结合 GBDT 与 Prophet 的沃尔玛销量预测方案

沙斌竹　计算机技术　2022214259　sbz22@mails.tsinghua.edu.cn

李洪君　建筑学　2022214144　lihj22@mails.tsinghua.edu.cn

摘要

本文首先对表格数据相关的模型设计按类别做出了文献综述，并且从现象、原因和数据方面在表格数据上对深度学习与传统机器学习进行了对比分析。……

关键词: kaggle 竞赛, 销量预测；梯度提升树；时序模型

Abstract

This paper begins with a literature review of model design related to tabular data by category, and a comparative analysis of deep learning versus traditional machine learning on tabular data in terms of phenomena, causes and data. ……

Keywords: kaggle, GBDT, sales forecasting, time series model

引言

沃尔玛销量预测本质上是对时间序列进行处理分析，其数据集类型属于表格数据 (Tabular Data)，尽管深度学习在图像、语言等领域有各类性能强大的算法模型，但在表格数据的背景之下，传统机器学习反而能够起到更好的效果。

基于经典的决策树模型，通过集成学习方法提出的梯度提升树模型 [1] (Gradient Boosting Decision Tree) 具有低偏差和强拟合能力，加入随机森林的采样思想后在结构化数据领域性能相当显著。在深度学习领域，也通过结合注意力机制、集成决策树和自动化机器学习等方法能够在表格数据上取得良好的效果。本文将 GBDT 模型作为基础，结合时序预测模型 Prophet 共同对沃尔玛销量进行预测。

1. 文献综述

1.1. 表格数据模型

除了图像和自然语言等数据，表格数据也在数据科学领域也是最为常见的类型之一。从传统的机器学习方法到如今结合 Transformer 模块的深度学习方法，都对表格数据有着不同的探索和应用。在本节文献综述中我们将按照类别依次介绍表格数据的模型设计。

机器学习. 基于树的梯度提升算法由于具有低偏差和强拟合能力对表格数据有较好的效果。GDBT（Gradient Boosting Decision Tree）属于传统机器学习算法，以 CART 回归树为基础单元，在加入了随机森林的采样思想后，在方差和偏差的方面都具有很好的表现，在结构化数据领域性能更为显著。并且该模型能够较好地处理非线性问题以及能够适应多种损失函数，同时具有相对较好的可解释性 [1]。

Xgboost 是基于决策树的集成机器学习算法，大体上沿袭了之前说过的 GBDT 的框架，区别在于将引入学习率缓解模型过拟合改为了引入树的正则化方法，并且相比 GBDT 使用了一、二阶梯度的信息，同时使用了多种近似分裂方式从而加速了建树的过程 [2]。

LightGBM 是一个轻量级的梯度提升机，作为工程实现 GBDT 算法的框架能够快速处理大量的数据并且支持高效率的并行训练。主要结合 Histogram 算法用以减少候选分裂点位数量、由 GOSS 算法的作用是减少样本的数量以及 EFB 算法的作用是减少特征的数量 [3]。

深度学习. 尽管受限于数据特征和模型原理，深度学习更适合于应用在图像和自然语言领域。但在表格数据领域中深度学习算法一直不断改进与探索。

FastAI 封装了专门的应用模块来处理表格数据，使用的是传统的神经网络算法；而 TabTransformer 建立在基于自注意力的 Transformer 之上，是用于对表格数据进行监督和半监督学习的深度学习架构。其中 Transformer 层将分类特征的嵌入转换成了更为稳健的上下文嵌入，以实现更高的预测精度。[4]

TabNet[5]是 Google 提出的针对表格数据设计的深度学习模型，同时将树模型和神经网络进行应用，因此既具有较好的可解释性和稀疏特征选择能力，能够量化每个输入特征在训练模型中的参与程度；又有表征学习和端到端训练的优点，在 TabNet 中不需要输入原始表格特征和进行任何特征预处理工作。

与 TabNet 相似的还有 NODE[6]，作为处理表格数据的深度学习框架，既使用了集成遗忘决策树（见图1），又具备基于梯度的端到端优化和多层分层表示学习的能力。

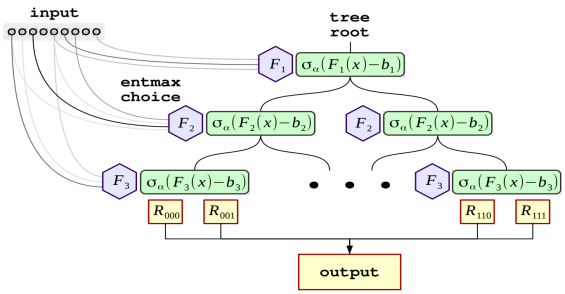

图 1. NODE 与集成决策树的相似性 [6]

在自动化机器学习方面，AutoInt[7]可以高效地学习输入特征的高维特征交互，能够应用于数值和分类输入特征。具体而言是讲数值特征和分类特征映射到同一个低维空间。之后，通过一种具有残差连接的多头自注意神经网络来显式建模低维空间中的特征交互，多头自注意力在神经网络的不同层可以对输入特征的不同顺序的特征组合进行建模（见图2）。

