作业3: Aliens游戏

王琛然（151220104、17721502736）

(南京大学 计算机科学与技术系, 南京 210093)

摘 要: 随着人工智能的迅速发展，如何训练机器、如何使机器学习成了主要的问题。机器学习的方法大体分为监督学习、无监督学习和半监督学习。本次实验仅涉及监督学习，讨论分析监督学习的几种常见算法：朴素贝叶斯算法、对数几率回归算法、决策树算法和随机森林四种算法。本次实验中，引入 Weka 机器学习包，对训练数据集分别调用不同的算法进行有监督学习。

关键词: 监督学习；朴素贝叶斯算法；对数几率回归；决策树；随机森林；Weka

中图法分类号: TP301　　　文献标识码: A

# 引言

监督学习是指：从已有的训练数据集中学习出一个模型参数，从而可以使用得到的模型参数对新的数据进行预测。监督学习的数据集包括输入和输出，即特征和目标。在本次实验中，通过运行Train.java文件进行游戏，获得训练数据集；然后通过运行Weka机器学习包中的不同算法得到不同的模型参数；再运行Test.java文件进行预测测试。

# 监督学习算法

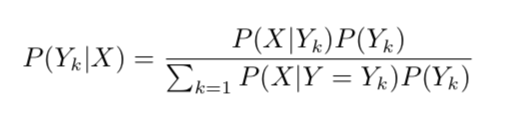
## 朴素贝叶斯算法

### 算法介绍

贝叶斯分类算法以贝叶斯定理为基础，朴素贝叶斯是其中运用最为广泛的算法之一，它基于特征属性之间相互条件独立的假设，对于给出的待分类特征求出该特征在各个类别下出现的概率并取最大概率的类别作为自己的分类结果。

### 算法过程

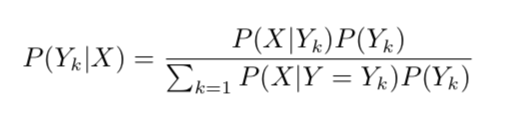
从概率论与统计学角度来说，我们已知贝叶斯公式为：



再从数据角度来分析，朴素贝叶斯分类算法是以贝叶斯定理为基础：

1. 分类模型样本X有m个特征属性集合
2. 特征输出分为k个类别，分别为Y1, Y2, …, Yk，
3. 假设X中的所有属性相互条件独立
4. 朴素贝叶斯的先验概率：P(Yk) = P(Y = Yk)

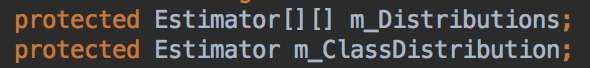
条件概率：



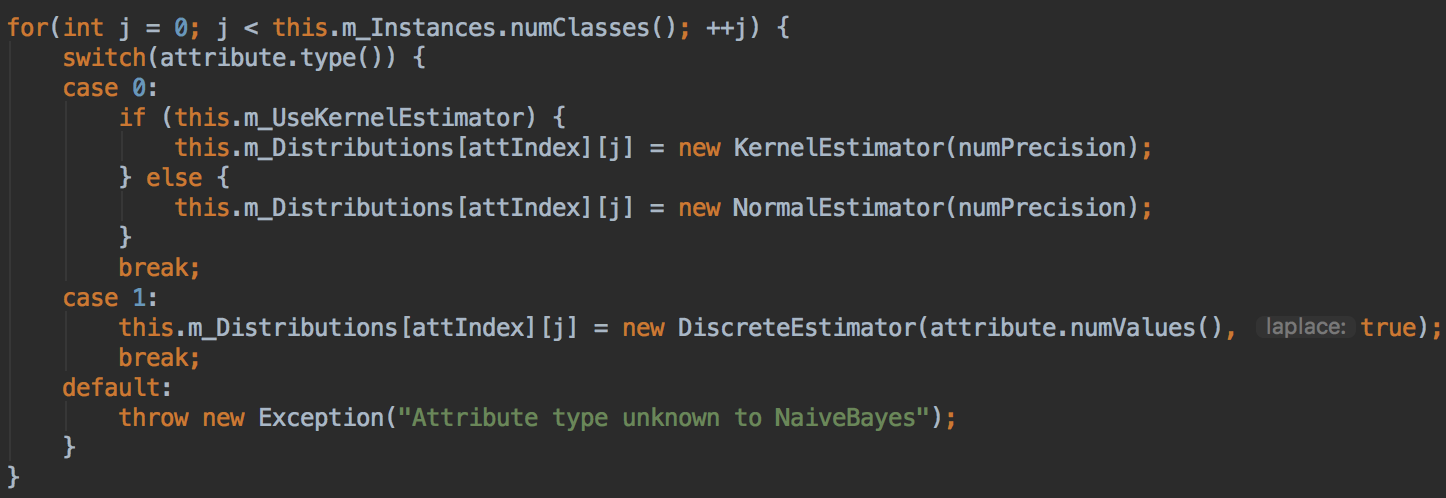
1. 对于每一个特征属性，比较（5）中的所有类别Yk条件概率的大小，有最大概率的类别Yk即为其分类结果。

### 算法源代码分析

1. 代码中，m\_Distributions表示条件概率，m\_ClassDistribution表示先验概率：



1. **buildClassifier**函数为每一个类创建一个Estimator分类器：



1. 样本分类函数**distributionForInstance**对输入样本进行分类：扫描每一个样本，temp存储计算得到的类条件概率，防止结果过小，将得到的结果放大



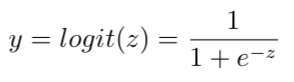
## 逻辑回归算法

### 算法介绍

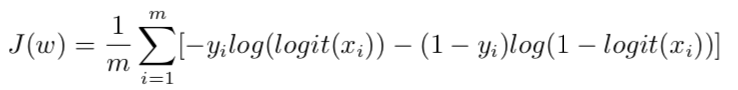
逻辑回归算法直接对分类的可能行建模，适合做分类任务，多处理二分类问题。它将数据拟合到一个logit函数，确立代价函数，通过优化方法求解出最优的模型参数，即一组权值，对于输入的测试集，这组权值与测试数据线性加和，并根据结果得到分类结果。

