Predict future sales

Pengcheng Jiang

JiLin University

2021-04-11

Overview

Problem Definition

Data Cleaning

Data analysis

Model

Lightgbm

Problem Definition

Predict future sales

- given: a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company.
- target: predict total sales for every product and store in the next month
- evaluation: Submissions are evaluated by root mean squared error (RMSE)

Data Cleaning

Date

- item_categories.csv:item_category_name item_category_id
- items.csv:item_id item_category_id
- sales_train.csv:date date_block_num shop_id item_id item_price item_cnt_day
- shops.csv:shop_name shop_id
- test.csv:shop_id item_id

take sales as an example

Data Information


```
import pandas as pd
import numpy as np
sales_train = pd.read_csv('data/sales_train.csv')
test =pd.read_csv('data/test.csv')
item = pd.read_csv('data/items.csv')
item_categories = pd.read_csv('data/item_categories.csv')
shop = pd.read_csv('data/shops.csv')
```

```
sales_train.head
```

Data Information


```
<bound method NDFrame.head of                 date  date_block_num  shop_id  item_id  item_price  \
0         02.01.2013               0       59    22154      999.00
1         03.01.2013               0       25     2552      899.00
2         05.01.2013               0       25     2552      899.00
3         06.01.2013               0       25     2554     1709.05
4         15.01.2013               0       25     2555     1099.00
...              ...             ...      ...      ...         ...
2935844   10.10.2015              33       25     7409      299.00
2935845   09.10.2015              33       25     7460      299.00
2935846   14.10.2015              33       25     7459      349.00
2935847   22.10.2015              33       25     7440      299.00
2935848   03.10.2015              33       25     7460      299.00

         item_cnt_day
0                 1.0
1                 1.0
2                -1.0
3                 1.0
4                 1.0
...               ...
2935844           1.0
2935845           1.0
2935846           1.0
2935847           1.0
2935848           1.0

[2935849 rows x 6 columns]>
```

Data Information


```
: print('------information------')
  print(sales_train.info())
```

```
------information------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   date            object
 1   date_block_num  int64
 2   shop_id         int64
 3   item_id         int64
 4   item_price      float64
 5   item_cnt_day    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB
None
```

Missing Value


```
print('-----missing value-----')
print(sales_train.isnull().sum())
```

```
-----missing value-----
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
```

Other data were similarly processed and no missing.

Non Value


```
print('--------nan value------')
print(sales_train.isna().sum())
```

```
--------nan value------
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
```

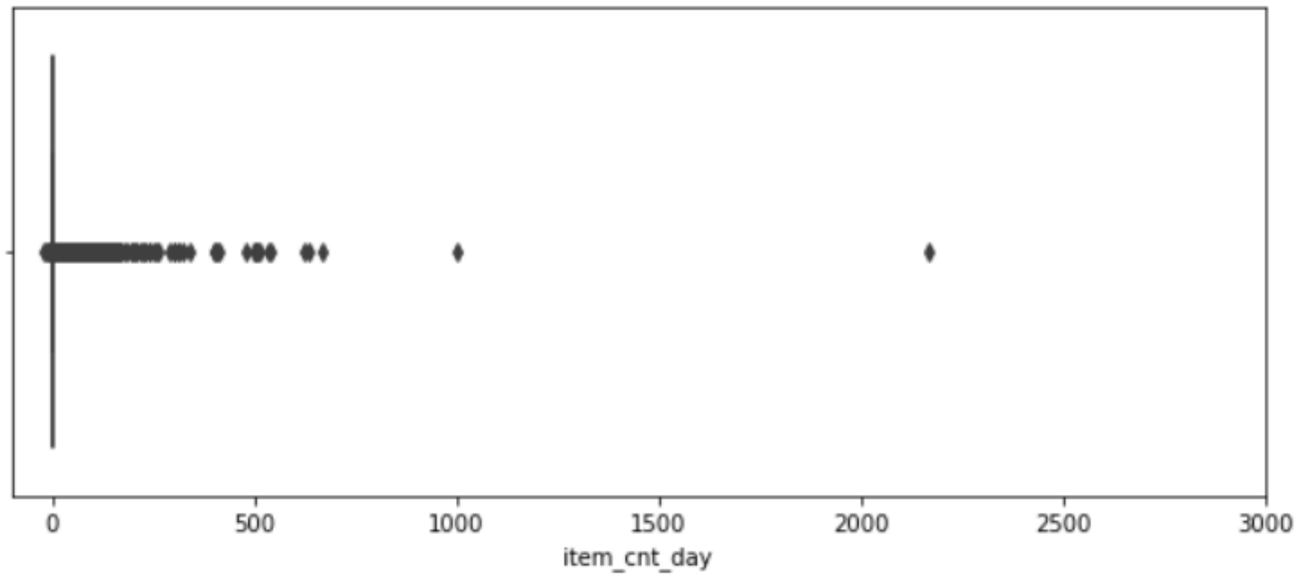
Other data were similarly processed.

