机器学习纳米学位

毕业项目 - Rossmann 销售预测

赵江柘 优达学城

2018年9月9日

I. 问题的定义

项目概述

该项目源自Kaggle的比赛Rossmann Store Sales。Rossmann是欧洲的一家大型连锁药店，Rossmann的经理希望能对各个门店未来六周的销售额进行预测，因为可靠的销售额预测能让经理提前进行合理的资源分配，从而提高资源的使用效率。门店的销售额会受很多因素的影响，该项目的目标就是要基于Rossmann各个门店的信息（比如促销、竞争、节假日、季节性和位置等）以及过往的销售情况来建模预测未来六周的销售额。项目使用到的数据集为1115个Rossmann门店的历史销售记录和这些门店的相关信息。

问题陈述

项目要求预测的是门店未来六周的销售额，所以按照机器学习对问题的分类方法，该项目属于回归问题，那么项目要完成的任务就是从所给的数据中提取出可能对销售额有影响的特征，建立有效的回归模型进行预测。具体来说，主要可以分为以下几步：

- 通过探索性数据分析（Exploratory Data Analysis）来尝试了解数据的一些基本情况，像数据的分布情况、缺失情况等，为后续步骤做准备和提供一些参考；
- 通过数据预处理（Preprocess the data）过程来处理诸如类别信息、缺省值、时间序列信息等，便于后续的特征提取；
- 利用提取到的特征建立回归模型，在建模的过程中可以尝试通过特征选择（Feature Selection）来争取达到比较好的预测效果。

最终期望达成的结果是，训练好的模型根据门店的相关信息（比如促销，竞争对手，位置等）和预测日当天以及前后一段时期的节假日等信息，能相对准确地对预测日的销售额进行预测。

评价指标

评价指标选用的是百分比均方根误差（the Root Mean Square Percentage Error(RMSPE)），计算方式如下：

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

其中y_i表示单个门店单天的销售额，$\widehat{y_i}$表示对应门店对应单天的销售额预测值，对于单个门店单天的销售额为0的情况不予评价。采用百分比误差可以有效降低大尺度的数据对最终误差的影响，对于门店的销售额数据来说，某些节假日或者某些特殊日子的销售额肯定比平常日要高出不少，如果采用均方根误差，那些很大的销售额数据就会对误差评估产生较大的影响，从而可能对模型好坏的评估产生误判。

II. 分析

2.1 数据的探索

项目提供的数据集中包含三个部分：train、test和store，train数据集中包含了各个门店的历史销售额数据，store数据集中包含了各个门店的一些补充信息，test数据集被用于测试和评估模型。下面对各个数据集做一个详细介绍。

2.1.1 数据的介绍

train数据集中一共包含1017209条数据记录，包含的数据域如下：

- Store: 门店的数字标识
- DayOfWeek: 周一到周日的数字标识
- Date: 日期，从2013-01-01到2015-07-31
- Sales: 单个门店在某个日期的销售额数据
- Customers: 单个门店在某个日期的顾客数
- Open: 指示门店是否开放（0表示关闭，1表示开放）
- Promo: 表示门店是否在进行促销（0表示否，1表示是）
- StateHoliday: 表示对应日期当天是否是法定假日（a表示是公共假日，b表示复活节，c表示圣诞节，0表示不是法定节日）
- SchoolHoliday: 表示对应日期当天是否是学校假日（0表示不是，1表示是）

store数据集中一共包含1115条数据记录，包含的数据域如下：

- Store: 门店的数字标识
- StoreType: 一共四种不同类型的门店（a,b,c,d）

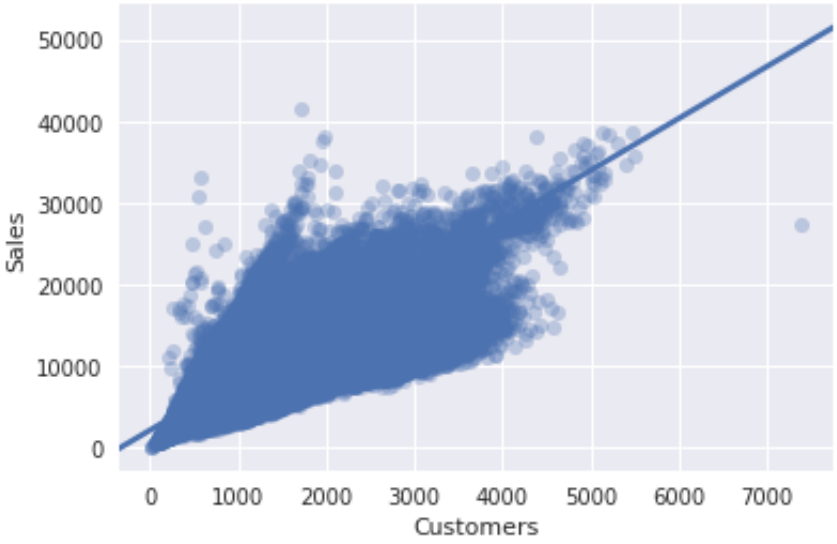
- Assortment: 描述门店上架的商品类型（a表示基本，b表示额外，c表示扩展）
- CompetitionDistance: 表示与最近的竞争对手门店的距离
- CompetitionOpenSinceMonth: 表示最近的竞争对手门店开业的月份
- CompetitionOpenSinceYear: 表示最近的竞争对手门店开业的年份
- Promo2: 表示一个门店是否进行了持续的促销（0表示否，1表示是）
- Promo2SinceWeek: 表示一个门店进行持续促销的开始的周数
- Promo2SinceYear: 表示一个门店进行持续促销的开始的年份
- PromoInterval: 表示门店开始促销的月份

test数据集包含41088条数据记录，数据域和train数据集一样，只是没给销售额数据和顾客数，额外的Id用于Kaggle上数据的提交。

store.csv	**train.csv**	**test.csv**
Store	Store	Store
StoreType	DayOfWeek	DayOfWeek
Assortment	Date	Date
CompetitionDistance	Sales	
CompetitionOpenSinceMonth	Customers	
CompetitionOpenSinceYear	Open	Open
Promo2	Promo	Promo
Promo2SinceWeek	StateHoliday	StateHoliday
Promo2SinceYear	SchoolHoliday	SchoolHoliday
PromoInterval		Id

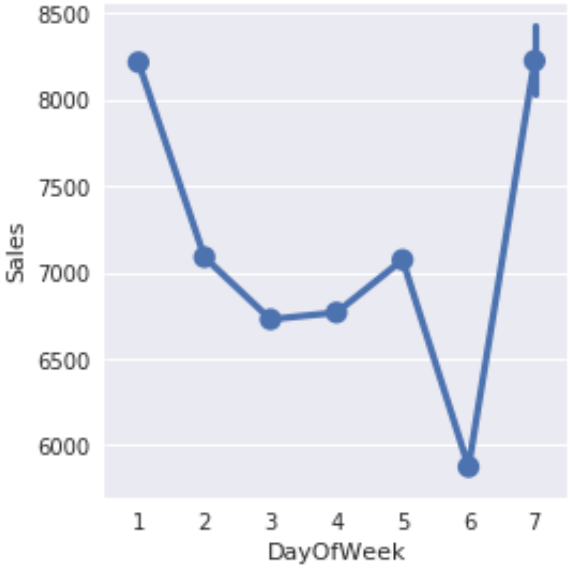
2.2 探索性可视化

2.2.1 Customers × Sales



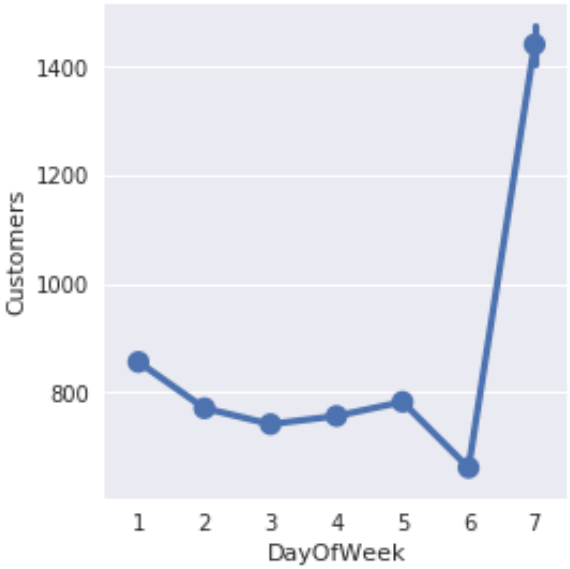
上图表达的是客流量(Customers)和销量(Sales)的关系，可以看出顾客数量和销量呈很强的正相关性。

