机器学习纳米学位

毕业项目

姜明

2018年5月2日

I. 问题的定义

项目概述

这是Kaggle的一个竞赛项目，目标是帮助Rossmann连锁药妆店的经理预测店铺未来6周的销售额，准确的预测店铺未来一段时间的销售额，能够帮助经理更加合理的安排员工的工作时间，同时也有利于提升员工的工作效率，可以看出这是一个典型的回归问题。

本项目使用的是XGBoost模型，XGBoost方法是由传统的GBDT方法发展而来，而GBDT的基学习器是CART，也就是基本的决策树模型。XGBoost方法最近几年屡次在各大数据比赛中获得第一[1]，该项目来自于kaggle，这个比赛的第一名使用的算法也是xgboost[2]。

该项目使用的数据集是由Kaggle提供，数据来源于坐落在德国的1115家店铺的销售数据（train.csv）和店铺信息（store.csv）数据。train.csv是店铺的销售数据，包含了每个店铺每天的**销售额**以及其他特征信息，比如说是否节假日，当天是否有促销活动等，该数据集中一共有1017209条数据，时间跨度为2013-01-01到2015-07-31。store.csv是1115家店铺的信息数据，共有1115条数据，主要特征有竞争对手开业时间，竞争对手距离，是否有长期促销活动等。模型需要预测的是未来6周每个店铺每天的销售额。

问题陈述

本项目的数据集来自真实世界中Rossmann连锁药妆店的销售数据和店铺数据，目标是要预测1115家店铺未来6周的销售额，这是一个标准的监督学习问题，同时也是一个时间序列的回归问题。

本项目一个很大的挑战在于，我们需要从现有的数据集中探索出店铺销售额的影响因素，从而才能预测店铺的销售额。另一个挑战在于，其预测周期为6周，较长的预测周期很有可能会给预测带来较大的误差，怎么才能找到影响店铺长期销售额的因素也很重要。

总之，大量的特征工程和数据分析需要花费较多的时间。寻找影响店铺销售额的影响因素需要从现有数据集的字段出发，比如说，可以从store.csv数据集获取更多店铺的信息，店铺的分类水平(Assortment)，促销活动(Promo)以及竞争对手店铺的距离(ComprtitionDistance)等都是影响销售额的影响因素。参考Gert[3]和Anton[4]的工作来构造更多的特征。

评价指标

本项目的评价指标由kaggle给出， Root Mean Square Percentage Error (RMSPE)，而且有比赛的结果作为对比。因此，选择RMSPE作为评价指标，定义为：

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

其中y_i表示真实的销售额，$\hat{y_i}$表示销售额的预测值。当销售额为0时，不带入计算。该指标的含义是预测值与真实值之间的误差占真实值比例的反映，RMSPE的值越小代表预测越准确。

II. 分析

数据的探索

特征探索

本项目主要使用两个数据集：train.csv（train）和store.csv（store）。其中train代表店铺的销售数据，store代表店铺的信息。

表 1 train的前6条数据

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

表1展示的是train的前6条数据，从表中可以看出该数据集一共包含了9个字段，每个字段的含义为：Store（店铺ID），DayOfWeek（星期几，5代表周五），Date（日期），Sales（销售额），Customers（到店人数），Open（是否开业，1代表开业，0代表未开业），Promo（是否有促销，1代表有促销活动，0反之）等，关于个字段的含义详见Rossmann Store Sales Data。

表 2 store的前6条数据

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
0	1	c	a	1270.0	9	2008	0	NaN	NaN	NaN
1	2	a	a	570.0	11	2007	1	13.0	2010.0	Jan,Apr,Jul,Oct
2	3	a	a	14130.0	12	2006	1	14.0	2011.0	Jan,Apr,Jul,Oct
3	4	c	c	620.0	9	2009	0	NaN	NaN	NaN
4	5	a	a	29910.0	4	2015	0	NaN	NaN	NaN

表2展示的是store的前6条数据，从表中可以看出，该数据集一共包含了10个字段，每个字段的含义为：Store（店铺ID），StoreType（陈列水平），Assortment（摆放水平）等，对于每个字段包含的详细信息，详见Rossmann Store Sales Data。

表 3 store的特征探索

	缺失值占比	最大值	最小值	均值
Store	0%	--	--	--
StoreType	0%	--	--	--
Assortment	0%	--	--	--
CompetitionDistance	0.2690%	75860.00	20.00	5404.90
CompetitionOpenSinceMonth	31.74%	--	--	--
CompetitionOpenSinceYear	31.74%	--	--	--
Promo2	0%	--	--	--
Promo2SinceWeek	48.78%	--	--	--
Promo2SinceYear	48.78%	--	--	--
PromoInterval	48.78%	--	--	--

> 注：--代表不存在数据。

表3展示了store数据集中每个特征缺失值的占比，以及该特征值的最大值，最小值和均值。从表中可以看出有48.78%的店铺没有参与Promo2的促销活动。有将近70%的店铺有竞争对手店铺的开业信息，从CompetitionDistance可以看出有99.7%左右的店铺有竞争对手店铺。

表 4 train的特征探索

	缺失值占比	最大值	最小值	均值
Store	0%	--	--	--
DayOfWeek	0%	--	--	--
Date	0%	--	--	--
Sales	0%	41,551.00	46.00	6,955.96
Customers	0%	7388	3	762
Open	0%	--	--	--
Promo	0%	--	--	--
StateHoliday	0%	--	--	--
SchoolHoliday	0%	--	--	--

> 注：(i) --代表不存在数据；(ii) 表4中Customers的最大值、最小值和均值是在店铺开门情况下计算出来的

表4展示了train数据集中每个特征缺失值的占比，以及该特征值的最大值，最小值和均值。从表中可以看出该数据集的各个字段没有缺失值，而train数据集的总行数为1017209，对于1115家店铺从2013-01-01到2015-07-31应该有1050330条数据，因此总体是存在缺失数据的，考虑对于某一店铺在某一天的缺失数据用0来填充。

销售额的描述性统计

本项目要预测的是Sales，由于店铺会有不营业的情况出现，为了体现其销售额真实的统计量，因此先去掉Sales=0的数据，其最小值、最大值、均值、中值和标准差分别为：

表 5 Sales的描述性统计

最大值	最小值	均值	中位数	标准差
41,551.00	46.00	6,955.96	6,369.00	3,103.82

图1给出了Sales频率的频率直方图

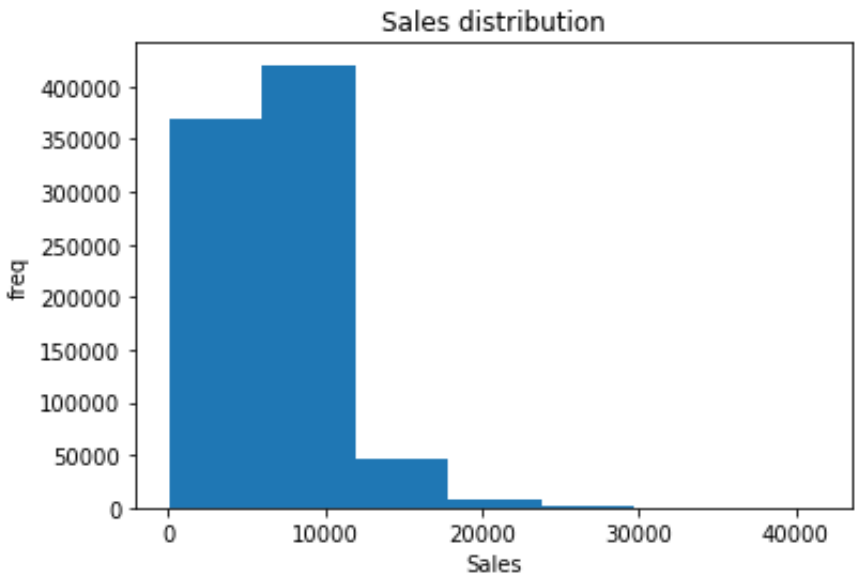

图 1 Sales的频率直方图

从图1可以看出来，Sales是左偏的一个分布，**因此需要对Sales进行log处理，在代码中进行了该处理**。从表5和图1可以发现，大部分店铺的销售额在10000左右。

