DOI:10.13718/j.cnki.xsxb.2021.11.007

# 面向软件缺陷数据的协同过滤抽样推荐算法①

吴克奇[1], 崔梦天[1], Mariani Manuel Sebastian[2],
张翼成[3], 谢 琪[1], 周绪川[1]

1. 西南民族大学 计算机系统国家民委重点实验室, 成都 610041;
2. 苏黎世大学 商业管理系, 瑞士 苏黎士 8050;
3. 弗里堡大学 物理系, 瑞士 弗里堡 1700

**摘要**: 基于没有一种抽样方法能在所有缺陷数据集上表现良好且为软件缺陷数据选择适用抽样方法是必要的这一前提, 提出了一种面向软件缺陷数据的协同过滤抽样推荐算法. 首先, 在历史缺陷数据上对主流抽样方法进行排序, 以得到在特定分类算法和度量指标下主流抽样方法的性能排序; 然后, 计算新缺陷数据和历史缺陷数据之间的杰卡德相似系数, 以挖掘数据之间的相似性; 最后, 将抽样方法排名和数据相似性的信息结合起来构建一个推荐网络, 利用协同过滤算法为新的软件缺陷数据推荐适用的抽样方法. 通过 Python 对多个 NASA 缺陷数据集进行仿真实验, 实验结果表明面向软件缺陷数据的协同过滤抽样推荐算法是可行和有效的.

**关 键 词**: 软件缺陷数据; 抽样推荐算法; 协同过滤; 数据相似性

**中图分类号**: TP311 **文献标志码**: A **文章编号**: 1000-5471(2021)11-0046-10

# Sampling Recommendation Algorithm Based on Collaborative Filtering for Software Defect Data

WU Keqi[1], CUI Mengtian[1], Mariani Manuel Sebastian[2],
ZHANG Yicheng[3], XIE Qi[1], ZHOU Xuchuan[1]

1. *The Key Laboratory for Computer Systems of State Ethnic Affairs Commission, Southwest Minzu University, Chengdu 610041, China;*
2. *Department of Business Administration, University of Zurich, Zurich CH-8050, Switzerland;*
3. *Department of Physics, University of Fribourg, Fribourg CH-1700, Switzerland*

**Abstract**: Based on the fact that no single sampling method can be performed well on all defect data sets and it is necessary to select suitable sampling methods for software defect data, a sampling recommendation algorithm based on collaborative filtering for software defect data has been proposed. Firstly, the mainstream sampling methods are sorted on historical defect data to obtain the performance ranking of the

① 收稿日期: 2020-09-25
基金项目: 国家自然科学基金项目(12050410248); 四川省科技计划项目(2021YFH0120); 四川省科技创新苗子工程项目(2020024, 2021010); 成都市国际科技合作资助项目(2021-GH03-00001-HZ); 西南民族大学研究生创新型科研项目(CX2020SZ07).
作者简介: 吴克奇, 硕士研究生, 主要从事软件缺陷研究.
通信作者: 崔梦天, 教授, 博士.

mainstream sampling methods under specific classification algorithms and metrics. Secondly, the Jaccard similarity coefficient between the new defect data and the historical defect data is calculated to mine data similarity. And finally, the information of sampling method ranking and data similarity is combined to build a recommendation network, and the cooperative filtering algorithm is used to recommend the applicable sampling method for the new software defect data. The simulation experiment is carried out on multiple NASA defect data sets by using Python. The experimental results show that the sampling recommendation algorithm based on collaborative filtering for software defect data is feasible and effective.

**Key words**: software defect data; sampling recommendation algorithm; collaborative filtering; data similarity

在这个信息爆炸的时代，时刻都产生着大量的数据，在这些数据中，有一类为不平衡数据，其各个类别的样本数目相差巨大[1]. 在软件缺陷预测领域，其缺陷数据集通常都是不平衡的[2]. 但是直接用传统分类方法解决不平衡数据的分类问题时，其效果往往都不理想. 这是因为不平衡数据集中多数类的数量远远大于少数类的数量，导致数据集没有足够的少数类信息进行分类预测[3]. 传统分类方法追求整体的准确率最大化，从而导致模型更偏向于多数类，但不平衡数据中的少数类在现实生活中的意义往往更大.

为了解决软件缺陷数据集的不平衡问题，有大量的方法被提出. 这些方法主要是在算法层面和数据层面上解决数据不平衡问题，算法层面有代价敏感[4]、集成学习和特征选择[5]方法. 数据层面的方法主要是过采样、欠采样和混合采样. 过采样中使用最频繁的是随机过采样(random over sampling)，因为该算法的实现较简单并且效果还比较好. 文献[6]使用簇心或最接近簇心的样本代替原数据，基于此提出了两种聚类的欠采样方法. SMOTE(synthetic minority oversampling technique)算法存在一些不足，比如易引入噪声点、合成的样本有重复等问题，于是产生了很多改进算法[7-9]. 文献[10]提出了将 SMOTE 算法和数据清洗方法相结合，增加了多数类和少数类的可分性.

