Rossmann 销售额预测

机器学习工程师纳米学位毕业项目

张立芹　2018 年 07 月 05 日

1. 定义

1.1. 项目概览

人们常常通过自己经历的经验来预测结果，但是很难量化去得到预测结果。如今在机器学习研究中，可以通过数学的形式，对数据进行训练，从而量化的方式得到预测结果。在机器学习研究中，分为监督学习与非监督学习两大类。监督学习是通过历史的数据学习，该数据有特征信息（输入信息）和目标信息（预测信息），通过分析得出输入与输出之间的关系搭建模型，然后通过模型输入的特性信息进行预测。非监督学习是对特征信息进行聚类分析，但没有目标结果。目前在金融领域已经广泛运用，所以监督学习实用价值会越来越高。

我选择的毕业项目是 Rossmann 销售预测。Rossmann 是德国的一家最大的日化日用品超市，拥有 3000 多家实体店和网店，并且很受中国游客、商务人士和留学生购买。每家商店的经理需要预测未来 6 周的日销售情况。每个经理根据自己的经验等依据预测销售情况但是结果的准确率不高。根据 Rossmann 商店的信息，比如促销、竞争对手、节假日、季节、气候、地域等因素，来预测 Rossmann 未来 6 周每天的销售额。提供可靠的预售销售额，可以有效管理商品，免受商品积压，带来的损失，同时也提供经理更加预售情况合理安排人员。

1.2. 问题说明

Rossmann 是德国最大的日化日用品超市，拥有 3000 多家实体店和网店，并且很受中国游客、商务人士和留学生购买。每家商店的经理需要预测未来 6 周的日销售情况。每个经理根据自己的经验等依据预测销售情况但是结果的准确率不高。根据 Rossmann 商店的信息，比如促销、竞争对手、节假日、季节、气候、地域等因素，来预测 Rossmann 未来 6 周每天的销售额。提供可靠的预售销售额，可以有效管理商品，免受商品积压，带来的损失，同时也提供经理更加预售情况合理安排人员，所以提供给他们一个预测模型是非常有需要的。使

用 1115 家门店的历史销售数据情况研究，分析后的数据进行创建模型，用于预测六周后的日销售情况。

对于有特征-标签模式的数据时，通过机器学习可以转化为简单数学问题，这类问题通过对大量数据的训练。而该项目提供了大量数据，所以可以根据该数据进行监督学习，得到预测结果。

在机器学习中，可以通过决策树回归算法、SVM 回归算法等等可以解决回归问题，然后如今通过这些基础算法，有随机森马算法、GBDT 这些性能优越的算法。如今 XGboost 被广泛运用。

首先对数据进行处理与整合，从而得到可以让机器学习训练的数据集，然后使用随机森林与 XGboost 算法搭建回归模型，最终预测出 1115 家门店的 41088 条未来的销售额。

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。

1.3. 指标

一般情况下，会使用数据集中的一部分作为训练集，其中另一部分为测试集。对于二分类问题，可以使用准确率、对数损失函数、ks、AUC 等评分方法。但是对回归的数据预测问题，使用平方根误差（RMSE）或者均方根百分比误差指标衡量模型效果，同时也可以使用召回率。由于 RMSPE 对数值的绝对值大小不敏感，更加适合作为评价模型的指标。

使用 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。其公式如下：

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

y_i 表示真实的销售数据，$\hat{y}_i$ 是预测数据，销售为零的数据不进行统计。该函数可以解决线性回归问题，预测结果是具体数值向量，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。

2. 分析

2.1. 数据研究

通过下载数据与 Kaggle 提供的信息，该信息与 csv 格式提供，其中包含训练与预测的数据集。输入数据集详情如下：

- *输入数据集*
  train.csv 表示过去的销售数据为训练集。包含字段：
  Store,DayOfWeek,Date,Sales,Customers,Open,Promo,StateHoliday,SchoolHoliday
- test.csv 表示测试数据。包含字段：
  Id,Store,DayOfWeek,Date,Open,Promo,StateHoliday,SchoolHoliday
- sample_submission.csv 预测数据格式样本。包含字段：Id,Sales

- store.csv 表示门店信息。包含字段：
  Store,StoreType,Assortment,CompetitionDistance,CompetitionOpenSinceMonth，CompetitionOpenSinceYear,Promo2,Promo2SinceWeek,Promo2SinceYear,PromoInterval
- *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。
- 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。
- 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。
- 以上每个数据集的统计情况如下表

