# 预测 Rossmann 未来的销售额

## 机器学习纳米学位-毕业项目

宋振
2018 年 12 月 5 日

## I. 问题的定义

### 项目概述

本项目将使用 Kaggle 上提供的数据集信息，对 Rossmann 未来的销售额进行预测。

Rossmann 是欧洲的一家药品连锁店，在 7 个欧洲国家拥有 3,000 家药店。Kaggle 比赛 Rossmann Store Sales 要求是利用 Rossmann 药妆店的信息（例如促销，竞争对手，节假日）以及在过去的销售情况，训练模型用于预测 Rossmann 未来的销售额。

本项目采用 Kaggle 中得分较高的 XGBoost 模型，对这里提出的回归问题（预测未来的销售额）进行预测。

训练模型使用的数据集是在 Kaggle 上下载的“train.csv”和“store.csv”，这里面分别包括了与 Rossmann 的销售额以及店铺信息有关的数据；Kaggle 还提供了“test.csv”用于预测销售额，相比“train”数据集少了要预测的“Sales”字段。

### 问题陈述

本项目要解决的问题是：提取 Rossmann 的“train.csv”和“store.csv”数据集里的特征信息，以“Sales”数据作为标签，训练学习器，使学习器可以对“test.csv”数据集里未来一段时间的销售额进行预测。这属于监督学习中的回归问题。

为实现这里的回归问题，需要对数据集进行以下处理：

- 首先需要对数据集进行查看和清理，填补或丢弃数据集里的缺失值和异常值；
- 对已经清理的数据集进行探索——可视化与销售额有关的变量，并且为了提高预测的准确率，需要进行数据集的合并，并从原有特征中提取生成一定数量的新特征，以增加特征数量；
- 定义评价标准，将提取的标签和特征按一定比例随机划分为训练集和测试集，使用训练集训练 XGBoost 学习器，使用测试集评价模型性能，检验模型的鲁棒性。

使用训练好的模型对销售额进行预测，将预测结果提交到 Kaggle，通过得分评估模型表现。

## 评价指标

由于本项目对于销售额的预测属于回归问题，所以使用均方根预测误差（rmspe）作为模型表现的评价指标，其公式如下所示：

$$\text{rmspe} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y - \text{yhat})^2}$$

其中，N 表示样本的数量；y 是实际值，这里是指标签集里真实的销售额；yhat 是预测值，即学习器预测的销售额。

# II. 分析

## 数据的探索

本项目使用了 3 个数据文件，分别是包含销售信息的“train.csv”，包含商店信息的“store.csv”，以及用于预测的“test.csv”，这些数据集中包含的具体字段信息如下：

表 1 数据集中包含的字段信息释义

| 字段名 | 释义 |
|---|---|
| **Store** | 每个商店独有的 Id |
| **Sales** | 销售额 |
| **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，有 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 |

首先查看各数据集中可能存在的缺失值，自上向下分别是“train.csv”“test.csv”和“store.csv”的数据缺失情况：

```
Store            0
DayOfWeek        0
Date             0
Sales            0
Customers        0
Open             0
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

```
Id               0
Store            0
DayOfWeek        0
Date             0
Open            11
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

```
Store                         0
StoreType                     0
Assortment                    0
CompetitionDistance           3
CompetitionOpenSinceMonth   354
CompetitionOpenSinceYear    354
Promo2                        0
Promo2SinceWeek             544
Promo2SinceYear             544
PromoInterval               544
dtype: int64
```

可以看到，train_data 中没有 NaN 值，但是否存在异常值？首先查看对“train.csv”数据集中“Sales”一列的统计性描述[1]：

表 2 数据集“Sales”列的统计性描述

| | |
|---|---|
| count | 1.017209e+06 |
| mean | 5.773819e+03 |
| std | 3.849926e+03 |
| min | 0.000000e+00 |
| 25% | 3.727000e+03 |
| 50% | 5.744000e+03 |
| 75% | 7.856000e+03 |
| Max | 4.155100e+04 |

可以看到，“train.csv”数据集中共有 1017209 条数据。可以看到，销售额的最小值为 0，表明销售数据中存在已经关闭（“Open”== 0）的商店，又或者是数据中存在虽然店铺未关闭（“Open”== 1），但是销售额为 0 的异常值，因此需要对这些数据进行清理，以防止对预测产生干扰。

可以看到，“store.csv”数据集中，'CompetitionDistanceSinceMonth'等系列的 NaN 值较多，缺失原因不明确，这里用 0 填充 NaN；另外，通过进一步探索可知 'Promo2SinceWeek'等字段的数据缺失是因为该店铺没有参加促销活动，所以也可以 0 填充 NaN。

“test.csv”数据集中‘Open’字段的缺失数据全部来源于 622 号商店，这里将‘Open’的 NaN 值全部用‘1’填充以用于测试。

对数据集完成数据清理后，我们将其合并，得到“train_and_store”和“test_and_store”数据集。查看“train_and_store”中以商店类型为分组依据的部分统计数据，其中“SMean”是人均消费额，“All”是每种商店的总数：

