机器学习工程师纳米学位毕业项目
基于集成树模型的零售店销量预测算法

李乙侠

2019年12月15日

目 录

1. 定义

1.1. 项目概述

项目利用了来自Rossmann商店的31个月真实数据，以预测接下来六个月的商店销量。商店销量预测具有重要的意义，它可以为零售企业的管理者提供一个预期销量，来更好的帮助商店优化商品配送问题，从而提高效率。本文基于决策树模型，建立了多种商店销量预测模型。

项目提供了三个数据集：训练集、测试集以及商店数据集。训练集和测试集主要用于训练模型，而测试集则可以对最终模型的预测效果进行量化。训练集中包含了1115家商店2014-2016年的日期、以及是否处于节假日；商店数据集中包含了1115家商店的详细信息，包含了商店类型、产品类型、竞争者和促销信息。

本文首先对三个数据集进行了清洗，对异常数据和缺失数据进行了转换，接着继续了简单的描述统计。在对数据集有大概的了解后，本文利用决策树模型对商店销量进行了初步预测，通过与其他算法的比较，发现决策树模型在销量预测上具有较高的准确性；在第二部分，本文通过不同的集成算法，建立了多个决策树集成模型，进一步提高了预测准确率。在第三部分，本文针对最优集成模型进行了优化，将预测准确率进一步提高。

1.2. 问题陈述

项目将需要通过大量的历史商店销售信息，来对不同商店销量进行预测，我们将首先使用决策树模型来对商店销量进行预测，通过网络搜索来寻找最优参数，并以此模型为基模型进行集成；随后利用不同的集成算法，对决策树模型进行提升，来提高预测性能，最终预测出不同商店的销量趋势。

1.3. 评价指标

本文将使用均方根误差（Root Mean Square Percentage Error, RMSPE）来对模型拟合效果进行量化：

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{y_i - \hat{y}_i}{y_i})^2}$$

均方根误差是预期值 $\hat{y}_i$ 与真实值 y_i 的偏差的平方与观测次数 n 比值的平方根，均方根误差可以衡量观测值与真实值之间的偏差，当均方根误差越小时，预测效果越好。在本文中，当实际值 y_i 为 0 时，该数据将会被跳过不被计算。

2. 分析

2.1. 数据探索

2.1.1. 数据清洗

首先我们将对三个数据集进行清洗，训练数据类型如下：

表 1 训练集数据类型及取值范围

变量名	变量含义	变量类型	取值范围	备注
Store	商店编号	离散型 - int	[1,1115]	
DayOfWeek	星期	离散型 - str	[1,7]	
Date	销售日期	离散型 - str	[2013-01-01,2015-07-31]	
Sales	销售量	连续型 - float	[0,41551]	
Customers	顾客数目	离散型 - int	[0,7388]	
Open	是否营业	分类型 - int	[0,1]	
Promo	是否有促销活动	分类型 - int	[0,1]	
StateHoliday	国定假日	分类型 - str	[0,a,b,c]	a=公共假日，b=复活节，c=圣诞节，0=无
SchoolHoliday	学校假日	分类型 - int	[0,1]	是否受到公共学校关闭的影响

我们通过对该数据中每一项类别对应的数据进行筛选，可以看到 StateHoliday 中包含了 ['0', 'a', 'b', 0] 四种类型，因此可以将 0 类型的数据转换为 '0' 型数据，其他变量均无缺失值。

测试数据类型如下：

表 2 训练集数据类型及取值范围

变量名	变量含义	变量类型	取值范围	备注
ID	记录编号	离散型 - int	[1,41088]	
Store	商店编号	离散型 - int	[1,1115]	
DayOfWeek	星期	离散型 - str	[1,7]	
Date	销售日期	离散型 - str	[2013-01-01,2015-07-31]	
Sales	销售量	连续型 - float	[0,41551]	
Customers	顾客数目	离散型 - int	[0,7388]	
Open	是否营业	分类型 - str	[0,1]	
Promo	是否有促销活动	分类型 - int	[0,1]	
StateHoliday	国定假日	分类型 - int	[0,a,b,c]	a=公共假日，b=复活节，c=圣诞节，0=无
SchoolHoliday	学校假日	分类型 - int	[0,1]	是否受到公共学校关闭的影响

对测试集进行清洗时发现，'2015-09-05' 至 '2015-09-17' 中除了周日，其他日期的 Open 数据都为缺失值，且均为 622 号商店数据。通过对 622 号商店的数据观察，可以发现其正常的开业时间为周一至周六，而考虑到在这段时间内没有其他法定假期，因此将其缺失数据填充为 1 。

商店数据类型如下：

表 3 商店数据集数据类型及取值范围

变量名	变量含义	变量类型	取值范围	备注
Store	商店编号	离散型 - int	[1,1115]	
StoreType	商店类型	离散型 - str	[a,b,c,d]	
Assortment	产品组合	分类型 - str	[a,b,c]	描述产品组合级别： a=基本，b=附加，c=扩展
CompetitionDistance	竞争者距离	连续型 - int	[20,75860]	单位：米
CompetitionOpenSinceMonth	竞争者开业月份	离散型 - int	[1,12]	nan 为未知
CompetitionOpenSinceYear	竞争者开业年份	离散型 - int	[1900,2015]	nan 为未知
Promo2	商店的连续促销	分类型 - int	[0,1]	0=未参与，1=正参与
Promo2SinceWeek	促销开始周数	离散型 - int	[1,50]	nan 为未知
Promo2SinceYear	促销开始年份	离散型 - int	[2009,2015]	nan 为未知
PromoIntervalu	促销期间	离散型 - str	[Jan,Dec]	nan 为未知