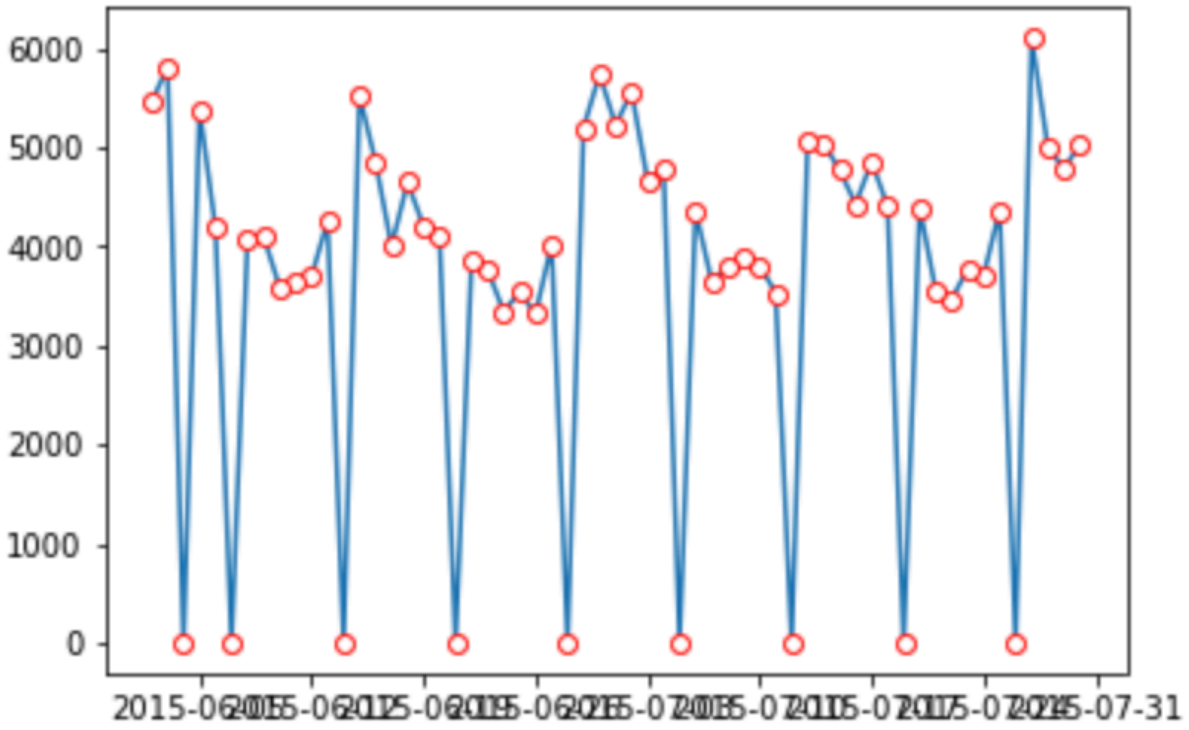

图 2 店铺1从2015-04-30到2015-07-30的销售额的点线图

图2展示了店铺1从2015-04-30到2015-07-30的销售额的点线图，从上图可以看出销售额存在一定的周期性，大约是每两个星期一个周期。

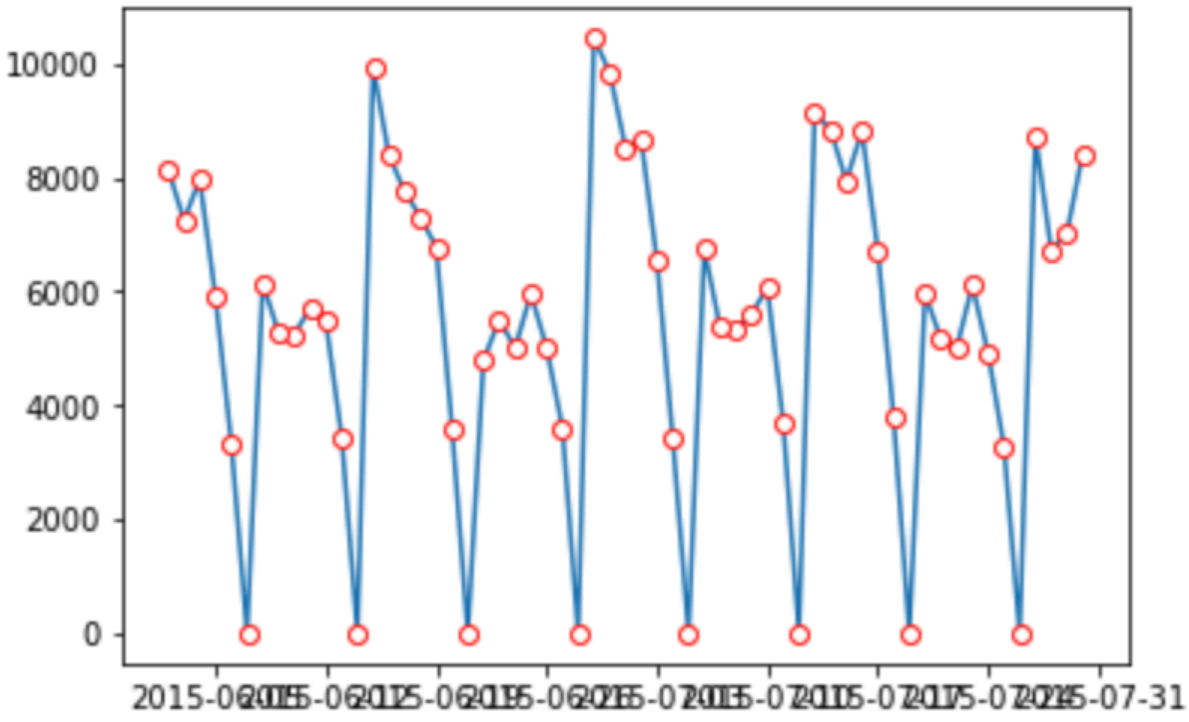

图 3 店铺8从2015-04-30到2015-07-30的销售额的点线图

图3展示了店铺8从2015-04-30到2015-07-30的销售额的点线图，从上图同样可以看出销售额存在一定的周期性，大约是每两个星期一个周期。

> Question：**给出某些特征和sales之间的相关性图是什么意思?**

算法和技术

本项目选择xgboost模型来解决该问题，xgboost是以CART作为基分类器的。CART能够解决回归问题，并且决策树算法能够给出每个特征的重要程度，有利于模型的特征筛选。同时，xgboost模型借鉴了随机森林的思想，支持列抽样，不仅能够防止过拟合，还能减少计算量。本项目的一个难点就在于特征的构造，然而可以利用xgboost的特点，来找出影响销售额的重要变量。

