Rossmann 销售预测报告

机器学习工程师纳米学位毕业项目

何伟健 2019.4.27

问题的定义

项目概述

销售预测是数据挖掘领域的一个行业应用。数据挖掘是一个跨学科的计算机科学分支，其定义是利用人工智能、机器学习、统计学和数据库的交叉方法在数据集中发现模式和规律的过程。[1]

本项目研究的销售预测问题，其主要出发点是通过准确的销售预测，帮助企业或商店做出更好的业务决策，合理有效地分配资源并提供员工的积极性。这里将尝试使用机器学习方法来解决问题。

有许多与本项目类似的公开数据集可供参考，如 DataMarket[2]上的魁北克月度汽车销售数据，肥皂销售数据等。在本项目我选取的是来自 kaggle 竞赛的 Rossmann 销售数据集，收集了 Rossmann 公司位于德国的 1115 家日化用品店的销售情况。

问题陈述

Rossmann 的商店销售预测问题是一个基于时间序列数据的预测问题，即基于时间顺序排列的数据从中发现变动的规律并进行预测。具体任务是根据各商店过去 2 年半的销售数据，来预测未来 6 周的销售额。这可以看作一个有监督回归问题，目的是从数据集中提取合适的特征并建立合适的模型对销售额进行准确的预测。

首先我将探索训练集，从中提取与销售额相关的特征。接着对适用于本研究问题的模型进行评估，从中选择最好的预测模型。最后使用最优模型对测试集进行预测。

输入数据

本项目用到的数据集包括 store.csv, train.csv, test.csv 文件，这些数据文件可以从 kaggle 上下载。其中 train.csv 训练集包含历史销售数据，test.csv 测试集是不含销售额的历史数据，两个数据集都是按天记录的，包括当天的销售额、客户量，等经营信息。store.csv 是关于商店的补充信息，包括商店的促销，竞争信息等。在本项目分析过程中，训练集与商店信息数训练据合并使用，用于训练模型，并在测试集上预测销售额。

评价指标

本项目观察值是每个商店的销售额，我将使用 Kaggle 竞赛中采用的指标 RMSPE 值来评价销售预测的好坏。

RMSPE (Root Mean Square Percentage Error)，即均方根误差百分比，其定义是预测值与真值偏差比值的平方，与观察次数 n 比值的平方根。计算方法如下：

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

其中 y_i 指单个商店单天的销售量，yhat_i 指相应的预测值。理想的 RMSPE 为 0。

分析

数据的探索

通过基本的目测评估和编程评估数据，整理了三个数据集的基本信息汇总表。详见表 1，表 2，表 3。

数据基本信息：研究的对象是 1115 个商店，每个数据以天为单位，时间跨度为 2013-1-1 到 2015-9-17，其中最后 6 周销售额为预测对象，之前的时间用于训练预测模型。商店按照类型分成 4 个类别，按照等级划分成 3 级。此外数据还包括促销信息，竞争关系，节假日信息等。

数据问题：

缺失值

- store 表的竞争信息和促销信息均存在数据丢失；
- test 表的 open 列信息存在部分丢失；

数据类型：数据集包含整型，文本型和浮点型数据。其中 Date 列数据以文本型存储，竞争商店开业的时间和促销时间则存储为浮点型。类别变量如 Promo2 用数值表示，而另一些如 StoreType 用字母表示。

表 1

store **数据集**				
总记录数： 1115				
列	**描述**	**值的范围**	**值的类型**	**非空值数**
Store	每个商店的代号	1 - 1115	int64	1115
StoreType	商店类别	a,b,c,d	object	1115
Assortment	商店级别	a=basic b=extra c=extended	object	1115
CompetitionDistance	与最近的竞争 商店的距离	/	float64	1112
CompetitionOpen SinceMonth	最近的竞争商店 开业的年份	1.0 – 12.0	float64	761
CompetitionOpen SinceYear	最近的竞争商店 开业的月份	1900.0 – 2013.0	float64	761
Promo2	是否在参与持续的 促销活动	0=没有参与 1=正在参与	int64	1115
Promo2SinceWeek	正在参与的促销活动 的开始年份	1.0 – 50.0	float64	571
Promo2SinceYear	正在参与的促销活动 的开始时间是第几周	/	float64	571
PromoInterval	促销活动开展的月份	[Jan,Apr,Jul,Oct] [Feb,May,Aug,Nov] [Mar,Jun,Sept,Dec]	object	571

表 2

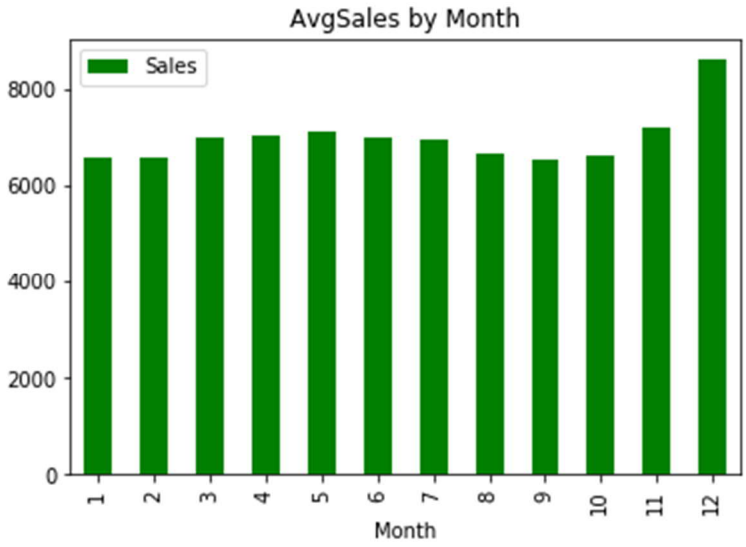
train **数据集**				
总记录数：101729				
列	**描述**	**值范围**	**值的类型**	**非空值数**
Store	每个商店的代号	1 - 1115	int64	1017209
DayOfWeek	当天是星期几	1 - 7	int64	1017209
Date	当天日期	2013-01-01 到 2015-07-31	object	1017209
Sales	当天的营业额	/	int64	1017209
Customers	当天的顾客数量	/	int64	1017209
Open	表示商店当天是否开门	1：是 0：否	int64	1017209
Promo	显示商店是否在当天促销	1：是 0：否	int64	1017209
StateHoliday	表示国家法定节假日	a=public holiday b=Easter holiday c=Christmas 0=None	object	1017209
SchoolHoliday	表示商店是否受 公立学校关门的影响	1：是 0：否	int64	1017209

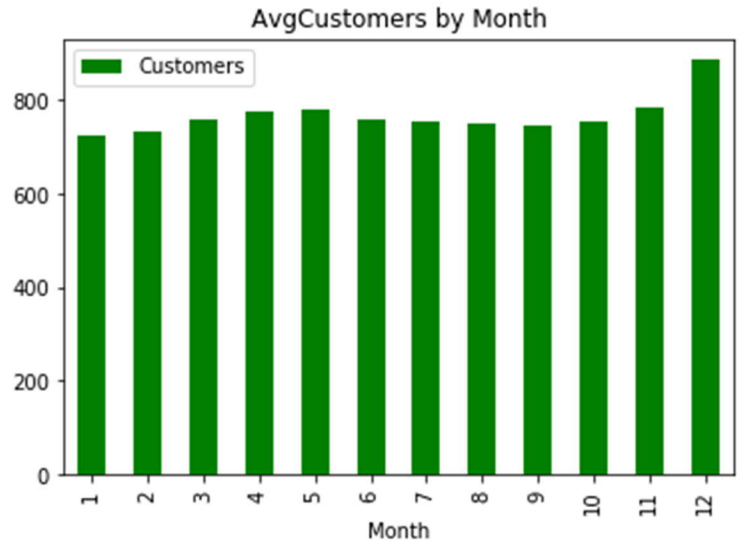
表 3

test **数据集**				
总记录数：41088				
列	**描述**	**值范围**	**值的类型**	**非空值数**
Id	测试集代表（商店 - 日期）组合的唯一号码	1 - 41088	int64	41088
Store	每个商店的代号	1 - 1115	int64	41088
DayOfWeek	当天是星期几	1 - 7	int64	41088
Date	当天日期	2015-08-01 到 2015-09-17	Object	41088
Open	表示商店当天是否开门	1：是 0：否	float64	41077
Promo	显示商店是否在当天搞促销	1：是 0：否	int64	41088
StateHoliday	表示国家法定节假日	a=public holiday b=Easter holiday	Object	41088
SchoolHoliday	表示商店是否受公立学校关门的影响	1：是 0：否	int64	41088

