

# 预测 Rossmann 未来的销售额

## 总结报告

---

Denny

## 定义

### 项目概览

如何合理有效地利用企业掌握的数据为企业决策服务成为各行业关注的焦点<sup>[1]</sup>。早在 70 年代，计算机已经开始被应用于销售额的预测，与传统的方法相结合，使用定量方法，以趋势及其变化为重点，依靠历史数据，建立线性或非线性函数完成预测。

这个项目是 Kaggle 的比赛项目，其任务是通过 Rossmann 所提供的往年商铺数据去预测商铺未来销售额。作为数据分析类型项目，数据可视化与特征工程都非常重要，可视化可更好的理解数据，做好特征工程可获得更准确的预测。在模型上这类型的工作也有较多的选择，从较简单的 random forest 到最常用的 XGboost，在时序预测上还有 2017 年 Facebook 开源的 Prophet。

### 问题说明

欧洲 Rossmann 连锁店的历年销售数据是本项目的数据，其在 7 个欧洲国家拥有 3,000 家药妆店。商店销售受到诸多因素的影响，包括促销，竞争，学校和国家假日，季节性和地点。成千上万的个人经理根据其独特的情况预测销售量，结果的准确性可能会有很大的变化。

在这个源自 Kaggle 比赛的 Rossmann Store Sales 中，需要根据 Rossmann 药妆店的信息以及在过去的销售情况，通过对原始数据的清洗和可视化分析，挖掘隐藏于数据背后的特征，预测 Rossmann 未来的销售额。为了解决这个问题，我会将重点放在特征工程中，然后对比不同模型取得的成绩从而选出最优解。

## 评价指标

根据 kaggle 官方的测评指标对于该任务这里只能使用 rmspe 进行测评，对于测试集只能获取 rmspe 的分数。计算方式如下：

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

其中  $y_i$  表示单日商店的销售额， $\hat{y}_i$  表示相应的预测。评分中忽略任何一天销售为 0 的商店。

## 分析

### 数据探索

数据集包含以下三个部分：

1. 训练集 (train.csv)：其前样本如图 1 所示。包含 1.02m 条销售数据，时间跨度为 2013/01/1—2015/7/31。从该数据集可以获得以下信息：

Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
1	5	2015/7/31	5263	555	1	1	0	1
2	5	2015/7/31	6064	625	1	1	0	1
3	5	2015/7/31	8314	821	1	1	0	1

图 1 训练集样本

表格 1 train 数据域

数据域	描述	类型
Date	日期	日期
Store	商铺编号	
DayOfWeek	星期	
Sales	销售额	
Customers	顾客量	数字/
Open	是否营业 (0=关闭, 1=营业)	字符
Promo	是否促销 (0=不促销, 1=促销)	
StateHoliday	国家假日 (a=公共, b=复活节, c=圣诞节, 0=无)	
SchoolHoliday	学校假日 (0=不是, 1=是)	

在训练集中并不存在数据缺失的情况，所有数据域都有填充。而由于预测时销售额为 0 不计入评分，所以需观测是否只要营业就有销售额。在经过条件筛选后发现，有 172817 家店铺销售为 0 是未营业状态，有 54 家店铺虽然是营业状态但营业额为 0。

2. 商铺信息集 (store.csv)：其前样本如图 2 所示。包含 1115 个商铺的补充信息。从该数据集可以获得以下信息：

Store	StoreType	Assortment	CompetitionDistance	OpenSinceMonth	OpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
1	c	a	1270	9	2008	0			
2	a	a	570	11	2007	1	13	2010	Jan,Apr,Jul,Oct
3	a	a	14130	12	2006	1	14	2011	Jan,Apr,Jul,Oct

图 2 store 样本

表格 2 store 数据域

数据域	描述	类型
Store	商铺编号	
StoreType	商店类型: a, b, c, d	
Assortment	分类级别: a=基础, b=额外, c=扩展	
CompetitionDistance	最近的竞争对手商店的距离	数字/字符
CompetitionOpenSinceMonth/Year	竞争对手开店的年份/月份	
Promo2	连续促销 (0=否, 1=是)	
Promo2SinceWeek/Year	开始连续促销的年份/周	
PromoInterval	连续促销的间隔月份	

可以看到数据集存在缺失数据，经统计后发现 CompetitionDistance 缺失 3 个，CompetitionOpenSinceMonth/Year 缺失 354 个，Promo2SinceWeek/Year、PromoInterval 缺失 544 个

