Udacity - 毕业项目报告

报告作者：蔡久兵

报告时间：2018-11-25

Rossmann销售预测开题报告

一、 定义

1.1 项目概览

> 学生从旁观者的角度概述项目。已提供的背景信息包括问题领域、项目源和相关数据集或输入数据。

Rossmann 是欧洲的一家连锁药店。在欧洲7个国家经营着3000多家药店。目前，Rossmann门店经理的任务是提前六 周预测他们的日常销量。商店的销售受到许多因素的影响，包括促销、竞争、学校和国家假日、季节性和所在地区。成千上万的门店经理根据其独特偏好进行销售预测，造成结果的准确性可能会偏差很大。

可靠的销售预测使商店经理能够制定有效的员工时间表，提高生产力和积极性。通过帮助 Rossmann 创建一个健壮的预测模型，将会帮助商店经理专注于他们最重要的事情:他们的客户和他们的团队。

1.2 问题说明

> 明确定义需要解决的问题。已制定解决问题的策略，并讨论了预期解决方案。

在这个源自Kaggle比赛Rossmann Store Sales中，Rossmann希望参赛者能通过给出的数据来预测德国各地1,115家店铺的6周销售量，销售量是数值型因为我们需要解决的是**“回归问题”**，基于有标签的历史数据集预测结果，这里我们用**“监督学习算法”**来实现此项目；再根据评价指标，所得的预测值与实际值的差值越小，准确性就越好。

Rosmann 提供了"train.csv"和"stores.csv"两个包括销售数据和店铺信息的数据集，同时还有一个"test.csv"的数据集，字段和"train.csv"一样，只是缺少需要提交的"Sales"字段。

项目采用的模型是 XGBoost、GBDT 和 RandomForest 通过最后模型效果选择最优模型来做最终模型，Kaggle 比赛中第一名采用的是 XGBoost 模型。

1.2 评价指标

> 明确定义了用于测量模型性能或结果的指标。指标根据问题特征进行了调整。

提交的内容是根据均方根百分比误差(RMSPE)计算的。RMSPE 被计算为

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

其中 yi 表示单个商店在一天内的销售额，$\hat{y_i}$ 表示相应的预测。任何一天和商店 0 的销售都被忽视得分。

RMSPE 对一组预测中的特大或特小误差反映非常敏感，因而它能够很好地反映出模型预测的精密度。对于项目中需要预测的销售量，采用 RMSPE 能够很好地表示模型的效果。

二、 分析

2.1 数据研究

> 如果数据集存在，则必须已报告与此问题相关的特征和经过计算的统计数据，并已对数据进行抽样。已对数据集中的输入空间或输入数据进行了充分说明。已经确认需要解决的数据或输入异常或者特征。
>
> 提供了可以总结或提取数据集或者输入数据相关特征的可视化图表，并进行了充分讨论。明确定义了可视化曲线。

Kaggle提供文件：

- train.csv -每家店销售数据集，用于训练
- store.csv -每家店信息数据集

- test.csv -跟"train.csv"结构一样，缺少需要提交的"Sales"字段，用于最终的测试
- sample_submission.csv -提交数据集的参考样本

数据集字段主要分为商店信息和商店销售数据两部分：

- #### 商店销售数据：####

    1. Store（商店 ID）
    2. DayOfWeek（日期所在的周几）
    3. Date（销售时间）
    4. Sales（销售额，目标真实值）
    5. Customers（消费者数量）
    6. Open（商店当日是否开放）
    7. Promo（商店当日是否促销）
    8. StateHoliday（国家节假日，a、b、c 是国家节假日，0 表示非节假日）
    9. SchoolHoliday（国家公立学校节假日，1：节日，0 非节日）
- #### 商店信息数据：####

    1. Store（商店 ID）
    2. StoreType（商店类型）
    3. Assortment（商店级别）
    4. CompetitionDistance（竞争对手距离）
    5. CompetitionOpenSinceMonth（竞争对手开店月份，部分缺失）
    6. CompetitionOpenSinceYear（竞争对手开店年份，部分缺失）
    7. Promo2（是否持续促销）

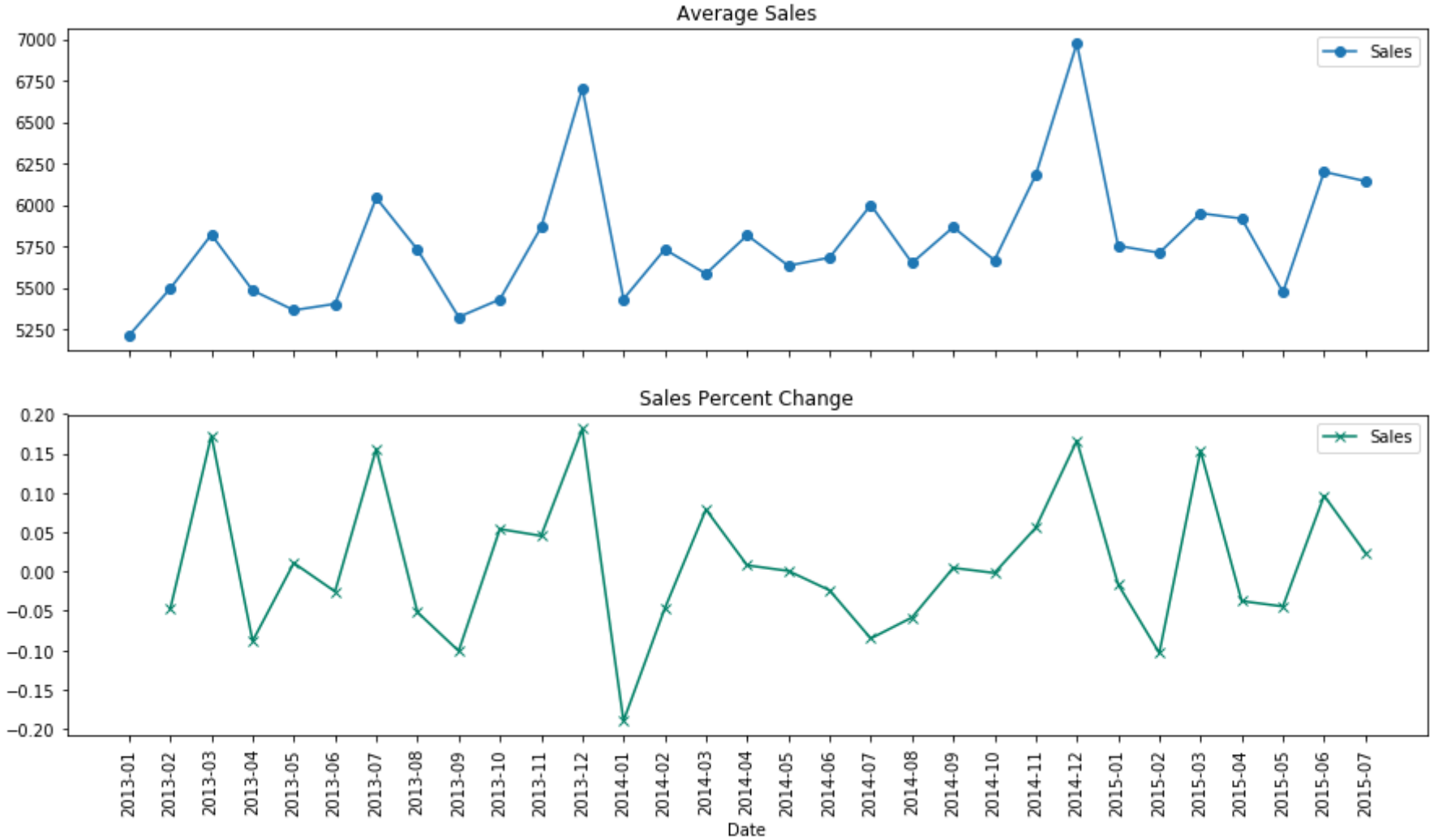
8. Promo2SinceWeek（持续促销的周）
9. Promo2SinceYear（持续促销的年）
10. PromoInterval（促销的月份）

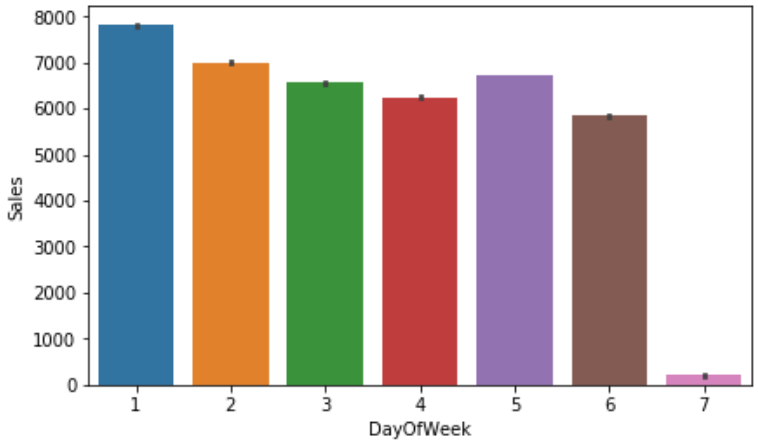
以上数据由kaggle提供，没有引入其他的数据集，因为这个问题比较特定并且有足够的数据，因此不需要额外的数据。

