机器学习纳米学位

2018 年 11 月 11 日

I.问题定义

项目概述

Rossmann 是欧洲最大的医药连锁商店，大约有 3000 种药品在 7 个不同国家的门店同时销售。为了能够创建一个有效的合理的员工时间表以提高生产力和动力，当前 Rossmann 的管理者要求使用机器学习等技能创建一个模型对未来六周之内的日常销售额进行预测。日常销售额会受到较多的因素影响，例如，促销，竞争对手，节假日，季节性因素等。输入的数据有训练集和测试集，以及门店补充信息。 总结来说，这个问题是一个预测性的问题，也正是机器学习所擅长的，在提供了足够多的数据后（大于 100 万条），机器学习可以更根据这些数据更加准确的进行建模，识别出对于价格因素影响较大的属性，以对未来的价格进行模拟预测。

问题陈述

这个项目是根据真实的数据预测未来的日常销售额，并且在数据集中已经事先给出了影响的因数，可以确定这是一个有监督的回归问题。我将依据历史的销售信息和影响因素(即训练数据和商店类型)，先对数据进行预处理，如处理缺省值或异常值，划分训练集，对 categorical data 和 numerical data 进行操作，选择重要特征或者融合新特征等。接下来， 要训练 XGBoost 和 DecisionTree 两个模型，根据验证集数据进行权重调整。或者选择单个模型，或者两个模型都选。

评估标准

因为是提交到 kaggle 进行测评，所以是采用 rmpse 方法来测评。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。其公式为:

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

其中 y_i 表示单日商店的销售额，yhat_i 表示相应的预测。 在评分中忽略任何一天和 0 销售的商店。

II. 分析

数据的探索

数据从 Kaggle 上下载，共有 3 个，分别是 store.csv, train.csv 和 test.csv。训练集（超过 101 万条数据）和测试集（4 万条数据左右）包含了日常营业额，顾客数，是否营业，是否促销，节假日等特征。门店数据集包含门店类型，分类级别，竞争信息，促销信息等特征。

训练集的特征中是否营业，是否促销，节假日等特征为 category data（即 0 或 1）；顾客数，竞争信息为 numerical data。而这些数据也是作为训练的主要特征。日常营业额将作为所预测的目标。

字段名	含义
Id	代表测试集里的一个 (Store, Date)
Store	每个商店独有的 Id
Sales	销售额 （测试集中需要预测）
Customers	客流量
Open	商店是否处于营业状态
StateHoliday	国家假日类型. a 代表公休假日，b 代表复活节，c 代表圣诞节，0 代表不处于假日状态
SchoolHoliday	商店是否受学校放假的影响
Storetype	商店的类型， a,b,c,d 四种
Assortment	商店销售品的分类等级
CompetitionDistance	和最近竞争者的距离
CompetitionOpenSince[Month/Year]	最近的竞争者开张时大概的年（月）
Promo	商店当天是否处于促销状态
Promo2	商店是否处于一个持续的促销状态中
Promo2Since[Year/Week]	店铺开始参与 Promo2 的年（月）份
PromoInterval	Promo2 持续运行的间隔

