机器学习纳米学位

毕业项目

Senga 优达学城

2020年3月11日

I. 问题的定义

项目概述

> 需要解决的问题涉及哪个领域？做这个项目的出发点？有哪些相关的数据集或输入数据？

本项目所解决的问题是**预测 Rossmann 未来的销售额**，该问题设计数据挖掘与金融领域。选择这个项目的出发点是因为对数据挖掘与金融感兴趣。使用了 kaggle 上所提供的 Rossmann 过去的销售记录作为输入数据。Rossmann在7个欧洲国家经营着3,000多家药店。目前，Rossmann商店经理的任务是预先提前六周预测他们的日常销售。商店销售受许多因素的影响。成千上万的个体经理根据他们独特的情况预测销售情况，结果的准确性可能会有很大差异。本次项目要求预测德国各地的1,115家商店6周中每周销售额。

> 问题的背景信息能够让完全没接触过这个问题的人充分了解这个问题吗？

问题的背景信息在 kaggle 上是比较具体的，对于输入数据的鱼问题的解释可以让完全没接触过这个问题的人充分了解这个问题

问题陈述

Rossmann Sales 是一个 regression 的 supervised learning，需要根据过去的销售数据提炼出 feature 然后进行预测。

预计使用 lightgbm 模型，每一步骤将会在分析部分具体展开：

- 首先要进行数据探索，探索 feature 的关系

- 数据预处理
- 对该模型进行调参优化
- 进行测试验证优化

评价指标

kaggle 给出的评价指标为 Root Mean Square Percentage Error (RMSPE),

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

- 其中 y_i 一家 store 一天的销售额
- $\hat{y}_i$ 代表对应的 prediction
- 任何 0 sales 在评分中都会被忽略

II. 分析

数据的探索

数据解释

- **Id** - 一个代表着 (Store, Date) 的 id
- **Store** - 一个商店的唯一 id
- **Sales** - 给定一天内的营业额
- **Customers** - 给定一天内的客户数量
- **Open** - 商店是否开门 0 = closed, 1 = open
- **StateHoliday** - 代表着国家假日，一般来说绝大多数商店都会在国家假日关门。注意学校在 public holiday 和周末都会关门。a = public holiday, b = Easter holiday, c = Christmas, 0 = None
- **SchoolHoliday** - 代表 (Store, Date) 是否有被学校放假所影响
- **StoreType** - 四种不同的商店类型: a, b, c, d
- **Assortment** - 商店的 assortment 等级: a = basic, b = extra, c = extended
- **CompetitionDistance** - 最近的竞争对手的距离，以米为单位
- **CompetitionOpenSince(Month/Year)** - 竞争对手开张的年/月
- **Promo** - 代表当日商店是否有打折
- **Promo2** - 持续性的打折 0 = store is not participating, 1 = store is participating
- **Promo2Since(Year/Week)** - 参与打折的年/周
- **PromoInterval** - 打折间隔 E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store
-