根据前面的分析可知，该问题是一个时间序列的回归问题。用于回归的算法有很多，比如ARIMA模型、支持向量回归、均值回归、神经网络和CART等，这些算法统称为机器学习算法或统计学习算法。本项目使用xgboost模型来解决该问题，xgboost是以CART作为基分类器的。CART能够解决回归问题，并且决策树算法能够给出每个特征的重要程度，有利于模型的特征筛选。而xgboost属于boosting模型，它基本思想是把成百上千个准确率较低的树模型组合起来，成为一个准确率很高的模型。这个模型会不断的迭代，每次迭代就生成一颗新的树。下面就来介绍一下xgboost模型[5]。

监督学习[6]

在讲解xgboost之前要从监督学习介绍，下面从模型，参数和目标函数三个方面来介绍监督学习。

模型是指给定输入x_i如何去预测输出y_i。比较常见的模型如线性模型（包括线性回归和logistic regression）采用了线性叠加的方式进行预测$\hat{y_i} = \Sigma w_j x_{ij}$。对于这里的预测值$y$，根据应用场景的不同，可以给出不同的解释，比如说它既可以是回归目标的输出，也可以通过sigmoid函数的变换得到概率，也可以作为排序的指标等。在模型中，我们需要学习的就是参数，而这个线性模型中，参数是指线性系数w。

目标函数通常的表示方法为$Obj(\theta) = L(\theta) + \Omega(\theta)$，主要由两部分组成，损失函数$L(\theta)$和正则化项$\Omega(\theta)$。在给定输入的情况下，可以通过模型和参数本身来预测，但是没有给定一个比较好的参数来实现比较准确的预测，这个时候就需要目标函数了。目标函数的作用就在于用来优化出模型最优的参数，从而使得模型实现较为准确的预测结果。常见的损失函数有平方损失$L = \Sigma(y_i - \hat{y_i})^2$，logistic损失函数$L = \Sigma(y_i ln(1 + e^{-\hat{y_i}}) + (1 - y_i) ln(1 + e^{\hat{y_i}})$等。而对于线性模型常见的正则化项有$L2$正则和$L1$正则。这样目标函数的设计来自于统计学习里面的一个重要概念叫做Bias-variance tradeoff。比较感性的理解，Bias可以理解为假设我们有无限多数据的时候，可以训练出最好的模型所拿到的误差。而Variance是因为我们只有有限数据，其中随机性带来的误差。目标中误差函数鼓励我们的模型尽量去拟合训练数据，这样相对来说最后的模型会有比较少的 bias。而正则化项则鼓励更加简单的模型。因为当模型简单之后，有限数据拟合出来结果的随机性比较小，不容易过拟合，使得最后模型的预测更加稳定。

优化算法是在给定目标函数后，来解决如何学习出最优参数的方法。常见的优化算法有梯度下降算法、拟牛顿算法插值法等。

Boosted Tree[6][7]

Boosted tree 最基本的组成部分叫做回归树(regression tree)，也叫做CART。CART假设决策树是二叉树，内部节点特征的取值为“是”和“否”，这样的决策树等价于递归的二分每个特征，将输入空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布。CART有特征选择、树的生成及剪纸组成，即可以用于分类也可以用于回归。

一个CART往往过于简单无法有效地预测，因此一个更加强力的模型叫做tree ensemble。对于tree ensemble，可以比较严格的把我们的模型写成是：

$\hat{y_i} = \sum_{k=1}^{K} f_k(x_i), f_k \in \Gamma$

其中每个f是一个在函数空间Γ里面的函数，而F对应了所有CART的集合。设计的目标函数也需要遵循前面的主要原则，包含两部分

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

其中第一部分$\Sigma l(y_i, \hat{y_i})$是训练误差，第二部分$\Sigma\Omega(f_k)$是每棵树的复杂度的和。因为现在模型的参数可以认为是在一个函数空间里面，不能采用传统的如SGD之类的算法来学习该模型，因此采用一种叫做additive training的方式。每一次保留原来的模型不变，加入一个新的函数f到模型中。

$\hat{y_i}^{(0)} = 0$

$\hat{y_i}^{(1)} = f_1(x_i) = \hat{y_i}^{(0)} + f_1(x_i)$

$\hat{y_i}^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y_i}^{(1)} + f_2(x_i)$

$\dots$

$\hat{y_i}^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y_i}^{(t-1)} + f_t(x_i)$

选取一个f来使得我们的目标函数尽量最大地降低。

$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$

更加一般的，会采用如下的泰勒展开近似来定义一个近似的目标函数，方便进行这一步的计算。

目标：$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$

用泰勒展开来近似原来的目标

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

其中，$g_i = \delta_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}, h_i = \delta^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

当把常数项移除之后，会发现如下一个比较统一的目标函数。这一个目标函数有一个非常明显的特点，它只依赖于每个数据点的在误差函数上的一阶导数和二阶导数。

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

其中，$g_i = \delta_{\hat{y}^{t-1}} l(y_i, \hat{y}^{(t-1)}), h_i = \delta^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

接下来定义树的复杂度，复杂度包含了一棵树里面节点的个数，以及每个树叶子节点上面输出分数的$L2$。

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

在这种新的定义下，我们可以把目标函数进行如下改写，其中I被定义为每个叶子上面样本集合
$I_j = \{i | q(x_i) = j\}$

$Obj^{(t)} \simeq \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) = \sum_{i=1}^{n} [g_i w_q(x_i) + \frac{1}{2} h_i w_q^2(x_i)] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2$

$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \lambda T$

这一个目标包含了T个相互独立的单变量二次函数。我们可以定义

$$Obj^{(t)} = \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \lambda T$$

$$= \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \lambda T$$

上述的主要内容是算法设计的推到和技术。xgboost是大规模并行boosted tree的工具，在数据科学方面，有大量kaggle选手选用它进行数据挖掘比赛，其中包括两个以上kaggle比赛的夺冠方案。

GBDT与xgboost[8]

参考wepon的文章中的见解，讨论XGBoost与GBDT的区别：

- XGBoost可以实现以CART作为基分类器的树模型，还能够支持线性回归和分类，也就是logistic回归和线性回归。
- XGBoost对损失函数进行泰勒展开，使用到了一阶导和二阶导信息，而GBDT优化时只用到了一阶导。
- XGBoost增加了与随机森林具有相同思想的列抽样，不仅具有防止过拟合的作用，还能够降低运算的复杂度。
- XGBoost支持并行计算。
- XGBoost的损失函数中增加了正则项，能够有效的降低模型的复杂度，防止模型过拟合。
- XGBoost对缺失值不敏感，可以自动学习其分裂方向。

参数调节[9]

