| 姓名 | 学号 | 成绩 |
|---|---|---|
| 姚施越 | 18120255 | |
| 胡亦得 | 18120257 | |
| 杨晨恩 | 18120256 | |

上海大学

# 数据分析程序设计

## 项目报告

| 课程设计成绩（总分 35） | | 分数 |
|---|---|---|
| 程序 | 程序设计完整，功能齐全（10-15 分） | |
| | 程序设计基本完整，功能基本完成（0-10 分） | |
| 报告 | 结构清晰完整，报告格式正确，语言通顺（15-20 分） | |
| | 格式不正确，封面或字体有误，语言欠通顺（0-15 分） | |
| | 总分 | |

项目名称：____基于决策树算法的幸福感预测___

组长：_____姚施越_____手机：_17721277085_

邮箱：_badwolf613@shu.edu.cn_

项目分工：

| 姓名 | 项目分工 | 自评成绩<br>（每人满分 35 分） |
|---|---|---|
| 姚施越 | 模型搭建及训练 | 35 |
| 胡亦得 | 数据预处理及可视化分析 | 35 |
| 杨晨恩 | 模型搭建及详解 | 35 |

任课教师：___李 颖_____

# 目　　录

# 一、项目介绍

## 1.1 赛事介绍

我组选择的项目名称为《快来一起挖掘幸福感!》，是来自阿里天池的新人实战赛，比赛链接：https://tianchi.aliyun.com/competition/entrance/231702/introduction。首先下载数据，其次在本地调试算法，最终在阿里天池提交结果，通过评测指标来评价模型的优劣，评测指标的计算详见 1.3。

赛题使用公开数据的问卷调查结果，选取其中多组变量，包括**个体变量**（性别、年龄、地域、职业、健康、婚姻与政治面貌等等）、**家庭变量**（父母、配偶、子女、家庭资本等等）、**社会态度**（公平、信用、公共服务等等），来预测其对**幸福感**的评价。详细的数据说明请查看下文的 2.2.1 项目文件说明。

## 1.2 赛题背景

在社会科学领域，幸福感的研究占有重要的位置。这个涉及了哲学、心理学、社会学、经济学等多方学科的话题复杂而有趣；同时与大家生活息息相关，每个人对幸福感都有自己的衡量标准。如果能发现影响幸福感的共性，生活中是不是将多一些乐趣；如果能找到影响幸福感的政策因素，便能优化资源配置来提升国民的幸福感。目前社会科学研究注重变量的可解释性和未来政策的落地，主要采用了**线性回归**和**逻辑回归**的方法，在收入、健康、职业、社交关系、休闲方式等经济人口因素；以及政府公共服务、宏观经济环境、税负等宏观因素上有了一系列的推测和发现。

赛题尝试了**幸福感预测**这一经典课题，希望在现有社会科学研究外有其他维度的算法尝试，结合多学科各自优势，挖掘潜在的影响因素，发现更多**可解释**、**可理解**的**相关关系**。

## 1.3 评测指标

提交结果为 csv 文件，其中包含 id 和 happiness 的预测值两列。

分数计算公式：

$$score = \frac{1}{n}\sum_{1}^{n}(y_i - y^*)^2$$

其中$n$代表测试集样本数，$y_i$代表第$i$个样本的预测值，$y^*$代表真实值。

## 1.4 项目思路

本次的幸福感预测任务实质上是一个回归问题（目标只有 5 个值，也可以理解为分类问题，但由于平台评测指标使用的是均方误差，这样考虑显然是不合适的），我们采用了决策树作为核心算法，挑选了三个强大的模型：XGBoost，LightGBM，CatBoost 算法，并且采用模型融合的方式将三个模型的结果融合在一起形成最终的模型。

# 二、项目实施

## 2.1 项目功能概述

本部分将介绍项目的具体实施过程，包括**数据预处理**、**数据可视化分析**、**模型详解**及**模型训练**、**结果预测**等，最后展示本项目的实验结果。其功能流程图如图 1 所示。

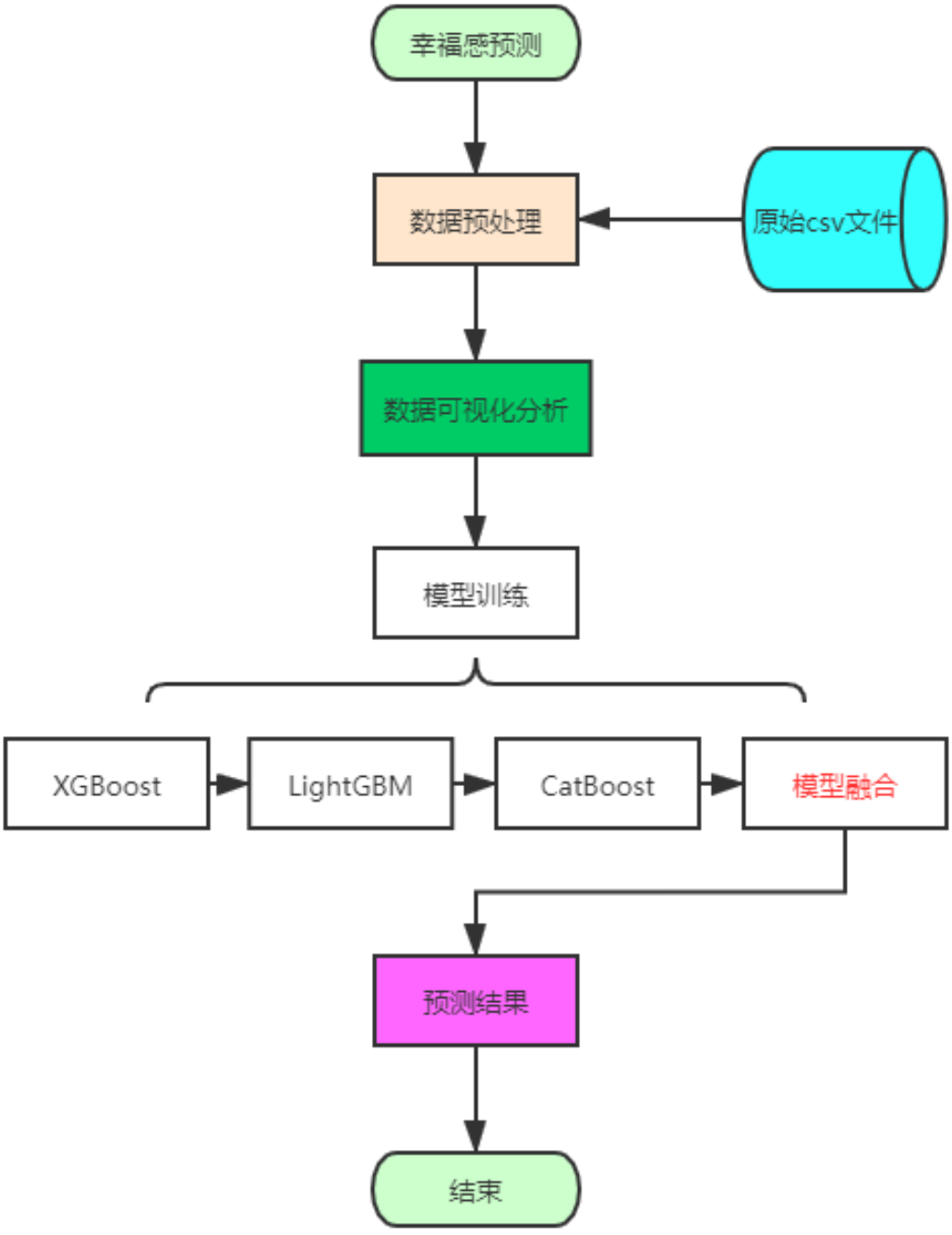


图 1 项目功能流程图

## 2.2 数据预处理

首先，对原始数据进行预处理。结合可视化，挖掘数据集本身的特征，并以此为启示对数据集进行处理，挖掘其潜在特征，扩展数据集的同时降低数据集的维度，大大提升了算法效率。