通过对商店数据集各个变量进行箱线图分析，我们发现在 CompetitionOpenSinceYear 中，存在着一个明显的异常值：

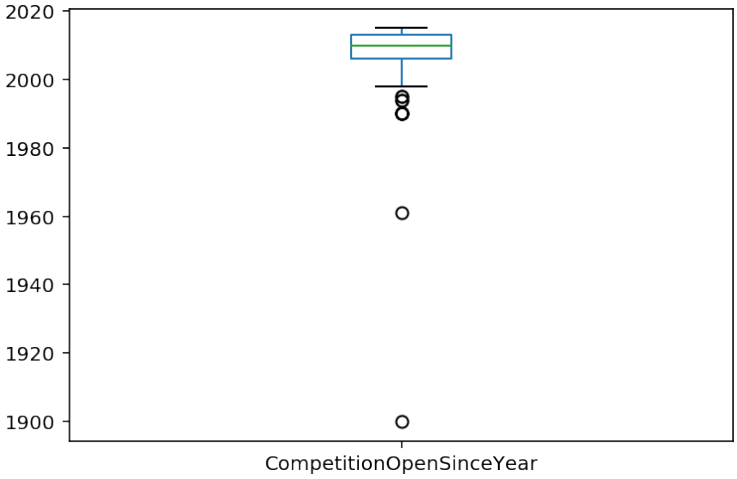

图 1 CompetitionOpenSInceYear 箱线图

由于没有其他的数据支撑，我们有理由相信该 1900 年开业的竞争者数据为异常值，因此可以将其赋值为未知。

而由于部分商店并没有存在着竞争者，因此其 CompetitionDistance 与 CompetitionOpen-Time将出现缺失值，考虑到当无竞争者时，可近似等同于竞争者的距离无限远，且开业时间为 0 ，故针对三种不同缺失情况，我们采取了不同的缺失值处理方法：

1. CompetitionDistance 缺失，CompetitionOpenTime 不缺失：使用 CompetitionDistance 均值填充；

2. CompetitionDistance 不缺失，CompetitionOpenTime 缺失：使用 CompetitionOpen-Time 均值填充；

3. CompetitionDistance 缺失，CompetitionOpenTime 缺失：使用 CompetitionDistance 最大值填充 CompetitionDistance，使用 0 填充 CompetitionOpenTime。

由于部分商店并没有促销策略（Promo2=0），因此其对应的 Promo2SinceYear 、Promo2SinceWeek 和 PromoInterval 为缺失值，故我们将对应数据赋值为 0 。

