Forecasting monthly sales

January 9, 2020

1 Predict Future Sales

This notebook was completed as part of the Coursera Advanced Machine Learning specialisation as the final project for course 2.

This competition involved working with a challenging time-series dataset consisting of daily sales data, provided by one of the largest Russian software firms - 1C Company.

The task was to predict total sales for every product and store in the next month (November 2015).

The dataset can be accessed here: https://www.kaggle.com/c/competitive-data-science-predict-future-sales/overview

[2]:
```
import numpy as np
import pandas as pd
import os
from itertools import product
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from xgboost import plot_importance
import lightgbm as lgb
import seaborn as sns
import matplotlib.pyplot as plt
import time
import sys
import gc
import pickle
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline

Validation = False
reduce_size = False
#num_first_level_models = 3
```

```
SEED = 0
start_time = time.time()

pd.set_option('display.max_rows', 99)
pd.set_option('display.max_columns', 50)

def downcast_dtypes(df):
    float_cols = [c for c in df if df[c].dtype == "float64"]
    int_cols =   [c for c in df if df[c].dtype in ["int64", "int32"]]
    df[float_cols] = df[float_cols].astype(np.float32)
    df[int_cols]   = df[int_cols].astype(np.int16)
    return df

#function to calculate RMSE
def rmse(actual, predictions):
    return np.sqrt(mean_squared_error(actual, predictions))
```

```
/Users/charlottefettes/opt/anaconda3/lib/python3.7/site-
packages/lightgbm/__init__.py:48: UserWarning: Starting from version 2.2.1, the
library file in distribution wheels for macOS is built by the Apple Clang
(Xcode_8.3.3) compiler.
This means that in case of installing LightGBM from PyPI via the ``pip install
lightgbm`` command, you don't need to install the gcc compiler anymore.
Instead of that, you need to install the OpenMP library, which is required for
running LightGBM on the system with the Apple Clang compiler.
You can install the OpenMP library by the following command: ``brew install
libomp``.
  "You can install the OpenMP library by the following command: ``brew install
libomp``.", UserWarning)
```

[258]:
```
#read data into notebook
items = pd.read_csv('items.csv')
shops = pd.read_csv('shops.csv')
cats = pd.read_csv('item_categories.csv')
train = pd.read_csv('sales_train.csv')
test = pd.read_csv('test.csv')
```

1.0.1 Test set analysis

[3]:
```
good_sales = test.merge(train, on=['item_id','shop_id'], how='left').dropna()
good_pairs = test[test['ID'].isin(good_sales['ID'])]
others = test[~(test['ID'].isin(good_sales['ID']))]
item_only = others[others['item_id'].isin(train['item_id'])]
no_data_items = others[~others['item_id'].isin(train['item_id'])]

print('1. Number of good pairs:', len(good_pairs))
print('2. Only Item_id Info:', len(item_only))
print('3. No Data Items:', len(no_data_items))
```

```
1. Number of good pairs: 111404
2. Only Item_id Info: 87550
3. No Data Items: 15246
```

This shows that, within the test set, there are 6,719 occurrences involving items that have not appeared preveiously in the training set, 5,615 occurrences where the shop-item combination has not occurred in the training set but the item has appeared with different shops, and 41,180 occurrences involving shop-item combinations that have been recorded in the training set.

1.1 Data Exploration

[4]:
```
train.describe()
```

[4]:
```
       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
```

1.1.1 Outliers and abnormal entries

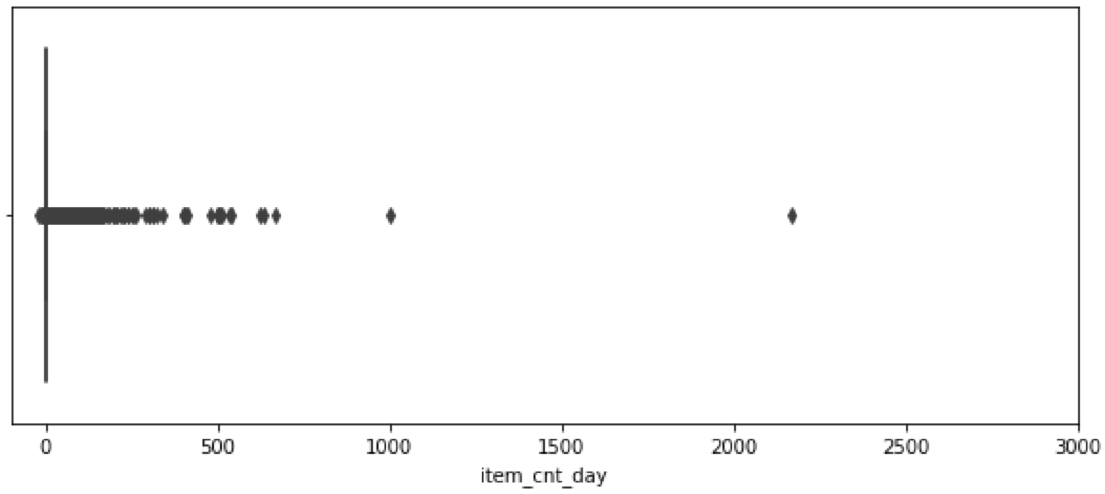
[5]:
```
negative_price = train[train.item_price < 0]
negative_price
```

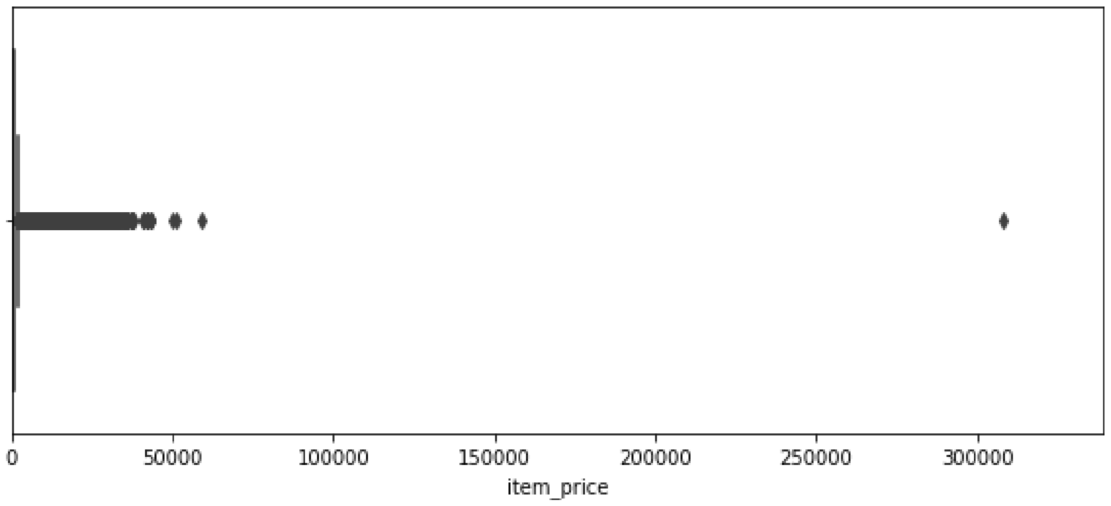
[5]:
```
              date  date_block_num  shop_id  item_id  item_price  item_cnt_day
484683  15.05.2013               4       32     2973        -1.0           1.0
```

[6]:
```
#item count per day boxplot
plt.figure(figsize=(10,4))
plt.xlim(-100, 3000)
sns.boxplot(x=train.item_cnt_day)

#item price boxplot
plt.figure(figsize=(10,4))
plt.xlim(train.item_price.min(), train.item_price.max()*1.1)
sns.boxplot(x=train.item_price)
```

[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7b8cc8450>
```





These boxplots show outliers. To deal with these, entries with item price above 100000 and sales above 1001 will be removed from the dataset.

The data also includes negative entries for price (one item). This will be replaced with the median price.

[7]:
```
train[train['item_id'] == 11373][['item_price']].sort_values(['item_price'])
```

[7]:
```
         item_price
2909818    0.908714
2257993   38.500000
2048642   71.000000
1058343   72.200000
2462729   75.454545
```

```
…               …
2608320  1790.000000
885303   1821.000000
1006939  1931.000000
1830098  2111.500000
1058333  2137.000000