Gorishniy 以表格数据为基础讨论了深度学习在该数据类型中的应用 [8]。主要实践了 MLP、ResNet 和 FT-Transformer 三大类深度学习模型，并通过计算

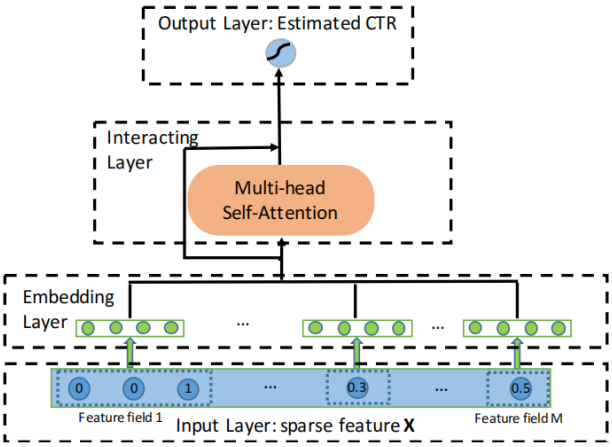

图 2. AutoInt 核心网络结构 [7]

RMSE （见图3）与基于树的模型比较在表格数据中的表现差异。

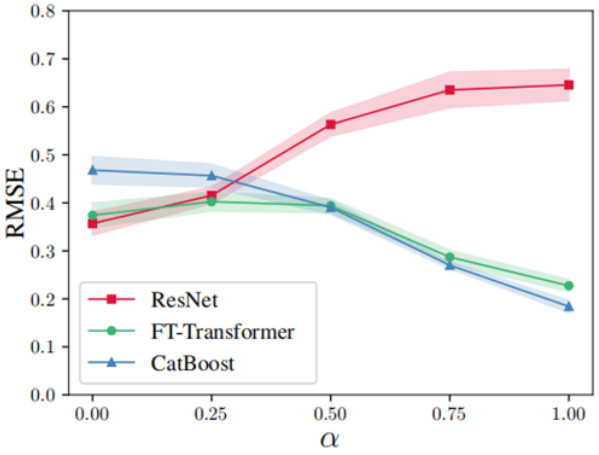

图 3. 不同模型的 RMSE[8]

1.2. 对比分析

现象. 深度学习在图像、语言等领域大放异彩，但在表格数据处理中效果不如传统机器学习算法。比如在 Shwartz-Ziv[9]的论文中，将深度学习模型和基于树的模型进行了性能上的对比（见图4），体现出了树模型强大的稳定性，易于优化训练的优点。

原理. 深度学习和传统机器学习在表格数据上的表现差异主要有以下三个原因：

1）在深度学习中，需要用到多个隐藏层才能够把输入空间变成线性可分割，并且数据集越复杂，需

Test results on tabular datasets. Presenting the performance for each model. MSE is presented for the YearPrediction and Rossman datasets, and cross-entropy loss (with 100X factor) is presented for the other datasets. The papers that used these datasets are indicated below the table. The values are the averages of four training runs (lower value is better), along with the standard error of the mean (SEM)

Model Name	Rossman	CoverType	Higgs	Gas	Eye	Gesture
XGBoost	490.18 ± 1.19	3.13 ± 0.09	21.62 ± 0.33	2.18 ± 0.20	**56.07** ±0.65	80.64 ± 0.80
NODE	488.59 ± 1.24	4.15 ± 0.13	21.19 ± 0.69	2.17 ± 0.18	68.35 ± 0.66	92.12 ± 0.82
DNF-Net	503.83 ± 1.41	3.96 ± 0.11	23.68 ± 0.83	**1.44** ±0.09	68.38 ± 0.65	86.98 ± 0.74
TabNet	**485.12**±1.93	3.01 ± 0.08	**21.14**±0.20	1.92 ± 0.14	67.13 ± 0.69	96.42 ± 0.87
1D-CNN	493.81 ± 2.23	3.51 ± 0.13	22.33 ± 0.73	1.79 ± 0.19	67.9 ± 0.64	97.89 ± 0.82
Simple Ensemble	488.57 ± 2.14	3.19 ± 0.18	22.46 ± 0.38	2.36 ± 0.13	58.72 ± 0.67	89.45 ± 0.89
Deep Ensemble w/o XGBoost	489.94 ± 2.09	3.52 ± 0.10	22.41 ± 0.54	1.98 ± 0.13	69.28 ± 0.62	93.50 ± 0.75
Deep Ensemble w XGBoost	485.33 ± 1.29	**2.99** ±0.08	22.34 ± 0.81	1.69 ± 0.10	59.43 ± 0.60	**78.93** ±0.73
		TabNet			DNF-Net	