至此，我们已经完成了对数据集的清洗。

2.1.2. 数据联合

通过对训练数据和商店数据在‘Store’上进行联合，我们可以得到一个拓展的数据集。

2.2. 数据可视化

我们可以通过分析不同变量下商店销量的差异，如图 2 所示：

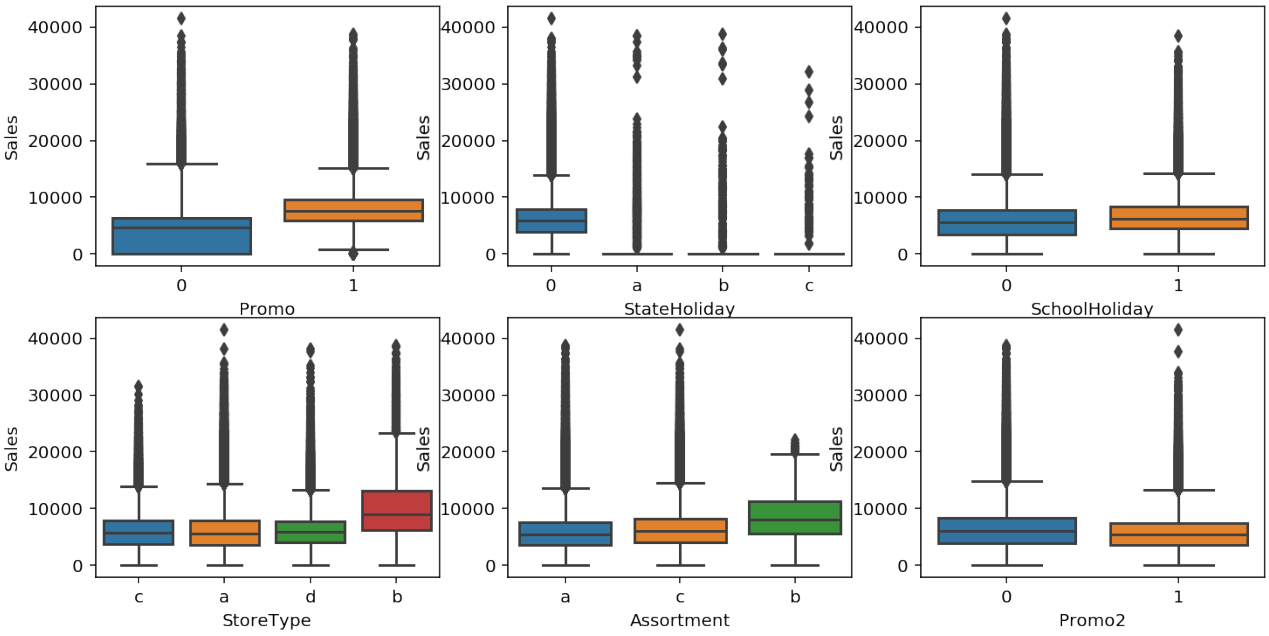

图 2 不同变量下商店平均销售量箱线图

从这 6 幅图中可以分别发现：

1. 可以看出当商店正处于促销期（Promo=1）时销量高于平时时期；
2. 当不处于法定假期（StateHoliday=0）时销量高于其他时期；
3. 学校是否放假对销量没有明显影响；
4. b 类商店的销量高于其他商店；
5. 销售 b 类商品组合的商店销量稍高于其他商店；
6. 是否有周期性的促销策略对销量没有明显提高。

接着，我们分析了一周中每一天的平均销量，如图 3 所示，周日的销量几乎为零，这也验证了大多数的商店周日都是关闭的。

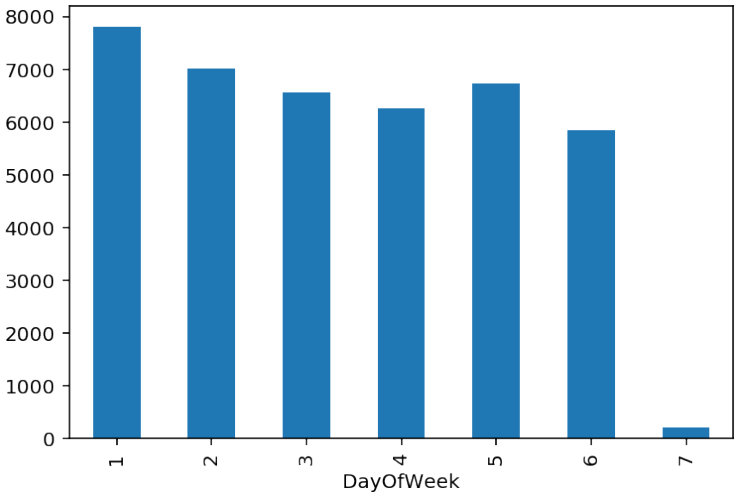

图 3 不同时间下商店平均销售量直方图

最后，我们对不同年份的商店平均销量绘制了折线图，如图 4 所示：

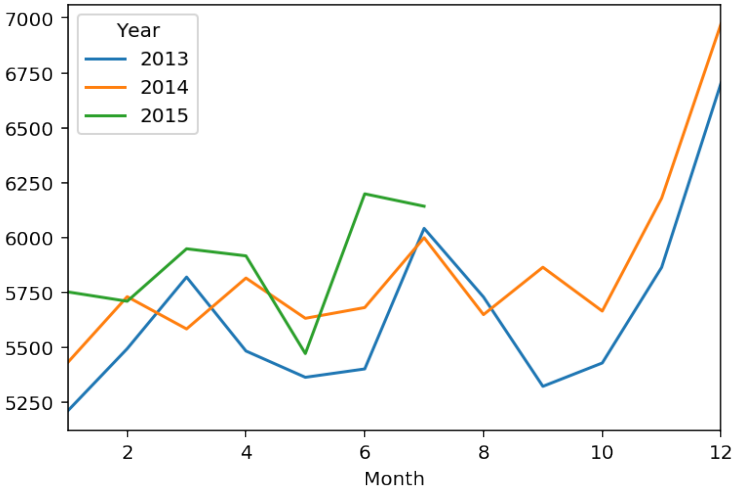

图 4 不同年份下商店平均销售量直方图

从图中可以发现，销量呈现逐年递增的趋势，且在 12 月时销量达到最高，可能的原因有圣诞节以及黑五等节日导致了销量激增。

3. 模型算法

3.1. 树模型

3.1.1. 决策树

决策树是一种基本的分类与回归方法，其在每一个特征进行最优划分，相当于在特征空间与类空间上的条件概率分布。决策树学习通常包含三个步骤：特征选择、决策树生成和决策树的修剪。这些决策树学习的思想主要来源于 Quinlan 在 1986 年提出的 ID3 算法和 1993 年提出的 C4.5 算法，以及由 Beriman 等人在 1984 年提出的 CART 算法。这三个算法最大的不同在于特征选取的指标上，分别使用信息增益、信息增益率以及基尼系数作为特征选择指标。[1]