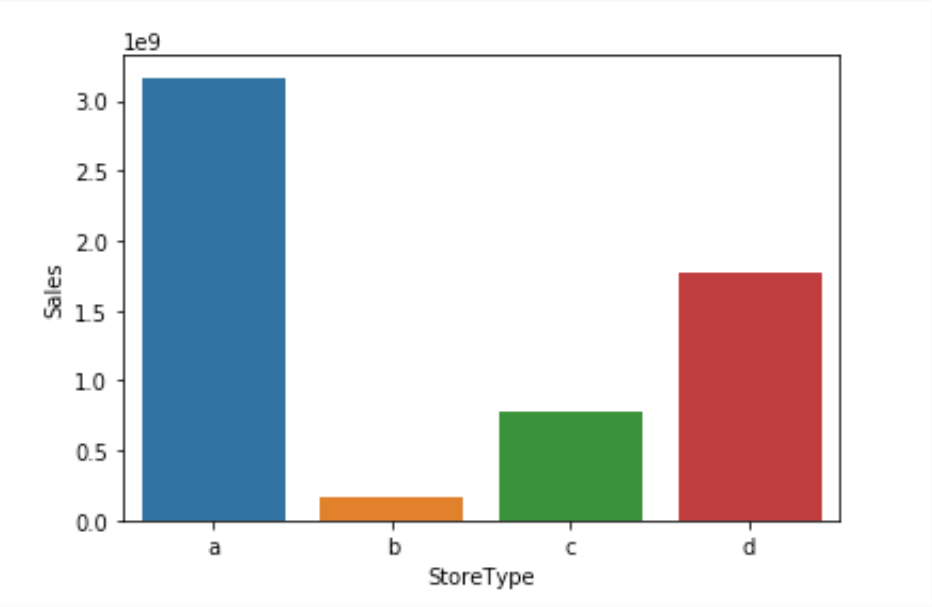
数据异常值

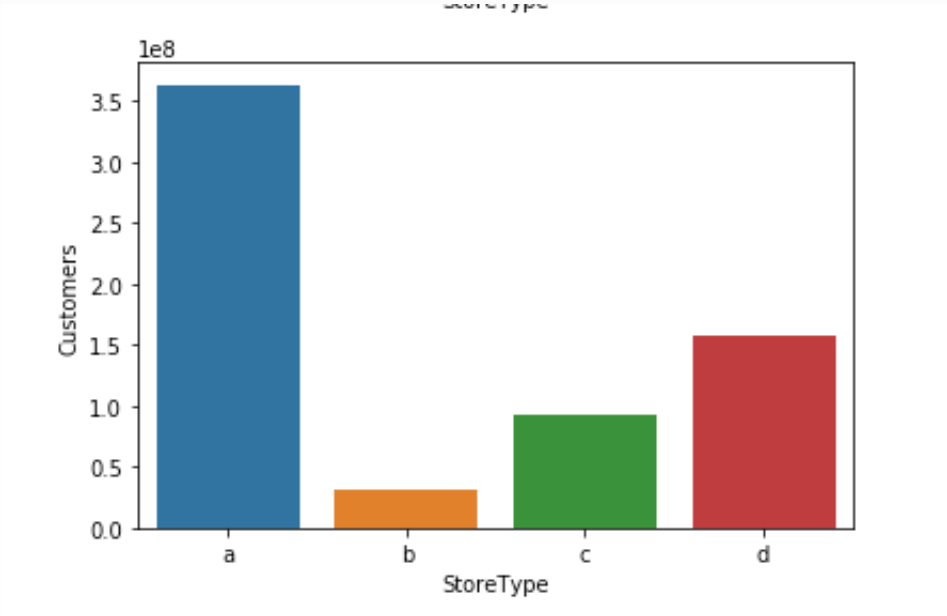
- 注意到在数据集中存在部分商店未开门却有营业额的情况，为了模型训练的准确性选择了商店开门且有营业额的数据作为训练集