Model Name	YearPrediction	MSLR	Epsilon	Shrutime	Blastchar
XGBoost	77.98 ± 0.11	55.43±2e−2	11.12±3e−2	13.82 ± 0.19	20.39 ± 0.21
NODE	76.39 ± 0.13	55.72±3e−2	**10.39**±1e−2	14.61 ± 0.10	21.40 ± 0.25
DNF-Net	81.21 ± 0.18	56.83±3e−2	12.23±4e−2	16.8 ± 0.09	27.91 ± 0.17
TabNet	83.19 ± 0.19	56.04±1e−2	11.92±3e−2	14.94± 0.13	23.72 ± 0.19
1D-CNN	78.94 ± 0.14	55.97±4e−2	11.08±6e−2	15.31 ± 0.16	24.68 ± 0.22
Simple Ensemble	78.01 ± 0.17	55.46±4e−2	11.07±4e−2	13.61± 0.14	21.18 ± 0.17
Deep Ensemble w/o XGBoost	78.99 ± 0.11	55.59±3e−2	10.95±1e−2	14.69 ± 0.11	24.25 ± 0.22
Deep Ensemble w XGBoost	**76.19** ±0.21	**55.38**±1e−2	11.18±1e−2	**13.10**±0.15	**20.18**±0.16
		NODE		New datasets	

图 4. 不同模型之间的效果对比 [9]

要的隐藏层就越多，变换过程很可能失败，反而让数据更加纠缠在一起。而即便成功了，相比梯度提升树的效率也很低。并且，即便人工创建了数据的特征，深度网络还是会再自己创建隐藏特征，然而表格数据本身的表头就是其数据特征。

2）神经网络偏向更平滑的解，在设计非平滑函数或者决策边界时，神经网络很难创造最适合的函数，而传统的随机森林能够在不规则模式下得到更好的效果 [10]。在不规则目标函数的数据集中，与基于决策树的模型相比，神经网络偏向于低频率函数从而难以拟合，因为梯度依赖可微的平滑搜索空间，所以无法区分不规则随机函数。而基于决策树的模型学习了分段常函数从而不会表现出这样的偏置。

3）表格数据包含许多非信息的特征，在神经网络中输入不相关的特征会浪费更多的资源训练模型，作者通过随机添加和删除非信息特征来比较树形模型和神经网络的性能差距，从图5可以看到，在数据集中添加随机特征使得神经网络衰退更为严重，类似 MLP 结构的神经网络在无信息的特征数据集并不够鲁棒。

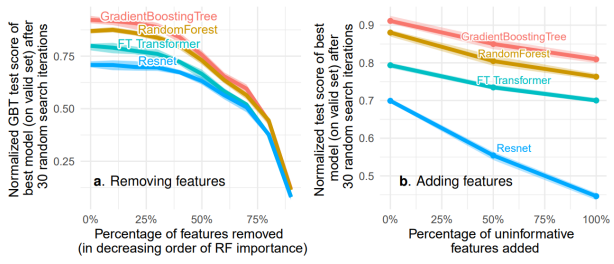

图 5. 改变特征前后的准确度测试 [10]

数据量. 在小数据集上深度学习容易过拟合，并且正则化的方法需要依赖许多条件。此外，参数越多的深度神经网络需要越大的数据量，如果提供的数据集有限并且数据维度比较低，就无法发挥深度学习应有的效果。

数据特点. 表格数据用 GBDT 这类机器学习算法能够得到较好表现，而深度学习适合譬如张量、图片、音频等非常大的非表格数据集。在数据特点上，除了上文提到的非信息特征之外，不同的深度网络本身也适用于不同的数据集，如 CNN 适合处理图像，RNN 适合处理特定的序列等。

2. 问题分析和评价指标

问题分析. 沃尔玛销量预测比赛官方提供了来自美国三个洲、三种产品类别、七个部门从 2011 年到 2016 年 3049 种产品的历史销量数据，参赛者需要根据该数据集预测 28 天的销量，属于时间序列预测问题，需要对庞大的数据集进行整理分析，并设计适用于表格数据的预测模型去解决问题。主要的几点问题分析：

1）数据集体量大且包含的因素复杂，同时还要考虑节假日对销量的营销，以及数据集本身存在许多零值，因此需要处理数据集分析和特征提取与构造的问题；

2）深度学习在表格数据上不一定比基于树方法的传统机器学习效果好，但深度学习能够表征学习以及端到端进行训练，所以存在设计合适的预测模型的问题；

3）需要讨论和实践单个模型相互之间的对比以及集成模型增加预测准确度，研究超参数的选取和对预测效果的影响；

评价指标. 沃尔玛销量预测要求参赛者分别完成点预测和概率预测的任务，销量预测的范围是 28 天，最终分数的计算首先要对每个系列进行单独计算然后再根据权重表取加权平均。

点预测. 该评价指标的准确性使用均方根误差（$RMSSE$，见式1）进行评估，这种方式适用于间歇性的销量数据，同时由于不依赖于等于或接近零值的除法，并对正负预测误差以及大小预测都进行了惩罚，因此相对其它度量方法更为安全和准确。

$$\boldsymbol{RMSSE} = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}\left(Y_t - \hat{Y}_t\right)^2}{\frac{1}{n-1}\,sum_{t=2}^{n}\left(Y_t - Y_{t-1}\right)^2}} \quad (1)$$

