机器学习工程师纳米学位毕业项目

预测 Rossmann 未来的销售额

柴伟

2018 年 7 月

目录

第一章　数据分析

1.1 项目概述

随着计算机技术以及数据信息技术的飞速发展，数据量呈现爆炸性增长，越来越多的企业重视挖掘这些数据背后的潜在信息。对于竞争愈演愈烈的超市行业而言，如果能够通过对销售额的预测，从而采取更有针对的日常调度以及更为有效的促销手段，那将是十分重要的。

本项目来源于 Kaggle 竞赛 Rossmann Store Sales, Rossmann 是德国首家平价日用商品，在欧洲拥有 3000 多分店。本项目所要解决的问题是，提前 6 周预测 1115 家商店的日常销售额。

本项目的数据集来源于 Kaggle 竞赛 Rossmann Store Sales 所提供的数据文件，主要分为以下三个文件：train.csv，test.csv 以及 store.csv。其中，train.csv 文件中是包含销售额数据，用来进行训练模型；test.csv 文件中不包含销售额数据，是用于预测的；store.csv 文件中记录的是与商店有关的信息。

1.2 研究内容

本项目基于 Kaggle 所提供的数据，采用 XGBoost 与 LightGBM 方法，对 1115 家商店未来 6 周的销售额进行预测。为了达到最终的目标，将问题分解为以下几个方面：

(1) 探索性数据分析：观察数据的特征，记录数据的清洁度以及数据质量问题，例如是否存在缺失值和异常值，分析特征之间的关系。

(2) 特征工程：对特征进行预处理，选择对销售额预测影响较大的特征，基于原始特征去构建与销售额更为相关的新特征。

(3) 基准模型：搭建一个解决问题的基准模型，为以后模型优化提供基础。

(4) 模型优化：对模型的参数进行调节，以提高模型的性能；同时组合不同的模型，来进一步地提高预测精度和泛化能力。

1.3 评价指标

根据 Kaggle 竞赛所提供的信息，评估标准采用 RMSPE 均方根百分误差：

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)} \tag{1.1}$$

其中，y_i是每日单个商店的真实销售额，$\hat{y}_i$是每日单个商店的预测销售额，n 是样本的个数。

RMSPE 评价标准是非对称的，并且对异常点十分敏感，如果一个模型在比

较特殊的一天的预测值和实际值相差较大，使得 RMSPE 很大，这会使得很好的模型效果很差。

第二章　数据分析与算法概述

2.1 数据探索与可视化

2.1.1 数据特征

train.csv 训练数据集中包含了 9 个特征，1017209 条记录。test.csv 测试数据集中包含了 8 个特征，41088 行记录，与 train.csv 训练数据集相比，缺少 Sales 和 Customers 特征，同时增加了 Id 特征，表示(Store, Date)组合的编号。表 2.1 对训练集和测试集中的特征进行了详细地说明。

表 2.1 训练集与测试集特征

特征名	特征类型	含义
Id	定量数据	记录编号
Store	定量数据	商店编号
DayOfWeek	定性数据	星期几
Date	定量数据	日期
Sales	定量数据	销售额
Customers	定量数据	客户数
Open	定性数据	是否营业
Promo	定性数据	是否有促销
StateHoliday	定性数据	假日
SchoolHoliday	定性数据	是否学校假日

store.csv 商店信息数据集中包含了 10 个特征，1115 条记录。表 2.2 对商店数据的特征进行说明：

表 2.2 商店特征

特征名	特征类型	含义
Store	定量数据	商店编号
StoreType	定性数据	商店类型
Assortment	定性数据	商店类别
CompetitionDistance	定量数据	与最近竞争商店的距离
CompetitionOpenSinceMonth	定性数据	最近竞争商店开张的月份
CompetitionOpenSinceYear	定量数据	最近竞争商店开张的年份
Promo2	定性数据	是否有持续性的促销活动
Promo2SinceWeek	定量数据	开始 Promo2 的日历周
Promo2SinceYear	定量数据	开始 Promo2 的年份
PromoInterval	定性数据	参与 Promo2 的月份列表

下面分别针对定性数据与定量数据，给出部分特征的统计特征信息。Store 的取值范围为 1-1115，Date 在训练数据集中的取值范围从 2013 年 1 月 1 日到 2015 年 7 月 31 日，在测试数据集中的取值范围为 2015 年 8 月 1 日到 2015

年 9 月 17 日。其余部分定量特征的统计信息如表 2.3 所示。

表 2.3 部分定量特征统计信息

	Sales	Customers	CompetitionDistance
count	1017209.00	1017209.00	1112
mean	5773.82	633.15	5404.9
std	3849.93	464.41	7663.17
min	0.00	0.00	20
25%	3727.00	405.00	717.5
50%	5744.00	609.00	2325
75%	7856.00	837.00	6882.5
max	41551.00	7388.00	75860

下面给出部分定性数据的取值，如表 2.4 所示。

表 2.4 部分定性数据的取值

特征名称	取值
Open	0：不营业，1：营业
StoreType	a, b, c, d 四类
Assortment	a：基础类，b：补充类，c：扩展类
StateHoliday	0：非假日，a 公共假日，b：复活节，c：圣诞节
SchoolHoliday	0：非假日，1：假日
Promo	0：无促销，1：促销
Promo2	0：无持续促销，1：有持续促销
PromoInterval	"Jan, Apr, Jul, Oct", "Feb, May, Aug, Nov", "Mar, Jun, Sept, Dec"

2.1.2 清洁度与数据质量

对于数据存在的问题，往往从清洁度和数据质量两个方面来考虑。所谓数据清洁度是指：表格是否每一列都是一个特征，每一行都是一条记录，每个位置是否都只记录一个数据。所谓数据质量问题是指：数据丢失，数据无效，数据不准确，数据不一致。不整洁的数据属于混杂数据，而低质量的数据属于脏数据。对数据进行评估，所记录的清洁度和质量问题如下。