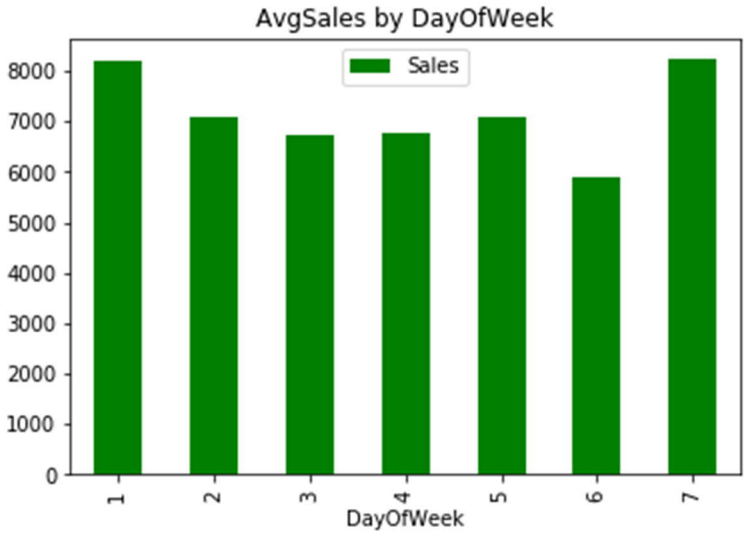
探索性可视化

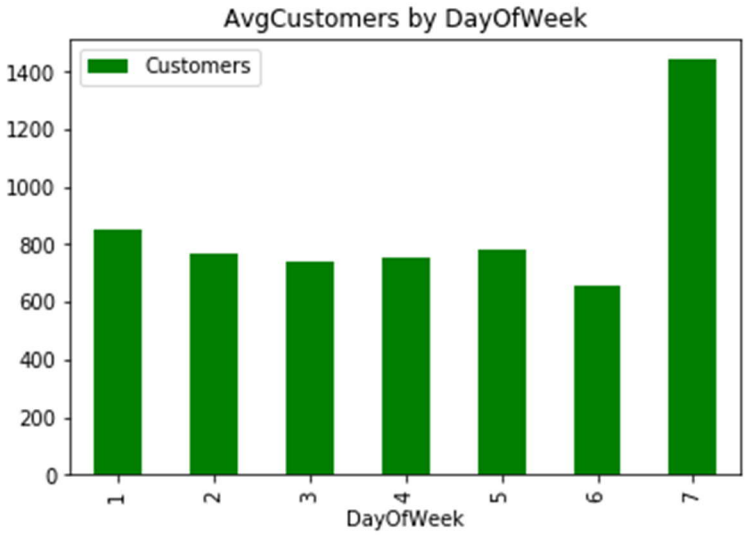
以下两图为商店每月平均销售和客户量的分布。可以看出在 12 月份二者都有明显的增长，可能与当月的圣诞节有较大联系。



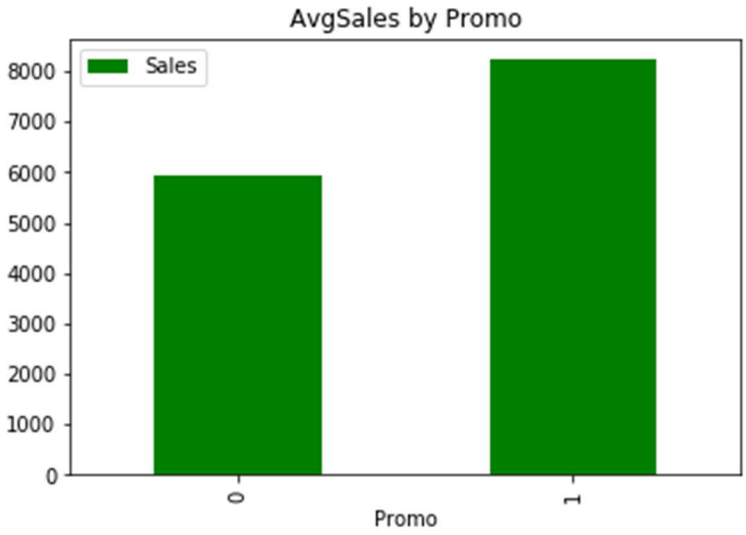


下图为商店周一到周日平均销售和客户量的分布。可以看出平均销售额在周一和周日都会达到一个峰值，可能由于周一是工作日的第一天以及周日有很多商店休息。然而每周平均客户量显示出较大的不同，周日比周一到周六的客户量有明显的激增。