训练数据特征统计数据表

	Store	DayOfWee	Sales	Customer	Open	Promo	SchoolHoliday
count	1.02E+06	1.02E+06	1.02E+06	1.02E+06	1.02E+06	1.02E+06	1.02E+06
mean	5.58E+02	4.00E+00	5.77E+03	6.33E+02	8.30E-01	3.82E-01	1.79E-01
std	3.22E+02	2.00E+00	3.85E+03	4.64E+02	3.76E-01	4.86E-01	3.83E-01
min	1.00E+00	1.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
25%	2.80E+02	2.00E+00	3.73E+03	4.05E+02	1.00E+00	0.00E+00	0.00E+00
50%	5.58E+02	4.00E+00	5.74E+03	6.09E+02	1.00E+00	0.00E+00	0.00E+00
75%	8.38E+02	6.00E+00	7.86E+03	8.37E+02	1.00E+00	1.00E+00	0.00E+00
max	1.12E+03	7.00E+00	4.16E+04	7.39E+03	1.00E+00	1.00E+00	1.00E+00

测试数据特征统计数据表

	Id	Store	DayOfWeek	Open	Promo	SchoolHoliday
count	41088	41088	41088	41077	41088	41088
mean	20544.5	555.899533	3.979167	0.854322	0.395833	0.443487
std	11861.22827	320.274496	2.015481	0.352787	0.489035	0.496802
min	1	1	1	0	0	0
25%	10272.75	279.75	2	NaN	0	0
50%	20544.5	553.5	4	NaN	0	0
75%	30816.25	832.25	6	NaN	1	1
max	41088	1115	7	1	1	1

门店特征计数据表

	Store	CompetitionDistance	Compet	CompetitionOpenSinceYear	Promo2	Promo2	Promo2SinceYear
count	1115	1112	761	761	1115	571	571
mean	558	5404.901079	7.224704	2008.668857	0.512108	23.59545	2011.763573
std	322.01708	7663.17472	3.212348	6.195983	0.500078	14.14198	1.674935
min	1	20	1	1900	0	1	2009
25%	279.5	NaN	NaN	NaN	0	NaN	NaN
50%	558	NaN	NaN	NaN	1	NaN	NaN
75%	836.5	NaN	NaN	NaN	1	NaN	NaN
max	1115	75860	12	2015	1	50	2015

通过对数据统计，可以得出如下信息：

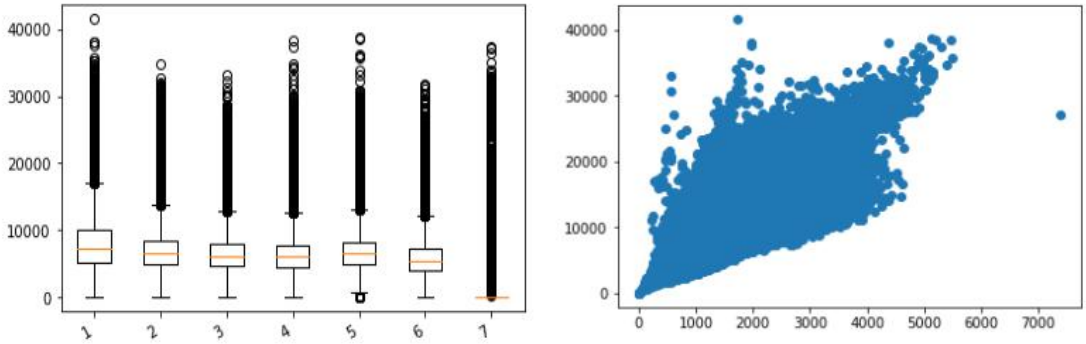
- 训练集样本有 1017209 个样本，测试数据有 41088 个样本。其中数据集没有缺失值，测试集 Open 字段数据有少量缺失，对缺失的数据进行观察，发现其当前日期、节假期情况可以判断该缺失部分都是 Open 为 1 的情况，所以对其进行补充。

	Id	Store	DayOfWe	Date	Open	Promo	StateHoliday	SchoolHoliday
479	480	622	4	2015/9/17	NaN	1	0	0
1335	1336	622	3	2015/9/16	NaN	1	0	0
2191	2192	622	2	2015/9/15	NaN	1	0	0
3047	3048	622	1	2015/9/14	NaN	1	0	0
4759	4760	622	6	2015/9/12	NaN	0	0	0
5615	5616	622	5	2015/9/11	NaN	0	0	0
6471	6472	622	4	2015/9/10	NaN	0	0	0
7327	7328	622	3	2015/9/9	NaN	0	0	0
8183	8184	622	2	2015/9/8	NaN	0	0	0
9039	9040	622	1	2015/9/7	NaN	0	0	0
10751	10752	622	6	2015/9/5	NaN	0	0	0

