Rossmann 销售额预测

机器学习工程师纳米学位毕业项目

薛威

2018 年 02 月 19 日

I. 定义

项目概述

该项目是 Kaggle 上由 Rossmann 两年前建立的一个预测比赛。比赛的目标是取得一个能够正确预测六周后的日销售情况。

Rossmann 在欧洲经营 7 国经营着 3000 家药店，目前，Rossmann 商店的经理被要求预测他们未来六周的日销售情况，商店销售收很多因素的影响，包括促销，竞争，学校放假和法定节假日，节气性和区域性。由于数千经营者依据他们独特的情况预测销售情况，结果的准确性可能有很大不同。 在这个项目中，将挑战预测 6 周的 1115 家德国境内的 Rossmann 商店的每日销售额，可靠的销售预报可以让商店经营者增加工作效率和积极性创建更高效的工作人员安排。通过帮助 Rossmann 创建一个强壮的预测模型，你将帮助经营者保持关注对他们来说最重要的是：他们的客户和他们的团队。背景知识主要来自于 kaggle 项目介绍相关链接在最后的相关引用里。

本项目将使用目前 Kaggle 上线性预测普遍表现很好的 XGBoost 算法模型来建模，并验证所取得的结果。

问题陈述

我要通过对旧的销售记录的学习建模，然后预测六周后的日销售情况。具体步骤和可能遇到的问题如下：

- 对数据做清洗和整理，可能遇到的问题有异常值的处理，缺失值的处理。我将针对存在缺失数据的特征的具体情况来补充或丢弃训练数据，对于每个特征的异常值做一次筛选判断。
- 对数据做单个变量分析，多个变量分析，对变量做特征处理，可能会遇到整合多个变量为一个有效特征的问题。我将通过对问题的理解来对某几个特征做一些基本运算将其转换为更直观的特征展现出来。
- 使用 XGBoost 来建模，如何调整训练模型的数据用来适合 XGBoost 模型的输入要求。我将依照 XGBoost 模型的传入参数的要求来对输入特征做数据转换。
- 在使用 XGBoost 的基础上如何找到最合适的超参。我将使用 CV 来筛选出最好的参数进行建模。
- 预测并评价结果，根据 Kaggle 的描述使用 RMSPE 来做为评价标准，需要对我建立的模型做一个评估。我将使用建立好的模型对于 test 数据进行预测然后将其上传到 Kaggle 上验证其 RMSPE 分数，分数越低表现越好。

评价指标

我将使用 Kaggle 在该项目建议的 RMSPE 来做为验证函数，该值越低代表差异性越小。它是指模型的预测值和实际观察值之间的差异的一种衡量方式。

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

y 是真实的销售数据而 y_hat 是预测数据，任何的为 0 的销售数据不参加此评价。采用该评价函数的好处是该问题是一个线性回归的问题，预测的结果是一个具体的数值向量，相关的预测数据只能是一个大概无法精确匹配，所以计算预测值和实际值之间的差额平方并累加起来平均最后再开方得到的结果相比直接实际值和预测值做差额来说的大大减少了预测值与实际值之间过于详细的数值匹配要求。

II. 分析

数据的探索

Rossmann 提供了相关的数据集，包含训练的和预测比赛所需要的。

- 输入数据集如下
  train.csv - 包括销售额的历史数据训练用
  包含有
  "Store","DayOfWeek","Date","Sales","Customers","Open","Promo","StateHoliday","SchoolHoliday"字段。
  test.csv - 包括销售额的历史数据测试用 historical data excluding Sales
  包含有
  "Id","Store","DayOfWeek","Date","Open","Promo","StateHoliday","SchoolHoliday"字段。
  sample_submission.csv - 预测数据格式样本
  包含有"Id","Sales"字段。
  store.csv - 关于商店的附加信息
  包含有
  "Store","StoreType","Assortment","CompetitionDistance","CompetitionOpenSinceMonth"，
  "CompetitionOpenSinceYear","Promo2","Promo2SinceWeek","Promo2SinceYear","PromoInterval"字段。
- 数据集特征如下
  Id - 测试集中表示一条记录的编号。
  Store - 每个商店的唯一编号。
  Sales - 任意一个给定日期的销售营业额。
  Customers - 给定那一天的消费者数。
  Open - 商店是否开门标志，0 为关，1 为开。
  StateHoliday - 表明影响商店关门的节假日，正常来说所有商店，除了极少数，都会在节假日关门，a=所有的节假日，b=复活节，c=圣诞节，所有

学校都会在公共假日和周末关门。
SchoolHoliday - 表明商店的时间是否受到公共学校放假影响。
StoreType - 四种不同的商店类型 a，b，c 和 d。
Assortment - 描述种类的程度，a = basic, b = extra, c = extended。
CompetitionDistance - 最近的竞争对手的商店的距离。
CompetitionOpenSince[Month/Year] - 最近的竞争者商店大概开业的年和月时间。
Promo - 表明商店该天是否在进行促销。
Promo2 - 指的是持续和连续的促销活动。: 0 = 商店没有参加, 1 = 商店正在参加。 Promo2Since[Year/Week] - 表示参加连续促销开始的年份和周。
PromoInterval - 描述持续促销间隔开始，促销的月份代表新的一轮，月份意味着每一轮的开始在哪几个月。

- 相关的初步统计分析如下
  test.csv 有 41088 条数据其中缺失数据是在特征 Open 上缺失了 11 条，根据缺失的 Date，StateHoliday 和 SchoolHoliday 来判断推测 Open 均为 1 并填入。
  以下是前五条输出样例

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
0	1	1	4	2015-09-17	1.000	1	0	0
1	2	3	4	2015-09-17	1.000	1	0	0
2	3	7	4	2015-09-17	1.000	1	0	0
3	4	8	4	2015-09-17	1.000	1	0	0
4	5	9	4	2015-09-17	1.000	1	0	0

  store.csv 有 1115 条数据其中缺失数据特征是 CompetitionDistance，CompetitionOpenSinceMonth，CompetitionOpenSinceYear，Promo2SinceWeek，Promo2SinceYear 和 PromoInterval。CompetitionDistance 缺失 3 条，我推测这三家商店在有效的距离内没有竞争对手用一个特别大的值来处理，CompetitionOpenSinceMonth 和 CompetitionOpenSinceYear 缺失的情况一直，我推测就在很早之前或者

说在 train 数据之前就存在这个竞争对手了我给一个默认的之前的时间 2010 年，Promo2SinceWeek，Promo2SinceYear 和 PromoInterval。这三个特征的确是情况也都一样，也就是没有参加 Promo2 的这三项均为空，那我就将时间设置为一个未来时间 2030 年，PromoInterval 用“0，0，0，0”来填补。