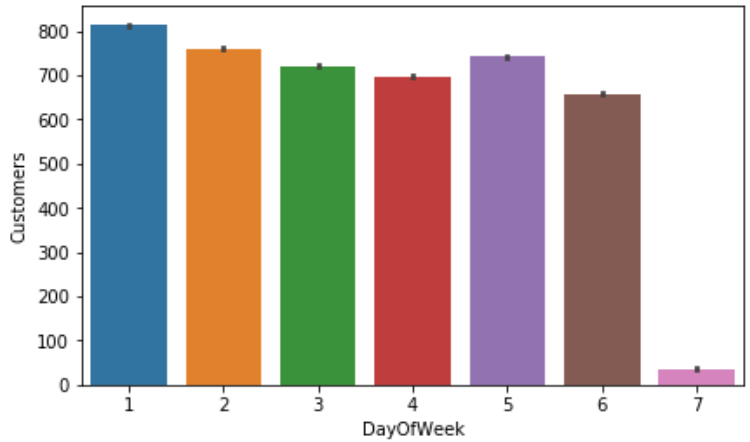
"train.csv" 共有 1017209 条数据，时间跨度从 { 2013-01-01 ~ 2015-07-31 } 大概两年半的时间的数据，"train.csv" 中的数据没有缺失值。而 “store.csv” 共有 1115 条数据，代表 1115 家店，其中跟竞争对手、促销活动相关的字段存在 null值，与竞争对手相关的字段可以使用平均值填充，而跟促销相关的字段存在null值是因为对应店铺没有参与促销活动，因此不能也不需要填充。

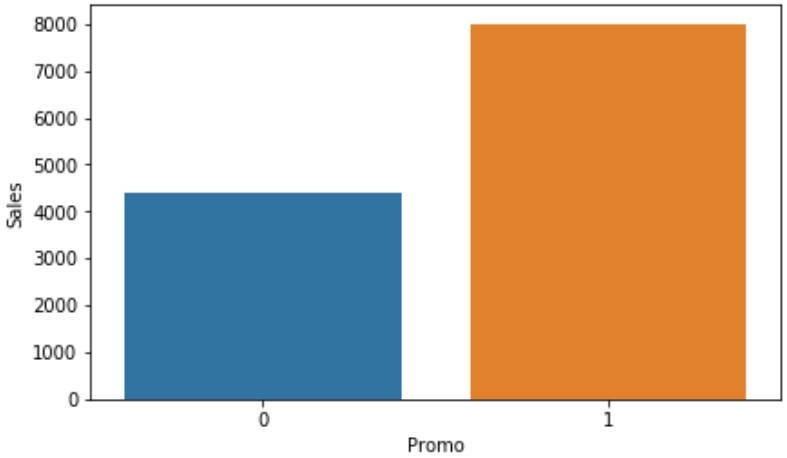
2.2 探索性可视化

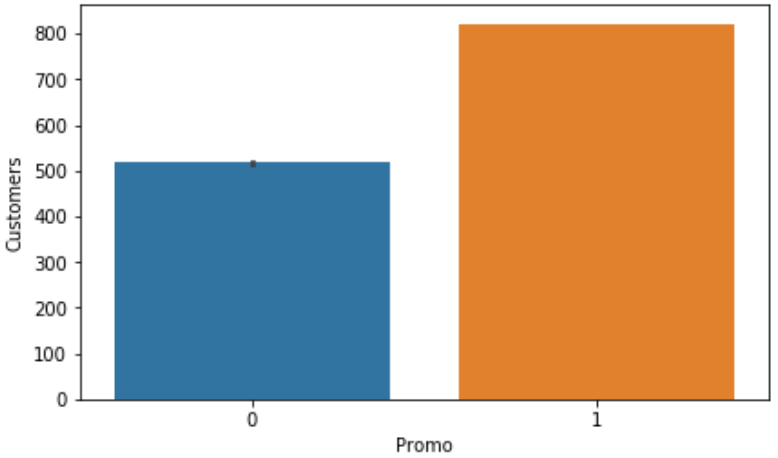
下面我们可视化部分数据图表来对数据进行分析。本项目是为了预测销售额，那我们先来看一下不同特征维度销售额变化图表：

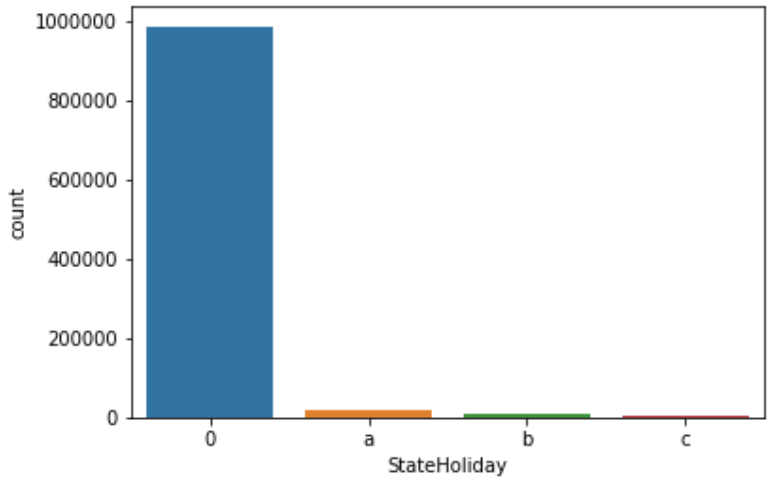

Average Sales
Sales
7000
6750
6500
6250
6000
5750
5500
5250
Sales Percent Change
Sales
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
2013-01
2013-02
2013-03
2013-04
2013-05
2013-06
2013-07
2013-08
2013-09
2013-10
2013-11
2013-12
2014-01
2014-02
2014-03
2014-04
2014-05
2014-06
2014-07
2014-08
2014-09
2014-10
2014-11
2014-12
2015-01
2015-02
2015-03
2015-04
2015-05
2015-06
2015-07
Date

