机器学习工程师纳米学位毕业项目
基于机器学习的数据分析

Rossman 销售预测

战柏瑞

2018 年 6 月 23 日

目录

1 定义

1.1 项目陈述

该项目是 Kaggle 上 Rossmann 公司举办的一个竞赛项目.Rossmann 是德国第二大药品销售链,在欧洲 7 个国家拥有近 3600 家药店.公司由 Dirk Rossmann 建立于 1972 年[i].在 2015 年 Rossmann 的管理层被指派预测近 6 周的日销售额.销售额会被很多因素印象例如, 促销,竞争对手,学校以及州节假日,季节性,和地域性.[ii]

1.2 问题陈述

在分类中，我们想了解模型隔多久正确或不正确地识别新样本一次。而在回归中，我们可能更关注模型的预测值与真正值之间差多少。[iii] 通过对 Rossmann 数据分割,得到 35 天的销售数据,然后使用 xgboost 进行回归预测,因为此次项目更关注预测值和真实值之间的差,所以是一个回归问题.

通常在处理此类问题上,有两种大方向的模型可以选择,一是定性分析法,二是定量分析法.定性分析法是在预测人员具备丰富的实践经验和广泛的专业知识的基础上，根据其对事物的分析和主观判断能力对预测对象的性质和发展趋势作出推断的预测方法，如市场调研法和判断分析法.在本次项目中使用的是定量分析法,也就是通过现代数学方法对历史资料进行分析加工,从而得出结果.

随着市场经济的发展和经济的全球化，企业面临着越来越残酷的市场竞争。企业要想赢得竞争、赢得客户，就必须在最快的时间内，以最低的成本将产品提供给客户，这使得进行正确及时的产品销售预测及由此产生的可靠的决策，成为现代企业成功的关键要素.所以假如可以针对销售进行更加准确的预测时,企业的盈利概率会加大.[iv]

1.3 评价指标

通过对测试集的 Root Mean Square Percentage Error (RMSPE)对已知模型进行评价.

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

[v]

Y_i 为单个门店单天的销售额 $Y_i$hat 是对此进行的预测.将单个门店单天销售额与预测值相减作为 error 差值.然后对每个门店 error 占单个门店单天销售额的百分比的方差进行求和并取平均值.通过这个值来评价模型的好坏.

2 分析

2.1 数据研究

使用 Kaggle website (https://www.kaggle.com/c/rossmann-store-sales)的数据,当训练时,使用 sklearn.cross_validation 包中的 train_test_split,分出十分之一的数据进行验证.

```
#查找缺失值
for i in df_train,df_test,df_store:
    print i.shape
    print i.isnull().sum()
    print ""
```

```
(1017209, 9)
Store            0
DayOfWeek        0
Date             0
Sales            0
Customers        0
Open             0
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64

(41088, 8)
Id               0
Store            0
DayOfWeek        0
Date             0
Open            11
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64

(1115, 10)
Store                         0
StoreType                     0
Assortment                    0
CompetitionDistance           3
CompetitionOpenSinceMonth   354
CompetitionOpenSinceYear    354
Promo2                        0
Promo2SinceWeek             544
Promo2SinceYear             544
PromoInterval               544
dtype: int64
```

在对各个数据集进行大小以及缺省值的检查.df_train 由 train.csv 导入.数据集无缺省值,大小为 1017209 条数据有 9 列表项,其中 Store 代表商店编码,DayofWeek 为数据所在的星期第几天,Date 通过 pandas.to_Datetime 在传入数据时进行格式编码,存入 train.Date.dt, Sales 为销售额,为所需要预测的目标,Customer 为顾客数量. Open 为是否开业. Promo 为是否促销,StateHoliday 为是否是国立节日,并且对不同节日有不同标识,SchoolHoliday 为是否是学校假日.