(1) 清洁度

train.csv 训练数据和 store.csv 商店数据应该合并成一个表格数据。

(2) 质量问题

Date 的数据类型不正确，CompetitionDistance, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2SinceWeek, Promo2SinceYear, PromoInterval 存在数据丢失。

2.1.3 可视化分析

由于需要预测销售额，因此整个数据可视化都是围绕着 Sales 展开的。首先来看一下 Sales 的数据分布情况，如图 2.1 所示，其中图 a 是原始的数据分布图，

图 b 是调整 bin 间隔以及坐标范围后的图，可以看出 Sales 属于右偏态分布。为了能够提高模型的精度，需要对 Sales 进行某种变换，使得变换后的数据分布趋向于比较理想的分布，比如正态分布。对 Sales 取对数变换，变换后的分布如图 2.2 所示，可以看出取对数后数据呈正态分布。

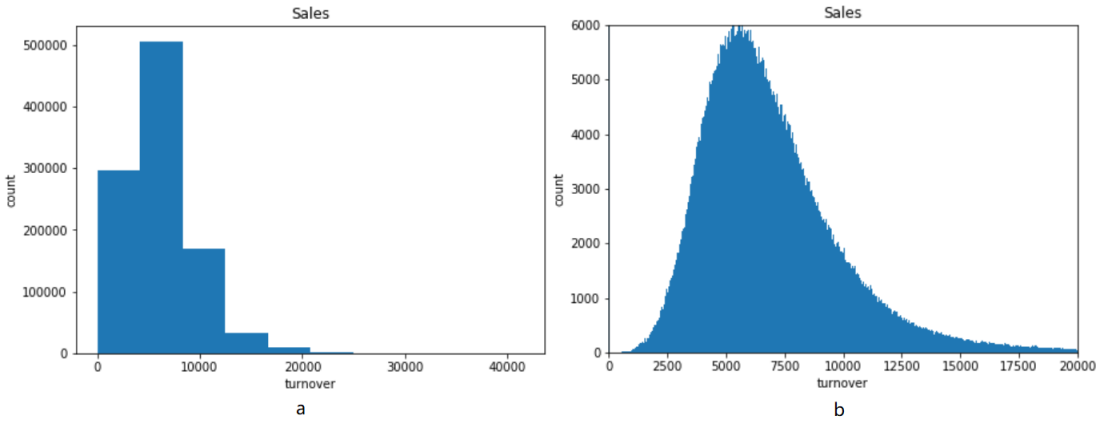

图 2.1 Sales 数据分布

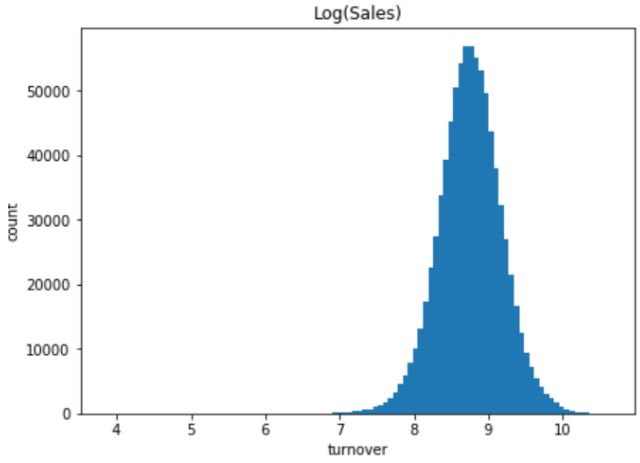

图 2.2 Log(Sales)数据分布

其余部分特征的分布如图 2.3 所示，

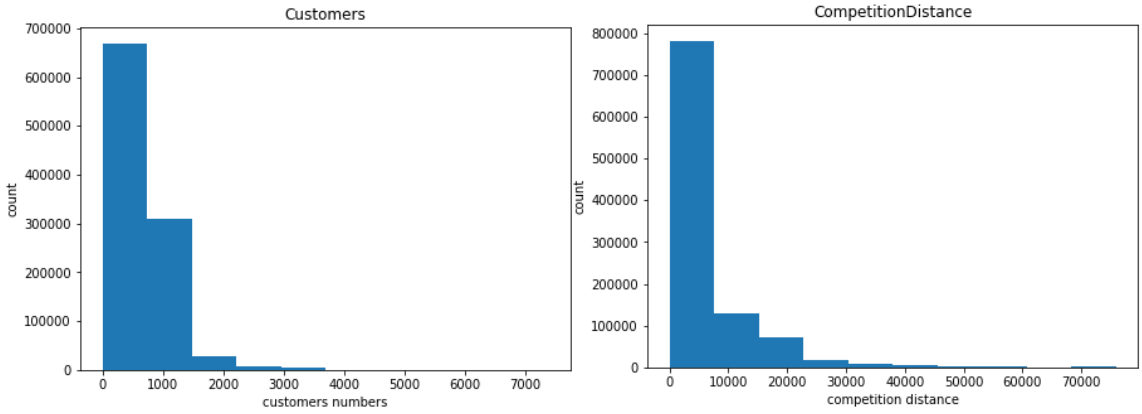

图 2.3 部分特征的数据分布

接下来分析其他特征与 Sales 之间的关系，首先从整体上来看各个特征的相关性。如图 2.4 所示，可以看到 Sales 与 Customers 具有很强的正相关性，其次与 Promo 也具有一定的正相关性。

