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 计算机类 9 班

学　　号　3120005035

学生姓名　王伟杭

2021 年 4 月 14 日

目录

一、 数据分析

对数据集整体做一个比较简单的把握，确定任务。

初步分析

首先，对训练集做一个初步的分析，先对前五行做一个简单的观测：

```
[50]: data = pd.read_csv("train.csv")
      data.head()
```

[50]:

	id_num	program_type	program_id	program_duration	test_id	test_type	difficulty_level	trainee_id	gender	education	city_tier	age	total_programs_enrolled
0	9389_150	Y	Y_1	136.0	150.0	offline	intermediate	9389.0	M	Matriculation	3.0	24.0	5.0
1	16523_44	T	T_1	131.0	44.0	offline	easy	16523.0	F	High School Diploma	4.0	26.0	2.0
2	13987_178	Z	Z_2	120.0	178.0	online	easy	13987.0	M	Matriculation	1.0	40.0	1.0
3	13158_32	T	T_2	117.0	32.0	offline	easy	13158.0	F	Matriculation	3.0	NaN	4.0
4	10591_84	V	V_3	131.0	84.0	offline	intermediate	10591.0	F	High School Diploma	1.0	42.0	2.0

```
[51]: data.columns
```

```
[51]: Index(['id_num', 'program_type', 'program_id', 'program_duration', 'test_id',
             'test_type', 'difficulty_level', 'trainee_id', 'gender', 'education',
             'city_tier', 'age', 'total_programs_enrolled', 'is_handicapped',
             'trainee_engagement_rating', 'is_pass'],
            dtype='object')
```

```
[52]: data.shape
```

```
[52]: (49998, 16)
```

可以发现，这是一个具有五万个数据，每个数据具有 15 个特征的数据集，而我们的任务为根据这 15 个特征来完成对数据的二分类。

同时根据常识与业务判断，id_num 是每个数据的独特 id，应该与 pass 与否无关。因此后续最先要对 id_num 进行去除。

查重

对训练集整体做一个简单的查重工作：

```
[54]: ## 查重
      data.duplicated().sum()
```

```
[54]: 0
```

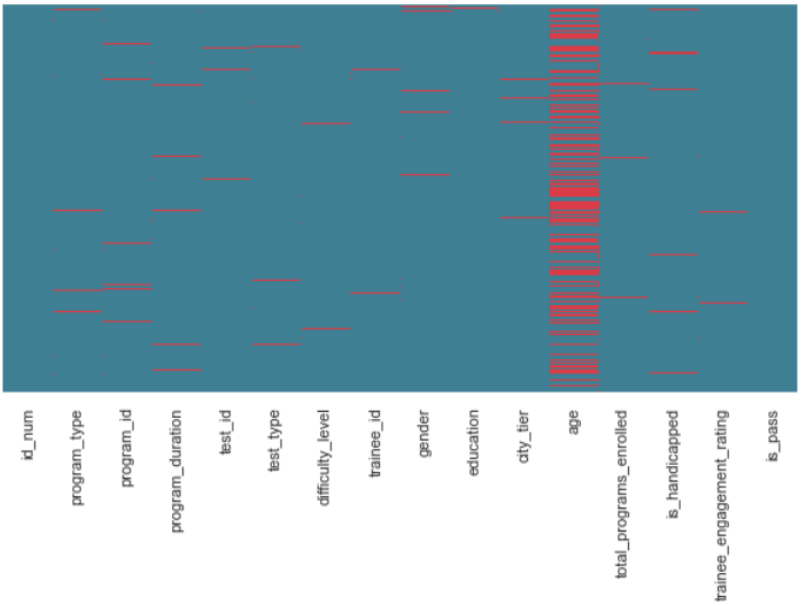
可以看到返回结果为 0，说明没有重复的数据，因此后续无需进行去重的工作。

缺失值观测

为了方便表述，下面会将 is_pass 列描述为标签，其他列描述为特征，接下来对全体数据做一个简单的整体缺失值查看：

	Total	Percent
age	19379	38.76
trainee_engagement_rating	772	1.54
trainee_id	739	1.48
program_type	731	1.46
test_id	725	1.45
is_handicapped	718	1.44
gender	707	1.41
difficulty_level	703	1.41
test_type	702	1.40
education	702	1.40
city_tier	700	1.40
program_id	699	1.40
total_programs_enrolled	692	1.38
program_duration	675	1.35
id_num	0	0.00
is_pass	0	0.00

可以发现除了 age 之外其余的特征缺失都在 1.5%左右，考虑对其使用均值或者中位数填充，最终使用哪一种通过对后续的分析才能给出；而 age 缺失值太多，考虑先对其进行填充，如果效果一般则直接对其去除；对此表格的可视化结果如下：



各个缺失值所在特征的分布都比较分散，所以不优先考虑对其他缺失特征进行去除，否则会有小部分的信息损失。

二、 特征分析

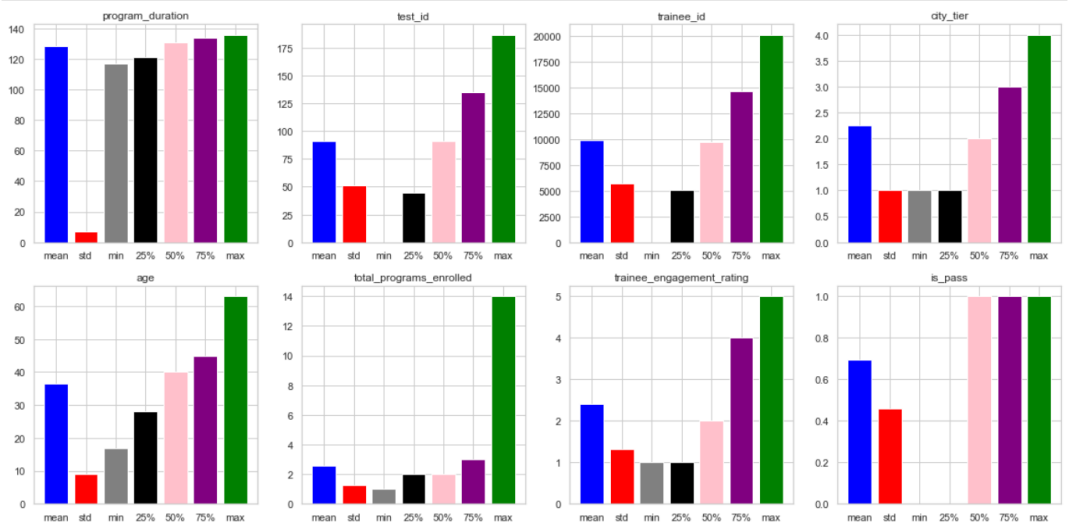
对所有特征做一个比较系统的分析。首先我们确定数据中正反例的一个分布情况：

```
[30]: data[data['is_pass'] == 1].shape[0] / data.shape[0]
```

```
[30]: 0.6962878515140606
```