过采样和欠采样虽然可以平衡数据分布，但欠采样可能会删除对分类有价值的数据，过采样则会增加过拟合的风险而且可能引入不合理的样本数据[11]. 在软件缺陷预测领域，使用不同的缺陷数据集、分类技术和度量指标得到的最好抽样方法有时会出现矛盾. 这意味着没有一种抽样方法可以在所有软件缺陷数据上表现得很好[12]. 因此，由于有大量不同的抽样方法，软件研究人员和从业人员为新的软件缺陷数据选择适用的抽样方法将是非常困难但相当重要的，故本文提出了一种抽样推荐算法，为新数据集推荐适用的抽样方法.

## 1　相关准备

### 1.1　欠采样

在不平衡数据中，负样本为数量多的样本，正样本为数量少的样本，且正样本在现实生活中的意义大于负样本. 欠采样是指选取一些具有代表性的负样本，这样大大减少了负样本的数量，使得负样本和正样本的数量相当. 虽然提高了正样本分类准确率以及分类效率，但同时也丢失了负样本的数据特征，分类模型不能充分学习到负样本的样本特征，导致负样本的分类准确率降低. 下面将介绍几种主流的欠采样方法：

随机欠采样(random under sampler)的思想就是随机选取一些多数类样本并剔除掉. 这种方法的缺点是被剔除的样本可能包含着一些重要信息，致使学习出来的模型效果不好.

NearMiss 本质上是一种原型选择(prototype selection)方法，为了在一定程度上解决随机欠采样的信息丢失问题，用于训练的样本都是从多数类样本中选取最有代表性的.

Tomek Link 表示不同类别之间距离最近的一对样本，即这两个样本互为最近邻且分属不同类别[13]. 如果两个样本形成了一个 Tomek Link，则要么其中一个是噪音，要么两个样本都在边界附近. 这样通过移除 Tomek Link 就能“清洗掉”类间重叠样本，使得互为最近邻的样本都属于同一类别，从而能更好地进行

分类.

ENN的主要思想是如果有超过一半的$k$近邻点都不属于多数类的多数类样本，那么这个多数类样本会被剔除.

Cluster Centroids算法不是随机抽取原始样本，其每一个类别的样本都会用$k$-Means算法的中心点进行合成.

### 1.2 过采样

过采样是目前比较主流的处理数据不平衡的方法，其通过增加正样本的数量来平衡数据集中的正负样本，工作原理与欠采样相反. 过采样增加了正样本的数量和多样性，进而增加了正样本的数据特征，使得分类模型能够学习到更多的正样本特征，但同时这些生成的特征可能成为样本噪声，反而不利于分类模型对正样本的正确分类. 下面将介绍几种主流的过采样方法：

随机过采样的核心思想是随机的复制、重复少数类样本[14]，最终使得少数类与多数类的数量相当从而得到一个均衡的数据集.

SMOTE的思想是通过在少数类样本之间插值来生成少数类的新样本. 具体地，对于一个少数类样本$X_i$使用$k$近邻法，求出离$X_i$距离最近的$k$个少数类样本，样本之间用$n$维特征空间下的欧氏距离进行度量. 然后从$k$个近邻点中随机选取一个，使用下列公式生成新样本：

$$X_{\text{new}} = X_i + (\hat{X}_i - X_i) \times \delta \tag{1}$$

其中$\hat{X}$为选出的$k$近邻点，$\delta$是一个随机数且取值范围为[0，1].

Border-line SMOTE算法会先将所有的少数类样本分成3类noise，danger和safe，根据少数类样本的$k$近邻数来判断属于哪一类，如果所有的$k$近邻数都是多数类就属于noise类别，超过一半的$k$近邻数是多数类就属于danger类别，超过一半的$k$近邻数是少数类就属于safe类别. 该算法只会从danger类别的少数类样本中随机选择，然后用SMOTE算法合成新的样本. 因为danger类别处于边界附近，而处于边界附近的样本更容易被误分.

ADASYN(adaptive synthetic sampling)名为自适应合成抽样，其最大的特点是每个少数类生成新样本的数量是自动机制决定的，而SMOTE对每个少数类样本生成的数量都相同. ADASYN给每个少数类样本施加了一个权重，周围的多数类样本越多则权重越高，导致它易受离群点的影响.

### 1.3 混合采样

欠采样和过采样相结合的方法称为混合采样，通过样本生成模型生成一部分新的正样本，通过样本筛选模型选取一部分具有代表性的负样本，从而达到正负样本的数量相当. 混合采样旨在减少负样本的特征丢失，同时减少正样本的噪声生成. 主流的混合采样方法有SMOTEENN和SMOTETomek.

### 1.4 协同过滤算法

协同过滤算法是目前推荐系统中最常用的一种推荐策略，许多互联网公司如阿里巴巴、百度和腾讯等已经成功应用到实际系统中，其主要思想是根据用户的历史评价数据进行分析，计算得出用户间的相似性，结合相似度给出最终的推荐列表[15-16]. 协同过滤算法一般分为两种类型，分别是基于模型的协同过滤和基于近邻的协同过滤[17]. 基于模型的协同过滤是找一个包含用户和项目之间关系的优良子空间，这样就可以计算出评分[18]. 基于近邻的协同过滤易于理解，算法比较简单，但是在用户项评分比较稀疏的情况下，很难找到稳定可靠的近邻[19].

