Rossemann销售额预测

王廷风 2017年12月12日

I. 问题的定义

项目概述

该项目源自于Kaggle 比赛Rossmann Store Sales 中,Rossmann 是欧洲一家连锁药店，我们需要根据过去各个Rossmann 药妆店每日的销售情况及相关信息情况，来预测Rossmann 未来3个月的销售额。

该问题可以看作是一个回归分析问题，每一家店面的销售额都与许多因素有关，通过各种因素建立与销量之间的回归模型，即通过各个自变量（Features）来预测因变量Y（销售额）。

本项目使用近年来备受欢迎的xgboost算法来训练模型，对原数据集进行特征提取后输入xgboost来训练模型，并使用其来预测最终的结果。

问题陈述

该问题可以看作是一个回归问题，给定的数据中含有自变量与因变量。因变量Y为商店的销售额Sales，而给定的数据中的自变量（Features）则有许多，有每个商店的日期数据以及自身相关的一些特征，如每个商店的情况（StoreTypeCompetitionDistance、Customers等）、时间相关的参数（日期、周几、假期等）

该题目要求我们用2013 年至2015 年8 月的数据来进行训练，然后给定2015 年剩余月份的feature 来让我们预测出各个商店对于的销售额。

我将通过特征工程与特征提取来选定合适的Features，并以此为基础来训练xgboost模型。于此同时xgboost模型的调参也是十分重要的。通过合适的特征与模型参数设置来训练出我所需要的xgboost模型，并以此来对未来的月份的销售额进行预测，再将预测的结果与真实值做比较来判定模型的好坏。

评价指标

该问题得到的结果为预测的销售额，因此自然销售额的预测值与真实值之间的误差是评估模型的指标。

本测试采用的误差为： Root Mean Square Percentage Error (**RMSPE**)

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

RMSPE与日常使用的方差或标准差相比，主要多了一步特征缩放(除以y_i)，因为本项目的label销售额数据相差巨大(最大值为：41551，而最小值为46)，对于如此大的差距，使用方差会造成结果数值相当巨大，而RMSPE通过特征缩放的方式，能将结果数值控制在0-1区间内。

II.分析

数据探索与可视化

本项目的测试数据集包含的数据有train.csv与store.csv。前者主要体现每天商店的日期与销售数据，而后者是每个商店自身的特征。各个特征的含义可参考此处

train.csv包含的特征有：

```
['Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open', 'Promo',
       'StateHoliday', 'SchoolHoliday']
```

其数据样例为：

`raw_train_df.head()`

	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

store.csv包含的特征有：

```
['Store', 'StoreType', 'Assortment', 'CompetitionDistance',
       'CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear', 'Promo2',
       'Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval']
```

其样例数据为：

`store_df.head()`

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	Pr
0	1	c	a	1270.0	9.0	2008.0	0	NaN	NaN	
1	2	a	a	570.0	11.0	2007.0	1	13.0	2010.0	Ja
2	3	a	a	14130.0	12.0	2006.0	1	14.0	2011.0	Ja
3	4	c	c	620.0	9.0	2009.0	0	NaN	NaN	
4	5	a	a	29910.0	4.0	2015.0	0	NaN	NaN	

可以看出原数据集中有着许多的异常值，例如train.csv中不开门的店`train_df[train_df.Open==0]`没有销售额，而store.csv中的缺失数据有：