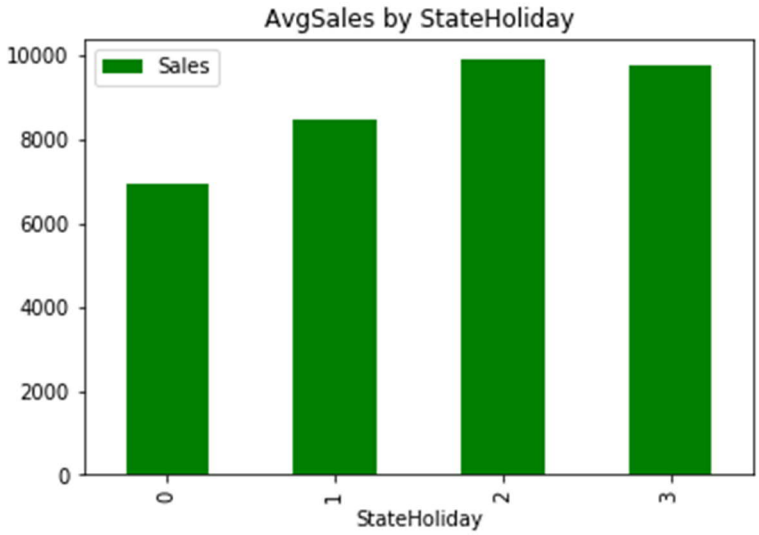


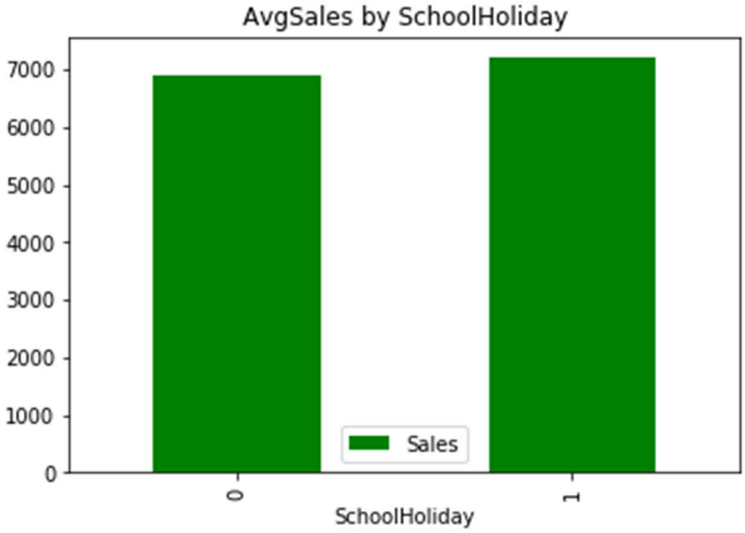


下图为当天营业的店有无促销的平均销售额分布。可以看出有促销对提升销售额有明显的作用。

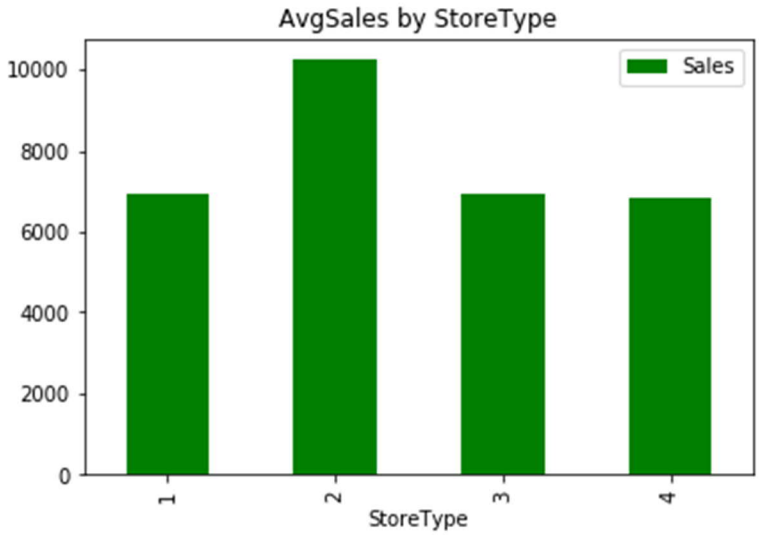


以下两图为节假日对平均销售额的影响。可以看出国家节假日对销售额均有明显促进，而学校假期对销售额几乎无影响，可能与学生的消费水平有关。

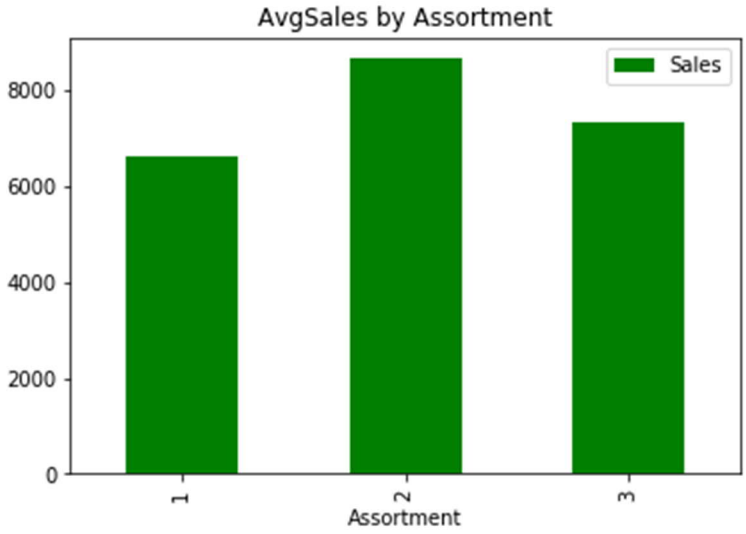




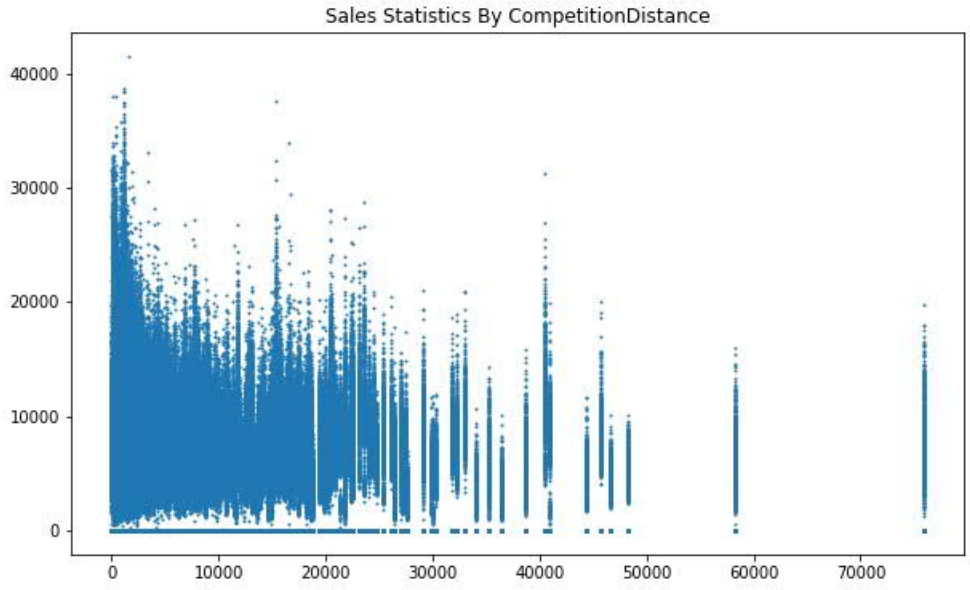
下图为不同商店类型的平均销售额分布。可以看出第二类型(b 型)商店的平均销售总额最高。



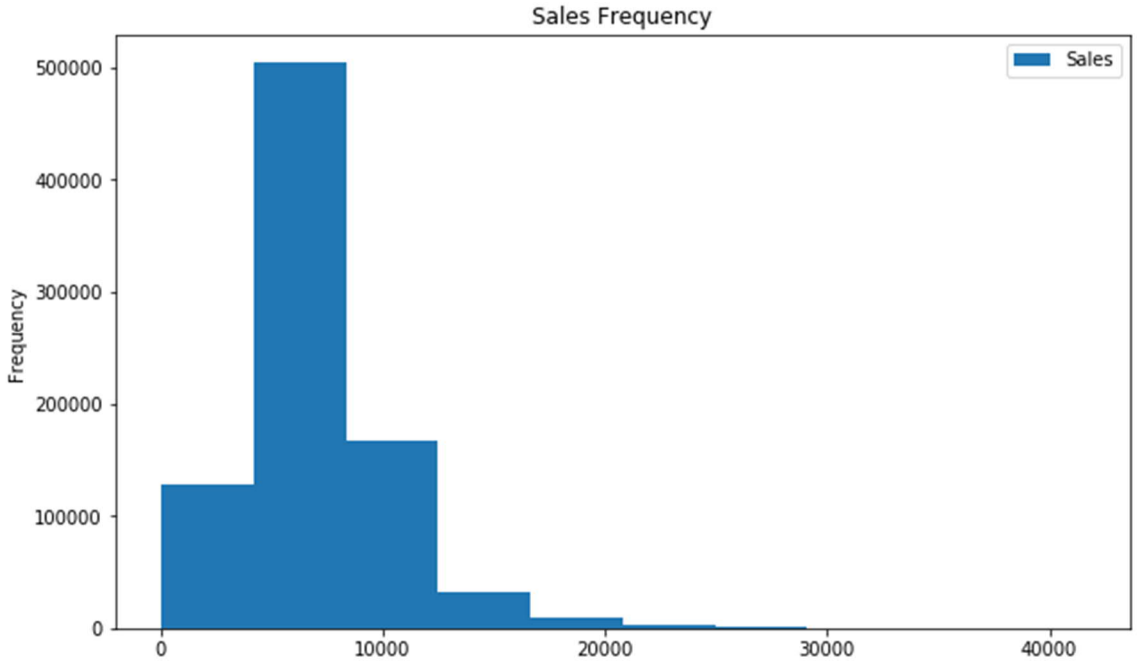
下图为不同等级商店的平均销售额分布。可以看出第二等级（ b 级）商店的平均销售额最高。