Outliers

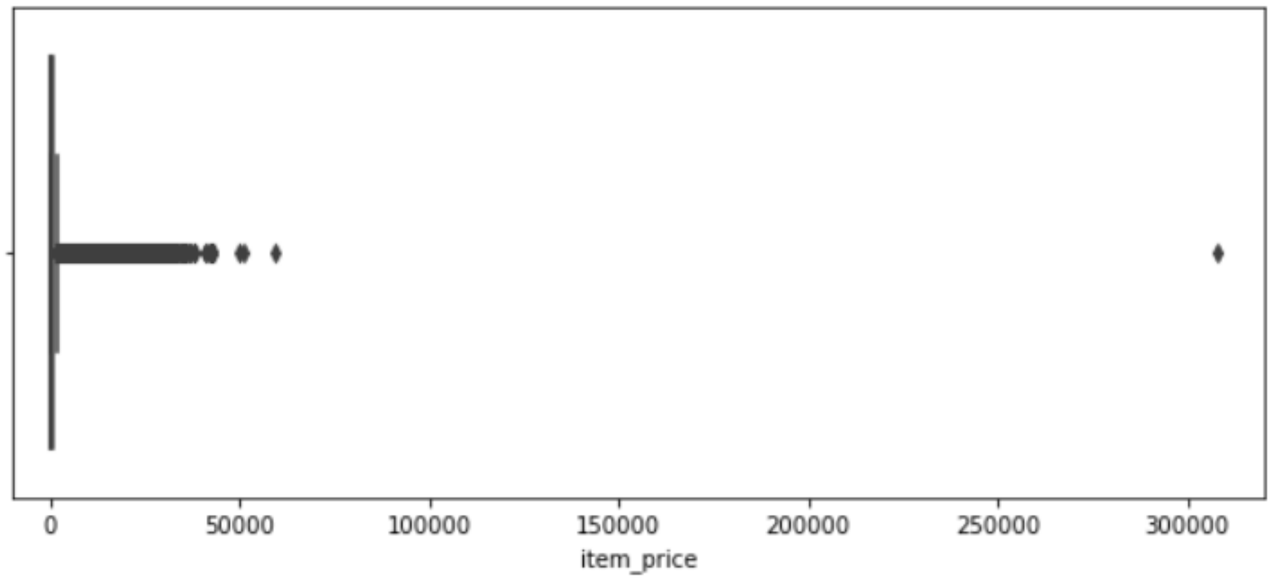

```
:  import matplotlib.pyplot as plt
   import seaborn as sns
   plt.figure(figsize=(10,4))
   plt.xlim(-100,3000)
   sns.boxplot(x = sales_train['item_cnt_day'])
   print('Sale volume outliers:',sales_train['item_cnt_day'][sales_train['item_cnt_day']>1001].unique())
   plt.figure(figsize=(10,4))
   plt.xlim(-10000,320000)
   sns.boxplot(x = sales_train['item_price'])
   print('Sale price outliers:',sales_train['item_price'][sales_train['item_price']>300000].unique())
```

```
Sale volume outliers: [2169.]
Sale price outliers: [307980.]
```

