

I. 问题的定义

项目概述

Rossmann 劳诗曼是德国最大的日化用品超市，成立于 1972 年。对于像 Rossmann 这样的大型连锁超市而言，能够准确地预测未来某段时间的销售额是公司发展的助力剂，一方面帮助决策者衡量其产品的市场需求量，另一方面帮助公司制定合理的财务及市场战略计划[1]。所以，该项目的主要目的是帮助 Rossmann 预测未来的销售额。

用于预测的数据集包含了 2013 年 1 月 1 日到 2015 年 7 月 31 日期间 1115 家 Rossmann 连锁商店的销售数据，共涉及 1017209 个样本和 18 个特征[2]。该项目旨在通过分析影响各连锁店日销售额的相关因素，达到预测未来六周（2015 年 8 月 1 日到 2015 年 9 月 17 日）各连锁店的日销售额的目的。

一般的时间序列预测问题运用传统模型比如 Exponential Smoothing 和 ARIMA 很简单，但这些模型一次只能对一组时间序列做预测，比如预测某家连锁店下某一天的销售额。而我们现在面对的时间序列预测任务是预测 1115 家店未来六周的日销售额，所以可以尝试采用机器学习算法一次性获得所有预测结果[3]。

问题陈述

为了到达预测的最终目的，我们首先需要充分的了解项目给定的数据集特点，包括样本数量、特征含义，各特征数据类型及缺失情况，有无异常值等。

其次，需要对数据集进行探索性分析，例如通过直方图了解销售额等连续数值变量的分布情况，通过箱型图了解店铺类型等分类变量的类别分布情况等。最重要的是通过单变量或多变量分析单个或多个特征对连锁店销售额的影响，以及销售额或促销等因素随时间变化的周期性和趋势。

第三，在前两步的基础上进行特征工程。对于时间序列数据集，不能简单的像截面时间数据集那样用给定的训练集训练模型，然后预测测试集标签。这里需要将训练集（历史特征）和测试集（未来特征）合起来构成总预测特征集合。如图 Fig 1 所示[4]。由于原始特征有限，只应用原始特征建模可能很难到达项目要求的指标标准，可以通过两种方法获取新特征。方法一是在 Kaggle 或 Github 等网站寻找补充数据集作为新的特征，例如各连锁店所在州等数据[7]。方法二是依据原始特征挖掘新的特征，而挖掘的依据则是前面的探索性分析结果。挖掘的思路有如下几种：对于时间序列预测问题可以根据销售额变化特点提取每个连锁店销售额的 Lag 特征[5]，即用滞后销售额预测当前销售额，也可以创建新的时间标签，例如分解观察日期的年月日等，还可以根据对销量影响明显的特征去衍生新的特征，例如根据最近竞争店铺开业时间提取出它距观察时间的营业月份数，根据销售额在周末和国家法定假日前后的变化趋势提取观察时

间距这些放假日的天数等。将获取的全部特征预处理，主要是将类别变量向量化。接着，我们还需从项目给定的训练集中分离出训练集与验证集，通过训练集训练模型，通过验证集检验模型性能。因为预测的是未来 6 周的销售额，所以不妨将验证集设为同等时间长度，即将 2013-01-01 至 2015-06-13 时间段的数据作为训练集，将 2015-06-14 至 2015-07-31 时间段的数据作为验证集。训练集通过基准模型‘随机森林回归模型’进行训练，根据特征重要性筛选对结果影响权重较大的特征，并去除相关性较高的特征之后，将剩余特征作为最终特征集。

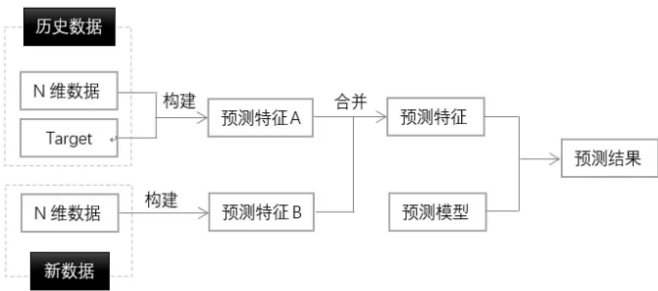


Fig 1. 总预测特征集的获取过程

第四，将最终特征集输入 XGBoost 模型进行训练，并进一步优化模型参数。参数调优的标准是使验证集预测销售额与真实销售额的 rmspe 值最小。最后，利用调优后的模型对测试集进行预测，将结果按照指定格式上传 Kaggle，得到测试集预测值与实际值的 rmspe 值。

评价指标

该项目中用于评价模型和结果的指标采用 rmspe，均方根百分比误差是用来衡量观测值同真值之间的偏差，计算公式如下[6]：

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

其中 y_i 代表各商店的实际销售额，yhat_i 代表模型的预测销售额。Rmspe 越小表示模型预测的越精准。

II. 分析

数据的探索

本项目数据集共包含 train.csv（训练集）、test.csv（测试集）、store.csv（各连锁店信息）和 sample_submission.csv（关于正确提交文件格式的示例数据集）四个数据集。其中 train.csv 包含了 2013 年 1 月 1 日至 2015 年 7 月 31 日期间 1115 个连锁店的历史销售数据，test.csv 包含了 2015 年 8 月 1 日至 2015 年 9 月 17 日期间 1115 个连锁店的销售数据（营业额 'Sales' 和客流量 'Customers' 除外），store.csv 包含了 1115 家连锁店店铺信息。销售数据包括店铺编号（Store）、营业时间（DayOfWeek 表示周几，date 表示具体的日期）、给定日期的营业额（Sales）、给定日期的客户数（Customers）、当日店铺是否开业（Open）、当日店铺是否促销（Promo）、给定日期是否是国家假日（StateHoliday: a 表公共节日；b 表示复活节；c 表示圣诞节；d 表示非节日）、当日店铺是否受学校停课的影响（SchoolHoliday）。连锁店信息包括店铺类型（StoreType）、店铺等级（Assortment）、最近竞争店铺的距离（CompetitionDistance）、最近竞争店铺开业时间（CompetitionOpenSince[Month/Year]）、店铺是否参与连续促销（Promo2）、参与连续促销的年或周（Promo2Since[Year/Week]）、连续促销的开始时间间隔（PromoInterval）。除此以外，为了丰富建模特征，增加了额外数据集 state.csv，该数据集是关于项目中 1115 个连锁店所在州的信息[7]。下面分别展示了建模用到的四个数据集缺失情况及其数据类型：

```
df_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 9 columns):
Store                1017209 non-null int64
DayOfWeek            1017209 non-null int64
Date                 1017209 non-null datetime64[ns]
Sales                1017209 non-null int64
Customers            1017209 non-null int64
Open                 1017209 non-null int64
Promo                1017209 non-null int64
StateHoliday         1017209 non-null object
SchoolHoliday        1017209 non-null int64
dtypes: datetime64[ns](1), int64(7), object(1)
memory usage: 69.8+ MB
```

上图展示了 train.csv 数据集基本信息，共包含 1017209 条样本，各字段均没有缺失。需要说明的是 'Date' 在原始数据集中是字符串数据类型，导入时直接转换为 datetime 数据格式。分类变量 'StateHoliday' 包含四个分类 'a' , 'b' , 'c' , '0'，但存在部分 '0' 显示为数值 0 的情况，需要统一转换为 '0'。

```
df_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41088 entries, 0 to 41087
Data columns (total 8 columns):
Id                    41088 non-null int64
Store                 41088 non-null int64
DayOfWeek             41088 non-null int64
Date                  41088 non-null datetime64[ns]
Open                  41077 non-null float64
Promo                 41088 non-null int64
StateHoliday          41088 non-null object
SchoolHoliday         41088 non-null int64
dtypes: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 2.5+ MB
```

上图展示了 test.csv 数据集基本信息，共包含 41088 条样本，仅 'Open' 字段有缺失，计划用 1 填充，即假设待预测连锁店均为开业状态，同时将该字段由 'float' 转换为 'int' 类型。考虑到前面提到时间序列训练集与测试集合并为特征集，计划添加目标字段 'Sales'，默认待预测销售额初值为 0。

```
df_store.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
Store                1115 non-null int64
StoreType            1115 non-null object
Assortment            1115 non-null object
CompetitionDistance  1112 non-null float64
CompetitionOpenSinceMonth  761 non-null float64
CompetitionOpenSinceYear  761 non-null float64
Promo2              1115 non-null int64
Promo2SinceWeek      571 non-null float64
Promo2SinceYear      571 non-null float64
PromoInterval        571 non-null object
dtypes: float64(5), int64(2), object(3)
memory usage: 87.2+ KB
```

上图展示了 store.csv 数据集基本信息，共包含 1115 条样本。其中，'PromoInterval' 缺失值用 '0' 填充。'CompetitionDistance'、'CompetitionOpenSinceYear'、'CompetitionOpenSinceMonth' 缺失值用相应字段中位数填充。因为只有参与长期促销活动的连锁店才有关于 'Promo2' 的字段，所以 'Promo2SinceYear' 和 'Promo2SinceWeek' 缺失值用 0 值填充更合理。同时为了分析方便，将后四个字段的 datatype 由 'float' 转换为 'int'。

```
df_state.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 2 columns):
Store    1115 non-null int64
State    1115 non-null object
dtypes: int64(1), object(1)
memory usage: 17.5+ KB
```

州	连锁店数量
NW	286
BY	180
SH	115
HE	112
BE	92
SN	75
BW	73
ST	56
RP	40
TH	36
HH	28
HB, NI	22

上图左展示了 state.csv 数据集基本信息及，共包含 1115 条样本，无缺失。上图右展示了各州连锁店数量分布。

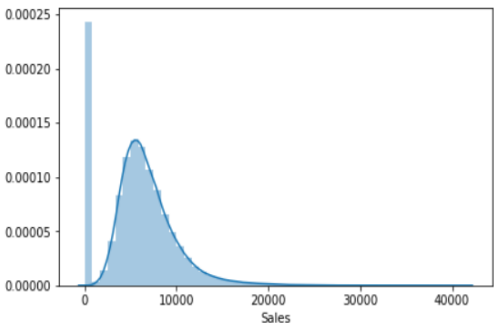


Fig 2. Sales 直方图

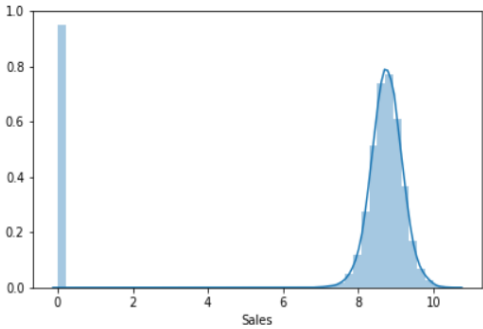


Fig 3. Sales 对数转换后直方图

Fig 2 是目标标签 ‘Sales’ 的直方图分布，Fig 3 是该特征做了对数转换后的分布图。‘Sales’ 不做任何形式转换时的直方图存在 ‘右拖尾’ 现象，考虑到一些较大的离群值会影响数据平稳性，所以应将该变量做 log 转换，取对数之后不会改变数据的性质和相关关系，但压缩了变量的尺度，使数据更加平稳，也削弱了模型的共线性、异方差性等[8]。‘Sales’ 做 log 转换后的分布更接近正态分布且数据更平稳。上述两个直方图均存在销售额为 0 的情况，后期建模时可以只考虑训练集中销售额大于 0 的样本。