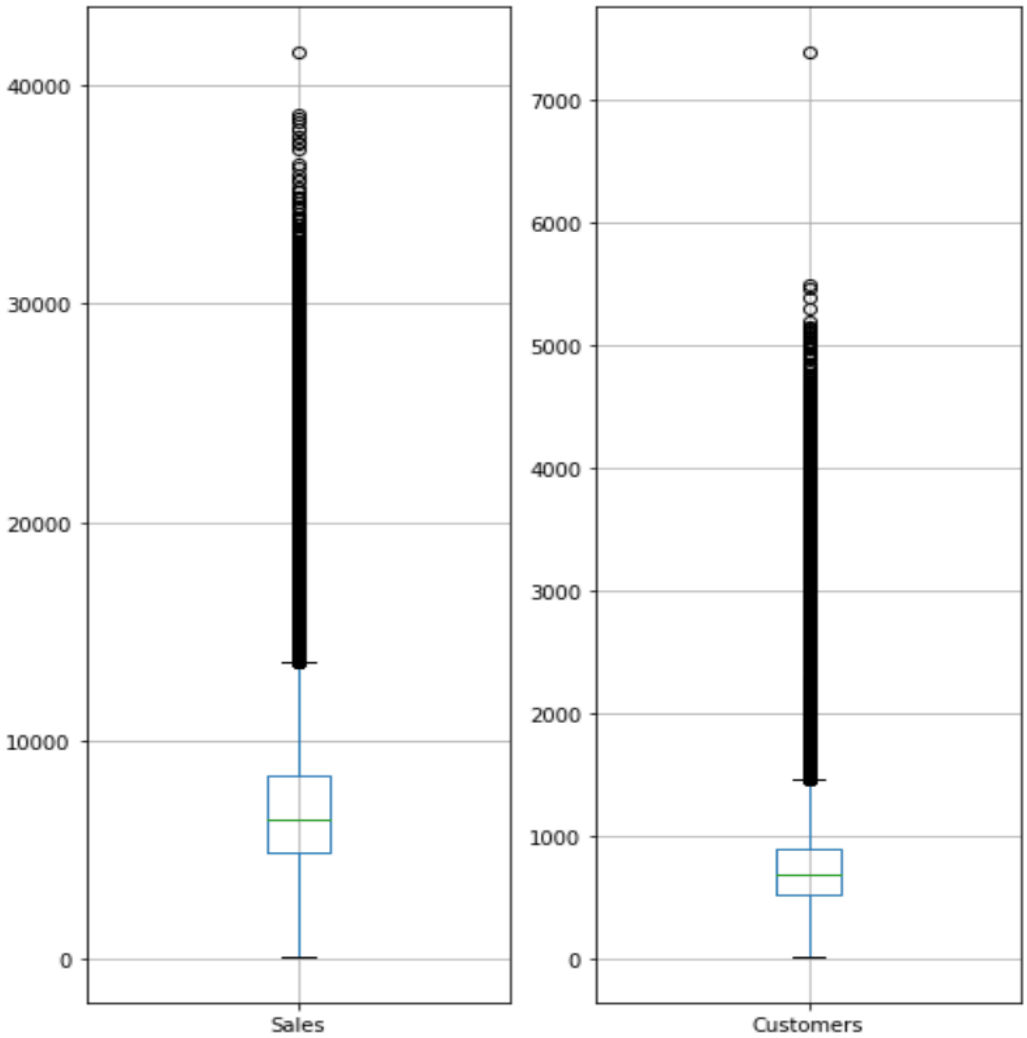
在如此大量的数据集中，应该会有缺省值或者异常值。对于缺省值，如果其所占的比例对于整个数据集而言不大，可以考虑将其删除或者用平均值来填补，比如，因为销售额为 0 的记录意义不大，所以本次实验只采用了店铺为开且销售额大于 0 的数据。如果占据相当大的比例，比如整个属性大部分都为空，可以考虑弃用这个属性。 另一方面，对于异常值，因为这个问题是一个回归问题，对于异常值较为敏感。可以使用