## 2 抽样推荐算法

### 2.1 抽样推荐算法的框架

由于数据特征与抽样方法性能之间存在一些内在的关系，即两个数据集的特征之间具有很高的相似度则被认为具有相同的适用抽样方法，故本文提出一种抽样推荐算法，用于自动选择新缺陷数据的适用抽样

方法. 抽样推荐算法的框架如图 1 所示，该算法由抽样方法排序、数据相似性挖掘和基于用户的推荐 3 个部分组成. 由图 1 可知，首先通过抽样方法在历史数据集上排序来建立一个在特定分类器下的抽样方法排名存储库. 然后基于数据特征，通过挖掘新数据与历史数据之间的相似性[20]，构建一个相似性存储库. 最后使用上述抽样方法排名存储库和数据相似性存储库构建一个三层网络，在此基础上为新数据集推荐适用的抽样方法.

图 2 给出了由 3 个历史数据集和 4 种抽样方法组成的推荐网络结构. 在图 2 中，第一层是新数据集，第二层的节点表示历史数据集. 第一层和第二层的连接权重是两个数据集之间的相似性得分. 此外，抽样方法都在第三层，排名得分作为第二层和第三层的连接权重. 在构建推荐网络后，根据所获得的推荐分数对所有抽样方法进行排序，并选择 Top-$N$ 排序方法向新数据集推荐抽样方法.

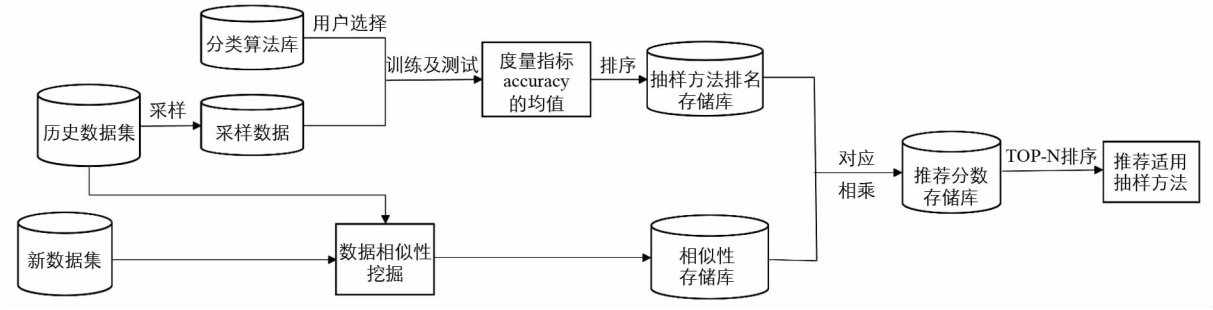


图 1　抽样推荐算法的框架图

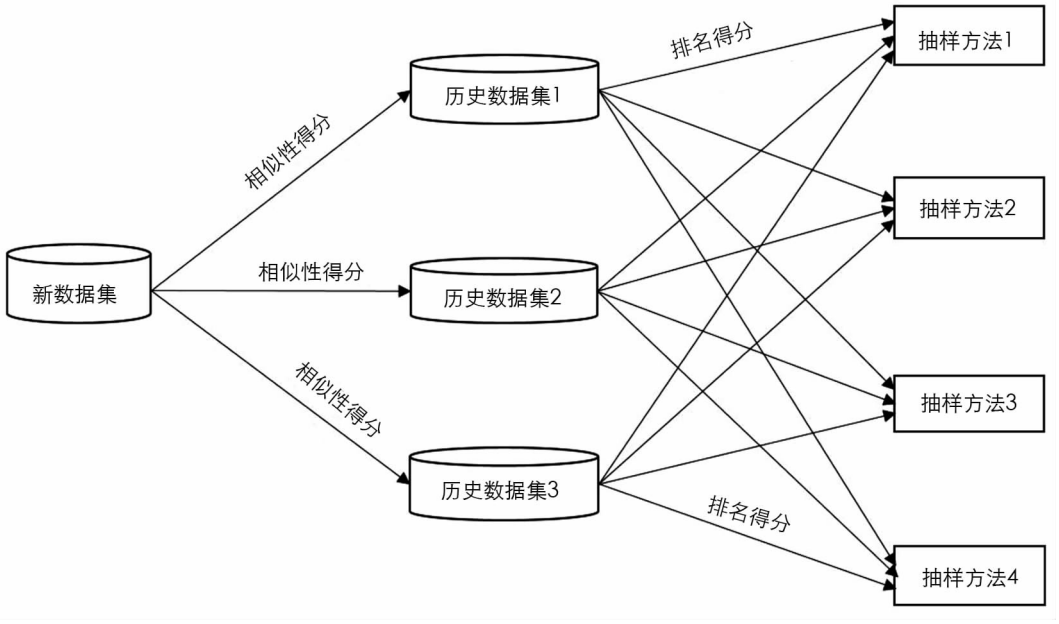


图 2　推荐网络结构图

## 2.2　抽样推荐算法流程

由于不同的抽样方法在处理给定的不平衡缺陷数据时可能表现出不同的性能，因此抽样方法排序的目的是根据这些不同抽样方法对指定数据的适用性提供详细的等级，这可以通过一些常见的分类评价指标来衡量，如准确率. 此外，抽样方法的性能可能随着各种分类算法的不同而变化. 算法 1 提供了获得抽样方法适用性的详细过程，在特定分类算法下采用 10 折交叉验证对软件缺陷数据进行训练，利用预测准确率作为对主流抽样方法排序的度量指标.