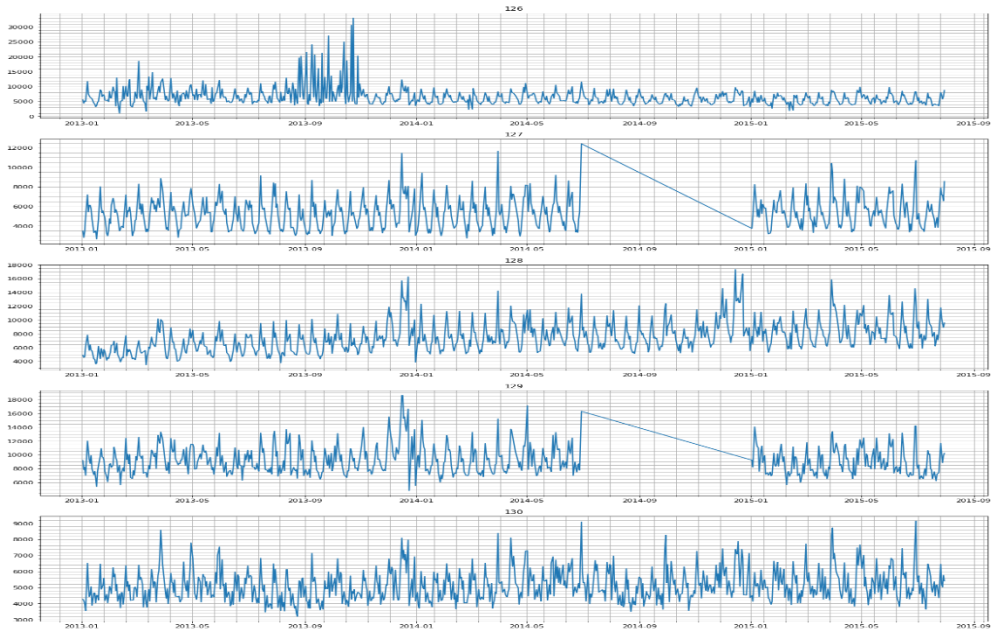
df_test 由 test.csv 导入.数据集有缺省值,open 属性缺少 11 个, 大小为 41088 条数据有 8 列表项,处理方式为补全为 1.理由是因为此次 kaggle 竞赛中,最终对销售为 0 的数据,也就是不开业的数据,不计入 rmspe 中,由此可见若是补全为 0,如果预测数据 open 为 1 对最终 rmspe 值影响巨大.而补全为 1, 如果预测数据 open 为 0,kaggle 评分系统自动筛除. df_test 的表项和 df_train 的表项基本一致,除了没有预测目标 sales,和 customer 顾客数量,多了 Id 用于测试集提交后便于 kaggle 评分.

df_store 为 store.csv 导入.有部分数据缺失,两列表项缺失数据约三分之一,还有三列表项缺失数据约二分之一.大小为 1115,有 10 个表项.其中 Store 与 df_test 和 df_train 中的 store 一致,可以用于合并.StoreType 为商店类型,Assortment 为品种类型.CompetitionDistance 为竞争对手距离, CompetitionOpenSinceMonth 为竞争对手开业月份. CompetitionOpenSinceYear 为竞争对手开业年份.竞争对手开业月份和年份各缺失 354 项.Prome2 为促销 2.

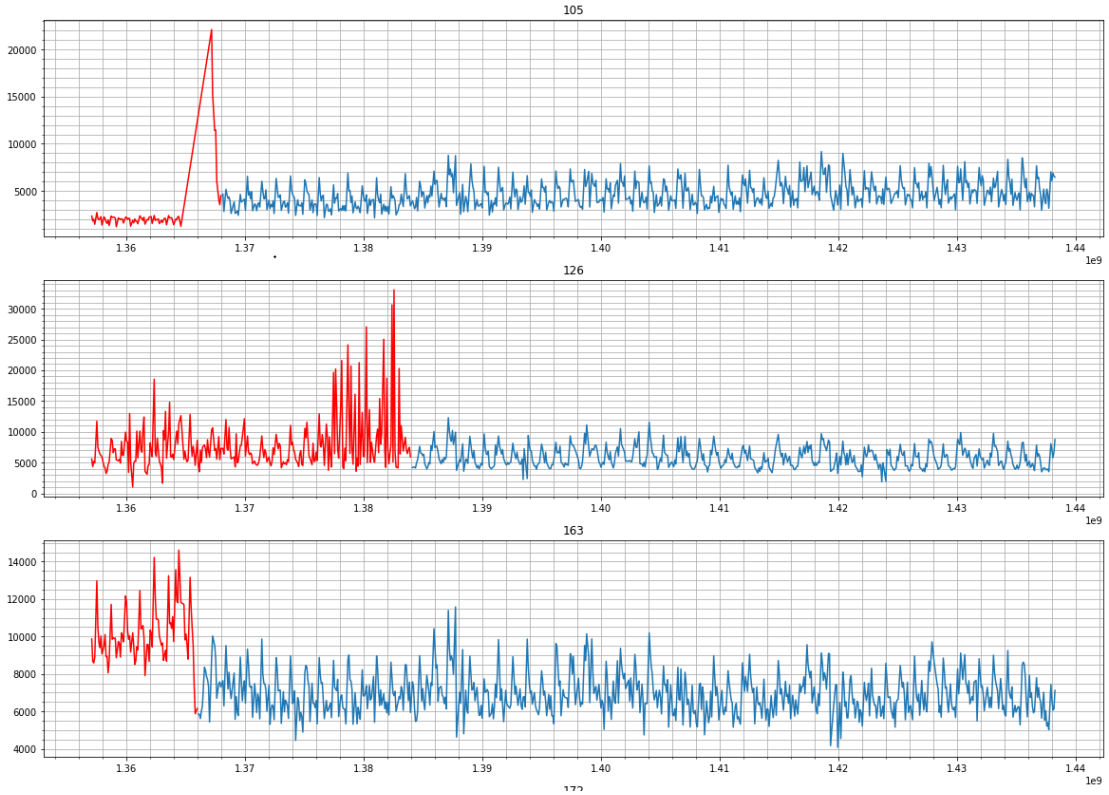
promo2sinceWeek 为促销 2 开始的周(通过查看转化为 category 并且查看,发现最大值为 50,大致判定这里指一年的第几周), promo2sinceYear 为促销 2 开始年份. PromoInterval 为促销的时间段,有 PromoInterval 有'Jan,Apr,Jul,Oct', 'Feb,May,Aug,Nov'以及'Mar,Jun,Sept,Dec'也就是一,四,七,十月促销, 或者二,五,八,十一月促销,或者三,六,九,十二月促销,上述三项各有 544 个缺省值.由于 xgboost. DMatrix 本身支持处理缺省值,而且默认为 np.nan,但是由于 cat.codes 处理时,nan 处理为-1,所以在进行测试之前,将缺省值再赋值为 np.nan.[vi]

