北京理工大學

本科生毕业设计（论文）

基于机器学习的奖学金管理系统设计

Design of Scholarship Management System Based on Machine Learning

学　　院：计算机学院

专　　业：计算机科学与技术

学生姓名：崔程远

学　　号：1120171189

指导教师：郑宏

2021 年 5 月 24 日

原创性声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

特此申明。

本人签名：　　日 期：　　年　月　日

关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名：　　日 期：　　年　月　日

指导老师签名：　　日 期：　　年　月　日

基于机器学习的奖学金管理系统设计

摘　要

奖学金管理是教务管理系统中的重要组成部分，目前学校的奖学金推荐主要是使用经验公式进行。本文尝试使用机器学习方法对奖学金进行推荐，以期减小教务系统的压力，并在一定程度上作为奖学金颁发的依据。最终使用的机器学习模型在实验数据上取得了一定效果。

本文主要分为 4 个组成部分，分别涵盖了实验过程中使用的算法或函数、数据仓储与数据导入、特征筛选和构建、使用多种算法进行训练、推荐与结果展示。其中最为重要的是特征筛选构建与模型训练部分。

在实验过程中使用了多种不同的机器学习模型，包括朴素贝叶斯分类器、XGBoost 决策树模型、DeepFM 神经网络等，并进行了新的特征工程与模型融合。此项目实现过程大致形成了推荐系统整体搭建流程。

关键词：奖学金；机器学习；特征工程

Design of Scholarship Management System Based on Machine Learning

Abstract

Scholarship Management is an import part of the educational administration system. At present, the school's scholarship recommendation is mainly carried out by using empirical formulas. This article attempts to use machine learning methods to recommend scholarships in order to reduce the pressure on the educational administration system and to a certain extent serve as the basis for scholarship awards recommendation. The machine learning model finally used has achieved certain results on the experimental data.

This article is mainly divided into 4 parts, covering the algorithms or functions used in the experiment, data warehousing and data importing, feature selection and engineering, and the use of multiple algorithms for training, recommendation and result display. The most important part is the feature selection and feature engineering as well as model training.

During the experiment, a variety of different machine learning models were used, including naive Bayes classifier, XGBoost model, DeepFM neural network, etc., and new features formed by high-dimensional feature interaction and new model fusion method were carried out. The implementation process of this project roughly formed the overall forming and construction process of the recommendation system.

Key Words: Scholarship Management；Machine Learning；Feature Engineering

目　录

摘　要 I
Abstract II
第 1 章　前言 1
1.1　系统背景 1
1.2　类似系统现状及问题 2
1.2.1　传统机器学习模型 3
1.2.2　深度学习模型 4
1.2.3　存在的主要问题 4
1.3　本文主要系统实现 5
第 2 章　算法理论 7
2.1　优化函数 7
2.1.1　固定学习率 7
2.1.2　自适应学习率 8
2.2　目标函数 10
2.2.1　最大似然估计 10
2.2.2　交叉熵 11
2.3　推荐系统评价指标 12
2.3.1　准确率 12
2.3.2　精确率 12
2.3.3　召回率 12
2.3.4　κ 系数 13
2.3.5　ROC-AUC 曲线与 AUC 值 13
2.4　特征降维算法 14
2.4.1　主成分分析 14
2.4.2　t-SNE 14
2.4.3　HDBSCAN 14
2.4.4　谱聚类 14
2.5　推荐算法 15
2.5.1　贝叶斯分类器 15
2.5.2　XGBoost 15
2.5.3　DeepFM 16

2.6 本章小结 21
第 3 章 数据预处理和数据仓储 22
3.1 原始数据分析 22
3.1.1 本科生信息字段 22
3.1.2 研究生信息字段 22
3.2 数据库模型构建 23
3.2.1 基础信息表 23
3.2.2 个人信息表 24
3.2.3 奖学金表 25
3.2.4 数据库 E-R 图 25
3.3 数据导入 25
3.4 本章小结 26
第 4 章 特征构建 27
4.1 特征分析 27
4.2 聚类方法 28
4.2.1 PCA/TSNE 特征降维 28
4.2.2 K-Means 聚类 29
4.2.3 HDBSCAN 聚类 31
4.2.4 谱聚类 33
4.3 特征构建总结 33
4.4 csv 生成 34
第 5 章 模型训练与结果预测 36
5.1 XGBoost 36
5.1.1 Hyperopt 自动化调参 36
5.1.2 数据编码方式效果对比 38
5.2 XGBoost 与逻辑回归融合 41
5.3 XGBoost 与贝叶斯分类器融合 43
5.4 神经网络模型 44
5.5 本章小结 45
结 论 46
1 特征构建与特征工程 46
2 模型训练与融合 47
参考文献 48

附　录 50
附录 A　GitHub 仓库地址 50
附录 B　DeepCTR API 说明 50
附录 C　XGboost 使用说明 50
附录 D　DeepCTR RunExample 50
致　谢 51

第 1 章 前言

1.1 系统背景

奖学金是教务、教学系统中的重要组成部分，对奖学金进行推荐与效果评估对教学管理与学生发展都有重要意义，这其中主要包括两部分的内容：分别是奖学金分配与奖学金评价，在此本节将分别进行介绍。

奖学金分配，即以奖学金规定为标准对奖学金进行分发处理，通常有两种方式。一种是根据现有的标准对学生的多种指标（成绩、论文、竞赛等）进行数值换算从而得到数值评价并直接按数值顺序等规则颁发奖学金，也就是现在在本科生奖学金颁发中常用的综合测评等评价指标，这种方法主要基于已有的经验公式进行奖学金计算与推荐；第二种则是以学生的获奖、论文、竞赛等为依据以奖学金申请条件为标准进行推荐，这种方法是以推荐系统的思想为基础进行推荐，但是这种推荐与常用的内容、商品推荐等亦有差别，主要体现为内容、商品等信息推荐能够通过用户在平台积累的大量历史数据信息获得较为完整的用户画像，并且平台累积的用户数据丰富，适合进行推荐，而本研究的系统数据量少而且标签分布不均匀，对推荐算法的设计与特征的构建而言是一个较大的困难与挑战。

奖学金评价，即为通过学生的历史奖学金记录与其获奖周边历史信息对奖学金效果进行评价，这里就需要对用户进行时序上的画像，那么如何构建评价指标也是比较困难的一个工作，因为衡量效果的指标有多种，包括成绩变化、论文发表、专利申请、竞赛获奖等内容，需要对评价指标进行设计。而这仅仅是数据的标签部分，甚至于数据的标签可能是一个向量输出多维度指标，同时训练的特征需要进行设计才能使模型进行有效的评价，这里可能需要用到时序的神经网络等能够提取时序特征的模型才能实现。由于本研究的数据量过少且分布不均匀而没有对奖学金效果评价进行处理。

在查询了与奖学金有关的推荐系统后，没有发现与本文有关的太多文献或信息。对于已知的和教务系统推荐有关的文献而言，与此类似的系统绝大部分集中于对应届生进行工作推荐。如大学生兼职电商平台工作推荐系统的实现[1]、一种面向冷启动学生用户的工作推荐系统及推荐方法[2] 等。国外的相关文献绝大部分也集中于对教学资源的分配与推荐。如一个用协同过滤以及情感探测的教学资源推荐系统

EduRecomSys[3] 等。

1.2 类似系统现状及问题

推荐的基础是数据，对于一个典型的推荐系统而言，获得的数据来源总体可以分为三种，分别是用户的信息、物品的信息以及用户的行为信息。所以依据不同的数据来源可以将推荐系统分为基于内容的推荐、基于用户行为的推荐。

总体而言，推荐系统的两大任务可以分为推荐和预测，所以推荐系统实现的评价指标也可以从上述两方面定义，分别是评分预测以及 Top-N 推荐。评分预测的目的就是通过已有的用户评价数据或者用户浏览数据对用户的评价模型进行学习。Top-N 推荐是给用户推送一个根据用户浏览历史或者相似度信息匹配得到的包含多个内容项的以推荐列表形式呈现的信息。在本系统中主要希望实现 Top-N 推荐，从目前已有的数据来推荐内容最多涉及 10 种奖学金。系统预期的功能是通过给目标用户返回按照特定规则排序的奖学金列表以提高推荐成功率，降低用户手工对奖学金进行选择的时间成本，提高奖学金推送的推送效率。

Top-N 推荐算法典型的应用场景为商品推荐系统，这也是目前研究、应用比较丰富的一类系统，在 Amazon、豆瓣、淘宝、当当等电商、网商平台都有大规模的应用。此类系统中大致有两种思路，一种是基于商品把商品推荐给用户，也就对应于上文中基于内容的推荐；第二种是以用户为基础通过计算用户的相似性将用户推荐给商品，对应于上文中基于人口统计学和用户行为规则的推荐。图1-1即为典型的 Top-N 个性化推荐系统。

图 1-1 网易云音乐歌曲 Top-N 推荐

Top-N 推荐的主要方法包括协同过滤算法 (CF)、矩阵分解、因子分解机（Factorization Machines）、逻辑回归、梯度提升决策树 (GBDT)、深度神经网络等为代表的一系列模型.

1. 2. 1 传统机器学习模型

在上述模型中基于协同过滤的推荐 (Collaborative Filtering)[4] 算法是应用最早和最为成功的技术之一，协同过滤一般分为两大类，分别是基于近邻关系的过滤 (neighborhood-based) 和基于模型的过滤 (model-based)。其中基于近邻的过滤又可以分为两大类，分别是 User-based 与 Item-based。这两类的区别在于基于用户的过滤[5] 原理是利用用户的历史喜好信息计算用户之间的距离，然后利用相似用户的加权商品评分来对目标用户可能的购买行为进行预测。它能够有效的在较少的用户反馈量当中个性化学习其他相似用户的反馈信息[6]。

而 Item-based[7] 的思想为通过计算商品和商品之间的相似度特征获取商品和商品之间的关联矩阵，依据得到的关联矩阵通过相似性计算获得物品之间的相似性特征，再根据相似性特征对相似度高的商品目标用户评分预测，进行得到用户的可能购买率。由于在最近邻的加权计算过程与相似度计算过程中，只分析了用户对用户或物品对物品的关系，所以它的运算速度相当快，并且在大型或小型的推荐系统都适用；而基于模型过滤则是最为常见的过滤算法类型，其原理是因为用户存在行为矩阵，用户的行为矩阵与物品之间存在隐变量，算法使用用户行为矩阵经过矩阵分解计算后的低阶用户矩阵和物品矩阵相乘来进行结果预测，其主流方法包括关联、聚类、分类、矩阵分解、神经网络及隐语义模型等。此类模型的推荐效果相对最近邻较好，但是算法训练时间较长复杂度较高使得推荐系统的训练以及在线推荐时间大幅上涨。由于基于模型的原理是用户和物品之间存在隐变量，所以早期推荐系统为了减少运算规模，常用推荐方法还有基于关联规则的推荐，由于用户的购买是存在关联性特征的，所以此类方法的主要思想就是通过算法统计用户的购买倾向，如对购买了商品集合 A 的用户和商品集合 A 与商品集合 B 存在关联的规则对用户推荐集合 B 中的商品。对于小型的推荐系统而言，使用较多的算法为 Item-based 的协同过滤算法，而对于大型的系统，由于存在丰富的用户特征以及商品特征，则可以考虑基于用户的协同过滤或基于模型的过滤算法。

但是随着数据量的积累，常用的数据特征工程方法 one-hot encoding 在面对大规模数据集的时候表现出了难以解决的缺陷，比如在一个有 10000 商品和 1000000 用

户的系统中如果对用户的商品购买特点进行 one-hot encoding 那么总的特征空间过大而导致最终的特征矩阵过于稀疏，所以这样对网络的训练而言造成了比较大的困难，而且也导致了很多不必要的存储空间的浪费，所以大数据的高效处理也是一个棘手的问题。对于巨大的特征空间而言，可采用的方法包括使用 PCA 进行主成分分析降低特征维度，而且在实际应用中 one-hot encoding 后得到特征向量再经过 PCA 主成分分析进行降维的组合非常有用。除此以外 FM、FFM 以及 DeepFM 等也在此领域大显身手。

上文中也曾介绍与教务系统有关的工作推荐[1]，使用的主要算法是通过用户行为数据构建协同过滤器挖掘用户相似性，再使用 PageRank 等算法通过点击率预测进行推荐。最终推荐结果标签主要为点击率信息，与本文主要推荐不一致。

1.2.2 深度学习模型

神经网络、机器学习技术的进步使协同过滤算法也有了一些新思路，其中较为典型的思路包括使用 GBDT 或 XGBoost 等集成学习模型进行混合推荐、使用矩阵分解的思路对矩阵进行更快速的降维处理与提取、基于深度学习的协同过滤算法等。

随着深度神经网络在近年的发展，越来越多的模型也选择采用神经网络作为推荐算法进行推荐，典型的算法包括 DeepFM、Wide & Deep 等，2016 年 Google 提出的 Wide & Deep 模型就在 YouTube 的视频推荐中应用并且取得了不凡的效果，离线和在线测试相比传统的算法均有较大提高。在 Wide &Deep 的基础上 2017 年华为联合高校提出了 DeepFM 算法，克服了 Wide & Deep 需要手工特征工程的缺点并结合 FM 和 Wide & Deep 的优点对低阶和高阶特征同时进行学习，训练了一个使用 FM+DNN 的端到端神经网络。

1.2.3 存在的主要问题

但是基于协同过滤的算法也包括一些固有的问题，比如典型的冷启动问题与特征空间稀疏问题。

冷启动问题[8] 是推荐系统中众所周知存在的一类问题。推荐系统的推荐原理是通过形成一种特定类型的信息过滤（IF）从而试图呈现用户可能感兴趣的信息项（电子商务，电影，音乐，书籍，新闻，图像，网页）。通常推荐系统将用户的个人资料与某些参考特征进行比较。这些参考特征可能与 Item 特征（基于内容的过滤）或用户的社交环境和过去的行为（协作过滤）有关。根据系统的不同，用户可以与各种类

型的交互相关联，典型的特征如：点赞，购买，喜欢，页面访问次数等。而冷启动则是指在缺乏以上用户交互特征或商品信息特征的情况下如何进行有效推荐。典型的冷启动有如下三种类型：

1. 新系统：指的是推荐系统或程序的启动，尽管在推荐系统中可能存在 Item 信息，但是由于缺少用户所以无法获得可靠的用户交互特征。

2. 新 Item：新 Item 已添加到推荐系统目录中，它可能有一些内容信息，但是没有交互作用。

3. 新用户：新用户注册并且尚未提供任何交互，因此无法提供个性化推荐。

特征空间稀疏[9]（Data Sparsity）是另一个存在于推荐系统中的典型问题。特征空间稀疏同样是由于用户数量和 Item 数量巨大导致的[8]。在推荐系统中一个典型用户可能只够买或浏览了有限商品，而整个商品空间巨大，所以如何进行有效特征提取从而实现有效推荐成为了一个棘手的问题。

这两类问题目前都没有比较好的解决思路。冷启动问题解决思路包括新用户资料填充、混合特征加权[10]、特征映射、正则化权重等。特征空间稀疏主要通过 embedding 方法将高维稀疏特征映射到低维稠密空间。其中冷启动问题在本文实验数据上也会涉及。由于此系统的用户数据量较少，对应奖学金（Item）数量也较少，所以这个问题同样无法适用于常见冷启动问题解决思路。

1.3 本文主要系统实现

本文系统主要实现基于协同过滤的奖学金推荐，数据集来自计算机学院 2016 级学生以及 2017 级学生的真实数据，可用标签规模为 186 条。本文首先进行了数据库架构构建，然后将数据 excel 文件转换为 sqlite 数据库记录，再根据 sqlite 记录进行筛选构建特征矩阵，而后用不同协同过滤算法进行推荐系统构建。

