机器学习工程师纳米学位毕业项目

预测 Rossman 未来的销售额

Xin Zeng

2020 年 6 月 28 日

1. 定义

1.1 项目概述

Rossmann 在欧洲 7 个国家/地区拥有 3,000 多家药店。目前，Rossmann 商店经理的任务是至少提前六周预测其每日销售额。商店的销售受到许多因素的影响，包括促销，竞争，学校和州假日，季节性和地区性等。

本项目来源于 Rossmann 公司在 Kaggle 上发布的竞赛“Rossmann Store Sale”。我们需要根据 Rossmann 药妆店的信息（比如促销，竞争对手，节假日）以及在过去的销售情况，来预测 Rossmann 未来的销售额。

1.2 问题说明

这个问题是一个有监督的回归问题，需要根据给定的数据，首先运用特征工程的方法选择，提取，合并特征，然后对数据集进行拆分处理，分割成训练集和验证集，最后运用机器学习构建模型，尽可能构建稳定的能够有效预测 Rossmann 商店销售额的模型，在新的数据集上取得很好的效果。

1.3 评价指标

最终提交结果的评价指标是均方根百分比误差（RMSPE），RMSPE 的计算公式为$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i-\hat{y}_i}{y_i}\right)^2}$，其中$y_i$表示一个商店在一天的销售额，$\hat{y}_i$表示与这个对应的预测值，评价时忽略销售额为 0 的点。

RMSPE 是在 RMSE 的基础上，将真实值与预测值之差再除以真实值，形成一个比例。相比于 MSE 和 RMSE，RMSPE 计算的是一个误差率，这样就避免了真实值之间大小的不同而对误差产生的影响。因此在这个项目中，采用 RMSPE 指标来评价模型。

2 分析

2.1 数据研究

项目的数据集一共有四个，分别是 train.csv，test.csv，store.csv 和 sample_submission.csv。

其中 train.csv 是包括销售额的历史数据，是这个项目的训练集。包括 'Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open', 'Promo', 'StateHoliday', 'SchoolHoliday'九个变量，其中 Sales 是需要预测的变量。训练集共有 1017209 条数据，包含了从 2013 年 1 月 1 日到 2015 年 7 月 31 日这段时间的数据。

test.csv 是不包括销售额的历史数据，是这个项目的测试集。包括'Id', 'Store', 'DayOfWeek', 'Date', 'Open', 'Promo', 'StateHoliday', 'SchoolHoliday'八个变量，不包括 Sales 这个需要预测的变量。测试集共有 41088 条数据，包含了从 2015 年 8 月 1 日到 2015 年 9 月 17 日这段时间的数据。

store.csv 是关于商店的补充信息数据。包括'Store', 'StoreType', 'Assortment', 'CompetitionDistance', 'CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear', 'Promo2', 'Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval'十个变量，共有 1115 条数据，通过'Store'这个变量和训练集和测试集的数据相对应。

三个数据集里面包含的特征具体如下：

表 1：Train.csv 特征说明

特征名称	特征含义	特征取值范围
Store	商店 ID	1-1115
DayofWeek	星期几	1-7
Date	日期	2013 年 1 月 1 日至 2015 年 7 月 31 日
Sales	销售额	正整数
Customers	用户数	正整数
Open	是否开张	0 = closed, 1 = open
Promo	当天是否有促销	0=否，1=是
StateHoliday	国家假日，通常，商店在国家假日不营业	a = public holiday, b = Easter holiday, c = Christmas, 0 = None
SchoolHoliday	学校假日	0 = None 1 = 放假

表 2：Test.csv 特征说明

特征名称	特征含义	特征取值范围
Id	记录的序列号	1-41088
Store	商店 ID	1-1115
DayofWeek	星期几	1-7
Date	日期，格式为 YYYY-MM-DD	2013 年 1 月 1 日至 2015 年 7 月 31 日
Open	是否开张	0 = closed, 1 = open
Promo	当天是否有促销	0=否，1=是
StateHoliday	国家假日，通常，商店在国家假日不营业	a = public holiday, b = Easter holiday, c = Christmas, 0 = None
SchoolHoliday	学校假日	0 = None 1 = 放假

表 3：Store.csv 特征说明

特征名称	特征含义	特征取值范围
Store	商店 ID	1-1115
StoreType	商店类型	a, b, c, d
Assortment	c	a = 基础类, b = 补充类, c = 扩展类
CompetitionDistance	最近的竞争对手的距离，单位是米	1115 个商店里有 761 个商店有
CompetitionOpenSinceMonth	竞争对手开张的月份	1-12
CompetitionOpenSince Year	竞争对手开张的年份	年份数据
Promo	当天是否有促销	0=否，1=是
Promo2	是否参与持续的、联系的促销	0=否，1=是
Promo2SinceWeek	开始参加 Promo2 促销的周数	日历上的第几周，1-54，有空的记录
Promo2SinceYear	开始参加 Promo2 促销的年份	年份数据

PromoInterval	每年 Promo2 开始的月份	"Jan,Apr,Jul,Oct", "Feb,May,Aug,Nov" , "Mar,Jun,Sept,Dec" 有空的记录