### 2.2.1 项目文件说明

本项目中共有多个数据文件，数据的有效时间为 2015 年，包括参考类文件：

(1)happiness_survey_cgss2015.pdf 居民调查问卷的 PDF 文件。

(2)happiness_index.xlsx 调查结果表中各个字段及其不同值的含义说明。

以及数据类文件：

(1)happiness_train_abbr.csv 简化版的训练集文件，包括目标以及所有特征中的一部分，特征较少，利于上手。

(2)happiness_train_complete.csv 完整版的训练集文件，包括目标以及所有特征中的全部，特征较多，也是我小组所使用到的。

(3)happiness_test_abbr.csv 简化版的测试集文件。

(4)happiness_test_complete.csv 完整版的测试集文件。

(5)happiness_submit.csv 平台提交结果示例文件，最终提交到平台所用。

参考文件当中一共涉及六部分：核心模块（家庭情况、社会人口属性、健康、移居、生活方式、社会观点、阶层认同、政治态度、认知和劳动、社会保障），十年回顾，EASS（东亚联合社会调查），ISSP（国际社会调查项目），能源使用，法制观点，联系方式。通过对比参考类文件和数据文件可以发现，所给的完整数据集选取了核心模块和部分的十年回顾内容。

### 2.2.2 具体处理方法

通过肉眼观察，结合 happiness_index.xlsx 文件，查看各字段含义，发现数据集整体可以划分为以下十余个方面：年代、信仰、受教育程度、政治面貌、房产、业余生活、财富等等。同时，数据集本身并不规整，存在空缺数据和不合理输入。使用 Pandas、Numpy 库进行预处理，将所有为空的数据归置为-1，如果该问题列当中很少有人回答则考虑删除该列。

根据字段含义的不同方面，考虑把同一类的重复问题合并，例如把有关于家庭投资的多个问题（股票、基金、活期）等通过问题顺序编制成二进制编码，例如：参与股票，基金不参与活期的样例改为 110，再转换为十进制整数 6，缩减数据规模，如果有特殊情况的则使用全 1 或者全 1 往上的数字进行编码。这样的合并一共有两组，一组是房产问题，一组是投资问题。此外，把日期更改为年龄这一可以进行分析的数据类型。

### 2.2.3 数据可视化分析

原始数据中，共 8000 个训练数据，2968 个测试数据，除目标外有 139 个特征。查看 happiness 属性的数据分布时，发现 12 个异常值为-8，将其删除。

```
 4     4818
 5     1410
 3     1159
 2      497
 1      104
-8       12
Name: happiness, dtype: int64
```

图 2 happiness 属性分布

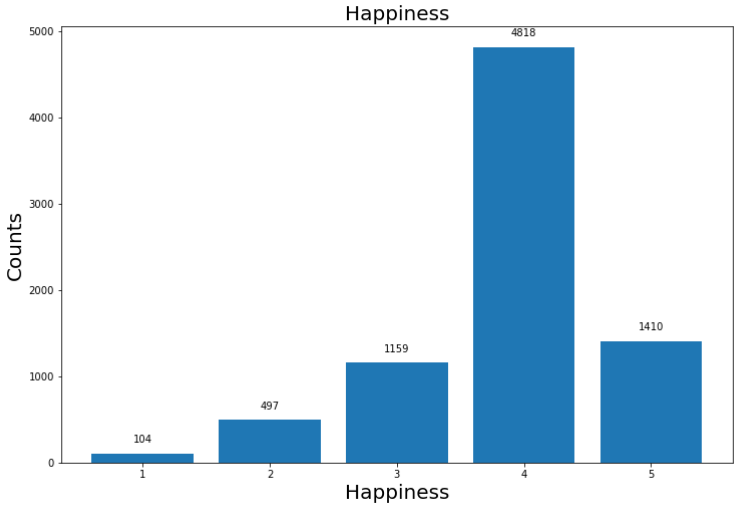


图 3 去除异常值后 happiness 属性分布（条形图）

该条形图表明受访者 happiness 属性的分布中，4 占大多数，3 和 5 次之，1、2 最少。

接下来查看各属性与 happiness 的相关性，利用相关系数这一指标来评估，首先输出前十的属性如图 4，并绘制所有属性与 happiness 的相关系数如图 5。

```
depression
class
health
equity
family_status
health_problem
class_10_after
public_service_6
class_10_before
public_service_7
```

图 4 与 happiness 相关系数前十的属性

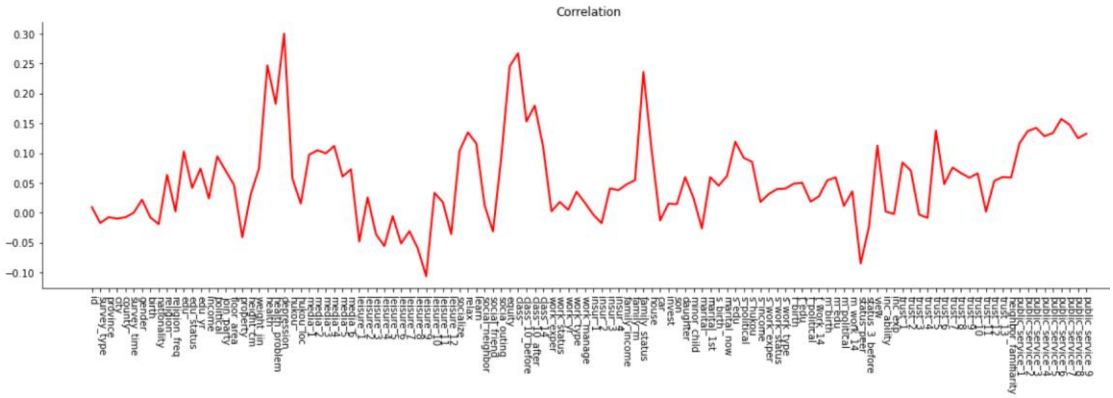


图 5 所有属性与 happiness 相关系数折线图

# 2.3 模型详解

## 2.3.1 XGBoost

### 2.3.1.1 GBDT 梯度提升树

XGBoost 模型实际上是对 GBDT[1]的工业化实现和进一步的改进。全称是 Gradient Boosting Decision Tree，其中 Gradient Boosting 指的是梯度上升算法，用来对分类器进行筛选的方法，Decision Tree 决策树指的是 CART 回归树弱分类器。