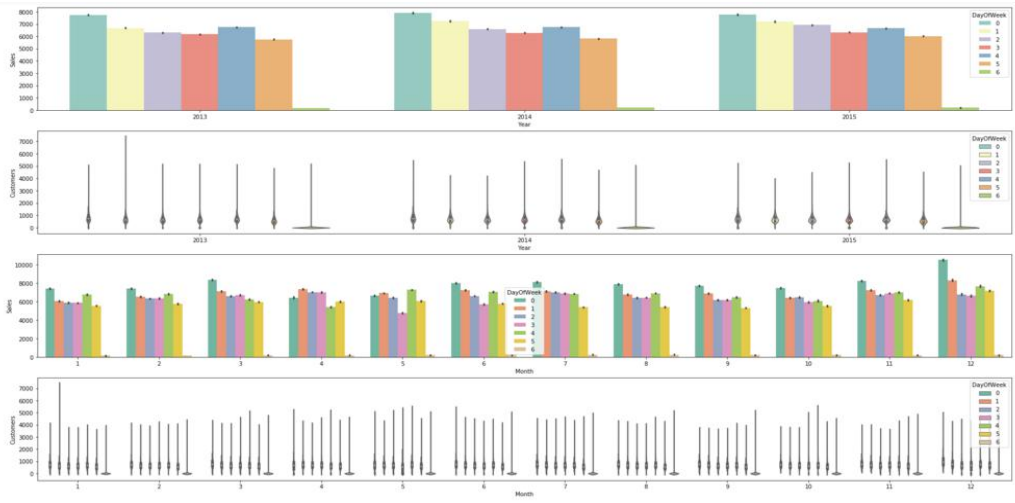
- 样本日期：year-month-day 字符串，需要进行拆分编码，明确样本的年、月、日。并对年进行独热编码。
- 门店类型：a,b,c,d 表征四种类型的门店，需要对该字段进行独热编码。
- 品类规模：使用 a,b,c 分别表示基础型、大型和扩充型的店铺，也使用独热编码。
- 对门店的缺失值，进行补充，使用 0 补充。

2.2. 探索性可视化

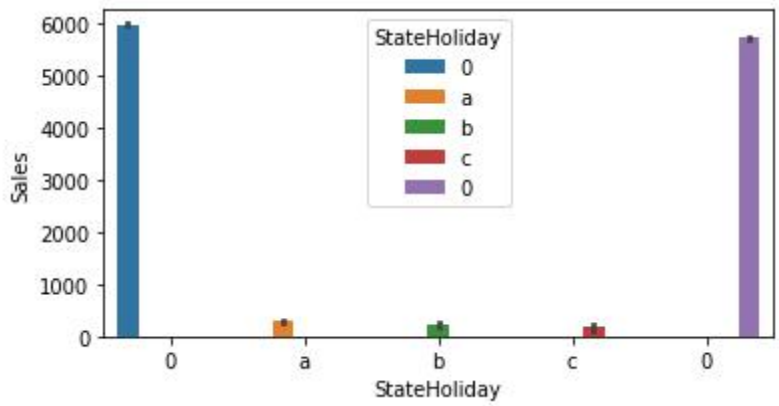
为了对数据进行直观的认识，对数据进行可视化。我通过对训练集的周几特征与销量的分布进行箱图、客户量与销售额的相关性进行可视化分析。结果如下图：



下图是每年与每月对应每周的七天对应的所有销售额的柱状图与其琴箱图。可以明显发现，周日门店都是关闭状态。



节假日对销售额的影响，从下图可以很明显观察出，法定节假日和销售额的柱状图。



2.3. 算法与方法

针对该项目，我选择随机森林与 XGboost 算法，这两种算法可以高效展开预测。在数据研究与数据探索的可视化，有的特征变量需要处理，对异常值个人不进行处理，因为该数据比较真实，同时对于特征缺失的都进行 0 简单方式填充。

2.3.1. 随机森林算法

随机森林是包含多个决策树的分类器，其输出室友每个树输出的类别的众数决定。该算法的推出由 Leo Breiman 和 Adele Cutler。其主要步骤如下：

1. 用 N 来表示训练样本的格式，M 表示特征数目。

2. 输入特征数目 m，用于确定决策树上一个节点的决策结果；其中 m 应远小于 M。

3. 从 N 个训练用例（样本）中以有放回抽样的方式，取样 N 次，形成一个训练集（即 bootstrap 取样），并用未抽到的用例（样本）作预测，评估其误差。

4. 对于每一个节点，随机选择 m 个特征，决策树上每个节点的决定都是基于这些特征确定的。根据这 m 个特征，计算其最佳的分裂方式。

5. 每棵树都会完整成长而不会剪枝（Pruning，这有可能在建完一棵正常树状分类器后会被采用）。

我选择调参如下几个参数：

1. n_estimators：随机森林中的决策树数目，默认为 10。

2. max_features：最大特征数，可以用整数形式考虑确定数目的特征，用浮点数形式来考虑确定比例的特征，也可以使用“log2”，“sqrt”来指定一种计算考虑特征数目的方法。

