概览

项目背景:

做一个基于电影评分的推荐系统

数据集选择:

Movielens100k 一个有名的数据集的缩略版本 具有一定的稀疏度

特征包括:

用户信息

职业（数字映射的名称），用户 id（数字），性别（字符串），年龄（数字），邮编（字符+数字）

电影信息

电影名（字符） 发行日期（数字） 类型（独热编码） URL（字符串） 视频发布日期（日期）电影 id（数字）时间戳（长时间）

分别存储在不同的字典里

项目执行思路:

数据集可视化——清洗——合并为干净数据

特征处理

输入不同模型

模型改良

特殊模型探究

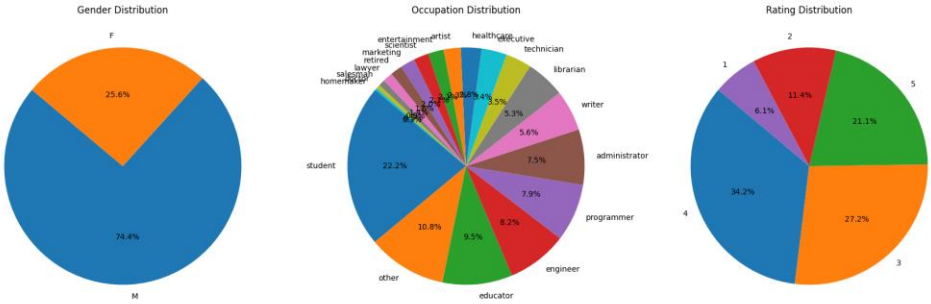
高级模型探究

对比模型&结论

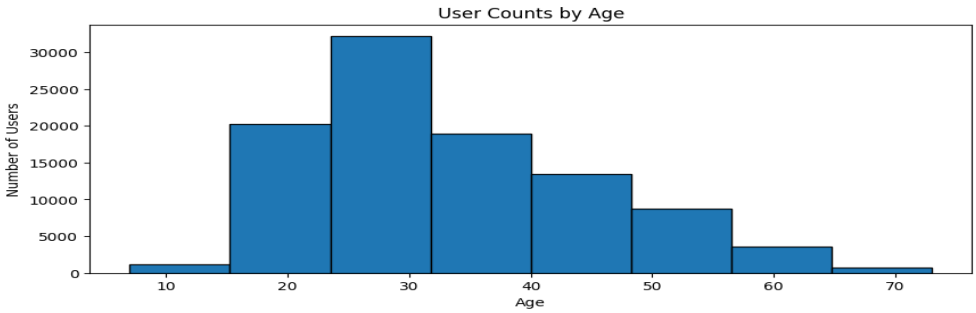
数据集可视化——清洗——合并

用户信息部分：针对 U.user 和 U.data，Head 进行统计信息分析，Null 值检查没有异常
将两表进行合并，再检查 null 并直接删除，发现了 901 行出现问题，检查发现是部分 user_id 没有覆盖导致，删除这部分互相没有覆盖的 user_id，组成 99099*8 的 df
电影信息部分：从 u.item 读取电影数据，后续处理包括：
只保留发行日期中的年份，算是一种平滑的手段，减少噪声
删除 URL 和电影名，对训练没有帮助
检查 nan，不存在问题
将该表和用户表合并，从而产生完整的用户信息——电影信息——评分表，便于后续使用，这里同样进行缺失检查并删除了 8 个值，相同的匹配问题。
合并后总表整理：99091*29 的矩阵