ID3 算法特征选择指标为信息增益 $g(D,A)$ ，其为集合 D 的经验熵 $H(D)$ 与特征 A 给定条件下的经验条件熵 $H(D|A)$ 之差：

$$g(D,A) = H(D) - H(D|A)$$

其中经验条件熵 $H(D|A)$ 为划分子代后的平均信息熵：

$$H(D|A) = \sum_{i=1}^{n} P(A_i)H(A_i)$$

$$H(A_i) = E[I(A_i)] = -E[log_2(P(A_i))] = -\sum_{i=1}^{n} P(A_i)log_2(P(A_i))$$

C4.5 算法特征选择指标为信息增益率 $g_R(D,A)$ 定义为其信息增益 $g(D,A)$ 与集合 D 的经验熵 $H(D)$ 之比:

$$g_R(D,A) = \frac{g(D,A)}{H(D)}$$

ID3 和 C4.5 算法在完成决策树生成和修剪后，可以根据每个分支上的平均值作为回归输出值。

CART 回归算法使用平方误差最小准则进行特征选择，生成二叉树，假设输入空间划分为 M 个单元 $R1,R2,...,R_M$ 并且每个单元 R_m 上有一个固定的输出值 C_m ，则回归树模型可以表示为：

$$f(x) = \sum_{m=1}^{M} C_m I(x \in R_m)$$

当输入空间划分确定时，可以用平方误差 $\sum_{x_i\ in R_m} (y_i - f(x_i))^2$ 来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值，每个单元 R_m 上的 C_m 的最优值 $\hat{C}_m$ 是 R_m 上所有输入实例 x_i 对应的输出 y_i 的均值，即：

$$\hat{C}_m = Ave(y_i|x_i \in R_m)$$

CART 回归决策树生成时通过遍历所有输入变量，找到最优的切分变量 j 并以此将输入空间划分为两个区域，接着在每个区域重复上述划分过程直至满足停止条件为止。

3.2. Bagging

个体学习器虽然能有较好的效果，但是通过集成可以以更高的偏差可以换取更低的方差，提高模型的泛化能力，集成有多种方式，如通过不同的基学习器（Voting），在分类时采用投票、在回归时采取均值的方式，又或者通过同一基学习器，利用不同的样本子集（Bagging）或者提升方法（Boosting）来提高模型泛化能力，本节将介绍以树为基础的集成算法。

Bagging 为采取多个相同基学习器、对样本采取有放回抽样（Bootstrap），其优点有泛化能力强且可并行化，能提高训练速度。

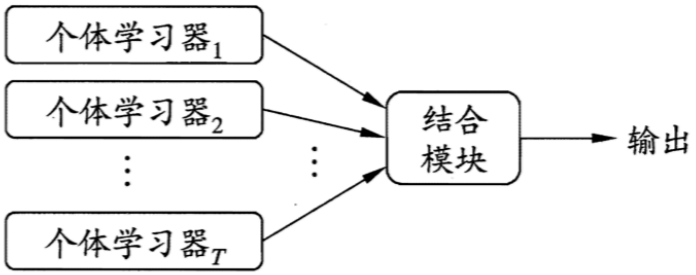

图 5 Bagging 算法示意图

3.2.1. 随机森林

随机森林与 Bagging 相似，都是通过同一基学习器以及随机子样本空间进行集成，不同的是其基学习器为决策树（无剪枝）且同时对特征空间进行随机抽样，以增强其泛化能力。

3.2.2. 极端随机树

随机森里内单颗树的生长过程中，每个节点在分裂时仅考虑了一个随机子集包含的特征，如果我们对每个特征使用随机阈值，而不是搜索得出的最佳阈值，则可以让决策树生长得更加随机。这种由极端随机决策树组成的森林，被称为极端随机树，同样也是以更高的偏差换去了更低的方差。

3.3. Boosting

Boosting（提升法）是指将几个弱学习器结合成一个强学习器的集成方法。大多数提升法的整体思路时循环训练预测器，每次对前序做出一些改正，目前最流行的方法是 AdaBoost 和梯度提升。

3.3.1. AdaBoost

AdaBoost（Adaptive Boosting）的思路是利用新的学习器对前序进行修正，通过对前序拟合不足的训练实例赋予更高的权重，使后续学习器更加注重于这些欠拟合的实例。

AdaBoost 集成算法有多种推导方式，较容易理解的是基于“加线性模型”（Additive Model），即利用基学习器 h 的线性组合：

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

来最小化指数损失函数：

$$\ell_{exp}(H \mid D) = E_{x\sim D}\left[e^{-f(x)H(x)}\right]$$

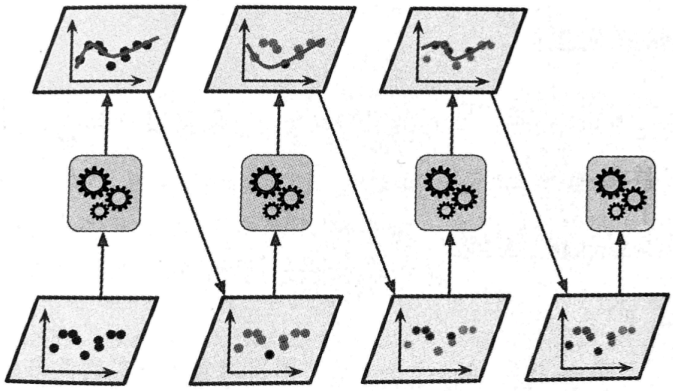

图 6 AdaBoost 算法示意图

3.3.2. GBRT

除 AdaBoost 外，另一个常用的提升方法就是梯度提升（Grandient Boosting），但其不像 AdaBoost 那样在每个迭代中调整实例的权重，而是让新的预测器针对前一个预测器的残差进行拟合。GBRT（梯度提升回归树）就是使用决策树作为基学习器，利用最小化二次残差来进行提升回归。