将 train.csv 与 test.csv 进行合并,通过给与新增特征 Train 和 Test 进行区分,其中对最终测试集,Train 特征设置为 0,Test 设置为 1,对训练集,Train 特征设置为 1,Test 特征为 0. 如图所示,合并后数据,拥有 Store,DayOfWeek,Date,Sales,Customers,Open,Promo,StateHoliday,SchoolHoliday, Train,Test,以及 Id 这些特征,其中对 stateholiday 和 schoolholiday 使用 DataFrame. astype('category').cat.codes 进行编码转换,由原来 a,b,c 等 string 转化为可训练数字 1,2,3,4 等.

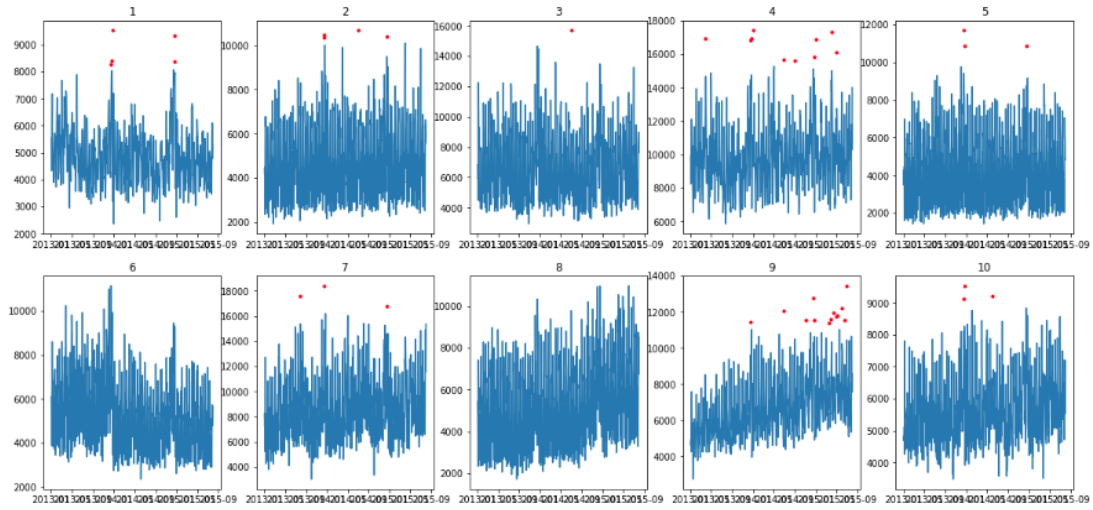
2.2 探索性可视化



由上图可见,store 为 126 的数据前半段数据有一段浮动较大,可以作为异常值删除,看别的数据可以发现,数据本身细微波动.



通过一轮 20 一个 Store 手动排查,发现很多数据,有一段靠前数据其实和靠后数据均值相差较大,通过时间戳直接将不需要的数据进行删除.