其中 Y_t 和 $\hat{Y}_t$ 分别是在 t 时刻检查的时间序列的预测值和实际值，训练样本的长度为 h。

在评估完所有时间序列的 $RMSSE$ 后，进一步使用加权的 $RMSSE$（$WRMSSE$，见式2）对参赛者进行排名。

$$\boldsymbol{WRMSSE} = \sum_{i=1}^{42{,}840} w_i \times RMSSE \quad (2)$$

其中 w_i 是 i_{th} 系列数据的权重，$WRMSSE$ 越小说明预测分数越好。

概率预测. 该评价指标通过所缩放弹球损失函数(Scaled Pinball Loss，见式3) 每个系列的数据和四分位数进行计算。

$$\boldsymbol{SPL}(\boldsymbol{u}) = \left(\frac{\sum_{t=n+1}^{n+h}\left(Y_t - Q_t(u)\right)u\mathbf{1}\left\{Q_t(u) \le Y_t\right\}}{\frac{1}{n-1}\sum_{t=2}^{n}\left|Y_t - Y_{t-1}\right|}\right.$$
$$\left. + \frac{\left(Q_t(u) - Y_t\right)(1-u)\mathbf{1}\left\{Q_t(u) > Y_t\right\}}{\frac{1}{n-1}\sum_{t=2}^{n}\left|Y_t - Y_{t-1}\right|}\right) \times \frac{1}{h} \quad (3)$$

其中 Y_t 是经过点 t 所研究的时间序列的实际未来值，$Q_t(U)$ 是对四分位数 u 生成的预测值，h 是预测范围，n 是训练样本的长度，即所谓历史观测值的数量；$\mathbf{1}$ 是指标函数，如果 Y 在假定的区间内则为1，否则为0。

同样地，在计算完 SPL 之后按照给定的权重表取加权平均（$WSPL$，见式4）作为最后的评估成果. w_i 是比赛中 i_th 个系列的权重，u_j 是被研究的 j_th 个四分位数。

$$\boldsymbol{WSPL} = \sum_{i=1}^{42{,}840} w_i \times \frac{1}{9}\sum_{j=1}^{9}\mathrm{SPL}\left(u_j\right) \quad (4)$$

3. 数据分析和处理

我们一共需要处理 42,840 个分层时间序列。数据来源于美国 3 个加利福尼亚州（CA），德克萨斯州（TX）和威斯康星州（WI）。并且一共有 10 个分层序列，这里的“分层”表示可以在不同级别上汇总数据：商店级别，部门级别，产品类别级别和州级别。销售信息可以追溯到 2011 年 1 月至 2016 年 6 月。除了销售数量，还有相关的商品价格，促销和节假日的相应数据。数据包括来自 3 个类别和 7 个部门的 3049 种单独产品，分别在 3 个州的 10 家商店中销售。

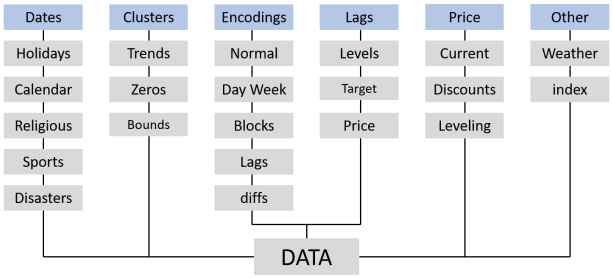

图 6. 数据集构成

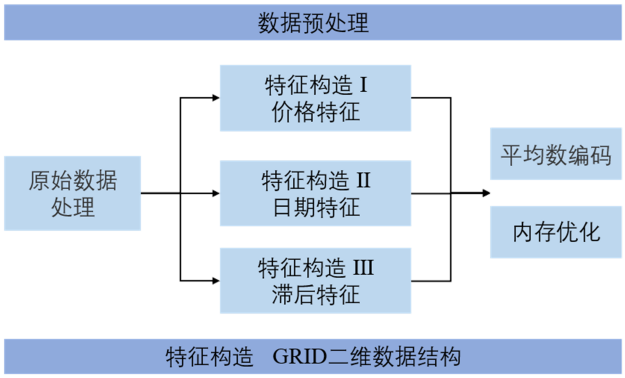

图 7. 特征构造过程

4. 模型设计

4.1. 梯度提升树

梯度提升树 (Gradient Boosting Decision Tree) 是集成学习的一种模型（见式5），其基学习器仍然是决策树，但被限定为回归树，因为每次迭代要拟合的是梯度值为连续值。

$$T(x;\Theta)=\sum_{j=1}^{J}c_jI\left(x\in R_j\right) \tag{5}$$

不同的凸优化问题对应着不同的损失函数，在提升树模型中表现为平方损失函数，但为了求解一般性问题，提出了通用的决策树模型，即梯度提升树。提升树用加法模型与前向分布算法对学习进行优化，但不适用于一般损失函数；梯度提升利用了最速下降法的近似方法（见式6）

$$-\left[\frac{\partial L\left(y_i,f\left(x_i\right)\right)}{\partial f\left(x_i\right)}\right]_{f(x)=f_{m-1}(x)} \tag{6}$$

