Kaggle销售预测—XGBoost算法应用

彭晴1 申鸿雨2

1西南财经大学统计学院2018级经济统计学 41809024

2西南财经大学统计学院2018级经济统计学 41809021

摘要 挖掘隐藏在异质信息网络中丰富的语义信息是数据挖掘的重要任务之一．离群点在值、数据分布、和产生（待写）

关键词 销售预测；聚类；XGBoost

# 1整体概要

## 1.1问题阐述

比赛提供了俄罗斯电商公司1C从2013年1月至2015年10月的每日销售数据，包含了商品信息、店铺信息、商品分类等，要求利用这些数据预测下一个月每家商店售卖每种商品的月销量。

## 1.2研究意义

通过数据预处理和模型构建，获取训练集样本中有用的信息，对未来一个月的销售情况做出预测，能够帮助电商更好准备商品和精准营销销售，提高利润空间。

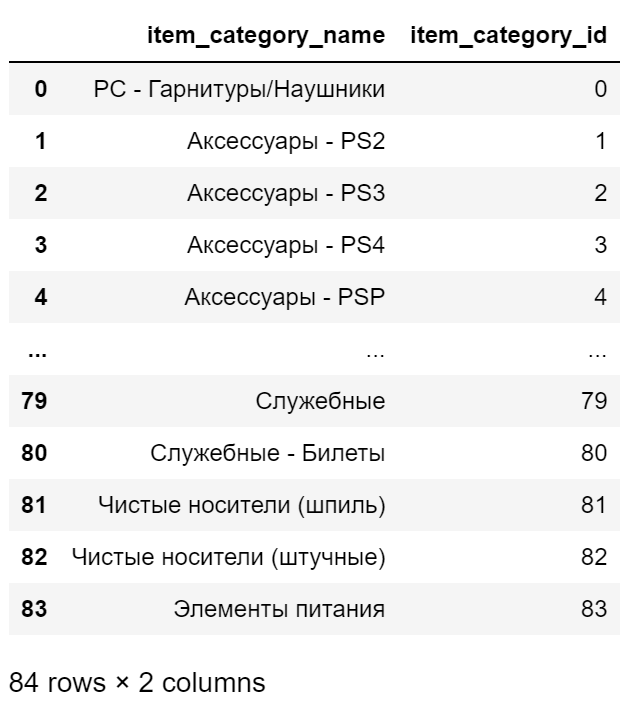
通过不断的尝试和调整，我们最终找到对我们的模型最优的参数设置，在10000余支队伍中获得前7%的成绩。

## 1.3研究思路

首先，我们对源数据进行预处理，清洗掉和填充异常值，并根据实际意义和信息价值进行分类和添加特征，构造完整和整齐的分类数据和计数值，对不同特征数据进行标准化之后，构建了XGBoost机器学习模型，手动尝试调整参数，根据最小均方 误差值判断预测结果准确度。 最后选取最优参数组合，得到结果。

