Part 1 Data Processing and Engineering

December 23, 2021

1 Predict Future Sales Part 1: Data Processing and Engineering

Motivation

This is the final project for the course "How to win a data science competition". In this competition you will work with a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company.

We are asking you to **predict total sales for every product and store in the next month**. By solving this competition you will be able to apply and enhance your data science skills.

You are provided with daily historical sales data. The task is to forecast the total amount of products sold in every shop for the test set. **Note that the list of shops and products slightly changes every month.** Creating a robust model that can handle such situations is part of the challenge.

Source: https://www.kaggle.com/c/competitive-data-science-predict-future-sales/overview

File Description - **sales_train.csv** - the training set. Daily historical data from January 2013 to October 2015. - **test.csv** - the test set. You need to forecast the sales for these shops and products for November 2015. - **sample_submission.csv** - a sample submission file in the correct format. - **items.csv** - supplemental information about the items/products. - **item_categories.csv** - supplemental information about the items categories. - **shops.csv**- supplemental information about the shops.

2 Table of contents

[1]:
```
#load packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from math import ceil
import copy

import os
import time
import gc
import pickle
from tqdm import tqdm #progress bar
from itertools import product
import warnings
warnings.filterwarnings("ignore")
from IPython.display import clear_output

from sklearn.feature_extraction import text
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA, TruncatedSVD, NMF
from sklearn.linear_model import LinearRegression, Ridge, Lasso, Lars,␣
 ↪ElasticNet
# from sklearn.externals import joblib
import joblib

import lightgbm as lgb
import xgboost as xgb
```

3 1. Importing data

[3]:
```
data_folder = "./data/"
sales_train = pd.read_csv(os.path.join(data_folder, "sales_train.csv"))
items = pd.read_csv(os.path.join(data_folder, "items.csv"))
test = pd.read_csv(os.path.join(data_folder, "test.csv")).set_index('ID')
item_categories = pd.read_csv(os.path.join(data_folder, "item_categories.csv"))
shops = pd.read_csv(os.path.join(data_folder, "shops.csv"))
```

[11]:
```
sales_train['sales'] = sales_train['item_cnt_day'] * sales_train['item_price']
print('shape: \n', sales_train.shape)
print('-----------------------------------------------------------------')
print('dtype: \n', sales_train.dtypes)
print('-----------------------------------------------------------------')
print('# of unique values \n', sales_train.nunique())
print('-----------------------------------------------------------------')
sales_train.head()
```

```
shape:
 (2935849, 7)
-----------------------------------------------------------------
dtype:
 date               object
date_block_num      int64
shop_id             int64
item_id             int64
item_price        float64
item_cnt_day      float64
sales             float64
dtype: object
-----------------------------------------------------------------
# of unique values
 date               1034
date_block_num       34
shop_id              60
item_id           21807
item_price        19993
item_cnt_day        198
sales             24771
dtype: int64
-----------------------------------------------------------------
```

[11]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	sales
0	02.01.2013	0	59	22154	999.00	1.0	999.00
1	03.01.2013	0	25	2552	899.00	1.0	899.00
2	05.01.2013	0	25	2552	899.00	-1.0	-899.00
3	06.01.2013	0	25	2554	1709.05	1.0	1709.05
4	15.01.2013	0	25	2555	1099.00	1.0	1099.00

[6]:
```
sales_train.describe()
```

[6]:
```
       date_block_num       shop_id       item_id    item_price  item_cnt_day  \
count    2.935849e+06  2.935849e+06  2.935849e+06  2.935849e+06  2.935849e+06
mean     1.456991e+01  3.300173e+01  1.019723e+04  8.908532e+02  1.242641e+00
std      9.422988e+00  1.622697e+01  6.324297e+03  1.729800e+03  2.618834e+00
min      0.000000e+00  0.000000e+00  0.000000e+00 -1.000000e+00 -2.200000e+01
25%      7.000000e+00  2.200000e+01  4.476000e+03  2.490000e+02  1.000000e+00
50%      1.400000e+01  3.100000e+01  9.343000e+03  3.990000e+02  1.000000e+00
75%      2.300000e+01  4.700000e+01  1.568400e+04  9.990000e+02  1.000000e+00
max      3.300000e+01  5.900000e+01  2.216900e+04  3.079800e+05  2.169000e+03

              sales
count  2.935849e+06
mean   1.157732e+03
std    5.683604e+03
min   -6.897000e+04
25%    2.490000e+02
50%    4.490000e+02
75%    1.078200e+03
max    1.829990e+06
```

Note The minimum item_price is negative, which is strange.

[12]:
```
print('shape: \n', items.shape)
print('-----------------------------------------------------------------')
print('# of unique values \n', items.nunique())
print('-----------------------------------------------------------------')
items.head()
```

```
shape:
 (22170, 3)
-----------------------------------------------------------------
# of unique values
 item_name           22170
item_id              22170
item_category_id        84
dtype: int64
-----------------------------------------------------------------
```

[12]:
```
                                                item_name  item_id  \
0          !              (   .)         D         0
1  !ABBYY FineReader 12 Professional Edition Full…         1
2      ***           (UNV)                      D         2
3    ***          (Univ)                       D         3
4        ***     (    )                        D         4

   item_category_id
0                40
1                76
```

```
2                40
3                40
4                40
```

[13]:
```
print('shape: \n', test.shape)
print('----------------------------------------------------------------')
print('# of unique values \n', test.nunique())
print('----------------------------------------------------------------')
test.head()
```

```
shape:
 (214200, 2)
----------------------------------------------------------------
# of unique values
 shop_id      42
item_id     5100
dtype: int64
----------------------------------------------------------------
```

[13]:

ID	shop_id	item_id
0	5	5037
1	5	5320
2	5	5233
3	5	5232
4	5	5268

[14]:
```
print('shape: \n', item_categories.shape)
print('----------------------------------------------------------------')
print('# of unique values \n', item_categories.nunique())
print('----------------------------------------------------------------')
item_categories.head()
```

```
shape:
 (84, 2)
----------------------------------------------------------------
# of unique values
 item_category_name    84
item_category_id       84
dtype: int64
----------------------------------------------------------------
```

[14]:

	item_category_name	item_category_id
0	PC - /	0
1	- PS2	1
2	- PS3	2
3	- PS4	3
4	- PSP	4

[15]:
```
print('shape: \n', shops.shape)
print('----------------------------------------------------------------------')
print('# of unique values \n', shops.nunique())
print('----------------------------------------------------------------------')
shops.head()
```

```
shape:
 (60, 2)
----------------------------------------------------------------------
# of unique values
 shop_name    60
shop_id      60
dtype: int64
----------------------------------------------------------------------
```

[15]:

	shop_name	shop_id
0	! , 56	0
1	! " "	1
2	" "	2
3	" - "	3
4	" "	4

4 2. Preprocessing data

In test dataset, each ID represents a unique pair of (shop_id, item_id)

4.1 2.1 Extracting categorical features from Russian item names

[16]:
```
shops.head()
```

[16]:

	shop_name	shop_id
0	! , 56	0
1	! " "	1
2	" "	2
3	" - "	3
4	" "	4

[17]:
```
shops.loc[shops.shop_name == '          "7 "', 'shop_name'] = '          "7 "'␣
↪#delete space between
shops['city'] = shops['shop_name'].str.split(' ').map(lambda x: x[0])
shops.loc[shops.city == '!    ', 'city'] = '    '
shops['city_code'] = LabelEncoder().fit_transform(shops['city'])
shops = shops[['shop_id', 'city_code']]
shops.head()
```

[17]:

	shop_id	city_code
0	0	29

```
1          1          29
2          2           0
3          3           1
4          4           2
```

[18]:
```
item_categories['split'] = item_categories['item_category_name'].str.split('-')
item_categories['type'] = item_categories['split'].map(lambda x: x[0].strip())
item_categories['type_code'] = LabelEncoder().
 ↪fit_transform(item_categories['type'])
item_categories['subtype'] = item_categories['split'].map(lambda x: x[1].
 ↪strip() if len(x) >1 else x[0].strip())
item_categories['subtype_code'] = LabelEncoder().
 ↪fit_transform(item_categories['subtype'])
item_categories = item_categories[['item_category_id', 'type_code',␣
 ↪'subtype_code']]
item_categories.head()
```

[18]:
```
   item_category_id  type_code  subtype_code
0                 0          0            29
1                 1          1             9
2                 2          1            10
3                 3          1            11
4                 4          1            13
```

[19]:
```
items.drop('item_name', axis=1, inplace=True)
items.head()
```

[19]:
```
   item_id  item_category_id
0        0                40
1        1                76
2        2                40
3        3                40
4        4                40
```

4.2 2.2 Creating text features with TF-IDF

[20]:
```
items1= pd.read_csv(os.path.join(data_folder, 'items.csv'))
item_categories1 = pd.read_csv(os.path.join(data_folder, 'item_categories.csv'))
shops1 = pd.read_csv(os.path.join(data_folder, 'shops.csv'))
```

[21]:
```
#for items1 names
nb_features = 25
tfidf = text.TfidfVectorizer(max_features=nb_features)
items1['item_name_len'] = items1['item_name'].map(len) #name length
items1['item_name_wc'] = items1['item_name'].map(lambda x: len(str(x).split('␣
 ↪'))) #word count
txtFeatures = pd.DataFrame(tfidf.fit_transform(items1['item_name']).toarray())
```

```
cols = txtFeatures.columns
for i in range(nb_features):
    items1['item_name_tfidf_'+ str(i)] = txtFeatures[cols[i]]

items1.head()
#txtFeatures.head()
```

[21]:

```
                                        item_name  item_id  \
0          !              (   .)         D        0
1  !ABBYY FineReader 12 Professional Edition Full…        1
2     ***           (UNV)                    D        2
3    ***          (Univ)                    D        3
4        ***     (   )                      D        4

   item_category_id  item_name_len  item_name_wc  item_name_tfidf_0  \
0                40             41            14                0.0
1                76             68             9                0.0
2                40             45            26                0.0
3                40             47            26                0.0
4                40             43            25                0.0

   item_name_tfidf_1  item_name_tfidf_2  item_name_tfidf_3  item_name_tfidf_4  \
0                0.0                0.0                0.0                0.0
1                0.0                0.0                0.0                0.0
2                0.0                0.0                0.0                0.0
3                0.0                0.0                0.0                0.0
4                0.0                0.0                0.0                0.0

   …  item_name_tfidf_15  item_name_tfidf_16  item_name_tfidf_17  \
0  …                 0.0            0.000000                 0.0
1  …                 0.0            0.403761                 0.0
2  …                 0.0            0.000000                 0.0
3  …                 0.0            0.000000                 0.0
4  …                 0.0            0.000000                 0.0

   item_name_tfidf_18  item_name_tfidf_19  item_name_tfidf_20  \
0                 0.0                 0.0                 0.0
1                 0.0                 0.0                 0.0
2                 0.0                 0.0                 0.0
3                 0.0                 0.0                 0.0
4                 0.0                 0.0                 0.0

   item_name_tfidf_21  item_name_tfidf_22  item_name_tfidf_23  \
0                 0.0                 0.0                 0.0
1                 0.0                 0.0                 0.0
2                 0.0                 0.0                 0.0
3                 0.0                 0.0                 0.0
```

```
4                 0.0                 0.0                 0.0

   item_name_tfidf_24
0            0.000000
1            0.483839
2            0.000000
3            0.000000
4            0.000000

[5 rows x 30 columns]
```

