Rossmann 销售预测

机器学习工程师纳米学位毕业项目

李俊

2018 年 3 月 29 日

1. 定义

1.1 项目概述

本项目来源于 Rossmann 公司在 Kaggle 上发布的竞赛 Rossmann Store Sales[1]，它的目标是构建一个模型来预测商店的每日销售量。

Rossmann 是一家连锁店，它在欧洲 7 个国家拥有超过 3000 个的药店。为了更加有效地管理采购和库存，Rossmann 药店的管理者的一项重要的任务就是利用销售量的历史数据来预测未来 6 周各家分店的每日销售量。药店销售量会受到很多因素的影响，包括促销活动、竞争者、节假日和一些季节性和局部的因素，Rossmann 公司希望通过这些历史数据，训练一个稳健模型来帮助他们预测未来的每日销售量。

1.2 问题说明

这是一个回归问题，即构建一个模型 $\hat{y} = f(X)$，使得目标变量的预测值 $\hat{y}$ 与真实值 y 之间的差别最小，这里目标变量 y 是每日销售量，X为输入变量，而预测值与真实值之间的差别则由评估标准来评估。

这个问题的难点在于如何从复杂的结构数据中构建输入X，也就是特征工程。解决回归问题的常用模型有 Linear Regression, SVM, 回归树以及神经网络。

1.3 指标

本项目使用 Kaggle 竞赛中采用的指标 Root Mean Square Percentage Error (RMSPE)[2]：

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

其中，y_i 表示单个商店在某一天的销售量，而 $\hat{y_i}$ 表示对应的预测值，销售量为 0 的样本将被忽略。从上式可以看出，RMSPE 相当于对 Root Mean Sqaure Error(RMSE)进行加权，权值为目标变量真实值平方的倒数，从而也得出 RMSPE 具有非对称性，即相同的绝对误差真实值偏小的 RMSPE 要比真实值大的 RMSPE 要大。

2. 分析

2.1数据研究

Kaggle 为参赛者提供的数据集主要包含在三个文件中：train.csv, test.csv, store.csv。

- train.csv 每家商店每天的销售相关的信息，例如是否营业，促销，节假日以及当天的顾客量和销售量等，用于训练。
- test.csv 与 train.csv 相似，但是不包括当天顾客量和销售量，用于测试。
- store.csv 每家商店的额外的信息，例如商店类型，等级，竞争者信息，以及长期的促销活动等。

表 1　train.csv 中的数据示例

Sample Id	1	2	3	4	5	6
Store	13	13	13	13	13	13
DayOfWeek	5	4	3	2	1	7
Date	2015/7/31	2015/7/30	2015/7/29	2015/7/28	2015/7/27	2015/7/26
Sales	8821	7648	6648	6819	7926	0
Customers	568	474	406	446	527	0
Open	1	1	1	1	1	0
Promo	1	1	1	1	1	0
StateHoliday	0	0	0	0	0	0
SchoolHoliday	0	0	0	0	0	0

表 2　store.csv 中的数据示例

Sample Id	1	2	3	4	5	6	7
Store	1	2	3	4	5	6	7
StoreType	c	a	a	c	a	a	a
Assortment	a	a	a	c	a	a	c
CompetitionDistance	1270	570	14130	620	29910	310	24000
CompetitionOpenSinceMonth	9	11	12	9	4	12	4
CompetitionOpenSinceYear	2008	2007	2006	2009	2015	2013	2013
Promo2	0	1	1	0	0	0	0
Promo2SinceWeek	NA	13	14	NA	NA	NA	NA
Promo2SinceYear	NA	2010	2011	NA	NA	NA	NA

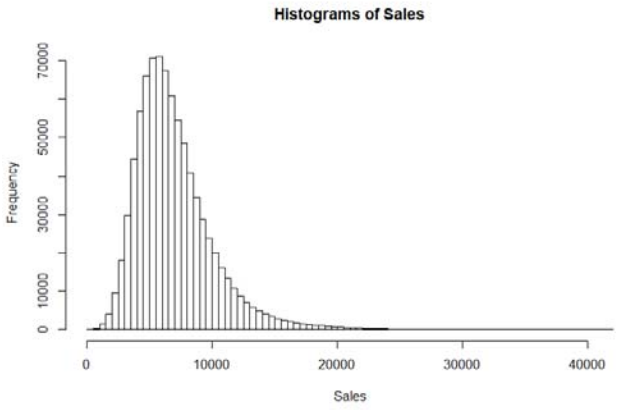
PromoInterval		Jan,Apr, Jul,Oct	Jan,Apr, Jul,Oct				

表 1 给出了 train.csv 中的一些样本示例，它包含 1017209 个样本，包括 1115 家商店从 2013 年 1 月 1 日至 2015 年 7 月 31 日约 2 年半的历史数据，但是有 180 家商店缺少从 2014 年 7 月 1 日至 2014 年 12 月 31 日约半年的数据；而 test.csv 包含 41088 个样本，包括 856 家商店从 2015 年 8 月 1 日至 2015 年 9 月 17 日约 6 周的数据。表 2 是 store.csv 中的数据示例，它有 1115 行，每一行表示一家商店的一些额外信息（如商店类型，评级等）。

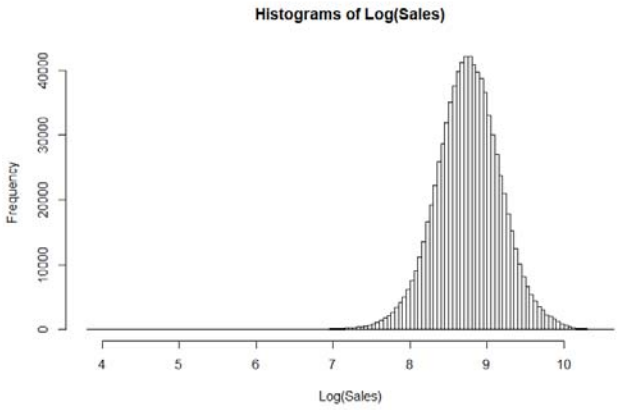
表 3 一些基本特征的说明

特征名	特征意义	特征类型	取值
Store	商店 Id	离散	1-1115
Sales	每日销售量	连续	
Customers	每日顾客量	连续	
Open	是否营业	离散	0,1
StateHoliday	所在州的节假日	离散	a,b,c,0
SchoolHoliday	是否受公立学校放假的影响	离散	0,1
StoreType	商店类型	离散	a,b,c,d
Assortment	商店评级	离散	a,b,c
CompetitionDistance	离最近的竞争者的距离	连续	存在 NA，占比 0.3%
CompetitionOpenSinceYear	最近的竞争者开业的年份	离散	1900,1961,...,2015，存在 NA，占比 31.7%
CompetitionOpenSinceMonth	最近的竞争者开业的月份	离散	1-12，存在 NA，占比 31.7%
Promo	当天是否有促销活动	离散	0,1
Promo2	商店是否参与连续的促销活动	离散	0,1
Promo2SinceYear	商店参与 Promo2 开始的年份	离散	2009-2015，存在 NA，占比 48.8%
Promo2SinceWeek	商店参与 Promo2 开始的周数	离散	1,5,...,50，存在 NA，占比 48.8%
PromoInterval	每年 Promo2 开始的月份	离散	“”，"Jan,Apr,Jul,Oct", "Feb,May,Aug,Nov" ，"Mar,Jun,Sept,Dec"