以下是前五条数据样本

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
0	1	c	a	1270	9	2008	0	0	2030	0,0,0,0
1	2	a	a	570	11	2007	1	13	2010	0,3,6,9
2	3	a	a	14130	12	2006	1	14	2011	0,3,6,9
3	4	c	c	620	9	2009	0	0	2030	0,0,0,0
4	5	a	a	29910	4	2015	0	0	2030	0,0,0,0

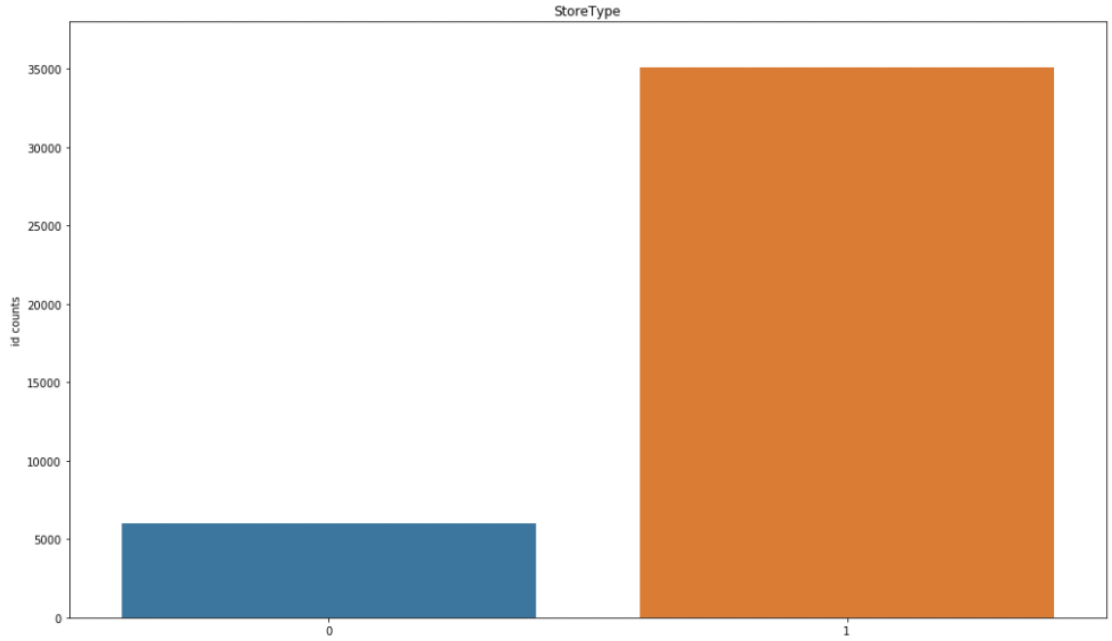
train.csv 有 1017209 条数据，无数据缺失情况，所有提供的数据总体来看没有异常值情况。

样本如下

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

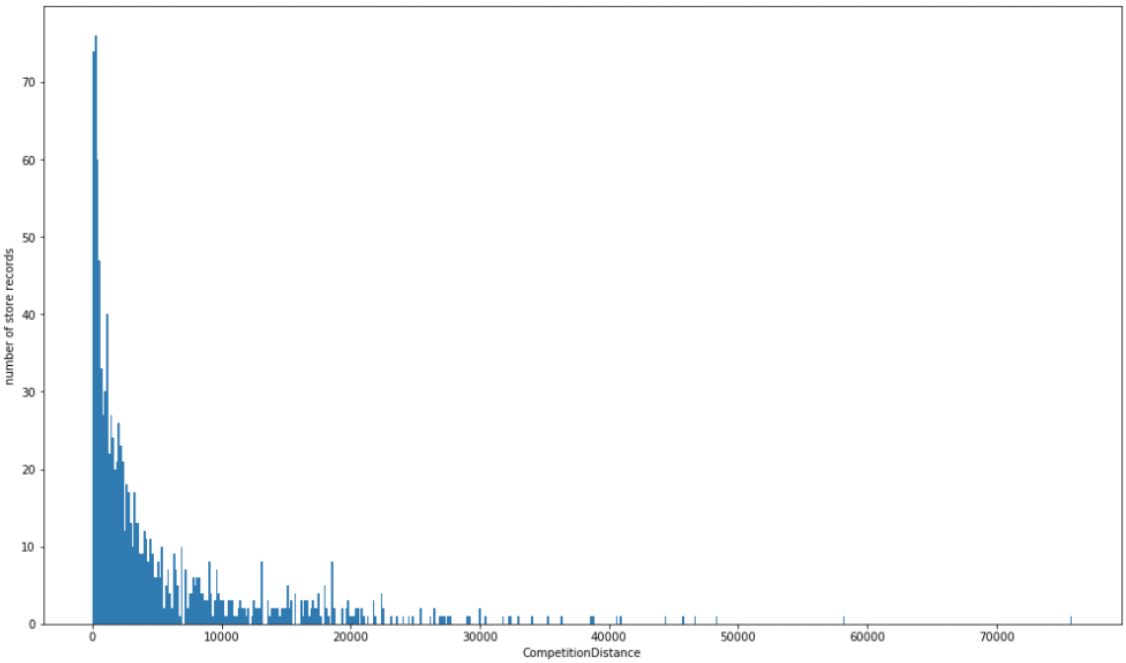
探索性可视化

针对 test 数据集做特征 open 的可视化



test 数据集里的 Open 为 0 的有 5984 条，直接将其预测值设置为 0

针对 store 数据集查看特征 CompetitionDistance 来看各个距离的商店的分布



并查看该特征的详细情况

```
store['CompetitionDistance'].describe()
```

```
count    1112.000
mean     5404.901
std      7663.175
min        20.000
25%       717.500
50%      2325.000
75%      6882.500
max     75860.000
Name: CompetitionDistance, dtype: float64
```

中位数在 2325m，还有部分大于 20km 的，对于三个缺失的值我也做一个最大值 99999 来填充，表示有效距离内无竞争对手。 20km 以上我之后都将标记为无竞争对手

