**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： CS2002

学 号： U202015324

姓 名： 屈绍博

成 绩：

指导教师： 何琨

**完成日期： 2022年 7月 6日**

目录

[1. 实验题目: 智能家居使用场景识别挑战赛 2](#_Toc108019991)

[2. 实验要求 2](#_Toc108019992)

[2.1 实验任务 2](#_Toc108019993)

[2.2 数据说明 2](#_Toc108019994)

[2.3 评测标准 2](#_Toc108019995)

[3. 算法设计 3](#_Toc108019996)

[3.1 数据处理 3](#_Toc108019997)

[3.2 模型算法 3](#_Toc108019998)

[3.3 模型融合 3](#_Toc108019999)

[4. 实验环境与平台 3](#_Toc108020000)

[5. 实现与分析 4](#_Toc108020001)

[6. 实验结果 9](#_Toc108020002)

[7. 个人体会 9](#_Toc108020003)

# 实验题目: 智能家居使用场景识别挑战赛

http://challenge.xfyun.cn/topic/info?type=smart-home-2022

# 2. 实验要求

## 2.1 实验任务

根据公开数据集，准确、快速的分析出所给的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，以此来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，系统会针对展厅类的场景做特殊的场景优化。

## 2.2 数据说明

赛事举办方提供了四类数据，账号信息、设备列表、控制操作日志、设备上报日志。其中账号基础数据的训练集被打上了使用场景是家庭用户还是体验厅的标签。具体字段说明如下：

**表1.1 数据说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类别 | 变量 | 数据格式 | 解释 |
| 账号信息cus.csv | uid | string | 账号ID，唯一标识一个账号 |
| label | int | 使用场景 |
| 设备列表  devList.csv | uid | string | 账号ID，唯一标识一个账号 |
| did | string | 设备id |
| type | string | 设备型号 |
| area | string | 设备房间标签 |
| 控制操作日志  control.csv | uid | string | 账号ID，唯一标识一个账号 |
| did | string | 控制的设备id |
| time | bigint | 控制时间 |
| form | string | 控制方式 |
| data | string | 从远程对设备下发的控制日志 |
| 设备上报日志  devUpdata.csv | uid | string | 账号ID，唯一标识一个账号 |
| did | string | 上报的设备id |
| time | bigint | 上报时间 |
| data | string | 设备上报的状态日志 |

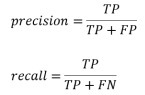
数据集会分为两个部分，第一部分数据有使用场景标签，用于识别算法训练，第二部分不包含场景标签，数据用于预测。通过对数据集数据量的观察，我们可以得到一个大致的估计，train和test数据量比例在3：1左右。

## 2.3 评测标准

依据结果文件，采用F1-score进行评价。

（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

利用pandas自带函数以及自己编写的数据读取函数，将训练以及测试数据集中的数据尽可能全面地读取进pandas的DataFrame表中，并通过一些手段比如将字符串转化为不同字符的数量的方法来将数据集中的数据转化为计算机可以处理的格式。

将所有文件中的数据读入之后，我们还要通过对数据进行整理归纳，不仅要通过分析题义以及测试的方法去除干扰数据，还要将训练和预测数据集的不同文件中的数据汇总到同一张表中，可以通过uid将数据对齐，将所有文件中的数据整合到一张表中，形成训练和预测两张feature表方便后续的训练和预测。

## 3.2 模型算法

主要训练了决策树相关模型对预测数据进行分类。

运用了三个模型算法：随机森林，XGBoost以及LightGBM。

## 3.3 模型融合

利用Linear Blending融合方式对模型进行线性加权融合，取上述三个模型的分类结果并赋予权重，形成最终的预测结果。

# 4. 实验环境与平台

硬件信息：

设备名称 LAPTOP-AR2NOGT7

处理器 Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz 1.61 GHz

图形处理器 NVIDIA GeForce MX350

机带RAM 16.0 GB (15.8 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

软件信息：

操作系统版本 Windows 11 家庭中文版

操作系统版本 22000.739

Python版本 Python 3.9.0 [MSC v.1927 64 bit (AMD64)] on win32

Jupyter版本 v2022.7.1001881029

模型的训练和预测均基于本地计算机CPU进行。

# 5. 实现与分析

本实验由个人完成。

数据处理：

数据处理可能是整个实验中最为耗时的部分，但它也同样重要，一个好的数据处理不仅可以让数据表示更加直观，也能让后续的训练取得更好的效果。

首先，由于数据集中某些数据编码的某些问题，直接利用pandas读取时产生了报错，所以我利用python的readline手写了一个处理csv文件的函数robust\_readcsv()，来帮助我们逐行将数据集中的数据得到panda表格中，解决了这个问题，成功读取了全部数据集中的数据并存入了DataFrame表格中。具体代码如下：