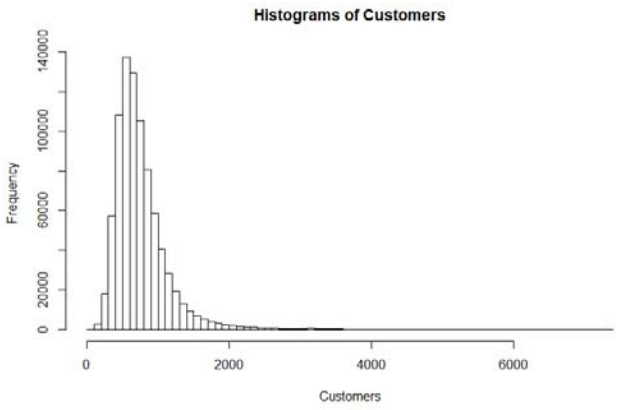
表 3 是数据集中用到的一些基本特征的说明，从表中发现，**数据集包含有大量的离散特征**，而连续特征非常少，这是本项目数据集的一个特点。



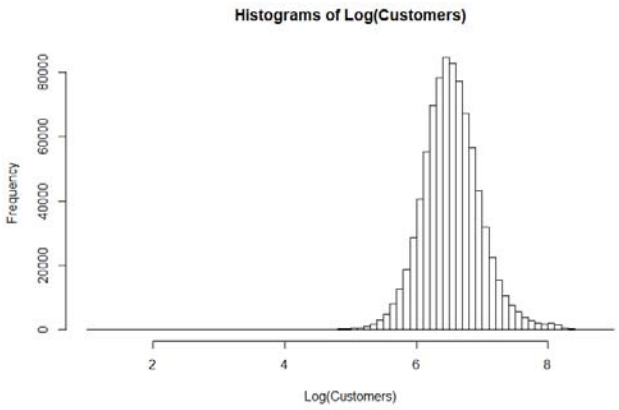
(a) Sales 的分布



(b) Log 转换后 Saels 的分布



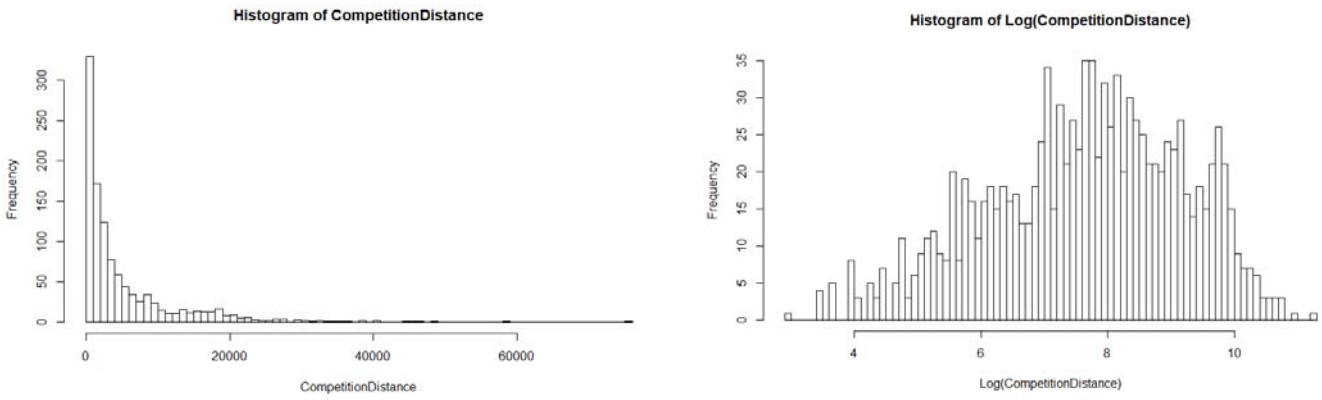
(c) Customers 的分布



(d) Log 转换后 Customers 的分布

图 1 销售量和顾客量分布图

图 1(a) 是销售量的分布图（不包括 Sale 为 0 的样本），可以观察到销售量存在着右偏斜，并且跨度很大，从最小的 46 到最大 41551，所以这里需对其进行 Log 转换。图 1(b) 是进行 Log 转换后的直方图，经过处理后的销售量分布很接近高斯分布。顾客量的分布与销售量类似，图 1(c)和(d)分别是顾客量处理前后的分布图，图 2(a)和(b)是 CompetitionDistance 处理前后的分布图。



(a) CompetitionDistance 的分布　　(b) Log 转换后 CompetitonDistance 的分布

图 2 CompetitionDistance 的分布

2.2 探索性可视化

1) 销售量随时间变化：

首先直观的观察一下销售量随时间的变化曲线。图 3 是 Store324 的销售量曲线（不包括 Sale 为 0 的点），可以观察到销售量大多数在 2000 到 10000 之间波动，红色的是销售量 1 个月的移动平均线，移动平均线没有呈现出明显趋势，表明**销售量的趋势性比较弱**。仔细观察销售量随时间的波动，直觉上其**波动呈现出周期性，波动周期大约是一周**。

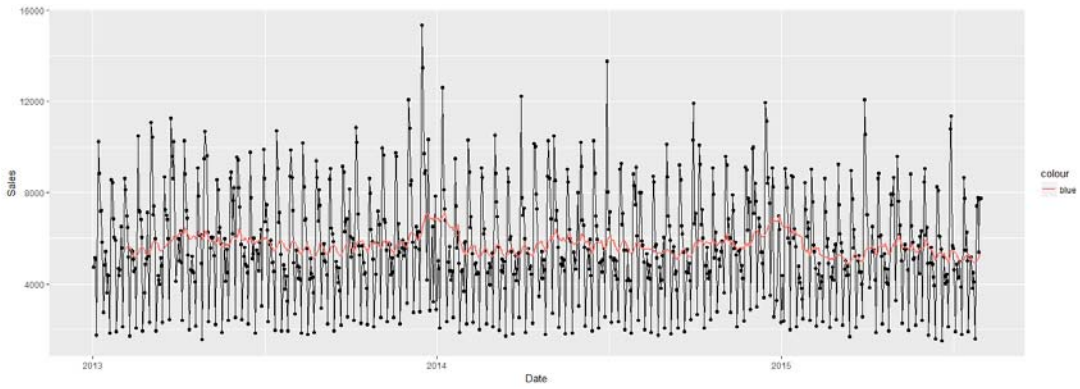


图 3　Store324 销售量曲线

2) Promo 对销售量的影响：

很容易猜想到 Promo 会对销售量产生积极的影响，这可以从图 4 中得到验证， 有促销的日子销售量明显要高于没有促销的日子。

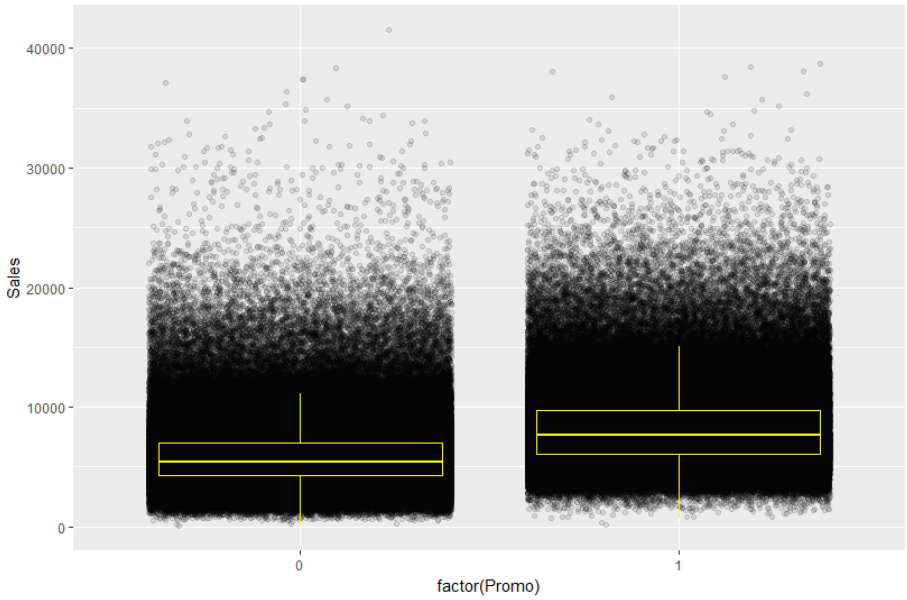


图 4　Promo 对销售量的影响

进一步的数据探索发现 **Promo 对销售量的影响会随时间而逐渐减弱**。由此可以提取出特征”PromoDecay”，它表示在一段连续的促销期内，距离促销开始的天数。图 5 以 Store58 为例，2013-1-7 到 2013-1-11 是一段促销期，那么它们所对应的 PromoDecay 值为 0-4，PromoDecay=5 表示非促销期。从图 6 中可以看出 PromoDecay=0, 1 时的销售量要明显大于其它时期。