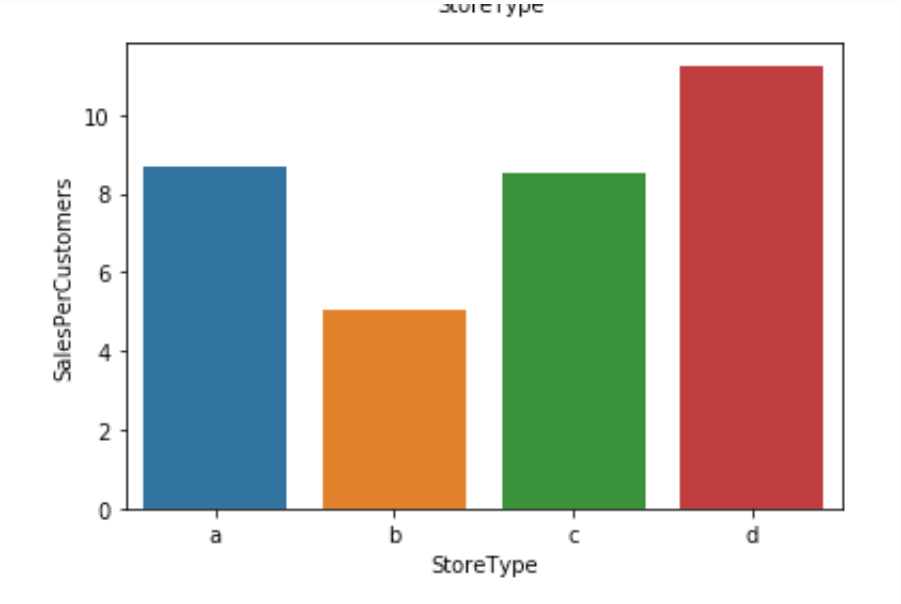
探索性可视化

商店类型与营业额之间的关系

增加了一个参数 `SalesPerCustomer`， 进行可视化后得出 a 的总营业额和客户数量最高，d的平均顾客画的钱最高，b类的商店在各个表现上均较差。



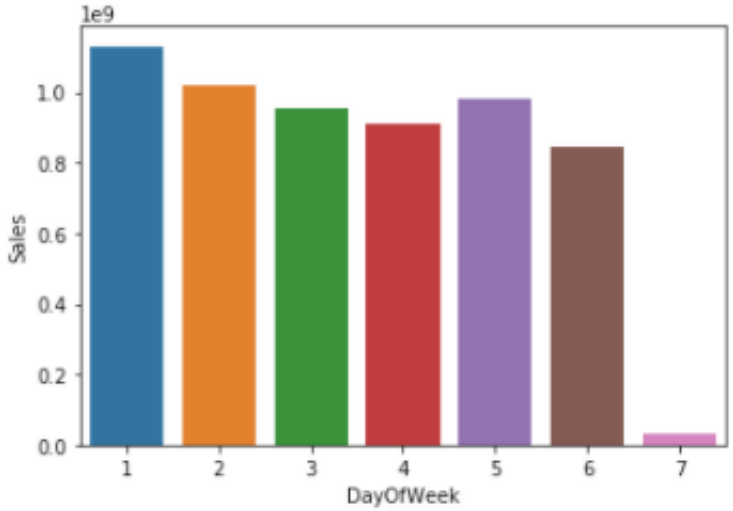




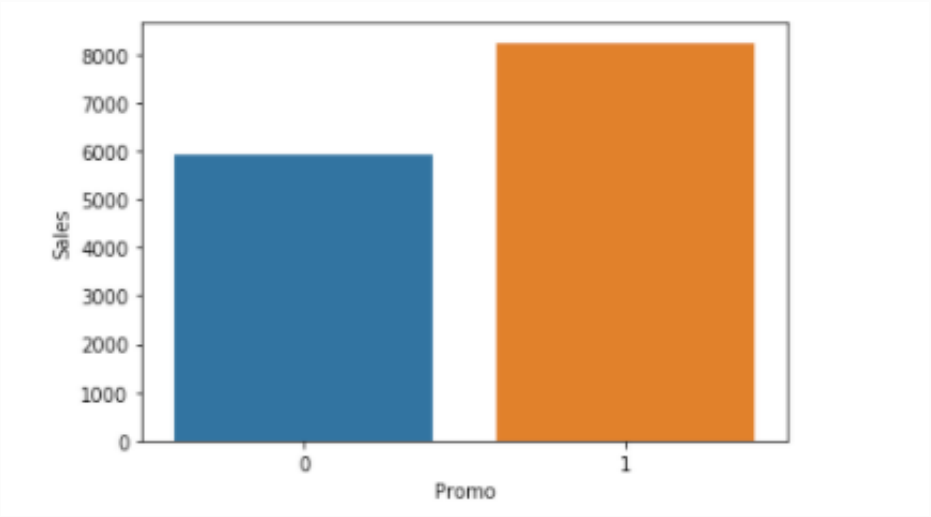
分析工作日与销售额之间的关系

发现周日销售额最少，周一销售额最多。可知周几对于销售额来说是有参考价值的。

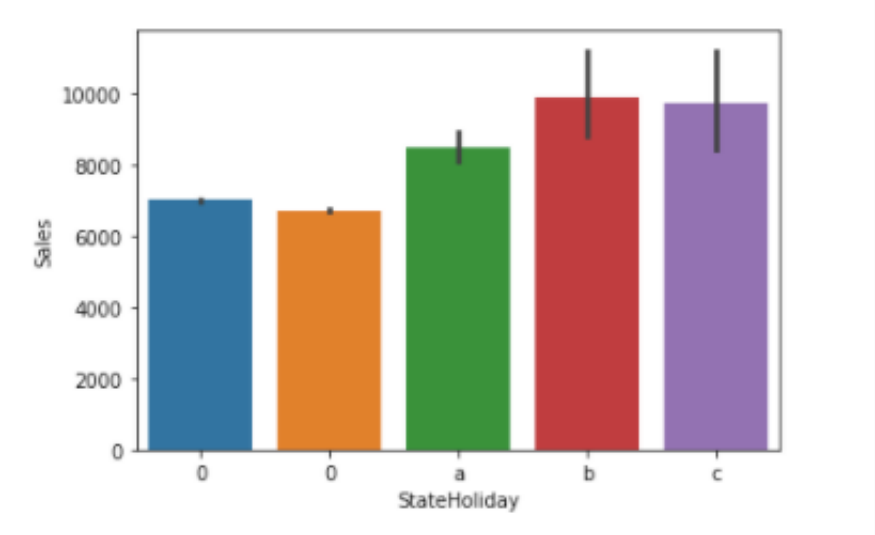
```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1221e4910>
```



