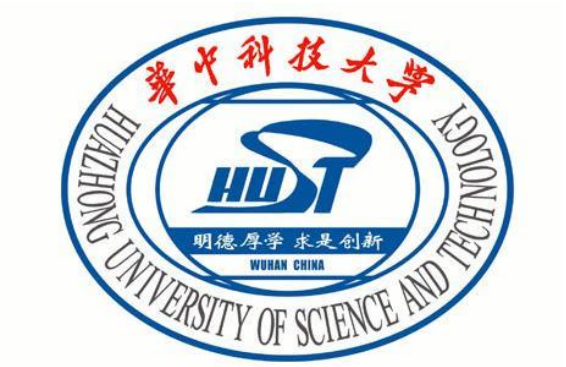
华中科技大学

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：计算机 202008 班

学　　号：U202015176

姓　　名：刘鉴之

成　　绩：

指导教师：何琨

完成日期：　2022 年　7　月　3　日

目录

1. 实验题目：智能家居使用场景识别挑战赛

某开展智能家居业务的公司在全国各地设有不同等级的代理商，为了让用户切身感受到智能家居产品的智能化和便捷性，每个代理商均有自己的智能家居体验店和展厅。

根据该公司的发展策略，需要系统能够准确、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

根据数据集，分析智能家居产品体系的应用场景。

2.2 数据说明

2.2.1 数据集组成说明

- 训练集和测试集均包含四类数据，测试集包含使用场景标签，用于识别算法训练，第二部分不包含场景标签，用于测试。

数据类别	对应文件	变量	格式	说明
账号信息	cus.csv	uid	string	账号 ID
		label	int	智能家居产品体系的使用场景
设备列表	devList.csv	uid	string	账号 ID
		did	string	设备 ID
		type	string	设备型号
		area	string	设备所在区域
控制操作日志	control.csv	uid	string	账号 ID
		did	string	远程控制的设备 ID
		time	bigint	远程控制设备的时间
		form	string	远程控制设备的方式
		data	string	远程对设备下发的控制日志
设备上报日志	devUpdate.csv	uid	string	账号 ID
		did	string	上报日志的设备 ID
		time	bigint	设备上报日志的时间
		data	string	设备上报的日志内容

表 2.1 数据集组成

2.2.2 数据量统计说明

数据集	数据类别	规模	备注
训练集	cus.csv	915×2	
	devList.csv	8858×4	area 列数据可能含逗号
	control.csv	334006×5	存在 json 型数据
	devUpdate.csv	346797×4	存在 json 型数据
测试集	cus.csv	267×1	
	devList.csv	2501×4	area 列数据可能含逗号

	control.csv	112371×5	存在 json 型数据
	devUpdate.csv	113574×4	存在 json 型数据

表 2.2 数据量统计

2.3 评测标准

本实验采用$F1-score$指标对预测结果进行评价：

（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）使用（1）中统计值，计算模型准确率$precision$和召回率$recall$，公式如下：

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

（3）使用第二步计算结果，计算$F1-score$,得到评测指标，公式如下：

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

3. 算法设计

本实验属于**二分类**问题，需要根据数据集预测智能家居的应用场景。

3.1 数据处理

3.1.1 数据读取

由于数据集中的部分单元格中的数据包含逗号，需要额外进行处理，因此定义了函数 my_readcsv 处理这类数据，具体实现见 5.2.1 内容。

3.1.2 数据统计与分析

为了便于算法训练与预测，需要对不同数据类别中的数据进行统计，以得到新的特征便于特征工程的构建，具体实现见 5.2.2 和 5.2.3 内容。

3.1.3 特征工程

使用相关系数法、递归特征消除法、模型分析法、人工选择法，分别选择相关度较高的特征，取并集后得到选择的特征子集，具体实现见 5.3 内容。

（1） 相关系数法

通过计算特征与特征之间的**相关系数**的大小，可**判定两两特征之间的相关程度**。

取相关系数值的绝对值，把值大于 90%~95%的两两特征中的某一个特征剔除，这样，便实现了对特征的筛选与数据降维，避免不重要的特征过多，使得模型的性能下降。

（2） 递归特征消除法

sklearn.feature_selection 提供的 RFE 利用递归剔除特征重要性最低的特征得到 k 个不同特征数量的特征子集，选择**分类精度最高的特征子集**作为最优特征组合。

