**华中科技大学**

**计算机科学与技术学院**

**《机器学习》结课报告**



专 业： 计算机科学与技术

班 级： 计卓2001

学 号： U202015293

姓 名： 杨释然

成 绩：

指导教师： 何琨

**完成日期： 2022年 7 月 5 日**

目录

[1. 实验题目: 智能家居使用场景识别 2](#_Toc107942309)

[2. 实验要求 2](#_Toc107942310)

[2.1 实验任务 2](#_Toc107942311)

[2.2 数据说明 2](#_Toc107942312)

[2.3 评测标准 3](#_Toc107942314)

[3. 算法设计 3](#_Toc107942315)

[3.1 数据处理 3](#_Toc107942316)

[3.2 模型算法 4](#_Toc107942317)

[3.3 模型融合 5](#_Toc107942318)

[4. 实验环境与平台 6](#_Toc107942319)

[5. 实现与分析 6](#_Toc107942320)

[6. 实验结果 8](#_Toc107942321)

[7. 个人体会 8](#_Toc107942322)

# 实验题目: 智能家居使用场景识别

为了公司更好发展，需要让终端用户在智能家居体验过程中能够更加强烈、清晰的感受到智能家居系统的优点。因而需要针对展厅类的场景做特殊的场景优化。为了实现这一场景下的优化，我们需要设计出一个系统，能够准确、快速的分析出当前智能家居产品使用环境是真实的家庭还是智能化体验的公共区域。

# 2. 实验要求

## 2.1 实验任务

设计出一个二分类的机器学习程序，能够准确、快速的根据智能家居产品使用环境的四类数据分析出当前智能家居产品使用环境是真实的家庭还是智能化体验的公共区域。

## 2.2 数据说明

如图2-1所示，比赛官方一共提供了4类数据：账号信息、设备列表、控制操作日志、设备上报日志。

## 

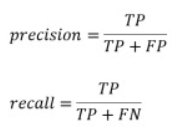
图2-1 数据集字段简要说明

数据集一共两个文件夹，一个是带有标签的训练集，一个是不带标签的测试集，我们把训练集按照8：2的比例分为训练集和验证集。

## 2.3 评测标准

1. 统计TP（正确预测环境场景），FP（错将家庭场景预测为公共区域），FN（错将公共区域预测为家庭记录）。

2. 通过第一步的统计值计算模型的precision和recall，计算公式如下：



3. 通过第二步计算结果计算F1-score,得到最后评测结果，计算方式如下：



# 3. 算法设计

## 3.1 数据处理

对于训练集和测试集的数据处理是完全一致的。

首先，对训练集的数据进行分析，从cus.csv中我们可以发现数据是非常不平衡的，0标签比较多，1标签比较少，0占比93.5%，1占比6.5%，所以后面就要用smote算法来解决数据类别不平衡的问题。我首先用了pandas.read\_csv发现会报错，数据中有异常的不规整的地方，于是我写了一个返回pandas.DataFrame的函数用于读入数据，主要就是对于异常的数据进行处理使得数据规整。

读入完成后就是对四个文件的数据分别进行处理，其中cus.csv读入后不进行处理。对于devUpdata.csv读入后的数据，统计按uid分组后的did，data的唯一值的个数。对于control.csv统计按uid分组后的did，form，data的唯一值的个数。对于devList.csv统计按uid分组后的did，type，area的唯一值的个数、area的唯一值、area的数量。用merge函数将以上三个数据按uid合并，得到特征数据。感觉area对分类的作用比较大，于是我使用了TfidfVectorizer得到areaunique的词频矩阵，最终将特征数据和词频矩阵合并得到处理好的数据。

（tfdi的主要思想：如果一个词语或短语在一篇文章中出现的概率高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。tf-idf作用：用于评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度，我们把这个用于提取area的特征。）