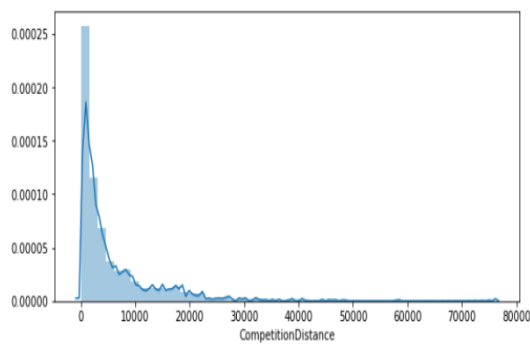


Fig 4. CompetitionDistance 直方图

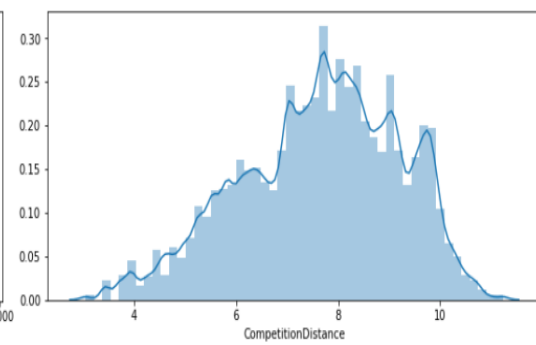


Fig 5. CompetitionDistance 对数转换后直方图

Fig 4 是最近竞争店铺距离的直方图分布，Fig 5 是该特征做了对数转换后的分布图。该特征不做任何形式转换时的直方图同 ‘Sales’ 一样，较大离群值使得分布 ‘右拖尾’，经过对数转换后数据分布更均匀和平稳，减少了离群值的干扰，所以也将该特征进行对数转换后入模。

探索性可视化

为了找到与研究问题相关的特征，从以下方面进行了探索性分析：

1. 单变量分析，主要研究单个原始特征对目标标签销售额的影响：

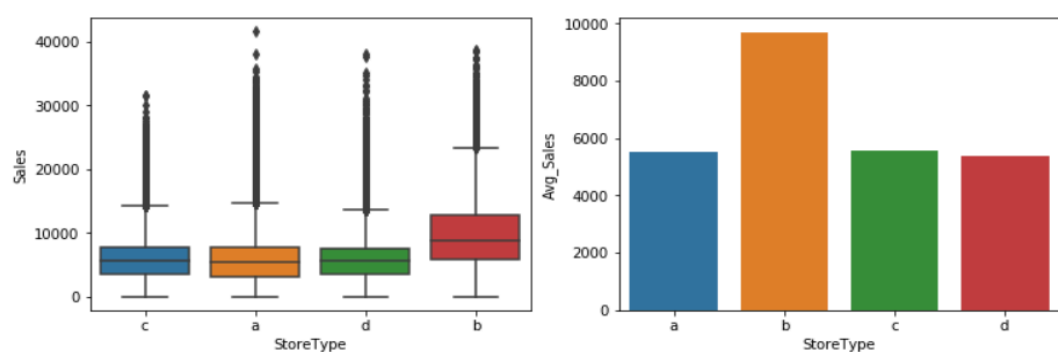


Fig 6. 连锁店类型 (StoreType) 对销售额的影响

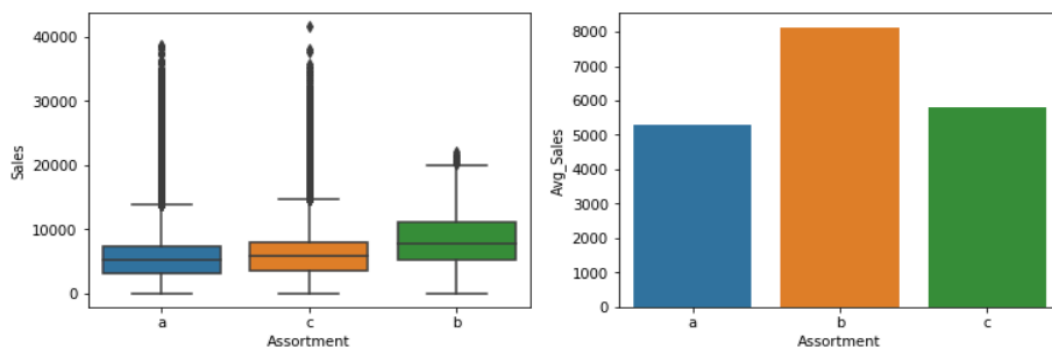


Fig 7. 连锁店分类等级 (Assortment) 对销售额的影响

由图 Fig 6 和 Fig 7 可见，连锁店类型与分类等级会影响销售额，其中 b 类型或 b 等级店铺具有相对较高的销售额，其它类型或等级的店铺销售额相差无几。

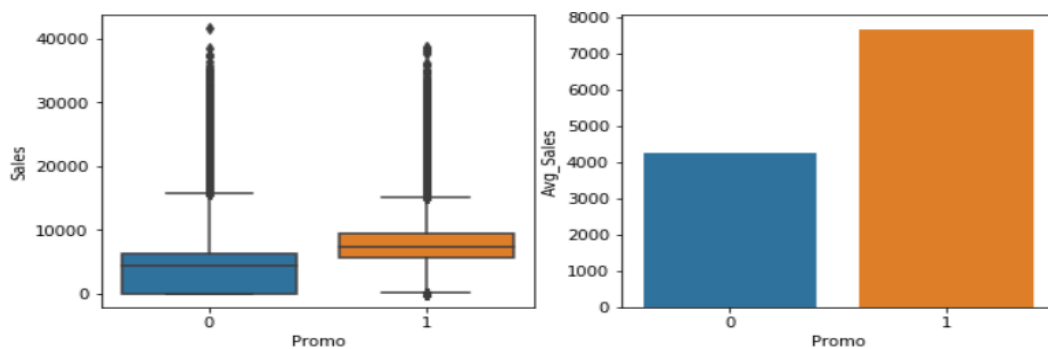


Fig 8. 促销 (Promo) 对销售额的影响

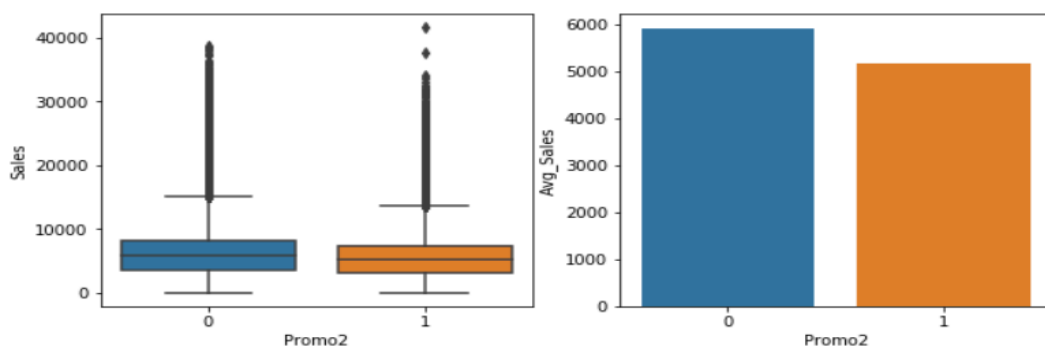


Fig 9. 长期促销 (Promo2) 对销售额的影响

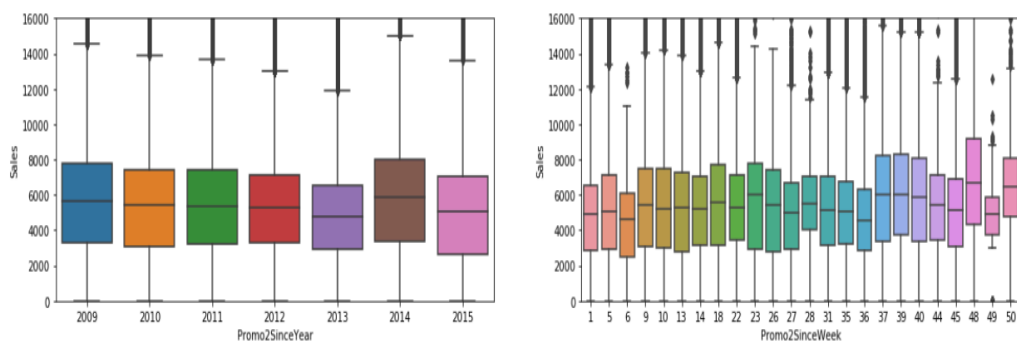


Fig 10. 长期促销开始年份或周 (Promo2SinceYear/Week) 对销售额的影响

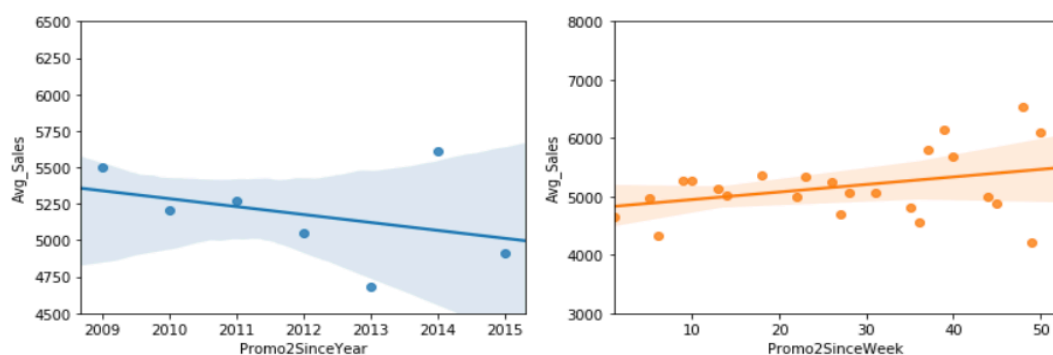


Fig 11. 长期促销开始年份或月份 (Promo2SinceYear/Week) 对平均销售额的影响

由图 Fig 8 可见，促销活动 (Promo) 能够明显地促进销售额增加。而图 Fig 9 显示，参与连续促销 (Promo2) 店铺的销售额总体上略低于未参与连续促销活动的店铺。对参与连续促销 (Promo2) 的店铺而言，连续促销开始的年份和周对销售额的分布影响不明显且无规律可循(如图 Fig 10)。但随着开始年份的增加，以该维度聚合的平均销售额总体呈下降趋势；随着开始周数的增加，以该维度聚合的平均销售额呈略微上升趋势（如图 Fig 11）。

