机器学习工程师纳米学位毕业项目

Rossmann 销售预测

王云波

2019 年 4 月 26 日

1. 定义

1.1. 项目概述

本项目源自 Kaggle 的一次竞赛【参考文献 1】，奖金提供方是 Rossmann，一个国际连锁药店，希望得到一个可靠的模型来预测销售额，用以改善业务。销售预测是大多数企业都关心的能力，准确的销售预测能力很多时候对企业提升生产效率，降低管理成本和降低经营风险有着重要作用。

Rossmann 在七个欧洲国家经营着 3000 多家药店。目前，Rossmann 门店经理的任务是提前 6 周预测他们的日常销售。商店的销售受到许多因素的影响，包括促销、竞争、学校和法定假日、季节和地区。由于成千上万的门店经理根据各自的情况预测销售状况，结果的准确性可能会非常不同。

Rossmann 向参赛者发出挑战，通过在德国各地的 1115 家商店的每日销售额来预测未来 6 周的销售额。可靠的销售预测能够让门店经理制定有效的员工时间表，提高生产效率和员工的积极性。本项目的目的是为 Rossmann 创建一个健壮的预测模型，用以帮助商店经理专注于他们最重要的事情: 客户和团队。

1.2. 问题陈述

项目所选的数据集为 Kaggle 中名为 Rossmann Store Sales 竞赛所提供的有关 Rossmann 销售的历史数据。根据 Kaggle 的规则，参赛者允许使用公开的其他数据，

本项目也将遵循这一规则。

为了实现一个可靠的销售预测模型，我们需要根据 Rossmann 药店的信息，其中包括促销，竞争对手，节假日，以及在过去的销售情况等信息，来预测 Rossmann 未来 6 周的日销售额。销售额预测首先是基于时间序列的，时间维度是模型需要考虑的重要元素，其次这是一个回归问题，预测结果是当天具体的销售额。

1.3. 指标评价

模型的性能指标是根据均方根百分比误差（RMSPE）计算的。RMSPE 的计算公式为

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

其中 y_i 表示其中一个商店在一天内的销售额，$\ddot{y}_i$ 表示相应的预测。如果任何一天中商店销售额为 0，则该预测在最终评分中被忽略。

比赛得分分为两个部分：

- 公开分数，通过对 39%的测试数据集验证计算后得出
- 隐藏分数，通过对剩下的 61%的测试数据集验证计算后得出

最终的比赛成绩是根据**隐藏分数**的排名而确定。

在 2015 年的这次比赛中，前三名的成绩分别是：

第一名	**Gert**	**0.10021**

第二名	NimaShahbazi	0.10386
第三名	Neokami Inc	0.10583

也就是说，预测的效果接近 10%的误差。

2. 分析

2.1. 数据可视化分析

输入数据包括两部分内容，分别是:

1. 商店信息：
    a. Store: 商店标识符
    b. DayOfWeek: 一周中的哪一天
    c. Date: 销售日期
    d. Sales: 销售额，预测目标
    e. Customers: 顾客人数，与预测目标强相关
    f. Open: 商店当时是否营业
    g. Promo: 商店当时是否进行促销
    h. StateHoliday: 法定假日。a,b,c 是假日，0 表示非节假日
    i. SchoolHoliday: 公立学校节假日，1 为节假日，0 为非节假日

2. 商店销售数据
    a. Store: 商店标识符

b. StoreType：商店类型

c. Assortment：商店级别

d. CompetitionDistance：离最近的竞争对手的距离

e. CompetitionOpenSinceMonth :最近的竞争对手距今开店月数 ,部分缺失

f. CompetitionOpenSinceYear :最近的竞争对手距今开店年数 ,部分缺失

g. Promo2：是否持续促销

h. Promo2SinceWeek：持续促销周数

i. Promo2SinceYear：持续促销年数

j. PromoInterval：促销月份

缺失数据分析

'train data'

	column	missing	total

'store data'

	column	missing	total
0	PromoInterval	544	1115
1	Promo2SinceYear	544	1115
2	Promo2SinceWeek	544	1115
3	CompetitionOpenSinceYear	354	1115
4	CompetitionOpenSinceMonth	354	1115
5	CompetitionDistance	3	1115

'test data'

	column	missing	total
0	Open	11	41088

如上表所示：

- 训练数据中没有缺失数据
- 附加商店数据中与促销相关的三个字段：PromoInterval, Promo2SinceYear,

Promo2SinceWeek 均有 554 个缺失值；与竞争对手相关的三个字段：CompetitionOpenSinceYear, CompetitionOpenSinceMonth, CompetitionDistance 分别有 354, 354 和 3 个缺失值

- 测试数据中 Open 这个字段有 11 个缺失值

商店数据情况

经计算：

- 从各个字段的基数来看，测试和训练中出现的商店，共 1115 个，都包含在 store data 中，数据的对齐比较容易
- 训练数据的量级在 100 万左右，足够用来划分数据集和验证集
- 从部分特征统计结果来看，测试数据与训练数据的分布大体一致

数据在时间上的分布

```
train period: 2013-01-01 2015-07-31
test  period: 2015-08-01 2015-09-17
test dates count: 48
expected stores count: 856.0
```

如上面结果所示：

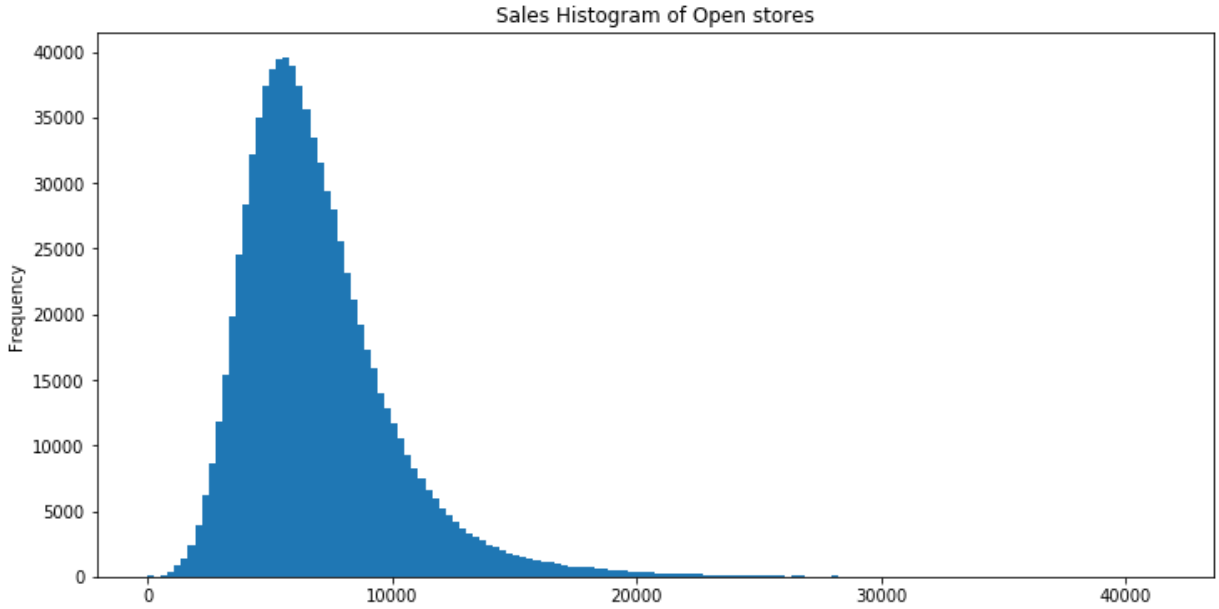
- 训练数据包含大概 2 年零 7 个月的数据
- 测试数据包含大概 6 个星期的数据，时间刚好紧靠训练数据的时间之后
- 从特征分布上的一致性加上时间分布上的连续性来看，目前不必过于担忧训练数据与测试数据之间的差异而造成的巨大方差