GBDT 是传统机器学习算法当中对真实分布拟合最好的算法之一，其主要思想是通过基函数的线性组合以及不断减少残差来得到回归预测。[3]通过不断地迭代生成一系列的树，然后对这一系列的树进行加权求和，目的是找到一个 CART 回归树模型的弱学习器，是的损失函数最小。其中的数学方法不多做介绍。

梯度提升树是提升树的改进算法，主要思想是在迭代中不断地接近拟合目标，减少拟合损失，例如目标是拟合一个距离是 50，第一次拟合到 30，下一次迭代用 10 来拟合差距 20，这样继续往下，最后每次迭代的结果 30+1…之和就是最终的模型输出结果。

**算法 1. 提升树算法**

(1)初始化$f_0(x) = 0$

(2)对$m = 1,2,\cdots,M$

(a)计算残差

$$r_{mi} = y_i - f_{m-1}(x), i = 1,2,\cdots,N$$

(b)拟合残差$r_{mi}$学习一个回归树，得到$h_m(x)$

(c)更新$f_m(x) = f_{m-1} + h_m(x)$

(3)得到回归问题提升树

$$f_M(x) = \sum_{m=1}^{M} h_m(x)$$

对于这样的模型需要的输入为学习率，迭代次数和树的深度，首先初始化弱学习器，把数据残差也就是负梯度——上一轮的学习器差值作为标签值，遍历所有特征可能，找到最佳划分节点，找到总平方损失最小的作为划分。迭代次数结束之后将多棵树进行加总成为最后的强学习器，形如式(1)。

$$f(x) = f_5(x) = f_0(x) + \sum_{m=1}^{5}\sum_{j=1}^{4} \Upsilon_{\mathrm{jm}} I\left(x \in R_{jm}\right) \tag{1}$$

### 2.3.1.2 CART 回归树

CART 算法既可以用于创建分类树，也可以用于创建回归树、模型树，两者在建树的过程稍有差异。[1][2]

在建树的过程当中使用基尼系数作为判定标准选择一个作为类别的单位，在划分树的过程当中不断地重复寻找能够得到最好值的特征值作为切分树的标准，一直循环执行直到无法切分。

#### 2.3.1.3 XGBoost 模型[5]

XGBoost[2]模型的逻辑和 GBDT 模型是一样的，而 XGBoost 使用贪心算法求解树结构，定义了一个增益值，对于每一层都用线性扫描寻找到最佳分裂，如下式(2)。

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_L+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \lambda \quad (2)$$

**总体算法流程：**

(1)迭代生成树。

(2)迭代初始对每个样本计算损失函数，这个计算和已经迭代好的树有关。

(3)使用目标函数采用贪心算法求当前树。

(4)新增树后，模型更新。

(5)使用收缩率改变新增树的函数，防止过拟合。

相比 GBDT 支持线性分类和正则化逻辑回归，通过列抽样有效降低了模型的方差。

### 2.3.2 LightGBM

#### 2.3.2.1 算法概述

前文说到GBDT和XGBoost在每次迭代选择新的划分点时，都需要遍历所有的样本点，而本数据集数量大，维度高，每次都对于数据集进行遍历是非常耗时的。LightGBM[3]算法针对于这两点做出了改进，提升了模型的训练速度，并且在某些数据集上具有更加良好的表现。改进主要体现在两个方面：一是应用了 GOSS(Gradient-Based One-Side Sample)算法，算法不会基于所有的样本点计算梯度，而是给予梯度的绝对值大的样本点更大的权重，并且对于小梯度的样本点进行采样，减少了参与梯度计算的样本点的个数，提升了计算效率；二是应用了 EFB(Exclusive Feature Bundle)，在划分特征时并不扫描所有的特征，从中找出最佳的划分特征，而是将互斥的特征合并成为特征包，降低总特征的个数，降低了寻找最佳划分点的消耗。

因此，LightGBM 实际上是应用了 GOSS 和 EFB 的 GBDT 算法，下面将详细介绍 GOSS 和 EFB 的实现方式。

#### 2.3.2.2 GOSS(单边梯度采样)

为什么我们能在计算梯度时忽略那些梯度较小的样本点？这是因为上一轮迭代计算的梯度本身就含有额外的信息，如果某个样本点的梯度较小，就代表该样本点的训练误差已经很小，已经经过了很好的训练，因此我们可以不将这些样本点纳入梯度计算的范围。而如果直接抛弃这些样本点，将会改变数据集的分布，这是我们不希望看到的。因此，在不改变数据集分布的情况下，通过采样的方式从数据集中抽取少部分的小梯度样本点就是 GOSS 算法的核心思想。

**具体做法如下：**

(1)首先根据样本点的梯度绝对值进行降序排序，随后从中选取前 a%(a 为超参数)的样本子集作为大梯度样本点，这些样本点将会参与梯度计算。

(2)对剩下的(1-a)%的样本点进行随机采样，采样系数为 b%，生成了小梯度样本点集合。

(3)合并大梯度样本点和小梯度样本点，将小梯度样本点乘上系数 1-a/b。

(4)用这样计算得到的样本点集合训练新的弱分类器，如此迭代直到算法收敛。

GOSS 算法的流程图如图 6 所示：

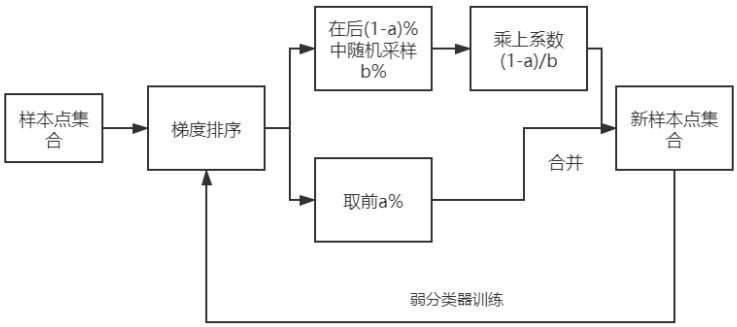


图 6 GOSS 算法流程图

### 2.3.2.3 EFB(Exclusive Feature Bundling)

在介绍 EFB 的原理之前，首先需要介绍什么是互斥特征。互斥特征实际上是特征取值不同时非 0，如 one-hot 编码。在本数据集上应用 EFB 是非常合适的，因为本数据集中有大量的 one-hot 编码特征如 Property，leisure 等属性都是具有很高互斥性的特征。而只有互斥的特征在合并时才不容易丢失特征，这是因为如果两个特征之间是互相纠缠的，那么这两个特征在合并之后势必会有重叠的部分，这样很有可能会丢失数据集重要的信息。EFB 的核心思想就是利用数据集高维稀疏的特性，合并这些互斥的特征，从而减少特征的维数。