可视化（如四分位法或方差处理）确认异常值不是重要的因素（如安然案件中两位老板的数据看似异常值，但不能去除）,如果不是重要的因素，则直接删除。

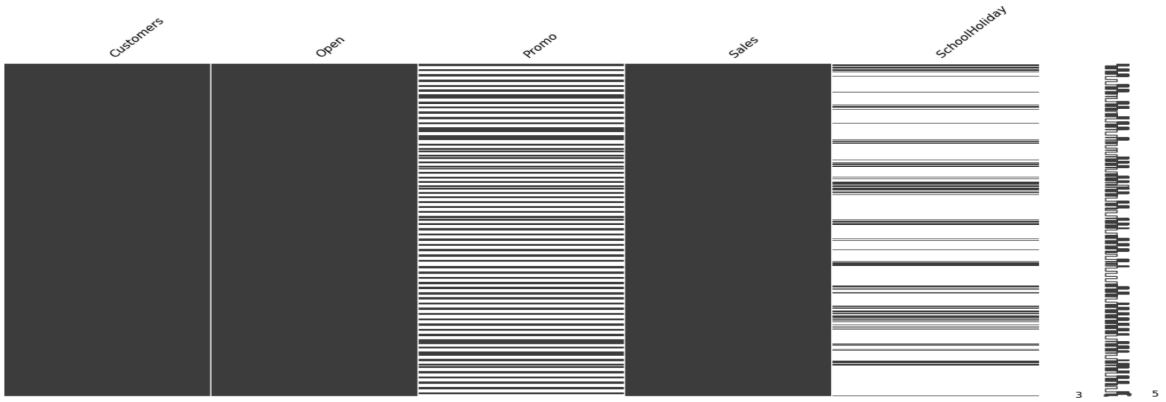
探索性可视化

经过初步的探索，发现 Customers 和 Sales 的异常值较多。

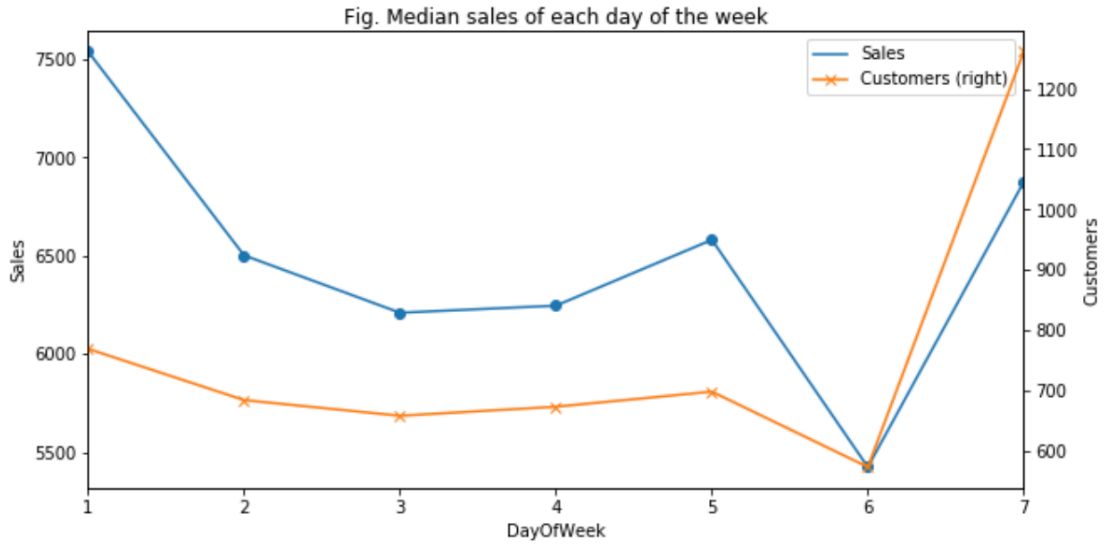
Fig. The box plot of Sales and Customers



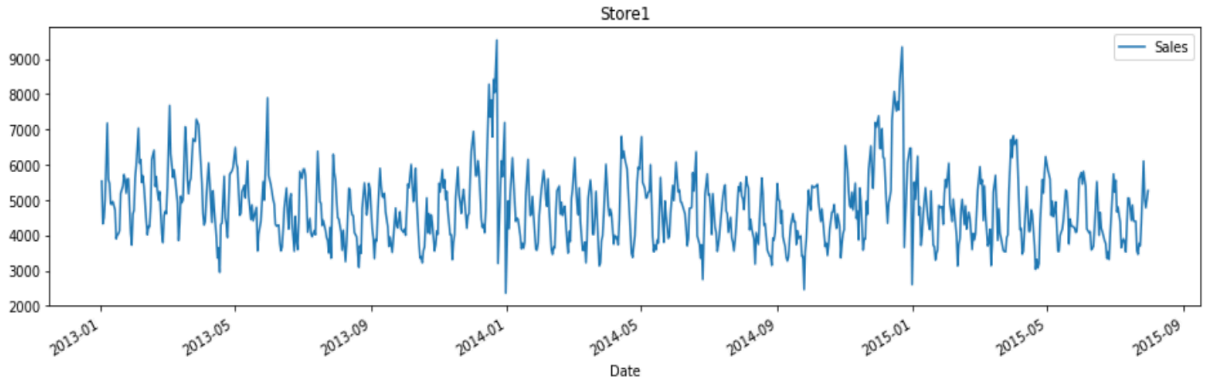
这是 Customers, Open, Promo 等几个特征的缺失情况，可以看到 Customers, Open 和 Sales 的缺失情况基本是一致的，因此在预处理时可通过简单过滤来丢弃缺失值。



