数据挖掘
大作业最终报告

学　　院：计算机学院

专　　业：计算机应用技术

年　　级：2016 级

组　　号：第六组

成　　员：赵 颖 2620160012

王晓媛 2620160007

李昱燃 2620160009

用户行为预测——以某电信公司为例

摘要

本文讨论了一个关于用户行为预测的数据挖掘问题，在对某电信公司客户数据集的处理、特征选取之后，尝试使用随机森林（Random Forest）、与 AdaBoost 结合的决策树、梯度提升决策树（Gradient Boosting Decision Tree）三种算法进行分类，最终较好地实现了对客户忠诚度、购买欲、增值性三个属性的预测，并且对三者的结果进行了分析和比较。

关键词：特征选取、决策树、分类器

1　导论

1.1　问题描述

本文所尝试解决的问题是根据 2009 年 KDD Cup 竞赛的问题改编而来，目的是预测客户三个维度的行为，包括：1、忠诚度（Churn）：用户切换运营商的可能性；2、购买欲（Appetency）：购买新服务的可能性；3、增值性（Up-selling）：客户升级或追加购买高利润产品的可能性。我们需要根据现有数据训练出分类器，分别对客户的忠诚度、消费欲和增值服务倾向性做出二元判别，并计算各分类器的准确率，比较它们的分类效果。

竞赛中的数据集来自法国 Orange 电信公司的客户特征描述数据，我们使用的数据集则是从比赛数据集中随机抽取的 10000 个带标签的训练数据和 10000 个无标签的测试数据（为了减轻类分布的不平衡，保留了大多数正值的数据），均有 230 个特征变量，前 190 个特征变量是数值型变量，后 40 个是类别型变量。

1.2　相关工作

在 KDD Cup 2009 中，参赛者对该问题主要采取了以下方案：Miller[2]等人采用交叉验证方法进行特征选取，分类器采用流行的 Boosting+决策树的融合方法；Lo[5]等人则分别采用了多种分类方法，比如选择朴素贝叶斯分类器、与 AdaBoost 结合的决策树等；Xie[12]等人采用了封装类型的特征选取方法，分类器则是随机梯度提升树。总的来看[3]，特征选取的方法相对多样化，封装类、过滤类以及混合类都有；分类方法则比较类似，虽然也有采用其它方法甚至多种方法的，但决策树类的方法仍然是最受参赛者青睐的方法。

1.3　问题分析及文章结构

本文分三步解决这一问题：数据预处理与特征选择，分类算法的选择，分类算法的实现。

首先，由于数据集本身的特性，比如有大量缺失数据需要处理，或者类分布的不均衡，需要我们对数据集中的数据进行预处理，并且选择合适的特征变量，方便分类器的训练。之后，出于提高准确率、加快训练速度的目的，需要选择适当的分类算法。最后，实现了分类算法并进行了分类器的训练。

本文第二部分对给出的特征值进行处理，以构造便于进行分类器训练的数据集；第三部分选择合适的分类算法并实现分类；第四部分为不同条件下的实验结果；最后是总结与分析。

2 特征值处理

在构建分类器之前，首先要对原始数据进行有选择的加工，得到合适样本。

给定的训练集共 10000 条数据，每条数据分别包括 230 个特征值，特征值的前 190 个为数值类型，后 40 个为 string 字符串类型，特征值之间以逗号（,）分隔，无特征值时为空（””）。

从 230 个特征值中选择特征值出现次数最多，较为完整的 40 余特征值进行训练，其余数据由于完整度不够被删去。选择特征值的方法如下：

a) 使用 java 编写程序，从 train.txt 中按行读入数据，每行记为 lineTxt，由于数据之间严格采用“,”分隔，故可调用函数 split()，得到 tempArray 数组，其元素为字符串类型，其长度为 230，其中的值为空或为某一特征值。

b) 定义全局变量 sum 数组用于统计每个特征值的缺失次数，sum 为 int[230]类型，初始值皆为 0，对 tempArray 数组中的元素逐个进行判断，若 tempArray[i]值为空，则令 sum[i]加 1，否则不变。

c) 统计结束后，为了在获得其中缺失次数的信息的同时保存其所在列的信息，将数组复制到另一数组 s 中，对 s 进行排序，调用 Arrays.sort(s)函数并输出，对输出数据进行观察统计，可得到下图结果：

```
61:1150
62:1150
63:1458
64:1458
65:1458
66:2755
67:4467
```