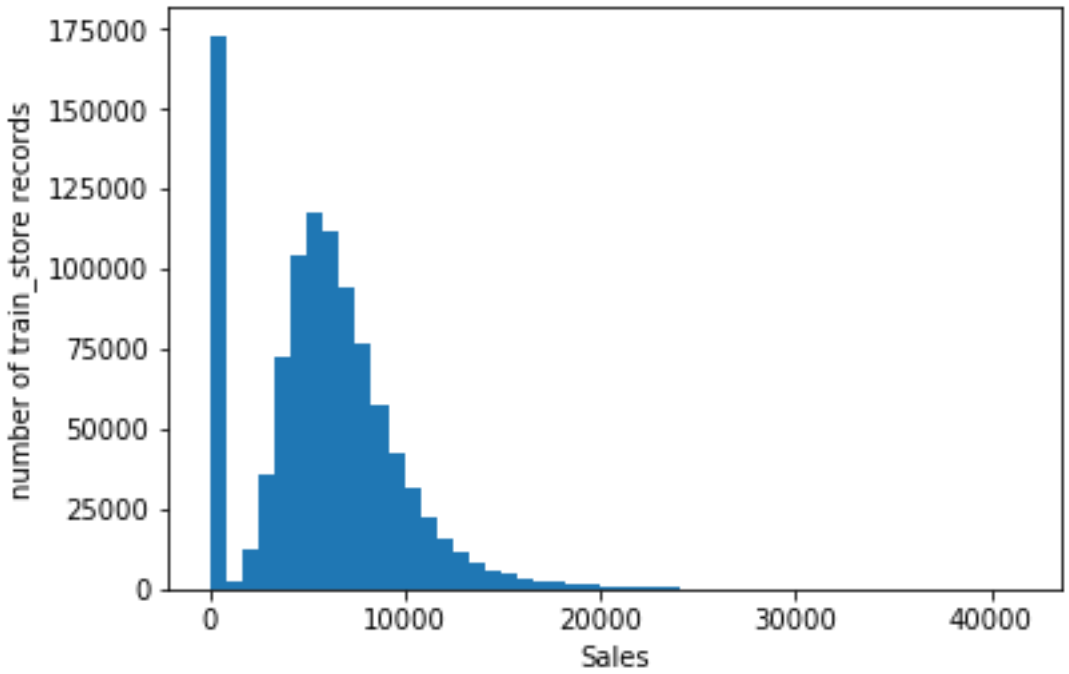
针对 store 数据集查看特征 CompetitionOpenSinceYear 来看竞争对手的开店时间

```
store['CompetitionOpenSinceYear'].describe()
```

```
count     764.000
mean     2008.753
std         6.326
min      1900.000
25%      2006.000
50%      2010.000
75%      2013.000
max      2030.000
Name: CompetitionOpenSinceYear, dtype: float64
```

缺失值就用中位数 2010 来填。

对于 train 数据集对 Sales 做可视化

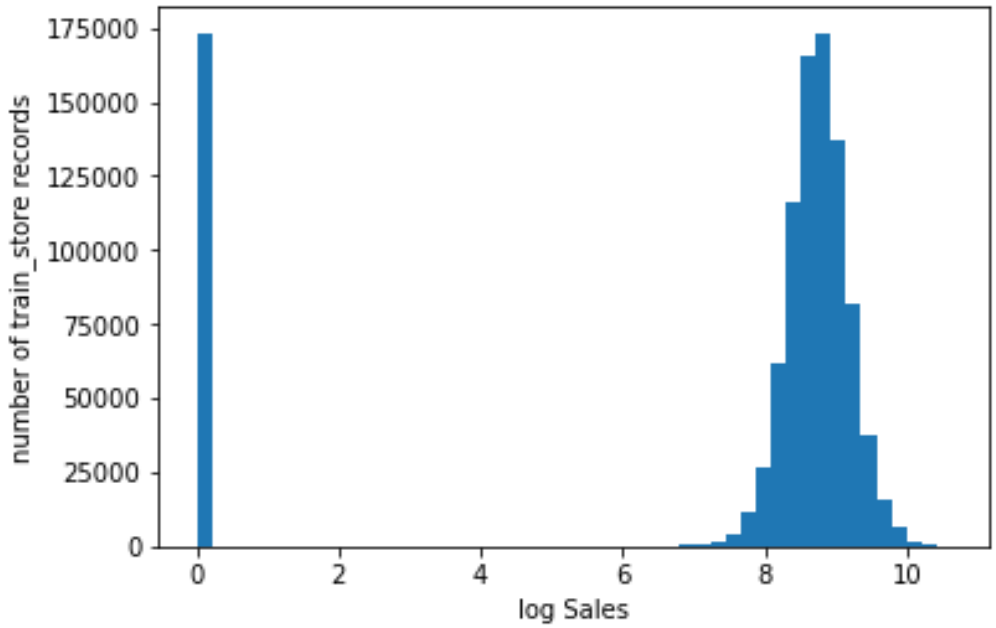


发现有大量为 0 的数据，Sales 为 0 对于训练来说并无意义，而且会干扰最终结果应该只训练 Sales 不为 0 的，同时发现 Open 特征为 0 和 Sales 为 0 表现一致

```
print(train.query('Open=="0"')['Sales'].value_counts())
print(train.query('Open=="0"')['Customers'].value_counts())
```

```
0    172817
Name: Sales, dtype: int64
0    172817
Name: Customers, dtype: int64
```

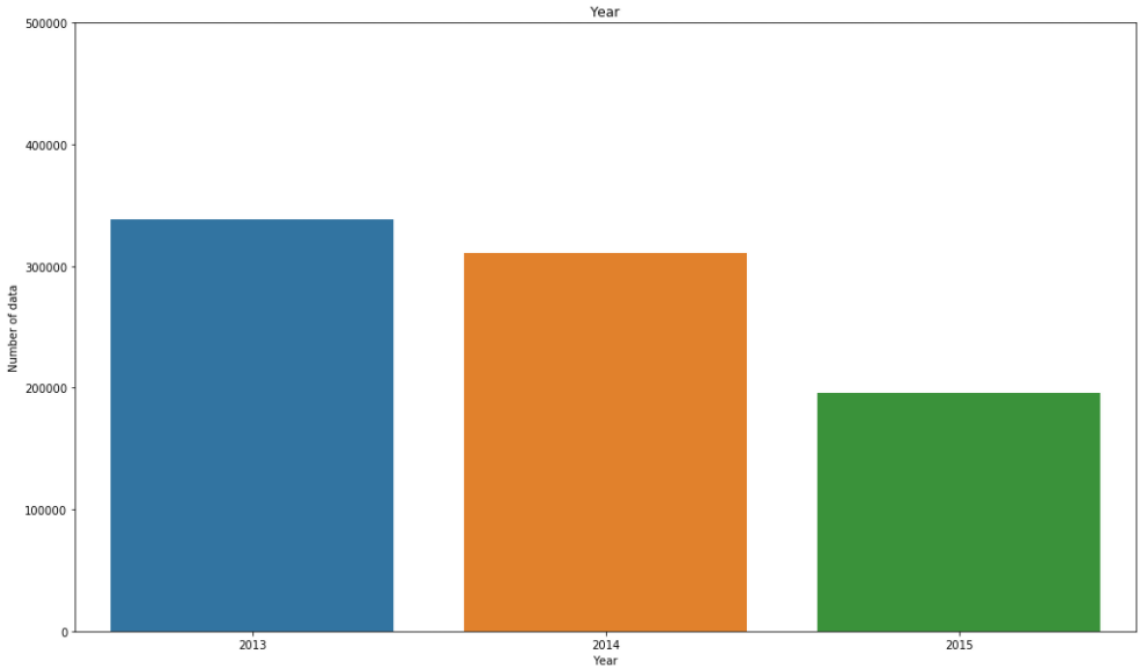
我将只训练 Open 为 1 的数据集，由于该结果集是一个斜态分布作为训练集它将预测结果为了更好的精度我将其转化为正态分布图形如下，不用考虑结果为 0 的它们将不参加训练