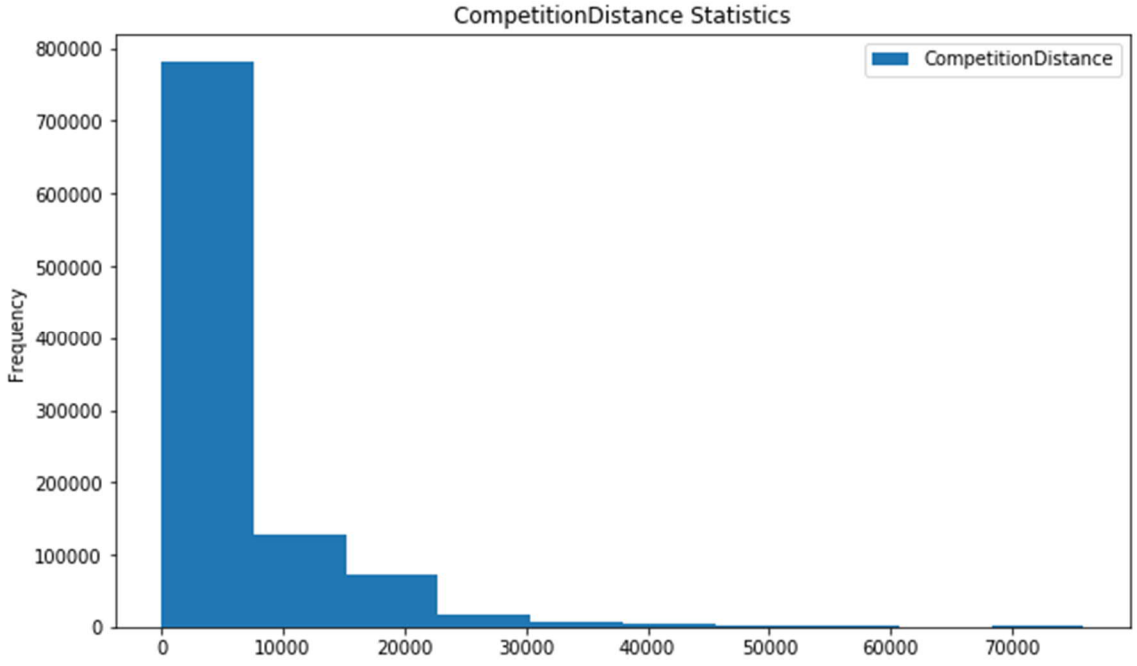


下图为商店销售额与竞争对手距离的散点图，可粗略看出当竞争对手与商店的距离越接近时，商店的销售额越小。



以下两个图为销售额、竞争距离统计分布图。可以看出两个变量均呈右偏分布，可考虑在训练模型时对它们进行进行对数变换。





算法和技术

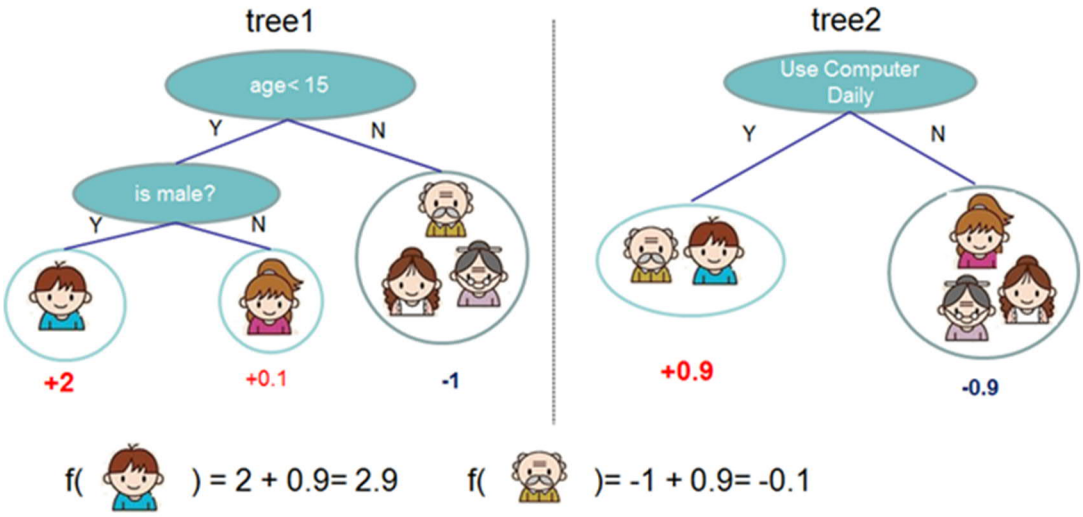
前面已介绍本项目研究问题为有监督回归问题，考虑到数据集中包含各种不同类型的特征数据，如数值型、类别型，而 XGBoost 擅长处理各种数据类型、关系和分布，以及具备大量的超参数。因此我将采用 XGBoost 的回归模型来解决本项目的预测问题。

XGBoost **介绍**

XGBoost 全称是 eXtreme Gradient Boosting，是大规模并行的 Boosted Tree 算法，属于 GBDT（梯度提升决策树）的一种高效实现。GBDT 是一种迭代的决策树算法，其基本原理是：首先使用训练集和样本真集训练一棵树，然后使用这棵树预测训练集，得到每个样本的预测值，预测值和真值相减可以得到“ 残差 ”。接下来训练第二棵树，此时不再使用真值，而是使用残差作为标准答案。第二棵树训练完成后，可以再次得到每个样本残差，再进一步训练第三棵树，以此类推。树的总数可以人为指定，也可以监控某些指标（如验证集上的误差）来停止训练。

在预测新样本时，每棵树都有一个输出值，将这些输出值相加，即可得到最终的预测值。

下面的例子是预测一个人是否会喜欢电脑游戏的 CART，可以把叶子的分数理解为这个人有多可能喜欢电脑游戏。我们用两棵树来进行预测，对于每个样本的预测结果就是每棵树预测分数的和。



XGBoost 相对于 GBDT 的改进：

1. 传统的 GBDT 以 CART 为基分类器，而 XGBoost 还支持线性分类器，相当于带 L1 和 L2 正则化项的 Logistic 回归和 Linear 回归。
2. GBDT 的核心在于后面的树拟合前面预测值的残差，这样可一步步逼近真值。主要是采用了平方损失作为损失函数：

   $$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

   GBDT 的损失函数只有一阶泰勒展开，而 XGBoost 进行了二阶泰勒展开，同时拥有一阶和二阶导数，可以使用前两阶作为改进的残差。因此 GBDT 是 XGBoost 的一个特例。详细的推导过程将不在本文体现，具体请参阅陈天奇论文 “ XGBoost: A Scalable Tree Boosting System ” [3]。
3. XGBoost 在代价函数中加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数，每个叶子节点上输出 score 的 L2 模的平方和。从 Bias-variance trade off 的角度来讲，正则项降低了模型的 variance，使模型更加简单，防止发生过拟合。
4. XGBoost 支持列抽样，即训练每棵树时，不是使用所有特征，而是从中抽取一部分来训练这棵树。既能降低过拟合，又减少了计算量。
5. XGBoost 模型在进行完一次迭代后，会将叶子节点的权重乘上一个学习率，能够削弱每棵树的影响，让后面的学习空间更大。
6. XGBoost 内置处理缺失值的规则，可以自动学习出它的分裂方向。