可以看到正例的比例在 70 左右，这将是后续对特征分析的基础。

接下来对数值型特征的分布情况做一个简单分析：

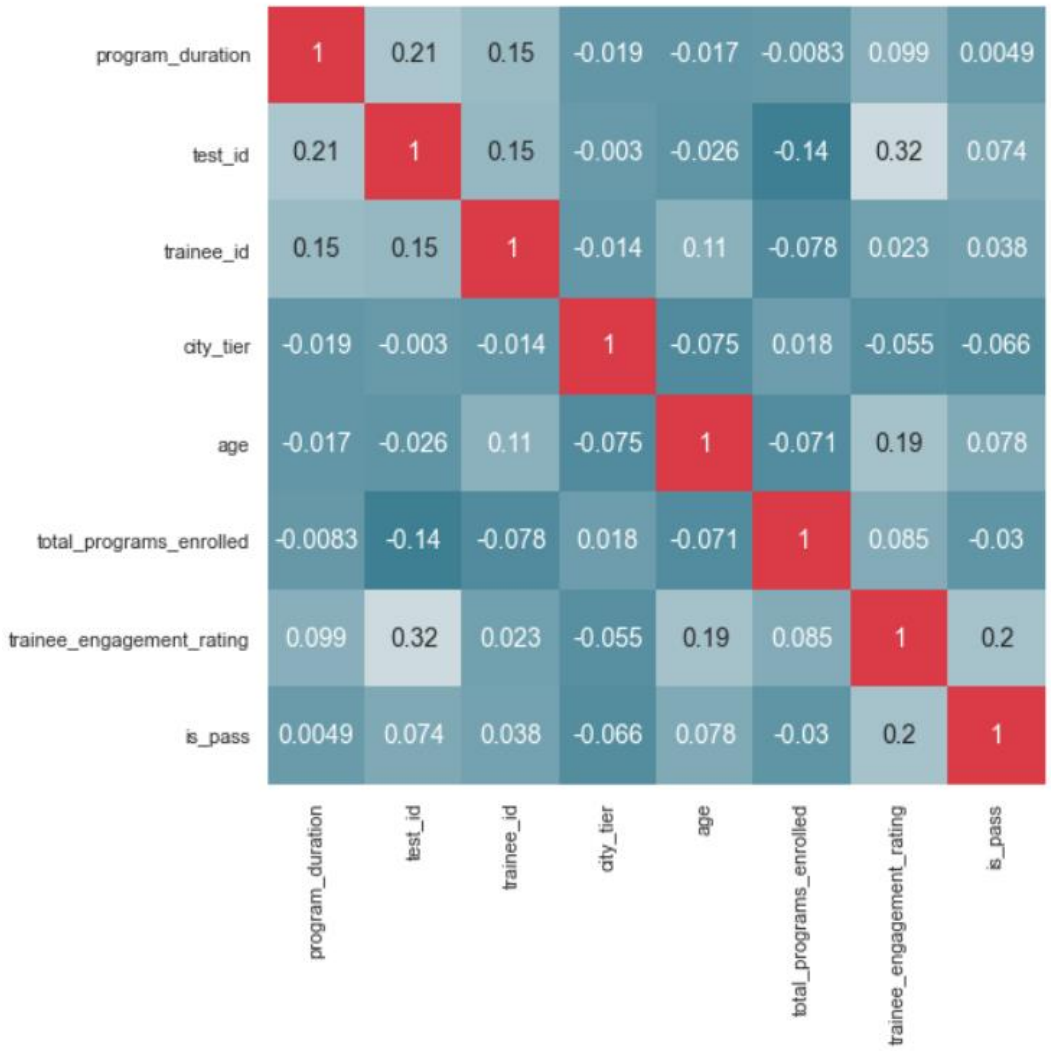


可以看出模型的分布还算是较为正常的，但是似乎 programs_enrolled 的分布稍微有些倾斜，后续看表现考虑将其删除。

数值型特征相关性分析

我们对整体上特征与标签之间的相关性可视化出来。

首先分析数值类特征与标签的相关性：



通过对这个数值矩阵的初步分析，我们得出以下的初步结论：

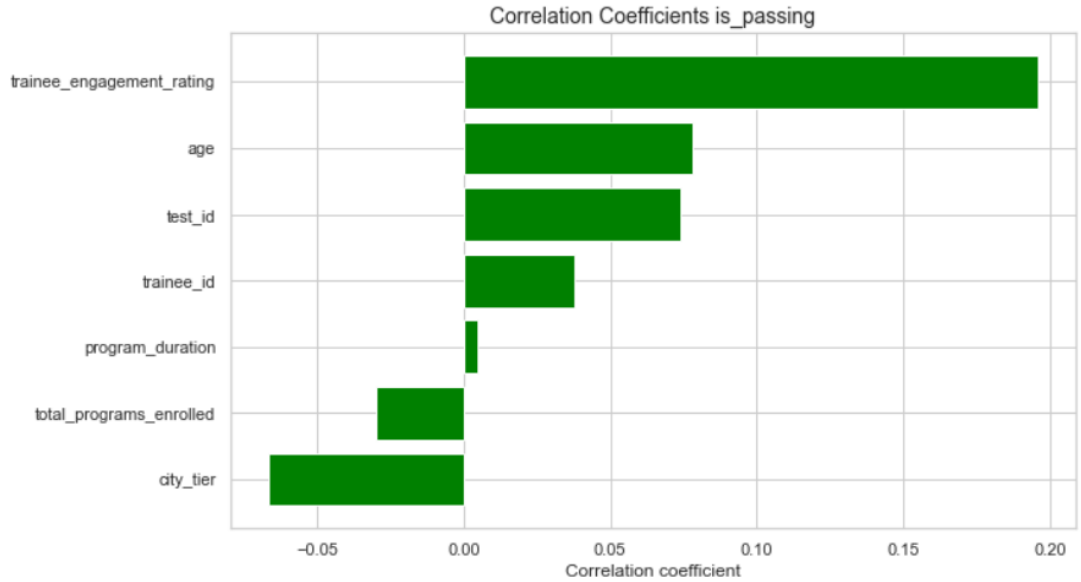
1.在数值类标签当中，rating——评分特征与标签的相关系数最高，达到了 0.2，远超其他特征，因此我们可以先初步认为，rating 在我们的预测当中会起到很重要的作用；