def robust\_readcsv(path, sep=','):

    try:

        lines = codecs.open(path).readlines()

    except:

        lines = codecs.open(path, encoding='latin-1').readlines()

    header = lines[0].strip().split(sep)

    content = []

    for line in lines[1:]:

        line = line.strip()

        try:

            index = [i for i, x in enumerate(line) if x == ',']

            if len(index) == len(header) - 1:

                content.append(line.split(sep))

            else:

                line\_content = []

                index = [0] + index

                for idx in range(len(header)-1):

                    line\_content.append(line[index[idx]:index[idx+1]].strip(sep))

                line\_content.append(line[index[len(header)-1]:].strip(sep))

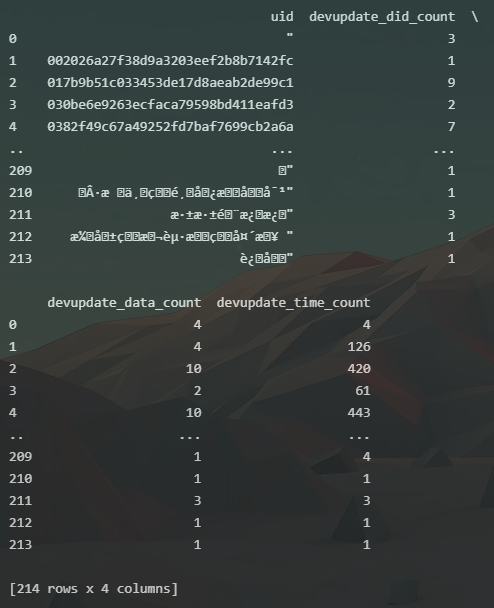
                content.append(line\_content)

        except:

            pass

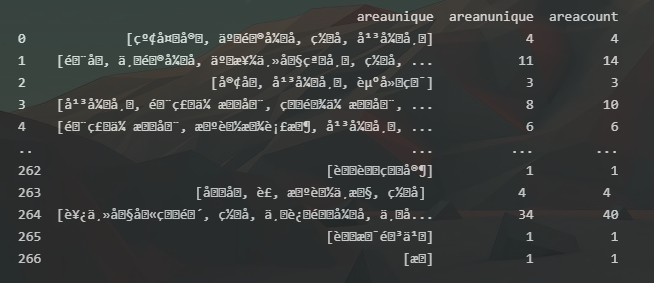
    return pd.DataFrame(content, columns=header)

然后我们开始对存储到表格中的数据进行处理，分别将train和test中的所有数据集按照标签‘uid’进行分组，并统计每个‘uid’的各种属性在数据中有多少种取值，将这些数据特征存储在一张新表中，命名为feat，方便后续进行数据分析。如图所示：



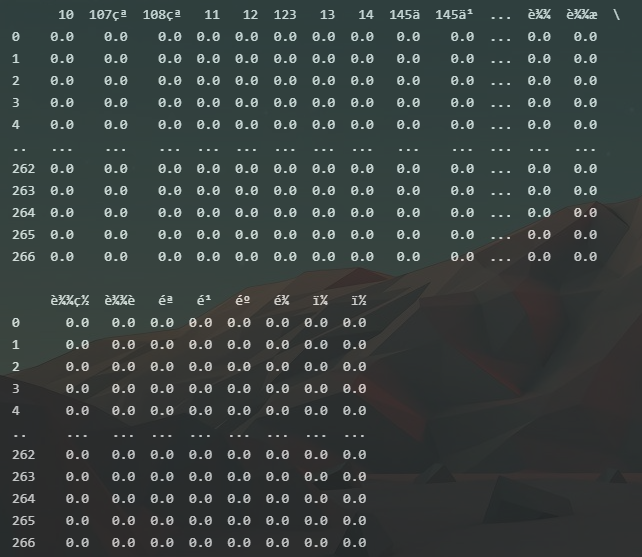
**图1 feat表中的数据**

需要特别说明的是，对于‘did’、‘data’、‘form’以及‘time’一类的数据我们采取统计不同取值的策略（nunique），而对于‘area’数据，由于其含有的数据量较大，我们则采用三个策略，分别统计所有取值（unique），不同取值的个数（nunique）以及总数（count），以此最大化利用已有数据。如图所示：



**图2 从devlist中的area属性中获取的数据**

由于我们得到的area所有的取值格式为一个string的列表，这个格式无法装载进入表格中进行处理，所以我们要对该项数据进行处理，我的处理方法是统计每个‘uid’的‘area’的每个token的字符的数量和。如图所示：



