机器学习工程师纳米学位毕业项目

Rossman 商店销售额预测

Capstone Report

陈俞飞

2019 年 5 月 28 日

1. 定义

1.1 项目概述

Rossman 是一个在欧洲 7 个国家有 3000 家分店的药品连锁店。现在各个 Rossman 连锁店的经理们正在研究如何提前预测接下来 6 周的日销售额。商店的销售额受到诸如促销，竞争，学校，节假日，区位等多种因素的影响。由于每家分店的经理们都根据各自分店的实际情况进行了销售额预测，导致预测结果的准确性很不一样。

Rossman 的管理者们希望可以通过本次项目得到一些优秀的模型和特征，以便其更准确地对各分店做出销售营业额预测，提高集团整体决策效率。

1.2 问题陈述

根据 Rossman 方面的要求和其提供的数据，我们需要根据过去一段时间内商店的经营情况来对未来一段时间商店的营业额作出预测。

这个问题可以用机器学习-监督学习中的回归方法来进行解决，我们可以训练相关的机器学习回归模型（线性回归，随机森林，xgboost, lightgbm 等）来对商店未来一段时间的营业额作出预测，并进行模型预测结果的评估。

1.3 评价指标

根据 Kaggle 上项目发起人的要求，我们选择 RMSPE 作为我们本次项目的评估指标，计算公式如下：

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

其中 y_i 为真实值，y_i-hat 为预测值。

2. 分析

2.1 数据探索

该项目共有三分数据文件，分别为 train.csv, test.csv, store.csv，
下载路径为：
https://www.kaggle.com/c/rossmann-store-sales/data
train.csv：具有 label 的历史销售数据，需要用于训练构建模型，共有 1017209 条记录
test.csv：无 label 的历史销售数据，需要用于测试训练的模型，共有 41088 条记录
sample_submission.csv：提交的预测数据的正确格式样本文件
store.csv： Rossmann 经营的 1115 家商店的属性信息

特征名称	描述	备注
Store	每家商店的编号	
Sales	表示当天的销售额	
Customers	表示当天的消费者数量	
Open	指示说明商店当天是否营业	0 表示商店关门，1 表示开门营业
DayofWeek	星期几	取值：1-7
Date	日期，格式为 YYYY-MM-DD	
StateHoliday	表明当天是否为国家法定节假日	a 表示公众假期;b 表示复活节假期;c 表示圣诞假期;0 表示不是假期
SchoolHoliday	表明当天是否为学校假期	
StoreType	说明商店的类型	a,b,c,d
Assortment	说明商店经营策略的类型	a 表示基础型,b 表示额外型,c 表示扩展型
CompetitionDistance	最近竞争商铺的距离	
CompetitionOpenSicne	竞争者从什么时候开始营业	CompetitionOpenSicneMonth=开始经营月份;CompetitionOpenSicneYear=开始经营年份
Promo	说明给定日期时商店是否有进行促销	0 表示不进行促销，1 表示进行促销
Promo2	说明商店是否有进行连续的促销活动	0 表示商店不参与连续促销;1 表示商店参与连续促销
Promo2Since	描述了开始参与连续性促销的日期	Promo2SinceWeek=参与促销的月份;Promo2SinceYear=参与促销的年份
PromoInterval	描述有连续性促销的间隔	哪些月份有连续性促销

```
#需要预测2015年8月1日至2015年9月17日之间的
print(test.Date.max())
print(test.Date.min())
```

```
2015-09-17
2015-08-01
```

```
#训练集提供2013年1月1日至2015年7月31日的数据
print(train.Date.max())
print(train.Date.min())
```

```
2015-07-31
2013-01-01
```

经过观察，我们得知训练集中提供了 2013 年 1 月 1 日至 2015 年 7 月 31 日的相关数据，需要我们去对 2015 年 8 月 1 日至 2015 年 9 月 17 日的营业额进行预测。

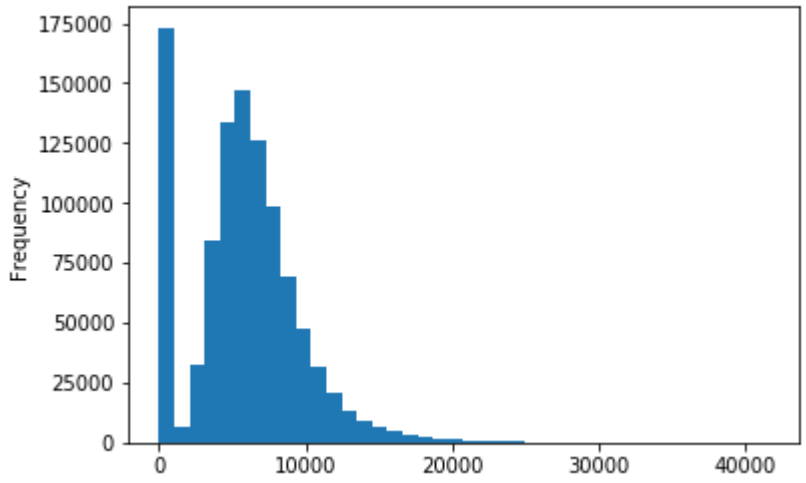
```
#train中没有缺失值
train[train.isnull().values==True]
```

Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday

```
#查看test中的缺失值
#622号商店从2015年9月5日至2015年9月17日的open字段缺失
test[test.isnull().values==True]
```

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
479	480	622	4	2015-09-17	NaN	1	0	0
1335	1336	622	3	2015-09-16	NaN	1	0	0
2191	2192	622	2	2015-09-15	NaN	1	0	0
3047	3048	622	1	2015-09-14	NaN	1	0	0
4759	4760	622	6	2015-09-12	NaN	0	0	0
5615	5616	622	5	2015-09-11	NaN	0	0	0
6471	6472	622	4	2015-09-10	NaN	0	0	0
7327	7328	622	3	2015-09-09	NaN	0	0	0
8183	8184	622	2	2015-09-08	NaN	0	0	0
9039	9040	622	1	2015-09-07	NaN	0	0	0
10751	10752	622	6	2015-09-05	NaN	0	0	0

随后我们观察 train.csv 和 test.csv 中的缺失值情况，发现训练集中没有缺失值而测试集中的 622 号店铺有缺失值，后续需要进一步关注。



观察训练集中 Sales 分布，发现除了销量为 0 的店铺之外，整体呈现右偏分布。

```
print(len(train[train['Sales']>20000]))
print(len(train[train['Sales']>35000]))
print(len(train[train['Sales']>40000]))
```

```
4099
18
1
```

进一步观察，发现训练集中只有 19 家店铺的营业额大于 35000，可以将其归类为极值点，后续在特征工程中进行剔除。

```
len(train[(train['Sales']==0) & (train['Open']==1)])
```

```
54
```