2.2 探索性数据分析

1. 训练集中各个变量之间的相关系数分析

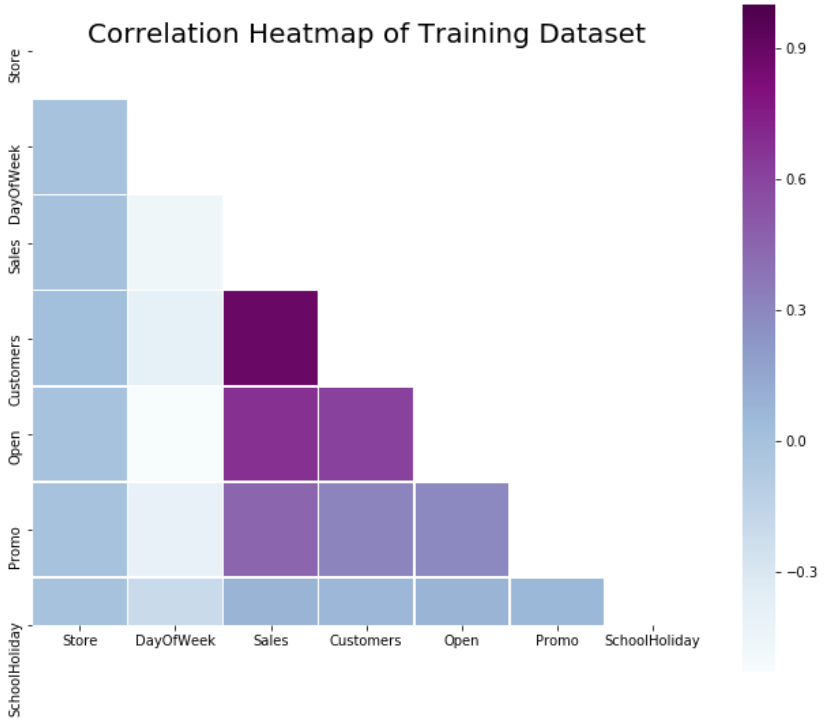

图 1 训练集的相关系数热力图

通过训练集的相关系数热力图，我们可以看出 Sales 变量和 Customers、Open 和 Promo 的相关性最大。Promo、Customers 和 Open 这三个变量也有一定的相关性。

2. 连续变量的分布分析

从三个数据的特征表格中可以看出，只有这三个字段是连续的：Sales，Customers，和 CompetitionDistance。下面对这三个连续变量的分布进行进一步分析。

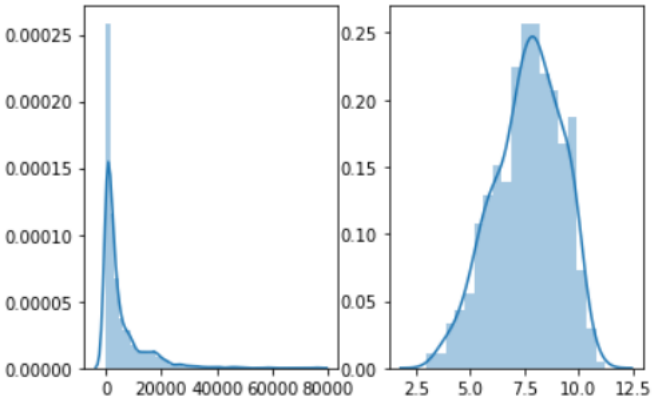

图 2 CompetitionDistance 和 log(CompetitionDistance)的分布

通过 CompetitionDistance 和 log(CompetitionDistance)的分布图我们可以看出，CompetitionDistance 呈明显的右偏分布，log(CompetitionDistance)近似服从正态分布。