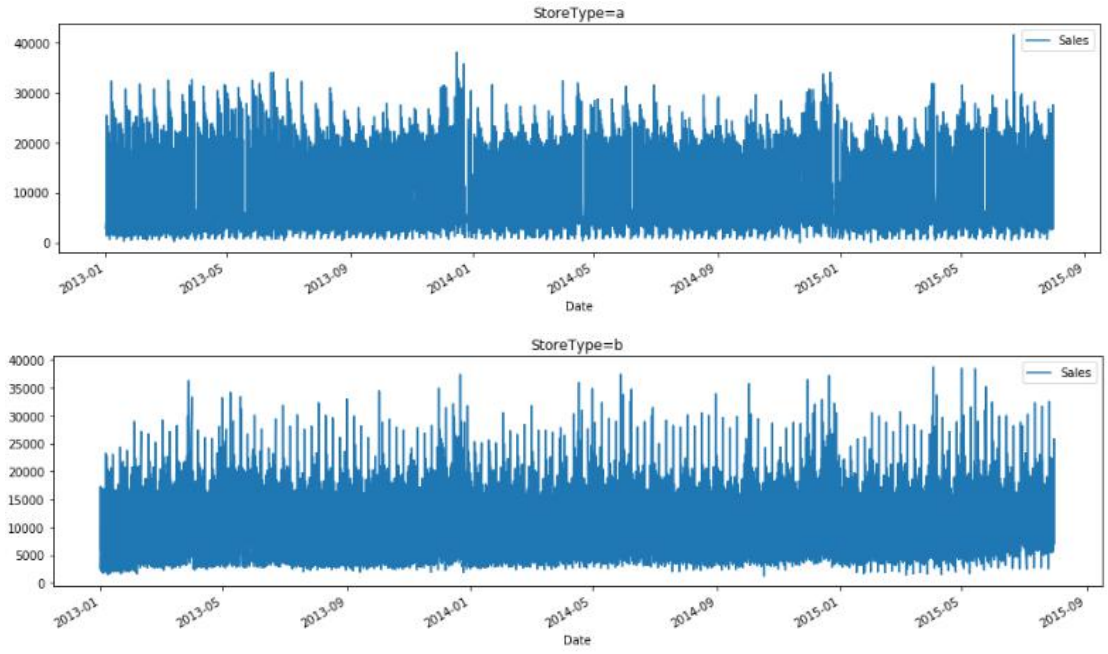
表 3 合并后的数据集信息统计

| | Sales | Customers | SMean | All | PromoP | PromoP2 |
|---|---|---|---|---|---|---|
| StoreType | | | | | | |
| a | 3165334859 | 363541431 | 8.706944 | 457042 | 44.75% | 46.60% |
| b | 159231395 | 31465616 | 5.060489 | 15560 | 38.16% | 28.86% |
| c | 783221426 | 92129705 | 8.501291 | 112968 | 44.91% | 49.76% |
| d | 1765392943 | 156904995 | 11.251350 | 258768 | 44.71% | 56.95% |

从表中可以得知，d 类型商店的人均消费是最高，尽管其总的销售额不如 a 商店多，而 a 类型则是是 Rossmann 数量最多的店铺类型。每种类型都有 50%左右的商店参加了两种促销活动，b 类型商店参加促销活动的商店占总数的比例最低。