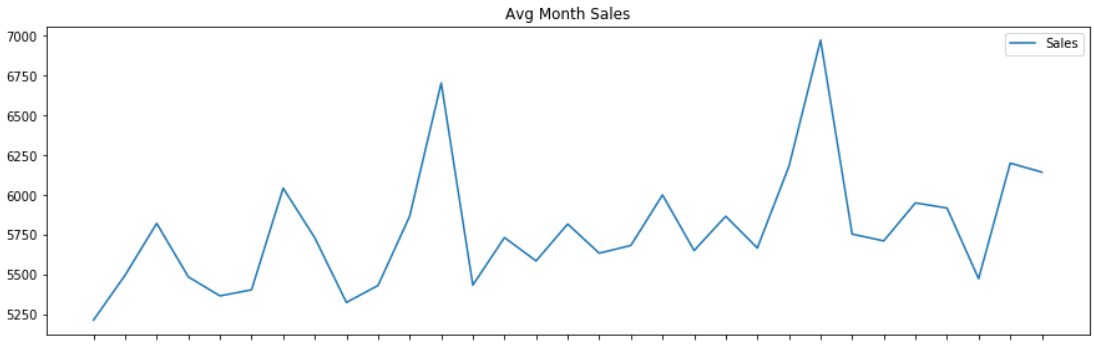
有54家店在某一天开门但是没有营业额

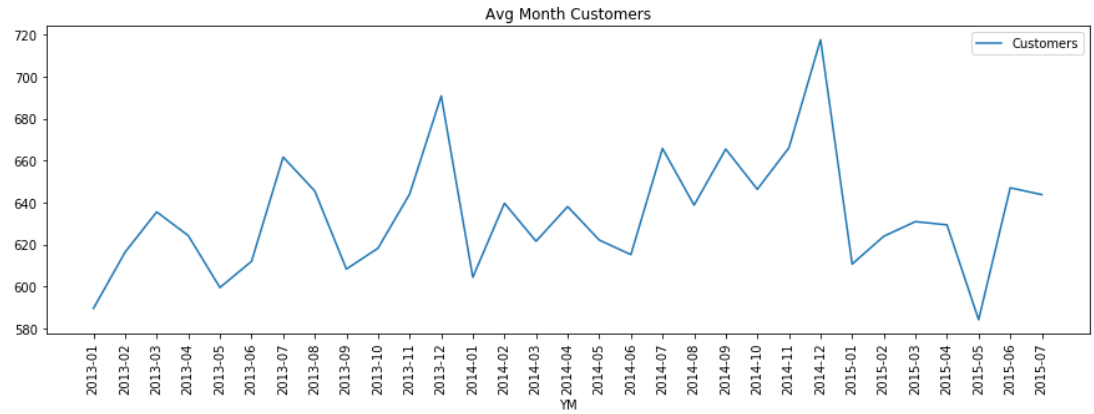
此外，还有 54 家店在某一天开门营业但是没有营业额，也属于异常值。

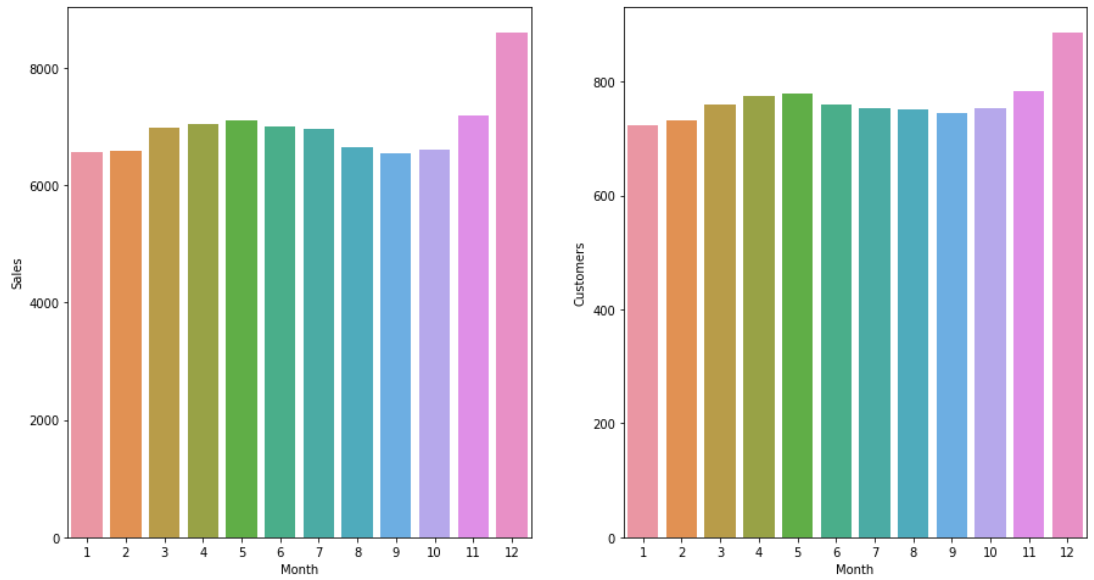
2.2 数据分析

我们剔除训练集中不营业的记录（Open 字段为 0），进行接下来的数据分析。

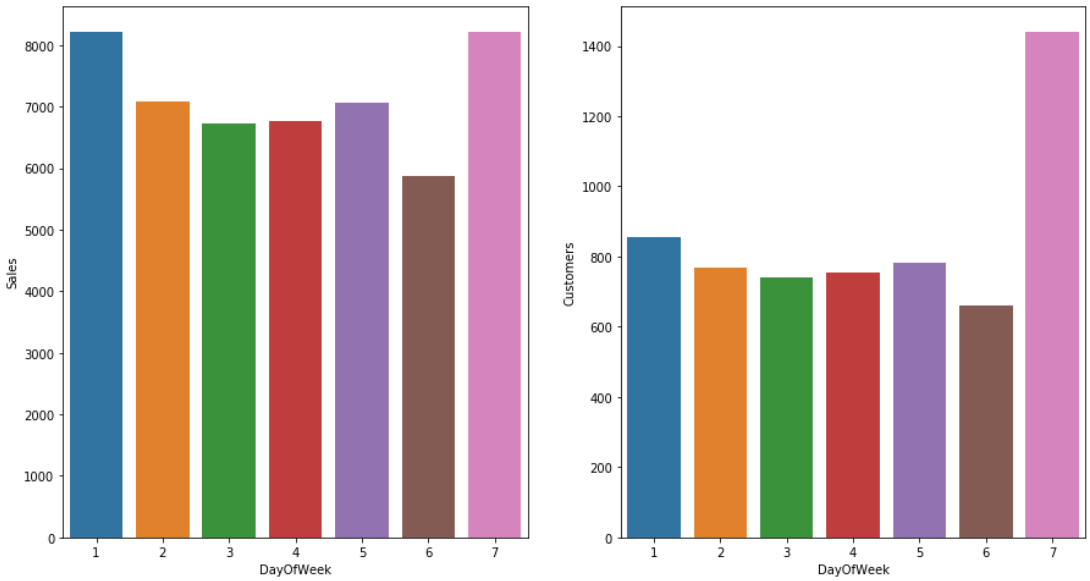
2.2.1 营业额 Sales 和顾客数量 Customers 与时间的关系







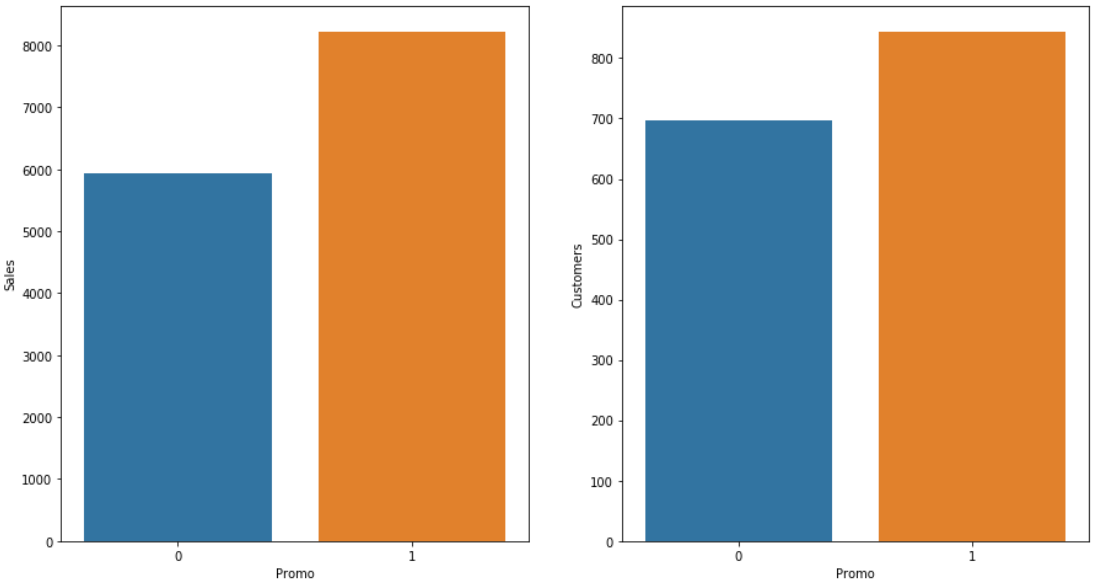
从上两张图我们发现每月平均营业额与每月平均顾客数量基本成正相关，且都在年底（11 月，12 月）达到阶段性高峰。