**算法 1** 获取抽样方法排名得分

输入：历史数据集$\{D_1, D_2, \cdots, D_m\}$
抽样方法$\{T_1, T_2, \cdots, T_n\}$
分类算法 $C$
输出：抽样方法排名得分
for $i=1$ to $m$ do
for $j=1$ to $n$ do
$R_{score}[i][j]=0$；
把历史数据集 $D_i$ 划分成 10 份；
for $k=1$ to 10 do
$D_{test}=b[k]$；
$D_{train}=D_i-D_{test}$；
使用抽样方法 $T_j$ 去训练并得到平衡训练集 $B_{train}$；
使用分类算法 $C$ 在 $B_{train}$ 上学习并得到一个分类预测器 $p$；
使用分类预测器 $p$ 在测试集上评估并得到相应的性能度量值 $a$；
$R_{score}[i][j]=R_{score}[i][j]+a$；
end for
$R_{score}[i][j]=R_{score}[i][j]/10$；
end for
end for
return $R_{score}$

挖掘数据集之间的相似性是推荐算法的重要组成部分，本文采用 Jaccard 系数来衡量两个数据集之间的相似性. Jaccard 系数的值越大，说明之间的相似性越高，其定义如下：

$$J(A, B)=\frac{|A \cap B|}{|A \cup B|} \tag{2}$$

具体获取数据集之间相似性得分的过程见算法 2.

**算法 2** 获取相似性得分

输入：历史数据集$\{D_1, D_2, \cdots, D_m\}$
新数据集 $N$
输出：相似性得分
for $i=1$ to $m$ do
$S_{score}=J(N, D_i)=\frac{|N \cap D_i|}{|N \cup D_i|}$；
end for
return $S_{score}$

通过前面的算法 1 和算法 2 可以得到抽样方法排名存储库和相似性存储库，利用这些信息就可以为用户的新数据集推荐适用的抽样方法，具体的过程见算法 3.

**算法 3** 抽样方法的推荐

输入：从算法 1 中获取 $R_{score}$
从算法 2 中获取 $S_{score}$
历史数据集的数量 $m$
抽样方法的数量 $n$
输出：抽样方法的 Top-$N$ 排序
使用抽样方法排名存储库和相似性存储库构建推荐网络；
for $j=1$ to $n$ do
$E_{score}[j]=\sum_{i=1}^{m} S_{score} \times R_{score}[i][j]$；
end for
根据推荐分数 $E_{score}$ 对抽样方法进行排序；
return Top-$N$ ranked methods

## 3　实例分析

### 3.1　实验数据

本文采用 NASA 提供的 MDP 软件缺陷数据集，对抽样推荐算法进行验证分析和推荐性能评价. 具体使用 MDP 数据集中的 10 个数据集文件构建了训练集，并选取了 PC3,JM1 和 PC5 这 3 个数据集进行验证，因为 PC3,JM1 和 PC5 的总模块数分别为 1 099,9 591 和 16 962，其数据集规模不断扩大，由此可以验证抽样推荐算法的可扩展性. 其相关信息如表 1 所示. 实验环境：Windows 10 操作系统，8GB 内存，使用 python 进行具体编程实现.

**表 1　NASA MDP 数据集**

| 数据集名称 | 语言 | 总模块数/块 | 有缺陷模块数/块 | 缺陷率/% | 特征数/个 |
|---|---|---|---|---|---|
| CM1 | C | 344 | 42 | 12.21 | 37 |
| JM1 | C | 9 591 | 1 759 | 18.34 | 21 |
| KC1 | C++ | 2 095 | 325 | 15.51 | 21 |
| KC3 | JAVA | 200 | 36 | 18.00 | 39 |
| KC4 | Perl | 125 | 61 | 48.80 | 40 |
| MC1 | C++ | 8 737 | 68 | 0.78 | 38 |
| MC2 | C | 125 | 44 | 35.20 | 39 |
| MW1 | C | 263 | 27 | 10.27 | 37 |
| PC1 | C | 735 | 61 | 8.30 | 37 |
| PC2 | C | 1 493 | 16 | 1.07 | 36 |
| PC3 | C | 1 099 | 138 | 12.56 | 37 |
| PC4 | C | 1 379 | 178 | 12.91 | 37 |
| PC5 | C++ | 16 962 | 502 | 2.96 | 38 |

### 3.2　评价指标

为了证明本文方法的有效性，通过准确率、平均准确率及命中率这些推荐算法的通用指标[16]进行评价. 在对抽样推荐算法进行评价时，需要兼顾考虑上述的评价指标.