- 另外，考虑到数据的时序性，而且测试数据的时间跨度紧跟在训练数据之后，可以选取训练数据中的最后 6 周的数据作为验证集

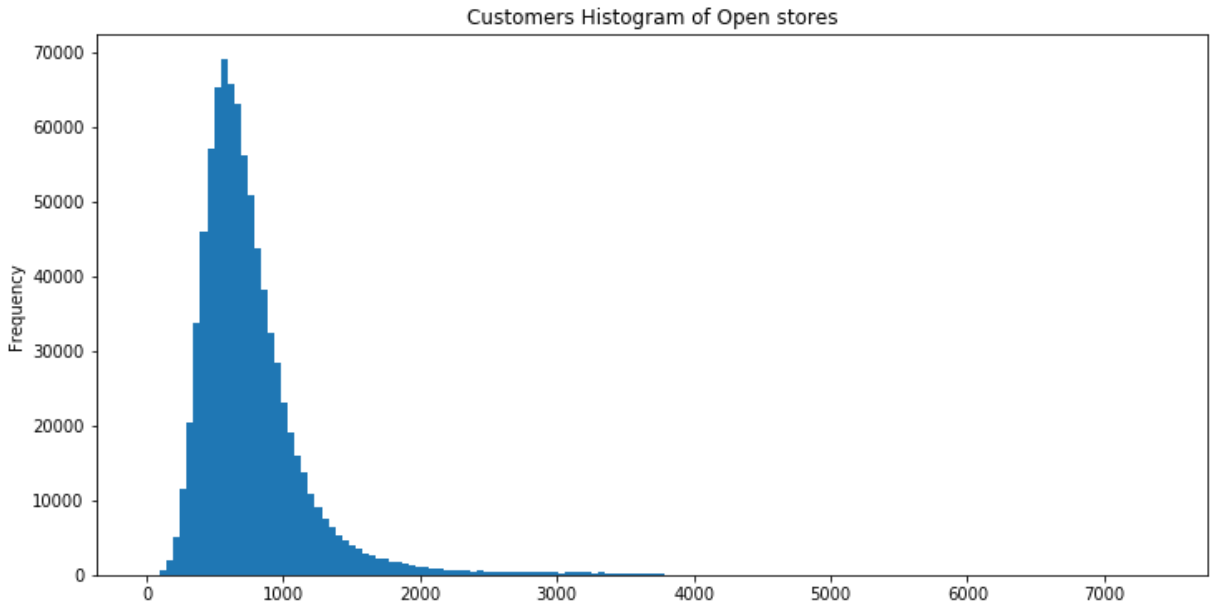
销售额分布情况

- 由于关门的商店的销售额为 0，影响图形显示，下面的两个销售额分布图都去除了销售额为 0 的数据。

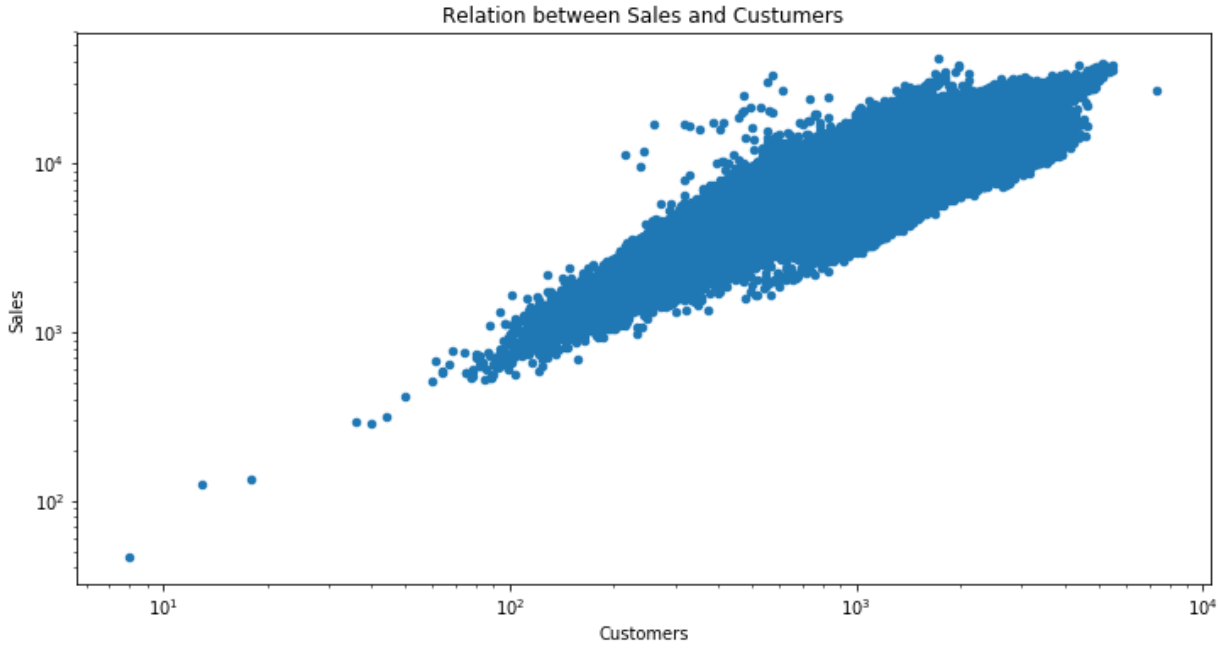
如下面的销售额分布图所示，销售额符合稍微偏斜的正态分布



不难推断，顾客数量分布与销售额的分布近似，如下图



进一步验证顾客数量（横轴）与销售额（纵轴）之间的相关性：

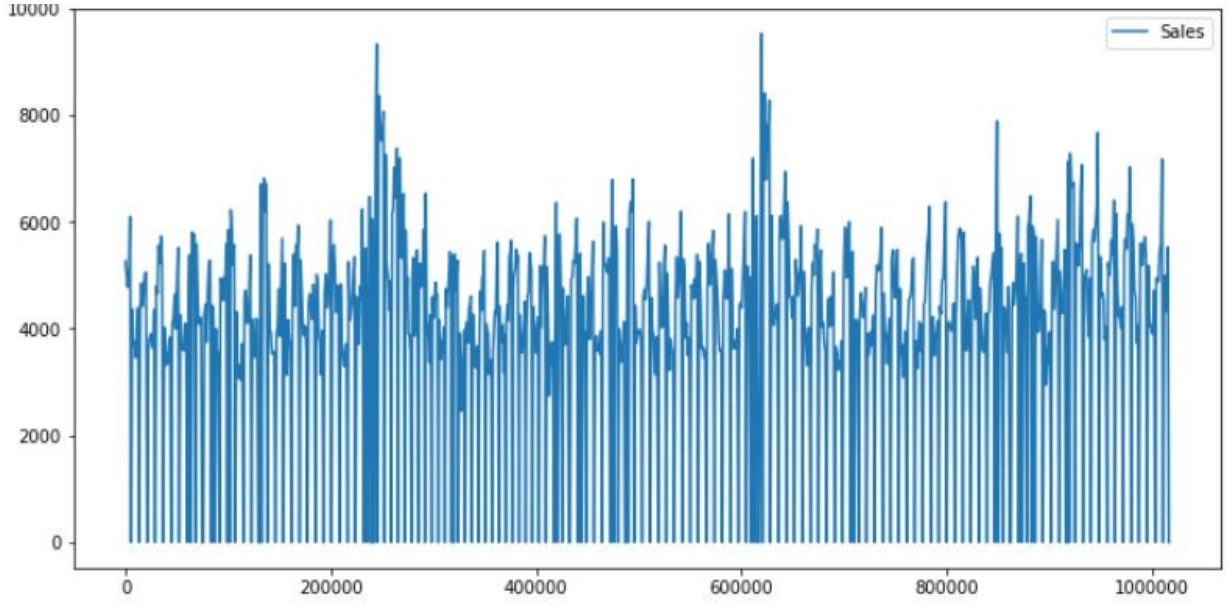


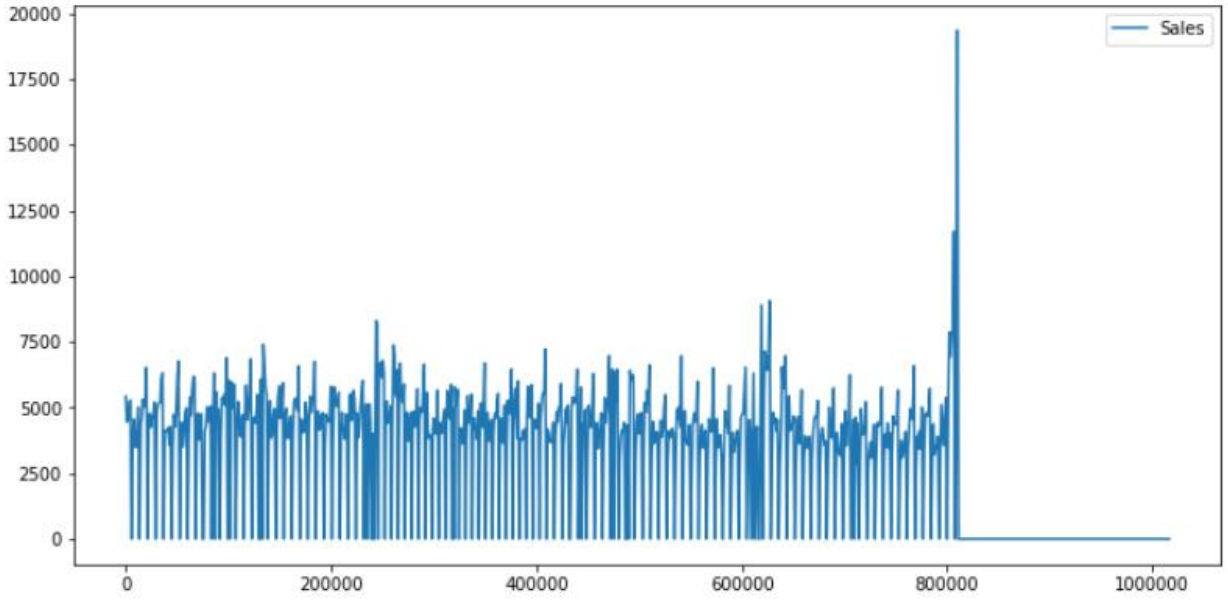
不难看出，客户数量与销售额之间有明显的线性相关性，符合预期。

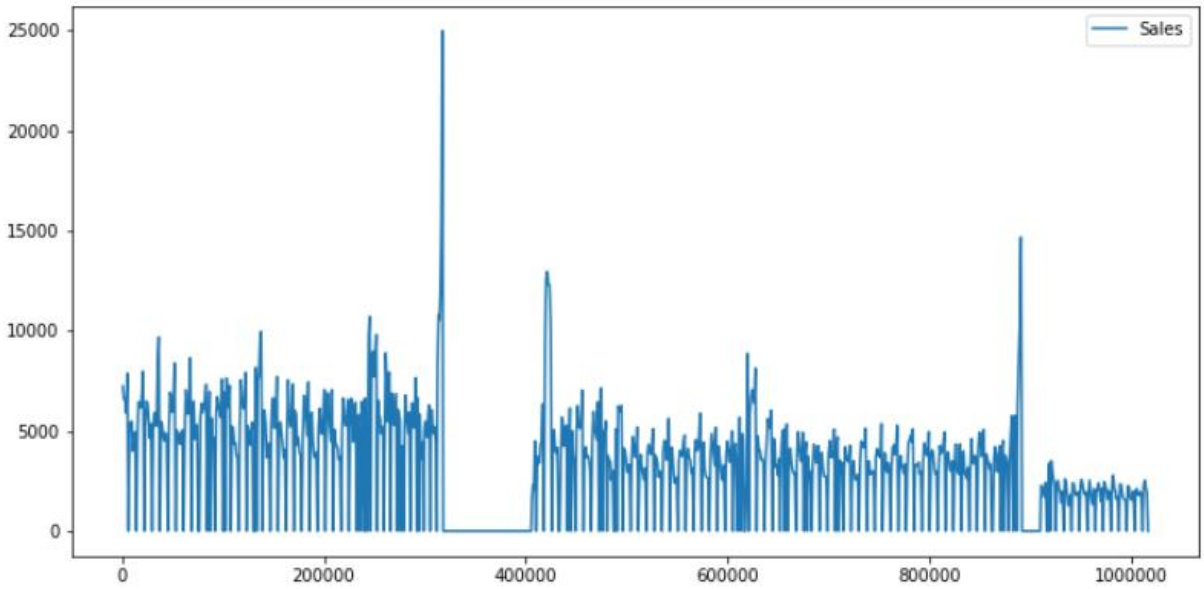
进一步分析销售数据分布

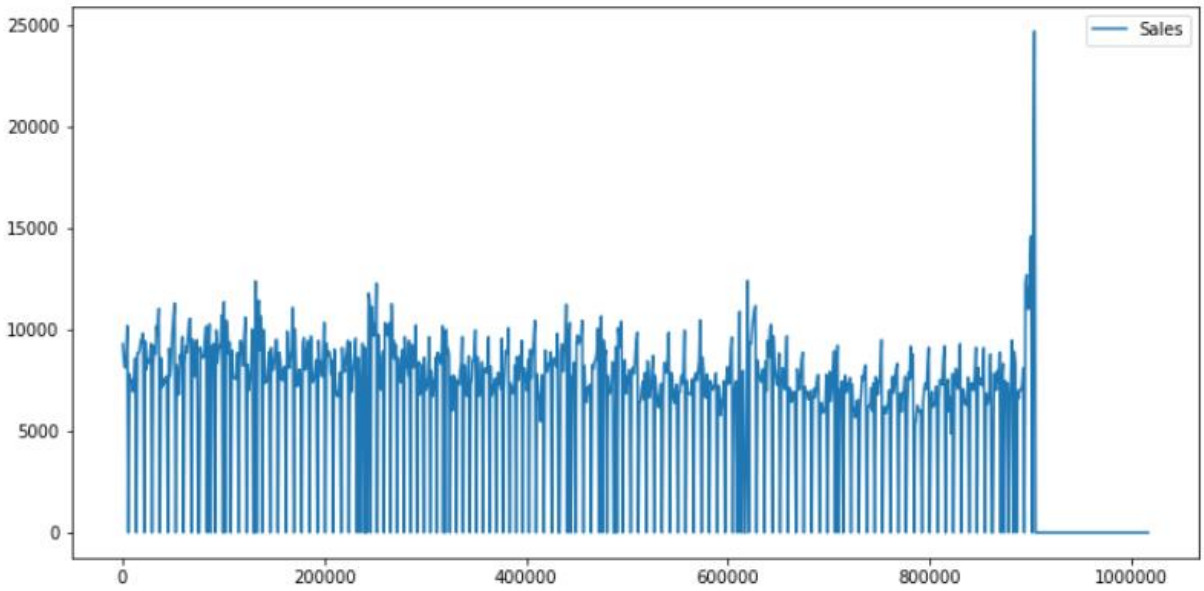
下面随机选取一些商店的销售情况来进一步观察数据特征

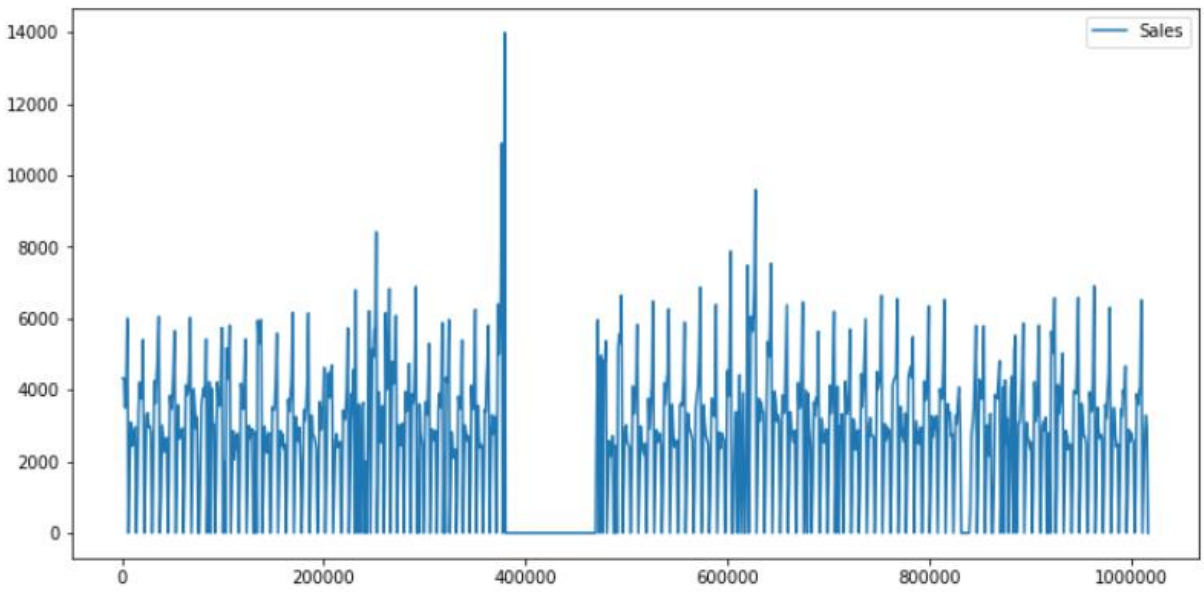
以下分别为 1 号，103 号，708 号，349 号，972 号以及 674 号 6 个商店的销售额分布图（按时间顺序）

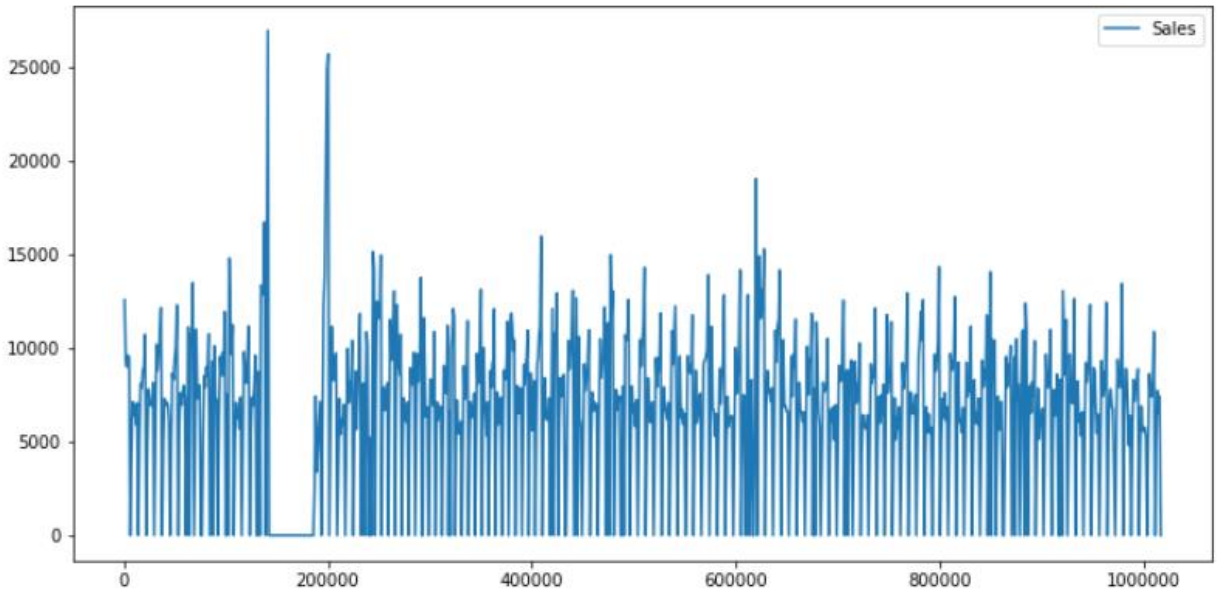





