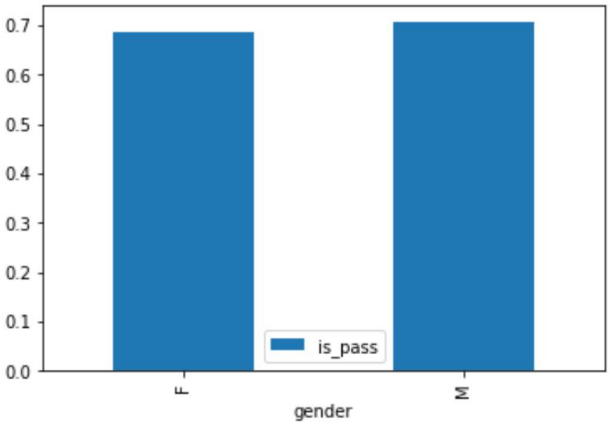


性别与是否通过的关系

可见男女通过率相差不多，但是男生略胜一筹

In [234]:
```
train[['gender','is_pass']].groupby(['gender']).mean().plot.bar()
```

Out[234]: <AxesSubplot:xlabel='gender'>

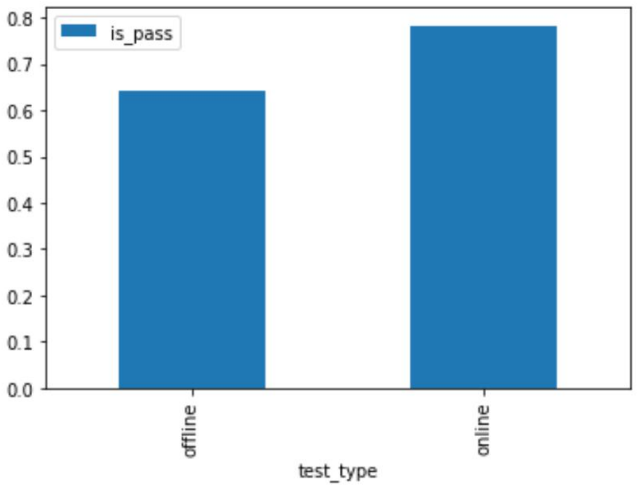


测试形式与是否通过的关系

分析可知线上测试的通过率更高

In [72]:
```
train.groupby(['test_type','is_pass'])['is_pass'].count()
train[['test_type','is_pass']].groupby(['test_type']).mean().plot.bar()
```

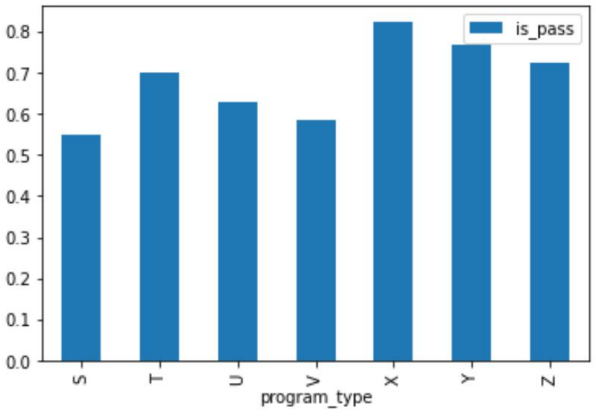
Out[72]: <AxesSubplot:xlabel='test_type'>



程序类型与是否通过的关系

In [73]:
```
train.groupby(['program_type','is_pass'])['is_pass'].count()
train[['program_type','is_pass']].groupby(['program_type']).mean().plot.bar()
```

Out[73]: <AxesSubplot:xlabel='program_type'>

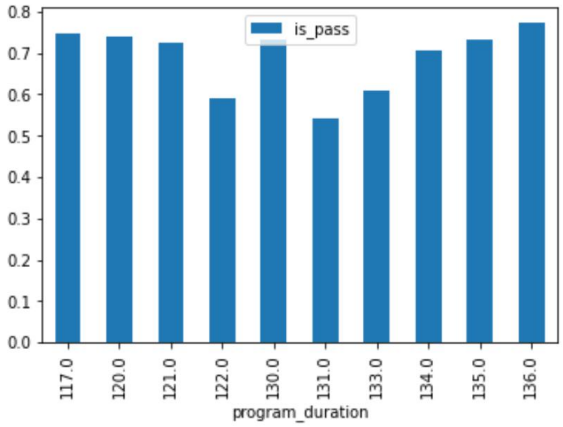


程序类型时 X 的通过率最高，程序类型是 S 的通过率最低

计划持续时间与是否通过的关系

In [74]:
```
train.groupby(['program_duration','is_pass'])['is_pass'].count()
train[['program_duration','is_pass']].groupby(['program_duration']).mean().plot.bar()
```