2.有些特征甚至与标签的相关系数为负值，这表明他们可能在预测当中起到的作用很小甚至会干扰我们的预测；

3.两个 id 特征（test_id，trainee_id）与特征的相关系数很高，这可能是巧合；

4.测试的评分似乎和测试的 id 具有一定的联系；

下面我们对这个矩阵进行可视化：

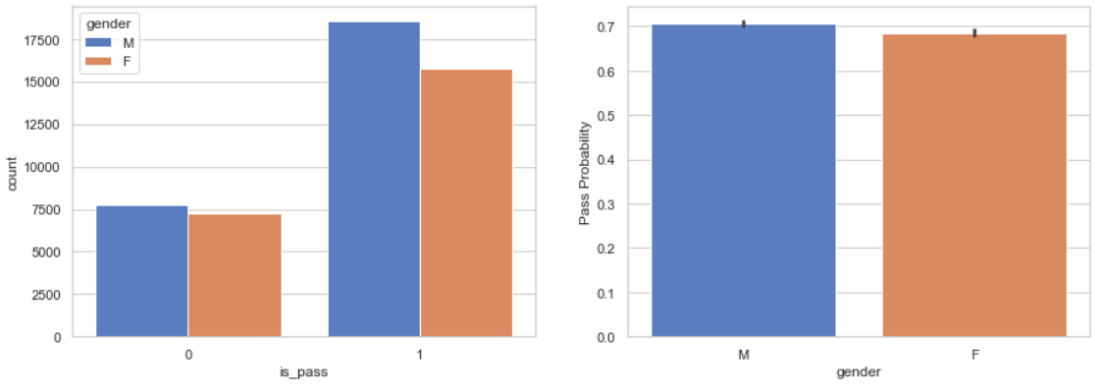


刚刚得出的结论在图上被清晰的表现出来。

标称型特征可视化分析

1.性别分析

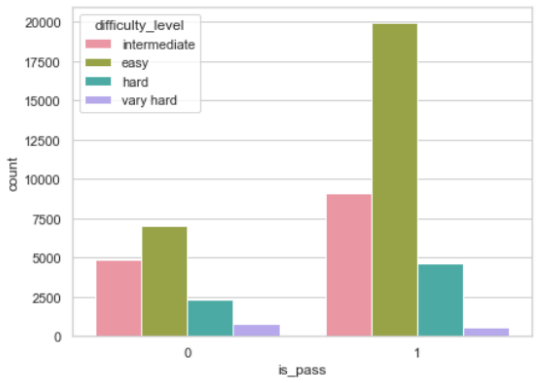
首先我们对性别的分布情况可视化：

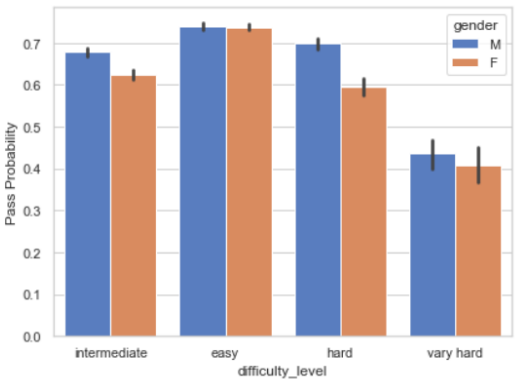


可以看出女性的数量略少于男性，但是男女的通过率非常接近。

2.难度分析

接下来我们对难度进行可视化：

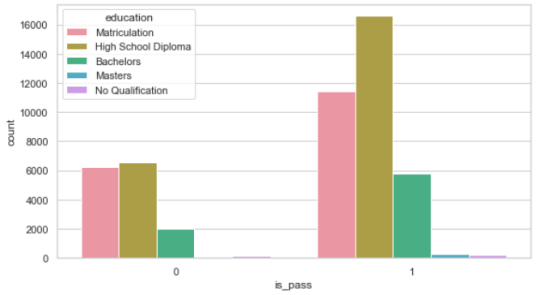


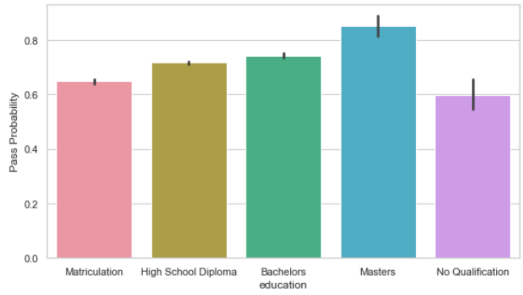


可以发现难度越低通过率就越高，符合常识，并且可以发现与性别并无非常大的关联；

3.学历分析

学历可视化：

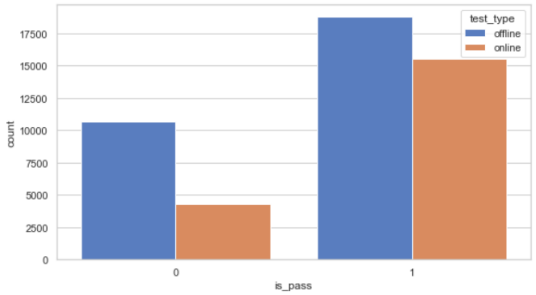


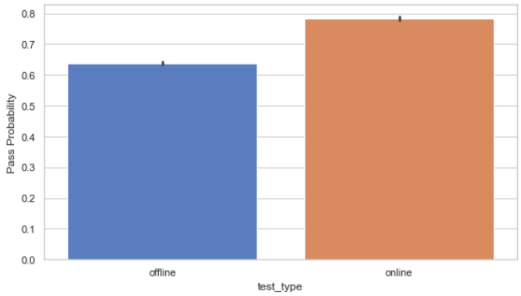


可以发现，学历越高的人通过测试概率越高，符合常识；

4.学习方式分析

学习方式可视化：

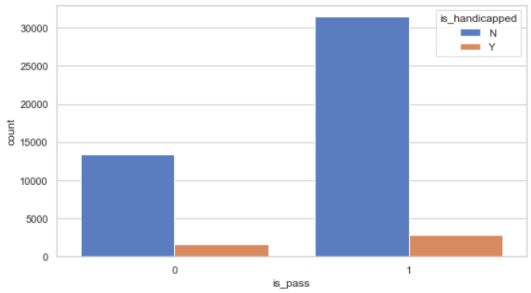


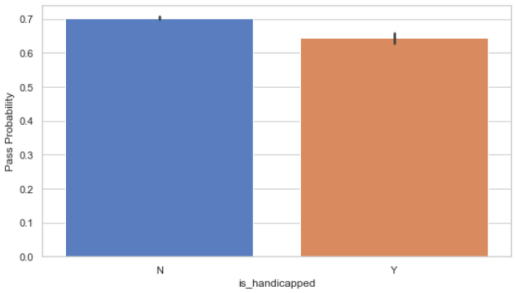


在线学习的效果远优于线下学习；

5.是否残疾分析

是否残疾可视化：

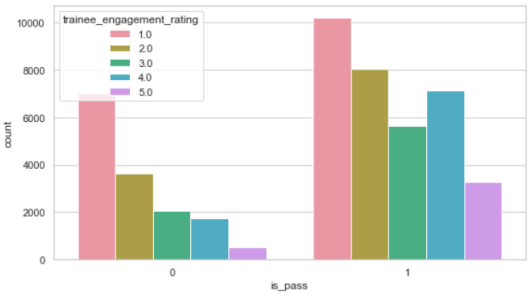


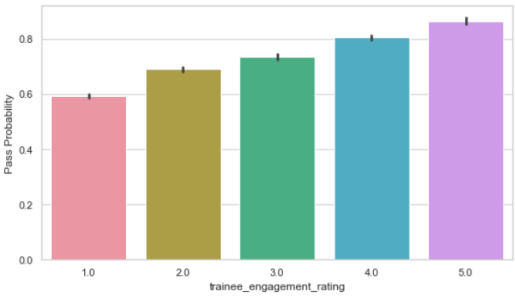


可以看出残疾的人学习效果要稍差；

6.评分分析

评分可视化：

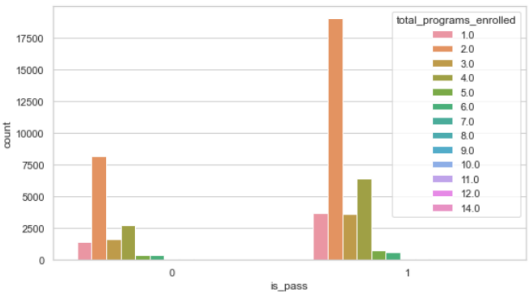


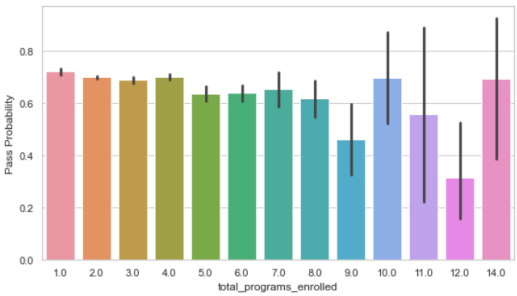


评分与结果的相关性很高；

7.选课数量情况分析

选课数量可视化：

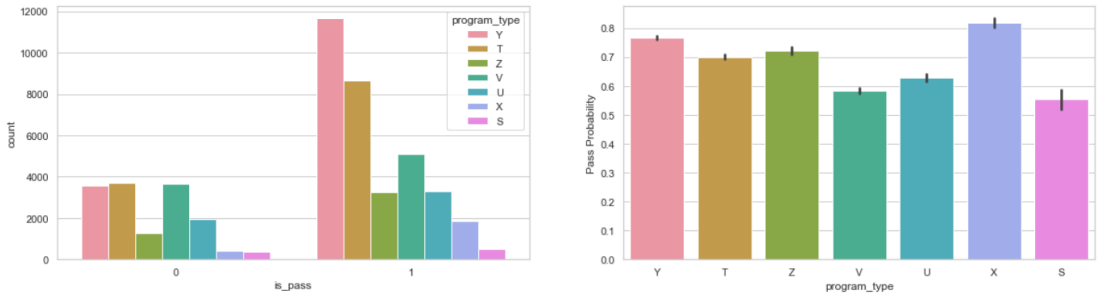




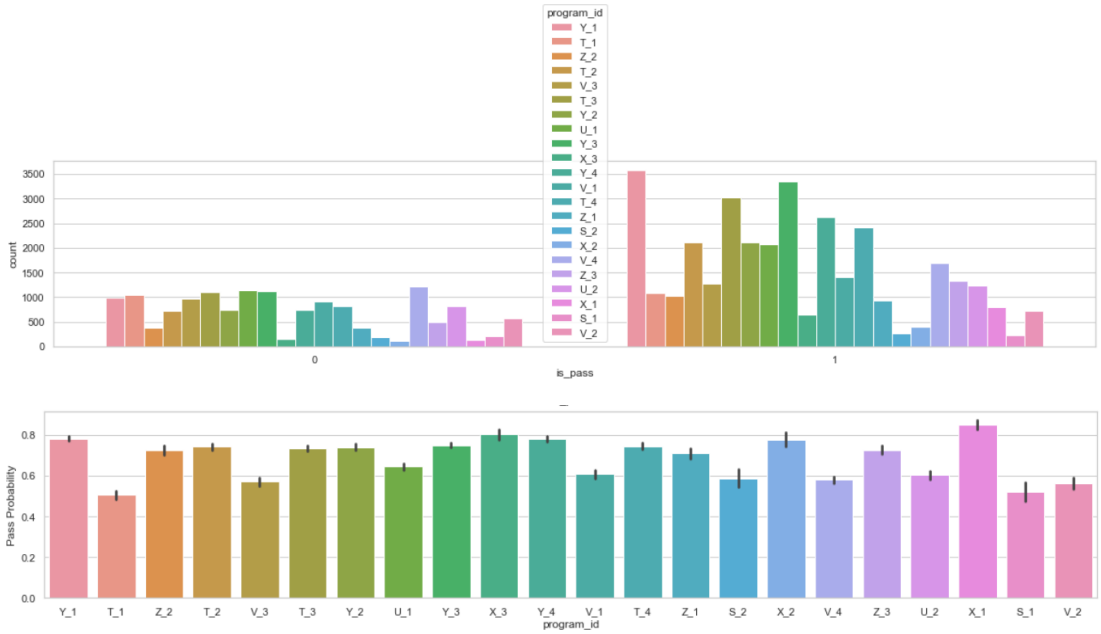
特征分布比较离散，而且特征效果不稳定，考虑噪声可能比较多，看拟合效果考虑将其删除。

8.程序类型分析

程序类型可视化：



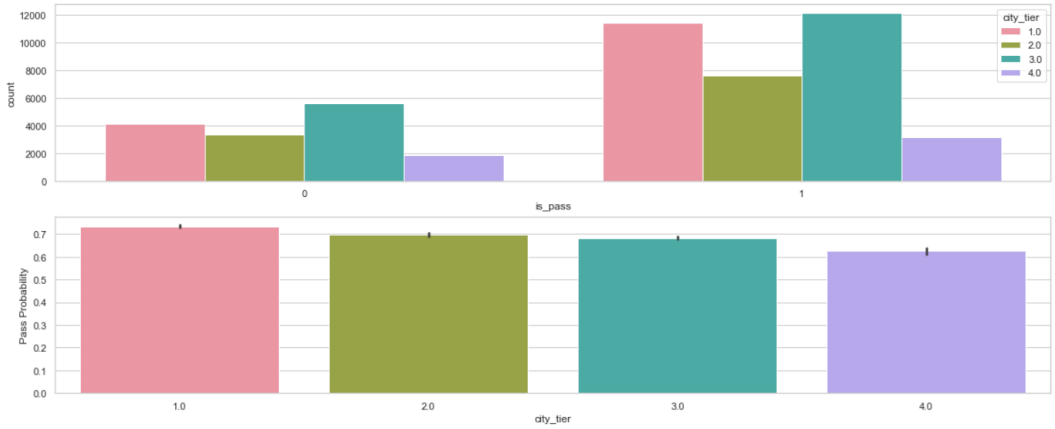
可以发现标签与类型选择有较强的相关性，进一步对 id 进行分析：



可以看出 id 的选择与标签有很大相关性，考虑对其进行编码后进行拟合；

9.城市等级分析

城市等级可视化：



可以看出城市等级越高通过率越高；

至此，已经完成了对非数值型特征的分析，得到以下分析结果：

1.性别对标签影响不大；

2.测试评分与学历与标签具有不强的联系；

3.选课数量值较为分散，并且选课多少对标签的影响并无直接联系，考虑对其去除；

4.测试类型与标签有较强联系，细分到 **id** 上联系变得更加清晰；