在合并特征之前，我们首先需要寻找互斥特征，我们可以把互斥的特征想象成是不同颜色的点，于是寻找互斥特征的问题就归结成了图着色问题，一个节点代表一个特征，一个特征应该和与它相连的特征是互斥的。然而图着色问题是一个 NP-hard 问题，也就是多项式时间内不可解的问题，于是还需要对问题进一步的简化。由于寻找完全互斥的特征是非常困难的，很容易想到是否可以允许两个特征之间不完全互斥而是近似互斥(允许小部分的冲突)。定义冲突率作为度量特征之间互斥性的指标，冲突率越小，互斥性越强，特征就越有可能被合并，算法将冲突性小的特征归入一个 bundle，然后在每个 bundle 中合并特征。

**具体做法如下：**

(1)首先构造一个带权图，权重为特征之间的冲突率。

(2)将特征按度数降序排列。

(3)检查有序列表当中的每一个特征，将其分配给当前具有最小冲突的 bundle，或者是创建新的 bundle。

流程图如下图 7 所示：

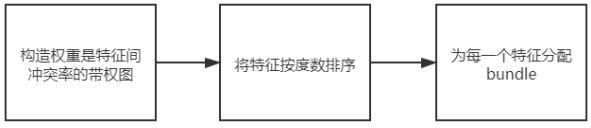


图 7 EFB 算法流程图

在寻找到近似互斥的特征之后，接下来要考虑如何高效地合并特征。

算法使用直方图来加速特征之间的合并。直方图的优势之一在于它可以将取值连续的特征映射到整型的编号，根据这个编号遍历寻找最优切分点。其次，由于直方图本身的计数特性，当前节点的直方图可以由父节点和其兄弟节点相减得到，这样又进一步减少了计算所需要的时间，如下图 8 所示。


图 8 加速直方图计算

另外在合并两个取值范围有重叠的特征时，可以采用添加偏移量的方式来合并两个特征。例如现有两个互斥特征 A，B，A 的取值范围是[0,10]，而 B 的取值范围是[0,20]，我们就可以给 B 增加一个偏移量取值为 10，B 就变为[10,30]，于是我们就可以合并特征 AB(其取值为[0,30])来取代原来的特征 A 和特征 B。

在应用了上述两个改进算法之后，就可以利用 GBDT 的核心思想每次迭代构造一个弱分类器，最终形成一个强分类器用于我们的幸福感预测任务。

## 2.3.3 CatBoost

### 2.3.3.1 算法概述

Catboost[4]算法是属于 Boosting 算法族中的算法的一员，其核心思想与 GBDT 算法无异。Catboost 的全称是 Category Boosting，顾名思义，它可以很好地处理 Categorial feature，也就是类别型特征。这也与本数据集的特性相符，幸福感数据集中含有大量类别型的特征，如年份特征，这也是我们选择该算法的原因之一。另外，Catboost 还具有性能卓越，鲁棒性强等优势，其结果势必会对最后的模型融合有巨大的帮助。

### 2.3.3.2 类别型特征到数字特征的转换

类别型特征在训练之前首先要转换为数字特征，Catboost 使用特殊的方式来处理这类数据。在传统的 GBDT 中，最简单的处理方式就是将类别特征用其所对应类别的平均值来替换，但是往往特征相比类别而言会包含更多的信息，采用这样简单的替换方式，毫无疑问会丢失关键信息，导致模型精度不足。而 Catboost 采用的方法是，基于数据集的先验知识(如数据集中某一类出现的概率)来构建类别型特征到数值型特征的转换，这样做的好处在于不单单考虑到某几个样本点的类别，而将整个数据集纳入考虑，这样得出的数据型特征是全局的，并且也省去了手动处理类别型特征的麻烦。

### 2.3.3.3 组合类别

Catboost 和 LightGBM 类似，同样具有特征降维的机制。Catboost 算法认为任意的类别型特征都可以进行组合而形成新的更强大的特征，并且将组合完成的类别特征转换成数字特征具有更好的效果。举个例子，由于幸福感数据集是由问卷收集而来，问卷又存在着多选题的形式，多选题中的选项的每一个组合都会形成一个特征，如 leisure 属性中有看电视、看报纸等等，而某些老年人的休闲娱乐方式就是看电视和看报纸，换言之，看电视和看报纸两个特征之间存在着一定的联系，那么如果将这两个特征单独转换成数字特征就会造成关键信息的丢失。所以，我们需要将“看电视”和“看报纸”这两个类别型特征组合起来，形成一个更加强大的特征------“看电视和报纸”，这样转换成数字型特征就不会造成信息的丢失。这实际上和 LightGBM 的特征合并有着异曲同工之妙。

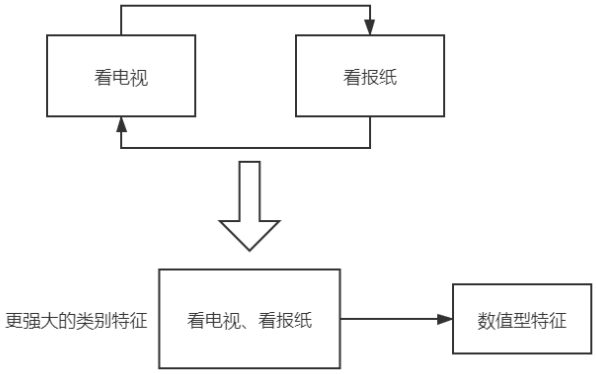


图 9 合并特征示意图

### 2.3.4 模型融合

在完成了上述三个算法之后，最后一步就是要将所得到的模型融合起来得到最终的回归模型，我们采用了模型融合的 Stacking 方法。

我们需要使用 Stacking 方法对于 XGBoost，LightGBM 和 CatBoost 进行融合，每一次迭代都需要分别对三个模型进行五折交叉验证，再综合每个模型五折交叉验证的结果得到本次迭代的模型。

五折交叉验证，顾名思义，就是将数据集划分为 5 份，其中四份作为训练数据，而剩余的一份作为测试集。一次交叉验证需要做两件事情：一是基于当前训练集训练模型并给出当前测试集的预测；二是给出原数据集测试集的预测。而对于一个相同的数据集，对数据集进行 4:1 的划分共有 5 种，因此每一次交叉验证又会进行 5 次，每一次交叉验证算法都会根据当前的训练集训练特定的模型，并且对当前的测试集给出预测。每一次迭代又需要将每次测试集的预测结果堆叠起来得到一个和原数据集大小相同的矩阵，这个矩阵将会作为下一次迭代的训练集。另外，模型除了会给出对当前测试集的预测，还会给出对原数据集测试集的预测，交叉验证结束后会对每次交叉验证的预测结果求平均值，以求得下一次迭代的测试集。对三个模型都采用上述的交叉验证步骤，采用堆叠的方式融合三个算法的结果，重复以上步骤就得到了最终的模型，算法的具体流程如下图所示。