[22]:
```
#for shops1 names
nb_features = 25
tfidf = text.TfidfVectorizer(max_features=nb_features)
shops1['shop_name_len'] = shops1['shop_name'].map(len)
shops1['shop_name_wc'] = shops1['shop_name'].map(lambda x: len(str(x).split('␣
↪')))
txtFeatures = pd.DataFrame(tfidf.fit_transform(shops1['shop_name']).toarray())
cols = txtFeatures.columns
for i in range(nb_features):
    shops1['shop_name_tfidf_' + str(i)] = txtFeatures[cols[i]]

shops1.head()
```

[22]:
```
                      shop_name  shop_id  shop_name_len  shop_name_wc  \
0   !           , 56           0             29             4
1   !      "      "           1             29             4
2                      "   "        2             16             3
3          "    -    "        3             30             3
4               "      "        4             24             4

   shop_name_tfidf_0  shop_name_tfidf_1  shop_name_tfidf_2  shop_name_tfidf_3  \
0                0.0                0.0                0.0                0.0
1                0.0                0.0                0.0                0.0
2                0.0                0.0                0.0                0.0
3                0.0                0.0                0.0                0.0
4                0.0                0.0                0.0                0.0

   shop_name_tfidf_4  shop_name_tfidf_5  …  shop_name_tfidf_15  \
0                0.0                0.0  …                 0.0
1                0.0                0.0  …                 0.0
2                0.0                0.0  …                 0.0
3                0.0                0.0  …                 0.0
4                0.0                0.0  …                 0.0

   shop_name_tfidf_16  shop_name_tfidf_17  shop_name_tfidf_18  \
0                 0.0                 0.0                 0.0
```

```
1                  0.0                  0.0                  0.0
2                  0.0                  0.0                  0.0
3                  0.0                  0.0                  0.0
4                  0.0                  0.0                  0.0

   shop_name_tfidf_19  shop_name_tfidf_20  shop_name_tfidf_21  \
0                 0.0                 0.0            0.000000
1                 0.0                 0.0            0.322815
2                 0.0                 0.0            0.498580
3                 1.0                 0.0            0.000000
4                 0.0                 0.0            0.423972

   shop_name_tfidf_22  shop_name_tfidf_23  shop_name_tfidf_24
0                 0.0            0.000000            1.000000
1                 0.0            0.689588            0.648274
2                 0.0            0.000000            0.000000
3                 0.0            0.000000            0.000000
4                 0.0            0.000000            0.000000

[5 rows x 29 columns]
```

4.3 2.3 Check duplicates

[24]:
```
#sales_train
cols = ['date', 'date_block_num', 'shop_id', 'item_id','item_cnt_day']
print(sales_train.duplicated(subset=cols).value_counts())
sales_train.drop_duplicates(subset=cols, inplace=True)
# sales_train.head()
```

```
False    2935825
True          24
dtype: int64
```

There are 24 duplicated rows in sales_train data.

4.4 2.4 Preprocessing for data engineering

[25]:
```
train1 = []
cols = ['date_block_num', 'shop_id', 'item_id']
for block_num in range(34): #range of date_block_num is 0~33,␣
↪max(sales_train['date_block_num'])
    temp = sales_train[sales_train.date_block_num == block_num]
    #iterate all combination of block_num, shop_id, item_id
    train1.append(np.array(list(product([block_num],temp['shop_id'].
↪unique(),temp['item_id'].unique()))))

train1 = pd.DataFrame(np.vstack(train1), columns=cols)
train1['date_block_num'] = train1['date_block_num'].astype(np.int8)
```

```
train1['shop_id'] = train1['shop_id'].astype(np.int8)
train1['item_id'] = train1['item_id'].astype(np.int16)
train1.sort_values(cols, inplace=True)
train1.head()
```

[25]:

	date_block_num	shop_id	item_id
139255	0	0	19
141495	0	0	27
144968	0	0	28
142661	0	0	29
138947	0	0	32

[26]:
```
temp = sales_train.groupby(['date_block_num', 'shop_id', 'item_id']).
 ↪agg({'item_cnt_day': ['sum']})
temp.columns = ['item_cnt_month']
temp.reset_index(inplace=True)

print(temp.shape)
temp.head()
```

```
(1609124, 4)
```

[26]:

	date_block_num	shop_id	item_id	item_cnt_month
0	0	0	32	6.0
1	0	0	33	3.0
2	0	0	35	1.0
3	0	0	43	1.0
4	0	0	51	2.0

[27]:
```
train1 = pd.merge(train1, temp, on=cols, how='left')
train1['item_cnt_month'] = (train1['item_cnt_month'].fillna(0).clip(0,20).
 ↪astype(np.float16))

print(train1.shape)
train1.head()
```

```
(10913850, 4)
```

[27]:

	date_block_num	shop_id	item_id	item_cnt_month
0	0	0	19	0.0
1	0	0	27	0.0
2	0	0	28	0.0
3	0	0	29	0.0
4	0	0	32	6.0

We are using sales' results from Jan.2013 to Oct.2015 to predict future sales of Nov.2015, so we add the test set to the training set by adding a 34th `date_block_num`.

[28]:
```
test['date_block_num'] = 34
test['date_block_num'] = test['date_block_num'].astype(np.int8)
test['shop_id'] = test['shop_id'].astype(np.int8)
test['item_id'] = test['item_id'].astype(np.int16)
test.head()

train2 = pd.concat([train1,test],ignore_index=True, sort=False, keys=cols)
train2.fillna(0, inplace=True)
print(train2.shape)

train2.tail()
```

```
(11128050, 4)
```

[28]:

	date_block_num	shop_id	item_id	item_cnt_month
11128045	34	45	18454	0.0
11128046	34	45	16188	0.0
11128047	34	45	15757	0.0
11128048	34	45	19648	0.0
11128049	34	45	969	0.0

4.5 2.5 Create lagged features

We want to capture past behavior of several item-shop, item category-shop or item-city pairings (among others):

- *date_avg_item_cnt*
- *date_item_avg_item_cnt*
- *date_shop_avg_item_cnt*
- *date_cat_avg_item_cnt*
- *date_shop_cat_avg_item_cnt*
- *date_shop_subtype_avg_item_cnt*
- *date_city_avg_item_cnt*
- *date_shop_type_avg_item_cnt*
- *date_item_city_avg_item_cnt*
- *date_type_avg_item_cnt*
- *date_subtype_avg_item_cnt*

[29]:
```
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
```

[30]:
```
shops.head()
```

[30]:
```
   shop_id  city_code
0        0         29
1        1         29
2        2          0
3        3          1
4        4          2
```

[31]:
```
#merge sales_train, test, shops, items, item_categories
train3 = pd.merge(train2, shops, on=['shop_id'], how='left')
train3 = pd.merge(train3, items, on=['item_id'], how='left')
train3 = pd.merge(train3, item_categories, on=['item_category_id'], how='left')

train3.head()
```

[31]:
```
   date_block_num  shop_id  item_id  item_cnt_month  city_code  \
0               0        0       19             0.0         29
1               0        0       27             0.0         29
2               0        0       28             0.0         29
3               0        0       29             0.0         29
4               0        0       32             6.0         29

   item_category_id  type_code  subtype_code
0                40         11             4
1                19          5            10
2                30          8            55
3                23          5            16
4                40         11             4
```

[32]:
```
########## 1. Create 'date_avg_item_cnt'
temp = train3.groupby(['date_block_num']).agg({'item_cnt_month': ['mean']})
temp.columns = ['date_avg_item_cnt']          ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num'], how='left')
train3 = lag_feature(train3, [1], 'date_avg_item_cnt')
train3.drop(['date_avg_item_cnt'], axis=1, inplace=True)
train3.head()
```

[32]:
```
   date_block_num  shop_id  item_id  item_cnt_month  city_code  \
0               0        0       19             0.0         29
1               0        0       27             0.0         29
2               0        0       28             0.0         29
3               0        0       29             0.0         29
4               0        0       32             6.0         29

   item_category_id  type_code  subtype_code  date_avg_item_cnt_lag_1
```

```
0                40          11              4                     NaN
1                19           5             10                     NaN
2                30           8             55                     NaN
3                23           5             16                     NaN
4                40          11              4                     NaN
```