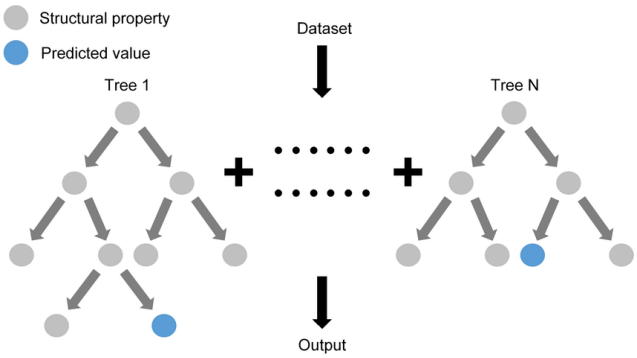

图 8. GBDT 算法

4.2. LightGBM

LightGBM 是一个轻量级的梯度提升机，作为工程实现 GBDT 算法的框架能够快速处理大量的数据并且支持高效率的并行训练。主要结合 Histogram 算法用以减少候选分裂点位数量、由 GOSS 算法的作用是减少样本的数量以及 EFB 算法的作用是减少特征的数量 [3]。通过引入这三种算法极大降低了生成叶子的复杂度从而节约了计算时间。

LightGBM 在预测任务中显示出优于其他机器学习方案的几个优势。例如在 M5 Accuracy 竞赛中考虑的一个：它允许有效处理各种类型（数字、二进制和分类），与典型的梯度提升（GBM）实现相比，它计算速度更快，不依赖于数据预处理和转换，并且只需要优化相对较少的参数（例如，学习率，迭代次数，特征值将被装入的最大箱数量、估计器数量和损失函数）。LightGBM 很容易针对互相关的大量序列数据提供准确泛化的解决方案。

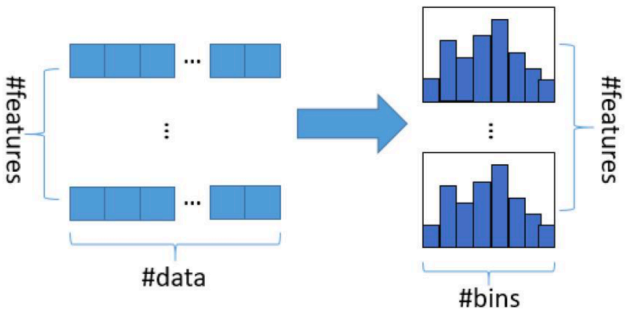

图 9. 直方图算法

4.3. Prophet 时序预测模型

Prophet 是 Facebook 开源的一个基于加法模型，通过时间序列分解和机器学习的拟合来预测时间序列数据的项目，适用于具有强烈季节性影响和多个季节历史数据的时间序列，此外 Prophet 还能自适应处理异常值和缺失值。Prophet 模型具有以下特点：

1) 准确快速 Prophet 在 Facebook 的许多应用程序中使用，用于为计划和目标设定生成可靠的预测。在大多数情况下，它的表现优于其他的时序分析方法。由于在拟合模型的时候使用了 pyStan 这个开源工具，所以可以快速得到预测结果, 兼顾长期趋势与局部震荡。
2) 自动化无需额外的数据预处理操作即可对原始数据进行合理预测。Prophet 对异常值、缺失数据和时间序列中的显著变化具有鲁棒性。
3) 可调预测 Prophet 提供了用户自适应调整模型的可能性。可以添加或修改参数，通过添加先验知识来改进预测结果。
4) 提供统计指标包括拟合曲线，上下界，还包含了用于衡量和跟踪预测准确性，并标记应手动检查的预测的部分。

Prophet 模型将时间序列 y_t 进行了如式7分解

$$y(t)=g(t)+s(t)+h(t)+\epsilon_t \tag{7}$$

其中趋势项 $g(t)$ 模拟时间序列值的非周期性变化，季节项 $s(t)$ 代表周期性变化（例如，每周和每年的季节性），假期项 $h(t)$ 表示假期的影响。误差项 ϵ_t 表示模型无法拟合的任何特殊变化，模型中假设误差项 ϵ_t 服从正态分布。