[464 rows x 1 columns]
```

[8]:
```
train[train['item_id'] == 11365].sort_values(['item_price'])
```

[8]:
```
               date  date_block_num  shop_id  item_id  item_price  \
1651714  16.05.2014              16       12    11365       124.0
2805487  21.08.2015              31       12    11365       170.0
1330776  13.01.2014              12       12    11365       180.0
1398688  25.02.2014              13       12    11365       194.0
661581   05.07.2013               6       12    11365       230.0
…               …               …       …        …          …
885161   28.09.2013               8       12    11365      9370.0
302568   12.03.2013               2       12    11365     10540.0
885165   23.09.2013               8       12    11365     11880.0
302544   05.03.2013               2       12    11365     14530.0
885138   17.09.2013               8       12    11365     59200.0

         item_cnt_day
1651714           5.0
2805487           2.0
1330776           3.0
1398688           5.0
661581            4.0
…               …
885161            1.0
302568            1.0
885165            1.0
302544            1.0
885138            1.0

[242 rows x 6 columns]
```

[9]:
```
#Correct train values
train['item_price'][2909818] = np.nan
train['item_cnt_day'][2909818] = np.nan
train['item_price'][2909818] = train[(train['shop_id'] ==12) &␣
↪(train['item_id'] == 11373) & (train['date_block_num'] == 33)]['item_price'].
↪median()
```

```
train['item_cnt_day'][2909818] = round(train[(train['shop_id'] ==12) &␣
↪(train['item_id'] == 11373) & (train['date_block_num'] ==␣
↪33)]['item_cnt_day'].median())
train['item_price'][885138] = np.nan
train['item_price'][885138] = train[(train['item_id'] == 11365) &␣
↪(train['shop_id'] ==12) & (train['date_block_num'] == 8)]['item_price'].
↪median()
```

[10]:
```
#remove extreme outliers
train = train[train.item_price<100000]
train = train[train.item_cnt_day<1001]
```

[11]:
```
#median of entries with shop id 32, item id 2973, month 4 and item price␣
↪greater than 0
median = train[(train.shop_id==32)&(train.item_id==2973)&(train.
↪date_block_num==4)&(train.item_price>0)].item_price.median()

#replace negative entry with median value
train.loc[train.item_price<0, 'item_price'] = median
```

[12]:
```
#examine shop entries
shops
```

[12]:
```
                                          shop_name  shop_id
0                     !           , 56           0
1                     !       "      "           1
2                                         "  "        2
3                            "    -    "        3
4                                  "      "        4
5                                    "    "        5
6                             (       , 13)        6
7                                    "    "        7
8                                -   "  "        8
9                                                   9
10                              .      39 ?      10
11                              .      39 ²      11
12                                   -            12
13                                     "    "      13
14                                 "    " II       14
15                                    "XXI   "       15
16                                        "  "       16
17                             "       "       17
18                                      "  "       18
19                                  "      "      19
20                                   "     "       20
21                                   "     "      21
22                                         21       22
```

```
23                        "      " (  . 2)       23
24                        "      " (  . 7)       24
25                                      "   "       25
26                             "  " (    )       26
27                          "          II"       27
28                         "          " II       28
29                      "     " (      )       29
30                                  "     "       30
31                                 "      "       31
32                              "        "       32
33                                        "XL-3"       33
34                              .        "  "       34
35                       .        "      "       35
36                      "           "       36
37                                      "  "       37
38                                         "  "       38
39                       "          "       39
40             "          "             40
41                                      "  "       41
42                                  "       "       42
43                                        "   "       43
44                                     "   "       44
45                                    "    "       45
46                                       "7 "       46
47                                   "     "       47
48                            "         "       48
49                                    "    "       49
50                                      "   "       50
51                               "       "       51
52                                   "      "       52
53                                       "  " 2       53
54                                         "  "       54
55                                   1 -           55
56                                    "    "       56
57                                      , 56       57
58                                 "      "       58
59                                    "   "       59
```

From looking at this dataframe, 4 duplicate entries have been found but listed separately due to spelling variations: + ! , 56 (shop id 0), , 56 (shop id 57) + ! “ ” (shop id 1), “ ” (shop id 58) + . 39 ? (shop id 10), . 39 ^{2} (shop id 11) + “ ” (shop id 39), “ ” (shop id 40)

These codes will be corrected to match in both the test and train sets.

From looking at the shop names, the first word is duplicated. From research, these are apparently the names of the cities in which these shops are located. This information will be useful for processing the data later.

[13]:
```
#              , 56
train.loc[train.shop_id == 0, 'shop_id'] = 57
test.loc[test.shop_id == 0, 'shop_id'] = 57

#        "      "
train.loc[train.shop_id == 1, 'shop_id'] = 58
test.loc[test.shop_id == 1, 'shop_id'] = 58

#         .      39 ²
train.loc[train.shop_id == 11, 'shop_id'] = 10
test.loc[test.shop_id == 11, 'shop_id'] = 10

#           "          "
train.loc[train.shop_id == 40, 'shop_id'] = 39
test.loc[test.shop_id == 40, 'shop_id'] = 39

#retain only shop_id present in test set
train = train.merge(test[['shop_id']].drop_duplicates(), how = 'inner')

#convert date to datetime
train['date'] = pd.to_datetime(train['date'], format = '%d.%m.%Y')
```

[14]:
```
#save altered train
train.to_pickle('train_alt.pickle.gzde', compression='gzip')
```

[15]:
```
del train
```

[16]:
```
cats
```

[16]:
```
                          item_category_name  item_category_id
0                    PC -      /                      0
1                                  - PS2                 1
2                                  - PS3                 2
3                                  - PS4                 3
4                                  - PSP                 4
5                               - PSVita                 5
6                             - XBOX 360                 6
7                             - XBOX ONE                 7
8                                  (   )                8
9                                                      9
10                               - PS2                10
11                               - PS3                11
12                               - PS4                12
13                               - PSP                13
14                            - PSVita                14
```

```
62               -     ,    ,                    62
63                       -                         63
64                      -                         64
65         -           (     )                 65
66                     -     ,                    66
67                               -                    67
68                    -       ,                  68
69                               -                    69
70                   -      (     )               70
71       -    ,    ,      /                    71
72                                -                    72
73                      - 1 :      8               73
74                           - MAC (   )                74
75                      -                          75
76              -            (   )                76
77                             -                     77
78                     -       (   )               78
79                                                        79
80                                -                    80
81                               (   )               81
82                             (    )               82
83                                                     83
```

[17]:
```
# Select all duplicate rows based on one column
duplicateRowsCats = cats[cats.duplicated(['item_category_name'])]
print("Duplicate Rows based on a single column are:", duplicateRowsCats,␣
↪sep='\n')
```

```
Duplicate Rows based on a single column are:
Empty DataFrame
Columns: [item_category_name, item_category_id]
Index: []
```

No apparent issues have been spotted from the item category dataframe.

It has been noted that each category name includes type (duplicated) followed by subtype, separated by a dash.

[18]:
```
items
```

[18]:
```
                                               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
…                                                  …       …
22165                      2 [PC,          ]     22165
```

```
22166            1 :        [         ]    22166
22167          1 :       8 (+CD).      …    22167
22168                                    Little Inu    22168
22169                              (         )    22169

       item_category_id
0                    40
1                    76
2                    40
3                    40
4                    40
…                   …
22165                31
22166                54
22167                49
22168                62
22169                69

[22170 rows x 3 columns]
```

[19]:
```python
# Select all duplicate rows based on one column
duplicateRowsItems = items[items.duplicated(['item_name'])]
print("Duplicate Rows based on a single column are:", duplicateRowsItems,␣
 ↪sep='\n')
```