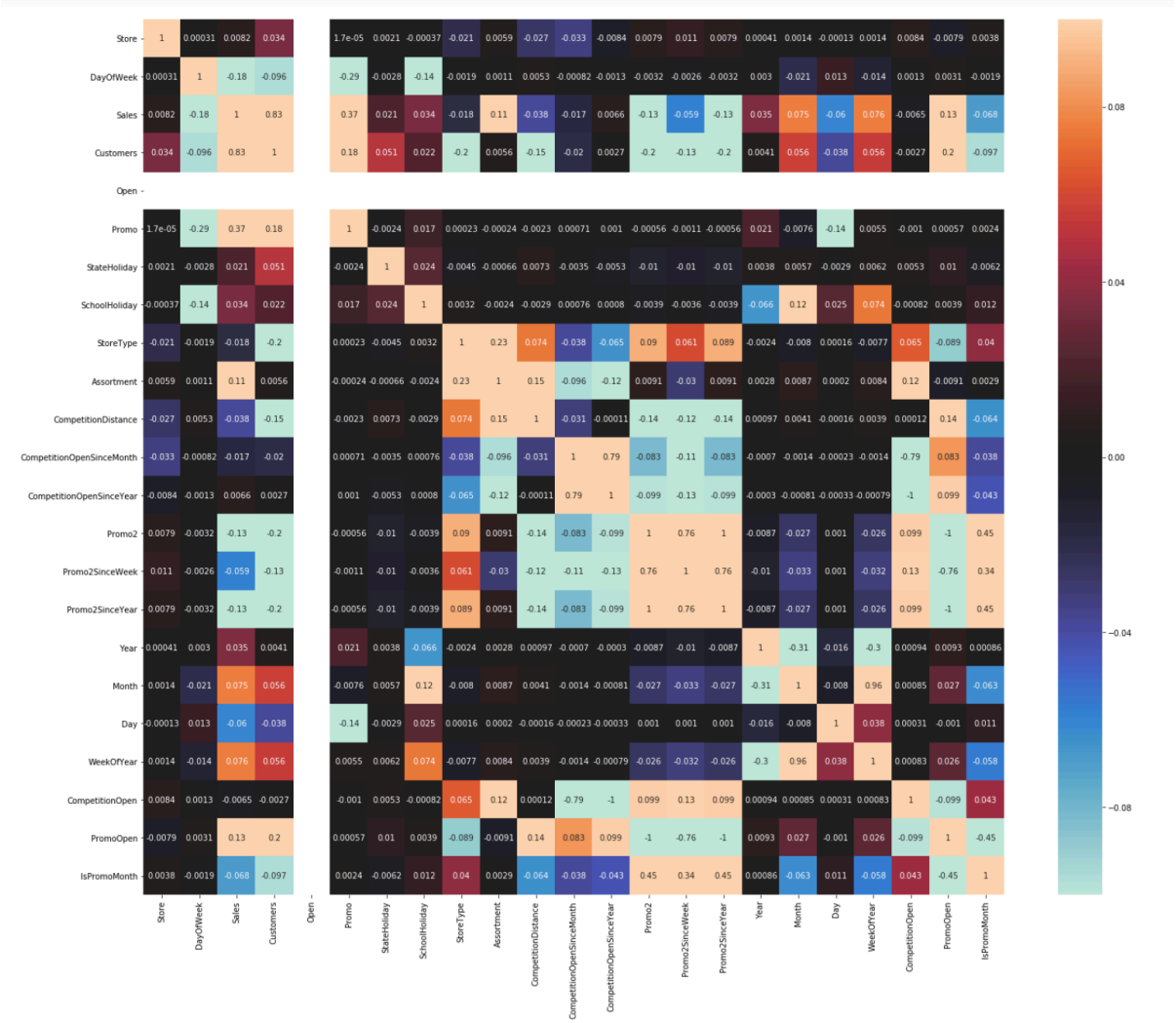
这是一周中每一天的平均销售量的时线图，从图中可以看到，在周末的时候，销量达到最高，而后开始下降，周六是平均客人最少的一天。在周日客流量最高的时候，销量反而不高，周一客流量不高的时候，销量反而是最高的。