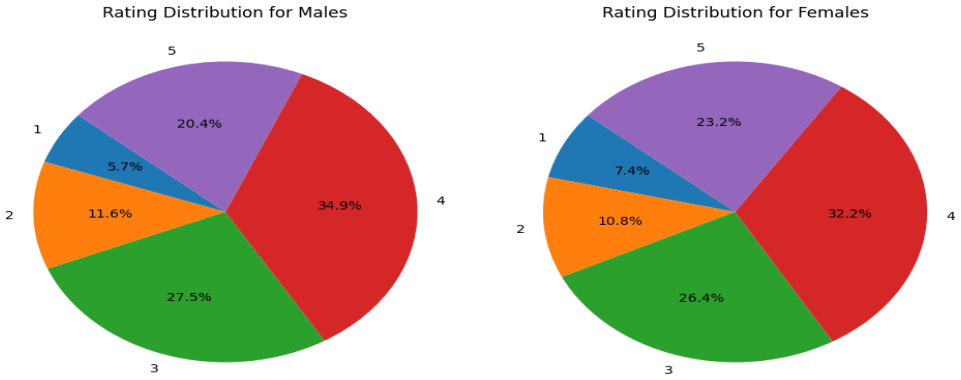
主要是针对用户画像进行可视化：从中可以得到一些 DA 的可用信息



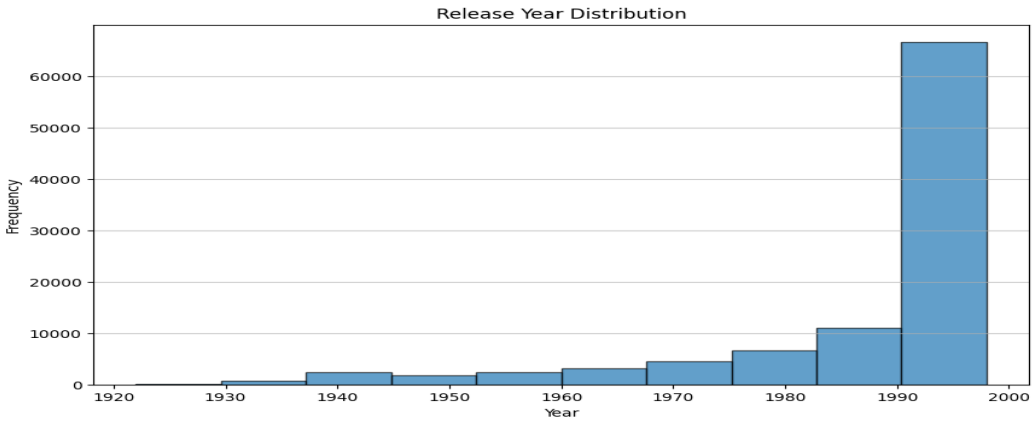
年龄刻画：



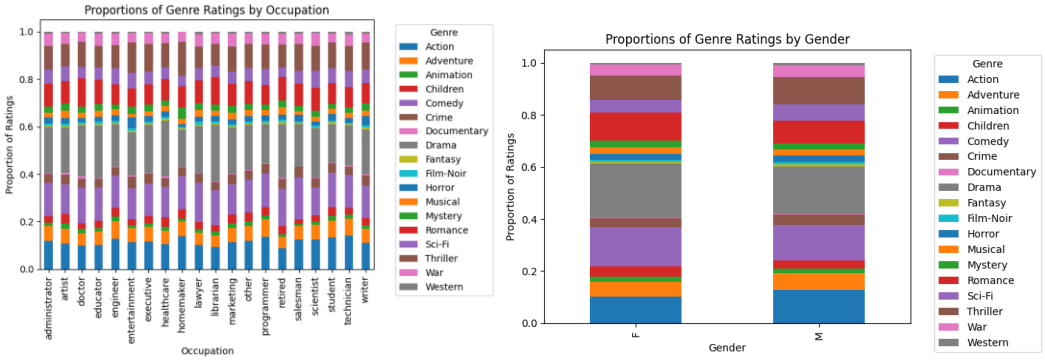
比较不同性别是否会影响评分：但这个比较很粗略，毕竟看的电影可能不一样

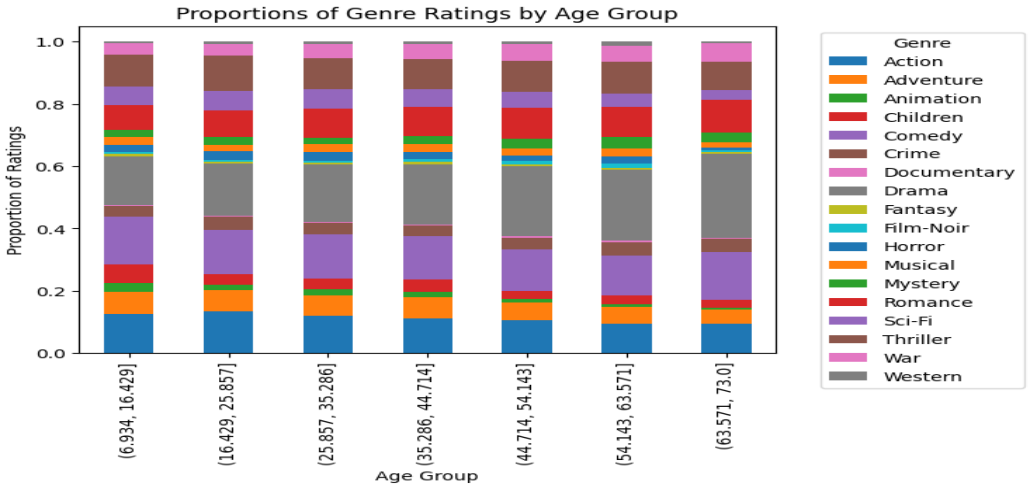