（3） 模型分析法

通过使用回归模型、LightGBM 模型、XGBoost 模型的特征选择功能选择特征。

（4） 人工选择

3.2 模型算法

通过对逻辑回归、决策树、随机森林等不同模型进行准确率评估，最终选取随机森林模型、LightGBM 模型、XGBoost 模型参与模型融合并进行最后的预测。

1. 逻辑回归模型

虽然逻辑回归模型中带有“回归”二字，但它其实是一个分类模型。

2. 决策树模型

由课内知识可知，构造决策树的基本算法有多种，如使用信息增益进行特征选择的 ID3 算法。对于分类问题，决策树的最后一层叶子结点才是分类标签。

3. 随机森林模型

随机森林模型通过建立多个决策树并将它们融合起来得到一个更加准确和稳定的模型，其随机性体现在随机选取特征和随机选取样本。

4. LightGBM 模型

LightGBM 是基于 GBDT（梯度提升决策树）算法的分布式梯度提升框架。

5. XGBoost 模型

XGBoost 实现了梯度提升树(GBDT)模型，可以自动处理缺失值，也常用于分类问题。

3.3 模型融合

对于分类问题，常用的模型融合方式有以下几种：

1. Voting

在基学习器的基础上得到一个投票的分类器，把票数最多的类作为最终预测的类别。

2. Stacking

Stacking 模型是一种分层的结构，使用大量的基分类器，将其预测的结果作为下一层输入的特征，这样的结构使得它比相互独立训练模型能够获得更多的特征。

最终，我们选择了 Voting 的模型融合方法。

4. 实验环境与平台

OS：Windows10

CPU：Core i7

编程语言：Python 3.10.5

编辑器：PyCharm

5. 实现与分析

5.1 成员分工

具体模块	分工划分
数据读取、数据统计	刘鉴之
特征工程	共同完成
逻辑回归、决策树、随机森林模型	刘鉴之
LightGBM、XGBoost 模型及调参	徐瑞达
模型融合	徐瑞达
代码整理与优化	徐瑞达

表 5.2 成员分工

5.2 数据预处理（撰写:刘鉴之）

5.2.1 数据读取

由于数据中存在 json 等特殊数据字段，因此定义了函数 my_readcsv 实现对特殊数据的额外处理，参数为文件路径和分隔符，返回 DataFrame 对象。具体实现的部分代码如下：

```
def my_readcsv(path, sep=','):
    # 使用 codecs 读取整个文件到 lines
    try:
        lines = codecs.open(path, encoding='utf-8').readlines()
    except:
        lines = codecs.open(path, encoding='latin-1').readlines()
    # 获取文件表头 header
    header = lines[0].strip().split(sep)
    # 获取文件内容列表 content
    content = []
    for line in lines[1:]:
        line = line.strip()
        try:
            # 如果不包含 json 型数据,则直接使用 sep 分割 line
            index = [i for i, x in enumerate(line) if x == ',']
            if len(index) == len(header) - 1:
                content.append(line.split(sep))
            else:
                json_list = []
                # 将非 json 型数据读取至 json_list 中
                index = [0] + index
                for idx in range(len(header) - 1):
                    json_list.append(line[index[idx]:index[idx + 1]].strip(sep))
                json_list.append(line[index[len(header) -
1]:].strip(sep).replace(',', ';'))
                content.append(json_list)
        except:
            pass
    # 返回 DataFrame
    return pd.DataFrame(content, columns=header)
```

代码块 5.1 函数 my_readcsv 的实现

5.2.2 数据统计

使用模块 pandas 的 groupby 方法对各个数据类别对应 DataFrame 的 uid 列进行分组统计，并使用 agg 函数对不同变量应用不同的统计方法，变量统计方法如表 3.3：