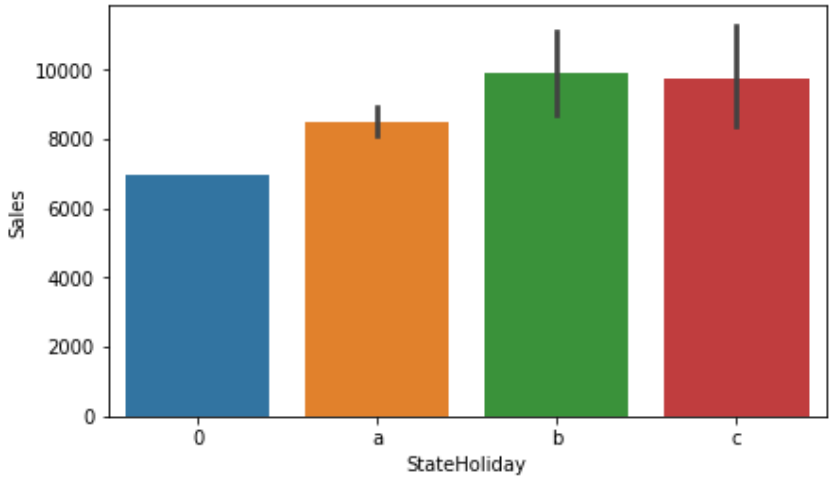

Sales
DayOfWeek
0
1000
2000
3000
4000
5000
6000
7000
8000
1
2
3
4
5
6
7

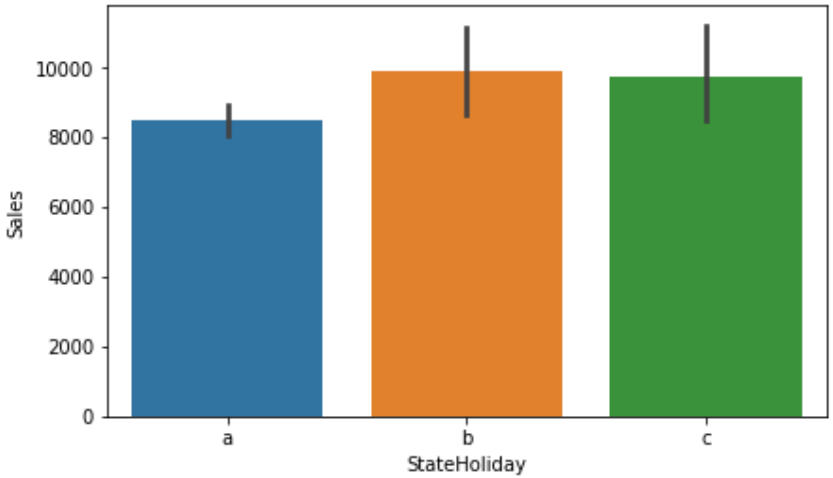

Customers
DayOfWeek
0
100
200
300
400
500
600
700
800
1
2
3
4
5
6
7

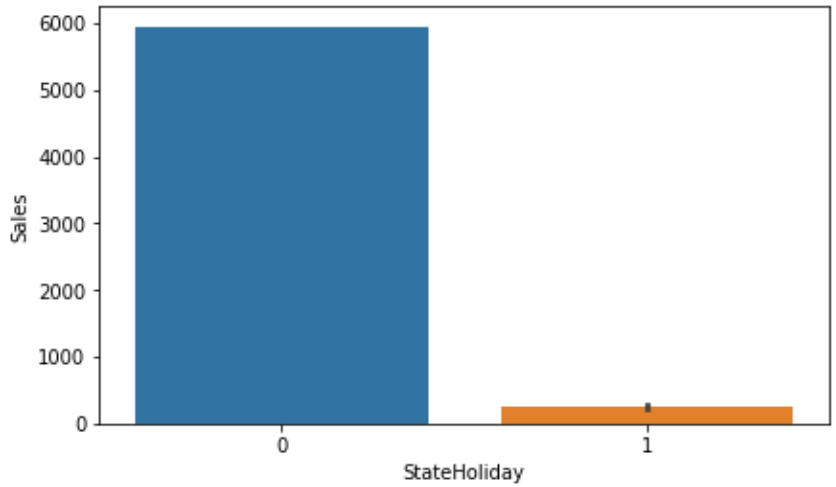

Sales
Promo
0
1000
2000
3000
4000
5000
6000
7000
8000
0
1

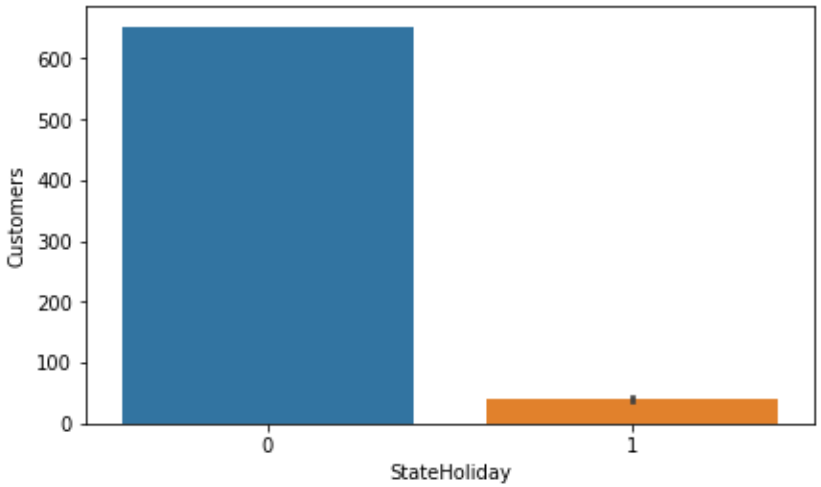

Customers
Promo
0
100
200
300
400
500
600
700
800
0
1

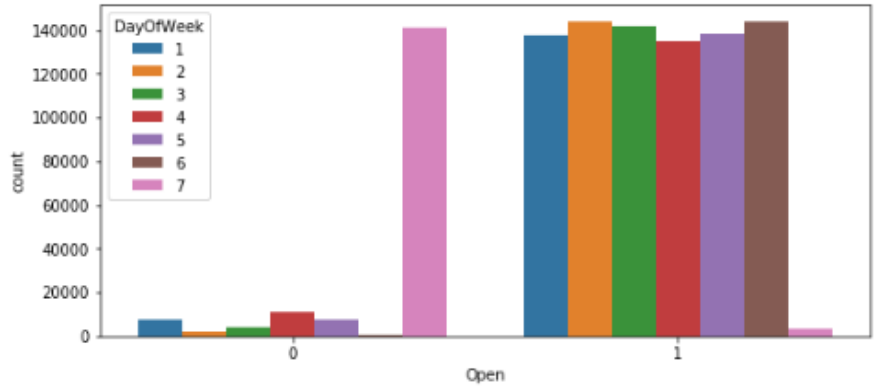

1000000
800000
600000
400000
200000
0
count
0
a
b
c
StateHoliday

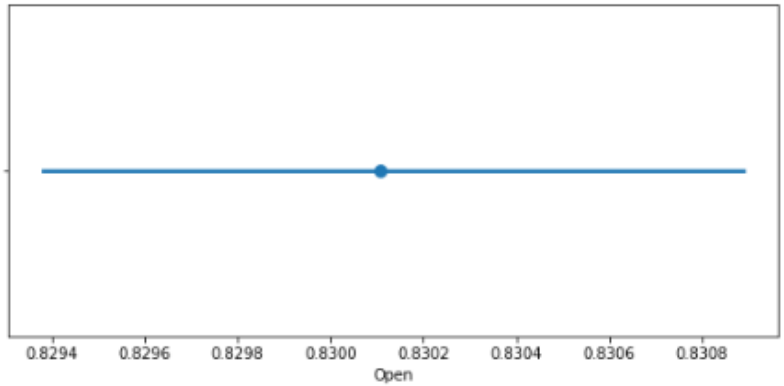

10000
8000
6000
4000
2000
0
Sales
0
a
b
c
StateHoliday

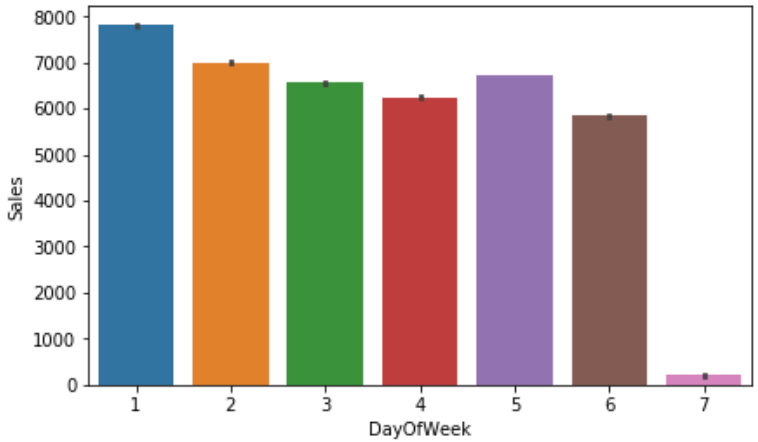

10000
8000
6000
4000
2000
0
Sales
a
b
c
StateHoliday

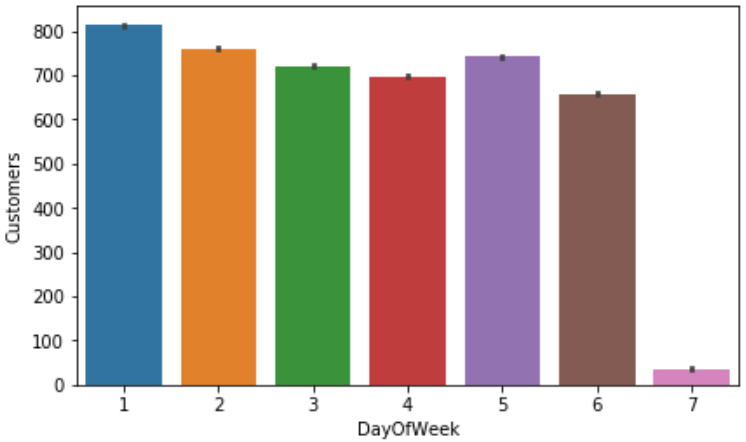

Sales
6000
5000
4000
3000
2000
1000
0
0
1
StateHoliday

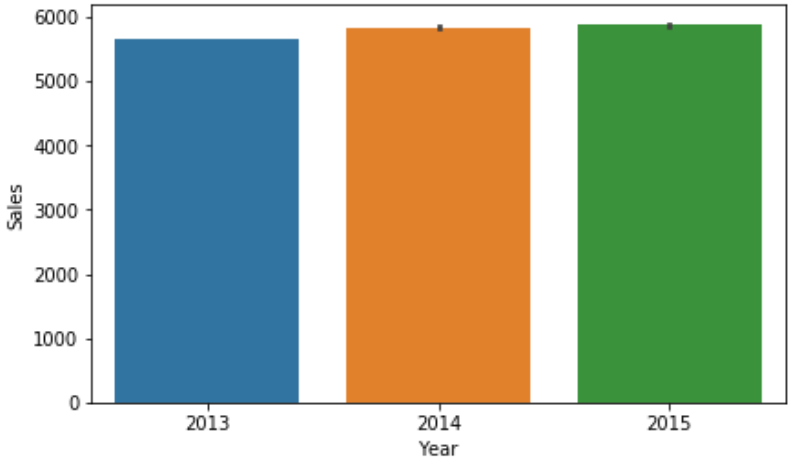

Customers
600
500
400
300
200
100
0
0
1
StateHoliday

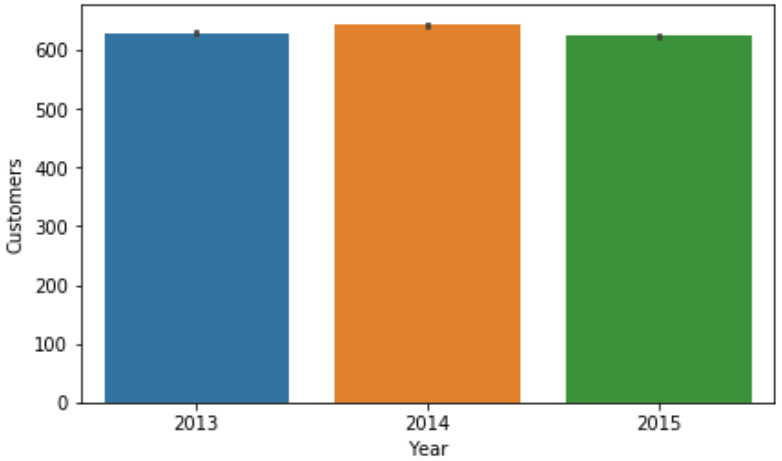

DayOfWeek
1
2
3
4
5
6
7
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
Open