Outliers



Outliers



Negative

Change item whose commodity price is negative to median(median)

```
import matplotlib.pyplot as plt
print(sales_train[sales_train['item_price']<0])
median = sales_train[(sales_train['date_block_num'] == 4)&\
                     (sales_train['shop_id'] == 32)&\
                     (sales_train['item_id'] == 2973)&\
                     (sales_train['item_price']>0)].item_price.median()
sales_train.loc[sales_train['item_price']<0,'item_price'] = median
print(sales_train[sales_train['item_price']<0])
```

```
             date  date_block_num  shop_id  item_id  item_price  item_cnt_day
484683  15.05.2013               4       32     2973        -1.0           1.0
Empty DataFrame
Columns: [date, date_block_num, shop_id, item_id, item_price, item_cnt_day]
Index: []
```

Process Steps

- features engineering
- Model training
- Performance analysis

Data analysis

Shop sales


```
sales_by_shop_id = sales_train.pivot_table(index=['shop_id'],values=['item_cnt_day'],\
                                           columns='date_block_num', aggfunc=np.sum, fill_value=0).reset_index()
#print(sales_by_shop_id)
#每一行是一个商店，列是月数，元素为一个商店一个月的销量
#print(sales_by_shop_id['shop_id'].nunique())#60个商店
sales_by_shop_id.columns = sales_by_shop_id.columns.droplevel().map(str)
sales_by_shop_id = sales_by_shop_id.reset_index(drop=True).rename_axis(None, axis=1)
sales_by_shop_id.columns.values[0] = 'shop_id'
for i in range(27,34):
    print('Not exists in month',i,sales_by_shop_id['shop_id'][sales_by_shop_id.loc[:,'0':str(i)].sum(axis=1)==0].unique())
#上一行筛选出了最新开的商店
for i in range(27,34):
    print('Shop is outdated for month',i,sales_by_shop_id['shop_id'][sales_by_shop_id.loc[:,str(i):].sum(axis=1)==0].unique())
#上一行筛选出了已经关闭的商店
shop2=sales_by_shop_id.iloc[2,1:]
#第一行，1到34列
shop2.plot(legend=True, label="shop sum")
#图为一个商店1-33月份的销量图
```

Objective: To prepare for feature extraction

Shop sales


```
Not exists in month 27 [36]
Not exists in month 28 [36]
Not exists in month 29 [36]
Not exists in month 30 [36]
Not exists in month 31 [36]
Not exists in month 32 [36]
Not exists in month 33 []
Shop is outdated for month 27 [ 0  1  8 11 13 17 23 30 32 40 43]
Shop is outdated for month 28 [ 0  1  8 11 13 17 23 30 32 33 40 43 54]
Shop is outdated for month 29 [ 0  1  8 11 13 17 23 29 30 32 33 40 43 54]
Shop is outdated for month 30 [ 0  1  8 11 13 17 23 29 30 32 33 40 43 54]
Shop is outdated for month 31 [ 0  1  8 11 13 17 23 29 30 32 33 40 43 54]
Shop is outdated for month 32 [ 0  1  8 11 13 17 23 29 30 32 33 40 43 54]
Shop is outdated for month 33 [ 0  1  8 11 13 17 23 27 29 30 32 33 40 43 51 54]
```

Item Information

The categories of items are: large categories, small categories, we separate them, and code them separately to facilitate subsequent feature extraction

```
categories['split'] = categories['item_category_name'].str.split('-')
categories['type'] = categories['split'].map(lambda x:x[0].strip())
categories['subtype'] = categories['split'].map(lambda x:x[1].strip() if len(x)>1 else x[0].strip())
categories = categories[['item_category_id','type','subtype']]
categories.head()
```

Shop Information

Shop information includes: the city where the store is located, the type of store, which we separate and encode separately for subsequent feature extraction