Filter

	Store	Date	Promo	PromoDecay	Sales
1	58	2013-01-01	0	5	0
2	58	2013-01-02	0	5	4309
3	58	2013-01-03	0	5	3865
4	58	2013-01-04	0	5	4239
5	58	2013-01-05	0	5	4452
6	58	2013-01-06	0	5	0
7	58	2013-01-07	1	0	8439
8	58	2013-01-08	1	1	6704
9	58	2013-01-09	1	2	5385
10	58	2013-01-10	1	3	5833
11	58	2013-01-11	1	4	5863
12	58	2013-01-12	0	5	4585
13	58	2013-01-13	0	5	0
14	58	2013-01-14	0	5	4560
15	58	2013-01-15	0	5	3995
16	58	2013-01-16	0	5	3882

图 5　PromoDecay 特征

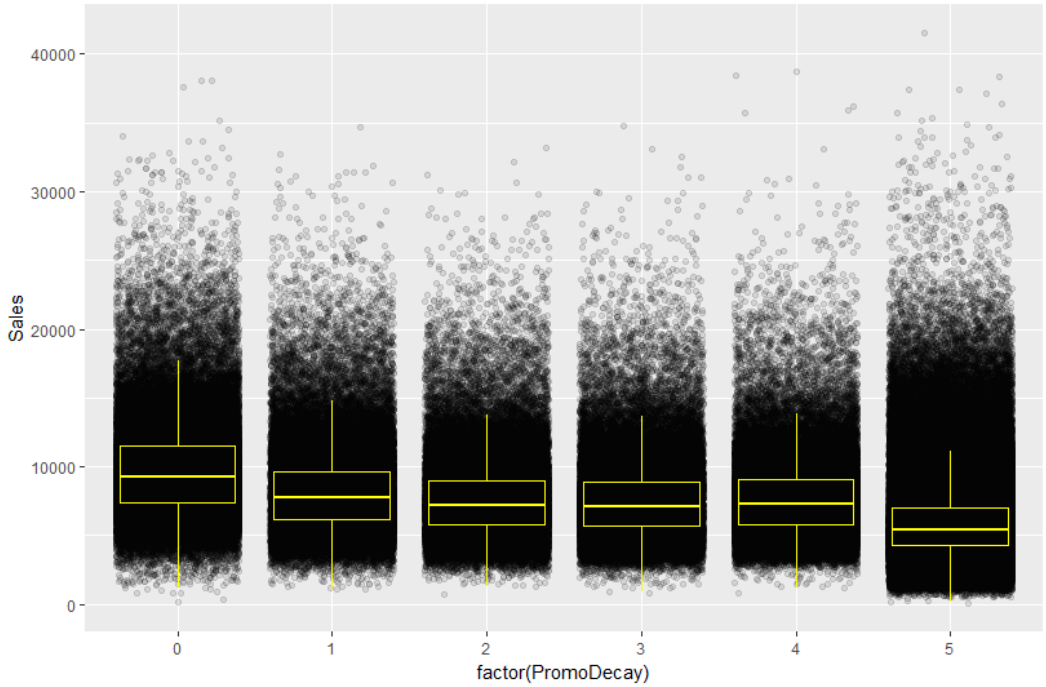


图 6　PromoDecay 与 Sales 的关系

3)　销售量与月份的关系：

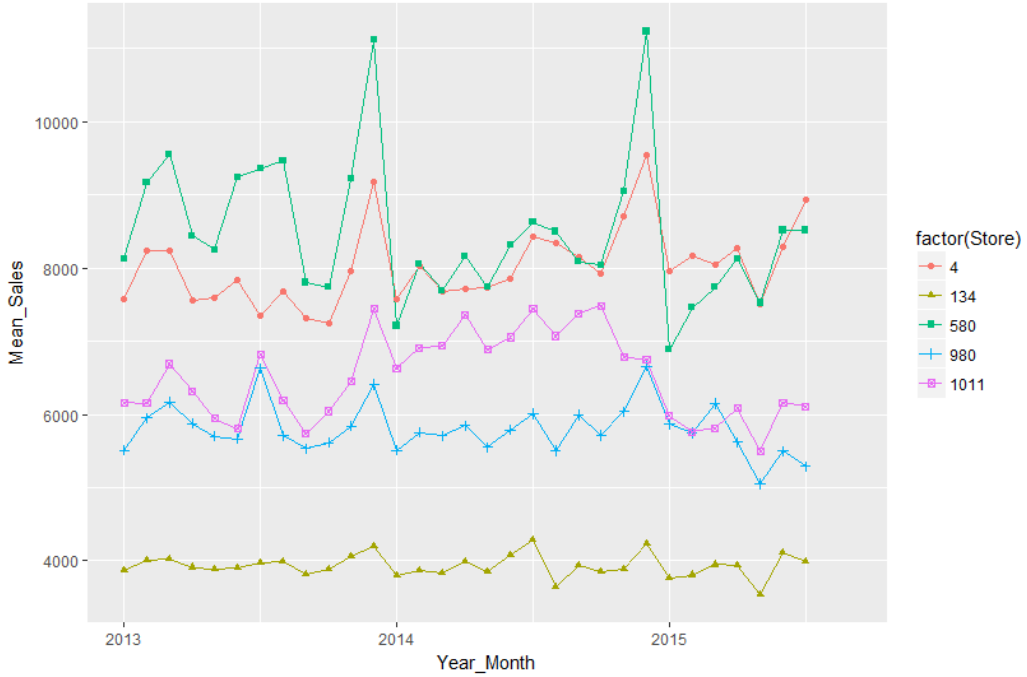


图 7　每月平均销售量曲线

图 7 是几家商店每月的平均销售量曲线，从图中可以看出，**大部分的商店在 12 月份销售量会迎来一个高峰，而到 1 月份会迅速回落**，这可能因为消费者为了迎接圣诞节，一般

会选择在节前进行大采购，这导致了 12 月份的销售高峰，而到 1 月份又因为家里的商品有富余，需求量下降导致销售量回落。所以月份对销售量有很大的影响，我们可以日期信息中提取”Month”这个特征。

2.3 算法和方法

本项目中我使用 R 进行数据探索、预处理、特征工程等，数据集处理使用的是 data.table 包，可视化使用的是 ggplot2 包，构建模型使用的是 Python。本项目要解决的是一个回归问题，并且数据集包含有大量的离散特征。对于离散特征的处理目前主要有以下几种方法：

1) One-hot 编码：将离散特征的每一种类型都编码为一个单独的特征。例如 StateHoliday 有四种类型值[0, a, b, c]，经过 One-hot 编码就会变成四个特征 StateHoliday_0、StateHoliday_a、StateHoliday_b、StateHoliday_c, 那么 StateHoliday=b 的样本就会编码为[0, 0, 1, 0]。这种编码的劣势就是在类型值数量很多的时候，其特征空间就会变得非常大，而如果样本数量不够的话，就会导致数据稀疏的问题。SVM，Linear Regression 模型经常使用这种处理方法。