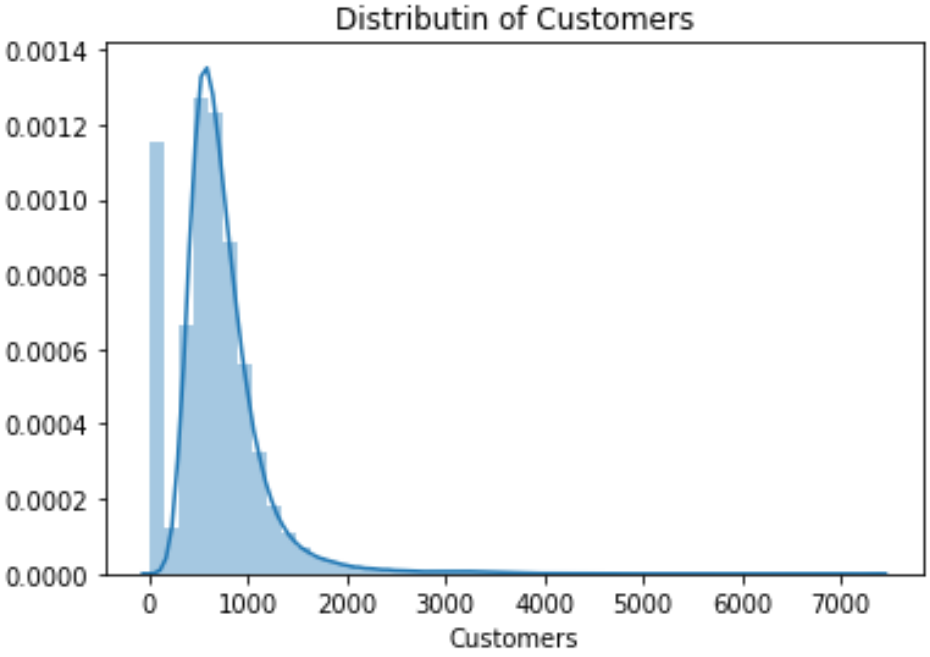

图 3 Customers 的分布

通过 Customers 的分布图我们可以看出，Customers 呈明显的右偏分布。

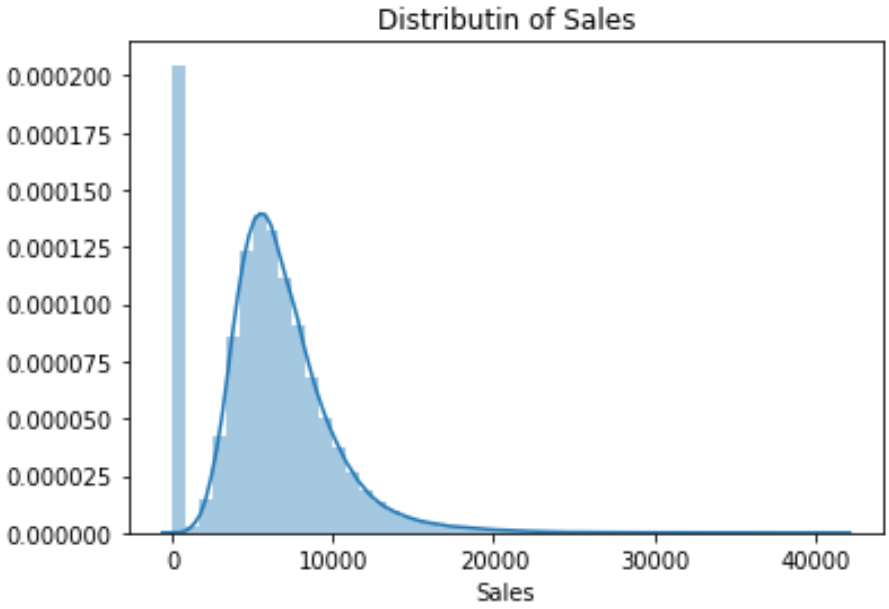

图 4 Sales 的分布

通过 Sales 的分布图我们可以看出，Sales 呈明显的右偏分布。

3. 数据相关关系分析

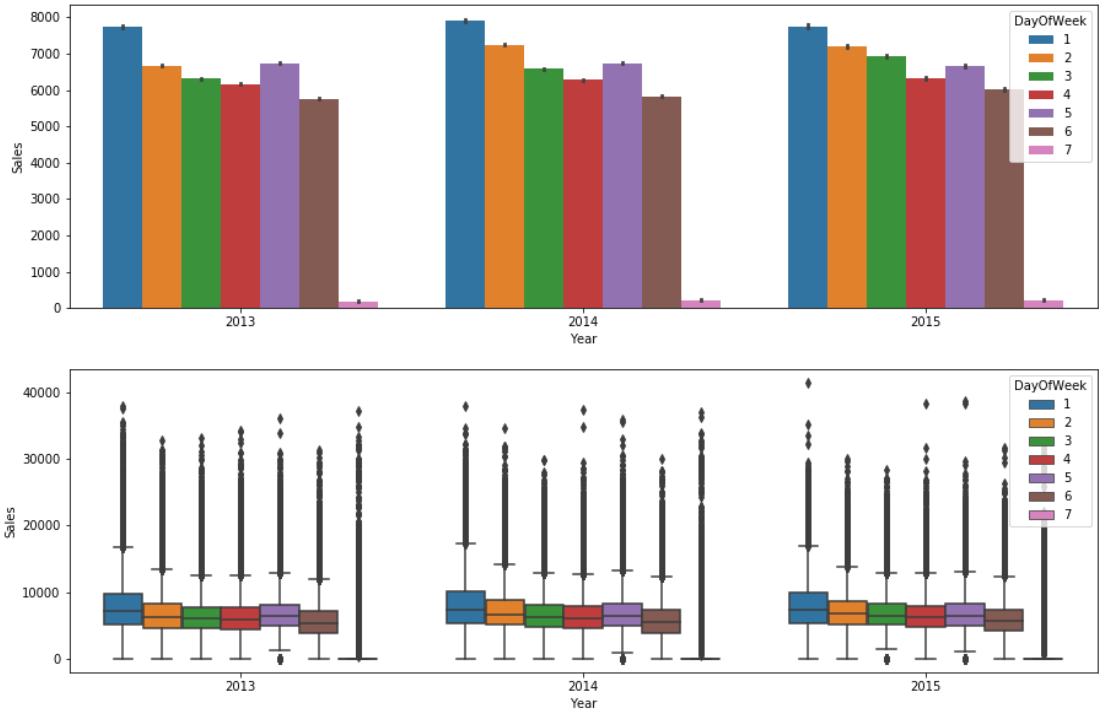

图 5 Sales 和 Year 以及 DayOfWeek 的关系

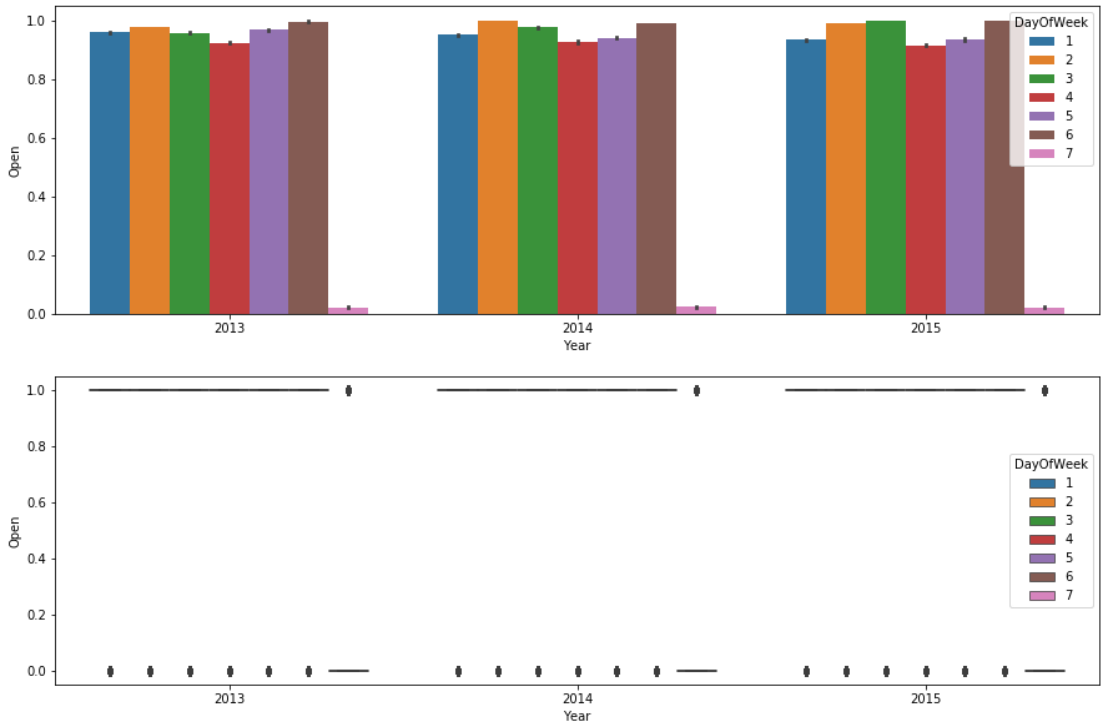

图 6 Sales 和 Year 以及 DayOfWeek 的关系

通过观察图 5 我们可以看出，不同年份，Sales 在不同 DayOfWeek 的分布很稳定，平均来看都是周一的 Sales 最高，周日的 Sales 最低。结合图 6 公司是否开张和 DayOfWeek 的关系，可以得出多数店周日不营业的结论，而这个也导致了周一客户的需求增加，因此周一的 Sales 最高。