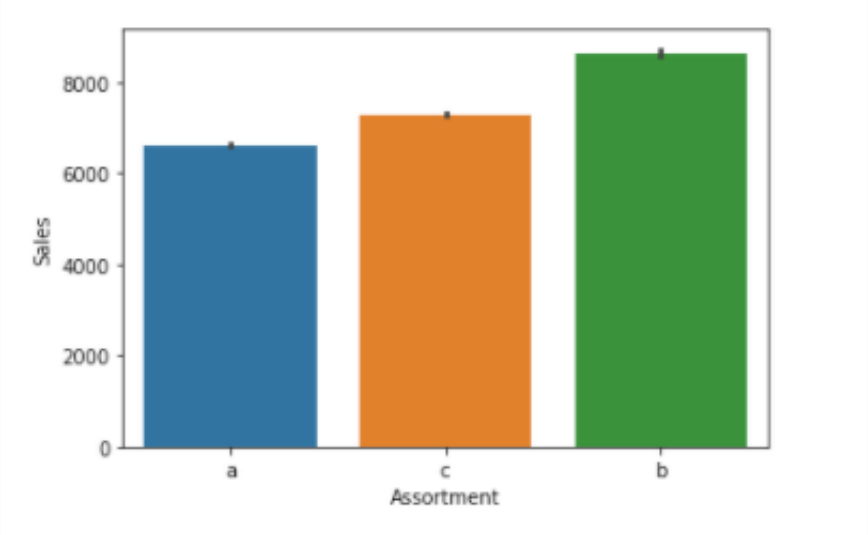
打折分析



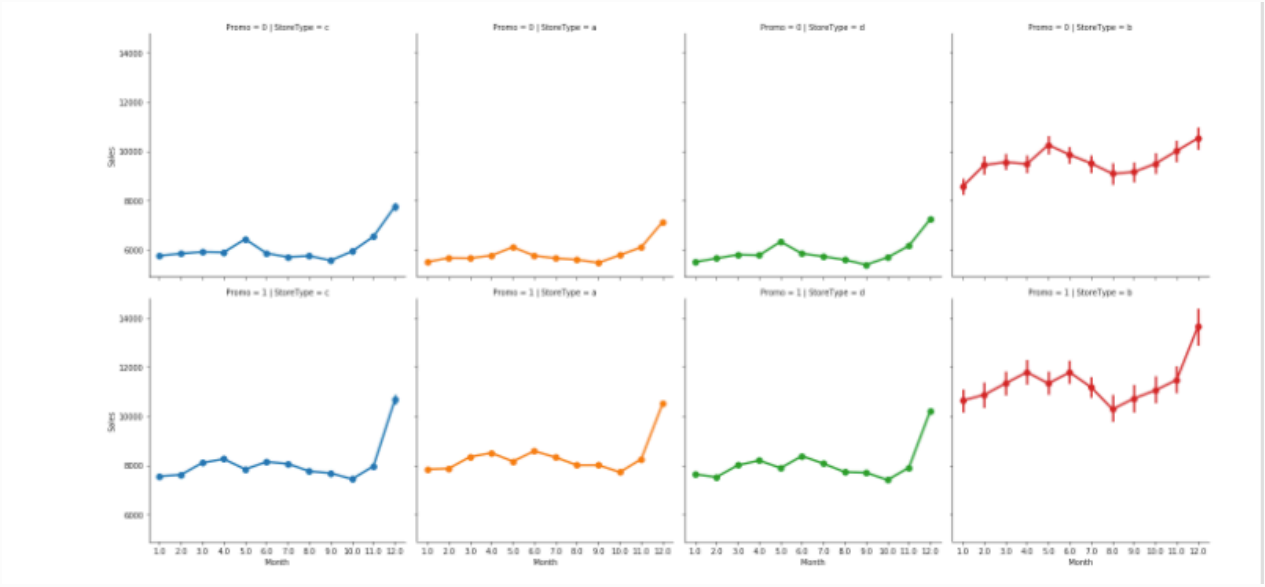
节假日分析



商店 AssortMent 分析

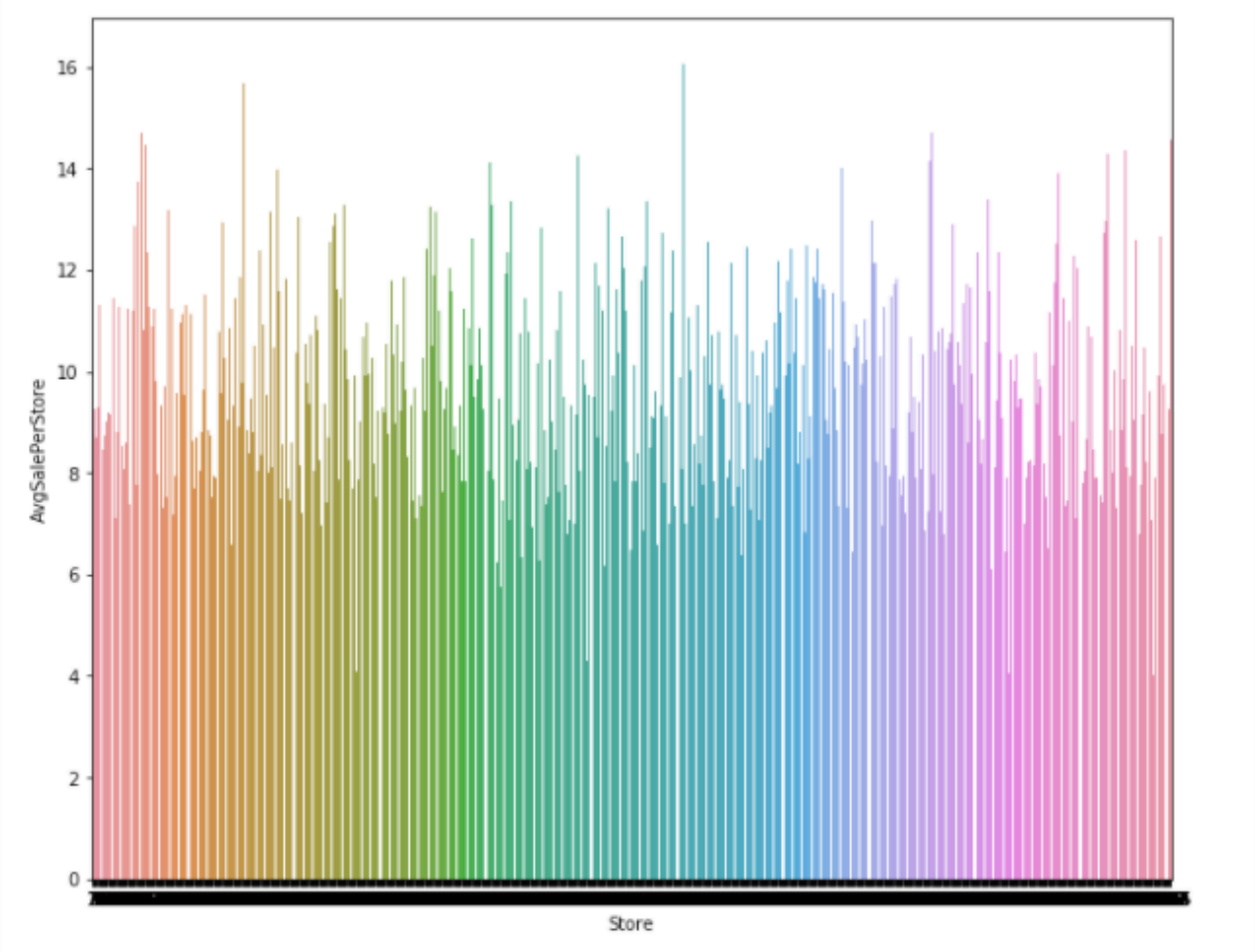


每月销量分析

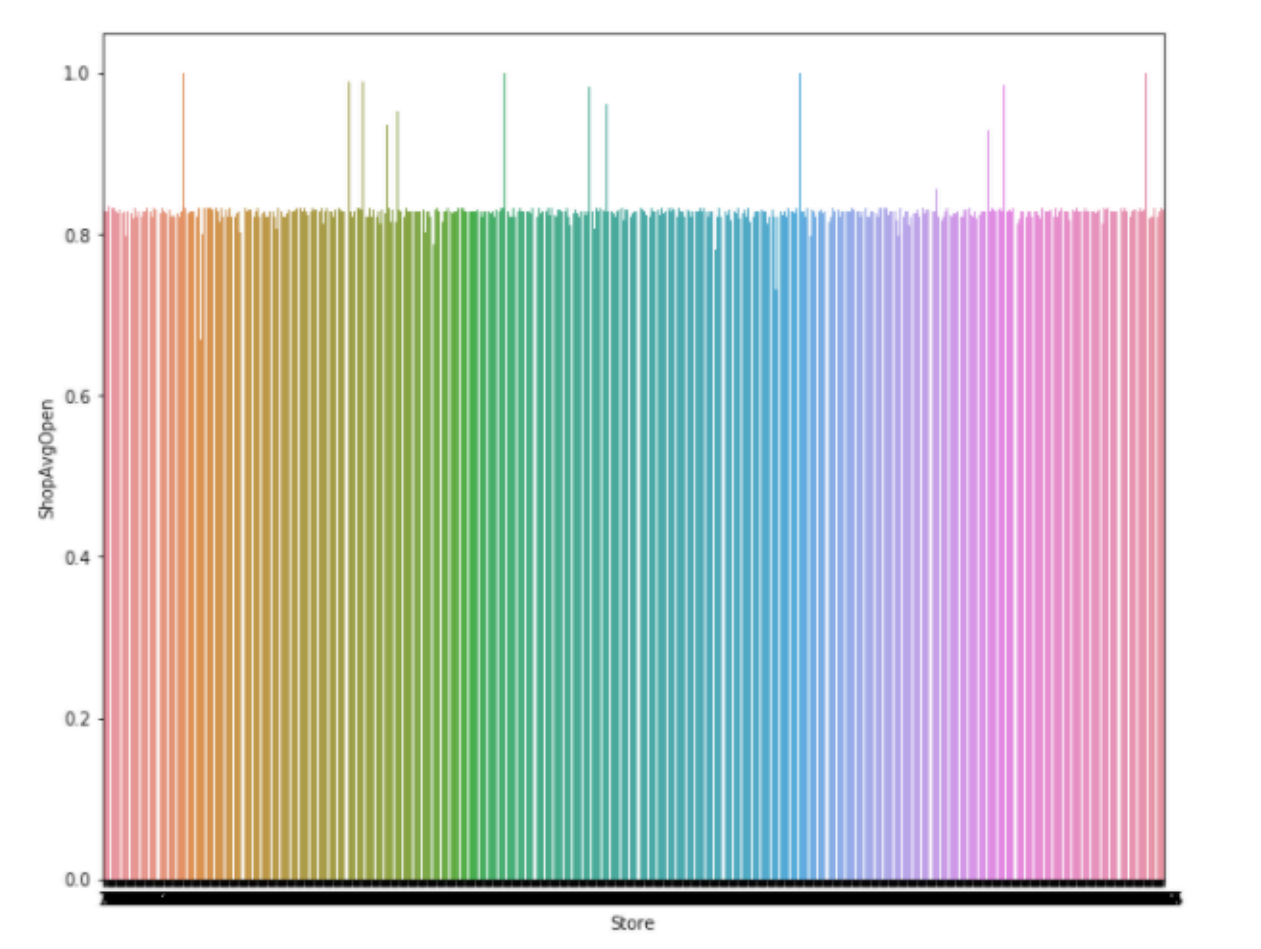


商店销售额分析

一千多个商店 id 每家商店的平均销售额有所不同。



商店平均开门几率分析



算法和技术

实现模型的 training error 足够小，且算法的自由度不能过大。真实数据存在很多的噪声，需要在数据处理部分进行清除。

基准模型

模型的 trainning error 要足够小，需要处理实际数据中的噪声的现象。采用的是 GBDT(Gradient boosting) 模型：

Algorithm 1. The Algorithm Framework of GBDT

1. $F^*(x) = argminE_y(L(y, F(x))|x)$
2. $F_0(x) = f_0(x)$
3. $for \quad m = 1 \quad to \quad \mathrm{M}:$
4. $\quad \mathrm{g}_m(x) = -\frac{\partial E_y[L(y,F(x))|x]}{\partial F(x)}|F(x) = F_{m-1}(x) \quad //descent\ direction$
5. $\quad \rho_m = argminE_y[L(y, F_{m-1}(x) + \rho g_m(x)|x] \quad //step\ size$
6. $\quad f_m(x) = \rho_m g_m(x)$
7. $\quad F_m(x) = F_{m-1}(x) + \rho_m g_m(x)$
8. $end\ for$
9. $F^*(x) \approx F_M(x) = f_0(x) + \sum_{m=1}^{M} \rho_m g_m(x)$