3. max_depth：决策树最大深度，可以指定一个整数来限制决策树建立过程中子数的深度。

其中随机森林 RandomForest 的分类类是 RandomForestClassifier，回归类是 RandomForestRegressor，所以该项目选择随机森林的 RandomForestRegressor 算法。

2.3.2. XGboost 算法

XGBoost 是在每轮迭代中生成一棵新的回归树，并综合所有回归树的结果，使预测值越来越逼近真实值。该算法有很强的泛化能力，其核心是每棵树通过学习之前所有树的残差。与随机森林不同，随机森林采取的多数投票的结果，而 XGBoost 采取是所有结果的累加。

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。
其主要步骤如下：

1、General parameters：参数控制在提升（boosting）过程中使用哪种 booster，常用的 booster 有树模型（tree）和线性模型（linear model）。

2、Booster parameters：这取决于使用哪种 booster

3、eta 默认值为 0.3,为了防止过拟合，更新过程中用到的收缩步长。在每次提升计算之后，算法会直接获得新特征的权重。 eta 通过缩减特征的权重使提升计算过程更加保守。

4、max_depth：数的最大深度,其默认值为 6。

5、Learning Task parameters：控制学习的场景的相关参数 。

6、objective [“reg:linear”] objective [default=reg:linear]

- 定义学习任务及相应的学习目标，可选的目标函数如下：
- “reg:linear” –线性回归。
- “reg:logistic” –逻辑回归。
- “binary:logistic” –二分类的逻辑回归问题，输出为概率。
- “binary:logitraw” –二分类的逻辑回归问题，输出的结果为 wTx。
- “count:poisson” –计数问题的 poisson 回归，输出结果为 poisson 分布。
- 在 poisson 回归中，max_delta_step 的缺省值为 0.7。(used to safeguard optimization)
- “multi:softmax” –让 XGBoost 采用 softmax 目标函数处理多分类问题，同时需要设置参数 num_class（类别个数）
- “multi:softprob” –和 softmax 一样，但是输出的是 ndata * nclass 的向量，可以将该向量 reshape 成 ndata 行 nclass 列的矩阵。没行数据表示样本所属于每个类别的概率。
- “rank:pairwise” –set XGBoost to do ranking task by minimizing the pairwise loss

7、feval 设置为 RMSPE 为评价函数。

8、num_boost_round 设置 boosting 的次数。

9、early_stopping_rounds 可以提前终止程序，这样可以找到最优的迭代次数。

2.3.3.独热编码方法

独热编码即 One-Hot 编码，又称一位有效编码，其方法是使用 N 位状态寄存器来对 N 个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。其优势为：能处理肥连续值特征、在一定程度上达到扩充特征。

2.3.4.空值填充方法

对于数据集中出现的空值，常用的填充方法使用 0 值填充，在数据集较为完善的情况下，也可以根据空值特征与其他特征的关系进行相关性填充，或者根据该特征的统计结果建立相应的填充策略。

2.4. 基准测试

检验模型是否达标，需要建立基准模型，根据这个基准指标来衡量模型的合理性。通过参考抗过了比赛排名结果，个人初步定的基准为 0.25。根据项目要求 Leaderboard 的 10% 作为基准，同理可得，测试集的评分可达到 0.11773。

3. 方法

3.1. 数据预处理

1、整理 test 数据集 Open 特征缺失，通过日期判断，使用 1 进行填充

2、DATE 做数据转换，变成 Year，Month，Day，WeekOfYear

3、StateHoliday 取值 a、b、c 的为 1，否则为 0 的转换，同时也对其训练集对 DATE 与 StateHoliday 相同处理，转化后进行独热编码。

4、对 test 与 train 集与门店信息整合，处理 CompetitionDistance 使用 9999 值填充，目标说明竞争对手距离很远（表明附件没有竞争对手门店）