2.2.2 DayOfWeek × Sales



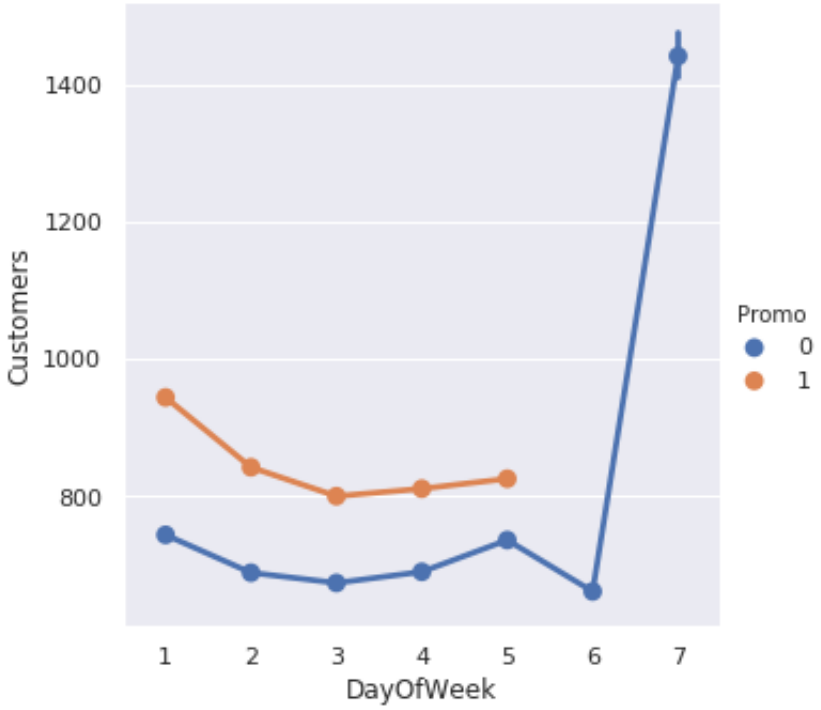
上图表达的是销量(Sales)随工作日和周末(DayOfWeek)的变化，由于国外大部分商店在周六周日会关门，所以对于在周日有购物需要的人来说，周日开门的店就会有较高的销量，所以在图中周日有较高的峰值。而周一店铺会重新开门，在周六日买不的东西会在周一去买，所以在图中周一也呈现较高的峰值。有意思的一点是周五出现小高峰，可能是因为人们在周末门店关门之前要把需要的东西买好。

2.2.3 DayOfWeek × Customres



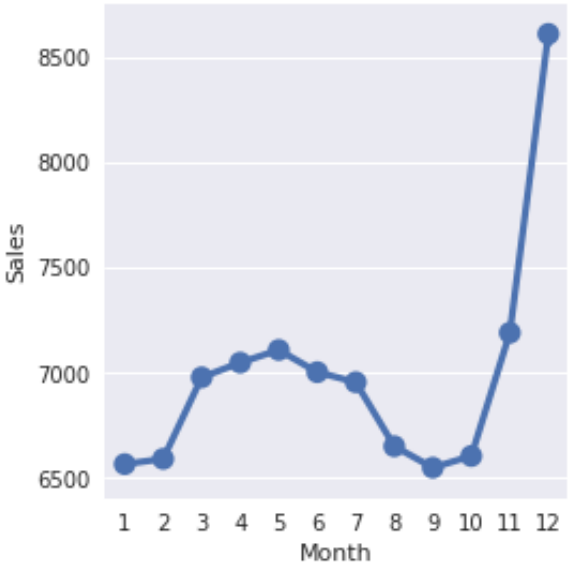
上图反应的是客流量(Customres)随工作日和周末(DayOfWeek)的变化,可以看出与DayOfWeek × Sales图中反应的规律基本一致。

2.2.4 Promo × DayOfWeek × Customres



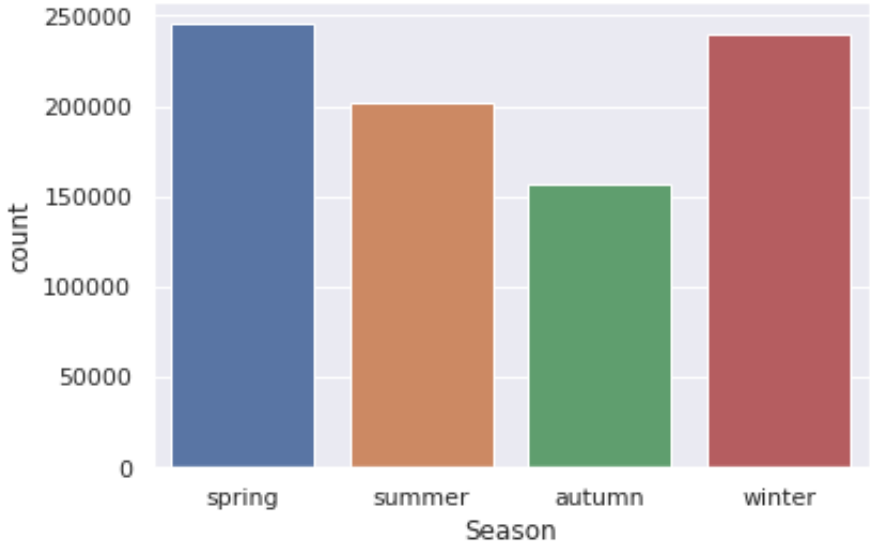
上图反应的是客流量(Customres)在是否有促销(Promo)的情况下,随工作日和周末(DayOfWeek)的变化,可以看出在有促销的情况下(Promo=1)的销量明显高于没有促销的情况(Promo=0),并且在周末商家不进行促销活动(Promo=0)。

2.2.5 Month × Sales



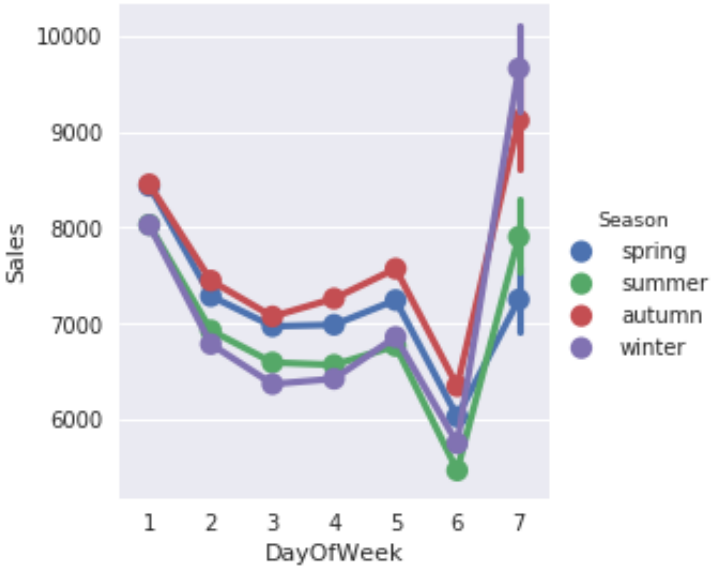
上图反应的是销量(Sales)随月(Month)的变化，可以看出在12月销量达到峰值，可能是由于圣诞节的原因，在1月销量骤然下降。这与国内的春节期间前后的消费活动非常相似。由于月份与季节相关，而季节与温度相关，温度的变化可能会影响人的消费欲望，基于这个发现，我建立"Season"特征。