2) Label 编码：把离散特征的类型与整数（一般是连续的整数）一一对应，不同的特征值编码为不同的整数。以 StateHoliday 为例，Label 编码会把它的四种类型与 0-3 一一对应，这样 StateHoliday=b 的样本就编码为 2。这种编码的劣势是它把原先不具有的数值尺度和位序关系赋给了离散特征，举例来说，经过 Label 编码后，以下等式 StateHoliday_b - StateHoliday_0 = 2*(StateHoliday_a – StateHoliday_0)成立，也就是说样本 StateHoliday_0 到 SateHoliday_b 的距离，是到样本 StateHoliday_a 距离的 2 倍，然而这在现实中是没有意义的。同样整数所具有的位序关系对 StateHoliday 也是没有意义的。当然有些离散特征也具有位序关系，像商店评级 Assortment，这类特征一般称为有序特征(ordinal feature)。Label 编码一般使用在树模型中。

3) Embedding: tensorflow 的 Programmer’s Guide 文档把 Embedding 定义为一个从离散对象到向量的映射[3]。Embedding 的特点在于这个从离散对象到向量的映射是通过训练从数据中学习得到的。经过训练之后，两个离散对象所对应的向量之间的距离，便成为它们之间某种相似性的可靠的度量。这种技术的一个很好的例子就是 word2vec[4]，训练后两个单词向量之间的距离是一种相似性的度量。king – queen ≈ man – woman, walking – walked ≈ swiming – swam，如图 8 所示。

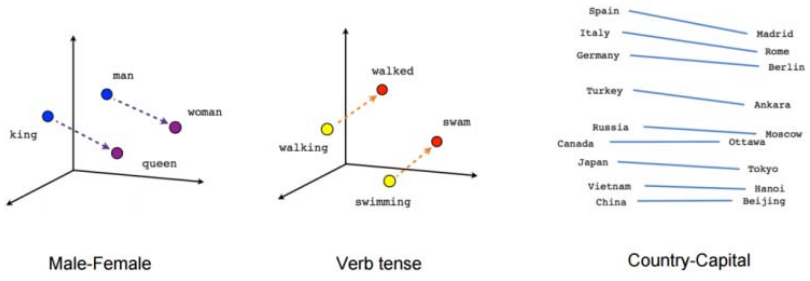


图 8 word vector[5]

本项目中我使用 XGBoost[6]和神经网络模型，XGBoost 是一种梯度回归树模型，它使用 Label 编码处理离散特征，而神经网络则使用 Embedding。对于训练集，首先我移除了 Sale 为 0 的样本，然后划分最后六周的样本作为验证集，把验证集的评分作为模型选择和参数调优的标准。

2.4 基准测试

本项目以 Cheng Guo 的 Entity-Embedding 模型[7]为基准，他在 Rossman Store Sales 竞赛中以 0.10583 的成绩取得了第三名。本项目的目标是在 Kaggle 私有测试集的评分超过 0.10583。

3. 方法

3.1 数据预处理

数据集 store.csv 中的特征 CompetitionDistance, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2SinceWeek, Promo2SinceYear 存在着 NA 值，我使用这些特征的中位数来填充 NA 值。数据集 train.csv 中有 180 家商店缺失半年的记录，我使用零值填充这些缺失的记录，使得数据在时间上保持连续性，以便进行时间序列处理。然后将 train.csv，test.csv 与 store.csv 合并形成待处理的数据集。

除了 Kaggle 提供的公开数据，还有一些额外的数据可以从 Kaggle 论坛上获取，比如各家商店所在州的数据[8]，天气数据[9]以及 GoogleTrend[10]数据，甚至还有一些体育赛事的信息[11]。还有一些像德国各州的节假日的信息[12]可以从互联网上获取。

离散特征使用 Label 编码转换为整数，而连续特征比如 CompetitionDistance、Customers 和 Sales，因为其分布存在偏斜，对其进行 Log 转换。连续特征归一化处理至 0 到 1。

3.2 实现

本项目实现了以下几种模型：XGBoost 模型，Entity-Embedding 基准模型，EE-Residual 模型，EE-tree 模型。

简单的 XGBoost 模型

首先，我实现了一个简单的 XGBoost 模型，这个模型只使用了一些基本的特征，如表 4 所示。features_base 列是这些特征的名称，它们的意义大部分都是不言自明的，需要说明的是 CompeteOpenMonths 表示最近的竞争者开业距当天的时间（以月为单位），Promo2OpenWeeks 表示 Promo2 开始距当天的时间（以周为单位）。

XGBoost 模型的参数我参考了 Kaggle 论坛上公开的代码[13]，不同的是，对于这个简单

的模型，我为了加快训练速度把学习率 eta 设为 0.3，并使用了 hist 方法。我把训练集的最后 6 周划分出来作为验证集，并没有使用 KFold 方法。

表 4　XGBoost 模型的设置

features_base	parameters	train parameters
Store, DayOfWeek, Promo, Year, Month, Day, WeekOfYear, DayOfYear, StateHoliday, SchoolHoliday, CompetitionOpenSinceYear, CompetitionOpenSinceWeek, Promo2SinceYear, Promo2SinceWeek, StoreType, Assortment, Promo2Interval, CompeteOpenMonths, Promo2OpenWeeks	objective: reg:linear max_depth: 8 eta: 0.3 subsample: 0.8 colsamples_bytree: 0.97 tree_method: hist	num_boost_round:500 early_stopping_rounds:50

Entity-Embedding 模型

我根据本次 Kaggle 竞赛第三名 Cheng Guo 的访谈[14]及源代码[7]，构建了 Entity-Embedding 神经网络模型，它的网络结构如图 9 所示。

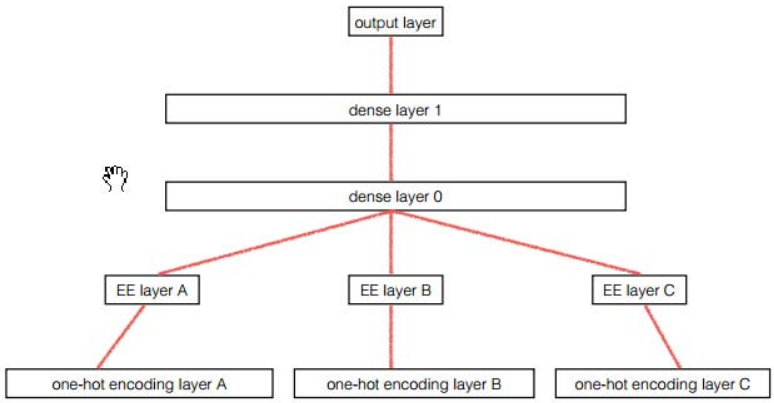


图 9　Entity-Embedding 模型结构[15]

我使用 Keras 深度学习库[16]来构建这个模型，图 10 是通过 Keras 构建的模型的结构图，图中只示意了少量的特征。输入的离散特征（如 Store，DayOfWeek）通过 Embedding 层转化为向量，而连续特征（如 Distance）或布尔特征（如 Promo，SchoolHoliday）则通过 Dense 层连接，这些 Embedding 层的输出和 Dense 层的输出，通过 Concatenate 层合并起来，然后在上方连接 2 个 Dense 层，最后是输出层。Concatenate 层使用了 dropout，概率设为 0.02。两个 Dense 层神经元数量分别是 1000 和 500，权值使用 random_uniform 初始化，激活函数选择 relu。输出层使用 sigmoid 激活函数，这样输出被限制在 0-1 之间，相应的目标变量 Sales 进行了 Log 转换，并归一化处理至 0 到 1。损失函数选择了 mean_absolute_error，使用 Adam