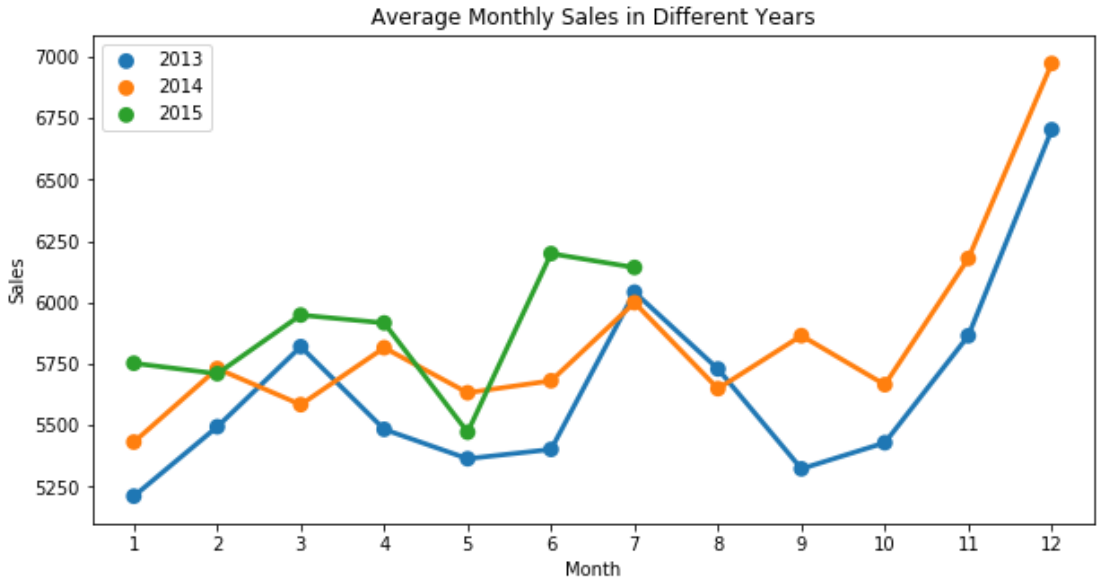

图 7 Sales 和 Month 的关系

通过观察图 7 我们可以看出，不同年份，Sales 在不同 Month 的分布和变化趋

势不完全一样，没有之前相对 DayOfWeek 的情况稳定。2013 年和 2014 年 Sales 都在 12 月达到最大值，在 1 月达到最小值，这可能和 12 月圣诞节等节假日促进消费有关。

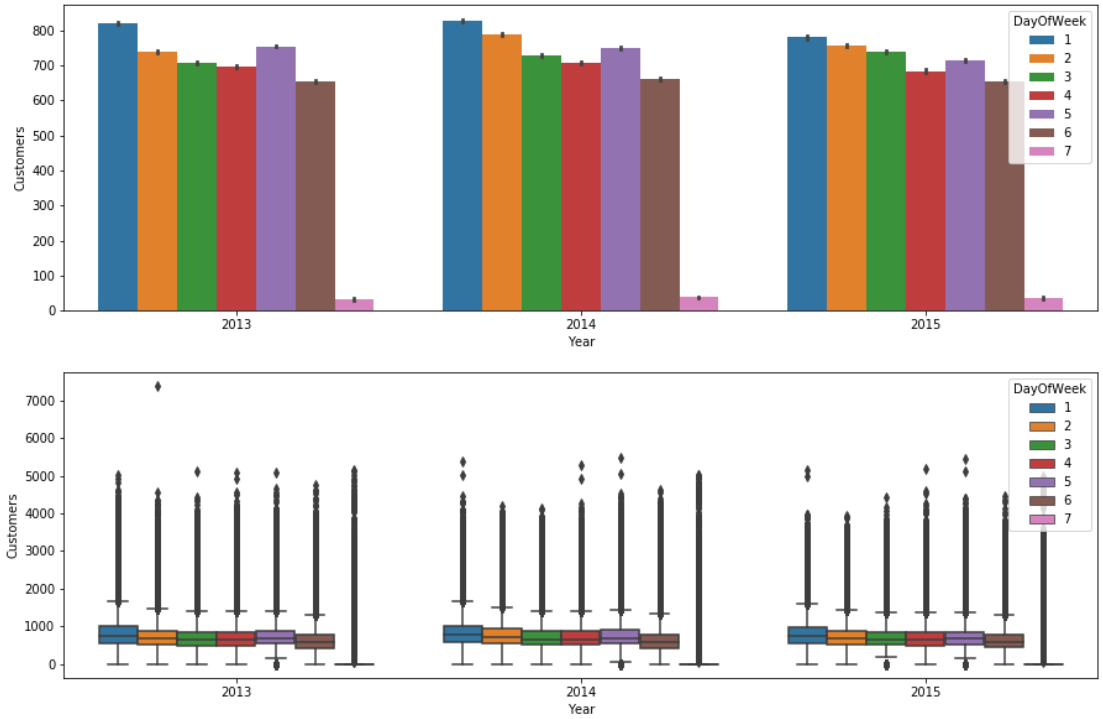

图 8 Customers 和 Year 以及 DayOfWeek 的关系

通过观察图 8 我们可以看出，不同年份，Customers 在不同 DayOfWeek 的分布很稳定，平均来看都是周一的 Customers 最多，周日的 Customers 最少。这个和图 5 和图 6 的结论一致。