2.2.6 Season



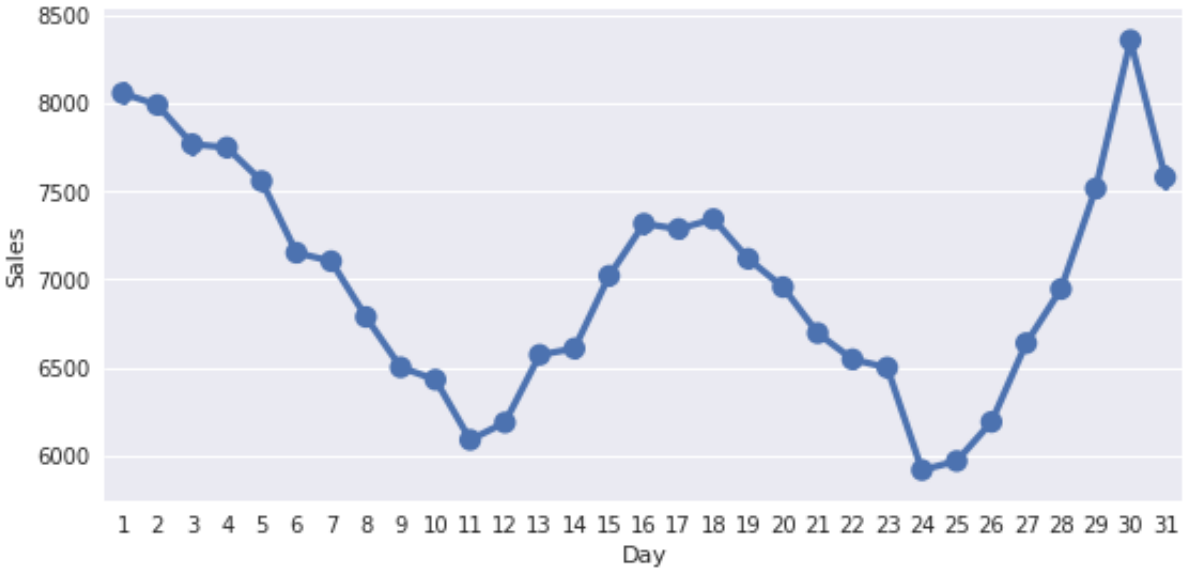
上图放映的是数据集中每个季节的数量，可以看出春天(spring)和冬天(winter)数量较多，秋天(autumn)数量最少。

2.2.7 Season × DayOfWeek × Sales



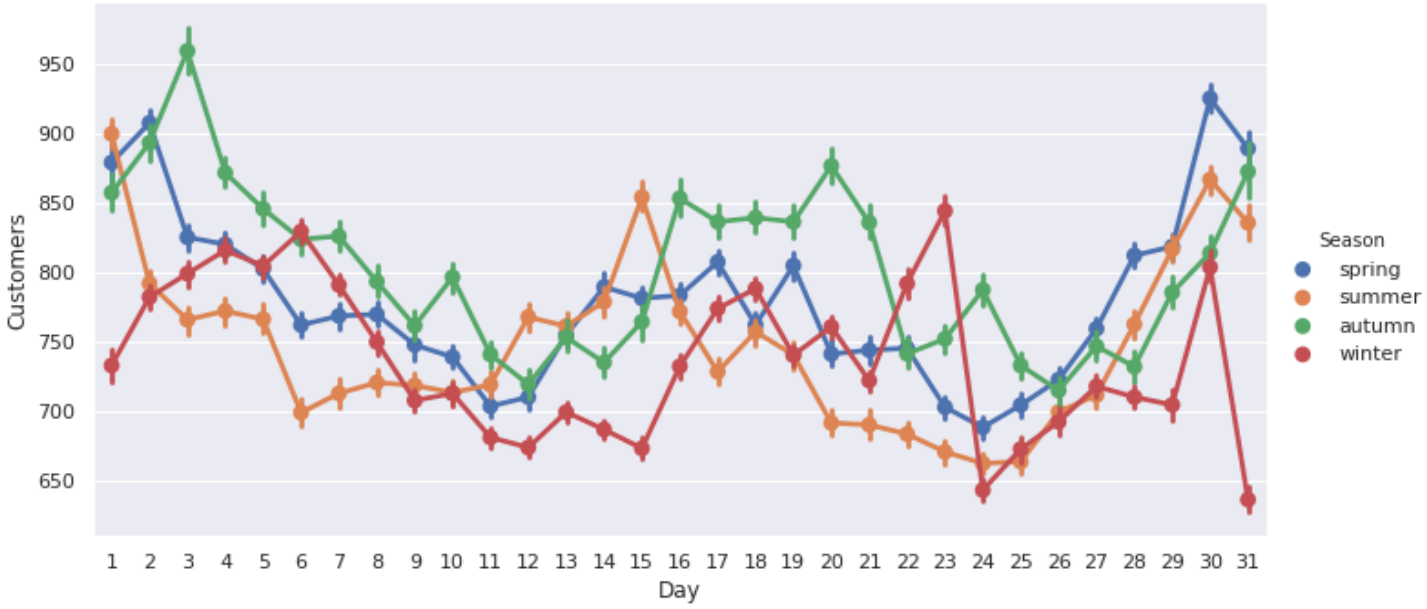
上图表达的是销量(Sales)在季节(Season)的区分下，随工作日和周末(DayOfWeek)的变化，可以看出虽然不同季节时，每周内的消费变化有区别，但每周的整体趋势相同。

2.2.8 Day × Sales



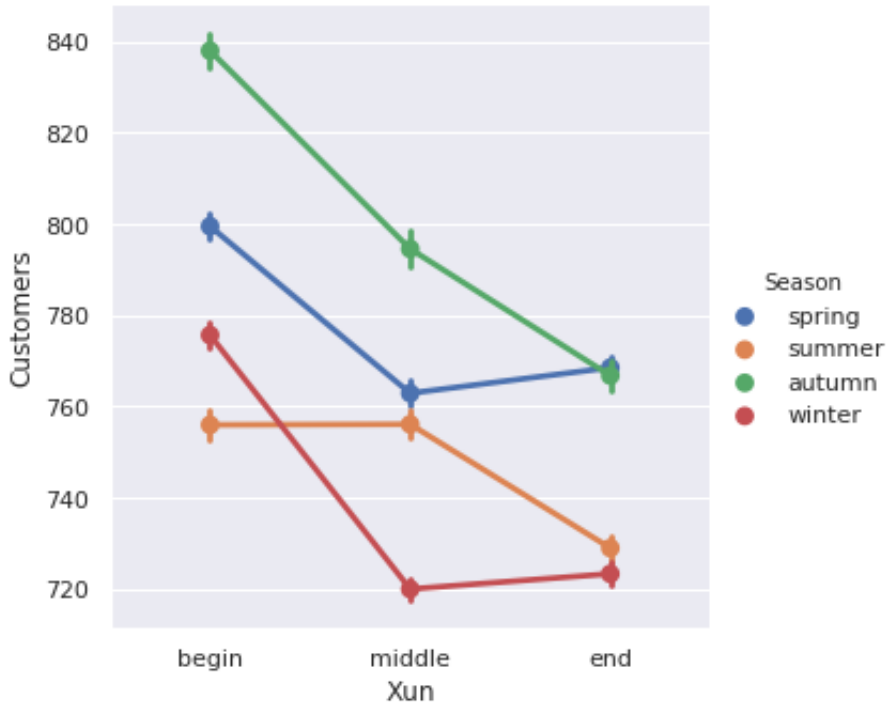
上图反应的是销量(Sales)随日(Day)的变化，从中可以看出明显的涨幅趋势，月初销量由高降低；在月中，由低到高再到低；月末，由低再次升高。基于这个发现，为了提取月内的变化趋势，我创建了Xun(旬)这个特征，希望从上旬，中旬，下旬的变化中发现一些规律。