图 2.1 数组 s 的标号与对应值

d) 由图中统计数字不难发现，在缺失次数超过 1500 次后，缺失次数过多，且急剧增长，不适合用于分类器的训练，因此仅考虑缺失次数不超过 1500 次的特征值所在列，在限定缺失次数不超过 1500 次的情况下对 sum 数组再次进行遍历，从而得到缺失次数较少的特征值所在列。结果如下（数字表示列号）：

[5,6,12,20,21,23,24,27,34,37,43,56,64,71,72,73,75,77,80,82,84,93,108,111,112,118,122,124,125,131,132,133,139,142,143,148,152,159,162,172,180]。

3 分类算法分析与选择

考虑到本问题中数据集的特性，参考了部分 KDD Cup 2009 参赛者的报告和分析，我们最终选择了决策树集合类的分类算法，主要进行实验的算法有：随机森林（Random Forest）、与 AdaBoost 结合的决策树、梯度提升决策树（Gradient Boosting Decision Tree）三种算法。

3.1 数据集特性分析

在使用的数据集中，训练数据和测试数据各 10000 个，数量比较大；而且该数据集中的 230 个特征变量并不是同一类型的，而是 190 个数值型变量和 40 个类别型变量的组合[3]；而且数据集中有大量数据缺失，例如：有 18 个完全无观测值的特征变量[4]。

3.2 参赛者选用的分类算法分析

参赛者使用最多的几种分类算法降序排列如下[3]：决策树集合类的分类算法（包括随机森林、与 AdaBoost 结合的决策树算法、梯度提升决策树等算法）、线性分类算法（特别是逻辑回归算法）、支持向量机。可见这几种算法对于该数据集的处理有一定优势。

3.3 决策树集合分类算法的优势

决策树集合分类算法，在本问题的处理过程中，有以下几个显著的优点[2],[3]：在大数据集上的表现良好，运行速度快；处理不同类型的变量比较容易；数据集中出现大比例的数据缺失时，也能够有效地估计缺失数据，保持较高的准确度。此外，这一类算法实现比较简单、鲁棒性好、准确度高、对分类不平衡等极端情况也能有效处理，这些因素也是选择随机森林算法的重要原因。

3.4 分类器的训练和保存

综合考虑所需时间、编写复杂度、匹配效果等因素，我们采用随机森林等三个算法对数据进行处理，由于在 python 语言的 scikit-learning 模块中，API 接口仅允许输入数字类型，我们应当将特征值的后 40 列转换为 int 类型，即统计每列 string 类型不同字符串的数量，分别用 1、2、3、……代替。但由于操作成本较高，时间上有一定限制，我们取消了该步骤，而简单的将后 40 列忽略，经测试，这样做得出的最终结果都在 0.8 以上，应当说是可以接受的。

在训练分类器之前，首先读入原始数据文件并把用逗号隔开的特征值保存在数组中，按特征值选取中选好的特征值将数组简化，再加入每行数据对应的标签，最后将数组中有空值的行删去，即完成了前期数据的处理，得到了用于训练分类器的训练集。

声明一个空的分类器，用训练数据及对应的标签进行训练，便得到了一个训练好的分类器。在本次任务中，我们把上述生成训练集的步骤整合成了一个函数 train()，该函数的调用需要输入的参数有两个，其一是原始数据文件，其二是标签文件，函数的返回值即为生成的一个分类器，只需要调用该函数，通过传入不同的参数，便可以生成不同的分类器。生成的分类器分别保存在名为 app.clf，chu.clf 和 ups.clf 的文件中，而后可以计算分类器在训练集上的准确率（train accuracy_score）。