# 2数据概况

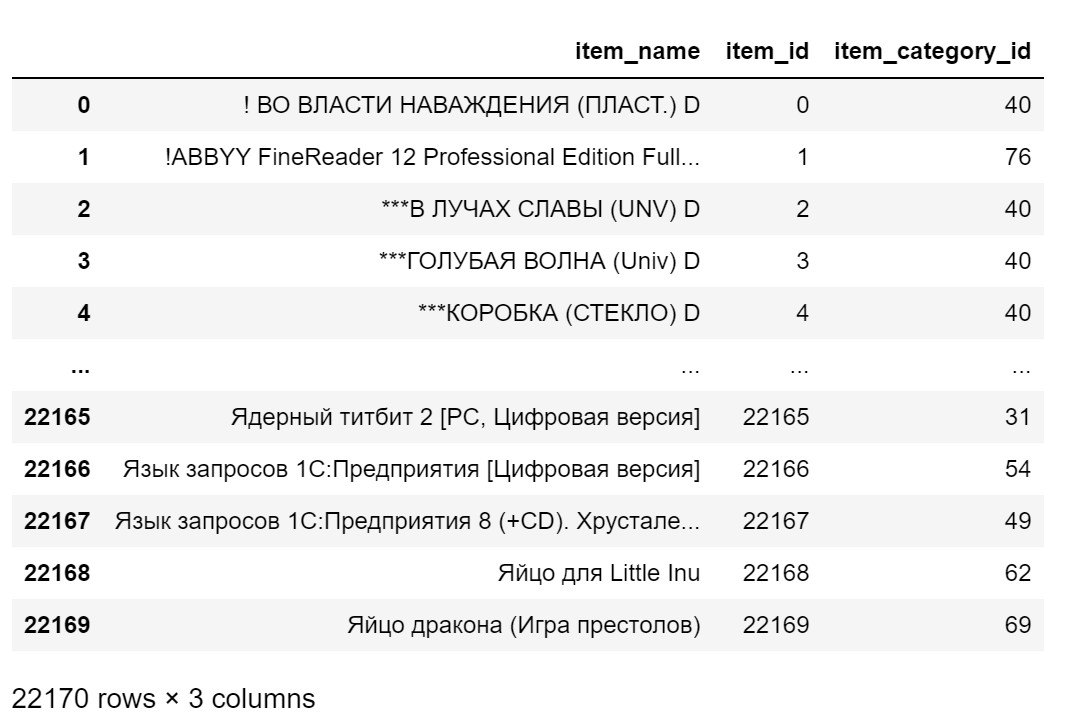
（1）item\_categories.csv



图表 1

分别描述了商品分类的全称和ID，全程中分隔符“-”前是商品品类，如“书籍”“礼品”等，“-”后为商品的具体型号。共有84个商品类别，根据中文翻译的实际含义，其中存在一些相似的品类命名。

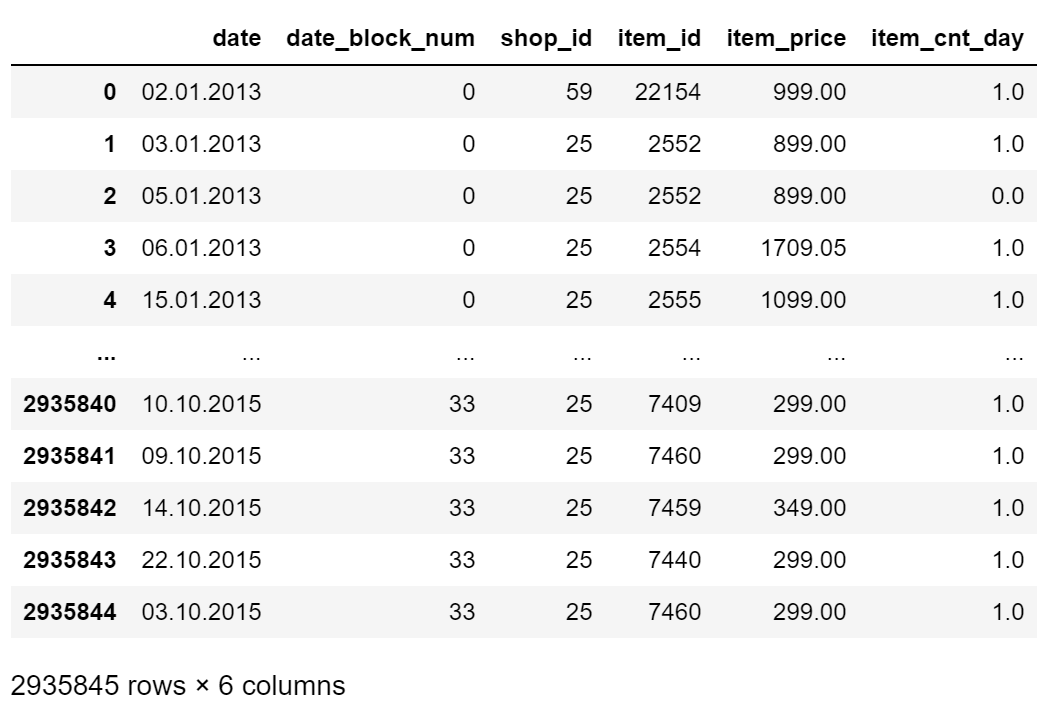
（2）item.cs



图表 2

商品全称item\_name中，“！”后跟的是商品的售卖地城市，“[]”中表示的是商店名称，观察发现存在相同商店名称与不同城市的组合，认为是同一家商店在不同城市的分店。