本项目中将会使用Python里sklearn库中的xgboost模型来训练，xgboost方法中涉及的参数众多，这里介绍以下几个重要参数。

- min_child_weight：这个参数默认是 1，是每个叶子里面 h 的和至少是多少，对正负样本不均衡时的 0-1 分类而言，假设 h 在 0.01 附近， min_child_weight 为 1 意味着叶子节点中最少需要包含 100 个样本。这个参数非常影响结果，控制叶子节点中二阶导的和的最小值，该参数值越小，越容易 overfitting。
- eta：shrinkage 参数，用于更新叶子节点权重时，乘以该系数，避免步长过大。参数值越大，越可能无法收敛。把学习率 eta 设置的小一些， 小学习率可以使得后面的学习更加仔细。
- max_delta_step：如果设立了该值，对叶子节点的权重值做了约束在[max_delta_step,max_delta_step]。以防在某些 loss 下权重值过大，默认是 0（其实代表 inf)。可以试试把这个参数设置到 1-10 之间的一个值。这样会防止做太大的更新步子，使得更新更加平缓。
- lambda： 控制模型复杂程度的权重值的 L2 正则项参数， 参数值越大，模型越不容易 overfitting。
- alpha： 控制模型复杂程度的权重值的 L1 正则项参数， 参数值越大，模型越不容易 overfitting。
- gamma： 后剪枝时，用于控制是否后剪枝的参数。
- max_depth：gbdt 每颗树的最大深度，树高越深，越容易过拟合。
- num_round：gbdt 的棵数，棵数越多，训练误差越小，但是棵数过多容易过拟合。需要同时观察训练 loss 和测试 loss，确定最佳的棵数。

对于模型参数设置，本文参考Abhilash Awasthi[10]的脚本来设计。

基准模型

在kaggle上，Gert在本项目上获得了第一名的成绩。以RMSPE为评价指标，其得分为0.10021。Gert[2]使用的就是集成的xgboost 模型，也就是多个采用不同特征的xgboost模型集成。该比赛获得金奖的指标最低得分是0.11037，获得银奖的指标最低得分是0.11552，获得铜奖的最低得分是0.11773。由于能够参考很多前人的研究，特别是Gert给出了自己获得第一名的比赛文档，本项目设定的目标为RMSPE指标得分在0.11552以内。

III. 方法

数据预处理

根据数据集已有的字段，进行预处理，构造特征，并且观察模型的训练效果。本项目主要利用两个数据集，train.csv和store.csv。

针对train.csv和store.csv做如下特征处理：

- CompetitionOpenDate：通过CompetitionOpenSinceYear和CompetitionOpenSinceMonth来得到竞争对手开业的时间CompetitionOpenDate
- Assortment_one_hot：把Assortment，StateHoliday处理成one-hot-vector
- CompetitionOpen判断竞争对手是否开门：Date >= CompetitionOpenDate和Open两个一起判断。也就是说首先判断CompetitionOpenDate是否早于Date，如果是，与该日期本店铺的Open一致，如果不是则为未开门0。
- Promo2Active：该特征是店铺持续促销活动的在当天是否在进行，而Promo2是一些店铺是否参与持续促销活动，这样一来Promo2就可以不用使用了。判断方法是：当前日期是否大于开始促销的日期，当前月份是否在促销的月份里。如果都满足则为1，否则为0。最后外面还要乘以Promo2

参考Gert[2]的工作，考虑一些特征：

- Week:根据Date计算当前日期是这一年的第几周
- Year: 根据Date得到当前日期的年份，年份作为特征是否需要数值化进行处理。
- Day: 根据Date得到当前日期的是哪一天，因为销售额与日期会有很大的关系，该特征可以体现出周期性。
- Month：同day一样，能够从月的角度来体现周期性。

参考Kaggle Discussion[11]中的工作，对数据集做如下特征处理：

- DayofYear：根据Date计算当前日期是这一年的第几天
- 删除'Open'= 1 and 'Sales' = 0的数据，因为该数据是具有很大影响的异常数据。
- Outlier：定义$Outlier = 0.6745 * \frac{diff}{diff_{median}}$，其中$diff = |y - y_{median}|$，$diff_{median}$为$diff$的中位值。（但是没明白原理是什么？）通过可是化可以看处理，的确能够找到偏差较大的值。

构建关于每个Store的特征：

- SalesPerDay：每天平均的销售额。$每天平均的销售额 = \frac{历史总销售额}{历史开业天数}$
- CustomersPerDay：每天平均的顾客数。$CustomersPerDay = \frac{历史总的顾客数}{历史开业天数}$
- SalesPerCustomersPerDay：每天平均每个顾客的消费额。$SalesPerCustomersPerDay = \frac{SalesPerDay}{CustomersPerDay}$

经过上述特征处理后的数据集称为sub_all_data

最终模型训练时使用的特征有：

```
features_x = ['DayOfWeek', 'Open', 'Promo','SchoolHoliday',
'Store','StateHoliday_0','StateHoliday_a', 'StateHoliday_b',
'StateHoliday_c', 'Day', 'Week','Month', 'Year', 'DayOfYear',
'Assortment_onehot', 'StoreType_a', 'StoreType_b', 'StoreType_c',
'StoreType_d', 'SalesPerDay','CustomersPerDay','SalesPerCustomersPerDay']
```

每个特征含义如上所述。

训练和预测

划分数据集

按时间划分

对数据集sub_all_data进行分割，原则是validation data分别保留6周的数据，因此按照时间划分后，从从2013-01-01到2015-06-19的数据作为train，从2015-06-20到2015-07-31的数据作为valid。

将test.csv作为预测集，对该数据集做同样上述的预处理，下面称为test。

随机划分

使用sklearn中的train_test_split来随机划分数据集，其中test_size=0.1。对sub_all_data使用该方法划分train和valid数据集。

这种划分方法训练完成后，可以用模型直接预测test数据集的结果。不需要将train和valid两个数据集合并后再去重新训练。

> 最终划分数据集方法为“随机划分”。

评价指标

本项目的评价指标由kaggle给出， Root Mean Square Percentage Error (RMSPE)，而且有比赛的结果作为对比。因此，选择RMSPE作为评价指标，定义为：

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

其中y_i表示真实的销售额，$\hat{y_i}$表示销售额的预测值。当销售额为0时，不带入计算。该指标的含义是预测值与真实值之间的误差占真实值比例的反映，RMSPE的值越小代表预测越准确。

模型训练

解释变量y_i是左偏的分布，为了减小被解释变量y_i的区间范围，**对y_i进行以e为底的对数(ln)处理**，在评价指标中将其还原后在计算指标值。其中`Sales`作为被解释变量，其余特征为解释变量。该项目使用主要尝试了xgboost模型，使用的参数，参考了Abhilash Awasthi[10]和Kaggle Discussion[11]中的工作。参数设置如下：

```
param = {'bst:max_depth':12,
         'bst:eta':0.0095,
         'subsample':0.8,
         'colsample_bytree':0.7,
         'silent':1,
         'objective':'reg:linear',
         'nthread':6,
         'seed':seed}
num_round = 10000
bst = xgb.train(plst, dtrain, num_round, evallist, feval=rmspe_xg,
verbose_eval=1, early_stopping_rounds=250)
```

`num_round = 10000`设置迭代次数为10000。训练时使用的数据为train，为了输出训练时的评价指标结果，将valid作为`watchlist`。