在此系统实现后，其包含的机器学习算法也可以给其他类似的问题提供思路，如教务系统中助学金推荐、学生成绩预测等。在未来随着数据量的扩充可结合 Top-N 推荐对模型进行修改以满足多分类需求。同时由于数据库使用 Django 框架的 ORM 映射，所以可以通过搭建 Django 的 apps 实现可视化操作。

本系统的主要难点在于如何进行用户特征提取，由于数据规模较小可用的评价指标维度不算丰富，所以对特征选择与推荐算法有效性构成较大挑战。在本文中将

分别对实验中使用的优化函数、目标函数、聚类算法、协同过滤算法等进行介绍。本文在模型构建处提出了一种特征融合方法，通过 XGBoost 将低维与高维特征进行融合，同时采用多种不同的机器学习算法进行推荐。除了使用传统的机器学习算法外，本文也应用 DeepFM 等深度神经网络进行模型构建与推荐。

第 2 章 算法理论

2.1 优化函数

梯度下降方法是本文中主要使用的优化函数算法。它是一种用于寻找可微函数局部最小值的一阶迭代优化算法。它的想法是在某点函数的梯度（或近似梯度）相反方向上重复执行上述步骤，因为梯度的下降方向是损失函数下降最快的方向，顺着梯度下降方向可以得到损失函数的局部最小值。反之沿梯度方向步进将得出损失函数的局部最大值，该过程称为梯度上升。梯度下降方法主要包括两大类，分别是固定学习率的梯度下降算法以及自适应学习率的算法。

2.1.1 固定学习率

恒定学习率的代表算法有 BGD、SGD、MBGD 等，这三种算法的主要区别即为使用训练集中不同数量的训练数据计算损失函数的梯度。

BGD（批量梯度下降）算法主要通过对整个训练集进行损失函数计算得出最优梯度。由于对训练集的每个样本都进行了梯度计算，所以 BGD 可以保证梯度下降方向为全局最优方向。但是在迭代过程中需要对每个参数求偏导，且在对参数求偏导的过程中还需要对训练集遍历一次，所以 BGD 的问题是整个训练计算量过大导致训练成本过高，而且不能投入新数据实时更新模型。BGD 伪代码如下：

Algorithm 1: Batch Gradient Descent

while *i in range(nb_epochs)* **do**

```
params_grad = evaluate_gradient(loss_function, data, params);
params = params - learning_rate * params_grad;
```

end

通过定义迭代次数 nb_epochs，首先计算 params_grad，然后沿着梯度的方向更新参数 params，其中 learning rate（学习率）的作用是决定了优化过程中梯度优化范围大小。

随机梯度下降（SGD）是另一种梯度下降迭代方法，用于以合适的参数（例如可微或亚可微）优化目标函数，可以将其视为批量梯度下降优化的随机近似值，因为它将实际的梯度（由整个数据集计算得出）替换为估计值（由训练集随机选择子集计算得出）。与 BGD 相比 SGD 在高维优化问题通过选取数据集的一个子集减少了迭

代过程中的计算量，从而以较低的收敛速度实现了更快的迭代。但是由于 SGD 的数据是从全部数据集中随机选出，所以 SGD 的噪音较 BGD 更多，这导致了 SGD 的下降方向不一定是向着整体迭代最优的方向。所以 SGD 虽然训练速度快，它的代价是训练准确度可能下降。

除此以外还有 MBGD 方法，它的原理是每次迭代过程中利用一小批样本（mini-batch）进行计算，MBGD 相比 SGD 可以降低参数更新时的方差从而使收敛更稳定，因为 MBGD 在迭代中每次使用一小批样本而不是一个样本进行梯度的计算与更新。

由于 SGD 的训练迭代速度更快，而且随机选择带来的负收益在大数据集上并不明显，所以它在大数据量上有较好的效果。所以绝大部分梯度下降方法的优化都是基于 SGD 及其固有问题实现。一般而言在机器学习中，设置学习速率（步长）太高会导致算法发散，而将其设置得太低会使收敛变慢。所以从概念上讲，随机梯度下降的简单扩展就是使学习速率成为迭代次数 t 的递减函数 $_t$，从而使学习速率随着训练迭代次数增加而调整。

2. 1. 2 自适应学习率

自适应学习率优化算法包括 Adagrad、Adadelta、RMSProp 与 Adam 等。

Momentum 算法是基于物理学中动量思想设计出的一种优化算法，由于 SGD 算法的学习率固定所以在收敛过程中存在震荡的问题，所以 Momentum 算法在梯度下降的过程中加入惯性的概念。它的主要思想是梯度的下降的方向由两个因素共同决定，分别是由当前点的梯度方向决定，还由此前的累积的梯度决定，在当前梯度方向与历史梯度一致时，会增强该方向的梯度从而加快收敛。

Momentum 算法虽然没有改变学习率但是从效果上讲起到了加快收敛的作用，并成为了 Adam 算法的基础。

在实验过程中，本文主要使用了 Adam(自适应矩估计) 优化函数进行梯度计算以及更新。Adam[11] 于 2015 年由 Diederik Kingma 以及 Jimmy Ba 在 ICLR 上提出。根据作者所述，Adam 是对 Adaptive Gradient Algorithm (AdaGrad）以及 Root Mean Square Propagation (RMSProp）的优化器的扩展。AdaGrad 算法核心在于可变的参数学习率。具体而言，对于频繁出现的参数，AdaGrad 算法会对其使用更小的更新速率；对于不频繁出现的参数使用更大的更新速率。AdaGrad 改善了对稀疏梯度问题的优化性能，在自然语言处理与计算机视觉等领域的深度学习模型中有优异效果。

RMSProp 算法则是对 AdaGrad 的进一步优化，它引入了一个衰减系数 r 并且此

系数在每个训练 epoch 都按一定比例衰减，从而解决了 AdaGrad 在深度学习中可能过早结束的问题。RMSProp 的第一个参数实际上与 AdaGrad 的第一个更新向量相同，同时 RMSProp 还引入了另一个参数即为学习率除以平方梯度的指数衰减平均值。RMSProp 算法适合处理非平稳的目标，对于 RNN 等网络的效果很好。

在 Adam 算法中，Adam 除了像 RMSProp 中那样根据平均第一矩（均值）调整参数学习率，同时也使用了梯度第二矩（无中心方差）的平均值（有偏方差）。具体而言，该算法计算梯度和平方梯度的指数移动平均值，并且使用两个超参数 β_1 和 β_2 控制这些移动平均值的衰减率。在移动过程中，均质的初始值和 β_1、β_2 的超参数的值接近 1，此时的误差接近 0，接下来通过计算带变差估计与偏差修正后的估计进行对比从而使得优化函数得到提升。Adam 本质上是带有动量项的 RMSprop，Adam 的主要优点在于经过偏置校正后，每一次迭代学习率都有确定范围，使得参数比较平稳。经过实验证明 Adam 在实践中比上述优化算法有更好的表现。

Adam 的主要参数包括 α（学习率）、β_1（一阶矩估计的指数衰减率）、β_2（二阶矩估计的指数衰减率）、ϵ（防止 $\hat{v}_t$ 不合适时导致震荡）。Adam 算法公式如下：

$$w_t = w_{t-1} - a * \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2-1}$$

其中 w 是当前的权重，t 是指修正次数，$\hat{m}_t$ 是 m_t 的修正，也就是修正一阶矩估计偏差，$\hat{v}_t$ 同理，即为二阶矩估计偏差的修正，a 是默认参数，通常取 0.001。其中 $\hat{m}_t$ ，$\hat{v}_t$ 的计算公式如下[11]：

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2-2}$$

其中 m_t 是梯度指数移动均值，即为 Momentum 项，v_t 是平方梯度，即为 RMSProp 项，g_t 为当前梯度，g_t^2 为梯度一阶导，m_t 、v_t 的计算公式如下[11]：

$$\begin{aligned} m_t &= \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \\ v_t &= \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \end{aligned} \tag{2-3}$$

2.2 目标函数

目标函数在广义上指经验函数 + 结构化函数，而训练模型的目的在于使经验风险与结构化风险最小化，经验风险即为在训练集上训练所得到的模型准确率高低，对于给定的训练集而言，对于模型拟合的越好那么经验风险（预测失败）的概率越低。但是需要注意的是模型的经验风险并非越低越好，因为在经验风险低的情况下虽然模型在训练集的准确率变高，但是模型可能因为拟合好而变得更为复杂从而缺少泛化能力。所以也引入了结构风险的概念，结构风险即为模型的复杂程度，降低结构化风险可以采取奥坎姆剃刀原则，在模型的经验风险与模型的复杂程度上进行取舍。过高的结构复杂度可能导致过拟合的情况，而过低的复杂度将导致模型训练不足产生欠拟合情况。而在本文实现的系统中由于推荐系统的衡量指标不仅仅是误差函数，同时也包括准确率、召回率以及 ROC 曲线或 AUC 值等，所以在这里以不包括结构化 objective function 定义目标函数。

对于一个优化问题而言，目标函数被定义为用来评估候选集结果（即一组权重）的函数。目标可能是最大化或者最小化此函数，意味着优化要在候选集中寻找有着相对最高/最低的目标函数值。具体而言在神经网络中，一般希望使误差最小化，在这种情况下，目标函数也常被称为损失函数或代价函数（cost function）。对于一个模型而言，损失函数的意义在于将一个复杂的系统所有优缺点降低为一个标量值，从而可以对候选解决方案进行排名和比较[12]。在本文使用的模型中主要使用的损失函数包括 mlogloss、binary-crossentropy 等。

首先目标函数分为两大类，一类是用于回归问题，另一类是用于分类问题，由于本文的系统主要实现的是分类功能，所以在这里着重介绍分类相关的目标函数。

2.2.1 最大似然估计

虽然有许多目标函数都可用于估计神经网络中一组权重的误差，但是从实验评估角度而言倾向于使用一种能够将候选权值的空间映射到平滑（高维）空间上，使得优化算法可以通过对模型权重进行迭代更新来更合理地进行权值修正。最大似然估计（MLE）是用于从历史训练数据中找到合适参数最佳统计估计的推理框架，这种方法正是在神经网络的修正工作中大量使用的。Neural Networks For Pattern Recognition 中曾指出：最大似然估计是通过计算训练集得到的似然函数而对参数进行优化[13]。最大似然估计损失函数是评估训练目标和模型预测值分布差异的方法。使用最大似

然估计的好处是随着训练数据的增长，模型参数预估的效果会提升，这是由于最大似然估计具有统计学中一致性的特点[14]。

2. 2. 2 交叉熵

从技术上讲，交叉熵（Cross-Entropy）概念来自于信息理论，并以“位”为单位。交叉熵是建立在最大似然估计的基础之上的，优化目标是希望通过使用交叉熵函数优化模型权重以最小化模型预测的概率分布以及训练集的概率分布之间的差异。除此以外，若在最大似然估计上使用高斯分布作为目标，MSE（最小均方误差）也可以被理解为模型预测与目标分布之间的交叉熵。所以在实际应用中，MSE 也同时被作为交叉熵的一部分用于回归问题而且在一部分分类问题的应用中也会使用 MSE 作为损失函数。在神经网络中，使用交叉熵损失函数的模型极大的提高了 sigmoid 和 softmax 层作为激活函数的输出神经元的效果。

对于二分类问题训练数据集中的一条数据而言，它存在一个已知的类别标签，概率为 1.0，所有其他标签的概率为 0.0，模型可以估计训练集中某数据属于每个列别标签的概率并且使用交叉熵来计算两个概率分布之间的差异。所以可以将数据标签映射到具有概率分布的随机变量上，若以 P 表示真实值，Q 表示预测值，那么每一条数据所代表的交叉熵公式[15] 就可表示如下:

$$H(P,Q)=\sum_{x}P(x)\log(\frac{1}{Q})=-\sum_{x}P(x)\log(Q(x)) \tag{2-4}$$

而由于对于一个特定的分类问题而言，输出结果可以用列向量 y=[y_1 ,y_2 ,...,y_n]T 表示，其中当 y_i 为预测判定时，y_i=1 其他为 0。此时 $p_j = y_j$，那么交叉熵公式可以如下表示:

$$H(P,Q)=-y_i\log(q_i)=-log(q_i) \tag{2-5}$$

那么对于具有 N 个样本的损失函数计算而言，它可以表示为如下形式[16]:

$$L=\frac{1}{N}\sum_{i}L_i=\frac{1}{N}\sum_{i}-[y_i\log(p_i)+(1-y_i)\cdot\log(1-p_i)] \tag{2-6}$$

y_i 表示样本 i 的 label，正类为 1，负类为 0;

p_i 为样本 i 预测为正的概率。

实际上在二分类问题中的交叉熵损失函数也被称为 logloss。

多分类问题本质上是对二分类问题的扩展，多分类问题的公式可以表示如下：

$$L = \frac{1}{N}\sum_{i} L_i = \frac{1}{N}\sum_{i} -\sum_{c=1}^{M} y_{ic}\log(p_{ic}) \tag{2-7}$$

其中 M 代表类别的数量

y_{ic} 代表指示变量, 如果该类别和样本 i 的类别相同就是 1，否则是 0；

p_{ic} 为对于观测样本 i 属于类别 c 的预测概率。

2.3 推荐系统评价指标

准确率、精确率和召回率是在推荐系统中对模型进行评价的几个重要指标，在这里首先要引入 TP、FP、FN 与 TN 的概念。TP 即为 True Positives，也就是将正类判断正确，FP 即为负类判断为正类，FN 也就是正类判断为负类，TN 即为负类判断为负类。

2.3.1 准确率

准确率即为所有判定正确的样本所占样本总数的比例，公式如下：

$$acc = \frac{TP + TN}{Total} \tag{2-8}$$

2.3.2 精确率

精确率是指所有判定为正类的样本中，真实正类所占比例，公式如下：

$$pec = \frac{TP}{TP + FP} \tag{2-9}$$

2.3.3 召回率

召回率 (真阳性率)，即指所有真实正类中，判定为正类的比例，公式如下：

$$rec = \frac{TP}{TP + FN} \tag{2-10}$$

2.3.4 κ 系数

除此以外还可以使用 κ 系数衡量分类效果。κ 系数是进行检验一致性的重要指标，对于分类问题而言，一致性即指模型预测结果与实际分类结果之间的差异，κ 系数是通过对混淆矩阵计算而得到的，取值在 [-1, 1] 之间，一般 κ 系数的值为正，当 κ 系数落在 [0, 0.20] 之间时，模型的一致性极低，[0.21, 0.40] 之间时，模型的一致性一般，[0.41, 0.60] 之间时，模型的一致性中等，[0.61, 0.8] 之间时，模型的一致性较高，[0.81, 1.0] 之间时，模型的一致性极高。κ 系数的计算公式如下：

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{2-11}$$

其中 p_0 即为 accuracy，p_e 为

$$p_e = \frac{\sum_i \text{第 } i \text{ 行元素之和} \times \text{第 } i \text{ 列元素之和}}{(\sum \text{矩阵所有元素})^2} \tag{2-12}$$

2.3.5 ROC-AUC 曲线与 AUC 值

ROC 曲线为 FPR 与 TPR 之间的关系曲线，其中 FPR 指假阳性率，也就是所有负样本中分类器将负样本预测为正样本的比例，而 TPR 指真阳性率，也就是 Recall 值。ROC 曲线的定义为以 FPR 作为 x 轴 TPR 作为 y 轴的曲线，这个组合总 FPR 对 TPR 即可理解为代价对收益，通过改变不同的阈值可以得到一系列 TPR 与 FPR 并绘制出 ROC 曲线。而 AUC 值即是指 ROC 曲线与坐标轴围成区域的面积，AUC 值反映了分类器对样本的排序能力。FPR 与 TPR 的公式如下所示。