接下来进行数据的清洗工作。

三、 数据清洗

对缺失值和无关特征去除填充。

注：数据清洗的效果采用模型拟合效果来衡量，默认对所有标称型特征进行硬编码，采用最大深度 19（采用学习曲线得出，在后续模型优化当中给出原因）的随机森林进行 5 折交叉验证来筛选出最适合的清洗方法。

缺失值处理

1.先考虑对数值型缺失值进行均值填充，对标称型特征进行众数填充，得到 74.1%的准确率；

2.对数值型缺失值进行中位数填充，对标称型特征进行众数填充，得到 74.1%的准确率；

综合上述表现可以发现用什么方式来对数值型特征进行填补并没有对结果产生太多影响。

3.考虑到 age 缺失值太多，考虑对 age 的所有缺失值进行去除，对其他标称型缺失值进行均值填充，对标称型特征进行众数填充，得到 74.6%的准确率；

4.考虑直接对所有缺失值进行去除，得到 74.4%的准确率；

综合上述表现，我们发现对 age 进行填充的表现稍差，原因可能是因为太多的缺失值导致填充之后的信息有了一定程度的干扰，对我们这个具有 5w 个数据的数据集来说，对全部缺失数据进行去除似乎也可以满足我们对模型拟合的要求，因此我们选择去除所有的缺失值，这样也可以降低我们模型的收敛速度，提高效率。

异常值处理

在对数据的初步观察中发现可能具有异常值的特征只有选课数量，尝试采用对所有缺失值进行去除之后，将选课数量这一个特征直接去除，然后进行模型的拟合，得到准确率 73.7%，**拟合的结果甚至不如原来的效果，因此我们不对这个可能的异常值进行处理。**

四、 特征工程

特征工程效果采用随机森林进行 5 折交叉验证来筛选出最适合的特征工程结果。

编码方式

首先，我们对有序特征采用硬编码，对名义特征采用独热编码：

```
[4]: newdata.shape

[4]: (25427, 44)
```

结果我们得到了一个高达 44 个特征的数据，为了得到最优的特征，我们选用 embedded 嵌入法来进行特征选择，结果得到一个具有 13 个特征的数组：

```
[14]: X_embedded.shape

[14]: (25427, 13)
```

这和原来的维数差不多，但是既然是模型帮我们选出来的结果，所以还是选择随机森林进行拟合尝试，得到准确率 74.0%，模型对比不做任何特征处理的效果要低了不少；

为了找到最好的特征编码方式，对五个名义特征分别进行独热编码，其余特征采用硬编码，得到准确率均不如全部采用硬编码效果更好，故最终选择对全部特征采用硬编码。

综合上述表现，考虑原因为树模型可能更倾向于连续型的特征，而对离散型的特征比较不敏感。

特征选择

接下来，为了选择最适合的特征来拟合模型，我们要进行特征选择。

一开始在进行特征选择时候，我走了一个非常错误的方向。在一开始查看特征的时候，我发现数据的两个 id 属性都有比较多的取值，想当然的认为这两个特征与标签的相关性并不高，于是对这两个标签进行去除，然后对剩下的特征进行了：

1.对不同变量进行硬编码和独热编码处理、全部进行硬编码处理；

2.去除 program_type 特征，对不同变量分别进行硬编码和独热编码处理、全部进行硬编码处理；

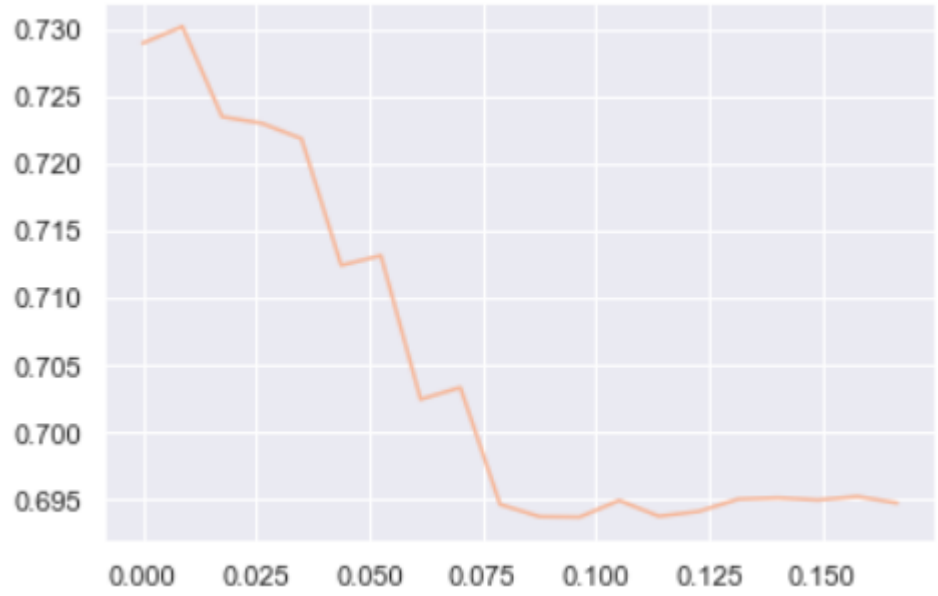
3. 去除 program_type、program_id 特征，对不同变量分别进行硬编码和独热编码处理、全部进行硬编码处理；