Out[74]: <AxesSubplot:xlabel='program_duration'>

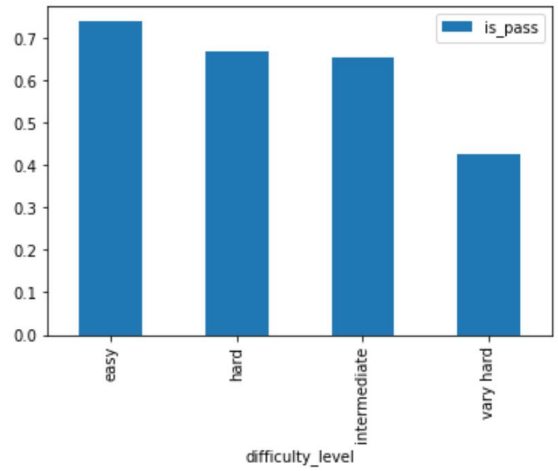


计划时间 136 的通过率最高，计划持续时间为 131 的通过率最低

难度等级与是否通过的关系

In [75]:
```
train.groupby(['difficulty_level','is_pass'])['is_pass'].count()
train[['difficulty_level','is_pass']].groupby(['difficulty_level']).mean().plot.bar()
```

Out[75]: <AxesSubplot:xlabel='difficulty_level'>

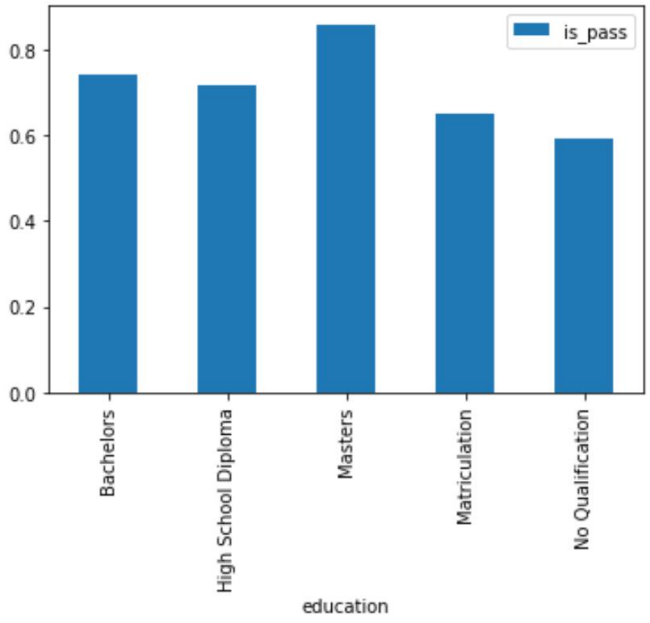


难度等级与通过率成反比例关系

教育与是否通过的关系

In [76]:
```
train.groupby(['education','is_pass'])['is_pass'].count()
train[['education','is_pass']].groupby(['education']).mean().plot.bar()
```

Out[76]: <AxesSubplot:xlabel='education'>

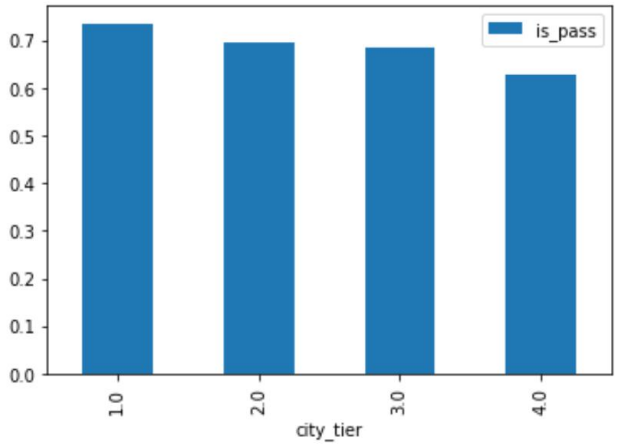


所受教育程度与通过率成正比

城市等级与是否通过的关系

In [77]:
```
train.groupby(['city_tier','is_pass'])['is_pass'].count()
train[['city_tier','is_pass']].groupby(['city_tier']).mean().plot.bar()
```