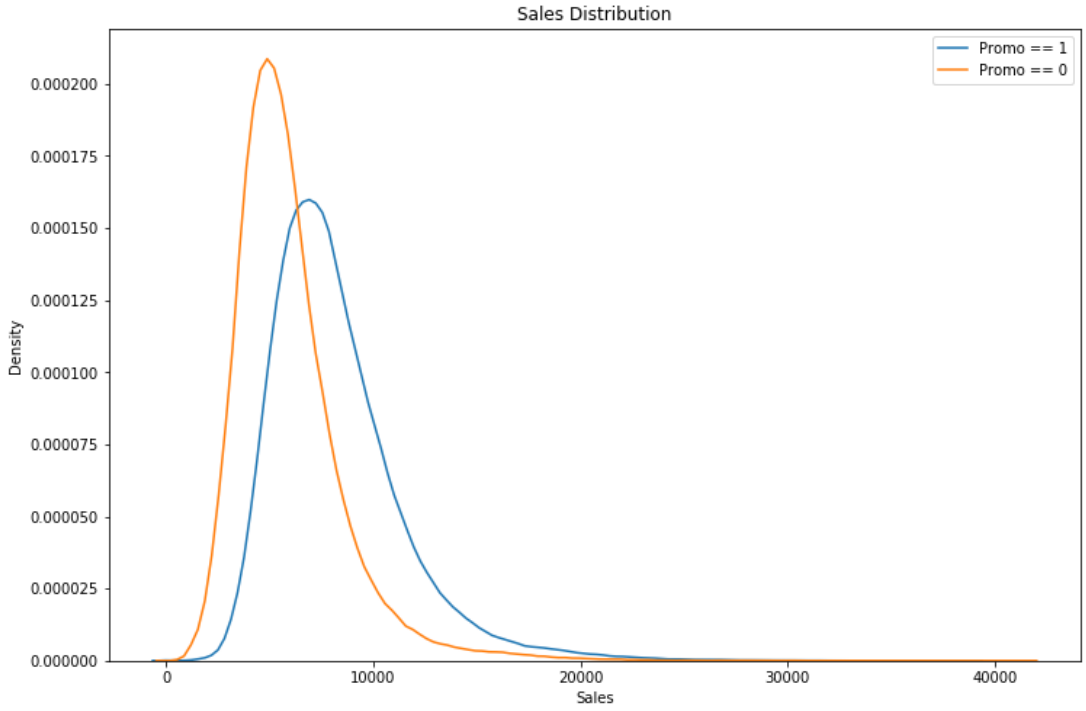


由上图可知，若我们将周日不营业的店铺剔除，则每周日的平均顾客数和平均营业额最高，和生活常识保持基本一致。若将周日不营业的因素考虑在内，那么周一的会比其他几天更高一些。