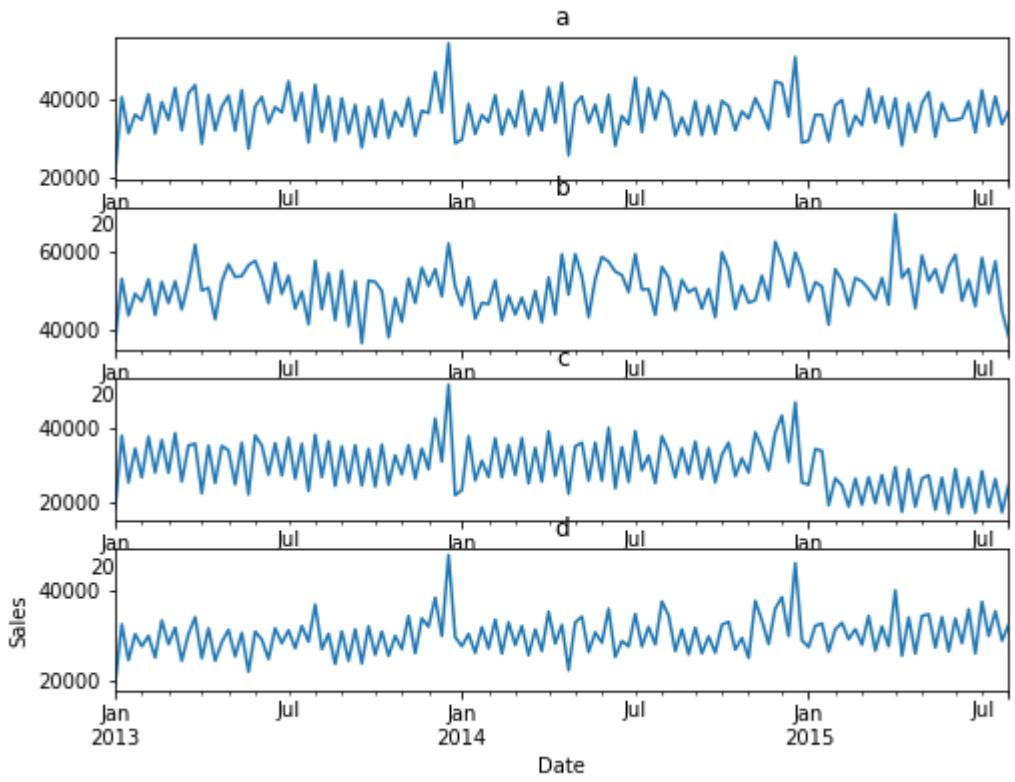
`store_df.isnull().sum()`

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

在数据预处理部分我将对异常值进行处理。

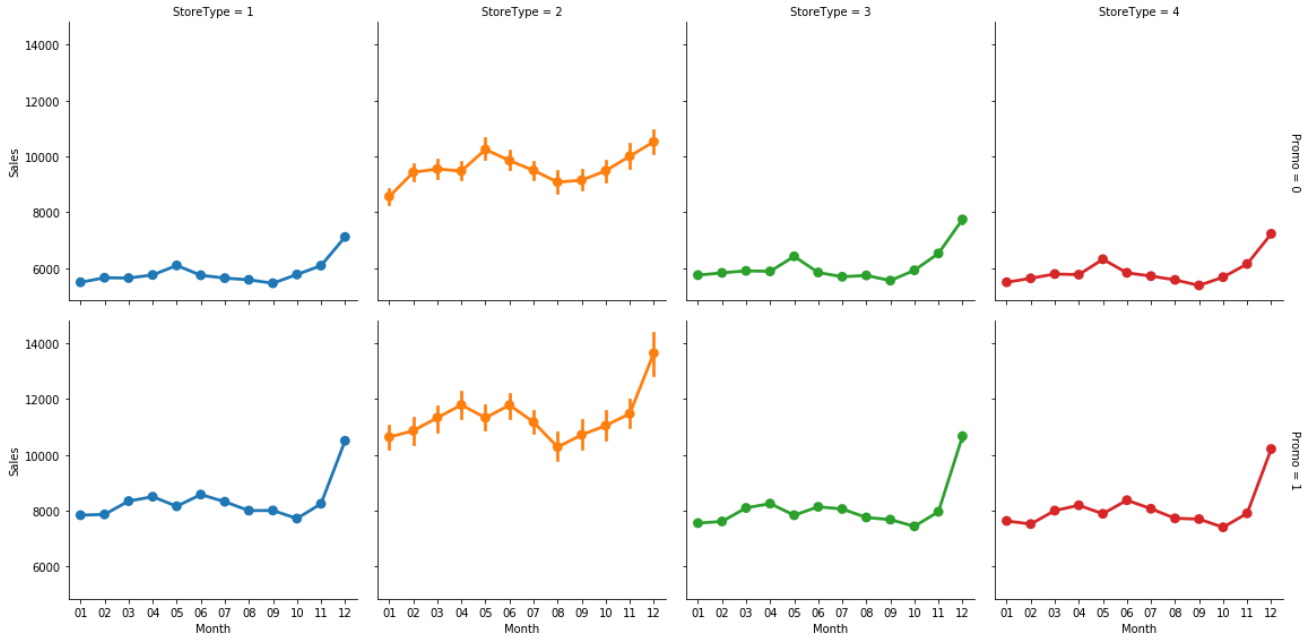
数据集共含有1,000,000条左右的数据，20多个特征，因此为了更好地进行预测，对于特征的选取与从中提取出新的有用特征十分的重要。