3.3.3. XGBoost

XGBoost 的提升模型与 GBRT 相似，都是对残差进行提升，但不同的是分裂节点选取的时候不一定是最小平方损失，而是考虑了 GBRT 中树模型的复杂度而加入了正则化项，因此更不易过拟合。XGBoost 的损失函数由两部分构成，前者优化经验误差，后者是控制泛化误差：

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

其中后者代表了树的复杂程度，由子叶树木 γT 和 L2 范数组成：

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$$

3.4. Voting

Voting 是最简单的集成方法，其通过对已训练好的多个弱学习器的预测结果进行简单平均或加权平均，可提高模型的泛化能力。

3.5. 基准模型

本文将以 Kaggle 竞赛平台 Rossmann Store Sales 竞赛项目 Private 榜前 1% 的 RMSPE 0.11773 为基准数据，通过不断优化本文算法来达到这个分数。

4. 方法

4.1. 数据预处理

4.1.1. 样本选择

由于我们只需要预测销量大于 0 的数据，因此在对模型进行训练时，我们可以将训练集中销量为 0 的数据舍弃。

4.1.2. 特征转换

对于数据中的 Date 属性数据，由于其为字符属性，如“2016-01-01”，并不利于算法的处理，因此我们将其拆分为了三类整型数据：“year”、“month”以及“day”。

对于数据中的 PromoInterval 属性数据，我们通过 BagOfWord 将其转换为取值范围从 Jan 至 Dec 的 12 维向量，如将“Jan July Dec”编码为：100000100001。

4.1.3. 特征编码

对于数据中的 StoreType、Assortment 与 StateHoliday 属性，我们将其原本的字符属性值按照下表对应关系映射到了数值空间上，如将 Assortment 取值为 “a” 的值映射为 1 。

表 4 映射关系

字符属性	数值属性
0	0
a	1
b	2
c	3

4.1.4. 特征选择

由于结果了数据联合、BagOfWord 等数据处理后，输入空间纬度已经达到了 29 维，这意味着模型的训练时长也将大大提高，因此我们利用 Lasso 变量选择法，将训练维数降低至了 14 维，以加速训练时间的同时提高模型精度。

4.1.5. 对数变换

由于数据 Sales 属性变量分布非正态，呈现明显的右偏，因此我们可以通过 Box-Cox 变化来修正这一不足，从分布图来看，经过对数变换后 Sales 数据分布接近于正态分布。

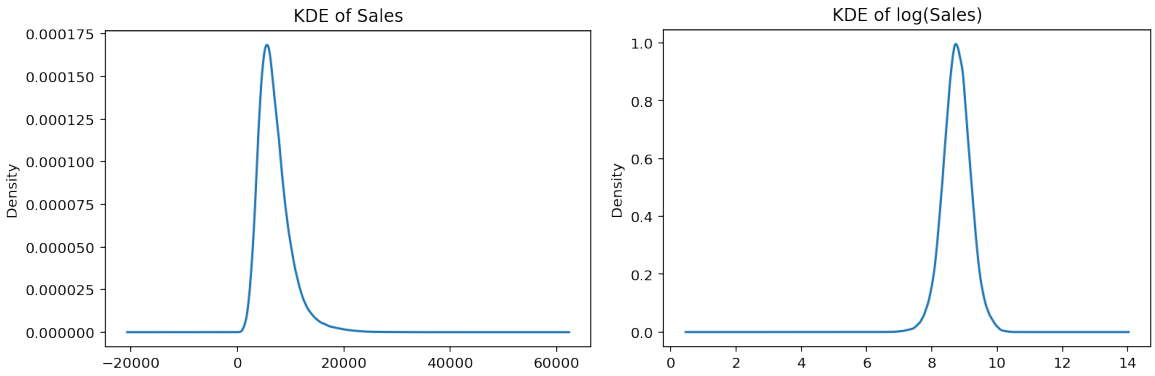

图 7 对数变换前（左）与对数变换后（右）的 Sales 数据分布图

4.1.6. 数据集划分

本文将数据集按照 75 : 25 的比例随机划分了训练集和验证集，通过交叉验证来选取最优参数。输入空间包含属性如下表所示：

表 5 变量选择后的输入空间属性

Month	Year	DayOfWeek	Assoerment
CompetitionDistance	CompetitionOpenTime	Promo	Promo2
Promo2SinceYear	PromoInterval_Dec	PromoInterval_Jun	PromoInterval_Mar
PromoInterval_Sept	StoreType		