### 算法过程

1. 构造预测函数logit：



1. 构造损失函数J，通过极大似然估计推导得：（二分类情况下）



1. 使用优化方法（如梯度下降法或牛顿迭代法）使损失函数J最小，得到模型参数。

### 算法源代码分析

1. nC：训练集的样本个数；

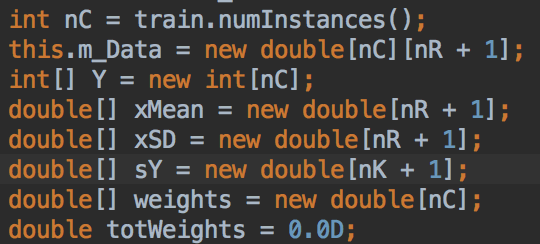
xMean[i]：前i个特征的平均值；

Xsd[i]：前i个特征的标准差；

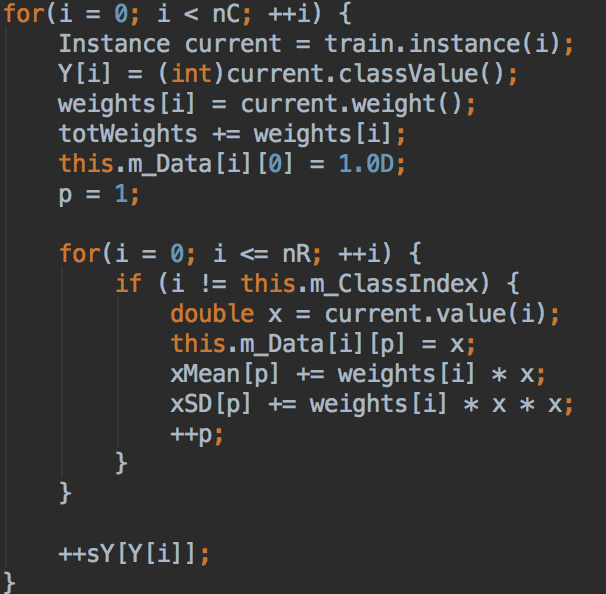
sY：分类的类别；

weights[i]：第i个特征的权重；

m\_Data：属性值



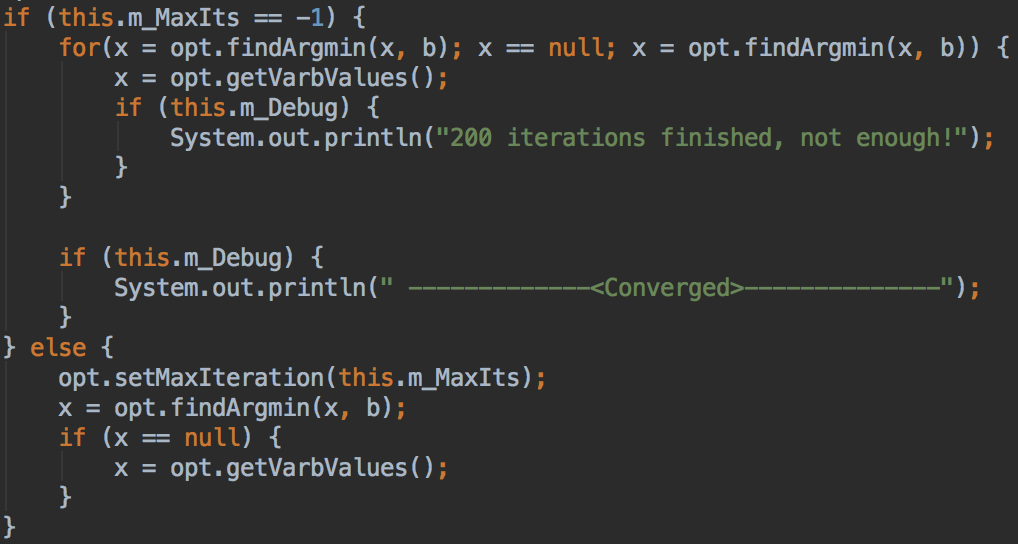
1. 输入属性值、权重，计算属性均值、属性标准差、类别数，Y[i]记录每个样本的类别值