5、CompetitionOpenSinceMonth 使用中位数填充。

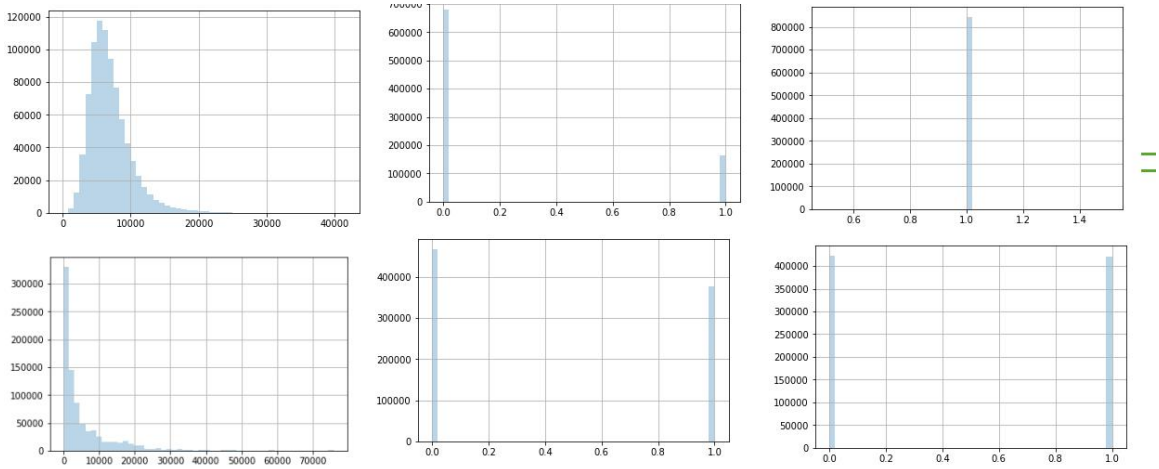
6、衍生特征 CompetitionMonths，通过销售年份与月份于竞争对手的年与月进行得出竞争对手营业几个月时间，缺失的话，使用 0 填充，表明没有竞争对手。

7、Promo2SinceWeek 使用 0 填充。

8、衍生特征 Promo2weeks，缺失值使用 0 填充

9、对 StoreType，Assortment，StateHoliday 三个特征进行独热编码。

对数据处理后，几个个特征与销售额分布分析。发现其分布属于偏正态分布。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。



3.2. 实施

算法实施基本以下面几个步骤进行：首先使用未优化的模型，也就是算法默认的参数进行对数据训练与预测，然后使用 GridSearchCV 方法进行优化模型，得出最优模型，最后得出最佳的效果。执行发现网络搜索耗时比较长，后期就改成使用手动调参。

由于预测销售是时间序列，所以划分训练集与验证集是按照时间分，其中在 2015-07-10 之前的作为训练集，2015-07-11 到 2015-07-30 为验证集,其验证集占总样本的 2.38%。

3.2.1. 未优化的随机森林

采用随机森林算法建模，开始先使用默认的参数，然后通过调整测试集的大小可以得出当测试集到达 0.1 以上，其模型效果相对稳定。

```
In  [41]: %%time
          RF_alpha = RandomForestRegressor()
          RF_alpha.fit(X_train,np.log1p(y_train))

          Wall time: 57.3 s
```

得出 RMSPE 为 0.4538。

3.2.2. 随机森林优化过程

接下来使用 GridSearchCV 方法进行参数调优，目前调试的参数有 max_features、max_depth、n_estimators。但是由于电脑配置的原因，对 n_estimators 的调参没有进行。（该部分只是的调试，后期再代码中删除该部分）

```
from sklearn.grid_search import GridSearchCV
start_time =time.time()
#使用GridSearchCV进行调优
param_grid = {'max_features': [0.6,0.8,1],'max_depth':[6,7,8]}
rf = RandomForestRegressor()
rf_grid = GridSearchCV(rf, param_grid=param_grid, cv=10, n_jobs=8, refit=True)
rf_grid.fit(X_train,y_train)
end_time = time.time()
longtime = end_time - start_time
```

```
D:\ProgramFiles\Anaconda3.4.2\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprecated in version
0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of
the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
D:\ProgramFiles\Anaconda3.4.2\lib\site-packages\sklearn\grid_search.py:42: DeprecationWarning: This module was deprecated in version 0.18 i
n favor of the model_selection module into which all the refactored classes and functions are moved. This module will be removed in 0.20.
  DeprecationWarning)
```

```
longtime
```

```
869.0467066764832
```

```
rf_grid.grid_scores_,rf_grid.best_params_, rf_grid.best_score_
```

```
([mean: 0.32248, std: 0.00334, params: {'max_features': 0.6, 'max_depth': 6},
  mean: 0.31735, std: 0.00219, params: {'max_features': 0.8, 'max_depth': 6},
  mean: 0.17625, std: 0.01397, params: {'max_features': 1, 'max_depth': 6},
  mean: 0.35266, std: 0.00573, params: {'max_features': 0.6, 'max_depth': 7},
  mean: 0.34992, std: 0.00400, params: {'max_features': 0.8, 'max_depth': 7},
  mean: 0.21153, std: 0.00937, params: {'max_features': 1, 'max_depth': 7},
  mean: 0.38800, std: 0.00355, params: {'max_features': 0.6, 'max_depth': 8},
  mean: 0.38640, std: 0.00605, params: {'max_features': 0.8, 'max_depth': 8},
  mean: 0.24068, std: 0.01373, params: {'max_features': 1, 'max_depth': 8}],
 {'max_depth': 8, 'max_features': 0.6},
 0.3879978423449633)
```