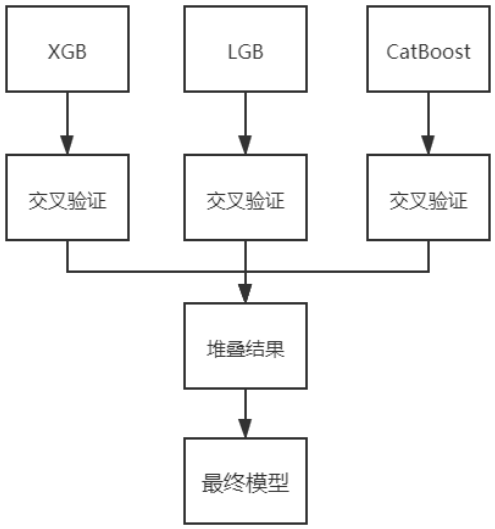


图 10 模型融合流程图

## 2.4 实验情况

### 2.4.1 环境说明

项目平台：Jupyter Notebook
项目环境：Python3.8
项目依赖包：pandas, numpy, sklearn,lightgbm, xgboost, catboost, matplotlib

### 2.4.2 使用参数

#### 2.4.2.1 XGBoost

表 1 XGBoost 模型参数设计意义及最优数值

| 参数 | 参数值 | 含义 |
|---|---|---|
| Booster | gbtree | 每次迭代使用树模型 |
| Eta | 0.005 | 迭代权重(学习率) |
| max_depth | 5 | 最大树高 |
| Subsample | 0.7 | 随机采样比例 |
| colsample_bytree | 0.8 | 列随机采样比例 |
| Objective | reg:linear | 任务类型 |
| eval_metric | rmse | 评价指标 |
| Silent | True | 不输出任何信息 |
| Nthread | 8 | 可用最大线程数 |

#### 2.4.2.2 LightGBM

表 2 LightGBM 模型参数设计意义及最优数值

| 参数 | 参数值 | 含义 |
|---|---|---|
| boosting_type | gbdt | 提升算法 |
| num_leaves | 20 | 叶子节点的数量 |
| min_data_in_leaves | 20 | 叶子节点中最少的记录数 |
| objective | regression | 任务类型 |
| max_depth | 6 | 最大树深 |
| learning_rate | 0.01 | 学习率 |
| min_child_sample | 30 | 最小孩子数 |
| feature_fraction | 0.8 | 每次迭代使用的参数比例 |
| bagging_fraction | 0.8 | 每次迭代使用的数据比例 |
| metric | mse | 评价指标 |
| lambda_l1 | 0.1 | 正则化方法 |

#### 2.4.2.3 CatBoost

表 3 CatBoost 模型参数设计意义及最优数值

| 参数 | 参数值 | 含义 |
|---|---|---|
| n_estimators | 10000 | 树的数量的最大值 |
| loss_function | RMSE | 损失函数 |
| eval_metric | RMSE | 评价指标 |
| learning_rate | 0.05 | 学习率 |
| depth | 5 | 最大树高 |
| use_best_model | True | 是否采用最佳模型 |
| subsample | 0.6 | 样本采样比例 |
| bootstrap_type | Bernoulli | 权重采样方式 |
| reg_lambda | 3 | 正则化 |

### 2.4.3 实验结果

本项目数据集由于本身数据量不多且都是数值类型的数据，因此模型的训练速度较快，在 CPU 环境下，采用 8 线程进行训练用时约为 4 分钟。最终模型在 RMSE 损失下的评分约为 0.47,在所有近 7000 个参赛队伍中排名 222 名。

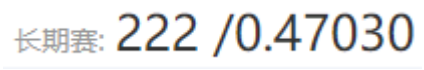


图 11 最终结果与排名

# 三、总结

18120255 姚施越

20-21 学年秋季学期的《数据分析程序设计》课程随着这篇报告的撰写完毕便告一段落了。在本次项目中，我主要负责模型的搭建和训练，回顾这十周的课程学习及项目实战，我对于 Python 和机器学习的认识加深了不少。对此，我有几点感触，在下文中展开叙述。

首先，关于数据预处理。作为一个回归问题，数据本身是相当重要的，因此预处理是必不可少的。同时，看似无关的数据可能存在相关性，这正是数据挖掘所要去探寻的。因而除了将缺失值较多的数据删去，对于剩余数据的处理，通过计算相关系数帮助我们去抉择，采用了如将独热编码看作二进制并转成十进制来降低数据规模的方法。大部分的数据都没有改动，通过后续模型来挖掘其深层关系。

其次，如何选择模型。在李老师的悉心指导下，我对于以 CART 和 C4.5 为首的决策树有了浅显的认识。因此，审视该问题之后，很快便决定使用决策树算法来解决。对于 XGBoost 这一“比赛大杀器”，我早有耳闻，在仔细剖析和上手使用后更是深深感受到了其的强大。对于其背后的数学原理企图顷刻之间理解，自然是不容易的。这时候他人的经验就显得很有参考价值了。通过阅读原文，相关书籍，博客分享等方式，收集到他人的理解和实现，对于自己如何考虑也能够有更好的认识，更能快速上手。针对不懂的问题与队友共同探讨，再加以自己的思考，能够更快地获得解答。以上两点也正是我的做法，很好地帮到了我。

最后，说说自己的收获。这种考核方式对于我来说是不小的挑战，但也是通过这几周的时间，经历了摸索的过程，我对于如何搭建完整的机器学习项目有了一些实战经验，针对决策树算法也有了一些浅显的认识。调参训练的过程，伴随着模型效果出众的喜悦，回想起来也是非常的难忘，也深感数据挖掘这一领域的魅力。虽然这次对于某些知识的运用和衔接还不够熟练，但是我将在今后的学习中继续努力、不断完善。

18120257 胡亦得

本次挖掘幸福感的项目当中，我主要负责了数据处理和 XGBoost 模型的测试方面。在数据处理上主要运用了 Python 的 Pandas 库，但是数据处理当中方法上主要是人工观察然后通过脚本方法一个个数据类处理，并没有达到可移植性很强，一旦源数据进行比较大的更新就难以进行处理，例如对于部分独热编码的操作我认为有更好的处理方法，可以通过获取列标 tolist 方法匹配字符串来找到同类项的方法，这样可以更高效的批量处理整个数据，对于处理空缺过多的数据删除也可以通过类似的对比方法来看。

在对数据进行相关性的分析的时候很容易发现，大部分的相关问题选项并不是直接合并或者类比就能够拿到最好情况，更多的情况下甚至会比原先更差，对于如何把同类型选项进行有效合并是需要进一步学习的，这个方法不一定是数据处理方法，也有可能是社会学统计方法，例如关于家庭财产的投资是否有基金、定期等等情况，看似全部都是一个类别的问题，能够体现一个家庭的资金富有程度和财富观念，对应的高级投资方式可能应该具有更大的权重，而不是一概而论。

在整个实验过程当中，使用 sklearn 的过程感觉还是不够清晰，对于算法的内部结构，具体操作，回归树的搭建和选择还是不够清楚，在后续过程当中还能够进一步的学习。