## 探索性可视化

实现看一下，不同店铺类型，其过去这段时间销售额的整体情况[3]：



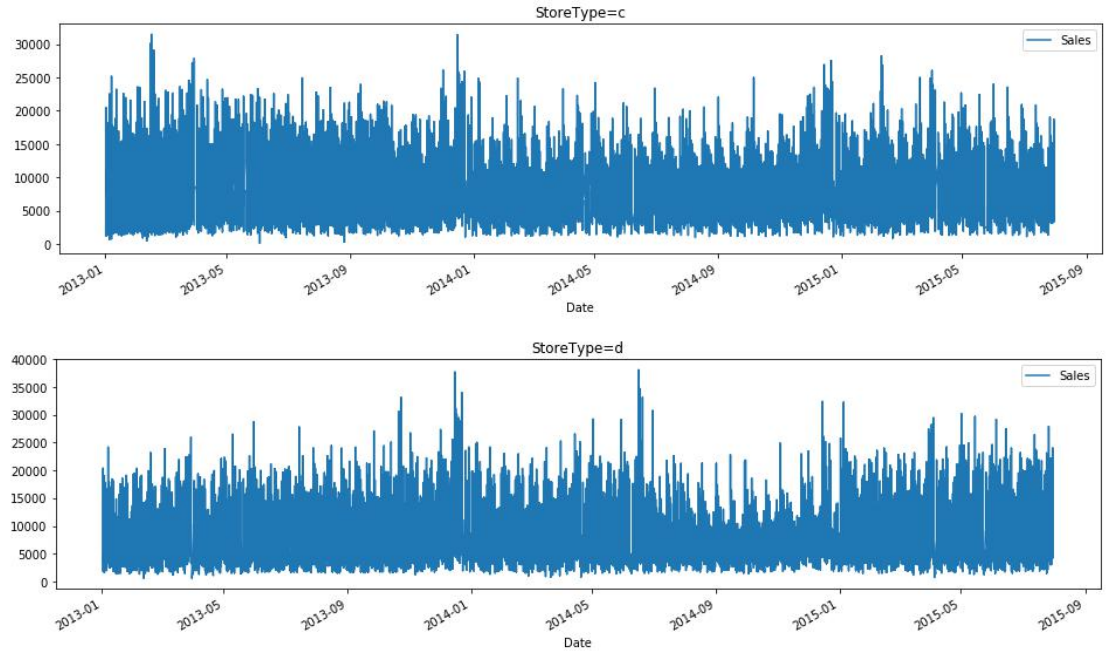
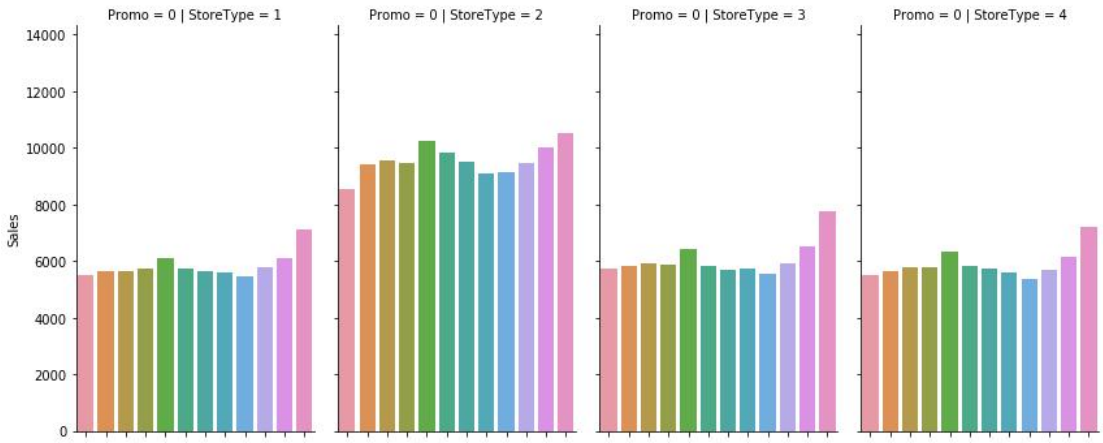



图 1 不同类型店铺在过去时间的销售额信息

可以看到，比较统一的规律是，在每年的年末各店铺的销售额普遍偏高，9 月份前后销售额有所下降。

通过对数据的探索，了解到不同店铺类型的销售情况存在较大差异，所以可以查看不同店铺类型，其销售额随着月份变化的情况，同时考虑该店铺是否参加了促销的情况（“Promo”== 0 or“Promo”== 1）以及是否处于持续促销的状态（（“Promo2”== 0 or“Promo2”== 1））[2]：



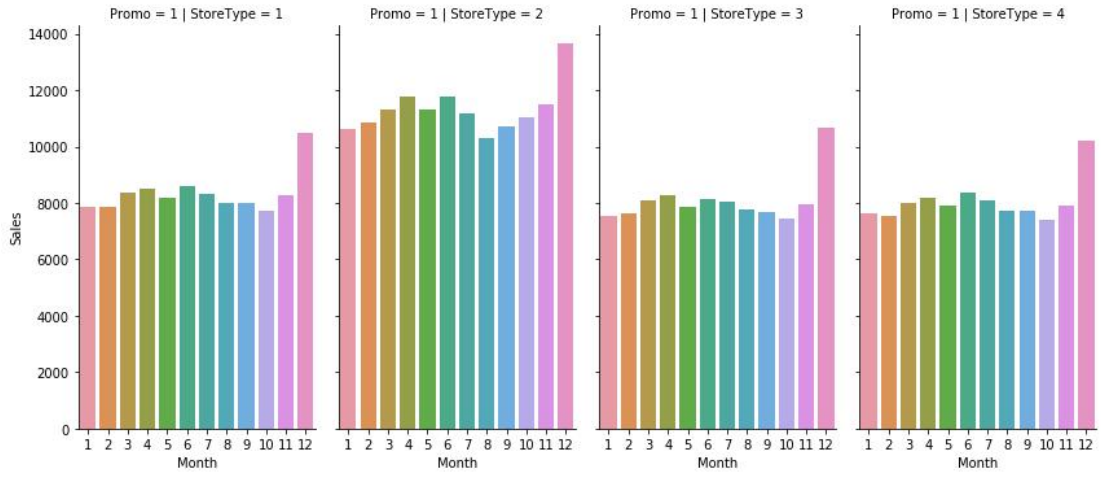



图 2 各类型商店参加/不参加促销每月的平均销售额情况

可以看到，所有店铺在有圣诞节的 12 月份的销售额都要比其余月份高，且 b 类型每个月的平均销售额明显比其他类型的商店高。参加促销的商店（“Promo”== 1）的每月的平均销售额要高于同类型的未参加促销的商店。