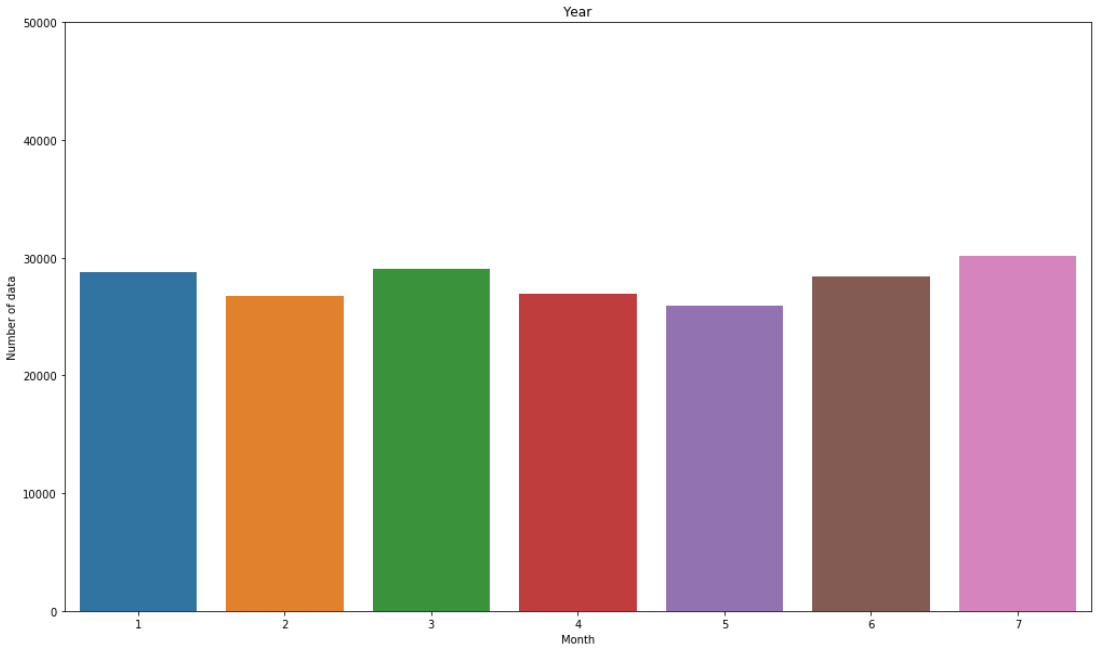
由图可见预测结果主要集中在 8-10 之间。

针对 train 的日期里的 Year 做可视化

```
Year
2013    337943
2014    310417
2015    196032
Name: Open, dtype: int64
```



我使用 2013，2014 做训练，2015 的最后一个月做为 test 验证数据这样可以保证时间的从过去到将来的顺序。
下图是 2015 年涉及的月份



将 7 月作为验证数据。
验证数据和总在的训练集的占比为 3.5751%

算法和技术

该问题属于监督学习里的回归问题，也就是依据已有的数据集建模预测测试集的问题。
在特征上通过预处理将空值根据相关的业务场景做填补，对于异常值经行处理避免过拟合数据，对日期做转换，然后对于部分分类特征比如 Assortment，StoreType 和 StateHoliday 做 one-hot，让 train 和 test 与 store 数据集做连接，最后只考虑 Open 的训练数据。
模型选择上，XGBoost 非常适合于这个领域。第一训练的数据规模并不算大，深度学习需要更大量的数据集，第二数据的形式主要是表格，深度学习主要是针对图像，音频和其他的，而传统的基于统计学的方法更适合，第三在机器学习方法里 xgboost 的速度快，精度高，而且是基于树型的模型模型的解释度高，在实践中 XGBoost 多次赢得 kaggle 比赛表现稳定良好。

XGBoost 该算法的解释首先要从 GB 开始，先生成多个的弱模型，再将每个弱模型的预结果相加，第一个模型是 F0 之后的模型比如 F1 是基于这个模型 F0 再加上 h（x）得来。这个 h 其实就是残差的表示，要通过函数空间的负梯度（残差已知，F(X)已知，真实结果 y 已知）学习来训练生成这个 h，然后通过得出 h 可以继续的更新模型。

接下来是介绍 GBDT 其实就是指 GB 里的弱模型是 DT（决策树）主要是指回归树，而且这个 DT 是个弱模型，叶子节点不超过 10，深度不超过 5，学习率小于 0.1，可以通过交叉验证的方法来选择出最优的参数。

最后回到 Xgboost 它是 GB 算法的高效实现，它的弱学习器可以是 gbtree 也可以是 gblinear。

具体进步在目标函数中细化了正则化项也就是可以更好的泛化避免过拟合，正则化项受叶子的数量和每个叶子的值来决定。

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss　　Complexity of the Trees

在损失函数部分用到了一阶和二阶导数（前者优化经验误差，后者控制泛化误差），而 GB 只是用了一阶导数。

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

Number of leaves　　L2 norm of leaf scores

在分割点的选择上 GBDT 使用了 GINI 系数寻找最小化均方差的贪心算法，而 xgboost 是用了最大化，lamda，gama 且与正则相关的优化的近似值算法提高了准确度和效率。
还有其他的计算过程的优化都提升了减少了计算量提高了计算速度。

在具体的 xgb 使用的时候需要考虑的参数有
General parameters：参数控制在提升（boosting）过程中使用哪种 booster，本项目使用默认值 gbtree。
Booster parameters：这取决于使用哪种 booster，设置如下：
eta [default=0.3]
max_depth [default=6]
Learning Task parameters：控制学习的场景的相关参数 。
objective [“reg:linear”] 。
feval 设置为 RMSPE 为评价函数。
num_boost_round 设置 boosting 的次数。
early_stopping_rounds 可以提前终止程序，这样可以找到最优的迭代次数。