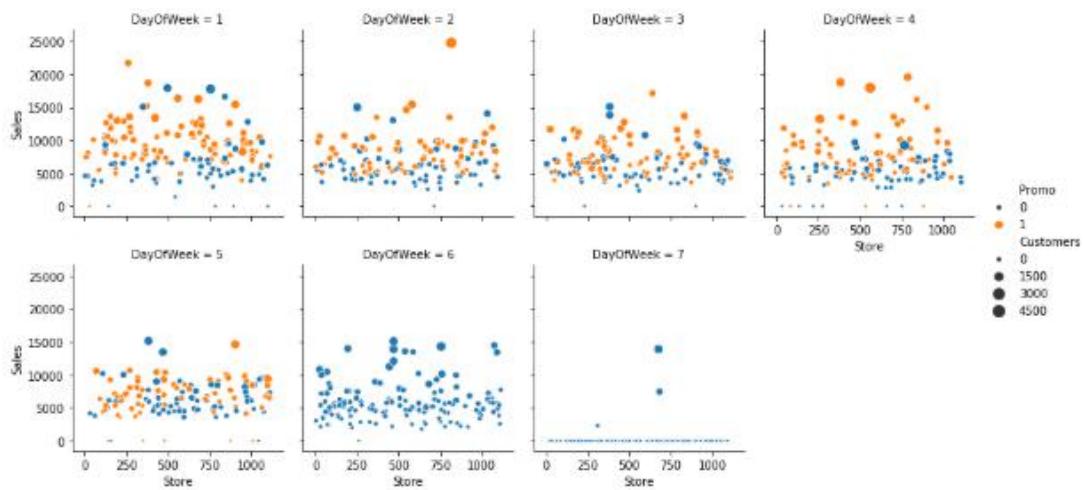
3. 测试集 (test.csv) : 包含 41.1k 条销售的历史数据, 时间跨度为 2015/08/1—2015/9/17, 该数据集的数据域与测试集相似, 仅缺少 Customer 与需预测的 Sales。在 Open 中存在 11 个缺失值

## 探索性可视化

为了可以获得更丰富的信息, 在可视化前先将 train 数据与 store 数据融合, 然后对可能影响销售额的因素逐个或组合观察。

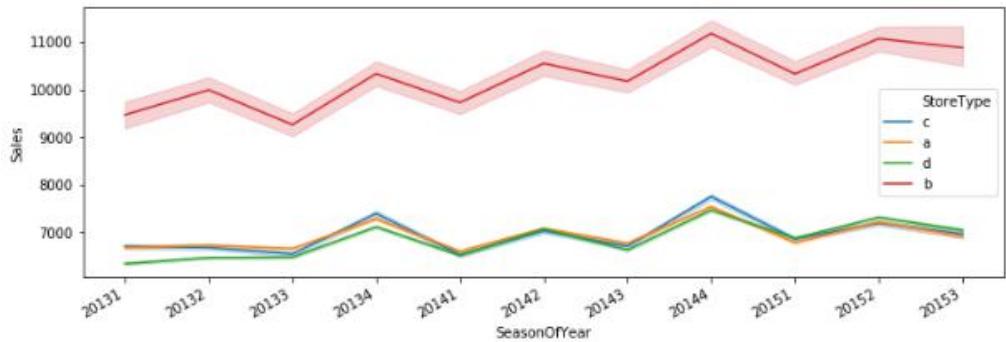
### 初步探索:

根据“星期”、“是否促销”、“顾客量”三个属性对 Sales 进行了初步可视化, 横轴为店铺, 纵轴为销量。可以看出 1. 顾客量与销售额成正比, 2. 促销能够提高销售额, 3. 周六几乎没有店铺促销, 4. 周日几乎所有店铺都没有销售额, 5. 周一至周六也有部分店铺没有销售额。



