Kaggle: Predict Future Sales

张驰　1801111587　前沿交叉学科研究院
伊照　1801214010　前沿交叉学科研究院

2019 年 4 月 26 日

目录

1 项目介绍

我们组选择参加 kaggle 比赛 (如图 1)，目前排名在 top5%(120/2876)。

图 1: Kaggle: Predict Future Sales

1.1 任务描述

给定从 2013.1-2015.10 的各个商店卖出各个商品的每日销售数据，预测下个月指定商店卖出指定商品的数量 (限制在 0-20 之间)。

1.2 数据集描述

文件描述：

- sales_train.csv：训练集。从 2013.1-2015.10 的每日销售数据。
- test.csv ：测试集。需要预测 2015.11 这些商店的商品月销售量。
- sample_submission.csv：样本提交文件。
- sales_train.csv：训练集。从 2013.1-2015.10 的每日销售数据。
- items.csv：关于商品的补充信息。
- item_categories.csv：商品类别补充信息。
- shops.csv：商店补充信息。

数据字段描述：

- ID：表示测试集中的（Shop，Item）元组的 Id。
- shop_id ：商店的唯一标识符。
- item_id：产品的唯一标识符。
- item_category_id：项目类别的唯一标识符。
- item_cnt_day：销售的产品数量。需要预测的是此字段的每月销售量。
- item_categories.csv：商品类别补充信息。
- item_price ：商品的当前价格。

- date ：日期格式为 dd // mm // yyyy。
- date_block_num：为了方便使用连续的月号。2013 年 1 月是 0，2013 年 2 月是 1，...，2015 年 10 月是 33。
- item_name：项目名称。
- shop_name：商店名称。
- item_category_name：项目类别的名称。