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{2-13}$$

AUC 的计算公式如下所示：

$$AUC = \frac{\sum_{ins_i \in positiveclass} rank_{ins_i} - \frac{M \times (M+1)}{2}}{M \times N} \tag{2-14}$$

$rank_{ins_i}$ 指按照预测为正样本的 prob 大小进行排序的第 i 个样本序号，其中 $\sum_{ins_i \in positiveclass}$ 是指所有属于正样本的数据符号之和，即为排序后的正样本序号之和，M 和 N 分别代表正样本个数与负样本的个数。

2.4 特征降维算法

在本文中使用了主成分分析、t-SNE 等传统降维方法进行了特征降维，除此以外还使用了聚类算法将多个特征以及相似的高维数据压缩到低维特征空间。分别使用了 K-Means、谱聚类以及 HDBSCAN 三种算法来进行聚类结果比较。其中 KMeans 是一种常用的机器学习算法，不做赘述。

2.4.1 主成分分析

主成分分析（PCA）是一种数据挖掘领域常用的线性特征提取降维方法。它的思想是通过将一大组变量转换为较小的变量（仍包含大数据集中的大多数信息）来减少大型数据集的维数。PCA 的原理是通过寻找原数据集中正交的坐标轴将高维数据压缩到低维并确保原始数据在低维特征中的方差尽可能大而保留并还原高维特征。

2.4.2 t-SNE

t-SNE[17] 是一种非线性的降维技术，它的思想与 PCA 类似。但是 t-SNE 原理上是以二维或者三维的点对高维对象进行建模，使得相似对象大概率由临近点表示，而差异对象由距离较远的点表示。

2.4.3 HDBSCAN

DBSCAN 是一种常用的基于密度的聚类方法，HDBSCAN[18] 相比于 DBSCAN 将传统的密度聚类转换为分层聚类扩展了 DBSCAN，然后基于聚类稳定性使用平面聚类提取技术提高了聚类效果的鲁棒性，最为重要的是基于密度的聚类算法可以通过指定最小生成类簇的大小自动推荐最优簇类结果生成最优聚类簇数 k，同时 HDBSCAN 相比于 DBSCAN 不用选择人工选择领域半径 R 和 MinPts（最小点集）。

2.4.4 谱聚类

谱聚类[19] 是一种来源于图论的算法，它将数据看做空间中的点，这些点之间可以用边连接起来。连接点的边的权值大小和两个点之间的距离有关，距离越近权值越高，距离越远权值越低。通过对所有数据点组成的图进行切图，目标是使得切图后不同子图间边的权重和尽可能的低，子图内边的权重和尽可能的高而实现聚类。谱聚类通过这些图上的点映射到一个低维易于分离的空间从而逐渐实现距离度量并最终聚类。谱聚类使用来自图形或数据集的特殊矩阵（即距离矩阵（Affinity Matrix），度矩阵（Degree Matrix）和拉普拉斯矩阵（Laplacian Matrix））的特征值（频谱）中

的信息。谱聚类是一种优秀的算法，较 KMeans 相比谱聚类对数据分布适应性更强，聚类效果也更优秀，同时计算量也小很多。

2.5 推荐算法

选择合适的推荐算法是一个推荐系统成功与否的关键，对于这个问题本文主要使用了 3 种不同的推荐算法，分别涵盖了最基础的机器学习算法–贝叶斯分类器，推荐领域大显身手的决策树模型–XGBoost 以及基于深度学习并且可以摆脱手动特征工程困扰的典型模型–DeepFM。

2.5.1 贝叶斯分类器

贝叶斯分类器是以贝叶斯定理为基础的简单概率分类器，它通过数据分布的先验概率利用贝叶斯公式计算后验概率，本文推荐中使用了三种不同的朴素贝叶斯分类器，分别是高斯、伯努利以及多项式。它们的区别是在使用最大似然估计对数据进行估计时假设的数据分布概率函数不同。

2.5.2 XGBoost

XGBoost[20] 是使用梯度提升（Gradient Boosting）的基于决策树的集成机器学习算法。在涉及非结构化数据（图像，文本等）的预测问题中，人工神经网络往往胜过所有其他算法或框架。但是，当涉及中小型结构化/表格数据时，基于决策树的算法目前被认为是同类中最好的。

虽然 XGBoost 与梯度提升算法一样，都是使用 Boosting 算法的思想将许多弱分类器如 CART 回归树模型等集成在一起在梯度下降的优化下形成一个强分类器，但是 XGBoost 通过对 GBM（Gradient Boosting Machine）进行改进从而提升了算法性能，具体的提升可以分为两大方面，分别是性能优化以及算法优化。

XGBoost 的系统性能优化如下：

1. 并行化: XGBoost 通过并行计算来实现树的构建过程。以传统的决策树构造过程为例，它包括内外两个循环，外层循环用于枚举一棵树的叶子节点构造，内存循环用于特征计算。循环嵌套限制了算法的并行化能力，因为如果不完成内部循环（计算量更大），就无法启动外部循环。因此交换内外层循环的顺序可以减少运行时间，循环的交换通过初始化多个运行在所有实例上的扫描与排序线程实现。交换通过并行计算减少算法运行时间。

2. 剪枝算法: 梯度提升框架中决策树分裂的停止准则本质上是贪婪的，并且取决于分裂时的负损失准则。XGBoost 使用“max_depth”参数代替 GBM 的分裂标准，然后开始向后剪枝。这种深度优先的方法显著提高计算性能。

3. 存储优化: XGBoost 通过在每个线程中分配内部高速缓冲区（Cache）来存储梯度统计信息来实现的。同时使用诸如“out-of-core”等方法增强对超大数据帧（DataFrame）的处理，并且进一步优化了内存和磁盘空间使用。

算法优化如下：

1. 正则化: XGBoost 通过使用 L1 和 L2 正则化参数计算模型的结构复杂度并防止过拟合。

2. 稀疏感知: 为解决稀疏特征问题，为每个树节点选择默认方向，如果某个样本对应特征缺失，直接划入默认方向[21]。

3. 加权分位数: XGBoost 通过加权分类数算法（Weighted Quantile Sketch）寻找最优分裂点。

4. 交叉验证: 算法通过内置交叉验证来获取最优参数并防止过拟合。

2. 5. 3 DeepFM

CTR 预测大规模应用的算法最初主要是 SVM 和 LR，但是 LR/SVM 这样的线性模型无法建模非线性的特征交互，于是需要手工进行特征交叉或组合，而这个过程是复杂而且需要大量专业人员的参与；为了弥补 LR/SVM 无法处理非线性特征交互的问题，因子分解机（Factorization Machine）这种方法被提出并得到大规模应用。

FM[22] 的模型由 Konstanz 大学 Steffen Rendle 于 2010 年提出，旨在解决稀疏数据下的特征组合问题。在 CTR 推荐过程中，主要存在三类数据，分别是用户的特征、广告的特征和上下文特征，这些特征很多是离散的类别特征，而这些离散的特征在使用 one-hot encoding 后将使特征空间急剧增大，而且变得稀疏，稀疏矩阵使得参数的计算变得困难。为了解决上述问题，FM 的做法是通过矩阵分解的方法将评分矩阵分解为用户矩阵和商品矩阵，其中用户和商品都可以使用隐向量进行表示[23]；如可以将 User 表示为一个二维向量，同时将 Item 表示为二维向量，那么两个向量的点积就是 User 对 Item 的打分，如图2-1所示。

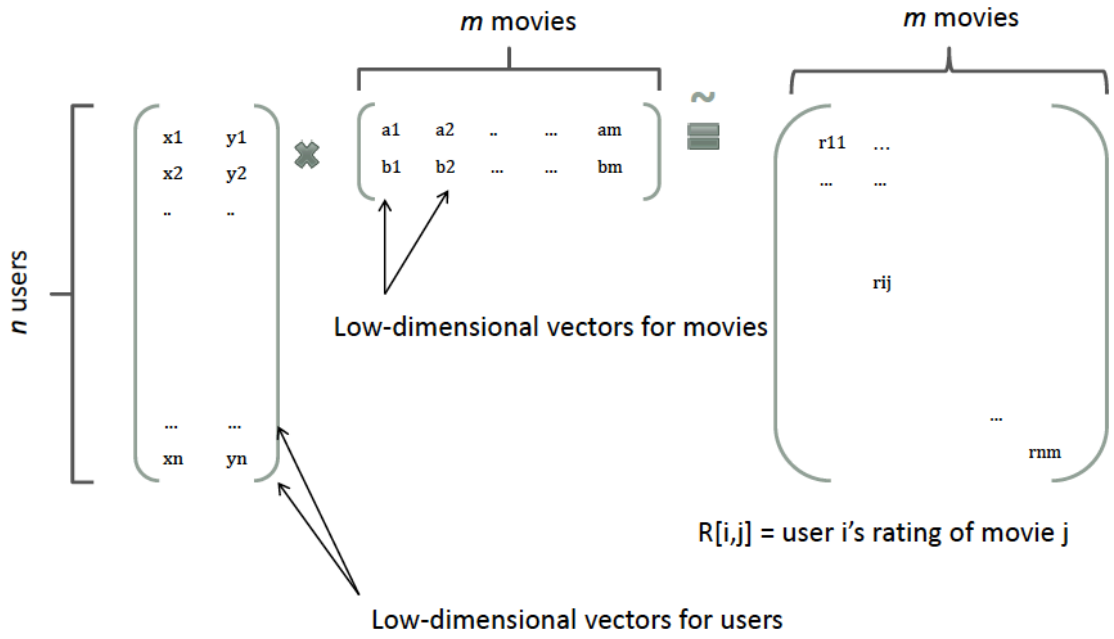

图 2-1 FM 矩阵分解[24]

类似地，所有二次项参数 W_{ij} 可以组成一个对称阵 W（为了方便说明 FM 的由来，对角元素可以设置为正实数），那么这个矩阵就可以分解为 $W = V^{\mathrm{T}}$，V 的第 j 列便是第 j 维特征的隐向量。换句话说，每个参数 $w_{ij} =< v_i, v_j >$，这就是 FM 模型的核心思想。因此，FM 的模型方程为（本文不讨论 FM 的高阶形式）：

$$y(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j \tag{2-15}$$

FM 的训练复杂度为 $O(kn)$ 级别，可以在线性时间对新样本进行预测。FM 通过给单独的特征引入低维稠密隐向量减少了二阶特征交互的参数，并且在这个过程中实现了信息共享，从而使稀疏特征向量的学习更加容易泛化。理论上 FM 可以建模二阶以上的特征交互，但是由于二阶以上的参数数量将会急剧增加而且更高维的特征交互可以使用 DNN、DCN 等神经网络模型实现，所以一般上只用二阶的模型。

但是由于 FM 模型也存在特征无法区分的问题，比如对于奖学金推荐系统而言数据的成绩特征与总学分的组合与论文的发表数量和论文的所属期刊的组合潜在意义是相同的，而 FM 模型无法捕捉这样的差异，所以又提出了 FFM 模型。FFM 即为 Field-aware Factorization Machines[25]，它在 FM 模型的基础上引入了 Field 的概念，对

于其中的每一维特征，都属于一个特定的 Field，Field 与 Feature 之间存在一对多的关系。具体的实现上看，由一个 Categorical feature one-hot 出的多个类别特征属于同一个域（Field），而连续的特征不需要进行 one-hot 自己作为一个单独的域存在。FFM 的公式如下：

$$\phi_{FFM}(w,x)=\sum_{j_1=1}^{n}\sum_{j_2=j_1+1}^{n}(w_{j_1,f_2}\cdot w_{j_2,f_1})x_{j_1}x_{j_2} \tag{2-16}$$

其中 x 是 n 维特征向量，f_1 f_2 代表 j_1 j_2 对应的 field，j_1 j_2 分别是特征向量中的两个特定 index 的值。FM 和 FFM 对比而言，FM 只有一个代表特征的向量，而 FFM 有一个代表特征向量和一个代表域的向量，也就意味着同一特征，要和不同的 fields 进行组合的时候，会用不同的 embedding，因而 FFM 参数更多。对于一个 vector 的特征，现在拓成 F 个 vector，只要跟它任意组合，就有一个 vector 来代表，这就是 FFM 的基本思想。

可以看到无论是 FM 或是 FFM，都是对两个特征进行的特征组合，那么如果有更高阶的特征组合，就需要更复杂的模型进行训练，随着深度神经网络的发展，研究发现可以使用多层的网络结构进行更好的高阶特征表示，从而学习高阶的非线性关系。而深度神经网络为代表的模型分化出了两种发展思路，一种是并行结构，一种是串行结构。

对于深度神经网络而言，都是首先将特征输入进行 embedding 后连接隐层进行预测，这一部分是一个公有结构，而另一个公有结构为上述介绍的 FM Function，负责学习模型的二阶特征组合。串行与并行模型的区别就在于 FM 和 DNN 的关系是并行（FM 与 DNN 同样接受 embedding 层的输入后将两者结果进行融合）或是串行（FM 先进行二阶特征组合后将结果输入 DNN 进行更高阶组合）。

并行结构的代表模型有 Wide & Deep、DeepFM、NeuralFFM、DeepFFM。

串行结构的代表模型有 DeepCross、xDeepFM。

首先提出的并行结构为 Wide & Deep[26]，它于 2016 年由 Google 发布，它分为两个部分，分别是 Wide 部分以及 Deep 部分，其中 Wide 部分的作用是通过交叉特征实现高效的记忆能力，达到准确推荐的目的，使用 LR 等广义线性模型实现特征交叉。Deep 部分则是一个前馈神经网络模型，由于基于嵌入的模型（如 FM 与 DNN）能将输入转化为一个低维稠密向量从而获得了一定的泛化能力，减轻了特征工程的负担。embedding 得到的向量作为 Deep 部分第一层隐藏层的输入并经过多层隐藏层厚并根

据得到的 loss 进行训练。此模型同时学习了低阶特征与高阶特征并且在输出层对上述 Deep 以及 Wide 的结果进行加权求和，在 Google Play 离线测试以及在线 A/B 测试中都取得了超越 Deep 或 Wide 的效果。但是 Wide 模型存在需要手工特征处理的情况，为了解决这一问题，华为联合哈尔滨工业大学提出了一种改进模型–DeepFM[27]。

DeepFM 结合了 FM 和 Wide & Deep 的特点，并通过共享 FM 和 DNN 的 Embedding 来减少参数量和共享信息，并且通过引入 FM 克服了手工特征处理的障碍。针对高维稀疏输入特征，DeepFM 采用 Word2Vec 的词嵌入思想，把高维稀疏的向量映射为相对低维且向量元素不为零的空间向量。DeepFM 的网络结构如图2-2所示。