[33]:
```
# ########## 1. Create 'date_avg_item_cnt'
# temp = train3.groupby(['date_block_num']).agg({'item_cnt_month': ['mean']})
# temp.columns = ['date_avg_item_cnt']          ###
# temp.reset_index(inplace=True)
# train3 = pd.merge(train3, temp, on=['date_block_num'], how='left')
# train3 = lag_feature(train3, [1], 'date_avg_item_cnt')
# train3.drop(['date_avg_item_cnt'], axis=1, inplace=True)

########## 2. Create 'date_item_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'item_id']).agg({'item_cnt_month':␣
↪['mean']})
temp.columns = ['date_item_avg_item_cnt']     ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num','item_id'], how='left')
train3 = lag_feature(train3, [1, 2, 3, 6, 12], 'date_item_avg_item_cnt')
train3.drop(['date_item_avg_item_cnt'], axis=1, inplace=True)

########## 3. Create 'date_shop_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'shop_id']).agg({'item_cnt_month':␣
↪['mean']})
temp.columns = ['date_shop_avg_item_cnt']     ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num','shop_id'], how='left')
train3 = lag_feature(train3, [1, 2, 3, 6, 12], 'date_shop_avg_item_cnt')
train3.drop(['date_shop_avg_item_cnt'], axis=1, inplace=True)

########## 4. Create 'date_cat_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'item_category_id']).
↪agg({'item_cnt_month': ['mean']})
temp.columns = ['date_cat_avg_item_cnt']      ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num','item_category_id'],␣
↪how='left')
train3 = lag_feature(train3, [1], 'date_cat_avg_item_cnt')
train3.drop(['date_cat_avg_item_cnt'], axis=1, inplace=True)

########## 5. Create 'date_shop_cat_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'shop_id', 'item_category_id']).
↪agg({'item_cnt_month': ['mean']})
temp.columns = ['date_shop_cat_avg_item_cnt']    ###
temp.reset_index(inplace=True)
```

```
train3 = pd.merge(train3, temp, on=['date_block_num', 'shop_id',␣
↪'item_category_id'], how='left')
train3 = lag_feature(train3, [1], 'date_shop_cat_avg_item_cnt')
train3.drop(['date_shop_cat_avg_item_cnt'], axis=1, inplace=True)

########## 6. Create 'date_shop_type_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'shop_id', 'type_code']).
↪agg({'item_cnt_month': ['mean']})
temp.columns = ['date_shop_type_avg_item_cnt']    ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num', 'shop_id', 'type_code'],␣
↪how='left')
train3 = lag_feature(train3, [1], 'date_shop_type_avg_item_cnt')
train3.drop(['date_shop_type_avg_item_cnt'], axis=1, inplace=True)

########## 7. Create 'date_type_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'type_code']).agg({'item_cnt_month':␣
↪['mean']})
temp.columns = ['date_type_avg_item_cnt']     ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num', 'type_code'], how='left')
train3 = lag_feature(train3, [1], 'date_type_avg_item_cnt')
train3.drop(['date_type_avg_item_cnt'], axis=1, inplace=True)

########## 8. Create 'date_city_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'city_code']).agg({'item_cnt_month':␣
↪['mean']})
temp.columns = ['date_city_avg_item_cnt']     ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num', 'city_code'], how='left')
train3 = lag_feature(train3, [1], 'date_city_avg_item_cnt')
train3.drop(['date_city_avg_item_cnt'], axis=1, inplace=True)

########## 9. Create 'date_shop_subtype_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'shop_id', 'subtype_code']).
↪agg({'item_cnt_month': ['mean']})
temp.columns = ['date_shop_subtype_avg_item_cnt']   ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num', 'shop_id',␣
↪'subtype_code'], how='left')
train3 = lag_feature(train3, [1], 'date_shop_subtype_avg_item_cnt')
train3.drop(['date_shop_subtype_avg_item_cnt'], axis=1, inplace=True)

########## 10. Create 'date_item_city_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'item_id', 'city_code']).
↪agg({'item_cnt_month': ['mean']})
```

```
temp.columns = ['date_item_city_avg_item_cnt']     ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num', 'item_id', 'city_code'],␣
↪how='left')
train3 = lag_feature(train3, [1], 'date_item_city_avg_item_cnt')
train3.drop(['date_item_city_avg_item_cnt'], axis=1, inplace=True)

########## 11. Create 'date_subtype_avg_item_cnt'
temp = train3.groupby(['date_block_num', 'subtype_code']).agg({'item_cnt_month':
↪ ['mean']})
temp.columns = ['date_subtype_avg_item_cnt']     ###
temp.reset_index(inplace=True)
train3 = pd.merge(train3, temp, on=['date_block_num', 'subtype_code'],␣
↪how='left')
train3 = lag_feature(train3, [1], 'date_subtype_avg_item_cnt')
train3.drop(['date_subtype_avg_item_cnt'], axis=1, inplace=True)
train3.head()
```

[33]:

	date_block_num	shop_id	item_id	item_cnt_month	city_code	\
0	0	0	19	0.0	29	
1	0	0	27	0.0	29	
2	0	0	28	0.0	29	
3	0	0	29	0.0	29	
4	0	0	32	6.0	29	

	item_category_id	type_code	subtype_code	date_avg_item_cnt_lag_1	\
0	40	11	4	NaN	
1	19	5	10	NaN	
2	30	8	55	NaN	
3	23	5	16	NaN	
4	40	11	4	NaN	

	date_item_avg_item_cnt_lag_1	…	date_shop_avg_item_cnt_lag_6	\
0	NaN	…	NaN	
1	NaN	…	NaN	
2	NaN	…	NaN	
3	NaN	…	NaN	
4	NaN	…	NaN	

	date_shop_avg_item_cnt_lag_12	date_cat_avg_item_cnt_lag_1	\
0	NaN	NaN	
1	NaN	NaN	
2	NaN	NaN	
3	NaN	NaN	
4	NaN	NaN	

	date_shop_cat_avg_item_cnt_lag_1	date_shop_type_avg_item_cnt_lag_1	\

```
0                                NaN                                NaN
1                                NaN                                NaN
2                                NaN                                NaN
3                                NaN                                NaN
4                                NaN                                NaN

   date_type_avg_item_cnt_lag_1  date_city_avg_item_cnt_lag_1  \
0                           NaN                           NaN
1                           NaN                           NaN
2                           NaN                           NaN
3                           NaN                           NaN
4                           NaN                           NaN

   date_shop_subtype_avg_item_cnt_lag_1  date_item_city_avg_item_cnt_lag_1  \
0                                   NaN                                NaN
1                                   NaN                                NaN
2                                   NaN                                NaN
3                                   NaN                                NaN
4                                   NaN                                NaN

   date_subtype_avg_item_cnt_lag_1
0                              NaN
1                              NaN
2                              NaN
3                              NaN
4                              NaN

[5 rows x 27 columns]
```

[34]:
```
train3.columns
```

[34]:
```
Index(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month', 'city_code',
       'item_category_id', 'type_code', 'subtype_code',
       'date_avg_item_cnt_lag_1', 'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6',
       'date_shop_avg_item_cnt_lag_12', 'date_cat_avg_item_cnt_lag_1',
       'date_shop_cat_avg_item_cnt_lag_1', 'date_shop_type_avg_item_cnt_lag_1',
       'date_type_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_1',
       'date_shop_subtype_avg_item_cnt_lag_1',
       'date_item_city_avg_item_cnt_lag_1', 'date_subtype_avg_item_cnt_lag_1'],
      dtype='object')
```

[35]:
```
# save data to local disk
output = open('./data_processed/train3.pkl', 'wb')
```

```
pickle.dump(train3, output) #764M

output.close()
# train3.to_csv('./data_processed/train3.csv') #1.3G
```

4.6 2.6 Creating time series trend features

[36]:
```
#Trend features for price
temp = sales_train.groupby(['item_id']).agg({'item_price': ['mean']})
temp.columns = ['item_avg_item_price']
temp.reset_index(inplace=True)
train4 = pd.merge(train3, temp, on=['item_id'], how='left')

temp = sales_train.groupby(['date_block_num', 'item_id']).agg({'item_price':␣
↪['mean']})
temp.columns = ['date_item_avg_item_price']
temp.reset_index(inplace=True)
train4 = pd.merge(train4, temp, on=['date_block_num','item_id'], how='left')

lags=[1,2,3,4,5,6]
train4 = lag_feature(train4, lags, 'date_item_avg_item_price')

train4.head()
```

[36]:

	date_block_num	shop_id	item_id	item_cnt_month	city_code	\
0	0	0	19	0.0	29	
1	0	0	27	0.0	29	
2	0	0	28	0.0	29	
3	0	0	29	0.0	29	
4	0	0	32	6.0	29	

	item_category_id	type_code	subtype_code	date_avg_item_cnt_lag_1	\
0	40	11	4	NaN	
1	19	5	10	NaN	
2	30	8	55	NaN	
3	23	5	16	NaN	
4	40	11	4	NaN	

	date_item_avg_item_cnt_lag_1	…	date_item_city_avg_item_cnt_lag_1	\
0	NaN	…	NaN	
1	NaN	…	NaN	
2	NaN	…	NaN	
3	NaN	…	NaN	
4	NaN	…	NaN	

	date_subtype_avg_item_cnt_lag_1	item_avg_item_price	\
0	NaN	28.000000	

```
1                              NaN          1461.228571
2                              NaN           310.010465
3                              NaN          1759.285714
4                              NaN           249.629240

   date_item_avg_item_price  date_item_avg_item_price_lag_1  \
0                 28.000000                             NaN
1               2325.000000                             NaN
2                549.000000                             NaN
3               2397.500000                             NaN
4                338.110349                             NaN

   date_item_avg_item_price_lag_2  date_item_avg_item_price_lag_3  \
0                             NaN                             NaN
1                             NaN                             NaN
2                             NaN                             NaN
3                             NaN                             NaN
4                             NaN                             NaN

   date_item_avg_item_price_lag_4  date_item_avg_item_price_lag_5  \
0                             NaN                             NaN
1                             NaN                             NaN
2                             NaN                             NaN
3                             NaN                             NaN
4                             NaN                             NaN

   date_item_avg_item_price_lag_6
0                             NaN
1                             NaN
2                             NaN
3                             NaN
4                             NaN

[5 rows x 35 columns]
```

[37]:
```
train4.columns
```

[37]:
```
Index(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month', 'city_code',
       'item_category_id', 'type_code', 'subtype_code',
       'date_avg_item_cnt_lag_1', 'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6',
       'date_shop_avg_item_cnt_lag_12', 'date_cat_avg_item_cnt_lag_1',
       'date_shop_cat_avg_item_cnt_lag_1', 'date_shop_type_avg_item_cnt_lag_1',
       'date_type_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_1',
```

```
       'date_shop_subtype_avg_item_cnt_lag_1',
       'date_item_city_avg_item_cnt_lag_1', 'date_subtype_avg_item_cnt_lag_1',
       'item_avg_item_price', 'date_item_avg_item_price',
       'date_item_avg_item_price_lag_1', 'date_item_avg_item_price_lag_2',
       'date_item_avg_item_price_lag_3', 'date_item_avg_item_price_lag_4',
       'date_item_avg_item_price_lag_5', 'date_item_avg_item_price_lag_6'],
      dtype='object')
```

[41]:
```
for i in lags:
    train4['delta_price_lag_'+str(i)] = \
  ␣
↪(train4['date_item_avg_item_price_lag_'+str(i)]-train4['item_avg_item_price'])␣
↪/ train4['item_avg_item_price']

def select_trend(row):
    for i in lags:
        if row['delta_price_lag_'+str(i)]:
            return row['delta_price_lag_'+str(i)]
    return 0

train4['delta_price_lag'] = train4.apply(select_trend, axis=1)
train4['delta_price_lag'].fillna(0, inplace=True)

train4.head()
```

[41]:

	date_block_num	shop_id	item_id	item_cnt_month	city_code	\
0	0	0	19	0.0	29	
1	0	0	27	0.0	29	
2	0	0	28	0.0	29	
3	0	0	29	0.0	29	
4	0	0	32	6.0	29	

	item_category_id	type_code	subtype_code	date_avg_item_cnt_lag_1	\
0	40	11	4	NaN	
1	19	5	10	NaN	
2	30	8	55	NaN	
3	23	5	16	NaN	
4	40	11	4	NaN	

	date_item_avg_item_cnt_lag_1	…	date_item_avg_item_price_lag_4	\
0	NaN	…	NaN	
1	NaN	…	NaN	
2	NaN	…	NaN	
3	NaN	…	NaN	
4	NaN	…	NaN	

	date_item_avg_item_price_lag_5	date_item_avg_item_price_lag_6	\

```
0                              NaN                              NaN
1                              NaN                              NaN
2                              NaN                              NaN
3                              NaN                              NaN
4                              NaN                              NaN

   delta_price_lag_1  delta_price_lag_2  delta_price_lag_3  delta_price_lag_4  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

   delta_price_lag_5  delta_price_lag_6  delta_price_lag
0                NaN                NaN              0.0
1                NaN                NaN              0.0
2                NaN                NaN              0.0
3                NaN                NaN              0.0
4                NaN                NaN              0.0

[5 rows x 42 columns]
```