2.2.9 Season × Day × Customres



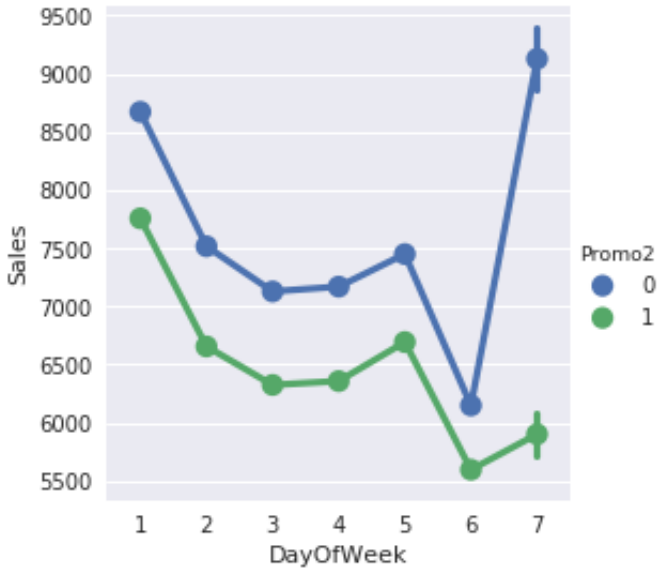
上图反应的是客流量(Customres)在不同季节(Season)下，随日(Day)的变化。可以看出在不同季节下，客流量(Customres)在月内的增长变化非常不同，为了捕捉总体趋势，结合Xun特征再观察一下。

2.2.10 Season × Xun × Customers



上图反应的是客流量(Customres)在不同季节(Season)下，随旬(Xun)的变化，可以看出春天客流量在上旬较多，在中旬降低，在下旬又升高；夏天上中旬客流量基本相同，在下旬降低；秋天天客流量在上旬较多，然后逐渐降低；冬天客流量上旬较多，中下旬降低，基本相同。总体客流量大小关系: 秋天 > 春天 > 夏天 > 冬天。

2.2.11 DayOfWeek × Sales × Promo2



上图表达的是销量(Sales)，在连续促销(Promo2)的情况下随工作日和周末(DayOfWeek)的变化,可以看出没有连续促销(Promo2 = 0)的情况反而大于有连续促销(Promo2 = 1)的情况，可能是因为实施连续促销的活动的店铺销量都偏低，才会推出连续促销的活动。

2.3 算法和技术

由前面的分析可知，这是一个回归问题。所以首先想到的是采用简单的线性回归模型进行拟合，观察拟合效果。鉴于给出的训练数据集中包含很多不同类型的特征数据，包括数值型、类别型等，于是尝试采用集成学习的回归模型就成了很自然的想法，常见的集成学习模型有Gradient Tree Boosting、Extreme Gradient Boosting（xgboost）等，而xgboost相比与其他Gradient Boosting的模型，速度更快，模型表现更好，所以该项目主要采用的就是基于xgboost的回归模型。下面对线性回归模型和Gradient Boosting方法做一个简单的介绍。

2.3.1 Linear Regression

给定数据集$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$，其中$x_i = (x_{i1}; x_{i2}; \ldots; x_{id}), y_i \in R$. 线性回归（linear regression）试图学得一个通过属性的线性组合来进行预测的函数。对于本项目来说，就是要从给定的数据集中提取特征，通过对这些特征的线性组合来预测销售额。线性回归使用最佳的拟合直线在因变量（销售额）和特征（自变量）之间建立一种关系。而最佳拟合直线的求解是通过最小二乘法来完成，在线性回归中，最小二乘法就是试图找到一条直线，使所有样本到直线上的欧氏距离之和最小。但是基于最小二乘的参数估计依赖于不同特征之间的独立性，希望不同特征之间的相关性越小越好，另外线性回归对异常值非常敏感，异常值会严重影响回归线，最终影响预测值。

2.3.2 Random Forest

随机森林是从原始训练样本集N中有放回地重复随机抽取k个样本生成新的训练样本集合，然后根据自助样本集生成k个分类树组成随机森林，新数据的分类结果按分类树投票多少形成的分数而定。其实质是对决策树算法的一种改进，将多个决策树合并在一起，每棵树的建立依赖于一个独立抽取的样品，森林中的每棵树具有相同的分布，分类误差取决于每一棵树的分类能力和它们之间的相关性。特征选择采用随机的方法去分裂每一个节点，然后比较不同情况下产生的误差。能够检测到的内在估计误差、分类能力和相关性决定选择特征的数目。单棵树的分类能力可能很小，但在随机产生大量的决策树后，一个测试样品可以通过每一棵树的分类结果经统计后选择最可能的分类。

在建立每一棵决策树的过程中，有两点需要注意采样与完全分裂。首先是两个随机采样的过程，random forest对输入的数据要进行行、列的采样。对于行采样，采用有放回的方式，也就是在采样得到的样本集合中，可能有重复的样本。假设输入样本为N个，那么采样的样本也为N个。这样使得在训练的时候，每一棵树的输入样本都不是全部的样本，使得相对不容易出现over-fitting。然后进行列采样，从M个feature中，选择m个（m << M）。之后就是对采样之后的数据使用完全分裂的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。一般很多的决策树算法都一个重要的步骤——剪枝，但是这里不这样干，由于之前的两个随机采样的过程保证了随机性，所以就算不剪枝，也不会出现over-fitting。

2.3.3 Gradient Boosting

Gradient Boosting（梯度提升）是一种集成弱学习模型的机器学习方法。机器学习模型主要的目标是得到一个模型F，使得预测值$\hat{y} = F(x)$与真实值y之间的误差尽可能小。Gradient Boosting采取分层学习的方法，通过m个步骤来得到最终模型F，其中第m步学习一个较弱的模型F_m，在第$m+1$步时，不直接优化F_m，而是学习一个基本模型$h(x)$，使得其拟合残差项$y - F_m$，这样就会使$m+1$步的模型预测值$F_m = F_m + h(x)$更接近于真实值y，因而目标变成了如何找到$h(x) = F_{m+1} - F_m$，最终就是要找到某类函数空间中的一组$h(x)$使得

$$F(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + const$$

Gradient Boosting算法的步骤如下：

对于给定的输入：训练数据集$\{(x_i, y_i)\}_i^n$，损失函数$L(y, F(x))$，迭代次数M。

1. 初始化模型为常数值：

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$

2. 迭代生成M个基学习器

i. 计算伪残差

$$r_{im} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)} \ for\ i = 1, \ldots, n.$$

ii. 基于$\{(x_i, r_{im})\}_{i=1}^{n}$生成基学习器$h_m(x)$

iii. 计算最优的γ_m

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

iv. 更新模型

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Gradient Boosting算法中最典型的基学习器是决策树。Gradient Boosting Decision Tree(GBDT)就是一种结合决策树的Gradient Boosting算法实现。这里的决策树是回归树，GBDT中决策树是个弱模型，树的深度和叶子节点的数量一般都较小。

另外一种很常用的Gradient Boosting算法的实现是XGBoost，XGBoost是Gradient Boosting算法的一种高效实现，以其出众的效率和较高的预测准确度在Kaggle比赛中崭露头角。

2.3.4 技术

项目中将会先使用开源的Python机器学习库scikit-learn中的线性回归模型 `LinearRegression` 简单看下效果，之后会主要使用XGBoost方法。scikit-learn是一个非常常用的机器学习方面的Python库，提供了大量监督学习和无监督学习算法，调用起来十分方便。XGBoost是Gradient Boosting Machine的一个C++实现，最大的特点在于它能够自动利用CPU的多线程进行并行，同时在算法上加以改进提高了精度。为了方便使用，作者陈天奇将XGBoost封装成了Python库。