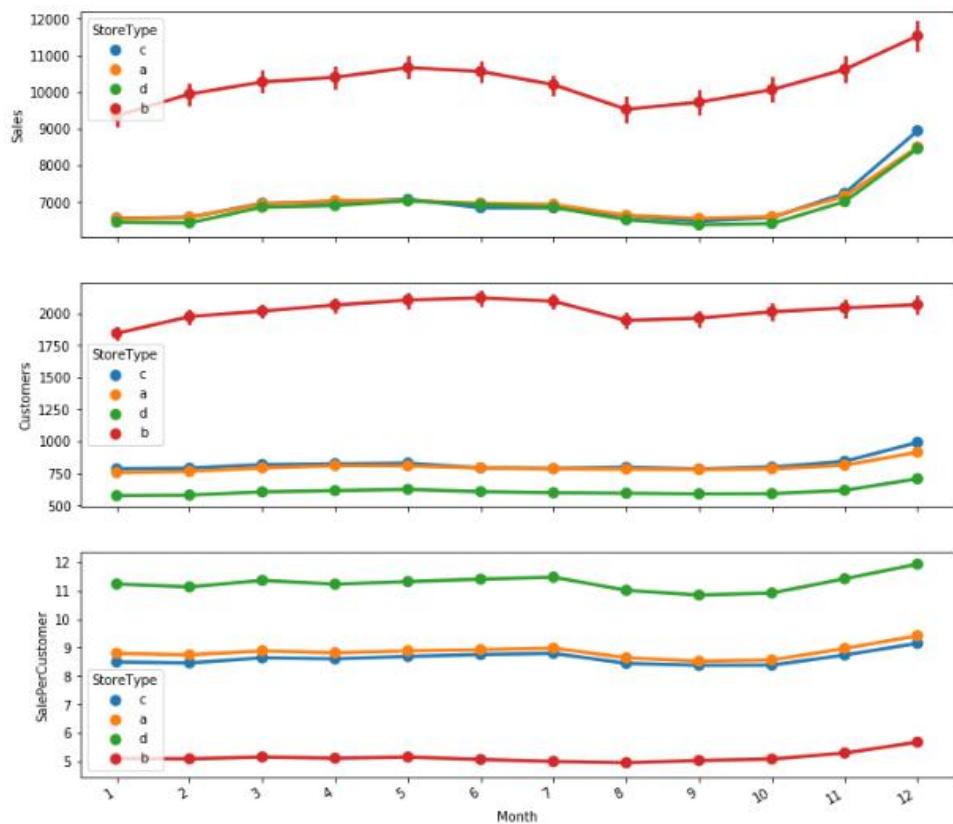
### 季度、店铺类型、与平均销售额:

下图根据每年每个季度与不同的店铺类型画出了销售额的变化, 从图中可看出 b 类店铺的平均销售额远高于其他四类店铺, 并且在 2013 年到 2015 年间波动增长, 而其他三类店铺则几乎没有改变, 并且 B 类商铺要比其他三类商铺的季度波动性更强。还可以看到四类店铺的销售额在第四季度都会有所上涨。



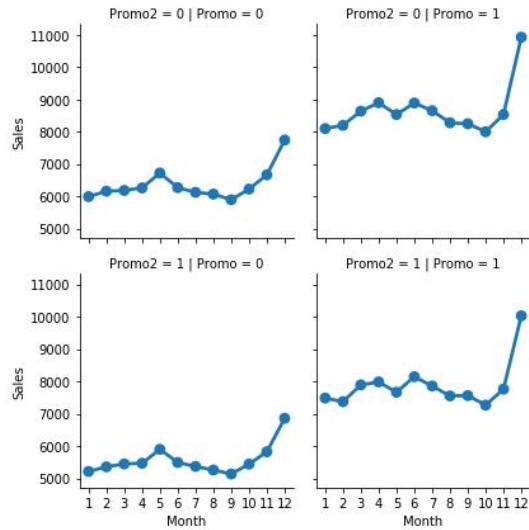
月份、店铺类型、顾客量、人均消费、平均销售额：

从图中可知四类店铺的销售、顾客量的峰值都在 12 月份出现，而 b 类店铺的高销售额是靠大量的顾客换取的，也就是说这些顾客人均消费非常低，相反 d 类店铺的人均消费是最高的。