18120256 杨晨恩

本次大作业我主要负责模型的搭建以及算法的学习。本次比赛项目是一个回归问题，因此我们选用了“比赛神器”梯度提升树作为核心算法。为了使得模型兼顾各种优点，有更强的鲁棒性，我们还采用了模型融合算法结合了 XGBoost，LightGBM，CatBoost 三大梯度提升决策树的训练结果，实验证明该方法取得了非常不错的成效。

对于我来说，在上这门课程之前，我对于决策树的理解仍然停留在非常粗浅的层面。由于此次大作业中项目的需要，从零开始学习类似于 XGBoost，LightGBM 等的高阶决策树算法是非常困难的，因此我也阅读了英文原文文献，查阅了大量博客。在学习决策树改进算法的过程当中，我意识到决策树作为十大机器学习算法不无道理，决策树不仅原理简单，并且效果良好，改进的决策树算法现如今也活跃在如今各种机器学习比赛中。后人对于决策树算法做出的改进和更新无不是站在巨人的肩膀上，使决策树能够应用于各种各样不同的场合，获得更高的准确率。GBDT 基于决策树提出了使用训练多个弱分类器最终得到强分类器的方法，而 XGBoost，LightGBM，CatBoost 则基于 GBDT 算法在各个方面做出改进，提升了训练效率，模型准确率，并且使得模型能够适应多种数据。

数据分析程序设计是一门非常有趣的课程，使用深度学习算法解决现实问题是和大数据打交道的过程，在算法不断发展的今天，人们开始深入研究数据的本质，我们通过某些方法挖掘各种各样数据的潜在信息，并且解释数据存在的意义，这样不仅可以推动算法的发展，也可以使我们对于深度学习有更加深入的理解。而这门课程以 Python 语言为媒介，介绍了数据处理的方法以及诸多机器学习经典算法，最后又采用趣味大作业的形式使我们真切地感受到了数据挖掘的魅力。最后，要感谢我的队友的殷勤付出和李颖老师的尊尊教诲，使我获益良多。

**致谢** 首先，非常感谢李老师一学期的辛勤付出，在课程学习中，我们的收获良多，对于 Python 的使用和数据分析有了更清晰的认识。其次，也向组员们的共同努力表示由衷的感谢。最后，在本篇报告中，也难免会有不少纰漏和不严密的地方，敬请李老师批评指正。

# 四、附录

## 4.1 参考文献


[1] H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics 2001, Vol 29, No.5, 1189-1232.

[2] Tianqi Chen, Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. ACM 2016.8.

[3] Guolin Ke, Qi Meng, Thomas Finley, Tianfeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. LightGBM:A Highly Efficient Gradient Boosting Decision Tree.Microsoft Research NIPS 2017.

[4] Anna Veronika Dorogush, Vasily Ershov,Andrey Gulin. CatBoost: gradient boosting with categorical features support.