对以上的处理分别进行了多种模型拟合、调参方式，但是准确率保持在 72%上下，不仅浪费了两天时间并且还一直找不出原因，是本次考核当中失误最大的地方。

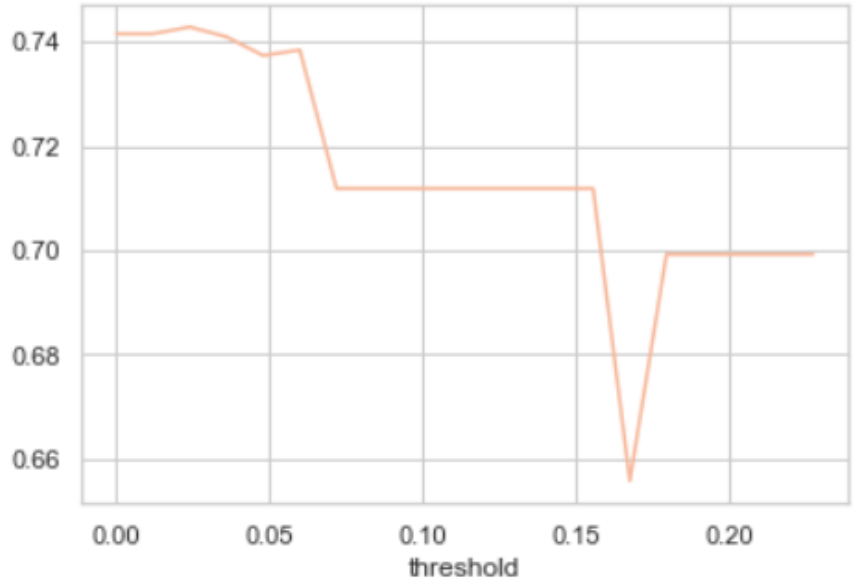
最后对数据重新进行分析时，才发现问题所在，重新对特征进行了分析，并借助

embedded 嵌入法与学习曲线来对特征进行选择，具体过程如下：

首先，根据刚刚采用独热编码的表现，发现 embedded 嵌入法在这个数据集下对于正常的编码方式（对有序特征采用硬编码，对名义特征采用独热编码）表现并不好，具体的原因我还没有找出来，但是我们可以采用学习曲线对 embedded 的 threshold 参数的选取进行监控：



看起来模型在去除重要性低于 0.005 左右的特征的时候表现会稍微提高那么一点点，其余时候特征选取的效果都是呈现降低的趋势，那么在全为硬编码的情况下呢？




```
最大准确率为： 0.7427927590381525 对应阈值为： 0.02396455287949477
```

输出最高点对应的阈值可以发现当阈值在大约 0.04 的位置时，模型的表现达到了最优，调用 get_support 接口可以发现被去除的特征为：

[28]:
```
# 进行特征选择
sfm = SelectFromModel(rfc, threshold=threshold[score.index(max(score))]).fit(newdata,y)
X_embedded = sfm.transform(newdata)
newdata.columns[~sfm.get_support()]
```

[28]: Index(['program_type', 'is_handicapped'], dtype='object')

即：程序类型、是否残疾，但是，这与我们一开始对数据整体的分析所得到的结果不同（与标签最不相关的特征应为城市等级与选课数量），这是什么导致的呢？

首先，嵌入法是使用特征的权重来对特征进行筛选，传入的阈值即代表低于这个阈值的权重所对应的特征都会被进行去除，而这里使用嵌入法传入的模型为随机森林，现在只能简单的了解到对于随机森林来说特征对应的权重即为该特征在模型当中的 feature_importances 属性：

feature	importance
trainee_id	0.224517
test_id	0.163358
age	0.155644
trainee_engagement_rating	0.084524
city_tier	0.065093
total_programs_enrolled	0.063342
program_id	0.057920
education	0.051900
program_duration	0.037791
test_type	0.034404
difficulty_level	0.034332
gender	0.027174

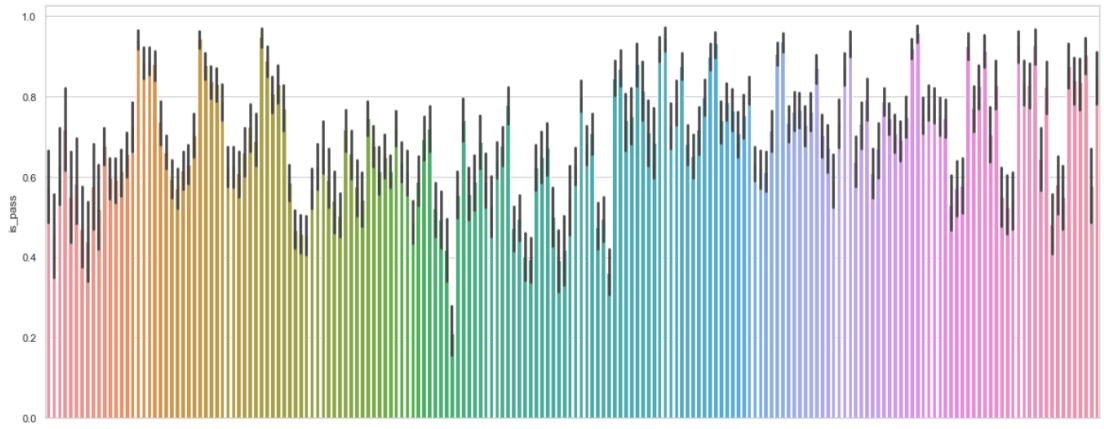
通过调用随机森林进行拟合之后的 feature_importances_接口进行分析之后，发现其对应的最重要属性依次为：trainee_id,test_id,age,rating…这与我们开头对数据的整体分析所得到的相关性最高的属性并不同（最高应为 rating），结合被去除的特征以及随机森林最倾向的特征可以发现，随机森林对连续性高的特征倾斜更多，也就是说，较为离散的特征在随机森林看来是不比连续性高的特征重要的。

因此，为了提高模型的拟合度，接下来我们对数据进行连续特征的分段。

连续型特征分段

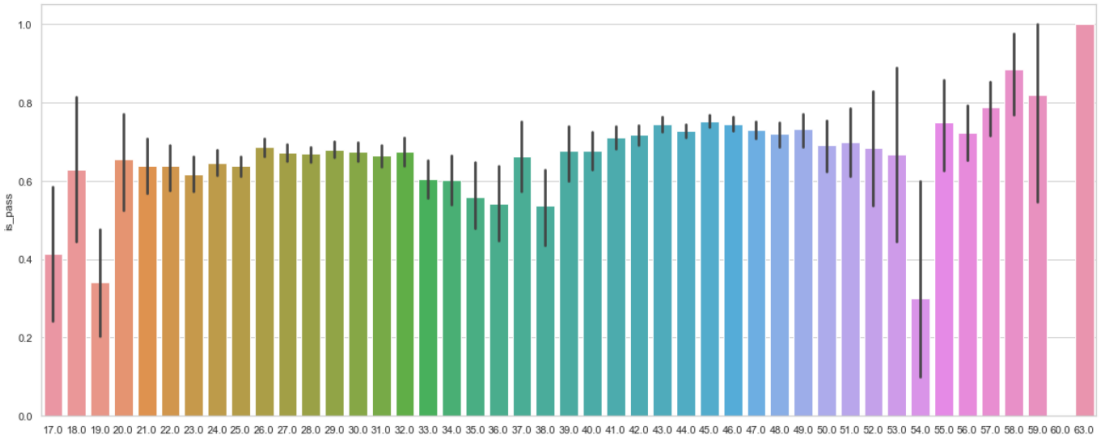
结合刚刚随机森林所选择的特征，我们现在先对三个排在 rating 之前的特征进行分析：