另外还有一部分数据为 outlier,通过函数 outlier 数据将会有特征 outlier 设为 true 将其剔除于训练集.剔除依据为 Modified Z-Scores.[vii]

2.3 算法与方法

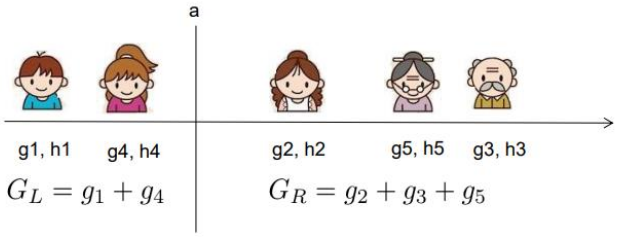
此次项目通过 XGboost 来完成. XGBoost 实现的是一种通用的 Tree Boosting 算法，此算法的一个代表为梯度提升决策树（Gradient Boosting Decision Tree, GBDT），又名 MART(Multiple Additive Regression Tree)。[viii]

GBDT 的原理是，首先使用训练集和样本真值（即标准答案）训练一棵树，然

后使用这棵树预测训练集，得到每个样本的预测值，由于预测值与真值存在偏差，所以二者相减可以得到“残差”。接下来训练第二棵树，此时不再使用真值，而是使用残差作为标准答案。两棵树训练完成后，可以再次得到每个样本的残差，然后进一步训练第三棵树，以此类推。树的总棵数可以人为指定，也可以监控某些指标（例如验证集上的误差）来停止训练。[ix]

Efficient Finding of the Best Split

- What is the gain of a split rule $x_j < a$? Say x_j is age



- All we need is sum of g and h in each side, and calculate

$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

- Left to right linear scan over sorted instance is enough to decide the best split along the feature

在 xgboost 中，函数会对各个数据进行分类，分类进行是，会计算 gain 函数，如果 gain 小于 0，则函数认为此次分类不太适合模型.

Elements continued: Objective Function

- Objective function that is everywhere

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

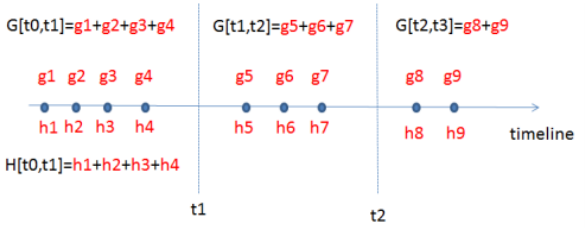
Regularization, measures complexity of model

- Loss on training data: $L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$
  - Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$
- Regularization: how complicated the model is?
  - L2 norm: $\Omega(w) = \lambda \|w\|^2$
  - L1 norm (lasso): $\Omega(w) = \lambda \|w\|_1$

Obj 代表了当我们指定一个树的结构的时候，我们在目标上面增多多减少多少。我们可以把它叫做结构分数(structure score)。你可以认为这个就是类似 gain 系数一样更加一般的对于树结构进行打分的函数。

Questions to check if you really get it

- Time series problem



- All that is important is the structure score of the splits

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

  - Top-down greedy, same as trees
  - Bottom-up greedy, start from individual points as each group, greedily merge neighbors
  - Dynamic programming, can find optimal solution for this case

当遇到时序问题是，xgboost 会将临近数据集分在一起，通过判断 obj 函数的大小，来将一定类的数据分组在一起.

2.4 基准测试

设定基准阈值为 kaggle 排行榜前 10%（330/3303），也就是在 Private Leaderboard 上的分数要低于 0.11733。[x]

3 方法

3.1 数据预处理

```
#显示合并后数据属性
df_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1058297 entries, 0 to 41087
Data columns (total 12 columns):
Store            1058297 non-null int64
DayOfWeek        1058297 non-null int64
Date             1058297 non-null datetime64[ns]
Sales            1017209 non-null float64
Customers        1017209 non-null float64
Open             1058286 non-null float64
Promo            1058297 non-null int32
StateHoliday     1058297 non-null object
SchoolHoliday    1058297 non-null int32
Train            1058297 non-null int64
Test             1058297 non-null int64
Id               41088 non-null float64
dtypes: datetime64[ns](1), float64(4), int32(2), int64(4), object(1)
memory usage: 96.9+ MB
```