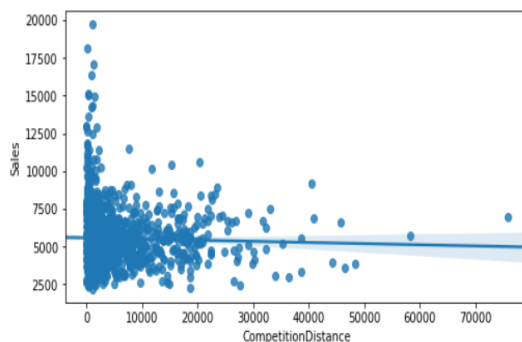


Fig 12. 最近竞争店铺距离 (CompetitionDistance) 对平均销售额的影响

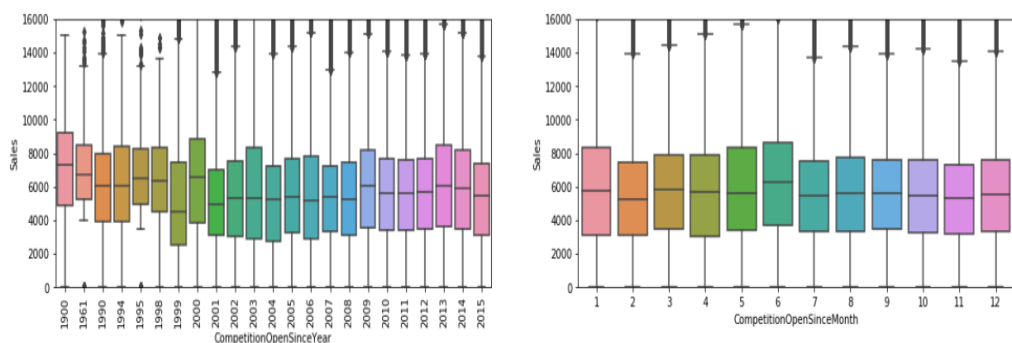


Fig 13. 最近竞争店铺开始年份或月份 (CompetitionOpenSinceYear/Month) 对销售额的影响

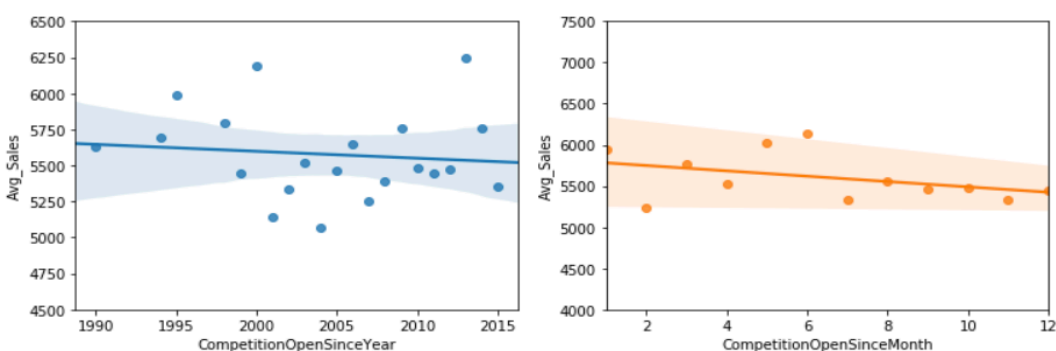


Fig 14. 最近竞争店铺开始年份或月份 (CompetitionOpenSinceYear/Month) 对平均销售额的影响

图 Fig 12 展示了距最近竞争连锁店距离对平均销售额的影响。随着距离的增加，平均销售额总体呈轻微下降趋势。图 Fig 13 分别描述了最近竞争店铺不同开始年份或月份下销售额的分布情况，无明显规律可循。以上两个变量聚合的平均销售额总体上随着开始年份或月份的增加而下降，但趋势很弱（如图 Fig 14）。

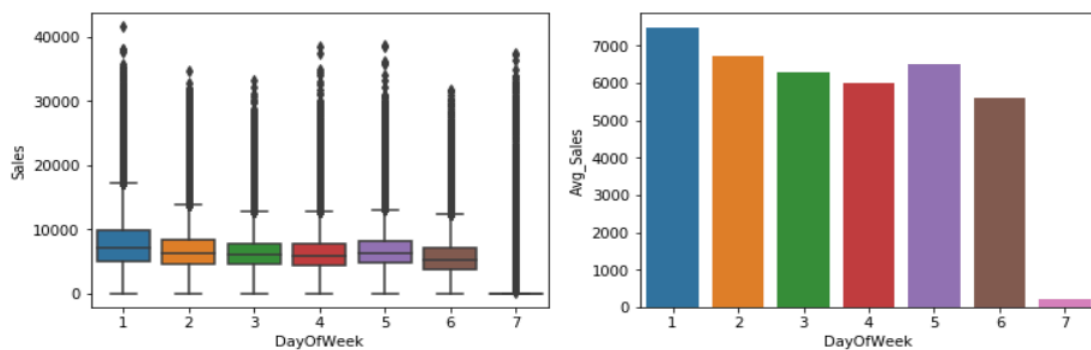


Fig 15. 周 (DayOfWeek) 对销售额的影响

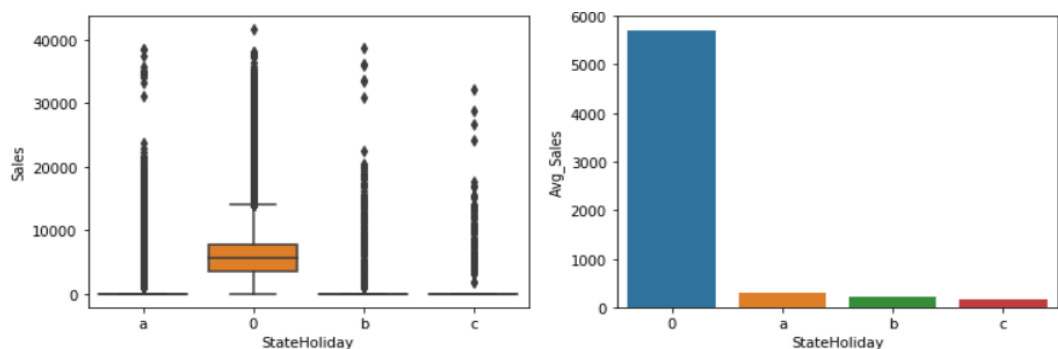


Fig 16. 国家法定假期 (StateHoliday) 对销售额的影响

图 Fig 15 探索了一周当中销售额的变化情况。从周一到周日，销售额总体呈下降趋势。其中周日由于大部分店铺不营业，所以销售额远远低于其它天数。图 Fig 16 探索了国家法定假日对销售额的影响，可以发现只有少部分店在节假日期间营业。对于 a(Public Holiday)、b(Easter Holiday)和 c(Christmas Holiday)三种假期而言，其平均销售额依次降低。

综上所述，就单变量分析结果而言，对销售额影响较强的因素依次是促销因素 (Promo, Promo2)，促销时间(DayOfWeek)、假期因素 (StateHoliday, SchoolHoliday)，店铺因素 (StoreType, Assortment)，最近竞争连锁店因素 (CompetitionDistance, CompetitionOpenSinceYear, CompetitionOpenSinceWeek)，这些都可以作为模型特征。

2. 双变量分析，主要研究两个特征对目标标签销售额的影响：

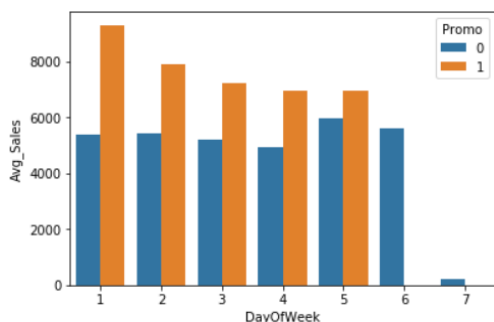


Fig 17. 周(DayOfWeek) 和促销(Promo)对平均销售额的影响

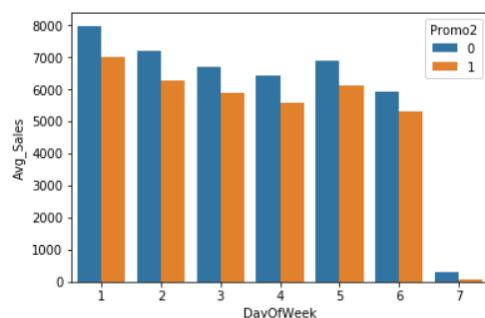


Fig 18. 周(DayOfWeek) 和连续促销(Promo2)对平均销售额的影响

由 Fig 17 可知，促销只在周一到周五开展，周六和周日无促销活动。周一到周六在非促销情况下所有店的平均销售额差别不大（周五和周六略高），周日大部门连锁店不营业，所以销售额最低。促销在某种程度上增加了营业额，促销的增额效果从周一至周五逐渐减弱，表现为在促销情况下周一到周五所有店的平均销售额逐渐下降。由图 Fig 18 可知，从周一至周日，无论参与或未参与连续促销的连锁店均有营