```
Duplicate Rows based on a single column are:
Empty DataFrame
Columns: [item_name, item_id, item_category_id]
Index: []
```

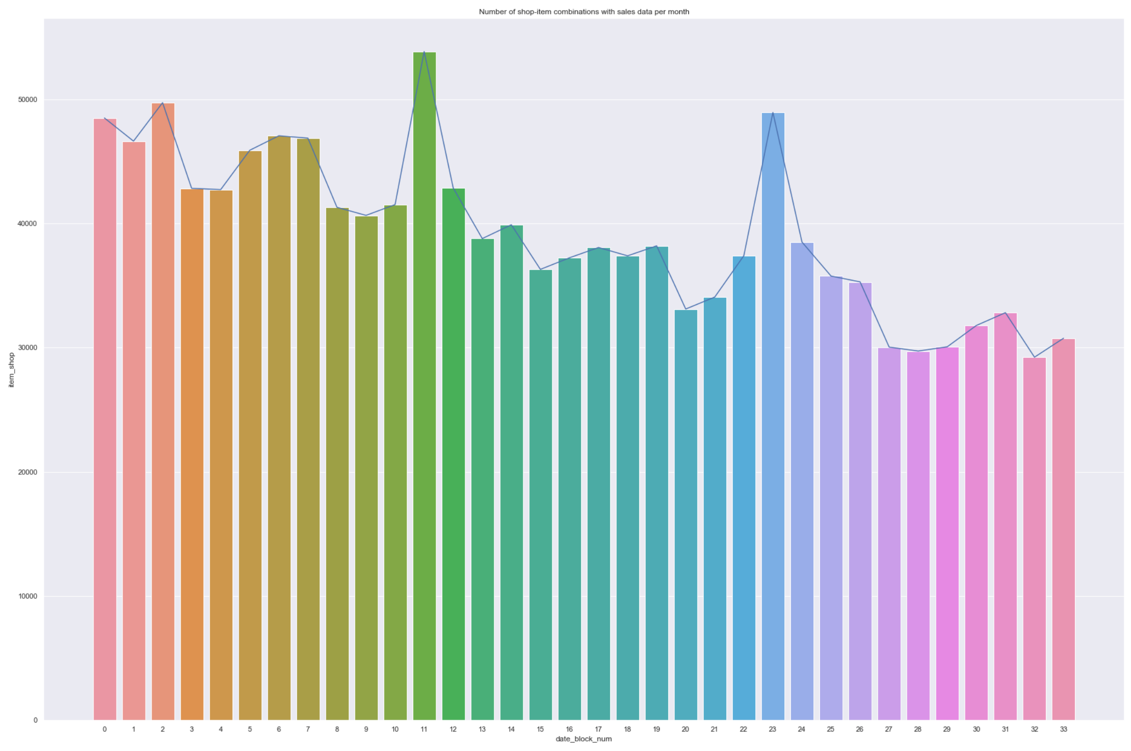
No apparent issues have been spotted in the items dataframe.

1.1.2 Further exploration

[20]:
```python
train = pd.read_pickle('train_alt.pickle.gzde', compression='gzip')
train.nunique()
```

[20]:
```
date              1034
date_block_num      34
shop_id             42
item_id          21085
item_price       16567
item_cnt_day       191
dtype: int64
```

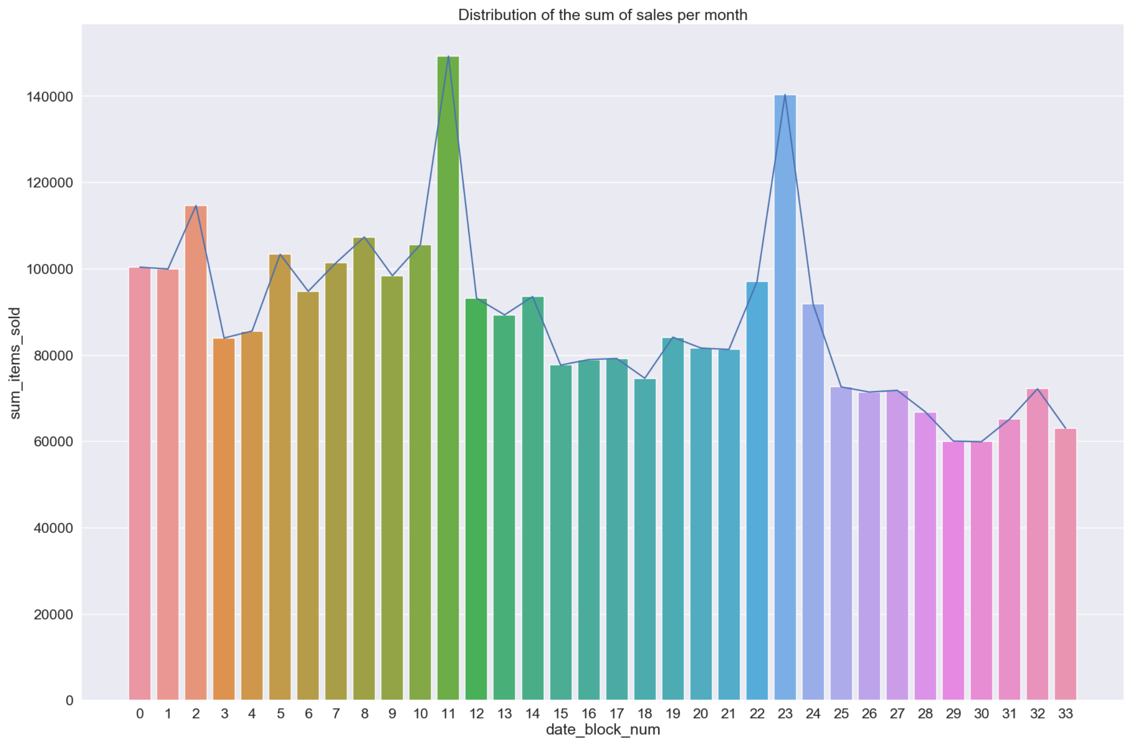
[21]:
```python
#number of sales-item combinations for which data is available per month
sns.set(rc={'figure.figsize':(30, 20)})
shop_item = pd.DataFrame(train[['date_block_num', 'shop_id',
                                        'item_id']].drop_duplicates().
↪groupby('date_block_num').size()).reset_index()
shop_item.columns = ['date_block_num', 'item_shop']
sns.barplot(x ='date_block_num', y='item_shop', data=shop_item);
plt.plot(shop_item['item_shop']);
plt.title('Number of shop-item combinations with sales data per month')
del shop_item
```



The plot above shows not all item-shop combinations have data for every month.

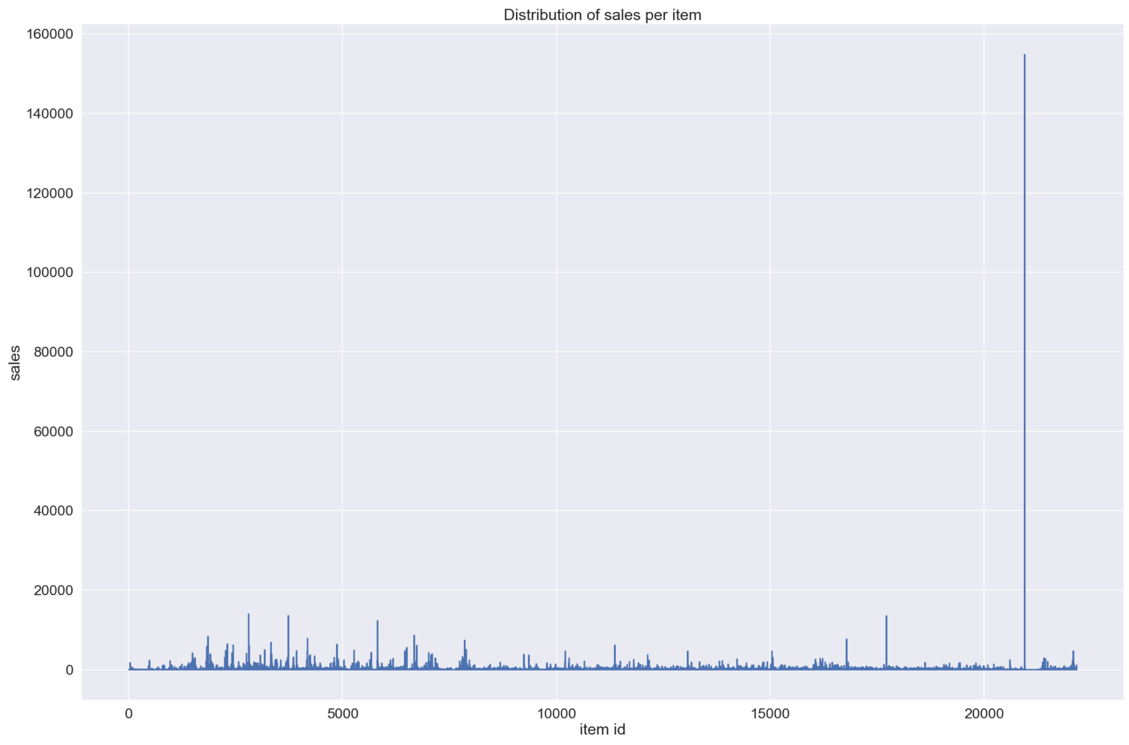
To enable calculation of these previously unrecorded combinations, the train dataset will need to be extended to include these missing combinations.

[22]:
```
#visualise sales per month total
sns.set(rc={'figure.figsize':(30, 20)})
sns.set_context("talk", font_scale=1.4)
sales_month = pd.DataFrame(train.groupby(['date_block_num']).sum().
↪item_cnt_day).reset_index()
sales_month.columns = ['date_block_num', 'sum_items_sold']
sns.barplot(x ='date_block_num', y='sum_items_sold',
            data=sales_month.reset_index());
plt.plot(sales_month.sum_items_sold)
plt.title('Distribution of the sum of sales per month')
del sales_month
```



This plot shows there is monthly variation in sales, therefore, date_block_num (month number) is a variable that should be accounted for.