趋势项. 该部分由饱和增长 (Saturating Growth Model) 模型和分段线性模型 (Piecewise Linear Model) 组合而成。对于增长预测，数据生成过程的核心组成部分是人口如何增长以及预计将如何继续增长的模型。Facebook 的增长建模通常类似于自然生态系统中的人口增长，其中非线性增长会在承载能力下饱和。这种增长通常使用逻辑增长（Logistic Growth Model）模型建模，其最基本的形式如式8所示：

$$g(t) = \frac{C}{1 + \exp(-k(t-m))} \tag{8}$$

其中 C 是承载能力，k 是增长率，m 是偏移参数。在现实应用中，这三个参数不可能为常量，所以在 Prophet 里面，将把这三个参数全部换成了随着时间而变化的函数，也就是

$$C = C(t), k = k(t), m = m(t)$$

Prophet 还通过明确定义允许增长率变化的变化点 (changepoints)，将趋势变化纳入增长模型。假设有 S 个变化点，分别定义在 s_j 时刻 $j = 1, \cdots, S$.Prophet 会定义一个增长率变化向量 $\delta \in \mathbb{R}^S$，其中 δ_j 表示在 s_j 时刻增长率的变化量。那么任意时刻 t 的增长率就可以表示为基本增长率 k 加上到该点为止所有变化点的变化量，即 $k + \sum_{j:t>s_j} \delta_j$。可以通过定义向量 $\mathrm{a}(t) \in \{0,1\}^S$，如式9：

$$a_j(t) = \begin{cases} 1, \text{ if } t \geq s_j, \\ 0, \text{ otherwise.} \end{cases} \tag{9}$$

同时将 t 时刻的增长率表示为 $k + \mathrm{a}^T\boldsymbol{\delta}$

当速率 k 变化时，偏移参数 m 也必须被调整以连接线段的端点。改变点 j 的调整应为式10：

$$\gamma_j = \left(s_j - m - \sum_{\ell<j} \gamma_\ell\right) \cdot \left(1 - \frac{k + \sum_{\ell<j} \delta_\ell}{k + \sum_{\ell \leq j} \delta_\ell}\right) \tag{10}$$

所以，分段逻辑增长模型可以表示为：

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \boldsymbol{a}(t)^t\boldsymbol{\delta}) \cdot (t - (m + \boldsymbol{a}(t)^T\boldsymbol{\gamma})}$$

而对于不表现出饱和增长的预测问题，分段线性模型提供了一个简约且通常有用的模型，这里的趋势模型可以表示为：

$$g(t) = (k + \boldsymbol{a}(t)\boldsymbol{\delta}) \cdot t + (m + \boldsymbol{a}(t)^T\boldsymbol{\gamma})$$

其中 k 表示增长率，δ 表示增长率的变化量，m 是偏移参数，与逻辑回归函数最大的区别在于 γ，在分段线性模型中 $\gamma_j = -s_j\delta_j$ 使得函数连续。

季节项. Prophet 利用傅立叶级数来提供灵活的周期效应模型。令 P 为时间序列具有的常规周期（例如，$P = 365.25$ 表示以年为周期，$P = 7$ 表示以周为周期）。任意平滑的季节性影响可以用标准傅里叶级数近似表示为式11：：

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right)\right) \tag{11}$$

拟合季节性需要估计 $2N$ 个参数 $\boldsymbol{\beta} = [a_1, b_1, \cdots, a_N, b_N]^T$，这是通过为历史和未来数据中的每个时刻构建一个季节性向量组成矩阵 X 来完成的，例如以年为周期的序列，N=10 时：

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25}\right), \cdots, \sin\left(\frac{2\pi(10)t}{365.25}\right)\right]$$

季节项于是可以表示为：

$$s(t) = X(t)\boldsymbol{\beta}$$

在模型中，采用 $\beta \sim Normal(0, \theta^2)$ 对 β 进行初始化，此外还可以修改参数以控制 θ 的值。

假期项. Prophet 中内置了全球及各个国家的特殊节假日，对于给定的预测问题，Prophet 使用全球假期集和国家特定假期集的联合，此外 Prophet 还允许自定义过去和未来事件的列表。

Prophet 假设假期的影响是相互独立的，将不同的节假日看作相互独立的模型，并且可以为不同的节假日设置不同的前后窗口值，表示该节假日会影响前后一段时间的时间序列。

对于每个假期 i，D_i 表示该假期前后一段时间。通过定义指示函数来表示时刻 t 是否在假期 i 期间，并为每个假期分配一个参数 κ_i，它表示预测中相应的变化。与季节性类似，这是通过生成回归矩阵完成的：

$$Z(t) = [\mathbf{1}(t \in D_1), \cdots, \mathbf{1}(t \in D_L)]$$

$$h(t) = Z(t)\boldsymbol{\kappa}$$

其中 $\boldsymbol{\kappa} \sim Normal(0, v^2)$，同样可以自定义 v 的值以控制节假日对模型的预测结果的影响。

4.4. 融合模型

对于时间序列的多步预测问题，通常有递归和非递归两种方法。其中多步预测方法如下：

直接预测. 为每个预测时间步开发建立模型，但是计算和维护代价高，尤其是当要预测的时间步数较大时。因为使用了单独的模型，这意味着没有对时间步之间的前后依赖进行建模。用于预测的模型与假设模型不匹配，不同的模型是独立估计的，结果可能会有很大的差异