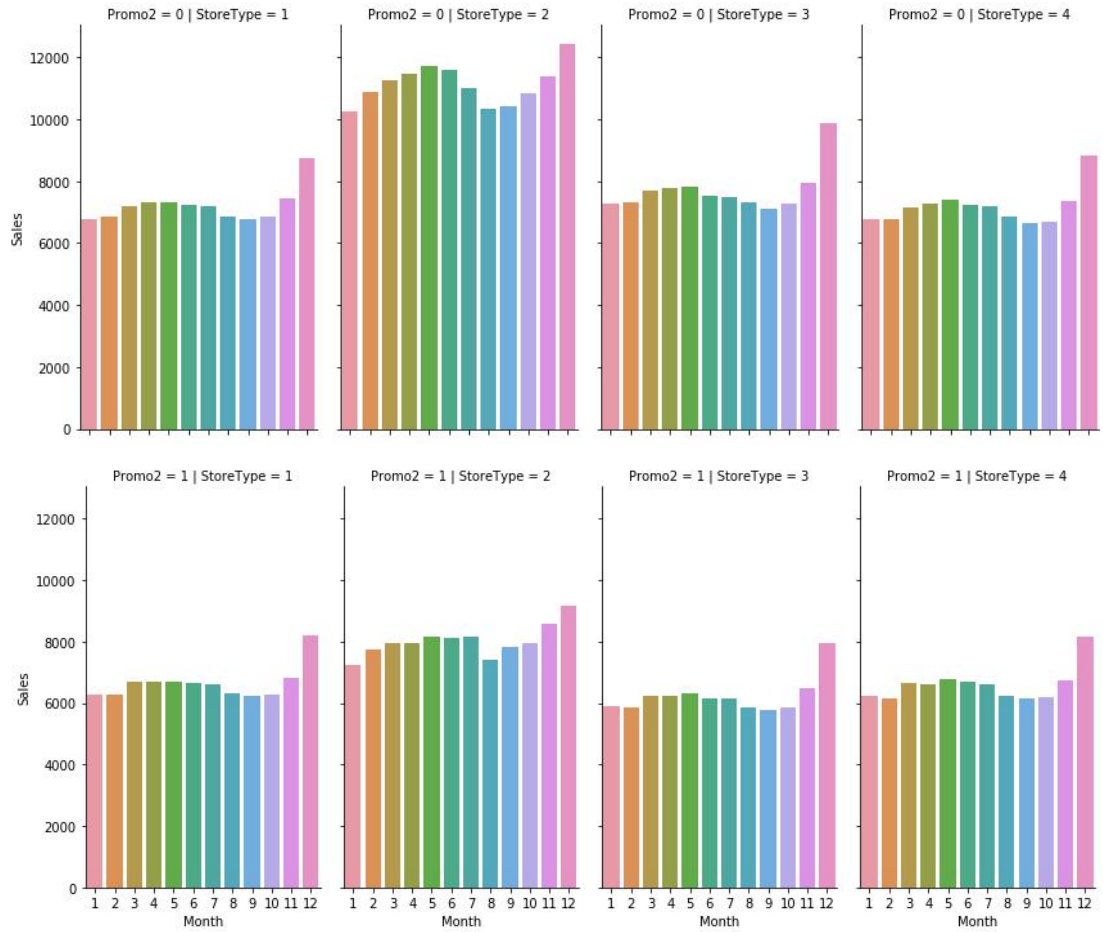



图 3 各类型商店参加/不参加持续促销每月的平均销售额情况

可以看到，不参加持续促销的店铺销售额明显高于同类型参与持续促销的店铺。这可能是因

为参与持续促销的店铺周围有比较有竞争力的同行业店铺，同业竞争导致了店铺的销售额的下降。并且我们注意到，尽管 b 类型店铺每月的平均销售额最高，但人均销售额（见表 4）却最低，这说明该类型的商店可能主要经营高档产品，到该店铺的每位顾客未必一定会购买这里的商品。