```
[23]: #sales per item and shop combination
sns.set(rc={'figure.figsize':(30, 20)})
sns.set_context("talk", font_scale=1.4)
sales_item_id = pd.DataFrame(train.groupby(['item_id']).sum().item_cnt_day)
plt.xlabel('item id')
plt.ylabel('sales')
plt.title('Distribution of sales per item');
plt.plot(sales_item_id);
```



The plot shows that the vast majority of total sales for shop-item combinations are low, with some showing very high numbers.

```
[24]: sales_item_id = sales_item_id.reset_index()
      large_item = sales_item_id.item_cnt_day.argmax()
      large_item
```

```
[24]: 20055
```

```
[25]: sales_item_id.iloc[20055]
```

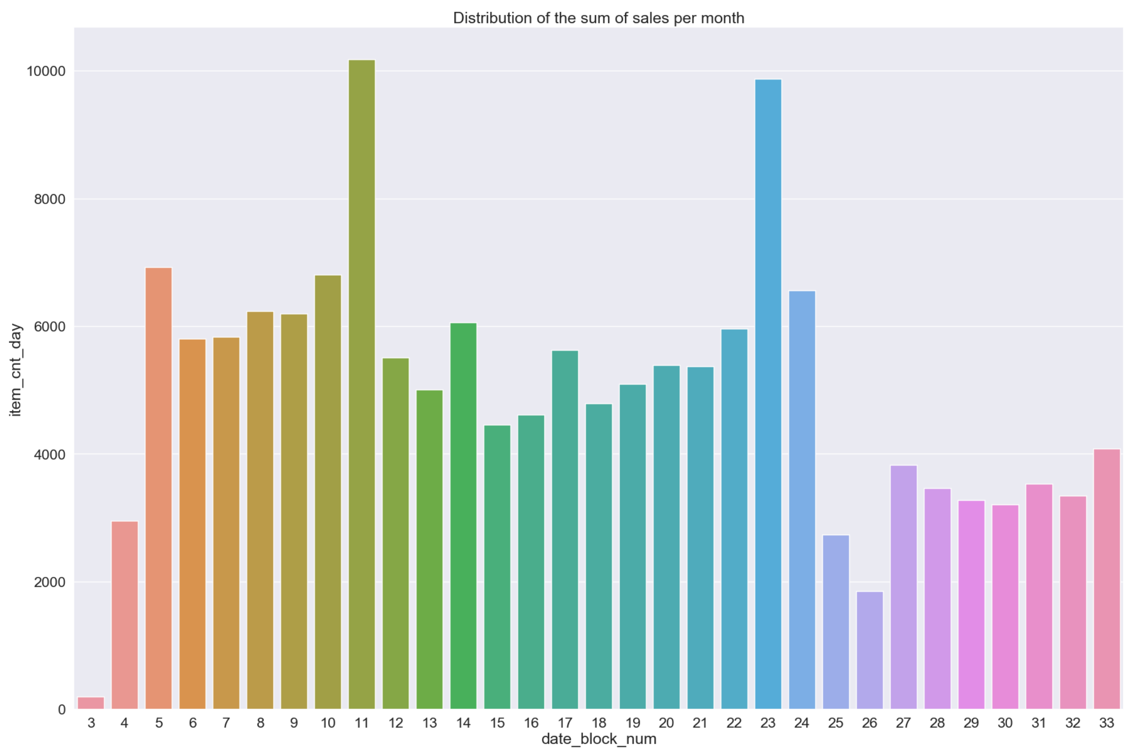
```
[25]: item_id            20949.0
      item_cnt_day      154771.0
      Name: 20055, dtype: float64
```

```
[26]: mask = (train.item_id == 20949)
      highest = train[mask]

