1. 项目概述

Rossmann 劳诗曼，是德国最大的日化用品超市。截止 2012 年，Rossmann 一共有 2600 家连锁店，其中 1600 多家位于德国境内。 2017 年 4 月，ROSSMANN 的 2000 多家实体店和网店将接受来自中国的游客、商务人士和留学生的支付宝支付。

Rossmann 的门店经理希望通过其历史销售数据，来预测未来六周的销售情况，以便提前进行合理的资源分配，从而提高各资源和存货的使用效率。

门店的销售额会受到很多因素影响，包括促销、同类门店竞争、节假日、季节和地域等。该项目的目标就是基于 Rossmann 各门店的信息(促销、竞争、节假日、季节性、门店类型等)以及历史销售额和顾客流量来预测未来六周的销售额。显然该问题属于机器学习中的回归问题(regression)，那么我们需要从数据中提取可能对销售额有影响的特征(feature)，并建立合适的回归模型进行预测。问题解决大致分为以下五个步骤：

1. 数据初步分析与探索：尝试了解数据的一些基本情况，例如数据分布、缺失值情况、不合理数据等，为数据预处理做准备。
2. 数据预处理：进行数据类型格式化、编码、缺失值填充、时间信息转化等，为数据特征提取做准备。
3. 特征提取：通过特征工程算法，进一步提取出额外的特征供后续模型训练使用。
4. 建立回归模型并进行预测：项目采用 xgboost 库建立模型。
5. 优化及结果进行分析，提高预测准确度。

1.1. 评价指标

考察两种评价指标，分别是均方根误差(the Root Mean Square Error -- RMSE)和均方根百分比误差(the Root Mean Square Percentage Error -- RMSPE)，计算方式如下：

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

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

其中 y_i 表示门店 i 某天的销售额， $\hat{\mathrm{y}}_i$ 表示该门店对应的某天销售额预测值。对于门店的销售额数据来说，采用百分比误差可以有效降低高销售额数据对最终误差的影响，因为促销日或者节假日可能会比平时的平均销售额高出不少，如果采用均方根误差，高销售额数据会对评估产生较大影响，从而影响模型好坏的评估。

2. 数据初步分析与探索

2.1. 原始数据解读

项目数据集包含 train、store 和 test。train 数据集记录了 1115 家门店从 2013 年 1 月 1 日到 2015 年 7 月 31 日的销售信息共计 1017209 条数据。 test 数据集记录了 1115 家门店从 2015 年 8 月 1 日到 2015 年 9 月 17 日(48 天)的销售信息共计 41088 条数据，但是缺少销售额(Sales)和顾客数(Customers)，其中销售额是我们需要进行预测的。 store 数据集中包含了这 1115 个门店的额外信息。下面详细介绍各数据集的字段含义

- Store: 门店的 Id 标识
- DayOfWeek: 周一到周日的数字标识
- Date: 日期
- Sales: 该门店在对应日期的销售额
- Customers: 该门店在对应日期的顾客数
- Open: 该门店在对应日期是否营业(0 表示关闭，1 表示营业)
- Promo: 表示该门店是否在对应日期进行促销(0 表示没有，1 表示进行促销)
- StateHoliday: 表示对应日期是否为国家法定节假日(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: 表示对应日期是否为学校假期(0 表示不是，1 表示是)
- 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起始周数
- Promo2SinceYear: 表示一个门店进行持续促销的起始年份
- PromoInterval: 表示门店促销周期的月份
- Id: test 数据集特有的用于标识提交数据顺序的编号

2.2. 数据探索

不难发现在 train data 里有存在销售额为 0 的数据项：一种是未营业销售额为 0；另一种是正常营业销售额依旧为 0。第一种情况无需预测，不营业销售额一定为 0，第二种情况可能是数据录入有误(比如该门店其实未营业)，因此在我们对数据做进一步观察前，先去除所有销售额为 0 的项。train data 没有缺失数据，无需进行缺失值填充。

在机器学习应用中，我们往往会使用很多列数据，并寻找他们之间的关联度。作为一种密度图，热力图一般使用具备显著颜色差异的方式来呈现数据效果，热力图中亮色一般代表事件发生频率较高或事物分布密度较大，暗色则反之。热力图最终效果常常优于离散点的直接显示，可以在二维平面或者地图上直观地展现空间数据的疏密程度或频率高低。