2 特征工程

该比赛主要是进行数据集的特征工程，构造大量的特征列，进行特殊处理，并利用回归模型与集成方法得到最终结果。本次实验中主要按照以下步骤进行处理的：

- 异常数据处理。
- 特征工程。
- 选择最佳特征。
- 选择模型训练。
- 模型集成。
- 特殊处理。

2.1 异常数据处理

如图 2 所示，先找到单品售价和单品日常销量的异常点，对异常点进行剔除，对于售价小于 0 的不正常值采取平均值赋值。

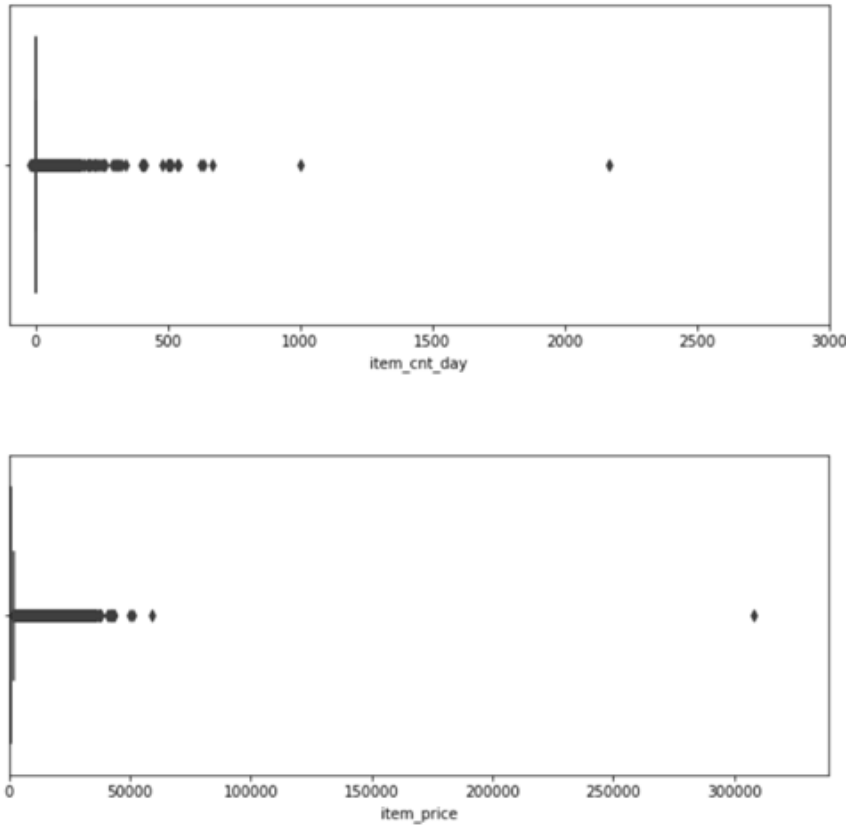

图 2: 数据集异常点

```
train = train[train.item_price<100000]
train = train[train.item_cnt_day<1001]

train.loc[train.item_price<0, 'item_price'] = median
```

2.2 特征工程

该步骤是整个项目关键的一环，特征选取的好坏会直接影响模型训练的结果。在实际中我们从以下几个方面来构造特征：

2.2.1 从源数据文件中构造特征

除了训练集的数据外，项目还提供了商店，商品，商品分类这些补充文件，我们可以充分挖掘这些这些文件中的隐含信息，构造新的特征，如图 3 所示。

- 每个商店的名字首项是城市名，取出城市后作为一列新特征，并对其因子化。
- 商品类别是每个商品的小类别，在类别名里还可分出两种大类别，作为两个新的特征 type 和 subtype。有两种情况，一种第一个有‘-’分割成两个元素第一个元素就是首选类别，第二个元素就是代替类别；没有‘-’则首选类别和代替类别都一样。并将这两个特征因子化。

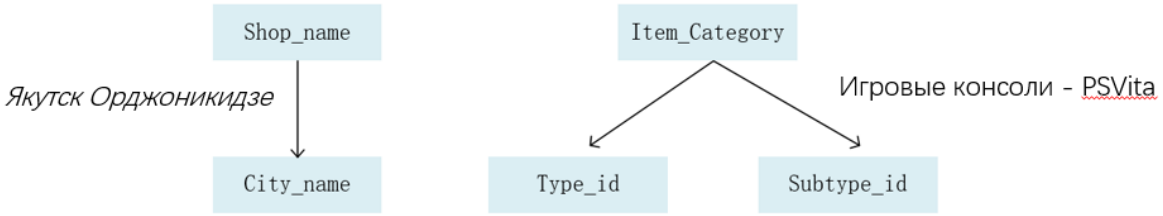

图 3: 源文件中构造特征

2.2.2 从项目要求中构造特征

在这里为了方便数据整理，我们新建了一个 matrix 矩阵，里面包含三个特征：date_block_num、shop_id、item_id, 三个特征数据来源于训练集且三个特征都不重复，每个月下每个商店卖出每个商品的矩阵，即使某一个月某商店没有卖出某件商品，但只要其他商店卖出过，也会记录。关键代码如下：

```
    for i in range(34):
      sales = train[train.date_block_num == i]
      matrix.append(np.array(list(product([i], sales.shop_id.unique(), sales.item_id.unique())),
                                          dtype='int16'))
```

之后将得到的各特征都存储到该 matrix 中，同时将测试集加入到 matrix 末尾，date_block_num 设为 34。

项目要求我们预测下个月商品的销售总量，而训练集中的数据给定的为商品的每日销售总量，因此可以构造训练集每月商品的销售总量以及每月销售次数 (即每月有销售记录的天数)，如图 4 所示：