[5] XGBoost Documentation.https://xgboost.readthedocs.io/en/latest/


## 4.2 项目源代码

Happiness.py 文件如下：

```
#!/usr/bin/env python
# coding: utf-8

# # 一、准备工作

# In[68]:


import sys
print(sys.version)
print(sys.version_info)


# ## 1.1 安装工具包

# In[1]:


# 安装 lgb、xgb、ctb 决策树、matplotlib
get_ipython().system('pip install -i https://pypi.tuna.tsinghua.edu.cn/simple lightgbm')
get_ipython().system('pip install -i https://pypi.tuna.tsinghua.edu.cn/simple
```

```
xgboost')
get_ipython().system('pip install -i https://pypi.tuna.tsinghua.edu.cn/simple catboost')
get_ipython().system('pip install -i https://pypi.tuna.tsinghua.edu.cn/simple matplotlib')
# ## 1.2 导入库

# In[2]:


# pandas 和 numpy 用于数据预处理；sklearn 用于模型训练；xgb、lgb、ctb 为决策树，用在模型主体；matplotlib 用于绘图
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
import lightgbm as lgb
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold, RepeatedKFold
from scipy import sparse
from catboost import Pool, CatBoostRegressor
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

# ## 1.3 导入数据

# In[3]:


# 导入数据
train_abbr=pd.read_csv("./Dataset/happiness_train_abbr.csv",encoding='ISO-8859-1')
train=pd.read_csv("./Dataset/happiness_train_complete.csv",encoding='ISO-8859-1')
test_abbr=pd.read_csv("./Dataset/happiness_test_abbr.csv",encoding='ISO-8859-1')
test=pd.read_csv("./Dataset/happiness_test_complete.csv",encoding='ISO-8859-
```

```
    1')
test_sub=pd.read_csv("./Dataset/happiness_submit.csv",encoding='ISO-8859-1')


# # 二、数据预处理

# ## 2.1 观察数据

# In[4]:


#观察数据大小
print(test.shape)
print(test_sub.shape)
print(train.shape)
#简单查看数据
print(train.head())
#查看数据是否缺失
print(train.info(verbose=True,null_counts=True))
print(train_abbr.columns)#查看 train_abbr 所有特征列
print(train.columns)#查看 train 所有特征列


# ## 2.2 处理原始数据

# ### 2.2.1 处理训练集数据

# In[5]:


# 对训练集（train）处理
# 已有 edu_status 属性，这一列大部分是空，去掉
train = train.drop(['edu_other'],axis = 1)
# 家中有多少个 18 岁以下的孩子，没填视为没有，填 0
train['minor_child'] = train['minor_child'].fillna(value=0)
list1 = []
train['invest_other'] = train['invest_other'].fillna(value=-1)
# 对 invest 独热编码，01 的组合看作二进制数，转整数
```

```
for a,b,c,d,e,f,g,h,i,j in zip(train['invest_0'],
    train['invest_1'],train['invest_2'],
    train['invest_3'],train['invest_4'],
    train['invest_5'],train['invest_6'],
    train['invest_7'],train['invest_8'],train['invest_other']):
    item = 0
    item = 1*a+2*b+4*c+8*d+16*e+32*f+64*g+128*h+256*i
    if(not('-1' in str(j))):
        item = 1+2+4+8+16+32+64+128
    list1.append(item)
# 添加此新属性为 invest
train.insert(71,'invest',list1)
# 删除原先的 invest
train = train.drop(['invest_0','invest_1','invest_2','invest_3','invest_5',
'invest_4','invest_6','invest_7','invest_8','invest_other'],1)
list1 = train.keys()
# 根据 index 的说明，空数据视为不适用，填成-1
for item in list1:
    train[item] = train[item].fillna(value=-1)

# ### 2.2.2 处理测试集数据

# In[6]:


# 对测试集（test）处理
# 已有 edu_status 属性，这一列大部分是空，去掉
test = test.drop(['edu_other'],axis = 1)
# 家中有多少个 18 岁以下的孩子，没填视为没有，填 0
test['minor_child'] = test['minor_child'].fillna(value=0)
list1 = []
# 对 property 独热编码，01 的组合看作二进制数，转整数
for a,b,c,d,e,f,g,h,i,j in zip(test['property_0'],
    test['property_1'],test['property_2'],
    test['property_3'],test['property_4'],
    test['property_5'],test['property_6'],
    test['property_7'],test['property_8'],test['property_other']):
    item = 0
```

```
        item = 1*a+2*b+4*c+8*d+16*e+32*f+64*g+128*h+256*i
        if('共' in str(j)):
            print(a,b,c,d,e,f,g,h,i,j)
            item = 1+2+4+8+16+32+64+128
        list1.append(item)
    # 添加此新属性为 property
    test.insert(20,'property',list1)
    # 删除原先的 property
    test = test.drop(['property_0','property_1','property_2','property_3',
    'property_5','property_4','property_6','property_7','property_8',
    'property_other'],1)
    list1 = []
    test['invest_other'] = test['invest_other'].fillna(value=-1)
    # 对 invest 独热编码，01 的组合看作二进制数，转整数
    for a,b,c,d,e,f,g,h,i,j in zip(test['invest_0'],
        test['invest_1'],test['invest_2'],
        test['invest_3'],test['invest_4'],
        test['invest_5'],test['invest_6'],
        test['invest_7'],test['invest_8'],test['invest_other']):
        item = 0
        item = 1*a+2*b+4*c+8*d+16*e+32*f+64*g+128*h+256*i
        if(not('-1' in str(j))):
            item = 1+2+4+8+16+32+64+128
        list1.append(item)
    # 添加此新属性为 invest
    test.insert(71,'invest',list1)
    # 删除原先的 invest
    test = test.drop(['invest_0','invest_1','invest_2','invest_3','invest_5',
    'invest_4','invest_6','invest_7','invest_8','invest_other'],1)
    list1 = test.keys()
    # 根据 index 的说明，空数据视为不适用，填成-1
    for item in list1:
        test[item] = test[item].fillna(value=-1)


    # ## 2.3 处理 happiness 属性

    # In[7]:
```

```
6
7
8   #查看 label 分布
9   y_train_=train["happiness"]
0   # y_train_.value_counts()
1   print(y_train_.value_counts())
2   #将含有-8 的行直接去掉
3   print(train.index[0])
4   for lin,q in zip(range(0,len(y_train_)),y_train_):
5       if(q == -8):
6           train = train.drop(index=[lin])
7   # y_train_=y_train_.map(lambda x:3 if x==-8 else x)
8   #让 label 从 0 开始
9   y_train_=y_train_.map(lambda x:x-1)
0   #train 和 test 连在一起
1   data = pd.concat([train,test],axis=0,ignore_index=True)
2   #全部数据大小
3   print(data.shape)
4
5
6   # ## 2.4 处理时间特征
7
8   # In[8]:
9
0
1   #处理时间特征
2   data['survey_time']                                                    =
      pd.to_datetime(data['survey_time'],format='%Y-%m-%d %H:%M:%S')
3   data["weekday"]=data["survey_time"].dt.weekday
4   data["year"]=data["survey_time"].dt.year
5   data["quarter"]=data["survey_time"].dt.quarter
6   data["hour"]=data["survey_time"].dt.hour
7   data["month"]=data["survey_time"].dt.month
8
9
0   # In[9]:
1
2
```

```
# 把一天的时间分段
def hour_cut(x):
    if 0 <= x < 6:
        return 0
    elif 6 <= x < 8:
        return 1
    elif 8 <= x < 12:
        return 2
    elif 12 <= x < 14:
        return 3
    elif 14 <= x < 18:
        return 4
    elif 18 <= x < 21:
        return 5
    elif 21 <= x < 24:
        return 6

data["hour_cut"] = data["hour"].map(hour_cut)

# 做问卷时候的年龄
data["survey_age"]=data["year"]-data["birth"]


# ## 2.5 处理其他特征

# In[10]:


# 让 label 从 0 开始
data["happiness"]=data["happiness"].map(lambda x:x-1)

data=data.drop(["happiness"], axis=1) # 需要预测的结果，去掉一列
data=data.drop(["survey_time"], axis=1) # 已处理成 survey_age 属性，删去

# 是否入党
data["join_party"]=data["join_party"].map(lambda x:0 if x == -1  else 1)