合并 train data 和 store data 后观察热力图(相关性矩阵)：

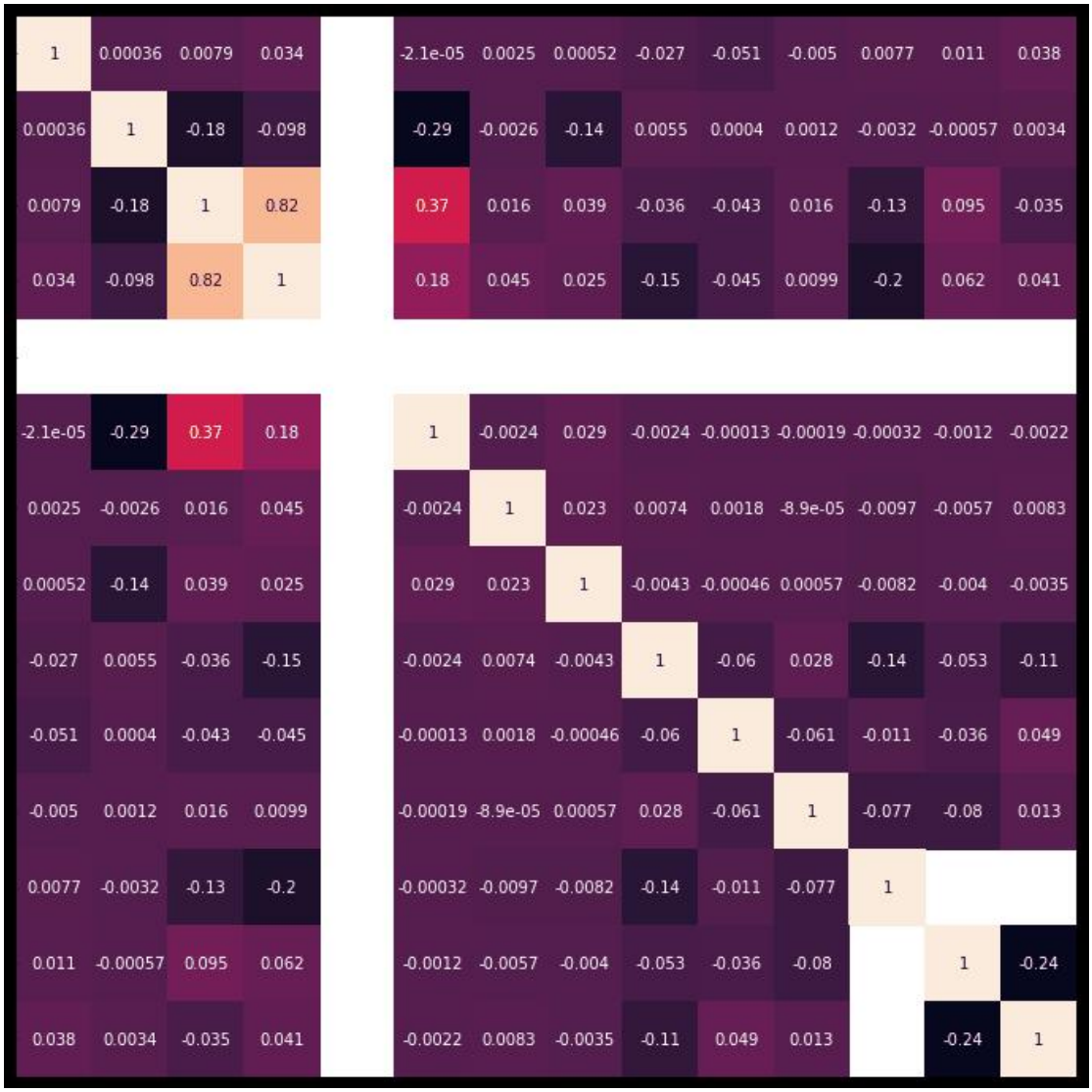


图 1

可以看出，很多特征之间都具有一定的正相关性或者负相关性，这往往意味着这些数据之间是有一定的关联度的， 也就是说我们可以将这些数据使用机器学习模型进行回归。

2.2.1. 销售额和顾客数的关系图