我先查看日期与销售额之间的关系，我在4种StoreType各任取一加店铺，查看销售额随日期的变化情况：



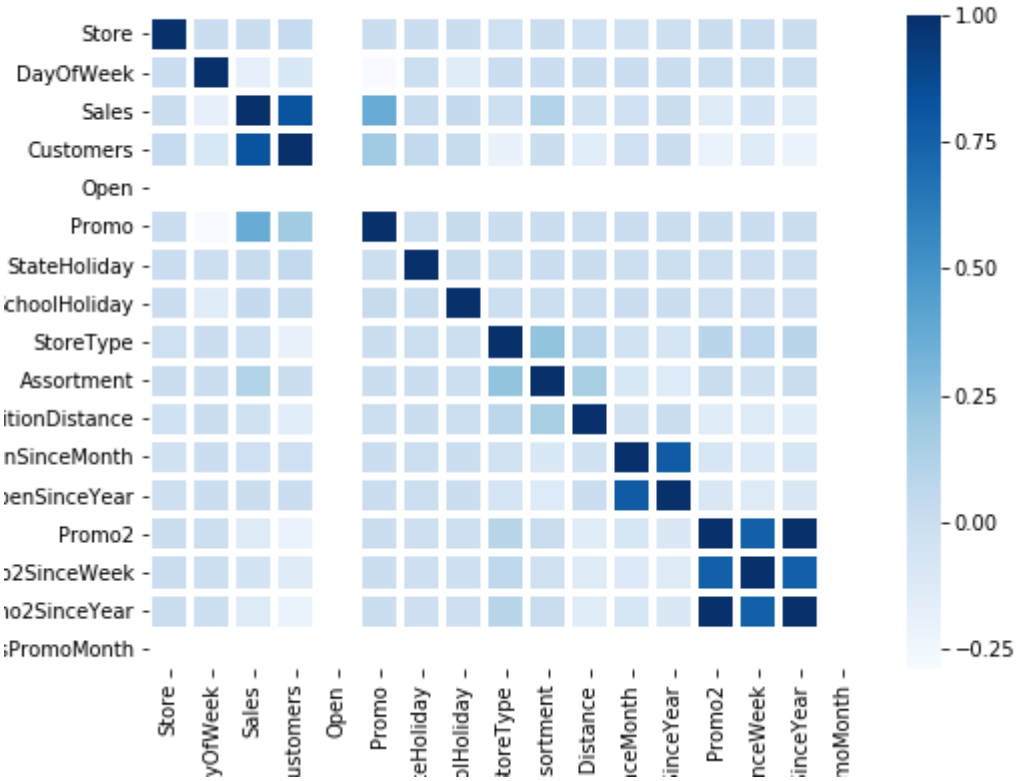
可以看出，Sales与日期之间是有着相关性的，Sales随日期的变化而大致呈现出某种趋势，例如12月份销售额为全年最高(圣诞大促销)。因此我认为日期方面的特征对模型的训练是十分重要的。

而对于促销Promo与商店类型StoreType特征，



可以看出不同商店类型间每年的销售情况存在着差异，并且当有促销时（Promo=1）销售额发生了显著的增长。

对于总体的特征情况，有：



可以看出一些特征间存在着一定的相关性，如Sales与Customers，Promo2、Promo2SinceYear与Promo2SinceMonth,Promo2SinceWeek。因此有着进一步特征提取的空间，如Sales与Customers的相关性很大，因此可以在训练时只使用Sales相关特征而不适用Customers的。