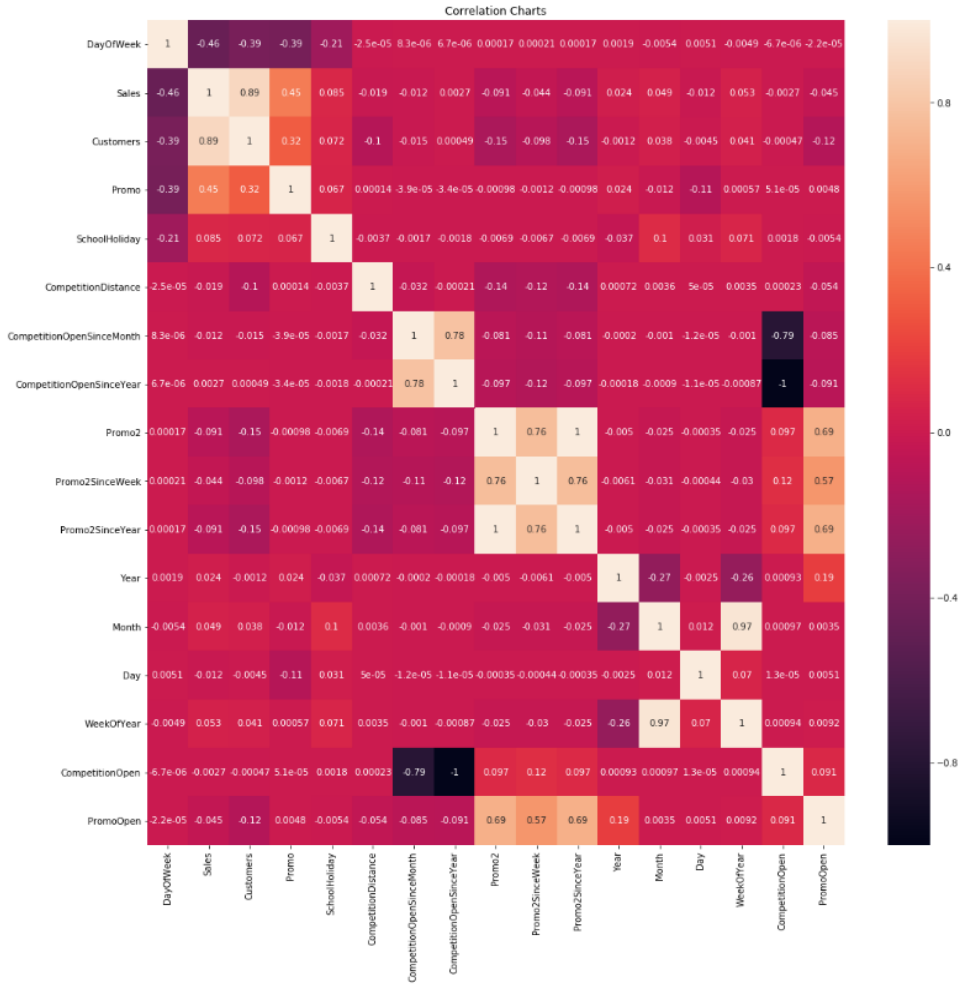

图 2.4 特征相关性

那么先来看看这些关系，如图 2.5 所示，分别对 Sales 与 Customers 取对数，可以看出这两者之间正相关性，同时观察 Sales 与 Promo 之间的关系，可以看出当 Promo=1 时,也就是促销时，销售额会高一些，说明促销时可以提高销售额的。

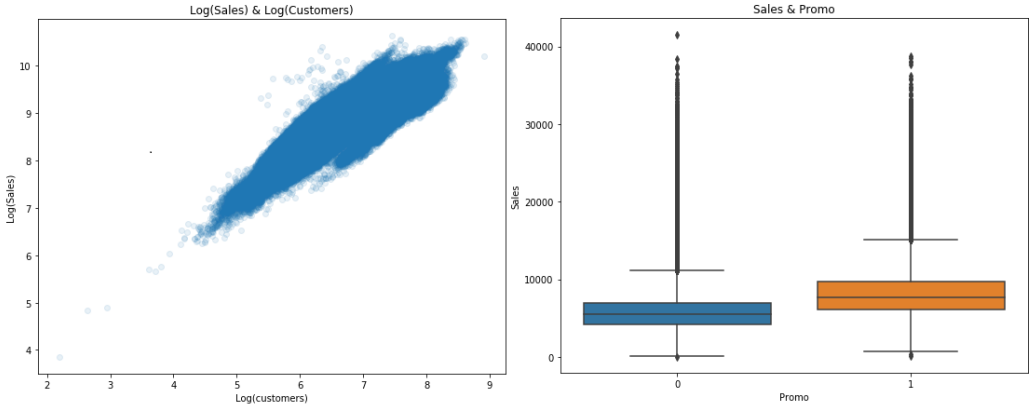

图 2.5 Sales 与 Customer、Promo 的关系

虽然其他特征与 Sales 从图 2.4 中看不出有什么相关性，但是还是对一些特征进行探索。先看一下节假日以及学校节日与销售额之间的关系，如图 2.6 所示，可以看出 StateHoliday 公共节日期间销售额比不放假期间要高，说明节假日对销售额有促进作用。同时可以看出学校放不放假，对销售额几乎没什么影响。

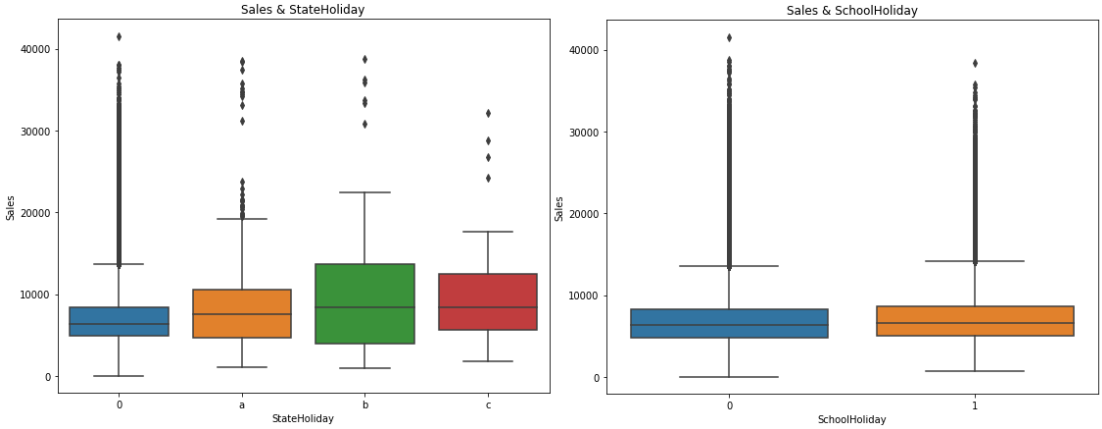

图 2.6 Sales 与 StateHoliday、SchoolHoliday 的关系

其余的特征之间的关系，通过多变量分面图来观察。如图 2.7 所示，可以看出 b 类型的商店的销售额要明显高于其他类型的商店，同时还可以看出周六周天从不进行促销活动，c 类型商店在周天不营业，周一的销售额比较高。