2.2.2 营业额 Sales 和顾客数量 Customers 与 Promo 的关系

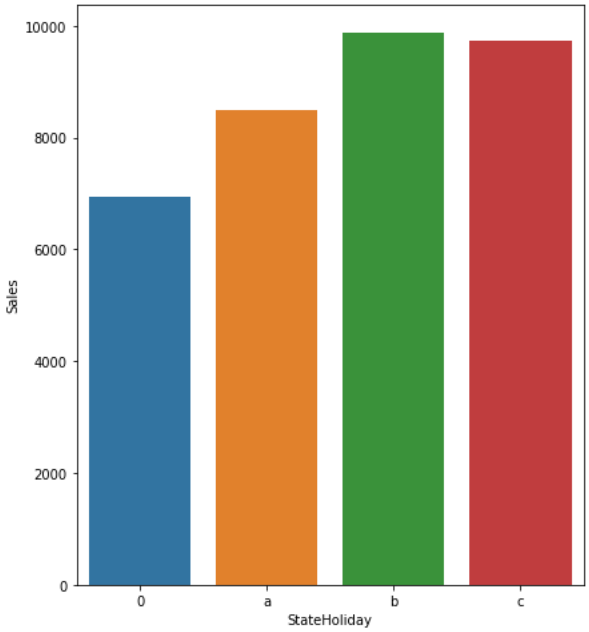
我们接下来观察营销活动对该日营业额和顾客数的影响。

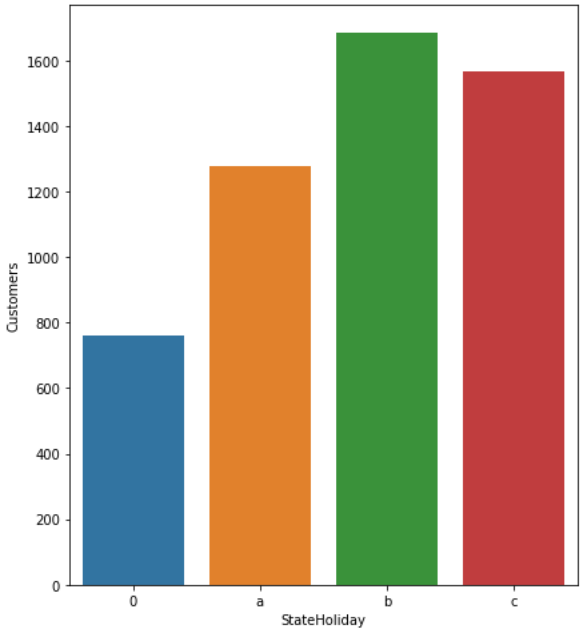


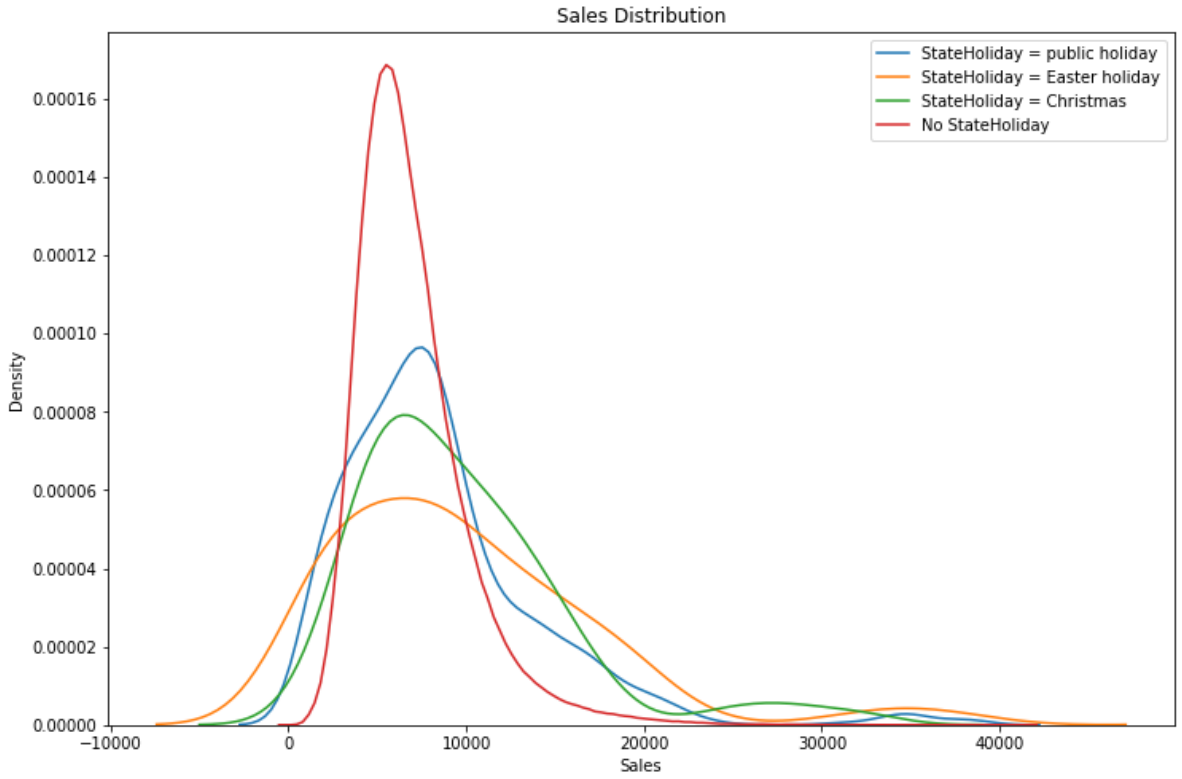


从均值来看，有营销活动对营业额和顾客数有正向促进效果。密度分布图来也印证了这一结论。

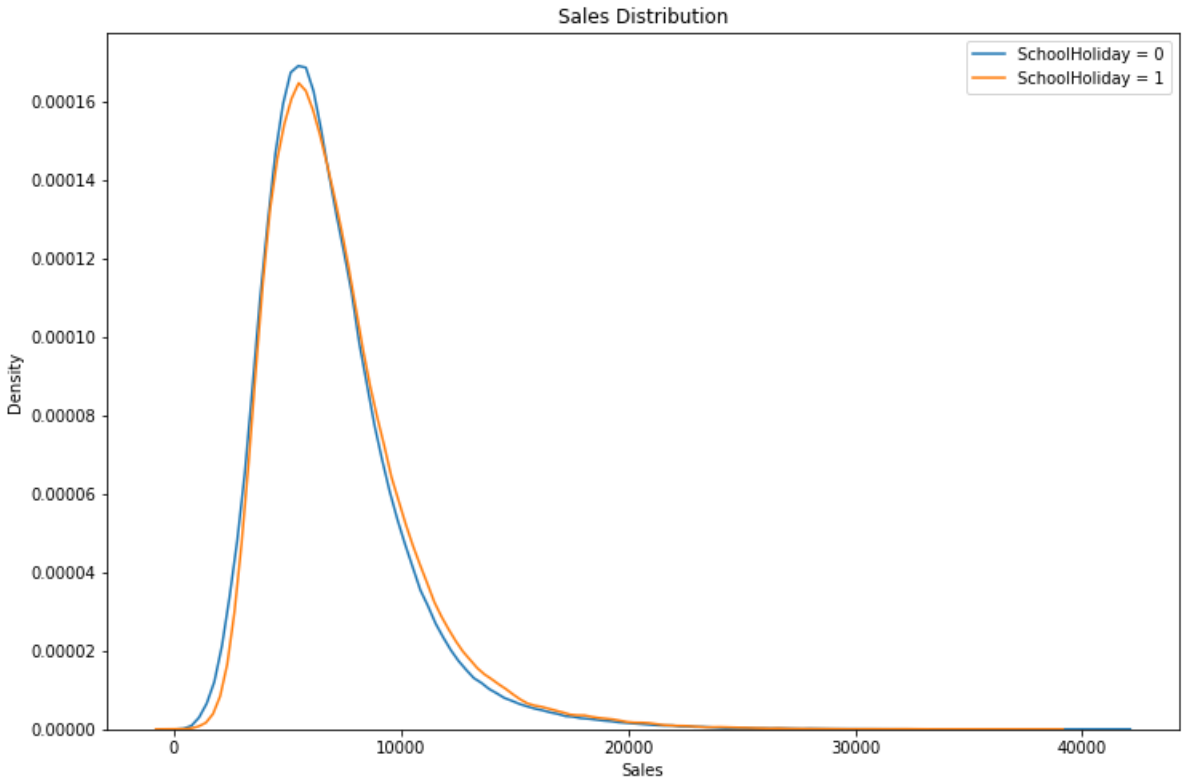
2.2.3 营业额 Sales 和顾客数量 Customers 与节假日的关系





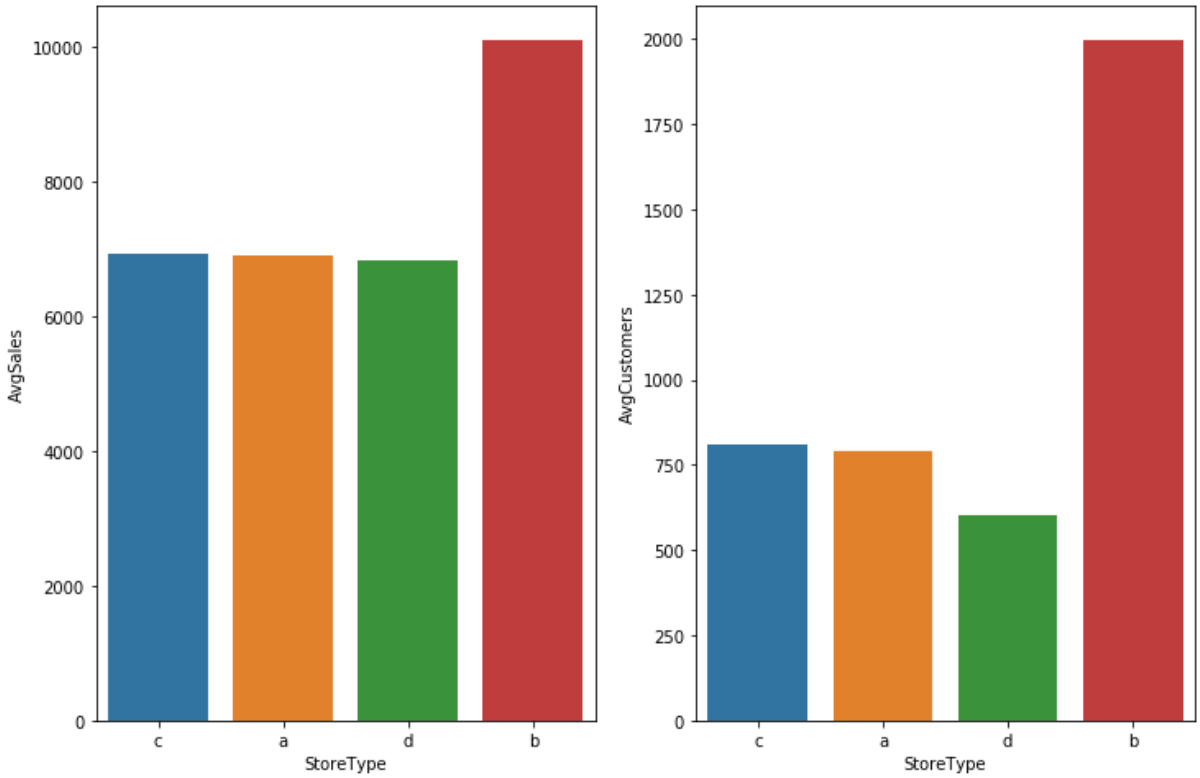


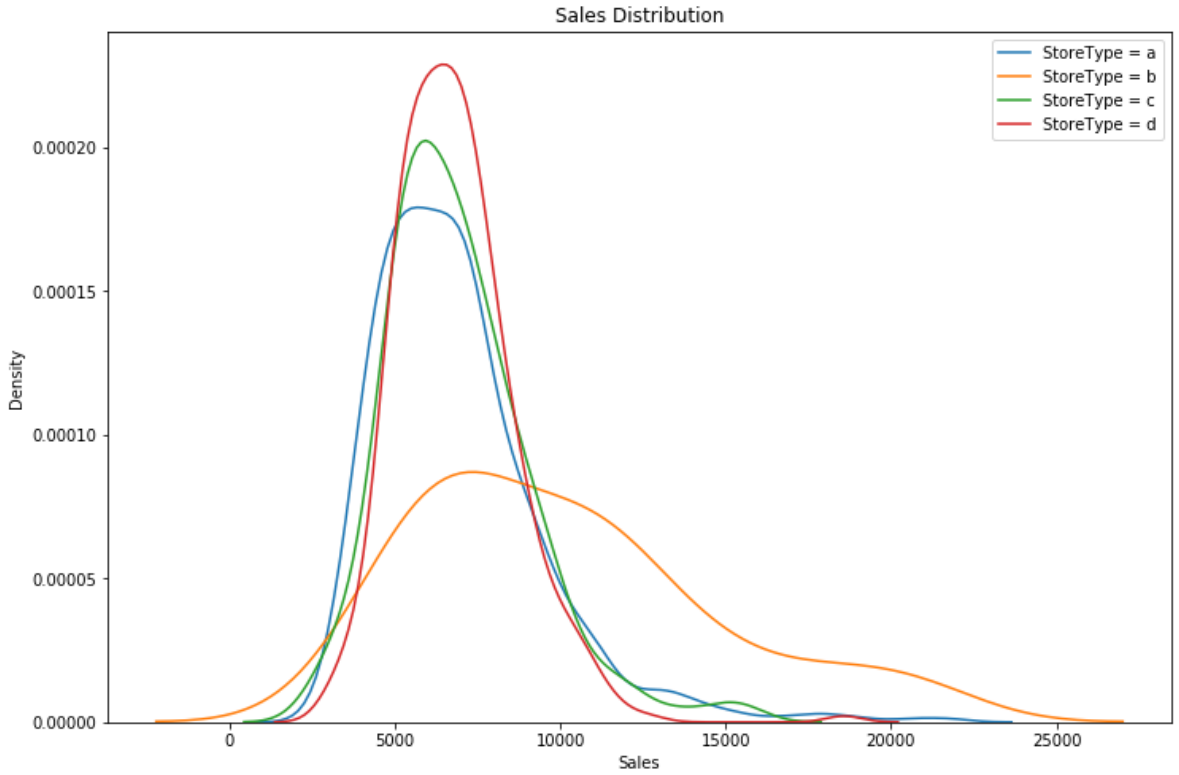
从上面两张图中我们不难发现，政府节假日对营业额和顾客数有正向影响，且各个类型的政府节假日对营业额的影响不尽相同。