传统GBDT在优化时只用到一阶导数信息，XGBoost则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数。XGBoost在代价函数中加入了正则项，用于控制模型的复杂度，学习出来的模型更加简单，防止过拟合。XGBoost在进行完一次迭代后，会将叶子节点的权重乘上缩减（shrinkage）系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。

基准模型

通过这个课程了解到,为构建RMSPE的baseline,

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

best constant(即α)取值应为目标值的加权平均(weighted target mean)。其中权重的取值为：

$$w_i = \frac{\sum_{j=1}^{N} \frac{1}{y_j}}{y_i}$$

通过在验证集上进行运算后得出baseline: 0.40737

考虑到我们将要尝试3个以上的模型，我期望一个模型具有良好的RMSE分数，注意在RMSE的情况下越低越好，0是完美的分数。

对销售有一个好主意，商店经理需要对模型的准确性有一定的信念。

可以接受一定的错误率，但如果误差几乎达到五分之一，那么该模型是不错的。 有经验的店经理将能够预测那么多错误率。 如果模型没有正确地预测至少五分之一的数据，则数据需要更多的处理，并且所选择的模型需要被优化或改变。 因此，RMSE的基准应为0.20

III. 方法

数据预处理

缺失值处理

通过对train和store数据集进行探索，发现train数据集没有缺失值，而store数据集中 `CompetitionDistance` , `CompetitionOpenSinceMonth` , `CompetitionOpenSinceYear` , `Promo2SinceWeek` , `Promo2SinceYear` , `PromoInterval` 存在缺失值。

下面详细介绍每个缺失值的处理

- CompetitionDistance

总共有3个缺失值，通过观察这3条数据，发现 `CompetitionOpenSinceMonth` , `CompetitionOpenSinceYear` 同时缺失，可以认为缺失的原因是随机缺失。通过观察 `CompetitionDistance` 的分布，呈现right skewed,所以用中位数填充缺失值。

- CompetitionOpenSinceMonth / CompetitionOpenSinceYear

`CompetitionOpenSinceMonth` 和 `CompetitionOpenSinceYear` 都有354个缺失值，可能其中一个确实另一个也会缺失，所以放在一起分析。

通过观察这些缺失数据后，并没有发现什么特别的原因，可能只是因为开店年和月无法收集到，

属于随机缺失。决定用0填补。

- Promo2SinceWeek/Promo2SinceYear/PromoInterval

`Promo2SinceWeek`， `Promo2SinceYear`， `PromoInterval`都有544个缺失值，可能相互之间有联系，所以放在一起分析。

通过观察这些缺失数据后，发现这些数据的`Promo2`值大部分为0。为了验证`Promo2`的取值与`Promo2SinceWeek`， `Promo2SinceYear`， `PromoInterval`缺失值的关联。当`Promo2`为1时， `Promo2SinceWeek`没有缺失值。所以可以确定`Promo2SinceWeek`,`Promo2SinceYear`,`PromoInterval`的缺失是因为`Promo2`为0。

因为有些店不进行`Promo2`的促销活动，所以和`Promo2`相关的特征缺失属于正常情况。决定用0对这些缺失值进行填充。

异常值处理

- Sales

在对`Sales`观察时发现有大量的异常值，这些异常值会对模型引入偏差，所以需要删除。利用Tukey's Test去这些极度异常值。同时因店铺装修或关门(`Open = 0`)的原因，一些数据的`Sales`为0，为了不对模型产生bias，需要删除这些`Sales`为0的数据。

- Customers

`Customers`在2013.1.22这一天取得最大值`7388`，这一天在进行促销活动(`Promo = 0`),不认为是异常值，所以保留。

- CompetitionOpenSinceYear

在对`CompetitionOpenSinceYear`进行探索时，发现有一天数据为1900年，虽然欧洲有很多百年老店，但是大部分数据都集中在90年代以后，可以认为是异常值，进行删除。

这里在后续的过程遇到了问题，当进行store与test数据集合并的时候，测试数据集中，Store为815的数据缺失，导致模型预测失败。所以这里保留这条数据，不进行删除。

数据划分

由于是时间序列，所以在划分训练集，验证集，测试集的时候需要注意不能有时空"穿梭",也就是不用用未来的时间去预测过去的时间。

把训练集中的最后最后六个月，作为验证集，其余作为训练集。此外，由于12月不在测试集的预测范围内，而且12月的平均销量比较高，不够稳定，也在训练集中删除。

训练集大小：784757
验证集大小：38641

特征提取

- Season

调查了一下Rossmann的经营范围和主要销售产品，和国内的屈臣氏有些类似，考虑到季节与温度的因素可能会对人们消费的产生影响，所以增加了`Season`特征，取值为`spring`,`summer`,`autumn`,`winter`。为增加合理性，取每年的春分，夏至秋分，冬至作为季节的节时间季节结点。

通过观察 Season × Day × Sales， 可以看出不同季节下每月的销量还是很不同的，具有较好的区分性。

- Xun

在观察Season × Day × Sales的时候，发现每月的月初，月中，月末呈现不同的趋势，大体是月初较高，月中降低，月末又上升。想到可能每月发工资的时间有关系，所以创建`Xun`特征，取值为`begin`,`middle`,`end`。

通过观察Season × Xun × Sales，更容易捕捉不同季节下，一个月的时间内`Sales`的变化趋势。

但是在后面的测试中发现，Xun和Season并不是很重要的特征，所以在后面的模型中就没再引入。

以下特征是借鉴了Kaggle discussion上的方案：

- Promo2OpenInMonth：表示从门店开始进行持续促销的日期开始，到当前日期所经历的时长，以月为单位来进行计算
- CompetitionOpenInMonth：表示门店的竞争对手从开业到当前日期所经历的时长，同样以月为单位来进行计算
- IsPromo2Month：表示当前月份是否在门店开始进行促销的月份里，如果是，标记该特征的值为1，否则标记为0
- AvgSalesPerStore / AvgCustomersPerStore : 表示每个门店在开放的天数里的平均销售额(顾客数)
- AvgSalesPerStore_Promo / AvgCustomersPerStore_Promo : 表示在有促销活动时每个门店在开放的天数里的平均销售额(顾客数)
- AvgSalesPerStore_No_Promo / AvgCustomersPerStore_No_Promo : 表示在没有促销活动时每个门店在开放的天数里的平均销售额(顾客数)
- AvgSales_2013 / AvgCustomers_2013 : 表示2013年每个门店在开放的天数里的平均销售额(顾客数)

- AvgSales_2014 / AvgCustomers_2014 : 表示2014年每个门店在开放的天数里的平均销售额(顾客数)
- AvgSales_2015 / AvgCustomers_2015 : 表示2015年每个门店在开放的天数里的平均销售额(顾客数)
- AvgSales_1(2,3,6)_Month_before / AvgCustomers_1(2,3,6)_Month_before : 表示每个门店在过去的1,2,3,6个月，在开放的天数里的平均销售额(顾客数)
- DaysToHoliday : 距离节假日的天数，正值表示还有几天到，负值表示已经过去了几天
- AvgSalesPerDow，medianSalesPerDow：表示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
- AvgCustsPerDow，medianCustsPerDow：表示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

执行过程

1. 首先我建立了Training & Predicting Pipeline，可以记录每个学习器的四个指标

   - train_time 训练时间
   - pred_time 预测时间
   - rmspe_train 在训练集上RMSPE得分
   - rmspe_val 在验证集上RMSPE得分

   这能够帮助我快速有效地使用不同大小的训练集并在验证集上做预测训练和验证。

2. 我根据回归任务，选择了3个备选模型,分别是scikit-learn的`LinearRegression`和`RandomForestRegressor`,以及XGBoost的`XGBRegressor`

3. 分别使用1%， 10%， 100%的训练数据进行观察每个备选模型的效果

4. 选择综合表现最好的模型。

5. 进行参数调优

6. 进行多模型选择和融合

7. 测试集上进行测试

完善

在对RandomForestRegressor进行GridSearch的过程中发现，运行时间较长，超过24小时，而且还是运行在服务器上。因为在XGBoost训练时间相对较快，所以改用XGBoost模型。