**图3 ‘area’中每个token的字符数量和的统计**

然后我们将上面得到的数据进行拼接，按照 ‘uid' 进行整合, 即同一个 uid 的数据才会合并到一起。这样即可以得到包含有数据中所有信息的feat表格，我们即可以利用train\_feat表格对模型进行训练，利用test\_feat表格进行预测。

用这组数据在基础的随机森林模型上进行初步训练之后，正确率达到了0.62左右，成绩并不是十分理想。

在深度分析数据之后，我认为如果将每个设备使用的时间，即“time”，作为特征的话 可能会影响模型的建模能力，因为每个设备使用的具体时间，直觉上不是固定的。而控制时间和日志上报时间也并非对应关系，所以我认为“time”是一组无用的特征信息，可能会影响决策树的构建，在从feat表中剔除了“time”特征量进行训练和预测后，在基础的随机森林模型上进行初步训练之后，正确率达到了0.68左右，比较符合预期。



**图4 去除“time”参数的随机森林模型预测结果得分**

算法模型的运用：

主要运用了决策树相关的模型：随机森林，XGBoost以及LightGBM。

随机森林部分：

我基于随机森林模型，利用刚刚得到的train\_feature表格，排除无关参数后，对label参数进行训练。

通过统计训练集中的正负样本，我们得到负样本一共有855条，正样本有60条，所以我们将正负样本的比重设置为0.93443：0.06557，这个比例可以在后续测试时进行微调。同时，我们还需要要调整n\_estimator数量在保证训练质量的同时，防止过拟合现象的出现。

训练过程中，我们需要统计不同的uid的对应的属性特点，此时，我们应该，让不同uid的属性去拟合该uid的label，又由于areaunique属性的格式为字符串，模型不方便处理，我们已经将其转化为其他格式存储在feature表格中，所以训练过程中需要将该属性剔除。

训练结束后对test\_feature表中的各个uid根据相关属性进行预测，并将结果存储在test\_cus表格中，方便后续进行模型融合。

代码如下：

# 随机森林模型代码

clf = RandomForestClassifier(n\_estimators=2500, class\_weight={0:0.89, 1:0.11}, bootstrap=False)

clf.fit(train\_feat.drop(['uid', 'label', 'areaunique'], axis=1), train\_feat['label'])

test\_cus['label'] = clf.predict(test\_feat.drop(['uid','areaunique'], axis=1),)

clf = RandomForestClassifier(n\_estimators=2500, class\_weight={0:0.89, 1:0.11}, bootstrap=False)

clf.fit(train\_feat.drop(['uid', 'label', 'areaunique'], axis=1), train\_feat['label'])

test\_cus['label'] = clf.predict(test\_feat.drop(['uid','areaunique'], axis=1),)

LightGBM部分：

同理，基于LightGBM模型，利用刚刚得到的train\_feature表格，排除无关参数后，对label参数进行训练。

调整n\_estimator数量，在保证训练质量的同时，防止过拟合现象的出现。

选择train\_feature表中的合适的参数对模型进行训练。

训练结束后对test\_feature表中的各个uid根据相关属性进行预测，并将结果存储在test\_cus表格中，方便后续进行模型融合。

代码如下：

# LightGBM模型

clf = LGBMClassifier(n\_estimators=2500,class\_weight={0:0.89, 1:0.11})

clf.fit(train\_feat.drop(['uid', 'label', 'areaunique'], axis=1), train\_feat['label'])

test\_cus['label\_lgbm'] = clf.predict(test\_feat.drop(['uid','areaunique'], axis=1),)

XGBoost部分：

同理，基于LightGBM模型，利用刚刚得到的train\_feature表格，排除无关参数后，对label参数进行训练。

调整n\_estimator数量，在保证训练质量的同时，防止过拟合现象的出现。

选择train\_feature表中的合适的参数对模型进行训练。

训练结束后对test\_feature表中的各个uid根据相关属性进行预测，并将结果存储在test\_cus表格中，方便后续进行模型融合。

代码如下：

# XGBoost模型

clf = XGBClassifier(n\_estimators=2000)

clf.fit(train\_feat.drop(['uid', 'label', 'areaunique'], axis=1), train\_feat['label'])

test\_cus['label\_xgb'] = clf.predict(test\_feat.drop(['uid','areaunique'], axis=1),)

三个模型整体的代码比较类似，都是进行了粗略的参数调节，没有进一步细化调节。

随机森林优化的参数主要是正负样本比参数，使用与训练集相同的正负样本比参数可以达到较好的结果，经过下方模型融合的分析，我怀疑测试集中正标签数据占比较多，所以我不断下调正负样本的比例，在将比例调整到0.89：0.11时，成绩又上涨了将近一个百分点，达到了0.735，这验证了我在下面模型融合阶段的猜想。但在调至0.87：0.13时成绩甚至有所下降。