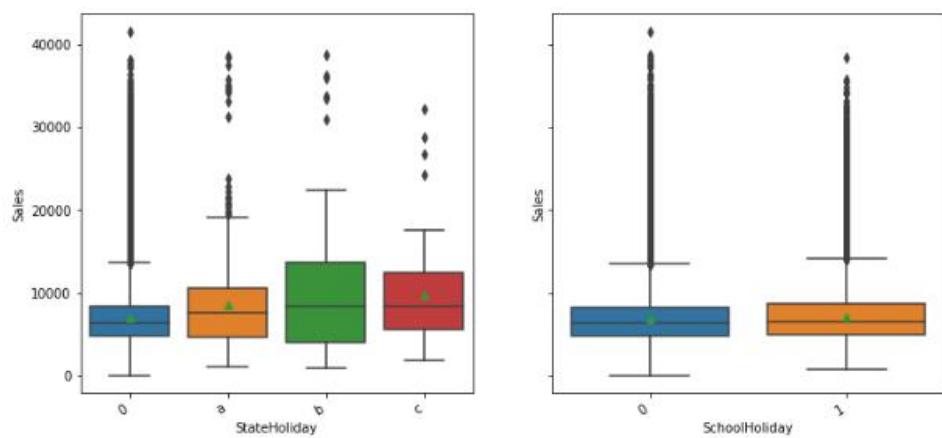
促销与销售额：

从图中可知有短期促销的店铺销额要大于无短期促销的店铺销额；有长期促销的店铺销额低于无长期促销的店铺销额



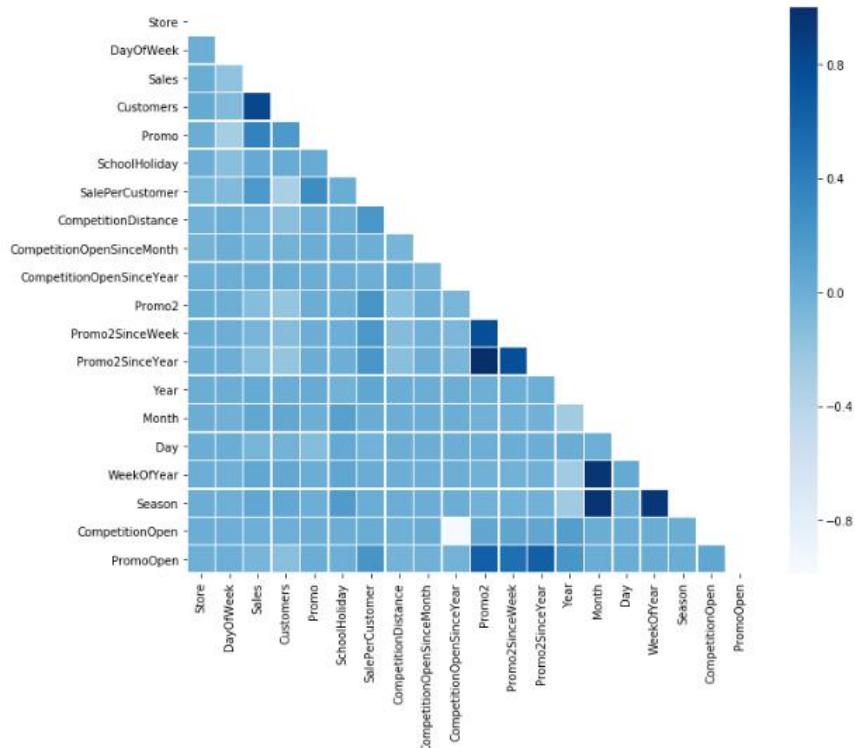
### 节假日与销售额：

从箱线图可发现在节假日开门的商铺的平均销售额要大于平日的平均销售额,在学校假期期间商铺的平均销售额也会略高于平日。



### 总体关联性：

鉴于篇幅有限无法一一给出所有特征与销售额的关系,所以最后通过热力图去大致呈现他们的关联性:



## 算法和技术

这个问题上销售额预测作为一个回归问题有很多的工具可供选择。这里我选择了回归决策树作为基准模型与 XGBoost 作为目标模型。

回归决策树：在 sklearn 中 DecisionTreeRegressor 使用的是 CART 算法，既可用于分类也可用于回归，如果待预测结果是连续型数据，则 CART 生成回归决策树。区别于 ID3 和 C4.5，CART 假设决策树是二叉树，回归树选取 Gain<sub>σ</sub> 为评价分裂属性的指标。选择具有最小 Gain<sub>σ</sub> 的属性及其属性值作为最优分裂属性以及最优分裂属性值。Gain<sub>σ</sub> 值越小，说明二分之后的子样本的“差异性”越小，说明选择该值作为分裂值的效果越好[2]。

XGBoost：XGBoost 可简单的理解为很多 CART 树的集成，是集成学习中 Boosting 流派中的其中一种算法，其本质上还是一个 GBDT。核心算法思想基本