将 train.csv 与 test.csv 进行合并,通过给与新增特征 Train 和 Test 进行区分,其中对最终测试集,Train 特征设置为 0,Test 设置为 1,对训练集,Train 特征设置为 1,Test 特征为 0. 如图所示,合并后数据,拥有 Store,DayOfWeek,Date,Sales,Customers,Open,Promo,StateHoliday,SchoolHoliday, Train,Test,以及 Id这些特征,其中对 stateholiday 和 schoolholiday 使用 DataFrame. astype('category').cat.codes 进行编码转换,由原来 a,b,c 等 string 转化为可训练数字 1,2,3,4 等.

```
#显示df_store数据属性
df_store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
Store                        1115 non-null int64
StoreType                    1115 non-null object
Assortment                   1115 non-null object
CompetitionDistance          1112 non-null float64
CompetitionOpenSinceMonth    761 non-null float64
CompetitionOpenSinceYear     761 non-null float64
Promo2                       1115 non-null int64
Promo2SinceWeek              571 non-null float64
Promo2SinceYear              571 non-null float64
PromoInterval                571 non-null object
dtypes: float64(5), int64(2), object(3)
memory usage: 87.2+ KB
```

store 处理相对复杂,对 StoreType,assortment 进行字符转数字处理,,将 competitionsincemonth 和 competitionsinceyear 合并并且转化为时间戳.同理,promo2sinceweek 和 promo2sinceyear 进行合并转化, 对于 PromoInterval 进行 DataFrame. astype('category').cat.codes 编码转换,并且使 'Jan,Apr,Jul,Oct'对应 1 'Feb,May,Aug,Nov'对应 2 'Mar,Jun,Sept,Dec'对应 0,缺省值 nan 对应-1,然后对后期和 df_raw 合并时,通过使用 PromoInterval 生

成新特征时更加方便.
对每个 store 的 sales, customer 和 open 求和,并且生成三个新特征,SalesPerDay,CustomersPerDay 以及 SalesPerCustomersPerDay 先存入 Store,然后在对 store 数据集和 df_raw 进行合并.
最终生成数据集 Train,通过 Month %3 和 PromoInterval 做对比,判断当前月份是否是在促销中,存入新特征 IsPromoMonth.

```
X_train[features].isnull().sum()
```

```
Store                         0
Promo                         0
SchoolHoliday                 0
StateHoliday                  0
StoreType                     0
Assortment                    0
Promo2                        0
CompetitionDistance        1957
CompetitionOpenTS        239559
Promo2TS                 376494
SalesPerDay                   0
CustomersPerDay               0
SalesPerCustomersPerDay       0
IsPromoMonth             376494
DayOfWeek                     0
Month                         0
Day                           0
Year                          0
WeekOfYear                    0
DateTS                        0
dtype: int64
```

最终训练集,含有 20 个特征,缺失值由 xgboost 处理.

3.2 实施

```
params = {"objective": "gpu:reg:linear",
          "booster" : "gbtree",
          "eta": 0.01,
          "max_depth": 12,
          "subsample": 0.7,
          "colsample_bytree": 0.5,
#           "min_child_weight": 1,
          "silent": 1,
          "seed": 42,
          'nthread':6,
#           "tree_method":"gpu_hist"
          }
num_boost_round = 20000

print("训练准备完成")
# X_train = train.loc[(train['Outlier'] == False) & (train['Delete'] != False) ]
# X_valid = valid.loc[(valid['Outlier'] == False) & (valid['Delete'] != False) ]
# y_train = np.log(X_train.Sales)
# y_valid = np.log(X_valid.Sales)
X_train, X_valid, y_train, y_valid = train_test_split(train.loc[(train['Train'] == 1) & (train['Open'] == 1) & (train['Outlier'] == False)][fea
                                                      np.log1p(train.loc[(train['Train'] == 1) & (train['Open'] == 1) & (train['Outlier'] == Fals
                                                      test_size=0.1)


dtrain = xgb.DMatrix(X_train[features], y_train)
dvalid = xgb.DMatrix(X_valid[features], y_valid)

evallist = [(dtrain, 'train'), (dvalid, 'eval')]
evals_results={}
```