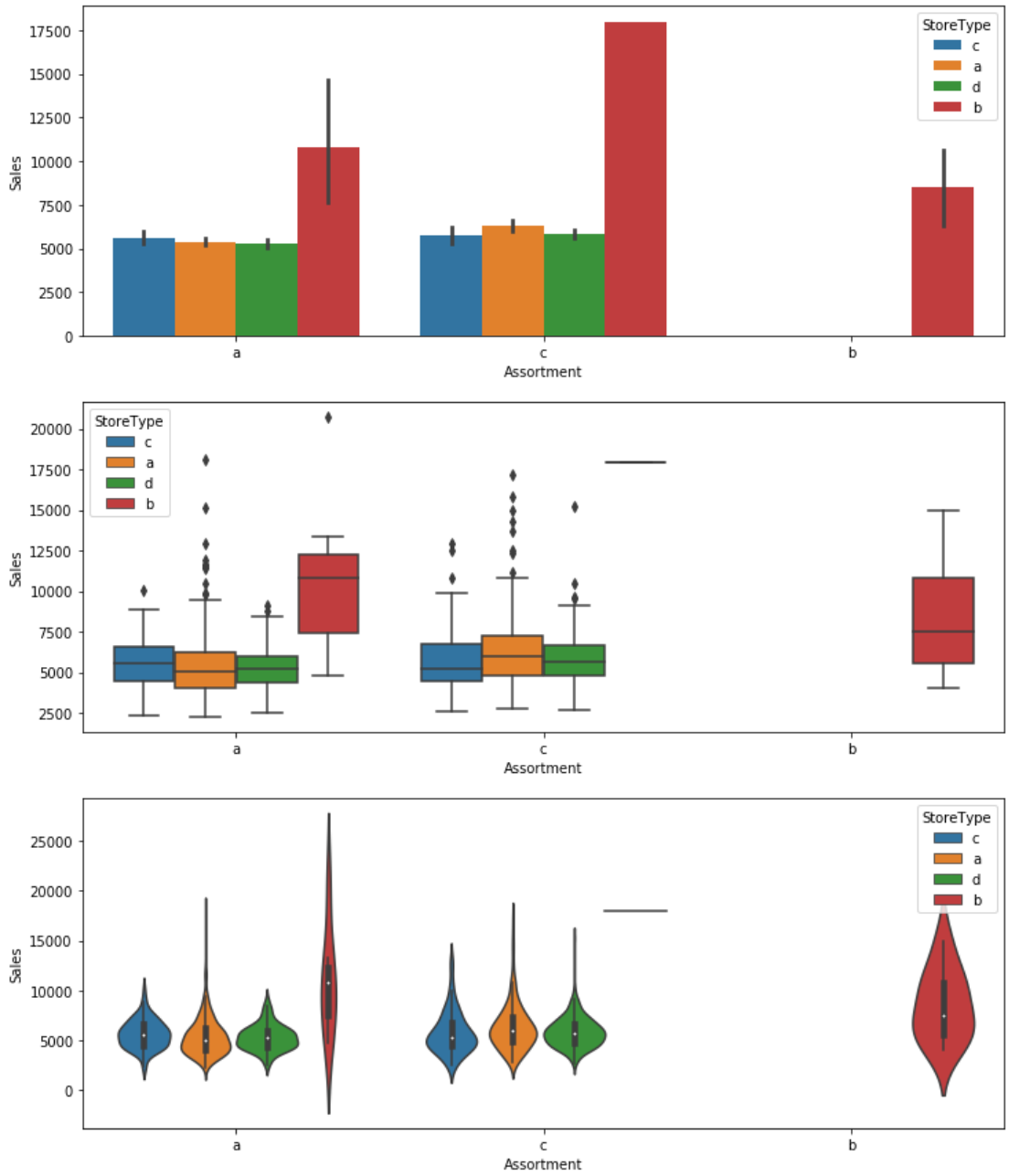

图 9 Sales 和 Assortment 以及 StoreType 的关系

通过观察图 9 我们可以看出，不同 Assortment，Sales 在不同 StoreType 的分布差异很大。Assortment 为 b 类的时候，a，d 和 c 类 StoreType 的 Sales 都为 0，Sales 都集中在 b 类，且呈现右偏分布。Assortment 为 a 类的时候，d 和 c 类 StoreType 的 Sales 分布比较对称，a 类 StoreType 的 Sales 呈现右偏分布，b 类 StoreType 的 Sales 呈现左偏分布。Assortment 为 c 类的时候，d 和 c 类 StoreType 的 Sales 分布比较对称，a 类 StoreType 的 Sales 呈现右偏分布，b 类 StoreType 的