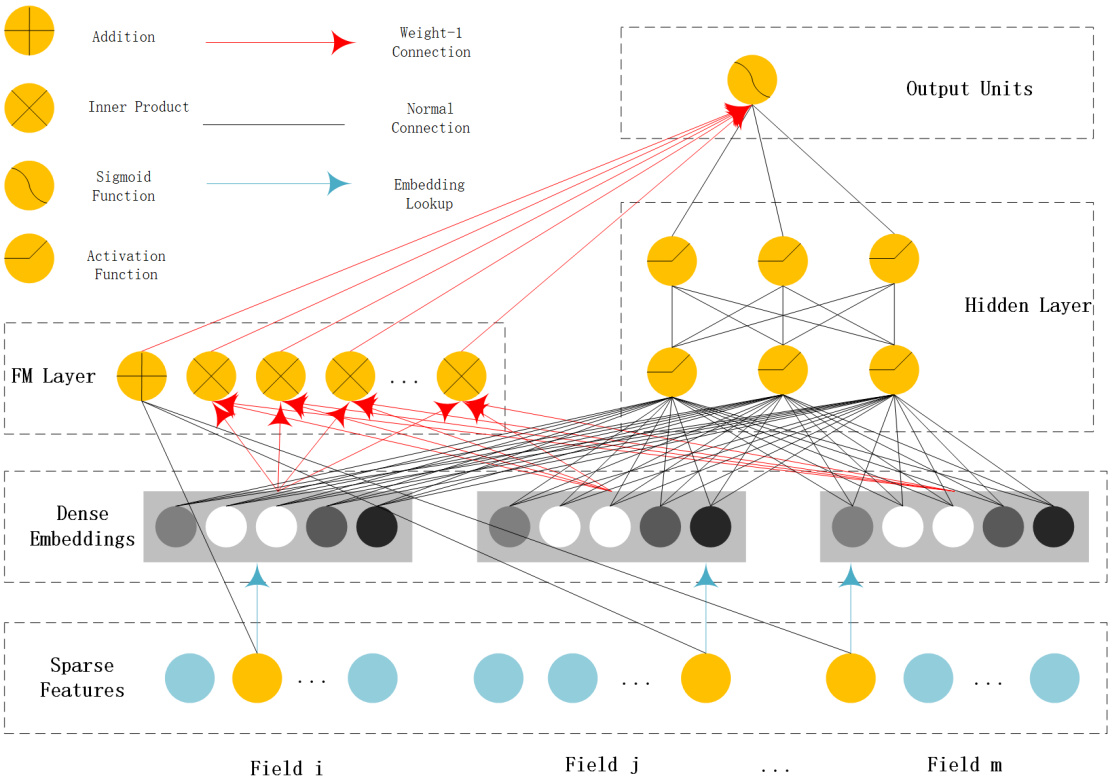

图 2-2 DeepFM 结构图[28]

可见 DeepFM 与 Wide & Deep 的框架极为类似，差异在于在 Wide 部分的特征交叉输入项被替换为 FM 部分，FM 代替 Wide 部分的手动特征工程构造二阶特征叉乘，从而实现了低阶的自动特征工程。DeepFM 的 Wide 与 Deep 模块共享相同的由各个 Field 的 one-hot 编码横向拼接而成的高维稀疏向量输入；除此以外 FM 层与 NN 层共享相同的 embedding 层，从而降低了模型的复杂度，并且在 embedding 层训练过程中

同时接受低维以及高维特征交互的反馈。在 DeepFM 的实验中作者对 embedding 层分别进行了共享与不共享的对比，发现共享效果更优。网络整体的输出为:

$$\hat{y} = sigmoid(y_{FM} + y_{DNN}) \tag{2-17}$$

FM 层需要注意的是 FM 中引入的辅助向量 V 现在为网络权重的一部分并且这些权重被学习并用于将 Embedding 输出的向量进一步压缩为嵌入向量。与原有的 FM 中辅助向量 V 是 FM 预训练得到并用于网络初始化不同，DeepFM 中没有使用辅助向量 V 来进行初始化，而是将 FM 作为网络结构的一部分。如此一来参数无需通过 FM 进行预训练，而是以端到端的方式联合训练整个网络。FM 对应的输出如下:

$$y(x) = \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j \tag{2-18}$$

FM 部分的结构如图2-3:

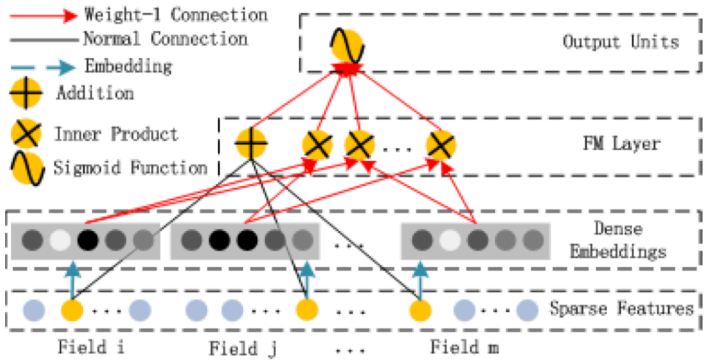

图 2-3 FM 结构[27]

Embedding 层的目的是对输入的稀疏特征矩阵进行嵌入转为稠密低维向量，因为 DNN 部分的输入和图片/音频等低维特征不同，CTR 的数据集比较稀疏，而且其中有大量 one-hot 后的离散类别特征，并且离散的特征和连续的特征混合在一起，所以需要使用 Embedding 层降维，通过 Embedding 后得到的低维稠密向量输入第一层隐含层。这样做可以防止网络参数过多导致网络训练时间长运算开销大。需要注意的是无论输入的某个 Field 的特征的长度如何 embedding 后的维数都一样。Embedding 层对应结构如图2-4:

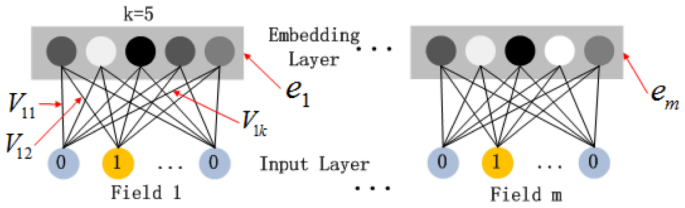

图 2-4 Embedding 层结构[27]

embedding 层的输出如下所示：

$$a^{(0)} = [e_1, e_2, ..., e_m], \tag{2-19}$$

其中 e_i 代表第 i 个 Field，m 为 Field 的个数，$a^{(0)}$ 是 FM 与 DNN 部分的输入。

网络的 Deep 部分是一个前馈神经网络，用于学习高维特征交叉。DNN 部分的输入是 embedding 层的输出向量，DNN 部分的结构如图2-5：

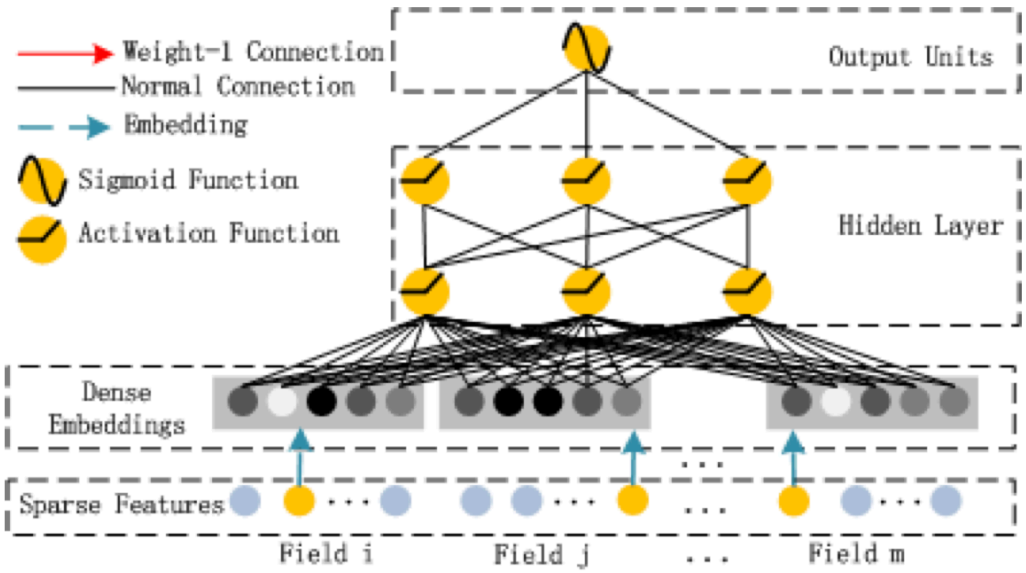

图 2-5 DNN 结构[27]

2.6 本章小结

本章主要介绍了模型使用的目标函数、优化函数、评价指标等内容，并介绍了一些数据降维的方法。除此以外还介绍了推荐使用的算法等内容。其中介绍的绝大部分函数以及算法将会在后文中出现。

第 3 章 数据预处理和数据仓储

3.1 原始数据分析

本文的数据来自于计算机学院提供的学生数据文件，其中分为两大部分，分别是本科生信息以及研究生信息，但是其中本科生信息只包含成绩信息，而研究生信息维度较为丰富，所以使用研究生信息进行建模和推荐。其中包含 2016 与 2017 两级信息。

数据来源是教务处提供的包含学生数据的 excel 文件，文件共有 4 个，内容分别是本科生的成绩信息、研究生成绩信息、研究生学业奖学金信息以及研究生国家奖学金信息。

3.1.1 本科生信息字段

本科生成绩信息字段 学生 ID、选课课号、学年学期、课程代码、课程名称、课程性质、成绩、折算成绩、课程归属、重修标记、学分创建时间、绩点、课程属性、课程种类、考试性质、年级、学院

3.1.2 研究生信息字段

研究生成绩信息与此类似，但不同的是研究生多了一个第二课堂的 excel 表单，研究生与本科生信息的主要差别在于奖学金信息部分，研究生学业、国家奖学金信息均提供了 8 个表单，分别是论文发表、专著论文出版、发明专利、科技获奖、科研项目、创新竞赛、荣誉称号以及其他成果，同时提供了对应每条记录的奖学金获奖情况。研究生信息字段如下所示。

研究生信息公有字段 学生 ID、学业奖学金申请等级、状态、学院培养层次、性别、民族、入学时间、学科、是否专业学位

研究生数据的论文字段 论文题目、刊物会议名称、作者排序情况、论文收录情况、论文层次、中科院 JCR 分区大类、中科院 JCR 分区小类、他引情况、录用发表状态、发表时间、作者排序情况

专利字段 专利名称、专利类别、专利申请号、专利申请时间、专利证书编号、专利批准日期、专利持有单位

科技获奖、创新竞赛获奖字段 奖项名称、主办单位、所获奖项、获奖级别、获奖日期、奖项等级、总人数、个人排名、获奖证书编号、颁发证书单位

科研项目字段 项目名称、项目类型、工作任务

出版专著字段 著作名称、出版社、工作量、书号、出版日期、编著类型、作者人数、第几作者

荣誉称号、其他成果称号字段 荣誉名称、荣誉级别、颁发单位、获得日期

3.2 数据库模型构建

以上文的 Excel 数据分析为基础，这里主要对 Django 的数据库模式进行建构，Django 是 Python 语言下一重要同步 B/S 服务器框架，使用 MVC 模式进行设计，在 Django 中有专门 models 文件对数据库模型进行定义，Django 通过封装好的 ORM 映射对数据库进行操作。由于给定的数据源（excel 文件）是关系型结构，所以本系统选用了关系型数据库 SQLite 进行数据库构建。

数据库模型构建首先需要对基础信息进行设计，基础信息按照特定构建顺序主要包括学年（期）信息、学院信息、专业信息、课程信息、学生信息。其中学年信息和学院信息是一切信息的基础，其他的信息或多或少都构建于这两个信息之上，下面对这些表中的字段与字段含义进行介绍。

3.2.1 基础信息表

学年（期）信息 包括学期的名称、学期起始日期、学期终止日期。通过学期信息进而转换成学年信息与数据表中的信息匹配。

学院信息 主要包括学院名称。

专业信息 包含专业名称、学生数量、学年 ID、学年信息（大一、研一等）、学期信息以及学院信息，学年 ID、学期信息以及学院信息是此表的外键，通过对学年信息进行筛选可以唯一的确定一条专业信息记录。虽然这张表可能存在一定冗余（专业名称、学生数量等）但是为了减少表之间取并的次数所以如此进行设计，未来可以将此表优化为拉链表节省存储空间加快检索速度。

课程信息 包括课程名称、考核方式、课程性质、学分、课程种类、学年信息等内容，主要课程的基本信息，其中有一额外生成表为 lessoninfo_affiliatedmajor，此表

生成原因是一门课程可能对应于多个学院（如高数等公共基础课），而一个学院也对应多门课程，所以产生了多对多的关系，在 Django 中自动将此多对多关系映射为一张额外系统表，表中内容是 lesson_info 表和 affiliated_major 表的主键关联信息。

学生信息 包括学生学号信息、是否为研究生、系统的登录名称、预留邮箱地址、登录密码等，学生信息与专业信息通过一额外多对多表相连，由于学生是一个实体，专业同样是一个实体，故将其之间的关系提取构建为单独关系表，在 Django 的代码中以 through 的形式表示此关系，下个条目对此关系表进行解释。

关系信息表 由上文可知关系信息表是通过 through 代码生成，Django 中关系表必须包括关联的两张表的主键作为此表外键且缺一不可，所以其中包含了学生的 ID（应为对应的用户 ID 而非学号）以及专业信息 ID，并且设置了冗余学年信息避免通过专业信息查找学年信息的连接操作提升了执行速度，同时设置关系属性字段，包括研究生、本科生等不同关系类型以及学年信息，包括研究生一年级、本科生一年级等不同信息区分，并且在最后设置起始日期与终止日期字段，使此表成为一个可通过日期查询的拉链表。

如上是对基础信息表的介绍，接下来是对学生部分的个人信息表进行构造。

3. 2. 2 个人信息表

个人信息表共包括 8 部分内容，其中有 7 个表是从所给数据文件中直接得到，分别是论文发表、专著论文出版、发明专利、科技获奖/创新竞赛、科研项目以及荣誉称号/其他成果，同时包括一个存储了学生奖学金获奖历史表，论文发表等表字段内容与 excel 所给字段内容一致不再赘述，值得注意的是有些空字段需要额外处理才能向数据库中录入内容，除此以外科技获奖、创新竞赛的表以及荣誉称号、其他成果表的字段内容基本一致，所以通过一个类型字段进行区分即可省略一张表，同时学生与奖学金信息同样构成了一个关系即为学生的奖学金获奖历史，这个表同样可以看作多对多关系生成的一个表，所以会额外生成一张带有学生 ID 和奖学金 ID 作为外键的表。

3.2.3 奖学金表

接下来是奖学金表的构造，因为之前考虑将系统设计为 B/S 架构，所以需要设计带有奖学金信息的推送网页内容，故设计奖学金信息表、奖学金信息分级表、奖学金网页内容表，这其中奖学金信息表主要包含奖学金的名称、奖励金额等，同时未来还可扩充额外要求字段。奖学金信息分类表主要是由于某个类型的奖学金可能有多个分级，需要对不同的分级进行考虑，此表能够减少冗余，并且在推送内容填充 URL 的时候可以直接根据奖学金是否有分级填充不同的 URL 提高奖学金在网页推送部分的效率，除此以外还可以增加数据库的层次化结构。同时表中的推送内容使用 Markdown 格式，方便如 Ueditor 等富文本编辑器进行编辑。同时可以使用 Python 的 word2md 包的 Markdown 转换工具将 word 文件上传自动转为 Markdown 方便在线编辑使用。

3.2.4 数据库 E-R 图

在完成了数据表的设计后，数据库生成的个人信息与基础信息 E-R 图如图3-3。

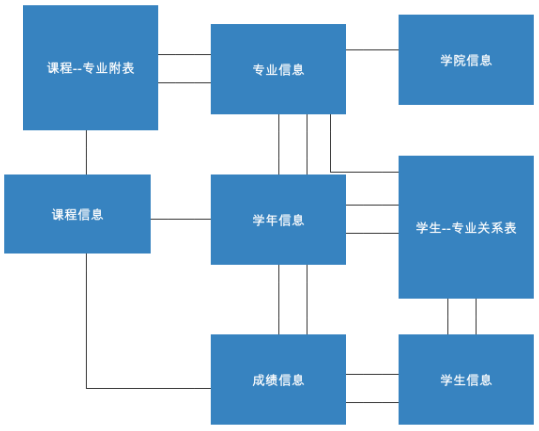

图 3-1 基础信息 E-R 图

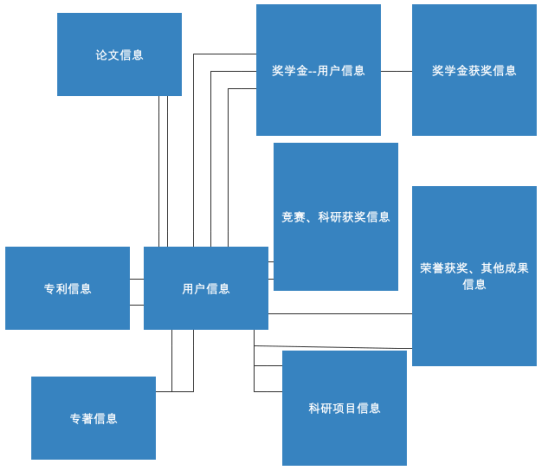

图 3-2 个人信息 E-R 图

图 3-3 数据库 E-R 图

3.3 数据导入

向数据库中填充数据使用的是 Django 自带的 ORM 映射，通过调用接口而非注入 SQL 命令实现更安全的数据操作方式，所有与数据填充有关的代码均在 utils