      sns.set(rc={'figure.figsize':(30, 20)})
      sns.set_context("talk", font_scale=1.4)
      highest_df = pd.DataFrame(highest.groupby(['date_block_num']).sum().
       ↪item_cnt_day)
      sns.barplot(x ='date_block_num', y='item_cnt_day',
                  data=highest_df.reset_index());
```

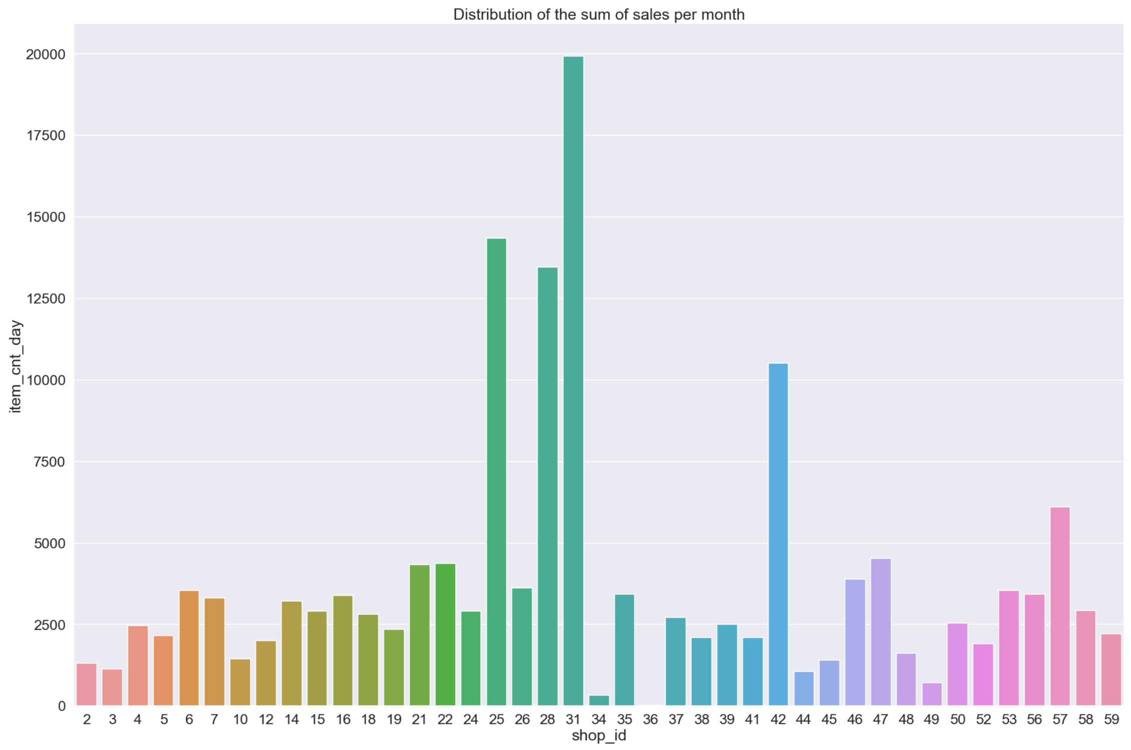
```
plt.title('Distribution of the sum of sales per month')
```

[26]:
```
Text(0.5, 1.0, 'Distribution of the sum of sales per month')
```



[27]:
```
sns.set(rc={'figure.figsize':(30, 20)})
sns.set_context("talk", font_scale=1.4)
highest_df2 = pd.DataFrame(highest.groupby(['shop_id']).sum().item_cnt_day)
sns.barplot(x ='shop_id', y='item_cnt_day',
            data=highest_df2.reset_index());
plt.title('Distribution of the sum of sales per month')
```

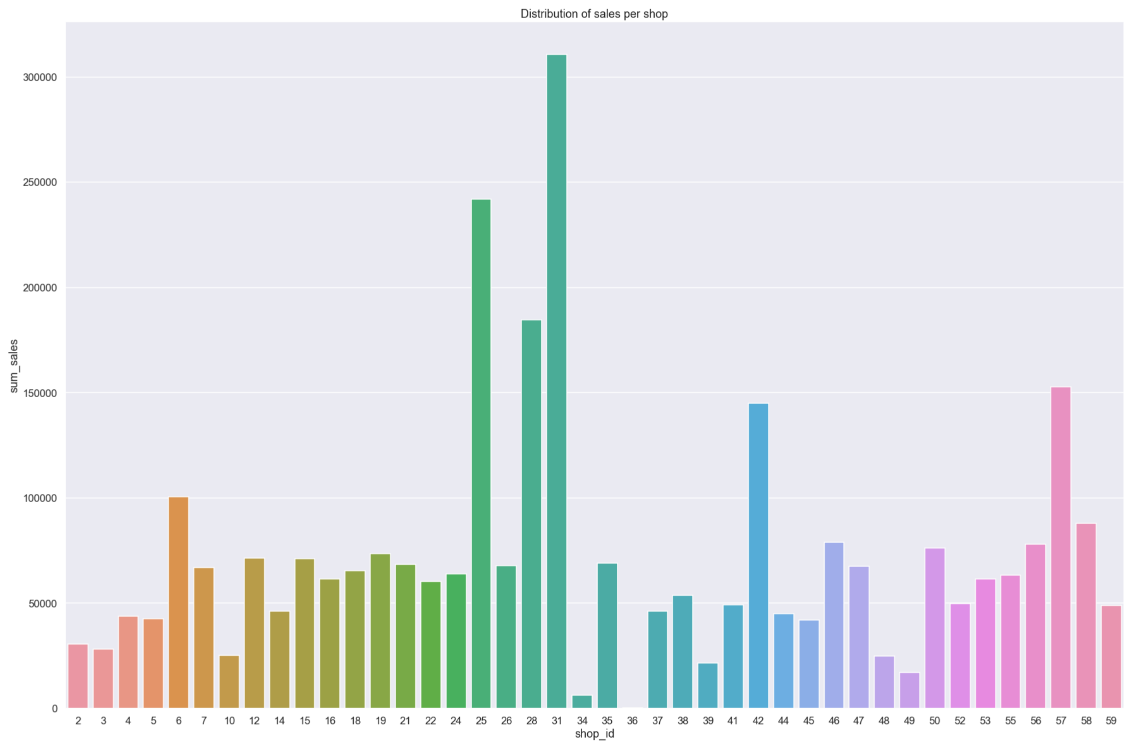
[27]:
```
Text(0.5, 1.0, 'Distribution of the sum of sales per month')
```



The plots above show the distribution of sales for the item with the highest total sales within the dataset. The first shows some variation across months. The second shows a great deal of variation between shops for sales of this item, with a very small number of shops dominating and driving total sales of the item up.

Based on this analysis, total sales will be capped to prevent the possibility of extremely high sales for some items within certain stores driving up predictions to ranges beyond that which are likely.

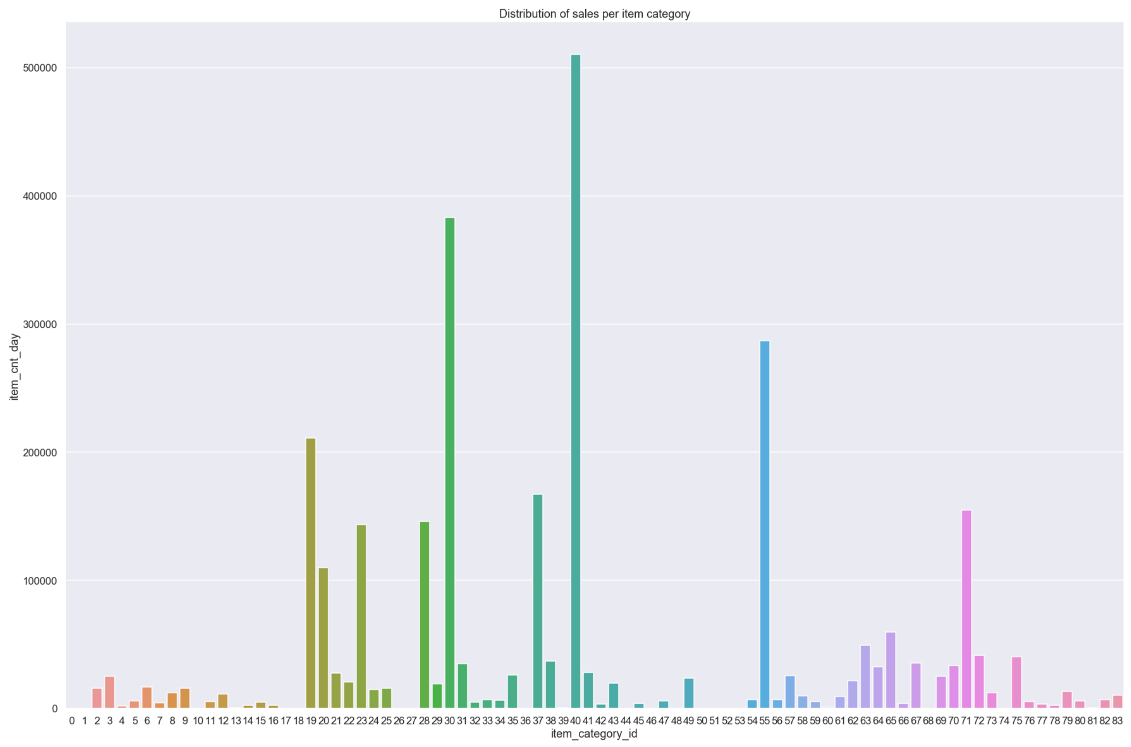
[28]:
```
#sales per shop id
sns.set_context("talk", font_scale=1)
sales_month_shop_id = pd.DataFrame(train.groupby(['shop_id']).sum().
 ↪item_cnt_day).reset_index()
sales_month_shop_id.columns = ['shop_id', 'sum_sales']
sns.barplot(x ='shop_id', y='sum_sales', data=sales_month_shop_id)
plt.title('Distribution of sales per shop');
```



The above plot shows the distribution of sales of all items per shop. There is a lot of variation in sales per shop (possibly based on location or otherwise), and so this is a variable that the model will need to account for.

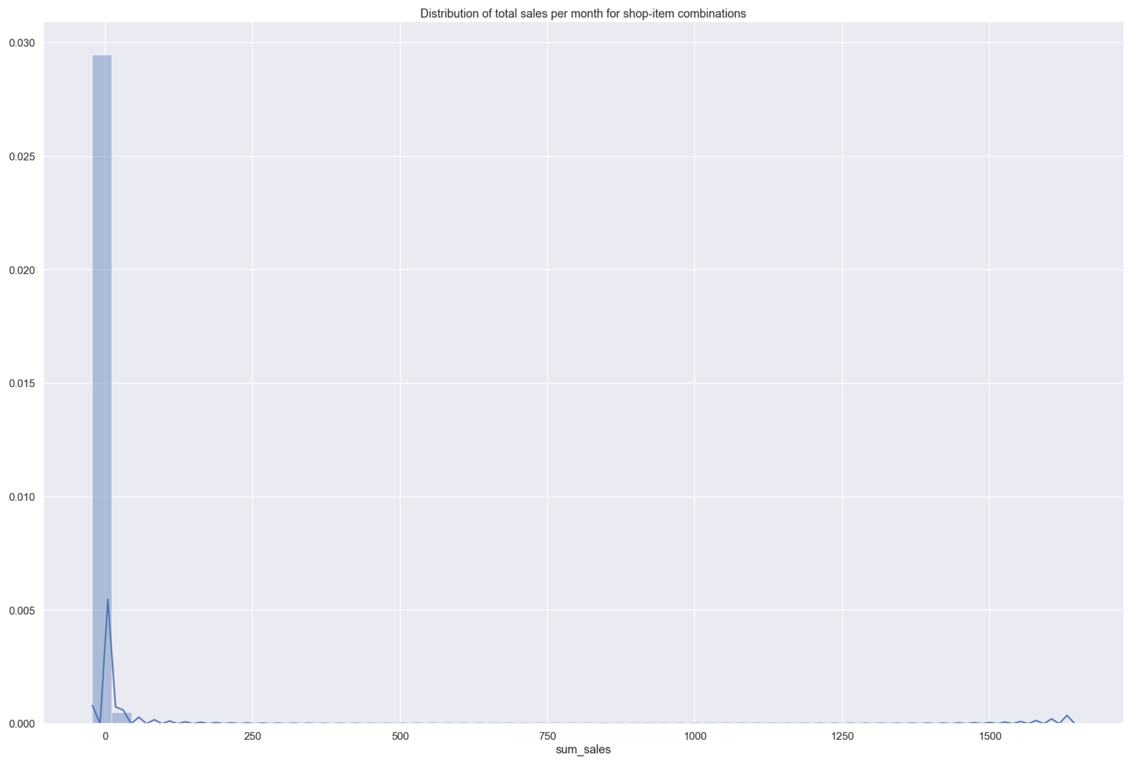
[29]:
```
del sales_month_shop_id
del sales_item_id
del highest_df
del highest
```

[30]:
```
#sales per item category
sales_item_category = train.merge(items, how='left',on='item_id').
 ↪groupby('item_category_id')['item_cnt_day'].sum()
sns.barplot(x = 'item_category_id', y = 'item_cnt_day', data =␣
 ↪sales_item_category.reset_index());
plt.title('Distribution of sales per item category');
del sales_item_category
```

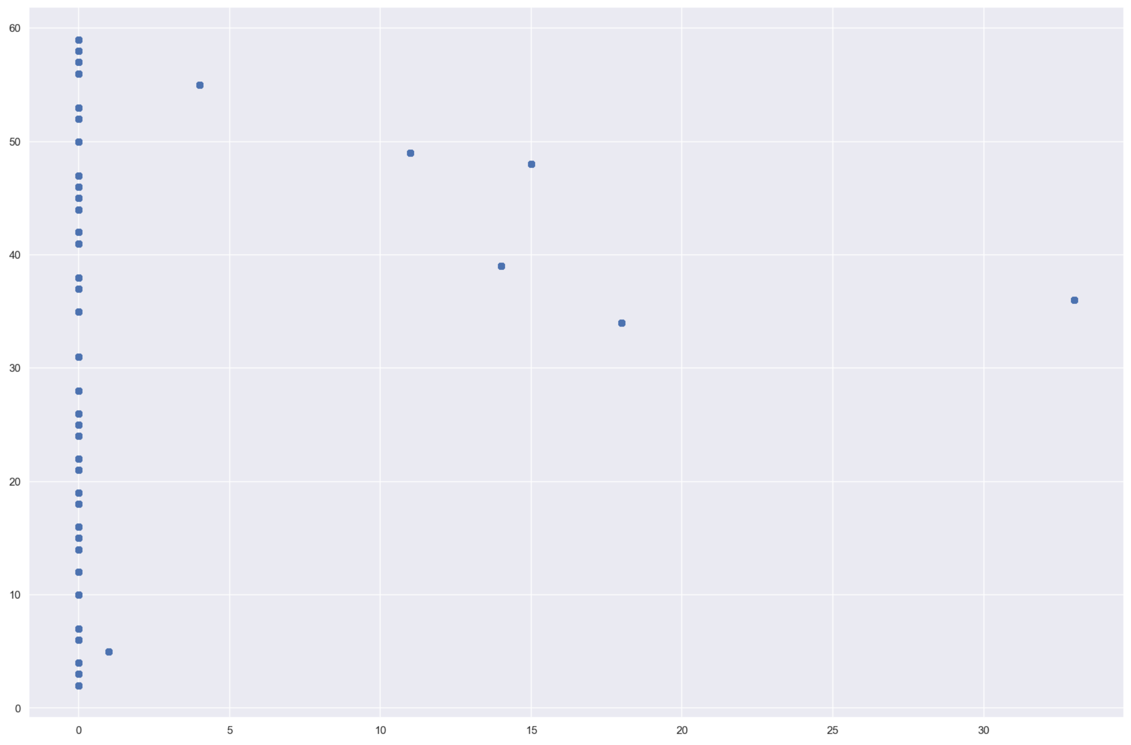


Again, there is a lot of variation in sales per item category id, which will likely be helpful in predicting sales.

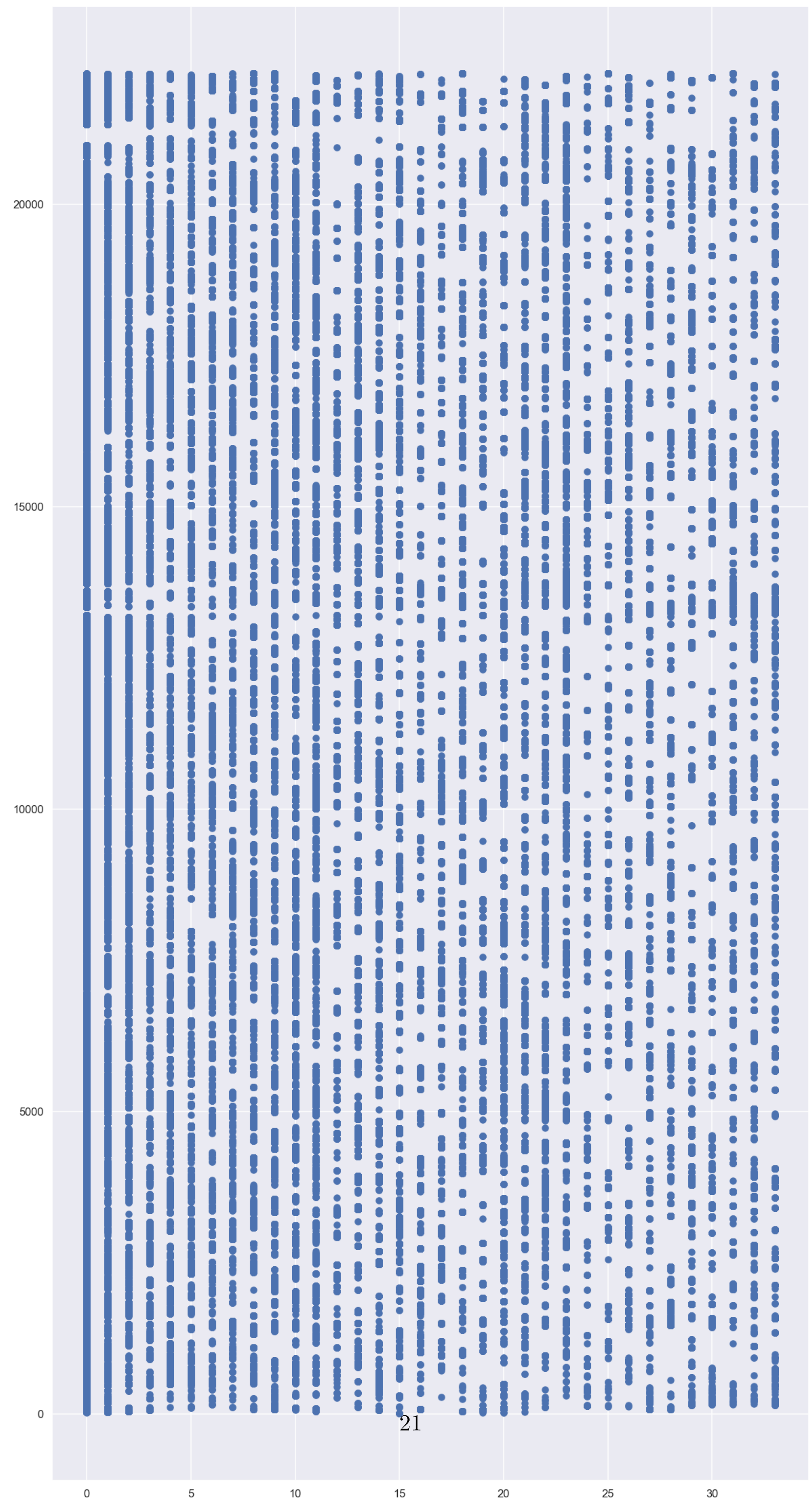
```
[31]: #sales per shop id
sns.set_context("talk", font_scale=1)
sales_month_shop_item = pd.DataFrame(train.
 ↪groupby(['date_block_num','shop_id','item_id']).sum().item_cnt_day).
 ↪reset_index()
sales_month_shop_item.columns =␣
 ↪['date_block_num','shop_id','item_id','sum_sales']
sns.distplot(sales_month_shop_item['sum_sales'])
plt.title('Distribution of total sales per month for shop-item combinations');
del sales_month_shop_item
```



[32]:
```
#scatter plot of item_id and date_block_num
sales1 = train.copy()
sales1['min_month'] = sales1.groupby(['shop_id'])['date_block_num'].
 ↪transform('min')
plt.scatter(sales1.min_month, sales1.shop_id)
del sales1
```