为了能够使用自定义的评价指标rmspe，构造函数rmspe_xg，特别设置了参数feval=rmspe_xg。

IV. 结果

模型的评价与验证

最终模型将预测结果提交到kaggle上，得到test数据集上的RMSPE得分为：

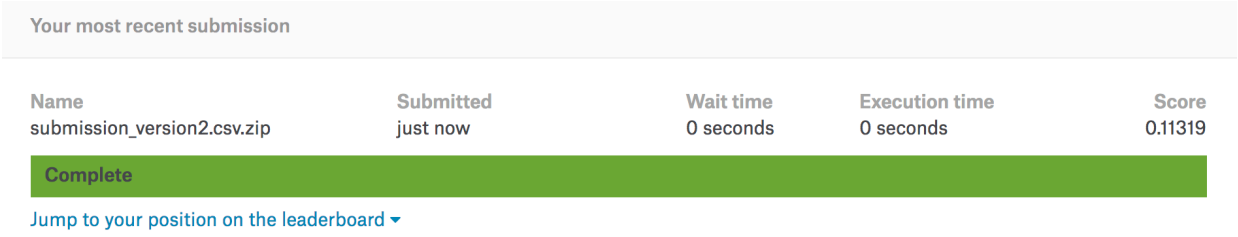

图 4 最终模型kaggle评价结果

从图4结果可以看出，该模型的结果达到了预期效果，并且在kaggle public Leaderboard中排名前100。

中间方案

实验1

第一次实验时，对数据集all_data进行分割，validation data和test data分别保留6周的数据，因此按照时间划分后，从从2013-01-01到2015-05-08的数据作为train，从2015-05-09到2015-06-19的数据作为valid，**从2015-06-20到2015-07-31的数据作为test**。

在训练时由于打印出了模型在valid数据集上的评价指标结果，可以看出train数据集上的拟合效果很好。第999次迭代时，在train数据集上的RMSPE结果为0.0672，在valid数据集上的RMSPE的结果为0.0772。为此，可以在test数据集上验证模型的预测效果。

该模型在test数据集上的预测效果为：0.0828。从预测结果上，可以看出，符合逻辑，在测试集上的RMSPE值要高于train和valid上的RMSPE值。

进一步展示模型预测结果与真实值之间的差距，对店铺1和店铺8进行可视化。

图5展示的是店铺1在2015-06-20至2015-07-31销售额的真实值和预测值，其中红色的圈代表真实值，蓝色的圈代表预测值，从图中可以看出，真实值与预测值之间的误差很小。

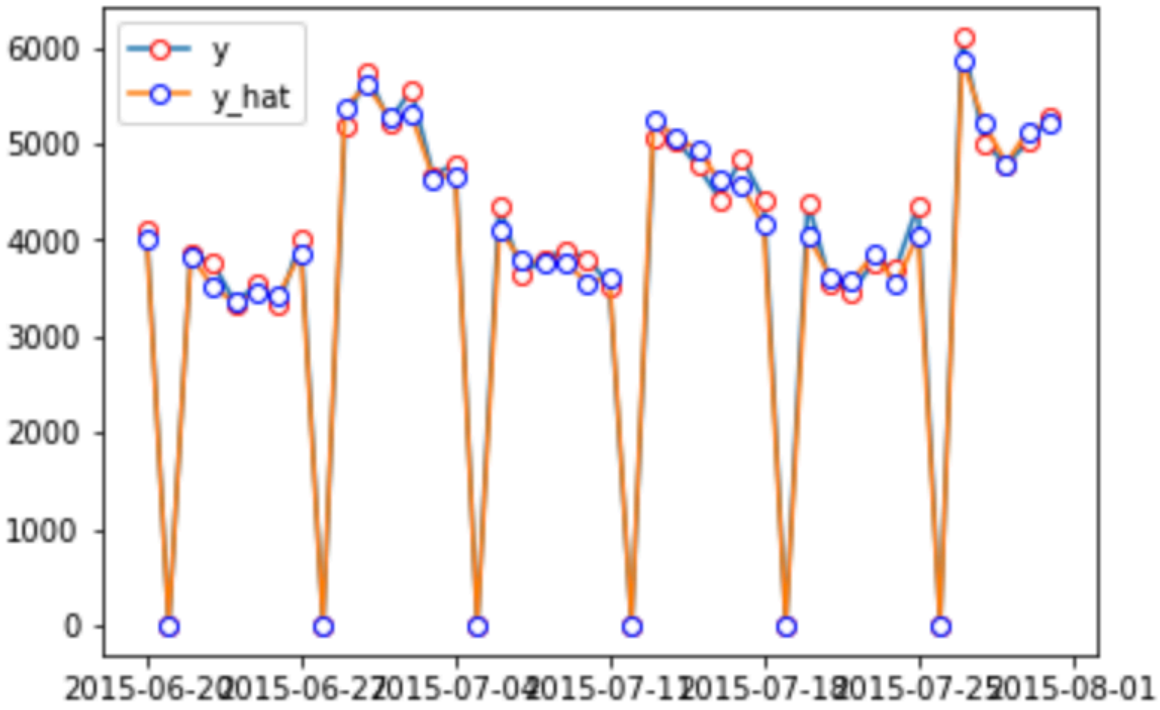

图 5 实验1中店铺1真实销售额和预测销售额的点线图

同样，图6对店铺8的销售额进行了可视化，我们观察店铺8在2015-06-20至2015-07-31销售额的真实值和预测值，其中红色的圈代表真实值，蓝色的圈代表预测值，从图中可以看出，真实值与预测值之间的误差同样很小。

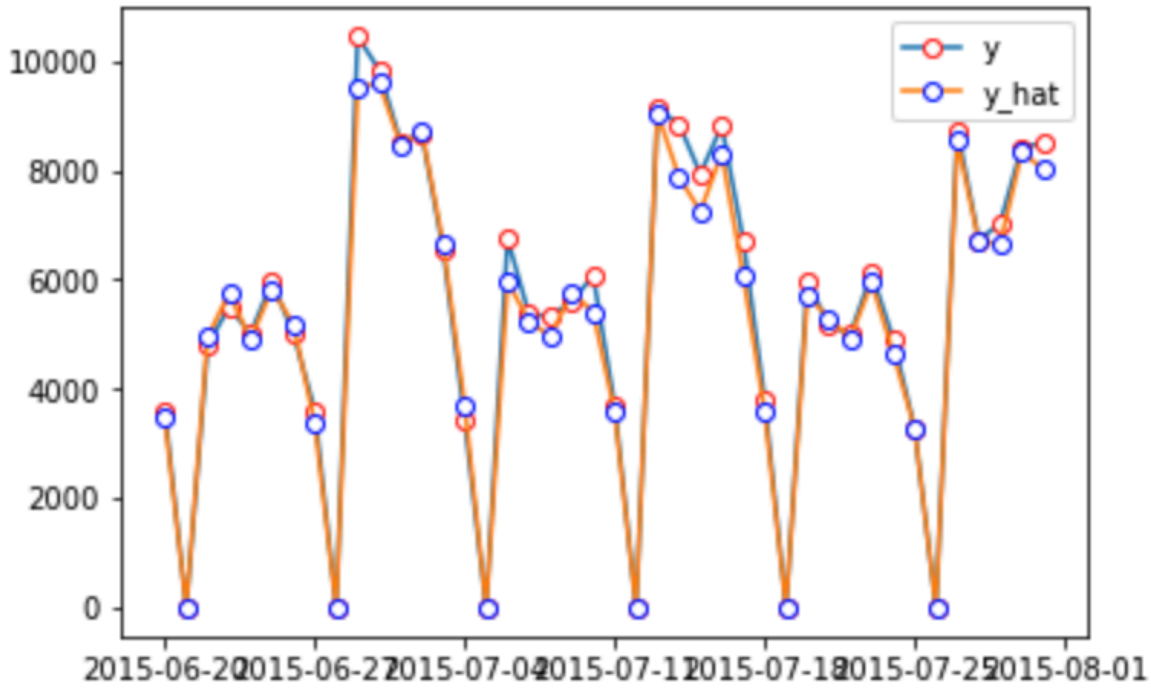

图 6 实验1中店铺8真实销售额和预测销售额的点线图

从上述测试集test上的RMSPE和可视化结果，可以看出，该模型达到了预期的效果，并且超过了预期的基准模型的效果。

> 反思：本次实验虽然得到了很好的结果，但是使用了真实test数据集上没有的特征Customers。实验证明Customers这个特征与Sales有很大的相关性，Customers的预测难度与Sales相差不大。