XGBoost 的主要参数解释：

通用参数	**定义/作用**	**值**
booster	选择每次迭代的模型 ，可选 gbtree（基于树的模型）或 gbline（线性模型），通常以 gbtree 为主。	默认 gbtree
silent	输出信息设置	0：输出相关模型信息，帮助理解模型。 1：无输出
nthread	用于进行多线程控制，输入系统的核数。	默认为最大可能的线程数

boost **参数**	**定义/作用**	**值**
eta	类似于 GBM 学习率，通过减少每一步的权重，提供模型的鲁棒性。	默认值 0.3 典型值为 0.01-0.2
min_child_weight	决定最小叶子节点样本权重和，值较大时可避免过拟合。但如果值过高，会导致欠拟合。	默认值 1
max_depth	树的最大深度，用于避免过拟合。值越大，模型越容易学到更具体的样本。	默认值 6
max_leaf_nodes	树上最大的节点或叶子的数量，与 max_depth 二选一	/
gamma	指定了节点分裂所需的最小损失函数下降值。在节点分裂时，只有分裂后损失函数的值下降了，才会分裂到这个节点。值越大，算法越保守。	默认值 0
max_delta_step	限制每棵树权重改变的最大步长	默认值 0：表示没有约束
subsample	控制对于每棵树，随机采样的比例。减小这个参数的值，算法会更加保守，避免过拟合。但是，如果这个值设置得过小，它可能会导致欠拟合	默认值 1 典型值 0.5-1
colsample_bytree	用来控制每棵随机采样的列数的占比(每一列是一个特征)	默认值 1 典型值 0.5-1
colsample_bylevel	用来控制树的每一级的每一次分裂，对列数的采样的占比	默认值 1
lambda	权重的 L2 正则化项。(和 Ridge regression 类似)。这个参数是用来控制 XGBoost 的正则化部分的	默认值 1
alpha	权重的 L1 正则化项，可以应用在很高维度的情况下，使得算法的速度更快	默认值 1
scale_pos_weight	在各类别样本十分不平衡时，把这个参数设定为一个正值，可以使算法更快收敛	默认值 1

学习目标参数:	**定义/作用**	**值**
objective	定义需要被最小化的损失函数	binary:logistic multi:softmax multi:softprob 默认值 reg:linear
eval_metric	对于有效数据的度量方法。	对于回归问题，默认值是 rmse，对于分类问题，默认值是 error
seed	随机数的种子,设置它可以复现随机数据的结果，也可以用于调整参数	默认值 0

模型融合

模型融合是本项目用到的另一个方法，也是 kaggle 等竞赛中常用的利器。将多个机器学习模型融合一起通常可以提高整体的预测能力，不管在多分类系统还是集成学习中，融合都是最重要的一个步骤。模型融合的基本理论假设是：不同的子模型在不同的数据上有不同的表达能力，我们可以结合他们擅长的部分，得到一个在各方面都很准确的模型。通常，最基本的假设是子模型的误差相互独立，但这是不现实的。即使子模型间的误差有相关性，适当的结合方法依然可以各取其长，达到提升的效果。

本项目我将采用一种简单的融合方式，即从竞赛的提交结果文件中进行融合，因为这样做不需要重新训练模型，只需要分别计算预测结果，然后采取某种措施得出最终结果。这里我将采用加权平均，可以一定程度上减轻过拟合现象，而且表现较好的模型将赋予更大的权重 [4]。算法为：

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) \ .$$

其中：

h_i 是每个模型的预测结果。

w_i 是分配给每个模型预测结果的权重，要求 $w_i \geq 0$，且所有 w_i 之和为 1。

基准模型

本项目的基准模型采用中位数模型，即对训练集按照 *Store*, *DayOfWeek*, *Promo* 进行合并后取销售额的中位数，再赋值到测试集的相应记录中作为预测值。

该基准模型是基于假设每个商店当天的销售额仅仅与当天是星期几及是否有促销活动相关。

测试集预测结果如下：

Private Score	0.14597
Public Score	0.14401

方法

数据预处理

数据问题

缺失值：

1. store 表的 CompetitionDistance, CompetitionSinceMonth，CompetitionSinceYear 的缺失值全部填充 0。（如果取平均值或者中位值填充效果反而不好）
2. store 表的 Promo2SinceYear 和 Promo2SinceMonth 的缺失值对应 Promo2 等于 0 的记录，因此可对缺失值填充 0。
3. test 表的 Open 列缺失值的记录对应的正常工作日，因此可以对缺失值填充 1。

特征转换：

1. 将 train 和 test 的 Date 列转化为时间戳类型，并提取出 Year，Month，Week，Day，DayOfYear 这几项时间特征。
2. train 表的 Sales 和 customers 特征的统计图呈现右偏分布，可进行对数转换，使之呈正态分布，有助于提高模型性能。
3. train 表的 CompetitionDistance 特征的统计图呈现右偏分布，同样进行对数转换
4. 对 store 表的 StoreType 和 Assortment 列数据替换为数字编码，对训练和测试集的 StateHoliday 列数据替换为数字编码。

特征提取：

在数据集已有特征及前面新增的时间特征基础上，根据数据分析和可视化，考虑增加以下特征：

- sales_avg: 每个商店平均销售额的平均值
- sales_median: 每个商店每天销售额的中位值
- customer_avg: 每个商店每天客户量的平均值
- customer_median: 每个商店每天客户量的中位值
- sales_by_customer: 每个商店单个客户带来的销售额
- sales_avg_per_DoW: 每个商店从周一到周日销售额的平均值
- sales_median_per_DoW: 每个商店从周一到周日销售额的中位值
- customer_avg_per_DoW: 每个商店从周一到周日客户量的平均值
- customer_median_per_DoW: 每个商店从周一到周日客户量的中位值
- holidays_thisweek: 本周是否有节假日
- holidays_lastweek: 前一周是否有节假日
- holidays_nextweek: 后一周是否有节假日
- Promo2ForMonth: 促销持续时间

执行过程

数据拆分：

本项目训练集将按照时间进行拆分，最后的六周数据将划为验证集，这样做是考虑验证集与测试集的时间上是相邻的，且时间跨度与测试集一样，两个数据集可能具有一些相同的特点。剩余的数据集用于模型训练。

建立 XGBoost 模型：

一开始我的 XGBoost 模型参数如下：

```
param = {
         'objective': 'reg:linear',
         'booster': 'gbtree',
         'eta': 0.03,
         'max_depth' = 10,
         'subsample': 0.9,
         'colsample_bytree': 0.5,
         'silent': 1,
         'seed': 17,
        }
```

训练的设置为：迭代次数 3000 次，早期停止次数 100 次。

先随机尝试一些特征组合来训练 XGBoost 模型，再用所有的特征运行一次模型，通过特征重要性分析，从中挑出重要性靠前的特征作为基本特征，重要性靠后的特征分成几组与基本特征结合，这样再得出一系列模型，选择 RMSPE 最小的模型用于后续调参。

随机特征组合 1（基本的时间、假期、商店信息特征）

模型 1	
特征 1	'Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance', 'Promo2', 'Year', 'Month', 'Day', 'Week'
训练集 RMSPE	0.06135
验证集 RMSPE	0.12934
Kaggle Public Score	0.12807
Kaggle Private Score	0.13397