[42]:
```
train4.columns
```

[42]:
```
Index(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month', 'city_code',
       'item_category_id', 'type_code', 'subtype_code',
       'date_avg_item_cnt_lag_1', 'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6',
       'date_shop_avg_item_cnt_lag_12', 'date_cat_avg_item_cnt_lag_1',
       'date_shop_cat_avg_item_cnt_lag_1', 'date_shop_type_avg_item_cnt_lag_1',
       'date_type_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_1',
       'date_shop_subtype_avg_item_cnt_lag_1',
       'date_item_city_avg_item_cnt_lag_1', 'date_subtype_avg_item_cnt_lag_1',
       'item_avg_item_price', 'date_item_avg_item_price',
       'date_item_avg_item_price_lag_1', 'date_item_avg_item_price_lag_2',
       'date_item_avg_item_price_lag_3', 'date_item_avg_item_price_lag_4',
       'date_item_avg_item_price_lag_5', 'date_item_avg_item_price_lag_6',
       'delta_price_lag_1', 'delta_price_lag_2', 'delta_price_lag_3',
       'delta_price_lag_4', 'delta_price_lag_5', 'delta_price_lag_6',
       'delta_price_lag'],
      dtype='object')
```

[44]:
```
dropped_cols = ['item_avg_item_price', 'date_item_avg_item_price']
for i in lags:
```

```
    dropped_cols += ['date_item_avg_item_price_lag_'+str(i)]
    dropped_cols += ['delta_price_lag_'+str(i)]

train4.drop(dropped_cols, axis=1, inplace=True)
train4.head()
```

[44]:

	date_block_num	shop_id	item_id	item_cnt_month	city_code	item_category_id	type_code	subtype_code	date_avg_item_cnt_lag_1	date_item_avg_item_cnt_lag_1	…	date_shop_avg_item_cnt_lag_12	date_cat_avg_item_cnt_lag_1	date_shop_cat_avg_item_cnt_lag_1	date_shop_type_avg_item_cnt_lag_1	date_type_avg_item_cnt_lag_1	date_city_avg_item_cnt_lag_1	date_shop_subtype_avg_item_cnt_lag_1
0	0	0	19	0.0	29	40	11	4	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	0	0	27	0.0	29	19	5	10	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	0	0	28	0.0	29	30	8	55	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	0	0	29	0.0	29	23	5	16	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0	0	32	6.0	29	40	11	4	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
   date_item_city_avg_item_cnt_lag_1  date_subtype_avg_item_cnt_lag_1  \
0                                NaN                              NaN
1                                NaN                              NaN
2                                NaN                              NaN
3                                NaN                              NaN
4                                NaN                              NaN

   delta_price_lag
0              0.0
1              0.0
2              0.0
3              0.0
4              0.0

[5 rows x 28 columns]
```

[45]:
```
#Trend feature for revenue
temp = sales_train.groupby(['date_block_num', 'shop_id']).agg({'sales':['sum']})
temp.columns = ['date_shop_revenue']
temp.reset_index(inplace=True)
train4 = pd.merge(train4, temp, on=['date_block_num', 'shop_id'], how='left')

temp = train4.groupby(['shop_id']).agg({'date_shop_revenue':['mean']})
temp.columns = ['shop_avg_revenue']
temp.reset_index(inplace=True)
train4 = pd.merge(train4, temp, on=['shop_id'], how='left')

train4['delta_revenue'] = (train4['date_shop_revenue'] -␣
↪train4['shop_avg_revenue']) / train4['shop_avg_revenue']
train4 = lag_feature(train4, [1], 'delta_revenue')

train4.head()
```

[45]:
```
   date_block_num  shop_id  item_id  item_cnt_month  city_code  \
0               0        0       19             0.0         29
1               0        0       27             0.0         29
2               0        0       28             0.0         29
3               0        0       29             0.0         29
4               0        0       32             6.0         29

   item_category_id  type_code  subtype_code  date_avg_item_cnt_lag_1  \
0                40         11             4                      NaN
1                19          5            10                      NaN
2                30          8            55                      NaN
3                23          5            16                      NaN
4                40         11             4                      NaN
```

```
   date_item_avg_item_cnt_lag_1  …  date_type_avg_item_cnt_lag_1  \
0                           NaN  …                           NaN
1                           NaN  …                           NaN
2                           NaN  …                           NaN
3                           NaN  …                           NaN
4                           NaN  …                           NaN

   date_city_avg_item_cnt_lag_1  date_shop_subtype_avg_item_cnt_lag_1  \
0                           NaN                                  NaN
1                           NaN                                  NaN
2                           NaN                                  NaN
3                           NaN                                  NaN
4                           NaN                                  NaN

   date_item_city_avg_item_cnt_lag_1  date_subtype_avg_item_cnt_lag_1  \
0                                NaN                              NaN
1                                NaN                              NaN
2                                NaN                              NaN
3                                NaN                              NaN
4                                NaN                              NaN

   delta_price_lag  date_shop_revenue  shop_avg_revenue  delta_revenue  \
0              0.0          2966412.0      3.319832e+06      -0.106457
1              0.0          2966412.0      3.319832e+06      -0.106457
2              0.0          2966412.0      3.319832e+06      -0.106457
3              0.0          2966412.0      3.319832e+06      -0.106457
4              0.0          2966412.0      3.319832e+06      -0.106457

   delta_revenue_lag_1
0                  NaN
1                  NaN
2                  NaN
3                  NaN
4                  NaN

[5 rows x 32 columns]
```

[46]:
```
train4.drop(['date_shop_revenue','shop_avg_revenue','delta_revenue'], axis=1,␣
 ↪inplace=True)
train4['delta_revenue_lag_1'] = train4['delta_revenue_lag_1'].fillna(0.0)
print(train4.shape)
train4.head()
```

```
(11128050, 29)
```

[46]:

	date_block_num	shop_id	item_id	item_cnt_month	city_code
0	0	0	19	0.0	29
1	0	0	27	0.0	29
2	0	0	28	0.0	29
3	0	0	29	0.0	29
4	0	0	32	6.0	29

	item_category_id	type_code	subtype_code	date_avg_item_cnt_lag_1
0	40	11	4	NaN
1	19	5	10	NaN
2	30	8	55	NaN
3	23	5	16	NaN
4	40	11	4	NaN

	date_item_avg_item_cnt_lag_1	…	date_cat_avg_item_cnt_lag_1
0	NaN	…	NaN
1	NaN	…	NaN
2	NaN	…	NaN
3	NaN	…	NaN
4	NaN	…	NaN

	date_shop_cat_avg_item_cnt_lag_1	date_shop_type_avg_item_cnt_lag_1
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

	date_type_avg_item_cnt_lag_1	date_city_avg_item_cnt_lag_1
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

	date_shop_subtype_avg_item_cnt_lag_1	date_item_city_avg_item_cnt_lag_1
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

	date_subtype_avg_item_cnt_lag_1	delta_price_lag	delta_revenue_lag_1
0	NaN	0.0	0.0
1	NaN	0.0	0.0
2	NaN	0.0	0.0
3	NaN	0.0	0.0

```
4                                NaN              0.0                    0.0

[5 rows x 29 columns]
```

[48]:
```
train4.columns
```

[48]:
```
Index(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month', 'city_code',
       'item_category_id', 'type_code', 'subtype_code',
       'date_avg_item_cnt_lag_1', 'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6',
       'date_shop_avg_item_cnt_lag_12', 'date_cat_avg_item_cnt_lag_1',
       'date_shop_cat_avg_item_cnt_lag_1', 'date_shop_type_avg_item_cnt_lag_1',
       'date_type_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_1',
       'date_shop_subtype_avg_item_cnt_lag_1',
       'date_item_city_avg_item_cnt_lag_1', 'date_subtype_avg_item_cnt_lag_1',
       'delta_price_lag', 'delta_revenue_lag_1'],
      dtype='object')
```

[47]:
```
# save train4
output = open('./data_processed/train4.pkl', 'wb')
pickle.dump(train4, output)

output.close()
```

4.7 2.7 Feature matrix

[49]:
```
df = train4[train4['date_block_num'] > 11]
print(df.shape)

for col in df.columns:
    if ('_lag_' in col) & (df[col].isnull().any()):
        if ('item_cnt' in col):
            df[col].fillna(0, inplace=True)
```

```
(6639294, 29)
```

Adding the processed text feature from section 2.2 to df.

[50]:
```
df = pd.merge(df, items1, on='item_id', how='left')
df = pd.merge(df, shops1, on='shop_id', how='left')
df.head()
```

[50]:
```
   date_block_num  shop_id  item_id  item_cnt_month  city_code  \
0              12        2       27             0.0          0
1              12        2       30             0.0          0
```

```
2              12        2       31             0.0          0
3              12        2       32             1.0          0
4              12        2       33             1.0          0

   item_category_id_x  type_code  subtype_code  date_avg_item_cnt_lag_1  \
0                  19          5            10                 0.411133
1                  40         11             4                 0.411133
2                  37         11             1                 0.411133
3                  40         11             4                 0.411133
4                  37         11             1                 0.411133

   date_item_avg_item_cnt_lag_1  …  shop_name_tfidf_15  shop_name_tfidf_16  \
0                      0.086975  …                 0.0                 0.0
1                      1.021484  …                 0.0                 0.0
2                      0.543457  …                 0.0                 0.0
3                      1.934570  …                 0.0                 0.0
4                      0.913086  …                 0.0                 0.0

   shop_name_tfidf_17  shop_name_tfidf_18  shop_name_tfidf_19  \
0                 0.0                 0.0                 0.0
1                 0.0                 0.0                 0.0
2                 0.0                 0.0                 0.0
3                 0.0                 0.0                 0.0
4                 0.0                 0.0                 0.0

   shop_name_tfidf_20  shop_name_tfidf_21  shop_name_tfidf_22  \
0                 0.0             0.49858                 0.0
1                 0.0             0.49858                 0.0
2                 0.0             0.49858                 0.0
3                 0.0             0.49858                 0.0
4                 0.0             0.49858                 0.0

   shop_name_tfidf_23  shop_name_tfidf_24
0                 0.0                 0.0
1                 0.0                 0.0
2                 0.0                 0.0
3                 0.0                 0.0
4                 0.0                 0.0