再来看一下从周一到周日，各类型店铺的营业情况：

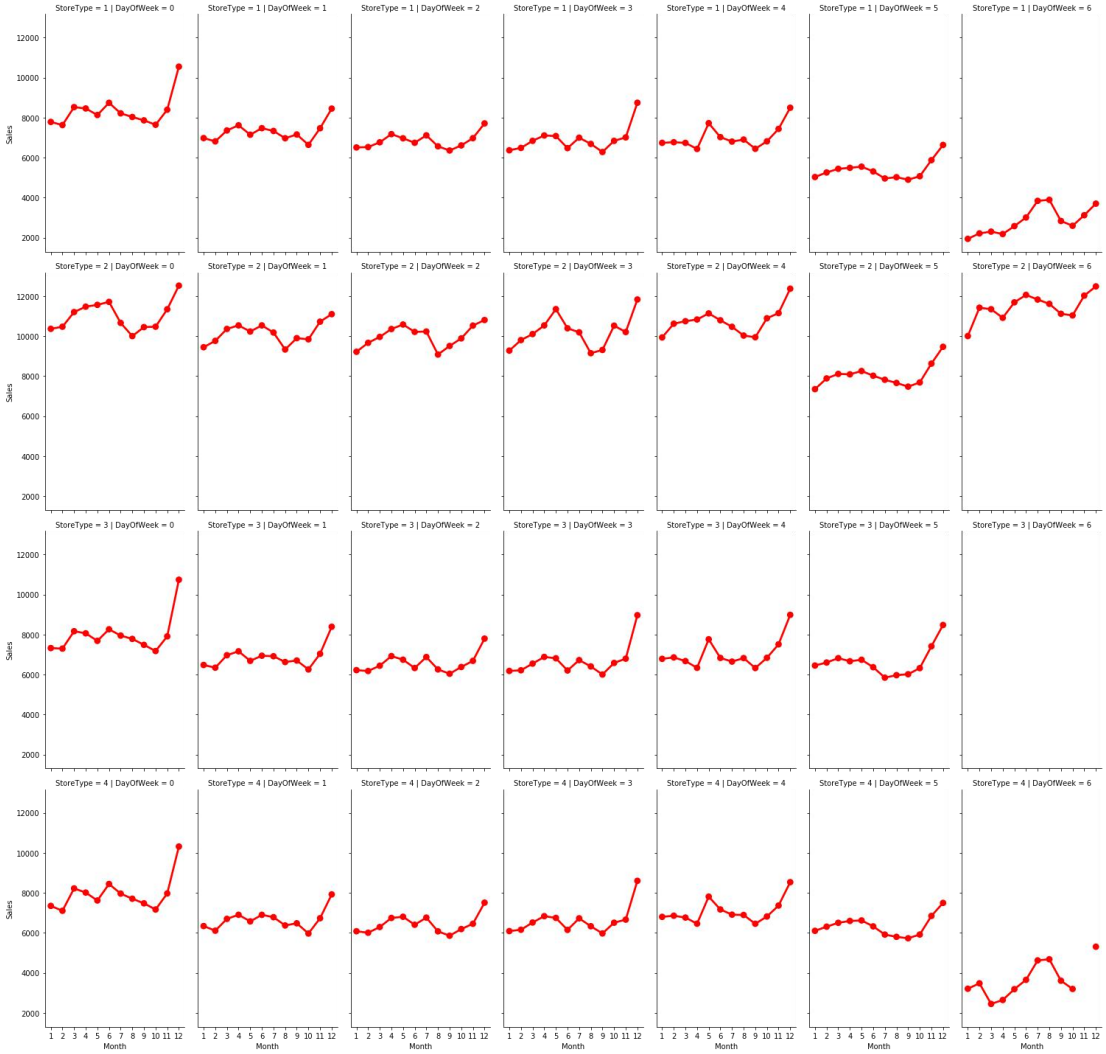



图 4 各个类型店铺周一到周日的营业情况统计

可以看到，很明显的，c 类型的店铺周末是不营业的，所以销售额为 0；a、d 周末的营业额降低较为明显，且 d 在每年的 11 月份的周末是不营业的；b 周末照常营业，且销售额相比其他几天更高。

利用已经实现特征合并的数据集 train_and_store，查看各特征间相关性的热图。从图中可以看到，与销售额相关性较高的特征有诸如“Promo”，“Assortment”，“PromoOpen”等。后期在模型训练结束，输出特征重要时进一步对这一结果进行验证[4]。

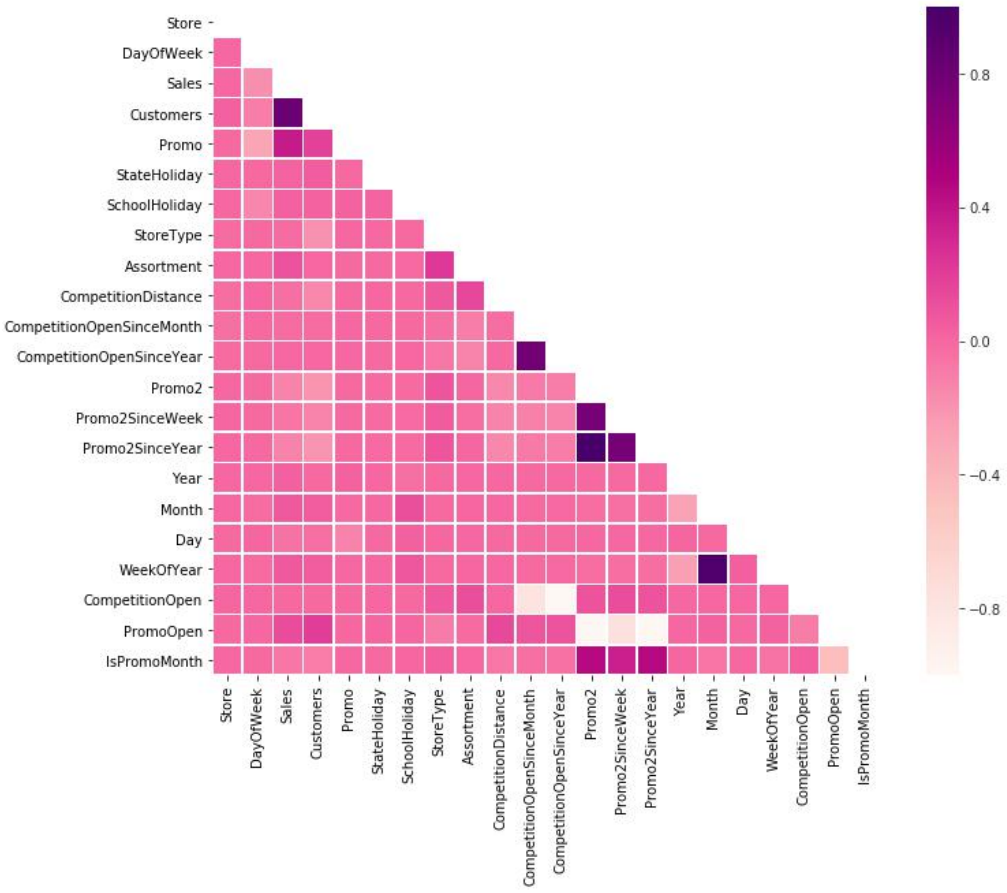



图 5 各特征间相关性热图

## 算法和技术

本项目采用的监督学习算法是 XGBoost，XGBoosting 在传统 Boosting 的基础上，利用 cpu 的多线程，引入正则化项，加入剪枝，控制了模型的复杂度，防止过拟合的出现。XGBoost 模型对应的是一堆 CART 树，该模型的目前函数包含两项：