具体的采用 **XGBoost 算法**：

作为GBDT的高效实现，XGBoost是一个上限特别高的算法，因此在算法竞赛中比较受欢迎。简单来说，对比原算法GBDT，XGBoost主要从下面三个方面做了优化：

1. 算法本身的优化：
    - 在算法的弱学习器模型选择上，对比GBDT只支持决策树，还可以直接很多其他的弱学习器。
    - 在算法的损失函数上，除了本身的损失，还加上了正则化部分。
    - 在算法的优化方式上，GBDT的损失函数只对误差部分做负梯度（一阶泰勒）展开，而XGBoost损失函数对误差部分做二阶泰勒展开，更加准确。
2. 算法运行效率的优化：
    - 对每个弱学习器，比如决策树建立的过程做并行选择，找到合适的子树分裂特征和特征值。在并行选择之前，先对所有的特征的值进行排序分组，方便前面说的并行选择。
    - 对分组的特征，选择合适的分组大小，使用CPU缓存进行读取加速。将各个分组保存到多个硬盘以提高IO速度。
3. 算法健壮性的优化：
    - 对于缺失值的特征，通过枚举所有缺失值在当前节点是进入左子树还是右子树来决定缺失值的处理方式。
    - 算法本身加入了L1和L2正则化项，可以防止过拟合，泛化能力更强。

还采用了 linear regression 和 decisiontree 作为基础模型进行比较，同时尝试了 LGB 算法，但是效果不尽如人意。选择 XGboost 的原因也是因为在 google 的 colab 上可以直接使用 GPU 进行训练。

对于数据进行了商店每个星期日的销售额，商店类型对销售额的影响，连续月的销售额变化，节假日的销售额变化，对于每个商店种类的用户平均购买力进行分析。同时加入了以商店 id 为基准的信息分析。

