数据挖掘大作业实验报告

银 琦

[1](南京大学 计算机科学与技术学院,江苏 南京 210046)

通讯作者: 银琦, E-mail: 141220132@smail.nju.edu.cn

摘 要: 在开发项目期间，代码更改可能导致项目构建失败，训练该模型并在软件测试平台定时进行预测，可以尽早得知项目构建失败，降低成本。我对给定的数据集进行预处理，选择合适的特征，对部分特征的数据进行了二次处理，通过采样选取训练数据，使用 J48, Neural Network, BayesNet, AdaBoost, 等算法训练模型，进行十折交叉验证，并对测试集进行预测。最终我选择了 5 种属性，用 BayesNet 训练模型，预测出结果，F1 为 0.42446。

关键词: 软件测试;数据挖掘;数据预处理;机器学习;模型评估

The report of Data Mining Practice

YIN Qi

[1](School of Computer Science and Technology, Nanjing University, Nanjing 210046, China)

Abstract: During the development of a software project, sometimes code changes lead to build failures, train this model and run build test periodically can make the developers notice the problem earlier and decrease cost. My work includes data preprocessing, choosing suitable attributes, processing original data to get more useful attributes, generating train set by sampling, training models with J48, Neural Network, BayesNet, AdaBoost, verifying the model with 10-fold cross validation with the train set, and making a prediction with the test set. Finally I chose 5 attributes, trained the model with BayesNet, and got the predicting results which F1 is 0.42446.

Key words: software prediction; data mining; data preprocessing; machine learning; model evaluation

1 实验分析

在开发项目期间，代码的更改有时会导致项目构建失败，构建测试会在若干次提交之后被某些特定的条件触发，频繁的测试使得成本提高，而项目过早的构建失败会导致效率低下，所以本次实验希望从已经测试过的项目的构建日志中提取一些重要信息，对已经提交但是还未测试的项目进行测试，有助于开发人员更在地注意到问题。本次试验的主要流程为：数据预处理、分类算法选择、训练模型、预测训练结果、结果评估。根据结果评估的信息重新处理数据，选择算法，直到结果不变或者达到预期目标。

在数据预处理中主要选取了特征，处理了缺失值，去除了噪声数据，并通过采样选取了训练集。

选取特征不一定要使用原始数据，本实验中对 project_name, author, committer 与 build_result 进行了二次处理，project_name 和 build_result 可以求出该工程 passed 和 failed 所占的比例，计算 failed 占的比例得到属性 rate；author 和 build_result 可以求出在所有的工程中该作者 passed 和 failed 各自占的比例，计算 passed 所占的比例得到属性 author_acc，committer 同理得到 committer_acc。

缺失值的处理是本次实验中的一个关键，因为可能存在 author 和 committer 在测试集中单不在训练集中，所以这部分 author_acc 和 committer_acc 就是缺失值，一开始将其均设为-1，但是由于训练模型时不会出现-1 的情况，所以对预测结果产生了不良的影响，后来将缺失值设置为所有 author_acc(committer_acc)的均值，对实验结果有了很大的提高。

2 算法原理

2.1 J48决策树

决策树算法基于从上到下的策略，递归的分治策略，选择某个属性放置在根节点，为每个可能的属性值产生一个分支，将实例分成多个子集，每个子集对应一个根节点的分支，然后在每个分支上递归地重复这个

过程。当所有实例有相同的分类时，停止。

决策树算法的优点：在相对短的时间内能够对大型数据源做出可行且效果良好的结果；

决策树算法的缺点：对于那些各类别样本数量不一致的数据，在决策树当中,信息增益的结果偏向于那些具有更多数值的特征；容易出现过度拟合的问题；忽略了数据集中属性之间的相关性。

2.2 NaiveBayes

朴素贝叶斯分类器采用了“属性条件独立假设”：对已知类别，假设所有属性相互独立。对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，此待分类项的类别为概率最大者。

朴素贝叶斯算法的优点：朴素贝叶斯模型发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率；朴素贝叶斯算法模型所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。

朴素贝叶斯算法的缺点：理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的，这给朴素贝叶斯模型的正确分类带来了一定影响。在属性个数比较多或者属性之间相关性较大时，朴素贝叶斯模型的分类效率比不上决策树模型。

2.3 BayesNet

由有向无环图和条件概率表组成，有向无环图的每个节点代表一个随机变量，变量之间的连线表示一个概率依赖。每个变量都有一个条件概率表。可以用贝叶斯网络在给定其他变量的观察值时推理出某些目标变量的值。贝叶斯学习理论将先验知识与样本信息相结合、依赖关系与概率表示相结合，是数据挖掘和不确定性知识表示的理想模型。