$$\mathrm{bj}(\theta) = \mathrm{L}(\theta) + \Omega(\theta)$$

θ 是模型的参数。对于该公式，第一项为损失函数，对于分类问题，常用的损失函数是对数损失函数：

$$\mathrm{L}(\theta) = \Sigma\,\mathrm{i}[\mathrm{yiln}(1 + \mathrm{e} - \mathrm{y\hat{}i}) + (1 - \mathrm{yi})\mathrm{ln}(1 + \mathrm{ey\hat{}i})]$$

第二项是正则化惩罚项，用于防止过拟合：

$$\Omega(\theta) = \sum_{\mathrm{i}=1}^{\mathrm{K}} \Omega(\mathrm{f}_k)$$

优化的目标是最小化目标函数，方法是目标函数对权重求偏导，得到一个能够使目标函数最小的权重，把这个权重代回到目标函数中，这个回代结果就是求解后的最小目标函数值。其公式如下：

$$\text{obj}^* = -\frac{1}{2}\sum_{\text{j=1}}^{\text{T}}\frac{{G_j}^2}{H_j+\lambda}+\gamma T$$

$$G_j = \sum_{i\in I_j} \partial_{y\hat{}(t-1)}\, \text{l(yi,y\^{} (t} - 1))$$

$$H_j = \sum_{i\in I_j} \partial^2{}_{y\hat{}(t-1)}\, \text{l(yi,y\^{} (t} - 1))$$

本项目希望可以利用其优秀的预测性能和运行效率获得较为优异的结果，并且， Kaggle 比赛的第一名就是采用的该算法对结果进行了精准的预测。对于该算法，主要包含以下三种参数类型：

- 通用参数，如 booster 等；
- 提升参数，如 eta（学习率），max_depth（树的最大深度），colsample_bytree（列采样率）等；
- 学习任务参数，如 objective（如本项目中选用“reg:linear”）等；

在训练数据的准备过程中，需要处理缺失值和异常值，将处理后的数据集合并形成新的含有较多特征的数据集，并有必要根据旧的特征，生成部分新特征以提升模型训练效果。

之后，从合并后的新数据集中提取标签和特征分别形成标签集和特征集，使用 sklearn 中的 train_test_split 将特征集和标签集随机地划分为训练集和测试集，使用训练集训练 XGBoost 模型，用测试集验证训练效果。

## 基准模型

本项目以模型在测试集上的 RMSPE 得分（Public Score）为评价基准，基准分数为 0.12，这也是进入 Kaggle 前 10%所需的分数。分数越低模型表现越优异。

# III. 方法

## 数据预处理

将 train.csv 原始数据中已经关闭的店铺，以及虽然未关店，但销售额为 0 的异常值进行了处理，将去除了这些值之后的数据存储为新的数据集 new_train_data 中；

对于 store.csv 原始数据中的“NaN”值，进行了填 0 的处理，并将处理后的数据放在新的数据集 new_store_data 中；将 test.csv 原始数据中“Open”字段的缺失值进行了填 1 处理，将处理后的数据放在新的数据集 new_test_data 中；之后，进行了数据集的合并处理；

最后，对合并后的“train_and_store”和“test_and_store”数据集进行了新特征的提取，从原有时间序列型特征中生成了“Year”、“Day”等新的数字类型特征，并对非数字特征如“Assortment”等进行了编码；

## 执行过程

首先，需要定义 rmspe 的计算方法：

In [107]:
```
#定义rmspe

def ToWeight(y):
    w = np.zeros(y.shape, dtype=float)
    ind = y != 0
    w[ind] = 1./(y[ind]**2)
    return w


def rmspe(yhat, y):
    w = ToWeight(y)
    rmspe = np.sqrt(np.mean( w * (y - yhat)**2 ))
    return rmspe


def rmspe_xg(yhat, y):
    # y = y.values
    y = y.get_label()
    y = np.exp(y) - 1
    yhat = np.exp(yhat) - 1
    w = ToWeight(y)
    rmspe = np.sqrt(np.mean(w * (y - yhat)**2))
    return "rmspe", rmspe

#以上代码引用自：https://www.kaggle.com/paso84/xgboost-in-python-with-rmspe
```

之后，定义 “train_and_store”数据集的标签和特征，再利用 train_test_split 数据拆分为训练集和测试集，测试集所占的比例为 0.2：

In [108]:
```
# 定义标签和特征

features = train_and_store.drop(['Date','Customers','Open','PromoInterval','monthStr','Sales'],axis=1)
labels = np.log1p(train_and_store.Sales)

# 将train_and_store拆分为训练集和验证集
import random
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
```