就是：不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是在学习一个新函数，去拟合上次预测的残差，当我们训练完成得到  $k$  棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值[3]。

但与 GBDT 不同，GBDT 的原理是所有弱分类器的结果相加等于预测值，然后下一个弱分类器去拟合误差函数对预测值的梯度/残差回归任务下，GBDT 在每一轮的迭代时对每个样本都会有一个预测值，此时的损失函数为均方差损失函数：

$$l(y_i, y^i) = \frac{1}{2}(y_i - y^i)^2$$

所以，当损失函数选用均方损失函数时，每一次拟合的值就是（真实值 - 当前模型预测的值），即残差。此时的变量是  $y^i$ ，即“当前预测模型的值”，也就是对它求负梯度。

如果不考虑工程实现、解决问题上的一些差异，xgboost 与 gbdt 比较大的不同就是目标函数的定义：

$$L(\phi) = \sum_i l(\hat{y}_i - y^i) + \sum_k \Omega(f_k)$$

这个目标函数分为两部分：损失函数和正则化项。且损失函数揭示训练误差，正则化定义复杂度。 $\hat{y}_i$  是整个累加模型的输出，正则化项  $\sum_k \Omega(f_k)$  是则表示树的复杂度的函数，值越小复杂度越低，泛化能力越强，其表达式为

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

## 基准模型

考虑到这个项目与波士顿房价项目具有一定相识性，同样是回归问题，并且波士顿房价项目作为入门项目，其中所用到的模型简单易于理解，所以决定选用决策树作为基准模型。

通过引入 sklearn 库中的 DecisionTreeRegressor，便得到了一个回归决策树模型，通过网格搜索，得到'max\_depth' 和 'min\_samples\_leaf' 最佳参数作为最优模型参数，使用这个最优参数得到的 RMSPE 得分即基准模型的分数。

## 方法

### 数据预处理

缺失值处理：在 store 数据中的缺失情况如图所示，竞争对手相关的三个数据域缺失原因不明，使用均值填充，还有促销相关的三项数据域缺失是由于对应的 Promo2 为 0，故用 0 作填充。在 test 中 Open 数据域有 11 个缺失，如图所示，缺失数据都来自 622 号店铺，时间分布在周一到周六，故认为店铺为营业状态，填充 1。

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

store 缺失数据

test 中 Open 缺失情况

类型转换：在数据集中存在部分数据域的取值为“a, b, c, d”，为了使

DecisionTree 和 XGB 能处理这些数据，这里使用 a=1, b=2, c=3, d=4 作替换。

特征构建：为了使数据细化达到更好的预测与可视化效果，将 Date 拆开构建了 Year, Month, Day 特征，同时构建了 WeekofYear(一年中的第几周)、Season(季度)、CompetitionOpen(竞争对手营业时长)、PromoOpen(促销时长)，共 7 个特征。此外删除了 Date, Customers, Open, Promointerval, Id。

数据合并：训练集与测试集中的数据信息不多，所以在训练与预测前将这两个数据集与 store 数据进行了合并，可以在训练与预测时得到店铺更精确的数据。

## 执行过程

数据准备：这个项目中使用了两个模型进行预测，DecisionTreeRegressor 作为基准模型，XGB 作为目标模型。在训练模型前对数据进行了可视化与预处理，构建与删除了部分特征，合并了数据。然后将数据集拆分为预测部分 X 与待预测部分 Y，再使用 sklearn 划分验证集与训练集，验证集比例为 0.3

特征维度：参与模型训练的特征共 21 个，分别是 Year、Month、'Day、DayOfWeek、'WeekOfYear、CompetitionOpen、PromoOpen、IsPromoMonth、CompetitionDistance、Store、CompetitionOpenSinceMonth、StoreType、CompetitionOpenSinceYear、Promo2SinceWeek、Promo2SinceYear、Assortment、Promo、Promo2、SchoolHoliday、StateHoliday、Season。

评价函数：由于两个模型的参数不同，所以定义两个评价函数 rmspe\_dtr、rmspe\_xg 分别作为决策树与 XGB 的评价函数，两个函数的计算方式相同。

模型参数与得分：对于 DecisionTreeRegressor，首次训练时使用默认参数（不设参数），之后使用网格搜索对 max\_depth 和 min\_samples\_leaf 两项参数