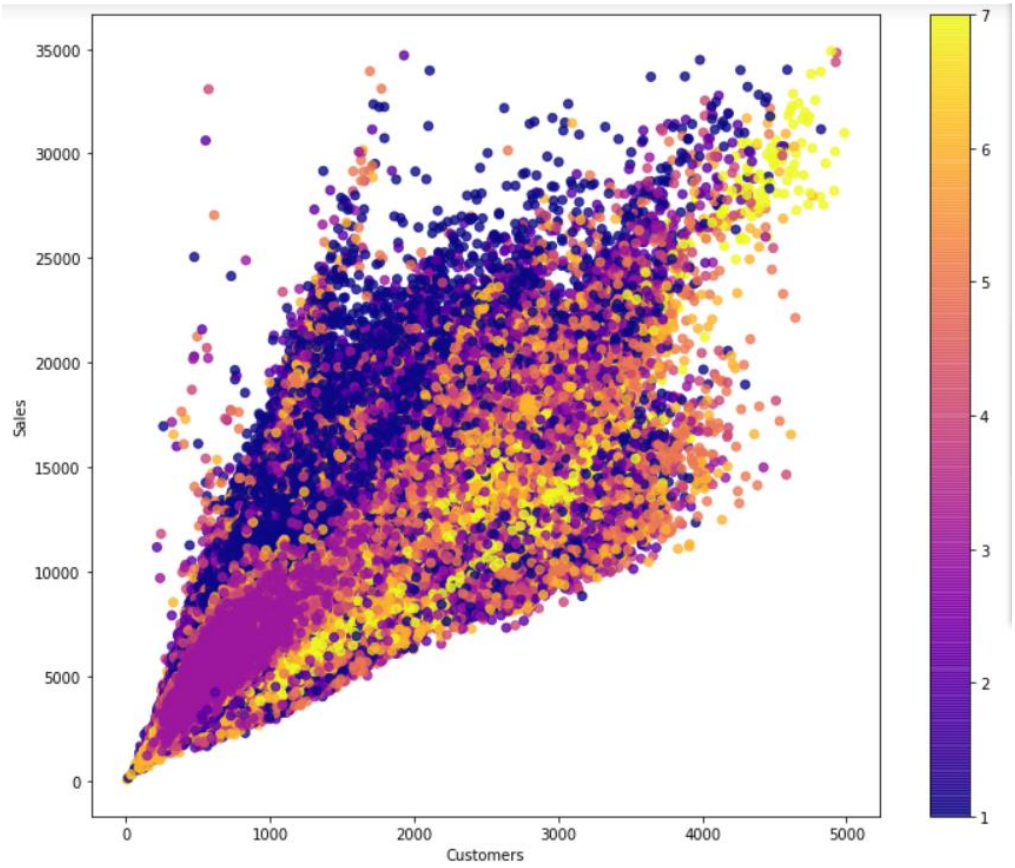


图 2

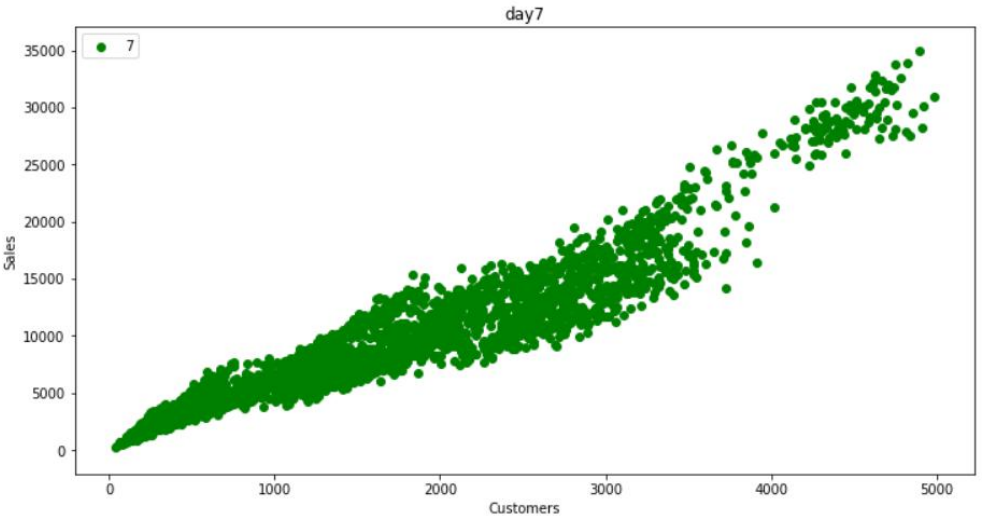


图 3

可以观察每周七天的数据，顾客和销售额基本呈线性递增关系(图 3 星期天数据呈现更为明显)。即顾客数越多，销售额就越高。在进行预测时，顾客数据也是未知数，因此训练模型时，不能使用 customer 特征值。为简化起见，下面的图示不再展现顾客数据(可在项目代码查看)。