```
[33]: #scatter plot of item_id and date_block_num
      sales1 = train.copy()
      sales1['min_month'] = sales1.groupby(['item_id'])['date_block_num'].
       ↪transform('min')
      plt.figure(figsize=(20,40))
      plt.scatter(sales1.min_month, sales1.item_id)
      del sales1
```


20000
15000
10000
5000
0
0
5
10
15
20
25
30

There is no apparent pattern for item id numbers overall - it does not appear to be the case that the later an item is released, the larger the id number.

This is a similar story for shop_id; no apparent relationship exists between shop_id and sales.

1.2 Data Preprocessing

1.2.1 Shops dataframe

This involves feature extraction from text, namely splitting city name from shop name and label encoded. These codes will be used later for mean encoding.

[34]:
```
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: '   ' if '   ' in x else␣
 ↪'  ' if '  ' in x else '  ' if '  ' in x else ' ' if ' ' in x else ' ' if ' '␣
 ↪in x else 'NO_DATA')
shops.head()
```

[34]:
```
                         shop_name  shop_id shop_city shop_type
0                                0         NO_DATA
1                               1
2                                     2
3                             3
4                                   4
```

[35]:
```
#label encode city name
shops['city_id'] = LabelEncoder().fit_transform(shops['shop_city'])
shops['shop_type_id'] = LabelEncoder().fit_transform(shops['shop_type'])