Out[77]: <AxesSubplot:xlabel='city_tier'>

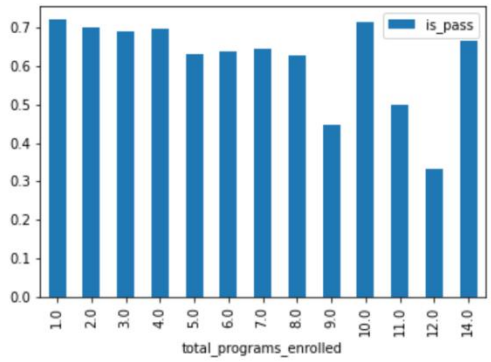


城市等级与通过率成正比

总课程学生通过实习与是否通过的关系

In [78]:
```
train.groupby(['total_programs_enrolled','is_pass'])['is_pass'].count()
train[['total_programs_enrolled','is_pass']].groupby(['total_programs_enrolled']).mean().plot.bar()
```

Out[78]: <AxesSubplot:xlabel='total_programs_enrolled'>

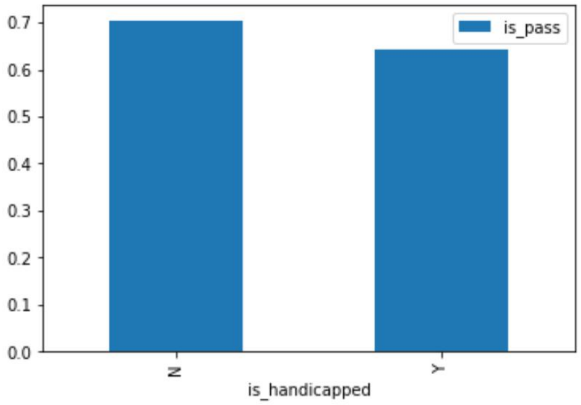


总课程是 12 的通过率最低

是否残疾与是否通过的关系

In [79]:
```
train.groupby(['is_handicapped','is_pass'])['is_pass'].count()
train[['is_handicapped','is_pass']].groupby(['is_handicapped']).mean().plot.bar()
```

Out[79]: <AxesSubplot:xlabel='is_handicapped'>

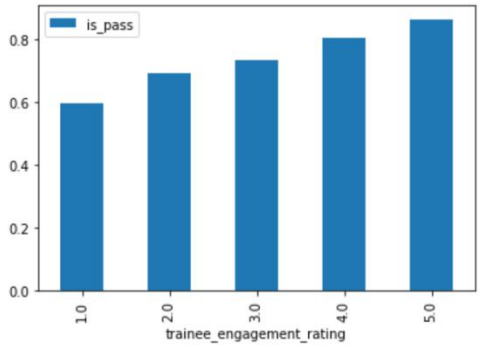


不残疾的通过率较高于残疾人士，不过差别也不明显

讲师/教学助理为课程提供学员参与度

In [80]:
```
train.groupby(['trainee_engagement_rating','is_pass'])['is_pass'].count()
train[['trainee_engagement_rating','is_pass']].groupby(['trainee_engagement_rating']).mean().plot.bar()
```

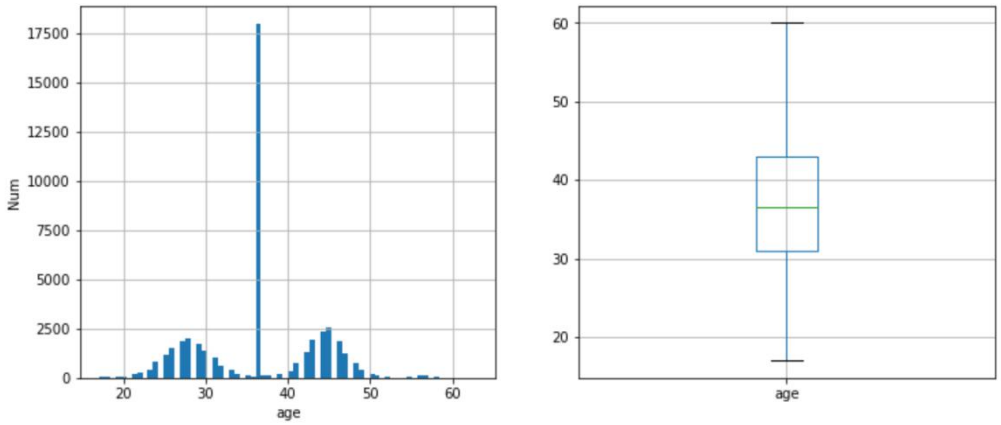
Out[80]: <AxesSubplot:xlabel='trainee_engagement_rating'>