3.2.3. 初始化的 XGBoost

执行默认的 XGBoost，看看其效果。

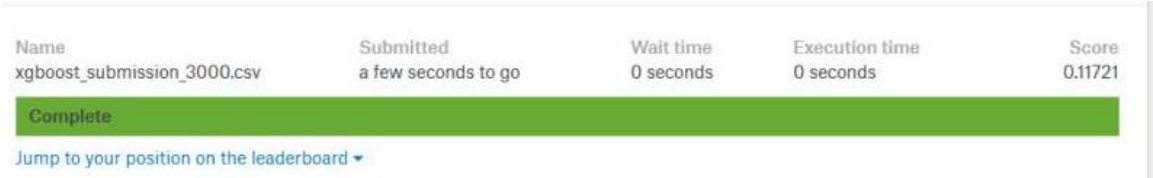
```
In [46]: dtrain = xgb.DMatrix(X_train, np.log1p(y_train))
         dvalid = xgb.DMatrix(X_val, np.log1p(y_val))
         watchlist = [(dtrain, 'train'), (dvalid, 'eval')]
         params = {"objective": "reg:linear",
                   "booster" : "gbtree",
                   "eta": 0.3,
                   "max_depth": 10,
                   "subsample": 0.9,
                   "colsample_bytree": 0.7,
                   "silent": 1,
                   "seed": 1000,
                   "learning_rate": 0.08
                   }
         num_trees = 100

In [47]: %%time
         gbm = xgb.train(params, dtrain, num_trees, evals=watchlist, early_stopping_rounds=100, feval=rmspe_xg, verbose_eval=True)
```

得到 RMSPE 为 0.2108。上传 kaggle 结果为 0.2138。

3.2.4. XGBoost 优化过程

Xgboost 算法建模，本想通过 GridSearchCV 方法进行参数调优。发现使用过程且，电脑内存不知，所以我手动调试。首先先调试 num_trees 的数目，发现调到 2000 时，结果降不下来，且在 877 时就开始收敛，上传 kaggle 结果在 0.1187。后来调

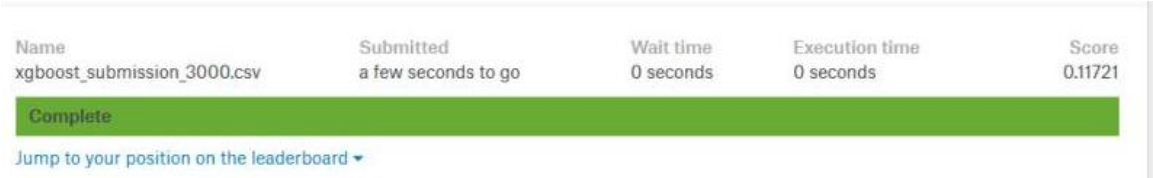
learning_rate 变低，但是分数也没有达到要求，然后对 eta 参数和 max_depth 进行调整，分别同 0.3 和 10 调整为 0.5 与 15，发现树的数据在 2000 并没有达到收敛，所以调整为 3000，可是不小心变成 30000，发现在 3328 时得到最佳。得出结果 RMSPE 结果为 0.2172，小于指定的 0.25 结果，其上传到 kaggle 结果为 0.11721，到达要求。



4. 优化与结果

对随机森林算法建模的 RMSPE 结果为 0.4538。对 max_features 与 max_depth 进行网络搜索，由于运行时间长，调试的值比较少，得到结果 0.37079907880090157。从调试中，可以发现，该最优的函数并非最佳的，所以需要把 max_depth 和 n_estimators 进行调试。

对 xgboost 的初始结果上传到 kaggle 是 0.2138，将其用上所有的训练集训练将树分别调为 300、800、2000 进行查看结果。虽然结果一步一步变好，但是调整到 2000，结果降不下来，且在 877 时就开始收敛，上传 kaggle 结果在 0.11721。后来调 learning_rate 变低，但是分数也没有达到要求，然后对 eta 参数和 max_depth 进行调整，分别同 0.3 和 10 调整为 0.5 与 15，发现树的数据在 2000 并没有达到收敛，所以调整为 3000，可是不小心变成 30000，发现在 3328 时得到最佳。得出结果 RMSPE 结果为 0.2172，小于指定的 0.25 结果，其上传到 kaggle 结果为 0.11721，到达要求。