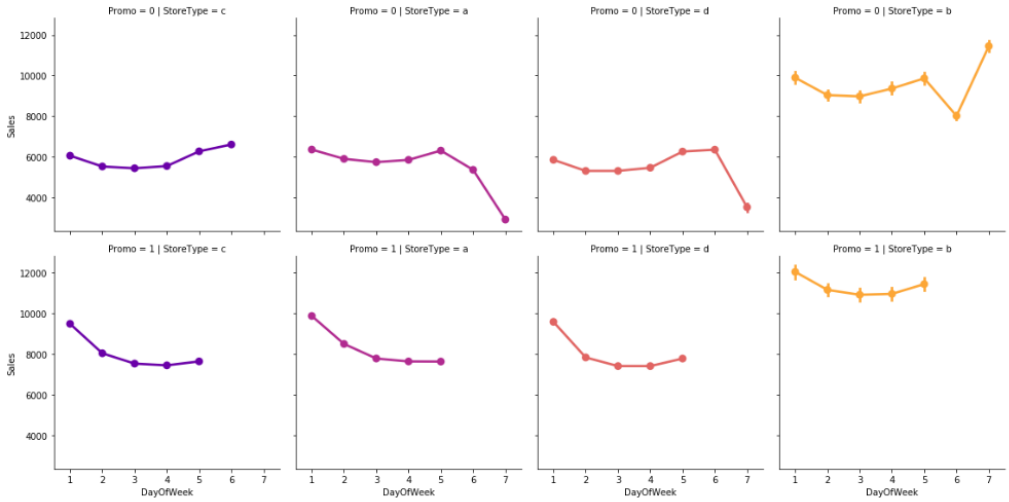

图 2.7 Sales 与 DayOfWeek、StoreType、Promo 的关系

2.2 算法与技术

由于本项目采用 XGBoost 与 LightGBM 算法，因此下面将重点介绍这两种算法。

2.2.1 XGBoost 算法

XGBoost 属于一种树集成方法，是 Gradient Boosting Machine 的一种扩展，能够利用 GPU 多线程并行加速树的构建，支持 YARN,MPI 等多个平台，实现分布式运算，加速训练。XGBoost 使用 CART 回归树作为基学习器，基于加法模型，采用前向分布算法去学习模型，具体的算法推导如下[1]。

第 i 个样本在第 t 轮的模型预测值$\hat{y}_i$，保留 t-1 轮的模型预测值$\hat{y}_i^{(t-1)}$后，加入一个新的函数 $f_t(x_i)$，尽可能地让目标函数最大程度地降低。

$$\begin{aligned}
&\hat{y}_i^{(0)} = 0 \\
&\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
&\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\cdots \\
&\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_1(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{2.1}$$

目标函数定义为：

$$\begin{aligned}
Obj^{(t)} = L^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant
\end{aligned} \tag{2.2}$$

对式(2.2)进行二阶泰勒展开可得：

$$L^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant \tag{2.3}$$

其中，$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$，$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$。

将式(2.3)中常量部分去除，可得：

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{2.4}$$

其中，$f_t(x) = w_{q(x)}, w \in R^T, q: R^d \to \{1, 2, \cdots, T\}$，$w$ 叶子权重，T 树叶数量。

将树模型的复杂度定义为：

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2 \tag{2.5}$$

联合式(2.4)和(2.5)，可得：

$$\begin{aligned}
L^{(t)} &\cong \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \\
&= \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T
\end{aligned} \tag{2.6}$$

定义 $G_j=\sum_{t\in I_j} g_i, H_j=\sum_{t\in I_j} h_i$ ，则式(2.6)可以改写为：

$$L^{(t)}\cong\sum_{j=1}^{T} G_j w_j+\frac{1}{2}(H_j+\lambda)w_j^2+\gamma T \tag{2.7}$$

假设树的结构 *q(x)*已知，则由式(2.7)可以计算叶子 j 的最优权重 w_j^*为：

$$w_j^*=-\frac{G_j}{H_j+\lambda} \tag{2.8}$$

由式(2.8)可得，最优目标函数值为：

$$L^{(t)}(q)=-\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda}+\gamma T \tag{2.9}$$

式(2.9)可以作为对一个树结构进行打分的函数，称为结构分数，分数越小，树的结构越好。那么利用这个结构分数来枚举所有的树结构，从而找到最优的树，但是往往无法枚举出所有的树，因此这里采用贪心算法，一次只优化树的一个层次，具体通过式(2.10)来找到当前层次的最优结构。

$$Gain=\frac{1}{2}[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}]-\gamma \tag{2.10}$$

其中，$\frac{G_L^2}{H_L+\lambda}$ 为左子树的分数，$\frac{G_R^2}{H_R+\lambda}$ 为右子树的分数，$\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$ 为不分割时的分数。

XGBoost 相比较于传统的 GBDT 算法，具有以下的优点[2]：

（1）传统 GBDT 以 CART 作为基分类器，XGBoost 还支持线性分类器，这个时候 XGBoost 相当于带 L1 和 L2 正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）。