贝叶斯网络优点：贝叶斯网络能够方便的处理不完全数据；贝叶斯网络能够学习变量间的因果关系，而因果关系是数据挖掘中极为重要的模式；贝叶斯网络与贝叶斯统计相结合能够充分利用领域知识和样本数据的信息。

2.4 BP神经网络

BP 神经网络首先初始化权重，然后向前传播输入，得到神经网络的输出后计算均方误差，并由均方误差得到梯度，反向传播误差，更新连接权与阈值，循环直至误差很小。

BP 神经网络的优点：分类的准确度高，对噪声神经有较强的鲁棒性和容错能力，能充分逼近复杂的非线性关系，具备联想记忆的功能等。

BP 神经网络的缺点：神经网络需要大量的参数，如网络拓扑结构、权值和阈值的初始值；不能观察之间的学习过程，输出结果难以解释，会影响到结果的可信度和可接受程度；学习时间过长,甚至可能达不到学习的目的。

2.5 SVM

SVM 方法是通过一个非线性映射 p，把样本空间映射到一个高维乃至无穷维的特征空间中（Hilbert 空间），使得在原来的样本空间中非线性可分的问题转化为在特征空间中的线性可分的问题。

SVM 的优点：可以解决小样本情况下的问题；可以提高泛化性能；可以解决高维问题；可以解决非线性问题；可以避免神经网络结构选择和局部极小点问题。

SVM 的缺点：对缺失数据敏感；对非线性问题没有通用解决方案，必须谨慎选择核函数来处理。

2.6 AdaBoost

AdaBoost 是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器，即弱分类器，然后把这些弱分类器集合起来，构造一个更强的最终分类器。

AdaBoost 优点：adaboost 是一种有很高精度的分类器；可以使用各种方法构建子分类器，adaboost 算法提供的是框架；当使用简单分类器时，计算出的结果是可以理解的，而且弱分类器构造极其简单；简单，不用做特征筛选；不用担心 overfitting。

AdaBoost 缺点：. 对分类错误的样本多次被分错而多次加权后，权重过大，影响分类器的选择，造成退化问题（需改进权值更新方式）；数据不平衡问题导致分类精度的急剧下降；算法训练耗时，拓展困难。

2.7 Bagging

Bagging 从训练集中采样得到新的训练集，重复多次，每次训练一棵树，平均每一棵树的预测值或采用少数服从多数得到分类结果。

Bagging 优点：泛化能力强。

2.8 Random Forest

Random Forest 从训练集中采样得到新的训练集，重复多次得到多个训练集，针对每个个不同的训练集分别训练一棵树，训练树的过程中，先从所有特征中随机选择 m 个特征作为候选，然后再从这 m 个特征中选择最优的一个来划分预测空间。

Random Forest 优点：训练可以高度并行化，对于大数据时代的大样本训练速度有优势；由于可以随机选择决策树节点划分特征，这样在样本特征维度很高的时候，仍然能高效的训练模型；在训练后，可以给出各个特征对于输出的重要性；由于采用了随机采样，训练出的模型的方差小，泛化能力强；相对于 Adaboost，Random Forest 实现比较简单；对部分特征缺失不敏感。

Random Forest 缺点：在某些噪音比较大的样本集上，Random Forest 模型容易陷入过拟合；取值划分比较多的特征容易对 Random Forest 的决策产生更大的影响，从而影响拟合的模型的效果。

3 数据预处理

3.1 数据分析及特征选择

数据包含 532 个工程，每个工程都有其测试集和训练集，每个文件都是一个 Json 数组，数组中每个元素包含四个字段：project_name, build_result, commits, build_id。选择的特征如下（会根据不同的数据预处理方式及算法从下列特征中选择部分特征）：