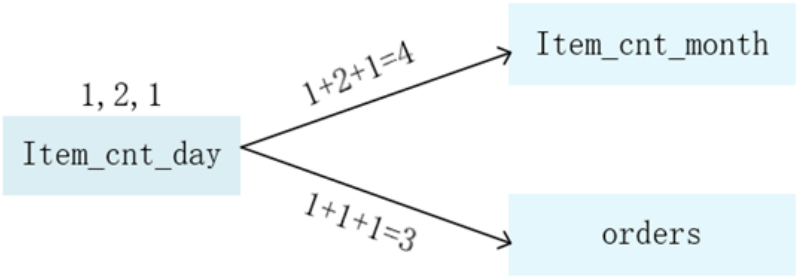

图 4: 每月商量销售总量以及每月销售次数

关键的代码为：

```
group = train.groupby(['date_block_num','shop_id','item_id']).agg({'item_cnt_day': ['sum', '
                                        count']})
group.columns = ['item_cnt_month', 'orders']
group.reset_index(inplace=True)

matrix = pd.merge(matrix, group, on=cols, how='left')
```

```
matrix['item_cnt_month'] = (matrix['item_cnt_month']
  .fillna(0)
  .clip(0,20) # NB clip target here
  .astype(np.float16))
matrix['orders'] = matrix['orders'].fillna(0).astype(np.float16)
```

2.2.3 从时间序列中构造特征

时间月份信息是一个非常重要的条件，如果有时间序列的稳定性足够强，那么可能存在着这个月的销量在一个月两个月三个月半年一年后还是这个销量，例如 date_block_num=1；shop_id=1；item_id=1 时的销量是 10，那么在其 1，2，3，6，12 月后也为 10。因此我们构造一个延迟处理函数，关键代码如下：

```
def lag_feature(df, lags, col):
tmp = df[['date_block_num','shop_id','item_id',col]]
for i in lags:
  shifted = tmp.copy()
  shifted.columns = ['date_block_num','shop_id','item_id', col+'_lag_'+str(i)]
  shifted['date_block_num'] += i
  df = pd.merge(df, shifted, on=['date_block_num','shop_id','item_id'], how='left')
return df
```

基于这个延迟函数，如图 5 所示，我们将已有的特征进行组合，以月份为基础，构造出大量的基于时间月份延迟的新特征，这里是该项目构造特征最关键的步骤。

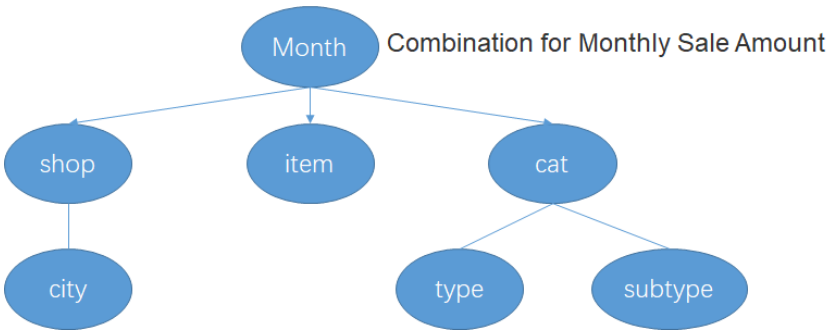

图 5: 基于月份的不同特征组合延迟

2.2.4 从特征关系中构造特殊特征

分析得出价格趋势与销售量有紧密关系,因此构造构造价格趋势变化的连续 6 个月的特征 delta_price_lag，构造每个月的商店收益趋势特征 delta_revenue_lag_1,构造相邻两次售出时间的间隔 item_shop_last_sale，构造某次与第一次售出的时间间隔 item_shop_first_sale 和 item_first_sale 等。

2.3 选择最佳特征

经过以上特征选择后，我们得到了大量的特征，我们需要先用一些模型进行训练，并得到最佳的 top 特征，图 6 为 xgboost 模型得到的前 20 特征，图 7 为 lgbm 模型得到的前 20 特征，图 8 位 catboost 模型得到的特征得分：

	index	feature	importance
0	7	item_cnt_month_lag_1	0.402294
1	12	orders_lag_1	0.145578
2	9	item_cnt_month_lag_3	0.028102
3	5	type_code	0.023219
4	13	orders_lag_2	0.021020
5	8	item_cnt_month_lag_2	0.020901
6	4	item_category_id	0.019423
7	89	item_first_sale	0.018124
8	72	date_item_city_sum_orders_lag_2	0.014545
9	18	date_item_avg_item_cnt_lag_1	0.014342
10	40	date_shop_cat_avg_item_cnt_lag_1	0.012385
11	25	date_shop_avg_item_cnt_lag_1	0.011641
12	6	subtype_code	0.011201
13	88	item_shop_first_sale	0.010436
14	1	shop_id	0.010144
15	14	orders_lag_3	0.008529
16	22	date_item_sum_orders_lag_1	0.007828
17	10	item_cnt_month_lag_6	0.007636
18	2	item_id	0.006787
19	3	city_code	0.006241