对电影信息进行刻画，如发行日期：

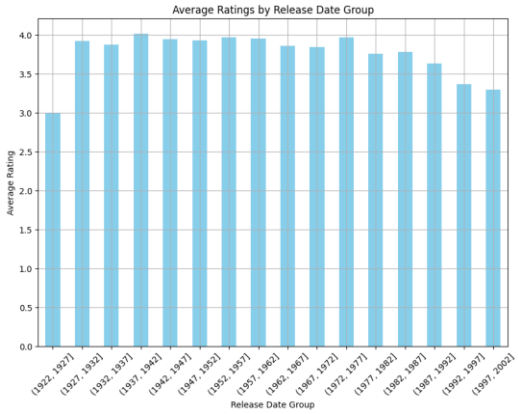


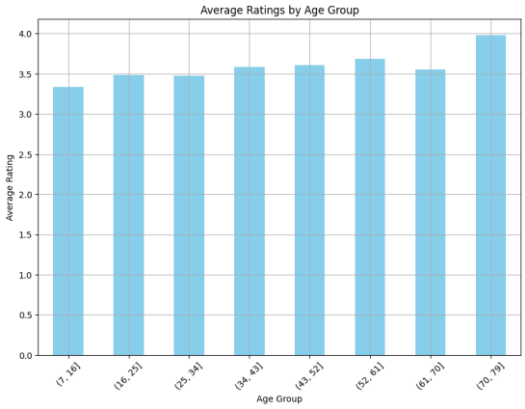
刻画不同职业/性别/年龄组的电影观看区别，均有一定程度的区别:



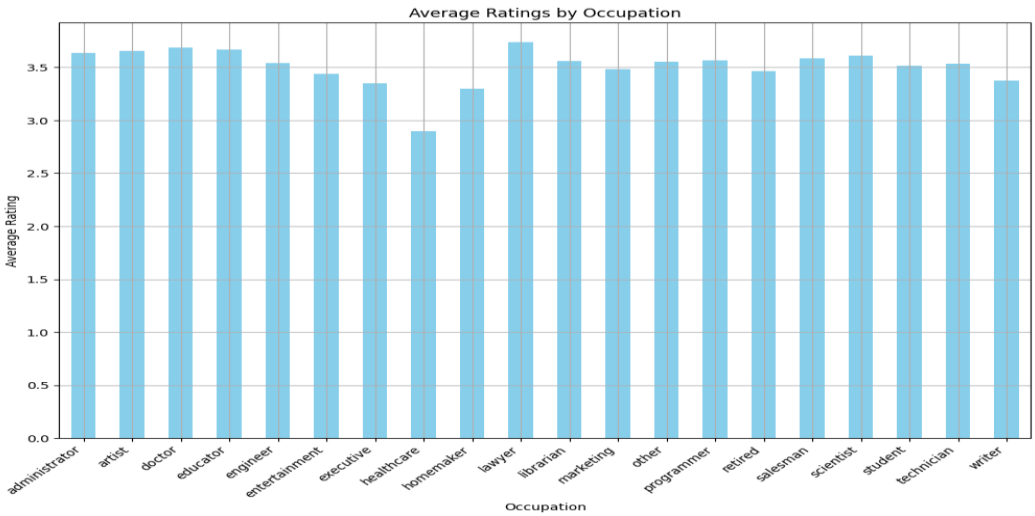


用户对不同时代发行电影的宽容度，中生代比较稳定，对于新片和太老的片似乎比较苛刻，老年用户也更加宽容:

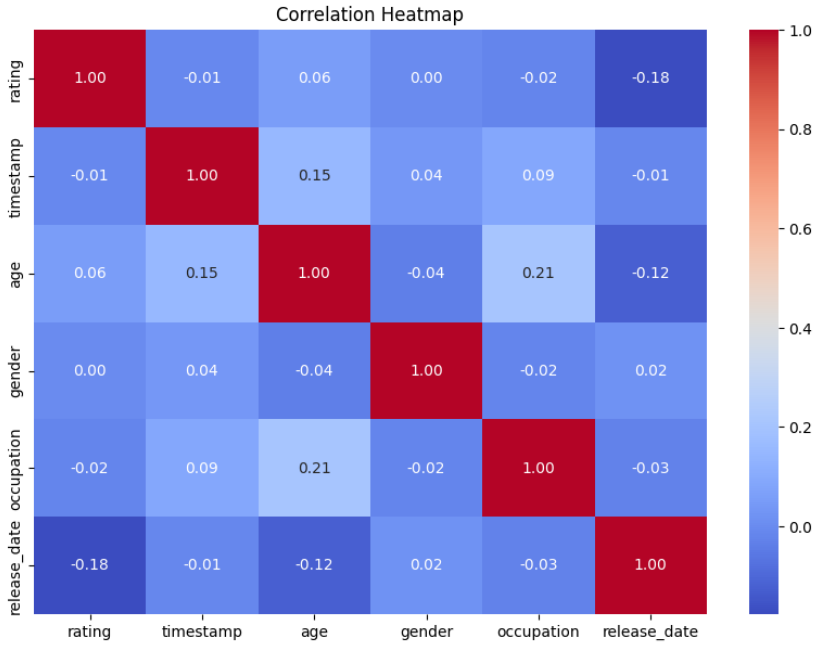




不同职业也显示了电影口味的不同：



接下来为了构建给模型使用的数据，对数据进行更细致的处理：
对职业字符串按照数字进行映射，以为了向量化输入准备
同时，这允许我们构建一阶关系矩阵来直观的衡量 feature 之间的关联性



对数据进行进一步处理：对性别按照 01 转换为独热编码，删除电影名称和 unknown 列（假定没有意义），检查了 zip_code,这一列相当混乱，包含字母，由于该特征按照一般规定是从前到后，地理位置逐渐精确，因此：保留前三位，并删除其中包含字母的列（2086：90991，约百分之 2），对于 timestamp 按照相同操作处理

特征处理

现在，初步的数据处理已经完成，按照不同策略分为三个对照组：

1. 对所有连续特征分组，包括年龄，邮编，时间戳，从而简化输入
2. 进一步简化输入，手动删除邮编和时间戳这两个直觉上可能无效的特征
3. 所有特征保留，也不做分组处理

所有输入按照 0.3，rs=42 分割数据集，选择 pointwise 的 mse 作为指标，这个指标也作为后续大量模型的损失函数直接使用

输入不同模型

LR 上：

1. MSE 1.1506543575864867
2. MSE 1.1511783663749202
3. MSE 1.1505094752067757

SVM：（十分钟一个，时间成本高）

1. MSE 1.1014728420440771
2. MSE 1.109374810515157
3. MSE 1.1051761930910795

GBDT(效果好计算快)

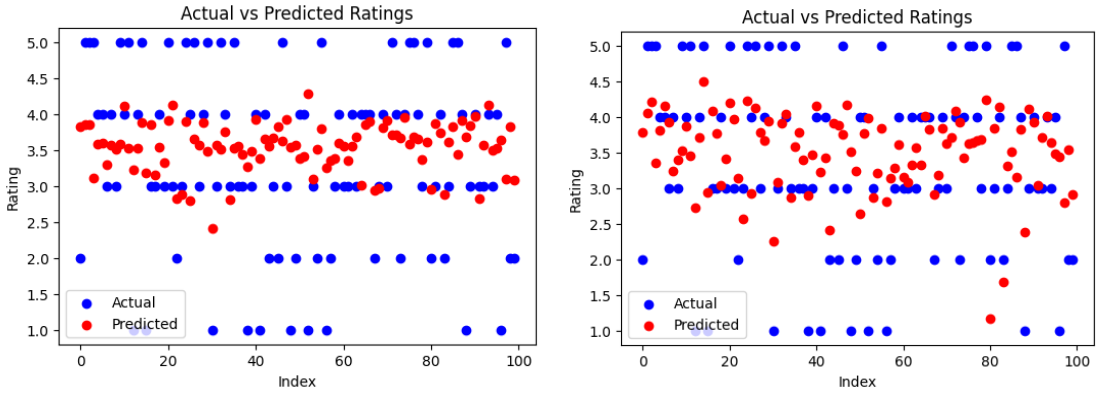
1. **MSE 0.9667582204973251**
2. MSE 0.9823934375332976
3. MSE 0.9688326652201468