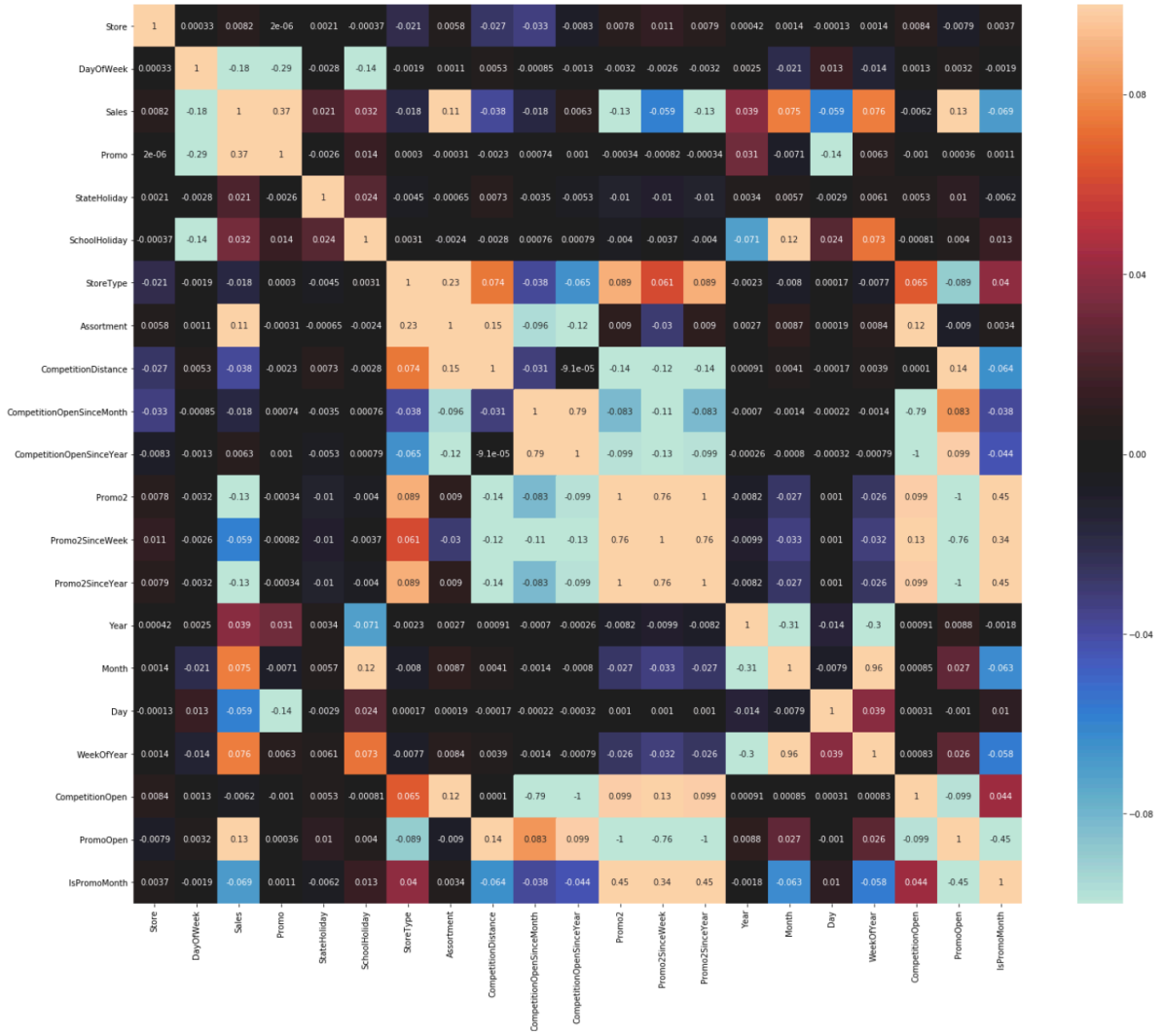
这是 2013 至 2015 年各月份平均销售量的变化折线图，销售的高峰是在圣诞节前后，圣诞过后有一波销售低谷。



各个特征与 ‘Sales’ 标签的相关性分析，可以帮助决定删除哪些相关性不是很大的特征，以避免训练过程中的过拟合和资源浪费。以下是未处理的特征相关性可视化



这是处理过后的可视化图，



算法和技术

从可视化中可以看到销售额与时间密切相关，但还会有很多其它特征对它造成影响，比如离竞争者的距离、是否处于学校放假期间和是否有进行广告推广等等。而且数据中的缺失值和离群值都会对销售额的预测造成影响，为了达到最好的预测精确度，首先需要丢弃掉数据中 Sales 和 Customers 这两个重要特征值偏离正常值过大的数据。

第二步数据预处理以及特征处理和转化，包括处理缺失值和对重要特征进行合适的改变或根据旧特征生成部分新特征以方便训练模型。比如，将不同字符的特征转化为数字，如 ‘mappings = {'0':0, 'a':1, 'b':2, 'c':3, 'd':4}’ 。由于和时间特征关系较为密切，所以将时间特征进行拆分和转化，并加入 ‘weekOfYear’ ，'CompetitionOpen'和'PromoOpen'作为新的特征。将'PromoInterval'特征转化为'IsPromoMonth'特征,表示某天某店铺是否处于促销月

第三步，我之前尝试过使用 cross_validation.train_test_split 将训练数据随机地划分为训练集和测试集，之后开始建模、测试并提交到 Kaggle 进行评分，但是始终无法达到之前所设定的 0.11 左右的标准，所以在通过搜索后，我尝试使用另一种做法处理数据。并且只训练 XGBoost 模型。