（2）XGBoost 在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的分数的 L2 模的平方和。从 Bias-variance trade off 角度来讲，正则项降低了模型方差，使学习出来的模型更加简单，防止过拟合，这也是 XGBoost 优于传统 GBDT 的一个特性。

（3）添加了 shrinkage 缩减，在每一步 tree boosting 之后增加了一个参数 n（权重），通过这种方式来减小每棵树的影响力，给后面的树提供空间去优化模型。

（4）采用了 column subsampling 列抽样，可以防止过拟合，并且有利于并行化处理算法。

（5）在划分点查找时，γ 其实相当于提供阈值功能，当划分前后的增益小于这个阈值时，不进行划分，也就是具有预剪枝的功能。

（6）对于特征的值有缺失的样本，XGBoost 可以自动学习出它的分裂方向。

（7）具有可扩展性与分布式计算特点，相对于 GBDT，XGBoost 设计并

建立了一种高可扩展的端到端的树提升系统。

（8）可并行学习，采用了高速缓存压缩感知算法，而且还可以进行核外计算。

2.2.2 LightGBM 算法

上述提及到的 XGBoost 算法虽然相比较于 GBDT 等常见的机器学习算法有了很大程度上的提升，但是当特征维度很高或者数据量很大的时候，其效率和可扩展性仍然不能达到满意的结果。究其主要原因是因为对于每个特征，我们都需要去扫描整个数据集来得到分割点。为了解决上述问题，LightGBM 算法[3]采用了两种解决方法：基于梯度的 one-side 采样（GOSS）和互斥的特征捆绑（EFB）。

1. GOSS 算法

在 AdaBoost 中，采样权重作为数据实例重要性的衡量指标。然而在 GBDT 中，没有内含的样本权重，于是基于采样方法的权重不能应用于 GBDT 中。幸运的是，如果实例梯度值小，这个实例的误差就小，说明这个实例已经训练的很好了，直接的想法就是抛弃拥有小梯度的实例数据，这样一来数据的分布就会发生改变，会损失学到的模型的精确度。为了避免这个问题，我们提出了一种叫做 GOSS 的方法。GOSS 保持有较大梯度的实例，在小梯度数据上运行随机采样。为了弥补这样做造成的数据分布的影响，当我们计算信息增益的时候，对于小梯度的数据 GOSS 引入了常量乘法器。特别的是，GOSS 首先根据梯度绝对值排序，再选出 a * 100%大梯度实例数据。之后在余下的数据随机采样 b * 100%。经过这个过程，当计算信息增益时，GOSS 使用小梯度放大了采样数据 1-a/b。这样做的好处在于，我们放更多的注意力在训练实例上，而没有改变原始数据的分布。

GBDT 使用决策树，来学习获得一个将输入空间映射到梯度空间的函数。假设训练集有 n 个实例 $x_1,\cdots,x_n$，特征为 u 度为 s。每次梯度迭代时，模型输出变量的损失函数的负梯度方向为 $g_1,\cdots,g_n$，决策树根据最大信息增益的最优切分点将数据划分到各个节点。对于 GBDT，通常通过方差衡量分割后的信息增益。

令 O 表示某个固定节点的训练集，则分割特征 j 的分割点 d 定义为：

$$V_{j|o}(d)=\frac{1}{n_o}\left(\frac{\left(\sum_{\{x_i\in o:x_{ij}\le d\}}g_i\right)^2}{n_{l|o}^j(d)}+\frac{\left(\sum_{\{x_i\in o:x_{ij}>d\}}g_i\right)^2}{n_{r|o}^j(d)}\right) \tag{2.11}$$

其中，$n_O=\sum I[x_i\in O], n_{l|O}^j=\sum I\{x_i\in O:x_i\le d\}, n_{r|O}^j=\sum I\{x_i\in O:x_i>d\}$。

遍历每个特征的每个分裂点，找到 $d_j^*=\arg\max x_d V_j(d)$ 并计算最大的信息增益 $V_j(d_j^*)$，然后根据特征 j^* 的分裂点 d_j^* 将数据分到左右子节点。

在 GOSS 算法中，首先根据数据的梯度将训练降序排序；保留梯度值前 a*100%的数据实例，作为数据子集 A；对于剩下的数据的实例 Ac，随机采样获得大小为 b*|Ac|的数据子集 B；最后我们通过以下方程估计信息增益：

$$\tilde{V}_j(d)=\frac{1}{n}\left(\frac{\left(\sum_{x_i\in A:x_{ij}\le d}g_i+\frac{1-a}{b}\sum_{x_i\in B:x_{ij}\le d}g_i\right)^2}{n_l^j(d)}+\frac{\left(\sum_{x_i\in A:x_{ij}>d}g_i+\frac{1-a}{b}\sum_{x_i\in B:x_{ij}>d}g_i\right)^2}{n_r^j(d)}\right) \tag{2.12}$$

在 GOSS 中，使用较小的数据集来估计 $\tilde{V}_j(d)$，而不是在所有数据集上的

信息增益$V_j(d)$，而且理论证明 GOSS 不会损失太多精度，比全部数据的随机采样效果要好，此处就不给予理论证明的过程。

2. EFB 算法

高维数据通常具有稀疏的特征。稀疏的数据提供给了我们减少特征数量而较小误差损失的可能。特别是在稀疏数据中，许多特征是互斥的，例如他们从不同时为非零值。我们绑定互斥的特征为单一特征（排他特征绑定）。通过仔细设计特征扫描算法，我们从特征绑定中建立相同的特征直方图作为单一特征。通过这种方式，构建直方图的复杂度从 O(data * feature)降到 O(data * bundle)，由于 bundle << feature，所以能够在无损精度的情况下极大地加速 GBDT 的训练。