MLP：（4 层 relu，Adam0.01，128-128-64-1，200 跑，十分钟）

1. MSE 1.1098673301711317 Loss:1.0910
2. MSE 1.0964974390258808 Loss:1.0778
3. MSE 1.1035022417407947 Loss:1.0825

初步实验表明，GBDT 又快又好，不做删除的数据集最好

一个可视化对比（LR，GBDT）



高级模型：（以后的统统用 data1 侧重比较模型表现）

FM：(接近 MLP，有提升)

1. MSE 1.1055232661992953 Loss:1.0946
2. MSE 1.1505582337161404 Loss:1.1725
3. MSE 1.107575899681599 Loss: 1.0924

Wide&Deep(类似 MLP 设置，但 lr 设置为 0.05，假设连续变量为 id 和 release_date，有嵌入层，对所有特征分别嵌入，维度为 4，使用 labelencoder 进行编码，这里连续变量这么设置其实也是因为 labelenocder 对于测试集没见过的就不会处理的，暂时不采用额外策略，结果就是耗时而且表现最差，不如 LR)

1. MSE 1.1610 Loss:1.1760
2. MSE 1.1715 Loss:1.1785
3. MSE 1.1620 Loss: 1.1298

GBDT+LR（youtube 的集成+转换做法，结果是不如 GBDT）

1. MSE 1.0024

2. MSE 1.0169
3. MSE 1.0098

GBDT+MLP（尝试提升，效果不好）

1. MSE 1.5324800905060942 Loss:1.5100
4. MSE 1.2776390513900824 Loss:1.2630
5. MSE 1.154118438283097 Loss: 1.1448

模型改良

首先对之前发现的 GBDT 最佳模型，进行网格搜索，找到最佳参数组合
深度：5 子树数：750（均是组合中的最大值）
MSE：0.8804840261712744
推荐系统设计：
拆分用户信息和电影信息，对于选定的用户 id，先提取其信息，然后和所有的备选电影信息进行组合，进行回归预测，去重，按照预测分数高低进行电影推荐

```
Movie: Monty Python's Life of Brian, Predicted Rating: 5.12
Movie: Close Shave, A, Predicted Rating: 4.82
Movie: Return of the Pink Panther, The, Predicted Rating: 4.82
Movie: Monty Python and the Holy Grail, Predicted Rating: 4.81
Movie: Sleeper, Predicted Rating: 4.80
Movie: Being There, Predicted Rating: 4.79
Movie: Manhattan, Predicted Rating: 4.77
Movie: Harold and Maude, Predicted Rating: 4.76
Movie: Ruling Class, The, Predicted Rating: 4.76
Movie: Private Benjamin, Predicted Rating: 4.76
```

Wide&Deep 改良

由于 Wide&Deep 表现最差和论文的表现相比十分反常，尝试通过调参加强，最终通过修改初始学习率为 0.01，极大增强了模型效果，详情如下（以下均用 data1，中庸的结果做参考）