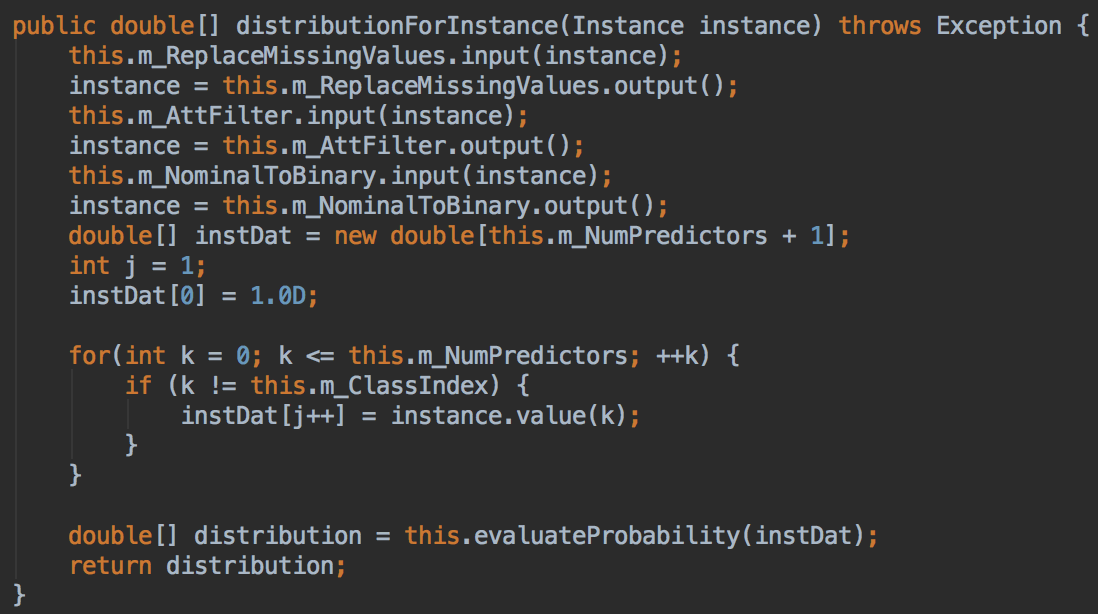
1. 计算xMean、xSD：



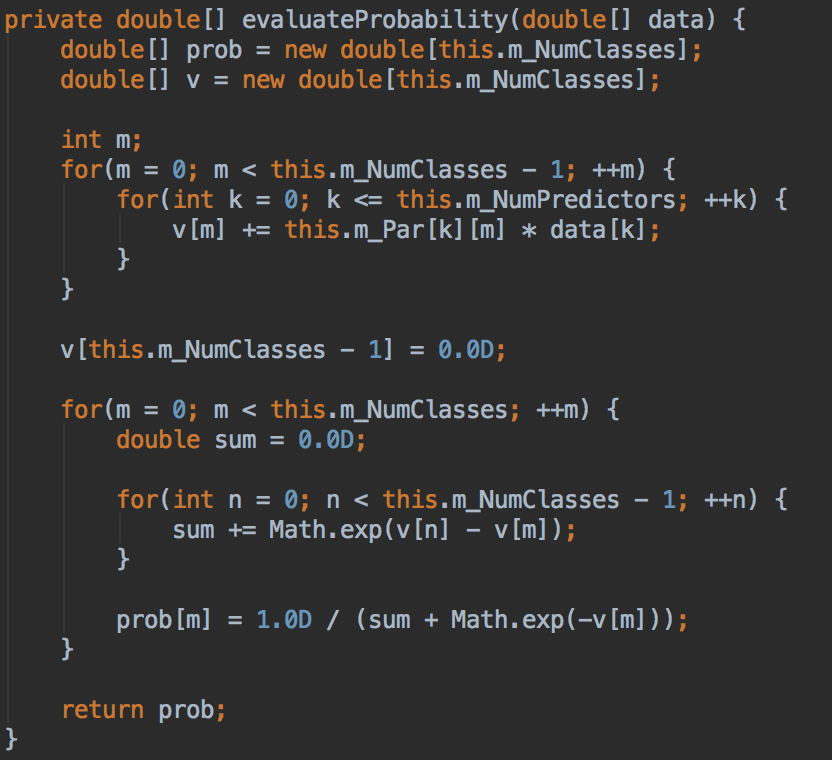
1. 优化：m\_MaxIts是优化迭代的次数，优化方法为findArgmin



1. 样本分类函数**distributionForInstance**对输入样本进行分类：



1. 计算概率：



## 决策树C4.5算法

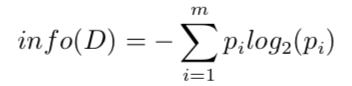
### 算法介绍

决策树是一个树结构，每个非叶节点表示一个特征属性上的测试，每个分支表示这个特征属性在某个值域上的输出，每个叶节点存放一个类别。决策树的决策过程大致是：从根结点开始，测试待分类项中相应的特征属性，按照其值选择输出分支直到到达叶节点，将叶节点中存放的类别作为决策结果。