随机特征组合 2（增加竞争和促销特征）

模型 2	
特征 2	'Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance', 'CompetitionForMonth', 'Promo2', 'Promo2SinceWeek', 'Promo2SinceYear', 'Promo2ForMonth', 'IsPromoMonth', 'Year', 'Month', 'Day', 'Week'
训练集 RMSPE	0.05449
验证集 RMSPE	0.13174
Kaggle Public Score	0.12920
Kaggle Private Score	0.13734

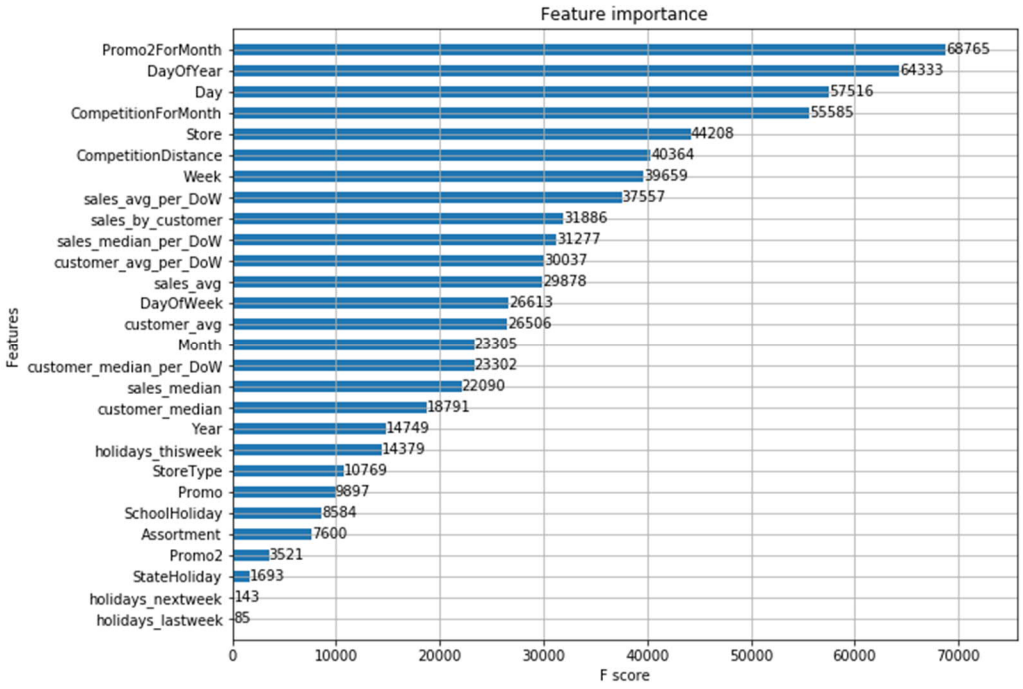
随机特征组合 3（增加平均销售，平均客户量，平均每个客户的销售量特征）

模型 3	
特征 3	'Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance', 'Promo2', 'Year', 'Month', 'Day', 'Week', 'sales_avg', 'customer_avg', 'sales_by_customer'
训练集 RMSPE	0.14987
验证集 RMSPE	0.11723
Kaggle Public Score	0.16079
Kaggle Private Score	0.14750

所有特征一起训练一次模型（包括数据集本身特征及特征工程新增的特征）

模型 5	
全部特征	'Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance', 'CompetitionForMonth', 'Promo2', 'Promo2ForMonth', 'Year', 'Month', 'Day', 'Week', 'DayOfYear', 'sales_avg', 'sales_median','customer_avg', 'customer_median', 'sales_by_customer', 'sales_avg_per_DoW', 'sales_median_per_DoW', 'customer_avg_per_DoW', 'customer_median_per_DoW', 'holidays_thisweek', 'holidays_lastweek', 'holidays_nextweek'
训练集 RMSPE	0.15509
验证集 RMSPE	0.12512
Kaggle Public Score	0.13387
Kaggle Private Score	0.13756

以下为用全部特征训练模型的特征重要性图，从中挑出排名前 15 的特征作为基本特征，剩余的特征随机分成几组分别与基本特征组合，再训练几个模型。



模型 7,8,9,10	
基本特征	'Promo2ForMonth', 'DayOfYear', 'Day', 'Store', 'CompetitionForMonth', 'sales_avg_per_DoW', 'CompetitionDistance', 'sales_avg', 'sales_median_per_DoW', 'sales_by_customer', 'customer_avg_per_DoW', 'DayOfWeek', 'Week', 'customer_avg', 'customer_median_per_DoW'
次要特征-1 （模型 7)	'sales_median', 'Month', 'Promo', 'holidays_nextweek'
次要特征-2（模型 8）	'customer_median', 'Year', 'Promo2'
次要特征-3（模型 9）	'holidays_thisweek', 'SchoolHoliday', 'StateHoliday'
次要特征-4（模型 10）	'StoreType', 'Assortment', 'holidays_lastweek'

经训练 4 个特征组合，其中对测试集预测分数最高的是模型 7，分数如下：

训练集 RMSPE	0.08103
验证集 RMSPE	0.11813
Kaggle Public Score	0.11968
Kaggle Private Score	0.12750

可以看出通过减少一些重要性较低的特征来训练，一定程度上带来了模型性能的提升，但是测试集的分数离项目要求仍有一定的距离。另外虽然设定的模型运行次数是 3000 次，但实际上大部分模型运行 1500 次左右其验证集误差基本趋于平稳。

完善

模型调参：

以模型 7 为基准，选取了 3 个 XGBoost 主要参数进行调参，分别是：

- max_depth: [9, 10, 11]
- subsample: [0.7, 0.8, 0.9]
- colsample_bytree: [0.5, 0.6]

另外迭代次数 nrounds 设定为 1500，其余参数维持不变。

经调参后，验证集结果：

Case	colsample_bytree	max_depth	subsample	valid_error
1	0.5	9	0.7	0.120147
2	0.6	9	0.7	0.121247
3	0.5	9	0.8	0.119239
4	0.6	9	0.8	0.121205
5	0.5	9	0.9	0.120159
6	0.6	9	0.9	0.121620
7	0.5	10	0.7	0.120220
8	0.6	10	0.7	0.119528
9	0.5	10	0.8	0.119948
10	0.6	10	0.8	0.119730
11	0.5	10	0.9	0.118153
12	0.6	10	0.9	0.119505
13	0.5	11	0.7	0.118013
14	0.6	11	0.7	0.118656
15	0.5	11	0.8	0.118931
16	0.6	11	0.8	0.118874
17	0.5	11	0.9	0.118279
18	0.6	11	0.9	0.118996

从 18 个组合的验证集误差来看，第 13 个组合（对应 max_depth=11, subsample=0.5, colsample_bytree=0.7）的验证集误差最小，为 0.11801，比原始模型 7 略提高一些（对应 max_depth=10, subsample=0.5, colsample_bytree=0.9），然而可以看到 18 个组合的验证集误差的差别都不是特别大，需要进一步的改善措施。最终调参后的模型参数如下：

```
param = {
         'objective': 'reg:linear',
         'booster': 'gbtree',
         'eta': 0.03,
         'max_depth' = 11,
         'subsample': 0.7,
         'colsample_bytree': 0.5,
         'silent': 1,
         'seed': 17,
        }
```

模型融合（预测结果融合）：

从上述所有模型的预测结果来看，分数最高的是模型 7 的第 13 个参数组合（简称模型 7-13），以及原始的模型 7，其次是用所有特征训练的模型 5，及随机特征的模型 1，最后的特征拆分的 3 个模型似乎对结果并没有改善。我将采用性能最好的前 4 个模型的预测结果参与融合：模型 7-13，模型 7，模型 1，和模型 5（包含一系列新增特征，有参考价值）。