数据类别	变量统计方法	说明
devList	'did': 'nunique', 'type': 'nunique', 'area':['nunique','unique', 'count']	统计设备列表中各账号对应的设备种类数、型号种类数、区域种类数、区域类型集合、区域数目
control	'did': 'nunique', 'data': 'nunique', 'form':['nunique','unique', 'count']	统计控制日志中各账号对应的设备种类数、日志种类数、命令类型种类数、命令类型集合、命令数目
devupdate	'did': 'nunique', 'data': 'nunique',	统计上报日志中各账号对应的设备数目、日志数目

表 5.3 数据统计方法

5.2.3 合并特征

使用模块 pandas 的 merge 方法将数据统计得到的 DataFrame 合并，然后使用 TF-IDF 技术对设备列表中的 areaunique 变量、操作控制日志中的 formunique 进行文本数据信息统计，此处举例对 formunique 的统计。

```
# 统计 formunique 列词频
tfidf = TfidfVectorizer(max_features=13)
# 使用'default'替代 formunique 为 0 的字段
train_control_tfidf=tfidf.fit_transform(train_feat['formunique'].apply(lambd
a x: 'default' if isinstance(x, int) is True else ' '.join(x)))
test_control_tfidf =tfidf.fit_transform(test_feat['formunique'].apply(lambda
x: 'default' if isinstance(x, int) is True else ' '.join(x)))
# 设置列名
features = list(tfidf.get_feature_names_out())
train_control_tfidf = pd.DataFrame(train_control_tfidf.toarray(),
columns=['form_unique_' + str(features.index(x)) for x in features])
test_control_tfidf = pd.DataFrame(test_control_tfidf.toarray(),
columns=['form_unique_' + str(features.index(x)) for x in features])
# 合并统计信息并删除 formunique 列
train_feat = pd.concat([train_feat, train_control_tfidf], axis=1)
train_feat = train_feat.drop(['formunique'], axis=1)
test_feat = pd.concat([test_feat, test_control_tfidf], axis=1)
test_feat = test_feat.drop(['formunique'], axis=1)
```

代码块 5.2 统计 formunique 变量的实现

以上工作完成后，即可得到处理后的 DataFrame，输出 train_feat 的特征如图 3.1：

```
Index(['uid', 'label', 'didnunique_x', 'typenunique', 'areanunique',
       'areacount', 'didnunique_y', 'datanunique', 'formnunique', 'formcount',
       ...
       'form_unique_3', 'form_unique_4', 'form_unique_5', 'form_unique_6',
       'form_unique_7', 'form_unique_8', 'form_unique_9', 'form_unique_10',
       'form_unique_11', 'form_unique_12'],
      dtype='object', length=425)
```

图 5.1 数据处理后 train_feat 的特征

在经过数据处理过后，被统计的变量变为了**数值型**特征，而且对于 area 和 form 变量，还利用 TF-IDF 技术得到了各 area 值和 form 值对应的统计信息，这样就根据原始数据构造出了便于进行算法训练的特征。具体统计结果见附件 train_feat.csv 和 test_feat.csv。

5.3 特征工程（撰写:共同完成）

在特征工程部分，我们利用不同的选择特征的方法对特征进行了筛选，具体介绍如下。

使用模块 sklearn 的 SelectKBest 方法可以根据特征间的相关系数选择特征，特征数目由参数 k 决定，由于总特征数目有 425 个，我们使用相关系数法选择了 20 个特征。

调用模块 sklearn 的 RFE 方法进行递归特征消除法得到包含 20 个特征的特征子集。

调用模块 sklearn 的 SelectFromModel 方法使用回归、LightGBM、XGBoost 模型进行特征选取，最后选择的部分特征子集如下：

['area_unique_3', 'area_unique_212', 'area_unique_170', 'area_unique_356', 'area_unique_201', 'area_unique_128', 'area_unique_53', …, 'area_unique_257']