优化函数，迭代次数为 20 次。Cheng Guo 的原模型 batchsize 设为 128，本项目中改为 2048（对模型影响不大，但是训练速度大大加快了）。

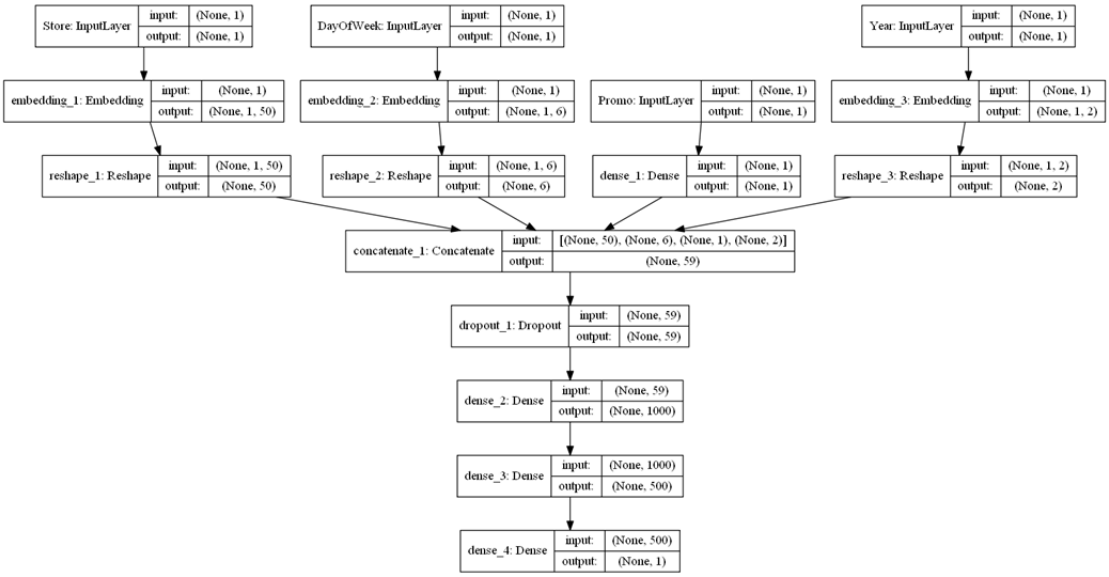


图 10　Entity-Embedding 模型的 Keras 实现

Cheng Guo 在提交到 Kaggle 的最终模型中使用了 30 多个特征。表 5 是这些特征的详情，其中 State 表示商店所在的州，Temperature, Humidity, Wind, Cloud, WeatherEvent 是当地的天气信息，GoogleTrend_DE 和 GoogleTrend_State 分别是德国以及各个州的 GoogleTrend 信息，这些特征是从[8], [9], [10]中的数据提取出来的。LatestPromo2Months 表示上一个 Promo2 为 1 的月距离现在的月数，XX_Forward(Backward)表示下一个（上一个）的 XX不为 0的日子到今天的天数，XX_Count_Forward(Backward)表示在下一周（上一周）周期内 XX 不为 0 的天数之和。最后模型 Concatenate 层的维度是 131（Cheng Guo 的原模型为 132）。

表 5　Entity-Embedding 模型使用的特征

特征	特征类型	类别数量	Embedding(Dense)层输出维度
Store	离散	1115	50
DayOfWeek	离散	7	6
Promo	布尔	2	1
Year	离散	3	2
Month	离散	12	6
Day	离散	31	10
StateHoliday	离散	4	3
SchoolHoliday	布尔	2	1
CompeteOpenMonths	离散	25	2
Promo2OpenWeeks	离散	26	1
LatesetPromo2Months	离散	4	1
Distance	连续	/	1
StoreType[1]	离散	*4*	*2*

Assortment[2]	离散	*3*	*2*
PromoInterval	离散	4	3
CompetitionOpenSinceYear	离散	18	4
Promo2SinceYear	离散	8	4
State	离散	12	6
WeekOfYear	离散	53	2
Temperature	连续	/	3
Humidity	连续	/	3
Wind	连续	/	2
Cloud[3]	离散	*9*	*1*
WeatherEvent	离散	22	4
Promo_Forward	离散	8	1
Promo_Backward	离散	8	1
StateHoliday_Forward	离散	8	1
StateHoliday_Backward	离散	8	1
StateHoliday_Count_Forward	离散	3	1
StateHoliday_Count_Backward	离散	3	1
SchoolHoliday_Forward	离散	8	1
SchoolHoliday_Backward	离散	8	1
GoogleTrend_DE	连续	/	1
GoogleTrend_State	连续	/	1

注：

1. StoreType 特征在源码中类别数量被记为 5，实际上应该是 4(取值为 a, b,c,d)；

2. Assortment 特征在源码中类别数量被记为 4，实际上应该是 3(取值为 a,b,c)；

3. Cloud 特征在源码中被当做了连续特征，后面连接了 Dense 层，实际上它是离散特征，应该用 Embedding 层。

EE-Residual 模型

Embedding 技术在很多地方得到了应用，Ying Shan 在赞助商搜索（Sponsored Search）应用[17]中使用 Embedding 处理文本特征，文本的词汇大小到达了 49,292。不同于 Cheng Guo 的 Entity-Embedding 模型在 Concatenate 层上方的使用 Dense 层连接，Ying Shan 借鉴了 ResNet[18]中 Residual Block 的概念，使用了 Residual Unit 代替 Dense 层，图 11 是 Ying Shan 所构建的 Deep Crossing 模型。Ying Shan 对 ResNet 中的 Residual Block 做了修改，去掉了其中的卷积层，图 12 是 Ying Shan 所使用的 Residual Unit。

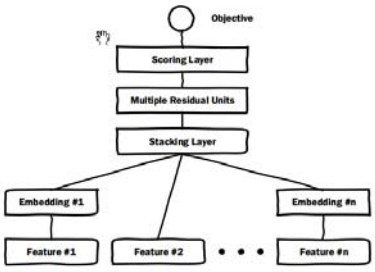


图 11　Deep Cross 模型结构 [17]

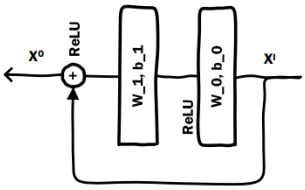


图 12　Residual Unit [17]

我把 Ying Shan 在[17]中所使用的模型应用于本项目，用两个 Residual Layer 代替了 Entity-Embedding 模型中的两个 Dense 层，这两个 Residual Layer 的神经元分别为 512 和 256。图 13 是 Residual Layer 的结构，它的内部使用了 dropout，概率为 0.2。损失函数同样选择 mean_absolute_error，使用 Adam 优化函数，迭代次数为 30 次。

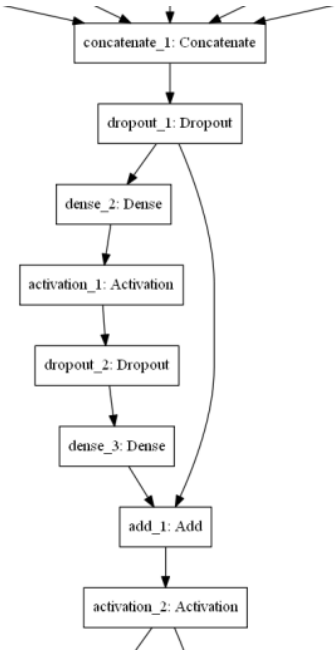


图 13 Residual Layer 结构

EE-tree 模型

训练完成后，Entity-Embedding 模型的 Concatnate 层的输出可以作为输入特征，提供给其他模型使用，本项目选择用 LightGBM[19]（与 XGBoost 相比，它的速度更快）。模型使用的参数如下：

```
params = {
    'boosting_type': 'gbdt',
    'objective': 'regression_l1',
    'num_leaves': 31,
    'min_data_in_leaf': 100,
    'learning_rate': 0.1,
    'feature_fraction': 0.95,
    'bagging_fraction': 0.8,
    'bagging_freq': 5,
    'max_bin': 255,
    'tree_learner': 'data',
    #'nthread': 4
}
```