讲师/教学助理为课程提供学员参与度与学生通过率成正比

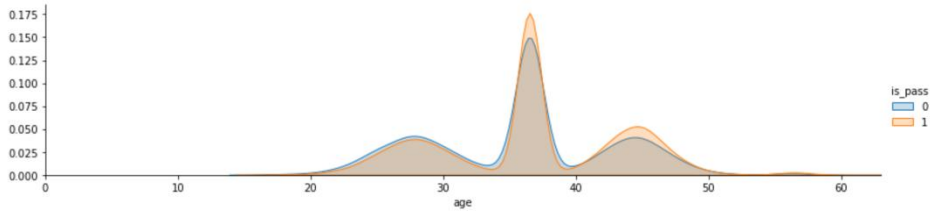
```
#总体的年龄分布
plt.figure(figsize=(12,5))
plt.subplot(121)
train['age'].hist(bins=70)
plt.xlabel('age')
plt.ylabel('Num')

plt.subplot(122)
train.boxplot(column='age', showfliers=False)
plt.show()
```



In [235]:
```
#不同年龄下的通过率分析
facet = sns.FacetGrid(train, hue="is_pass",aspect=4)
facet.map(sns.kdeplot,'age',shade= True)
facet.set(xlim=(0, train['age'].max()))
facet.add_legend()
```