III. 方法

数据预处理

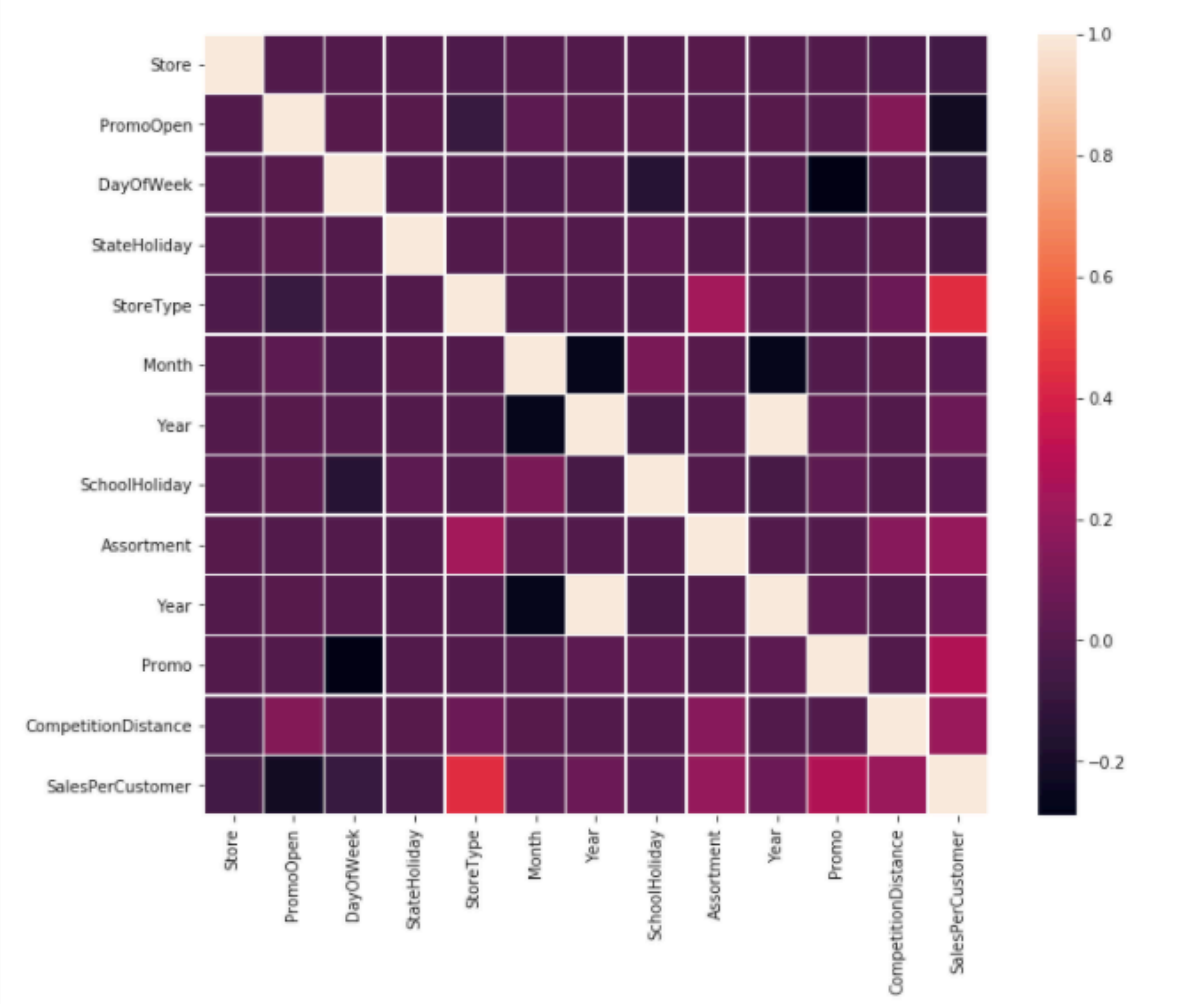
1. 填充缺失数据

- test 中有部分 Open 没有值：填充默认为 1，关闭状态下的销售额预测没有意义。
- store 数据中的 CompetitionDistance 部分缺失：使用中位数进行填充

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	P
290	291	d	a	NaN	NaN	NaN	
621	622	a	c	NaN	NaN	NaN	
878	879	d	a	NaN	NaN	NaN	

- 将其余为空数据填充为 0

2. 将日期拆解
    - 将日期拆解成为年月日，这年的第几周与第几天的形式
3. 对于打折信息与竞争对手信息进行处理
    - 将打折信息处理为是否有打折
    - 将竞争对手信息处理为是否开张
4. 查看 heatmap



执行过程

1. 算法比较

使用了 linear regression 和 decisiontree

```
In [43]: clf_a = LinearRegression()
         clf_a.fit(X_train,y_train)
         y_predict = clf_a.predict(X_test)
         print('LinearRegression',rmse(y_predict,y_test))

         LinearRegression 0.3850515340414187
```

```
In [45]: clf_d = tree.DecisionTreeRegressor()
         clf_d.fit(X_train,y_train)
         print('DecisionTreeRegressor',rmse(clf_d.predict(X_test),y_test))

DecisionTreeRegressor 0.2371703031952833
```

2. gridsearch 的调参数
    - 项目实现过程中使用了 LGB 与 XGBoost 算法
    - 在基准模型的基础上使用了 GridSearch 进行控制变量地参数调整，最后选择了

```
max_depth=10
learning_rate=0.03
colsample_bytree=0.7
subsample=0.9
```

    - LGB 模型的 rmspe 最后卡在了 0.14 因此没有继续进行下去
3. 进行预测找出需要执行的 best_iteration
    -
```
[0]  validation_0-rmse:8.0208   validation_1-rmse:8.02753   validation_0-rmspe:0.999809   validation_1-rmspe:0.999811
Multiple eval metrics have been passed: 'validation_1-rmspe' will be used for early stopping.

Will train until validation_1-rmspe hasn't improved in 100 rounds.
[100]  validation_0-rmse:0.445049   validation_1-rmse:0.44999   validation_0-rmspe:0.354106   validation_1-rmspe:0.345255
[200]  validation_0-rmse:0.161075   validation_1-rmse:0.173896   validation_0-rmspe:0.205586   validation_1-rmspe:0.180458
[300]  validation_0-rmse:0.129715   validation_1-rmse:0.146825   validation_0-rmspe:0.174411   validation_1-rmspe:0.156482
[400]  validation_0-rmse:0.112373   validation_1-rmse:0.133352   validation_0-rmspe:0.154978   validation_1-rmspe:0.14218
[500]  validation_0-rmse:0.102011   validation_1-rmse:0.126068   validation_0-rmspe:0.143288   validation_1-rmspe:0.134588
[600]  validation_0-rmse:0.095485   validation_1-rmse:0.122267   validation_0-rmspe:0.134943   validation_1-rmspe:0.130369
[700]  validation_0-rmse:0.090322   validation_1-rmse:0.118957   validation_0-rmspe:0.126323   validation_1-rmspe:0.126575
[800]  validation_0-rmse:0.086856   validation_1-rmse:0.117416   validation_0-rmspe:0.111744   validation_1-rmspe:0.124896
[900]  validation_0-rmse:0.083745   validation_1-rmse:0.116026   validation_0-rmspe:0.105528   validation_1-rmspe:0.123285
[1000] validation_0-rmse:0.081315   validation_1-rmse:0.115045   validation_0-rmspe:0.100294   validation_1-rmspe:0.122054
[1100] validation_0-rmse:0.079179   validation_1-rmse:0.11442    validation_0-rmspe:0.095875   validation_1-rmspe:0.121354
[1200] validation_0-rmse:0.077322   validation_1-rmse:0.113963   validation_0-rmspe:0.092447   validation_1-rmspe:0.120879
[1300] validation_0-rmse:0.07566    validation_1-rmse:0.113713   validation_0-rmspe:0.088757   validation_1-rmspe:0.120634
[1400] validation_0-rmse:0.074128   validation_1-rmse:0.113375   validation_0-rmspe:0.084704   validation_1-rmspe:0.120284
[1500] validation_0-rmse:0.072764   validation_1-rmse:0.113289   validation_0-rmspe:0.081288   validation_1-rmspe:0.120192
[1600] validation_0-rmse:0.07148    validation_1-rmse:0.113088   validation_0-rmspe:0.079007   validation_1-rmspe:0.120011
[1700] validation_0-rmse:0.070334   validation_1-rmse:0.112896   validation_0-rmspe:0.077053   validation_1-rmspe:0.119811
[1800] validation_0-rmse:0.069213   validation_1-rmse:0.112754   validation_0-rmspe:0.075145   validation_1-rmspe:0.119709
[1900] validation_0-rmse:0.068237   validation_1-rmse:0.112642   validation_0-rmspe:0.073536   validation_1-rmspe:0.119577
[2000] validation_0-rmse:0.067275   validation_1-rmse:0.112582   validation_0-rmspe:0.072165   validation_1-rmspe:0.119523
[2100] validation_0-rmse:0.066353   validation_1-rmse:0.112521   validation_0-rmspe:0.070887   validation_1-rmspe:0.119451
[2200] validation_0-rmse:0.065505   validation_1-rmse:0.112517   validation_0-rmspe:0.069639   validation_1-rmspe:0.119456
Stopping. Best iteration:
[2108] validation_0-rmse:0.066289   validation_1-rmse:0.112513   validation_0-rmspe:0.070805   validation_1-rmspe:0.119433
```
4. 进行预测