3.3 改进

本项目改进措施主要集中在特征工程上，我在 Entity-Embedding 模型使用的特征的基础上，尝试了很多其他的特征。

表 6 项目所构建的特征

特征	意义	特征类型	类别数量	Embedding(Dense)层输出维度
SchoolHoliday_Count_Forward	下一周 SchoolHoliday 的天数	离散	8	4
SchoolHoliday_Count_Backward	上一周 SchoolHoliday 的天数	离散	8	4
Promo_Count_Forward	下一周 Promo 的天数	离散	6	3
Promo_Count_Backward	上一周 Promo 的天数	离散	6	3
PromoDecay	在一段连续 Promo 期间，Promo 开始的天数（见 2.2）	离散	6	3
AvgSales_Per_Day	每天平均销售量	连续	/	1
AvgCustomers_Per_Day	每天平均顾客量	连续	/	1
AvgSales_Per_Customer	每个顾客平均消费	连续	/	1
Sales_Promo	Promo 时的销售量占比	连续	/	1
Sales_Holiday	在 StateHoliday 时的销售量占比	连续	/	1

Sales_Saturday	在周六时的销售量占比	连续	/	1
Open_Ratio	开门营业时间占比	连续	/	1
SchoolHoliday_Ratio	SchoolHoliday 时间占比	连续	/	1
Before_Long_Closed	长时间关门事件开始之前的天数	离散	6	3
After_Long_Closed	长时间关门结束之后的天数	离散	6	3
Before_SummerHoliday_Start	暑假开始之前的天数	离散	16	3
After_SummerHoliday_Start	暑假开始之后的天数	离散	16	3
Before_SummerHoliday_End	暑假结束之前的天数	离散	16	3
After_SumerHoliday_End	暑假结束之后的天数	离散	16	3

Cheng Guo 在 Entity-Embedding 模型中，依靠经验法则决定各个 Embedding 层的输出维度，在本项目中使用下面的公式计算输出维度，其中m_i是离散特征x_i的类别数量。

$$D_i = \min\left(50, floor\left(\frac{m_i + 1}{2}\right)\right)$$

项目中我还尝试了使用训练集的不同子集进行训练。因为训练集包含 1115 家商店，而测试集只有其中的 856 家，为了保持训练集与测试集分布规律的一致性，我尝试从训练集中移除了那些商店不包含在测试集中的数据，移除数据后训练集大约是原来的 76%。

4. 结果

4.1 模型评估与验证

简单 XGBoost 模型

本项目我使用训练集中最后 6 周的数据（约占训练集的 4.8%）作为验证集，以 RMSPE 作为评估标准。我构建的简单 XGBoost 模型，大约迭代了 320 次后结束训练，其在验证集上的得分为 0.1338。然后我使用 100%的训练集重新训练模型，其在 Kaggle 测试集上，Public Score 为 0.12513，Private Score 为 0.12871。

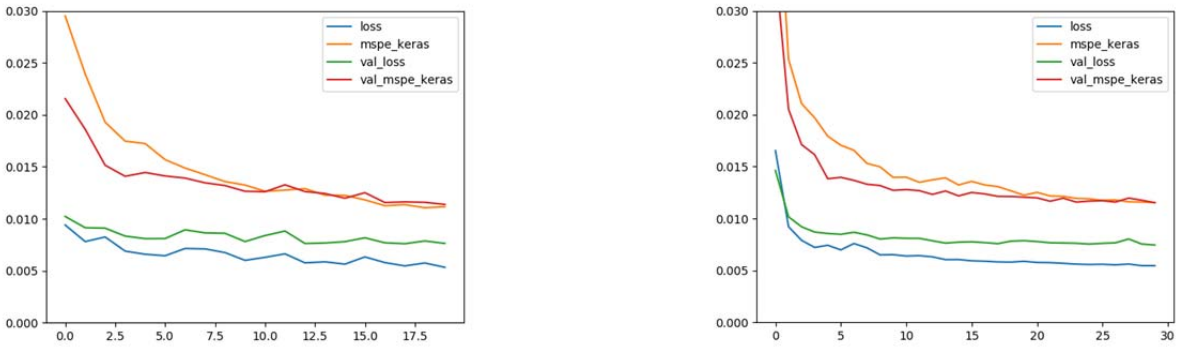
Entity-Embedding 模型与 EE-Residual 模型的对比

为了比较两种模型的效果，我使用相同的训练集，特征都选择表 5 所示 Entity-Embedding 原模型的特征。两种模型每个都运行 3 次，比较其在验证集上的评分，然后再在整个训练集上运行 3 次，比较在 Kaggle 测试集上的评分。

表 7　Entity-Embedding 模型与 EE-Residual 模型结果对比

Model	Concatnate dimension	Parameters (thousand)	Run	Val-rmspe	Public Score	Private Score
Entity-Embedding	131	690	1	0.10667	0.10355	0.11683
			2	0.10845	0.10678	0.11538
			3	0.10975	0.10414	0.11942
			mean	**0.10829**	**0.10482**	**0.11721**
EE-Residual	131	259	1	0.10732	0.09905	0.11208
			2	0.10930	0.09992	0.11412
			3	0.11285	0.10015	0.11084
			mean	**0.10982**	**0.09971**	**0.11235**

表 7 是两个模型的结果对比，EE-Residual 模型在验证集上的评分略高于 Entity-Embedding 模型，但是在 Kaggle 测试集上的 Public Score 和 Private Score 都低于 Entity-Embedding 模型，总体来说，两个模型的效果相当，但是 EE-Residual 模型的参数却只有 Entity-Embedding 模型的 38%。图 14 是这两个模型的训练曲线。



(a) Entity-Embedding 模型　　(b) EE-Residual 模型

图 14　Entity-Embedding 模型与 EE-Residual 模型训练曲线

Fine-tune EE-Residual 模型

经过尝试不同的特征组合，优化模型参数，得到最终的 Fine-tune EE-Residual 模型。模型所用的特征见表 8。模型使用 2 个 Residual Layer，神经元数量分别为 512 和 256，第一个 Residual Layer 设置了 dropout，概率为 0.2。Concatnate 层维度为 249，dropout 设为 0.03。BatchSize 设为 2048，epochs 设为 25。Fine-tune EE-Residual 模型运行的结果见表 9。

表 8　Fine-tune EE-Residual 模型所用特征

Features	Store, DayOfWeek, Promo, Year, Month, Day, StateHoliday, SchoolHoliday, CompeteMonths, Promo2Weeks, Lastest_Promo2_Months, Distance, StoreType, Assortment, PromoInterval, CompetitionOpenSinceYear, Promo2SinceYear, State, WeekOfYear, Temperature, Humidity, Wind, Cloud, WeatherEvent, Promo_Forward, Promo_Backward, StateHoliday_Forward, StateHolidy_Backward, SchoolHoliday_Forward, SchoolHoliday_Backward,Promo_Count_Forward, Promo_Count_Backward, StateHoliday_Count_Forward, StateHoliday_Count_Backward, SchoolHoliday_Count_Forward, SchoolHoliday_Count_Backward, GoogleTrend_DE, GoogleTrend_State, Sales_Per_Day, Customers_Per_Day, Sales_Per_Customer, Sales_Promo, Sales_Holiday, Sales_Saturday, Open_Ratio, SchoolHoliday_Ratio, Before_Long_Closed, After_Long_Closed