决策树中最为关键的是属性选择度量，即一种选择分裂准则，将给定的类标记和训练集合数据最好的划分为个体类。本实验介绍C4.5算法。

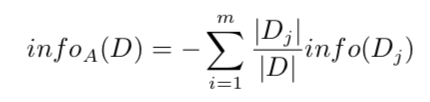
### 算法过程

1. D的信息熵为：



其中，D是一个用类别对训练集进行的划分结果，pi是第i类在整个训练集中出现的概率

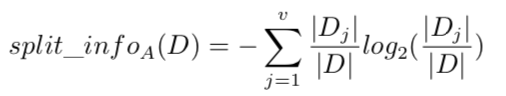
训练集D按属性A进行划分得到的信息熵为：



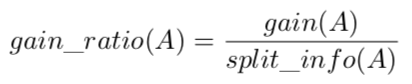
1. 信息增益：

../../屏幕快照%202018-05-01%2014.34.06.png

1. 定义“分裂信息”，训练集按属性A进行划分的分裂信息为：



1. 定义增益率：

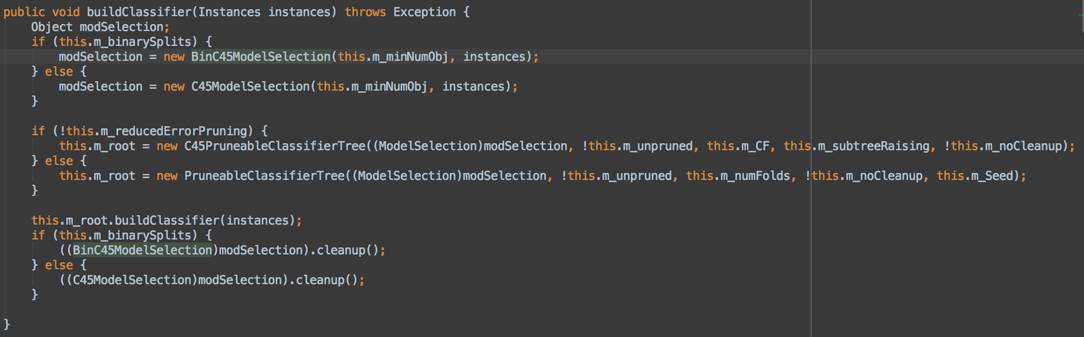


1. 选择增益率最大的属性作为分裂属性

### 算法源代码分析

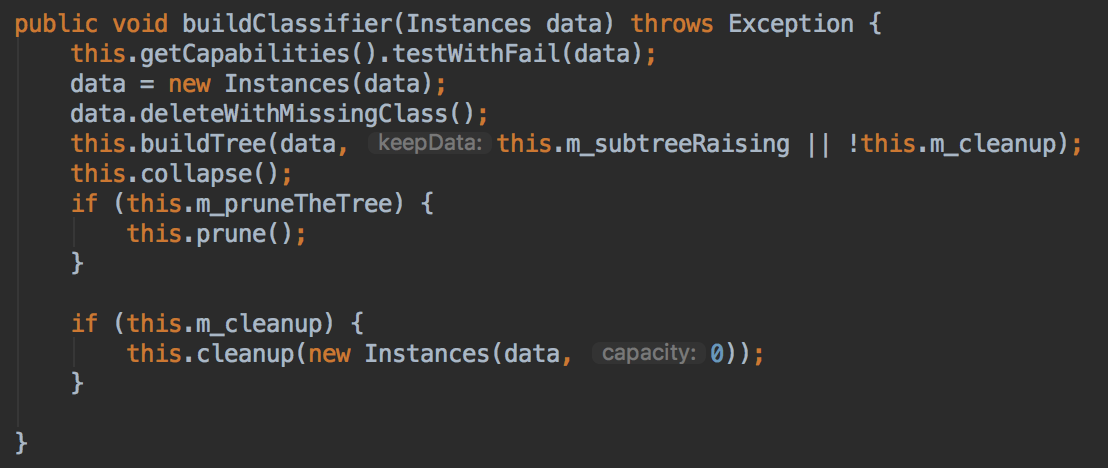
1. buildClassifer函数创建一个树结构：

首先判断是否为二叉树，再根据m\_reducedErrorPruning选择构建树的方法，modSelection用于选择分裂模型，m\_root是树的根结点



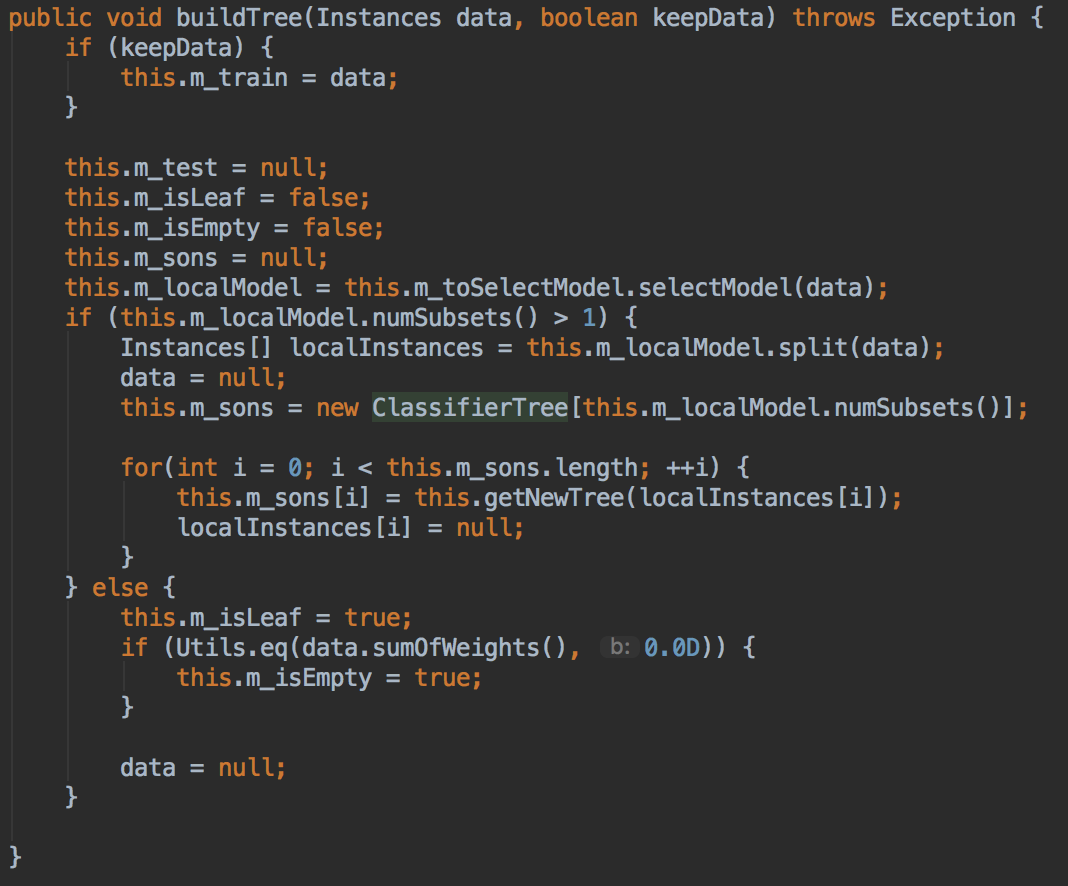
1. C45PruneableClassifierTree方法：

首先检测data是否能够分类，调用buildTree构建分类树，调用collapse进行树的塌缩，如果需要剪枝，则调用prune剪枝



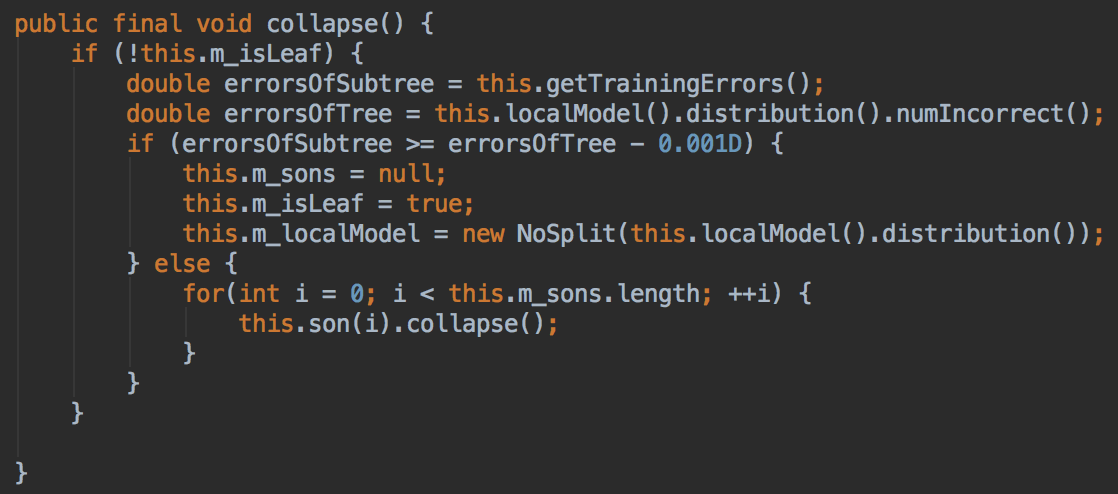
[1] buildTree函数：

根据m\_toSelectModel来选择一个模型把传入的数据集按相应的规则分成不同的子集，查看分裂子集的数量，若只有一个，则直接返回；否则根据localModel将传入的数据集分成不同的子特征，接着为每一个子特征建立新的ClassiferTree节点，并将其作为自己的子节点，再给子节点创建新树



[2] collapse函数：

若子节点的出错率较高，则将这些子节点删除



1. PruneableClassifierTree方法：

与C45PruneableClassifierTree不同的是，该方法在构建树的时候，还传入了测试集并且去除了collaspe步骤

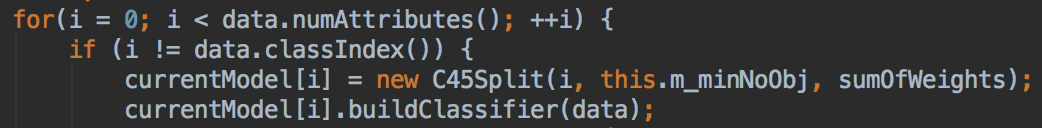


1. C45选择分裂模型：

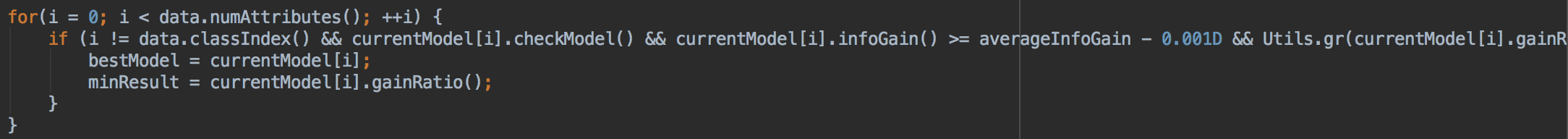
[1] 在public final ClassifierSplitModel selectModel(Instances data) 函数中，currentModel存储在每个属性上构建出的分裂模型