Out[235]: <seaborn.axisgrid.FacetGrid at 0x200baaa6ca0>



第三，对多个变量的统计分析

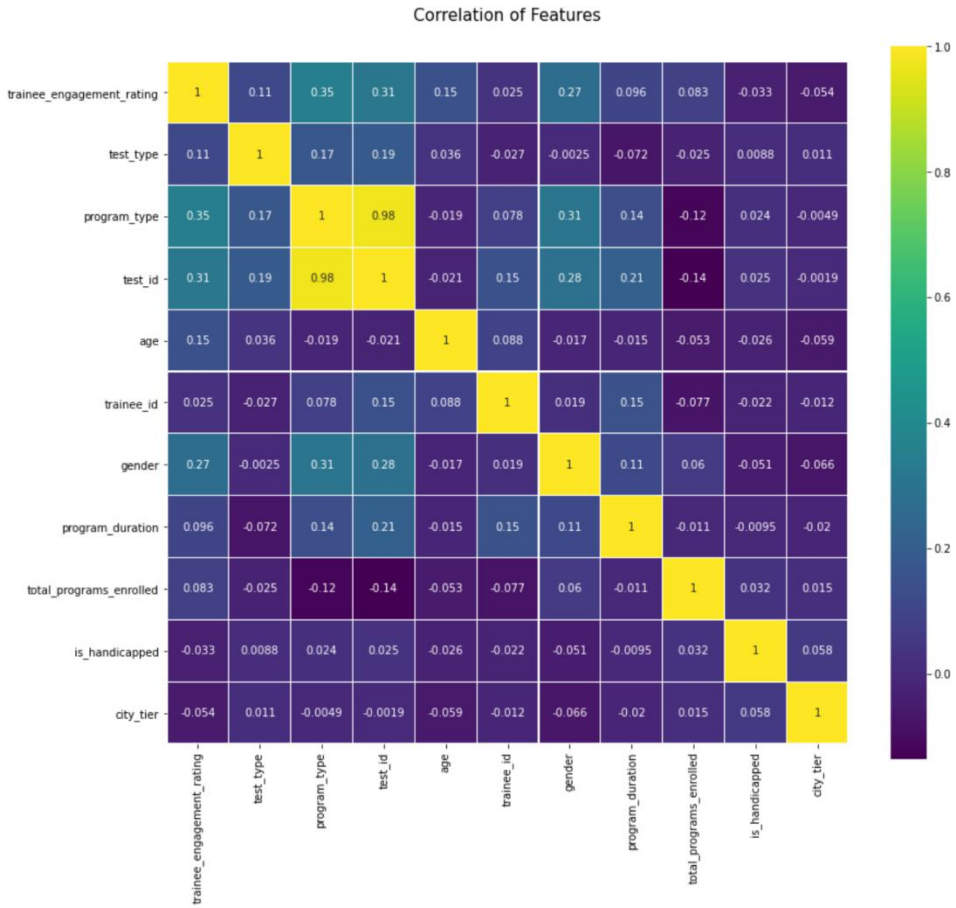
先看看相关性分析

In [270]:
```
corr = train.corr()
corr['is_pass'].sort_values(ascending=False)[1:].to_frame()\
.style.background_gradient(axis=1,cmap=sns.light_palette('green', as_cmap=True))
```

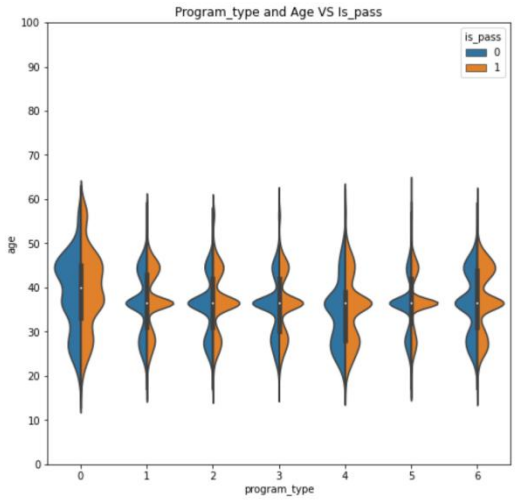
Out[270]:

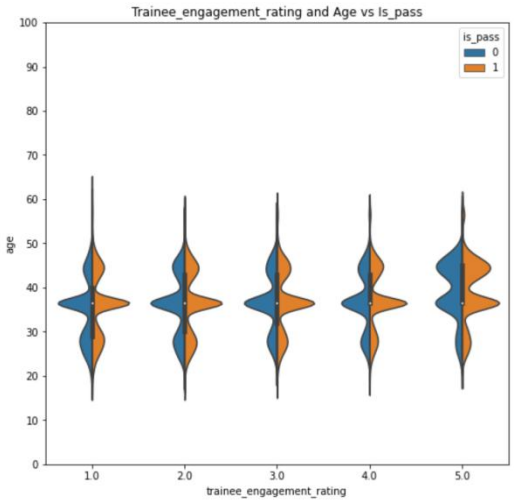
	is_pass
trainee_engagement_rating	0.194157
test_type	0.150572
program_type	0.082905
test_id	0.074343
age	0.060022
trainee_id	0.037678
gender	0.021949
program_duration	0.005361
total_programs_enrolled	-0.031246
is_handicapped	-0.037329
city_tier	-0.065120
education	-0.080005
difficulty_level	-0.115164

相关性可视化



根据相关性做出的多变量之间的关系





反思：多变量之间的相关性可视化还不够。

四、第二阶段

特征工程

做数据竞赛时最重要的一项内容。

1. 输入机器学习模型的数据必须是标准的向量形式。但处理的数据并没有以格式规范的特征向量的形式呈现。呈现的数据是数据库记录、字母、文字等形式，同时还存着在大量的噪声数据。所以需要一定的方法把非结构化的数据转化为结构化的数据。

处理过程：

数据预处理原因

a. 海量原始数据中存在大量信息缺失、不一致、冗余值、异常值等，会影响模型的学习效果。

b. 在用各种模型算法时的监督学习的假设，训练集和测试集样本是独立同分布的。

c. 在模型训练时，数据规范化的操作可以让梯度下降算法收敛得更快，更快地找到最优超参数。

数据转换

a. 首先把数据型的列放在一个列表中，把数据型的列放在另一个列表中

b. 先处理数据型：归一化

做聚类分析的时候，聚类的效果往往特别受其中一列数据的影响，使得原本应该散布在二维平面图上的点，变成聚集在一条线上的点。所以为了避免这个情况，采用数据归一化处理，使数值落在【0，1】之间。