最后，调用了SMOTE对象的fit\_resample解决不平衡问题。再按照8：2的比例划分了训练集和验证集。

## 3.2 模型算法

以下是我们对采用的模型和算法的说明。

1. 随机森林：

随机森林即由多个决策树组成，每个决策树并不相同，在构建决策树时，我们从训练数据中有放回的随机选取一部分样本，并且也不会使用数据的全部特征，而是随机选取部分特征进行训练。每棵树使用的样本和特征都不相同，训练出的结果也不相同。在机器学习中，随机森林是一个包含多个决策树的分类器，并且其输出的类别是由个别树输出的类别的众数而定。

随机森林 = Bagging + 决策树

随机森林构造过程中的关键步骤(M表示特征数目)：（1）一次随机选出一个样本，有放回的抽样，重复N次(有可能出现重复的样本)。（2）随机去选出m个特征, m <<M，建立决策树

2. 极端随机森林Extra Tree

ExtRa Trees是Extremely Randomized Trees的缩写，意思就是极度随机树，它是随机森林的一个变种：

对于每个决策树的训练集，随机森林采用的是随机采样bootstrap来选择采样集作为每个决策树的训练集，而extra trees一般不采用随机采样，即每个决策树采用原始训练集。

在选定了划分特征后，随机森林的决策树会基于基尼系数，均方差之类的原则，选择一个最优的特征值划分点，这和传统的决策树相同。但是extra trees比较的激进，他会随机的选择一个特征值来划分决策树。

从第二点可以看出，由于随机选择了特征值的划分点位，而不是最优点位，这样会导致生成的决策树的规模一般会大于RF所生成的决策树。也就是说，模型的方差相对于RF进一步减少，但是偏差相对于RF进一步增大。在某些时候，extra trees的泛化能力比RF更好。

3. 梯度提升决策树

梯度提升树GBDT (Gradient Boosting Decision Tree) 。是一种提升树。提升树通过多轮迭代，每轮迭代产生一个弱分类器，每个分类器在上一轮分类器的残差基础上进行训练。

GBDT与提升树的区别就是GBDT采用的损失函数可以有很多形式，采用负梯度求解残差。

## 3.3 模型融合

我们将多个不同的个体学习器 (Individual Leaner)进行融合提高模型效果。

对于个体学习器来说，它们的不同体现在（不同训练数据：数据集使用比例、预处理方法 ）、不同模型结构（RF、XGBoost、LightGBM、CatBoost、CNN、LSTM等）、不同超参（随机种子数、权重初始化、收敛相关参数、损失函数、子采样比例等）。对于融合方法我们采用了单层Stacking。Stacking的思路是基于原始数据，训练出多个基学习器，然后将基学习器的预测结果组合成新的训练集，去训练一个新的学习器。

单层Stacking是指在基学习器上只堆叠一层元学习器，基学习器可以是同质或异质的模型，而元学习器在传统做法中是选用逻辑回归模型，当然也能使用非线性模型作为元学习器，例如GBDT, KNN, NN, RF等。而在本次比赛中我们模型选用了超参不同的随机森林、极端随机森林和梯度提升树来作为基学习器，采用一个梯度提升树作为元学习器。

本项目一共使用十种（包括参数的不同）基学习器融合。

1. RandomForestClassifier(n\_estimators=100, n\_jobs=-1, criterion='gini')
2. RandomForestClassifier(n\_estimators=200, n\_jobs=-1, criterion='entropy')
3. ExtraTreesClassifier(n\_estimators=100, n\_jobs=-1, criterion='gini')

4.ExtraTreesClassifier(n\_estimators=200, n\_jobs=-1, criterion='entropy'),

5.GradientBoostingClassifier(learning\_rate=0.05, subsample=0.5, max\_depth=6, n\_estimators=5)

6.RandomForestClassifier(n\_estimators=300, n\_jobs=-1, criterion='gini')

7.RandomForestClassifier(n\_estimators=400, n\_jobs=-1, criterion='entropy')