假设 $R(u)$ 是为新数据集在特定分类算法下推荐的抽样方法列表，$T(u)$ 是新数据集在特定分类算法下实际的抽样方法列表. 准确率表示预测正确的样本数占总样本数的比例，准确率 Precision 记为 $P_1$，其定义如下：

$$P_1 = \frac{\sum_{u \in U} | R(u) \cap T(u) |}{\sum_{u \in U} | R(u) |} \tag{3}$$

对于用户 $u$，$\Omega_u$ 表示推荐的抽样方法在实际的抽样方法列表中，$p_{ui}$ 表示抽样方法 $i$ 在推荐列表中的位置，$p_{ui} > p_{uj}$ 表示抽样方法 $j$ 在推荐列表中排在抽样方法 $i$ 之前. 平均准确率 AP 记为 $P_2$，那么 $u$ 的平均准确率为

$$P_2 = \frac{1}{\Omega_u} \sum_{i \in \Omega_u} \frac{\sum_{j \in \Omega_u} h(p_{uj} < p_{ui}) + 1}{p_{ui}} \tag{4}$$

在 Top-$N$ 推荐中，HR 是一种常用的衡量召回率的指标，分母是所有推荐的抽样方法的个数总和，分子表示推荐的抽样方法出现在实际 Top-$N$ 列表中的个数总和.

### 3.3　实验结果分析

本文选取 CM1,KC1,KC3,KC4,MC1,MC2,MW1,PC1,PC2 和 PC4 这 10 个数据集作为历史数据集进行训练，并在 ADASYN,Borderline SMOTE,Cluster Centroids,ENN,Near Miss,Random Over Sampler,

Random Under Sampler,smote,SMOTEENN,SMOTETomek 和 TomekLinks 这 11 种主流的抽样方法上进行实验. 最邻近分类算法(KNN)、支持向量机(SVM)和决策树(DTC)是机器学习中的 3 种经典分类算法，被广泛应用于软件缺陷预测领域，故本文选取 KNN,SVM 和 DTC 作为特定的分类算法. 按照算法 1 的过程，得到训练集在 KNN,SVM,DTC 分类算法下的预测准确率，并作为抽样方法的排名指标，具体见表 2－4. 由于 ADASYN 抽样方法在 KC4 上并不适用，故在表中的值为 0.

**表 2 训练集在 KNN 分类算法下的预测准确率**

| 抽样方法名称 | CM1 | KC1 | KC3 | KC4 | MC1 | MC2 | MW1 | PC1 | PC2 | PC4 |
|---|---|---|---|---|---|---|---|---|---|---|
| ADASYN | 0.699 | 0.816 | 0.681 | 0 | 0.962 | 0.567 | 0.688 | 0.751 | 0.898 | 0.770 |
| Borderline SMOTE | 0.740 | 0.847 | 0.817 | 0.715 | 0.978 | 0.563 | 0.737 | 0.800 | 0.960 | 0.801 |
| Cluster Centroids | 0.500 | 0.517 | 0.495 | 0.757 | 0.712 | 0.493 | 0.441 | 0.476 | 0.560 | 0.562 |
| ENN | 0.900 | 0.898 | 0.833 | 0.941 | 0.994 | 0.733 | 0.882 | 0.906 | 0.991 | 0.852 |
| NearMiss | 0.812 | 0.975 | 0.732 | 0.732 | 0.863 | 0.648 | 0.859 | 0.792 | 0.910 | 0.865 |
| Random Over Sampler | 0.840 | 0.841 | 0.787 | 0.738 | 0.989 | 0.598 | 0.863 | 0.899 | 0.989 | 0.858 |
| Random Under Sampler | 0.692 | 0.668 | 0.591 | 0.743 | 0.741 | 0.544 | 0.459 | 0.568 | 0.800 | 0.523 |
| smote | 0.713 | 0.827 | 0.685 | 0.736 | 0.961 | 0.569 | 0.686 | 0.740 | 0.899 | 0.783 |
| SMOTEENN | 0.986 | 0.966 | 0.983 | 0.933 | 0.995 | 0.853 | 0.960 | 0.959 | 0.993 | 0.943 |
| SMOTETomek | 0.747 | 0.858 | 0.826 | 0.841 | 0.966 | 0.744 | 0.700 | 0.773 | 0.918 | 0.793 |
| Tomek Links | 0.831 | 0.837 | 0.804 | 0.809 | 0.993 | 0.645 | 0.845 | 0.906 | 0.990 | 0.838 |

**表 3 训练集在 SVM 分类算法下的预测准确率**

| 抽样方法名称 | CM1 | KC1 | KC3 | KC4 | MC1 | MC2 | MW1 | PC1 | PC2 | PC4 |
|---|---|---|---|---|---|---|---|---|---|---|
| ADASYN | 0.630 | 0.657 | 0.589 | 0 | 0.720 | 0.573 | 0.640 | 0.601 | 0.704 | 0.583 |
| Borderline SMOTE | 0.688 | 0.700 | 0.670 | 0.595 | 0.683 | 0.520 | 0.715 | 0.618 | 0.819 | 0.618 |
| Cluster Centroids | 0.565 | 0.559 | 0.455 | 0.619 | 0.546 | 0.507 | 0.429 | 0.519 | 0.710 | 0.617 |
| ENN | 0.863 | 0.866 | 0.819 | 0.79 | 0.992 | 0.533 | 0.857 | 0.915 | 0.991 | 0.833 |
| NearMiss | 0.673 | 0.798 | 0.732 | 0.589 | 0.580 | 0.707 | 0.794 | 0.684 | 0.680 | 0.741 |
| Random Over Sampler | 0.609 | 0.689 | 0.598 | 0.572 | 0.772 | 0.627 | 0.665 | 0.602 | 0.764 | 0.594 |
| Random Under Sampler | 0.577 | 0.674 | 0.527 | 0.565 | 0.541 | 0.567 | 0.529 | 0.505 | 0.640 | 0.562 |
| smote | 0.638 | 0.695 | 0.588 | 0.595 | 0.768 | 0.651 | 0.648 | 0.602 | 0.714 | 0.596 |
| SMOTEENN | 0.698 | 0.766 | 0.639 | 0.678 | 0.781 | 0.747 | 0.795 | 0.703 | 0.807 | 0.652 |
| SMOTETomek | 0.608 | 0.706 | 0.570 | 0.656 | 0.773 | 0.636 | 0.666 | 0.609 | 0.739 | 0.586 |
| TomekLinks | 0.873 | 0.854 | 0.798 | 0.674 | 0.992 | 0.694 | 0.895 | 0.915 | 0.991 | 0.860 |