图 6: xgboost 模型得到的 top 特征

	index	feature	importance
0	2	item_id	997
1	105	month	940
2	23	date_item_sum_orders_lag_1	921
3	18	date_item_avg_item_cnt_lag_1	845
4	103	delta_price_lag	716
5	1	shop_id	671
6	4	item_category_id	589
7	0	date_block_num	461
8	17	date_avg_item_cnt_lag_1	418
9	48	date_shop_cat_avg_item_cnt_lag_1	392
10	93	date_type_avg_item_cnt_lag_1	342
11	43	date_cat_sum_orders_lag_1	329
12	6	subtype_code	314
13	12	orders_lag_1	295
14	13	orders_lag_2	277
15	53	date_shop_cat_sum_orders_lag_1	263
16	38	date_cat_avg_item_cnt_lag_1	258
17	83	date_item_city_avg_item_cnt_lag_1	247
18	8	item_cnt_month_lag_2	229
19	44	date_cat_sum_orders_lag_2	225

图 7: lightgbm 模型得到的 top 特征

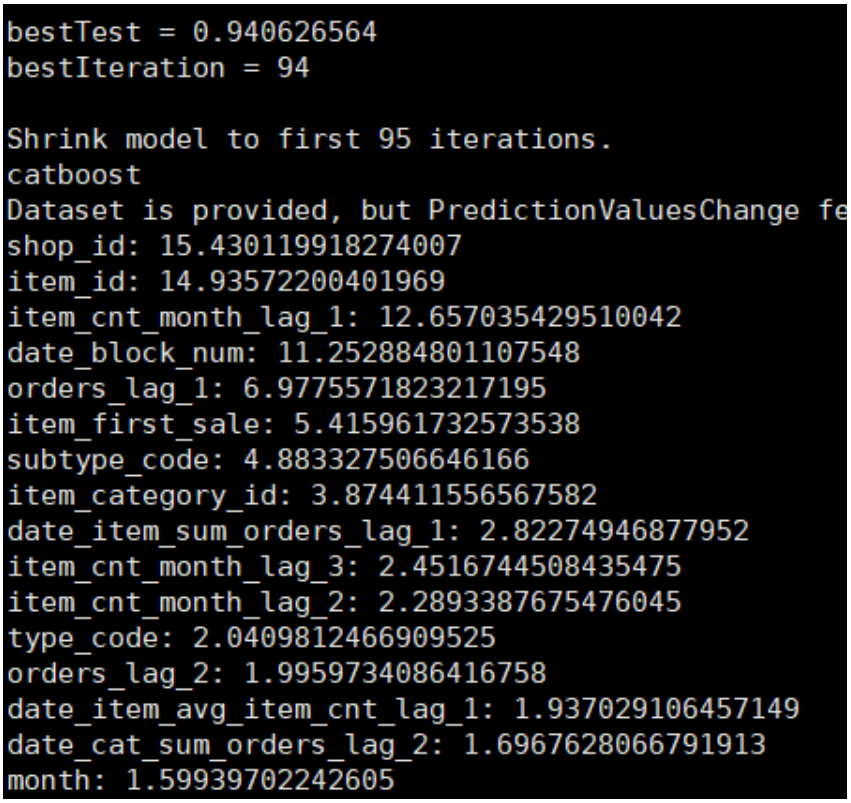

图 8: catboost 模型得到的 top 特征

根据特征得分和重要性，各自剔除掉 10% 左右的低作用特征作为新的训练特征。

2.4 选择模型训练

在这次实验中，我们初步选择了 xgboost，lightgbm，catboost 模型来用上面得到的新特征来进行训练，得到的结果如表 1 所示：

表 1: 各模型分别最佳的测试结果

模型	提交得分
xgboost	0.91145
lightgbm	0.90526
catboost	0.96969

由于提交的结果是与标准结果的均值方差，因此得分越低效果越好。就实验结果我们发现，单独使用以上三个模型，lightgbm 模型得到的结果最佳，catboost 得到的结果最差，实际中 xgboost 模型运行的时间最长，将近是其它两个模型 10 倍的时间，这与模型的原理有关。这里面存在调参的问题，不同模型对应的参数选择对最终的模型预测结果具有一定的影响，在本项目中，我们使用的是 GridSearch 方法遍历搜索每个模型选择的最佳参数，如图 9 所示为 lightgbm 通过 GridSearch 方

法遍历后得到的最佳模型参数。

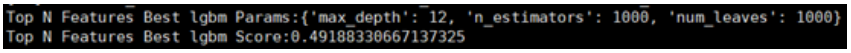

图 9: lightgbm 模型最佳参数

关于参数的选择的问题，在之后章节会继续讨论。

2.5 模型集成