中，其中首先需要调用 os.environ.setdefault 命令读取 Django 启动配置，然后使用 django.setup() 命令实现 Django 的初始化，Django 启动后可以通过引入不同的 model 对模型直接进行操作，同时还使用了 python 的 xlrd 包对 excel 文件进行读取。

数据库填充首先填充的是学年、学院、专业、课程以及学生信息，在填充学生个人的时候使用集合结构对所有数据表中的学生 ID 进行去重统计，除此以外还需要注意对某些特殊字段进行处理，包括对空字段进行填充等。由于在不同 apps 的 model 中设置好了部分分类字段的映射（分类字段是以 CharField 的形式进行保存，用元组设置了映射规则），所以需要用字典对文本进行处理与转换，最终在数据库中保存缩写。

3. 4 本章小结

本章主要内容是对 excel 文件进行分析并且根据不同数据表的特征进行数据库构建。数据库建模主要分为四大部分，分别是用户信息、与奖学金申领有关的个人信息、基础信息以及奖学金信息。具体的数据库模型如 GitHub 仓库中 E-R 图所示。数据填充部分使用 Django 关系数据库 ORM 映射的 API 进行数据填充，这样可以避免使用 SQL 注入语句从而提高数据库的安全性。除此以外还对数据库导入的数据信息进行了初步统计，以下是对导入内容进行统计：

表 3-1 数据库内容统计

	总人数	获奖人数	奖项个数	奖项个数 >10
研究生信息	1469	186	9	4
本科生信息	1441	无	无	无

可以看到只有研究生有奖学金获奖信息，而由于标签个数问题其中可供聚类的标签数量不超过 4 个，而且在数量大于 10 的 4 个标签中，有两种为无法推荐的奖学金，分别是硕士一年级学业奖学金于硕士新生奖学金。由于给定的数据集中缺少有效的时间特征，所以无法对这两种奖学金进行推荐。另外两种为硕士一等学业奖学金与硕士其他等级奖学金，本文主要工作即是对这两种奖学金进行推荐。

第 4 章 特征构建

特征的选择与构建基本上是整个推荐系统的核心所在，也是这个项目中最为复杂的部分，在项目特征的选择上，我们进行了很多的探索，接下来将一一介绍我们的工作。

4.1 特征分析

首先最简单的特征构建方法就是按照经验公式直接将学生的各项指标换算成评分来作为训练数据进行推荐，甚至于可以直接按照分数高低进行推荐，这种方法也是本科生奖学金评选经常使用的一种方法，那么在这个系统中，由于缺少对应研究生已有的经验公式，所以我没有对这部分内容进行实验，而是选择了其它方法。

对于奖学金推荐系统而言，因为推荐使用的特征是多维度的，包括论文、竞赛等诸多部分，所以我选择单独对每种类型的特征进行考虑，由于对应一个学生而言，他可能在一篇论文中有不同的作者等级，发表的论文会有不同的期刊（会议）等级，所以这些都应当作为特征构建中的评价指标，除此以外还应当考虑一个极其重要的问题即为一个学生可能有多篇论文的情况存在，而且这种现象确实已经在系统中出现，在系统中有的学生论文数量达到了 3 篇，更有甚者在荣誉称号一个单项上就存在 7 项之多，如何对其进行特征构建就成为了一个棘手的工作，如果要囊括所有学生的记录的话，那么这个特征矩阵将及其庞大，因为对应绝大部分学生来说，在某个单项（论文等）只有一项纪录，而要表示完整的单项需要的特征向量维度较高，如果只考虑竞赛一项按单人最大数量来算，那么只竞赛的表示就至少有 7*n（n 代表对每个记录选择的特征维数，如论文中的期刊等级以及作者排序等）个维度，所以最后我产生了两种思路，第一种思路是尽可能多的保留各单项信息，每个单项最多选取 3 个有价值的记录构成特征向量，或者将一个单项的所有信息累加，用公式换算为得分或者对单项信息进行无监督的聚类，将聚类标签作为特征输入推荐算法进行训练。

以上是产生的两种思路，我主要实验了第一种思路，也就是按单项最大个数对记录进行保留，但是我并没有选择简单的对特征进行筛选后直接构建特征向量，而是同时对特征使用聚类的方法以期减少特征的数量以及训练的复杂度。在这里以论文的特征向量构建为例展开分析。

以论文的特征向量构建为例，论文的特征向量主要包括论文刊载的期刊（会议）等级以及学生的作者排序，这两个数据是我们已有的，但是在给定我们的 excel 文件中，可以看到其他还存在一些额外的信息，比如论文期刊层次的中科院 JCR 分区大类、中科院 JCR 分区小类，以及论文他引情况等，至于为什么没有选择剩下的这 3 个特征进行构建，是因为对于绝大部分论文而言，前两个特征已经足够进行评级并且剩下的 3 个特征存在的问题分别是 JCR 分区大类和分区小类没有包括部分国际期刊的分类而完全归于其他分区，导致对部分国际、国内和部分国际、国内会议的区分不够明确，他引情况的问题是他引情况可能随时间变化较大，如较早发表的论文的引用数量较多，而较晚发表的论文可能引用数量不足或根本为 0，同时如何获取论文引用数量也是一个很困难的现实，因为我一开始曾经考虑对知网信息进行爬取获得论文的关键词、摘要等按照中文 NLP 方法以及 one-hot 编码对论文内容进行简单分类，但是一方面知网有完整的反爬机制除此以外对知网内容进行爬取可能会导致学校 IP 被暂时封禁，所以最终放弃了这种想法。但是我们选择了使用聚类的方法对论文数据进行处理，接下来将对聚类的实现展开介绍。

4. 2 聚类方法

4. 2. 1 PCA/TSNE 特征降维

聚类使用的主要方法均为无监督聚类，以典型的 KMeans 方法为例，我们首先对论文数据进行了清洗，并且保留了聚类所用到的两个特征–论文作者排序以及期刊（会议）等级，并且使用 PCA、t-SNE 两种方法对数据进行了主成分分析、降维以及可视化处理，同时我们使用了 one-hot encoding 以及 label encoding 两种方法将文字特征转化为数值特征，理论上而言期刊（会议）等级和作者排序均存在大小关系，所以我们构建的 label encoding 也是以高等级期刊、高作者排序对应数值低而进行，因此我们自主设定了排序等级，以期刊（会议）等级的 label encoding 为例，mapping 如图4-1所示：

```
In [65]: conference_mapping = {
             'tm': 1, # 顶级期刊
             'ic': 1, # 国际会议
             'om': 2, # 国际期刊
             'im': 2, # 重要期刊
             'nc': 3, # 国内会议
             'nm': 3, # 国内期刊
             'os': 4, # 其他SCI期刊
             'ot': 5, # 其他
         }
```

图 4-1 论文期刊 (会议) 的 Mapping

由上可知期刊（会议）等级排序共 8 类，所以我们对其进行了一一映射，同时也可以对其进行 one-hot encoding，对 one-hot encoding 后的样本进行 t-SNE 可视化以及主成分分析后的结果如图4-4所示：

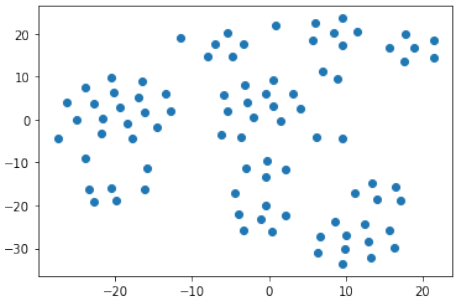

图 4-2 t-SNE

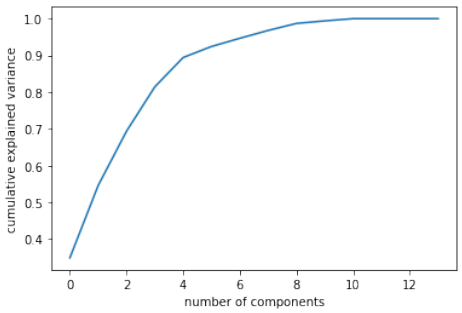

图 4-3 PCA

图 4-4 one-hot 编码的 TSNE 与 PCA 可视化结果

可见主成分在 6～8 左右 (one-hot 后的特征向量) 对原始特征的还原度较好，在这里可以选择对主成分进行提取；而 TSNE 的可视化结果则显示样本呈现某种特殊的分布规律，初步验证数据呈现规律分布可以进行聚类。

4. 2. 2 K-Means 聚类

接下来我首先进行的是使用 KMeans 方法聚类，根据 KMeans 的官方文档，对类别特征最好进行 one-hot encoding 后去除自身的欧几里得距离特征进行聚类更为合适，虽然 KMeans 对噪声敏感而且初始点的选择可能会影响聚类效果，但 KMeans 作为一种容易理解简单易懂而且执行速度较快的聚类算法在这种小量数据集上可以有较好的表现，所以我首先选择了 KMeans 对其进行聚类。由于 KMeans 要手动选取聚类类数 k，所以我首先进行了 k=5 的聚类，效果虽然不太理想但是证明了 KMeans 在这个数据集是有效的。

接着我对 KMeans 的聚类结果进行了可视化展示，分别展示了聚类不同类中数据个数以及 PCA 主成分分析后提取的主成分分布图，其中左图展示的是聚类个数以及聚类的标签之间的关系，右图是 PCA 的主成分分布情况，其中上面两图是 label encoding 的结果，下面是 one-hot encoding 的结果。从图中来看 label 中有一个类的数量极多，这个类根据后续的查询也是所有论文中最多的两个成分组成的类，分别是第一作者以及国际会议。结果如图4-9所示：

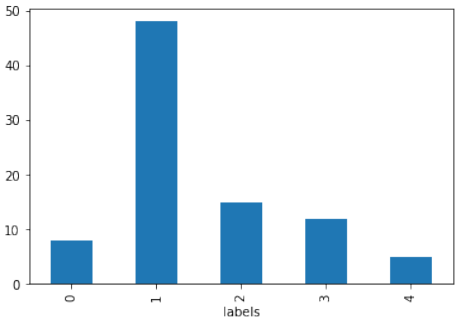

图 4-5 label-encoding 标签分布

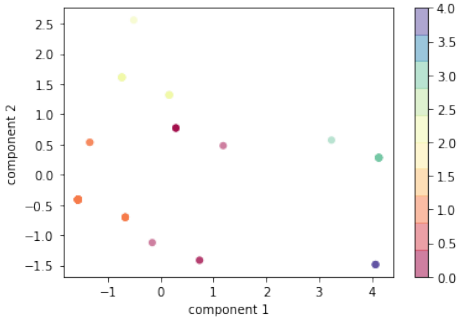

图 4-6 label-encoding PCA

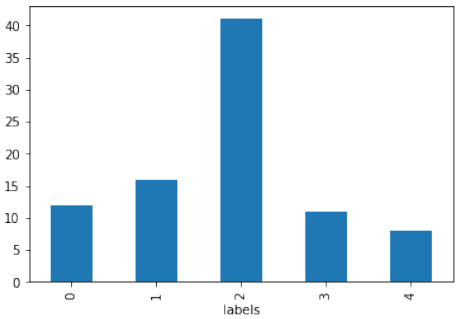

图 4-7 独热编码标签分布

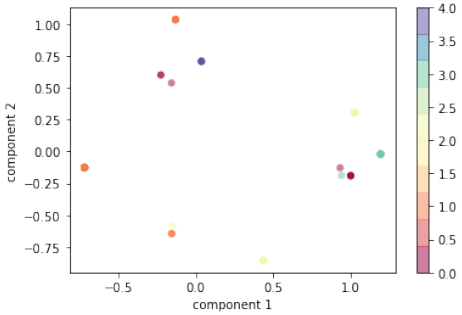

图 4-8 独热编码 PCA

图 4-9 不同编码的 TSNE 与 PCA 可视化结果

接着我对 KMeans 进行了优化，主要的优化为以不同的 k 值进行实验并评价聚类效果，对于聚类效果的评价，我选用的是轮廓系数 (silhouette score) 和 Calinski-Harabasz 系数，下面将分别对轮廓系数与 Calinski-Harabasz 系数展开介绍。

1. 轮廓系数取值在 [-1,1] 之间，接近 0 表示重叠的群集，负值通常表示样本分配给错误的聚类，轮廓系数的值大表示同类样本相距近，不同样本相距远，聚类效果较好，反之则代表同类样本与不同类相本之间距离不明显，聚类效果较差；

2. C-H 系数的主要作用是计算同一类别内部的协方差大小，并以此衡量类内的相似性，类内数据的协方差越小越好，类间的协方差越大越好，对于 C-H 系数而言 C-H 系数值越高说明聚类的区分度越高。

那么我通过 S-C 值与 C-H 值这两个指标可视化输出不同 k 值的轮廓系数与 C-H 系数变化曲线来确定聚类簇数 k 的最优值。结果如4-14所示：

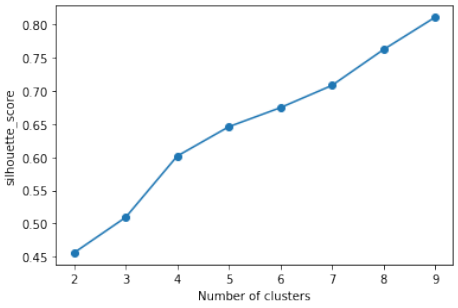

图 4-10 label-encoding 轮廓系数曲线

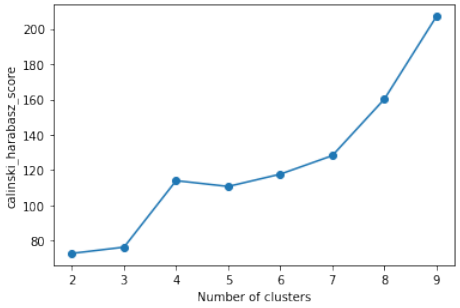

图 4-11 label-encoding C-H 曲线

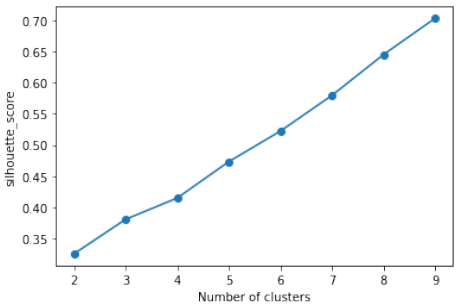

图 4-12 one-hot 轮廓系数曲线

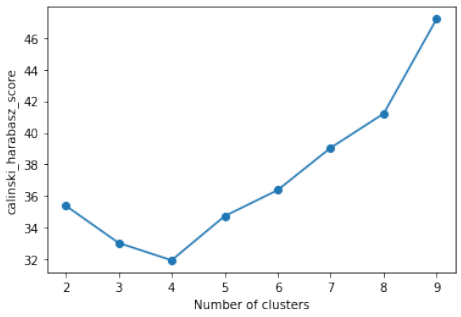

图 4-13 one-hot C-H 曲线

图 4-14 one-hot label-encoding 不同 k 值轮廓系数、C-H 曲线变化

可以看到 k=7 左右的聚类效果较好，那么在这样的情况下我们考虑使用其它方法进行聚类并评价聚类效果选择最优算法进行聚类，选择的其他算法包括谱聚类以及 HDBSCAN。