Sales 只有一个值。此外 Assortment 为 a 类和 c 类的时候 a，d 和 c 类 StoreType 的 Sales 的均值非常接近。

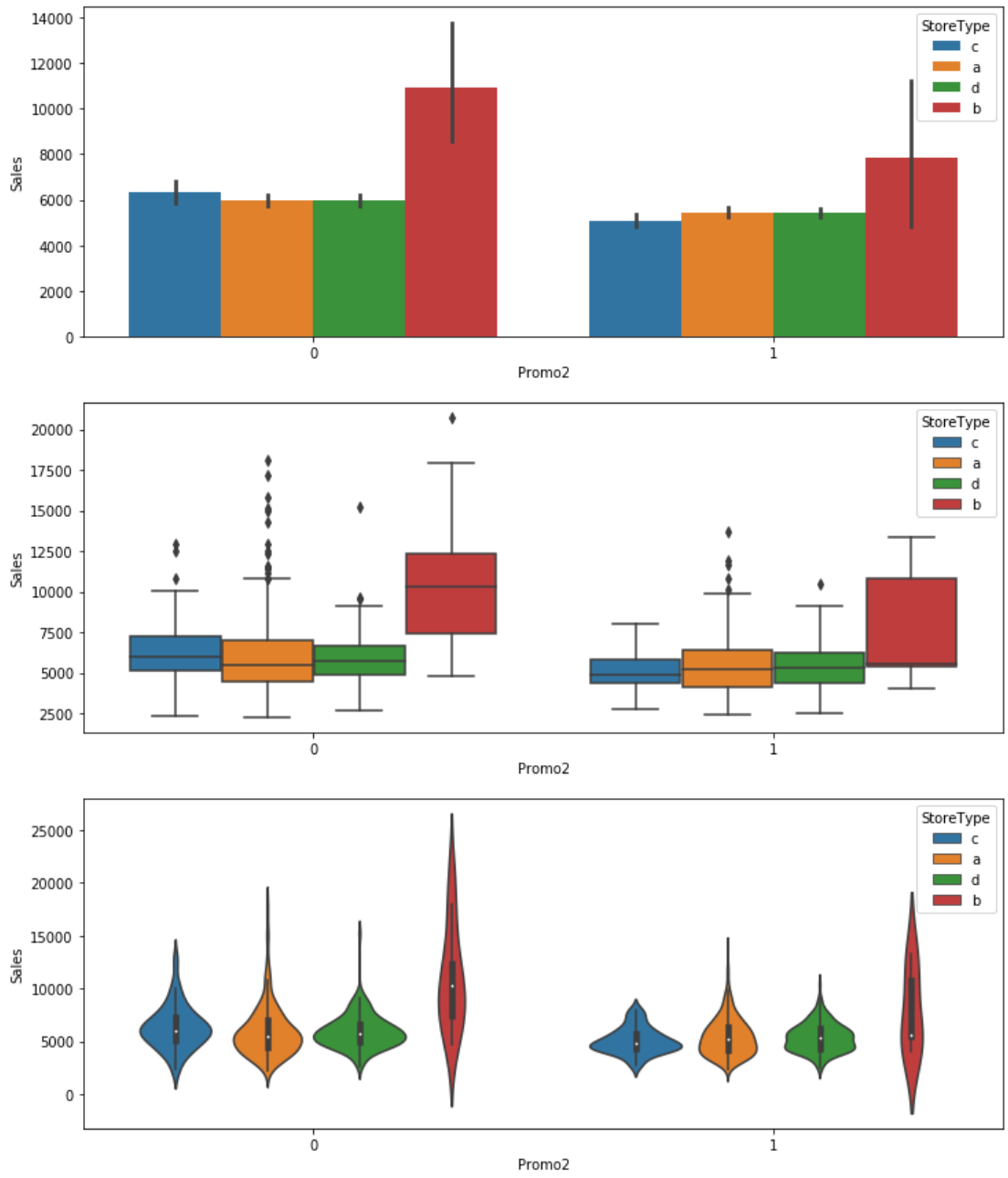

图 10 Sales 和 Promo2 以及 StoreType 的关系

通过观察图 10 我们可以看出，不同 Promo2，Sales 在不同 StoreType 的分布有一定差异，但相比图 9 差异较小。对不同的 Promo2，相同 StoreType 类的 Sales 均值基本一致。对不同的 Promo2，c 类和 b 类 StoreType 的 Sales 呈现对称分布，a 类和 d 类 StoreType 的 Sales 呈现右偏分布。

3 方法

3.1 数据预处理

1. 缺失值处理

表 4 统计了 store.csv 中字段的缺失值状况。其中 CompetitionDistance 特征，有 3 个缺失值，根据特征的含义，采用最大值填充这三个字段。CompetitionOpenSinceMonth 和 CompetitionOpenSinceYear 特征各有 354 个缺失值，根据特征的含义，使用最早的时间来填充，即最小值。Promo2SinceWeek，Promo2SinceYear 和 PromoInterval 特征各有 544 个缺失值，根据特征的含义，用 0 来填充缺失值。此外，将 test.csv 中的缺失值填充为 1，

表 4：Store.csv 缺失数据统计

特征名称	缺失值数量
StoreType	0
Assortment	0
CompetitionDistance	3
CompetitionOpenSinceMonth	354
CompetitionOpenSinceYear	354
Promo2	0
Promo2SinceWeek	544
Promo2SinceYear	544
PromoInterval	544

2. 其他处理

此外，对离散特征使用 Label 编码。对训练集中 CompetitionOpenSinceYear 小于 1990 的数据，将 CompetitionDaysOpen 的值填充为 0。

3. 特征工程

基于 Date 字段，可以获得新增字段：Year、Month、Day、WeekofYear 和 DayofYear。

构造特征 SalesHoliday，将训练集中的 Sales 根据 store 分类，然后根据 StateHoliday 是否为 0 求和，计算出比例。

构造特征 AvgOpen，将训练集中的 Sales 根据 store 分类，然后计算出每个 store 下的平均 open。

构造特征 CompetitionOpen，根据 Year 和 Month 对 CompetitionOpenSinceYear 和 CompetitionOpenSinceMonth 进行转换。

构造特征 PromoOpen，根据 Year 和 WeekofYear 对 Promo2SinceYear 和 Promo2SinceWeek 进行转换。

最终使用的特征有'SalesHoliday', 'AvgOpen', 'Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'SchoolHoliday', 'Year', 'Month', 'Day', 'WeekofYear', 'StoreType', 'Assortment','Promo2', 'CompetitionDistance', "CompetitionOpen", "PromoOpen"。

4. 训练集验证集划分

根据 Store ID 把 train.csv 和 store.csv 合并成一个数据文件，用最后的六周的数据作为验证集，而前面的数据作为训练集。此外，只把 Sales 大于 0 以及 Open 不等于 0 的数据纳入模型进行训练。

5. 模型分析

GBDT 模型是 GBM（Gradient Boosting Machine） 中的一种，采用 CART 树作为基学习器，所以称为 GBDT，为 Boosting 方法中的一员。GBDT 模型是一个加法模型，它串行地训练一组 CART 回归树，最终对所有回归树的预测结果加和，由此得到一个强学习器，每一颗新树都拟合当前损失函数的负梯度方向。其表达式为

$$f_m(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

用来实现 GBDT 模型的主要有 XGBoost 和 LightGBM 两个框架。相比 XGBoost，LightGBM 有很多改进的点：

（1） Histogram 算法：直方图算法的基本思想是先把连续的浮点特征值离散化成 k 个整数（其实又是分桶的思想，而这些桶称为 bin，比如 [0,0.1)→0, [0.1,0.3)→1），同时构造一个宽度为 k 的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了需要的统计量，然后根据直方图

的离散值，遍历寻找最优的分割点。这样做最明显的优点就是内存消耗的降低，直方图算法不仅不需要额外存储预排序的结果，而且可以只保存特征离散化后的值，而这个值一般用 8 位整型存储就足够了，内存消耗可以降低为原来的 1/8。然后在计算上的代价也大幅降低，预排序算法每遍历一个特征值就需要计算一次分裂的增益，而直方图算法只需要计算 k 次。