设置完 XGBoost 模型后使用手动将训练集里最后一个月的数据作为 vaild 集而其它时间的数据作为 train 集来训练，最后用全部的 train.csv 的非 0 数据建模预测 test 结果再加上之前提取的结果为 0 的数据做组合之后上传 kaggle 来评估结果。

基准模型

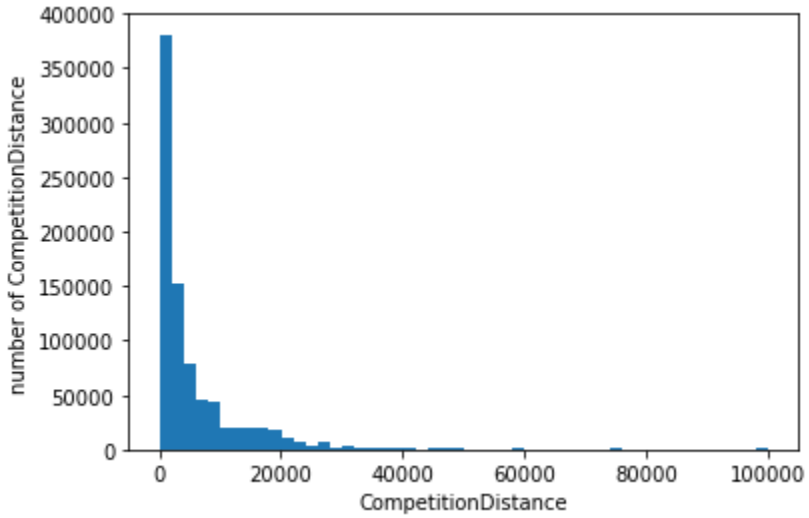
由于是销售预测模型，我对数据预测的错误率认为假设为 20%都是可以接受的范围，我的选择是 XGBoost 来作为我的预测模型，衡量结果的方式是 Kaggle 推荐的 RMSPE。
采用最朴素的均值基准模型用全量测试数据训练，上传 Kaggle 的分数为 0.45019，因此我选择的模型结果至少要超过这个值。

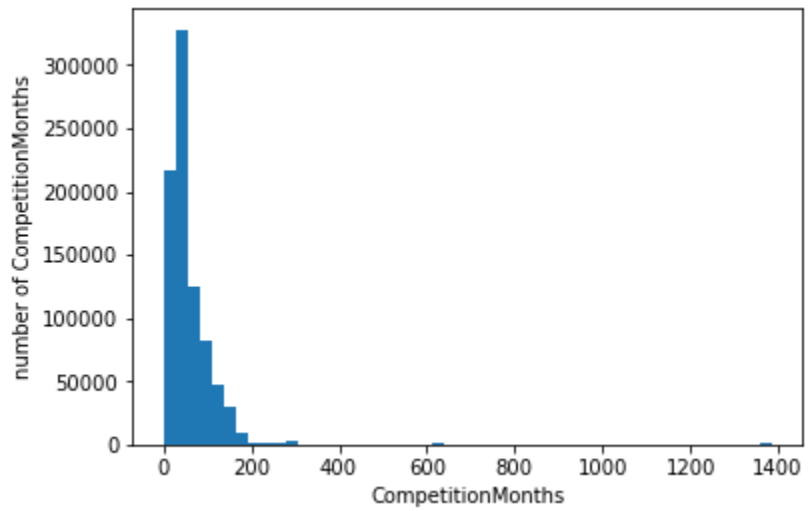
mean_submission.csv	a few seconds ago	3 seconds	0 seconds	0.45019

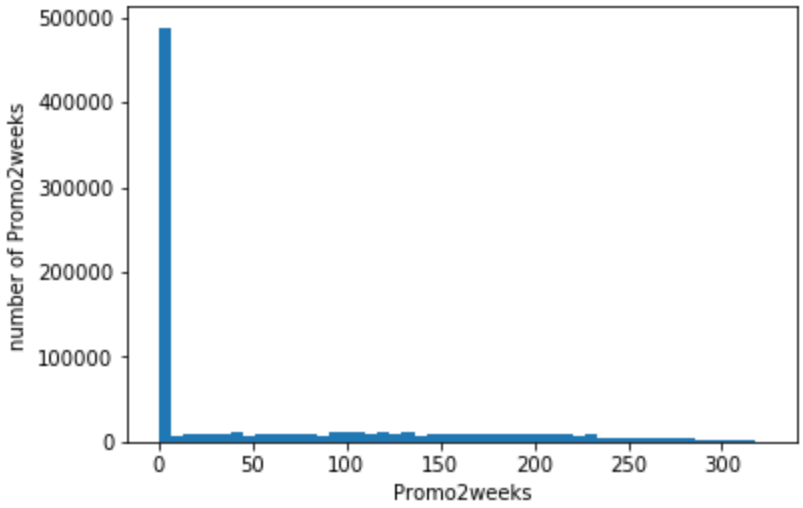
Complete

由于是监督学习的线性回归问题，所以我还将采用 Ridge 回归模型和 Lasso 回归模型来作为基准回归模型计算其结果。

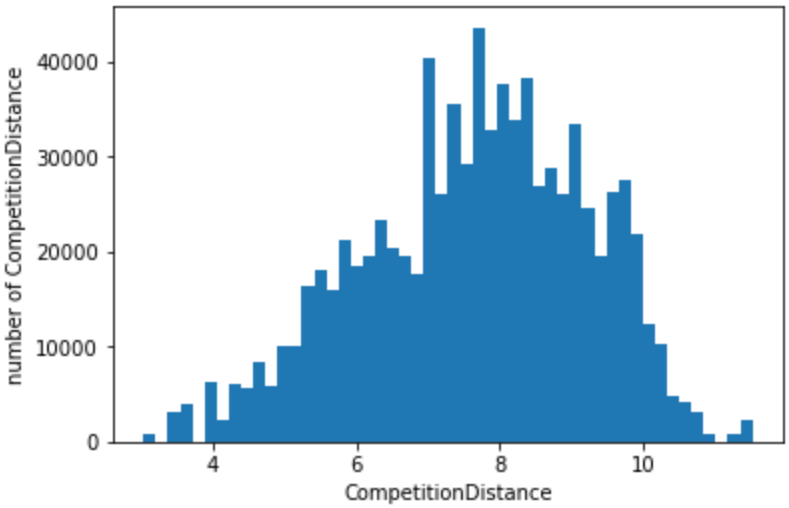
由于是线性算法，特征值是连续值的需要做正态化处理将其分布平滑化，依据是当因变量服从正态分布，误差项满足高斯马尔科夫条件，回归参数的最小二乘估计是一致最小方差无偏估计。涉及的特征有 CompetitionDistance，CompetitionMonths，Promo2weeks。