4. 2. 3 HDBSCAN 聚类

为了实现自动化的聚类个数 n 选择，我使用了 HDBSCAN 进行实验，在 HDBSCAN 的可视化工作上我生成了 HDBSCAN 的集群层次结构、生成的提取簇以及压缩聚类树，其中集群层次结构是通过并查集原理根据边之间的距离对不同的簇进行簇聚，从而使每一个簇形成一个单链接，此算法的目的在于寻找单链接的最终点；压缩聚类树的目的是根据最小簇大小对树进行分裂，从而使得算法能够标记在簇外的点，也就是数据标签中为-1 的点；提取簇的目的则是将每个簇最终形成一个特定的平面聚类，从而使不同簇映射到不同平面上增强聚类效果。我们同样在 one-hot encoding 以及 label encoding 两种不同标签上进行验证，聚类的效果如4-19：

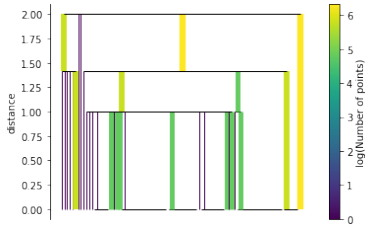

图 4-15 label-encoding 集群层次结构

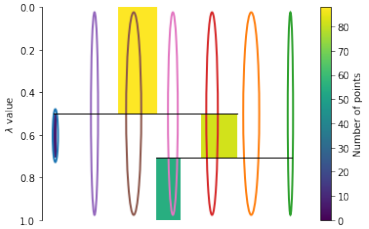

图 4-16 label-encoding 提取簇

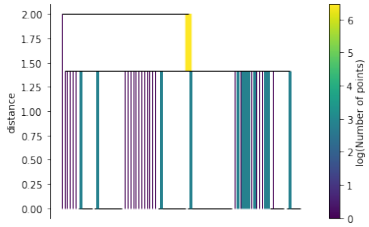

图 4-17 one-hot 集群层次结构

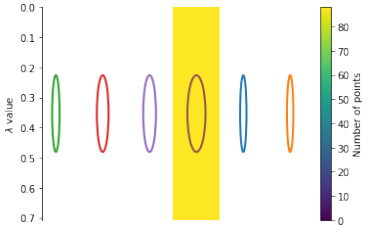

图 4-18 one-hot 提取簇

图 4-19 one-hot label-encoding 集群层次结构、提取簇变化

可以看到虽然 HDBSCAN 的效果不甚理想，主要原因是由于 HDBSCAN 本身是对平面上的数据进行分类，对于大量数据重合的情况分类效果尤其是提取簇和压缩树的效果不明显，但是在 Label encoding 中的 HDBSCAN 也将聚类簇数设置为 6，如果再加上 label 为-1 的列（对于 HDBSCAN 不属于任何聚类簇的标签）那么簇数为 7，而如果使用 one-hot encoding 的结果那么应该分为 7+1 共 8 类。两种不同 encoding 方式下聚类簇数的自动化选择与优化后的 KMeans 效果相似，再一次印证了 k=7 是较优结果。除此以外可以看到 HDBSCAN 的提取簇最终效果 one-hot encoding 和 label encoding 的差异非常大，这主要是由于 one-hot 后数据更加稀疏，但是在同一个 category 的特征距离进行计算时相较于 label encoding 所得出的欧式距离较小，所以数据分布更为致密；而 label encoding 在距离计算时由于平方使同一个 categorical 的特征距离大幅扩大，所以显得更为分散。

同时我们也使用了 DBSCAN 方法，DBSCAN 方法取 eps=0.01, min_samples=3，最终也是 7-8 类；但是效果不明显而且由于需要手动对 EPS 进行选择，所以在这里不展示结果。

4.2.4 谱聚类

除此以外我们也使用了谱聚类对聚类结果进行测试，谱聚类同样使用了轮廓系数以及 Calinski-Harabasz 系数对不同的聚类簇数进行测试，同时也对 gamma 值进行了调参，gamma 是谱聚类算法核函数参数，可视化输出效果如4-24下：

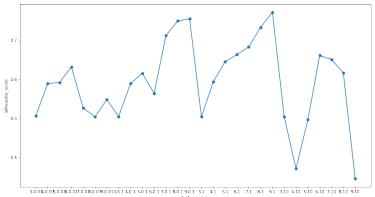

图 4-20 label-encoding k-gamma 轮廓系数曲线

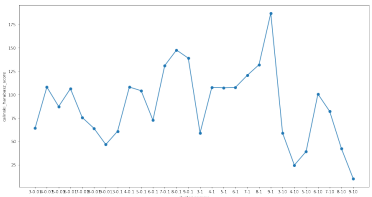

图 4-21 label-encoding k-gamma C-H 曲线

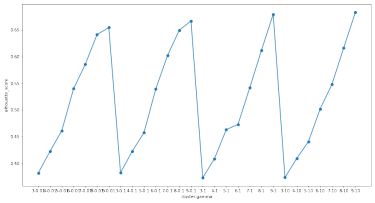

图 4-22 one-hot k-gamma 轮廓系数曲线

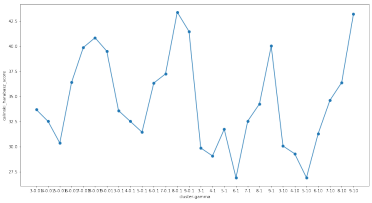

图 4-23 one-hot k-gamma C-H 曲线

图 4-24 one-hot label-encoding k-gamma 轮廓系数与 C-H 系数曲线

同样可以看到在 7-1 的组合左右谱聚类的轮廓系数达到最优，但是值得注意的是 C-H 系数呈现下降的态势，但是总体而言最终还是选择 k=7，gamma=1 进行聚类较为合适。

4.3 特征构建总结

其他的方法包括单独构建特征，不通过聚类的方法对特征进行 one-hot encoding 或 label encoding 直接作为数据特征进行训练或者将聚类结果作为特征而删除所有单独特征节省特征矩阵的长度，但是以上方法由于时间原因未能尝试，尤其是将聚类结果作为全部特征的特征构建也不失为一个构建模型好的选择，这样的方法更适合与对推荐模型进行融合。

除此以外可以考虑的方法还有对连续特征离散化，比如将成绩将成绩分段统计作为特征进行训练。这样可以避免连续取值需要归一化的问题，并且可以更清晰的对分数分布等特征在模型中的作用进行统计。

同时我们可以看到使用 one-hot encoding 和 label encoding 两种不同编码方式得到的聚类结果有很大差别，而这种差别在下文的模型构建过程中也有很明显的体现。使用不同方法聚类的 label 以及对应 label 数量如图4-29所示。

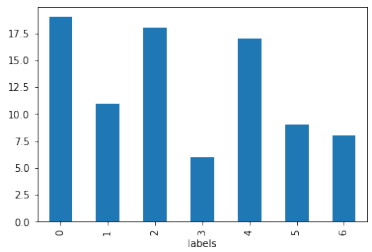

图 4-25 谱聚类 label-encoding 标签分布

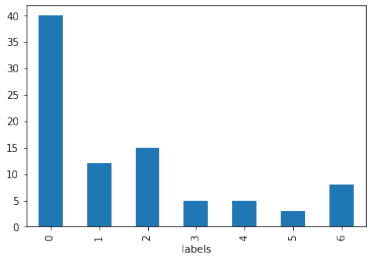

图 4-26 KMeans label-encoding 分布

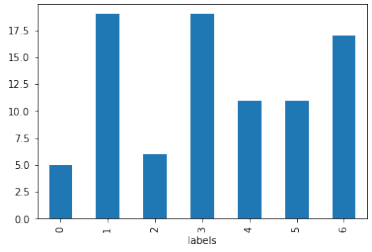

图 4-27 谱聚类 one-hot 标签分布

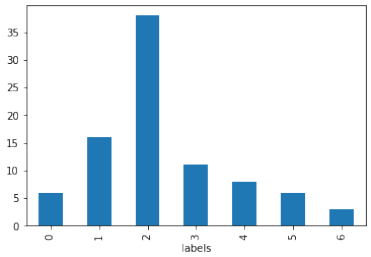

图 4-28 KMeans one-hot 标签分布

图 4-29 one-hot label-encoding 谱聚类、KMeans 标签分布

4. 4 csv 生成

在生成 csv 的过程中我们也充分考虑了论文等级论文期刊存在的偏序关系，以论文的排序举例，首先是按照用户的 user_id 进行排序，接下来按照论文层次进行排序，然后按照论文的作者等级进行排序，最后按照论文的期刊等级排序，在这样的排序条件下，虽然可能出现一个人有多个论文的情况，但是可以保证所有的数据都是以特定规则排列而不是无规则排列，这样在模型训练过程中会更为准确。除此以外对于论文等特征我们也统计了论文数量作为特征之一，而且在实验中证明论文数量也是影响奖学金评定的重要因素。

在特征设计的过程中为了将 one-hot 后离散的稀疏特征进行降维可以选择多种方法，其中比较典型的方法为借用 word2Vec 的思想加入 embedding 层进行嵌入训练将特征转化为低维稠密且元素不为 0 的特征，或者通过决策树等方式对特征进行降

维处理。我们在 DeepFM 中验证了前一种方法，并且使用了 XGBoost 验证了第二种方法。

CSV 构建过程中只含有两种标签，分别是硕士一等学业奖学金以及硕士其他等级奖学金，选择这两类奖学金是因为这两类在所有标签中数量最多。而且在 CSV 的标签选择中没有选择国家奖学金的理由是部分国家奖学金的数据和硕士学业奖学金重复，因为数据缺少时间特征所以无法区分哪些是申请国家奖学金所用的成果哪些是硕士学业奖学金的申请成果，在未来可以将其他更具体的奖学金加入其中实现 Top-N 推荐的功能。

	user_id	award_id	paper_count	paper_authOrder_0	paper_conferenceChoice_0	paper_paperState_0	paper_labels_0	paper_authOrder_1	paper_conference
0	2993.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	2995.0	3.0	1.0	1.0	1.0	4.0	5.0	0.0	
2	2997.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	3001.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	3005.0	2.0	2.0	1.0	1.0	1.0	6.0	1.0	
...	...	...	...	...	...	...	...	...	
121	4317.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	
122	4318.0	2.0	2.0	1.0	1.0	2.0	1.0	1.0	
123	4323.0	3.0	1.0	1.0	3.0	4.0	2.0	0.0	
124	4324.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	
125	4327.0	2.0	1.0	3.0	1.0	1.0	3.0	0.0	

图 4-30 生成 csv

最终生成的 csv 如4-30图所示。在上述 CSV 中可以看到在 CSV 生成以及读取后很多数据列都变成了 Double 浮点类，在 CSV 保存过程以及对应读取过程中应当注意将这些列改为 Int 型变量，优点是减小误差，其次更改数据类型能提高训练速度并节省内存。

第 5 章 模型训练与结果预测

在实验过程中，我们首先对数据集进行了划分，使用常见的经验公式对数据集进行了 8-2 划分，划分后的训练集共 100 条数据，测试集共 26 条数据。数据集划分使用相同的随机种子，以防止不同的模型使用的训练集或者测试集不同。在实验过程中，我们使用了多种模型分别进行实验并对结果进行比较。

5. 1 XGBoost

XGBoost 是一种常用的提升树模型算法，它在推荐系统和分类问题中有着优异的效果，XGBoost 基于 GBDT（梯度提升决策树）实现，它和 GBDT 相同都是 Boosting 算法一种，Boosting 算法的思想是将许多弱分类器如 CART 回归树模型等集成在一起，形成一个强分类器。

在我们的系统中，首先使用了 XGBoost 单独进行分类并查看分类的准确率、auc 等评价指标，并且将此作为 baseline，同时我们使用 Hyperopt 对 XGBoost 进行自动化调参。

接下来就是对 XGBoost 进行建模，XGBoost 使用 python 的 xgboost 包实现调用，在调用 XGBoost 的过程中，我们使用了 K-fold 的方法进行了 5 折交叉，通过调用 XGBoost 自带的 cv 函数实现比 sklearn 中 kfold split 更快的交叉运算。在将训练集和测试集输入 XGBoost 之前，首先通过 xgb 自带的 DMatrix 格式转化为 XGBoost 自有格式并且进行了填空处理，这样的优点是运算速度更快。

5. 1. 1 Hyperopt 自动化调参

然后我们通过引入 hyperopt 定义目标函数，hyperopt 是一个基于 Python 实现的 Sklearn 超参数优化类库，一般而言使用 hyperopt 有四个主要部分，分别是定义最小化/最大化的目标函数作为优化目标；其次是定义搜索空间，也就是超参数的；接下来定义存储搜索过程中所有点组合以及效果的方法；最后要确定在搜索过程中要使用的搜索算法。

我们在实验过程中对 XGBoost 选择的目标函数有两种，分别是验证集的 mlogloss 以及 auc 值，对于 mlogloss 而言，我们需要对其进行最小化，而 auc 则是进行最大化搜索；定义的参数搜索空间如5-1所示：

In [17]:
```
params_space = {
    'max_depth' : hp.choice('max_depth', range(5, 30, 1)),
    'learning_rate' : hp.quniform('learning_rate', 0.01, 0.5, 0.01),
    'n_estimators' : hp.choice('n_estimators', range(20, 205, 5)),
    'gamma' : hp.quniform('gamma', 0, 1.0, 0.01),
    'min_child_weight' : hp.quniform('min_child_weight', 1, 10, 1),
    'subsample' : hp.quniform('subsample', 0.1, 1, 0.01),
    'colsample_bytree' : hp.quniform('colsample_bytree', 0.1, 1.0, 0.01),
    'booster': 'gbtree',
    'tree_method': 'exact',
    'silent': 1,
}
```

图 5-1 hyperopt 参数空间

其中 max_depth 是决策树的最大深度，在这里搜索空间从 5 到 30 以 1 递增，learning_rate 是学习率，学习率从 0.01 到 0.5 以 0.01 递增，n_estimators 是决策树的数量，gamma 是控制后剪枝的参数，min_child_weight 是控制孩子节点当中最小的节点的样本权重和，如果叶子节点的样本权重和小于 min_child_weight 则对叶子节点拆分结束，在现行的回归模型中，这个参数的主要作用是控制过拟合，它代表了建立模型需要的最小样本数；subsample 同样是用于控制模型过拟合的一个参数，它的含义是用于训练模型的子样本在总样本中的比例；colsample_bytree 是在建立决策树时对特征随机采样的比例；booster 使用的是 gbtree，也就是树模型；tree_method 表示计算时使用的树方法，这里选择的是准确的计算值，同样可以使用近似计算来加快运算速度。

在定义搜索空间后定义目标函数，这里的目标函数代码如5-2所示：

In [16]:
```
import hyperopt
from hyperopt import fmin, tpe, hp, partial, STATUS_OK, Trials

def hyperopt_objective(params):

    model = xgb.XGBClassifier(n_estimators = params['n_estimators'],
                              max_depth = int(params['max_depth']),
                              learning_rate = params['learning_rate'],
                              gamma = params['gamma'],
                              min_child_weight = params['min_child_weight'],
                              subsample = params['subsample'],
                              colsample_bytree = params['colsample_bytree'],
                              num_class=2,
                              objective="multi:softmax",
                              # eval_metric="auc",
                              # auc doesn't work with multi-classification https://github.com/dmlc/xgboost/issues/1208
                              eval_metric="mlogloss",
                              seed=31415926,
                              nthread=-1,
                              silent=1
                              )

    res = xgb.cv(model.get_params(), dtrain, num_boost_round=30, nfold=5,
            callbacks=[xgb.callback.print_evaluation(show_stdv=False),
                       xgb.callback.early_stop(5)])

    # return np.min(res['test-mlogloss-mean']) # as hyperopt minimises
    return np.min(res['test-mlogloss-mean'])
```