然而，从上面的学校假期分布密度图来看，是否为学校假期并不显著影响商店营业额的分布。

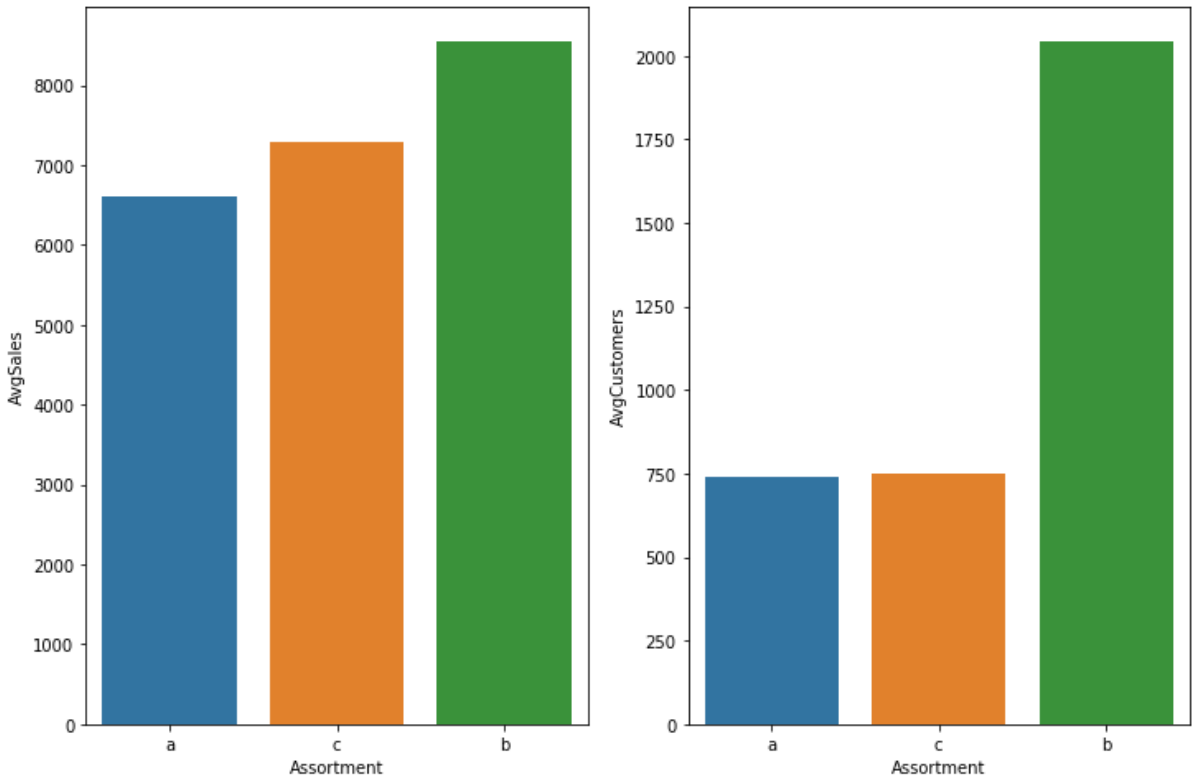
2.2.4 营业额 Sales 和顾客数量 Customers 与 StoreType 的关系

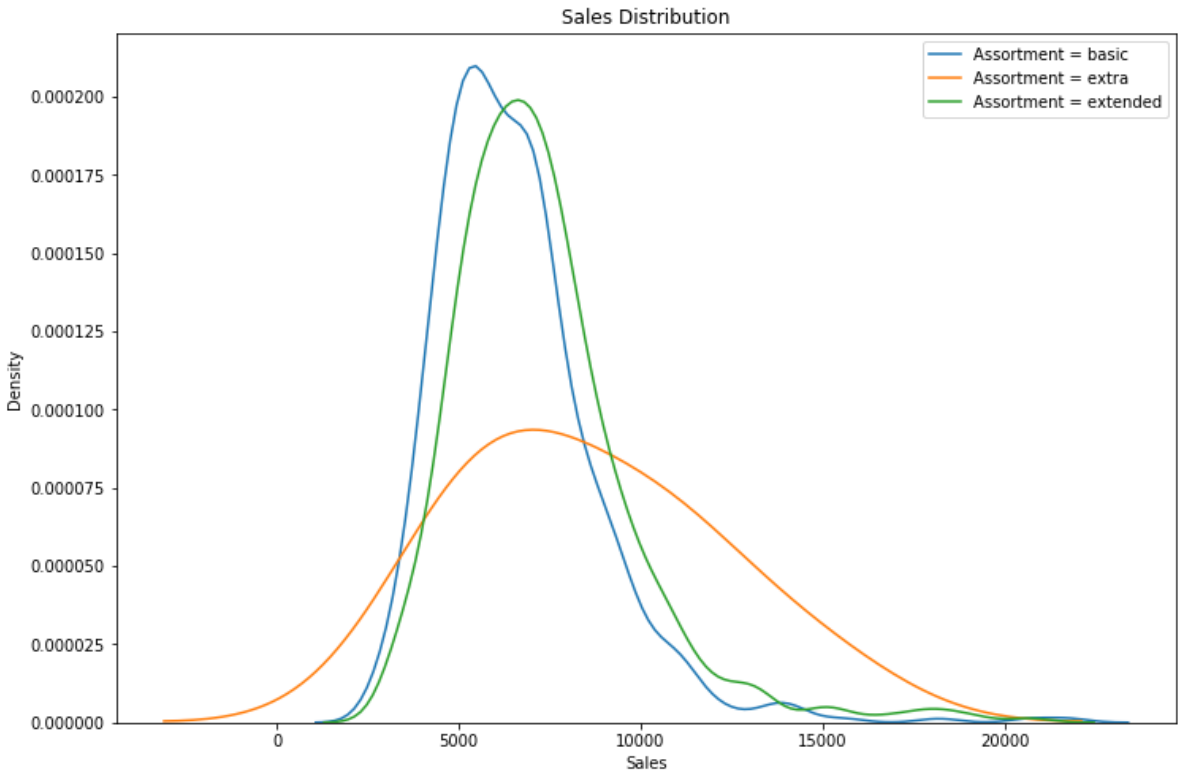




b 类商店的平均营业额和平均顾客数显著高于其他三个类型的店铺。

2.2.5 营业额 Sales 和顾客数量 Customers 与 StoreType 的关系





从上两张图中我们发现商店经营策略为 extra 的店铺在吸引顾客和扩大营业额方面更有优势。

2.3 算法与方法

此次项目通过 XGBoost 算法来实现。xgboost 是属于 boosting 模型大类 GBDT 模型中的一种。GBDT 和 xgboost 在竞赛和工业界使用都非常频繁，能有效的应用到分类、回归、排序问题。

GBDT 是以决策树（CART）为基学习器的 GB 算法，GBDT(Gradient Boosting Decision Tree) 又名 MART (Multiple Additive Regression Tree)，是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终答案。不同于 Adaboost 算法利用前一轮迭代弱学习器的误差率来更新训练集的权重，GBDT 用损失函数的负梯度来拟合本轮损失的近似值，进而拟合一个 CART 回归树。其原理是，假设前一轮迭代得到的强学习器是 $f_{t-1}(x)$，损失函数是 $L(y,f_{t-1}(x))$，我们本轮迭代的目标是找到一个 CART 回归树模型的弱学习器 $h_t(x)$，让本轮的损失函数 $L(y,f_t(x)=L(y,f_{t-1}(x)+h_t(x))$最小。也就是说，本轮迭代找到决策树，要让样本的损失尽量变得更小。其主要流程可表示为：

（1）假设输入样本 $T=\{(x_1,y_1),(x_2,y_2),...(x_m,y_m)\}$，最大迭代次数为 T，损失函数为 L，初始化弱学习器：

$$f_0(x) = \underbrace{arg\ min}_{c} \sum_{i=1}^{m} L(y_i, c)$$

（2）对每一次迭代，计算负梯度：

$$r_{ti} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}\ \ (x)}$$

（3）拟合基学习器的伪残差，其对应的叶子节点区域为 $R_{tj,j}=1,2,...,J$，其中 J 为基模型叶子节点个数。

（4）计算最佳拟合值：

$$c_{tj} = \underbrace{arg\ min}_{c} \sum_{x_i \in R_{tj}} L(y_i, f_{t-1}(x_i) + c)$$

（5）更新强学习器：

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{J} c_{tj} I(x \in R_{tj})$$

$$f(x) = f_T(x) = f_0(x) + \sum_{t=1}^{T} \sum_{j=1}^{J} c_{tj} I(x \in R_{tj})$$

XGBoost 是 Gradient boosting 算法的高效实现，它扩展和改进了 GDBT，该算法更快，准确率也相对高一些。具体来说，两者的区别主要有这么几点：

（1）XGBoost 对于损失函数加入了正则化项，用于控制模型的复杂度，而普通 GBDT 并没有：

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

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

T 为叶子节点的数量，W 为叶子的权重。
Y 帽子 为预测值，Y 为目标值。
gamma ,delta 为参数

（2）此外，XGBoost 损失函数是误差部分是二阶泰勒展开，GBDT 是一阶泰勒展开。所以损失函数定义的更精确。

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$$\text{where } g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \text{ and } h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

（3）XGBoost 借鉴了随机森林的做法，支持列抽样（col_sampling），不仅能降低过拟合，还能减少计算，这也是 xgboost 异于传统 gbdt 的一个特性。