**表 4 训练集在 DTC 分类算法下的预测准确率**

| 抽样方法名称 | CM1 | KC1 | KC3 | KC4 | MC1 | MC2 | MW1 | PC1 | PC2 | PC4 |
|---|---|---|---|---|---|---|---|---|---|---|
| ADASYN | 0.858 | 0.867 | 0.829 | 0 | 0.990 | 0.680 | 0.856 | 0.902 | 0.987 | 0.915 |
| BorderlineSMOTE | 0.881 | 0.871 | 0.841 | 0.719 | 0.993 | 0.705 | 0.871 | 0.909 | 0.988 | 0.915 |
| Cluster Centroids | 0.704 | 0.647 | 0.596 | 0.730 | 0.883 | 0.578 | 0.752 | 0.784 | 0.690 | 0.837 |
| ENN | 0.803 | 0.868 | 0.767 | 0.922 | 0.993 | 0.709 | 0.790 | 0.874 | 0.981 | 0.869 |
| Near Miss | 0.808 | 0.973 | 0.705 | 0.717 | 0.908 | 0.659 | 0.821 | 0.741 | 0.790 | 0.911 |
| Random Over Sampler | 0.937 | 0.905 | 0.897 | 0.710 | 0.992 | 0.746 | 0.946 | 0.960 | 0.993 | 0.954 |
| Random Under Sampler | 0.642 | 0.682 | 0.655 | 0.732 | 0.810 | 0.618 | 0.581 | 0.638 | 0.860 | 0.823 |
| smote | 0.863 | 0.867 | 0.840 | 0.717 | 0.991 | 0.708 | 0.854 | 0.911 | 0.989 | 0.914 |
| SMOTEENN | 0.914 | 0.937 | 0.915 | 0.953 | 0.998 | 0.793 | 0.919 | 0.955 | 0.994 | 0.945 |
| SMOTETomek | 0.856 | 0.876 | 0.837 | 0.820 | 0.991 | 0.675 | 0.870 | 0.904 | 0.987 | 0.914 |
| Tomek Links | 0.802 | 0.825 | 0.753 | 0.803 | 0.992 | 0.652 | 0.837 | 0.868 | 0.979 | 0.878 |

本文选取 PC3,JM1 和 PC53 个数据集进行验证，按照算法 2 的流程，得到 PC3,JM1 和 PC5 与另外 10 个训练集之间的相似度，并作为其相似性得分，具体见表 5.

**表 5　测试集的相似性得分**

| 数据集名称 | PC3 | JM1 | PC5 |
|---|---|---|---|
| CM1 | 37/37 | 21/37 | 36/39 |
| KC1 | 21/37 | 21/21 | 21/38 |
| KC3 | 37/39 | 21/39 | 38/39 |
| KC4 | 37/40 | 21/40 | 38/40 |
| MC1 | 36/39 | 21/38 | 38/38 |
| MC2 | 37/39 | 21/39 | 38/39 |
| MW1 | 37/37 | 21/37 | 36/39 |
| PC1 | 37/37 | 21/37 | 36/39 |
| PC2 | 36/37 | 20/37 | 35/39 |
| PC4 | 37/37 | 21/37 | 36/39 |

本文选取 TOP-5 进行推荐性能评价，按照算法 3 的流程，得到了 PC3,JM1 和 PC5 的排名前五的推荐抽样方法，具体见表 6.