进行优化，得到 max\_depth: 35, min\_samples\_leaf: 8，最终得分为 0.172。对于 XGB 模型，其主要参数设定如图所示，经过 300 次收敛，最终训练集得分为 0.192，而测试集为 0.131，分数优于 DecisionTreeRegressor。

```
params = {  
    'booster': 'gbtree',  
    'objective': 'reg:linear',  
    'subsample': 0.8,  
    'colsample_bytree': 1,  
    'eta': 0.3,  
    'max_depth': 6,  
    'seed': 42}  
  
xgb_model = xgb.train(params, dtrain, 300,  
                      evals=watchlist,  
                      early_stopping_rounds=50,  
                      feval=rmspe_xg,  
                      verbose_eval=True)
```

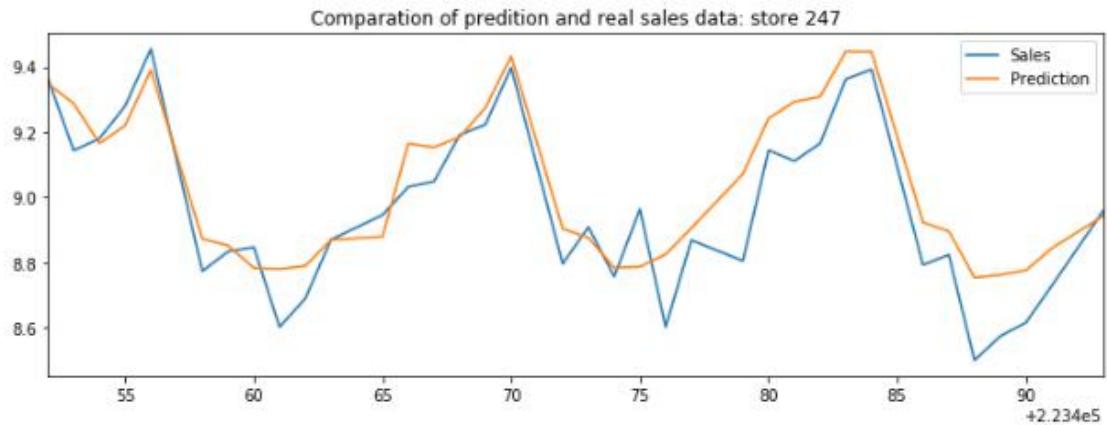
## 完善

这里从两个方面考虑去完善模型，首先是数据的预处理，然后是模型的参数选择。在数据划分时根据时间顺序对数据进行划分，前面六周作为验证集，后面六周作为训练集。此外仍有些可能有用的特征未被利用，如促销月特征；在参数选取方面，要考虑使用网格搜索找到最优参数，同时降低学习率（0.3），提高迭代次数（6000）。

## 结果

### 模型评价与验证

在使用三折交叉验证作为基础的网格搜索对 ‘max\_depth’、‘gamma’、‘reg\_alpha’ 四个参数搜索后，得到的最优结果分别是：15、0.9、10。使用上述参数建立的模型的最终经过 3379 次迭代，训练集得分是 0.183，验证集得分是 0.146。下图为验证集中 247 号商铺实际销售额与预测销售额对比结果，从图中可以看到预测曲线在大趋势上与实际销售额吻合，但存在偏差。



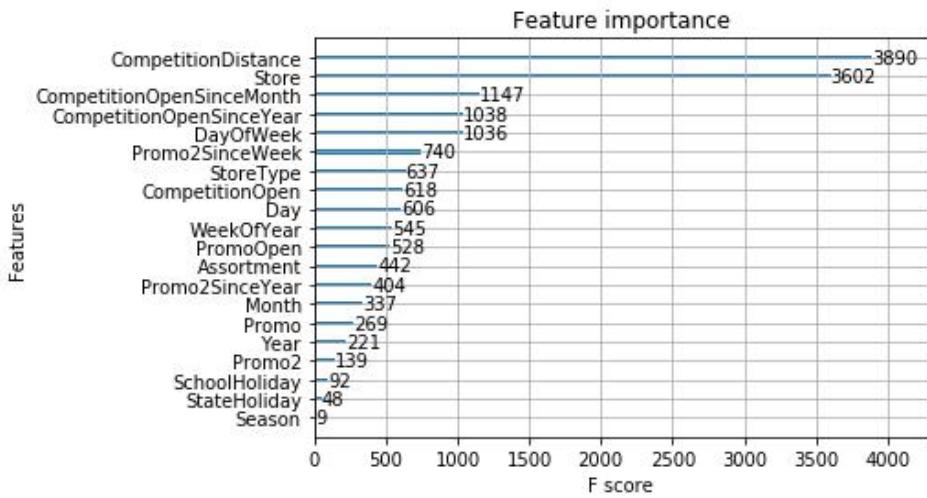
## 合理性分析

最终的预测模型使用的是网格搜索得到的参数，使用这个模型预测的结果提交到 kaggle 上得到的分数是 0.133，而 DecisionTreeRegressor 模型预测结果的得分是 0.156，XGB 模型得分明显优于 DecisionTreeRegressor，说明预测结果也更准确。