Sales
25000
20000
15000
10000
5000
0
0
200000
400000
600000
800000
1000000


Sales
25000
20000
15000
10000
5000
0
0
200000
400000
600000
800000
1000000


Sales
14000
12000
10000
8000
6000
4000
2000
0
0
200000
400000
600000
800000
1000000

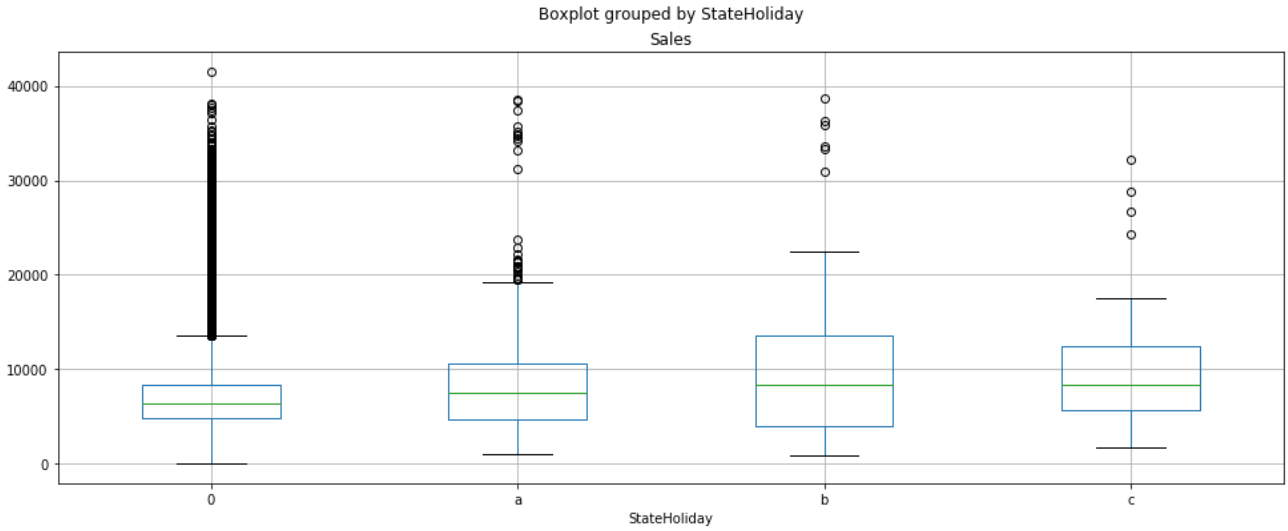


以上图线反应出几个重要信息：

- 每个商店都包含大量的 0 销售额，不难推测，其中应该包含周末和商店关门的日期
- 有的商店出现连续一周以上的 0 销售额，这个应该对应项目介绍中提到的装修事件
- 由于装修的时间是通过 0 销售额的分布决定的，为了保存这部分关键信息，不应该将所有为 0 销售额的数据剔除
- 然而 0 销售额必然对模型造成严重干扰，需要将它们作为异常信息进行处理

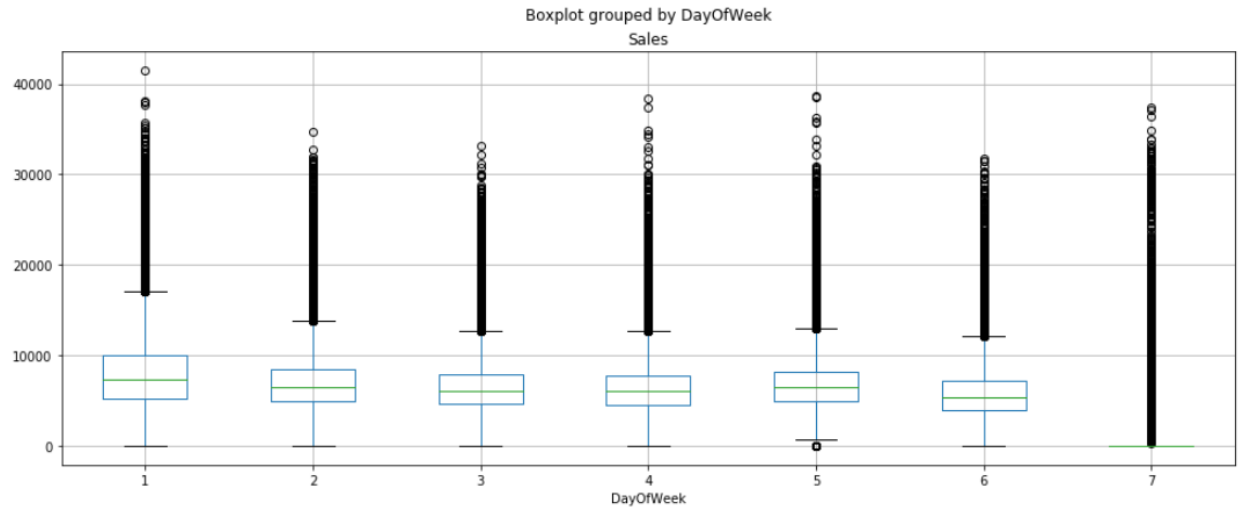
假期与促销对销售的影响

分别来看法定假日和学校假日的销售额箱图：



- 从上面箱图不难看出，法定节假日对销售的影响比较明显，人们更趋向于节假日的时候购买药品，这与大众消费的时间预期一致
- 学校假日对消费的影响很小，或许学生不是药店的主要消费群体