1.test_id 可视化：



可以发现，其数值大小与标签关联并不大，猜测其 id 大小与效果无关，故不对其分箱；

2.age 可视化：



可以发现结果与 test_id 相似，年龄的大小并不能很直接的影响标签，因此我们也不对其进行分箱；

3.对 trainee_id，由于其数量庞大，不适合对其画柱状图进行分析，因此我们对其进行二值化分箱处理，得到准确率 73.6%，效果比不进行分箱要差，原因可能与上面两个特征相同，tranee_id 与标签无明显线性关系，所以我们放弃对其进行分箱。

综合上述的三种处理，最终我们选取 embedded 嵌入法所选择的 12 个特征，对所有离散特征做硬编码，开始进行下一步的工作。

五、 模型选择

接下来进行模型选择工作，下面将在 KNN、logistic 回归、随机森林、梯度提升树这四个模型当中选取出最适合我们上面挑选出来的特征的模型。下面将进行五折交叉验证来绘制其学习曲线。

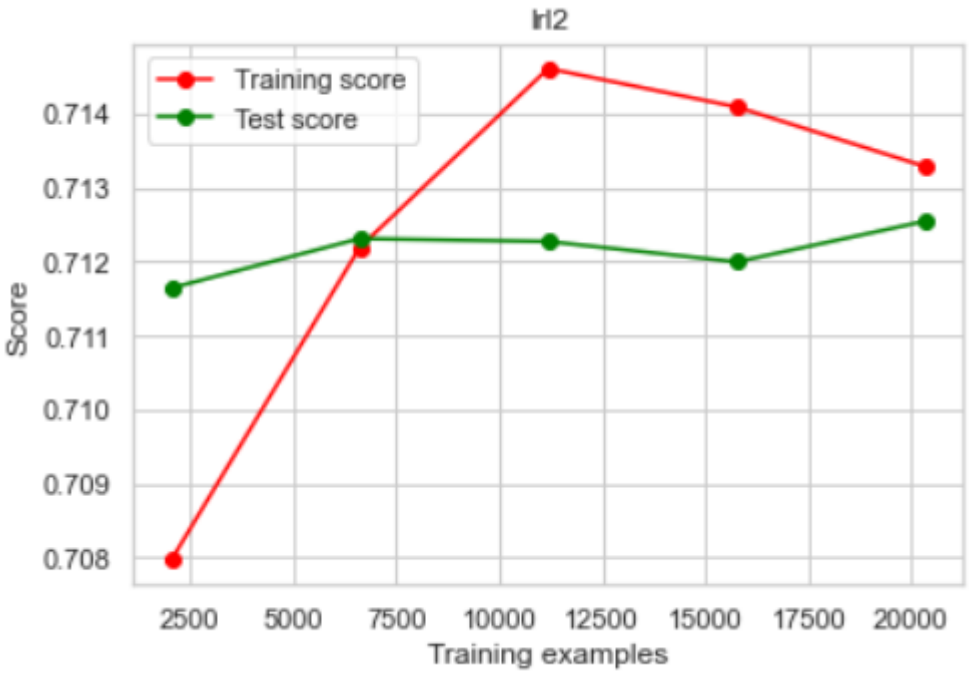
逻辑回归

考虑到 logistic 回归对于数值型的特征比较敏感，但是我们上述已经将特征进行了硬编码，因此我们先对原始数据中名义特征进行独热编码，有序特征进行硬编码，然后先进行一次 embedded 嵌入法，之后对其采用学习曲线观察其学习潜力：

```
[29]: X_embedded.shape

[29]: (25427, 35)
```

embedded 嵌入法选取了它认为最优秀的 35 个特征来拟合未调参的逻辑回归，接下来看看逻辑回归的学习曲线：



可以发现，逻辑回归的预测分数要远远的低于我们之前使用的树模型的拟合结果，而且根据学习曲线来看，模型的调参余地已经不是很多了，模型已经接近收敛；

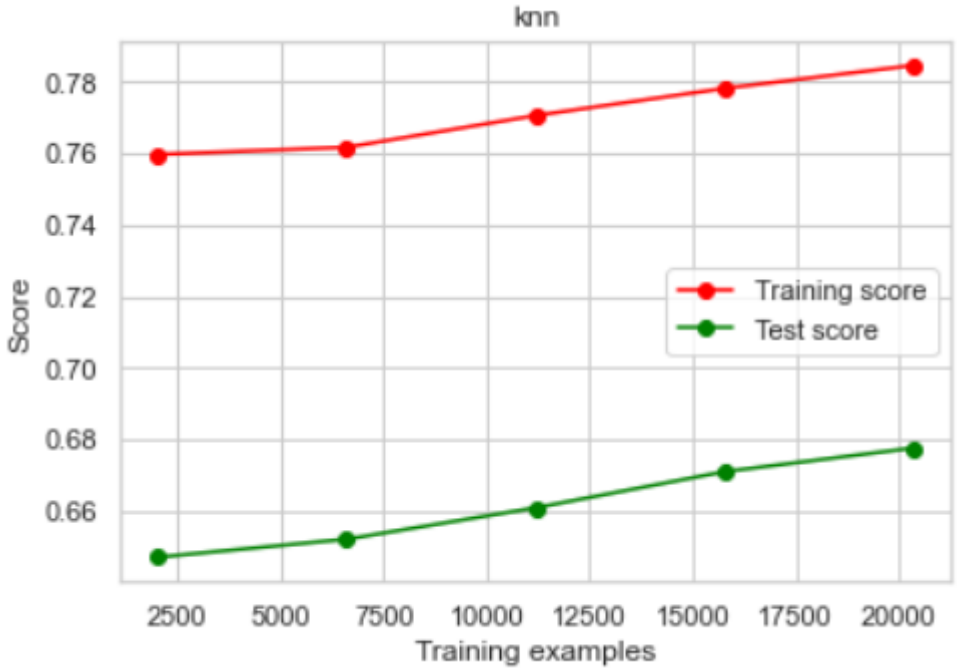
KNN

KNN 是通过计算欧氏距离来进行分类的，这里我们不考虑其是否对数值型敏感了，对全体进行一次硬编码后，采取方差过滤来进行特征选择，然后用学习曲线观察其潜力：

```
[41]: X_fsvar.shape

[41]: (25427, 7)
```