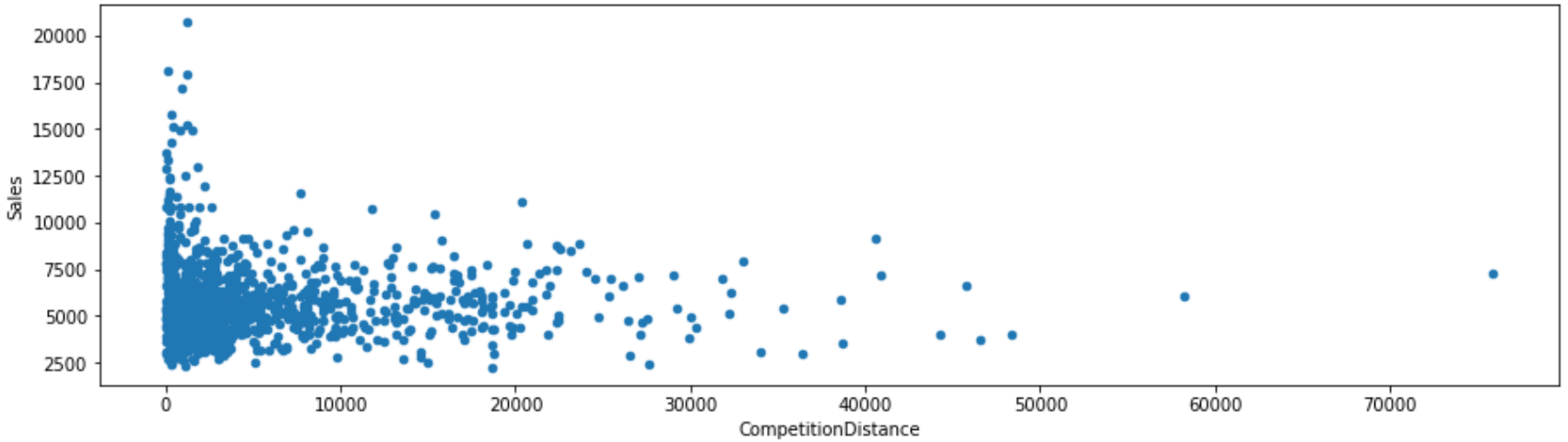

0.8294
0.8296
0.8298
0.8300
0.8302
0.8304
0.8306
0.8308
Open

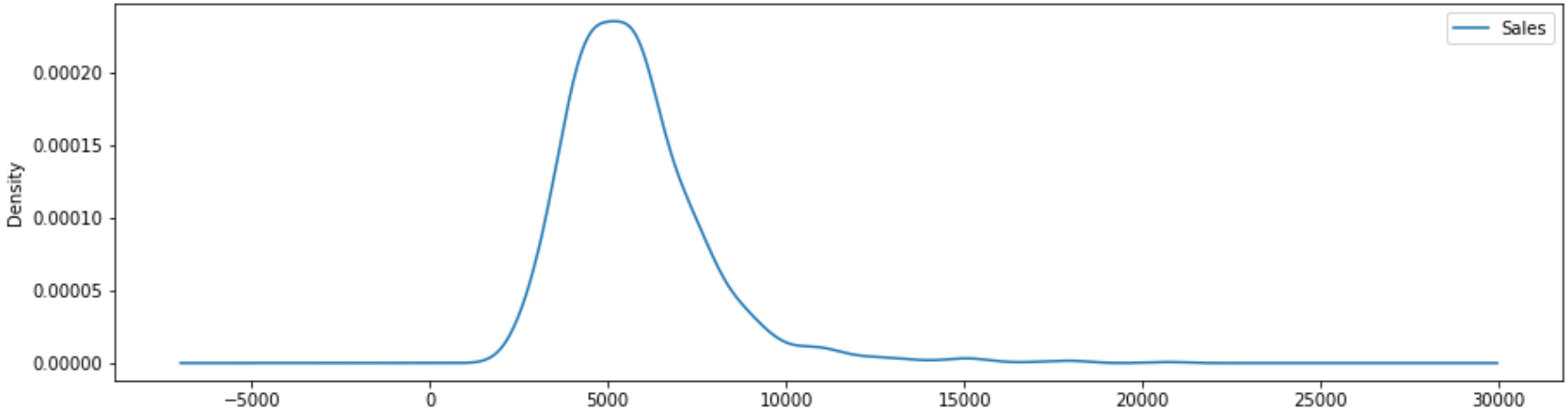

Sales
DayOfWeek
0
1000
2000
3000
4000
5000
6000
7000
8000
1
2
3
4
5
6
7


Customers
DayOfWeek
0
100
200
300
400
500
600
700
800
1
2
3
4
5
6
7

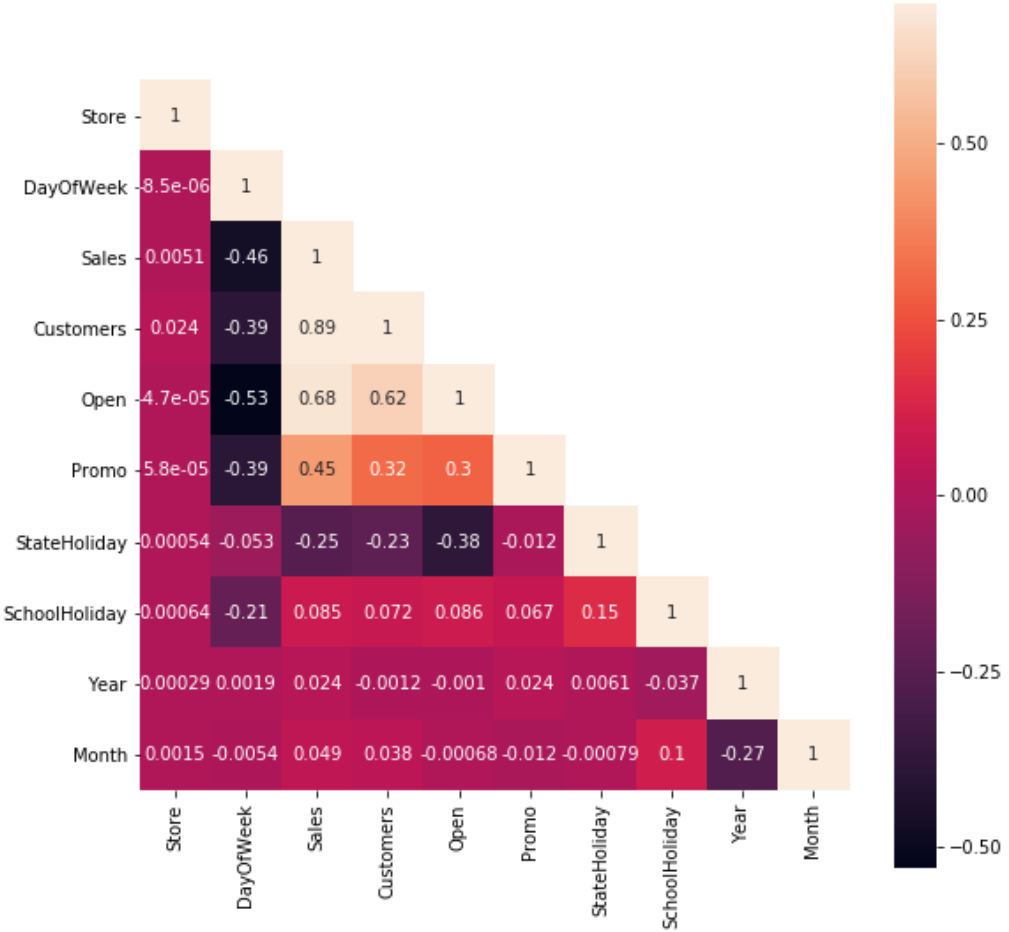

Sales
Year
0
1000
2000
3000
4000
5000
6000
2013
2014
2015


Customers
Year
0
100
200
300
400
500
600
2013
2014
2015

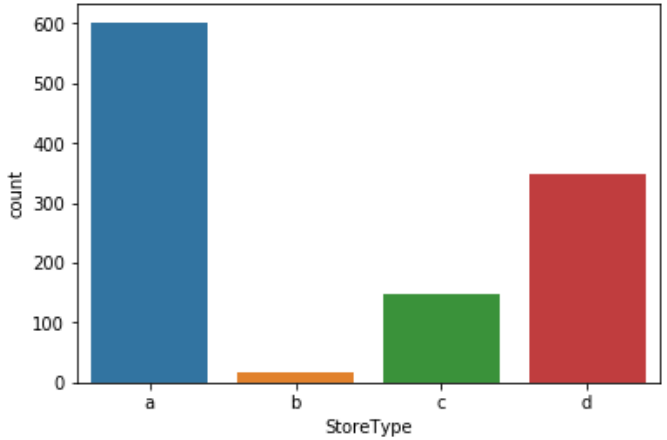


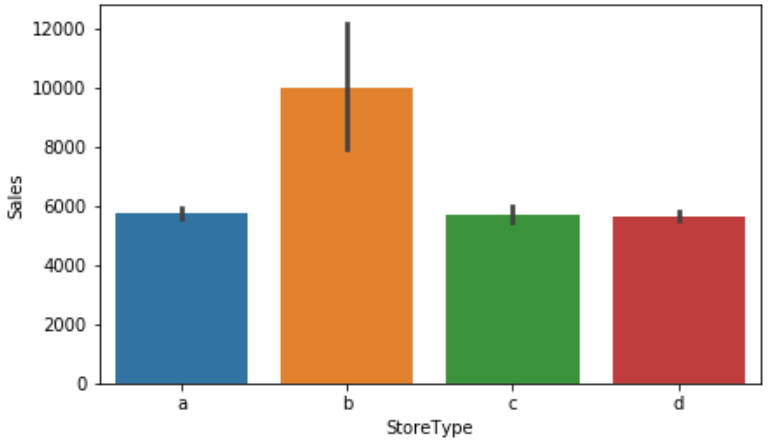


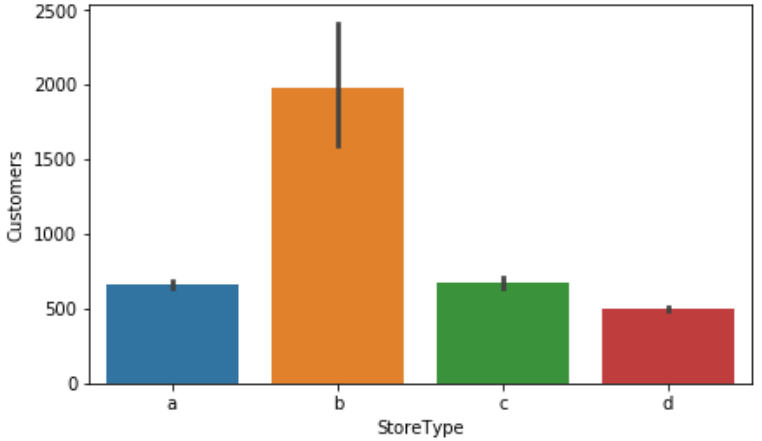
接下来分析特征与 Sales 之间的关系，首先从整体上来看各个特征的相关性。



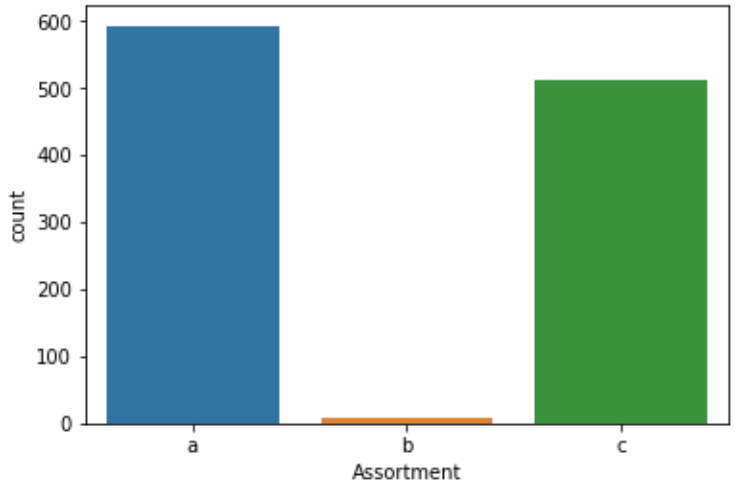
不同商店类型的销售情况

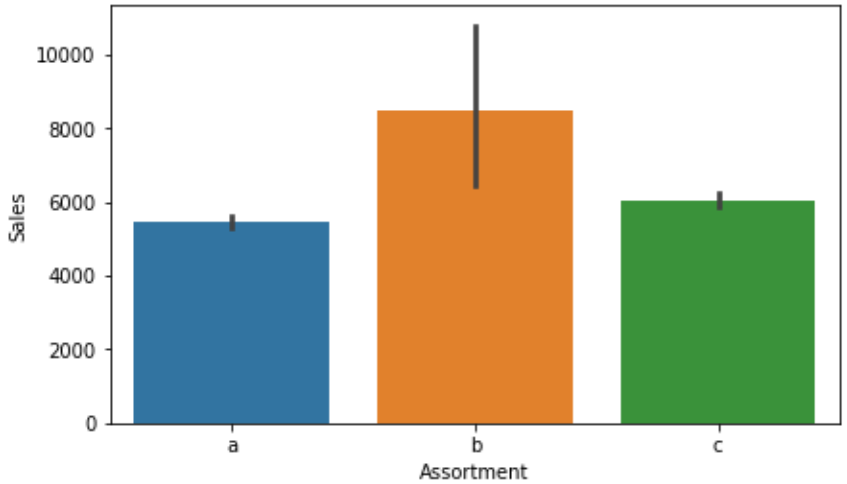


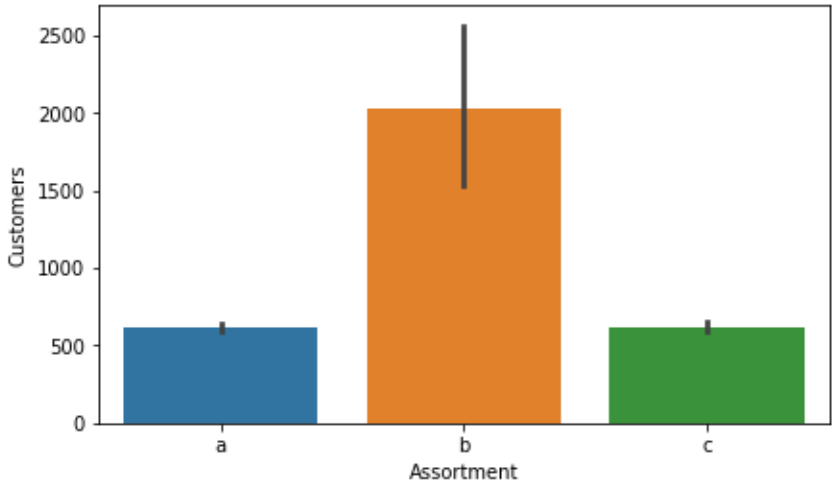




不同商家级别之间的销售对比

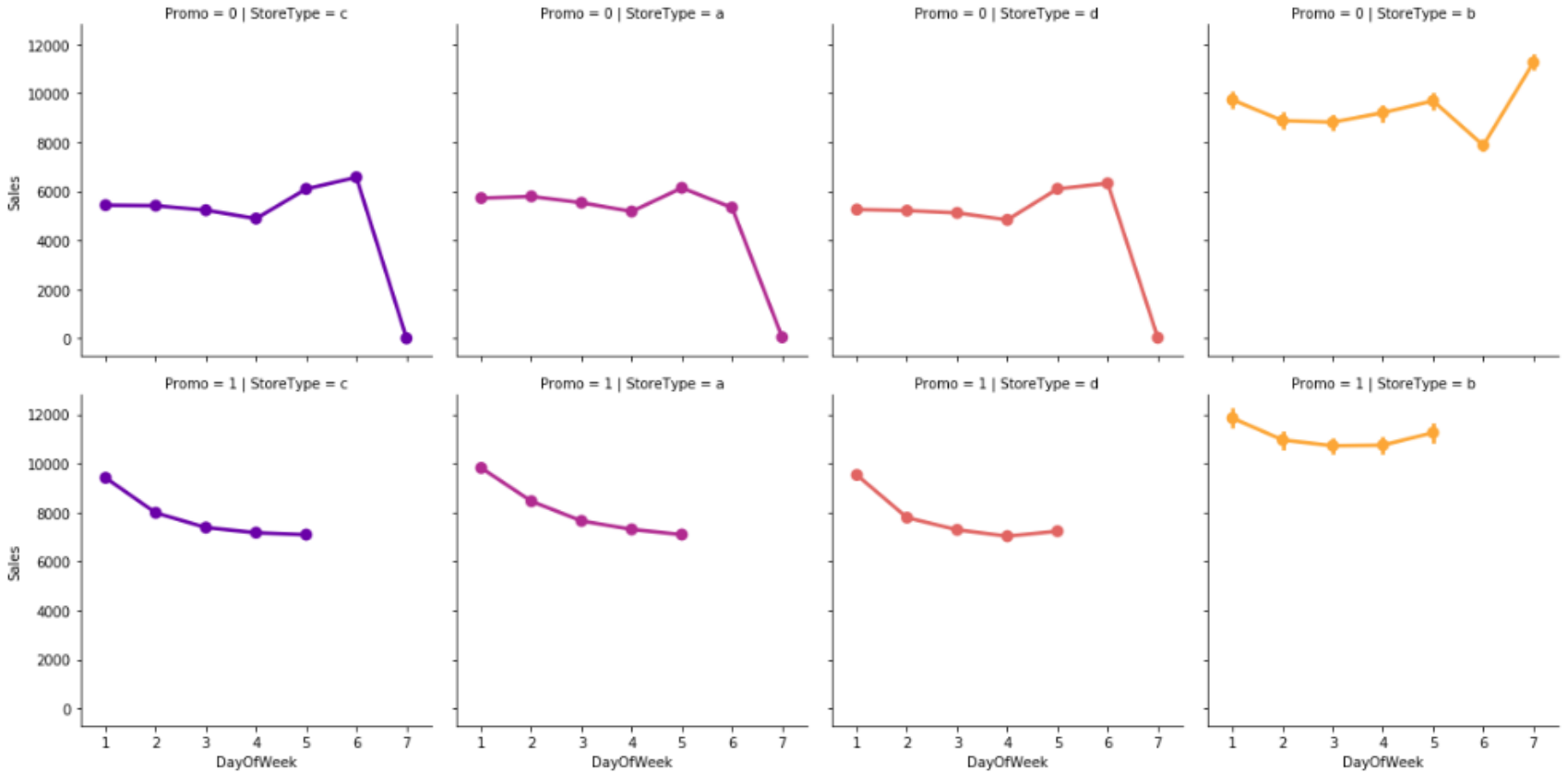


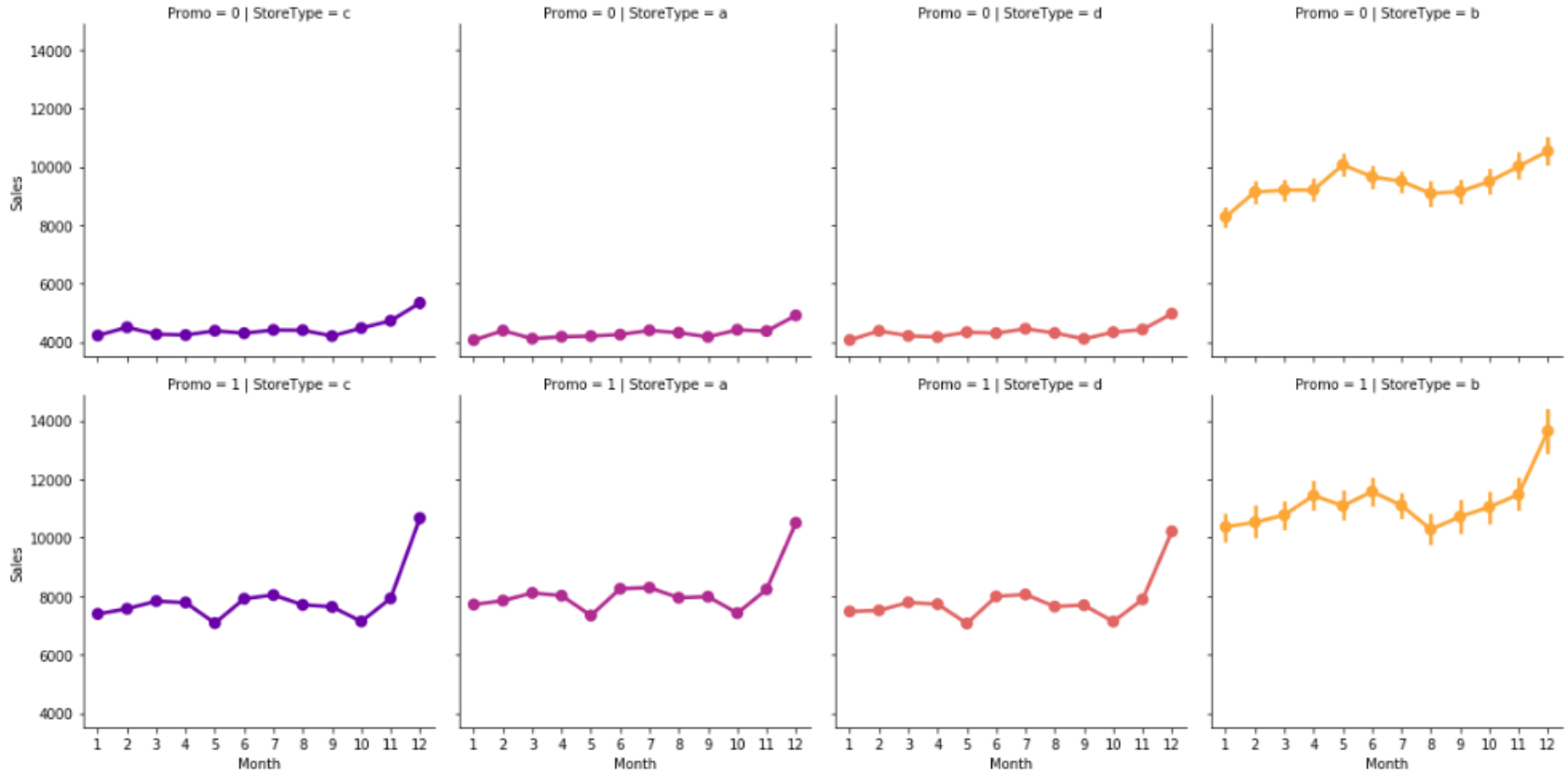


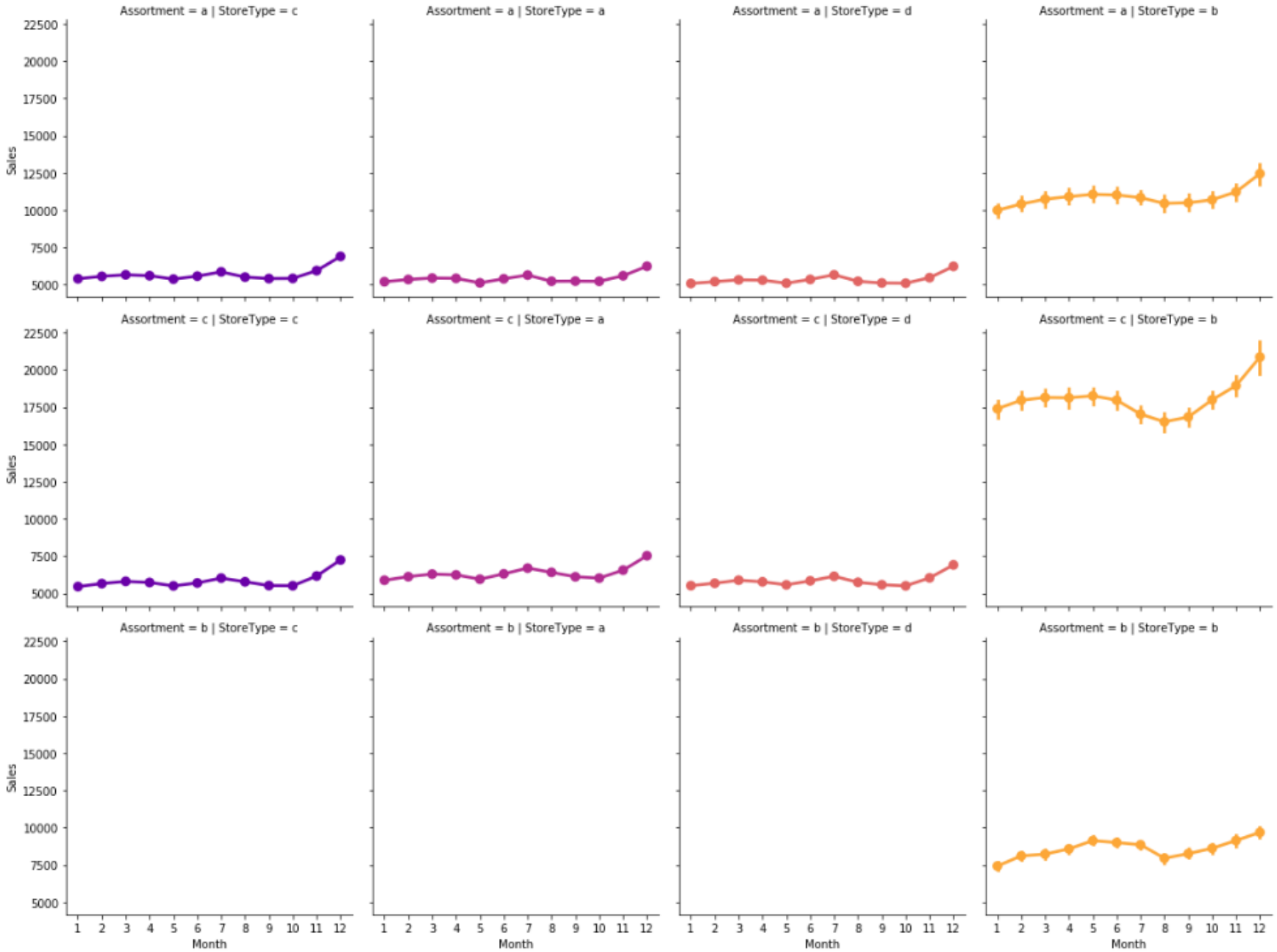


竞争对手距离对销量的影响


Sales & CompetitionDistance
40000
30000
20000
10000
0
0
10000
20000
30000
40000
50000
60000
70000


Promo = 0 | StoreType = c
Promo = 0 | StoreType = a
Promo = 0 | StoreType = d
Promo = 0 | StoreType = b
Promo = 1 | StoreType = c
Promo = 1 | StoreType = a
Promo = 1 | StoreType = d
Promo = 1 | StoreType = b
Sales
DayOfWeek
0
2000
4000
6000
8000
10000
12000
1
2
3
4
5
6
7


Promo = 0 | StoreType = c
Promo = 0 | StoreType = a
Promo = 0 | StoreType = d
Promo = 0 | StoreType = b