除了假日之外，周末对销售的影响应当也比较明显，下图为一周中每天的销售额箱图：



- 经验证得知，大部分商店在周日关门
- 另外周六的销售额相对较低
- 一周中的第几天是一个影响销售额的重要特征

接下来看促销对销售的影响，下图为不同促销选择（1 为实施促销）的销售额箱图：

不难看出，促销活动对销售的影响很明显，符合预期。

商店数据分析

商店概况

- 商店总数为 1115 个
- 商店类型的分别分别为

a 602

d 348

c 148

b 17

- 品种分布分别为

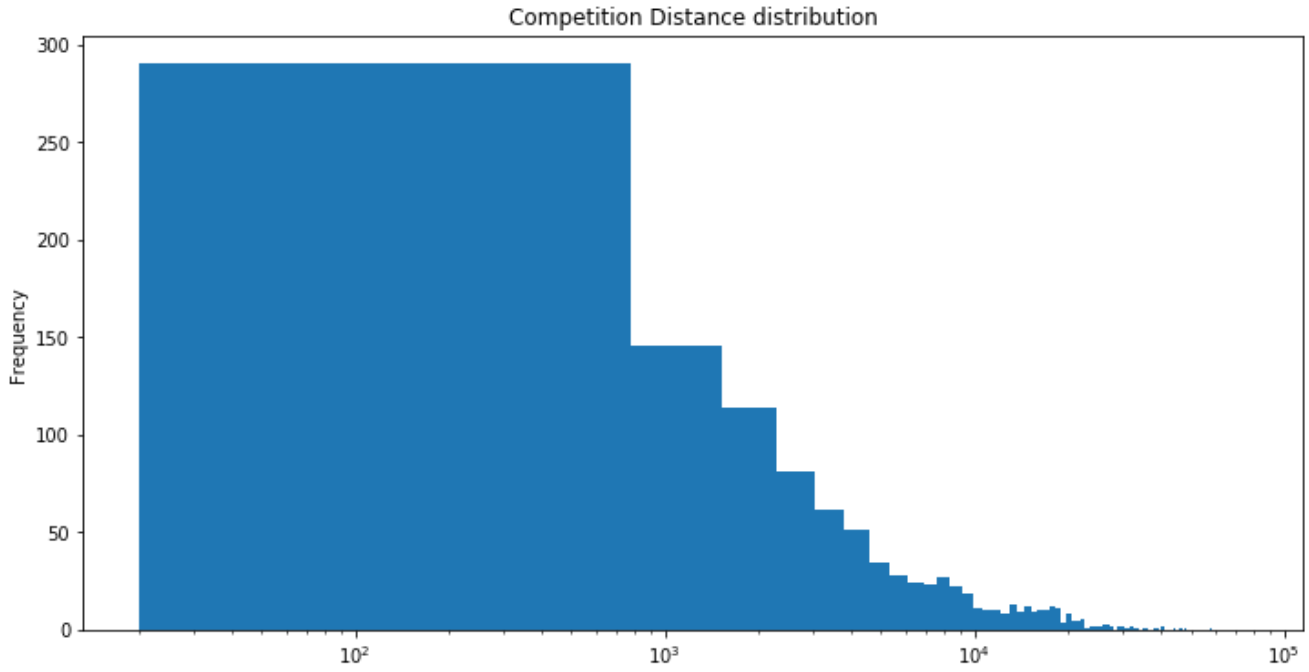
a 593

c 513

b 9

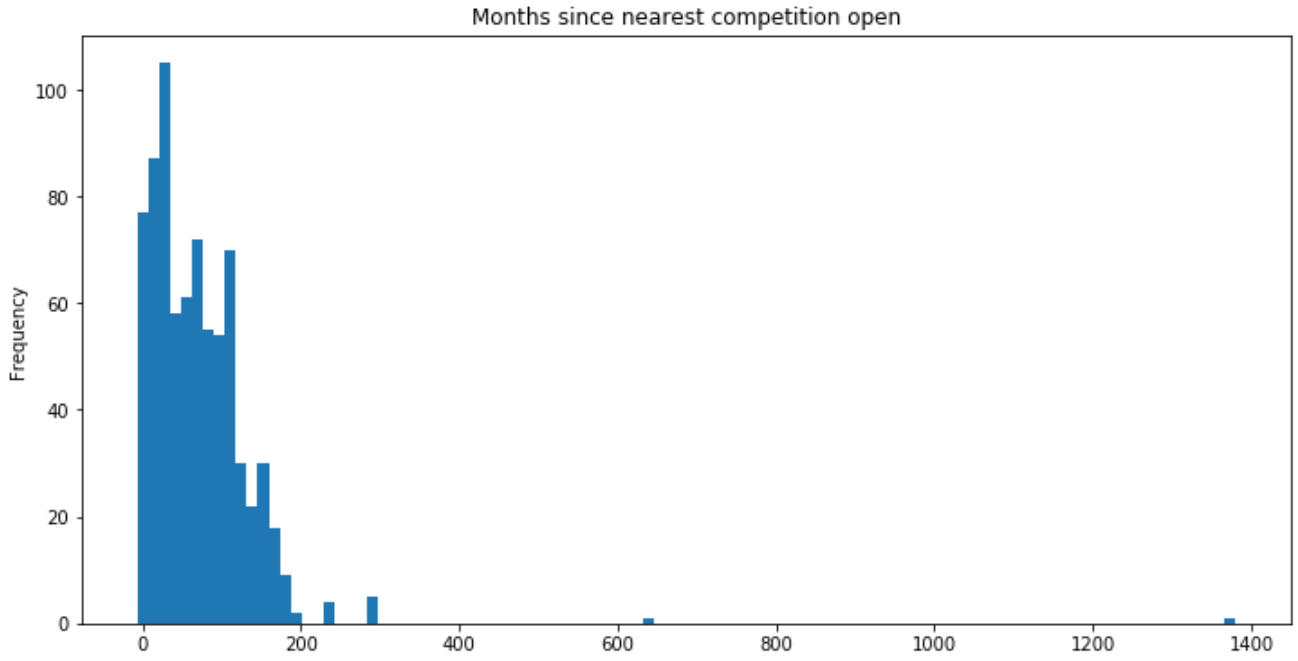
- 1115 个商店中有 571 个商店参与持续促销，有 513 个不参与，差不多各参一半

竞争对手距离分布图



从上图中可以看出绝大部分竞争对手都在商店附近，我推测这一现象的原因是大部分商店都集中在大城市的商业中心，所以不是因为竞争对手故意将商店开在附近，而是所有商店都优先选择开设在人流量大的地段。竞争对手距离是数据中少有的与空间维度有关的数据。

竞争对手开业时间分布图

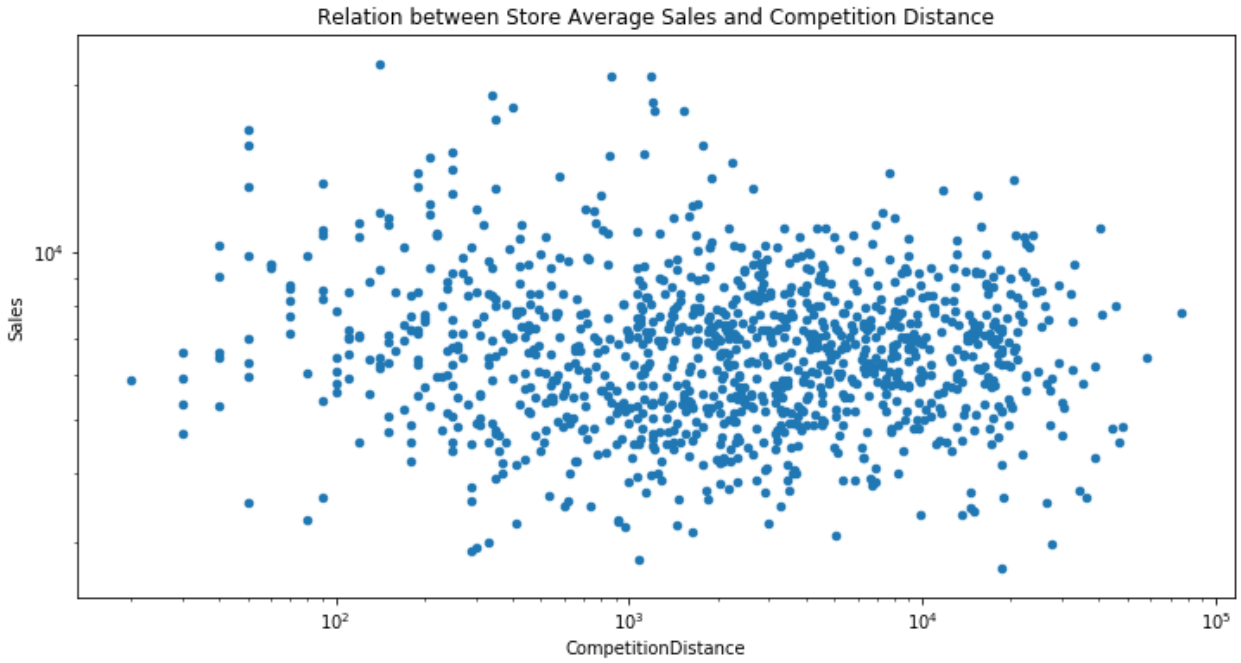


不妨设定截止日期为 2015-08-01，从上图中我们不难得出:

- 有 354 项缺失数据
- 开业时间处于 2012 年至 2013 年的较多
- 绝大部分竞争对手都在十五年之内开业

销售额与竞争对手距离的关系

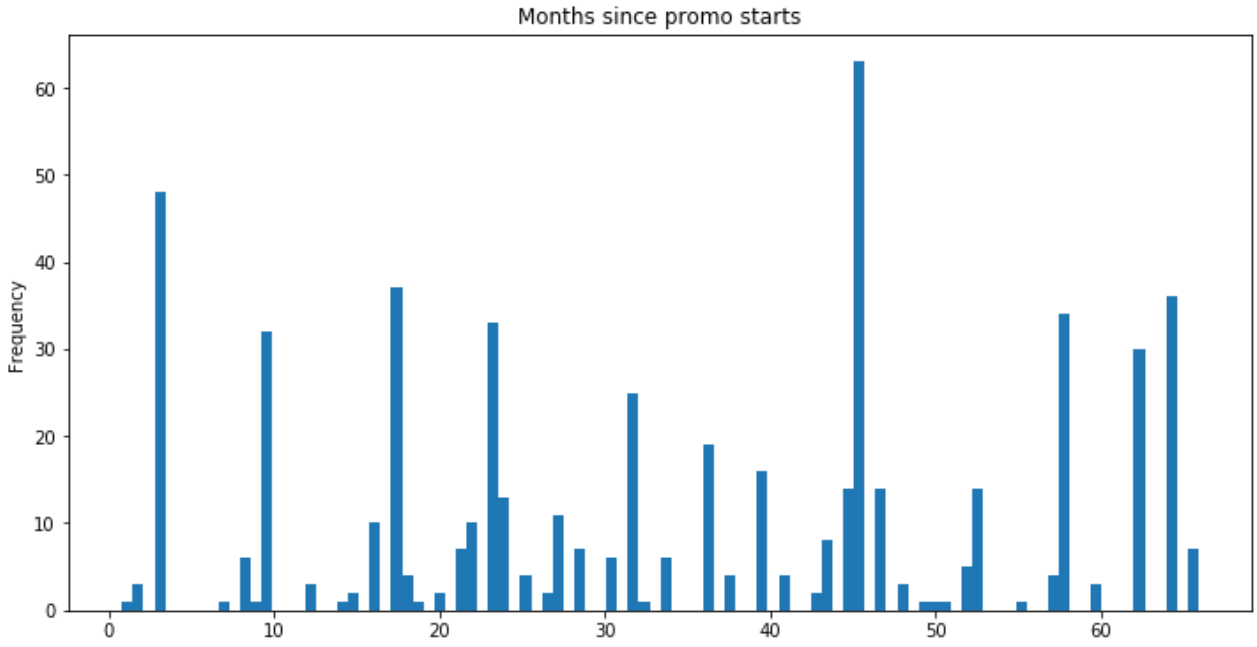
下面为竞争对手距离（横轴）与销售额（纵轴）的关系点阵图：



上图体现的情况有一些反直觉，距离竞争对手近一些的销售额要偏高，这里说明地理位置的影响更重要，因为好的地理位置带来高的销售额，无论竞争对手的多少。

持续促销策略时长分布

下面为促销持续时间（按月）的分布图，纵轴为频率

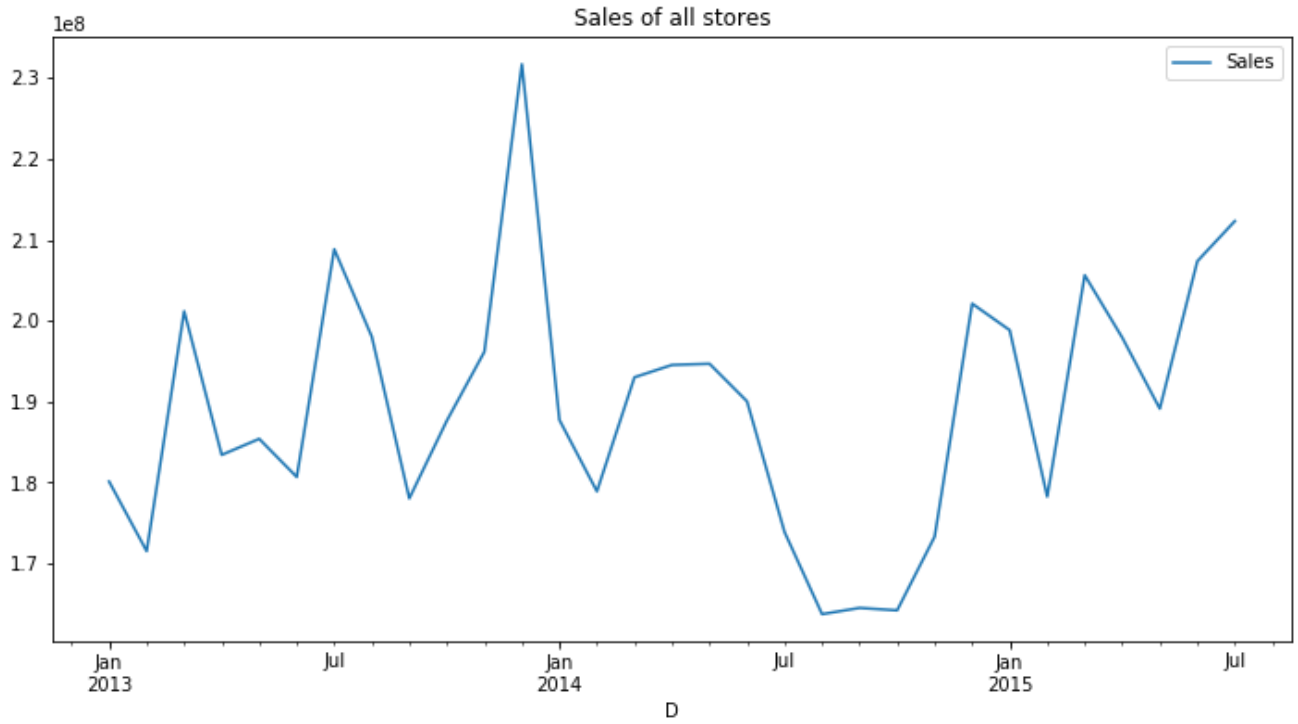


上图显示持续促销策略的时长分布没有明显规律。另外值得一提的是，持续促销

策略是有周期性的，周期为三个月，不同的商店促销开始时间不同。

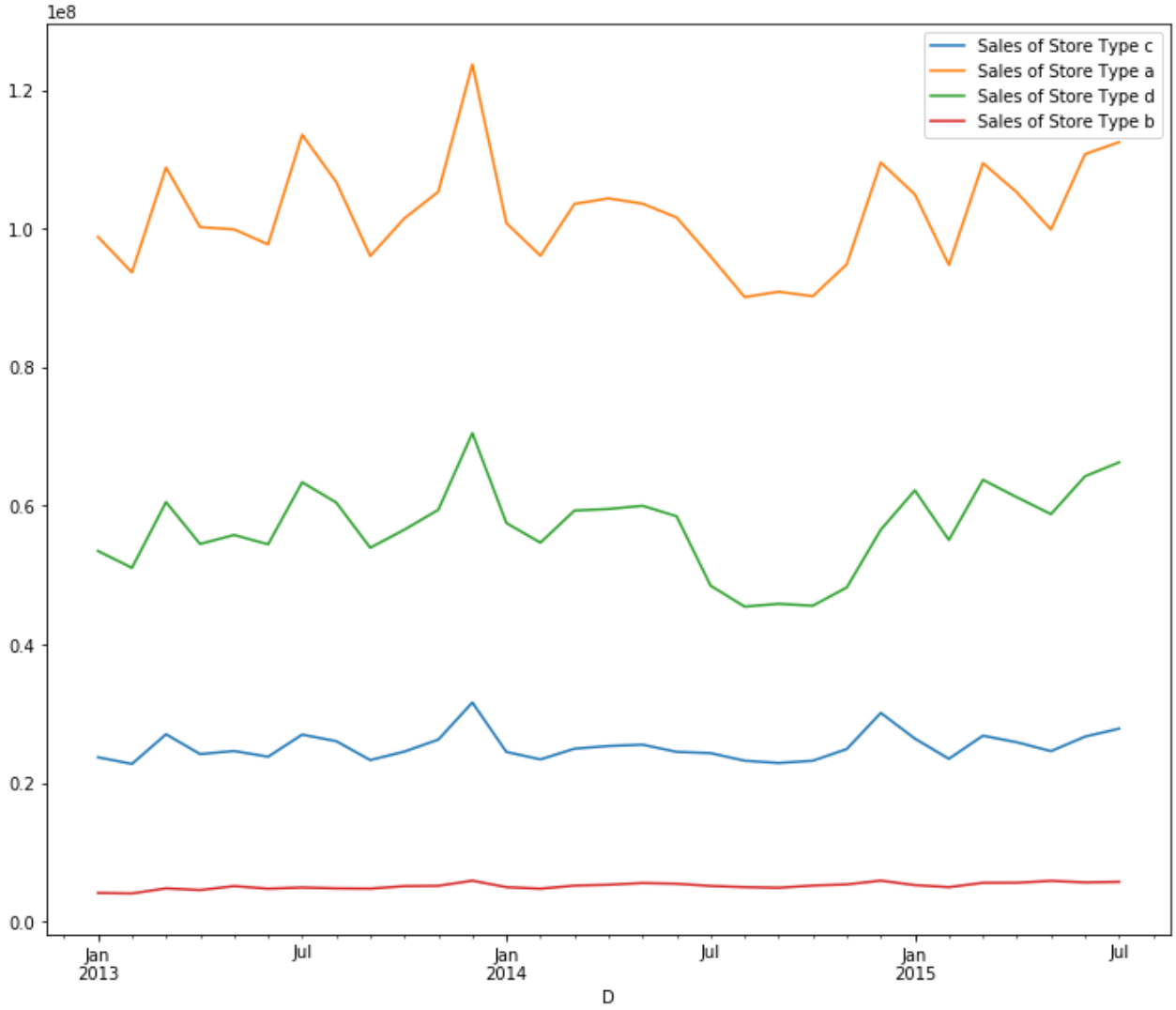
促销活动通常对销售有直接影响，应该考虑对促销策略的周期性进行进一步的特征处理。

商店在时间维度上的销售曲线



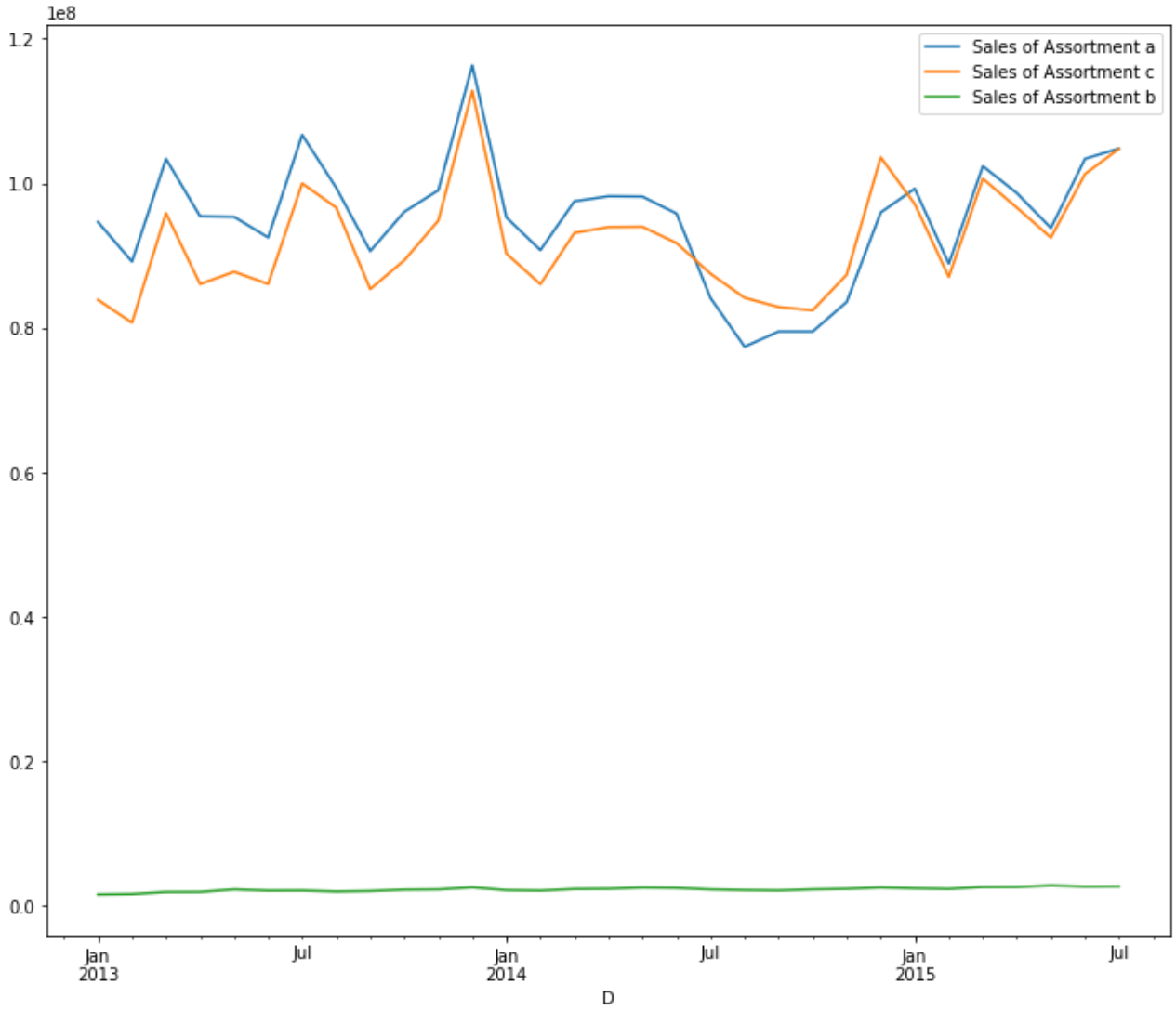
从图中可以大致看出销售额在时间上有明显的周期波动，猜测有潜在的季节性，后面进一步分析季节性。