对XGBoost模型选取了4个xgboost参数进行调参，分别为：

- max_depth：一棵树的最大深度值，用于控制模型的拟合情况；
- subsample：用于训练模型的子样本占整个样本集合的比例
- colsample_bytree：在建树时对特征采样的比例
- eta：更新过程中用到的收缩步长，eta通过缩减特征的权重使提升计算过程更加保守

模型参数

```
params = {'objective': 'reg:linear',
          'booster' : 'gbtree',
          'eta': 0.02,
          'max_depth': 10,
          'subsample': 0.9,
          'colsample_bytree': 0.7,
          'silent': 1,
          'seed': 1301
          }

num_boost_round = 20000
```

使用全部特征进行训练参数调优后在验证集上的RMSPE得分为: 0.025991

IV. 结果

模型的评价与验证

训练集大小：784757条数据
验证集大小：38641条数据

- 训练时间(单位：秒)

	1%	10%	100%
LinearRegression	0.02277	0.24828	4.50656
RandomForestRegressor	1.99506	25.34112	395.26783
XGBRegressor	1.54158	15.90852	241.01569

- 预测时间(单位：秒)

	1%	10%	100%

LinearRegression	0.01190	0.01388	0.01165
RandomForestRegressor	0.10138	0.17072	0.29855
XGBRegressor	0.22423	0.21947	0.24332

- 在训练集上的RMSPE得分

	1%	**10%**	**100%**
LinearRegression	0.14405	0.16239	0.19326
RandomForestRegressor	0.04515	0.04431	0.04767
XGBRegressor	0.09593	0.12075	0.15288

- 在测试集上的RMSPE得分

	1%	**10%**	**100%**
LinearRegression	0.56020	0.16725	0.19362
RandomForestRegressor	0.18902	0.04268	0.04487
XGBRegressor	0.18283	0.11641	0.14277

可视化结果如下：

回顾整个建模过程，从数据预处理、特征提取、构建基准模型，之后从简单线性回归模型开始，发现线性回归模型拟合效果都不理想。在初始，选择使用RandomForestRegressor模型，但在进行GridSearch的过程中发现，运行时间较长，超过24小时，而且还是运行在服务器上。因为XGBoost在训练时间相对较快，所以改用XGBoost模型。

在应用XGBoost模型的过程中，

首先特征工作做的比较少时，利用所有特征训练一个基准XGBoost模型，出现了较严重的过拟合现象，所以使用模型融合技术来减轻单个模型的过拟合现象。接下来再构建几个XGBoost模型。XGBoost基准模型出现过拟合现象的原因之一可能是特征数太多，于是采取从所有特征中随机选取不同数量的特征子集来训练XGBoost模型的策略，构建了10个模型，然后从这10个模型中选出4个验证集误差最小的模型进行调参，最终将这四个模型进行了简单的平均数融合。但最终结果效果都不是很好，没能达到10%的要求。

后来在查看Kaggle的Discussion后，引入了

- AvgSalesPerStore_Promo / AvgCustomersPerStore_Promo
- AvgSalesPerStore_No_Promo / AvgCustomersPerStore_No_Promo
- AvgSales_2013(4,5) / AvgCustomers_2013(4,5)
- AvgSales_1(2,3,6)_Month_before / AvgCustomers_1(2,3,6)_Month_before

- DaysToHoliday

用同样的参数训练时，结果要有显著提升。但还是没能达到10%的线。

再查阅资料后,发现一个小技巧，在销量比较小的时候，预测出来和实际可能方差会稍大，所以可以进行适当补偿，从而获得更小的RMSPE分数，我对最后的结果乘以0.98后，基本可以提升0.004的效果。最后的分数在Private Leaderboard 可以达到167名的位置。

整个过程中使用了模型选择、模型调参、模型融合等技术。

合理性分析

错误率的基准值为0.20。 由于XGBoost得分为0.11557，错误率降低了40%。 如果可以预测销售额的错误率只有0.12，那么管理者很容易做出必要的改变，看看是什么增加或减少了销售。

因此，对整个数据集进行了最后的训练，并且预测了测试集的销售。 可以肯定地说，这个模型将精确地预测所需的值。 因此，这个项目的目标已达到。

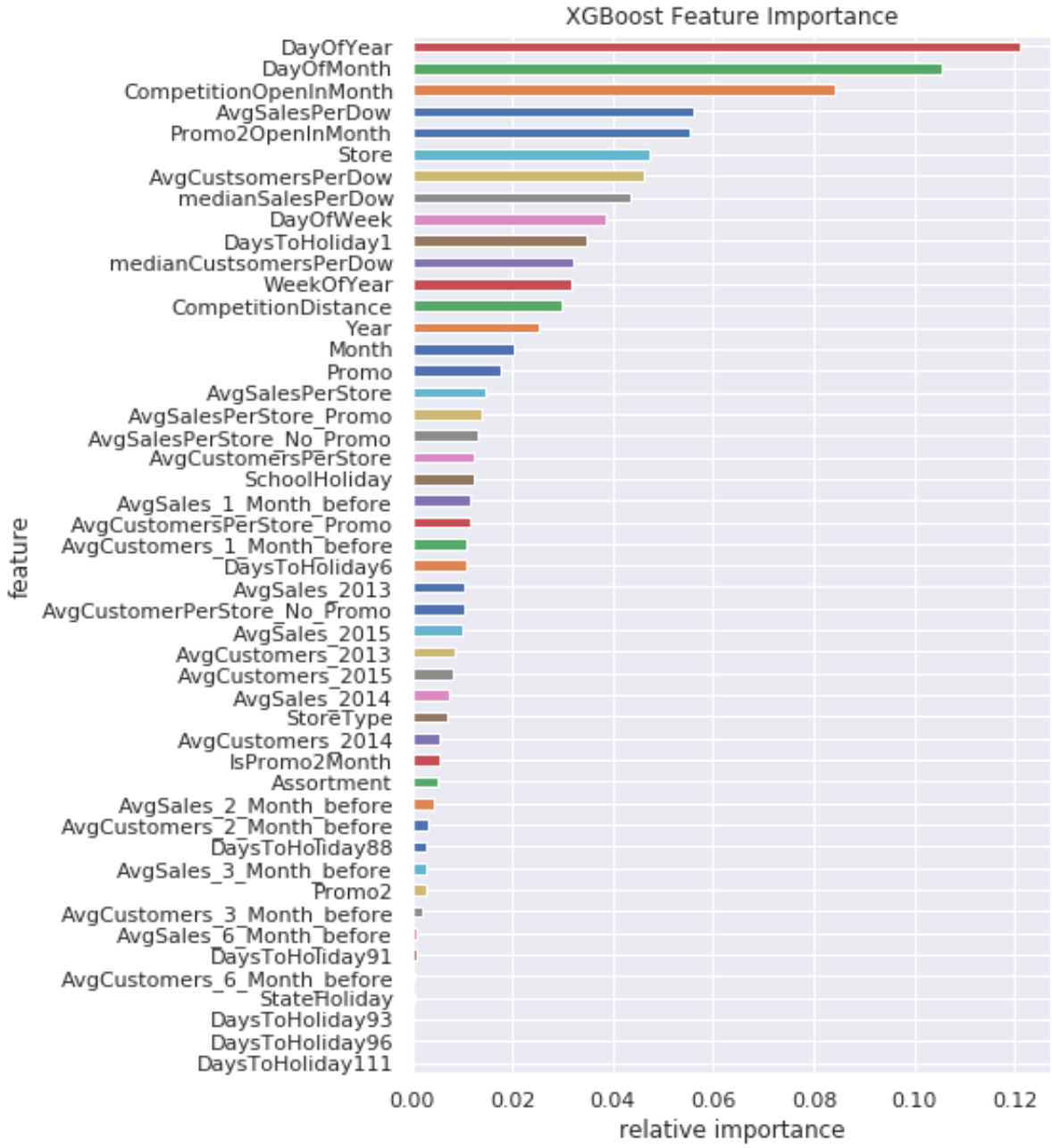
最终在测试集上的Public Score和Private Score与baseline的对比如下表所示：