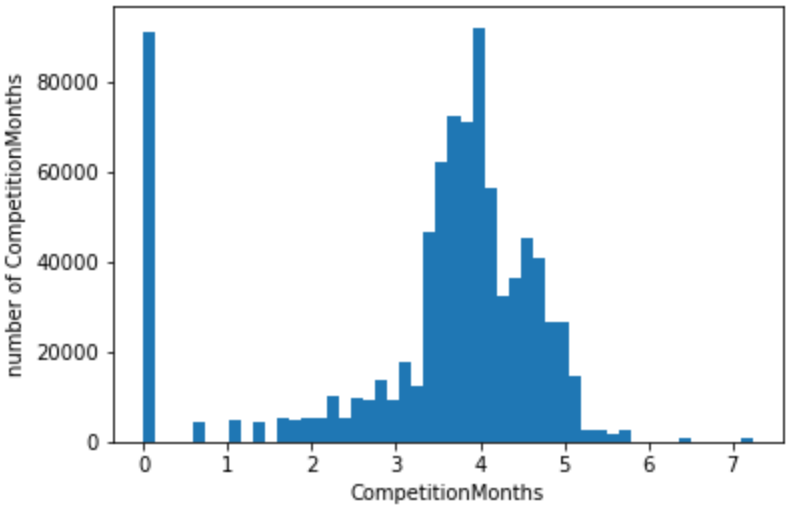


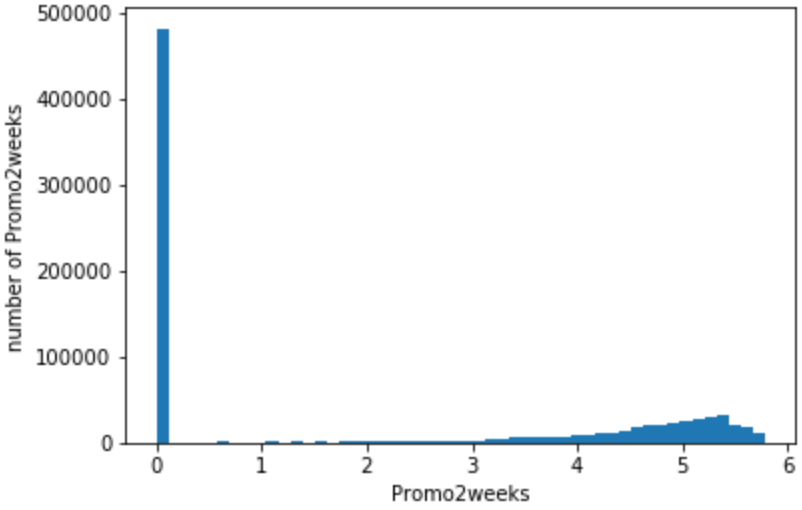



500000
400000
300000
200000
100000
0
number of Promo2weeks
0
50
100
150
200
250
300
Promo2weeks

处理之后特征分布为：


number of CompetitionDistance
40000
30000
20000
10000
0
4
6
8
10
CompetitionDistance


number of CompetitionMonths
80000
60000
40000
20000
0
0
1
2
3
4
5
6
7
CompetitionMonths


number of Promo2weeks
500000
400000
300000
200000
100000
0
0
1
2
3
4
5
6
Promo2weeks

得出的结果如下 Ridge 模型的 Kaggle 评估分数是 0.41781，

Lasso 模型的 Kaggle 评估分数是 0.41467，

这两个结果都略好于朴素预测结果。

选择使用 xgboost 算法预期就是要远远超过这三个算法。

III. 方法

数据预处理

整理 test 数据集的缺失数据，仅有 Open 特征通过判断日期推测出缺失的均为 1，对于 DATE 做数据转换，变成 Year，Month，Day，对 StateHoliday 做 One-Hot。
整理 store 数据集的缺失数据，处理 CompetitionOpenSinceMonth,CompetitionOpenSinceYear 缺失通过取中位数来填补，Promo2SinceWeek，Promo2SinceYear 和 PromoInterval 的缺失通过设置为 0，远大于目前的时间 2030，（0，0，0，0）来处理为了后面连接 train 和 test 来设置没有参加 Promo2 活动，对于 Date 也做相同的数据转换，将 PromoInterval 里的 string 用数字来做替换，对 StoreType 做 One-Hot。
整理 train 数据集，转换特征 Date，对 StateHoliday 做 One-Hot，但是只对

Open=1 的做训练。
将 train 和 test 都与 store 数据通过 store 特征做连接得出 train_data 和 test_data。对这两个数据集相同的处理，将 CompetitionOpenSinceMonth,CompetitionOpenSinceYear 合并为一个特征 CompetitionMonths 表示竞争对手的开业持续月份。将 Promo2SinceWeek Promo2SinceYear PromoInterval 合并为一个特征 IsPromo2 表示当前时间里是否在参加 Promo2。
对于结果值由于 Sales 是一个偏正分布，为了把数据到统一到一个数量级，把它转化为一个正态分布，一般采用对数的方式，它可以额加快梯度下降求最优解的速度，还可以提高计算的精度。所以得到的预测结果要再做一个逆运算才能提交评估打分。

执行过程

- 先建立测试函数 rmspe
  rmspe = np.sqrt(np.mean(w * (y - yhat)**2))
  用来评价模型的准确率。
- 采用 Ridge 算法建模
  使用 Sklearn 的 linear_model 里的 Ridge 建模，
  clf = Ridge(alpha=1)
- 采用 Lasso 算法建模
  使用 Sklearn 的 linear_model 里的 Lasso 建模，
  clf = Lasso (alpha=1)
- 采用 Xgboost 算法建模
  通过在 windows 平台下重新编译 xgboost 库调用 python 接口来使用我选择的是 CPU 版本
  gbm = xgb.train(params, dtrain, num_trees, evals=watchlist, early_stopping_rounds=100, feval=rmspe_xg, verbose_eval=True)

完善

对于 Ridge 算法模型的使用了 RidgeCV 来做参数的选优
clf = RidgeCV(alphas=[0.1, 0.5, 1.0, 10.0], cv=10,
fit_intercept=True,scoring=rmpse_estimator)
优化后结果变化不大基本还是在 0.370419。