用生成的分类器对测试数据进行分类，可以得到测试数据的标签，并计算测试集准确率（test accuracy_score）。

改变所构造树的参数：n_estimators（树木个数）和 max_depth（树木深度），再次计算 train accuracy_score 和 test accuracy_score。

4　实验结果

根据之前筛选得出的约 40 个类型特征值项，以及相关论文中作者们的经验，采用 AdaBoost 算法与决策树（Decision Tree）的结合、GradientBoosting 梯度提升算法以及随机森林（Random Forest）算法三种方式来分别构造分类器。实现分类器所用的语言为 Python，所用的工具包为 Scikit-learn，所搭建的环境为 Pydev for eclipse +Python2.7。

本次实验中，构造树的参数选定 n_estimators=10（即树木个数为 10），max_depth=3（树木深度为 3），其余为默认值。控制台输出结果依次为：购买欲（Appetency）、忠诚度（Churn）、增值性（Up-selling）的三个分类器的准确率。

1)　AdaBoost 算法与决策树分类器：

```
clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=3),
                         learning_rate=1.0,
                         n_estimators=10,
                         )
```

```
train accuracy_score:
0.963345379453
0.828600929272
0.815952503872
test accuracy_score:
0.9519
0.8061
0.8019
```

图 4.1　构造 AdaBoost 分类器参数设定及结果

2)　Gradient Boosting 梯度提升算法分类器：

```
clf = GradientBoostingClassifier(n_estimators=10,
                                 learning_rate=1.0,
                                 max_depth=3,
                                 random_state=0)
```

```
train accuracy_score:
0.969024264326
0.833505420754
0.822405782137
test accuracy_score:
0.9093
0.7232
0.7972
```

图 4.2 构造 Gradient Boosting 分类器参数设定及结果

3)　Random Forest 随机森林分类器：

```
clf = RandomForestClassifier(max_depth=3,
                             n_estimators=10,
                             max_features='auto',
                             )
```

```
train accuracy_score:
0.961280330408
0.821631388745
0.807176045431
test accuracy_score:
0.9544
0.8151
0.8214
```

图 4.3 构造 Random Forest 分类器参数设定及结果

表 4.1 n_estimators=10，不同 max_depth 对三种分类器的影响

max_depth	1	2	3	4	5	None
ABC	0.9544 0.8154 0.8205	0.9526 0.8118 0.8095	0.9515 0.8061 0.802	0.944 0.7927 0.8042	0.9438 0.7756 0.7752	0.9118 0.6337 0.6912
GBC	0.9544 0.8149 0.8197	0.9504 0.8122 0.8113	0.9093 0.7232 0.7972	0.9334 0.7141 0.7871	0.9152 0.7738 0.7617	0.9106 0.6353 0.6991
RFC	0.9544 0.8154 0.8215	0.9544 0.8154 0.8215	0.9544 0.8154 0.8213	0.9544 0.8154 0.8214	0.9544 0.8154 0.8213	0.9542 0.7638 0.7967

（max_depth=None 代表的是不设深度限制，可以达到最大树深）

表 4.2max_depth=3，不同 n_estimators 对三种分类器的影响

n_estimators	5	10	15	20	50	100
ABC	0.9542 0.8109 0.8181	0.9516 0.8061 0.8091	0.9473 0.7978 0.7973	0.9441 0.7911 0.7895	0.9421 0.7562 0.7599	0.9463 0.7398 0.7421
GBC	0.9465 0.8077 0.8082	0.9093 0.7232 0.7972	0.9067 0.7907 0.7901	0.8977 0.7837 0.7876	0.8933 0.7632 0.7614	0.8984 0.7463 0.7512
RFC	0.9544 0.8155 0.8216	0.9544 0.8154 0.8214	0.9544 0.8154 0.8215	0.9544 0.8154 0.8213	0.9544 0.8154 0.8215	0.9544 0.8154 0.8215

5 总结

5.1 实验结果分析

由上述实验结果可以看到，max_depth 与 n_estimators 对 Random Forest 随机森林分类器基本没有影响，于是我们去查阅了 Random Forest 算法的相关资料，发现其算法中并没有剪枝这一步，而且树的数量是随机的，因此实验中设定的值并没有起到作用。