[5 rows x 86 columns]
```

[51]:
```
df.columns.values
```

[51]:
```
array(['date_block_num', 'shop_id', 'item_id', 'item_cnt_month',
       'city_code', 'item_category_id_x', 'type_code', 'subtype_code',
       'date_avg_item_cnt_lag_1', 'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6',
       'date_shop_avg_item_cnt_lag_12', 'date_cat_avg_item_cnt_lag_1',
       'date_shop_cat_avg_item_cnt_lag_1',
       'date_shop_type_avg_item_cnt_lag_1',
       'date_type_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_1',
       'date_shop_subtype_avg_item_cnt_lag_1',
       'date_item_city_avg_item_cnt_lag_1',
       'date_subtype_avg_item_cnt_lag_1', 'delta_price_lag',
       'delta_revenue_lag_1', 'item_name', 'item_category_id_y',
       'item_name_len', 'item_name_wc', 'item_name_tfidf_0',
       'item_name_tfidf_1', 'item_name_tfidf_2', 'item_name_tfidf_3',
       'item_name_tfidf_4', 'item_name_tfidf_5', 'item_name_tfidf_6',
       'item_name_tfidf_7', 'item_name_tfidf_8', 'item_name_tfidf_9',
       'item_name_tfidf_10', 'item_name_tfidf_11', 'item_name_tfidf_12',
       'item_name_tfidf_13', 'item_name_tfidf_14', 'item_name_tfidf_15',
       'item_name_tfidf_16', 'item_name_tfidf_17', 'item_name_tfidf_18',
       'item_name_tfidf_19', 'item_name_tfidf_20', 'item_name_tfidf_21',
       'item_name_tfidf_22', 'item_name_tfidf_23', 'item_name_tfidf_24',
       'shop_name', 'shop_name_len', 'shop_name_wc', 'shop_name_tfidf_0',
       'shop_name_tfidf_1', 'shop_name_tfidf_2', 'shop_name_tfidf_3',
       'shop_name_tfidf_4', 'shop_name_tfidf_5', 'shop_name_tfidf_6',
       'shop_name_tfidf_7', 'shop_name_tfidf_8', 'shop_name_tfidf_9',
       'shop_name_tfidf_10', 'shop_name_tfidf_11', 'shop_name_tfidf_12',
       'shop_name_tfidf_13', 'shop_name_tfidf_14', 'shop_name_tfidf_15',
       'shop_name_tfidf_16', 'shop_name_tfidf_17', 'shop_name_tfidf_18',
       'shop_name_tfidf_19', 'shop_name_tfidf_20', 'shop_name_tfidf_21',
       'shop_name_tfidf_22', 'shop_name_tfidf_23', 'shop_name_tfidf_24'],
      dtype=object)
```

[52]:
```
df.drop(['item_name', 'item_name_len', 'item_name_wc', 'item_category_id_y',
         'shop_name', 'shop_name_len', 'shop_name_wc'], axis=1, inplace=True)
print(df.shape)
```

```
(6639294, 79)
```

[53]:
```
cat_features = ['shop_id', 'item_id', 'city_code', 'item_category_id_x',␣
 ↪'type_code', 'subtype_code']
```

[54]:
```
del train1, train2, train3, train4, temp
gc.collect()
```

[54]:
```
6753
```

[56]:
```
df['item_cnt_month'].unique()
```

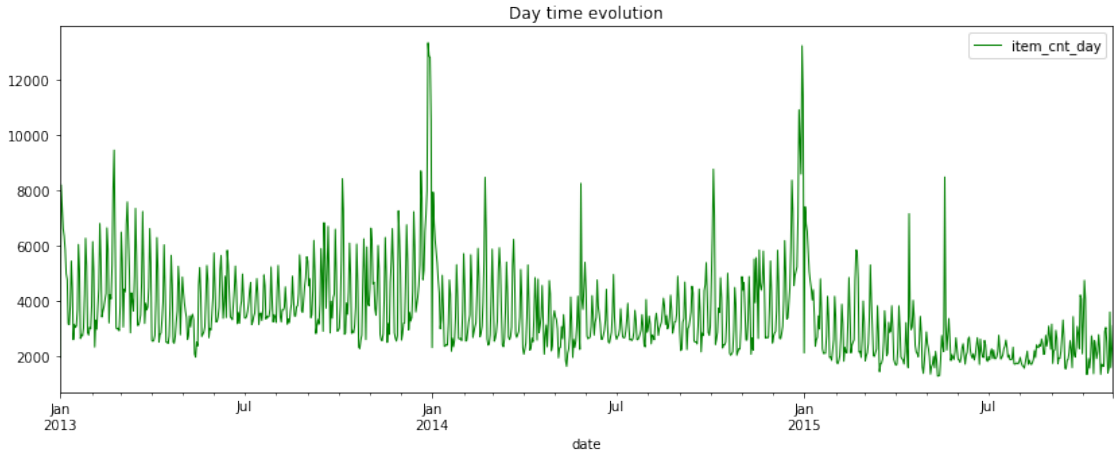
[56]:
```
array([ 0.,  1.,  2.,  7.,  9.,  3.,  4., 10.,  6.,  5., 20., 13., 11.,
       12.,  8., 16., 18., 19., 14., 15., 17.], dtype=float16)
```

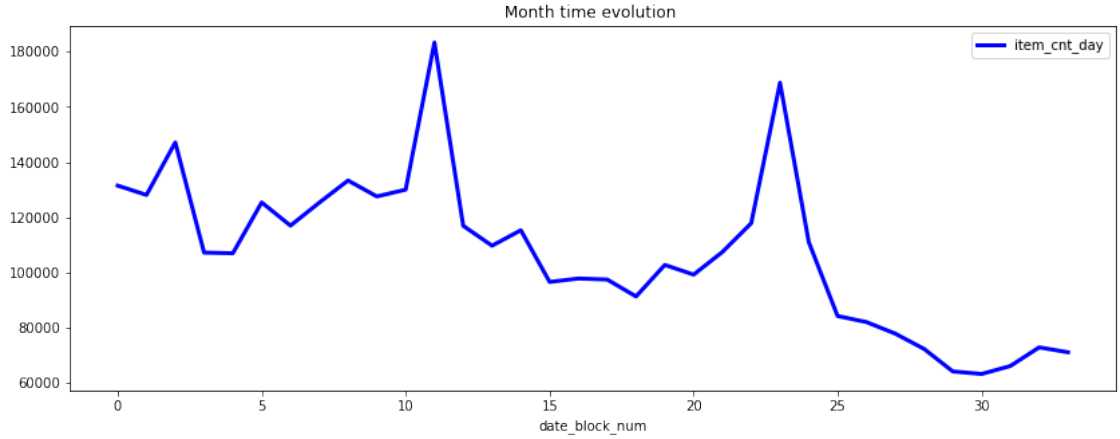
5 3. Exploratory data analysis

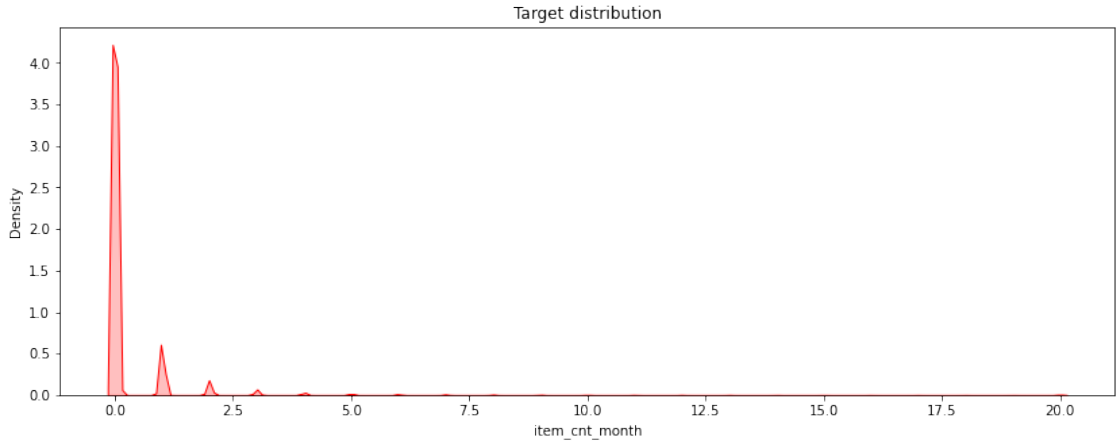
5.1 3.1 Target variable

The **target variable** to be predicted is `item_cnt_month`. Let's check its time series evolution and KDE distribution.

[57]:
```
sales_train['date'] = pd.to_datetime(sales_train['date'], format='%d.%m.%Y')
sales_train.groupby('date').agg({'item_cnt_day': 'sum'}).plot(figsize=(14,5),␣
 ↪title='Day time evolution', lw=1.0, color='green')
sales_train.groupby('date_block_num').agg({'item_cnt_day': 'sum'}).
 ↪plot(figsize=(14,5), title='Month time evolution ', lw=3.0, color='blue')
plt.show()

fig, ax = plt.subplots(figsize=(14,5))
ax = sns.kdeplot(df['item_cnt_month'].values, shade=True, color="red", ax=ax)
ax.set_xlabel('item_cnt_month')
ax.set_ylabel('Density')
ax.set_title('Target distribution')
plt.show()
```



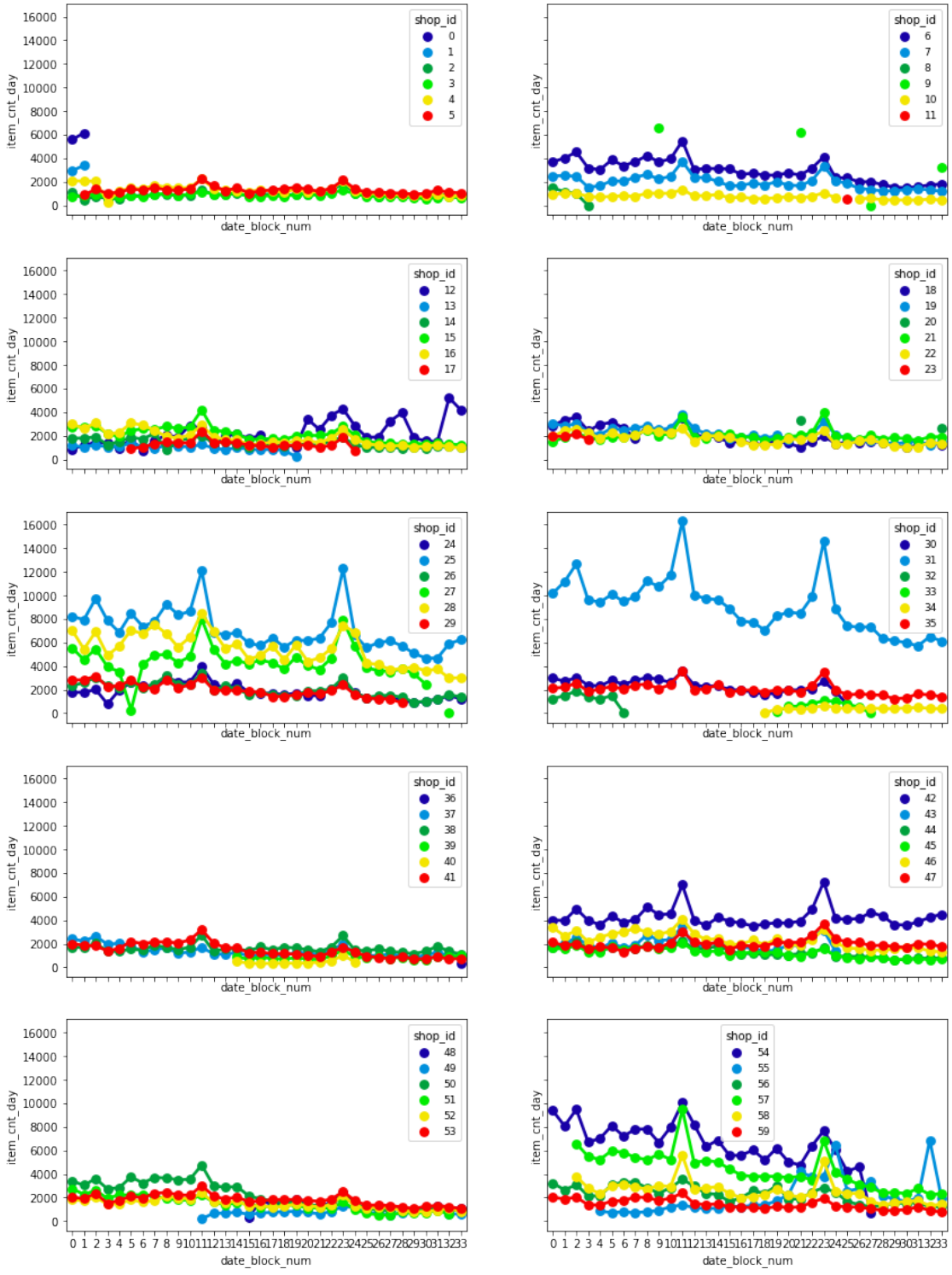




Now check the monthly performance of each shop:

[58]:
```
temp = pd.DataFrame(sales_train.groupby(['shop_id',␣
 ↪'date_block_num'])['item_cnt_day'].sum().reset_index())
fig, axes = plt.subplots(nrows=5, ncols=2, sharex=True, sharey=True,␣
 ↪figsize=(14,20))
num_graph = 10
id_per_graph = ceil(temp.shop_id.max() / num_graph) #temp.shop_id.max() is 59
x = 0
for i in range(5):
    for j in range(2):
        sns.pointplot(x='date_block_num', y='item_cnt_day', hue='shop_id',␣
 ↪palette='nipy_spectral',
                      data = temp[np.logical_and(x * id_per_graph <=␣
 ↪temp['shop_id'],
```