4.2. 执行过程

对于决策树模型，通过大范围的网络搜索法，可以将参数范围缩小至以下范围：

表 6 决策树参数搜索范围

参数	搜索范围
Max Depth	25 30 35 40 45 50
Min Samples Split	7 10 15 20 30

通过在以上参数中再次进行搜索并对模型进行 5 5折交叉验证，我们可以得到以下可视化得分分布图（图 8）。在下图的左侧为训练集得分，右侧为测试集得分，颜色越深代表着训练结果越接近真实值。可以看到，在训练集上，随着 Min Samples Split 从 7 增加到 20 ，训练效果逐渐降低，而测试效果逐渐提高，当参数继续增大到 30 时，测试效果不再提升，因此可以将 Min Samples Split 设定为 15 ；而随着 Max Depth 逐渐增大到 35 ，训练误差和测试误差均有所下降，而当其继续增大到 50 时，两类误差均不在变化，因此可以将 Max Depth 设定为 35 。

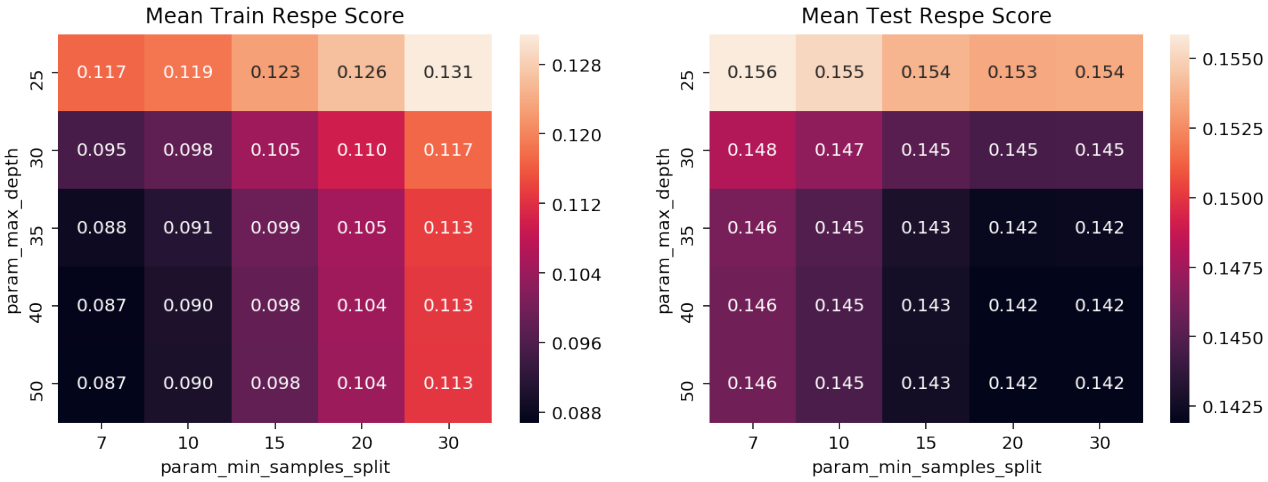

图 8 决策树模型网络搜索训练误差（左）及测试误差（右）

通过以上参数设置，我们的决策树模型最终在测试集上的误差 Respe 为 0.16006 。我们将在后续的集成算法中，以此最优模型做为基学习器，对训练效果做进一步提升。

我们又分别生成了一个包含 100 颗树的随机森林和极端随机树，得到测试误差分别为 0.16447 和 0.16773 ，效果非常接近。通过对决策树、随机森林以及极端随机树的投票集成，将训练误差进一步降低至 0.15886 。

在决策树模型的基础上，我们又测试了其他的集成模型，最终总结如下表：

表 7 各模型得分与参数设置

模型	训练时间(s)	训练集得分	训练集预测时间(s)	测试集得分	测试集预测时间(s)	Kaggle Private Score	模型参数
决策树	353	0.19027	0	**0.16006**	0	**0.15000**	min_samples_split=15 max_depth=35
随机森林	162	0.13618	25	**0.16447**	5	**0.14688**	n_estimators=100 base_model=Decision Tree
极端随机树	126	0.13985	25	**0.16773**	10	**0.14787**	n_estimators=100 base_model=Decision Tree
Voting	0	0.15212	50	**0.15885**	18	**0.14305**	base_model=[Decision Tree, Random Forest, Extra Forest]
Bagging	91	0.18787	7	**0.15228**	2	**0.14106**	n_estimators=30 base_model=Decision Tree
Adaboost	115	0.15907	5	**0.15879**	1	**0.15510**	n_estimators=25 base_model=Decision Tree
GBRT	207	0.15961	9	**0.15607**	3	**0.14184**	n_estimators=30 base_model=Decision Tree
XGBoost	2812	0.10596	89	**0.12963**	14	**0.12041**	n_estimators=1000 base_model=Decision Tree subsample=0.9 colsample_bytree=0.7

其中，XGBoost 的训练 rmse 与测试 rmse 在结果大约 150 轮训练后趋于收敛（图 9 上），而训练 rmspe 与 测试 rmspe 则大约在 1000 轮训练后趋于收敛（图 9 下）。