（2） 在 XGBoost 中，树是按层生长的，称为 Level-wise tree growth，同一层的所有节点都做分裂，最后剪枝。在 Histogram 算法之上，LightGBM 进行进一步的优化，采用的 Leaf-wise 则是一种更为高效的策略，每次从当前所有叶子中，找到分裂增益最大的一个叶子，然后分裂，如此循环。因此同 Level-wise 相比，在分裂次数相同的情况下，Leaf-wise 可以降低更多的误差，得到更好的精度。Leaf-wise 的缺点是可能会长出比较深的决策树，产生过拟合。因此 LightGBM 在 Leaf-wise 之上增加了一个最大深度的限制，在保证高效率的同时防止过拟合

（3） 在树中，一个叶子的直方图可以由它的父亲节点的直方图与它兄弟的直方图做差得到。利用这个方法，LightGBM 可以在构造一个叶子的直方图后，可以用非常微小的代价得到它兄弟叶子的直方图，在速度上可以提升一倍。

考虑到上述改进，采用 LightGBM 框架而不是 XGBoost 框架实现模型。

为了增强模型的泛化能力，防止过拟合，引入早期停止。并且用 RMSPE 作为自定义的模型评价函数。

根据模型在验证集上的表现，多次调整模型的参数，最终将 LightGBM 中的参数设置为 params = {'boosting_type': 'gbdt', 'objective': 'regression', 'metric': 'None', 'learning_rate': 0.1, 'feature_fraction': 0.9, 'bagging_fraction': 0.8, 'bagging_freq': 5, 'max_depth': 100, 'min_sum_hessian_in_leaf': 0.01}。此外 num_boost_round=10000，early_stopping_rounds=1000。

4 结论

1. 模型提交结果

图 11 提交结果的评分

将最终构建的模型预测 test.csv，将得到的预测结果提交到 Kaggle 平台，可以得出在 Private 上分数为 0.11489，在 Public 上分数为 0.11068，符合项目提交要求。

2. 模型结果分析

表 5：模型训练的记录

轮数	训练集 RMSPE	验证集 RMSPE
1	0.535447	0.483031
2000	0.145794	0.135323
4000	0.127289	0.130886
6000	0.109867	0.128412
8000	0.0974907	0.127112
10000	0.0893849	0.126447

表 5 记录了最终采用的参数在训练过程中训练集和验证集 RMSPE 的变化情况。可以看出 RMSPE 都在不断减小，训练集和验证集的变化一致。但是当训练论数逐渐增大的时候，RMSPE 减小的速度在递减，通过调整参数中的最大训练轮数和提前停止的轮数，我们可以在得到很好的模型训练结果的同时防止过拟合。

表 6：模型调参的记录

参数	模型轮次	训练集 RMSPE	验证集 RMSPE	Private Score	Public Score
参数组合 1	9990	0.0892242	0.126448	0.11489	0.11068
参数组合 2	10000	0.17848	0.151855	0.14210	0.13546
参数组合 3	10000	0.0929173	0.133968	0.12584	0.12482
参数组合 4	10534	0.0837219	0.126189	0.11502	0.11231

参数组合 5	10000	0.0898378	0.127604	0.11897	0.11921

其中参数组合 1 为最终结果使用的参数，为 params = {'boosting_type': 'gbdt', 'objective': 'regression', 'metric': 'None', 'learning_rate': 0.1, 'feature_fraction': 0.9, 'bagging_fraction': 0.8, 'bagging_freq': 5, 'max_depth': 100, 'min_sum_hessian_in_leaf': 0.01, num_boost_round=10000，early_stopping_rounds=1000}。参数组合 2 在参数组合 1 的基础上调小了 learning rate，调整成 0.01，参数为 params = {'boosting_type': 'gbdt', 'objective': 'regression', 'metric': 'None', 'learning_rate': 0.01, 'feature_fraction': 0.9, 'bagging_fraction': 0.8, 'bagging_freq': 5, 'max_depth': 100, 'min_sum_hessian_in_leaf': 0.01, num_boost_round=10000，early_stopping_rounds=1000}。参数组合 3 在参数组合 1 的基础上调小了 max depth，调整成 50，参数为 params = {'boosting_type': 'gbdt', 'objective': 'regression', 'metric': 'None', 'learning_rate': 0.1, 'feature_fraction': 0.9, 'bagging_fraction': 0.8, 'bagging_freq': 5, 'max_depth': 50, 'min_sum_hessian_in_leaf': 0.01, num_boost_round=10000，early_stopping_rounds=1000}。参数组合 4 在参数组合 1 的基础上调大了 num_boost_round，调整成 12000，参数为 params = {'boosting_type': 'gbdt', 'objective': 'regression', 'metric': 'None', 'learning_rate': 0.1, 'feature_fraction': 0.9, 'bagging_fraction': 0.8, 'bagging_freq': 5, 'max_depth': 100, 'min_sum_hessian_in_leaf': 0.01, num_boost_round=12000，early_stopping_rounds=1000}。参数组合 5 在参数组合 1 的基础上调整了 feature_fraction 和 bagging_fraction，调整成 feature_fraction=0.8 和 bagging_fraction=0.9，参数为 params = {'boosting_type': 'gbdt', 'objective': 'regression', 'metric': 'None', 'learning_rate': 0.1, 'feature_fraction': 0.8, 'bagging_fraction': 0.9, 'bagging_freq': 5, 'max_depth': 100, 'min_sum_hessian_in_leaf': 0.01, num_boost_round=10000，early_stopping_rounds=1000}。

根据表 6 中模型调参的记录可以得出，当模型参数没有达到最优的时候，调整模型参数能够提升模型表现，当参数达到最优范围的时候，相同特征工程下，再调整参数对模型的影响不大，比如当 num_boost_round 继续增大的时候吗，RMPSE 和提交 Kaggle 平台后得到的 private score 和 public score 的减少不