#retain only numerical codes
shops = shops[['shop_id','shop_type_id','city_id']]

shops.head()
```

[35]:
```
   shop_id  shop_type_id  city_id
0        0             0       29
1        1             5       29
2        2             5        0
3        3             3        1
4        4             5        2
```

1.2.2 Category dataframe

As noted in exploration, category name contains type and subtype of item. This section involves feature extraction from text, namely splitting item category name into type and subtype of the item and label encoding. These codes will be used later for the purposes of mean encoding.

```
[36]: #split and separate out item type
      cats['split'] = cats['item_category_name'].str.split('-')
      cats['type'] = cats['split'].map(lambda x: x[0].strip())
      cats.head()
```

```
[36]:        item_category_name  item_category_id                      split  \
      0  PC -      /                           0  [PC ,       /    ]
      1                - PS2                       1          [       ,  PS2]
      2                - PS3                       2          [       ,  PS3]
      3                - PS4                       3          [       ,  PS4]
      4                - PSP                       4          [       ,  PSP]

                type
      0           PC
      1
      2
      3
      4
```

```
[37]: #label encode item type
      cats['cat_type_id'] = LabelEncoder().fit_transform(cats['type'])
```

```
[38]: # separate out item subtype, if no subtype replace with type
      cats['subtype'] = cats['split'].map(lambda x: x[1].strip() if len(x) > 1 else␣
       ↪x[0].strip())

      #label encode item subtype
      cats['cat_subtype_id'] = LabelEncoder().fit_transform(cats['subtype'])
```

```
[39]: #retain only nnumerical codes
      cats = cats[['item_category_id','cat_type_id', 'cat_subtype_id']]
      cats.head()
```

```
[39]:    item_category_id  cat_type_id  cat_subtype_id
      0                 0            0              29
      1                 1            1               9
      2                 2            1              10
      3                 3            1              11
      4                 4            1              13
```

1.2.3 Items dataframe

```
[259]: items.head()
```

```
[259]:                                          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
```

Encode "features" that many items have.

The structure is always the same Item name [category feature] (additional feature)

This can be split, and encoded.

```
[260]: from collections import Counter
from operator import itemgetter
items['name_1'], items['name_2'] = items['item_name'].str.split('[', 1).str
items['name_1'], items['name_3'] = items['item_name'].str.split('(', 1).str

items['name_2'] = items['name_2'].str.replace('[^A-Za-z0-9 - - ]+', ' ').str.
 ↪lower()
items['name_3'] = items['name_3'].str.replace('[^A-Za-z0-9 - - ]+', ' ').str.
 ↪lower()
items = items.fillna('0')

result_1 = Counter(' '.join(items['name_2'].values.tolist()).split(' ')).items()
result_1 = sorted(result_1, key=itemgetter(1))
result_1 = pd.DataFrame(result_1, columns=['feature', 'count'])
result_1 = result_1[(result_1['feature'].str.len() > 1) & (result_1['count'] >␣
 ↪200)]

result_2 = Counter(' '.join(items['name_3'].values.tolist()).split(" ")).items()
result_2 = sorted(result_2, key=itemgetter(1))
result_2 = pd.DataFrame(result_2, columns=['feature', 'count'])
result_2 = result_2[(result_2['feature'].str.len() > 1) & (result_2['count'] >␣
 ↪200)]

result = pd.concat([result_1, result_2])
result = result.drop_duplicates(subset=['feature'])

print('Most common additional features:', result)
```

```
Most common additional features:            feature  count
130             284
131              340
132                399
133               400
134            360    465
135          jewel    552
136           xbox    589
137            ps3    611
138               1428
139              1995
140             pc   2585
141               3427
1981           box    246
1983            3d    409
1985           dvd    503
1986      digipack    541
1988                757
1991           mp3    854
1992            cd    871
1993              1849
1994            bd   2320
```

Item name correction

For our basic “name feature” it is enough to find identical items (not similar but identical),

```
[261]: print('Unique item names:', len(items['item_name'].unique()))
```

```
Unique item names: 22170
```

```
[262]: items.name_1.nunique(), items.name_2.nunique(), items.name_3.nunique(), items.
        ↪item_category_id.nunique()
```

```
[262]: (20611, 175, 1666, 84)
```

```
[263]: import re
def name_correction(x):
    x = x.lower()
    x = x.partition('[')[0]
    x = x.partition('(')[0]
    x = re.sub('[^A-Za-z0-9 - - ]+', ' ', x)
    x = x.replace('  ', ' ')
    x = x.strip()
    return x

items['name_1'] = items['name_1'].apply(lambda x: name_correction(x))
items.head()
```

```
[263]:                                          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                                       name_1  \
       0                40
       1                76  abbyy finereader 12 professional edition full
       2                40
       3                40
       4                40

                       name_2    name_3
       0                    0       d
       1  pc                    0
       2                    0     unv d
       3                    0    univ d
       4                    0      d
```

```
[264]: items.name_1.nunique(), items.name_2.nunique(), items.name_3.nunique(), items.
        ↪item_category_id.nunique()
```

```
[264]: (18121, 175, 1666, 84)
```

```
[45]: print('Unique item names after correction:', len(items['item_name'].unique()))
```

```
Unique item names after correction: 18121
```

```
[149]: #label encode name_1, 2, and 3
       #items['item_name_id'] = LabelEncoder().fit_transform(items['name_1'])

       items['item_type_id'] = LabelEncoder().fit_transform(items['name_2'])
       items['item_subtype_id'] = LabelEncoder().fit_transform(items['name_3'])

       items = items.drop(['item_name','name_2','name_3'], axis=1)
```

1.2.4 Monthly sales

As noted previously, some item-shop combinations in the test set are not present in the train set - the train set includes only items that have been sold in the past, thus the test items not present in the train have not, and as predictions are for the purposes of this task in the future, the target value should be zero. To ensure these items are accounted for in the model, all possible item-shop combinations need to be included for each month and set at zero.

Furthermore, the task is to predict monthly sales. So, daily sales reported need to be aggregated to monthly.

[48]:
```
# For every month create a grid from all shops/items combinations from that␣
↪month

grid = []

for block_num in train['date_block_num'].unique():
    cur_shops = train[train['date_block_num']==block_num]['shop_id'].unique()
    cur_items = train[train['date_block_num']==block_num]['item_id'].unique()
    grid.append(np.array(list(product(*[cur_shops, cur_items,␣
↪[block_num]])),dtype='int32'))