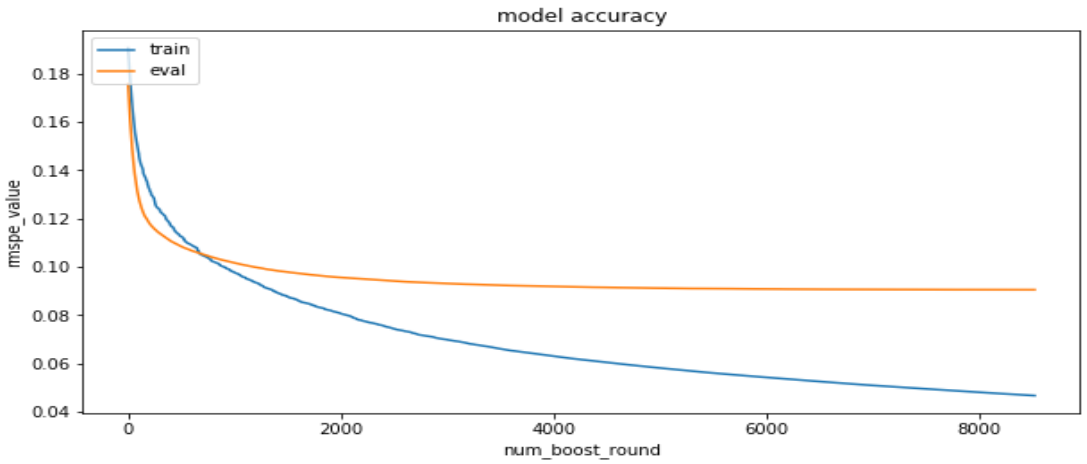
训练准备完成

参数设置为 max_depth:12， subsample:0.7， colsample_bytree:0.5 随机种子为 42.
并且在训练准备过程中，对 sales 进行取 log 处理.

```
gbm = xgb.train(params, dtrain, num_boost_round, evals=evallist,evals_result=evals_results, early_stopping_rounds=200, \
  feval=rmspe_exp, verbose_eval=50)
```

```
[8100]  train-rmse:0.048186     eval-rmse:0.081274      train-rmspe:0.048948    eval-rmspe:0.09059
[8150]  train-rmse:0.048059     eval-rmse:0.081271      train-rmspe:0.04881     eval-rmspe:0.090583
[8200]  train-rmse:0.047933     eval-rmse:0.081266      train-rmspe:0.048676    eval-rmspe:0.090576
[8250]  train-rmse:0.047793     eval-rmse:0.081268      train-rmspe:0.048523    eval-rmspe:0.090583
[8300]  train-rmse:0.047661     eval-rmse:0.081266      train-rmspe:0.048382    eval-rmspe:0.090576
[8350]  train-rmse:0.047531     eval-rmse:0.081262      train-rmspe:0.048242    eval-rmspe:0.090571
[8400]  train-rmse:0.047412     eval-rmse:0.081256      train-rmspe:0.048113    eval-rmspe:0.090564
[8450]  train-rmse:0.047281     eval-rmse:0.081255      train-rmspe:0.047972    eval-rmspe:0.090568
[8500]  train-rmse:0.047144     eval-rmse:0.081252      train-rmspe:0.047826    eval-rmspe:0.090565
[8550]  train-rmse:0.047014     eval-rmse:0.081253      train-rmspe:0.047685    eval-rmspe:0.090565
[8600]  train-rmse:0.046884     eval-rmse:0.08125       train-rmspe:0.047547    eval-rmspe:0.090558
[8650]  train-rmse:0.046766     eval-rmse:0.081248      train-rmspe:0.047422    eval-rmspe:0.090552
[8700]  train-rmse:0.046646     eval-rmse:0.081245      train-rmspe:0.047296    eval-rmspe:0.09055
[8750]  train-rmse:0.046529     eval-rmse:0.081243      train-rmspe:0.047171    eval-rmspe:0.090552
[8800]  train-rmse:0.046406     eval-rmse:0.081245      train-rmspe:0.047041    eval-rmspe:0.090558
[8850]  train-rmse:0.046283     eval-rmse:0.081244      train-rmspe:0.046907    eval-rmspe:0.090558
[8900]  train-rmse:0.046162     eval-rmse:0.081243      train-rmspe:0.04678     eval-rmspe:0.090555
Stopping. Best iteration:
[8724]  train-rmse:0.046584     eval-rmse:0.081243      train-rmspe:0.047229    eval-rmspe:0.090546
```

用 xgboost 最终数据验证集上的 rmspe 为 0.90546，early_stopping_rounds 设为 200 轮，也就是 200 只能验证集的 rmspe 没有任何变化之后，模型停止训练.