此处存在两个问题：一是怎么判定那些特征应该绑在一起，二是怎么把多个特征绑为一个特征。下面给出判定那些特征应该绑在一起的算法：

（1）建立一个图，每个点代表特征，每个边有权重，其权重和特征之间总体冲突相关。

（2）按照降序排列图中的度数来排序特征。

（3）检查排序之后的每个特征，对他进行特征绑定或者建立新的绑定使得操作之后的总体冲突最小。

那么如何将多个特征合并成一个特征呢，关键在于确保原始特征值可以从特征包中区分出来。具体来说，鉴于直方图算法存储离散值而不是连续特征值，我们通过将互斥特征放在不同的箱中来构建 bundle 特征。这可以通过将偏移量添加到特征原始值中实现，例如，假设 bundle 特征中有两个特征，原始特征 A 取值[0, 10]，B 取值[0, 20]。我们添加偏移量 10 到 B 中，因此 B 取值[10, 30]。通过这种做法，就可以安全地将 A、B 特征合并，使用一个取值[0, 30]的特征取代特征 A 和 B。

EFB 算法能够将许多互斥的特征变为低维稠密的特征，就能够有效的避免不必要零值特征的计算。实际，通过用表记录数据中的非零值，来忽略零值特征，达到优化基础的直方图算法。通过扫描表中的数据，建直方图的时间复杂度将从 O(data)降到 O(non_zero_data)。当然，这种方法在构建树的过程中需要而额外的内存和计算开销来维持预特征表。我们在 LightGBM 中将此优化作为基本函数，因为当 bundles 特征是稀疏的时候，这个优化与 EFB 不冲突。

LightGBM 主要有以下的特点[4]：

（1） 基于 Histogram 的决策树算法

（2） 使用带深度限制的 Leaf-wise 的叶子生长策略

（3） 利用直方图相减来进一步加速训练

（4） 直接支持类别特征

（5） Cache 命中率优化

（6） 基于直方图的稀疏特征优化

（7） 多线程优化

2.2.3 技术

本项目使用目前十分流行的基于 Python 语言的 Scikit-learn 机器学习框架以及 XGBoost 和 LightGBM 官方提供的算法接口。Scikit-learn 来源于 2007 年的 Google Summer of Code 项目，最初由 David Cournapeau 开发。它是一个简洁、

高效的算法库，提供一系列的监督学习和无监督学习的算法，以用于数据挖掘和数据分析。Scikit-learn 的基本功能主要被分为六大部分：分类，回归，聚类，数据降维，模型选择和数据预处理。

总体上来说，作为专门面向机器学习的 Python 开源框架，Scikit-learn 可以在一定范围内为开发者提供非常好的帮助。它内部实现了各种各样成熟的算法，容易安装和使用，样例丰富，而且教程和文档也非常详细。

2.3 基准指标

本项目所采用的基准指标是在 Kaggle 上 Private LeaderBoard 的得分，具体来说，对 test.csv 中各商店的销售额进行预测，然后将预测结果提交到 Kaggle 上，利用 Kaggle 竞赛 Rossmann Store Sales 的 Private LeaderBoard 的得分作为基准，目标是最终的得分要达到 0.117 及以下。

第三章　实验与模型优化

3.1 数据预处理

根据在数据探索和可视化小节中的发现，对数据做出了如下的预处理。

1. 数据清洁度处理

分别将 train.csv，test.csv 数据和 store.csv 数据合并成一个表格数据。

2. 数据质量处理

(1) 将 Date 数据类型转化成 datetime64 类型。

(2)对 test 数据中 Open 特征的缺失值用 1 进行填充。

(3)分别对 train 和 test 数据表中的特征 CompetitionOpenSinceMonth，CompetitionDistance，CompetitionOpenSinceYear，Promo2SinceWeek，Promo2SinceYear 以及 PromoInterval 的缺失值用 0 值进行填充。

(4)去除 Open 和 Sales 为 0 的数据。

3. 数据变换

将 train 数据中的 Sales 值取对数。

4. 对定性特征进行编码

分别将=对 train 和 test 数据中的 StateHoliday，StoreType 以及 Assortment 分类特征进行编码。具体来说将 StateHoliday 特征的取值为 0、a、b、c 重新编码成 0、1、2、3。将 StoreType 特征取值为 a、b、c、d 重新编码成 1、2、3、4。将 Assortment 特征取值为 a、b、c 重新编码成 1、2、3。

5. 构造新特征[5]

(1) 将特征 Date 分解成新的特征 Year、Month、Day、WeekofYear，分别表示年、月、日以及一年中的第几周。

(2) 利用特征 CompetitionOpenSinceYear、CompetitionOpenSinceMonth 以及新特征 Year、Month 构建新的特征 CompetitionOpen，表示最近竞争对手开业有多少个月的时长。

(3) 利用特征 Promo2SinceYear、Promo2SinceWeek 以及新特征 Year、WeekOfYear 构造新特征 PromoOpen，表示持续促销开始有多少个月的时长。

6. 特征选择

将 train 数据中的 Customers、Sales、Date 特征丢弃，用于后面的模型的训练。

将 test 数据中的 Id、Date 特征丢弃，用于以后的模型的测试。

3.2 具体实现

将经过预处理后的 train 训练数据根据时间划分为训练数据以及交叉验证数据，在这里取 train 中后两周的数据作为交叉验证集，即处于 2015/7/18-2015/7/31 这一时间范围内的数据。然后分别使用 XGBoost 与 LightGBM 的默认参数建立回归预测模型，其默认参数，如表 3.1 和表 3.2 所示。

表 3.1 XGBoost 默认参数

参数	值
booster	gbtree
objective	reg:linear
eta	0.3
max_depth	6
min_child_weight	1
subsample	1
colsample_bytree	1
gamma	0
lambda	1
alpha	0
random_state	23

表 3.2 LightGBM 默认参数

参数	值
Boosting_type	gbdt
objective	regression
learning_rate	0.1
num_leaves	31
max_bin	255
min_data_in_leaf	20
bagging_fraction	1
bagging_freq	0
feature_fraction	1
lambda_l1	0
lambda_l2	0
min_split_gain	0