业，且平均营业额均逐渐降低（周五稍有例外），同时未参与连续促销的连锁店平均营业额略高于参与连续促销的连锁店，可能是连续促销无价格优势造成的。

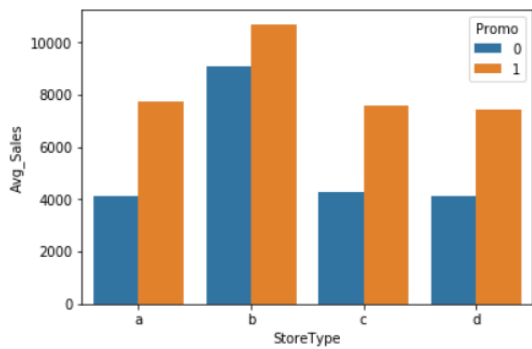


Fig 19. 连锁店类型(StoreType) 和促销(Promo)对销售额的影响

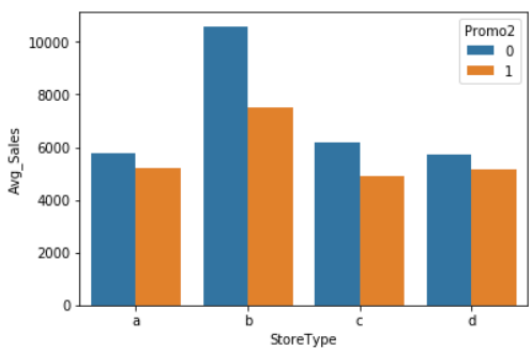


Fig 20. 连锁店类型(StoreType) 和连续促销(Promo2)对销售额的影响

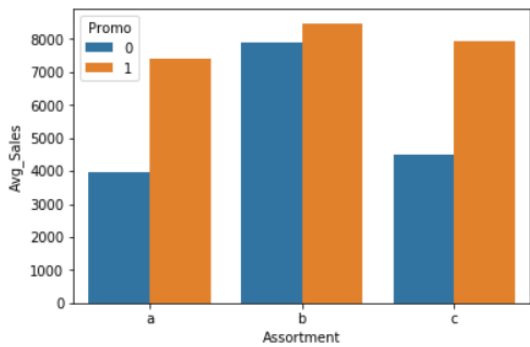


Fig 21. 商店等级(Assortment) 和促销(Promo)对平均销售额的影响

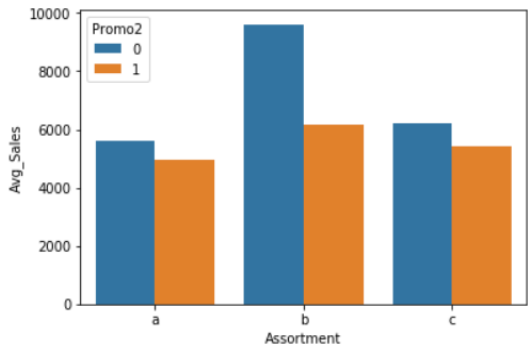


Fig 22. 商店等级(Assortment) 和连续促销(Promo2)对平均销售额的影响

无论（连续）促销与否，b类或b等级连锁店的营业额总体上高于其它类型或等级的连锁店。图 Fig 19 表明，b类连锁店由促销带动营业额增加的效果远远不及其它类型连锁店。a、c和d类店铺促销活动总体上会带来营业额的翻倍，但是b类连锁店营业额增加幅度不到 1/4。图 Fig 20 显示连续促销店铺的平均销售额均低于未参加连续促销的店铺，尤其对于b类店铺而言，参与连续促销导致的平均营业额下降幅度高于其它店铺类型。图 Fig 21 和 Fig 22 所展示的商店等级和促销对销售额的影响规律与之类似。表名促销对不同类型或等级连锁店销售额的影响强度不一样。

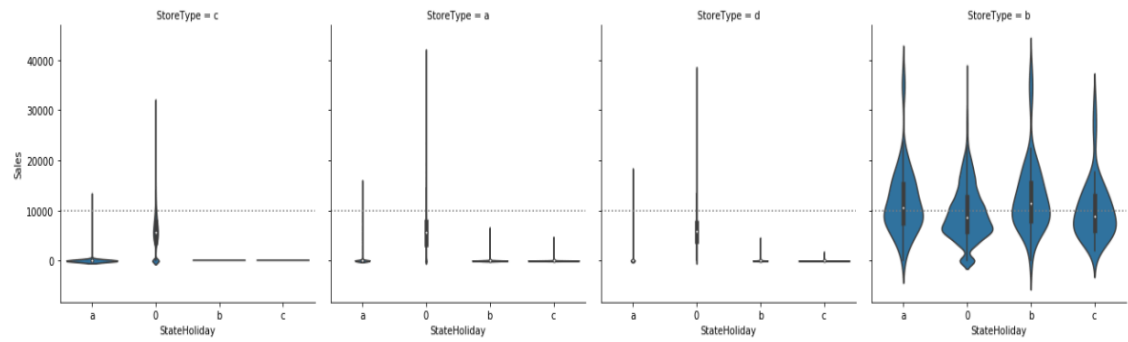


Fig 23. 连锁店类型(StoreType) 和全国假期(StateHoliday)对销售额的影响

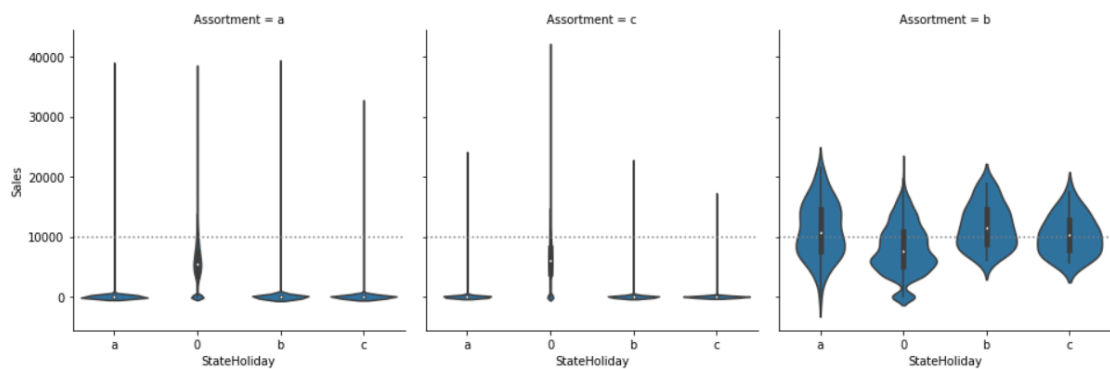


Fig 24. 连锁店等级(Assortment) 和全国假期(StateHoliday)对销售额的影响

图 Fig 23 中 a、c 和 d 类连锁店在全国假期期间销售额相对 b 类较少。对于 b 类连锁店，在假期的销售额总体上高于非假期，例如 Easter holiday(b)销售额 > public holiday(a)销售额 > Christmas(c)销售额。可以推测，b 类连锁店在假期营业的销售额是其营业额的重要组成部分。图 Fig 24 所展示的商店等级和全国假期对销售额的影响规律与之类似，b 等级连锁店在全国假期的销售额总体较非假期高。

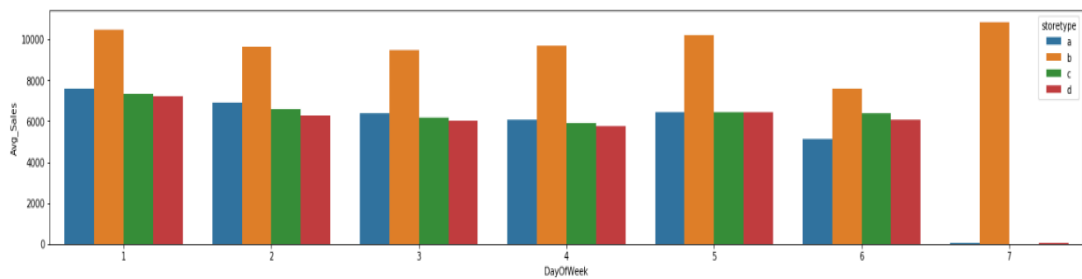


Fig 25. 连锁店类型 (StoreType) 和周 (DayOfWeek) 对平均销售额的影响

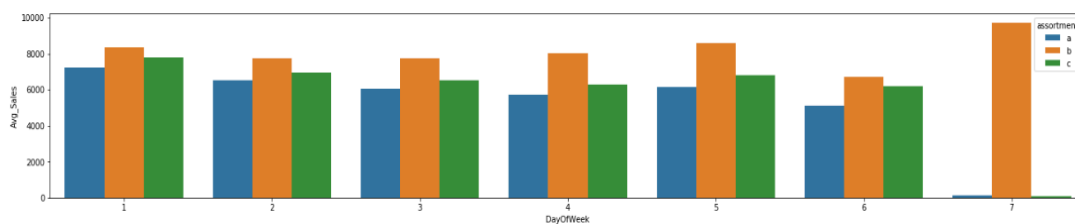


Fig 26. 连锁店等级 (Assortment) 和周 (DayOfWeek) 对平均销售额的影响

图 Fig 25 显示，周一到周六 b 类连锁店平均销售额最高，其它类连锁店平均销售额差距不大，但都大于 b 类销售额的一半。对于 b 类连锁店而言，周日平均销售额是一周中最高的，而其它类连锁店平均销售额几乎可以忽略不计。可以推测，b 类连锁店在周日营业的销售额是其营业额的重要组成部分。图 Fig 26 所展示的商店等级和周对销售额的影响规律与之类似，连锁店在周一到周日的平均销售额总体上是 b 等级 > c 等级 > a 等级，周日销售额几乎都来自 b 等级连锁店。

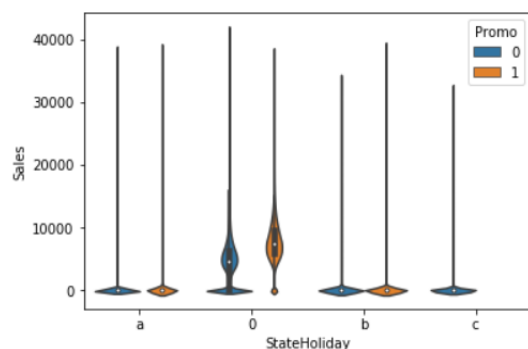


Fig 27. 全国假期 (StateHoliday) 和促销(Promo)对销售额的影响

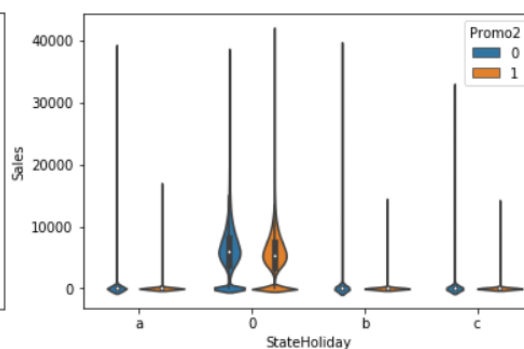


Fig 28. 全国假期 (StateHoliday) 和连续促销(Promo2)对销售额的影响

图 Fig 27 显示圣诞节(c)无促销活动。促销活动对全国假期期间销售额的增加几乎无影响，但是促销活动可以促进非假期期间的销售额。图 Fig 28 也表明了长期促销活动在全国假期期间对销售额没有很大影响。