../../屏幕快照%202018-05-01%2015.16.57.png

对于每一个特征，构建模型：

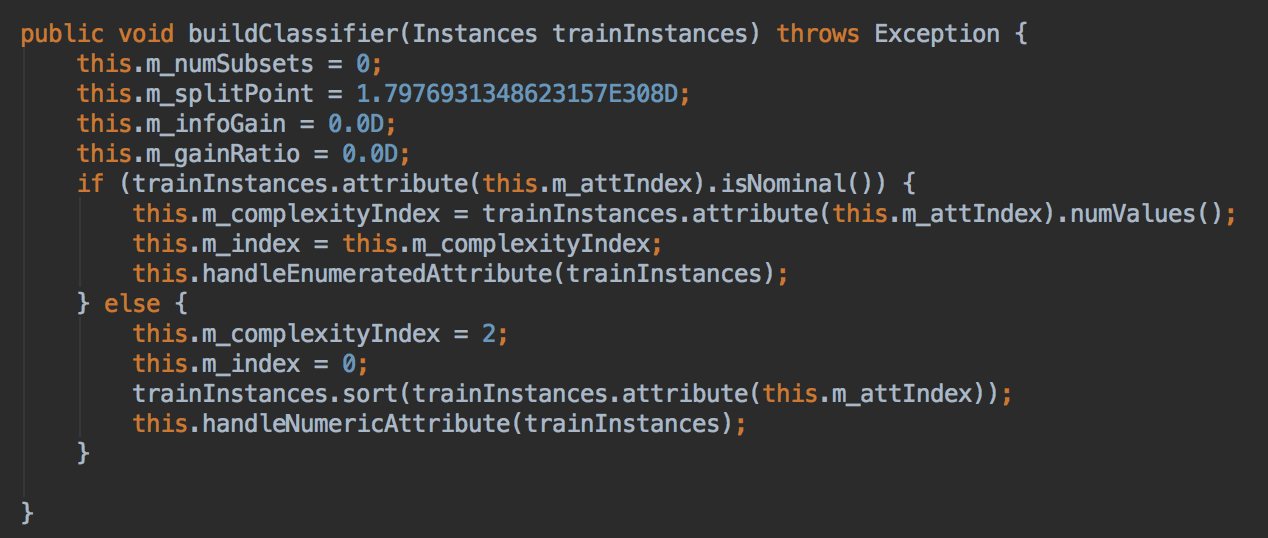


对所有的属性构建完分裂模型后，选择信息增益率最大的模型作为最优模型：

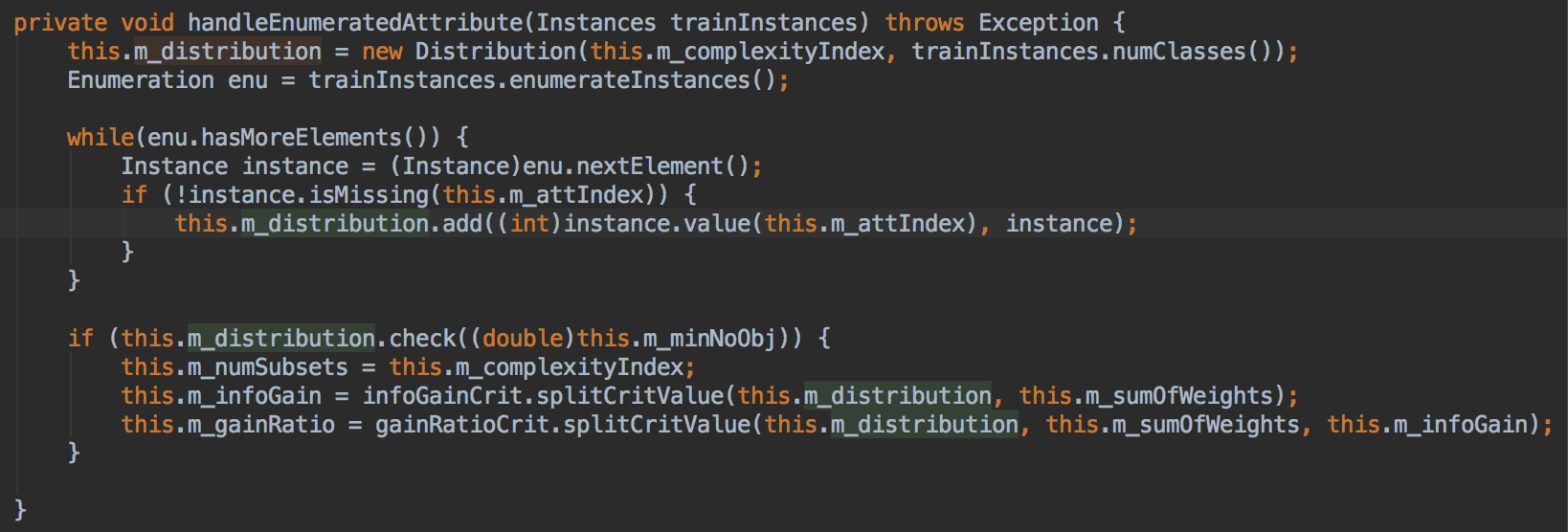


[2] 模型构建函数buildClassifier:

handleEnumeratedAttribute对枚举型进行分裂，handlerNumericAttribute对数值型进行分裂



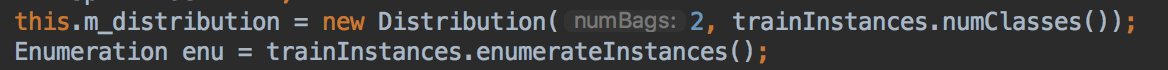
[3] 枚举型：handleEnumeratedAttribute函数

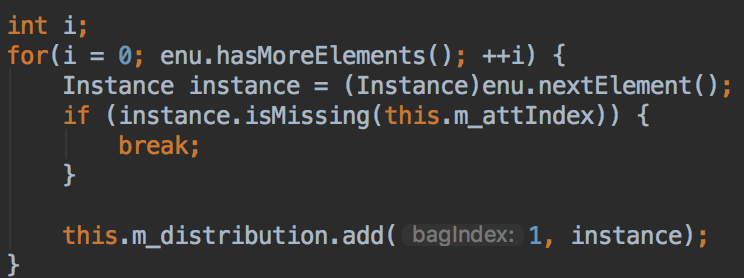


遍历所有的样本，若分裂属性不为空，则放入不同的bag，并检查分裂是否满足要求。若满足要求，则设置子集的数量，计算信息增益和信息增益率；否则子集数量为0，在buildClassifer函数中认定无效