c. 处理非数据型：LabelEncoder　OneHotEncoder

LabelEncoder 是用来对分类型特征值进行编码，即对不连续的数值或文本进行编码。其中包含以下常用方法：

fit(y) ： fit 可看做一本空字典，y 可看作要塞到字典中的词。

fit_transform(y)：相当于先进行 fit 再进行 transform，即把 y 塞到字典中去以后再进行 transform 得到索引值。

inverse_transform(y)：根据索引值 y 获得原始数据。

transform(y) ： 将 y 转变成索引值

OneHotEncoder

对于这个样本，如[" M "，"vary hard "，" N"]，我们需要将这个分类值的特征数字化，最直接的方法，我们可以采用序列化的方式：[0,1,3]。但是这样的特征处理并不能直接放入机器学习算法中。【这里说个题外话，是哪个师兄或者师姐写的数据集？？？我从小到大都没看到过有人把 very 能写成 vary，直到这次，直接去世（）】

对于上述的问题，性别的属性是二维的，困难等级是四维的，是否残疾也是二维得，这样，采用 One-Hot 编码的方式对上述的样本“[" M "，"vary hard "，

" N"]”编码，“M”则对应着[1，0]，同理“vary hard”对应着[0，0，0，1]，“N”对应着[0, 1]。则完整的特征数字化的结果为：[1,0,0,0,0,1,0,1]。

反思和改进

其实数据预处理还包含了很多的部分，数据清洗部分：缺失值的插补有些可以利用最近邻插补，离群值可以采取替换，或者是直接删除，异常，重复的数据直接删除。我首先只考虑到了处理缺失值，这次是比较幸运没有遇到有个性的数据。下一次做一个比赛的时候，希望自己考虑的更加周全。这一次做的比较好的：是数据变换和数据规范化。

模型融合以及测试

这一次的数据预测的特征工程部分我只做了数据的预处理，然后直接通过相关性，探究变量与变量之间的相关性，手动筛选特征，尝试了五种不同的特征组合，尝试了

（1）模型选择

最开始的尝试：使用简单线性回归，knn，逻辑回归和 XGBoost 单个模型来预测，但是效果都不太好。然后学会了一个小技巧，交叉验证.

就简单说一下，如果将数据集分为 10 折，做一次交叉验证，实际上它是计算了十次，将每一折都当做一次测试集，其余九折当做训练集，这样循环十次。通过传入的模型，训练十次，最后将十次结果求平均值。将每个数据集都算一次，然后再用打印出返回值。可以比较方便的知道那哪一个模型预测得更好。然后直接上传最好的那一份 submission。

交叉验证优点：

1：交叉验证用于评估模型的预测性能，尤其是训练好的模型在新数据上的表现，可以在一定程度上减小过拟合。

2：还可以从有限的数据中获取尽可能多的有效信息。

（2）模型优化

优化参数

在机器学习模型中，需要人工选择的参数称为超参数。比如随机森林中决策树的个数，人工神经网络模型中隐藏层层数和每层的节点个数等等，需要事先指定。超参数选择不恰当，就会出现欠拟合或者过拟合的问题。而在选择超参数的时候，有两个途径，一个是凭经验微调（很显然我是没的经验的），另一个就是选择不同大小的参数，带入模型中，挑选表现最好的参数。

微调的一种方法是手工调制超参数，直到找到一个好的超参数组合，这么做的话会非常冗长，所以我使用了 Scikit-Learn 的 GridSearchCV 来做这项搜索工作。

GridSearchCV 网格搜索和交叉验证。网格搜索，搜索的是参数，即在指定的参数范围内，按步长依次调整参数，利用调整的参数训练学习器，从所有的参数中找到在验证集上精度最高的参数，是一个训练和比较的过程。

GridSearch 穷举搜索：在所有候选的参数选择中，通过循环遍历，尝试每一种可能性，表现最好的参数就是最终的结果。其原理就像是在数组里找到最大值。这种方法的主要缺点是比较耗时！

就比如我下面需要找到最适宜的决策树的个数以及最适宜的决策树的最大深度。 我首先设置的 n_estimators 参数择优的范围是:1~101，步长为 10。太

慢了。

实现过程：

sklearn 根据param_grid的值，首先会评估n_estimators和max_features的组合方式，接下来在会在 bootstrap=False 的情况下（默认该值为 True），评估 n_estimators 和 max_features 的组合方式，每一种组合方式要在训练集上训练 10 次，训练结束后，通过 best_params_获得最好的组合方式。我得到的最好的参数是 n_estimators：80；max_features：但是竟然发现还没有默认参数预测得高。实在是慢极了，只尝试了一次，