我们分两种集成方法：1. 我们将不同模型得到的预测结果，选择最优的提交结果，再利用线性回归模型将验证集的预测结果作为新的训练集，将对应验证集的真实结果作为训练集的标签，并将对测试集的预测结果作为新的测试集，来用 LinearRegression 模型进行训练，2. 将最好的两个预测结果取平均，得到的结果如表 2 所示，

表 2: 集成后的测试结果

模型	提交得分
LinearRegressiot	0.90689
模型结果平均	0.89647

我们可以发现两个 0.905,0.906 的结果平均后达到了 1% 的提高，效果十分显著。

2.6 特殊处理

最终的预测结果均为小数形式，但是实际的销售额应为整数，我们选择将销售数量小数点后的结果小于某个阈值的小数去掉，或大于某个阈值的结果加 1，以阈值来进行四舍五入，结果如表 3 所示：

表 3: 特殊处理后的测试结果

低阈值	高阈值	提交得分
0.1	无	0.89674

效果虽然没有明显的提高，但是仍然是一个关键的步骤，我们尚在尝试如何选择阈值以及对预测结果的影响。目前这一块仍需要研究如何选择。最终的排名成果如下：

图 10: 比赛结果

3 问题思考与未来工作

- 1. 特征选择问题。我们在实践中发现，当选择不同特征然后用相同的模型来进行训练时，结果差异非常大，当我们细心选择 37 个特征时，得到模型结果接近 0.91，但是用了其他方法得到 100 多个特征时，提交的结果只有 0.96，相差非常大。所以特征选择依然是结果提高一个很关键的地方，这里仍需要继续思考。
- 2. 模型参数。当使用 lightgbm 模型时，不同的参数对结果有些微 (大概 1%) 的影响，由于运行一次的结果非常长，我们仍不能盲目的用 GridSearch 方法去找最佳的特征，还是应该了解模型的原理，选在一个合适的范围中使用 GridSearch 方法才是最重要的
- 3. 集成方法。一种思路是：由于我们使用的 xgboost，lightgbm，catboost 模型本质上都是同一类型模型，我们还应该去尝试不同类型的模型如 SVR，随机森林，岭回归等“真正的回归模型”来预测结果，再来集成，而不是仅仅依靠一个模型的结果。这里估计能有细微的提高。另一种思路是将预测得到的结果以权重方式相加。也是可以尝试的方法。
- 4. 特殊处理。还是之前说的，我们得到的结果是小数，而实际的预测结果应该是整数，我们是否可以从分布，概率的条件将预测结果调整为整数，从而更贴近真实的情况。简单综上所述，我们认为特征工程依然是最大的影响条件，预计能有质的提高，而模型调参以及集成方法能有百分位的提高，特殊处理存在未知，但却是必须要做的一个处理。

参考文献

[1] https://www.kaggle.com/dlarionov/feature-engineering-xgboost

[2] https://github.com/dlarionov/1c/tree/master/submission

[3] https://www.kaggle.com/jagangupta/time-series-basics-exploring-traditional-ts