- **XGBoost**：这是一种适用于监督学习的模型，Gradient Boosting 本身优秀的性能加上 XGBoost 高效的实现能令它在本项目中很好的表现，而且 Kaggle 比赛的第一名主要就是采用了这个算法对结果进行预测。XGBoost 自定义了一个数据矩阵类 DMatrix，会在训练开始时进行一遍预处理，从而提高之后每次迭代的效率。在监督学习下，一般线性模型使用给定的训练集（包含多维特征 X_i）预测目标变量 Y_i，其公式为：

$$\hat{y}_i = \sum_j \theta_j x_{ij}$$

并根据任务是回归还是分类的不同，预测值有不同的解释。且参数是不确定，需要从数据中学习。在线性回归里，参数是指系数 θ。

一般来说目标函数包含两部分：训练集损失和正则化项

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta)$$

Logistic 损失的计算依据如下公式：

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})]$$

正则化项控制模型的复杂度，避免过拟合。基于以上的公式，推导出的 XGBoost 模型数学公式为：

$$\text{obj}(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

XGBoost 的原理是在预测每个树的得分，将训练错误（得分较低）的样例增加更大的权重，而后投入到下一次训练中，这样下次训练就更加容易辨别出错误分类的样例。最终将这些弱分类器进行加权相加。在本次训练中，根据使用的的权重更新调整，有如下公式：

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

其中：

$$G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$$

而 XGBoost 模型需要调试的主要参数如下

- eta: Shtinkage 参数，用于更新子节点权重时，乘以该系数，相当于学习率
- nthread: 线程数
- subsample: 样本随机采样，较低的值使得算法更加保守，防止过拟合，但是太小的值也会造成欠拟合
- colsample_bytree: 列采样，对每棵树的生成用的特征进行列采样.一般设置为： 0.5-1

基准模型

项目采用 RMSPE 作为评价指标，其分数越低模型性能越好。根据 Kaggle 上的数据，我猜测合格数值大概是 0.2 左右 。但是我将会把基准设定在 0.11773，争取进入 private leaderboard 的 top 10%。

III.方法

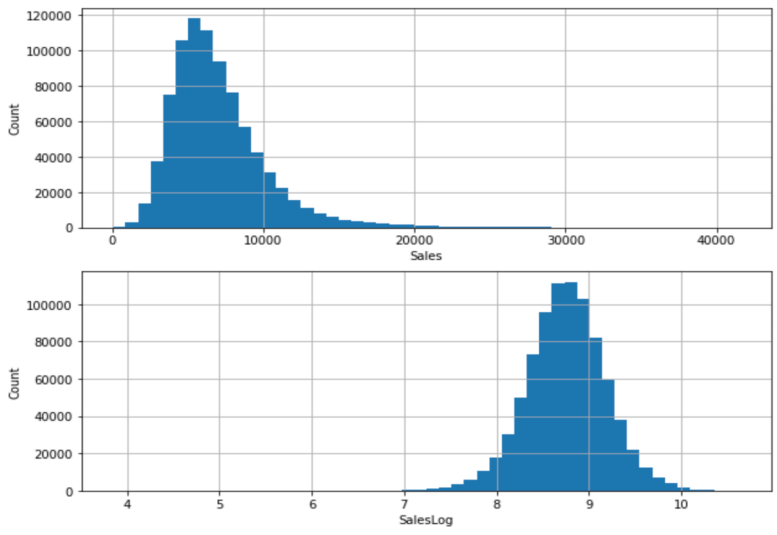
数据预处理

首先是对缺失值用 0 或 1 进行填充，如对 test 数据中的 open 数据补位 1，而 store 中的缺失数据大多数与竞争对手和促销相关，测试发现填充 1 的效果并不好，所以使用 0。之后如上所述对时间特征进行了比较细致的划分，主要是将包含年月日的日期拆分为 Day、Month、Year，DayOfWeek, WeekOfYear 等特征。同时商店信息中包含的竞争者信息和推广信息这里也进行了处理，新增两个新特征：

1. CompetitionOpen 是根据 12 * (Year - CompetitionOpenSinceYear) + (Month - CompetitionOpenSinceMonth)计算得出,并过滤出大于 0 的数据，如果不大于 0 则为 0.
2. PromoOpen 是根据 12 * (Year - Promo2SinceYear) + (WeekOfYear - Promo2SinceWeek) / 4.0 计算得出，并过滤出大于 0 的数据，如果不大于 0 则为 0.