3. 多变量分析，主要研究三个及以上特征对目标标签销售额的影响：

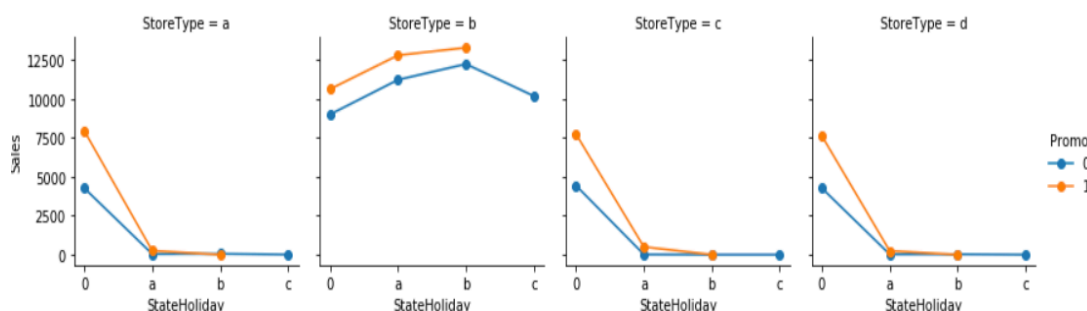


Fig 29. 连锁店类型 (StoreType) 、全国假期 (StateHoliday) 和促销(Promo)对销售额的影响

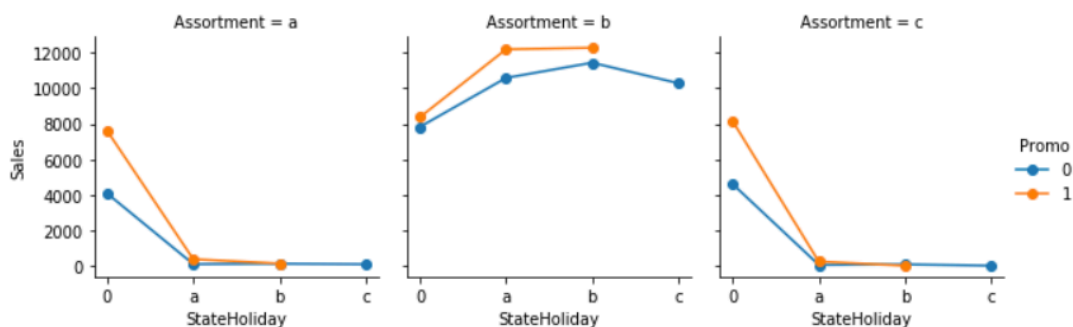


Fig 30. 连锁店等级 (Assortment) 、全国假期 (StateHoliday) 和促销(Promo)对销售额的影响

图 Fig 29 研究了连锁店类型 (或等级)、全国假期和促销活动对平均销售额的影响。对于 a, c, d 类商店, 促销活动只会增加非全国假期的销售额。而对于 b 类商店, 促销对于除圣诞节之外 (圣诞节无促销活动) 所有日期起到增加销售额的作用。图 Fig 30 是关于连锁店或等级、全国假期和促销活动对销售额的影响。与前面类似, 对于 a, c 等级商店, 促销活动不会增加全国假期的销售额。而对于 b 等级商店, 促销增额作用较之明显。

算法和技术

本项目属于多时序类预测问题, 数据集真实且数据量足够大, 所以可以采用有监督的回归类机器学习算法建模。在此类预测问题中, XGBoost 算法作为集成算法 (将若干分类或回归器组合成强学习算法) 的王牌, 以其独特的优势在历届 Kaggle 比赛中被获胜者所使用。

此处我们尝试采用 XGBoost (eXtreme Gradient Boosting, 极端梯度提升) 建模, 其所应用的算法就是 GBDT (gradient boosting decision tree) 的改进, 同时适用于分类和回归。XGBoost 是大规模并行提升树的工具。在预测算法中, 提升树采用了树集成的方法。什么是树集成呢? 在实际的预测模型建立过程中, 我们通过不断地增加新的回归树, 并给每个回归树赋予合适的权重, 在此基础上综合不同的回归树得分获得更为准确的预测结果, 这也就是树集成的基本思路[9]。

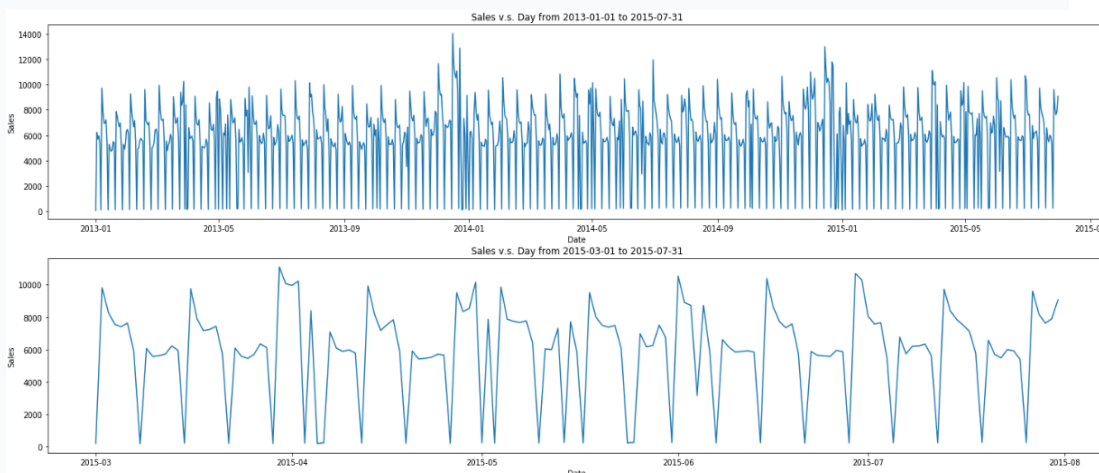


Fig 31. 历史日均销售额变化趋势图

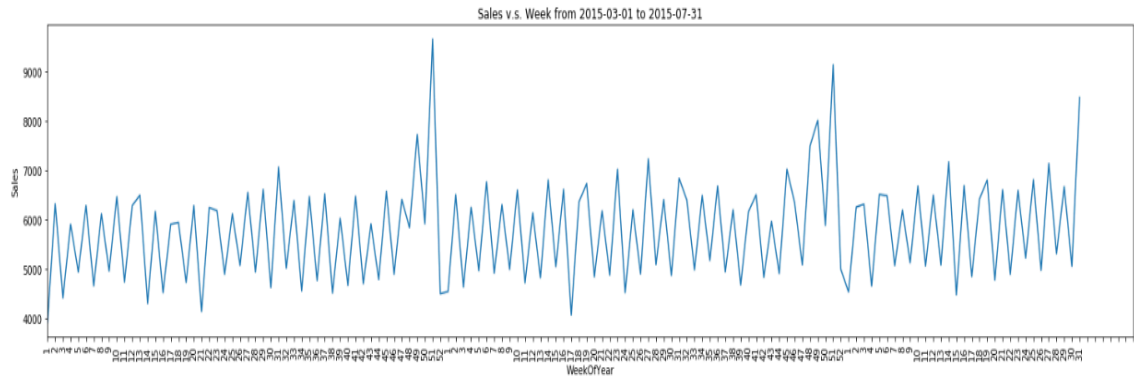


Fig 32. 历史周均销售额变化趋势图

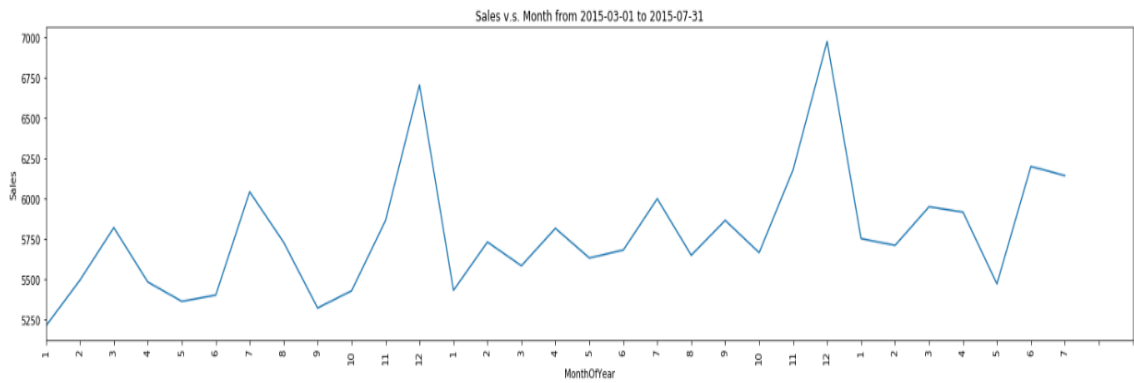


Fig 33. 历史月均销售额变化趋势图

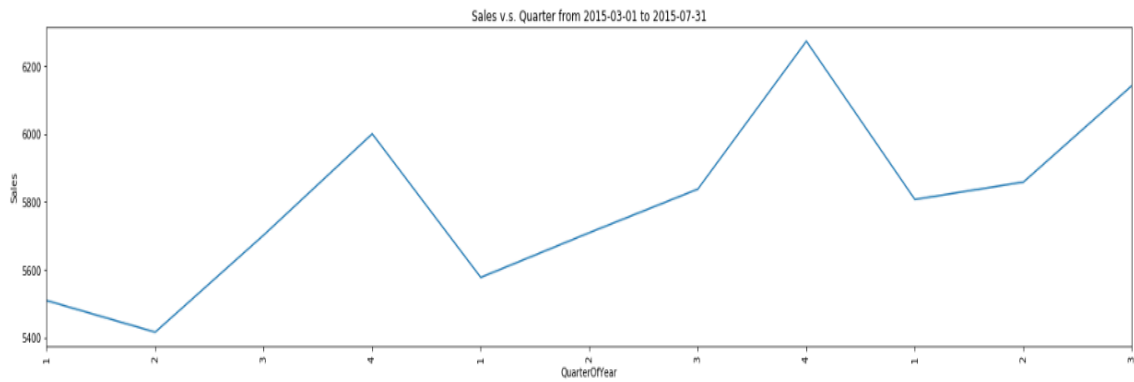


Fig 34. 历史季均销售额变化趋势图

在建模之前需要先做特征工程，即需要先确定输入模型的变量。基于单变量和多变量探索结果，在原有特征的基础上添加了如下新特征：1.各连锁店所在州数据[7]；2.时间点特征：由图 Fig31-Fig34 可知，所有店铺平均销售额具有日、周、月和年的周期性变化，所以添加观察日期的年（'Year'）、月