```
                                                temp['shop_id'] < (x+1) *␣
↪id_per_graph)],
                     ax=axes[i][j], grid=True, lw=1.0)
        plt.setp(axes[i][j].get_legend().get_texts(), fontsize='9')
        plt.setp(axes[i][j].get_legend().get_title(), fontsize='10')
        x += 1
# print(temp.shop_id.max())
# temp.head(10)
```


item_cnt_day
date_block_num
shop_id
0 1 2 3 4 5
6 7 8 9 10 11
12 13 14 15 16 17
18 19 20 21 22 23
24 25 26 27 28 29
30 31 32 33 34 35
36 37 38 39 40 41
42 43 44 45 46 47
48 49 50 51 52 53
54 55 56 57 58 59

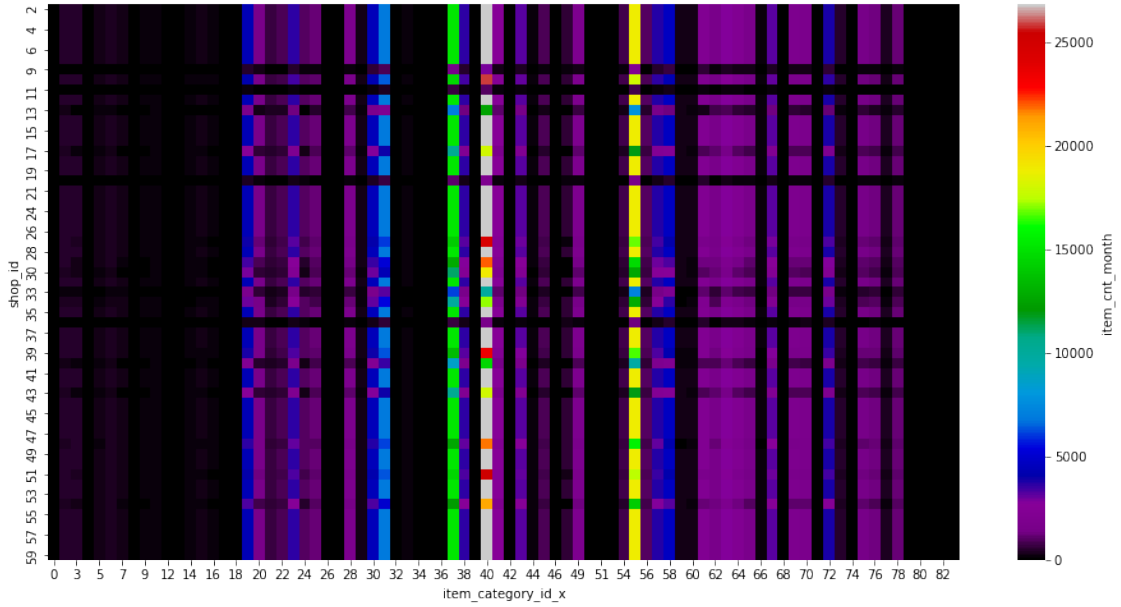
5.2 3.2 Multivariate heatmaps

We can pari different categorical variables (ex. `item_category_id`, `shop_id`, `city_code`) together and check the performance of different pairs.

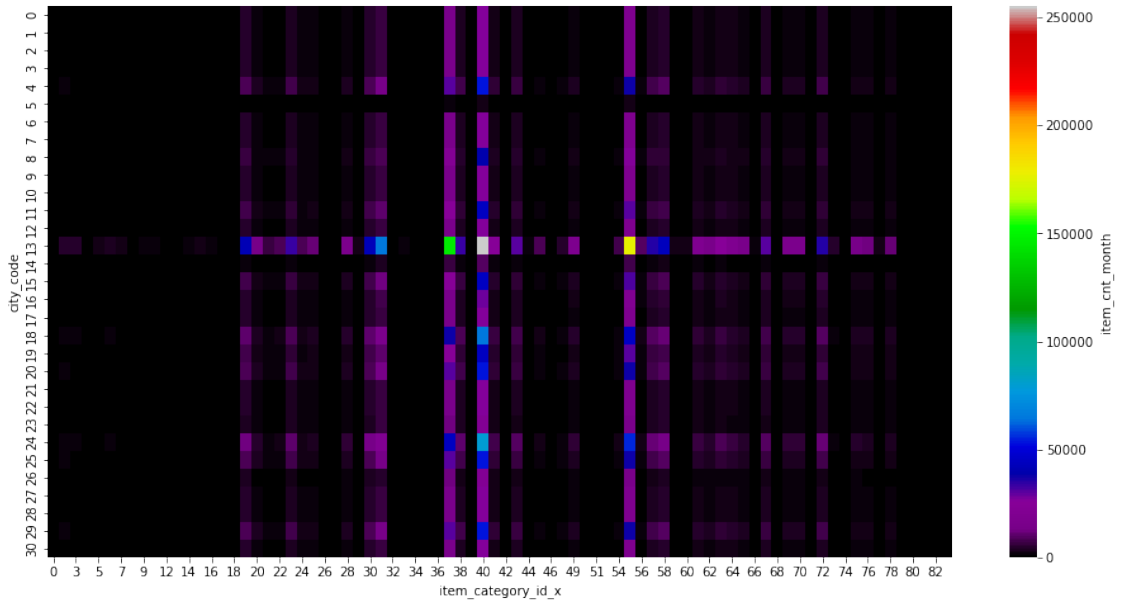
[59]:
```
#pair (item_category_id, shop_id)
stores_hm = df.pivot_table(index='shop_id', columns='item_category_id_x',␣
 ↪values='item_cnt_month', aggfunc='count', fill_value=0)
stores_hm.head()
```

[59]:
```
item_category_id_x  0    2    3   4    5    6    7   8    9    11  …  74  \
shop_id                                                             …
2                    1  513  467  58  265  414  276  49  153  198  …  22
3                    1  513  467  58  265  414  276  49  153  198  …  22
4                    1  513  467  58  265  414  276  49  153  198  …  22
5                    1  513  467  58  265  414  276  49  153  198  …  22
6                    1  513  467  58  265  414  276  49  153  198  …  22

item_category_id_x    75    76   77    78  79  80  81  82  83
shop_id
2                   1368  1290  315  1236  23  36  12  22  92
3                   1368  1290  315  1236  23  36  12  22  92
4                   1368  1290  315  1236  23  36  12  22  92
5                   1368  1290  315  1236  23  36  12  22  92
6                   1368  1290  315  1236  23  36  12  22  92

[5 rows x 80 columns]
```

[60]:
```
fig, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(stores_hm, cmap='nipy_spectral', ax=ax, cbar=True,␣
 ↪cbar_kws={'label': 'item_cnt_month'});
```

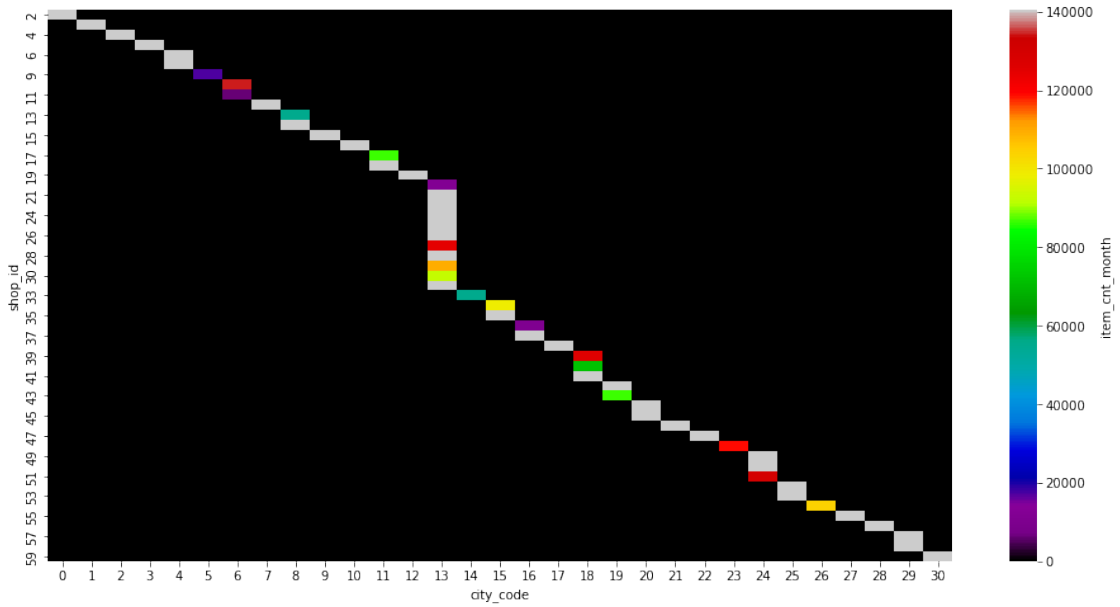


[61]:
```
#pair (item_category_id, city_code)
stores_hm = df.pivot_table(index='city_code', columns='item_category_id_x',␣
 ↪values='item_cnt_month', aggfunc='count', fill_value=0)
# stores_hm.head()
fig, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(stores_hm, cmap='nipy_spectral', ax=ax, cbar=True,␣
 ↪cbar_kws={'label': 'item_cnt_month'});
```



The next figure reveals the number of shops per cities and their sales performance.

[62]:
```
#pair (shop_id, city_code)
stores_hm = df.pivot_table(index='shop_id', columns='city_code',␣
 ↪values='item_cnt_month', aggfunc='count', fill_value=0)
# stores_hm.head()
fig, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(stores_hm, cmap='nipy_spectral', ax=ax, cbar=True,␣
 ↪cbar_kws={'label': 'item_cnt_month'});
```



Several pairs of shop-item perform better than others, let's exploit this by using target/mean encodings.

6 4. Advancedd Feature Engineering

6.1 4.1 Mean encoding on categorical features

Mean encodings look to map a high cardinality categorical feature into a 1D array (instead of high numbers of them had we used one-hot encoding) based on **how often the target variable appears on average in the categorical feature.**

In other words, for each unique value of the categorical feature, encode it based on **the ratio of occurrence of the positive class in the target variable.**

It is a convenient approach since it also has a clever way of imputing missing values among categorical variables.

[45]:
```
import copy
```

[63]:
```
df1 = copy.copy(df)
df1.reset_index(inplace=True)

cv = KFold(n_splits=5, shuffle=False)
new_features = []

check = False
for train_idx, valid_idx in cv.split(df1):
    # Train/validation split
    X_train, X_valid = df1.iloc[train_idx,:], df1.iloc[valid_idx,:]