Promo = 1 | StoreType = c
Promo = 1 | StoreType = a
Promo = 1 | StoreType = d
Promo = 1 | StoreType = b
Sales
Month
14000
12000
10000
8000
6000
4000
1 2 3 4 5 6 7 8 9 10 11 12


Assortment = a | StoreType = c
Assortment = a | StoreType = a
Assortment = a | StoreType = d
Assortment = a | StoreType = b
Assortment = c | StoreType = c
Assortment = c | StoreType = a
Assortment = c | StoreType = d
Assortment = c | StoreType = b
Assortment = b | StoreType = c
Assortment = b | StoreType = a
Assortment = b | StoreType = d
Assortment = b | StoreType = b
Sales
Month
22500
20000
17500
15000
12500
10000
7500
5000
1 2 3 4 5 6 7 8 9 10 11 12

视图分析小结：

根据可视化图表显示，我们可以看出销售额的变化受到各属性不同程度影响；

日期的转换对于预测非常重要的，不同月份的销售变化很大。

我们也看到竞争相关的属性，会影响销售的客户数量，从而影响销量。

客户数与销量成正比关系。促销也很重要需要对"Promo"属性进行编码。

可以看出 StateHoliday 公共节日期间销售额比不放假期间要高，说明节假日对销售额有促进作用。同时可以看出学校放不放假，对销售额几乎没什么影响。

看出 b 类型的商店的销售额要明显高于其他类型的商店，同时还可以看出周六周天从不进行促销活动，c 类型商店在周天不营业，周一的销售额比较高。

2.3 算法与方法

由于本项目采用 XGBoost

| XGBoost 是 “Extreme Gradient Boosting” 的缩写，XGBoost 算法的步骤和 GBDT 基本相同，都是首先初始化为一个常数，GBDT 是根据一阶导数，XGBoost 是根据一阶导数 gi 和二阶导数 hi，迭代生成基学习器，相加更新学习器。

泰勒公式是一个用函数在某点的信息描述其附近取值的公式。基本形式是：

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

泰勒公式

一阶泰勒展开：

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

一阶泰勒展开

二阶泰勒展开：

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + f''(x_0)\frac{(x - x_0)^2}{2}$$

二阶泰勒展开

XGBoost 的损失函数不仅使用到了一阶导数，还使用二阶导数。