图 5-2 hyperopt 目标函数

可以看到这里使用的是 test-mlogloss-mean 作为优化函数进行优化，每次选择 test-mlogloss-mean 的最小值返回给 hyperopt 进行效果验证，它的返回值来自于 xgboost 的 k-fold 方法，同时为了防止过拟合的现象使用了 early_stopping 方法在验证集验证，使用的验证函数同样是 mlogloss。

同时在这里需要特别注意的一点在于 XGB 是无法使用 auc 作为 eval_metric，在 XGBoost 的 issue 中可以看到 auc 目前不支持在 XGB 上使用。

除此以外对于 XGBoost 模型分别在两种不同编码格式的数据集上进行了测试，首先我们在 one-hot encoding 的数据集上进行了测试。

5. 1. 2 数据编码方式效果对比

根据 XGBoost 模型的文档，它推荐我将所有的类别特征转化为独热编码，XGBoost 对于特征的值仅接受数字类型的特征，因此必须进行独热编码。但是如果有类似的数字特征变量并且对类别使用数字占位符（例如，男，女，未知分别为 1,2,3），则对于 XGBoost 的学习而言这些特征是不能正确表示的，因为这些特征是存在偏序关系的，而原始类别没有与之关联的任何固有顺序，但是 XGBoost 模型可能会错误的学习数值带有的偏序关系而导致误差。

所以我们考虑将论文的作者顺序等通过 one-hot encoding 转化为独热编码后输入 XGBoost，虽然也有人指出 one-hot encoding 可以将距离计算变得更为合理，但对于离散特征而言，不需要使用 one-hot encoding 即可较好的进行距离计算，对于某些基于树的算法而言，变量的处理并不是基于向量空间而只是作为类别符号，其不存在偏序关系故而不需要进行 one-hot encoding。而独热编码反而增加了决策树的深度。

hyperopt 的默认搜索算法是 tpe，这是一种类似于遗传算法的搜索算法，它是一种启发式的最优化方法。

首先将 one-hot encoding 后的特征输入 XGBoost 中通过 hyperopt 自动化调参，数据集共有 118 维特征，学习目标为多分类 (multi:softmax)，验证集使用目标函数为 mlogloss，测试后的最优参数下测试集的 loss 为 0.41，而在此情况下训练集的 loss 为 0.21，可以看到 XGBoost 对于训练集出现了一定程度的过拟合情况。同时在这种情况下 gamma 值为 0.84，学习率为 0.45，树的最大深度为 12，subsample 取值为 1，这里是值得注意的，因为 subsample 为控制过拟合的参数，它使用全部训练集输入对 XGBoost 的树模型进行训练在一定程度上也导致了测试集的 loss 较差，特征随机采样比例为 0.79，叶子节点的最小权重和为 2，决策树个数为 14。

使用最优参数对 XGBoost 模型进行训练，得到的 CrossValMean 为 0.78，准确率为 0.61，可以看到这个结果不甚理想，同时特征重要性结果如图5-3所示：

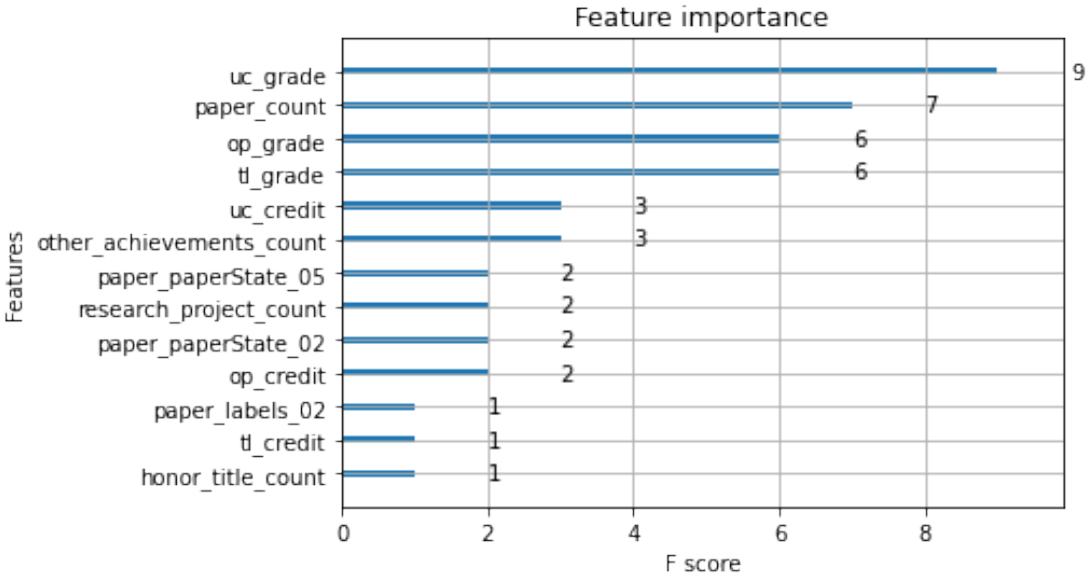

图 5-3 XGBoost one-hot 特征重要性

可以看到其中最为重要的特征是未分类课的成绩，其次是发表论文的数量，然后是选修课成绩、总加权平均成绩以及未分类课学分，然后是其他类型奖励的数量、以及论文的层次等。综上可以看到在整个过程中对模型影响较大的因素为成绩因素和论文质量。

图 5-4 XGBoost one-hot 生成树 (其一)

图5-4是其中一棵生成树的图片，可以看到树的深度为 6，首先分裂的节点是论

文数量。

接下来我们又对使用 Label Encoding 的编码数据集进行了测试，测试的结果大大出乎了我们的意料，在 hyperopt 优化参数不变的情况下，Label Encoding 的效果远好于 One-hot 编码后的效果，虽然 XGBoost 的文档推荐使用 One-hot 编码，而且原理上 One-hot 不影响决策树的效果，但是实际上 XGBoost 在小量数据上对大量特征的分裂决策效果依然不是很好。

Label Encoding 的数据集中特征向量的维数为 61，经过测试 XGBoost 在 Label Encoding 数据集上的最优参数为特征随机采样比例 0.45，gamma 为 0.99，学习率为 0.17，最大深度为 23，最小叶子节点权重和为 2.0，决策树个数为 26，决策树模型训练样本比例 subsample=0.7，在最优参数下进行模型训练得到的 FinalCrossMean 为 0.79，虽然和使用 One-hot encoding 的结果相差不大，但是实际上模型在测试集的准确率提升到了 96%，远高于独热编码后的结果，特征重要性排序如图5-5所示：

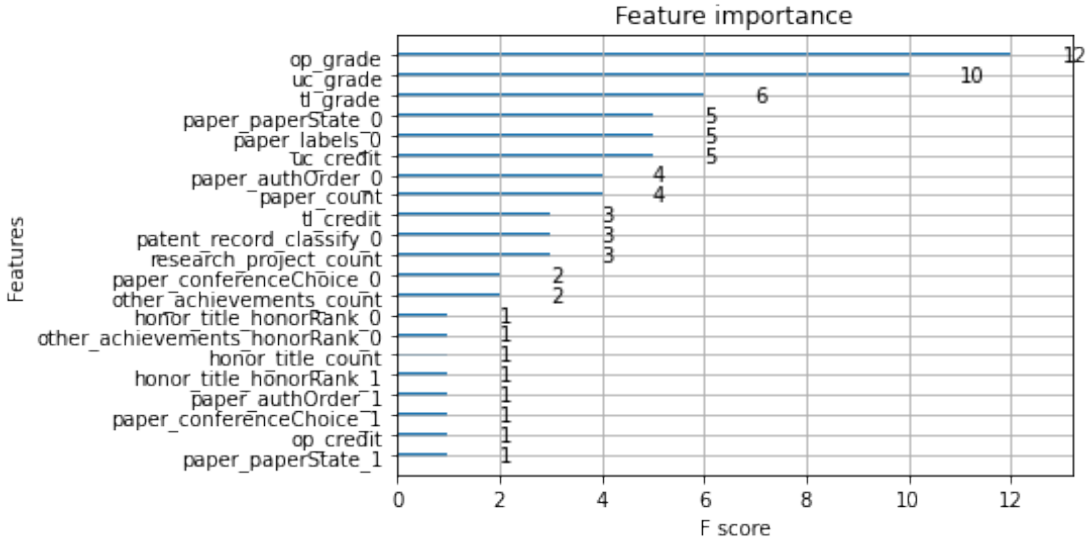

图 5-5 XGBoost label-encoding 特征重要性

可以看到参与训练的特征数量明显变多，虽然成绩特征依然重要，但是最重要的特征从未分类加权平均成绩变为选修课加权平均成绩，总成绩的重要性基本不变，接下来第一篇论文的论文层次和论文聚类的标签也对结果有重要作用，除此以外第一篇论文的作者排序以及发表论文的总数量等起次等作用。未分类课的学分以及总学分在特征重要性中也有一定比重。但是对于使用独热编码的特征，专利、研究项目

等很多特征并没有考虑在内。而在 Label-Encoding 方法下，专利和研究项目的数量、其他成果的数量、荣誉等级和荣誉数量等都在模型决策树构建中发挥作用，除此以外模型对第二篇论文的情况也纳入了考虑范围，包括第二篇论文的作者顺序、期刊等级以及论文层次等。

所以综上可以看出 One-hot encoding 的模型相比于 Label encoding 的模型效果相差甚远，可能原因是 One-hot 的特征矩阵较为稀疏，模型的迭代次数不足，而且模型在自动化调参过程中 subsample 比例过高导致模型出现了比较严重的过拟合问题，以上都是未来可以进行实验和改进的方向。Label encoding 的 XGBoost 的一棵生成树如图5-6所示。

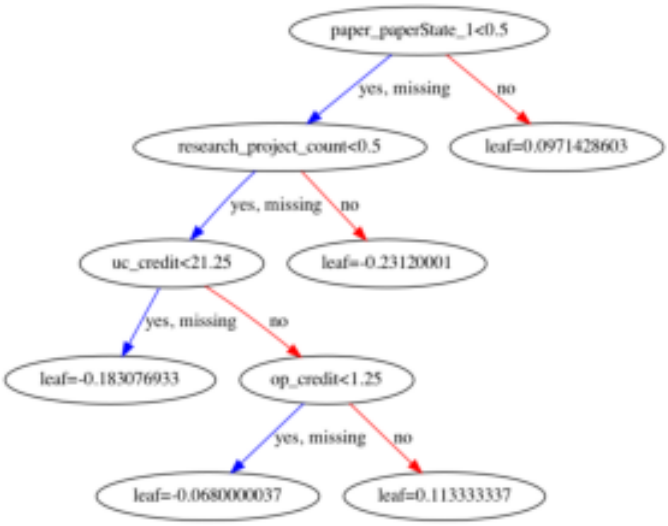

图 5-6 XGBoost label-encoding 生成树

在构建 XGBoost 单独进行预测后，我们也考虑将 XGBoost 的决策树输出作为新的特征与其他简单模型融合，可以考虑使用 XGBoost+ 贝叶斯或者 XGBoost+LR 逻辑回归等。

5. 2 XGBoost 与逻辑回归融合

我们同样进行了对比实验，以上述表现效果较好的 Label-Encoding XGBoost 模型作为新特征的全部或者一部分分别进行实验。

首先是对 XGBoost 决策树的特征进行构建，特征构建的原理是由于 XGBoost 的决策树存在多个输出，对于一条数据而言它的权值会由不同的叶子节点输出，所以

将所有的叶子节点视为 0-1 编码的 one-hot 向量，即可构建一个长度 = 叶子节点总数的特征向量。

对叶子节点输出进行获取的方法是 XGB 自带的 apply 方法，apply 的结果是决策树对应的叶子节点值，若没有叶子节点 (max_length = 1)，那么此值为 0，否则 >=1 的整数，所以先去除没有叶子节点的决策树，通过自定义的 get_cols_del 函数获取 apply 后形成的 DataFrame 中需要删除的列标号，然后用 del_cols 函数删除 DataFrame 中对应的无信息的列 (全 0)，再通过 one-hot encoding 的形式自定义一个 One-hot Label Encoder 进行 one-hot 编码后得到对应的稀疏特征矩阵，这样可以在一定程度上压缩特征空间的大小，并且能够提取高维的特征交互信息，然后将此结果输入一个简单的线性分类模型中，同时还可以省去归一化的步骤。

在这里需要对网络的叶子节点输出进行 one-hot 编码，一方面是因为这些节点输出本身不存在偏序关系，可以用 one-hot 去除这种偏序关系，第二方面 one-hot 的离散化处理可以增强逻辑回归模型的性能。由于逻辑回归在处理离散化特征时离散特征自身带有单独的权重能够引入非线性特征，并且离散化的特征有利于对异常点的处理，这样使网络的鲁棒性更强，而离散性特征本身能够进行特征交叉，增强了网络的表达能力，从而增强网络拟合。

将构建好的特征输入到一个简单的逻辑回归分类器当中，此时输出经过 one-hot encoding 后的特征共 98 维，略微少于完全使用 one-hot 的 118 维，在使用默认参数进行训练后对测试集进行了准确率测试，测试结果与 XGBoost 相同但是 auc 值高于单独使用 XGBoost 结果 0.25，此时 auc 值为 0.97，虽然提升效果不明显但是在全数据集上进行准确率测试发现准确率从 0.89 提升到 0.97，auc 从 0.83 提升到 0.963，所以逻辑回归和 XGBoost 决策树节点输出结果结合比较好的解决了模型的拟合问题，并且进一步提升了模型的准确率。

除此以外我们还使用 XGBoost 决策树节点输出结果与原有特征进行融合，以此构建一个既包含高维特征交叉也包含低维特征的特征向量。在这种情况下，逻辑回归可以学习低维的特征交叉，同时也可以将低维的特征与 XGB 的输出高维特征交叉进一步增强模型的表达能力。同样将 80% 的数据用于训练 20% 数据用于测试，在这种情况下模型出现了一定的过拟合情况，对测试集的准确率下降到 92%，auc 下降到 93%，但全数据集的准确率提升到 98%，auc 提升到 98.7%，虽然提升不明显，但是无论在 accuracy 或者 auc 上均有提升，所以可以看出融合了原有低维特征和 XGBoost

输出高维特征的特征向量还是使结果呈现了一定的上升趋势。

两种融合模型的结构如图5-9所示:

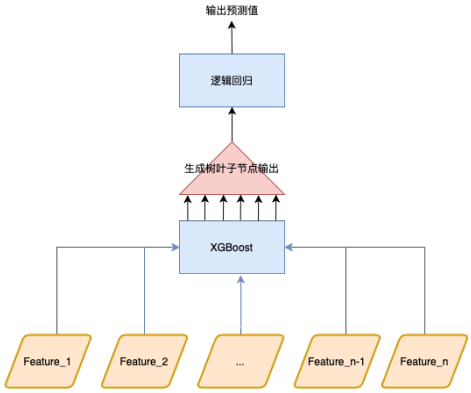

图 5-7 原始模型结构

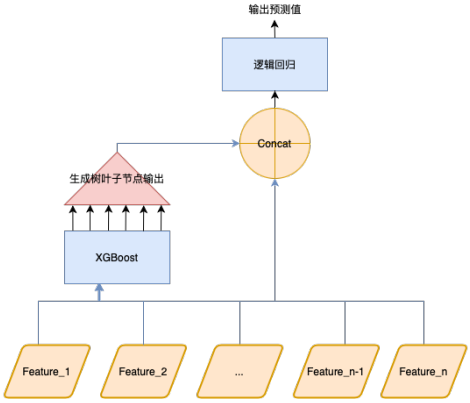

图 5-8 扩展低维特征模型结构

图 5-9 两种模型结构对比

主要区别在于逻辑回归部分输入不同，左侧视为串行模型，右侧将特征 concat 起来再送入模型训练则是并行模型。

5. 3 XGBoost 与贝叶斯分类器融合

那么顺着这个思路我们又进行了一系列的实验，包括将逻辑回归模型替换为 Bayes 分类器或 SVM 等并分别对它们的表现进行测试，结果如下:

表 5-1 贝叶斯分类器效果比较

	Test-Accuracy	Test-AUC	Train-Accuracy	Train-AUC
Pure BernoulliNB	0.884	0.888	0.72	0.704
BernoulliNB + XGBoost	0.923	0.918	0.83	0.792
GaussianNB + XGBoost	0.885	0.906	0.82	0.862
MultinomialNB + XGBoost	0.923	0.918	0.81	0.818

可以看到不同分布函数的贝叶斯分类器的效果不同，其中效果最好的是高斯分布的贝叶斯，这种情况出现可能和成绩、竞赛等分布符合高斯分布有关。可以看到多项式贝叶斯模型在测试集的效果最好，但是从训练集的准确度和 auc 值看，可能存在欠拟合的问题，并且在整个数据集上进行测试后可以发现它整体的效果不如高斯分布贝叶斯。伯努利分布的贝叶斯与多项式贝叶斯一样在测试集上效果好但是在训练集上效果一般，同样可以考虑测试集和训练集划分的问题。单独使用贝叶斯对不包含 XGB 输出的效果是最差的，说明 Bayes 没有对高维和低维特征的融合进行较好的处理。

同时我们还使用了 GBDT、SVM 等作为对特征融合的向量的输入模型。其中 GBDT+XGBoost 的效果最好，这是由于 GBDT 与 XGBoost 一样都是树模型，但是没有使用 GBDT 的原因是 GBDT 的模型较为复杂，不利于降低模型复杂度，而且训练比逻辑回归要困难得多。

5. 4 神经网络模型

关于神经网络模型，我们使用了 DeepFM，通过 FM 进行自动特征交叉，并验证了算法的效果。

首先对 DeepFM 中的 DenseFeature 与 SparseFeature 进行处理，DeepFM 中的 DenseFeature 指数值类特征或类似连续或带有特定偏序关系的特征，SparseFeature 一般为 CategoricalFeature 或类似离散数值特征。通过函数 SparseFeat 实现稀疏特征矩阵构造，其中 DeepFM 使用默认参数进行实验（dnn_hidden_units=(128, 128), l2_reg_linear=1e-05, l2_reg_embedding=1e-05, l2_reg_dnn=0, seed=1024, dnn_dropout=0, dnn_activation='relu', dnn_use_bn=False, task='binary'), 其使用的 embedding 层的 size 为 4，以以上参数训练了 40 个 epoch。

以 0.5 为界对 DeepFM 的输出进行处理，则在使用相同数据集划分 (测试集-训练集) 种子情况下 DeepFM 在测试集上准确率为 0.92，略低于使用 label-encoding 的 XGboost，在整个训练集上的准确率是 0.82，整体为 0.84 左右，与单独使用 XGBoost 相比存在一定差距。但是相比使用 one-hot encoding 的 XGB 模型进步明显。

对 DeepFM 的网络结构进行了可视化处理，由于网络 embedding 层过长导致模型输出结果过大，所以在这里只放 FM 的一部分以及 DNN 部分和最后的 Add 隐含层神经元与输出神经元，如图5-10所示。

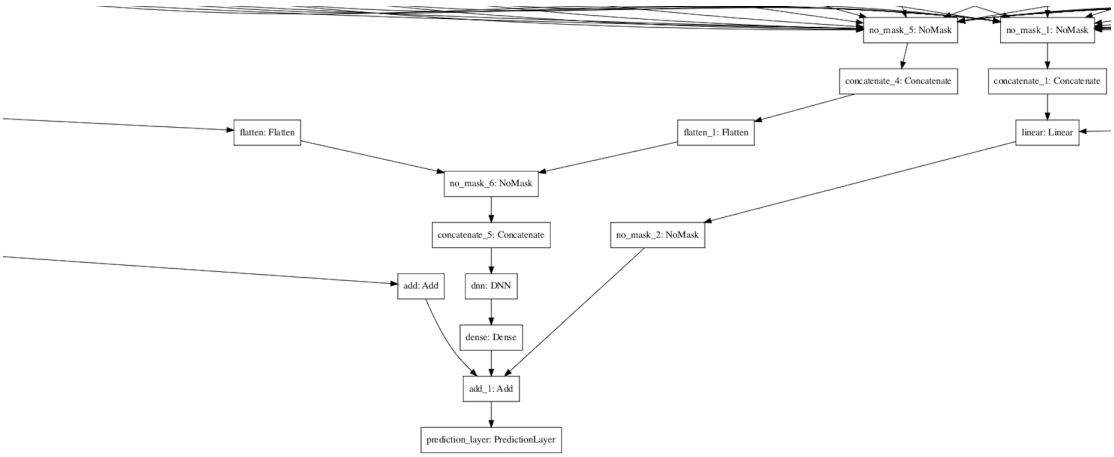

图 5-10 DeepFM 模型

可以看到 DeepFM 的 DNN 输入是 embedding 层的输出后进行 concat 的结果，FM 中 Add 层清晰可见，最终 FM 的结果和 DNN 的结果通过 Add 层融合叠加。

5. 5 本章小结

本章是涵盖了训练的整个过程，其中也包括高维特征构建部分，在这里通过使用 XGB 替代 FM 实现了高维的特征交叉，并且将特征交叉的结果与原有低维特征连接送入逻辑回归模型中训练，得到了非常好的效果，在整体数据集的准确率达到了 99.2%，也就是只有一条数据没有被正确分类。虽然这个数据量比较小，但是测试集的准确率是 100%，远超其他模型。而且值得注意的是 DeepFM 也在测试集上表现出了较强的分类效果，但是在训练集上的效果却比另外两个模型差。DeepFM 可以考虑继续增加 epoch，其最终结果与高斯分布贝叶斯类似。

表 5-2 模型效果比较

	Test-Accuracy	Test-AUC	All-Accuracy	All-AUC
XGBoost	0.9615	0.95	0.8889	0.8316
XGBoost+LR	1.0	1.0	0.9921	0.9875
DeepFM	0.9231	0.9	0.8413	0.7701

结　论

通过这个毕业设计，我基本上完整的进行了数据预处理、数据仓储、特征构建、特征工程、算法选择与比较以及预测推荐的完整流程。虽然整个系统在训练当中的数据量偏少，特征维数较少而且很多功能都还没有实现，但是整体而言这让我体会到了一个完整的数据挖掘的流程，使我获益匪浅。

1 特征构建与特征工程

值得注意的是在挖掘过程中，XGBoost 使用 one-hot 编码以及 label encoding 两种编码方式的结果有明显差异，再一次提示我在进行特征构建时要注意考虑特征是否存在偏序关系。如果只是简单的类别特征，如国家等，可以考虑使用 one-hot encoding 的方法进行编码，否则最好考虑两种方法融合或者进行比较。

除此以外特征构建过程中我们没有使用特征重要性或者 PCA 主成分分析等方法对特征进行降维，而是使用了聚类的方法以期可以对特征降维。从结果来看聚类的标签在模型权重的重要性比重不大，可能是由于数据量较少导致模型训练过程中没有使用到全部的特征，这一点在 XGB 模型的特征重要性当中有比较明显的体现，为了日后系统扩展的需求，暂时先不对特征进行筛选或主成分分析来降维。

在特征构建的过程中我们也没有使用手工特征交叉，主要原因是对于数据的理解不够深入，不知道怎么进行特征交叉的效果比较好。而且整体来看最后的特征哪怕是 label encoding 的结果也有超过 60 维，而且很多特征在训练中起到的作用很小，甚至对模型是负优化，可以考虑在未来剔除这些特征，如专利的分类 (发明专利/其他) 等。在特征构建的过程中只使用了两类具有代表性或数量较多的标签作为 label 进行训练，未来也可以加入国奖等其他奖学金将 XGBoost 模型修改为多分类，多分类下可以将一类作为正类其他类作为负类计算某条数据的概率并以阈值为过滤条件进行推荐，这样可以实现 Top-N 推荐的功能。

同时在多分类下也可以对网络输出层进行修改，使神经元个数与分类标签个数相等，从而构建一多分类器。也可以加入 Sigmoid 进行分类器构建。

2 模型训练与融合

在这一部分中我们分别进行了单独的 XGBoost 分类器构建、朴素贝叶斯分类器构建以及 SVM 模型构建等多种模型，结果上 XGBoost 无论在训练精度或者 AUC 值上都占有明显优势。体现了 XGBoost 在处理这种少量有关联特征的数据时的优秀效果，

在 XGBoost 的特征构建过程中，可以看做我们对模型进行了融合，将 XGBoost 的结果与逻辑回归分类器进行融合。但是这里的缺点是最后形成的模型并不是一个端到端的模型，而是需要先对 XGB 进行训练，并没有达到联合训练的效果。所以这种方法整体的训练难度较高，但是值得肯定的是这种方法的收益率是很明显的，XGBoost 的 apply 方法的叶子节点输出可以看作是高维的特征交叉结果，而对低维特征的保留与逻辑回归方法的运用则可以看作是另一种 Wide & Deep 模型的实现思路，不同的是 Wide & Deep 还使用了 embedding 层进行了降维，而我们没有对特征进行 embedding 处理。下一步可以考虑将 XGBoost 的 apply 输出当作 Wide & Deep 中 Wide 部分的输入，代替手动特征工程的结果对网络进行训练。

但是无论如何 XGBoost 和神经网络相比它的优势是很明显的，具体体现在 XGBoost 的模型训练速度更快、精度更高。而且可以很方便的将 XGBoost 的结果与其他模型融合。也进一步验证了对于中、少量的结构化数据 XGBoost 的效果一般比神经网络更优。

可以考虑使用融合模型的方法训练，但是没有采用这种方法的原因是单纯的 XGBoost 效果很好，采用这种方法融合意义不大。

同时还应当注意 XGBoost 在训练过程中的特征重要性，可以看到论文数量以及成绩两因素对得奖结果的影响是非常明显的，从特征上可以看出学分以及对应的单项平均成绩在使用两种不同编码方法的 XGBoost 模型中所占权重都比较大，其次在使用 label-encoding 的 XGB 模型中它学习到了更多特征相关性，值得注意的是未分类课成绩、选修课成绩、加权平均成绩在两模型中所占比重都比较高，第一篇论文在模型中所占权重较高，而且使用 label encoding 的模型在使用论文特征、其他成果特征、荣誉获奖特征、科研项目特征之外还学习到了专利这个大类。但是两个模型都没有学习到竞赛这个大类的特征，可能是数据量较少或竞赛大类缺少竞赛获奖特征而只包含参加竞赛和竞赛等级，导致此类特征不适合用于奖学金分类。

综上可以看到 XGBoost+LR 的强大性能以及特征构建过程中可改进的方向。

参考文献

[1] 郭洁畅, 杜鹏. 大学生兼职电商平台工作推荐系统的实现[J]. 数码世界, 2017, 8.
[2] 刘睿，荣文戈，唐翠，欧阳元新，熊璋. 一种面向冷启动学生用户的工作推荐系统及推荐方法. CN106097204A[P]. 中国. 2016-11-09.
[3] Bustos López M, Alor-Hernández G, Sánchez-Cervantes J L, et al. EduRecomSys: An Educational Resource Recommender System Based on Collaborative Filtering and Emotion Detection[J]. Interacting with Computers, 2020, 32(4): 407-432.
[4] Su X, Khoshgoftaar T M. A survey of collaborative filtering techniques[J]. Advances in artificial intelligence, 2009, 2009.
[5] 荣辉桂, 火生旭, 胡春华, 等. 基于用户相似度的协同过滤推荐算法[J]. 通信学报, 2014, 35(2): 16.
[6] 姜保庆, 王奇伟, 韩景景. 基于社交网络的一种个性化推荐算法[J]. 网络安全技术与应用, 2014, 3.
[7] Deshpande M, Karypis G. Item-based top-n recommendation algorithms[J]. ACM Transactions on Information Systems (TOIS), 2004, 22(1): 143-177.
[8] Guo G. Resolving data sparsity and cold start in recommender systems[C]//International Conference on User Modeling, Adaptation, and Personalization. 2012: 361-364.
[9] Sharma L, Gera A. A survey of recommendation system: Research challenges[J]. International Journal of Engineering Trends and Technology (IJETT), 2013, 4(5): 1989-1992.
[10] Tahir M A, Bouridane A, Kurugollu F. Simultaneous feature selection and feature weighting using Hybrid Tabu Search/K-nearest neighbor classifier[J]. Pattern Recognition Letters, 2007, 28(4): 438-446.
[11] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. ArXiv preprint arXiv:1412.6980, 2014.
[12] Russel D.Reed R J. Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks[M]. Boston, USA: MIT Press, 1999.
[13] Bishop C M. Neural Networks for Pattern Recognition[M]. Oxford, UK: Clarendon Press, 1995.
[14] Ian Goodfellow A C, Yoshua Bengio. Deep Learning 2016[M]. Boston, USA: MIT Press, 2016.
[15] RobertKidder. 分类任务交叉熵与 softmax 函数[EB/OL]. 博客园. (2021-04-04) [2021-04-04]. https://www.cnblogs.com/RobertKidder/p/14615362.html.
[16] 飞鱼 Talk. 损失函数丨交叉熵损失函数[EB/OL]. 知乎. (2018-04-15) [2018-04-15]. https://zhuanlan.zhihu.com/p/35709485.
[17] Van der Maaten L, Hinton G. Visualizing data using t-SNE.[J]. Journal of machine learning research, 2008, 9(11).
[18] McInnes L, Healy J, Astels S. Hdbscan: Hierarchical density based clustering[J]. Journal of Open Source Software, 2017, 2(11): 205.
[19] Von Luxburg U. A tutorial on spectral clustering[J]. Statistics and computing, 2007, 17(4): 395-416.
[20] Chen T, He T, Benesty M, et al. Xgboost: extreme gradient boosting[J]. R package version 0.4-2, 2015, 1(4).
[21] Zcsh. XGBoost 原理解析[EB/OL]. 博客园. https://www.cnblogs.com/zcsh/p/14752659.html.
[22] Rendle S. Factorization machines[C]//2010 IEEE International Conference on Data Mining. 2010: 995-1000.

[23] Pachocki J. Factorization Machines[EB/OL]. Carnegie Mellon University. 2015 [2016]. http://www.cs.cmu.edu/~wcohen/10-605/2015-guest-lecture/FM.pdf.
[24] del2z 大. 深入 FFM 原理与实践[EB/OL]. 美团技术团队. (2016-03-03) [2016-03-03]. http://www.cs.cmu.edu/~wcohen/10-605/2015-guest-lecture/FM.pdf.
[25] Juan Y, Zhuang Y, Chin W S, et al. Field-aware factorization machines for CTR prediction[C]// Proceedings of the 10th ACM conference on recommender systems. 2016: 43-50.
[26] Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]// Proceedings of the 1st workshop on deep learning for recommender systems. 2016: 7-10.
[27] Guo H, Tang R, Ye Y, et al. DeepFM: a factorization-machine based neural network for CTR prediction[J]. ArXiv preprint arXiv:1703.04247, 2017.
[28] DeepCTR. DeepFM[EB/OL]. DeepCTR. 2017 [2017]. https://deepctr-doc.readthedocs.io/en/latest/Features.html#deepfm.

附　录

附录 A　GitHub 仓库地址

Skybluewater: AwardSystem

附录 B　DeepCTR API 说明

DeepCTR Document

附录 C　XGboost 使用说明

XGBoost Document

附录 D　DeepCTR RunExample

DeepCTR RunExample

致　谢

值此论文完成之际，老师我改论文能只改致谢么 □?