下图为销售额数值分布图：

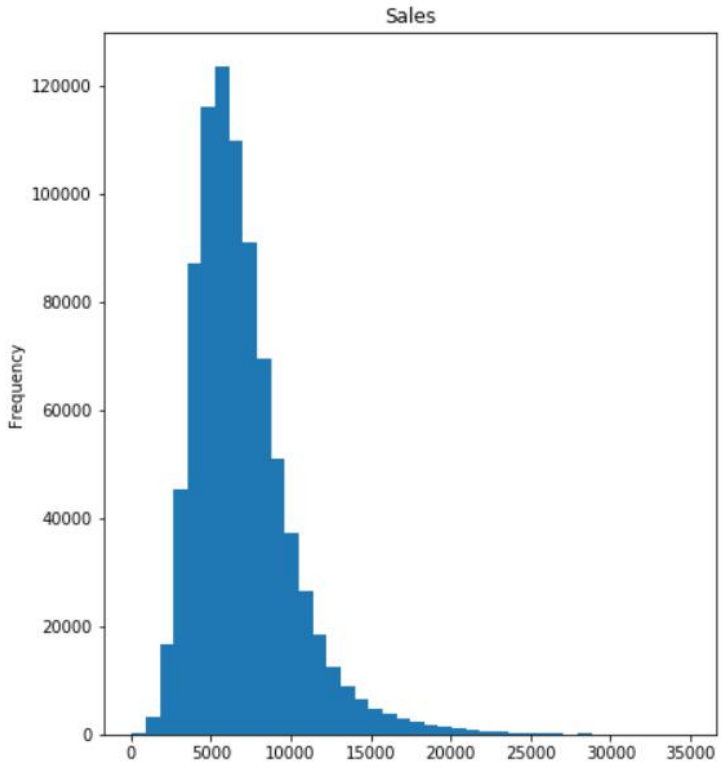


图 4

可以看出销售额数值呈正态分布，大部分集中在 5000-6000 范围。接着观察每周七天平均每天的销售额情况：

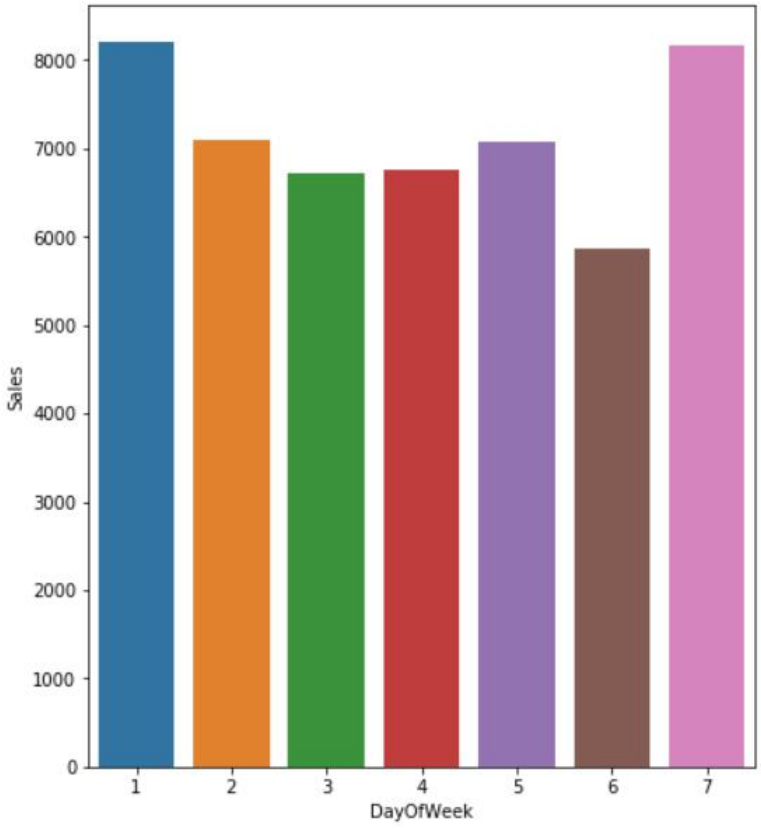


图 5

可以看出星期一和星期天的平均销售额高于其它时间。以每月为单位观察一年的销售额情况：

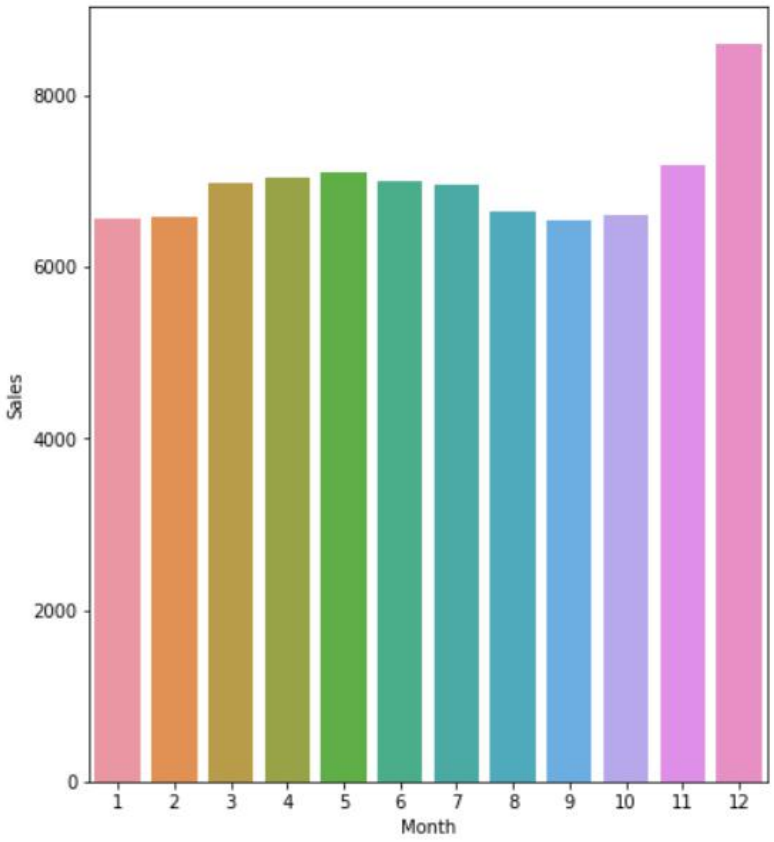


图 6

可以看出年末 12 月的销售额要远高于其它时间，可能是因为年末的商家促销和大家年末购物习惯导致。观察所有数据图 7 可以发现符合上述结论，每年年末都会有一个销售额的增长：