```

```
# ## 2.6 查看数据并保存到 csv 文件

# In[11]:


data.head(40) # 查看处理完的数据


# In[12]:


path = "data_1126.csv"
data.to_csv(path,sep=",",index=False,header=True)


# # 三、数据可视化分析

# In[65]:


#绘韧 happiness 条形图
train_type = train.happiness.value_counts()
x = train_type.index
y = train_type.values
plt.figure(figsize=(12,8))
#条形图
plt.bar(x,y)
#设置标题、横轴、纵轴
plt.title('Happiness',fontsize=20)
plt.xlabel('Happiness',fontsize=20)
plt.ylabel('Counts',fontsize=20)
#坐标轴
plt.tick_params(labelsize = 10)
#设置数字标签
for a,b in zip(x,y):
    plt.text(a,b+100,b,ha= 'center', va = 'bottom')
plt.savefig('happiness.png')
plt.show()
```

```
9
0   #绘制 happiness 饼图
1   plt.figure(figsize =(20,10))
2   train_type = train.happiness.value_counts()
3   plt.pie(train_type,labels = train_type.index, autopct='%.2f%%', shadow =
     False)
4   plt.title('happiness',fontsize=20)
5   plt.axis('equal')
6   plt.legend()
7   plt.show()
8
9   #绘制 happiness 与其他指标相关系数
0   plt.xlim((0, 30))
1   plt.figure(figsize =(20,5))
2   plt.xticks(rotation=270)
3   data                                                                    =
     pd.read_csv('./Dataset/happiness_train_complete_washed.csv',encoding='gbk')
4   name = data.columns.values.tolist()
5   name.remove("happiness")
6   happ = pd.Series(data['happiness'])
7   corr = []
8   for i in name:
9       q = data[i]
0       s1 = pd.Series(q)
1       try:
2           corr.append(happ.corr(s1))
3       except TypeError:
4           corr.append(0)
5   corr_sort = np.argsort(corr) #按降序排列
6   corrlen = len(corr_sort)
7   corr_max10 = []
8   for j in range(corrlen - 10, corrlen):
9       corr_max10.append(name[corr_sort[j]])
0   #输出相关系数前 10 的属性
1   for k in range(len(corr_max10)):
2       print(corr_max10[len(corr_max10) - k - 1])
3   ln1= plt.plot(name,corr,color='red',linewidth=2.0,linestyle='-')
4   plt.title("Correlation") #设置标题及字体
```

```
ax = plt.gca()
ax.spines['right'].set_color('none')  # right 边框属性设置为 none 不显示
ax.spines['top'].set_color('none')    # top 边框属性设置为 none 不显示
plt.savefig('corr.png')
plt.show()


# # 四、模型训练

# ## 4.1 划分数据集

# In[42]:


data=data.drop(["id"], axis=1)
X_train_ = data[:train.shape[0]] # 训练集
X_test_  = data[train.shape[0]:] # 测试集
target_column = 'happiness'
feature_columns=list(X_test_.columns)
feature_columns


# ## 4.2 转 numpy 数组

# In[46]:


X_train = np.array(X_train_)
y_train = np.array(y_train_)
X_test  = np.array(X_test_)


# In[47]:


print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
```

```
3
4
5   # ## 4.3 自定义评价函数
6
7   # In[48]:
8
9
0   #自定义评价函数
1   def myFeval(preds, xgbtrain):
2       label = xgbtrain.get_label()
3       score = mean_squared_error(label,preds)
4       return 'myFeval',score
5
6
7   # ## 4.4xgb
8
9   # In[49]:
0
1
2   ##### xgb
3
4   xgb_params = {"booster":'gbtree','eta': 0.005, 'max_depth': 5, 'subsample':
     0.7,
5                 'colsample_bytree': 0.8, 'objective': 'reg:linear',
6                 'eval_metric': 'rmse', 'silent': True, 'nthread': 8}
7   folds = KFold(n_splits=5, shuffle=True, random_state=2018)
8   oof_xgb = np.zeros(len(train))
9   predictions_xgb = np.zeros(len(test))
0
1   for fold_, (trn_idx, val_idx) in enumerate(folds.split(X_train, y_train)):
2       print("fold n° {}".format(fold_+1))
3       trn_data = xgb.DMatrix(X_train[trn_idx], y_train[trn_idx])
4       val_data = xgb.DMatrix(X_train[val_idx], y_train[val_idx])
5
6       watchlist = [(trn_data, 'train'), (val_data, 'valid_data')]
7       clf = xgb.train(dtrain=trn_data, num_boost_round=20000, evals=watchlist,
8       early_stopping_rounds=200,  verbose_eval=100,  params=xgb_params,feval  =
     myFeval)
```

```
    oof_xgb[val_idx] = clf.predict(xgb.DMatrix(X_train[val_idx]),
    ntree_limit=clf.best_ntree_limit)
    predictions_xgb += clf.predict(xgb.DMatrix(X_test),
    ntree_limit=clf.best_ntree_limit) / folds.n_splits

print("CV score: {:<8.8f}".format(mean_squared_error(oof_xgb, y_train_)))


# ## 4.5lgb

# In[50]:


##### lgb

param = {'boosting_type': 'gbdt',
          'num_leaves': 20,
          'min_data_in_leaf': 20,
          'objective':'regression',
          'max_depth':6,
          'learning_rate': 0.01,
          "min_child_samples": 30,

          "feature_fraction": 0.8,
          "bagging_freq": 1,
          "bagging_fraction": 0.8 ,
          "bagging_seed": 11,
          "metric": 'mse',
          "lambda_l1": 0.1,
          "verbosity": -1}
folds = KFold(n_splits=5, shuffle=True, random_state=2018)
oof_lgb = np.zeros(len(X_train_))
predictions_lgb = np.zeros(len(X_test_))

for fold_, (trn_idx, val_idx) in enumerate(folds.split(X_train, y_train)):
    print("fold n° {}".format(fold_+1))
    # print(trn_idx)
    # print(".............x_train.........")
```

```
    # print(X_train[trn_idx])
    #  print(".............y_train.........")
    #  print(y_train[trn_idx])
      trn_data = lgb.Dataset(X_train[trn_idx], y_train[trn_idx])

      val_data = lgb.Dataset(X_train[val_idx], y_train[val_idx])

      num_round = 10000
      clf = lgb.train(param, trn_data, num_round, valid_sets = [trn_data, val_data],
      verbose_eval=200, early_stopping_rounds = 100)
      oof_lgb[val_idx] = clf.predict(X_train[val_idx],
      num_iteration=clf.best_iteration)

      predictions_lgb += clf.predict(X_test, num_iteration=clf.best_iteration) / folds.n_splits

print("CV score: {:<8.8f}".format(mean_squared_error(oof_lgb, y_train_)))


# ## 4.6ctb

# In[51]:


from catboost import Pool, CatBoostRegressor
from sklearn.model_selection import train_test_split

kfolder = KFold(n_splits=5, shuffle=True, random_state=2019)
oof_cb = np.zeros(len(X_train_))
predictions_cb = np.zeros(len(X_test_))
kfold = kfolder.split(X_train_, y_train_)
fold_=0
for train_index, vali_index in kfold:
    print("fold n° {}".format(fold_))
    fold_=fold_+1
    k_x_train = X_train[train_index]
    k_y_train = y_train[train_index]
```

```
    k_x_vali = X_train[vali_index]
    k_y_vali = y_train[vali_index]
    cb_params = {
        'n_estimators': 100000,
        'loss_function': 'RMSE',
        'eval_metric':'RMSE',
        'learning_rate': 0.05,
        'depth': 5,
        'use_best_model': True,
        'subsample': 0.6,
        'bootstrap_type': 'Bernoulli',
        'reg_lambda': 3
    }
    model_cb = CatBoostRegressor(**cb_params)
    #train the model
    model_cb.fit(k_x_train, k_y_train,eval_set=[(k_x_vali, k_y_vali)],
    verbose=100,early_stopping_rounds=50)
    oof_cb[vali_index] = model_cb.predict(k_x_vali,
 ntree_end=model_cb.best_iteration_)
    predictions_cb += model_cb.predict(X_test_,
 ntree_end=model_cb.best_iteration_) / kfolder.n_splits



print("CV score: {:<8.8f}".format(mean_squared_error(oof_cb, y_train_)))


# ## 4.7 模型融合

# In[52]:


from sklearn import linear_model
# 将 lgb 和 xgb 和 ctb 的结果进行 stacking
train_stack = np.vstack([oof_lgb,oof_xgb,oof_cb]).transpose()
test_stack = np.vstack([predictions_lgb,
 predictions_xgb,predictions_cb]).transpose()

```

```
8
9   folds_stack = RepeatedKFold(n_splits=5, n_repeats=2, random_state=2018)
0   oof_stack = np.zeros(train_stack.shape[0])
1   predictions = np.zeros(test_stack.shape[0])
2
3   for fold_, (trn_idx, val_idx) in
      enumerate(folds_stack.split(train_stack,y_train)):
4       print("fold {}".format(fold_))
5       trn_data, trn_y = train_stack[trn_idx], y_train[trn_idx]
6       val_data, val_y = train_stack[val_idx], y_train[val_idx]
7
8       clf_3 = linear_model.BayesianRidge()
9       #clf_3 =linear_model.Ridge()
0       clf_3.fit(trn_data, trn_y)
1
2       oof_stack[val_idx] = clf_3.predict(val_data)
3       predictions += clf_3.predict(test_stack) / 10
4
5   print("CV score: {:<8.8f}".format(mean_squared_error(oof_stack, y_train_)))
6
7
8   # # 五、预测结果
9
0   # In[53]:
1
2
3   result=list(predictions)
4   result=list(map(lambda x: x + 1, result))
5   test_sub["happiness"]=result
6   test_sub.to_csv("submit_20201122_2.csv", index=False)
```