明显。最终的结果是由特征工程和模型参数共同决定的，特征工程决定了机器学习的上限，通过调整模型参数可以逼近这个上限。

3. 特征重要程度

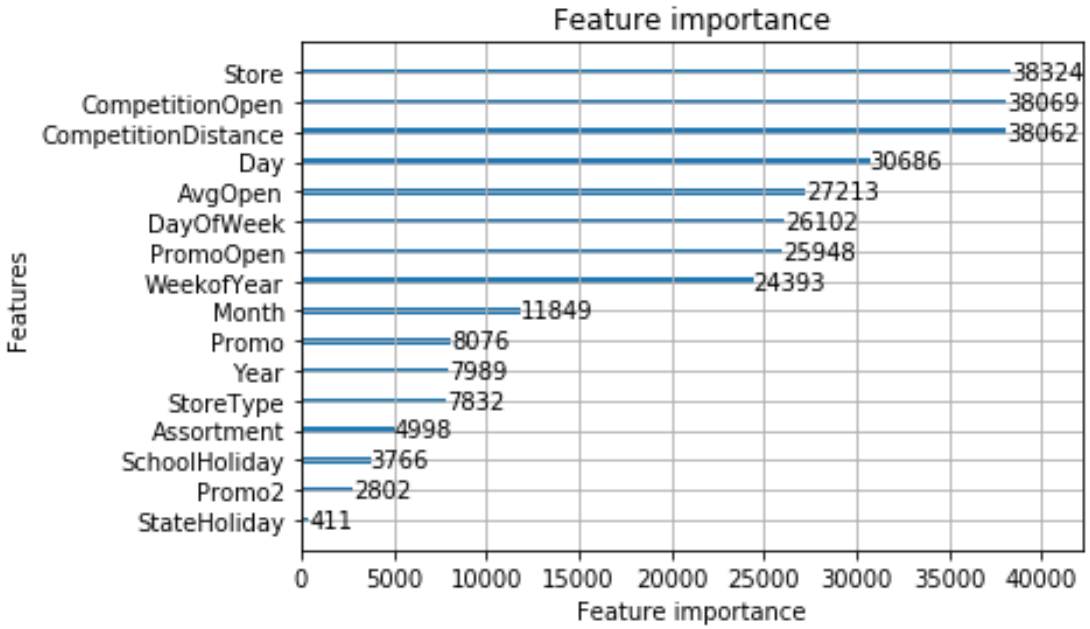

图 12 特征重要程度打分

根据图 12 可以得出，模型中最重要的三个特征为 Store，CompetitionOpen 和 CompetitionDistance。此外，通过特征工程得出的 AvgOpen，DayOfWeek，和 WeekOfYear 也对模型的准确预测起到了作用。这可以看出特征工程对实际业务是有一定作用的，更好的特征意味着更强的灵活度，更好的特征意味着只需要用简单的模型，更好的特征意味着更好的结果。

特征重要程度的结果也对构造特征提供了思路，既可以从包含很多维度信息的特征中提取新的特征，比如从原始的 Date 特征中提取出 DayOfWeek 和 WeekOfYear，也可以通过组合不同特征来得到新的特征，比如通过组合多个特征我们得到 AvgOpen 和 PromoOpen。

4. 后续改进

可以尝试使用 XGBoost 和 CatBoost 框架训练模型，并比较 XGBoost、CatBoost 框架和 LightGBM 的结果、速度和内存占用情况。

可以尝试用 LightGBM 和 XGBoost 的 GPU 模式提升训练速度。

可以进一步进行模型参数调优。

可以尝试时间序列模型，以及复杂的神经网络模型。

现有数据中大部分都与时间有关，与空间相关的数据很少。可以尝试使用外部获取的天气数据和超市所在地的数据来提高预测的精确度。

5. 改进尝试及结果

我尝试使用 XGBoost 框架训练模型并且和 LightGBM 框架进行比较。表 7 比较了 XGBoost 框架和 LightGBM 框架训练结果。此外，通过观察模型不同轮数的 RMSPE，我观察到 XGBoost 框架更容易出现过拟合的问题，当训练集 RMSPE 减小的时候，验证集 RMSPE 在增加，这样能够将训练集 RMSPE 降低到更小，但是验证集 RMSPE 更大，最终导致 Private Score 和。在模型运行过程中观察到 LightGBM 框架训练速度更快，内存占用更小。

表 7：XGBoost 框架和 LightGBM 框架训练结果比较

框架	训练集 RMSPE	验证集 RMSPE	Private Score	Public Score
LightGBM	0.0892242	0.126448	0.11489	0.11068
XGBoost	0.038704	0.141714	0.13578	0.13987

此外，我还尝试使用 store_states.csv 这个提供了超市所在地的外部数据和 store.csv 相结合，作为一个新的 feature 加入现有的模型，虽然样本内的 RMSPE 有所上升，但是样本外预测的精度略微上升，这可能是因为只引入了一个 feature 的原因，如果有更多的外部数据能够构造更多有效的 feature，对结果的提升应该会更多。

表 8：原模型和引入新的 feature 的训练结果比较

框架	训练轮数	训练集 RMSPE	验证集 RMSPE	Private Score	Public Score
LightGBM	9990	0.0892242	0.126448	0.11489	0.11068
XGBoost	7203	0.0936124	0.133151	0.11428	0.11020

参考文献

[1] Jason Brownlee. Supervised and unsupervised machine learning algorithms. *Machine Learning Mastery*, 2016.

[2] https://www.kaggle.com/c/rossmann-store-sales

[3] http://blog.kaggle.com/2015/12/21/rossmann-store-sales-winners-interview-1st-place-gert

[4] https://zhuanlan.zhihu.com/p/85044159

[5] Sanjay Ranka and V Singh. Clouds: A decision tree classifier for large datasets. *In Proceedings of the 4th Knowledge Discovery and Data Mining Conference*, pages 2–8, 1998.

[6] 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*, pages 785–794. ACM, 2016.

[7] Alin Dobra and Johannes Gehrke. Secret: a scalable linear regression tree algorithm. *In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–487. ACM, 2002.