**表 6　TOP-5 的推荐抽样方法**

| 分类 | PC3 预测 | PC3 实际 | JM1 预测 | JM1 实际 | PC5 预测 | PC5 实际 |
|---|---|---|---|---|---|---|
| KNN | SMOTEENN | SMOTEENN | NearMiss | SMOTEENN | SMOTEENN | SMOTEENN |
| | ENN | RandomOver Sampler | SMOTEENN | NearMiss | ENN | ENN |
| | TomekLinks | ENN | ENN | ENN | TomekLinks | RandomOver Sampler |
| | RandomOver Sampler | TomekLinks | SMOTETomek | RandomOver Sampler | RandomOver Sampler | SMOTETomek |
| | Borderline SMOTE | SMOTETomek | Borderline SMOTE | SMOTETomek | Borderline SMOTE | TomekLinks |
| SVM | ENN | TomekLinks | ENN | TomekLinks | ENN | ENN |
| | TomekLinks | ENN | TomekLinks | ENN | TomekLinks | TomekLinks |
| | Borderline SMOTE | NearMiss | NearMiss | NearMiss | SMOTEENN | Borderline SMOTE |
| | SMOTEENN | SMOTEENN | SMOTEENN | SMOTEENN | SMOTETomek | SMOTEENN |
| | NearMiss | Borderline SMOTE | SMOTETomek | SMOTETomek | RandomOver Sampler | ADASYN |
| DTC | SMOTEENN | RandomOver Sampler | NearMiss | NearMiss | SMOTEENN | SMOTEENN |
| | RandomOver Sampler | SMOTEENN | SMOTEENN | SMOTEENN | ENN | ENN |
| | smote | Borderline SMOTE | RandomOver Sampler | RandomOver Sampler | Borderline SMOTE | RandomOver Sampler |
| | Borderline SMOTE | SMOTETomek | SMOTETomek | SMOTETomek | RandomOver Sampler | SMOTETomek |
| | ADASYN | ADASYN | Borderline SMOTE | ADASYN | TomekLinks | Borderline SMOTE |

表 6 中不仅有 Top-5 的推荐抽样方法，还给出了测试集在特定分类器下实际表现前 5 的抽样方法. 本文选取 Top-5 推荐列表中的第一个抽样方法作为实际推荐给用户的抽样方法，因为在 9 次推荐中就有 5 次推荐分数最高的第一个抽样方法与实际表现最好的抽样方法相同，而在剩下的 4 次推荐中，推荐的第一个抽样方法也位于实际表现的第二位.

从表 6 中可以看出，在 SVM 作为分类器的前提下，JM1 和 PC3 的预测前 5 的抽样方法都出现在实际

表现前5的抽样方法列表中，特别是在JM1的预测中，除了第一和第二的抽样方法顺序不一样外，其余都完全一样. 不论是在小规模数据集PC3上，还是在中等规模数据集JM1以及大规模数据集PC5上，最差的预测也有3个抽样方法预测正确，而且在实际中表现良好的抽样方法也位于推荐列表的前面，说明本文的推荐算法性能良好，并且可扩展性好.

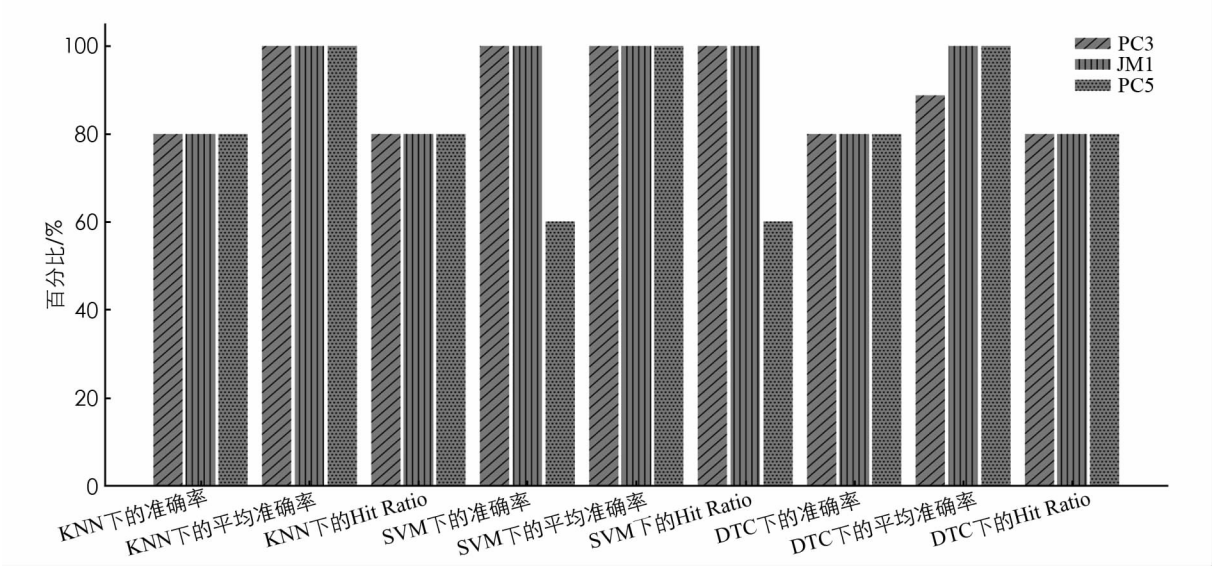


图3 抽样推荐算法的性能评价

图3给出了PC3,JM1和PC5在特定分类器下的准确率、平均准确率和Hit Ratio，由图3可知，本文提出的抽样推荐算法在测试集上总的准确率达到了0.82，其准确率是非常高的，平均推荐的5种抽样方法中就有4种抽样方法在实际中表现良好. 平均准确率也是一个重要的推荐指标，在9次测试中，本文提出的算法就有8次的平均准确率为1，最差的也能达到0.8875，说明实际表现良好的抽样方法基本上都在推荐列表的前面. Hit Ratio的均值达到了0.823，根据这些评价指标综合考虑，可以看出本文提出的抽样推荐算法的推荐性能良好且可扩展性好，可以为用户推荐出适用的抽样方法.