不同类型商店在时间维度上的销售曲线



从上图中能看出来不同的商店类型的销售额所在区间完全不同，因此商店类型是一个影响销售额的重要因素。

不同品类在时间维度上的销售曲线



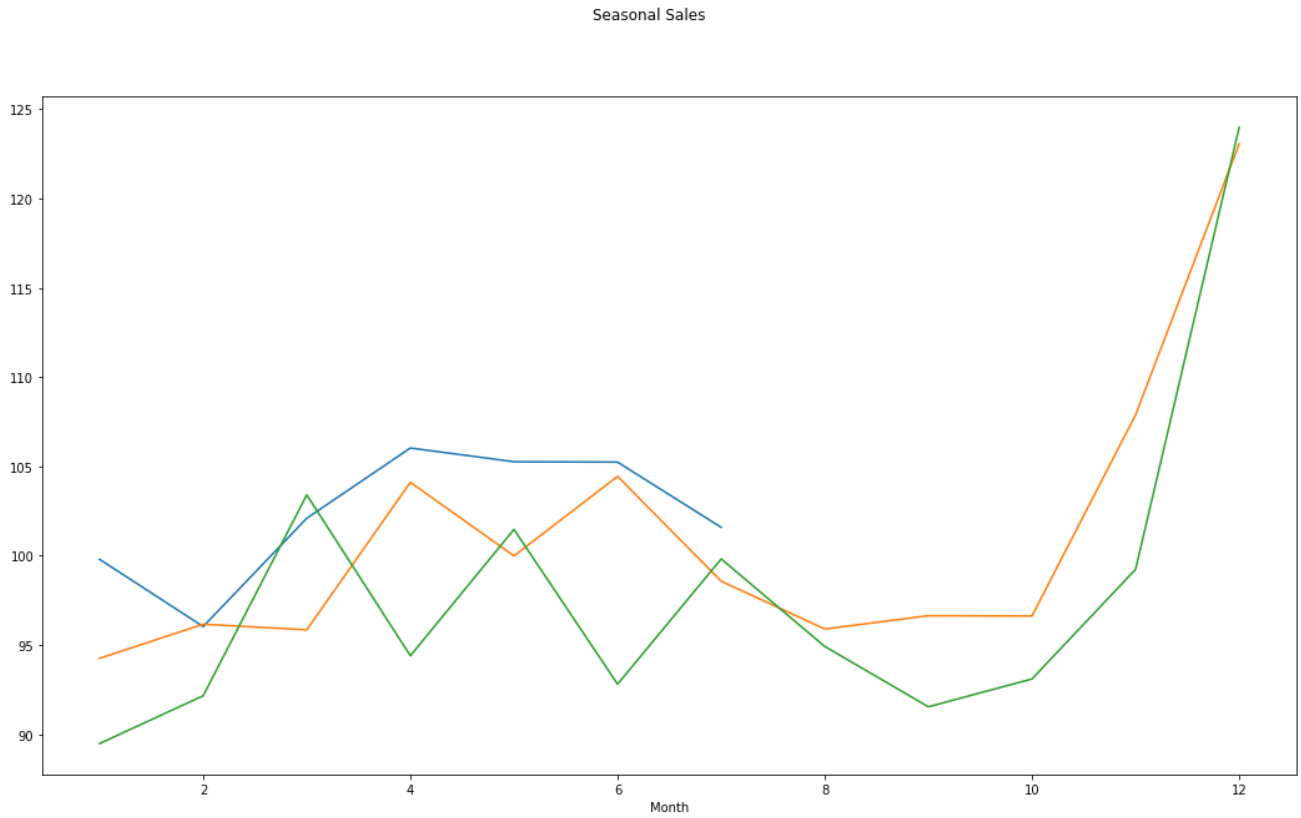
同理，不同的品类对销售额也有较大影响。

季节性分析

为了让图示更能显示特性并且尽量消除不必要的干扰，我对销售额在时间维度上的曲线做了如下处理：

- 销售额用相对于商店平均销售额的百分数来表示

- 相对销售额对月份取平均数
- 目的是大幅减少缺失数据和商店关门造成的影响
- 不同的年份用不用的颜色表示，如下图所示



从以上图看出，销售额的季节性非常明显，例如：1，2 月的低销售额，11，12 月的高销售额是普遍现象。由此可以推断出月份以及日期都是影响销售的重要特征。

2.2. 算法和技术

2.2.1. 回归算法

根据前面的分析可知，销售预测本质上是一个回归问题。用于回归的算法有很多，例如线性回归（Linear Regression），脊回归（Ridge Regression），拉索回归（LASSO Regression），弹性网路（ElasticNet），K 近邻回归（KNN），分类回归树（CART），支持向量回归（SVR）。

线性回归是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析，它的假设条件是自变量满足高斯分布，并且自变量和因变量是相关的，而自变量之间不是高度相关的。

脊回归是线性回归算法的拓展，通过改进损失函数来最小化模型复杂度。复杂度通过相关系数的平方和来测量，即 L2 正则。

拉索回归，与脊回归类似，通过改进损失函数来最小化模型复杂度。复杂度通过相关系数的绝对值之和来测量，即 L1 正则。

弹性网路是一种正则化的回归，其组合了脊回归和拉索回归的属性，使用 L2 正则 和 L1 正则来惩罚模型，最小化模型复杂度。

K 近邻为新数据实例确定 K 个最相似的实例。K 个实例的均值或者中位数作为预测值。所谓最相似指的是距离新数据实例的距离最近，这个距离的定义很大程度上通过相关领域知识来获得。

分类回归树是基于树的回归算法，使用训练数据选择最优点去分类数据，使损失最小。

支持向量回归是用于解决二值分类问题的支持向量机（SVM）用于回归问题的拓展，基本原理是找到一个回归平面，让一个集合的所有数据到该平面的距离最近。

2.2.2. 模型的选择

为了更准确的预测商店的销售额，我们从以下几方面来考虑模型的选择：

理论模型

一个机器学习模型的性能好坏，主要与下面两个主要因素相关：

1. 训练误差，也就是模型在训练数据上的表现，误差越小，表现越好。
2. 模型的复杂度，通常体现为模型的自由度，模型的参数越多，自由度越大，模型的复杂度越高，也就会导致模型过拟合。

基于以上考虑，我们可以通过分析数据和项目本身特征来评估模型抗过拟合的能力来初步筛选模型，然后通过训练误差的大小来进一步筛选模型。

实际项目与数据

我们知道，Rossmann 销售数据来源于真实世界，真实世界的数据中往往含有不少噪音，从抗噪音的角度，我们首先排除 K 近邻，因为它的抗噪音能力较差，容易过拟合。

其次通过上面可视化分析，我们可以看出特征的类型是相当丰富的，其中包括不同范围的数值类型和很多类别型特征。我们可以排除支持向量回归，因为它对多样性的数据容易过拟合。

由于我们暂时不能排除销售额与其中一些特征之间有较强的线性相关性，线性回归作为一个复杂度低，容易实现的回归算法，我们应当保留。

基于树的算法对多样性的数据有很好的支持，如果训练过程使用集合树（tree ensemble)的方法，更可以降低过拟合的风险，具有很好的抗噪音能力。常用的集

合树有随机森林（Random Forest）和增强树（Boosting Tree)。

模型的实现

一个良好的模型除了具备理论优势和匹配具体项目之外，还需要有以下工程方面的条件：

- 可以快速验证模型效果和模型参数
- 灵活性好，可以深度定制
- 可扩展性好，可以适应不同量级的数据量

综合上面分析，我们可以选择线性回归，随机森林以及具有代表性的 XGBoost 这三个经典模型。下面分别对这三种模型的优缺点进行分析。

线性回归算法

线性回归算法的原理众所周知，这里不再赘述，下面来分析一下优缺点：

- **优点**
  - 线性回归的理解和解释都很直观
  - 简单性使得线性回归的实现和更新的成本可控
  - 线性回归容易与其它方法结合以提升效果，例如，通过正则化避免过拟合，通过随机梯度下降来更新模型

- **缺点**
  - 线性回归的前提假设是线性相关 ,而真实世界的很大部分数据是非线性的
  - 模型过于简单，难以拟合含噪音的真实数据

随机森林算法

随机森林以决策树为基础学习器构建引导（Bagging）集成，进一步在决策树的训练过程中引入了随机特征选择，因此可以概括随机森林包括四个部分：

1、随机选择样本（有放回抽样）

有放回的随机选择样本指的是在训练决策树之前，先从样本中按照一定占比随机选出训练数据，下一次训练仍然从样本中随机抽选训练数据，就好比每次训练完成后都将抽样放回样本一样。