（4）XGBoost 对每颗子树增加一个参数（Shrinkage），使得每颗子树的权重降低，防止过拟合。在进行完一次迭代后，XGBoost 会将叶子节点的权重乘上该系数，削弱了每棵树的影响，让后面有更大的学习空间。

（5）对于数据集中的缺失值，XGBoost 算法可以自动学习出分裂方向。

2.4 基准模型

根据项目要求，设定基准阈值为 Kaggle Private Leaderboard 中前 10%，也就是分数要低于 0.11773

3. 方法

3.1 数据预处理

3.1.1 将 StateHoliday 中的字符 0 转化为数字 0

通过观察 Python 代码运行结果，我们发现 StateHoliday 中有两个相同的“0”，为了统一后续运算，我们将两种“0”统一化为字符类型的“0”

train['StateHoliday']=train['StateHoliday'].astype(str)

```
test['StateHoliday']=test['StateHoliday'].astype(str)
```

3.1.2 剔除开门营业但是没有营业额的数据

在本次报告的数据分析部门，我们就发现有 54 条记录显示为开门营业但是没有营业额，我们将这些记录从训练集中剔除

```
abindex_1=train[(train['Sales']==0) & (train['Open']==1)].index
train=train.drop(abindex_1)
```

3.1.3 删除 Sales 的极值点

```
train=train[train['Sales']<=35000]
```

3.1.4 用最大值填充 Store 表中 CompetitionDistance 的空值

```
store['CompetitionDistance']=store['CompetitionDistance'].fillna(max(store['CompetitionDistance']))
```

3.1.5 用最早的时间来填充 CompetitionOpenSinceMonth 和 CompetitionOpenSinceYear

```
store['CompetitionOpenSinceMonth']=store['CompetitionOpenSinceMonth'].fillna(1.0)
store['CompetitionOpenSinceYear']=store['CompetitionOpenSinceYear'].fillna(1900.0)
```

3.2 特征工程

3.2.1 营业额和顾客数特征

利用 Python 的 groupby 函数，生成：

每个商店平均销量 AvgSales

每个商店平均销量 MediSales

每个商店平均顾客数 AvgCustomers

每个商店顾客的中位数 MediCustomers

每个商店平均销量/每个商店平均顾客数 AvgSalesPerCustomer

每个门店每周一到每周日的平均销售额和销售额的中位数 AvgSales_DayX, MediSales_DayX

每个门店每周一到每周日的平均顾客数和顾客数的中位数 AvgCustomers_DayX, MediCustomers_DayX

同样地，根据各家商店所属营业的月份的营业额和客户量，来计算出营业的商店每月的中位数销售额、平均客户量以及中位数客户量。

此外，我还计算了每个商店的平均 sales 和 customers 以及（sales/customers）偏离自己这一类型（assortment，storetype，promo2,interval）均值的偏离量（deviation）。

3.2.2 营销和竞争特征

新增一个字段 IsPromoMonth，如果商店运营月份在 promo2 的时间段内，则为 1

新增字段 IsPromoMonth 表明该商店该日是否在 promo2 的月份

新增字段 DeltaCompetition 和 DeltaPromo2Weeks，分别表示当前日期与竞争开始日期与 Promo2 开始日期的差额

新增字段 AbsDeltaCompetition，是 DeltaCompetition 字段的绝对值

新增两个字段，分别为“BeforeCompetition”和“AfterCompetition”，用 apply 函数填充。

开业的前后几天，对销量的影响最明显。时间离的越远，则影响越小。

Before Competition	After Competition
半年之前，值为 1	0
一个月之前，值为 10	0
一周之前，值为 100	0
一周之内，值为 1000	0
0	一周之内，值为 1000
0	一个月之内，值为 100
0	半年之内，值为 10
0	超过半年，值为 1

```
def inmonthlist(a,b):
    return (a in b)

data_all_store['IsPromo2Month']=data_all_store.apply(lambda x: 1 if
((inmonthlist(x['Month'],x['Promo2MonthList'])) and (x['DeltaCompetition']>0)) else 0, axis=1)

data_all_store['BeforeCompetition']=data_all_store.apply(lambda x: 1 if int(x['DeltaCompetition'])<-12
else (10 if int(x['DeltaCompetition'])<-6 else (100 if int(x['DeltaCompetition'])<-3 else (1000 if
int(x['DeltaCompetition'])<0 else 0))), axis=1)

data_all_store['AfterCompetition']=data_all_store.apply(lambda x: 1 if int(x['DeltaCompetition'])>12
else (10 if int(x['DeltaCompetition'])>6 else (100 if int(x['DeltaCompetition'])>3 else (1000 if
int(x['DeltaCompetition'])>0 else 0))), axis=1)
```

3.2.3 假日特征处理

根据各家商店的营业年份，所属年份的周数来计算出本周学校的假期数、下周假期数以及上周

假期数，分别创建了变量 HolidayThisWeek，HolidayNextWeek，HolidayLastWeek

同样根据各家商店的营业年份，所属年份的月数来计算出本月的法定节日假期数、下月法定节假日期数以及上月法定节假日假期书，分别创建了变量 StateHolidayNextMonth，StateHolidayThisMonth，StateHolidayLastMonth

3.2.4 分类编码

考虑到 assortment 有实际排序意义，我们根据实际意义对其赋予 1,2,3 的值进行编码

```
dict_assortment={'a':1,'b':2,"c":3}
data_all_store['AssortmentEncoding']=data_all_store['Assortment'].map(dict_assortment)
```

对剩余的分类变量（DayOfWeek, StateHoliday，StoreType，PromoInterval 进行独热编码

此外，针对 Date 字段单独提取月份，年份，日期。

3.2.5 数值转换

最后对 sales 进行对数处理，对 test 预测出来的数据进行 exp 还原

3.2算法实施

作为尝试，我们将全部特征加入 XGBoost 模型，并将训练集中最后 6 周的记录作为验证集，其余的记录作为训练集进行数据分割。

该模型参数最初被设定为：

```
params = {"objective": "reg:linear",
          "booster": "gbtree",
          "eta": 0.01,
          "max_depth": 12,
          "min_child_weight": 6,
          "subsample": 0.7,
          "colsample_bytree": 0.7,
          "silent": 1,
          #"lambda": 0.1,
          "seed": 523}

num_boost_round = 15000, early_stopping_rounds=200
```

```
[4000]  train-rmse:0.061595  eval-rmse:0.107256  train-rmspe:0.070735  eval-rmspe:0.108671
[4050]  train-rmse:0.061421  eval-rmse:0.107249  train-rmspe:0.070415  eval-rmspe:0.108666
[4100]  train-rmse:0.061208  eval-rmse:0.10725   train-rmspe:0.070045  eval-rmspe:0.10867
[4150]  train-rmse:0.060999  eval-rmse:0.107243  train-rmspe:0.069624  eval-rmspe:0.108666
[4200]  train-rmse:0.060807  eval-rmse:0.107236  train-rmspe:0.069172  eval-rmspe:0.10866
[4250]  train-rmse:0.060616  eval-rmse:0.10724   train-rmspe:0.068798  eval-rmspe:0.108668
```

[4300] train-rmse:0.060433 eval-rmse:0.107249 train-rmspe:0.068489 eval-rmspe:0.108673
Stopping. Best iteration:
[4139] train-rmse:0.061051 eval-rmse:0.107237 train-rmspe:0.069787 eval-rmspe:0.108657

提交到 Kaggle 上得到如下分数

Private Score	Public Score
0.11990	0.11284

3.3 改进与完善

初次尝试，模型的最后得分并没有达到我的理想分数。

经过分析，我认为可能有几方面的原因：

1）特征过多造成过拟合，比如我在特征工程部分既构造了营业额的均值，又构造了营业额的中位数，这两类变量的相关性很高，会造成潜在过拟合，需要舍弃一些相关性高的变量后再次尝试训练

2）模型参数中的相关参数（max_depth, subsample, colsample_bytree）设置得不合理，导致潜在的过拟合，需要修改参数后再次尝试训练。

最后考虑到均值已受极值的影响，我决定保留中位数（Medi）相关的变量，舍弃了均值(Avg)相关的变量。此外，我把模型中 max_depth 参数由 12 调整为 10，subsample 参数调整为 0.65，colsample_bytree 参数调整为 0.6，并把最后的预测值进行一个 0.965 的缩放。最终的得分有 0.002 左右的提高。