算法和技术

本项目中我将使用xgboost算法。本项目是一个监督学习的项目，同时也可以看作是一个无约束的线性回归问题，而xgboost在线性回归问题方面有着很好的表现，也是解决该类问题的一个十分受欢迎的算法，故选择xgboost算法。

xgboost对输入使用每一个特征来进行划分得到一个树，我将使用默认树模型来对其进行操作`booster:"gbtree"`. 因为该问题是线性回归问题，所以目标函数使用线性回归`objective:"reg:linear"`。模型设置好后，在训练每个训练循环时模型会遍历所有特征，找出分割数据后能得到最大gain值的特征，再在分割后的每个叶子结点上实行同样的分割步骤，直至gain值为负或达到了设定的`max_depth`，最终得到一个树即$f_i(x)$，而最终得到的预测的因变量Sales值为各个树模型对应叶子结点的加权（eta）总和$y = \sum_{i=0}^{T} \eta * f_i(x)$.

对于该项目的实现，我将使用numpy与pandas来讲输入数据转化成dataframe，使用pyplot与seaborn进行可视化分析，使用xgboost来建立模型，并对得到的dataframe使用sklearn来进行数据集的切分，得到的训练数据传入xgboost模型来进行训练，并使用训练过后的模型来对给定的测试集做出预测，得出预测结果。

xgboost的具体参数与设置可见下“执行过程”部分。

基准模型

在该项目对应的kaggle竞赛上，冠军也使用了xgboost模型，其rmspe的得分为0.100。而本次竞赛前10%的成绩为0.11773，因此我认为rmspe的得分小于0.11773的模型就可以认为是一个优秀的模型。

III. 方法

数据预处理

我首先进行了的是异常值的处理。原数据集中仍有着许多的异常值，例如train.csv中不开门的店没有销售额，因此我将Open=0的数据进行了删除。而store.csv中的缺失数据有：

`store_df.isnull().sum()`

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

CompetitionDistance仅有3个为null，因此我认为它是缺失值，进行赋平均值的fillna处理。对其余的进行赋0值的fillna处理。

再将train与store数据进行整合，形成初步的train数据集。

`train_df = pd.merge(train_df, store_df, on='Store', how='outer')`

为了使模型能够更好的使用输出特征，我对类别型的特征如`StoreType, Assortment, StateHoliday`进行独热编码。

下面需要进行的工作是对特征的进一步选择与提取，我进行的特征提取有：

1.拆分日期数据，将`Date`特征拆分为`Year, Month, Day`三个特征。

2.进一步进行日期拆分，得到特征`DayOfYear`

3.`PromoInterval`特征表示该商店进行促销季的月份，因此我引入一个新的特征`IsPromoMonth`表示本月 是否在进行促销，并删除掉`PromoInterval`

4.对于`'CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear'`,我将其合并为表示竞争对手开店总月数的特征`'CompetitionOpenMonth'`，并删除上述2特征。

5.对于`'Promo2SinceWeek', 'Promo2SinceYear'`,我将其合并为表示商店参与促销季(Promo2)总月数的特征`'Promo2OpenMonth'`，并删除上述2特征。

6.对于每个商店的有关于销售额的数据，我引入了特征：

```
"MeanLogSalesByStore","MeanLogSalesByStateHoliday","MeanLogSalesByStore_DayOfWeek",
"MeanLogSalesByStore_Promo"
```

因为销售额的数据数值自身与跨度十分的大，因此我对销售额数据采用了**log**的处理，并最终选取了以上与销售额有关的特征。

执行过程

首先对数据进行分割，一开始因为该项目数据与日期相关，因此我采用训练集中后六周的数据来作为测试集：

```
x_valid = train_df.loc[(train_df.Year==2015)&(train_df.WeekOfYear>=26)]
y_valid = np.log1p(x_valid.Sales)
x_train = train_df.drop(x_valid.index)
y_train = np.log1p(x_train.Sales)
```

但之后训练出的模型得分为0.12030，不是最优的模型，因此我最终采用了随机分割的方式。

```
x_train, x_valid, y_train, y_valid = train_test_split(train_df, np.log1p(valid_df), test_size =
0.025, random_state = 66, shuffle = True)
```