模型融合采用对预测结果进行加权平均的方法，我将通过调整验证集的预测结果（即训练集的最后 6 周），使用权重因子和微调因子进行模型调整，公式如下：

```
model_new = [model_1 * x1 + model_5 * x2 +model_7 * x3 + model_7_13* x4] * y
```

其中：
x1,x2,x3,x4 是权重因子，决定分配给 4 个预测结果的权重，x1+x2+x3+x4=1

y 是微调因子，在融合结果的基础上对总体结果再进行微调

x1,x2,x3,x4	0	0.05	0.1	．．．	0.90	0.95	1
y	0.9	0.905	0.910	．．．	1.090	1.095	1.1

最终取 x1=0.2，x2=0.25，x3=0.25，x4=0.30, y=0.975，本地验证集的 RMSPE 可得到最小值 0.11337。

对 4 个模型的测试集预测值使用该融合模型，最终得到的分数是：

Private Score	0.11893
Public Score	0.11258

相比于单个最好模型的预测性能，多个模型融合的预测精度得到了比较明显的提升，kaggle 的 private score 从 0.12750 降到了 0.11893，进一步接近目标分数。

预测结果校正：

先观察融合后的模型对验证集的预测误差分布情况，下图为销售额 log 值的 RMSPE 误差分布，以 Store 为观察对象。取 log 值是考虑把值的范围标准化，便于观察和后续的校正。可以看到绝大多数的商店平均误差在 0.01 附近，分布规律非常清晰，只有极少数商店的误差超过 0.02。因此我将尝试对每个商店的销售预测值进行校正，寻求对应每个商店最小误差的权重。

结果校正的代码参考了 Rossmann kernel [7] 上的方法。校正方法：设置权重集合 W，其中每个商店的权重 w 取值范围为 0.990 到 1.009，共遍历 20 个值，步进为 0.001，这样覆盖了±1%的误差范围，校正计算为 sales_log_new = sales_log * w。经校正后最终的验证集的 RMSPE 误差进一步降低为 0.10445，相比校正前的验证集误差整整缩小了近 8%，模型性能预测性能得到大幅提升。

最后对测试集采取执行相同的步骤，先对 4 个模型的测试结果融合， 再采用上述权重集合 W 对每个商店的销售预测值进行校正。提交 kaggle 计算结果如下：

Private Score	0.11116
Public Score	0.10587

可以看出，经过对预测结果的融合并进行校正后，获得了目前最好的分数，达到了本项目的要求，我将以此为最终结果。

结果

模型的评价与验证

本项目在模型构建的过程中，我采用了不同的方式对数据进行预处理；特征选择方面，通过模型的特征重要性进行分析并择取有用的特征。验证方式上我使用的是训练集中最后 6 周的数据作为验证对象（占整个训练集约 5%），以 RMSPE 作为评估指标，同时通过 kaggle 计算的测试集误差结合评估模型。

从整个建模分析过程来看，最终模型的获得经历了四个重要过程：

- 从不同特征组合中挑选较优基础模型：有针对性的特征工程及特征重要性分析
- 模型调参优化：对 XGBoost 关键参数的不同组合进行模型优化
- 多模型融合： 充分发挥各模型的优点，降低过拟合
- 预测结果校正：有针对性地按不同商店修正模型误差

经过这四个过程模型预测性能是一直在提升的，最终测试集误差也满足本项目要求。如下表所示：

	验证集 RMSPE	测试集 RMSPE Public score	测试集 RMSPE Private score
基础模型（从不同特征组合中选出）	0.11813	0.11968	0.12750
模型调参优化	0.11801	0.11950	0.12735
多模型融合	0.11337	0.11258	0.11893
预测结果校正	0.10445	0.10587	0.11116

另外，由上面已给出的验证集误差分布散点图来看，1115 个点当中仅有不到 20 个预测误差（log 值）超过 0.02，其余所有点基本上稳定在 0.01 附近。因此输入的微小失误基本不会影响模型对数据整体的预测性能。

综上最终模型的不管从结果还是改进趋势来讲都是可信的，符合我的预期。

合理性分析

最终模型与基准模型在测试集上的 public score 和 private score 的对比如下：

	Public score	Private score
基准模型	0.14001（2316/3303）	0.14597（2094/3303）
最终模型	0.10587 （1006/3303）	0.11116（34/3303）
（基准模型-最终模型）/基准模型	24.0%	23.8%

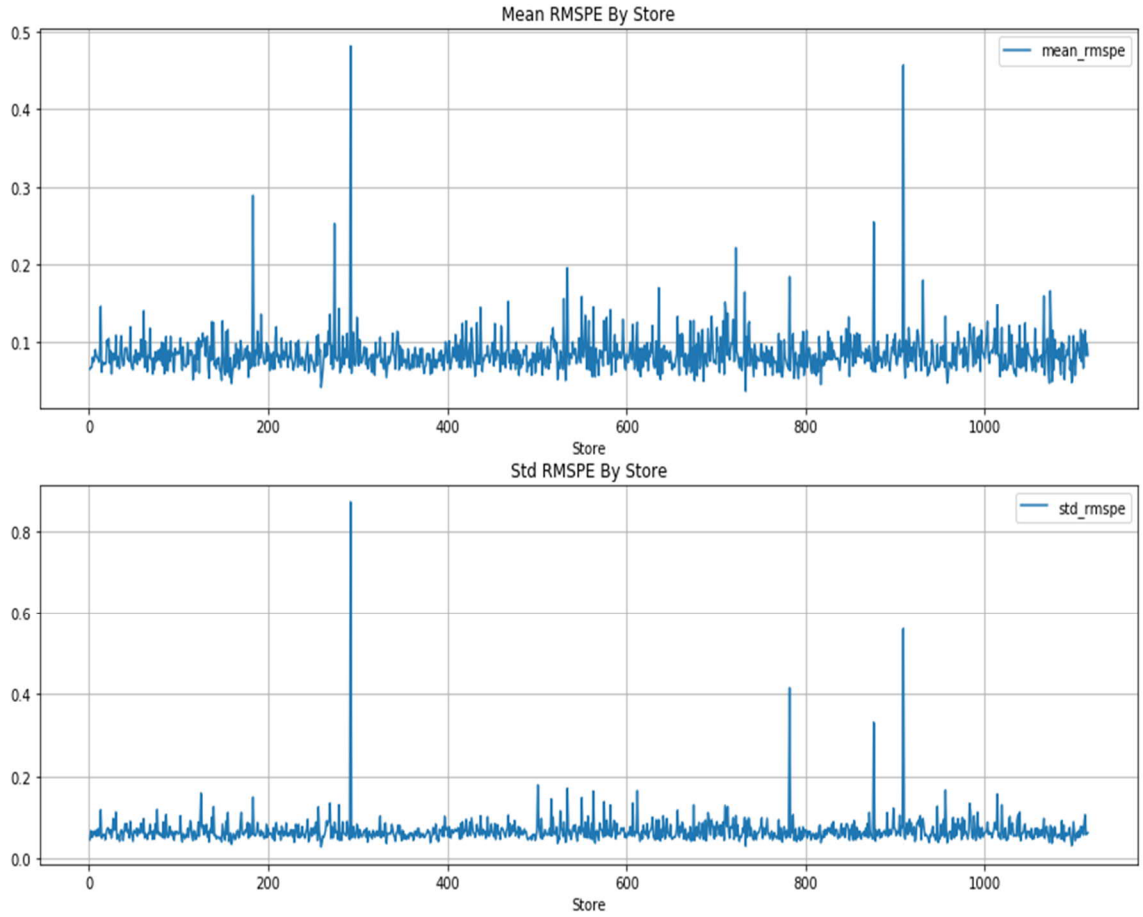
可以看出最终的模型在测试集上的误差均远远小于基准模型，而且优化幅度达到了24%，不管是性能提升幅度还是误差绝对值来看都是比较理想的结果。另外结合评价指标 RMSPE 的定义，最终模型确实具备较好的泛化性能，。因此我认为最终模型对未来六周的销售额预测准确度是比较高的，具有比较理想的参考价值。