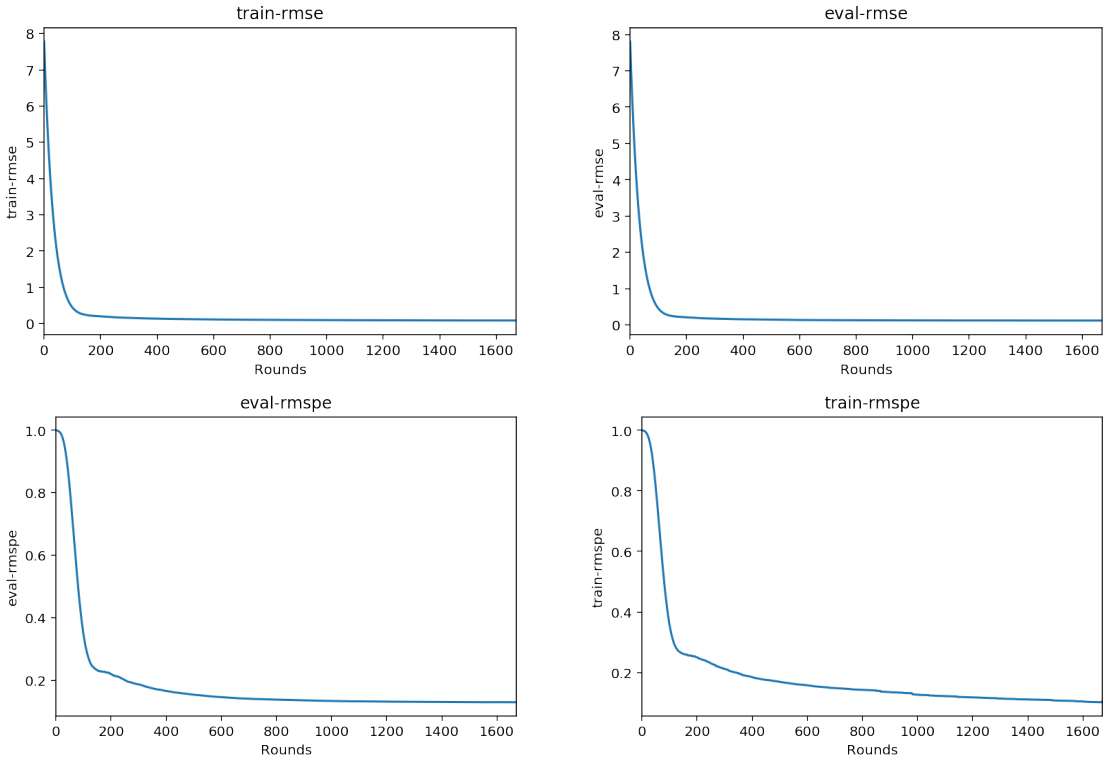

图 9 XGBoost 训练误差图

4.3. 完善

在本节中，我们选取在之前表现最佳的 XGBoost 对其进行优化完善，为了更加直观地观察拟合程度、发现其中缺陷，我们随机选取了两个店铺进行销量预测，并画出估计值与实际值之比的曲线，如下所示：

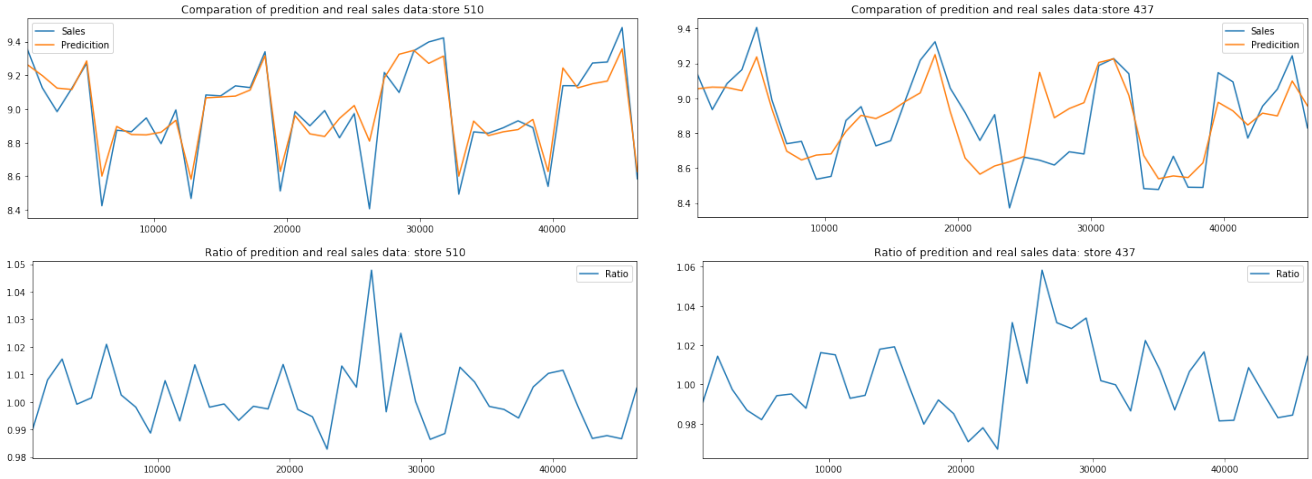

图 10 XGBoost 拟合效果（上）以及相应的估计值与实际值之比（下）

从图中可以看到，XGBoost 模型已经基本可以预测出商店销量的大体趋势，但是整体的预测效果均高于实际值（图 10 下图折线值大于 1 ）。考虑到在对 XGBoost 模型进行训练时并没有考虑不同商店的特征，因此我们可以对每个商店进行单独优化：通过对预测值进行一定范围（0.9 ~ 1.1）的放大或缩小，来选取使各个商店的预测效果达到最优的校正系数。