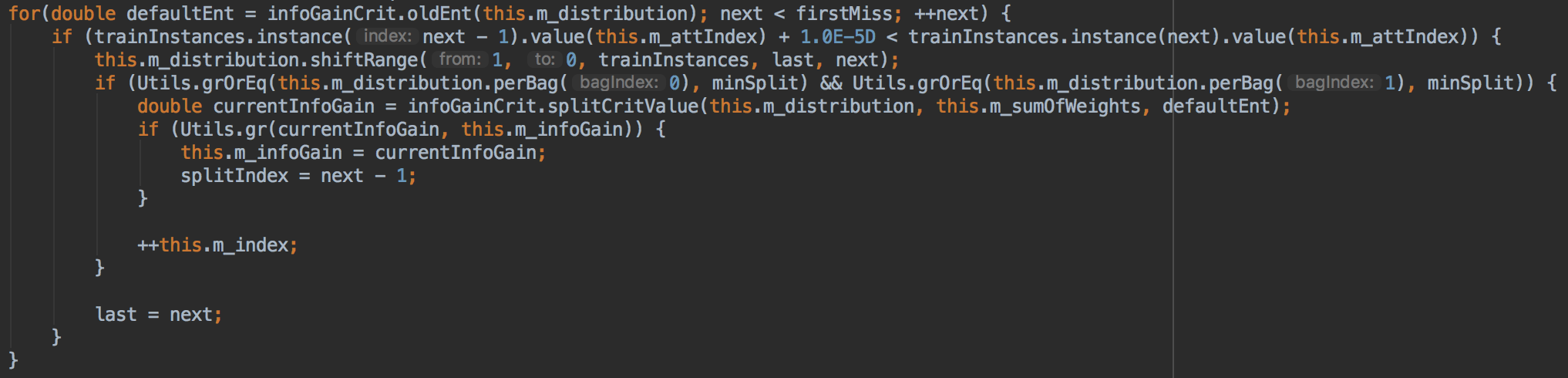
[4] 数值型：handlerNumericAttribute函数

新建分布，数值型默认为二维分布，有效的样本均放在bag1中

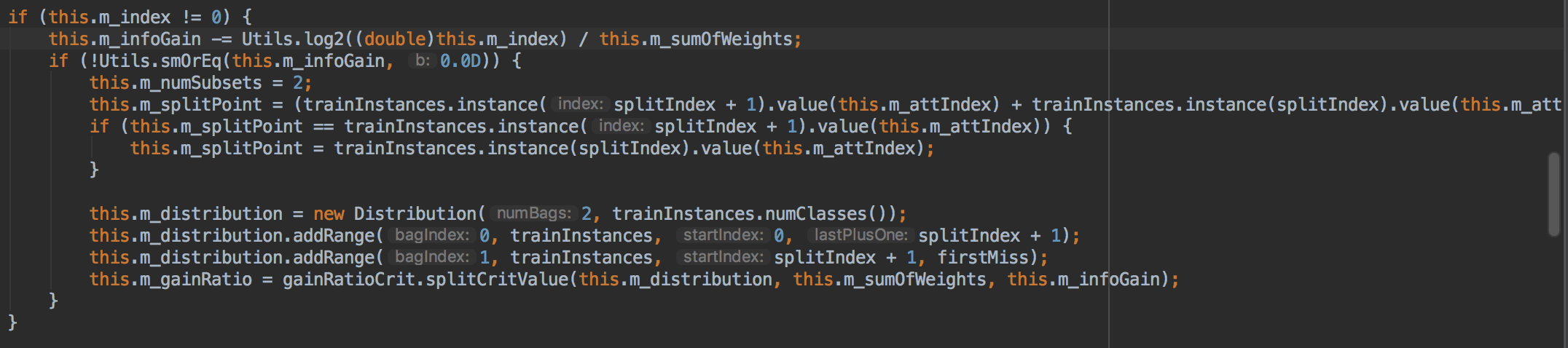




寻找合适的分裂点：



计算最大信息增益和信息增益率：



## 随机森林算法

### 算法介绍

随机森林是一种简单且有效的算法，基于Bagging的集成学习方法，常用来做分类、回归问题。核心思想为通过训练和组合不同的决策树，形成森林，最终的分类结果是个别树输出类别的众数。准确率高，训练速度快，抗噪能力好。

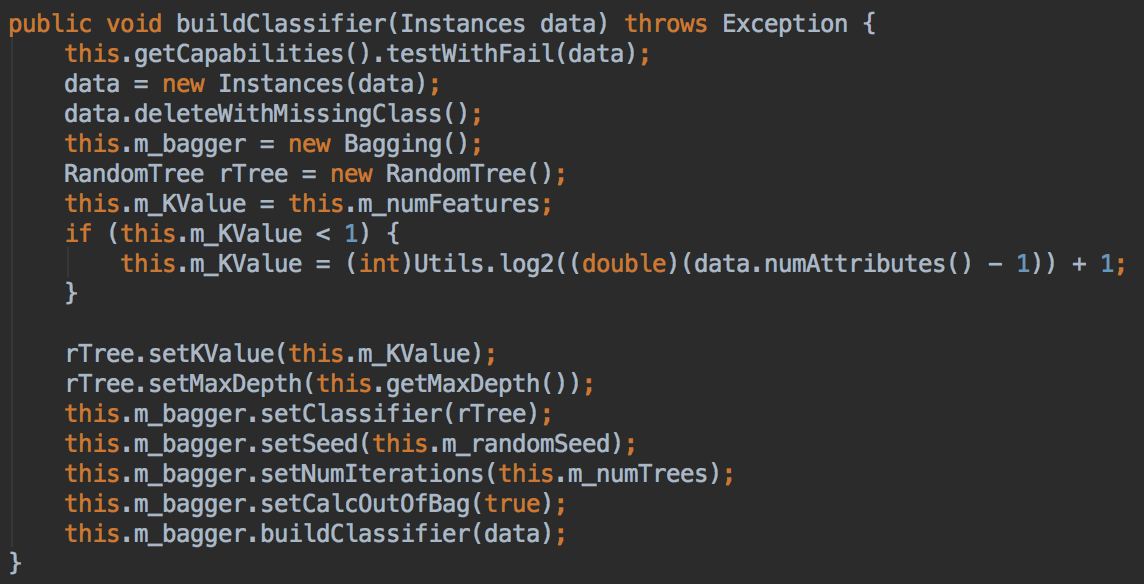
### 算法过程

1. 对训练集进行有放回的抽样N次，得到的子集作为新的训练集
2. 在新的训练集中随机抽出训练集的K个属性，训练一个决策树模型，不做剪枝操作
3. 重复上述过程M次，得到M个决策树模型，即M个分类器
4. 对于测试用例，使用M个分类器进行分类，最终的分类结果由这M个分类器投票决定。