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

损失函数

对上述损失函数做二阶泰勒展开，其中 g 为 一阶导数，h 为二阶导数，最后一项为正则项，

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

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

损失函数展开

XGBoost 对分类前后的增益计算采用了如下方式（ID3 采用信息增益，C4.5 采用信息增益比，CART 采用 Gini 系数），

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

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

|

增益公式

这个公式形式上跟 ID3 算法、CART 算法是一致的，都是用分裂后的某种值减去分裂前的某种值，从而得到增益。 为了限制树的生长，我们可以加入阈值，当增益大于阈值时才让节点分裂，上式中的 gamma 即阈值，它是正则项里叶子节点数T的系数，所以 XGBoost 在优化目标函数的同时相当于做了预剪枝。另外，上式中还有一个系数 lambda，是正则项里 leaf score 的 L2 模平方的系数，对 leaf score 做了平滑，也起到了防止过拟合的作用，这个是传统 GBDT 里不具备的特性。

优势

- 在寻找最佳分割点时，考虑传统的枚举每个特征的所有可能分割点的贪心法效率太低，XGBoost 实现了一种近似的算法。大致的思想是根据百分位法列举几个可能成为分割点的候选者，然后从候选者中根据上面求分割点的公式计算找出最佳的分割点。同时当分裂时遇到一个负损失时，GBM 会停止分裂。XGBoost 会一直分裂到指定的最大深度(max_depth)，然后回过头来剪枝。如果某个节点之后不再有正值，它会去除这个分裂。这种做法的优点，当一个负损失（如-2）后面有个正损失（如+10）的时候，就显现出来了。GBM 会在-2处停下来，因为它遇到了一个负值。但是 XGBoost 会继续分裂，然后发现这两个分裂综合起来会得到+8，因此会保留这两个分裂。
- 标准 GBM 的实现没有像 XGBoost 这样的正则化步骤。正则化对减少过拟合也是有帮助的。
- XGBoost 考虑了训练数据为稀疏值的情况，可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率。
- 列抽样，XGboost 借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算。
- 特征列排序后以块的形式存储在内存中，在迭代中可以重复使用；虽然 boosting 算法迭代必须串行，但是在处理每个特征列时可以做到并行。
- 按照特征列方式存储能优化寻找最佳的分割点，但是当以行计算梯度数据时会导致内存的不连续访问，严重时会导致 cache miss，降低算法效率。paper 中提到，可先将数据收集到线程内部的 buffer，然后再计算，提高算法的效率。