之后设定模型参数。对于首次训练模型选取的 booster 参数：

- eta：与 learning rate 参数类似，通过减少每一步的权重，提高模型的鲁棒性，这里选取 0.3；
- max_depth：用于避免过拟合的。max_depth 越大，模型会学到更具体更局部的样本，暂设定为 8；
- subsample：控制对于每棵树随机采样的比例。减小该参数可以避免过拟合。但是值设置得过小的话可能会导致欠拟合，这里设置为 1；
- colsample_bytree：用于控制每棵随机采样的列数的占比，这里选取 0.7；

最后进行模型的训练，在训练模型之前，需要将加载的数据存储在对象 DMatrix 中，以优化存储和运算速度：

```
In [109]: #设定XGboost的参数

params = {'objective': 'reg:linear',
          'booster': 'gbtree',
          'eta': 0.3,
          'max_depth': 8,
          'subsample': 0.7,
          'colsample_bytree': 0.7,
          'learning_rates': 0.1,
          'silent': 1
          }

num_trees = 600

dtrain = xgb.DMatrix(X_train, y_train)
dvalid = xgb.DMatrix(X_test, y_test)
evallist = [(dvalid, 'eval'), (dtrain, 'train')]

#训练模型
gbm = xgb.train(params, dtrain, num_trees, evals=evallist, early_stopping_rounds=50, feval=rmspe_xg, verbose_eval=True)
```

输出结果如下：

```
[595]   eval-rmse:0.093196      train-rmse:0.078411      eval-rmse:0.103317      train-rmse:0.085257
[596]   eval-rmse:0.093209      train-rmse:0.078376      eval-rmse:0.103384      train-rmse:0.08521
[597]   eval-rmse:0.093204      train-rmse:0.078346      eval-rmse:0.103379      train-rmse:0.085163
[598]   eval-rmse:0.093205      train-rmse:0.078333      eval-rmse:0.103382      train-rmse:0.085146
[599]   eval-rmse:0.093202      train-rmse:0.078296      eval-rmse:0.103392      train-rmse:0.085095
```

在训练 599 轮后得到了最佳本地 rmspe score，该模型在测试集上也得到了较好的预测结果：

```
In [110]: #检验模型的预测效果

yhat = gbm.predict(xgb.DMatrix(X_test))
error = rmspe(np.expm1(y_test), np.expm1(yhat))

print('The rmspe is {:.6f}.'.format(error))
```

```
The rmspe is 0.093758.
```

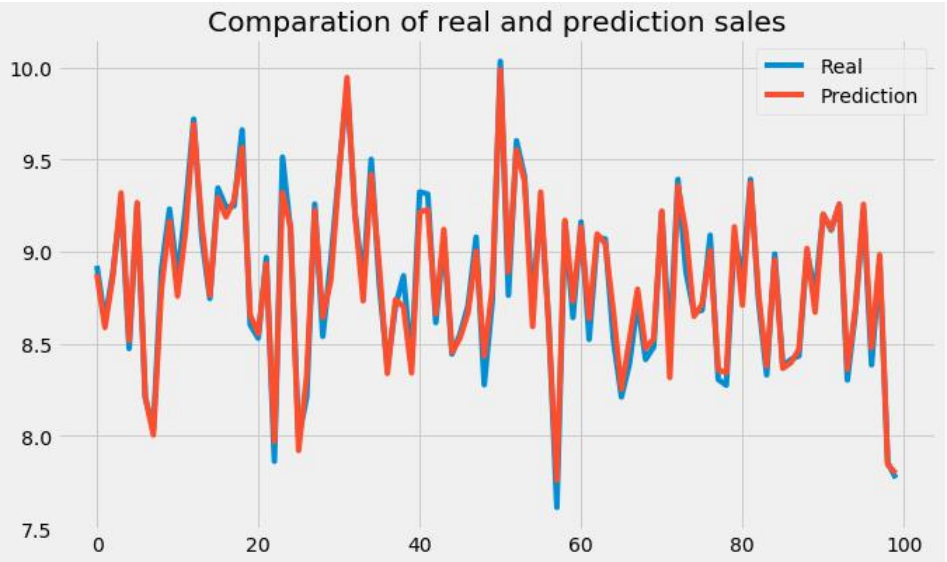
## 完善

对该模型性能影响较大的超参数主要是 eta，max_depth、colsample_bytree 以及 num_trees，本项目对上述参数进行了优化:eta 减小为 0.05，max_depth 增加至 10，colsample_bytree 增加为 0.8，num_trees 增加为 1000。经过 999 轮训练后得到本地最佳得分，在测试集上的得分为 0.090415。

# IV. 结果

## 模型的评价与验证

通过对模型的参数进行网格搜索优化，得到了最终的模型。在 X_test 中随机抽取 100 个样本，通过对比模型对这些样本的预测值和真实值查看模型表现：




可以看到，预测值和销售值走线吻合的比较好，表明预测较为准确。同时，该模型在 kaggle 上的最终得分为 Private Score：0.12687，Public Score：0.11543，其性能较好，与期待的结果一致。下图是参数优化前后的预测结果在 Kaggle 上的得分[5]：