(1) project_id(project_name)：每个 project_name 对应一个 project_id，若使用 project_name 作为属性，则需要去除字符串中的“’”，“,”等多余的字符，仅保留数字字母和下划线。
(2) build_id：每条记录的 build id。
(3) rate：工程的构建失败率，即在同一个工程下，build failed 的次数占 build 总次数的比重。
(4) author_acc：在所有项目中，某个 author build passed 的次数占该 author build 总次数的比重。
(5) author_id：该 author 的 id。
(6) committer_acc：在所有项目中，某个 committer build passed 的次数占该 committer build 总次数的比重。
(7) committer_id：该 committer 的 id。
(8) stats_total：记录中的 stats_total，若一条记录包含多个 commit，则将其累加起来。
(9) stats_additions：记录中的 stats_ additions，若一条记录包含多个 commit，则将其累加起来。
(10) stats_delections：记录中的 stats_ delections，若一条记录包含多个 commit，则将其累加起来。
(11) files_tatus：记录中的 files_tatus，有 5 种取值，added, removed, modified, renamed, none 若一条记录包含多个 commit，则统计这 5 种取值出现的次数，选择最多的一个作为该条记录的取值。
(12) files_additions：记录中的 files_additions，若一条记录包含多个 commit，则将其累加起来。
(13) files_delections：记录中的 files_ delections，若一条记录包含多个 commit，则将其累加起来。
(14) files_changes：记录中的 files_ changes，若一条记录包含多个 commit，则将其累加起来。
(15) num_commit：该条记录中 commit 的数量。
(16) build_result：分类结果。

3.2 数据预处理

3.2.1 训练集选取

(1) 每个工程单独训练模型

该种处理方式下，选择的特征有：build_id, author_id, committer_id, stats_total, stats_additions, stats_delections, files_status, files_additions, files_delections, files_changes, build_result。

build_id, author_id 和 committer_id 加上后面的特征经过训练可能会得到不同的 author 和 committer 对该工程 build passed 的贡献程度；stats 和 files 的各项数据可能会影响工程是否 build passed，理论上这些数据的数量越大说明改动越多，那么 build failed 的可能性也越大。

(2) 所有工程训练一个模型

该种处理方式下，选择的特征有：project_id, rate, author_acc, committer_acc, stats_total, stats_additions, stats_delections, files_status, files_additions, files_delections, files_changes, num_commit, build_result。

因为将所以的训练数据放到了一起，所以需要 project_id 来代表每个工程，在处理过程中发现若将其设置为 numeric 类型，则 weka 在离散化时会将其按区间离散化，这会导致模型训练不准确，所以将 project_id 设置为枚举类型，1-532 分别代表一个工程。

author_id 和 committer_id 具有同样的问题，但不太好处理，因为训练集和测试集的 author 和 committer 并不一致，所以删除了这两个属性，并且根据训练集提前统计出了每个 author 和 committer 的通过率，在测试集中，如果 author 在训练集中出现过，那么将 author_acc 设置为训练集中得出的 author_acc；若测试集中的 author 没有在训练集中出现过，即为缺失，此处对缺失值的处理为取所有 author_acc 的平均值，committer_acc 的处理方式与 author_acc 处理方式相同。

rate 特征对结果有挺大的影响，使用训练集计算出该工程的 rate，在测试集对应的工程中使用同样的 rate。

commit 的数量可能会对结果产生影响，因为 commit 数量越多，build failed 可能性越大，所以选择了属性 mun_commit。

3.2.2 缺失值处理

可能出现缺失值的特征有：author_id , committer_id, author_acc, committer_acc。

author_id , committer_id 若缺失，则赋值为-1；author_acc, committer_acc 在测试集的数据处理时可能会缺失，若缺失，则将 author_acc, committer_acc 赋值为均值。

3.2.3 噪声数据处理

在训练集中，build result 共有三种情况：passed、failed、errored。其中 errored 为噪声数据，在处理训练集时将其去除。

对于测试集，根据 non_error_build_ids.csv 中的 build id，将多余的测试数据去除，最终得到的结果只包含 passed 和 failed。

3.2.4 数据采样

使用了两种方式对数据进行处理。

(1) 不采取采样措施，只按照类别比例进行简单随机抽样，多次学习并预测，对得到的结果取平均值。如果 D 被划分成互不相交的部分，称作“层”，则通过对每一层的简单随机抽样就可以得到 D 的分层抽样，特别是当数据倾斜时，这可以帮助确保样本的代表性。

(2) 过采样：failed 的数量小于 passed 的数量，所以对 failed 采取过采样，即增加一些 failed 的样本，使得 failed 与 passed 数据量接近，从过采样的数据中按照类别比例进行简单随机抽样，多次学习并预测，对得到的结果取平均值。

4 算法选择

我选择的算法：贝叶斯网络。

在我所选择的特征中，我认为存在一些因果关系，如 author_acc, committer_acc 如果较高，会导致 rate

降低等，贝叶斯网络能够挖掘变量中的因果关系；贝叶斯网络具有强大的不确定性问题处理能力，贝叶斯网络用条件概率表达各个信息要素之间的相关关系，能在有限的，不完整的，不确定的信息条件下进行学习和推理；贝叶斯网络能有效地进行多源信息表达与融合。