任选 10 个特征绘制出的相关系数热力图如图 3.2 所示：

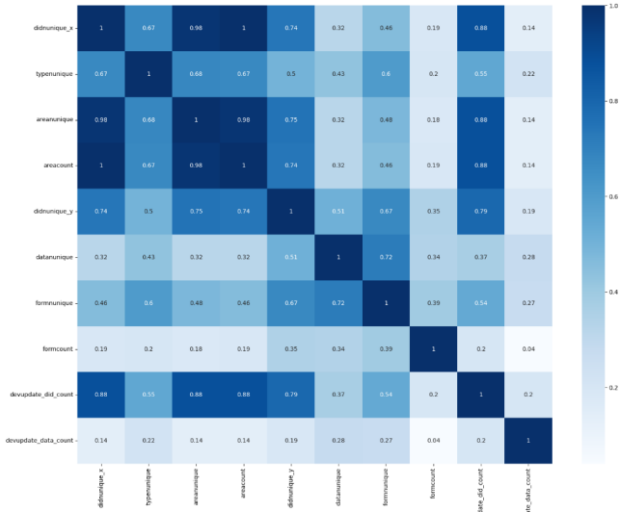


图 5.2 特征的相关系数热力图

5.4 模型预测一（撰写:刘鉴之）

在本项目中，分别采用了逻辑回归模型、决策树模型、随机森林模型、LightGBM 模型、XGBoost 模型等模型进行预测。其中前三个模型的预测由**刘鉴之**完成，后两个模型的预测和调参由**徐瑞达**完成。