('MonthOfYear')、周 ('WeekOfYear') 和日 ('DayOfMonth'和'DayOfYear')；3.时间间隔特征：由图 Fig35 可知，周末及全国假期所处时间点销量大多处于峰谷或较小值，观察时间接近这些时间点时销量通常不高，所以我们可以尝试构建时间间隔变量，例如观察日期距离最近全国假期的时间间隔 ('StateHoliday_DayDiff')、观察日期距离最近周末时间间隔 ('Weekends_DayDiff')、观察日期距离最近周末或全国假期的时间间隔 ('Weekends_StateHoliday_DayDiff')；4.时间段特征：最近竞争店铺距离观察时间的开业月份数 ('Competition_month_num') 和连锁店开始参与长期促销的周数 ('Promo2_week_num')。入模前所有分类变量做了数值映射，因为 XGBoost 模型不支持分类变量。'Sales' 和 'CompetitionDistance' 两个变量做了对数转换。删除不用于建模的变量 'Customers' 和 'Id'。然而，原始特征和上述这些特征不能直接为 XGBoost 模型所用，因为部分特征权重低对建模结果影响微弱，或某些特征有可能存在多重共线性。所以我们采取基模型 Random Forest 回归模型进行训练（注：训练集、验证集和测试集的划分方法请见问题陈述部分，其中标签 'Sales' 需要做对数转换，同时建模时只考虑营业且销售额大于 0 的样本），通过其 feature_importance_属性筛选出重要性大于 0.001 的特征，并最终保留 'spearman' 线性相关系数不大于 0.7 的特征作为最终特征集，包括如下特征：'Promo', 'CompetitionDistance', 'State_level', 'Store', 'Promo2SinceYear', 'Weekends_StateHoliday_DayDiff', 'Assortment_level', 'CompetitionOpenSinceMonth', 'StoreType_level', 'DayOfWeek', 'CompetitionOpenSinceYear', 'DayOfYear', 'Promo2_week_num', 'StateHoliday_DayDiff', 'Competition_month_num', 'SchoolHoliday', 'DayOfMonth'。

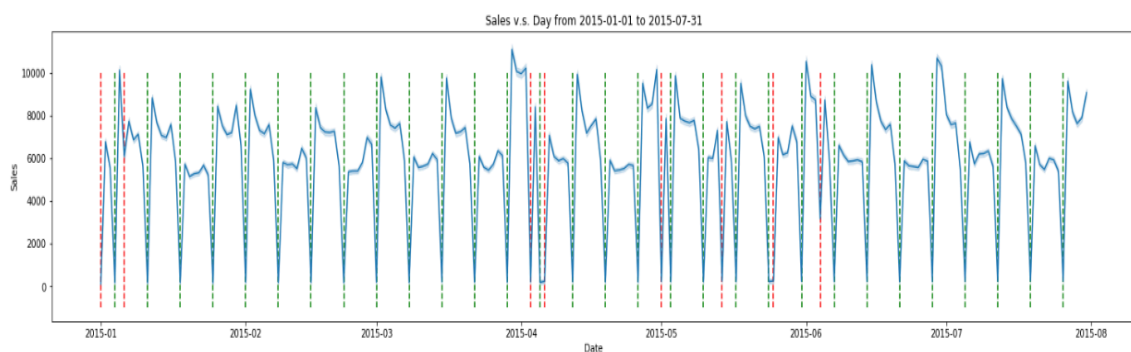


Fig 35. 历史日均销售额变化趋势图（绿色和红色虚线分别表示周末及全国假期所处时间线）

Xgboost 模型有 3 种类型的参数：通用参数、辅助参数和任务参数。通用参数确定提升过程中提升模型类型，常用树或线性模型；辅助参数取决于所选的提升模型；任务参数定义学习场景[10]。此处 XGBoost 线性回归（'objective' 选取 'reg:linear'）选取的提升模型是树模型 (gbtree)，通用参数 'verbosity' 表示是否打印信息，默认为 1。该模型辅助参数包括 'eta'：学习速率，取值范围 (0, 1]，在更新叶子节点的时候，权重乘以该速率，以避免在更新过程中过拟合；'max_depth'：每棵树的最大深度，取值范围[0,∞]，树越深，越容易过拟合；'subsample'：训练集的子样本比

例，取值范围是（0,1]，值为0.5时表示Xgboost随机抽取一半的训练样本数据来生成树模型，这样可以防止过拟合；‘colsample_bytree’：在构建每棵树时，列（特征）的子样本比，取值范围是（0,1]。任务参数只使用了随机数种子‘seed’，以确保数据的可重现性[11]。

用上述给定的参数训练提升树模型，除此之外，训练时还需要设置一些其它参数[11]。

‘num_boost_round’：提升迭代次数；‘evals’：是由（矩阵，字符串）元素组成的验证集列表，在训练模型时验证集的度量标准会评估并追踪模型性能；‘feval’：自定义评估函数。此处该函数设置为均方根百分比误差（见评价指标），其中标签值与预测值需要做指数转换（因为建模时标签做了对数转换）；‘early_stopping_rounds’：如果训练时迭代次数过多模型会进入过拟合。表现就是随着迭代次数的增加，测试集（或验证集）上的测试误差开始下降。当开始过拟合或者过训练时，测试集上的测试误差开始上升，或者说波动。该参数就是为解决因为迭代次数过多而导致过拟合，即如果误差在所设定值的迭代次数之内都没有提升的话，就结束迭代[12]。

Xgboost 是很多 CART 回归树集成，我们希望通过建立 K 个回归树，使得树群的预测值尽量接近真实值，而且有尽量大的泛化能力，从数学角度看这是一个最优化问题，目标函数如下：

$$L(\phi) = \sum_i l(\hat{y}_i - y_i) + \sum_k \Omega(f_k)$$

其中第一项表示所有样本的预测误差，第二项表示树的复杂度函数，复杂度越小泛化能力越强。我们可以进一步将损失函数表示为：

$$L(\phi) = \sum_i l(w - y_i) + \gamma + \frac{1}{2} \lambda \|w\|^2$$

将误差项表示为所有样本的平方误差，复杂度函数表示为 L2 正则项。那么该函数就变成了一个关于 w 的二次函数。我们需要不断选取分裂的特征，即选择损失函数效果最好的那个特征，也就是对上述二次损失函数求最小值，取最小值的点就是这个节点的预测值。按这种贪心策略不断分裂下去，形成一颗又一颗的树，直到树达到设置的最大深度或满足‘early_stopping_rounds’条件时停止[13]。

基准模型

为了与 XGBoost 模型性能做对比，引入了另一个集成模型随机森林回归模型

(RandomForestRegressor)，选择该基模型的另一个目的是通过特征重要性做特征筛选。虽然该模型也属于集成模型，但是原理与 XGBoost 大不相同。集成学习的目的是通过结合多个基学习器的预测结果来改善基本学习器的泛化能力和鲁棒性。根据基本学习器的生成方式，目前的集成学习方法大致分为两大类：即基本学习器之间存在强依赖关系、必须串行生成的序列化方法，以及基本学习器间不存在强依赖关系、可同时生成的并行化方法；前者的代表就是 Boosting（代表模型是 XGBoost），后者的代表是

Bagging（代表模型是 Random Forest）。随机森林模型的构建包括四个部分：1、随机选择样本（放回抽样）；2、随机选择特征属性；3、构建决策树；4、随机森林投票（平均）[14]。

我们将前面特征工程选定的特征训练集输入 RandomForestRegressor 模型训练，设置模型参数最大树深 ‘max_depth’ 为 10，随机数种子 random_state 为 0，其它参数采用默认值。模型在验证集上的 rmspe 约为 0.28540，不是很理想。希望验证集在 XGBoost 模型上的预测误差小于该值。

III. 方法

数据预处理

数据探索部分已详尽描述了缺失数据、异常数据及需要做转换的数据的处理方式。针对特征集中的分类变量，此处均采用 map 数值映射的方法处理成数值变量。其它数值变量较为完整无需预处理。关于特征选取，在算法和技术部分的特征工程（第三段）里已做记录。

执行过程

关于 XGBoost 模型最终参数设置与训练代码如下：

```
## XGBoost模型参数
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.06,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.7,
          'verbosity':1,
          "seed": 10
        }

dtrain = xgb.DMatrix(X_train[features], y_train)
dvalid = xgb.DMatrix(X_valid[features], y_valid)
watchlist = [(dtrain, 'train'), (dvalid, 'eval')]

## 模型训练参数
model_xgboost = xgb.train(
    params,
    dtrain,
    num_boost_round=1000,
    evals = watchlist,
    feval= rmspe_xg,
    early_stopping_rounds=100,
    verbose_eval=1)
```

这里需要补充说明一下该模型的自定义评估函数。这是基于项目评价指标的要求，同时结合数据探索部分分析结果，将目标标签 ‘Sales’ 做对数转换，所以评估函数需要相应地做指数还原，如下所示：

```
# XGBoost模型自定义评估函数
def rmspe_xg(yhat, y):
    y = np.expml(y.get_label())
    yhat = np.expml(yhat)
    return "rmspe", rmspe(y,yhat)
```

因为参数 ‘verbosity’ 设置为 1，所以可以打印出模型执行信息，下面分别展示开头、中间和结尾 10 行的信息：

```
[0]      train-rmse:7.77174      eval-rmse:7.80559      train-rmspe:0.999707      eval-rmspe:0.999719
Multiple eval metrics have been passed: 'eval-rmspe' will be used for early stopping.

Will train until eval-rmspe hasn't improved in 100 rounds.
[1]      train-rmse:7.30655      eval-rmse:7.34077      train-rmspe:0.999433      eval-rmspe:0.999455
[2]      train-rmse:6.86927      eval-rmse:6.90362      train-rmspe:0.999029      eval-rmspe:0.999068
[3]      train-rmse:6.45812      eval-rmse:6.49244      train-rmspe:0.998455      eval-rmspe:0.998516
[4]      train-rmse:6.07176      eval-rmse:6.10632      train-rmspe:0.997651      eval-rmspe:0.997743
[5]      train-rmse:5.70851      eval-rmse:5.74309      train-rmspe:0.996558      eval-rmspe:0.996692
[6]      train-rmse:5.36706      eval-rmse:5.40127      train-rmspe:0.995102      eval-rmspe:0.995291
[7]      train-rmse:5.04635      eval-rmse:5.08264      train-rmspe:0.993196      eval-rmspe:0.993469
[8]      train-rmse:4.74478      eval-rmse:4.78106      train-rmspe:0.990761      eval-rmspe:0.99113
[9]      train-rmse:4.46133      eval-rmse:4.49868      train-rmspe:0.987704      eval-rmspe:0.988206
[10]     train-rmse:4.19487      eval-rmse:4.23193      train-rmspe:0.983939      eval-rmspe:0.984589

...

[494]     train-rmse:0.088755      eval-rmse:0.122069      train-rmspe:0.110199      eval-rmspe:0.120647
[495]     train-rmse:0.088695      eval-rmse:0.122038      train-rmspe:0.110221      eval-rmspe:0.120619
[496]     train-rmse:0.088535      eval-rmse:0.121969      train-rmspe:0.110071      eval-rmspe:0.120552
[497]     train-rmse:0.088516      eval-rmse:0.121958      train-rmspe:0.110055      eval-rmspe:0.120533
[498]     train-rmse:0.088476      eval-rmse:0.12193      train-rmspe:0.10999      eval-rmspe:0.120509
[499]     train-rmse:0.088458      eval-rmse:0.121916      train-rmspe:0.109973      eval-rmspe:0.120496
[500]     train-rmse:0.088428      eval-rmse:0.121895      train-rmspe:0.109943      eval-rmspe:0.120479
[501]     train-rmse:0.088383      eval-rmse:0.121877      train-rmspe:0.109889      eval-rmspe:0.120451
[502]     train-rmse:0.088323      eval-rmse:0.121808      train-rmspe:0.109825      eval-rmspe:0.1204
[503]     train-rmse:0.088272      eval-rmse:0.121796      train-rmspe:0.109771      eval-rmspe:0.120387
[504]     train-rmse:0.088188      eval-rmse:0.121752      train-rmspe:0.109662      eval-rmspe:0.120331

...

[990]     train-rmse:0.073009      eval-rmse:0.116414      train-rmspe:0.080431      eval-rmspe:0.11541
[991]     train-rmse:0.072986      eval-rmse:0.116431      train-rmspe:0.080397      eval-rmspe:0.115429
[992]     train-rmse:0.072973      eval-rmse:0.116427      train-rmspe:0.080385      eval-rmspe:0.115424
[993]     train-rmse:0.072955      eval-rmse:0.116424      train-rmspe:0.080369      eval-rmspe:0.115424
[994]     train-rmse:0.072903      eval-rmse:0.116397      train-rmspe:0.080314      eval-rmspe:0.115398
[995]     train-rmse:0.072883      eval-rmse:0.116409      train-rmspe:0.080292      eval-rmspe:0.115407
[996]     train-rmse:0.072878      eval-rmse:0.116409      train-rmspe:0.080287      eval-rmspe:0.115407
[997]     train-rmse:0.072873      eval-rmse:0.116408      train-rmspe:0.080281      eval-rmspe:0.115407
[998]     train-rmse:0.072852      eval-rmse:0.1164      train-rmspe:0.08026      eval-rmspe:0.115397
[999]     train-rmse:0.072831      eval-rmse:0.116404      train-rmspe:0.080238      eval-rmspe:0.115399
```