我最终采用了随机分割的方式，是因为该数据间存在时间关系，最终预测时也是基于一段连续的时间数据来进行预测，因此让模型充分学习数据间的时间关系是十分必要的。train.csv中包含了一年8个月的数据，横跨了一整年，自然不会有时间信息的泄露问题，因此最终我采用随机分割的方式，让训练集train_df尽可能的包含了一年中的各个数据，以便模型更好地学习一年中所有日期与数据的联系，让模型在预测测试集中连续几个月的数据集时能获得更好的成绩。另一方面，虽然数据集中包含了各个商店的数据，在混洗中可能会造成商店数据的不均匀，但因为测试数据集十分庞大（80万+），根据大数定律，随机混洗后各商店间的数据比例会十分接近原有的占比，因此我认为在上述代码中不必特意使用`stratify`参数来控制各数据的比例。

经过查看该项目的discussion我发现将因变量Sales采用log操作后会取得比原来更好的结果，经过本地实际测试后也是如此，便将Sales进行了log操作`np.log1p(valid_df)`,采用log1p：$log(x+1)$防止出现负值。

再开始建立一个xgboost模型，因为训练一个模型的时间非常耗时，同时特征选取也有许多的工作要进行，因此我并未使用GridSearch来进行参数选取，而是经过几轮测试最终选定了如下的参数：

```
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.03,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.3,
          "silent": 1,
          "seed": 66
          }
num_boost_round = 4000
```

我使用默认树模型来对其进行操作`booster:"gbtree"`. 因为该问题是线性回归问题，所以目标函数使用线性回归`objective:"reg:linear"`。

为了防止过拟合，我选择设置常用的防止过拟合参数例如`eta`，即最终模型中的η 为默认值0.03、数的最大深度`max_depth`为10、而`subsample, colsample_bytree`则是训练数时使用的样本比例与特征的比例。经过不断测试与训练，我选择了如上的参数设置。

而后进行模型的训练

```
watchlist = [(dtrain, 'train'), (dvalid, 'eval')]
gbm = xgb.train(params, dtrain, num_boost_round, evals=watchlist, \
  early_stopping_rounds=100, feval=rmspe__xg, verbose_eval=True)
```

设置`evals=watchlist`使得我能在训练时查看到验证集的得分情况。因为该项目采用rmspe得分，因此我自己写了一个评估函数`feval=rmspe__xg`:

```
def rmspe_xg(preds, dtrain):
    labels = np.expm1(dtrain.get_label())
    preds = np.expm1(preds)
    score = np.sqrt(np.mean([((y_true-y_pred)/y_true)**2 for (y_pred, y_true) in zip(preds,
                        labels) if y_true != 0]))
    return 'RMSPE', score
```

模型输入的数据dtrain为：`dtrain = xgb.DMatrix(x_train[features], y_train)`

对于特征的选取也花费了我一定的时间，一开始为了防止过拟合我选取了特征集合的一些子集来送入模型进行训练，但效果都不竟如人意，因此我最终决定选用所有的特征来进行训练，即：

```
features = ['Store', 'DayOfWeek',  'Promo', 'StateHoliday', 'SchoolHoliday',
       'StoreType', 'Assortment', 'CompetitionDistance',
        'Promo2', 'Month', 'Day','Year','WeekOfYear','DayOfYear',
       'IsPromoMonth', 'CompetitionOpenMonth', 'Promo2OpenMonth',
        "MeanLogSalesByStore", "MeanLogSalesByStateHoliday",
        "MeanLogSalesByStore_DayOfWeek","MeanLogSalesByStore_Promo"]
```

最后，便是对模型进行训练与保存了。

```
print("Train a XGBoost model")
dtrain = xgb.DMatrix(x_train[features], y_train)
dvalid = xgb.DMatrix(x_valid[features], y_valid)

watchlist = [(dtrain, 'train'), (dvalid, 'eval')]
gbm = xgb.train(params, dtrain, num_boost_round, evals=watchlist, \
  early_stopping_rounds=500, feval=rmspe_xg, verbose_eval=True)

#保存模型
gbm.save_model('xgboost_15_with_all_features_moretrain.model')
```