### 算法源代码分析

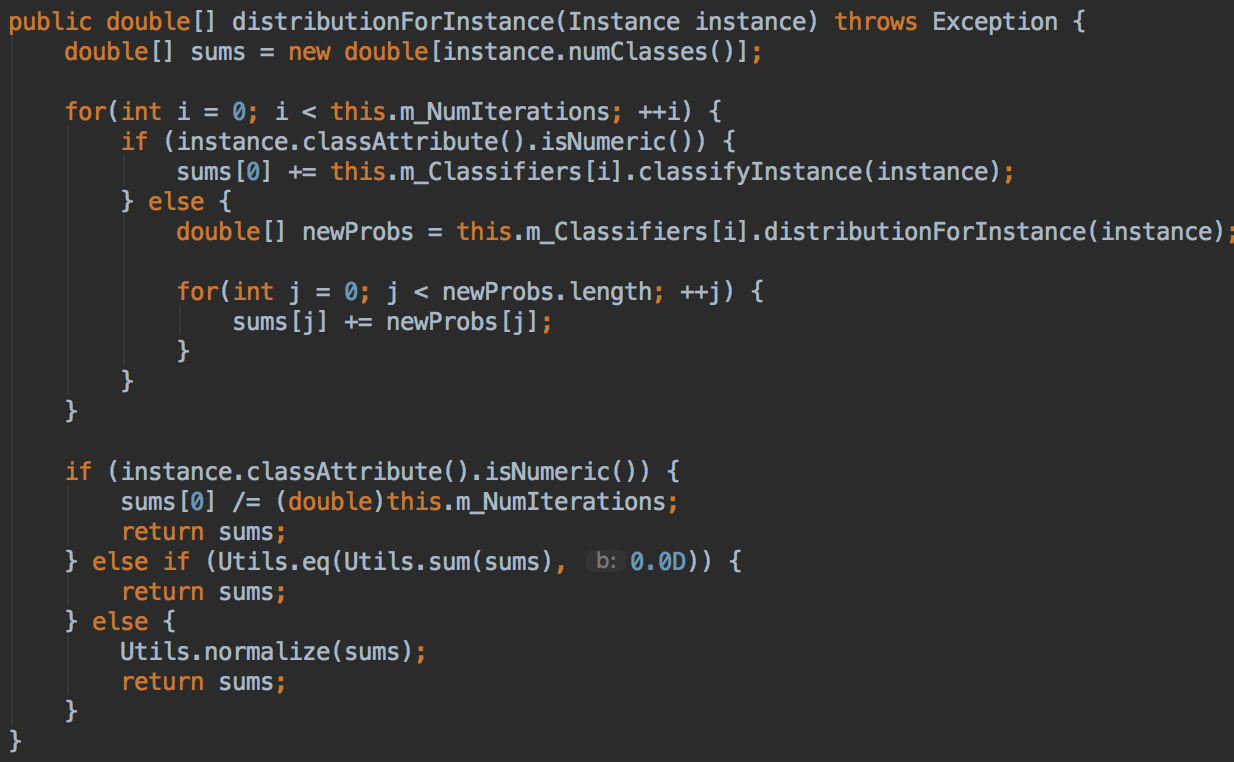
1. buildClassifier函数创建分类器：

首先去除无效数据，构建一个随机树，设置属性值，设置最大深度，并将该随机树传给Bag，调用bagging训练方法进行训练



1. 分类过程使用bagging的distributionInstance：

计算出概率的最大值并作为返回结果



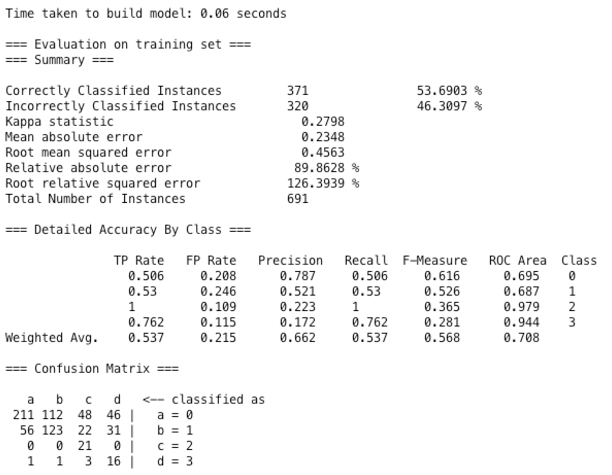
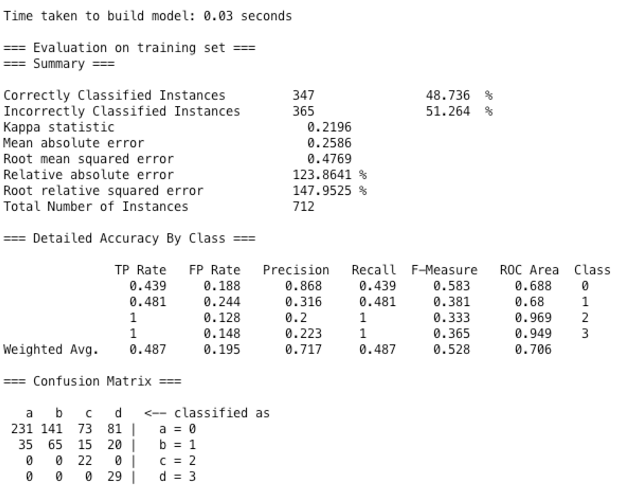
# 实验结果

分别在lv0-lv4关卡游戏胜利，将结果存储到AliensRecorder.arff文件中，在weka中分别调用上述4个算法查看结果

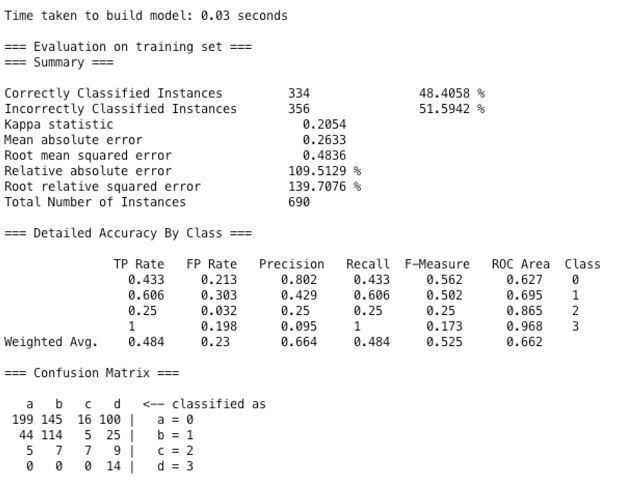
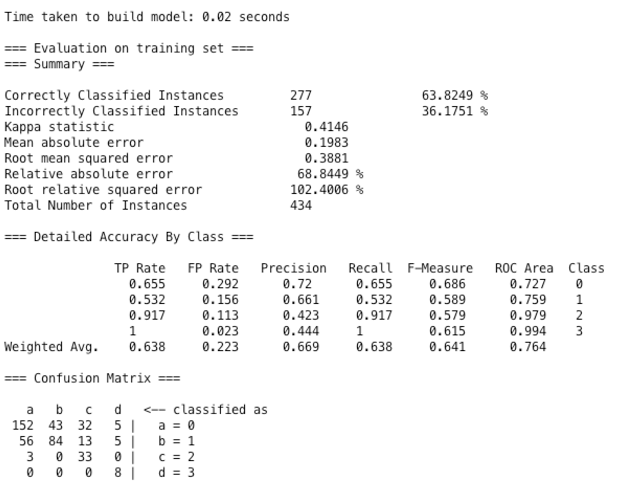
## 朴素贝叶斯算法

（1）weka运算结果

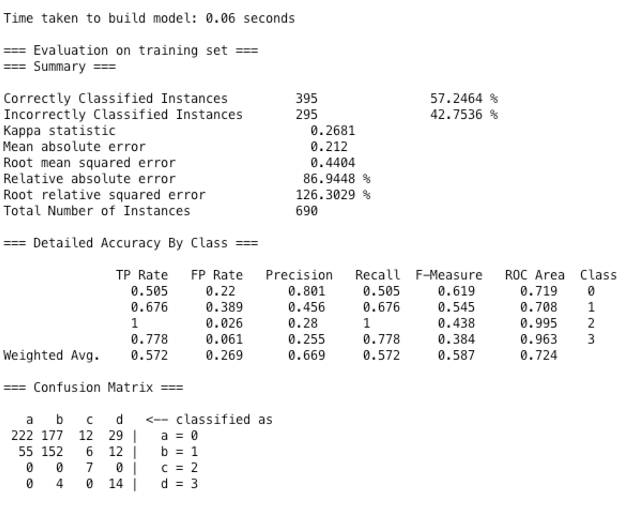
Lv0: Lv1:

Lv2: Lv3:

Lv4:



（2）结果统计

Lv0:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 40 | 853 |
| 2 | Win | 42 | 593 |
| 3 | Lose | 43 | 853 |
| 4 | Win | 42 | 521 |
| 5 | Win | 42 | 533 |
| Average | 60% | 42 | 671 |

Lv1:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 42 | 301 |
| 2 | Lose | 44 | 893 |
| 3 | Win | 42 | 301 |
| 4 | Win | 46 | 539 |
| 5 | Lose | 23 | 214 |
| Average | 60% | 42 | 671 |

Lv2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 44 | 511 |
| 2 | Lose | 40 | 457 |
| 3 | Lose | 18 | 244 |
| 4 | Lose | 42 | 937 |
| 5 | Win | 44 | 511 |
| Average | 40% | 38 | 532 |