2、随机选择特征

随机选择特征是指在决策树的构建中，会从样本的特征集合中随机选择部分特征，然后再从这个子集中选择最优的属性用于划分，这种随机性导致随机森林的偏

差会有稍微的增加，但是由于随机森林的‘平均’特性，会使得它的方差减小，而且方差的减小补偿了偏差的增大，因此总体而言是较好的模型。

3、构建决策树

构建随机森林的决策树时，每棵决策树都最大可能的进行生长而不进行剪枝（pruning）。

4、随机森林平均

在对预测输出时，随机森林通常使用简单平均法。

下面分析优缺点：

- **优点**
  - 对不同类型数据预测准确度的综合性能较好
  - 能够评估特征的重要性
  - 能够处理高维数据
  - 随机森林的一大特性是在生成树的过程中可以对误差进行无偏估计，即‘包外估计’
  - 如上所述，随机森林通常有较小的方差
  - 随机不需要做特征正规化，降低了部分特征工程的工作量【参考文献 15】

- 容易实现并行计算

- **缺点**
  - 随机森林对噪声较大的数据容易过拟合，对较大偏差的抵抗力不行
  - 由于构建树的复杂性，即便可以并行计算，训练时间仍然相对较长

XGBoost 算法

XGBoost 是 “Extreme Gradient Boosting” 的缩写，其中 “Gradient Boosting” 一词在论文 Greedy Function Approximation: A Gradient Boosting Machine 中，由 Friedman 提出。 XGBoost 基于这个原始模型。

XGBoost 与随机森林类似，是基于决策树的，它与随机森林最大的区别在于训练方式。随机森林中的每一个棵树是独立选择数据和选择特征并进行训练的，而 XGBoost 的梯度提升方法是每一次基于前一棵树进行优化后的到下一颗树，每次提升一点，每提升一点新建一棵树，预测时使用的平均策略与随机森林的策略类似。XGBoost 的另外一个特点是目标函数包含了结构复杂度的部分，所以构建树的过程会进行剪枝。

在原理上，XGBoost 与标准的 Gradient Boosting 算法没有本质区别，值得指出的是，XGBoost 在算法的实现上引入了很多工程方面的优化思想，使得这个模型在落地实现的时候具备了很多独有的有优势。

下面来分析一下 XGBoost 的优缺点：

- **优点**
  - 与随机森林一样，XGBoost 具备集合树的优势：
    - 对不同类型数据预测准确度的综合性能较好
    - 能够处理高维数据
    - 能够评估特征重要性
  - Boosting 的特性使得 XGBoost 能够快速修正偏差
  - XGBoost 还有一些独有优势：
    - 很好的可扩展性，目前 XGBoost 已经能在很多大规模集群上训练，例如 Spark 或者谷歌云
    - 很好的灵活性，可以深度定制每一个子回归器
    - XGBoost 不需要做特征正规化，降低了部分特征工程的工作量【参考文献 14】
    - 很好的可移植性，XGBoost 有丰富的各类平台支持
    - XGBoost 在工程上有经过验证的系统优化，它的准确性和性能都在学术界和工业界得到认可
- **缺点**
  - 由于每棵树之间的关联性，XGBoost 对高方差的数据表现不理想
  - 与随机森林一般，训练时间相对较长，需要足够的算力资源支持

2.2.3. 技术

本项目的代码我将会用 Python3 语言来实现，其中主要用到的库和框架有：

a. sklearn（http://scikit-learn.org/stable/）：机器学习算法以及训练过程

框架

b. xgboost（https://xgboost.readthedocs.io/en/latest/）：XGBoost 实现
c. pandas（https://pandas.pydata.org/）：数据分析工具箱
d. numpy（http://www.numpy.org/）：科学运算工具库
e. matplotlib（https://matplotlib.org/）：可视化工具箱
f. seaborn（https://seaborn.pydata.org/）：可视化工具箱
g. jupyter (http://jupyter.org/)：基于 Web 的交互式编程平台
h. Google AI Platform (https://cloud.google.com/ml-engine/docs/): 谷歌云端 AI 平台，支持在线训练，利用高性能计算机提升运算速度
i. Test of Grubbs: (https://pypi.org/project/outlier_utils/)：异常值检验
j. sqlite (https://www.sqlite.org/index.html): 便携数据库

2.3. 基准指标

2015 年的这次比赛中第 30 名(Someperson53)的成绩为 0.11108 ,第一名(Gert ）成绩为 0.10021，本项目的基准指标阈值设定为至少进入前 30 名，即

RMSPE <= 0.11108

3. 实现方法

3.1. 数据预处理与特征工程

经过数据探索分析，特征工程主要包含以下要点：

- 0 销售额为噪音，但是提供如装修等重要事件信息，应该保留数据，但是同时设置为异常值，并进一步处理

- 销售额（Sales）数据中存在异常值，需要进行处理；
    1. 首先利用 Grubbs 检验来识别异常值【参考文献 16】
    2. 利用基于商店的中值对异常销售额进行重新估计，替代原来数值
    3. 然后转化为对数坐标系（SalesLog），并以对数值作为预测变量

- 根据商店的持续促销策略，生成与持续促销季节性相关的特征：
    1. MonthsSincePromo2：距离周期性持续促销开始的月数
    2. Promo2Weeks：距离第一次持续促销开始的周数

- 销售额的预测最大的特点是时序序列，应当把时间因素考虑进特征的选择和抽取，这里考虑下面三类时间相关特征：
    1. 季节性特征：包括
        - DayOfWeek: 一周中的第几日
        - DayOfMonth: 当月日期
        - Month: 月份
        - Year: 年份

- WeekOfYear: 一年中的第几周

2. 与时间相关的事件：例如假日，促销，竞争对手开业，商店装修等。对关键事件发生的前后两周时间点用泊松分布进行模拟，生成相应的特征，包括
    - SoonChristmas：圣诞节前，用 0 或者 1 表示
    - WasChristmas：圣诞节后，用 0 或者 1 表示
    - SoonRefurbishments：装修前，用 0 或者 1 表示
    - WasRefurbishments：装修后，用 0 或者 1 表示
    - SoonPromo：促销前，用 0 或者 1 表示
    - WasPromo：促销后，用 0 或者 1 表示
    - IsWeekend: 0 表示工作日，1 表示周末
    - HolidayNextWeek: 下周是否有节假日，用 0 或者 1 表示
    - HolidayLastWeek：上周是否有节假日，用 0 或者 1 表示
    - CompetitionOpenSince：竞争对手开业时长，按月算

3. 近期数据：对刚过去的一个季度，一年以及整个历史的统计信息，这些统计计量主要包括：
    - SalesPerCustomer：每个商店的平均单个顾客的销售额
    - RatioOnSchoolHoliday：假期销售额占总销售额的比值
    - RatioOnPromo：促销销售额占总销售额的比值
    - RatioOnSaturday：周六销售额占总销售额的比值
    - Avg3MonthsSales：季度平均销售额

        - Avg3MonthsCustomers：季度平均顾客量
        - AvgSchoolHoliday：假期平均销售额
        - AvgCustomersSchoolHoliday：假期平均顾客量
        - AvgYearCustomers：年平均顾客量
        - AvgYearSales：年平均销售额
        - AvgCustomers：综合平均顾客量
        - AvgSales：综合平均销售额
        - AvgPromo ：促销期平均销售额
        - AvgCustomersPromo：促销期平均顾客量

- 对类别型变量进行数字转换，其中包括：
    1. SchoolHoliday：学校假期，不做处理
    2. StateHoliday：法定假日，转化为数字类型
    3. StoreType：商店类型，转换为数字类型
    4. Assortment：品类，转换为数字类型
    5. Promo：是否促销，二元特征，不做处理

- 与空间相关的特征
    1. CompetitionDistance ：与竞争对手的距离，转换为对数坐标系

- 数据划分应当根据时间来划分：用最后六周的数据作为验证集

- 模型训练过程以及调优过程所得结果都记录在名为 model.db 的 sqlite 数据库中，以便前后对照分析

3.2. 实现

通过对线性回归，随机森林以及 XGBoost 分别建立模型，使用默认参数，三個模型在验证集上的表現分别如下：

模型	RMSPE
线性回归	0.20952
随机森林	0.13466
XGBoost	0.15391

线性回归的表現（0.2095）明显不如其它模型，可以排除销售额与自变量之间有很强的线性关系。我决定去除线性回归模型。在下面改进步骤，我只考虑随机森林和 XGBoost 的性能调优。

- 随机森林 调参

利用随机网格搜索对随机森林算法的参数进行微调，微调后验证集得分为 0.12324，在 Kaggle 上的得分为 0.13566，并不理想。在做进一步调优前，先来看看 XGBoost 的性能。

- XGBoost 调参

同样利用随机网格搜索对 XGBoost 算法的参数进行微调，微调后验证集得分为 0.11684，在 Kaggle 上的得分为 0.12617。效果明显优于随机森林。

- 特征筛选与模型选择

分别用调参后的随机森林和 XGBoost 分别对特征进行筛选，筛选过程如下：

1. 记录模型当前得分 X
2. 每次从现有特征列表中抽走一个特征 i
3. 训练模型并且通过验证集得到新的分数 Yi
4. 将特征放回列表，回到步骤 2，直到所有特征都检验过为止
5. 最终对特征按照 X-Yi 得分的从大到小排序
6. 选取前 5 个特征为候选特征

经验证后，对两个模型选出的候选特征列表做交集运算，仅得到一个特征：StateHoliday，虽然这其中的原因并不清楚，我决定暂时将 StateHoliday 从特征列表中剔除。

此时，筛选后的模型表现分别为

模型	验证集 RMSPE	Kaggle RMSPE
随机森林	0.12156	0.13498
XGBoost	0.11501	0.12192

虽然离目标还有些距离，但是性能有比较明显的提升。此时，随机森林的性能已经

明显落后于 XGBoost，后面的改进步骤将剔除随机森林，并且确定 XGBoost 作为最终模型选择。

3.3. 改进

- 补充数据

现有数据中大部分都与时间有关，与空间相关的数据很少。考虑到天气因素对销售额可能造成的影响，我尝试获得一些与天气相关的公开数据，以提升模型性能。经过验证和分析，我导入了【参考文献 17】讨论中所提供的天气数据，与天气相关的数据是按照各个药店所在的州统计的，其中有效的特征有：

1. Max_TemperatureC：当日最高温度
2. Precipitationmm：降雨量

模型参数不变，加入上面两个特征之后，分数为：

模型	验证集 RMSPE	Kaggle RMSPE
XGBoost	0.11011	0.11521

天气数据的补充对模型的提升非常明显。

- 对抗过拟合

进一步分析此时模型的表现：

模型	训练集	验证集 RMSPE	Kaggle 公开集 (39%) RMSPE	Kaggle 隐藏集 (61%) RMSPE
XGBoost	0.90256	0.11011	0.11313	0.11521

由于模型得分离目标不远，我判断可以通过微调来达到目标，从目前在不同数据集上的表现来看，模型的泛化能力还值得提升，于是我调整了 XGBoost 下面两个参数来削弱过拟合造成的影响：

- 经过多轮验证，将参数 reg_lambda 从 100 调整到 230
- 经过多轮验证，将参数 gamma 从 0.5 调整到 0.54

调整后的表现为：

模型	训练集	验证集 RMSPE	Kaggle 公开集 (39%) RMSPE	Kaggle 隐藏集 (61%) RMSPE
XGBoost	0.100251	0.10612	0.11148	0.11451