EE-tree 模型

EE-tree 模型经过 5000 次迭代后，结束训练。这个模型训练比较耗时，在 i7-4710MQ CPU@2.50GHz 的机器上，花费了大约 2 小时，表 10 是模型运行的结果。

表 9　Fine-tune EE-Residual 模型运行结果

Model	Concatnate dimension	Parameters (thousand)	Run	Val-rmspe	Public Score	Private Score
Fine-tune EE-Residual	249	443	1	0.10625	0.09768	0.10850
			2	0.10534	0.09547	0.10739
			3	0.10446	0.09560	0.10566
			mean	**0.10535**	**0.09625**	**0.10718**

表 10　EE-tree 模型结果

Model	Val-rmspe	Public Score	Private Score
EE-tree	**0.09626**	**0.09976**	**0.10725**

表 11 是所有模型运行结果的汇总表，注意 Val-rmspe 是模型在划分出验证集后的训练集上训练的结果，而 Public Score 和 Private Score 是模型在整个训练集上训练的结果， 树模型（Simple XGBoost 和 EE-tree）是在本机 i7-4710MQ 上训练的，神经网络模型（Entity-Embedding、EE-Residual 和 Fine-tune EE-Residual）是在 aws p2.xlarge 上训练的。 从表中可以看出，EE-tree 模型与 Fine-tune EE-Residual 模型相比，前者在验证集上的表现更好，但是在公开测试集和私有测试集上要弱于后者，而且训练时间远远超过后者，这主要是因为 Fine-tune EE-Residual 模型使用了 GPU。Fine-tune EE-Residual 模型与 Entity-Embedding 模型对比，前者使用了更少的参数（70%），但在验证集、公开测试集和私有测试集上的评分都优于后者。综合来说，Fine-tune EE-Residual 模型表现最优。

表 11 所有模型的运行结果

Model	Concatnate dimension	Parameters (thousand)	Val-rmspe	Public Score	Private Score	Training Time
Simple XGBoost	/	/	0.01338	0.12513	0.12871	18 mins
Entity-Embedding	131	690	0.10829	0.10482	0.11721	**200 secs**
EE-Residual	131	259	0.10982	0.09971	0.11235	300 secs
Fine-tune EE-Residual	249	443	0.10535	**0.09625**	**0.10718**	250 secs
EE-tree	/	/	**0.09626**	0.09985	0.10745	2 hours

模型融合

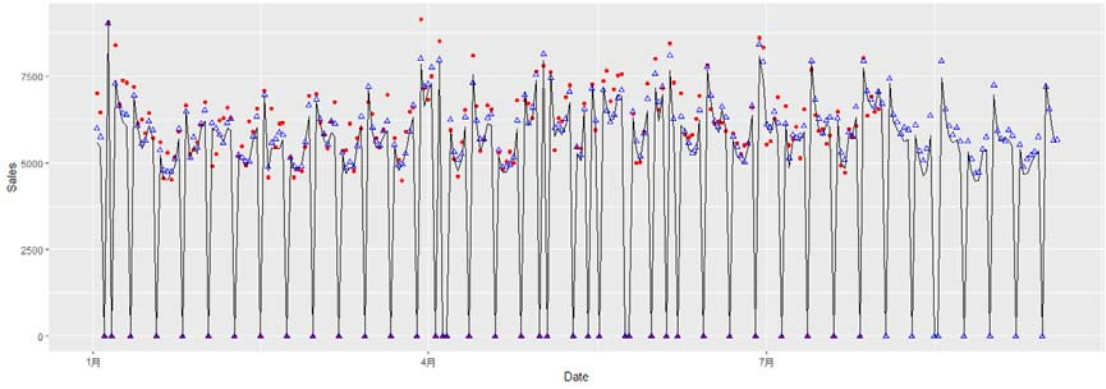
Cheng Guo 的原模型融合了 10 个 Entity-Embedding 模型，在 Kaggle 测试集上，Public Score 为 0.09563，Private Score 为 0.10583。本项目采用均值法，融合 10 个 Fine-tune EE-Residual 模型，其中单个模型 Private Score 最优的可以达到 0.10479。融合模型在 Kaggle 测试集上，Public Score 为 0.09059，Private Score 为 0.10311，两者皆优于 Cheng Guo 的原模型。融合模型在 Kaggle Private Leaderboard 上排名为 2/3303，达到了项目的预期目标。

表 12 Fine-tune EE-Residual 融合模型与 Cheng Guo 的原模型对比

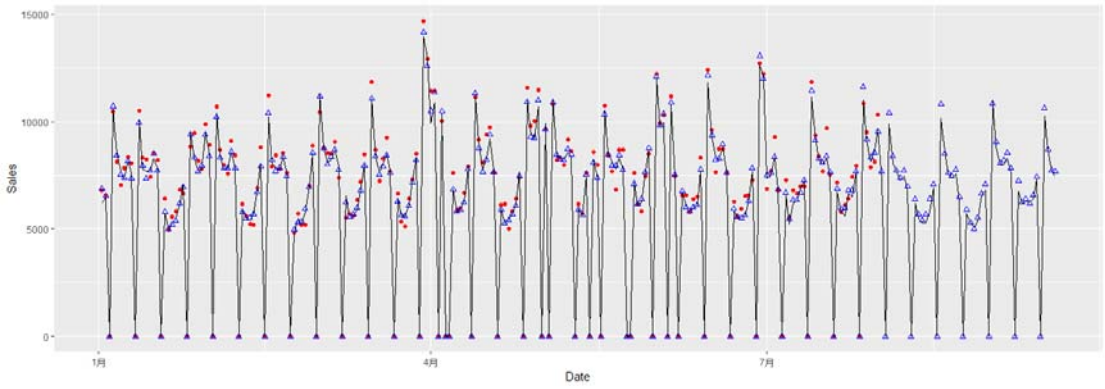
Model	Run	Public Score	Private Score
Fine-tune EE-Residual	1	0.09372	0.10479
	2	0.09415	0.10776
	3	0.09493	0.10904
	4	0.09415	0.10776
	5	0.0943	0.10643
	6	0.09436	0.10758
	7	0.09935	0.10957
	8	0.09606	0.10856
	9	0.09845	0.10975
	10	0.09582	0.10518
	ensemble	**0.09059**	**0.10311**
Cheng Guo’s Entity-Embedding	ensemble	0.09563	0.10583

5. 结论

5.1 可视化



(a) Store 174



(b) Store 77

图 15 模型的预测结果

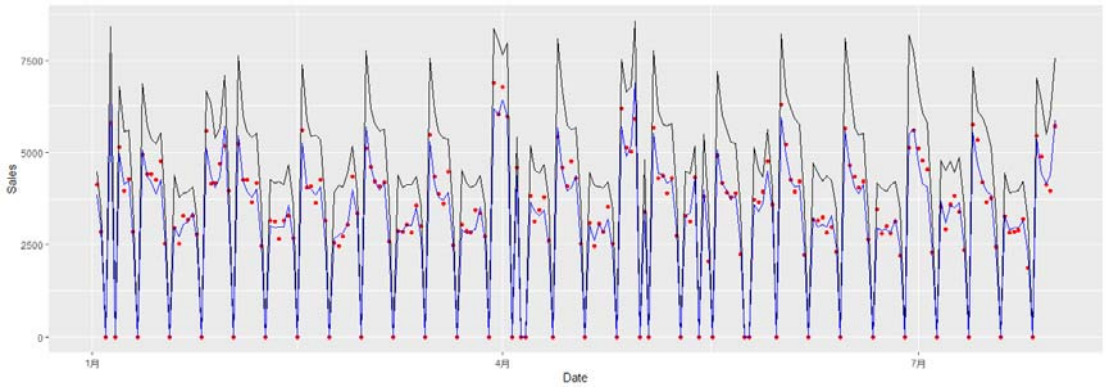


图 16 Store78 预测结果

图 15 是模型预测的结果，红色实心圆表示真实值，实线是单个模型的预测值，蓝色三角是融合模型的预测值，对 Store177 模型在波动剧烈的地方预测不够好，融合模型在这些地方要优于单个模型，Store77 预测结果整体上要优于 Store177。