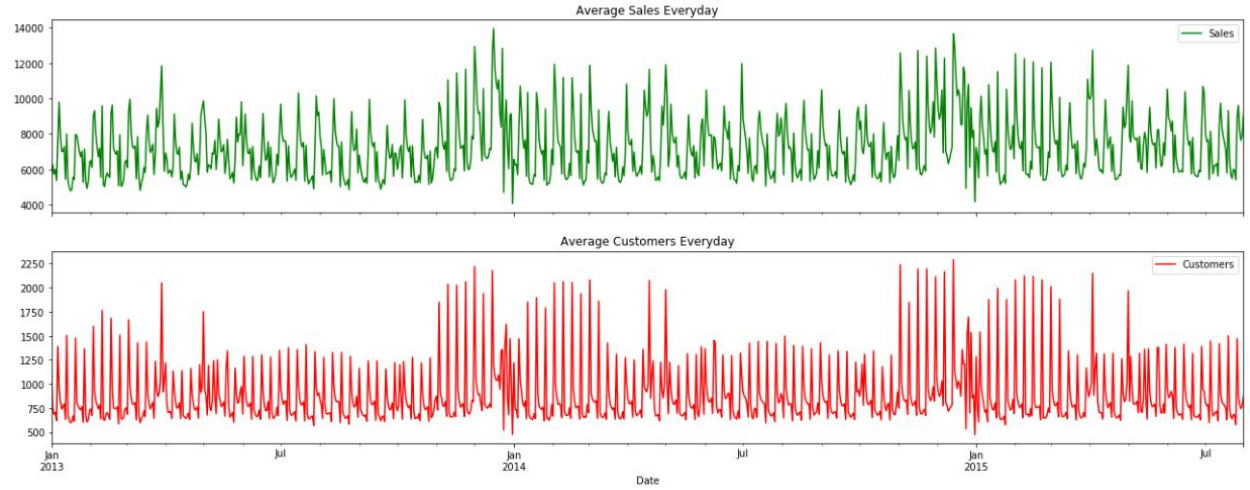


图 7

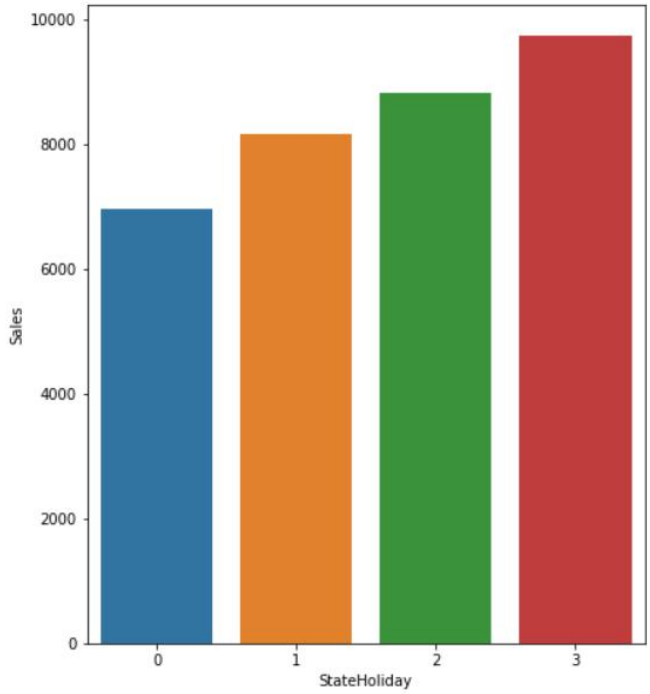


图 8

图 8 可以看出圣诞节销售额会提高许多，这也符合刚才年末销售额增加的结论。

下面观察一下竞品商店距离和销售额的关系图：

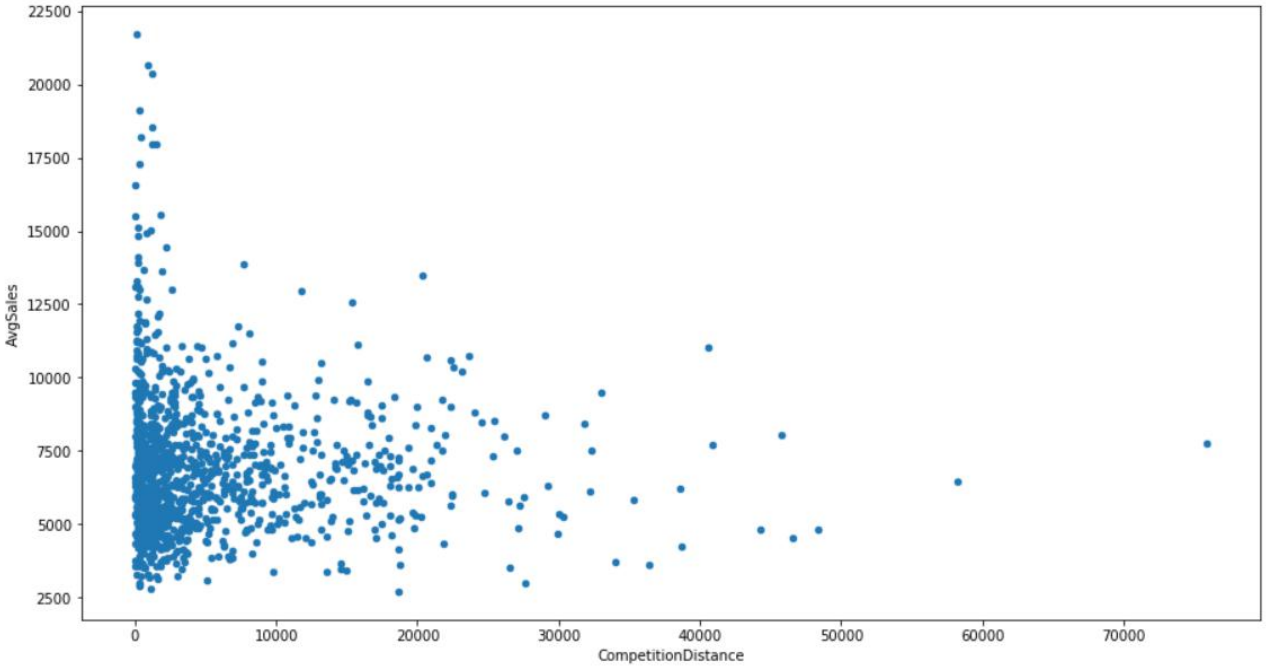


图 9

可以看到在竞品商店距离为 0(附近没有竞品商店)的情况下，平均销售额可能会高于其它情况。另外如果竞品商店在 0-20000 范围内，销售额的区间会整体降低。可以看出竞品商店确实会对销售额有影响。

