1. 具体实施

对于本次的幸福感预测任务本质上是一个回归问题，我们采用了决策树作为核心算法，挑选了三个强大的模型：XGB，LGB，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。另外，我们在将数据送入模型训练之前还根据数据集本身的特征对数据集进行了一定的预处理，挖掘了潜在特征扩展了数据集的同时降低了数据集的维度，提升了算法效率。下面，将介绍数据预处理的具体方式、三种模型的详细细节以及模型融合算法的实现过程。

1. . 数据预处理

本项目中共有多个数据文件，数据时间的的有效时间为2015年，包括参考类文件：调查问卷原卷happiness\_survey\_cgss2015.pdf、问卷数据的数据目录索引happiness\_index.csv；数据类文件：精简版测试、训练数据集happiness\_test\_abbr.csv、happiness\_train\_abbr.csv；完整版测试、训练数据集：happiness\_test\_complete.csv、happiness\_train\_complete.csv。

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。

数据集本身不规整，有空缺数据和不合理输入。通过pandas、numpy库进行预处理，把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。

通过直观观察问题可以发现，整体可以划分为以下几个内容：年代、信仰、受教育程度、政治面貌、房产、业余生活、财富等十余个方面。根据该情况，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。同时把日期更改为可以进行分析的数据类型。

（二）. 模型详解

1. XGB

1.1 GBDT梯度提升树

Xgb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做好的算法之一，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。

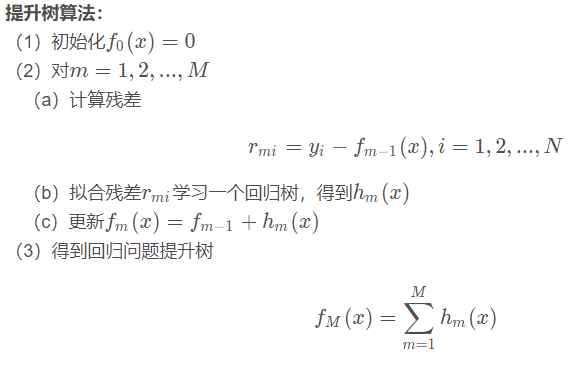


图 提升树算法伪代码

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。

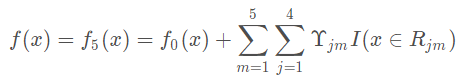


图 生成强分类器公式

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。

1.3 XGB模型【4】

XGB[2]模型的逻辑和GBDT模型是一样的，而XGB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。

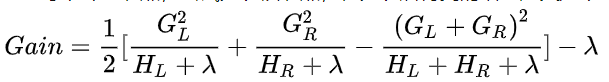


图 线性扫描寻找最佳分裂

总结算法流程：a.迭代生成树；b.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；c.使用目标函数采用贪心算法求当前树；d.新增树后，模型更新；e.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

1. Lightgbm

2.1 算法概述

前文说到GBDT和XGB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。

2.2 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算法的流程图如下图所示：

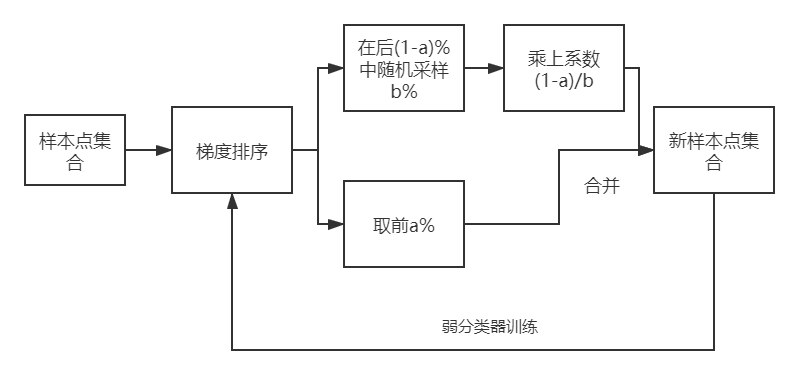


图 GOSS算法流程图

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。

流程图如下图所示：



图 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，如下图所示。



图 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。

1. Catboost

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。

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。

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。

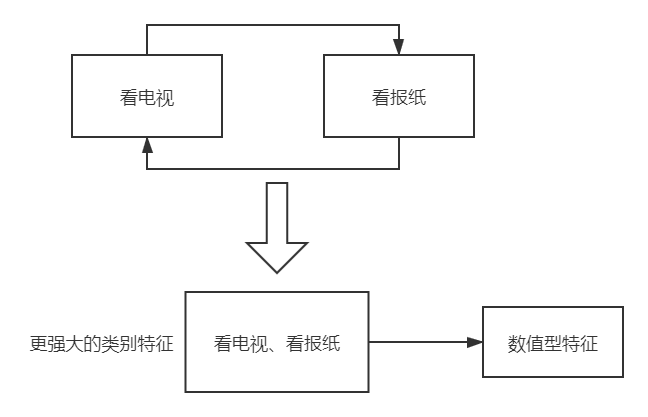


图 合并特征示意图

1. 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XGB，LGB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。

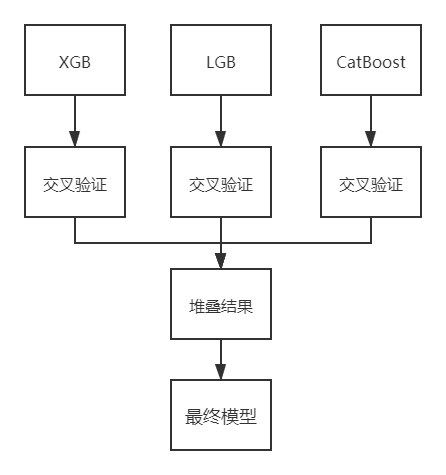


图 模型融合流程图

（三）. 实验过程

1. 使用参数
   1. Xgboost

|  |  |  |
| --- | --- | --- |
| 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 |

* 1. LightGBM

|  |  |  |
| --- | --- | --- |
| 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 |

* 1. CatBoost

|  |  |  |
| --- | --- | --- |
| 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 |

1. 实验结果

幸福感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,在所有约5000个参赛队伍中排名221名。



图 最终结果与排名

【参考文献】

1. Jerome 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.

【参考网站】

1. CART源码：<https://blog.csdn.net/weixin_40479663/article/details/84781500>
2. CART源码解析：<https://zhuanlan.zhihu.com/p/32164933>
3. GBDT算法解析：<https://blog.csdn.net/horizonheart/article/details/78782622>
4. XGB：<https://www.cnblogs.com/bnuvincent/p/9693190.html>