我们过滤了一半的特征，接下来观察其学习曲线：

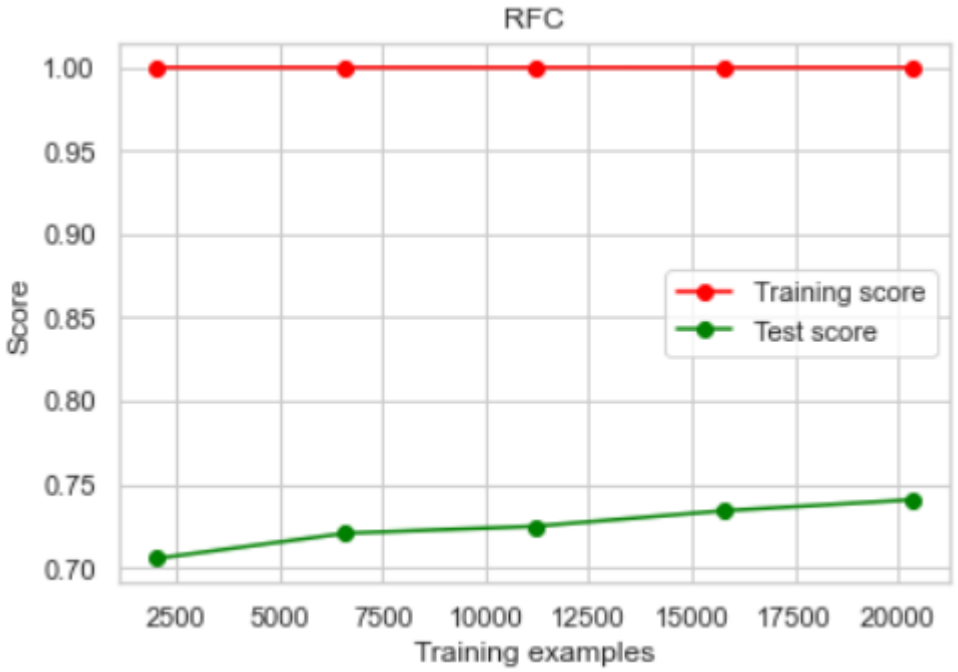


显示出较为明显的学习不足，明明已经缩减了一半的特征，且传入了两万个数据进行学习，仍然存在学习不足的现象，如果想要优化这个模型可能调参的难度会比较高；

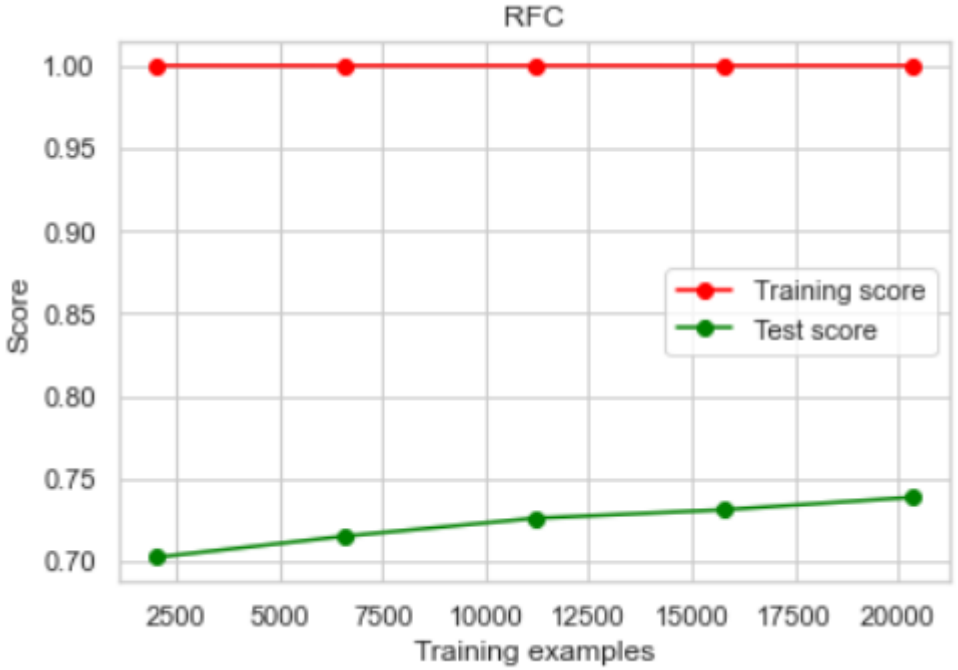
接下来对两个树模型的拟合都是采用前面特征工程所得到的特征进行拟合。

随机森林

首先，我们观察其在十四个特征下未调参的学习曲线：



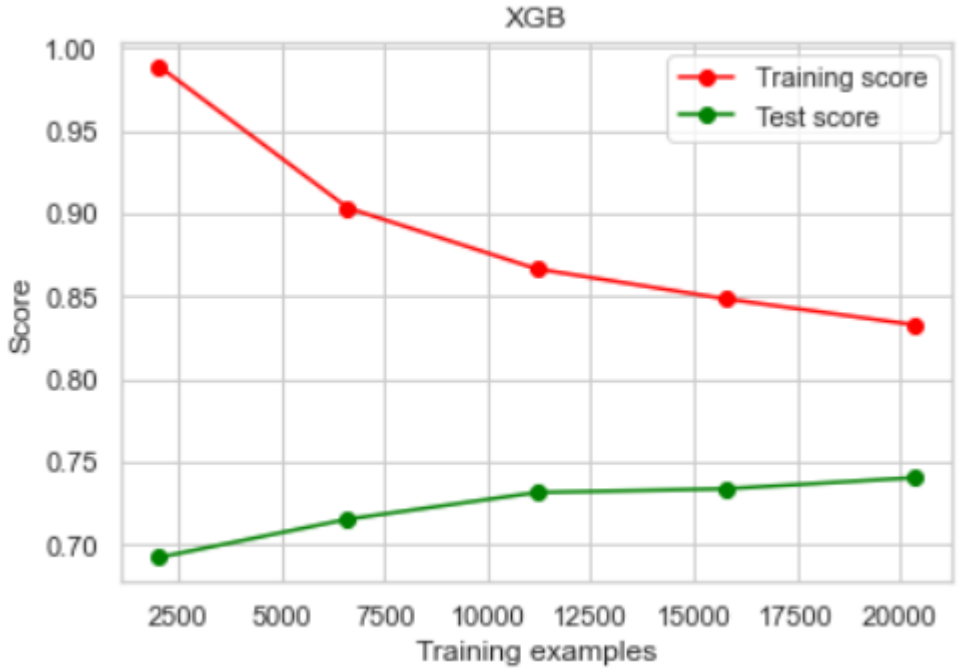
随机森林的输出要明显优于刚刚的两个模型，然后输出对其进行嵌入法之后得到的曲线：



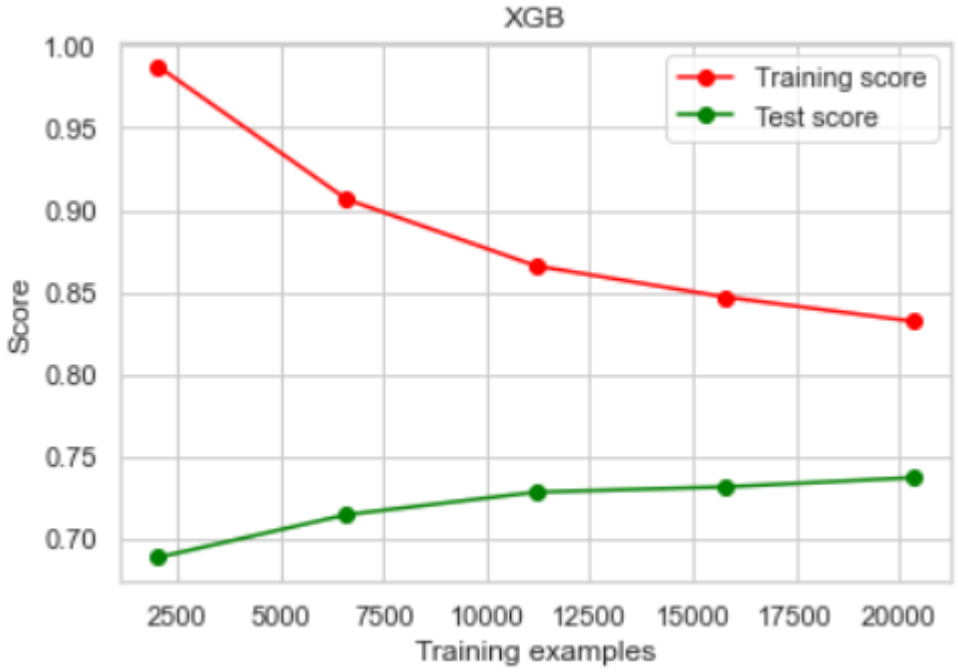
看不出有什么明显的效果，但是两者都有一个共同点，那就是都有严重的过拟合，再看看梯度提升树的表现。