4. 结果

4.1 模型评价与验证

All Successful Selected

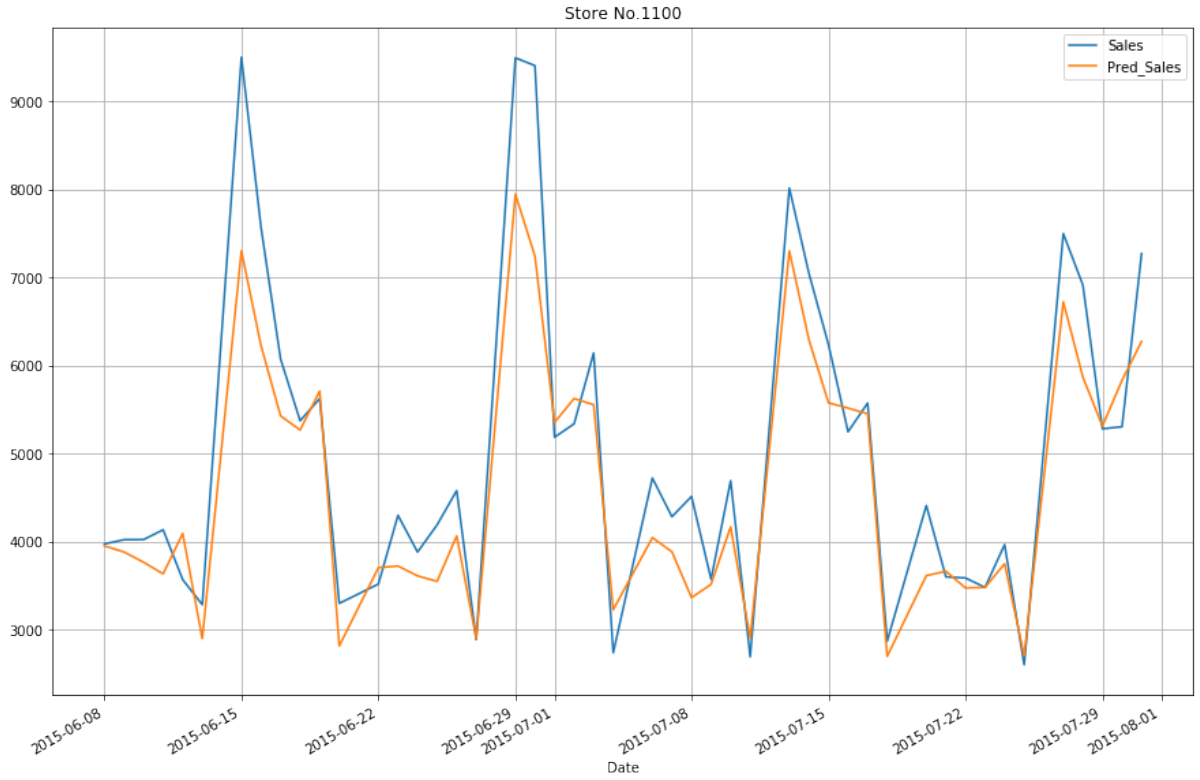
Submission and Description	Private Score	Public Score
submission_ensemble_11.csv a day ago by Felix Chen add submission details	0.11709	0.11072
submission_ensemble_14.csv 19 hours ago by Felix Chen add submission details	0.11715	0.11095
submission_ensemble_13.csv 19 hours ago by Felix Chen add submission details	0.11721	0.11121
submission_ensemble_10.csv 2 days ago by Felix Chen add submission details	0.11727	0.11102
submission_ensemble_9.csv 2 days ago by Felix Chen add submission details	0.11730	0.11089
submission_ensemble_12.csv 19 hours ago by Felix Chen add submission details	0.11740	0.11139
submission_ensemble_7.csv 2 days ago by Felix Chen add submission details	0.11773	0.11150
submission_ensemble_5.csv 2 days ago by Felix Chen add submission details	0.11774	0.11155
submission_partial_para_7.csv 2 days ago by Felix Chen 10,0.65,0.6	0.11775	0.11171

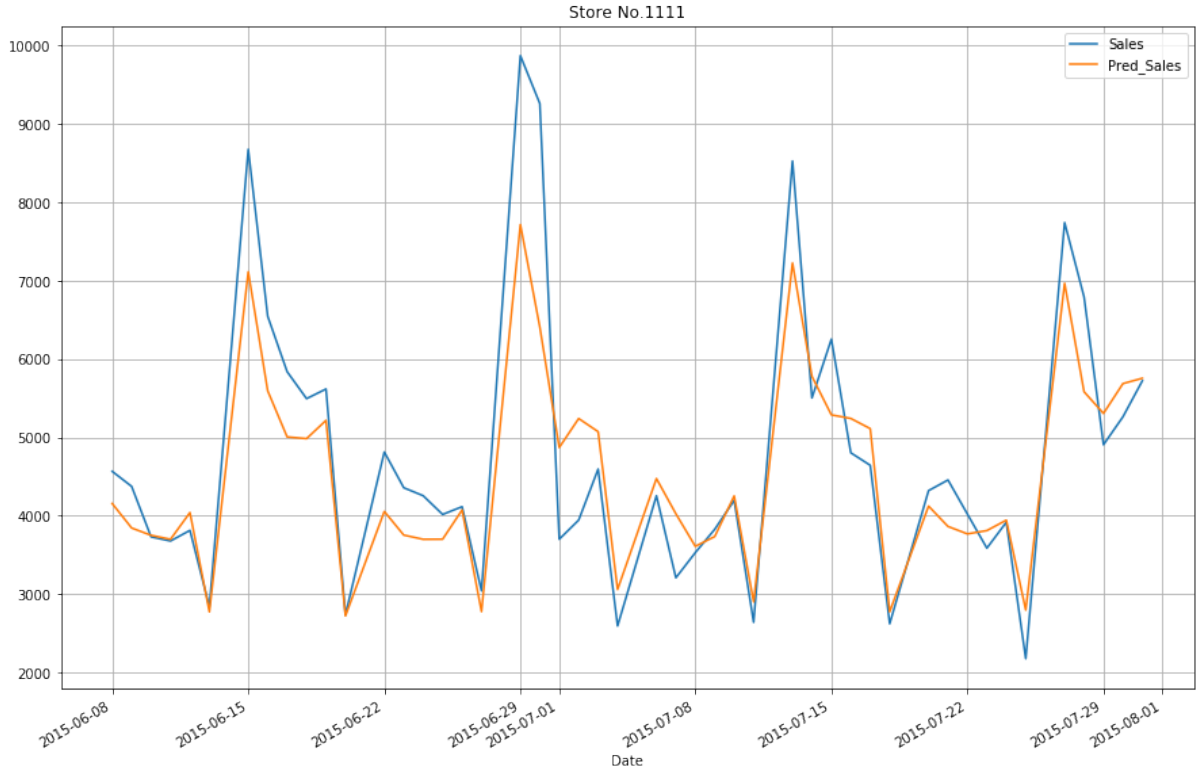
最终单个模型的预测值取得了 0.11775 左右的得分，我又微调了一些参数，多跑了几次模型，并将这些结果进行融合，取得了额外约 0.0007 左右的得分提升。