可以发现，开始时训练集和验证集的预测误差（rmspe）下降迅速，到中间时虽有下降但是幅度放缓，直至最后误差值几乎徘徊稳定在某值附近。这也完全符合 XGBoost 损失函数的下降趋势。

最终上述模型在验证集上的 rmspe 值是 0.12623，在测试集上的 rmspe 值是 0.11736，达到了该毕业项目的要求。

完善

该模型初始参数如下所示，部分参数参考了 kaggle 文件[15]:

```
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.05,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.7,
          'verbosity':1,
          "seed": 10
        }
```

在此基础上，尝试调整学习速率 ‘eta’，观察是否可以提升模型性能。表 Tab1 和图 Fig36 显示了不同学习速率下模型在训练集、验证集和测试集上的 rmspe 值。当学习速率 eta=0.06 时，模型在测试集上的 rmspe 值最小，即预测误差最小、模型性能最好。

Tab 1 不同学习速率下模型在不同数据集上的 rmspe 值

eta	训练集 rmspe	验证集 rmspe	测试集 rmspe
0.04	0.08177	0.12518	0.12273
0.05	0.07658	0.12503	0.12070
0.06	0.07283	0.12623	0.11736
0.07	0.06996	0.12724	0.11838
0.08	0.06692	0.12765	0.11983

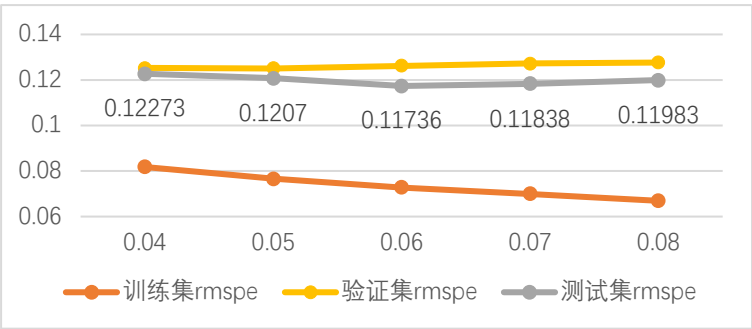


Fig 36 不同学习速率下模型在训练集、验证集和测试集上的 rmspe 值变化趋势

虽然已经达到了毕业项目要求，但是可以再尝试调整其它参数，看能否进一步优化模型性能。使 eta=0.06 时，继续调整训练集的特征子样本比例 ‘colsample_bytree’，表 Tab2 显示了不同

‘colsample_bytree’ 下模型在各个数据集上的 rmspe 值。可以发现初始值已是最优值，因为 colsample_bytree=0.7 时，模型在测试集上的 rmspe 值最小。

Tab 2 不同 colsample_bytree 下模型在不同数据集上的 rmspe 值

colsample_bytree	训练集 rmspe	验证集 rmspe	测试集 rmspe
0.6	0.07380	0.12675	0.11783
0.7	0.07283	0.12623	0.11736
0.8	0.07161	0.12188	0.12044

使 eta=0.06、colsample_bytree=0.7 时，继续调整训练集的子样本比例 ‘subsample’，由表 Tab3 可知，subsample=0.9 时，模型在测试集上的 rmspe 值最小。

Tab 3 不同 subsample 下模型在不同数据集上的 rmspe 值

subsample	训练集 rmspe	验证集 rmspe	测试集 rmspe
0.8	0.07251	0.12488	0.11740
0.9	0.07283	0.12623	0.11736
1	0.07460	0.12500	0.12262

最后，调整参数最大深度 max_depth，增加该值会使得模型更复杂且更容易过拟合。所以该值不是越大越好。当最大深度从 9 增加到 10 时，模型在测试集上的 rmspe 下降明显，但是从 10 增加到 11 时，下降微弱，所以选择 max_depth=10。

Tab 4 不同 max_depth 下模型在不同数据集上的 rmspe 值

max_depth	训练集 rmspe	验证集 rmspe	测试集 rmspe
9	0.08296	0.12647	0.11964
10	0.07283	0.12623	0.11736
11	0.06297	0.12558	0.11756

综上所述，在原始参数基础上，只调整学习速率 eta=0.06，此时模型在测试集上的性能最优，rmspe=0.11736，符合毕业项目要求。

IV. 结果

模型的评价与验证

我认为最终的模型是合理的，跟期待的结果基本一致，设置的各种参数都在合理的取值范围内。预测模型也足够稳健，从表 Tab1 (Fig 36)-Tab4 可知，部分参数微小的改变并不会极大影响测试集 rmspe 值。这个模型得出的结果是可信的。

合理性分析

XGBoost 模型最终结果比基准模型 RandomForestRegressor 表现更好。由表 Tab5 可知，前者在训练集、验证集和测试集上的 rmspe 值都远远小于后者。图 Fig37、Fig38 和 Fig39 分别以连锁店 1、66、666 为例，展示了两种模型在验证集上的效果。整体上 XGBoost 模型在验证集上的预测值更接近真实值。尤其在 1 号店上的效果尤为明显。XGBoost 模型最终结果确实解决了销售量预测问题。

Tab 5 RFR 和 XGBoost 模型在不同数据集上的 rmspe 值

模型	训练集 rmspe	验证集 rmspe	测试集 rmspe
RandomForestRegressor	0.33664	0.28540	0.30866
XGBoost	0.07283	0.12623	0.11736

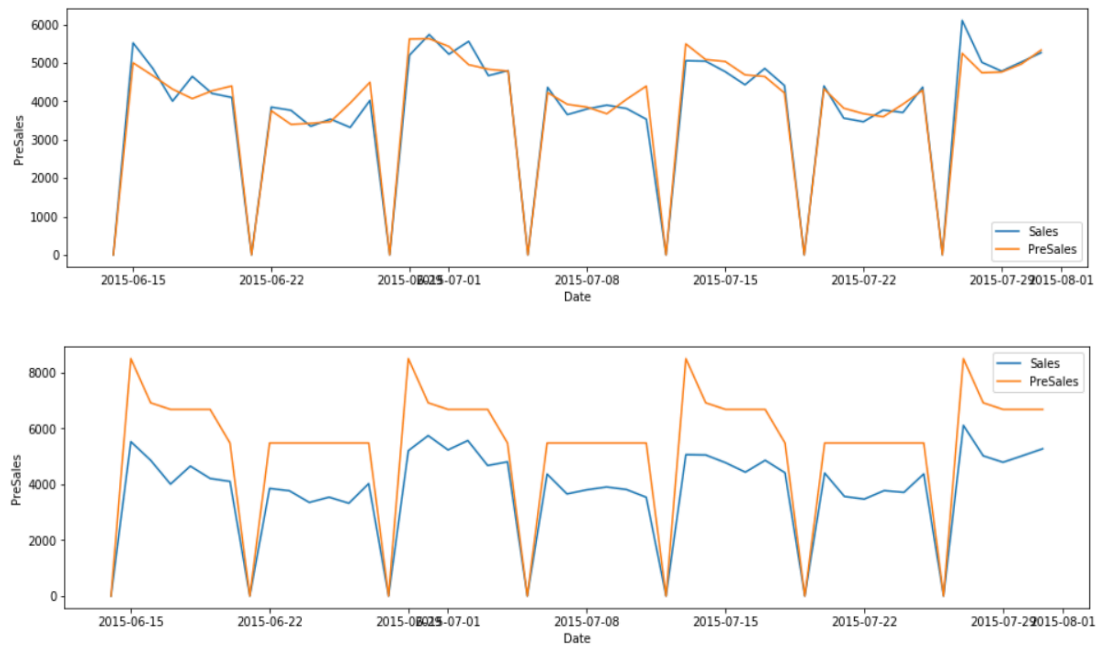


Fig 37 1号店 RFR(下)和 XGBoost (上) 模型在验证集上的预测值(PreSales)和真实值(Sales)