下面观察促销对销售额影响：

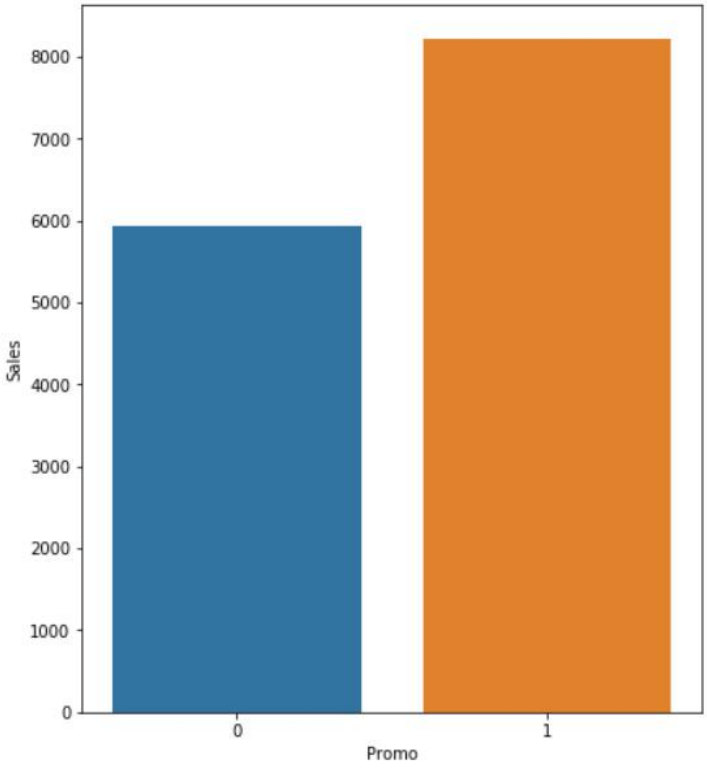


图 10

毫无疑问，促销会对销售额带来显著提升。

最后观察商店类型和上架商品类型对销售的情况影响。

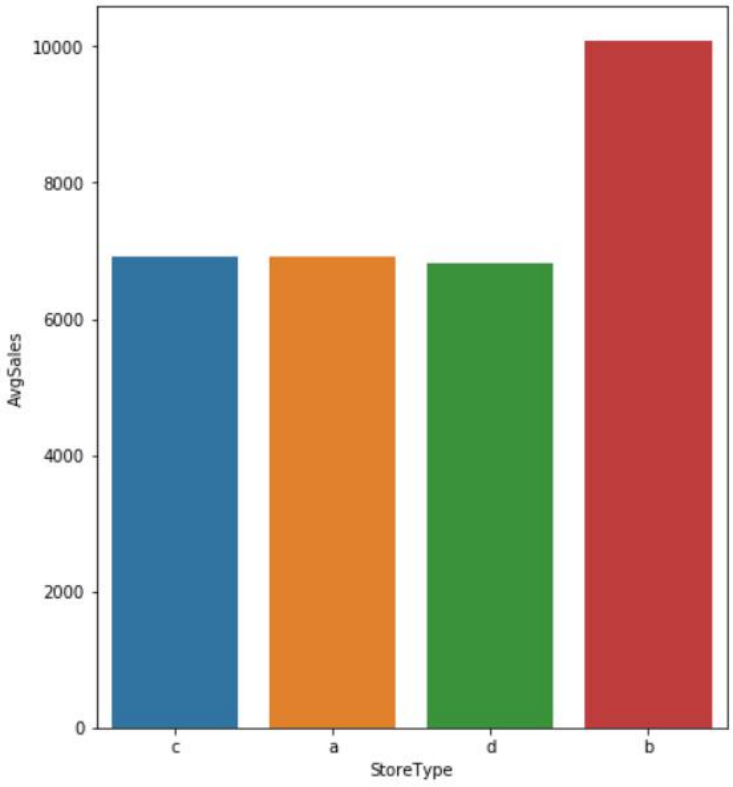


图 11

可以看出 adc 三种类型的商店销售额无太大差异，但是 b 类型的商店销售额明显高于其它类型。

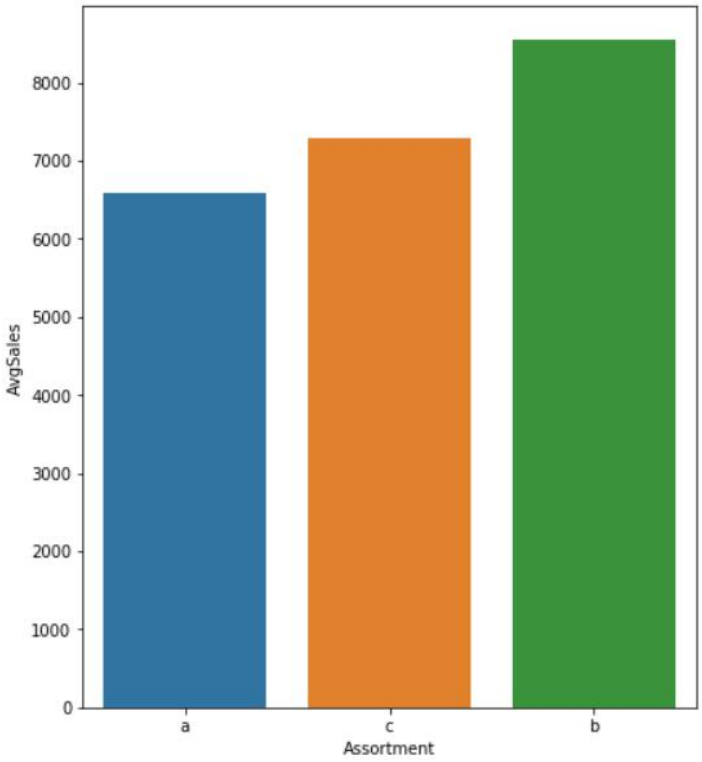


图 12

上架商品的类型对销售额有较大影响，a 类最低，c 类中等，b 类可以获得最高的销售额。

在对训练数据有了较为深刻的认识以后，开始着手数据预处理工作。

3. 数据预处理

3.1. 缺失值处理

通过检查发现，训练数据 train data 不存在缺失值。

测试数据 test data 存在[open]缺失值。显然 open 如果为 0 则销售额肯定为 0，没有预测的必要。因此我们将所有测试数据的 open 缺失值填充为 1。至此所有数据的 open 值为 1 在后续不作为特征值进行训练。

商店信息 store 存在很多缺失值。比如 CompetitionDistance 缺失值，这里我们将其全部填充为 0，表示附近没有竞争商店。最后将其它所有缺失值也填充为 0。

3.2. 去除训练数据中销售额为零数据

训练数据存在销售额为 0 的情况，这里分为开业和未开业的两种场景。未开业销售额一定为零，由于我们的测试数据已经全部是开业情况，因此将训练数据的未开业项也删除。开业但是销售额为 0 的情况也存在，其中有顾客 52 项，没有顾客 2 项。这里可能的原因也许

是录入数据出错，或者确实没有顾客到访。为了降低此特殊情况对预测的影响，将所有销售额为零的项全部去除。

3.3. 日期处理

将测试数据和训练数据的日期Date进行分离，转换为Year、Month、DayOfMonth、WeekOfYear、DayOfYear 五个新 feature。这样就把日期 date 类型数据转变为了整型日期值。

3.4. 格式化数据类型

数据中的 StateHoliday 有零值和字符'0'的情况，它们均表示相同的国家法定节假日。将数值 0 转变为字符'0'。训练数据集和测试数据集的 StateHoliday、DayOfWeek，以及 store 数据集的 Assortment、StoreType 都是字符类型，全部转换成 category 类型并进行编码。

这样所有数据的格式处理完成。

4. 特征值提取

提取特征的好坏很大程度上决定了模型最终的表现，所以特征提取是整个项目执行过程中非常重要的一个环节。参考了很多其他同学的经验总结，此次项目我最终选取了以下特征值：

- Store: 门店的 Id 标识
- DayOfWeek: 周一到周日的数字标识
- Promo: 表示该门店是否在对应日期进行促销(0 表示没有，1 表示进行促销)
- Promo2: 表示一个门店是否进行了持续的促销
- Year: 年份
- Month：月份
- WeekOfYear：一年之中的周数
- DayOfWeek：星期几
- DayOfMonth：几号
- DayOfYear：一年之中第几天
- StateHoliday: 表示对应日期是否为国家法定节假日
- SchoolHoliday: 表示对应日期是否为学校假期
- StoreType: 一共四种不同类型的门店
- Assortment: 描述门店上架的商品类型
- CompetitionDistance: 表示与最近的竞争对手门店的距离
  - CompetitionOpen：竞争商店已营业时间(月数)
  - PromoOpen：店铺已经促销的时间(月数)
  - IsPromoMonth: 店铺是否处于促销月份
  - AvgSales：店铺历史平均销售额
  - AvgCustomers：店铺历史平均顾客数