实验1中的模型在真实的test数据集是预测结果，提交到kaggle上，得到RMSPE得分为：

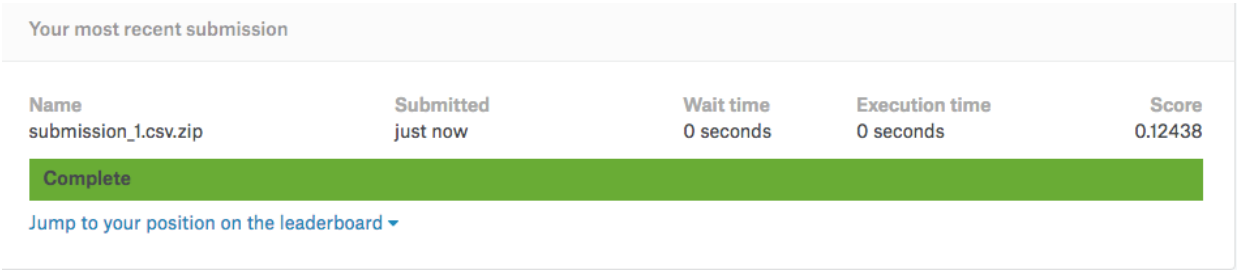

图 7 实验1在kaggle上的评价结果

实验2

通过分析发现Customers与Sales有很强的相关性，考虑将Customers的预测值（Customers_pred）作为特征之一。实验结果表明预测Customer与预测Sales难度相当。图8展示了Customer在valid数据集上的预测效果：

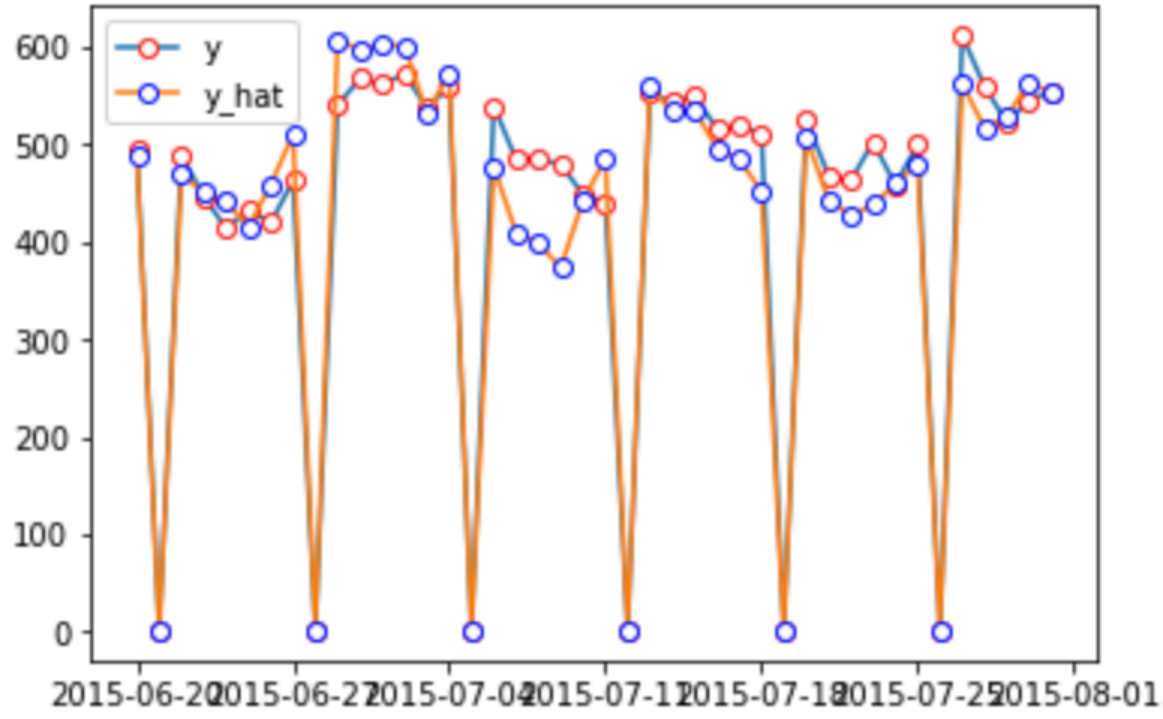

图 8 Customer真实值与预测值的点线图

而Customer中的预测误差会在训练Sales模型时带来更大的误差。下图展示了增加了Customers_pred特征后的训练过程截图：

```
[1879]  train-rmse:0.074198     valid-rmse:0.291015     train-RMSPE:0.086192    valid-RMSPE:0.394625
[1880]  train-rmse:0.07419      valid-rmse:0.291014     train-RMSPE:0.086184    valid-RMSPE:0.394625
[1881]  train-rmse:0.074188     valid-rmse:0.291014     train-RMSPE:0.086178    valid-RMSPE:0.394625
[1882]  train-rmse:0.074177     valid-rmse:0.291014     train-RMSPE:0.086166    valid-RMSPE:0.394625
[1883]  train-rmse:0.07417      valid-rmse:0.291022     train-RMSPE:0.086156    valid-RMSPE:0.394635
[1884]  train-rmse:0.074149     valid-rmse:0.291017     train-RMSPE:0.086128    valid-RMSPE:0.394633
[1885]  train-rmse:0.074148     valid-rmse:0.291017     train-RMSPE:0.086126    valid-RMSPE:0.394633
[1886]  train-rmse:0.074138     valid-rmse:0.291016     train-RMSPE:0.08612     valid-RMSPE:0.394632
[1887]  train-rmse:0.074126     valid-rmse:0.291015     train-RMSPE:0.086106    valid-RMSPE:0.394632
[1888]  train-rmse:0.07412      valid-rmse:0.291014     train-RMSPE:0.086089    valid-RMSPE:0.394631
[1889]  train-rmse:0.074117     valid-rmse:0.291017     train-RMSPE:0.086087    valid-RMSPE:0.394634
[1890]  train-rmse:0.074108     valid-rmse:0.291016     train-RMSPE:0.086082    valid-RMSPE:0.394633
[1891]  train-rmse:0.074104     valid-rmse:0.291017     train-RMSPE:0.086077    valid-RMSPE:0.394633
[1892]  train-rmse:0.074101     valid-rmse:0.291016     train-RMSPE:0.086074    valid-RMSPE:0.394632
[1893]  train-rmse:0.074088     valid-rmse:0.291012     train-RMSPE:0.086058    valid-RMSPE:0.394629
[1894]  train-rmse:0.074086     valid-rmse:0.291011     train-RMSPE:0.086055    valid-RMSPE:0.394628
Stopping. Best iteration:
[1844]  train-rmse:0.074467     valid-rmse:0.291015     train-RMSPE:0.086495    valid-RMSPE:0.394616
```