首先需要使用训练集对模型进行训练，划分训练集的代码如下：

```
# 划分训练集
X = train_feat.drop(['uid', 'label'], axis=1)
y = train_feat['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

代码块 5.3 划分训练集的实现

5.4.1 逻辑回归模型

使用 sklearn 库中的 LogisticRegression 就可以进行训练，该模型对于训练集的预测准确率较高，但是在线上测试时效果不佳。

5.4.2 决策树模型

使用 sklearn 库中的 DecisionTreeClassifier 就可以进行训练，该模型对训练集的预测准确率往往是所选取的几种模型中最低的，线上结果也不尽人意，因此没有进行详细的调参。

5.4.3 随机森林模型

使用 sklearn 库中的 RandomForestClassifier 就可以进行训练，在未采用 LightGBM 模型和 XGBoost 模型进行测试之前，随机森林模型的线上测试结果最佳，是值得选取的模型之一。使用随机森林模型对训练集进行预测时的预测结果热力图如图 3.3 所示：

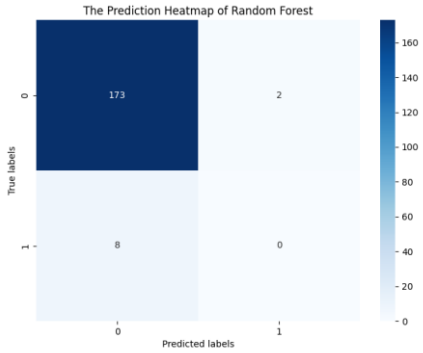


图 5.3 RF 在训练集上的预测热力图

5.5 模型预测二（撰写:徐瑞达）

5.5.1 LightGBM 模型

直接调用 lightgbm 模块中的 LGBMClassifier 方法进行训练即可。未经过调参的 LightGBM 模型对训练集的预测结果已经超过了逻辑回归模型和决策树模型，接近随机森林模型的结果，经过调参后，其准确率提升了 0.02 左右，高于随机森林模型。

使用调参前的 LightGBM 模型对训练集进行预测时的预测结果热力图如图 3.4 所示：

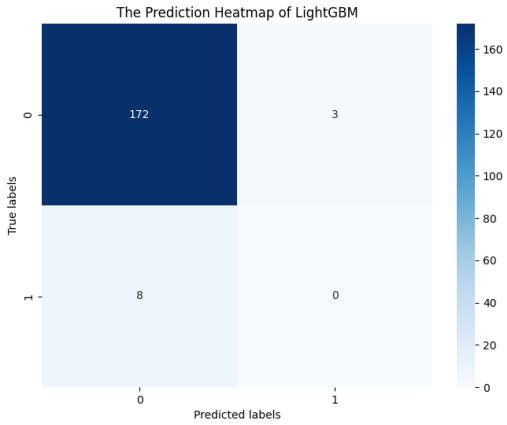


图 5.4 LightGBM 在训练集上的预测热力图

5.5.2 LightGBM 模型调参

在对 LightGBM 模型进行调参时，使用网格调参的方法，依次对 n_estimators 等参数进行了调整。具体参数说明与选取如下：

参数名称	选取	说明
num_leaves	5	设置树的模型复杂度，较大时准确率较高，但可能导致过拟合
max_depth	3	设置树的深度，防止过拟合
feature_fraction	0.7	使用特征的子抽样
bagging_fraction	0.6	
learning_rate	0.001	选择较小的学习率能获得稳定较好的模型性能
num_iterations	2	boosting 的迭代次数，过大会导致过拟合

min_child_samples	22	一个叶子上的最小数据量，用于提高模型泛化能力
min_child_weight	0.001	

表 5.4 LightGBM 主要参数表

使用调参后的 LightGBM 模型对训练集进行预测时的预测结果热力图如图 3.5 所示：

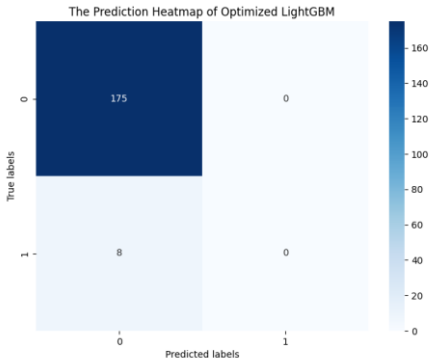


图 5.5 调参后的 LightGBM 在训练集上的预测热力图

可见预测失败的 label 数量由调参前的 11 变为了调参后的 8，训练集准确率上升。

5.5.3 XGBoost 模型

直接调用 xgboost 模块中的 XGBClassifier 方法进行训练即可。未经过调参的 XGB 模型对训练集的预测结果接近 LightGBM 模型的结果，经过调参后，其准确率与调参后的 LightGBM 模型相当，因此后期尝试使用这两种模型与随机森林模型进行融合。

使用调参前的 XGBoost 模型对训练集进行预测时的预测结果热力图如图 3.6 所示：

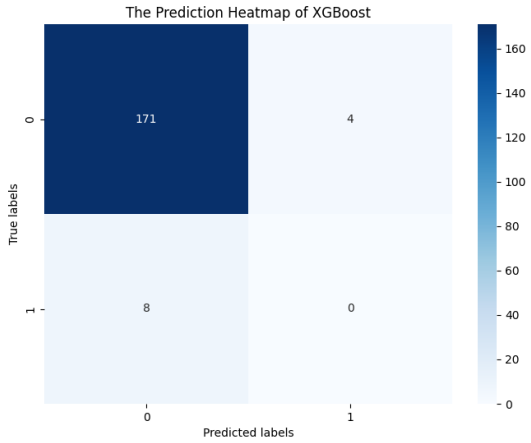


图 5.6 XGBoost 在训练集上的预测热力图

5.5.4 XGBoost 模型调参

与 LightGBM 模型调参方式相同，这里也采用了网格调参的方法，调参结果如下表：

参数名称	选取	说明
gamma	0.6	指定节点分裂所需最小损失函数下降值，值越大，算法越保守