- AvgSalesPerCustomer：店铺历史平均每位顾客消费额
- MedianCustomers：店铺历史顾客中位数
- HolidayThisWeek：本周节假日天数
- HolidyLastWeek：上周节假日天数
- HolidayNextWeek：下周节假日天数
- AvgSalesPerDow：历史星期几的平均销售额
- MedianSalesPerDow：历史星期几的中位数销售额
- AvgCustomersPerDow：历史星期几的平均顾客数
- MedianCustomersPerDow：历史星期几的中位数顾客数

其中空心圆部分是额外提取的特征值。

5. 建立回归模型并进行预测

5.1. 建立模型

xgboost 是 2014 年 2 月诞生的专注于梯度提升算法的机器学习函数库(Gradient Boosting Machine Learning)，作者陈天奇，采用 C++实现，充分利用 CPU 的多线程进行并行计算。此函数库因其优良的学习效果以及高效的训练速度而获得广泛的关注。仅在 2015 年，在 Kaggle 竞赛中获胜的 29 个算法中，有 17 个使用了 xgboost 库，而作为对比，近年大热的深度神经网络方法，这一数据则是 11 个。在 KDDCup 2015 竞赛中，排名前十的队伍全部使用了 xgboost 库。xgboost 不仅学习效果很好，而且速度也很快，相比梯度提升算法在另一个常用机器学习库 scikit-learn 中的实现，xgboost 的性能经常有十倍以上的提升。鉴于以上原因，此次项目也采用 xgboost 建立模型。

Gradient Boosting 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 $\hat{\mathrm{y}} = \mathrm{F}(x)$ 与真实 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 $\mathrm{F_m}$，在第 m+1 步时，不直接优化 $\mathrm{F_m}$，而是学习一个基本模型 $\mathrm{h}(x)$，使得其拟合残差项 $\mathrm{y} - \mathrm{F}_m$，这样会使 m+1 步的模型预测值 $\mathrm{F}_m = \mathrm{F}_m + h(x)$ 更接近于真实值 y，因而目标变成了如何找到 $h(x) = \mathrm{F}_{m+1} - F_m$，最终就是要找到某函数空间中的一组 $h(x)$ 使得：

$$\mathrm{F}(x) = \sum_{\mathrm{i}=1}^{\mathrm{M}} \gamma_{\mathrm{i}} h_i(x) + c$$

我们使用 xgboost 默认的训练模型 gbtree 决策树来进行训练。其它参数选择如下：

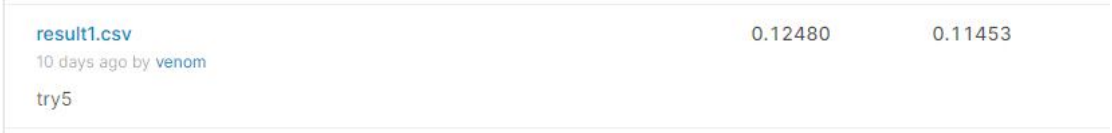
- reg = linear：定义学习任务的目标函数为线性回归。
- eta = 0.03：为了防止过拟合，更新过程中用到的收缩步长。
- booster = gbtree：使用决策树模型
- max_depth = 10: 树的最大深度，默认值为 6
- subsample = 0.9：用于训练模型的子样本占整个样本集合的比例
- colsample_bytree = 0.5: 在建立树时对特征采样的比例
- silent = 1: 打印训练中间过程
- seed=999: 随机数种子

模型的参数参考了其他同学的经验和结果表现。自己也设计了代码来微调 max_depth、subsample、以及 colsample（耗时方法，需要持续运行 36 个小时以上，详见 param_tune_xgboost.ipynb），最终选定以上参数值。

关于训练集和验证集的选取，因为数据具有时间先后顺序，预测总是发生在未来的时间点。因此我们将训练数据的最后 6 周的数据作为验证集，其它作为训练集。另外为了防止高营销额对训练结果产生较打偏差，我们对所有 y 值进行对数 log 运算。最终的训练验证集如下：

```
X_train = df_train[6*7*1115:]
X_valid = df_train[:6*7*1115]
y_train = np.log1p(X_train['Sales'])
y_valid = np.log1p(X_valid['Sales'])
```