    #Mean encoding
    #cat_features = ['shop_id', 'item_id', 'city_code',
    #'item_category_id_x', 'type_code', 'subtype_code']
    for col in cat_features:
        means = X_valid[col].map(X_train.groupby(col).item_cnt_month.mean()) #
        col_new = col + '_target_enc'
        X_valid[col_new] = means

        #results
        df1.loc[valid_idx, col_new] = X_valid

        #store new columns
        if check is False:
            new_features.append(col_new)
    check = True

print(new_features)
# df1.head()
```

```
['shop_id_target_enc', 'item_id_target_enc', 'city_code_target_enc',
'item_category_id_x_target_enc', 'type_code_target_enc',
'subtype_code_target_enc']
```

[64]:
```
X_train.head()
```

[64]:

	index	date_block_num	shop_id	item_id	item_cnt_month	city_code	\
0	0	12	2	27	0.0	0	
1	1	12	2	30	0.0	0	
2	2	12	2	31	0.0	0	
3	3	12	2	32	1.0	0	
4	4	12	2	33	1.0	0	

	item_category_id_x	type_code	subtype_code	date_avg_item_cnt_lag_1	…	\
0	19	5	10	0.411133	…	
1	40	11	4	0.411133	…	
2	37	11	1	0.411133	…	
3	40	11	4	0.411133	…	

```
4                  37         11             1                  0.411133  …

   shop_name_tfidf_21  shop_name_tfidf_22  shop_name_tfidf_23  \
0             0.49858                 0.0                 0.0
1             0.49858                 0.0                 0.0
2             0.49858                 0.0                 0.0
3             0.49858                 0.0                 0.0
4             0.49858                 0.0                 0.0

   shop_name_tfidf_24  shop_id_target_enc  item_id_target_enc  \
0                 0.0            0.141602            0.040802
1                 0.0            0.141602            0.186646
2                 0.0            0.141602            0.315186
3                 0.0            0.141602            0.575195
4                 0.0            0.141602            0.336426

   city_code_target_enc  item_category_id_x_target_enc  type_code_target_enc  \
0              0.141602                       0.550781              0.570312
1              0.141602                       0.230835              0.197021
2              0.141602                       0.157837              0.197021
3              0.141602                       0.230835              0.197021
4              0.141602                       0.157837              0.197021

   subtype_code_target_enc
0                 0.511230
1                 0.230835
2                 0.158691
3                 0.230835
4                 0.158691

[5 rows x 86 columns]
```

[65]:
```
X_valid.head()
```

[65]:
```
           index  date_block_num  shop_id  item_id  item_cnt_month  city_code  \
5311436  5311436              29        4     2270             0.0          2
5311437  5311437              29        4     2271             0.0          2
5311438  5311438              29        4     2272             0.0          2
5311439  5311439              29        4     2275             0.0          2
5311440  5311440              29        4     2277             0.0          2

         item_category_id_x  type_code  subtype_code  date_avg_item_cnt_lag_1  \
5311436                  23          5            16                 0.259521
5311437                  24          5            17                 0.259521
5311438                  31          8            61                 0.259521
5311439                  31          8            61                 0.000000
5311440                  31          8            61                 0.000000
```

```
         …  shop_name_tfidf_21  shop_name_tfidf_22  shop_name_tfidf_23  \
5311436  …            0.423972                 0.0                 0.0
5311437  …            0.423972                 0.0                 0.0
5311438  …            0.423972                 0.0                 0.0
5311439  …            0.423972                 0.0                 0.0
5311440  …            0.423972                 0.0                 0.0

         shop_name_tfidf_24  shop_id_target_enc  item_id_target_enc  \
5311436                 0.0            0.173706            0.752930
5311437                 0.0            0.173706            0.167480
5311438                 0.0            0.173706            0.037903
5311439                 0.0            0.173706            0.032013
5311440                 0.0            0.173706            0.030106

         city_code_target_enc  item_category_id_x_target_enc  \
5311436              0.173706                       0.634766
5311437              0.173706                       0.480469
5311438              0.173706                       0.062103
5311439              0.173706                       0.062103
5311440              0.173706                       0.062103

         type_code_target_enc  subtype_code_target_enc
5311436              0.633789                 0.615723
5311437              0.633789                 0.515625
5311438              0.543457                 0.070801
5311439              0.543457                 0.070801
5311440              0.543457                 0.070801

[5 rows x 86 columns]
```

[66]:
```
#fill missing value with mean of item_cnt_month
new_features = ['shop_id_target_enc', 'item_id_target_enc',␣
 ↪'city_code_target_enc', 'item_category_id_x_target_enc',␣
 ↪'type_code_target_enc', 'subtype_code_target_enc']
prior = np.mean(df1['item_cnt_month'].values)
df1[new_features] = df1[new_features].fillna(prior)
df1.head()
```

[66]:
```
   index  date_block_num  shop_id  item_id  item_cnt_month  city_code  \
0      0              12        2       27             0.0          0
1      1              12        2       30             0.0          0
2      2              12        2       31             0.0          0
3      3              12        2       32             1.0          0
4      4              12        2       33             1.0          0

   item_category_id_x  type_code  subtype_code  date_avg_item_cnt_lag_1  …  \
```

```
0                  19          5            10                 0.411133  …
1                  40         11             4                 0.411133  …
2                  37         11             1                 0.411133  …
3                  40         11             4                 0.411133  …
4                  37         11             1                 0.411133  …

   shop_name_tfidf_21  shop_name_tfidf_22  shop_name_tfidf_23  \
0             0.49858                 0.0                 0.0
1             0.49858                 0.0                 0.0
2             0.49858                 0.0                 0.0
3             0.49858                 0.0                 0.0
4             0.49858                 0.0                 0.0

   shop_name_tfidf_24  shop_id_target_enc  item_id_target_enc  \
0                 0.0            0.141602            0.040802
1                 0.0            0.141602            0.186646
2                 0.0            0.141602            0.315186
3                 0.0            0.141602            0.575195
4                 0.0            0.141602            0.336426

   city_code_target_enc  item_category_id_x_target_enc  type_code_target_enc  \
0              0.141602                       0.550781              0.570312
1              0.141602                       0.230835              0.197021
2              0.141602                       0.157837              0.197021
3              0.141602                       0.230835              0.197021
4              0.141602                       0.157837              0.197021

   subtype_code_target_enc
0                 0.511230
1                 0.230835
2                 0.158691
3                 0.230835
4                 0.158691

[5 rows x 86 columns]
```

[68]:
```
df1.drop(cat_features, axis=1, inplace=True)
df1.drop('index', axis=1, inplace=True)
cat_features = new_features
df1.columns.values
```

[68]:
```
array(['date_block_num', 'item_cnt_month', 'date_avg_item_cnt_lag_1',
       'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6', 'date_shop_avg_item_cnt_lag_12',
```

```
'date_cat_avg_item_cnt_lag_1', 'date_shop_cat_avg_item_cnt_lag_1',
'date_shop_type_avg_item_cnt_lag_1',
'date_type_avg_item_cnt_lag_1', 'date_city_avg_item_cnt_lag_1',
'date_shop_subtype_avg_item_cnt_lag_1',
'date_item_city_avg_item_cnt_lag_1',
'date_subtype_avg_item_cnt_lag_1', 'delta_price_lag',
'delta_revenue_lag_1', 'item_name_tfidf_0', 'item_name_tfidf_1',
'item_name_tfidf_2', 'item_name_tfidf_3', 'item_name_tfidf_4',
'item_name_tfidf_5', 'item_name_tfidf_6', 'item_name_tfidf_7',
'item_name_tfidf_8', 'item_name_tfidf_9', 'item_name_tfidf_10',
'item_name_tfidf_11', 'item_name_tfidf_12', 'item_name_tfidf_13',
'item_name_tfidf_14', 'item_name_tfidf_15', 'item_name_tfidf_16',
'item_name_tfidf_17', 'item_name_tfidf_18', 'item_name_tfidf_19',
'item_name_tfidf_20', 'item_name_tfidf_21', 'item_name_tfidf_22',
'item_name_tfidf_23', 'item_name_tfidf_24', 'shop_name_tfidf_0',
'shop_name_tfidf_1', 'shop_name_tfidf_2', 'shop_name_tfidf_3',
'shop_name_tfidf_4', 'shop_name_tfidf_5', 'shop_name_tfidf_6',
'shop_name_tfidf_7', 'shop_name_tfidf_8', 'shop_name_tfidf_9',
'shop_name_tfidf_10', 'shop_name_tfidf_11', 'shop_name_tfidf_12',
'shop_name_tfidf_13', 'shop_name_tfidf_14', 'shop_name_tfidf_15',
'shop_name_tfidf_16', 'shop_name_tfidf_17', 'shop_name_tfidf_18',
'shop_name_tfidf_19', 'shop_name_tfidf_20', 'shop_name_tfidf_21',
'shop_name_tfidf_22', 'shop_name_tfidf_23', 'shop_name_tfidf_24',
'shop_id_target_enc', 'item_id_target_enc', 'city_code_target_enc',
'item_category_id_x_target_enc', 'type_code_target_enc',
'subtype_code_target_enc'], dtype=object)
```

6.2 4.2 Matrix factorization of TFIDF processed features

We look to reduce the *dimensionality* of the TFIDF-processed features (from 50 dimensions to 10) and extract nonlinear relationships between the text features with **Non-Negative Matrix Factorization (NMF)**:

[69]:
```
tfidf_features = ['item_name_tfidf_0', 'item_name_tfidf_1', 'item_name_tfidf_2',
                  'item_name_tfidf_3', 'item_name_tfidf_4', 'item_name_tfidf_5',
                  'item_name_tfidf_6', 'item_name_tfidf_7', 'item_name_tfidf_8',
                  'item_name_tfidf_9', 'item_name_tfidf_10',␣
 ↪'item_name_tfidf_11',
                  'item_name_tfidf_12', 'item_name_tfidf_13',␣
 ↪'item_name_tfidf_14',
                  'item_name_tfidf_15', 'item_name_tfidf_16',␣
 ↪'item_name_tfidf_17',
                  'item_name_tfidf_18', 'item_name_tfidf_19',␣
 ↪'item_name_tfidf_20',
                  'item_name_tfidf_21', 'item_name_tfidf_22',␣
 ↪'item_name_tfidf_23',
```