刚刚说了那么那么多，实际上预测的增幅仍然小极了。超级 big boss 即将登场。

这里先给大 boss 做个“美美”的铺垫。

（3）模型融合（Model Ensemble)

集成学习，即分类器集成，通过构建并结合多个学习器来完成学习任务。一般结构是：先产生一组“个体学习器”，再用某种策略将它们结合起来。

常见的模型融合方法有：Bagging、Boosting、Stacking、Blending。这一次使用的是 Bagging

Bagging:

自助聚合(Bagging)，该方法通常考虑的是同质弱学习器，相互独立地并行学习这些弱学习器，并按照某种确定性的平均过程将它们组合起来。这一次使用的是决策树分类器，500 棵决策树。对训练集随机采样

分别基于不同的样本集合训练 500 个弱分类器（决策树）。

对每个弱分类器（决策树）输出预测结果，并投票

每个样本取投票数最多的那个预测为该样本最终分类预测

五、应用

原因分析小结

可以看到受教育程度，城市等级与通过率正比

测试难度与通过率成反比

受训持续时间长度，程序测试类型，测试形式（线上或线下），助教参与程度，是否残疾，性别等对通过率均有不等的影响。

制定方案

如何提高受训者参与度和表现呢?

我认为有以下几点:

1. 如果公司有足够大的影响力，认为提供的福利足以吸引到高端人才，不妨在招聘时提高招聘门槛。

2. 受训时间长度 117-121 或者是 134-136 之间

3. 程序类型：按需求选择

4. 鼓励助教积极参与，积极帮助

六、总结

1. 缺陷

数据探索篇：数据的质量分析部分没有分析正负样本的比例，歧义值；然后 test_id 与 program_type 之间有很大的关系，都有一个共同的字母，所以这两者之间的去是指可以通过两者关系来填补（虽然在这份数据里影响不大）；数据探索的五个步骤：对比，分组，频数，抓大放小，可视化。我所做的工作，比较简单，只是单纯地把所有的变量与通过率之间的关系可视化，然后简单可视化了年龄的分布，几乎没有其他更加深层次的分析。还有可视化部分只是简单的一些直方图，还不够直观。不太好分析变量之间的联系。学会使用 sns 画图，非常有艺术气息。

特征工程篇：数据和特征决定了机器学习的上限，而模型和算法只能逼近这个上

限，可见特征工程是有多么重要了，但是非常遗憾，我在这一部分除了做了些必要的数据预处理之外，没有做好其他的工作。这也是我使用了大 boss，但增幅仍然很小的重要原因。

编程能力篇：之所以这一次花了大量时间，但是却没有做好特征工程的原因是，自己遇到了很多 bug，找 bug 的过程是辛苦的，没错，但是解决 bug 的过程却是令人兴奋的，在这一次的中期考核中，我很开心自己每天都游荡在 stackflow，认识了好多和我有着相同错误的人，（这里所说的认识，也许是确认过经历（眼神）），感谢开源！Yyds. 最为重要的是，在实战中才体验到了机器学习算法的魅力，这与之前看书推导，看原理相比，完全不在同一个频道，至少我更喜欢了。

2. 有效的技术方法：

数据转化，数据可视化，集成学习

3. 规划未来设想：

（一）用好提分三板斧：

a. 数据清洗和特征工程

b. 模型参数调节

c. 模型集成

（二）数据挖掘还需掌握的技能：

a.数据挖掘相关知识和理论的储备

（相关算法的原理和特征工程的理论）

【我觉得这一次并没有完完全全的体验到预测的快乐，很大一部分原因是远离不太清楚的情况下使用一些算法，没有加入太多的分析，以至于后期我不在去尝试调整参数刷新分数，一个原因是，就算调整了也不见得提高，自己的局限很清楚的摆在这里，不如好好看看算法原理，好好看看如何做特征工程】

b.工具应用

比如可视化工具，还有 python 一些有趣的库，sklearn，当然 python 三剑客也不见得自己很熟悉。

c.逻辑能力 （有待提升）

比如解决问题的条理性，考虑问题是否周到。希望自己用好 X-mind 这个工具，可视化自己的想法，可视化自己的工程

当然希望自己不仅限于此。希望自己以后可以摆脱暴力调参的惨状，以问题为导向，以解决问题为根本目的。