## 项目结论

### 结果可视化

这里我可视化了特征重要性，从图中看出 CompetitionDistance 与 Store 两个特征的分数远高于其他特征，应该可以理解为店铺销售额非常受竞争对手的影响，并且大部分不同店铺间销售额差异较大。



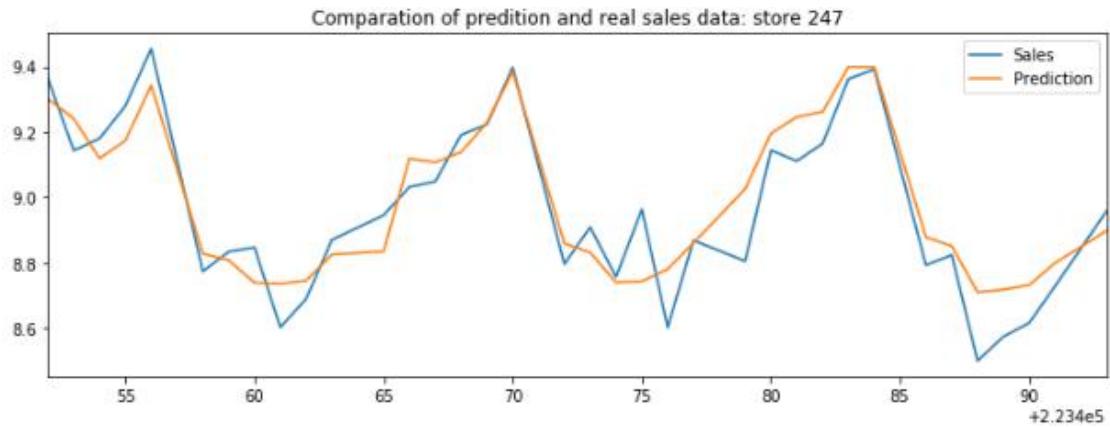
## 对项目的思考

通过这个数据挖掘的项目，使我对数据分析有了更深一层的认识，在处理这种问题时有更明确的方向。对于数据，首先要做的是清洗，去除异常值；然后是数据域数据类型的转换，以适应模型需求；最后是最关键的特征构建，好的特征能够明显的改善预测结果，但要做好特征构建并不简单，需要大量的经验与可视化分析。对于模型，在这个项目上我使用的 DecisionTreeRegressor 与 XGBoost 模型。DecisionTreeRegressor 作为基准模型我之关注了两个参数，所以在调整时相对简单，在使用 XGBoost 时关注参数较多，所以即使使用了网格搜索也消耗了大量时间，我认为理解模型调整参数也算是难点之一。

## 需要作出的改进

在特征工程方面对已有特征的利用以及对与现实情况考虑不够充分，比如消费水平、通货膨胀等问题都会对实际销售额的数据造成影响，使数据产生偏移；在参数调整上，网格搜索的参数太少，每钟参数只有三个选择，应考虑适当加大选择的密度与数量从而获得更精确的结果，但同时需要更多的时间计算。此外在

数据处理上欠缺对异常值的处理。最终的改进方案参考了 Kaggle 上的一个 kernel[4]，对结果乘以 0.995 调整分布，最终得分为 0.126。调整后验证集预测结果对比如图所示。



[1]赵啸彬，基于数据挖掘的零售业销售预测[D].上海：上海交通大学，2010

[2][https://blog.csdn.net/zhihua\\_oba/article/details/72230427](https://blog.csdn.net/zhihua_oba/article/details/72230427)

[3][https://blog.csdn.net/v\\_JULY\\_v/article/details/81410574](https://blog.csdn.net/v_JULY_v/article/details/81410574)

[4]<https://www.kaggle.com/xwxw2929/rossmann-sales-top1/notebook>