表明了该特征对模型样本内拟合效果确实得到了很大的提升，但是样本外的结果却更差了。造成这个的原因可能主要是因为拟合时Customers_pred特征被训练的参数与预测时参数不同，因为train中的Customers_pred和valid中的Customers_pred存在一定误差，这个误差会造成预测时更大的误差，因此导致结果偏差更大了。

> 反思：使用其他方法，利用Customers构建其他特征。

实验3

本次实验对数据集all_data进行分割，原则是validation data保留6周的数据，因此按照时间划分后，从2013-01-01到2015-06-19的数据作为train，从2015-06-20到2015-07-31的数据作为valid。

与**实验1**的不同在于，对特征进行了如下处理：增加了特征"DayofYear"；删除'Open'= 1 and 'Sales' = 0的数据；删除了Outlier。含义如**数据预处理**中所述。

实验3中的模型在真实的test数据集是预测结果，提交到kaggle上，得到RMSPE得分为：

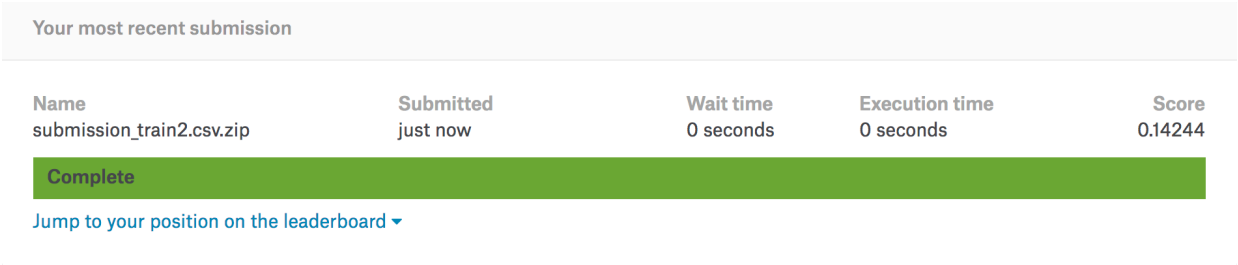

图 9 实验3在kaggle上的评价结果

部分训练过程如下：

```
bst = xgb.train(params=param, dtrain=xg_train, num_boost_round=num_round, evals=watchlist, verbose_eval = 100,
                early_stopping_rounds = 100, maximize = False, feval=rmspe_xg)
```

```
[8200]  train-rmse:0.055419     valid-rmse:0.124137     train-rmspe:0.057274    valid-rmspe:0.118159
[8300]  train-rmse:0.055124     valid-rmse:0.124091     train-rmspe:0.056917    valid-rmspe:0.118116
[8400]  train-rmse:0.054838     valid-rmse:0.124042     train-rmspe:0.056584    valid-rmspe:0.118065
[8500]  train-rmse:0.054563     valid-rmse:0.124021     train-rmspe:0.056267    valid-rmspe:0.118037
[8600]  train-rmse:0.054277     valid-rmse:0.123974     train-rmspe:0.055933    valid-rmspe:0.117996
[8700]  train-rmse:0.054016     valid-rmse:0.123941     train-rmspe:0.055631    valid-rmspe:0.117961
[8800]  train-rmse:0.053739     valid-rmse:0.123918     train-rmspe:0.055315    valid-rmspe:0.11794
[8900]  train-rmse:0.053458     valid-rmse:0.123893     train-rmspe:0.054982    valid-rmspe:0.117912
[9000]  train-rmse:0.053193     valid-rmse:0.123859     train-rmspe:0.054679    valid-rmspe:0.117884
[9100]  train-rmse:0.052937     valid-rmse:0.123848     train-rmspe:0.054391    valid-rmspe:0.117876

[9200]  train-rmse:0.052677     valid-rmse:0.123812     train-rmspe:0.054096    valid-rmspe:0.117843
[9300]  train-rmse:0.052424     valid-rmse:0.123786     train-rmspe:0.053803    valid-rmspe:0.11782
[9400]  train-rmse:0.052157     valid-rmse:0.123775     train-rmspe:0.053503    valid-rmspe:0.117808
[9500]  train-rmse:0.05191      valid-rmse:0.123755     train-rmspe:0.053223    valid-rmspe:0.117788
[9600]  train-rmse:0.051653     valid-rmse:0.123734     train-rmspe:0.052938    valid-rmspe:0.117771
[9700]  train-rmse:0.051422     valid-rmse:0.123713     train-rmspe:0.052682    valid-rmspe:0.117755
[9800]  train-rmse:0.05118      valid-rmse:0.123689     train-rmspe:0.052415    valid-rmspe:0.117732
[9900]  train-rmse:0.050948     valid-rmse:0.123675     train-rmspe:0.052154    valid-rmspe:0.117717
[9999]  train-rmse:0.050722     valid-rmse:0.123672     train-rmspe:0.051904    valid-rmspe:0.117707
```

从上图可以看出，RMSPE的结果不如实验1的效果。从训练过程可以看出来train-rmspe的值很小，说明样本内出现了过拟合，从而导致了在真实的test数据集上的结果较差。**好的训练过程和结果应该是，train-rmspe和valid-rmspe的值都很小，而且差距不大。**

试验4

使用特征

```
['DayOfWeek', 'Open', 'Promo', 'SchoolHoliday', 'Store',
       'StateHoliday_0', 'StateHoliday_a', 'StateHoliday_b',
'StateHoliday_c',
       'Day', 'Week', 'Month', 'Year', 'DayOfYear', 'Assortment_onehot',
       'StoreType_a', 'StoreType_b', 'StoreType_c', 'StoreType_d',
       'SalesPerDay', 'CustomersPerDay', 'SalesPerCustomersPerDay',
       'CompeteOpen', 'Promo2Active']
```

与最终模型方案相比，增加了两个特征'CompeteOpen', 'Promo2Active'，其他均相同。结果如下：

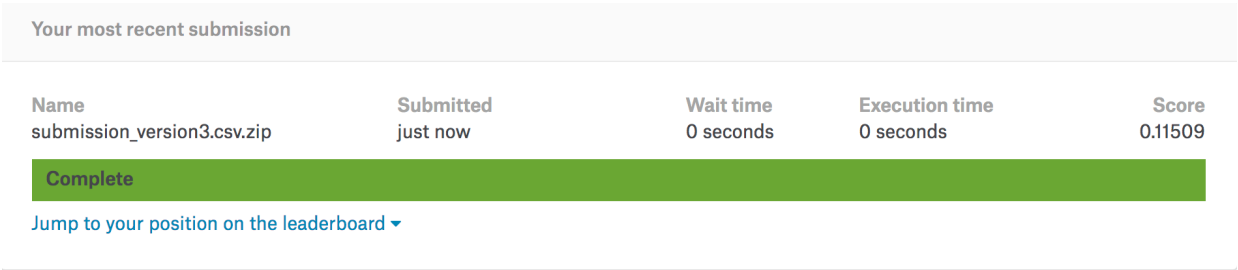

图 10 实验4在kaggle上的评价结果

通过训练过程来看，好像是要优于最终模型的。但是kaggle评价结果却不如最终模型。

V. 项目结论

特征分析

表 6 特征序号及对应特征名称

序号	特征
f0	DayOfWeek
f1	Open
f2	Promo
f3	SchoolHoliday
f4	Store
f5	StateHoliday_0
f6	StateHoliday_a
f7	StateHoliday_b
f8	StateHoliday_c
f9	Day
f10	Week
f11	Month
f12	Year
f13	DayOfYear
f14	Assortment_onehot
f15	StoreType_a
f16	StoreType_b
f17	StoreType_c
f18	StoreType_d
f19	SalesPerDay
f20	CustomersPerDay
f21	SalesPerCustomersPerDay