```
                'item_name_tfidf_24', 'shop_name_tfidf_0',␣
↪'shop_name_tfidf_1',
                'shop_name_tfidf_2', 'shop_name_tfidf_3', 'shop_name_tfidf_4',
                'shop_name_tfidf_5', 'shop_name_tfidf_6', 'shop_name_tfidf_7',
                'shop_name_tfidf_8', 'shop_name_tfidf_9',␣
↪'shop_name_tfidf_10',
                'shop_name_tfidf_11', 'shop_name_tfidf_12',␣
↪'shop_name_tfidf_13',
                'shop_name_tfidf_14', 'shop_name_tfidf_15',␣
↪'shop_name_tfidf_16',
                'shop_name_tfidf_17', 'shop_name_tfidf_18',␣
↪'shop_name_tfidf_19',
                'shop_name_tfidf_20', 'shop_name_tfidf_21',␣
↪'shop_name_tfidf_22',
                'shop_name_tfidf_23', 'shop_name_tfidf_24']

X_train = df1[df1['date_block_num'] < 34][tfidf_features]
X_test = df1[df1['date_block_num'] == 34][tfidf_features]
print(X_train.shape, X_test.shape)
```

```
(6425094, 50) (214200, 50)
```

[70]:
```
nmf = NMF(n_components=10, init=None, solver='cd', beta_loss='frobenius', tol=0.
↪0001, max_iter=200)
nmf.fit(df1[tfidf_features])
```

[70]:
```
NMF(init=None, n_components=10)
```

[71]:
```
X_train = nmf.transform(X_train)
X_test = nmf.transform(X_test)
print(X_train.shape, X_test.shape)
```

```
(6425094, 10) (214200, 10)
```

[72]:
```
df1.drop(tfidf_features, axis=1, inplace=True)
```

[73]:
```
df1.head()
```

[73]:

	date_block_num	item_cnt_month	date_avg_item_cnt_lag_1	\
0	12	0.0	0.411133	
1	12	0.0	0.411133	
2	12	0.0	0.411133	
3	12	1.0	0.411133	
4	12	1.0	0.411133	

	date_item_avg_item_cnt_lag_1	date_item_avg_item_cnt_lag_2	\
0	0.086975	0.044434	
1	1.021484	1.022461	

```
2                      0.543457                      0.600098
3                      1.934570                      1.799805
4                      0.913086                      0.333252

   date_item_avg_item_cnt_lag_3  date_item_avg_item_cnt_lag_6  \
0                      0.130493                      0.065247
1                      0.521973                      0.891113
2                      0.543457                      0.304443
3                      1.260742                      1.891602
4                      0.717285                      1.000000

   date_item_avg_item_cnt_lag_12  date_shop_avg_item_cnt_lag_1  \
0                       0.155518                      0.148071
1                       0.000000                      0.148071
2                       0.000000                      0.148071
3                       5.378906                      0.148071
4                       1.355469                      0.148071

   date_shop_avg_item_cnt_lag_2  …  date_item_city_avg_item_cnt_lag_1  \
0                      0.100647  …                                0.0
1                      0.100647  …                                0.0
2                      0.100647  …                                0.0
3                      0.100647  …                                0.0
4                      0.100647  …                                1.0

   date_subtype_avg_item_cnt_lag_1  delta_price_lag  delta_revenue_lag_1  \
0                         1.075195        -0.282795             1.230799
1                         0.291504        -0.483264             1.230799
2                         0.234009        -0.137618             1.230799
3                         0.291504        -0.407143             1.230799
4                         0.234009        -0.225177             1.230799

   shop_id_target_enc  item_id_target_enc  city_code_target_enc  \
0            0.141602            0.040802              0.141602
1            0.141602            0.186646              0.141602
2            0.141602            0.315186              0.141602
3            0.141602            0.575195              0.141602
4            0.141602            0.336426              0.141602

   item_category_id_x_target_enc  type_code_target_enc  \
0                       0.550781              0.570312
1                       0.230835              0.197021
2                       0.157837              0.197021
3                       0.230835              0.197021
4                       0.157837              0.197021

   subtype_code_target_enc
```

```
0                  0.511230
1                  0.230835
2                  0.158691
3                  0.230835
4                  0.158691

[5 rows x 29 columns]
```

[74]:
```
tfidf_reduced_df = pd.concat([pd.DataFrame(X_train), pd.DataFrame(X_test)],␣
↪axis=0)
tfidf_reduced_df.columns = ['tfidf_interaction_1', 'tfidf_interaction_2',␣
↪'tfidf_interaction_3',
                            'tfidf_interaction_4', 'tfidf_interaction_5',␣
↪'tfidf_interaction_6',
                            'tfidf_interaction_7', 'tfidf_interaction_8',␣
↪'tfidf_interaction_9',
                            'tfidf_interaction_10']
print(tfidf_reduced_df.shape)
tfidf_reduced_df.head()
```

```
(6639294, 10)
```

[74]:
```
   tfidf_interaction_1  tfidf_interaction_2  tfidf_interaction_3  \
0             0.002103                  0.0             0.000000
1             0.001955                  0.0             0.000000
2             0.001766                  0.0             0.035762
3             0.001955                  0.0             0.000000
4             0.001766                  0.0             0.035762

   tfidf_interaction_4  tfidf_interaction_5  tfidf_interaction_6  \
0                  0.0                  0.0             0.030325
1                  0.0                  0.0             0.029987
2                  0.0                  0.0             0.030042
3                  0.0                  0.0             0.029987
4                  0.0                  0.0             0.030042

   tfidf_interaction_7  tfidf_interaction_8  tfidf_interaction_9  \
0             0.015853                  0.0                  0.0
1             0.000000                  0.0                  0.0
2             0.000000                  0.0                  0.0
3             0.000000                  0.0                  0.0
4             0.000000                  0.0                  0.0

   tfidf_interaction_10
0              0.000039
1              0.000000
2              0.000000
```

```
3              0.000000
4              0.000000
```

[75]:
```
for col in tfidf_reduced_df.columns:
    print(col)
    test1 = tfidf_reduced_df[col].values
    df1[col] = test1
df1.head()
```

```
tfidf_interaction_1
tfidf_interaction_2
tfidf_interaction_3
tfidf_interaction_4
tfidf_interaction_5
tfidf_interaction_6
tfidf_interaction_7
tfidf_interaction_8
tfidf_interaction_9
tfidf_interaction_10
```

[75]:

	date_block_num	item_cnt_month	date_avg_item_cnt_lag_1	\
0	12	0.0	0.411133	
1	12	0.0	0.411133	
2	12	0.0	0.411133	
3	12	1.0	0.411133	
4	12	1.0	0.411133	

	date_item_avg_item_cnt_lag_1	date_item_avg_item_cnt_lag_2	\
0	0.086975	0.044434	
1	1.021484	1.022461	
2	0.543457	0.600098	
3	1.934570	1.799805	
4	0.913086	0.333252	

	date_item_avg_item_cnt_lag_3	date_item_avg_item_cnt_lag_6	\
0	0.130493	0.065247	
1	0.521973	0.891113	
2	0.543457	0.304443	
3	1.260742	1.891602	
4	0.717285	1.000000	

	date_item_avg_item_cnt_lag_12	date_shop_avg_item_cnt_lag_1	\
0	0.155518	0.148071	
1	0.000000	0.148071	
2	0.000000	0.148071	
3	5.378906	0.148071	
4	1.355469	0.148071	

```
   date_shop_avg_item_cnt_lag_2  …  tfidf_interaction_1  \
0                      0.100647  …             0.002103
1                      0.100647  …             0.001955
2                      0.100647  …             0.001766
3                      0.100647  …             0.001955
4                      0.100647  …             0.001766

   tfidf_interaction_2  tfidf_interaction_3  tfidf_interaction_4  \
0                  0.0             0.000000                  0.0
1                  0.0             0.000000                  0.0
2                  0.0             0.035762                  0.0
3                  0.0             0.000000                  0.0
4                  0.0             0.035762                  0.0

   tfidf_interaction_5  tfidf_interaction_6  tfidf_interaction_7  \
0                  0.0             0.030325             0.015853
1                  0.0             0.029987             0.000000
2                  0.0             0.030042             0.000000
3                  0.0             0.029987             0.000000
4                  0.0             0.030042             0.000000

   tfidf_interaction_8  tfidf_interaction_9  tfidf_interaction_10
0                  0.0                  0.0              0.000039
1                  0.0                  0.0              0.000000
2                  0.0                  0.0              0.000000
3                  0.0                  0.0              0.000000
4                  0.0                  0.0              0.000000

[5 rows x 39 columns]
```

[76]:
```
print(df1.shape)
print(df1.columns.values)
```

```
(6639294, 39)
['date_block_num' 'item_cnt_month' 'date_avg_item_cnt_lag_1'
 'date_item_avg_item_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_item_cnt_lag_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_cnt_lag_6' 'date_shop_avg_item_cnt_lag_12'
 'date_cat_avg_item_cnt_lag_1' 'date_shop_cat_avg_item_cnt_lag_1'
 'date_shop_type_avg_item_cnt_lag_1' 'date_type_avg_item_cnt_lag_1'
 'date_city_avg_item_cnt_lag_1' 'date_shop_subtype_avg_item_cnt_lag_1'
 'date_item_city_avg_item_cnt_lag_1' 'date_subtype_avg_item_cnt_lag_1'
 'delta_price_lag' 'delta_revenue_lag_1' 'shop_id_target_enc'
 'item_id_target_enc' 'city_code_target_enc'
 'item_category_id_x_target_enc' 'type_code_target_enc'
 'subtype_code_target_enc' 'tfidf_interaction_1' 'tfidf_interaction_2'
```

```
'tfidf_interaction_3' 'tfidf_interaction_4' 'tfidf_interaction_5'
'tfidf_interaction_6' 'tfidf_interaction_7' 'tfidf_interaction_8'
'tfidf_interaction_9' 'tfidf_interaction_10']
```

[77]:
```
df1.to_csv('./data_processed/data.csv') #2.27G
```

[ ]:
```
# df1 = pd.read_csv('data.csv', index_col=0)
```