项目结论

结果可视化

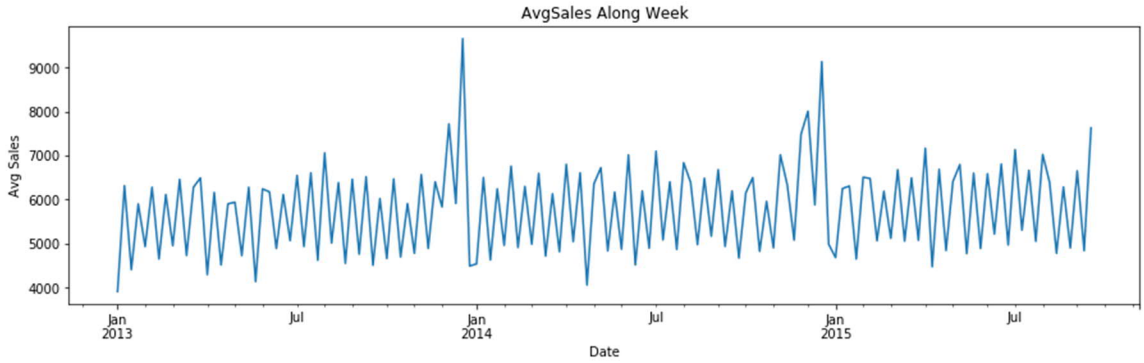
最终结果的可视化分三部分：

1. 对每个商店的实际销售和预测销售的 RMSPE 均值和标准差进行可视化。



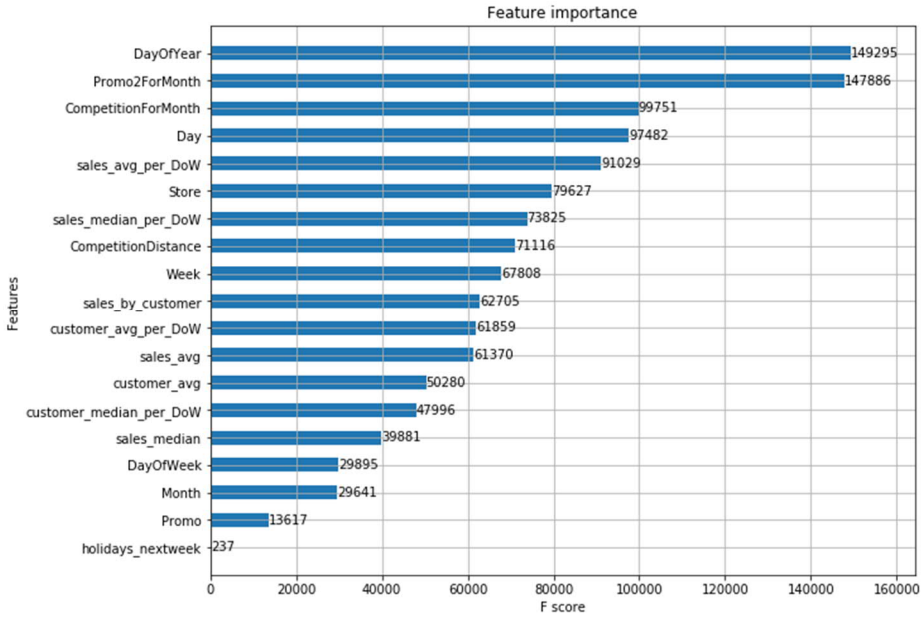
从以上两图可以看出，大部分商店在验证集的六周时间内的 RMSPE 值和标准差基本上在 0.1 左右波动，只有极少部分商店是超过 0.2 的。总体上看，模型的预测结果是可接受的。

2. 对每个时间周期（以周为单位）内所有商店的平均销售额的变化趋势可视化（测试集的预测销售额合并到训练集一并观察）



由上图可以看出，测试集的预测平均销售额与去年和前年同期的趋势和幅值比较一致，符合销售周期性变化的规律。

3. 特征重要度可视化



上图为模型 7（性能最好）的特征重要度统计图，纵坐标表示该模型每一个特征，横坐标表示每个特征的分数 F score，分数越高，特征被细分的次数越多，在该模型中的重要度就越高。可以看出，DayOfYear 和 Promo2ForMonth 对该模型的贡献最多，说明在该模型中当天的日期以及商店持续促销的时长影响是最大的。另

一点是 CompetitionForMonth 和 CompetitionDistance 特征均排在前 10，说明竞争关系对销售额也有关键的影响。此外 sales_avg_per_DoW 也占据重要地位，说明每周日均销售额对销售预测也有一定的作用，这一点与日均销售额在一周七天的分布有明显差异是相吻合的。

对项目的思考

本项目是一个销售预测问题，即利用过去 2 年半的销售数据，预测未来 6 周的销售额，这属于一个典型的时间序列预测问题。要预测每一天的销售量，除了要了解当天的因素之外，还要考虑过去以及未来一段时间内某些因素的影响，因此可以尝试创建一些特征，如促销活动持续的时长，前后的一段时间是否有节假日等。另外基于不同属性的销售平均值具有一定的共同性，如基于商店/客户量/星期几的平均销售额，可以进一步提高销售预测的准确性。

在选择最佳特征组合的过程中，我尝试了把所有原始特征和新增特征一起训练，然后按照特征重要度划分特征，将主要特征与次要特征随机组合，结果发现虽然可以一定程度上提升模型性能，但由于数据集本身已足够大，特征 shuffle 的改善力度不明显，而且运行一系列模型消耗的时间成本较大。因此我应当考虑把更多的精力放在模型调参和模型的进一步改善的工作中。

本项目的完整执行按照为：数据分析和探索性可视化 – 数据预处理 – 特征工程 – 训练模型 – 模型调参 – 模型融合 – 结果校正。 最终结果在 kaggle 上的 private score 分数为 0.11116， 达到了排名榜的前 1%，符合我对解决此问题的期望，对于解决同类型销售预测问题具有参考意义。

后续改进

采取的改进措施：

1. 一开始我采用的建模方式是训练多个单模型+模型融合，其预测结果离项目目标有一定的距离。后续增加了模型结果校正，大大改善了最终预测结果。因此我认为对模型结果的后处理是处理这类问题必不可少的方法。

本项目后续可能改进的地方：

1. 把销售量在不同时间窗口的移动平均值作为特征加入到模型中训练，比较是否会改善模型。比如创建前半个月/1 个月/3 个月的销售平均值特征。
2. XGBoost 模型运行的时间较长，从半个钟到 1 个钟不等，以至于没有尝试很多参数调优以及模型融合。后续可以考虑尝试 lightgbm。
3. 尝试采用 Entity-Embedding 模型，与 XGBoost 相比，一定程度上可减少特征工程的工作量。然后通过不同类型的模型融合是否有更好的提升。

引用

[1] Data Mining：https://en.wikipedia.org/wiki/Data_mining

[2]DataMarket：https://datamarket.com

[3] XGBoost 论文：XGBoost: A Scalable Tree Boosting System

[4] Kaggle Essemble Guide:
https://mlwave.com/kaggle-ensembling-guide/

[5] Time Series Analysis and Forecasts with Prophet:
https://www.kaggle.com/elenapetrova/time-series-analysis-and-forecasts-with-prophet

[6] Introduction to Boosted Trees：
https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

[7] Rossmann kernel:
https://www.kaggle.com/xwxw2929/rossmann-sales-top1/notebook