梯度提升树

先查看其没有进行特征选择下的表现：



测试集效果与随机森林相似，不过整体有收敛趋势，再看看进行了特征选择之后的曲线：



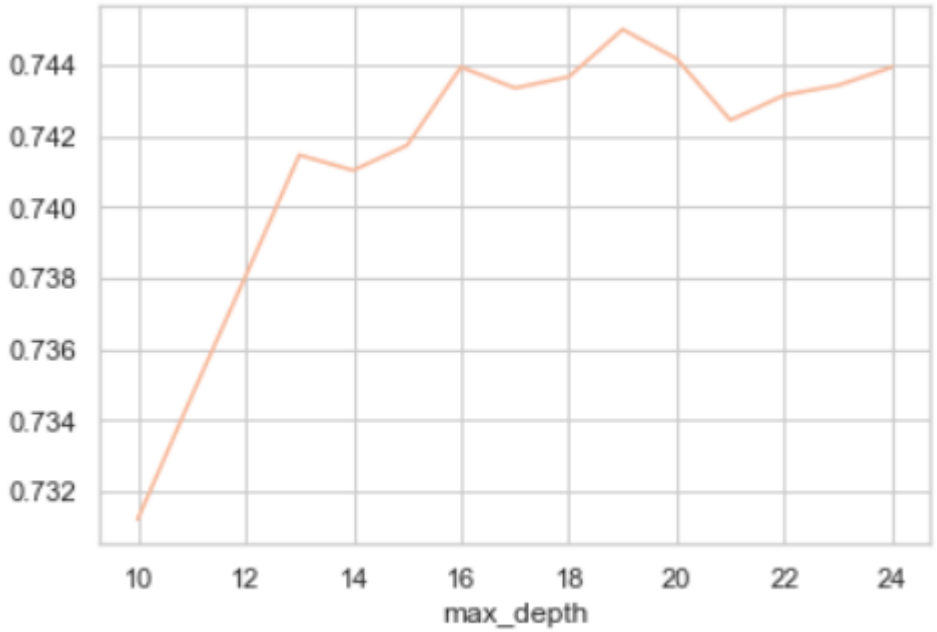
与刚刚基本无异，说明但是运算速度会稍快一点。

综合上面四个模型的表现，发现还是树模型的优化潜力更大一点，因此接下来我们选择对随机森林以及梯度提升树进行评估和优化。

六、 模型评估与优化

随机森林

首先，我们对随机森林进行优化，根据经验，我们一般对随机森林的 max_depth 参数进行调整，因此，我们先绘制学习曲线：



最大准确率为： 0.7449955184671023 对应深度为： 19

可以看出在深度 19 下，模型的表现最为优秀，我们选取深度为 19 的随机森林进行交叉验证：

```
# 调参后模型
rfc = RandomForestClassifier(max_depth=19,random_state=0)
score_t = cross_val_score(rfc,X_embedded,y,cv=10).mean()
print('随机森林准确率为：',score_t)
```

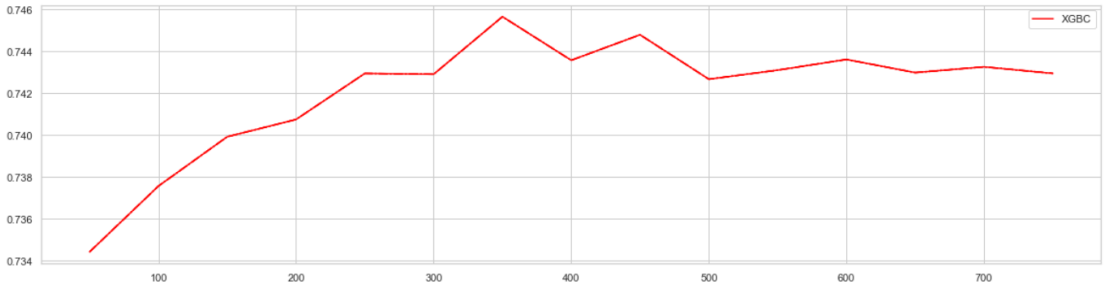
随机森林准确率为： 0.7449955184671023

10 折交叉验证表现达到了 74.5%左右的准确率，表现非常不错！接下来对梯度提升树进行优化。

梯度提升树

对于梯度提升树，我们优先调他的 n_estimators 参数，由于他的计算方式比随机森林更加复杂，我们不能和随机森林一样只是单纯提升它的取值，在这里，我们结合 bias- variance 的情况，来对参数进行监控：

```
偏差最低时对应的参数为： 350 此时准确率为： 0.7456640493443935 此时方差为：1.598724221604885e-05
方差差最低时对应的参数为： 150 此时准确率为： 0.7399222575245599 此时方差为：1.0847454871420201e-05
最低泛化误差对应参数为： 350 此时准确率为： 0.7456640493443935 此时方差为： 1.598724221604885e-05
```



在这个图中我其实还绘制了方差线来衡量，但是方差线没有很好的显示出来，这也说明了我们模型的方差很低，根据学习曲线，我们选取 350 来作为参数的取值：

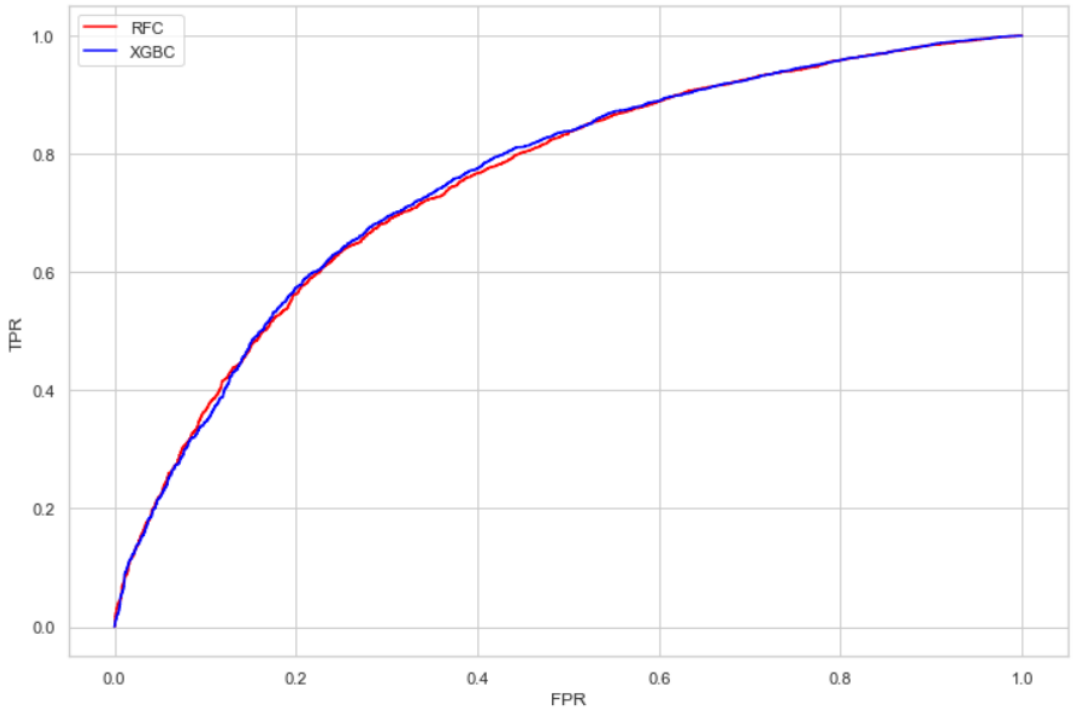
[62]:
```
# 调参后模型
reg = XGBC(n_estimators=350,random_state=50,use_label_encoder=False,eval_metric='logloss')
score_t = cross_val_score(reg,X_embedded,y,cv=10).mean()
print('梯度提升树准确率为：',score_t)
```

```
梯度提升树准确率为： 0.7490855785601733
```

可以看到调参后的梯度提升书准确率直逼 75%!是我们目前为止得到的最高的准确率。

模型评估

为了进一步衡量两者的拟合效果，考虑到我们的原始数据虽然不平衡，但是我们对正例的要求并不高，所以 PR 曲线和 ROC 曲线均可用来进行评估，这里我们就采用 ROC 曲线来对我们的两个模型进行最后的评估：




```
随机森林的AUC分数为： 0.752042896238245 梯度提升树的AUC分数为： 0.7544593075411922
```

可以看到两者的 AUC 值非常接近，但是梯度提升树的效果略优于随机森林，所以我们可以考虑采用梯度提升树来对我们的数据进行拟合。

七、 结语

在对两个模型的最终评估当中，我们认为梯度提升树的效果要优于我们的随机森林，但是，在提交 submission 时，随机森林的表现却总是比梯度提升树表现高了一个 1%左右，明明梯度提升树所采用的 Boosting 方法应该是要比随机森林所采用的 Bagging 方法要更加稳定效果更好的（事实上在留出法和交叉验证法上的表现也确实是这样的），但是在真正提交时却会出现随机森林效果更好这样的结果，这其中的原因我目前还没有分析出来。

梯度提升树所涉及的数学原理要比随机森林要复杂的多，分析起来也更加困难，采用梯度提升树来进行训练也只是尝试下在我的数据预处理下模型的上限到底怎么样，当然对于随机森林来说它背后所涉及的算法原理也还有很多需要我去探究。