微调后的模型最终成绩有一些提升，而在验证集和公开集上的表现有明显提升，过拟合的影响有所削弱。此时我注意到，模型的表现随随即参数（random_state）的变化有比较明显的波动，接下来考虑利用集成模型来降低这种随机波动造成的影响。

- 集成模型

我的基本思路是：

1. random_state 的初始值设为 767，每次加 1，迭代 99 次，终止值为 866
2. 对应每个 random_state 的值，训练一个 xgboost 模型，总共 99 个模型

模型

3. 将每个模型所得结果提交 Kaggle，获得分数 Si
4. 将 99 个模个模型按照分数排序
5. 分别选取前 5，10，20，30，40 个模型作为不同的模型集合
6. 对于每个模型集合，利用单个模型分别进行结果预测
7. 每个集合模型的最终预测结果为集合内所有单个模型预测结果的加权平均，权重为 1 – Si
8. 按照集合模型的 Kaggle 得分，挑选出优胜的集合模型组合

按照这个思路我得到 99 个模型和它们对应的得分，下面是排名前 50 的模型：

编号	得分
835	0.11353
846	0.11401
792	0.11413
779	0.11416
853	0.11420
800	0.11422
778	0.11435
789	0.11436
832	0.11440

856	0.11445
852	0.11449
814	0.11451
841	0.11451
845	0.11452
768	0.11456
857	0.11460
812	0.11462
831	0.11472
858	0.11472
808	0.11473
849	0.11475
859	0.11475
787	0.11476
809	0.11479
776	0.11484
819	0.11484
791	0.11485
782	0.11486
827	0.11487
799	0.11491
826	0.11492

788	0.11505
772	0.11507
854	0.11507
817	0.11508
818	0.11508
842	0.11511
861	0.11512
824	0.11521
796	0.11522
801	0.11523
774	0.11524
823	0.11524
783	0.11525
805	0.11525
806	0.11526
821	0.11526
829	0.11528
769	0.11529
804	0.11529

基于以上原理组合的集成模型的得分分别为：

集成模型	集合大小	Kaggle 公开得分	Kaggle 隐藏得分 (最终成绩)
model_7-ensemble-top-40	40	0.11117	0.11128
model_7-ensemble-top-30	30	0.11136	0.11123
model_7-ensemble-top-20	20	0.11126	0.11115
model_7-ensemble-top-10	10	0.11135	0.11106
model_7-ensemble-top-5	5	0.11128	0.11119

可以选择排名前 10 附近的集成模型作为最终预测模型。

另外，在筛选过程中，我刻意将 StateHoliday 这个特征重新加入训练，结果依然是带来负面效果，我猜测是在测试集中含有 StateHoliday 的数据极少，因此在训练过程中这部分信息反而成为了噪音，而不是信号。

- 提升训练速度：

值得一提的是，在整个训练和优化过程中，由于模型选择和模型调参的过程需要耗费大量的运算资源，在我的个人电脑上（处理器 i7x2 2.8GHz，内存 16GB）这些过程通常需要超过 10 个小时的运算时间，为了提升运算速度，并且能让模型训练的同时我能够使用个人电脑进行其它工作，我提高了 XGBoost 训练的并行度参数 n_jobs，然后利用 Google 云提供的 API 将训练过程部署到 Google 云端 AI 平台执行，模型调参和训练时间从 11.6 个小时降到了 2.3 个小时，明显的缩短了项目推进时间，并且使我的个人电脑闲置出来，让我能够持续完成特征工程和数据分析验证等工作。

4. 结果

4.1. 模型评价与验证

为了选择最终模型，我在排名前 10 附近的不同组合，选择并验证了下面 5 个集成模型：

集成模型	集合大小	Kaggle 公开得分	Kaggle 隐藏得分 (最终成绩)
model_7-ensemble-top-8	8	0.11091	0.11108
model_7-ensemble-top-9	9	0.11115	0.11106
model_7-ensemble-top-10	10	0.11135	0.11106
model_7-ensemble-top-11	11	0.11155	0.11108
model_7-ensemble-top-12	12	0.11126	0.11114

最终胜出的集合模型是 model_7-ensemble-top-9， 最终得分为 **0.11106**，满足基准阈值：

0.11106 <= 0.11108

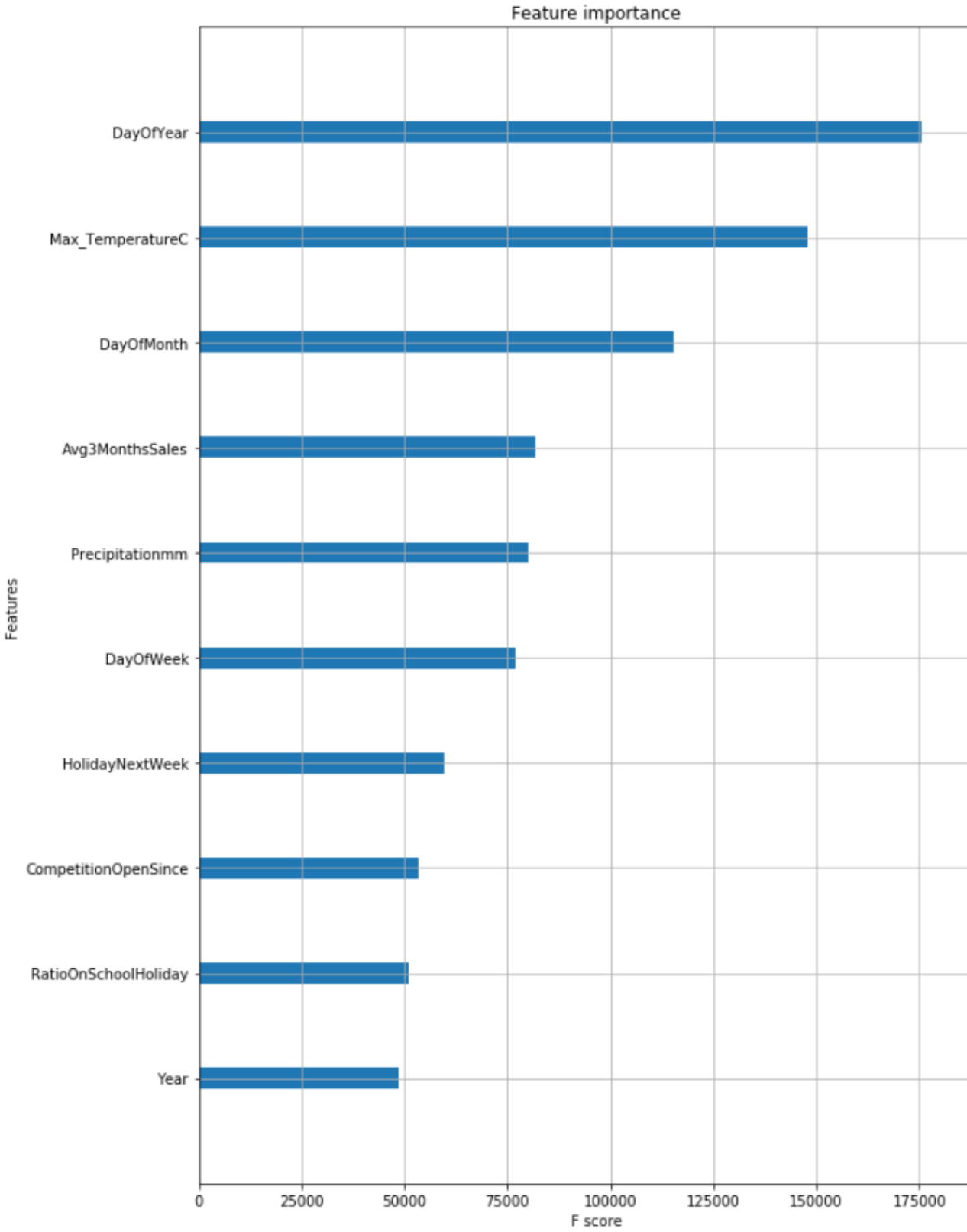
4.2. 结果分析

从上面的几个集合模型表现上来看，综合成绩最优的是 model_7-ensemble-top-8，不过 Kaggle 的比赛排名最终是根据隐藏得分来计算的，按照这个规则我选择了 model_7-ensemble-top-9。在真实应用中，我们应该选择 model_7-ensemble-top-8。不过，从实际的角度出发，这两个模型之间的差异已经非常微小，并没有本质的区别，对于商店预测未来 6 周的销售额，11%的误差是可以接受的范围。

5. 结论

5.1. 总结与思考

根据最终模型对特征的重要性排序如下图：



与我们对商店运营的常识相符，药店的经营受环境的因素影响很大，在所有因素里面，季节性的影响是最明显的，其中包括 DayOfYear，DayOfMonth，

DayOfWeek，其次与天气相关的信息也非常重要，这部分信息的填补对后期模型的微调起到了关键作用。除此之外与假期相关的事件和竞争环境相关的信息也对药店的销售有较大的影响。从现有数据的情况来看，绝大部分有效信息都是在时间维度上的特性，然而与商店相关的在空间维度上的信息非常稀少，天气信息在某一程度上间接补充了空间维度上的特征。回到真实的业务场景，在现有信息的基础上如果能收集更全面的空间维度上的特征，例如商店附近的人口密度分布，人口流动情况，所在城市规模等信息，应该能够更进一步降低预测误差，提升预测的准确率。

5.2. 后续改进

竞赛中第一名的模型得分达到了接近 10%的误差 ,比我的模型效果还要低了 1%，而且他并没有使用更多的额外信息，说明我的模型还没有充分挖掘出数据中的有用信息，对比之后，我发现他 【参考文献 3】 时间维度上的特征更丰富，另外他使用了另外一个模型先对顾客数量进行评估 ,再进一步预测销售额。这些思路都非常值得借鉴，因此还有进一步提升预测效果的空间。

6. 参考文献

1. https://www.kaggle.com/c/rossmann-store-sales
2. https://www.kaggle.com/thie1e/exploratory-analysis-rossmann
3. https://www.kaggle.com/c/rossmann-store-sales/discussion/18024
4. https://www.kaggle.com/abhilashawasthi/xgb-rossmann
5. https://xgboost.readthedocs.io/en/latest/tutorials/model.html

6. http://scikit-learn.org/stable/

7. https://xgboost.readthedocs.io/en/latest/

8. https://stackoverflow.com/questions/22354094/pythonic-way-of-detecting-outliers-in-one-dimensional-observation-data

9. https://pandas.pydata.org/

10. http://www.numpy.org/

11. https://matplotlib.org/

12. https://seaborn.pydata.org/

13. http://jupyter.org/

14. https://github.com/dmlc/xgboost/issues/357

15. https://stackoverflow.com/questions/8961586/do-i-need-to-normalize-or-scale-data-for-randomforest-r-package

16. https://en.wikipedia.org/wiki/Grubbs%27s_test_for_outliers

17. https://www.kaggle.com/c/rossmann-store-sales/discussion/17229