目的是为了计算某天某店铺的竞争对手已营业时间和店铺已促销时间。
然后在通过可视化探索之后，我将异常值和相关性不高的特征进行删除处理。以避免影响到回归模型的训练和预测。
对数据集连续型特征根据数值分布情况做归一化处理，例如 Sales 特征，我将会进行对数缩放处理，其结果可视化如下：

Fig. The distribution of Sales and SalesLog



对离散型特征则进行 One-hot coding 处理 。

执行过程

将最近的 6 周数据作为 hold_out 数据集进行测试，代码如下：

train = train.sort_values(['Date'],ascending = False)

ho_test = train[:6*7*1115]，ho_train = train[6*7*1115:]

并只取 ‘open’ 不为 0，且销售额大于 0 的数据。然后将特征与标签进行拆分，并将标签进行对数处理，使数据分布趋于正态，以避免数据分布出现倾斜。

接下来 XGBoost 模型的训练参数如下：

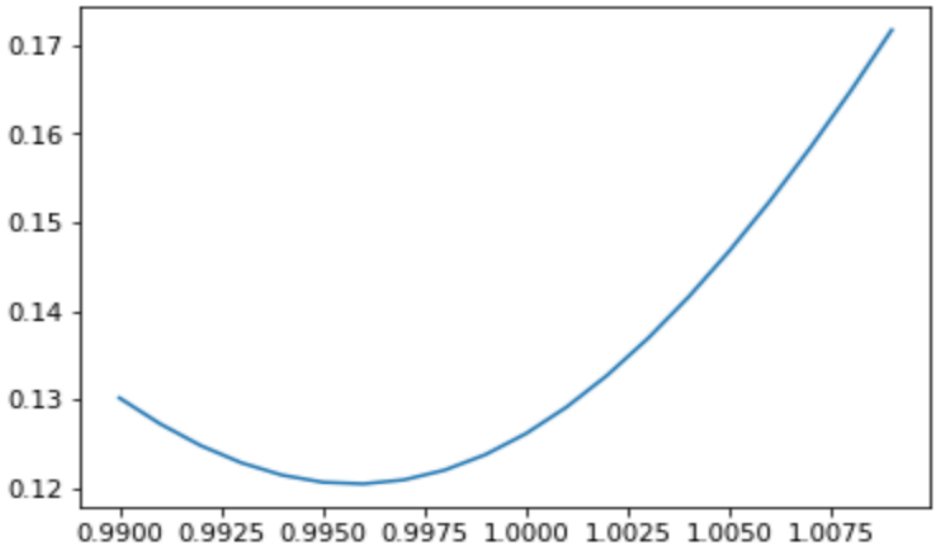
```
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.03,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.7,
          "silent": 1,
          "seed": 10
          }
num_boost_round = 6000
```

在训练完成之后，用之前保留下来的测试集进行检测，并保留数据集的预测结果。

完善

在完成预测后，需要对结果进行分析，这里只分析偏差最大的 10 个预测结果。从以下的测试结果中可以看到，最佳的权重为 0.996，得分为 0.120463。

```
Best weight for Score:0.996    0.120463
dtype: float64
```



但是在考虑到模型的泛化性之后，接下来的调整，还是以 0.990 为校正系数进行模型的校正。代码如下：

```
for i in L:
    s1 = pd.DataFrame(res[res['Store']==i+1],columns = col_1)
    s2 = pd.DataFrame(xtest[xtest['Store']==i+1])
    W1=[(0.990+(i/1000)) for i in range(20)]
    S =[]
    for w in W1:
        error = rmspe(np.expm1(s1.Sales), np.expm1(s1.Prediction*w))
        S.append(error)
    Score = pd.Series(S,index=W1)
    BS = Score[Score.values == Score.values.min()]
    a=np.array(BS.index.values)
    b_ho=a.repeat(len(s1))
    b_test=a.repeat(len(s2))
    W_ho.extend(b_ho.tolist())
    W_test.extend(b_test.tolist())
```

校正过后，将多个校正模型和初始模型融合并重新训练。

通过资料的查询，这里使用加权融合，理由是根据资料的查询，其效果比简单的平均融合更好。

IV. 结果

模型的评价与验证

以下是初始模型和融合修正后的模型的评分:

Submission and Description	Private Score	Public Score	Use for Final Score
Rossmann_submission_1.csv a few seconds ago by Howie Chen raw model	0.13065	0.12750	☐
Rossmann_submission_weight.csv 5 minutes ago by Howie Chen testing one	0.11246	0.11181	☐