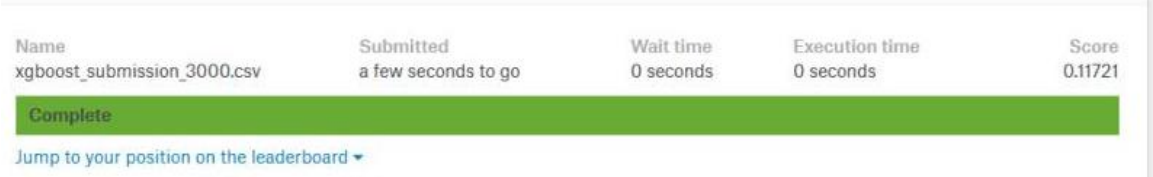


5. 结果

5.1 模型的评价与验证

随机森林建模
训练使用时间是 57.3 s
需要预测的 test
预测所用时间是 120 ms

XGBoost 建模（初始化）
训练使用时间是 1min 31s
需要预测的 test 收
预测所用时间是 184 ms
最后优化后的 XGBoost 建模
训练使用时间是 1h 6min 19s
需要预测的 test
预测所用时间是 2min 21s
预测结果上传 Kaggle 进行评分是 0.11721



5.2 合理性分析

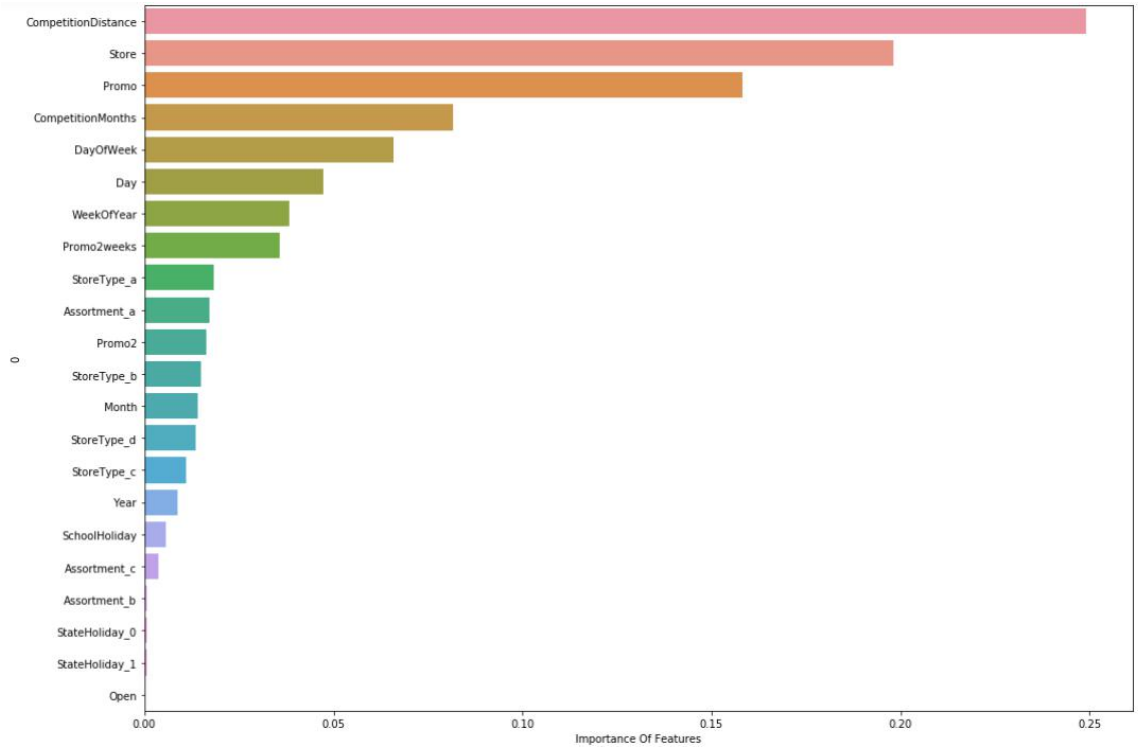
XGBoost 训练时间比较长，特别是深度增加，树的数量增加以后，学习率降低，但是模型的预测表现很好，已经满足了一开始制定的 0.25 的目标，进行初步的设置和参数的优化之后达到了 0.2172。