3.3 模型优化

由于上述 XGBoost 和 LightGBM 模型都是采用算法的默认参数，可以调整算法的参数，来进一步地提高分数。基于网格搜索法分别地对 XGBoost 和 LightGBM 进行参数调节。

XGBoost 中的参数分为常规参数，用于树提升的参数，学习任务参数以及命令行参数。XGBoost 有两种方式控制模型过拟合[6]：

(1) 第一种是直接控制模型的复杂度，包括 max_depth、min_child_weight 和 gamma；

(2) 第二种是增加随机性，加强模型的鲁棒性，参数包括 subsample 和 colsample_bytree，同时也可以减小步长 eta，增大 num_round。

因此，我们需要调节的参数有：eta、max_depth、gamma、min_child_weight、subsample、colsample_bytree、lambda 以及 alpha。

LightGBM 中的参数分为核心参数，学习控制参数，IO 参数，目标参数，度量参数，网络参数，GPU 参数以及模型参数。LightGBM 处理过拟合有几下几种方式[7]：

(1) 使用较小的 max_bin。

(2) 使用较小的 num_leaves。

(3) 使用 min_data_in_leaf 和 min_sum_hessian_in_leaf。

(4) 通过设置 bagging_fraction 和 bagging_freq 来使用 bagging。

(5) 通过设置 feature_fraction 来使用特征子抽样。

(6) 使用 lambda_l1, lambda_l2 和 min_gain_to_split 来使用正则。

(7) 尝试 max_depth 来避免生成过深的树。

对于本项目而言，进行调节的参数有 max_bin、num_leaves、min_data_in_leaf、bagging_fraction、bagging_freq、feature_fraction、lambda_l1,、lambda_l2 以及 min_gain_to_split。

各自调节完的参数，如表 3.3 和 3.4 所示。

表 3.3 调节后的 XGBoost 默认参数

参数	值
booster	gbtree
objective	reg:linear
eta	0.03
max_depth	10
min_child_weight	20
subsample	0.8
colsample_bytree	0.7
gamma	0
lambda	1
alpha	0.8
random_state	23

表 3.4 调节后的 LightGBM 默认参数

参数	值
Boosting_type	gbdt
objective	regression
learning_rate	0.05
num_leaves	140
max_bin	255
min_data_in_leaf	20
bagging_fraction	0.6
bagging_freq	8
feature_fraction	0.4
lambda_l1	0.1
lambda_l2	0.9
min_split_gain	0

在进行参数优化之后，最后还会将这两个优化之后的模型进行加权平均，用来作为最终的模型。

第四章　实验结果分析

4.1 模型验证

由于在之前的处理中，将 train 训练数据划分成两个部分：用于训练的数据以及用于验证的数据，因此我们可以通过验证集数据，来对模型进行评估和验证。由于 XGBoost 和 LightGBM 官方 API 接口 train 函数，本身就具有交叉训练的功能，因此直接使用官方接口。在没有进行参数调节前，XGBoost 和 LightGBM 模型在训练过程中终止 RMSPE 得分如表 4.1 所示。

表 4.1 基础模型训练终止 RMSPE 得分

Model	train-rmspe	eval-rmspe
XGBoost	0.114547	0.113521
LightGBM	0.14739	0.113133

其中，train-rmspe 代表模型在训练集上的 RMSPE 分数，eval-rmspe 表示模型在验证集上的 RMSPE 分数。

此时 XGBoost 模型的训练时间为 16min 46s，预测时间为 953ms。LightGBM 模型的训练时间为 3min 38s，预测时间为 1.13s。

在 XGBoost 和 LightGBM 模型调节参数后，此时在验证集上的得分如表 4.2 所示。

表 4.2 优化模型训练终止 RMSPE 得分

Model	train-rmspe	eval-rmspe
XGBoost	0.068837	0.099949
LightGBM	0.105873	0.104763

此时 XGBoost 模型的训练时间为 1h 18min 17s，预测时间为 8.86s。LightGBM 模型的训练时间为 9min 42s，预测时间为 8.35s。

对比表 4.1 和表 4.2，可以看出模型在进行参数调节后，在训练集上的 RMSPE 分数都降低了，说明模型拟合地更加精确。同时我们还可以看出，在训练集得分下降很多地情况下，在验证集上的得分却并没有下降或者上升太多，说明更加具有鲁棒性。

4.2 结果分析

由于之前都是在训练数据上进行的，下面来看看模型在测试集上的表现。利用上面建立的基础模型以及参数优化过的模型，分别对 test 数据进行预测，然后将预测结果提交到 Kaggle 上 Rossmann Store Sales 竞赛的 Leaderboard。在未进行参数调节前，两个基础模型在 Kaggle 上 Leaderboard 的 Private Score 得分如表 4.3 所示。

表 4.3 基础模型 Kaggle 得分

Model	Private Score
XGBoost	0.13440
LightGBM	0.12910

使用调节后的模型再去预测，此时的得分如表 4.4 所示。

表 4.4 优化模型 Kaggle 得分

Model	Private Score
XGBoost	0.11514
LightGBM	0.11610

可以看出，进行参数优化过的两个模型的分数提升了很多，而且单独的 XGBoost 和 LightGBM 的得分都降到了 0.117 以下。下面考虑将这两个模型进行集成，来看看最终的得分情况。

设 XGBoost 的结果为 xgb_prod，LightGBM 的结果为 lgb_prod，取 0.57*xgb_prod+0.43*lgb_prod 作为最终的结果提交到 Kaggle 上去，其得分如图 4.1 所示。

图 4.1 最终得分

由图 4.1 可知，最终的 Private Score 为 0.11429，满足项目的要求，也说明了最终的模型是可行的。

第五章　结论与展望

5.1 结论

本项目基于 XGBoost 和 LightGBM 方法对 Rossmann1115 家商店的销售额进行预测，最终的预测结果也到达要求。分别基于 XGBoost 和 LightGBM，画出特征的重要性，其结果如图 5.1 和图 5.2 所示。