## 4 结束语

本文提出了面向软件缺陷数据的协同过滤抽样推荐算法. 该算法首先在特定的分类器下计算训练集在11种主流的抽样方法处理后的预测准确率，并以此为衡量标准对抽样方法进行排序，然后使用杰卡德相似系数计算测试集与训练集之间的相似度，最后通过前面的排名分数和相似度值就能得到推荐分数，根据推荐分数为用户推荐适用的抽样方法. 在NASA的数据集上开展的验证实验证明所提算法能为新数据集推荐出适用的抽样方法，为后面的软件缺陷预测奠定了良好的基础.

为了证明抽样推荐算法的可行性和有效性，本文的训练集和测试集都是在NASA缺陷数据集中选取的，是因为这样能更容易地找到与测试集相似度高的训练集，不需要大量的缺陷数据集进行训练. 如果训练集从其他的缺陷数据集中选取，或者只选取与测试集相似度低的数据集进行训练，其推荐性能会下降，推荐效率会降低，但只要训练集达到一定的规模，同样可以为新的缺陷数据集推荐出性能良好的抽样方法，因此本文中抽样方法的排名结果不具有局限性. 软件缺陷数据集的数量是比较多的，比如常用的有NASA MDP,PROMISE和Eclipse等数据集，所以能找到足够数量的缺陷数据集进行训练，这样就能在很大程度上解决上述问题. 本文在算法1中使用预测准确率作为抽样方法排名的指标，还有一些其它的度量指标，比如AUC和F-Measure，使用其它的度量指标进行抽样方法排序是进一步的研究方向.

**参考文献：**

[1] 邓文凯. 不平衡数据分类研究及其在污水处理系统中的应用[D]. 广州：华南理工大学，2017.

[2] 娄丰鹏. 基于相关性分析的跨项目软件缺陷预测方法研究 [D]. 南京：南京邮电大学，2018.
[3] 张　艳. 面向不平衡数据的离群点检测研究 [D]. 青岛：青岛科技大学，2017.
[4] ZHAO H，LI X J. A Cost Sensitive Decision Tree Algorithm Based on Weighted Class Distribution with Batch Deleting Attribute Mechanism [J]. Information Sciences，2017，378：303-316.
[5] PÉREZ-RODRÍGUEZ J，ARROYO-PEÑA A G，GARCÍA-PEDRAJAS N. Simultaneous Instance and Feature Selection and Weighting Using Evolutionary Computation：Proposal and Study [J]. Applied Soft Computing，2015，37：416-443.
[6] LIN W C，TSAI C F，HU Y H，et al. Clustering-Based Undersampling in Class-Imbalanced Data [J]. Information Sciences，2017，409-410：17-26.
[7] 古　平，杨　炀. 面向不均衡数据集中少数类细分的过采样算法 [J]. 计算机工程，2017，43(2)：241-247.
[8] 易　未，毛　力，孙　俊，等. 改进 Smote 算法在不平衡数据集上的分类研究 [J]. 计算机与现代化，2018(3)：83-88.
[9] 杨　毅，卢诚波，徐根海. 面向不平衡数据集的一种精化 Borderline-SMOTE 方法 [J]. 复旦学报(自然科学版)，2017，56(5)：537-544.
[10] BATISTA G E A P A，PRATI R C，MONARD M C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data [J]. ACM SIGKDD Explorations Newsletter，2004，6(1)：20-29.
[11] 崔　鑫，徐　华，宿　晨. 面向不均衡数据集的过抽样算法 [J]. 计算机应用，2020，40(6)：1662-1667.
[12] WOLPERT D H，MACREADY W G. No Free Lunch Theorems for Optimization [J]. IEEETransactions on Evolutionary Computation，1997，1(1)：67-82.
[13] 邱凌峰. 基于机器学习的社会安全风险分析研究 [D]. 北京：中国人民公安大学，2019.
[14] 覃　朗，朱建军，衣柏衡，等. 非均衡数据下基于信息增益的 SMOTE 改进 SVM 模型研究 [J]. 中国管理科学，2016，24(S1)：128-136.
[15] 卢竹兵，马小琴，吴汶娟，等. 基于情感分析和情感遗忘的协同过滤推荐策略 [J]. 重庆师范大学学报(自然科学版)，2020，37(5)：103-108.
[16] 漆　月，周　欢. 基于图书分类号的自适应个性化图书推荐系统的研究 [J]. 西南师范大学学报(自然科学版)，2014，39(4)：210-214.
[17] SARWAR B，KARYPIS G，KONSTAN J，et al. Item-Based Collaborative Filtering Recommendation Algorithms [C]// Proceedings of the Tenth International Conference on World Wide Web. New York：ACM Press，2001：285-295.
[18] 纪　平，胡学友，杨文娟，等. 基于矩阵分解的协同过滤推荐算法 [J]. 合肥学院学报(综合版)，2020，37(5)：10-18.
[19] KOREN Y，BELL R，VOLINSKY C. Matrix Factorization Techniques for Recommender Systems [J]. Computer，2009，42(8)：30-37.
[20] 褚　菲. CCPP 煤气系统建模与运行优化研究 [D]. 沈阳：东北大学，2013.

**责任编辑**　张　枸