$$p(t+1) = model1(obs(t-1), obs(t-2), ..., obs(t-n))$$

$$p(t+2) = model2(obs(t-2), obs(t-3), ..., obs(t-n))$$

递归多步预测. 使用单步模型，其中前一时间步长的预测被用作对下一时间步长进行预测的输入。由于使用预测值来代替观测值，递归策略可能造成预测误差累积，从而随着预测时间的增加，性能迅速下降。

$$p(t+1) = model1(obs(t-1), obs(t-2), ..., obs(t-n))$$

$$p(t+2) = model2(p(t+1), obs(t-1), ..., obs(t-n))$$

直接预测与递归预测混合. 上述两种方案的结合，可以为要预测的每个时间步长构建单独的模型，但是每个模型可以使用模型在先前时间步长做出的预测作为输入值。

$$p(t+1) = model1(obs(t-1), obs(t-2), ..., obs(t-n))$$

$$p(t+2) = model2(p(t+1), obs(t-1), ..., obs(t-n))$$

多步预测. 多输出模型更复杂，因为它们可以学习输入和输出之间以及输出之间的依赖结构。更复杂可能意味着它训练速度较慢，需要更多的数据来避免过度拟合问题。

$$p(t+1), p(t+2) =$$

$$model(obs(t-1), obs(t-2), ..., obs(t-n))$$

在递归与非递归策略之间进行选择实际上在误差和方差之间进行权衡，当基础模型是非线性时，递归预测会产生偏差，但直接预测具有较高的方差，因为它在估计模型时使用较少的观测值，特别是对于较长的预测范围。

当且仅当 f 是线性的，使用线性模型 m，并且 embedding dim 正确时，递归预测是无偏的；在所有其他情况下，预测步长大于等于 2 时，递归预测都是有偏的。

递归策略的优势是只需要一个模型，节省了大量的计算时间，特别是当涉及大量时间序列和预测范围时。递归策略还确保拟合模型 m 与假设的数据生成过程 f 尽可能地匹配。另一方面，即使模型与数据生成过程等价，递归预测的结果也不等于条件平均值。

结合 Kaggle 上的讨论，发现无论是非递归方法还是递归方法，在划分的不同验证集上表现差异较大，多个模型的 public score 差异也较大。总体而言，递归方法比非递归方法得分更高，而非递归方法在验证集 3 上得分最高。为了提高鲁棒性，本文采用了递归与非递归结合的方法。

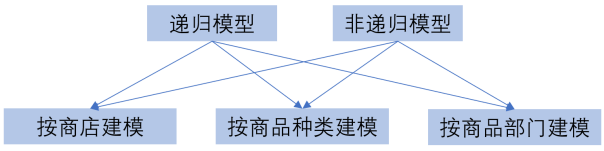

图 10. 递归方案与非递归方案

5. 超参数搜索

5.1. 网格搜索

网格搜索适用于重复的、迭代的进行，仅适合三个及以下数量的超参数，用户列出一个较小的超参数值域，这些超参数值域的笛卡尔集（排列组合）为一组组超参数。网格搜索算法使用每组超参数训练模型并挑选验证集误差最小的超参数组合。

然而当超参数的数量增长时，网格搜索的计算复杂度会呈现指数增长，这时要换用随机搜索。

5.2. 随机搜索

随机搜索指数级比网格搜索更为高效，因为网格搜索将大量的计算浪费在了指数级的对结果无影响的参数中，而随机搜索一般会根据超参数的边缘分布采样，几乎每次都搜索了对结果有影响的参数的值 [11]。

6. 实验结果及分析

算法缺陷. LGBM 的主要思想是利用弱分类器迭代训练以得到最优模型，通过不断寻找特征上最优的数值分割点进而生成子节点，最后通过大量弱分类器的搜索集成得到结果，相当适用于非线性的输入和输出之间的数据关系并拟合其间的波动特征。

然而销量预测和时序相关，不仅需要考虑时间序列的季节性和周期性，还需要结合趋势性进行分析。GBDT 由于不断寻找特征最佳分裂点，所以能够很好地拟合周期性，但对趋势性却由于算法本身的缺陷无法很好控制。我们在实验中结合了 Prophet，由于其将趋势变化纳入了增长模型因此能够得到不错的效果，得分0.53744，后续再次基础上不断改进。

外界因素. 在本次销量预测中能够发现，即便算法足够 SOTA，ensemble 足够多，特征工作做得细致也不一定能够得到好的结果，在时间数据序列中存在一个小的波动就能让最后的数据与预期相差巨大。比如在德州的数据序列中存在 28 天的洪水灾害，这对于出行采购的人会带来影响进而影响销量，因此在实验最后的结果中我们也考虑乘以保险系数 0.97，从而取得了更好的分数，表现为我们实验的最终结果，得分为0.51945。

7. 结论

1) 从商品属性、价格以及日期三个维度完成特征构造，尤其是通过滞后特征来表达时间序列数据间的依赖关系；
2) 按照商店-类别-部门的层级分别训练模型，并以递归和非递归两种方法分别实现多步的时间序列预测，使得模型具有较强的鲁棒性；
3) 引入了时间序列预测模型 Prophet，进一步挖掘销量的时序依赖关系，最终提交得分为 0.51945；

8. 所完成的加分项

表 1. 所完成的加分项

加分项	对应子章节
梯度提升决策树	章节1 分析对比两种学习方法
特征构造	章节3 构造新特征并分析效果
对比多种模型	章节4 对比观察多种模型性能

如表1所示，本文包含了作业中的三个加分项.