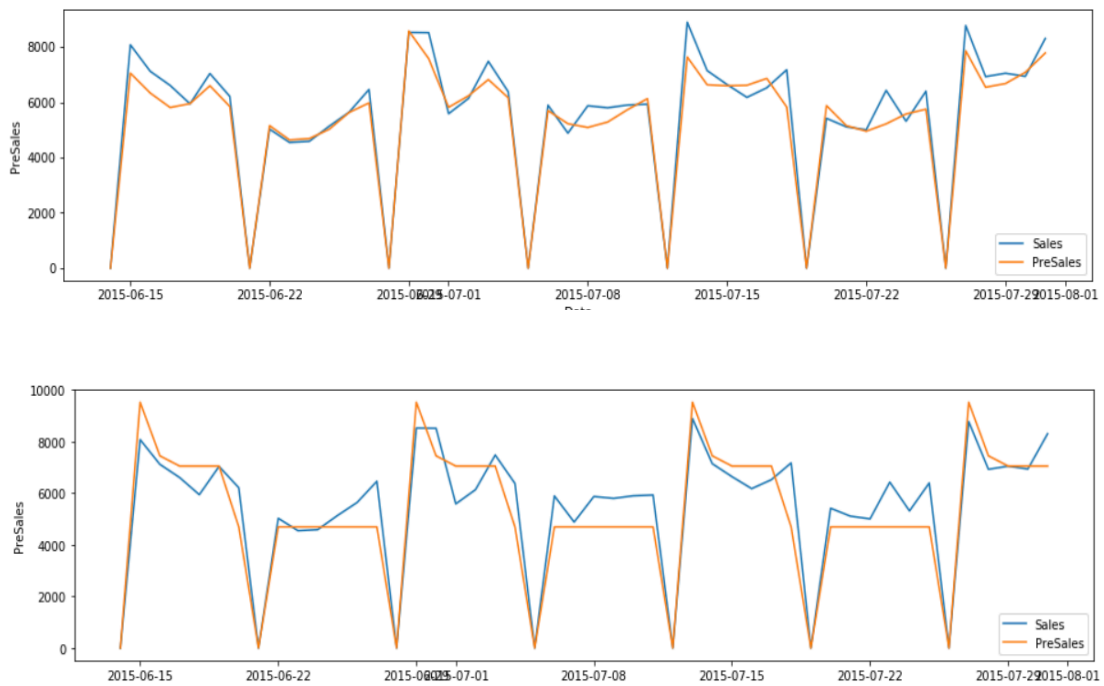


Fig 38 66号店 RFR(下)和 XGBoost (上) 模型在验证集上的预测值(PreSales)和真实值(Sales)

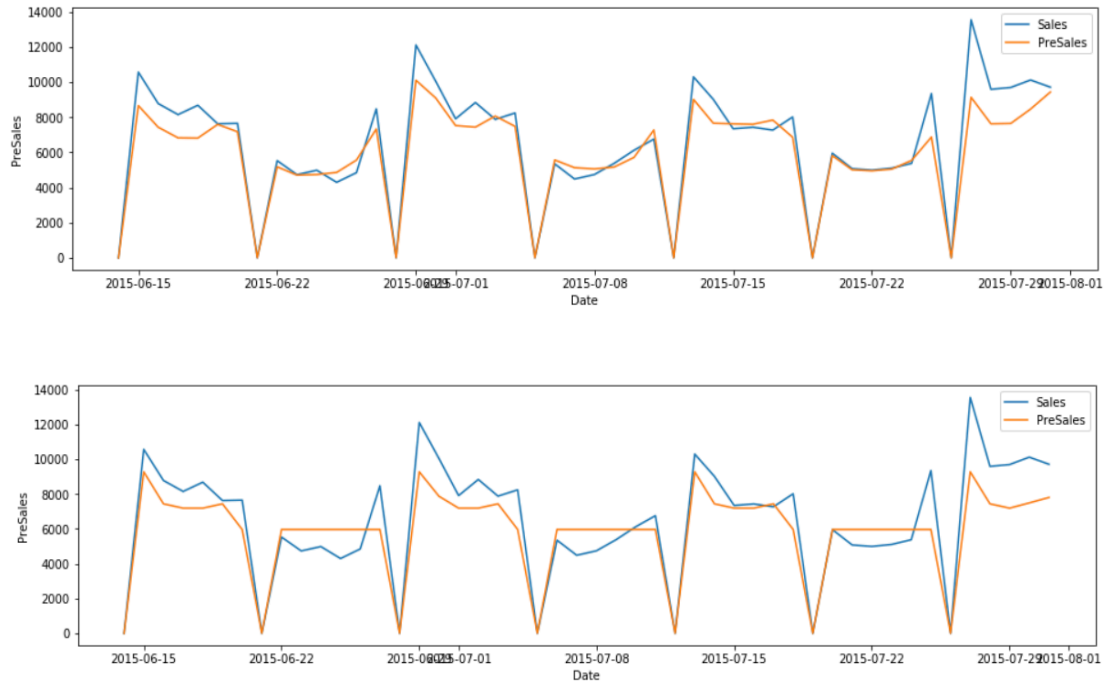


Fig 39 666 号店 RFR(下)和 XGBoost (上) 模型在验证集上的预测值(PreSales)和真实值(Sales)

V. 项目结论

结果可视化

在数据探索部分，我们已经通过可视化图形分析了输入数据的特点、处理方式，以及各变量对预

测标签的影响。而在算法和技术部分，在挖掘新变量的过程中，也进一步通过图形 (Fig31-Fig35) 探讨了多时序类数据集的特性。在模型完善部分，通过图 (Fig36) 表展示了不同参数对预测结果的影响。在结果分析部分，以三个连锁店为例，对比了 RFR 基模型和 XGBoost 模型在验证集上的预测效果 (Fig37-Fig39)。Fig40 和 Fig41 分别展示了 RFR 模型和 XGBoost 模型中特征重要性排序，二者锁定的特征基本一致，但重要性排序有所差别。

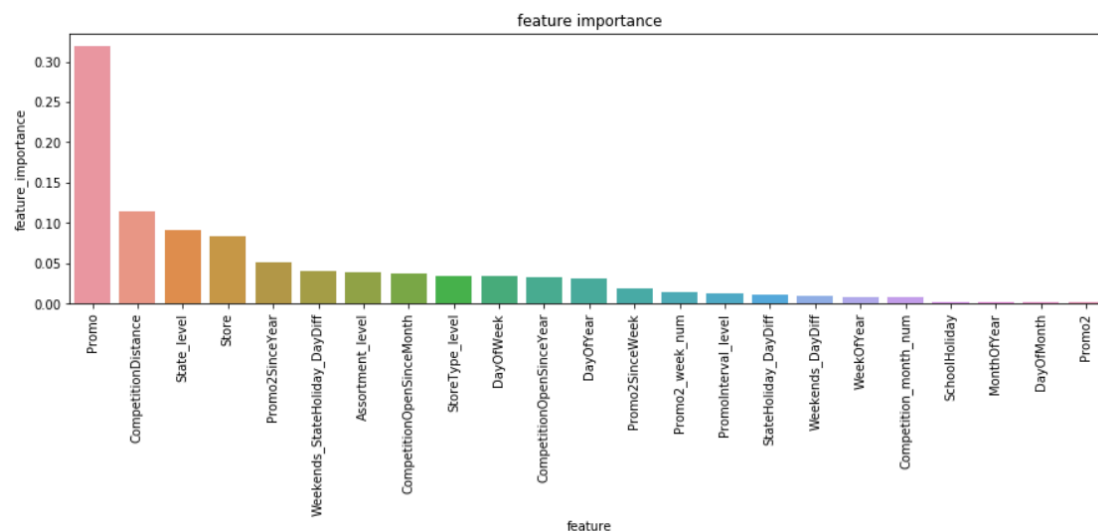


Fig 40 RFR 模型特征重要性

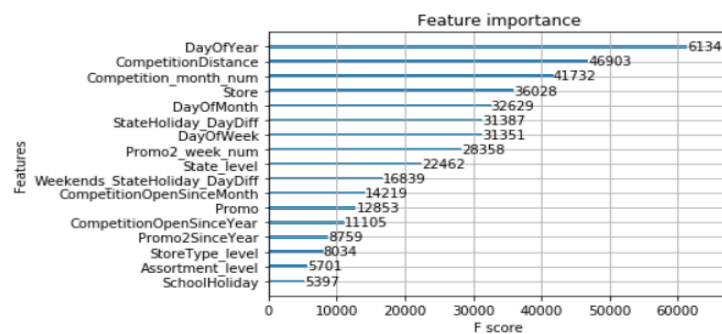


Fig 41 XGBoost 模型特征重要性


对项目的思考

报告的各个部分已经充分总结了项目的整个流程。项目中比较有意思的是挖掘特征变量的部分。我们可以寻找外部数据源，同时可以在已有的特征上衍生出新的特征，而这些特征会直接关系到模型的性能。因为数据和特征决定了模型的上限，而算法和模型只能让我们无限接近这个上限。另一个有意思的部分是模型调参，例如对学习速率微小的改变却可以使模型性能得到较大的提升。项目中比较困难的地方这是个多时序类预测问题，建模数据集的处理和划分不同于常规的时间截面数据问题，直接使用时间序列常用分析模型 AR、MA、ARMA、ARIMA 有些困难，但是集成算法类模型（例如 XGBoost）能够很好的解决这类问题。最终模型和结果符合我对这个问题的预期。我认为该模型可以通用于此类场景的预测问题。

需要作出的改进

在项目中有可以进一步完善的地方，例如模型参数调优环节，我用了枚举法逐个调整并确定最优参数，很幸运地在尝试几次调整后达到了项目要求的结果。但是此处可以尝试用网格搜索法，能够快速找到最优的参数组合，与前者相比缺点是运行时间长。如果将最终模型作为新的基准，我认为参数还有调优的空间。

模型得分截图

Rossmann Store Sales				
<div> Forecast sales using store, promotion, and competitor data \$35,000 · 3,303 teams · 4 years ago</div>				
Overview	Data	Kernels	Discussion	Leaderboard
			Rules	Team
				My Submissions
				Late Submission
Your most recent submission				
Name submission.csv	Submitted a day ago	Wait time 0 seconds	Execution time 0 seconds	Score 0.11736
Complete				
Jump to your position on the leaderboard				

参考文献

1. Pete Furseth and Jason Stumbaugh . The Importance of Sales Forecasting. ORM Technologies August 24, 2017.
2. 'Rossmann Store Sales', Kaggle.Com, URL: <http://www.kaggle.com/c/rossmann-store-sales>
3. <https://www.jianshu.com/p/e81ab6846214>
4. <https://www.jianshu.com/p/e81ab6846214>
5. <https://www.kaggle.com/c/rossmann-store-sales/discussion/17919#latest-295558>
6. <https://www.kaggle.com/c/rossmann-store-sales/overview/evaluation>
7. https://github.com/entron/entity-embedding-rossmann/blob/master/store_states.csv
8. https://blog.csdn.net/anshuai_aw1/article/details/83866105
9. <http://www.imooc.com/article/257226>
10. https://blog.csdn.net/weixin_39541558/article/details/80746084
11. <https://xgboost.readthedocs.io/en/latest/parameter.html>
12. <https://blog.csdn.net/lujiandong1/article/details/52777168>
13. http://www.sohu.com/a/226265476_609569
14. <https://blog.csdn.net/zwqjoy/article/details/82150528>
15. <https://www.kaggle.com/danspace/rossmann-store-sales-xgboost>