图 16 是模型预测 Store78 的结果，蓝色实线是用整个训练集训练的模型（Model A）预测结果，而黑色实线是用训练集的子集（即移除商店不在测试集中的数据）训练的模型（Model B）的预测结果。Model A 比 Model B 预测偏差明显大，这是因为 Store78 不在测试集中，Model A 没有学习 Store78 的数据。仔细观察两条实线，还可以发现它们的波动规律很相似，即如果把 Model B 的预测线（黑色线）整体下移，它与 Model A 的预测线（蓝色线）就基本重合。这说明 Model B 可以从其他的 Store 中学习到波动规律，只是因为它缺少 Store78 的平均销售量信息，所以才会出现预测值整体偏高的现象。从这个角度也说明模型的泛化能力很强。

Entity-Embedding 模型的一个缺点在于它是一个黑箱模型，无法识别出各个特征的重要性。然而如果把 Embedding 特征输入到树模型中去，训练完成后就可以得到 Embedding 特征各个分量的重要度。 图 17 是 EE-tree 模型训练后得到的特征重要度，Embedding 特征总共有 249 个，图中标出了排名前 50 的特征，横坐标特征名后面的数字表示该特征对应的 Embedding 特征的第几个分量。从图中可以看出，Sales_Per_Day 重要性最高，说明平均销售量起决定性作用，销售量始终围绕着平均值波动。另一点是 DayOfWeek 总共有 6 个分量，它有 3 个分量都排在前 20， 这表明了 DayOfWeek 特征的重要性，这一点与销售量呈现出的周期性是吻合的。

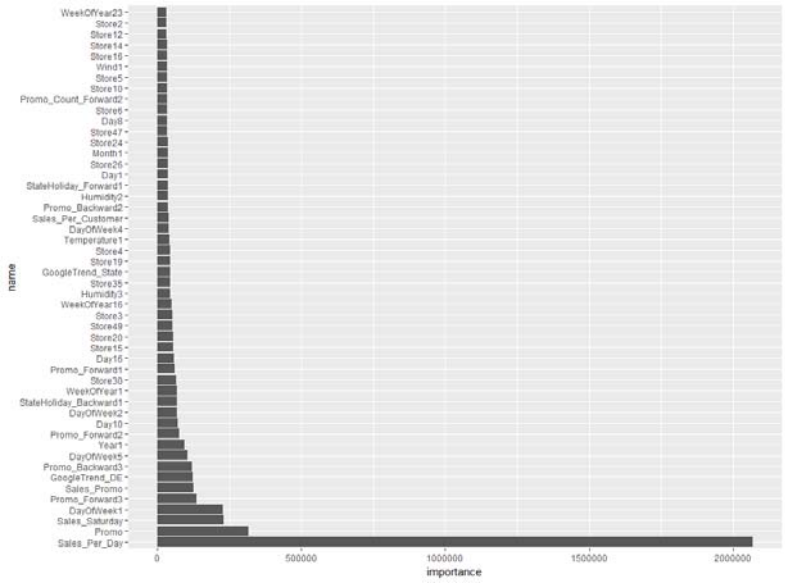


图 17 Embedding Features 重要度

5.2 思考

本项目解决的是一个销售量预测问题，它要求利用过去 2 年半的历史数据，来预测未来 6 周的销售量，这是一个**时间序列预测**问题。解决这类问题的一个关键在于，要意识到销售

量**不仅仅跟当天的因素有关，还会受到过去或未来的因素的影响，**因此需要把过去或未来一段时间的某些因素的平均值、计数等作为当天的特征。另一个需要注意的，也是所有机器学习问题需要注意的地方，是务必保留一部分数据作为验证集，以监测模型是否过拟合。本项目采用按时间分割的方法，划分训练集最后 6 周的数据作为验证集。

在 Rossmann Store Sales 的竞赛中，大多数参赛者使用 XGBoost 模型，而 Cheng Guo 利用 Embedding 技术构建深度神经网络模型，取得了第三名的好成绩。与 XGBoost 相比，Entity-Embedding 模型一定程度上减少了特征工程的工作量，这得益于 Entity-Embedding 能够从数据中学习到各个离散特征的固有的特性[15]。另一方面 Embedding 减少了特征空间的维度，使得模型更不容易过拟合。

本项目在 Entity-Embedding 模型的基础上，借鉴 Ying Shan 的 Deep Cross 模型[17]，使用残差块构建了 EE-Residual 模型，相比于原始 Entity-Embedding 模型，使用了更少的参数，获得了更好的精度。

5.3 后续改进

本项目后续可能改进的地方：

一是可以把销售量在不同的时间窗口的移动平均值作为特征，加入到模型中去，比较是否会改善模型。第二，Embedding 层的维度应该有一个更优的方法来确定，这可能跟数据在 Embedding 空间的分布有关。第三，项目没有很好地利用 Customers 数据。可以构建一个 Customers 的模型，来预测测试集的 Customers，并把预测结果作为特征，添加到 Sales 模型中去。还有一种方法是可以构建一个既输出 Sales 又输出 Customers 的多输出模型来利用 Customers 信息。第四，EE-tree 模型因为训练时间太长，没有进行很多的参数调优以及模型融合。

6. 参考文献

[1] Rossmann Store Sales, https://www.kaggle.com/c/rossmann-store-sales

[2] RMSPE, https://www.kaggle.com/c/rossmann-store-sales#evaluation

[3] Embeddings(tensorflow Programmer's Guide), https://www.tensorflow.org/programmers_guide/embedding

[4] Vector Representations of Words, https://www.tensorflow.org/tutorials/word2vec

[5] https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12

[6] XGBoost, https://xgboost.readthedocs.io/en/latest/

[7] Cheng Guo's Entity-Embedding 模型源代码,
https://github.com/entron/entity-embedding-rossmann/tree/kaggle

[8] Putting stores on the map,

https://www.kaggle.com/c/rossmann-store-sales/discussion/17048

[9]Weather at Berlin US Airport ,
https://www.kaggle.com/c/rossmann-store-sales/forums/t/17058/weather-at-berlin-us-airport

[10]Google Trends Data,

https://www.kaggle.com/c/rossmann-store-sales/discussion/17130#97196

[11]External data and Other information ,
https://www.kaggle.com/c/rossmann-store-sales/discussion/17229

[12] https://www.tuxx.uk/countries/germany/schoolholidays/2014/

[13] xgb-rossmann 代码, https://www.kaggle.com/abhilashawasthi/xgb-rossmann

[14] Rossmann Store Sales 第三名 Cheng Guo 的访谈，

http://blog.kaggle.com/2016/01/22/rossmann-store-sales-winners-interview-3rd-place-cheng-gui/

[15] Cheng Guo, Felix Berkhahn, Entity Embeddings of Categorical Variables ,
http://www.reuters.com/article/us-china-pollution-idUSKBN0UB1KB20151229

[16] Keras, https://keras.io

[17] Ying Shan, T.Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, JC Mao, Deep Crossing: Web-Scale Modeing without Manually Crafted Combinatorial Features.

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

[19]LightGBM, https://lightgbm.readthedocs.io/en/latest/index.html

[20] Gert 的帖子 Model documentation 1st place
https://www.kaggle.com/c/rossmann-store-sales/discussion/18024