而另外的 AdaBoost 分类器与 Gradient Boosting 分类器基本上随着树木深度增加，对于训练集数据的准确率提升，甚至可以趋近于 1.0，但是同时对测试集的准确率就开始下降；对于树木数量也是相似的情况。但总体来说，此次试验中，AdaBoost 分类器效果稍好于 Gradient Boosting 分类器，而且在选定树深为 3，树数量为 10 时，可以使得训练集与测试集的数据拟合准确率最接近。

根据以上分析得到的简单结论是，选用树深为 3，树数量为 10 的 AdaBoost 分类器，或者使用简单的随机森林分类器，效果都比较好。

5.2 项目过程总结

该实验的整体设计与实现是我们同一小组三名同学倾力合作的成果。我们对其进行了多次讨论，从一开始实验内容的选择确定，到查找文献、选取前辈们成功的算法，再到分工实现、每个人负责一个环节，到整合成功运行并测试，最后形成目前这篇并不算出众但却凝聚着心血的论文，我们三人通力合作，明确分工，才有了最终的结果。

在整个实验中，李昱燃同学设计了特征值处理的 java 程序，实现了特征值的过滤和筛选，从 10000 行数据、230 个特征值中成功选择了缺失率低、适用性好的特征值，从而为实验的成功进行打下了基础；王晓媛同学以丰富的智慧和学习能力对问题进行了分解和剖析，参考了多篇文献之后带领我们选择了最终的三种算法；赵颖同学设计了代码的框架，实现了三种不同分类器的生成和训练，并进行了多参数的调试以取得最优解。

由于我们三人的水平有限，能够进行学习和咨询的资源、时间也有一定限制，本次实验的设计可能并不完美，也确实存在很多问题，在之后我们会进行改正和提高。实验虽然告一段落，但数据挖掘的学习之路还很长，相信我们会一步步走下去，不断提高自己。

参考文献

[1] Friedman J, Hastie T, Tibshirani R. Additive Logistic Regression: a Statistical View of Boosting[J]. Annals of Statistics, 1998, 28(1):2000.

[2] Hugh Miller et al. Predicting customer behaviour: The University of Melbourne's KDD cup report. In JMLR W&CP, volume 7, KDD cup 2009, Paris, 2009.

[3] Guyon I, Lemaire V, Dror G. Analysis of the KDD cup 2009: Fast scoring on a large orange customer database[J]. Kdd Cup in Press, 2009, 11(2):2009.

[4] P. Doetsch, C. Buck, P. Golik, N. Hoppe, M. Kramp, J. Laudenberg, et al. Logistic model trees with AUC split criterion for the KDD Cup 2009 Small Challenge. JMLR: Workshop and Conference Proceedings, 7 (2009), pp. 77–88.

[5] Lo H Y, Chang K W, Chen S T, et al. An Ensemble of Three Classifiers for KDD Cup 2009: Expanded Linear Model, Heterogeneous Boosting, and Selective Naive Bayes[J]. Proceedings of KDD-Cup 2009 competition, 2009, 7 of jmlr workshop and conference proceedings:57-64.

[6] Hall P, Miller H. Using generalized correlation to effect variable selection in very high dimensional problems[J]. Journal of Computational & Graphical Statistics, 2009, 18(3):533-550.

[7] Breiman L. Random Forests[J]. Machine Learning, 2001, 45(1):5--32.

[8] Biau G, Devroye L, Lugosi G. Consistency of Random Forests and Other Averaging Classifiers.[J]. Journal of Machine Learning Research, 2008, 9(1):2015-2033.

[9] Nikulin V, Mclachlan G J. Classification of Imbalanced Marketing Data with Balanced Random Sets[J]. Journal of Machine Learning Research, 2009, 7:89-100.

[10] Jainjun Xie et al. A combination of boosting and bagging for KDD cup 2009 - fast scoring on a large database. In JMLR W&CP, volume 7, KDD cup 2009, Paris, 2009.

[11] Friedman J H. Greedy Function Approximation: A Gradient Boosting Machine[J]. Annals of Statistics, 2000, 29(5):1189--1232.