对于 Lasso 模型采用了 LassoCV 的方式来优化参数
clf = LassoCV(cv=20)
优化后的结果是 0.471350。

xgboost 的初始结果是 0.194368
用上所有的训练集训练将树调成 8000，然后再对测试集做预测，结果提交 kaggle 是 0.11742

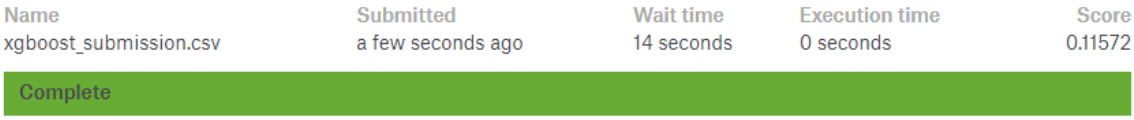
IV. 结果

模型的评价与验证

Ridge 建模使用 RidgeCV 方法训练数据集大小为 844392
训练使用时间是 27.753535 秒
需要预测的 test_data 数据集大小为 35104
预测所用时间是 0.008522
预测结果上传 Kaggle 进行评分是 0.41700

Lasso 建模使用 LassoCV 方法训练数据集大小为 844392
训练使用时间是 13.514766 秒
需要预测的 test_data 数据集大小为 35104
预测所用时间是 0.009526
预测结果上传 Kaggle 进行评分是 0.43365

XGBoost 建模使用 xgb.train 方法训练数据集大小为 844392
训练使用时间是 11 分 8 秒
需要预测的 test_data 数据集大小为 35104
预测所用时间是 1.973393
预测结果上传 Kaggle 进行评分是 0.11572



合理性分析

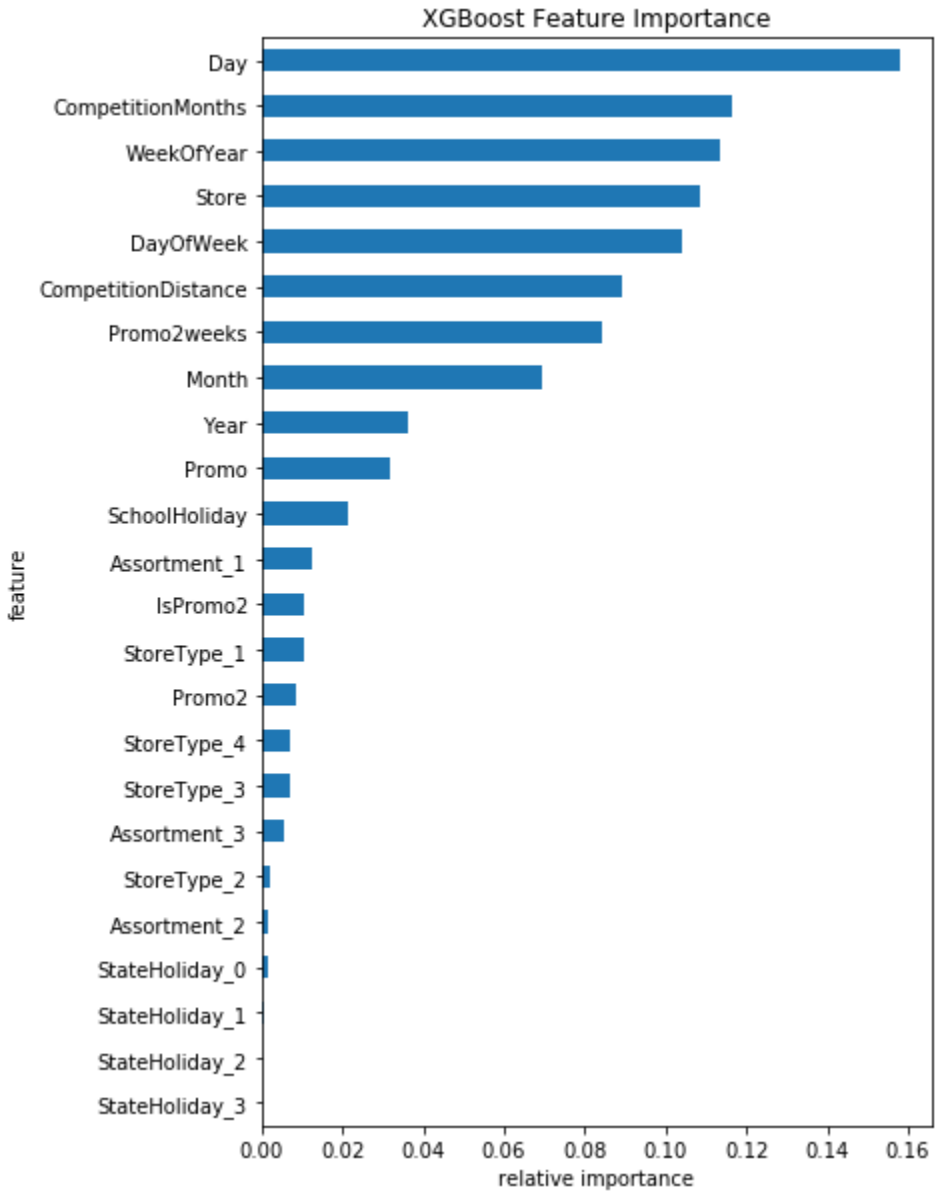
Ridge 和 Lasso 模型成本低于预期主要是训练时间短，对复杂输入数据无法很好的拟合，它的惩罚方式也造成它很难校准，根据本项目的输入特征来看得出 0.4-0.45 左右的结果也不算意外。

XGBoost 训练时间比较长，特别是深度增加，树的数量增加以后，但是模型的预测表现很好，已经满足了一开始制定的 0.2 的目标，进行初步的设置和参数的优化之后达到了 0.11572，这和它自身的特点分不开，它在代价函数里加入正则项可以很好的控制复杂度，作为树型的模型然后还可以从底到顶反向减枝避免陷入局部最优，因此可以相对比较轻松的拿下这个成绩。

V. 项目结论

结果可视化

展现 Xgboost 训练过程中特征重要性的排序



点评一下排名前五的相关度特征。

时间是第一重要的特征，因为比如节假日，竞争对手开业时间，学校放假，经济的发展情况都会与时间相关联，因此就会和销售额关联度高度相关。

竞争对手的情况也是一个很重要的特征，毕竟会导致客户的分流影响销售额。

商店不同预期的结果也不同，所以 store 也是很重要的。

Weekofyear 和 Dayofweek 也是与时间关联所以相关度高也正常。

对项目的思考

项目应该分为数据整理，模型选择，评估及优化三个部分，后面两个部分可以合并互相影响。
最重要的地方是在数据整理，这也是最有意思，最困难的地方，需要了解这个问题的背景和各个特征的意义，根据各个特征的实际情况和常识来补充缺失值，然后找出那些有密切实际联系的特征，将其转换为新的更直观更适合算法模型的特征。
模型选择方面，该项目是一个监督学习的回归问题，主流的监督学习回归算法都是可以适用的，通过设定一个基准分数尝试多个回归模型来比较选择其中表现最好的。模型训练时间在建模的时候成本很高，但是在建模成功使用其进行预测的时候建模成本并不影响。
模型的评估和优化是一个永恒的主题，训练建模的时间制约了尝试训练的次数，可以采用 cv 的方式寻找到最佳的参数。但是最好的优化还是在第一步数据整理上，一个项目的数据整理决定了模型的上限，再好的模型和优化都只是无限逼近这个上限。
目前我的 XGBOOST 建立的模型达到了我的预期，在通用场景的话可以按照这个思路来训练建立模型，但还是要考虑每个场景的具体情况。