**图5 测试结果**

LightGBM以及XGBoost模型主要调节了预测器数量参数防止其产生过拟合现象，在大于3500时会产生过拟合现象使成绩降低，所以最终我将参数设置在2500。

由于时间和能力问题，细节参数没有进行进一步调整。

模型融合：

这一部分我利用了Linear Blending融合方式对模型进行线性加权融合，取上述三个模型的分类结果并赋予权重，形成最终的预测结果。

代码如下：

ans=np.zeros(267,dtype=int)

print(test\_cus)

for i in range(0,267):

    row\_labels = test\_cus.index[i]

    tmp=1\*int(test\_cus.at[row\_labels,'label'])+1.4\*int(test\_cus.at[row\_labels,'label\_lgbm'])+0.6\*int(test\_cus.at[row\_labels,'label\_xgb'])

    print(tmp)

    if(tmp>=0.6):

        ans[i]=1

    else:

        ans[i]=0

print(ans)

ans\_cus['label']=ans

若仅采用1：1：1的比例，有两个模型的预测结果为1时最终结果才为1，整体预测的准确度仅能达到0.69左右，如图所示：



**图6 测试结果**

但当我将lgbm模型权重调高1.4，xgb模型的模型调低至0.6时，准确度有所上升，达到了0.71左右。



**图7 测试结果**

在我不断调节不同模型的加权时发现，我所给定的将最终结果赋为1的条件越宽松，成绩就越高，比如我将条件设为除xgb模型结果以外，其他任意一种模型预测值为1时结果都为1，所有模型中有两个及以上模型为1时预测结果为1，此时正确率达到了将近0.72。



**图8 测试结果**

根据这个思路，我认为我的模型训练时可能对赋1 的条件有些苛刻，或是测试集中label为1的元素占比较多，所以最终我将条件设为仅有三个模型预测结果均为0时预测结果才为0，这样成绩来到了将近0.727。



**图9 测试结果**

经过上述模型融合，最终我已经将各模型的加权设置到了极限，已将无法再通过调整模型加权来提高成绩，必须依靠对各个模型的优化来取得更高的成绩。根据上面调节模型加权的经验，我认为要提高成绩必须优化模型使其更容易做出1的预测，这样才能让成绩进一步提高。

# 6. 实验结果

最终得分0.73529，基本实现了测试数据的较高准确度的预测，截至提交时排名23。



**图10 成绩及排名**

# 7. 个人体会

这是我首次完成机器学习的一整套项目，整套流程说起来简单，真正实现起来还是有一定的难度的，在不同的阶段都遇到了各型各色的问题。

在数据处理的过程中，我遭遇了一些问题导致几个csv格式文件中的数据无法正常读取，我怀疑可能是数据格式的问题，所以重新实现了一个函数来解决了这个问题。

进行初步的训练时，成绩总卡在0.61，十分不理想，在分析了题意后，我逐步排除无用干扰信息，最终确定了实践标签会降低模型性能，将其去掉后，在初步的随机森林上也能达到将近0.68的成绩，符合了预期。

训练XGB和lgbm模型时还遇到了过拟合的问题，当我把n\_estimator参数设置到4000左右时，成绩迅速下降，但当我将其降低到2500左右时，成绩回升，我怀疑此处出现了过拟合现象。

在后期模型融合的过程中，同样也遇到了一些挑战，在我不断调整模型结果加权的过程中出现了奇怪的现象，我所给定的将最终结果赋为1的条件越宽松，成绩就越高。根据这个思路，我认为我的模型训练时可能对赋1 的条件有些苛刻，或是测试集中label为1的元素占比较多，所以最终我将条件设为仅有三个模型预测结果均为0时预测结果才为0，成绩得到了进一步的提高。

我还根据模型融合时得到的经验，进一步优化模型，通过调整正负样本的比例，成功再一次提高了成绩。

总的来说，本次实验比较考验我们的快速学习能力以及快速理解能力，提交截止日期延长之后减轻了一些压力，但还是很难能力有限的条件下进一步深挖相关知识。

但这也是一次还不错的机器学习入门体验，不断分析问题，优化参数，并看到自己的成绩不断提升还是很有成就感的。也让我知道了要做出一套准确的预测，需要有的不仅是精良的模型，还需要精炼到位的数据处理，四两拨千斤的参数调整，以及各个模块之间的协同作业，相互优化，这样才能让整个模型的预测效果不断提升。