评价指标	baseline	Private Score	Public Score
RMSPE	0.2	0.11557	0.10796

V. 项目结论

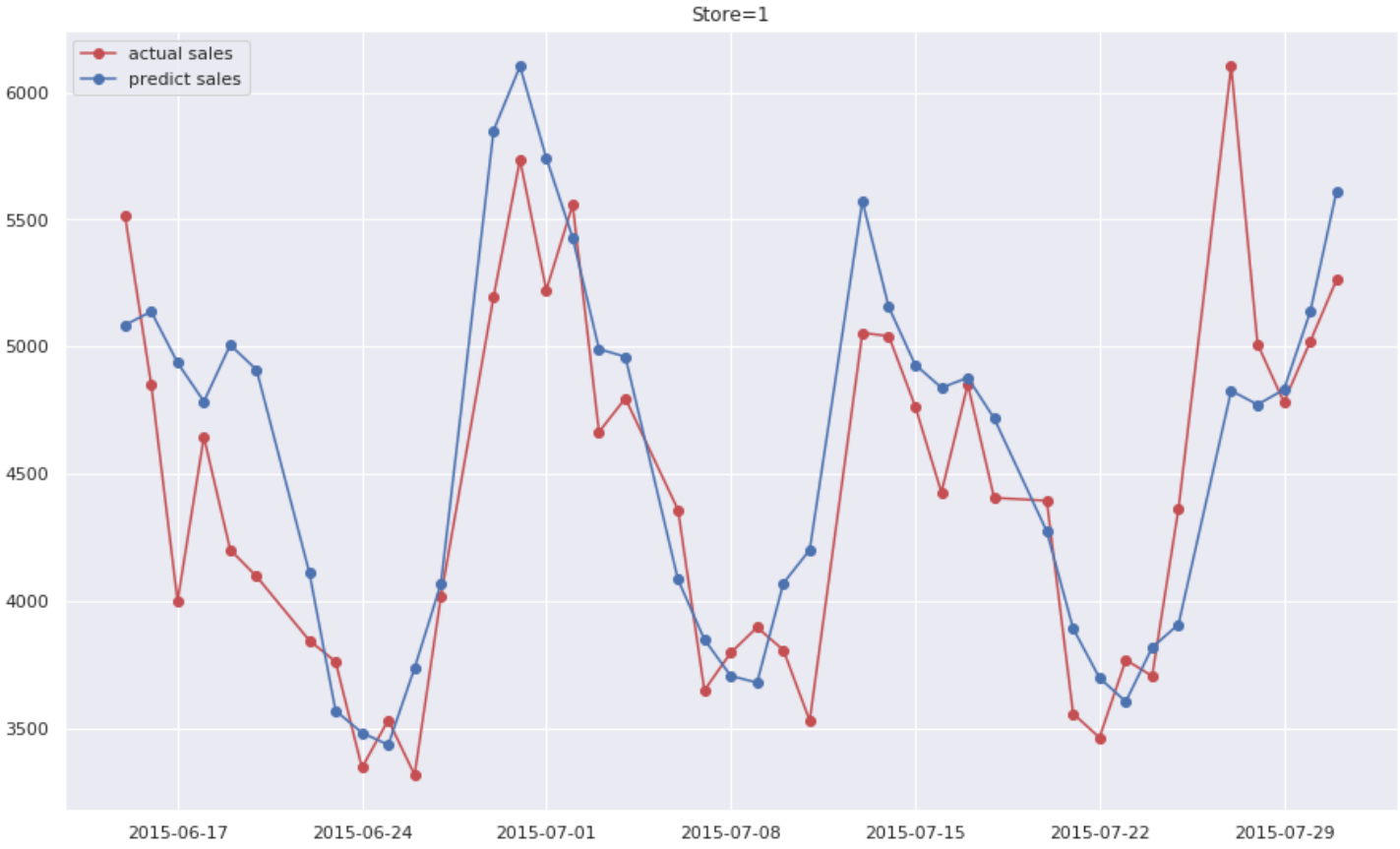
结果可视化

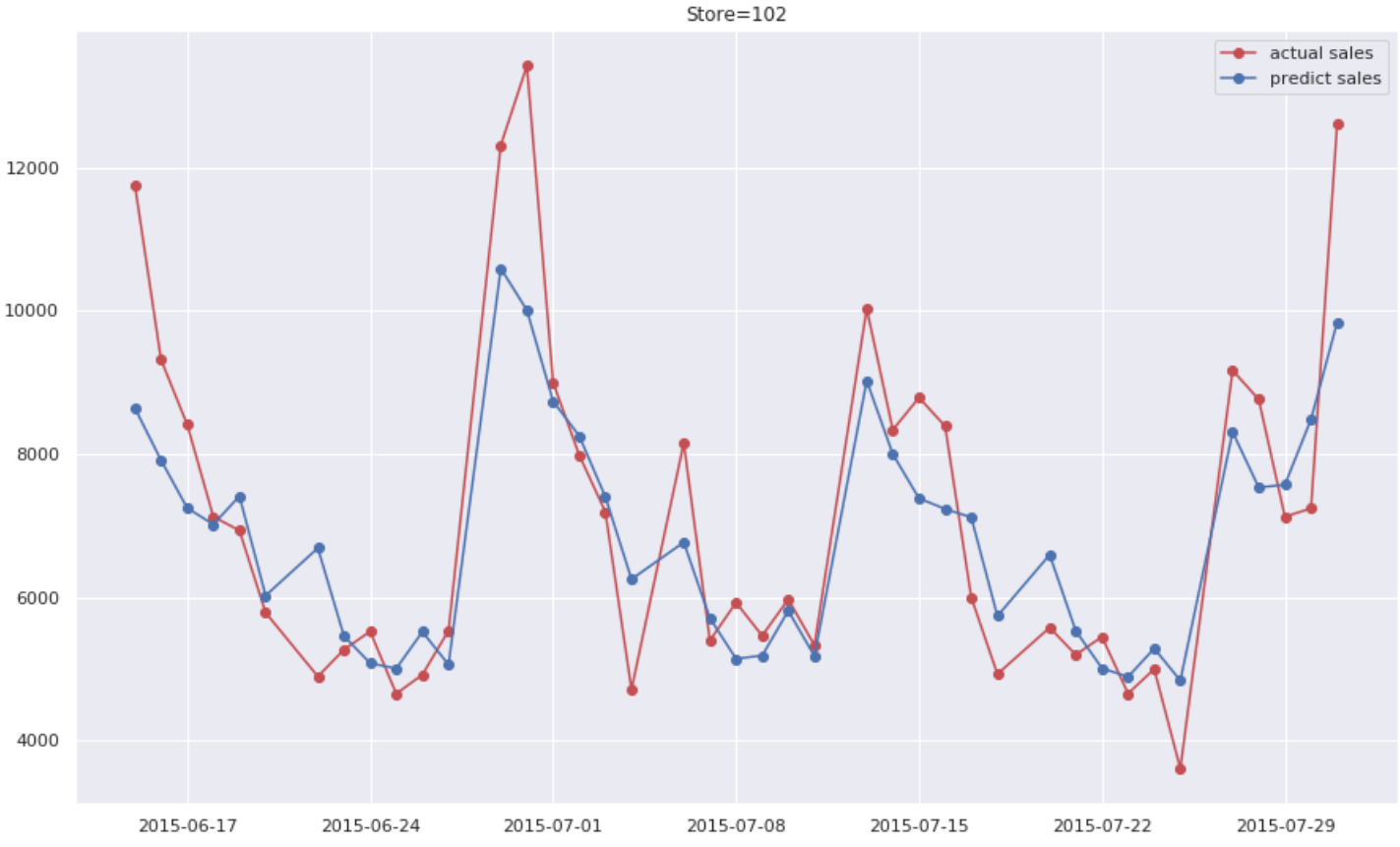
5.1 特征重要性

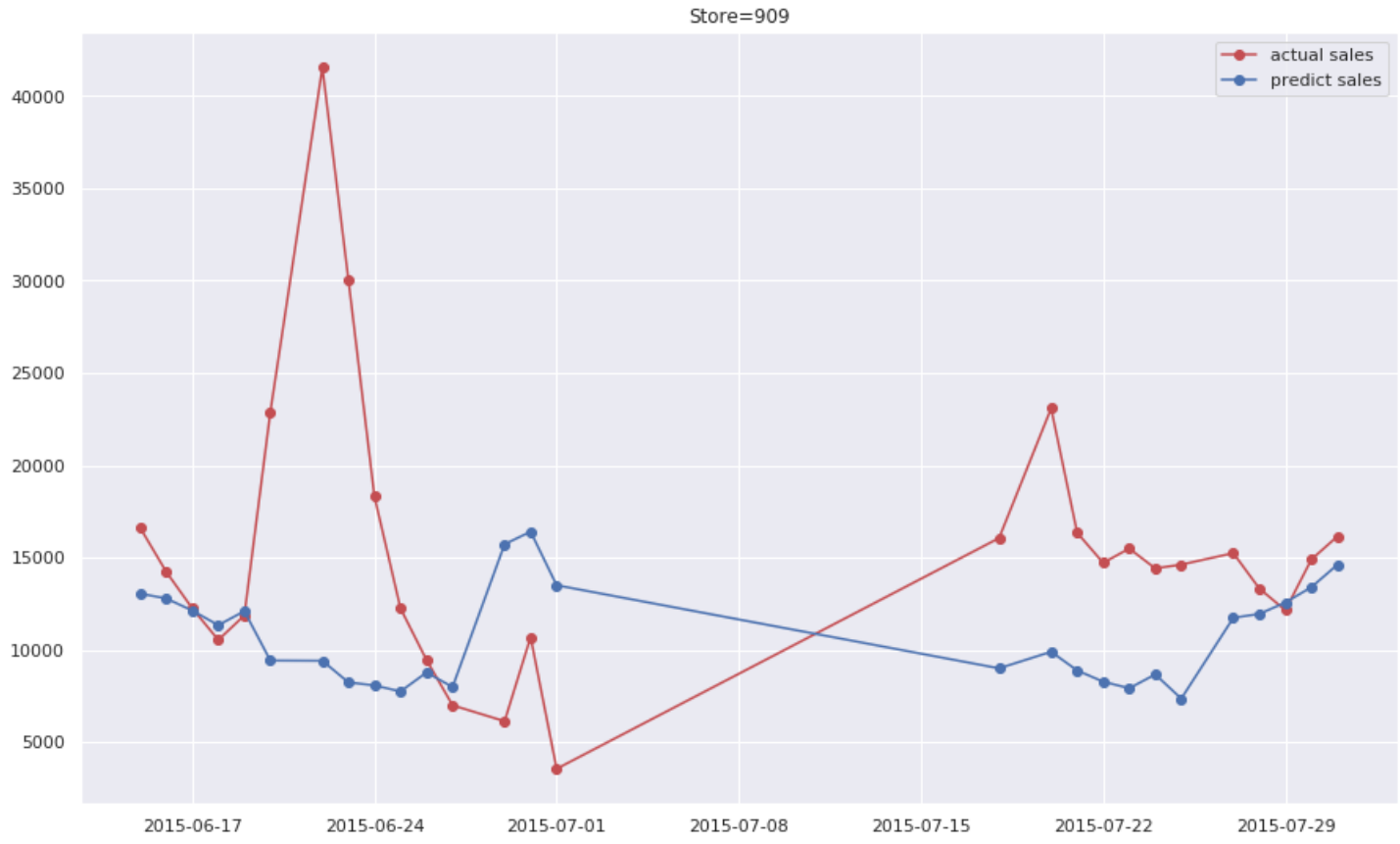


我们可以看到`DayOfYear`,`DayOfMonth`,`CompetitionOpenInMonth`,`AvgSalesPerDow`,`Promo2OpenInMonth`是前5个比较重要的特征。时间戳的转换和组合新的特征的确对训练更高精度的模型起到了显著的效果。另外，还可以发现过去一个月的相关特征重要性往往比过去两个月，一季度，半年的重要性高，这些在销量和顾客数量都有体现。还有一些特征重要性非常低，后续想提高模型训练速度可进行删除。

5.2 选取的几个门店的六周内的实际销售额和预测销售额的走势图


Store=1
actual sales
predict sales
6000
5500
5000
4500
4000
3500
2015-06-17
2015-06-24
2015-07-01
2015-07-08
2015-07-15
2015-07-22
2015-07-29


Store=102
actual sales
predict sales
12000
10000
8000
6000
4000
2015-06-17
2015-06-24
2015-07-01
2015-07-08
2015-07-15
2015-07-22
2015-07-29



可以看出，对于Store 1和Store 201来说，六周内的销售额预测走势和实际值比较吻合，对于Store 909来说，六周内的销售额预测走势和实际值相差就比较大。。在最终模型下

对项目的思考

开始着手项目后，首先需要了解数据，对数据进行探索性分析，我个人觉得这一步是非常有意思的地方，可以通过数据的可视化来发掘变量之间的关系，同时也可以在探索的过程中创建新的特征来进一步验证自己的发现。除了观察变量之间的联系，还需要注意数据集中的缺失值，异常值。处理异常值时，注意在训练集要删除的值没有在测试集中出现，否则会产生问题。

然后需要对数据进行预处理，如分割测试集和验证集,为后续使用学习算法做准备。

到了模型选择的部分，用备选模型的默认参数观察在训练集和验证集的表现，再选择要使用的模型。在这里借鉴了课程中其他项目的思路，首先建立train & predict pipeline，并对结果进行可视化。