8.ExtraTreesClassifier(n\_estimators=300, n\_jobs=-1, criterion='gini')

9.ExtraTreesClassifier(n\_estimators=400, n\_jobs=-1, criterion='entropy'),

10.GradientBoostingClassifier(learning\_rate=0.05, subsample=0.5, max\_depth=6, n\_estimators=5)

# 4. 实验环境与平台

操作系统：Microsoft Windows 10。

处理器：Intel(R) Core(TM) i7-7700HQ CPU。

开发环境：anaconda 4.12.0 和jupyter notebook

# 5. 实现与分析

实验开始我们的思路很简单，直接用pandas的read\_csv读取数据，然后将数据one-hot编码后用xgboost来进行训练。其中，数据处理将所有缺失值全部删除，忽略输入函数的报错。如图5-1和图5-2是我们一开始尝试方案的核心代码。（这部分代码没有放在最终提交的文件中）

文本

描述已自动生成

图5-1 初始思路训练代码

图形用户界面, 文本, 应用程序

描述已自动生成

图5-2 初始思路数据处理代码

但是实验的效果并不好，我们的F1-score只有0.48。在查阅资料和请教一些比较擅长机器学习的人后，我们从两个方向进行了改进，在数据处理方面一是自己编写了输入函数来进行输入保障了数据的完整性，二是采用tf-idf对特征重要性分析来提高预测的有效性；

而在模型方面我们考虑主要从模型上来改进并且采用模型融合的方法，首先用三种不一样的算法获得五个基学习器分别是两个RandomForestClassifier、两个ExtraTreesClassifier和一个GradientBoostingClassifier。在模型融合上，我们采用了Stacking的方法，把每一个基学习器得出的预测结果合成一个预测结果，建立新的梯度提升决策树模型，把五个基学习器循环预测完后合成的结果作为训练集去训练新的模型，标签为开始的标签, 使用新的模型对测试集进行预测，最终输出测试集的预测结果。

在这次的优化完成后，我们的F1-score成功达到了0.66。后面我们开始了调参的不断尝试过程，最终我们使用了十个基分类器来融合，并且增加了最终模型的训练深度，对于学习率也提升了一些，F1-score达到了最终的0.70588。

我们的分工：

我们一起想到用随机森林去完成这个题目。

韩磊负责一开始的数据处理包括输入函数的编写、各个文件数据的聚合、特征提取，提出并实现用tf-idf来更好的提取数据特征。

庞彤针对性地解决了样本不平衡问题，实现了用SMOTE算法实现了样本平衡性，并划分了训练集和验证集。

杨释然提出了模型优化的思路，学习并实现了stacking的模型融合方法，提出了加入ExtraTreesClassifier。

韩磊和杨释然在项目后期共同调试参数，实现了用十个基学习器去模型融合，并把F1-score提升到了最后的分数。

# 6. 实验结果



图6-1 团队队员组队截图



图6-2 最终提交结果

如图6-2所示，我们最终的F1-score为0.70588。

# 7. 个人体会

通过这次实验完整的进行了一次kaggle型比赛，了解并熟悉了比赛的流程，摆脱了最开始的陌生畏难心理，开始动手尝试之后发现也并不是很难搞定。

在过程中遇到的第一个问题就是数据的读取，因为数据并不规整导致直接用pd.read\_csv读取会产生报错，一开始查阅网络得到了一种解决方法是设置一些参数来忽略掉报错信息并将报错的行跳过，后来，想到这些数据有可能也有影响，所以我写了一个自己的数据读取来完成。

之后遇到的问题就是如何提高成绩（大哭）。一开始我的模型只有0.48左右，在引入了模型融合之后显著的上升了。但之后就一直卡在0.65左右不能提升，后来查阅资料发现应该调整弱分类器和梯度提升的层数的关系很有作用，最后终于提高到了0.7，这个成绩我也比较满意了。