最终的得分为：

Private Score	Public Score
0.11709	0.11072

该分数达到了在 Private Leaderboard 上排名前 10%的基准线，排名为（272/3303）。

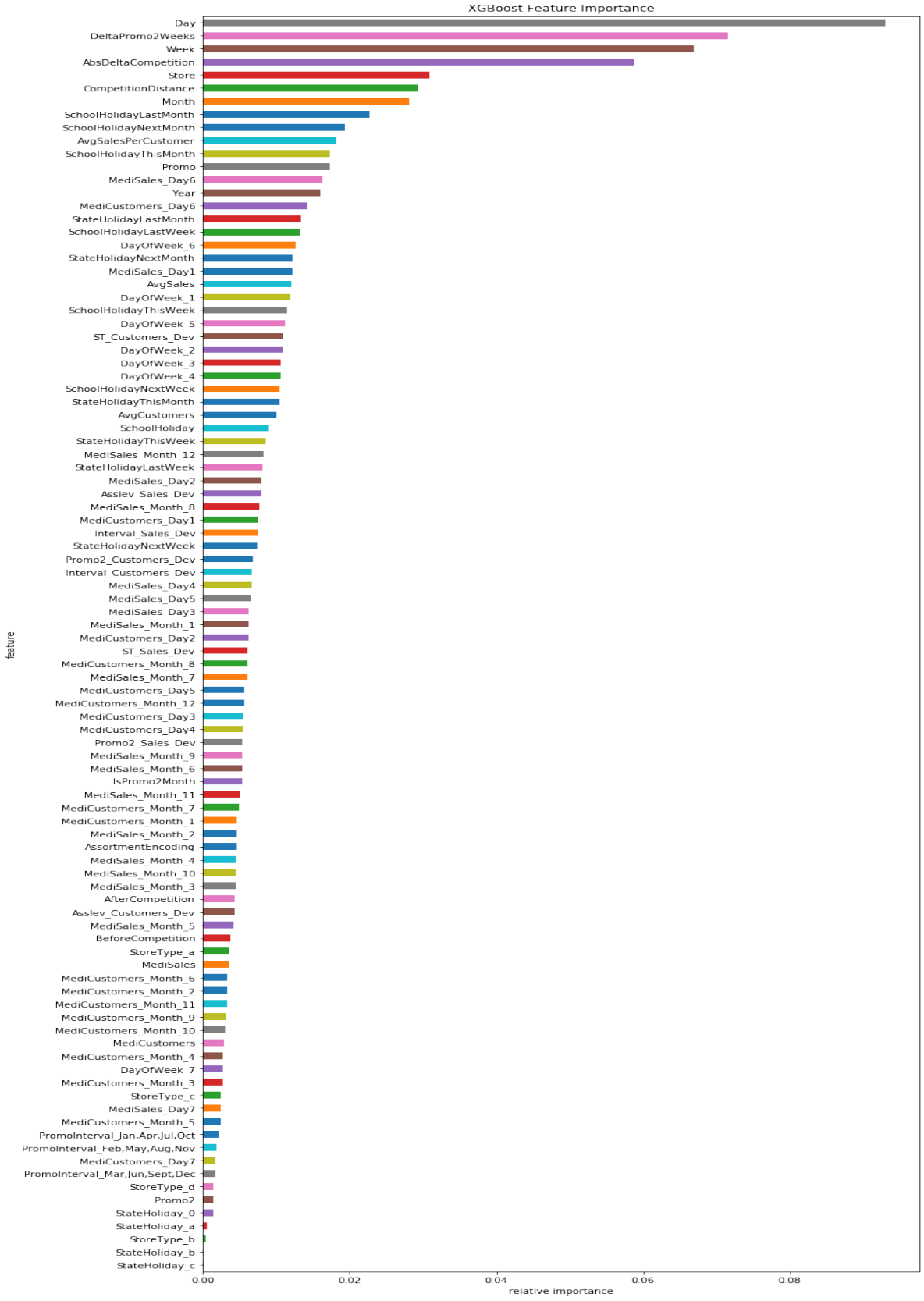




我们分别选取第 1100 号店铺和 1111 号店铺，观察模型的预测值和真实值在验证集上的表现情况。从上面两张图中我们发现总体来说预测值和真实值的走势基本一致，其大小范围相比于真实值来说要小一些。

5. 项目结论

5.1 可视化


XGBoost Feature Importance
Day
DeltaPromo2Weeks
Week
AbsDeltaCompetition
Store
CompetitionDistance
Month
SchoolHolidayLastMonth
SchoolHolidayNextMonth
AvgSalesPerCustomer
SchoolHolidayThisMonth
Promo
MediSales_Day6
Year
MediCustomers_Day6
StateHolidayLastMonth
SchoolHolidayLastWeek
DayOfWeek_6
StateHolidayNextMonth
MediSales_Day1
AvgSales
DayOfWeek_1
SchoolHolidayThisWeek
DayOfWeek_5
ST_Customers_Dev
DayOfWeek_2
DayOfWeek_3
DayOfWeek_4
SchoolHolidayNextWeek
StateHolidayThisMonth
AvgCustomers
SchoolHoliday
StateHolidayThisWeek
MediSales_Month_12
StateHolidayLastWeek
MediSales_Day2
Asslev_Sales_Dev
MediSales_Month_8
MediCustomers_Day1
Interval_Sales_Dev
StateHolidayNextWeek
Promo2_Customers_Dev
Interval_Customers_Dev
MediSales_Day4
MediSales_Day5
MediSales_Day3
MediSales_Month_1
MediCustomers_Day2
ST_Sales_Dev
MediCustomers_Month_8
MediSales_Month_7
MediCustomers_Day5
MediCustomers_Month_12
MediCustomers_Day3
MediCustomers_Day4
Promo2_Sales_Dev
MediSales_Month_9
MediSales_Month_6
IsPromo2Month
MediSales_Month_11
MediCustomers_Month_7
MediCustomers_Month_1
MediSales_Month_2
AssortmentEncoding
MediSales_Month_4
MediSales_Month_10
MediSales_Month_3
AfterCompetition
Asslev_Customers_Dev
MediSales_Month_5
BeforeCompetition
StoreType_a
MediSales
MediCustomers_Month_6
MediCustomers_Month_2
MediCustomers_Month_11
MediCustomers_Month_9
MediCustomers_Month_10
MediCustomers
MediCustomers_Month_4
DayOfWeek_7
MediCustomers_Month_3
StoreType_c
MediSales_Day7
MediCustomers_Month_5
PromoInterval_Jan,Apr,Jul,Oct
PromoInterval_Feb,May,Aug,Nov
MediCustomers_Day7
PromoInterval_Mar,Jun,Sept,Dec
StoreType_d
Promo2
StateHoliday_0
StateHoliday_a
StoreType_b
StateHoliday_b
StateHoliday_c
feature
0.00
0.02
0.04
0.06
0.08
relative importance

从模型给出的特征重要性来看，日期，距离 Promo2 开始的时间差，周数，距离竞争店铺开业的时间差，商店编号，竞争店铺距离，月份等特征重要性较高，而政府节假日，商店种类，促销区间等特征的重要性较低。

```
df.sort_values(by='fscore',ascending=False)[:10]
```

	feature	fscore
95	Day	0.093005
94	DeltaPromo2Weeks	0.071532
93	Week	0.066891
92	AbsDeltaCompetition	0.058730
91	Store	0.030855
90	CompetitionDistance	0.029294
89	Month	0.028043
88	SchoolHolidayLastMonth	0.022633
87	SchoolHolidayNextMonth	0.019308
86	AvgSalesPerCustomer	0.018247

```
df.sort_values(by='fscore',ascending=True)[:10]
```

	feature	fscore
0	StateHoliday_c	0.000076
1	StateHoliday_b	0.000127
2	StoreType_b	0.000412
3	StateHoliday_a	0.000558
4	StateHoliday_0	0.001346
5	Promo2	0.001464
6	StoreType_d	0.001471
7	PromoInterval_Mar,Jun,Sept,Dec	0.001680
8	MediCustomers_Day7	0.001759
9	PromoInterval_Feb,May,Aug,Nov	0.001821

5.2 思考

在整个项目中，最花时间的部分是探索性数据分析（EDA）和特征工程（feature engineering）。

我根据自己的商业理解和 EDA 的结果构造了许多特征，数量较多也比较花时间，从训练的结果和 Kaggle 排行榜上的两个分数来看，多多少少有一些过拟合的问题。营业额预测与时间序列有一定的联系，可以考虑后续通过 ARMA 模型对营业额和顾客数构造出一些新的预测值作为新的特征，以提高模型的准确性。

此外，XGBoost 模型使用的学习速率为 0.01，导致学习训练的速度比较慢，初次尝试训练便花了 6 到 7 个小时，模型的运行效率比较低。

5.3 改进

现有数据集中的数据大部分均与时间相关，与空间相关的比较少，可以考虑加入与地理位置相关的外部特征数据，比如每个地区的天气类型（后期进行独热编码），降雨量，温度等特征。

尝试 lightgbm, random forest regressor 等不同的回归模型（boosting 和 bagging），将不同算法跑出的结果进行融合，或许可以提高最后的得分。

学习并尝试深度学习 Entity-Embedding 模型进行预测。Guo Cheng 在 Kaggle 的 Rossman Store Sales 竞赛中运用该方法以 0.10583 的成绩取得了第三名。

使用相关无监督学习技术，比如 PCA 等对特征进行降维，以缓解潜在的过拟合问题。

可以对模型的预测值进行最优系数调整探索，进行最优系数调整后的预测值可能会有超越原预测值的得分。

参考文献：

1. https://www.kaggle.com/c/rossmann-store-sales
2. https://blog.csdn.net/owenfy/article/details/79631144
3. https://zh.wikipedia.org/wiki/XGBoost
4. https://xgboost.readthedocs.io/en/latest/
5. http://www.cnblogs.com/wxquare/p/5541414.html
6. https://www.cnblogs.com/peizhe123/p/5086128.html
7. https://github.com/entron/entity-embedding-rossmann/tree/kaggle
8. http://blog.kaggle.com/2015/12/21/rossmann-store-sales-winners-interview-1st-place-gert/
9. https://www.cnblogs.com/pinard/p/6140514.html
10. https://www.kaggle.com/xwxw2929/rossmann-sales-top1/notebook
11. 机器学习导论，周志华，清华大学出版社
12. 统计学习方法，李航，清华大学出版社