在选择模型后需要对模型进行参数优化，这一步是比较难的地方，因为，我刚开始选了RandomForest模型，但对它进行GridSearch时间很长，在服务器上运行都要24小时以上。所以选择XGBoost模型,因为训练时间较快，在进行调参后得到了一个单一模型，但出现了过拟合现象。

首先特征工作做的比较少时，利用所有特征训练一个基准XGBoost模型，出现了较严重的过拟合现象，所以使用模型融合技术来减轻单个模型的过拟合现象。接下来再构建几个XGBoost模型。

XGBoost基准模型出现过拟合现象的原因之一可能是特征数太多，于是采取从所有特征中随机选取不同数量的特征子集来训练XGBoost模型的策略，构建了10个模型，然后从这10个模型中选出4个验证集误差最小的模型进行调参，最终将这四个模型进行了简单的平均数融合。但最终结果效果都不是很好，没能达到10%的要求。

在增加了新的特征后，模型的效果明显提升很多。说明机器学习的特征工程部分仍然是很重要的一步，调参并不能有显著提升。

需要作出的改进

可尝试使用多模型融合的方法来进一步提升效果。

在模型方面，除了xgboost模型，还可以尝试使用其他的技术，比如深度学习，这个比赛的第三名就是使用的深度学习方法。

VI. 引用

https://www.kaggle.com/stefanozakher94/eda-and-forecasting-with-rfregressor-final-updated

https://github.com/littlewizardLI/Udacity-ML-nanodegrees/blob/master/CapstonePoject/capstone_report.ipynb

https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/

https://blog.csdn.net/qq547276542/article/details/78304454

https://github.com/Wang-Shuo/Kaggle-Rossman-Store-Sales/blob/master/report.md

https://www.coursera.org/learn/competitive-data-science/home/welcome

https://www.cnblogs.com/hunttown/p/6927819.html

https://github.com/dmlc/xgboost/issues/2334

https://www.zhihu.com/question/56784594