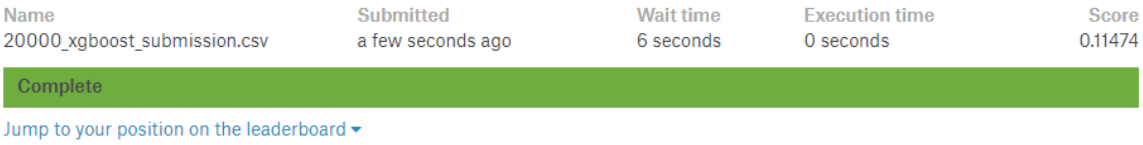
对于该项目我觉得还有些特征的理解我有疑问，商店里的商品那么多做 Promo2 的商品是哪个牌子的哪个型号，难道预测的是笼统销售活动；竞争对手开店时间 1900 应该是个默认值所以应该不是很准确；顾客数在预测工作中应该是很重要的，而我在这个模型里却没有想到改如何使用。

需要作出的改进

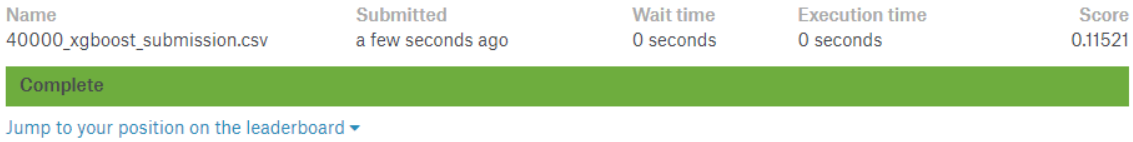
在数据整理分析阶段我还可以进一步的尝试处理 CompetitionDistance 这个特征，将其分段离散化，可以更好的正则化模型，CompetitionMonths 这个特征也

可以考虑分段离散化，而且它们都是重要性排名靠前的特征，深入研究应该会有不错的提升。
算法方面通过 CV 来找到 XGBOOST 的最佳参数继续做试验优化，由于本项目的数据特点在做 CV 操作的时候可以使用 TimeSeriesSplit 来切割做 CV，我通过调大树的数量来提高成绩在 20000 颗树的情况下耗费 1 小时 57 分成绩提升到 0.11470，但是再提高到 40000 除了训练时间变长，分数反而由于过拟合变高为 0.11521。

20000 的结果

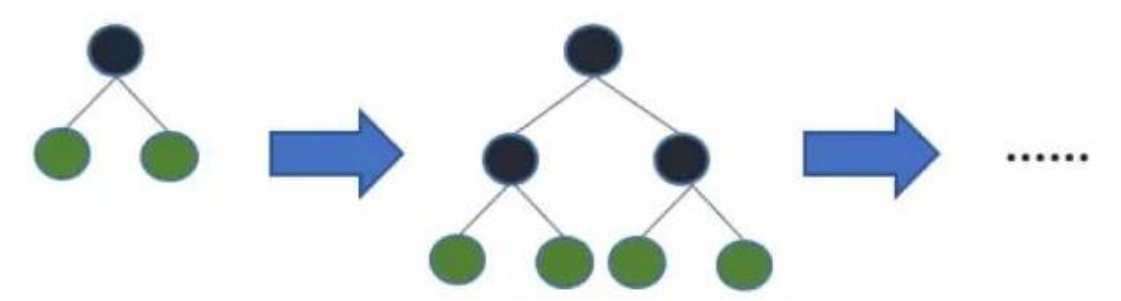


40000 的结果

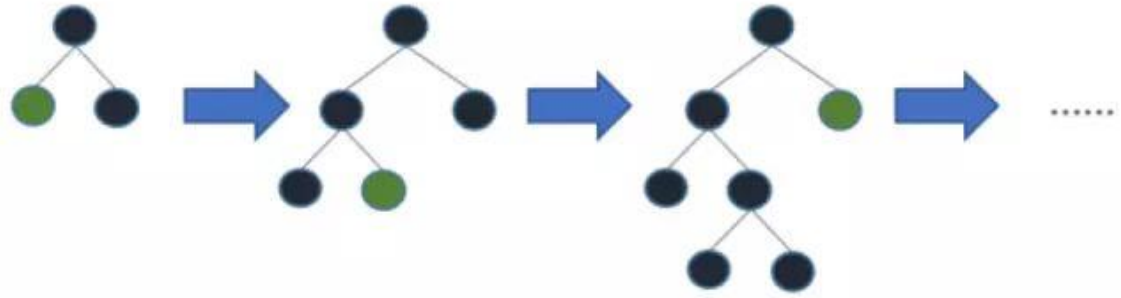


调参技巧方面升级到 GPU 版本进行尝试缩短调参后跑数据的时间。

还可以尝试使用微软刚开源不久的 lightGBM 算法来做一个比较看看成绩会不会更好。
相对于 XGBoost 我了解到的 lightGBM 区别和特点有：
1，决策树算法不一样前者是 pre-sorted 算法找分割点更精确，后者是 histogram 占用内存低，分割复杂度低。
2，树生长策略不一样，前者是 level（depth）-wise 能够同时分裂同一层叶子实现多线程并行，不易过拟合，缺点是不区分对待同层叶子，带来了多的开销。后者是采用 leaf-wise，每次从当前叶子中找出分裂增益最大的一个叶子，开始进行分裂，然后循环，缺点是会产生较深的决策树，导致过拟合。



Xgboost 的 level（depth）-wise



lightGBM 的 leaf-wise

3，通信量不一样，前者采用 pre-sorted 通信代价并行时是 histogram 算法，后者是 histogram，再使用集合通信算法实现并行计算加速。

可能的结论就是在结果差距不大的情况下 lightGBM 的速度会快些。论证需要后续继续试验。

VI.相关引用

http://xgboost.readthedocs.io/en/latest/model.html

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

https://www.kaggle.com/beiwenwu/xgboost-in-python-with-rmspe

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

https://www.quora.com/Why-is-xgboost-given-so-much-less-attention-than-deep-learning-despite-its-ubiquity-in-winning-Kaggle-solutions

http://www.cnblogs.com/wxquare/p/5541414.html

http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

https://zhuanlan.zhihu.com/p/24498293