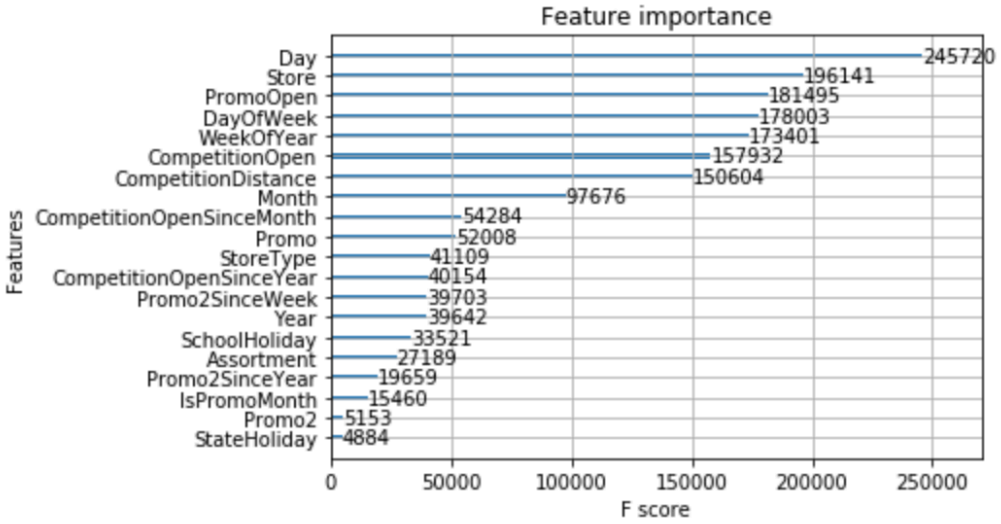
可以看出，最开始训练出来的模型的评分已经较高，但是，融合模型把分数提高了 0.01 分还更多，表现更加的优秀，public 分数大于 private 分数，也从侧面说明了这个表现是合理可信的，且融合模型的泛化能力比原始模型更加优秀。

合理性分析

XGBoost 模型相比较于综合决策树模型，最大的优势就是把时间给大大缩短了。在调整后训练出来的模型因为其相互之间较为独立，打下了融合的基础，在优化过后，分数又有大幅度的提高，更说明融合的模型是合理的。

V. 项目结论

结果可视化



从图中可以看出 XGBoost 对时间比较敏感，这个学习器也主要是从 Rossmann 销售额的时间规律中学习，而后是各个店铺的特性。可以说，XGBoost 模型对于每个不同特性的店铺都能有较为准确的预测。这也说明 XGBoost 模型对于以时间为基本数据的预测较为可靠，且修正融合后效果更好。

在得到最终模型后，对于任意的三个店铺进行了预测和真实销量对比，其中一个店铺的可视化如下：

可以看出，模型所展现出的预测与真实销量对比而言，大致还是较为接近的。也从侧面说明模型训练的较好。

对项目的思考

其实这个项目做了两次，耗费了大量的时间进行训练，第一次是根据之前的经验使用决策树模型和 XGBoost 融合，并进行预测，最终的得分在 0.15 左右，无论经过几次的参数调整，分数一直都是固定在这个范围，始终无法达到之前定下的目标。后来经过探索和查询，发现可以使用 XGBoost 模型单独进行训练，且校正调整融合之后的模型表现的更好。当然，在进行这个训练时，比较困惑的是项目特征的选择，清理和转

化，有几次把重要特征给删除了，有时候不知道哪个是重要特征；有时候不知道使用什么可视化图去更加直观的表达我所想呈现的效果；在调整的时候，校正系数无法确定，不知道哪个权重是最为合适；模型的融合方式是均值融合还是加权融合。所幸，最后一步步都是完成了，达到了预定的目标。

需要作出的改进

虽然最终的分数已经达到了预设的目标，但是还是有很多的提高的空间，比如：

- 预处理：对缺失值和异常值的处理，在项目中，异常值是直接丢弃的，如果能先预训练一个模型对缺失值和离群值进行重新预测也许能得到更好的效果；此外，离群值也许可以采用基于方差的选择会更加合理
- 特征的选择：一个好的特征划分或者特征转化，能够大大的提升预测结果，我想在特征的选择和处理方面还可以提高
- 模型选择：这里只选用了 XGBoost 模型，或许可以尝试一些线性模型来进行 Ensemble
- 尝试使用 Stacking 方法

参考链接

理解在 Otto 数据库中的 XGBoost 模型，来自

https://www.kaggle.com/tqchen/understanding-xgboost-model-on-otto-data

Rossmann-kaggle 项目，来自

https://www.kaggle.com/c/rossmann-store-sales/leaderboard

监督回归学习，来自

https://cn.udacity.com/

【机器学习】模型融合方法概述，来自

https://zhuanlan.zhihu.com/p/25836678

如何在 Kaggle 首战中进入前 10%

https://dnc1994.com/2016/04/rank-10-percent-in-first-kaggle-competition/