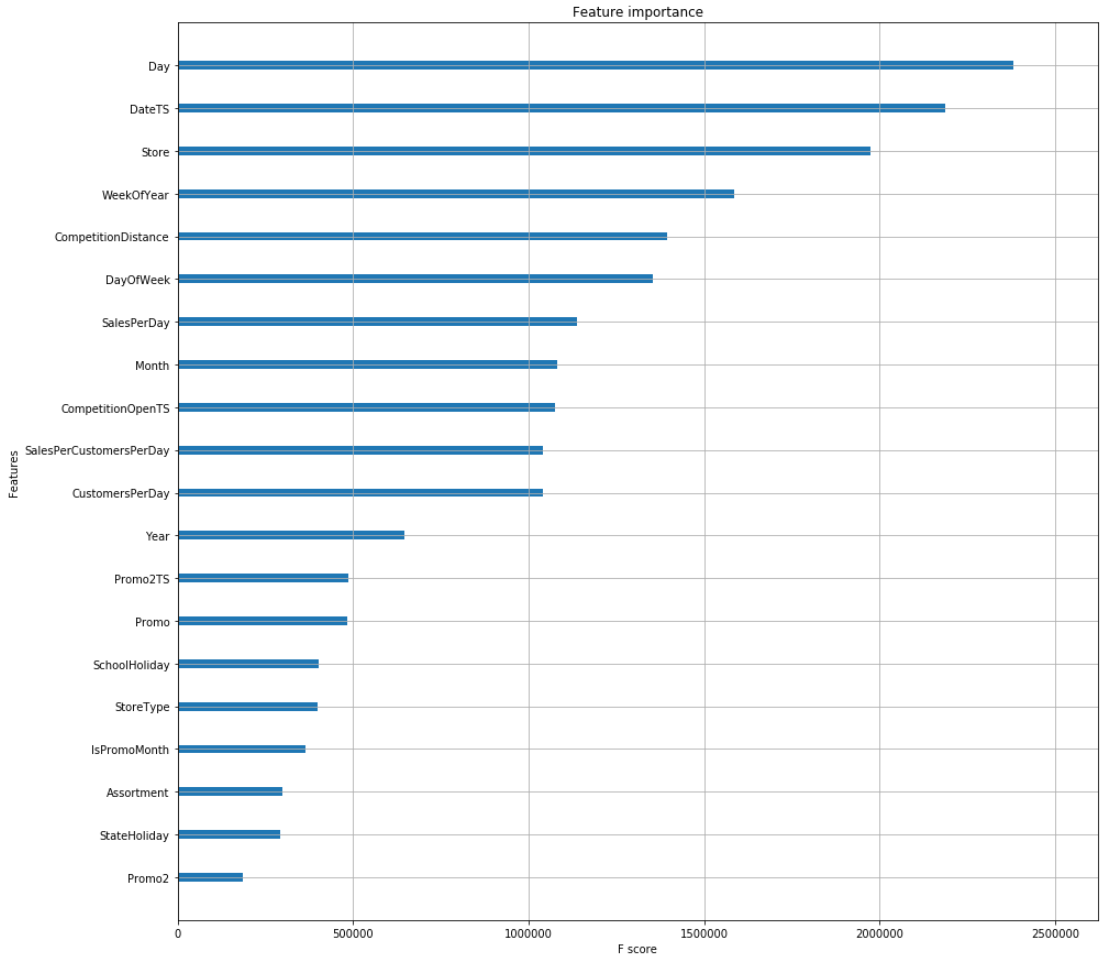


对训练过程进行可视化，上图去 400 轮以后的数据做成训练集和验证集 rmspe 变化的图像，可以发现图像的 1000 数值左右，也就是训练轮数 1400 之前，验证集的 rmspe 迅速收敛，之后，开始缓慢收敛.

```
print "生成测试集"
dtest = xgb.DMatrix(train.loc[(train['Test'] == 1)][features])
test_probs = gbm.predict(dtest)
result = pd.DataFrame({"Id": train.loc[(train['Test'] == 1)]['Id'].astype('int'), 'Sales': np.expm1(test_probs)*0.965})
result.to_csv("xgboost_submission10.csv", index=False)
```

```
生成测试集
```

对于最终结果，用 exp()恢复预测值. 在最终结果上乘以 0.965 用于调整最终提交结果. [xi]



对于 feature_importance 可以发现,最重要的 feature 是 day,其次是 DateTS 也就是 date 的时间戳,接着是 store 编号,然后是 WeekofYear.

3.3 改进

max_depth:12 由原先的 10 增大到 12, subsample: 由原先的 0.3 增大到 0.7,为了控制欠拟合,colsample_bytree:1,减小到 0.5,因为,colsample_bytree 也就是控制使用的特征数量,太大反而对结果不太理想,还有一点就是将数据原本缺省值,保留到了训练集当中,最终分数 0.02 左右的提高
在此有使用过自定义的 cv 但是太花费时间,最终放弃使用交叉验证.

```
np.exp(np.log(5000))
```

```
5000.0000000000036
```