完善

增加一些以商店 id 为基准的 feature

观察发现以商店 id 为基准下有很多数据是有参考价值的，因此加入 feature 中，实验结果进步很大

IV. 结果

模型的评价与验证

```
Will train until validation_0-rmspe hasn't improved in 100 rounds.
[100]   validation_0-rmse:0.444758      validation_0-rmspe:0.353499
[200]   validation_0-rmse:0.161195      validation_0-rmspe:0.200256
[300]   validation_0-rmse:0.129785      validation_0-rmspe:0.168512
[400]   validation_0-rmse:0.113178      validation_0-rmspe:0.150021
[500]   validation_0-rmse:0.102567      validation_0-rmspe:0.134289
[600]   validation_0-rmse:0.095952      validation_0-rmspe:0.125939
[700]   validation_0-rmse:0.0907        validation_0-rmspe:0.118542
[800]   validation_0-rmse:0.087042      validation_0-rmspe:0.11431
[900]   validation_0-rmse:0.084001      validation_0-rmspe:0.11025
[1000]  validation_0-rmse:0.081624      validation_0-rmspe:0.102826
[1100]  validation_0-rmse:0.07938       validation_0-rmspe:0.095269
[1200]  validation_0-rmse:0.077511      validation_0-rmspe:0.091722
[1300]  validation_0-rmse:0.07593       validation_0-rmspe:0.089288
[1400]  validation_0-rmse:0.074443      validation_0-rmspe:0.085823
[1500]  validation_0-rmse:0.073116      validation_0-rmspe:0.082742
[1600]  validation_0-rmse:0.071865      validation_0-rmspe:0.080582
[1700]  validation_0-rmse:0.070716      validation_0-rmspe:0.078107
[1800]  validation_0-rmse:0.069649      validation_0-rmspe:0.07615
[1900]  validation_0-rmse:0.068678      validation_0-rmspe:0.07454
[2000]  validation_0-rmse:0.067746      validation_0-rmspe:0.07297
[2100]  validation_0-rmse:0.066841      validation_0-rmspe:0.071779
[2107]  validation_0-rmse:0.066789      validation_0-rmspe:0.071725
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.7, gamma=0,
             importance_type='gain', learning_rate=0.03, max_delta_step=0,
             max_depth=10, min_child_weight=1, missing=None, n_estimators=2108,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=0.9, tree_method='gpu_hist',
             verbosity=1)
```

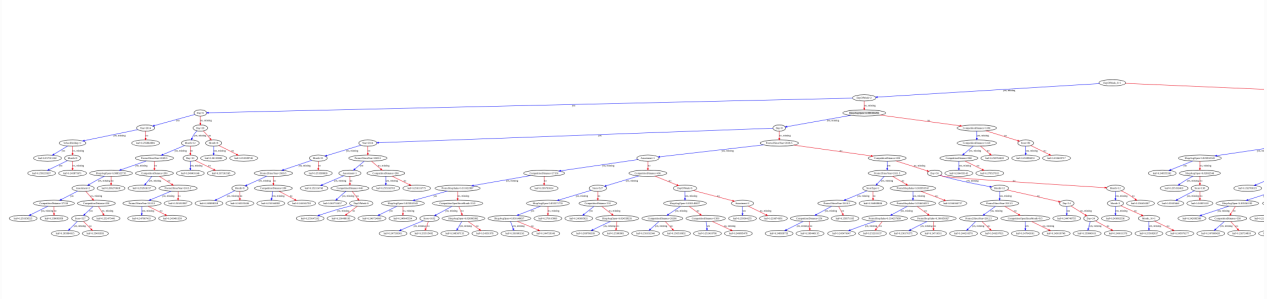
合理性分析

综合各个参数的优化，发现 max_depth 为 10，learning_rate 为 0.03，colsample_bytree为0.7，subsample为0.9，n_estimators 为 2108 时表现最佳

V. 项目结论

结果可视化

结果可视化见附件 xgb.dot.pdf 对树结构的可视化，缩略图截图如下：



对项目的思考

- 对数据的处理占很大一部分的因素

需要作出的改进

- 可将优化后的 feature 作用到 LGB 模型上