XGBoost 还考虑了当数据量比较大，内存不够时怎么有效的使用磁盘，主要是结合多线程、数据压缩、分片的方法，尽可能的提高算法的效率。

2.4 基准指标

本项目所采用的基准指标是在 Kaggle 上 Private LeaderBoard 的得分，具体来说，对 test.csv 中各商店的销售额进行预测，然后将预测结果提交到 Kaggle 上，利用 Kaggle 竞赛 Rossmann Store Sales 的 Private LeaderBoard 的得分作为基准，目标是最终的得分要达到 0.117 及以下

三、方法

3.1 数据预处理

1. 其他非数值字段属性处理

- "Null"值填充:test 的 open 字段填充为营业状态1、
- "Null"值填充:store 的 CompetitionOpenSinceMonth、CompetitionOpenSinceYear、CompetitionDistance、Promo2SinceYear、Promo2SinceWeek 其中Promo2SinceYear、Promo2SinceWee是由于 Promo2 字段为 0 导致的填充为0即可； 其他也用0填充

2. 新特征的获取

- 时间相关：通过 Date 一个字段，挖掘出Year、Quarter、Month、Day、WeekOfYear、IsWorkDay6 个字段。
- 促销相关：根据 1 中挖掘的 Month 字段，通过 Promo2 判断商店是否参与了持续促销，如果参与了再判断当前数据的 Month 是否在 PromoInterval 的促销月中，挖掘出 IsInPromo

字段。而通过IsInPromo 字段判断是否处于促销月，如果是，那么再通过 Day 字段获取该月已经持续的天数来挖掘出当前 处于持续促销活动的第几天，即 PromoDays 字段。

- 竞争对手相关：对于竞争对手相关字段，CompetitionOpenSinceYear、CompetitionOpenSinceMonth 两个字段并不是常规意义上的数值型字段，关系由大小决定，而是时间点字段，这种字段直接使用的话很难保证实际效果，但是我们可以从这两个字段中挖掘出竞争对手针对于当前数据的总营业时间，单位为月，而这一字段明显是一个典型的字段关系由数值大小来表示的字段，即 CompetitionOpenMonths。

3. 无用字段的抛弃

- 无用字段主要包括Date、Customers、Open、PromoInterval、monthStr。

4. 枚举性字段One-Hot编码

- StateHoliday:国家假日，一般假日国家假期都会关门，所有学校在公共假日都会关门，a=公共假日，b=东部假日，c=圣诞节，0=不是假日。
- StoreType:商店类型，有四种，abcd。
- Assortment:分类级别，a=基础，b=额外，c=扩展。

  对于这三个字段，由于其取值中有 a、b 等字符型数据，因此我们通过映射表{'0':0, 'a':1, 'b':2, 'c':3, 'd':4}将其映射至数字上。

5. 数值型数据归一化处理

避免由数值大小导致的字段在对预测结果的影响中权重不一致，因此做归一化处理。

- CompetitionDistance。
- PromoDays。
- CompetitionOpenMonths。

6. 提取目标字段

- 将 Sales 从 train 数据中提取出来，如果不提取，那么结果就是在训练集、验证集上表现完美的模型，在测试集上结果大打折扣，甚至因为字段的不匹配导致模型无法预测测试数据。

7. 训练集、验证集的划分方式

- 由于数据是时序数据，因此采取按照时间先后顺序划分的方式，选取最后 6 周作为验证集数据，之所以选择 6 周是因为测试集数据就是 6 周的数据，验证集跟测试集数据越接近，起到的模型验证效果越好。

8. 数据合并

- 将train 和 store合并，test 和 store合并

3.2 实施

模型实施步骤如下：

1. 模型构建：创建最初的模型参数、迭代次数以及转换训练数据的格式为xgboost需要的DMatrix。
2. 模型训练：利用xgb的train方法通过指定的参数使用指定的训练数据进行模型训练，返回一个Booster对象。
3. 使用Booster的predict进行验证集上的预测，并通过RMSPE方法计算分数。
4. 模型保存到本地，方便后续继续使用。
5. 模型优化：
    1. 参数优化。
    2. 校正系数。
    3. 增量训练验证集数据。

    A. 提交kaggle。

首先使用 XGBoost默认参数建立回归预测模型，其默认参数

** XGBoost 默认参数 **

参数	**默认值**
booster	gbtree
objective	reg:linear
eta	0.03
max_depth	10
min_child_weight	20
subsample	0.8

参数	默认值
colsample_bytree	0.7
silent	1
gamma	0
lambda	1
alpha	0.8
random_state	23
seed	100

num_boost_round = 6000的情况下

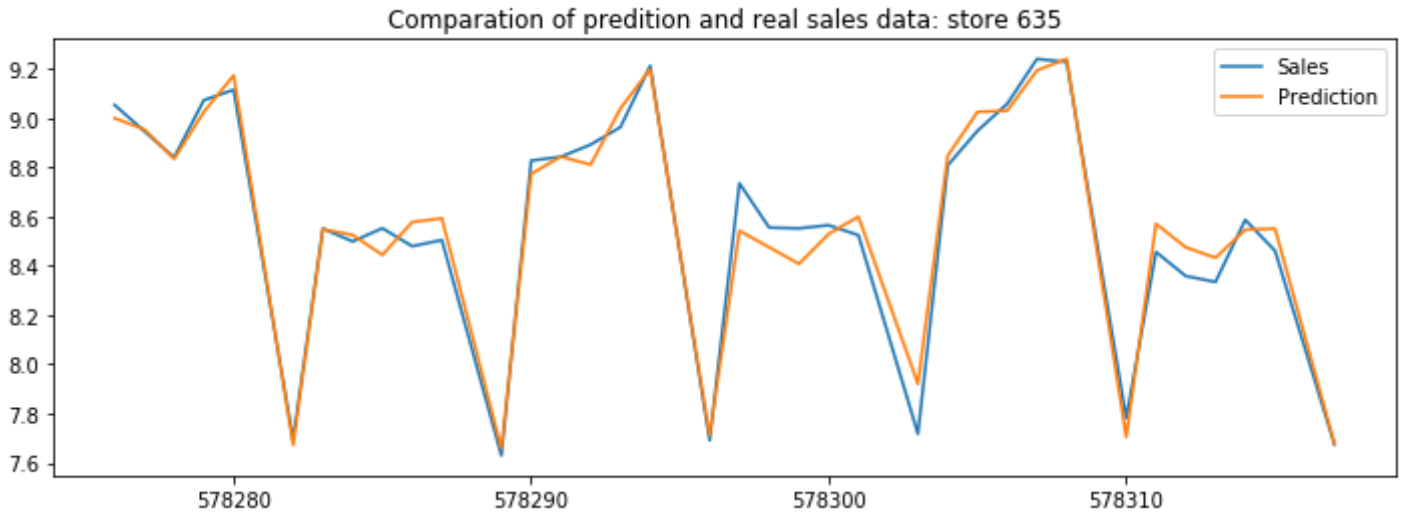
默认参数经过训练得到的结果：

train-rmse:0.077464 eval-rmse:0.117202 train-rmspe:0.104716 eval-rmspe:0.126201

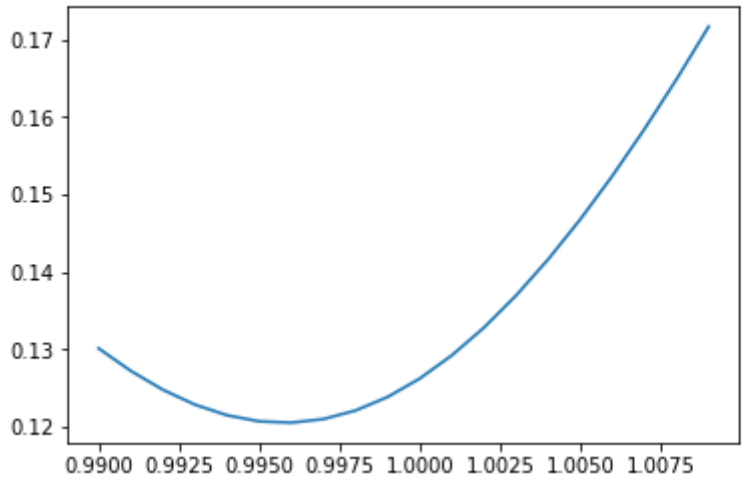
- Training time is 2590.958860 s.
- validating
- RMSPE: 0.126201

3.3 模型优化

下面我们应用校正系数，前面也提到，应用校正系数的前提是预测数据与实际数据有明显偏差，如图：



可以看到，直观感觉是整体偏高，通过赋予其一个校正系数来校正这一问题，如下：



通过优化参数后可以看到weight = 0.996位置效果最好，从整体系数训练也可看到模型结果可以看到结果从原先的 0.126201 上升到 0.120532 ，有了大幅的提升。

在此基础上进一步针对每个商家，通过同样的方法寻找weight值最优。最后 RMSPE 为:0.113504,效果越来越好。

最后完整数据训练

所有训练都是通过划分数据集得到训练集和验证集来训练的，因为最终要放到kaggle去测试，因此最后一步优化我们将训练集+验证集统一作为训练数据去训练我们的模型，同时应用之前的所有优化手段。经过kaggle验证，使用全数据的模型在参数不变、校正系数不变的情况下，private score为0.11655，相比于之前的最佳得分0.11077升高了很多，主要原因在于没有验证集能够进行模型参数的选择，导致模型表现稍差。