6.项目结论

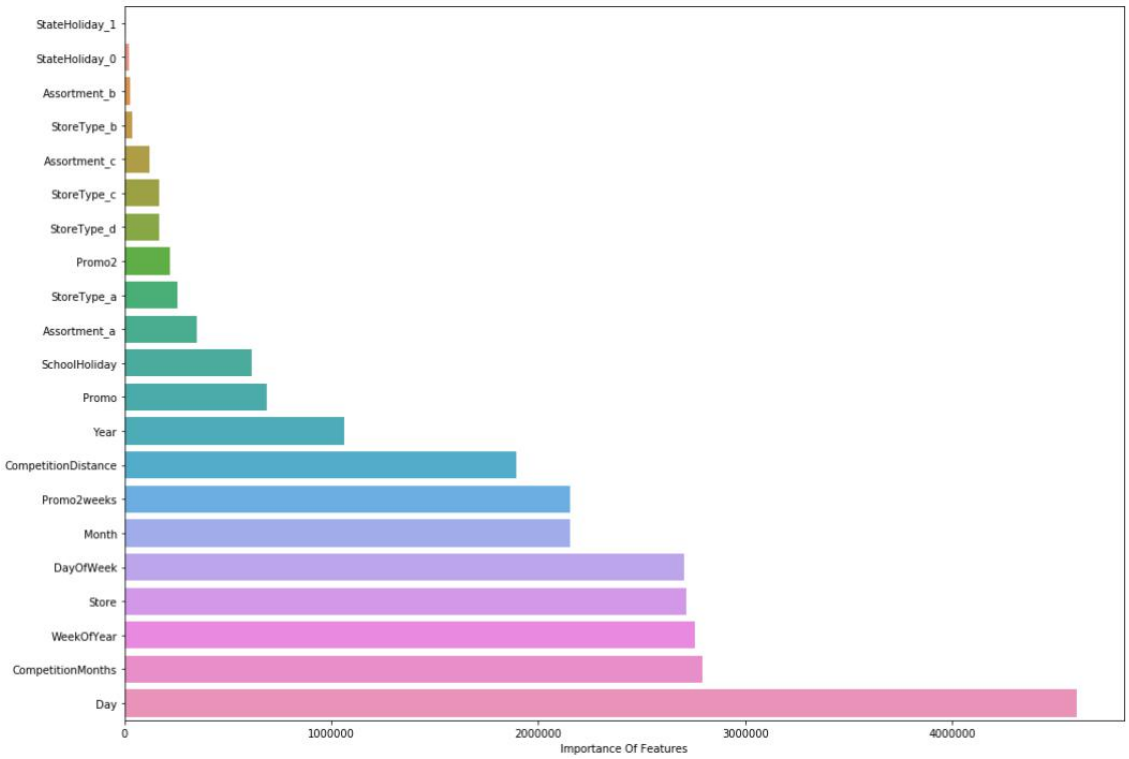
6.1 结果可视化

展现随机森林训练过程中特征重要性的排序 ，对前三个特征进行解释。

CompetitionDistance 是对手的一个重要特征，根据与竞争对手门店的距离，可以说明客户流是否受到影响，同理得出销售额受到影响。Store 门店与销售额有关，例如开业很久的门店，有一定的客源，所以对销售额有明细的影响。Promo 促销，因为促销，会促进销售量，所以销售额与促销也有相关性。



展现 Xgboost 训练过程中特征重要性的排序 ，对前三个特征进行解释。



Day 是第一个重要的特征，因为法定节假日、学校放假等等都与时间相关，所以与销售额有非常高的相关度。CompetitionDistance 是对手的一个重要特征，根据与竞争对手门店的距离，可以说明客户流是否受到影响，同理得出销售额受到影响。Dayofweek 也是与时间关联所以相关度高也正常。

6.1 对项目的思考

通过对数据的理解，数据的整理，模型的选择、模型的评估、模型的优化这几个部分。其中数据的整理是比较难的，需要对数据进行分析，是否有缺失值、是否需要转化等等来转化成监督学习可以训练的数据集。模型的选择，个人通过选择两个模型，一个随机森林，一个 XGBoost 来搭建模型，发现随机森林虽然训练时间段，但是模型收敛较慢，达不到制定目标，但是通过 XGBoost 可以比较快的达到模型效果。通过网络搜索进行调参，发现自己电脑硬件配置不佳，多个参数调优使用时间较长，所以只好改成手动，一个一个进行，最终有参数达到要求。

6.2 需要作出的改进

我目前使用的特征，其实相对比较少，我觉得如果加入更多有效的特征，可能准确性会更高，例如 Promo2 等特征。为了简便，在模型上，我个人没有对模型进行修改，只是把线下划分的训练集与验证集进行合并，使用全部的数据重新训练，得出结果进行对比,，其耗时长达 12 个小时的训练，可能和电脑配置有光吧！所以这时候，需要对模型进行改进，由于时间问题，改进工作后期继续。

参考文献

[1] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.