除了理论分析外，我对数据在训练集上采取了十折交叉验证，比较了 J48，NaiveBayes，BayesNet，AdaBoost 的 F-Measure，得出了 BayesNet 得到的 F-Measure 值最高的结论。

综上，我选择了算法贝叶斯网络。

5 实验结果

5.1 模型评估

最终的 F1 值为 0.42446，各处理方式、算法得到的 F1 值如下：

注：表中 stats 代表了与 stats 相关的三个属性，files 代表了与 files 相关的 4 个属性，√表示选择了该特征或采用了该方法。

5.1.1 每个工程单独训练模型

	build_id	author_id	committer_id	stats	files	num_commit	F1
神经网络	√	√	√	√	√		0.35886
神经网络	√	√	√	√	√	√	0.3618
libsvm	√	√	√	√	√	√	0.36181
Adaboost	√	√	√	√	√	√	0.3403

5.1.2 所有工程训练一个模型

(1) 本地十折交叉验证（包含了 3.2.1 (2)中的所有特征）

	J48	NaiveBayes	BayesNet	AdaBoost
F-Measure	0.421	0.515	0.544	0.459

(2) 测试集的 F1，根据十折交叉验证结果及 2.4 中的分析选择了算法 BayesNet。

project_id	rate	author_acc	committer_acc	stats	files	num_commit	采样	F1
√				√	√	√		0.37624
√	√			√	√	√		0.38596
√	√	√	√	√	√	√		0.38823
√	√	√	√	√	√	√	√	0.38596
√	√	缺失值重新处理		√	√	√	√	0.42359
√	√	√	√			√	√	**0.42446**

5.2 结果分析

(1) 总体来看，每个工程单独训练模型的效果不如所有工程训练一个模型，因为单独训练模型数据量小，数据量不足可能对结果产生影响，并且有些数据可能不具有代表性。

(2) 在每个工程单独训练模型的方法中，神经网络和 libsvm 均取得了较好的效果，但是在只有一个模型的方法中没有采用这两种算法，因为这两种算法需要花费大量的时间和空间，还有可能得不到预期的效果。

(3) 一开始没有对数据进行二次处理，只是简单的获取记录中的数据，用其训练模型并进行预测，但是结果并不理想，后来首先加入了属性 rate，有所提升，但是效果不是很明显，于是进一步处理，加入了 author_acc 和 committer_acc，也略有提升；将缺失值从-1 改为均值后，F1 才有了大幅提升，所以对数据的预处理十分重要。

(4) 尝试去掉了部分属性，F1 值反而有所提升，说明了并不是属性个数越多训练出的模型效果就会更好。

6 心得体会及不足之处

(1) 对缺失值的处理十分重要。加入 author_acc 和 committer_acc 这两个属性后，一开始将其缺失值均记为-1，另成一类，最高的 F1 值为 0.38596，后来做出修改，将缺失值改成了均值，F1 值提升到了 0.42359，说明在该实验中，缺失值取均值分类效果更好。

(2) 仅看最终结果，0.42446 这个数据不太理想，说明模型建立的不够好，可能有两个方面的原因：第一是特征的选择不够恰当或者有些需要二次处理的数据没有处理；第二是算法选择的不对或者算法的参数未能调节到最理想的值。

(3) 调用 weka 的分类函数直接使用有些弊端，大部分分类算法不能处理连续值，所以要对数据进行离散化，weka 默认的离散化方式用区间划分，但是如果选择了 build_id, author_id, committer_id 这些属性，用区间来离散化就毫无意义，需要直接将其处理为枚举类型，而这一步是我所缺少的。

References:

[1] https://cs.nju.edu.cn/lim/courses/IntroDM/MiningPractice.htm

[2] http://www.itkeyword.com/doc/4549987322741556x247/SMOTE-WEKA-Resample

[3] http://news.ifeng.com/a/20170613/51239799_0.shtml

[4] http://blog.csdn.net/xiaozezepingping/article/details/26494401

[5] http://www.cnblogs.com/en-heng/p/5013995.html

附中文参考文献:

[1] Jiawei Han, Micheline Kamner, Jian Pei 著，范明，孟小峰译. 数据挖掘概念与技术 第三版

[2] 周志华著 机器学习