完善

在最初的模型中，为了进行参数选取，我对参数的选取倾向于了更少的训练轮数，即：

```
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.3,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.85,
          "silent": 1,
          "seed": 66
          }
num_boost_round = 300
```

并在一开始没有引入与销售额有关的变量，此时模型得分在0.13-0.15左右，较好的模型得分为：

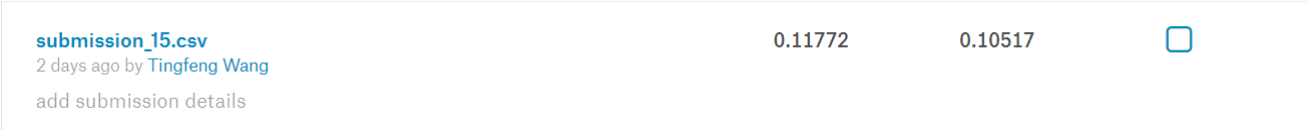
可以看到离10%仍有一段距离，在之后的调试过程中，我在该竞赛的Disussion上收获到了许多的思路与灵感，特别是此处的思路分享让我收获颇丰，其中对我帮助最大的是对销售数据的引入。经过对数据集的观察之后，我引入了几个与销售额有关的特征：

```
"MeanLogSalesByStore","MeanLogSalesByStateHoliday","MeanLogSalesByStore_DayOfWeek",
"MeanLogSalesByStore_Promo"
```

经过初步训练后发现效果有所提升，便开始对其进行最终的训练。使用更多的训练轮数，为了防止过拟合采用更小的eta，因为采用了全部的特征，特征选取的比例也设为更低，最终模型参数为：

```
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.03,
          "max_depth": 10,
          "subsample": 0.9,
          "colsample_bytree": 0.3,
          "silent": 1,
          "seed": 66
          }
num_boost_round = 4000
```

经过4000轮的训练过后，模型最终的得分为：



达到了10%的既定目标，可以认为是一个比较有效的模型。

IV. 结果

模型的评价与验证

该模型最终在kaggle上的得分达到了10%的既定目标，但离排名更靠前的模型仍有着一定的差距。比如说特征的选取，我的特征提取工作其实仍相当的初步，比如按月度、季度的特征分类，商店不同时间间隔的销售趋势，前列的模型中有的还添加了天气相关的数据。与之相比我的特征提取还是远远不够的。其次是模型的调整，许多参赛选手采用了多个模型进行聚合来预测的思路。我实现了一个以4个商店类别来分别训练的4个模型，与原先得到的模型进行加权聚合之后得到的结果不如原有的单一模型，因此我最终使用了单一的模型来进行预测。

对于模型的可靠性，在模型训练过程中，验证集的得分也十分优秀，达到了0.096783.

```
[3999]  train-rmse:0.067509 eval-rmse:0.084618  train-RMSPE:0.072135    eval-RMSPE:0.096783
```