All Successful Selected

| Submission and Description | Private Score | Public Score | Use for Final Score |
|---|---|---|---|
| Rossmann_Sales_submission.csv<br>3 minutes ago by Zhen Song<br>add submission details | 0.12687 | 0.11543 | ☐ |
| Rossmann_Sales_submission.csv<br>a day ago by Zhen Song<br>Rossmann Store Sales | 0.13441 | 0.12647 | ☐ |

进一步的，为验证模型的鲁棒性，我们用一条样本数据，用训练好的模型循环验证多次，看是否得到相同的测试误差：

```
#对X_test进行多次预测验证模型健壮性

import warnings
warnings.filterwarnings('ignore')

#多次运行测试结果
print ("Error Results\n------")
for i in range(10):
    yhat = gbm.predict(xgb.DMatrix(X_test))
    error = rmspe(np.expm1(y_test), np.expm1(yhat))
    print ("The No.{} time error result for the X_test is: {} ".format(i+1, error))
```

```
Error Results
------
The No.1 time error result for the X_test is: 0.09041482589671332
The No.2 time error result for the X_test is: 0.09041482589671332
The No.3 time error result for the X_test is: 0.09041482589671332
The No.4 time error result for the X_test is: 0.09041482589671332
The No.5 time error result for the X_test is: 0.09041482589671332
The No.6 time error result for the X_test is: 0.09041482589671332
The No.7 time error result for the X_test is: 0.09041482589671332
The No.8 time error result for the X_test is: 0.09041482589671332
The No.9 time error result for the X_test is: 0.09041482589671332
The No.10 time error result for the X_test is: 0.09041482589671332
```

可以看到，连续预测 10 次的结果相同，表明模型对于该问题足够稳健可靠，也说明模型得出的预测结果是可信的。

## 合理性分析

该模型对测试集的预测，在 Kaggle 上的最终得分为 0.11543，高于基准得分 0.12，表明该模型较好的解决了对商店未来一段时间销售额的预测问题。

# V. 项目结论

## 结果可视化

关于特征重要性的可视化结果如下：

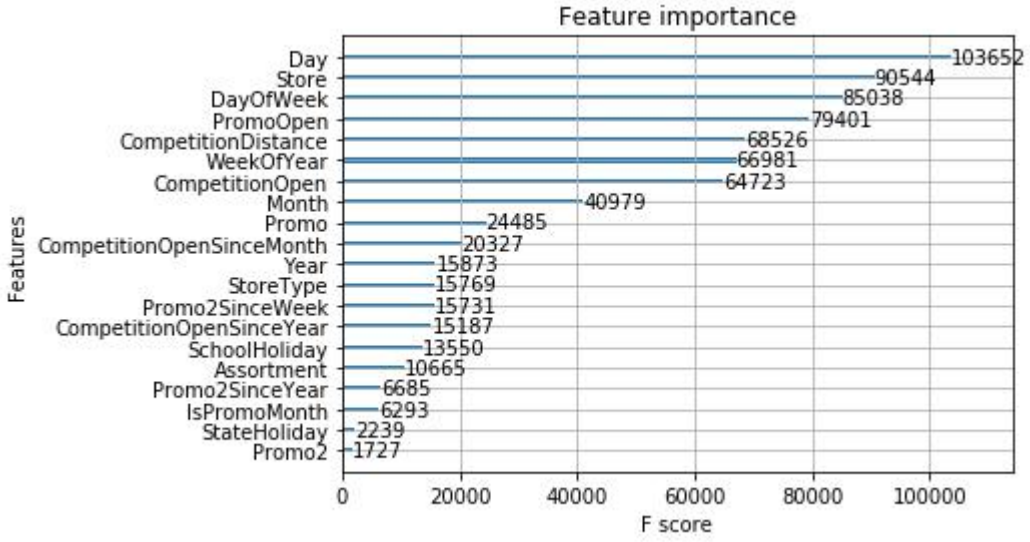



图 6 特征重要性可视化

## 对项目的思考

本项目比较顺利的解决了对 Rossmann 未来一段时间销售额预测的问题，这个项目的最有意思的地方是自己尝试了整个数据分析的流程，从提出问题，到数据的收集和清理，在到数据探索，最后根据数据建模和预测。

主要困难主要出现在数据探索阶段，由于对 Python 语言熟悉度不够，在对数据进行格式处理（如将“Date”分解成新的特征）和进行数据可视化时花费了比较多的时间完成，选取的可视化的方式可能也不是最好的表现方式。对 XGBoost 的运用不够熟练，对于模型参数的熟悉程度不够。

## 需要作出的改进

在数据处理阶段，对于 NaN 值的处理是直接填充 0，没有尝试填充中位数的方案，可能采用后一种方案会得到更好的预测效果；

模型的选取上使用了 XGBboost 模型模型，该模型的使用较为简单，但参数优化较为困难。本项目在 Kaggle 上的得分一般，所以在参数优化上面仍然有待提高；

可以尝试采用新的模型，如决策数，AdaBoost 等，评价不同模型的表现。

## 参考资料


[1] Kaggle 竞赛项目——Rossmann 销售预测 Top1%
[2] 20 Newsgroup Document Classification Report
[3] Stackoverflow
[4] Time Series Analysis and Forecasts with Prophet
[5]维基百科