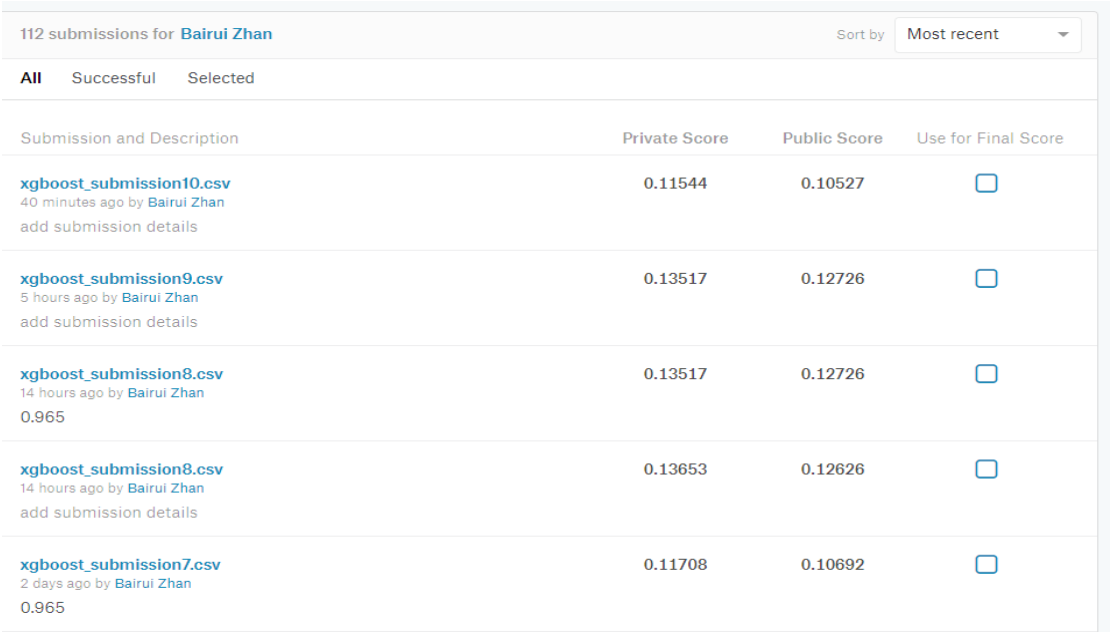
```
np.expm1(np.log1p(5000))
```

```
5000.0000000000027
```

本身数据训练时使用的由原先的 sales 变为了 log1p(sales),然后对于测试集用 expm1(pred)还原函数,精度大于去 log 然后用 exp 恢复.[xii]

4 结果

4.1 模型评估与验证



最终结果对预测结果进行了一个缩放，也就是所有数据乘以 0.965，最终结果为 0.11544 低于 0.11733，之前对缺省值没有很好的做保留，有部分缺省值变成了-1，所以当 缺省值完整保留下来后，xgboost 很好的处理了缺省值.将最终模型分数提高.

5 结论

5.1 思考

在单一模型的情况下，xgboost 效果理想，对于原始数据的处理，效果已经接近阈值，在对数据进行剔除异常值，和进行一部分数据处理后，效果就打到了阈值.

5.2 改进

通过融合多个模型会比阈值更好，较为简单的 ensemble 可以是随机种子，然后训练并预测 10 次，然后取平均值.或者可以采用 10 个 Entity Embedding+Neural Network 在 Private Leaderboard 上得分 0.11098.[xiii].又或者，融合不同的模型，如 xgboost, random forest regressor，和 linearregression(线性回归)等差异较大的模型.或者使用 PCA 技术，主动降维，而不是使用 xgboost 自动筛选.[xiv]
在 ipynb 文件中，本人尝试观察了 sales 在不同特征中是否有规律，如果加入新的特征，如 salesPerStoreType，也就是 sales 在不同商店类型的平均销售额，应该会对模型分数有进一步提高，在此，由于临近最终通过期限，未做尝试.

参考文献

[i] https://en.wikipedia.org/wiki/Rossmann_(company)
[ii] https://www.kaggle.com/c/rossmann-store-sales#description
[iii] Udacity,机器学习(进阶)-课程 12 评估指标-分类指标与回归指标
[iv] http://wiki.mbalib.com/wiki/%E9%94%80%E5%94%AE%E9%A2%84%E6%B5%8B
[v] https://www.kaggle.com/c/rossmann-store-sales#evaluation
[vi] http://xgboost.readthedocs.io/en/latest/python/python_api.html
[vii] https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h.htm
[viii] https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf
[ix] http://www.a-site.cn/article/714295.html
[x] https://www.kaggle.com/c/rossmann-store-sales/leaderboard
[xi] https://kaggle2.blob.core.windows.net/forum-message-attachments/102102/3454/Rossmann_nr1_doc.pdf
[xii] https://blog.csdn.net/liyuanbhu/article/details/8544644
[xiii] https://github.com/robininin/rossmann
[xiv] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html