Shop Information


```
from sklearn.preprocessing import LabelEncoder
shops['shop_name'] = shops['shop_name'].apply\
(lambda x: x.lower()).str.replace('[^\w\s]', '').str.replace\
('\d+','').str.strip()
shops['shop_city'] = shops['shop_name'].str.partition(' ')[0]
shops['shop_type'] = shops['shop_name'].apply(lambda x: 'мтрц' if 'мтрц' in x \
                                              else 'трц' if 'трц' in x else 'трк' \
                                              if 'трк' in x else 'тц' if 'тц' in x else 'тк' \
                                              if 'тк' in x else 'NO_DATA')
shops.head()
shops['shop_city_code'] = LabelEncoder().fit_transform(shops['shop_city'])
shops['shop_type_code'] = LabelEncoder().fit_transform(shops['shop_type'])
shops.head()
```

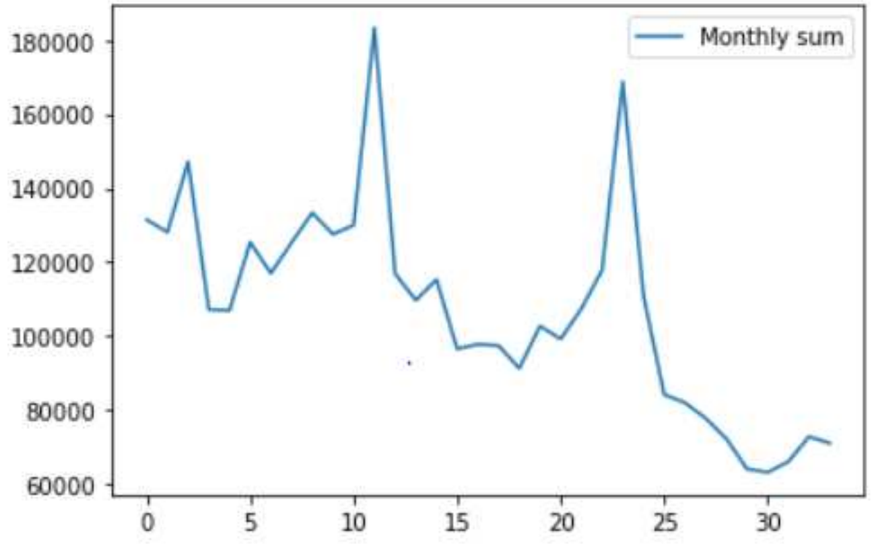
Items Information

The training set contains only the items that the store actually sold that month, for items not sold during the month, you should add them and set them to 0

```
for i in range(34):
    sales = sales_train[sales_train.date_block_num==i]
    matrix.append(np.array(list(product([i], sales.shop_id.unique(), sales.item_id.unique())), dtype='int16'))
#product:将i,shopid,itemid的结合起来。n*m*h
```

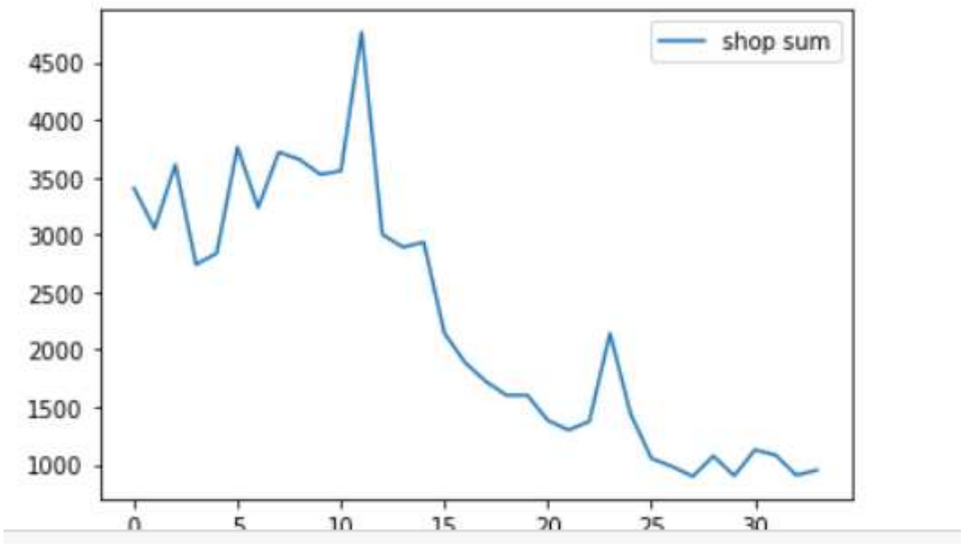
Cartesian product

Monthly total sales



Sales per store

It is known that the city to which the store belongs and the type of store affect sales