而在最终提交在2015年后几个月的连续数据上训练时也取得了0.11772的得分，因此我认为模型能较好的对数据进行预测。

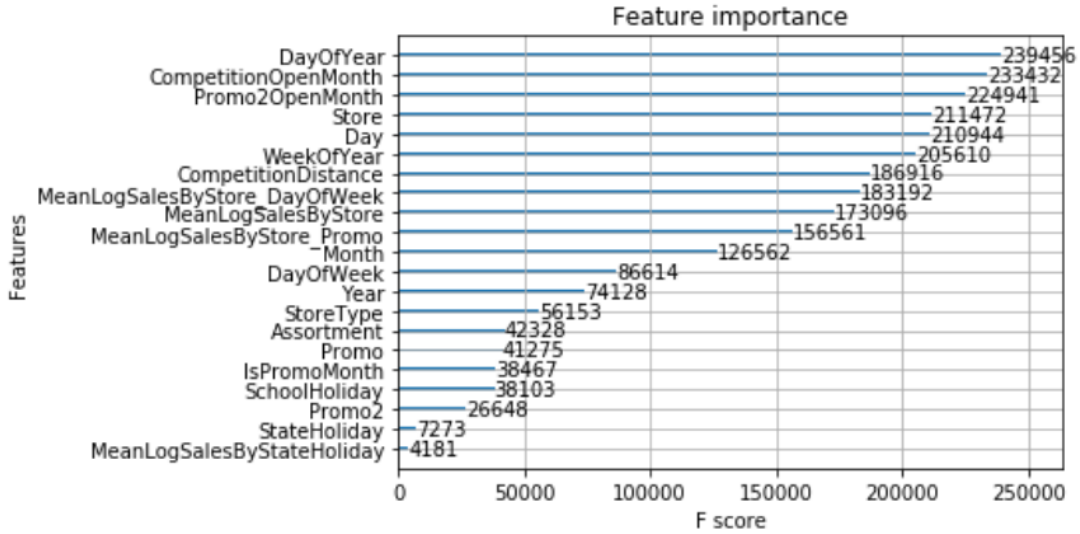
合理性分析

我所的的最终模型达到了我初步的10%的基准模型的得分。而对于我模型的合理性，得益于xgboost模型的优点与决策数的结构，对于数据中的缺失值与极端值都能够通过分叉来避免对模型造成极端的影响，而树形的结构能够挖掘特征中隐藏的联系并得出最好的特征划分，这些都保证了模型的合理性。而在kaggle的private得分（61%的测试数据）为0.11772。我认为模型对问题做出了比较有效的预测。

V. 项目结论

结果可视化

使用xgboost模型的一大好处便是模型决定划分时的依据是使gain值最大的特征，得分越高的特征对于模型划分来说就越重要，因此模型也可以反映出特征的重要情况，通过下图我们就能看出我所选取的各个特征在模型中的重要性情况。



可以看出，因为Sales与日期间有着很大的相关关系，因此特征DayOfYear无疑是最重要的，而其他日期特征如Day、WeekOfYear等也十分重要。而其他的比如竞争对手开业月数，竞争对手距离这些特征也反映了商店面临的竞争，自然对销售额影响比较重要，其他的特征也都有着各自对Sales的影响。

对项目的思考

我在这个项目的流程是一个比较典型的kaggle竞赛流程。

1.数据清理：处理数据中的缺失值，并合并出一整个的训练集

2.数据探索与可视化处理：初步探索所得的数据集，查看其均值等，并通过可视化探索初步确定下一步特征选取的方向。

3.特征选择与提取：在原有特征的基础上提炼出所需的新的特征，并进行选择哪些特征要送入模型进行训练。

4.建立模型与训练模型：将原数据集划分为训练集与测试集，使用训练集与选取的特征送入模型进行训练。

5.通过验证集评估模型与改进：训练后得到的模型在验证集上进行验证，判断模型的好坏，若不竟如人意则继续重复3，4步来得到新的模型。

回望整个训练过程，该项目让我体会到了特征工程的重要性，可以说本项目最难的地方也是我花费时间与精力最多的地方就是这个特征工程，原数据集中大约只有20个左右的特征，相比与近100万的样本来说是微不足道的。但这些特征十分的粗糙，并不能很好的反映原数据集的情况，这时候我就需要进行特征工程。将数据进行拆分与提取，比如将日期特征拆分为更详尽的数据如月日、将竞争对手开店的时间转化为开店的时长等，联系多个特征提取出新的数据比如本月是否有促销季等。相比较之下xgboost模型的调参工作反而花费的时间没那么多。

虽然最终的模型仍不够好（排名为10%）但其rmspe得分为0.11772，仍比较符合我对这个问题的期望，在相似场景下依然能够解决这类型的问题。

需要作出的改进

该项目的特征工程仍可以做出许多新的花样，比如更多不同分类情况下的销售额，更多的时间类型的划分。但由于我在笔记本上训练的时间较长，没能充分尝试更多的特征，不得不说也是个遗憾。其他的例如训练出不同的模型来进行聚合也是排名前列的选手使用的方法，我尝试了一些组合但结果都不甚理想。因此最终得到了现在使用的模型。

我的模型得分在10%，仍有很大的提升空间，我认为在多进行些特征工程与不同特征、不同类别下训练出的模型的聚合最终能得到比现在还要更好的模型。