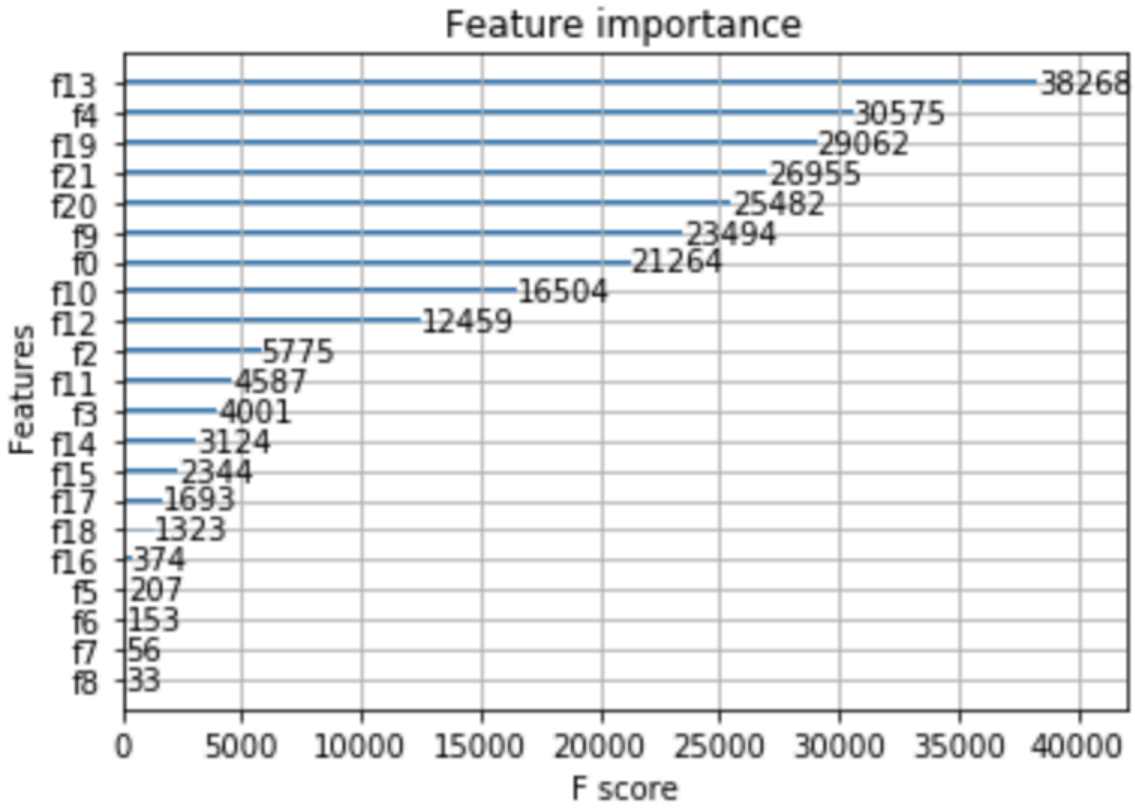

图 11 最终模型中各特征重要程度

从图11中可以看出DayOfYear是最重要的特征，说明销售额与时间有很强的相关性，也就是有时间趋势，同样与实时间相关的特征Day，DayOfWeek，Week，Year都具有很强的相关性。同时DayOfWeek这个特征也体现出销售额具有很强以周为周期的周期性，这一点从销售额的点线图中也可以发现。而销售额以月为周期的周期性要明显弱于以周为周期的周期性，因为Month这个特征重要性要远小于其他时间特征。其次Store特征占据第二，说明了对于每个店铺销售额的具有很大的差异性，构建的特征SalesPerDay，CustomersPerDay，SalesPerCustomersPerDay重要性也很强，同样体现出了这种差异性在预测销售额时的重要性。

总的来看重要特征主要由两部分组成：具有时间特性的特征和具有店铺差异性的特征。进一步提升模型的效果，可以从这两个角度继续努力，构建特征。

对项目的思考

对Rossmann连锁药妆店销售额的预测是一个典型的机器学习项目，项目的流程很清晰，主要分为3个步骤：问题的定义（明确要解决的问题），数据预处理（包括数据清理，特征工程等），算法选型（包括算法选择，模型训练等）。其中大部分时间花费在数据预处理上面，由于参考了Anton Lebedevich[12]的R代码和Abhilash Awasthi[10]的脚本，节省了很多自己探索的时间。在参考和学习的过程中，也有很多的收获。

- 将时间序列数据处理为横截面数据时，如何利用时间信息处理为特征？可以将`Date`时间处理为`week`，`month`，`year`，`day`这些特征来作为时间特征。
- 当特征的数值比较离散时，可以进行对数处理。
- 模型参数的设置对模型的训练效果和过程影响很大，先使用基本的特征选择合适的模型参数范围。对比不同特征训练效果，一定要在相同参数下，不然会造成一定误差。
- 重视训练过程反映的信息，及时暂停训练，有利于节省时间。

- 对Pandas库的使用更加熟悉了。
- 提升了自己对于xgboost模型的理解和运用。
- 做好记录。花一些时间记录好每次试验的数据以及保存的中间过程数据的含义，能省下更多的时间。
- 不要随意删除数据，删除要给出充分的理由和分析。特别是利用其它特征构造其它特征时，删除的数据很可能会给构造的特征带来更大的偏差。处理方法：通过增加列的方式来标记，想要在训时删除的数据。

Reference

1. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016:785-794. ↩

2. Gert's model doucmentation.↩↩↩

3. https://www.kaggle.com/c/rossmann-store-sales/discussion/18024↩

4. http://mabrek.github.io/↩

5. https://cosx.org/2015/03/xgboost↩

6. http://www.52cs.org/?p=429 ↩↩

7. 统计学习方法 ↩

8. http://wepon.me/2016/05/07/XGBoost%E6%B5%85%E5%85%A5%E6%B5%85%E5%87%BA/ ↩

9. xgboost导读与实战 ↩

10. https://www.kaggle.com/abhilashawasthi/xgb-rossmann?scriptVersionId=94227 ↩↩↩

11. https://nbviewer.jupyter.org/github/JohanManders/ROSSMANN-KAGGLE/blob/master/ROSSMANN%20STORE%20SALES%20COMPETITION%20KAGGLE.ipynb#Export-data ↩↩

12. github.com/mabrek/kaggle-rossman-store-sales ↩