Model

Model selection

- GBDT
- Xgboost
- lightgbm
- neural network

Method One

features:shop_id,item_id,item_cnt_month

Method:lightgbm

```
Early stopping, best iteration is:
[495]   training's rmse: 1.20578        valid_1's rmse: 1.12147
```

attention:After some data preprocessing

Method Two

Some historical information needs to be generated by delayed operations. For example, you can use the 0-33 month sales as a historical feature of the 1-34 month (one month delay).

Method Two

- Historical information on monthly sales (per item-store).
- Historical information on the average monthly sales (all merchandise-store) value
- Average monthly sales (per item) and historical characteristics
- Average monthly sales (per store) and historical characteristics
- Average monthly sales (per commodity category) and historical characteristics
- Average monthly sales (commodity category-store) and historical characteristics
- Average and historical characteristics of monthly sales volume (commodity category _ class)
- Average and historical characteristics of monthly sales (commodity-commodity category _ class)
- Average monthly sales (store _ city) and historical characteristics
- Average monthly sales (merchandise-store-city) and historical characteristics
- Trends, price changes over the past six months
- Number of days per month
- Sales beginning and ending

Method Two


```
print([column for column in X_train])
```

```
['date_block_num', 'shop_id', 'item_id', 'item_category_id', 'cat_type_code', 'cat_subtype_code', 'shop_city_code', 'shop_type
tem_cnt_month_lag_1', 'item_cnt_month_lag_2', 'item_cnt_month_lag_3', 'item_cnt_month_lag_6', 'item_cnt_month_lag_12', 'date_a
t_lag_1', 'date_avg_item_cnt_lag_2', 'date_avg_item_cnt_lag_3', 'date_avg_item_cnt_lag_6', 'date_avg_item_cnt_lag_12', 'date_i
em_cnt_lag_1', 'date_item_avg_item_cnt_lag_2', 'date_item_avg_item_cnt_lag_3', 'date_item_avg_item_cnt_lag_6', 'date_item_avg_
ag_12', 'date_shop_avg_item_cnt_lag_1', 'date_shop_avg_item_cnt_lag_2', 'date_shop_avg_item_cnt_lag_3', 'date_shop_avg_item_cn
'date_shop_avg_item_cnt_lag_12', 'date_cat_avg_item_cnt_lag_1', 'date_cat_avg_item_cnt_lag_2', 'date_cat_avg_item_cnt_lag_3',
avg_item_cnt_lag_6', 'date_cat_avg_item_cnt_lag_12', 'date_cat_shop_avg_item_cnt_lag_1', 'date_cat_shop_avg_item_cnt_lag_2',
hop_avg_item_cnt_lag_3', 'date_cat_shop_avg_item_cnt_lag_6', 'date_cat_shop_avg_item_cnt_lag_12', 'date_type_avg_item_cnt_lag_
type_avg_item_cnt_lag_2', 'date_type_avg_item_cnt_lag_3', 'date_type_avg_item_cnt_lag_6', 'date_type_avg_item_cnt_lag_12', 'da
pe_avg_item_cnt_lag_1', 'date_item_type_avg_item_cnt_lag_2', 'date_item_type_avg_item_cnt_lag_3', 'date_item_type_avg_item_cnt
'date_item_type_avg_item_cnt_lag_12', 'date_city_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_2', 'date_city_avg_item_cnt_
ate_city_avg_item_cnt_lag_6', 'date_city_avg_item_cnt_lag_12', 'date_item_city_avg_item_cnt_lag_1', 'date_item_city_avg_item_c
'date_item_city_avg_item_cnt_lag_3', 'date_item_city_avg_item_cnt_lag_6', 'date_item_city_avg_item_cnt_lag_12', 'delta_price_l
h', 'days', 'item_shop_last_sale']
```

Method Two


```
[230]   training's rmse: 0.831437       valid_1's rmse: 0.923975
```

Lightgbm