1）梯度提升决策树. 首先我们在章节 1 中做出数据表格领域的相关文献综述，列举了 GBDT、Xgboost 和 LightGBM 等相关机器学习模型和框架，以及 TabTransfomer、TabNet、NODE 和 AutoInt 等深度学习框架，然后从现象、三条主要原理、数据集和数据类型等方面对比分析了传统机器学习和深度学习，解释了深度学习在表格数据上应用效果不佳的原因。

2）特征构造. 在进行数据 EDA 之后，我们对数据进行了特征构造，其中包括价格特征、日期特征和时序数据中的滞后特征，在每一步处理中都进行了内存优化，且在最后进行了平均数编码，从而形成了 GRID 二维数据结构为模型分析做准备。

3）对比多种模型. 除了 LightGBM 的基础模型之外，考虑到其算法缺陷结合了 Prophet 模型增加对趋

势性的控制，此外还进行了融合模型的尝试，得到了不断提升的得分。

9. 成员分工及贡献比

组内成员分别来自计算机技术和建筑学两个专业，在合作中合理分工以充分发挥自身学科优势，成员分工及贡献比如表2所示。沙斌竹同学负责主体代码和相应的算法模型优化，李洪君同学负责实验报告、海报和部分代码，同时两人共同参与到实验报告和海报汇报的工作中。各自贡献比均为50%。

表 2. 成员分工及贡献比

姓名	分工	贡献比
沙斌竹	主要负责代码、实验报告	50%
李洪君	海报、实验报告、部分代码	50%

10. Kaggle 比赛提交记录截图及每两次提交之间的改进

提交记录. 图11为四次有效提交，从下到上依次为独立 Prophet 模型、融合模型、两者求加权平均以及加权平均后乘以其它参数的提交结果，能够发现成绩依次提升。

改进说明. 如图11所示，我们首先使用 Prophet 模型单独预测销量，其初始分数为0.53744；在第二次提交中的改进为将递归与非递归方案结合使用预测销量的提交结果，将分数提升到了 0.52339；在第三次提交中的改进为将前二者的预测结果取加权平均，但效果不够明显，仅将得分提升到 0.52131；最后的提交在第三次的基础上，把预测结果乘以 $magic\_number(0.98)$，获得了最佳的分数0.51945。

11. 心得体会

沙斌竹.

李洪君. 感谢队友沙斌竹同学在算法方面的细心讲解，以及完成销量预测任务的积极和负责，也因为合理分工和共同努力，Kaggle 的提交结果达到了0.51945的理想分数，从而获得了本学期课程的《最佳成果奖》。

在逐渐完成大作业各个部分的同时，自己对数据科学的了解和应用能力也逐步加深。在最初的文献综述中，阅读了许多表格数据相关的机器学习和深度学习算法，并在 Kaggle 平台上精读了排行靠前的 notebook，不仅了解了算法的发展脉络，还明白了理论和实践的差异与结合。在数据预处理中也接触了许多特征构造和可视化表达的内容，模型设计阶段对 LightGBM 的理论知识进行了更为基础的学习和探索。并通过队友明白了 Prophet 模型在时序预测中的实用性，以及对比了融合模型不同方法对于预测的提升效果。此外，也通过这次大作业了解了学术会议的流程，并在整个过程中明白了合作的重要性。

最后，感谢吴老师和助教们对这门课程的付出，在课堂或者课间的问答交流中能够明显看出老师对于传道授业解惑的热情与担当，同时也感谢老师提供其它专业同学选课的机会。

参考文献

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000. 1

[2] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794. 1

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017. 1, 5

[4] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin, "Tabtransformer: Tabular data modeling using contextual embeddings," *arXiv preprint arXiv:2012.06678*, 2020. 2

[5] Sercan Ö Arik and Tomas Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 6679–6687. 2

图 11. Kaggle 比赛提交记录截图

[6] Sergei Popov, Stanislav Morozov, and Artem Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," *arXiv preprint arXiv:1909.06312*, 2019. 2

[7] Manu Joseph, "Pytorch tabular: A framework for deep learning with tabular data," *arXiv preprint arXiv:2104.13638*, 2021. 2

[8] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko, "Revisiting deep learning models for tabular data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18932–18943, 2021. 2

[9] Ravid Shwartz-Ziv and Amitai Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022. 2, 3

[10] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 3

[11] James Bergstra and Yoshua Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012. 8

A. 附录

大作业相关文件清单及其相应说明