1.假如把 id 先转为独热编码，然后对于每个特征分别嵌入，表现更差了，说明了独热编码转换再嵌入问题很大，不应当使用这个办法

MSE 1.1978 Loss: 1.2208

2. 将 id 不转换，当作回归特征处理，表现好了很多，更说明了方法 1 的问题

MSE 1.0176 Loss: 0.9401

3. 整体嵌入多热编码区域，符合一般的论文做法，表现不错

MSE 1.0003 Loss: 0.8747

4. 将没见过的数据和特征映射到同一个保留值，表现较好，但这里其实没用到，因为目前的特征处理不存在这样的情况

MSE 1.0379 Loss: 0.8894

5. 从方法 4 的基础上，将 id 从回归值换成嵌入，但不转独热而是直接嵌入，模型训练效果很好，但过拟合严重

MSE 1.0389 Loss: 0.6215

6. 如果直接删除 id 不参加训练，发现效果很差，说明了 id 嵌入本身带来了有效的信息

MSE 1.0466 Loss: 0.9280

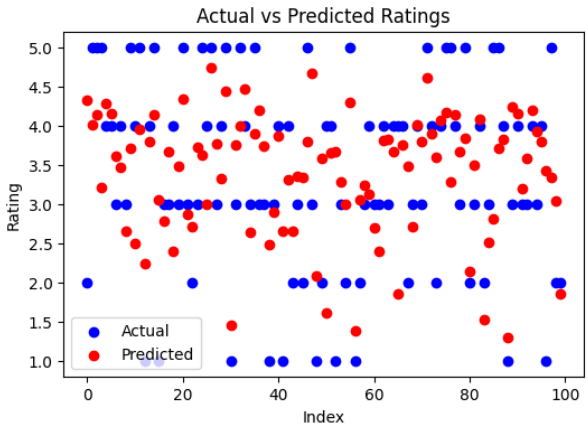
7. 在 5 的基础上，如果做一些手动的特征筛选，归类更多连续特征不嵌入，效果很好

MSE: 0.9761 Loss: 0.6714

8. 在 5 的基础上，如果换成全部嵌入，效果更好了

MSE: 0.9589 Loss: 0.6399

可视化：



9. 减轻模型复杂度的情况下（128-64），情况能稍微好些

MSE: 0.9234 Loss: 0.6676

10. 进一步减轻模型复杂度就会变差

MSE: 0.9788 Loss: 0.6919

11. 在 8 的基础上使用 L2 正则化（10-4），效果很好

MSE: 0.8536 Loss: 0.7769

12.残差连接，表现一般（已经加了正则化，学习能力下降了）

MSE:0.8989 Loss:0.8660

其他高级模型的探究

双塔模型（依然采用之前的多一个维度处理没见过的 id，使用 cos 损失并缩放为 0-5）：
分为用户塔和物品塔，并严格按照各自的特征进行输入，有个特殊的地方就是物品塔由于输入大部分是类型标签，因此这部分进行整体嵌入 表现进一步加强
MSE: 0.9063 Loss: 0.7877
双塔模型的好处在于这时候组合用户物品进行排序的思路比较清晰，最后可以得到如下的效果

```
Randomly selected training samples:
Training sample - True Rating: 2.00, Predicted Rating: 2.86
Training sample - True Rating: 3.00, Predicted Rating: 3.10
Training sample - True Rating: 4.00, Predicted Rating: 3.95
Training sample - True Rating: 4.00, Predicted Rating: 3.46
Training sample - True Rating: 3.00, Predicted Rating: 3.19

Randomly selected test samples:
Test sample - Predicted Rating: 3.97
Test sample - Predicted Rating: 3.30
Test sample - Predicted Rating: 3.30
Test sample - Predicted Rating: 3.13
Test sample - Predicted Rating: 3.78
```

但如果修改一下处理未见过的 id 的方式，比如直接按照最热门的嵌入结果进行嵌入，表现会反而不好，但不能就否定这个方式，毕竟长尾效应不严重，也不一定适合该任务
MSE: 0.9119 Loss: 0.7908
特征交叉系：
DeepFM,表现也不错
MSE: 0.8581 Loss: 0.7717
Deep&Cross,效果最好，看起来增强信息交叉输入比传统的过拟合解决办法更有效（SOTA）
MSE: 0.8486 Loss: 0.7546
不过要是阶数更高就反而不好了，毕竟 deep&cross 本质还是为了处理显式的低阶交叉特征，更高阶的角色和 deep 有些重合。所以表现不如前者
MSE: 0.8672 Loss: 0.7668

对比&结论

特征不删除比较好，删除了普遍不好

嵌入的表现普遍好，但是要注意深度模型的参数选择，方差很大

有 id 嵌入的普遍好，但不能是独热编码

交叉系普遍较好，其中 deep&cross 最好

过拟合存在，一定程度减轻模型复杂程度或使用正则化都不错

残差系存在使用限制