max_depth	3	设置树的深度，防止过拟合
learning_rate	0.2	选择较小的学习率能获得稳定较好的模型性能
subsample	0.6	用于避免过拟合

colsample_bytree	0.6	控制每棵随机采样的列数的占比，即控制特征采样
min_child_weight	5	
n_estimators	25	

表 5.5 XGBoost 主要参数表

使用调参后的 XGBoost 模型对训练集进行预测时的预测结果热力图如图 3.7 所示：

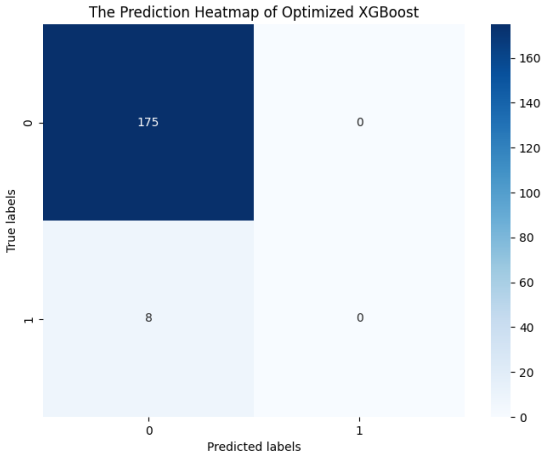


图 5.7 XGBoost 在训练集上的预测热力图

可见预测失败的 label 数量由调参前的 12 变为了调参后的 8，训练集准确率上升。

5.6 模型融合（撰写:徐瑞达）

经过模型测试环节，最终得到的模型准确度对比表如图 3.7：

Accuracy of Different Models

Model Name	Accuracy
Logistic Regression	0.9272727272727272
Decision Tree	0.92
Random Forest	0.9309090909090909
LightGBM	0.9236363636363636
Optimized LightGBM	0.9345454545454546
XGBoost	0.9272727272727272
Optimized XGBoost	0.9418181818181818

图 5.8 在训练集上的模型准确度对比表

由表可知，随机森林模型、调参后的 LightGBM 模型、调参后的 XGBoost 模型的预测准确率较高，可以选择这三种模型进行模型融合，以提高线上测试准确率。

1. Voting

使用模块 sklearn 的 VotingClassifier 方法即可对多个模型使用投票法进行融合，具体采用硬投票方式。使用投票方式进行模型融合后，线上结果提升了 0.05104。

2. Stacking

使用模块 sklearn 的 VotingClassifier 方法即可对多个模型使用投票法进行融合，具体采用硬投票方式。使用 Stacking 方式进行模型融合后，线上结果提升不明显，因此最终采用了

Voting 方式进行模型融合。

6. 实验结果

最终得分与排名如下图：

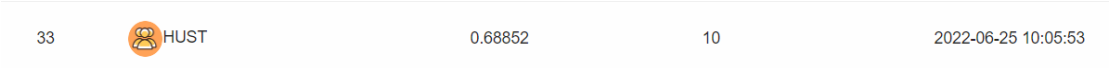


图 6.1 得分与排名

7. 个人体会

在这次实验中，我们小组遇到了许多问题，如没有学过 python，不知道机器学习如何下手，线上结果正确率总是上不去。

在网上查找相关资料后，我们小组制定了先学习基础知识如 python 语言、一些必要的库，接着进行数据处理，数据分析，模型选择的路线来完成这个项目。

我们花费了近五天的时间学习 python 语言，学习网上别人的项目思路，学习进行数据挖掘的整个流程。接下来，我们开始学习别人的经验，将项目分解为若干个步骤一步步地完成。

首先确定数据的特点，是否有缺失值，是否包含特殊字段，如何将 string 类型的数据抽象为可以进行算法训练的数值型数据等等，最后由我完成了数据的读取、统计和其他处理，这其中，pandas 库起到了很大作用。

接下来就是特征工程，从网上我们也得知特征工程对于整个预测准确率有很大影响。我们使用了各种方法筛选特征，比如相关系数法、模型选择法，最后我们还利用可视化手段分析了 form 列，area 列哪些数据字段对结果的影响较大，然而结果却不是很理想，在准确率迟迟徘徊在 0.6 左右的情况下，我们不得不暂时放弃了对特征工程的建立，而最终在进行模型的选取和融合后，线上正确率最高也只达到了 0.68852。

在选取模型上，我负责尝试逻辑回归模型、随机森林模型、决策树模型，而这几个模型所需要调整的参数也比较少，最后只选取了线下准确率较高的随机森林模型与瑞达同学负责的两种模型进行融合。

这次实验让我学到了很多，前前后后总共跨越了近三周时间。从学习基础知识到上手进行数据处理、特征工程、模型选择，虽然结果不是很理想，但我确实也学到了如何使用机器学习的手段来预测数据。