Lv3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 43 | 646 |
| 2 | Lose | 13 | 232 |
| 3 | Win | 43 | 646 |
| 4 | Win | 43 | 519 |
| 5 | Win | 43 | 646 |
| Average | 80% | 38 | 538 |

Lv4:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 8 | 163 |
| 2 | Lose | 42 | 777 |
| 3 | Win | 49 | 695 |
| 4 | Lose | 29 | 295 |
| 5 | Win | 44 | 493 |
| Average | 40% | 34 | 485 |

## Logistic算法

Lv0:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 62 | 872 |
| 2 | Win | 56 | 713 |
| 3 | Lose | 57 | 812 |
| 4 | Lose | 56 | 926 |
| 5 | Lose | 53 | 671 |
| Average | 40% | 57 | 799 |

Lv1:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 52 | 798 |
| 2 | Lose | 30 | 535 |
| 3 | Lose | 10 | 163 |
| 4 | Lose | 20 | 226 |
| 5 | Lose | 43 | 989 |
| Average | 20% | 31 | 542 |

Lv2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 49 | 523 |
| 2 | Win | 47 | 634 |
| 3 | Win | 48 | 592 |
| 4 | Win | 49 | 523 |
| 5 | Win | 46 | 543 |
| Average | 80% | 48 | 563 |

Lv3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 13 | 238 |
| 2 | Lose | 36 | 889 |
| 3 | Lose | 40 | 761 |
| 4 | Lose | 24 | 352 |
| 5 | Lose | 9 | 163 |
| Average | 0 | 24 | 481 |

Lv4:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 31 | 277 |
| 2 | Lose | 29 | 352 |
| 3 | Lose | 41 | 686 |
| 4 | Win | 56 | 917 |
| 5 | Lose | 37 | 457 |
| Average | 20% | 38.8 | 598 |

## 决策树C4.5算法

Lv0:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 42 | 444 |
| 2 | Win | 52 | 573 |
| 3 | Win | 42 | 444 |
| 4 | Win | 42 | 444 |
| 5 | Win | 55 | 697 |
| Average | 100% | 47 | 520 |

Lv1:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 39 | 977 |
| 2 | Win | 42 | 301 |
| 3 | Win | 42 | 301 |
| 4 | Win | 42 | 401 |
| 5 | Lose | 17 | 690 |
| Average | 60% | 36 | 534 |

Lv2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 15 | 346 |
| 2 | Win | 45 | 567 |
| 3 | Win | 44 | 583 |
| 4 | Lose | 40 | 841 |
| 5 | Lose | 34 | 729 |
| Average | 40% | 34 | 613 |

Lv3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 44 | 524 |
| 2 | Win | 44 | 524 |
| 3 | Win | 44 | 524 |
| 4 | Lose | 25 | 283 |
| 5 | Lose | 27 | 325 |
| Average | 60% | 37 | 436 |

Lv4:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 48 | 567 |
| 2 | Lose | 45 | 997 |
| 3 | Lose | 43 | 901 |
| 4 | Lose | 15 | 193 |
| 5 | Lose | 45 | 853 |
| Average | 20% | 40 | 702 |

## 随机森林算法

Lv0:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 49 | 687 |
| 2 | Lose | 42 | 496 |
| 3 | Lose | 32 | 346 |
| 4 | Win | 51 | 514 |
| 5 | Win | 51 | 755 |
| Average | 60% | 45 | 560 |

Lv1:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 42 | 301 |
| 2 | Win | 42 | 301 |
| 3 | Win | 42 | 301 |
| 4 | Win | 42 | 301 |
| 5 | Lose | 19 | 175 |
| Average | 80% | 37 | 276 |

Lv2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 47 | 589 |
| 2 | Lose | 42 | 719 |
| 3 | Win | 44 | 464 |
| 4 | Win | 45 | 593 |
| 5 | Win | 47 | 528 |
| Average | 80% | 45 | 579 |

Lv3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Lose | 16 | 247 |
| 2 | Win | 45 | 787 |
| 3 | Win | 46 | 500 |
| 4 | Win | 45 | 787 |
| 5 | Lose | 16 | 250 |
| Average | 60% | 43 | 514 |

Lv4:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Result | Score | TimeStep |
| 1 | Win | 46 | 735 |
| 2 | Win | 46 | 735 |
| 3 | Win | 46 | 735 |
| 4 | Win | 46 | 735 |
| 5 | Win | 46 | 735 |
| Average | 100% | 46 | 735 |

本文形式地给出了“继承反常现象”的一种一般性定义.定义对每一种Subtyping关系都有其相对应的“渐增式继承”,使其更具有普适性.文中“渐增式继承”是一个相对的概念,这有助于对人们“使继承反常现象得到缓解”的努力得以分类和评价.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是北京大学计算机科学技术系袁崇义教授、屈婉玲教授领导的讨论班上的同学和老师表示感谢.

References:

1. Clifton C, Leavens GT, Chambers C, Millstein T. MultiJava: modular open classes and symmetric multiple dispatch for Java. ACM SIGPLAN Notices, 2000,35(10):130-145.
2. Wegner P, Zdonik SB. Inheritance as an incremental modification mechanism or what like is and isn’t like. In: Gjessing S, Nygaard K, eds. Proc. of the ECOOP’88. LNCS 322, Heidelberg: Springer-Verlag, 1988. 55-77.
3. Waxman BM. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications, 1988,6(9):1617-1622.
4. Yonezawa A. ABCL: An Object-Oriented Concurrent System. Cambridge: MIT Press, 1990.
5. Matsuoka S, Yonezawa A. Analysis of inheritance anomaly in object-oriented concurrent programming languages. In: Agha G, Wegner P, Yonezawa A, eds. Research Directions in Concurrent Object-Oriented Programming. Cambridge: MIT Press, 1993. 107-150.
6. Hemige V. Object-Oriented design of the groupware layer for the ecosystem information system [MS. Thesis]. University of Montana, 1995.
7. Rose A, Perez M, Clements P. Modechart toolset user’s guide. Technical Report, NML/MRL/5540-94-7427, Austin: University of Texas at Austin, 1994.
8. Keene SE. A Programmer’s Guide to Object-Oriented Programming in Common LISP. Boston: Addison-Wesley Longman Publishing Co., Inc., 1988.
9. Guo L, Tang ZS. Specification and verification of the triple-modular redundancy fault-tolerant system. Journal of Software, 2003,14(1):28~35 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/28.htm
10. Schutze, H. Dimensions of meaning. In: Whitelock P, ed. Proc. of the Supercomputing’92. Los Alamitos, 1992. 787~796. ftp://parcftp.parc.xerox.com/pub/qca/papers/
11. Wang XW. Research on quality-of-service management and group communication mechanisms in distributed multimedia systems [Ph.D. Thesis]. Shenyang: Northeastern University, 1998 (in Chinese with English abstract).

附中文参考文献:

[9] 郭亮,唐稚松.三机冗余容错系统的描述和验证.软件学报,2003,14(1):28-35. http://www.jos.org.cn/1000-9825/14/28.htm

[11] 王兴伟.分布式多媒体系统服务质量管理与组通信机制的研究[博士学位论文].沈阳:东北大学,1998.