（3）salestrain.csv



图表 3

记录了每家商店每天售卖每种商品的信息和数量。

（4）shop\_name.csv



图表 4

上表截取了商品名称存在高度相似的几家店铺信息，并将0号店铺ID更改为57，1号店铺ID更改58，10号店铺ID更改为11。



图表 5

观察发现shop的命名规则为“(!+)城市+商店主营商品大类+后缀”，可借此提取新的特征。

# 2模型调整

## 2.1模型简介

XGBoost属于Boosting算法的一种，将多个弱估计器集成形成强估计器从而获得更强大的预测能力。我们选择XGBoost的理由主要如下：

1.与GBDT相比，XGBoost有更多的策略去防止过拟合，

2.XGBoost是目前机器学习算法中训练速度快较快的一种，在同层节点中支持并行化并且树与树之间是串行关系。

3.对于稀疏数据处理更具有优势。

4.由于本次收集数据大多有缺失值，而XGBoost能够自动化处理缺失值，使结果更具有鲁棒性。

基于上述优点，文本选取XGBoost作为预测模型。

## 2.2核心思想

不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到k棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数。最后只需要将每棵树对应的分数加起来就是该样本的预测值。

## 2.3模型建立

本模型为集成算法模型。

XGBoost是GBDT的扩展与改进，同属于boosting算法。而其本质是集成多个弱分类器，建立起多个相关联的决策树进行联合决策，并且每一颗决策树的输入样本将与前置决策树有关。

XGBoost的目标函数为：

其中，用于描述实际值和预测值之间的差异,为T模型的正则化项，T表示叶子结点数量,ω表示叶子结点的分值，γ用来控制叶子结点的个数，用来控制叶子结点的分值，用于避免模型过拟合。

在XGBoost中，每次的学习过程中总会依赖前轮的效果，因此在上述公式中加入t来表示与前t-1轮的结果相关。

为了使目标函数，也即使预测值与实际值差异最小，XGBoost的思想是利用其 =0的二阶泰勒展开式：

上式中由于只受函数f的影响，因此可以看做常数项，故直接删除。

又由于表示的是一个叶子结点，故可用来替换，于是目标函数可变化为：

求偏导可得到该目标函数的最优权重，将其带回到单个树中可得：

以上推导即为树结构的确定。

由此，提高精度和防止过拟合使我们调整参数的出发点。

测试集和训练集

题目一共提供了34个月的销售数据，我们将前33个月作为训练集进行拟合，将第34个月作为验证集，检验模型的精度，将需要预测的第35个月作为测试集。

第35个月中有5100间商品和42个商店，共有2142条数据需要预测

# 3参数调整

我们使用的是手动调参的方法，这需要一定的调参经验结合损失函数的变化。我们根据每次运行的得分作为依据调整参数变动方向，我们使用验证集进行实验，以找到理想的拟合效果。选的的评价标准是score,数值上等于均方误差，当score值越高时，说明精度越低。

又由于调参顺序也会会影响调参结果。所以一般会优先调对模型影响较大的参数。于是我们选择先调整n\_estimators和early\_stopping\_rounds，然后是max\_depth，最后是min\_chiild\_weight。

## 3.1 n\_estimators

在欠拟合与过拟合图中，该估计量太低的值会导致欠拟合，即对训练数据和新数据的预测都不准确。太大的值会导致过度拟合，即对训练数据的准确预测，但对新数据的不准确预测。这在很大程度上取决于下面讨论的学习率。

在我们的实验中，首先从1000开始在其他参数保持不变的情况下，该估计量从1000到1300到1500的过程中，得分保持一个下降的趋势，表明精度在不断升高，说明此前可能存在欠拟合。再变到1800时，精度开始下降，此时可能出现了过拟合，于是我们选择1500作为该参数的最优值。

## 3.2 early stopping rounds

该参数提供了一种自动找到理想值的方法。当验证分数停止改善时，提前停止会导致模型停止迭代。防止模型过拟合，比较好的组合是，为迭代次数估计量设置一个高值，然后使用early\_stopping\_rounds来找到停止迭代的最佳时间。

在我们的尝试中，我们选择20作为开局点，并且与迭代次数想组合，得到当迭代次数为1500，早停轮为20时得到较高的精度。

3.3learning\_rate

对于更好的XGBoost模型，即高精度与防止过拟合，不能简单地将每个组件模型的预测相加，而是将每个模型的预测乘以一个小数字，然后再相加。这意味着添加到集合中的每一棵树提供的信息更少。实际上，这降低了模型过度拟合的倾向。