经过校正后的模型，在测试集上的误差降低到了 0.11617 ，相比于校正前的 0.12963 有了明显的提高，并且最终在 Kaggle Private 榜上达到了 0.11145 ，超过了基准模型前 10% 的要

求。

Submission and Description	Private Score	Public Score	Use for Final Score
submission.csv 7 days ago by XavierLee XGBost_New	0.11145	0.10845	☐

图 11 Kaggle Private Score

5. 结果

本文通过对各种基于树的集成模型进行测试，选取了其中性能最好的模型进行优化，最终达到了毕业要求。从图 9 中的预测折线图可以发现，XGBoost 模型较为合理，能够很好的预测不同时间下的销量变化，与期待的结果基本一致。

通过对 XGBoost 模型进行优化，本文的最终模型略优于基准模型，且实实在在地预测出了不同商店不同时间的销量值，能够为商店决策者提供一定的参考数据，提前估计商店销量，用于优化商品分配等问题。

6. 项目结论

6.1. 结果可视化

我们可以利用 XGBoost ，对数据的属性重要性做出判断：

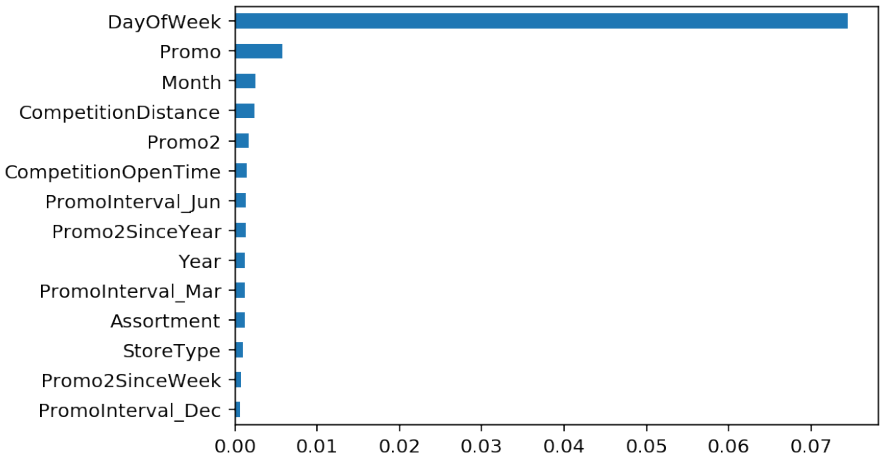

图 12 基于 XGBoost 模型的特征重要

上图中左图是对所有属性的重要性值进行了排序，可以看到对与销量影响最大的是星期几，通过在 2.2 中（图 3 ）对不同工作日的销量进行画图，我们也可以发现这一明显特征。其次是是否促销，可以看到促销对于销量的影响仅次于工作日的影响，因此店铺可以考虑加大促销力度来拉动销量。

6.2. 对项目的思考

本文从开篇对数据进行了清洗以及统计分析，在清洗的过程中发现部分数据缺失、异常，基于实际意义，本文对其一一进行了处理，以为下一步训练提供了一个可靠的数据集。通过可视化数据，我们对数据的各类属性有了一个初步的认识。

在第二部分，我们详细地介绍了本文使用的所有算法的原理以及区别，从单个模型到不同的集成策略，逐步地泛化了模型的性能。

在第三部分，我们首先对数据进行了预处理，通过对字符属性以及时间属性的转换，将样本数据转换为能够提供给算法训练的实例；然而在属性的转换过程中，数据维度迅速上升，因此本文引入了 Lasso 对属性进行了选择，降低了输入空间的维度。接着我们通过决策树模型，介绍了利用网络搜索法以及可视化对超参数进行选择，并通过不同的集成策略，泛化了模型的性能。在对各类集成模型进行测试后，我们选取了其中性能最佳的 XGBoost 集成算法进行优化，通过为每个商店寻找一个最优的校正系数，进一步降低了误差，并达到了基准模型的水平。

在本文中，主要的困难在于集成模型的训练时间长、参数不易调节，这使得无法更加密集地所有参数，且可能陷入局部最优。好在最终模型依然能够对销量做出较好的预测，同时也可以应用于其他领域的时间序列预测问题。

6.3. 需要提出的改进

在本文中，由于时间以及精力的问题，并没有测试神经网络在时间序列预测问题上的效果，在深度学习中，模型能够捕捉到更多传统算法所无法捕捉到的细节，这或许能过为某些特殊日期的销量进行更加精准地预测。

其次，在训练样本上，本文只考虑了商店的周边环境因素以及时间因素，没有考虑到天气因素，而天气因素对于商店销量的影响也是非常明显的，因此在接下来的工作中可以考虑扩大训练样本属性已提高精度

7. 参考文献

[1] 李航. 统计学习方法[M]. 北京: 清华大学出版社, 2012: 55-66.