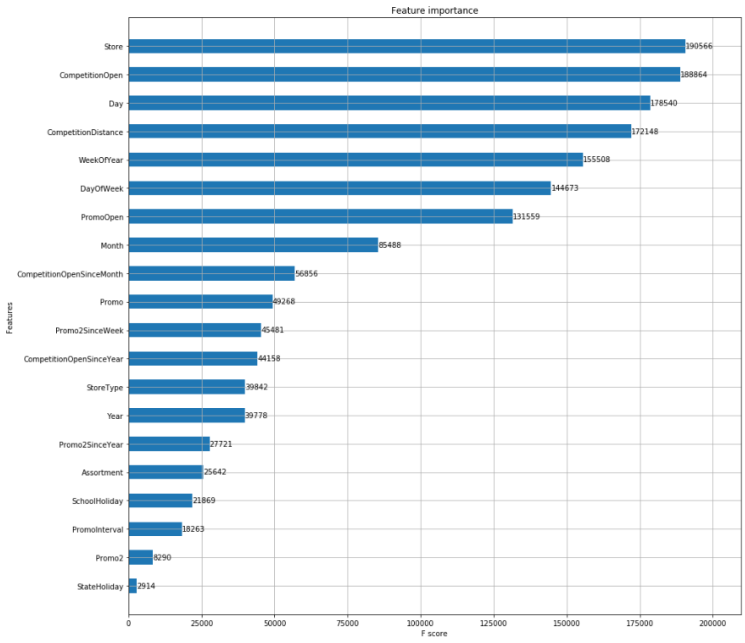

图 5.1 XGBoost 特征重要性

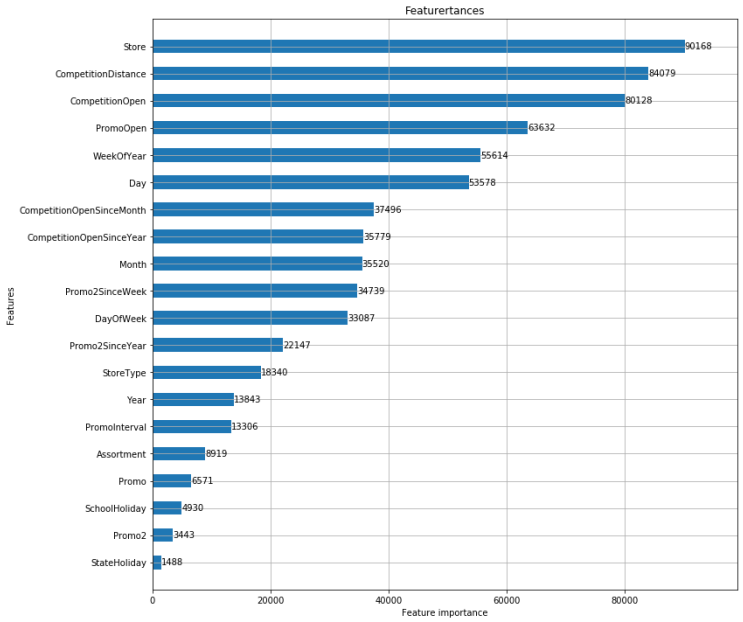

图 5.2 LightGBM 特征重要性

从图 5.1 和 5.2 中可以看出，虽然 XGBoost 和 LightGBM 对特征重要性的排序不一样，但是在这个两个模型中，重要性都靠前的几个特征有 Store、CompetitionOpen、Day、CompetitionDistance、WeekofYear，还可以看出来这些特征大部分都是与时间有关的特征。

5.2 项目思考

本项目主要分为 4 个部分：数据预处理、数据探索、特征工程、模型选择与优化，其中数据探索和特征工程这两个部分是十分重要的。数据探索在整个流程中是起到承上启下的作用，因为数据预处理和特征工程往往不能一次性地就能得到好的结果，需要通过数据探索来不断完善。也只有在对数据有了充分了解的前提下，才有可能对数据做出比较合理的处理，才能做好特征工程。对于机器学习来说，特征工程是决定上限的关键，而算法只是去逼近这个上限。在特征工程中，特征预处理，特征选择都有各自对应的方法，就本人而言，我认为特征构造是比较关键也是最困难的部分，因为这需要对数据有很充分的理解。在模型优化过程中，我们需要通过参数调整，来得到比较理想的结果。目前比较常用的方法有网格搜索法，这种方法比较耗时，可以尝试使用贝叶斯优化来调整参数，不仅训练时间会降低，而且最终的效果也很不错。

5.3 展望

本项目虽然在样本数据上满足了要求，但是依然还有一些地方有待提高，具体如下：

1. 参数调优：对于模型参数的调节不一定是最优的，因此还可以进一步地优化。

2. 特征工程：目前使用的特征并不多，而且还可以构造出更多的特征，例如可以搭建一个神经网络去构造新的特征，用来作为模型的输入，相信能取得更好的结果。

3. 模型尝试：由于本项目其实是跟时间序列相关的，因此可以尝试其他处理时间序列的模型。

参考文献

[1] Tianqi Chen,Tong He. Higgs Boson Discovery with Boosted Trees[C]. JMLR: Workshop and Conference Proceedings, 2015 (42): 69-80.

[2] https://blog.csdn.net/hfzd24/article/details/76889428.

[3] Wang, D., Zhang, Y. and Zhao, Y. LightGBM: An Effective miRNA Classification Method in Breast Cancer Patients.Proceedings of the 2017 International Conference on Computational Biology and Bioinformatics, Newark, NJ, 18-20 October 2017, 7-11.

[4] https://blog.csdn.net/u011630575/article/details/79616530

[5] https://www.kaggle.com/ananya77041/rossmann-store-sales/randomforestpython/code

[6] http://xgboost.apachecn.org/cn/latest/how_to/param_tuning.html.

[7] http://lightgbm.apachecn.org/cn/latest/Parameters-Tuning.html.