因此，可以使用更高的迭代次数，而无需过度拟合。一般来说，较小的学习率（和大量的估计器）将产生更精确的XGBoost模型，尽管由于模型在整个周期中进行了更多的迭代，因此也需要更长的时间来训练。

在我们的尝试中，我们选择从0.05开始，增加到0.1时，精度陡然下降，表明存在严重的过拟合，变到0.03时，精度相比0.05还是下降，表明此时为欠拟合。于是我们确定0.05为合适的值。

## 3.4 max\_depth

树的最大深度是决策树中的剪枝法宝，算是最常用的剪枝参数，这个值也是用来避免过拟合的。

max\_depth越大，模型会学到更具体更局部的样本。不过在XGBoost中，最大深度的功能与参数gamma相似，因此如果先调节了gamma，则最大深度可能无法展示出巨大的效果。通常来说，这两个参数中只使用一个，我们选择了最大深度这个参数。

在我们的尝试中，首先选择了10，当参数由10分别变到13和7时，精度都下降了，表明此时分别存在过拟合与欠拟合的问题。于是我们选择10作为合适的值。

## 3.5 min\_child\_weight

叶子节点中最小的样本权重和。如果一个叶子节点的样本权重和小于min\_child\_weight则拆分过程结束。在现行回归模型中，这个参数是指建立每个模型所需要的最小样本数。该参数越大算法越趋于保守。越小，极端情况为一个叶子节点只有一个样本，此时容易学到非常具体的特征，产生过拟合的问题。

在我们的尝试中，首先选择0.5作为起点，当参数分别由0.5变到1和0.3的，精度下降，表明此时模型可能分别存在欠拟合与欠拟合，于是我们选择0.5作为合适的值。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| max\_depth | n\_estimators | early\_stopping\_rounds | min\_child\_weight | learning rate | score |
| 10 | 1000 | 20 | 0.5 | 0.05 | 0.89868 |
| 10 | 1300 | 20 | 0.5 | 0.05 | 0.89668 |
| 10 | 1300 | 10 | 0.5 | 0.05 | 0.89668 |
| 10 | 1500 | 20 | 0.5 | 0.05 | 0.8896 |
| 10 | 1300 | 5 | 0.5 | 0.05 | 0.90736 |
| 10 | 1500 | 25 | 0.5 | 0.05 | 0.89868 |
| 10 | 1800 | 25 | 0.5 | 0.05 | 0.8896 |
| 10 | 1800 | 30 | 0.5 | 0.05 | 0.89668 |
| 10 | 1500 | 10 | 0.5 | 0.05 | 0.89668 |
| 10 | 1500 | 20 | 0.5 | 0.1 | 0.90094 |
| 10 | 1500 | 20 | 0.5 | 0.03 | 0.89716 |
| 10 | 1800 | 20 | 0.5 | 0.03 | 0.89716 |
| 13 | 1500 | 20 | 0.5 | 0.05 | 0.9012 |
| 7 | 1500 | 20 | 0.5 | 0.05 | 0.89911 |
| 10 | 800 | 20 | 0.5 | 0.05 | 0.89668 |
| 10 | 1500 | 20 | 0.3 | 0.05 | 0.89668 |
| 10 | 1500 | 20 | 0.8 | 0.05 | 0.8896 |
| 10 | 1500 | 20 | 1 | 0.05 | 0.89668 |

图表 6

# 4反思

## 4.1不足

试验次数少回顾整个比赛过程，我们发现了不少值得改进的地方。

首先，我们的试验次数太少了，由于是手动调参，非常耗时，电脑也哒哒哒的，再加上Kaggle平台每支队伍每天只能提交5次结果，导致最终我们的试验次数不到20份，一些重要的参数没有进行更细致的调试。只有一个粗略的范围。

其次我们参数的组合不太合理，

我们选择“控制变量法”的原理，即每次只调整一个或两个参数变化，其他参数保持不变。而且调整参数是基于上一次的结果，在本次比赛中，由于我们在第4次实验得到了最高的精度，之后的每次实验都是基于第四次实验进行。这对新手很不友好，往往导致局部最优，而全局不乐观。

## 4.2改进

针对出现的问题，再加上我们只选择了4个超参数，网格搜索或许是不错的解决方案、通过4个参数之间的组合，可以解决第二个问题，同时它能够根据我们设置的参数范围，自动调参，也能解决手动调参的不足，希望在未来，我们能尝试加入这个方法。