四、结果

4.1 模型验证

XGBoost模型通过数据训练进行有监督的回归学习，输入数据使用了kaggle提供的train和store，输出为数值型的"Sale"，满足项目需求，基准模型的RMSPE：0.117（kaggle前10%），在初始参数训练结果的基础上校正系数模型最终得到的结果低于0.117，因此合理可用的，结果令人满意。

我们来比较一下本地数据的训练结果：

模型	RMSPE
基准模型	0.439239
XGB初始模型	0.126028
XGB整体校正模型	0.120532
XGB细致校正模型	0.113504

通过优化参数weight = 0.996整体训练模型结果由原先的 0.126201 上升到 0.120532 ， 这说明通过系数调整的方法可以提升结果。 因此在此基础上进一步针对每个商家校正独立的weight最优值。经过训练得出结果0.113504。

因此最后一步优化我们将训练集+验证集统一作为训练数据去训练我们的模型，同时应用之前的所有优化手段。经过kaggle验证，使用全数据的模型在参数不变、校正系数不变的情况下，private score为0.11655，相比于之前的最佳得分0.11077升高了很多， 主要原因在于没有验证集能够进行模型参数的选择，导致模型表现稍差。

4.2 结果分析和评估

由于之前都是在训练数据上进行的，下面来看看模型在测试集上的表现。利用上面建立的基础模型以及参数优化过的模型，分别对 test 数据进行预测， 然后将预测结果提交到 Kaggle 上 Rossmann Store Sales 竞赛的 Leaderboard。

基准模型

submission_baseline_kaggle.csv
a minute ago by Jo Choi
基准模型提交

0.53325 0.48972

xgb初始模型

submission_xgb_kaggle.csv
4 minutes ago by Jo Choi
add submission details

0.12444 0.12449

xgb整体校正模型

submission_xgb_factor_kaggle.csv
4 minutes ago by Jo Choi
add submission details

0.12067 0.11977

xgb细致校正模型

submission_xgb_sfactor_kaggle.csv
4 minutes ago by Jo Choi
add submission details

0.11077 0.11226

xgboost细致校正模型+合并拆分的训练集

submission_xgb_final_kaggle.csv
a few seconds ago by Jo Choi
add submission details

0.11655 0.11014

经过XGBoost模型经过参数优化、校正系数+ 完整数据在Kaggle的得分：

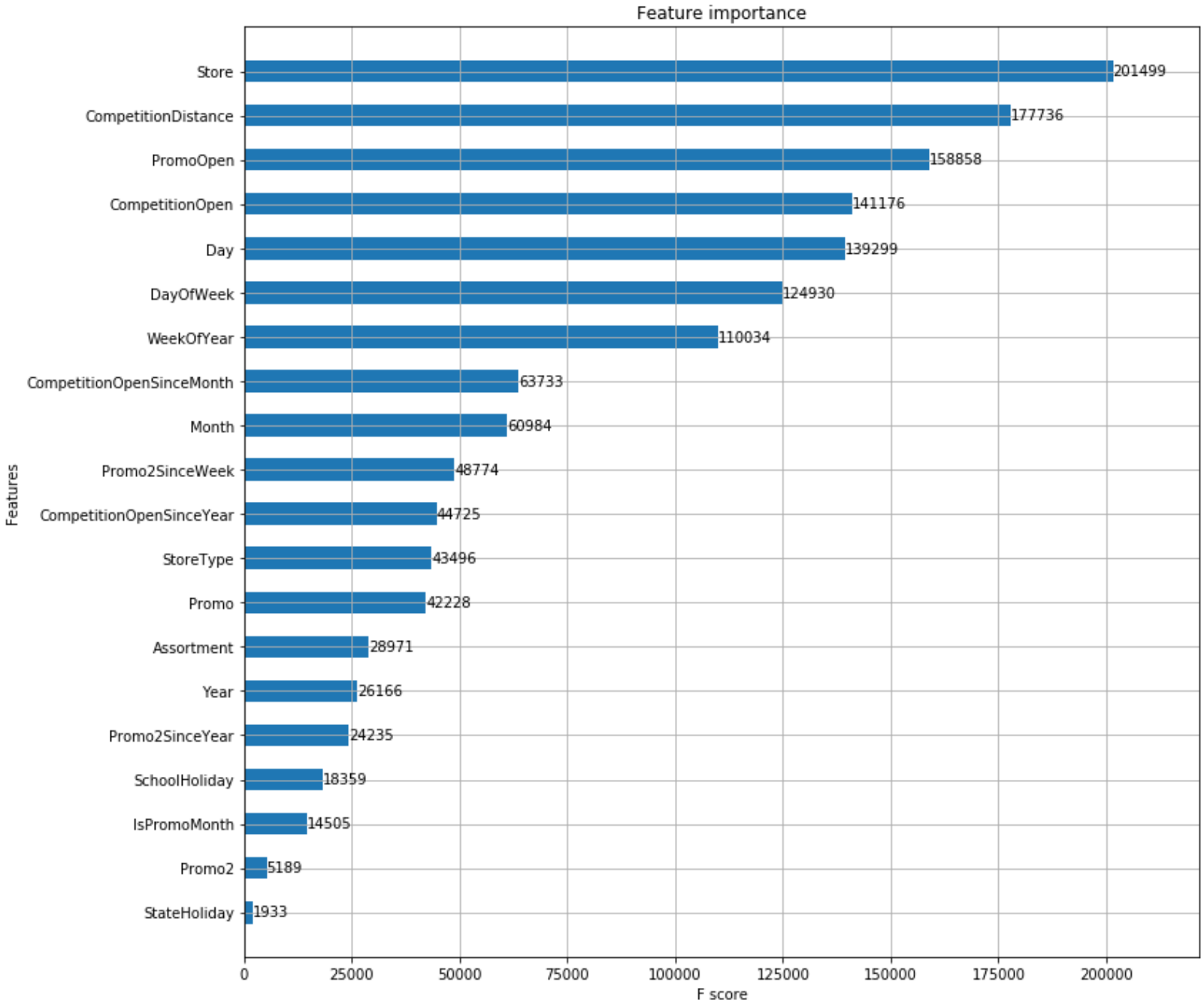
- private score:0.11655
- public score:0.11014

可以看到XGBoost模型达到预期的效果，并且在系数优化后有了很大的效果，证明模型的校正确实有效。

五、结论

5.1 结论

本项目基于 XGBoost方法对 Rossmann1115 家商店的销售额进行预测，最终的预测结果也到达要求。XGBoost特 征的重要性，其结果如图所示。


Feature importance
Store
CompetitionDistance
PromoOpen
CompetitionOpen
Day
DayOfWeek
WeekOfYear
CompetitionOpenSinceMonth
Month
Promo2SinceWeek
CompetitionOpenSinceYear
StoreType
Promo
Assortment
Year
Promo2SinceYear
SchoolHoliday
IsPromoMonth
Promo2
StateHoliday
201499
177736
158858
141176
139299
124930
110034
63733
60984
48774
44725
43496
42228
28971
26166
24235
18359
14505
5189
1933
Features
0
25000
50000
75000
100000
125000
150000
175000
200000
F score

从上图可以看出，虽重要性靠前的几个特征有 Store、CompetitionOpen、Day、CompetitionDistance、WeekofYear，还可以看出来这些特征大部分都是与时间有关的特征。

5.2 思考

本项目主要分为 几 个部分：

- 1. 初始化开发环境，加载所需库。
- 1. 加载数据，数据完整性探索。（可视化图表见数据可视化.ipynb (数据可视化.ipynb)）
- 1. 数据预处理。
- 1. 数据探索。
- 1. 特征工程。
- 1. 模型构建、训练、调参、优化 。
- 1. 生成kaggle的提交文件。
- 1. 记录kaggle得分情况。

其中数据探索和特征工程这两个部分是十分重要的。数据探索在整个流程中是起到承上启下的作用，因为数据预处理和特征工程往往不能一次性地就能得 到好的结果，需要通过数据探索来不断完善。也只有在对数据有了充分了解的前提下，才有可能对数据做出比较合理的处理，才能做好特征工程。对于机器学习 来说，特征工程是决定上限的关键，而算法只是去逼近这个上限。在特征工程中，特征预处理，特征选择都有各自对应的方法，就本人而言，我认为特征构造是比 较关键也是最困难的部分，因为这需要对数据有很充分的理解。在模型优化过程中，我们需要通过参数调整，来得到比较理想的结果。 目前比较常用的方法有网格搜索法，这种方法比较耗时，可以尝试使用贝叶斯优化来调整参数，不仅训练时间会降低，而且最终的效果也很不错。

5.3 展望

本项目虽然在样本数据上满足了要求，但是依然还有一些地方有待提高，具体如下：

- 1. 参数调优：对于模型参数的调节不一定是最优的，因此还可以进一步地优化。
- 1. 特征工程：目前使用的特征并不多，而且还可以构造出更多的特征，例如可以搭建一个神经网络去构造新的特征，用来作为模型的输入，相信能取得更好 的结果。
- 1. 训练集分出去的验证集未校正系数的情况下提交给了kaggle，实际上在验证集也通过方法获取细致系数的情况下，相信结果会更好。
- 1. 模型尝试：由于本项目其实是跟时间序列相关的，因此可以尝试其他处理时间序列的模型。

参考文献

- 【1】XGBoost官方文档：https://xgboost.readthedocs.io/en/latest/ (https://xgboost.readthedocs.io/en/latest/)
- 【2】XGBoost参数优化：https://blog.csdn.net/aicanghai_smile/article (https://blog.csdn.net/aicanghai_smile/article)
- 【3】Pandas官方文档：http://pandas.pydata.org/pandas-docs/stable/api.html (http://pandas.pydata.org/pandas-docs/stable/api.html)
- 【4】Kaggle项目地址：https://www.kaggle.com/c/rossmann-store-sales (https://www.kaggle.com/c/rossmann-store-sales)

- 【5】XGBoost使用方法：
  http://xgboost.apachecn.org/cn/latest/how_to/param_tuning.html
  (http://xgboost.apachecn.org/cn/latest/how_to/param_tuning.html).