模型训练完成后，验证集结果为 0.11602
提交 test 预测结果为：



观察特征值重要性示意图：

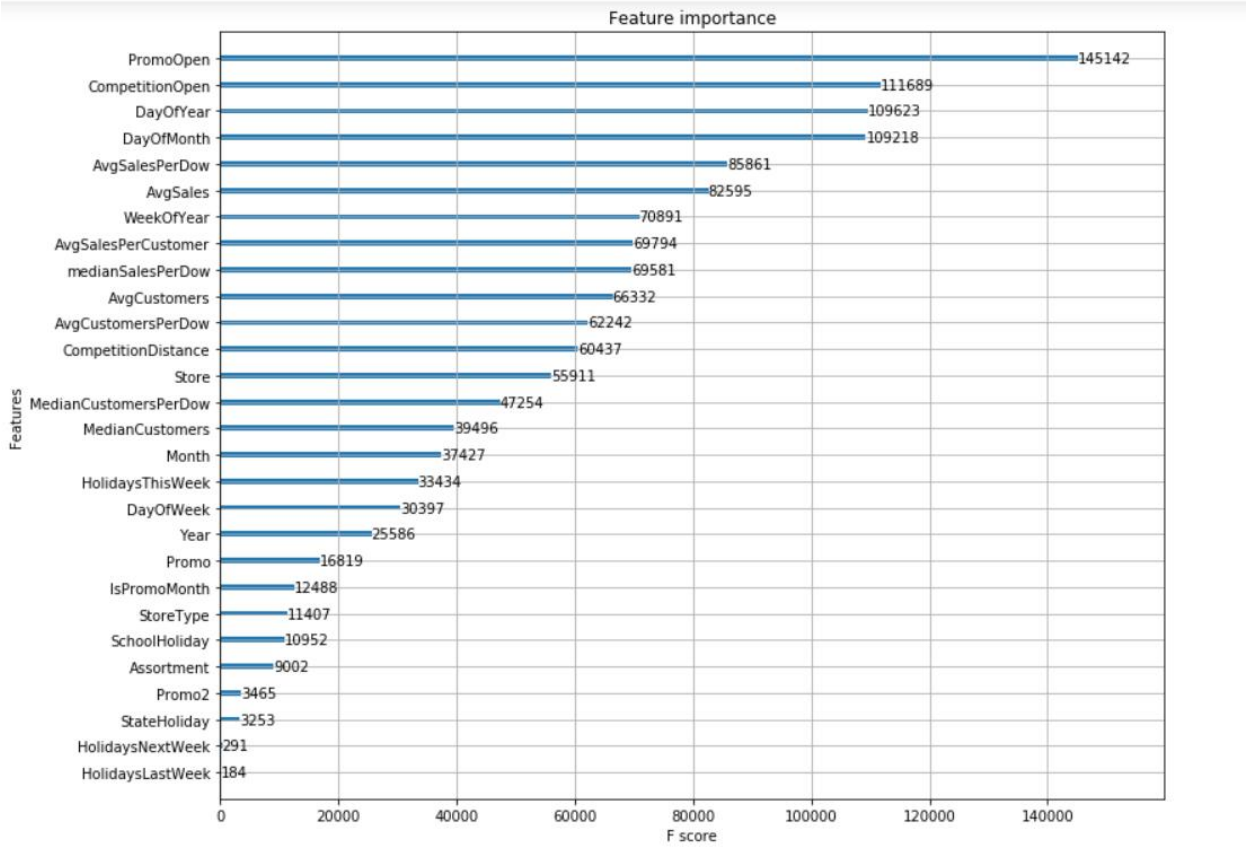


图 13

可以看出最重要的特征值是 PromoOpen， 最不重要的是特征值是 HolidayLastWeek。现在开始对模型进行调参优化和融合优化。

6. 优化及结果分析

6.1. 最优系数校正

假设每个商店都有一个常量系数 factor，将它乘以我们的模型预测结果会得到更好的得分。如下图所示：

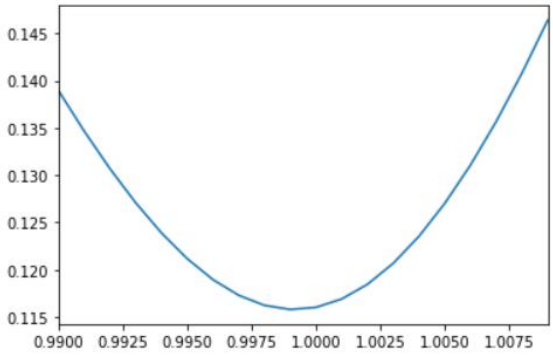


图 14

现在使用穷举法寻找最适合某商店的 factor_i。算法如下：设 factor 的取值范围是[0.995, 1.005]，每 0.001 为一个取值步长。对于模型预测结果的每一个值 y_predict_i，均乘以所属商店的 factor_i，作为最终预测结果。最后选取得分最高的一种 factor 组合。

$$[\text{y}_0, y_1 \ldots y_{n-1}\, y_n][factor_0, factor_1 \ldots factor_{n-1}, factor_n]^T = [\text{y}'_0, y'_1 \ldots y'_{n-1}\, y'_n]$$

详细算法代码如下：

```
col_1 = ['Sales','Prediction']
s_dict = {}

L=range(1115)

for i in L:
    s1 = pd.DataFrame(valid_pred_res[valid_pred_res['Store'] == i+1],columns = col_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)
    s_dict[(i+1)] = Score.idxmin()
```

```
temp = X_valid['Store']
temp.head()
temp_s = pd.Series(temp.values)
w_valid = []
for i, v in temp_s.items():
    a = 1.000
    if v in s_dict.keys():
        a = s_dict[v]

    w_valid.append(a)
```

图 15

算出每个商店最合适的 factor 以后，乘以测试预测值，最终 kaggle 得分数据如下：

Submission and Description	Private Score	Public Score
result3.csv 10 days ago by venom try-7	0.11350	0.10627

可以看出进行该算法修正的预测得分有较大提高。

7. 多模型融合

通过调整模型系数 seed 随机值，来建立十个不同模型。每个模型都通过系数校正。这里需要算每个模型的“贡献值”，即哪个模型在验证集上预测上错误更低，哪个模型的贡献值就越高。具体算法如下所示：

```
model_param = [1.0 - a for a in model_error]
sum = np.sum(model_param)
ratio = [a / sum for a in model_param]

final_result = [0] * len(blend_test_result[0])
for i in range(rounds):
    final_result = final_result + blend_test_result[i] * ratio[i]
```

图 16

最终该方法在 kaggle 得分为：

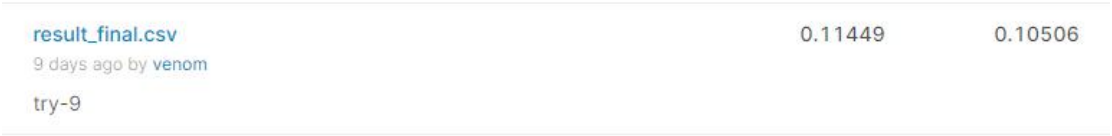


图 17

可以看出融合模型的 private score 降低了，但是 public score 有提高。

8. 结论

本项目先确定评价指标，再对数据进行初步探索和可视化分析。对已经给出的特征值有了较好的认识。然后通过数据预处理和特征工程，挖掘了潜在特征值。接着我们使用 xgbboost 库来建立模型和进行预测，并对结果进行了分析。最后我们进行了系数校正优化和多模型融合优化，得到了不错的预测结果。

【参考资料】

1. Github
https://github.com/udacity/cn-machine-learning/tree/master/Rossmann_Store_Sales
2. Github https://github.com/li-kai/rossman-store-sales
3. 《xgboost 参数解释及调参》
https://blog.csdn.net/iyuanshuo/article/details/80142730
4. 《如何在 Kaggle 首战中进入前 10%》
https://dnc1994.com/2016/04/rank-10-percent-in-first-kaggle-competition/