# #turn the grid into pandas dataframe
index_cols = ['shop_id', 'item_id', 'date_block_num']
grid = pd.DataFrame(np.vstack(grid), columns = index_cols,dtype=np.int32)


index_cols = ['shop_id', 'item_id', 'date_block_num']
train['item_cnt_day'] = train['item_cnt_day'].clip(0,20)
gb_cnt = train.groupby(index_cols)['item_cnt_day'].agg(['sum']).reset_index().
↪rename(columns = {'sum': 'item_cnt_month'})
gb_cnt['item_cnt_month'] = gb_cnt['item_cnt_month'].clip(0,20).astype(np.int)
```

[49]:
```
#join aggregated data to the grid
train = pd.merge(grid,gb_cnt,how='left',on=index_cols).fillna(0)
train['item_cnt_month'] = train['item_cnt_month'].astype(int)
train = downcast_dtypes(train)
```

[50]:
```
#sort the data
train.sort_values(['date_block_num','shop_id','item_id'],inplace=True)
```

[51]:
```
#add additional column for shop-item
train['shop_item_id'] = train['shop_id'].apply(str) + '_' + train['item_id'].
↪apply(str)
test['shop_item_id'] = test['shop_id'].apply(str) + '_' + test['item_id'].
↪apply(str)
```

1.2.5 Add codes from category, item and shop dataframes to main dataset

These codes are needed for mean encoding next.

[52]:
```
sales = pd.merge(train, shops, on=['shop_id'], how='left')
sales = pd.merge(sales, items, on=['item_id'], how='left')
sales = pd.merge(sales, cats, on=['item_category_id'], how='left')

test = pd.merge(test, shops, on=['shop_id'], how='left')
test = pd.merge(test, items, on=['item_id'], how='left')
test = pd.merge(test, cats, on=['item_category_id'], how='left')
```

```
[53]: #reduce data size
      sales = downcast_dtypes(sales)
```

```
[54]: del shops
      del items
      del cats
```

```
[55]: sales.head()
```

```
[55]:    shop_id  item_id  date_block_num  item_cnt_month shop_item_id  \
      0        2       19               0               0         2_19
      1        2       27               0               1         2_27
      2        2       28               0               0         2_28
      3        2       29               0               0         2_29
      4        2       32               0               0         2_32

         shop_type_id  city_id  item_category_id  item_type_id  item_subtype_id  \
      0             5        0                40             4               42
      1             5        0                19            77               42
      2             5        0                30           108               42
      3             5        0                23           124               42
      4             5        0                40             4               42

         cat_type_id  cat_subtype_id
      0           11               4
      1            5              10
      2            8              55
      3            5              16
      4           11               4
```

1.2.6 Mean encode features

Mean encode categorical features using KFold, LOO, Smoothing and Expanding and select the version of each feature with the highest correlation coefficient.

As item_item_month for the test set will not be available to include in mean encoding, these encodings will be conducted on the data minus the test set.

```
[56]: #global mean set at group mean as train_df produces NaN due to large number of␣
       ↪0 values
      mean_encoded_col =␣
       ↪['shop_id','item_id','shop_item_id','shop_type_id','city_id','item_category_id','item_type_
                           'item_subtype_id','cat_type_id','cat_subtype_id']

      from tqdm import tqdm
      from sklearn.model_selection import KFold

      Target = 'item_cnt_month'
```

```
global_mean = sales[Target].mean()
y_tr = sales[Target].values

for col in tqdm(mean_encoded_col):
    col_tr = sales[[col] + [Target]]
    corrcoefs = pd.DataFrame(columns = ['Cor'])

    # 3.1.1 Mean encodings - KFold scheme
    kf = KFold(n_splits = 5, shuffle = False, random_state = 0)
    col_tr[col + '_cnt_month_mean_Kfold'] = np.nan

    for tr_ind, val_ind in kf.split(col_tr):
        X_tr, X_val = col_tr.iloc[tr_ind], col_tr.iloc[val_ind]
        means = X_val[col].map(X_tr.groupby(col)[Target].mean())
        X_val[col + '_cnt_month_mean_Kfold'] = means
        col_tr.iloc[val_ind] = X_val

    col_tr.fillna(global_mean, inplace = True)
    corrcoefs.loc[col + '_cnt_month_mean_Kfold'] = np.corrcoef(y_tr, col_tr[col␣
 ↪+ '_cnt_month_mean_Kfold'])[0][1]


     # 3.1.2 Mean encodings - Leave-one-out scheme
    item_id_target_sum = col_tr.groupby(col)[Target].sum()
    item_id_target_count = col_tr.groupby(col)[Target].count()
    col_tr[col + '_cnt_month_sum'] = col_tr[col].map(item_id_target_sum)
    col_tr[col + '_cnt_month_count'] = col_tr[col].map(item_id_target_count)
    col_tr[col + '_target_mean_LOO'] = (col_tr[col + '_cnt_month_sum'] -␣
 ↪col_tr[Target]) / (col_tr[col + '_cnt_month_count'] - 1)
    col_tr.fillna(global_mean, inplace = True)
    corrcoefs.loc[col + '_target_mean_LOO'] = np.corrcoef(y_tr, col_tr[col +␣
 ↪'_target_mean_LOO'])[0][1]


     # 3.1.3 Mean encodings - Smoothing
    item_id_target_mean = col_tr.groupby(col)[Target].mean()
    item_id_target_count = col_tr.groupby(col)[Target].count()
    col_tr[col + '_cnt_month_mean'] = col_tr[col].map(item_id_target_mean)
    col_tr[col + '_cnt_month_count'] = col_tr[col].map(item_id_target_count)
    alpha = 100
    col_tr[col + '_cnt_month_mean_Smooth'] = (col_tr[col + '_cnt_month_mean'] *␣
 ↪ col_tr[col + '_cnt_month_count'] + global_mean * alpha) / (alpha +␣
 ↪col_tr[col + '_cnt_month_count'])
    col_tr[col + '_cnt_month_mean_Smooth'].fillna(global_mean, inplace=True)
    corrcoefs.loc[col + '_cnt_month_mean_Smooth'] = np.corrcoef(y_tr,␣
 ↪col_tr[col + '_cnt_month_mean_Smooth'])[0][1]
```

```
    # 3.1.4 Mean encodings - Expanding mean scheme
    cumsum = col_tr.groupby(col)[Target].cumsum() - col_tr[Target]
    sumcnt = col_tr.groupby(col).cumcount()
    col_tr[col + '_cnt_month_mean_Expanding'] = cumsum / sumcnt
    col_tr[col + '_cnt_month_mean_Expanding'].fillna(global_mean, inplace=True)
    corrcoefs.loc[col + '_cnt_month_mean_Expanding'] = np.corrcoef(y_tr,␣
↪col_tr[col + '_cnt_month_mean_Expanding'])[0][1]

    sales = pd.concat([sales, col_tr[corrcoefs['Cor'].idxmax()]], axis = 1)
    print(corrcoefs.sort_values('Cor'))
```

```
 10%|          | 1/10 [00:04<00:38,  4.28s/it]

                                       Cor
shop_id_cnt_month_mean_Kfold      0.172836
shop_id_target_mean_LOO           0.174991
shop_id_cnt_month_mean_Smooth     0.175016
shop_id_cnt_month_mean_Expanding  0.175150

 20%|          | 2/10 [00:09<00:36,  4.53s/it]

                                       Cor
item_id_cnt_month_mean_Kfold      0.312957
item_id_cnt_month_mean_Smooth     0.479641
item_id_target_mean_LOO           0.481724
item_id_cnt_month_mean_Expanding  0.565665

 30%|          | 3/10 [00:51<01:50, 15.81s/it]

                                            Cor
shop_item_id_cnt_month_mean_Kfold      0.423936
shop_item_id_cnt_month_mean_Expanding  0.542637
shop_item_id_target_mean_LOO           0.577498
shop_item_id_cnt_month_mean_Smooth     0.600000

 40%|          | 4/10 [00:55<01:13, 12.30s/it]

                                            Cor
shop_type_id_cnt_month_mean_Kfold      0.034009
shop_type_id_target_mean_LOO           0.037720
shop_type_id_cnt_month_mean_Smooth     0.037738
shop_type_id_cnt_month_mean_Expanding  0.039478

 50%|          | 5/10 [00:59<00:49,  9.83s/it]

                                       Cor
city_id_cnt_month_mean_Kfold      0.117109
city_id_target_mean_LOO           0.119897
city_id_cnt_month_mean_Smooth     0.119918
city_id_cnt_month_mean_Expanding  0.120103
```

```
 60%|        | 6/10 [01:04<00:32,  8.21s/it]

                                                 Cor
item_category_id_cnt_month_mean_Kfold      0.270138
item_category_id_cnt_month_mean_Smooth     0.289958
item_category_id_target_mean_LOO           0.289988
item_category_id_cnt_month_mean_Expanding  0.292890

 70%|       | 7/10 [01:08<00:21,  7.14s/it]

                                             Cor
item_type_id_cnt_month_mean_Kfold      0.194394
item_type_id_cnt_month_mean_Smooth     0.213486
item_type_id_target_mean_LOO           0.213759
item_type_id_cnt_month_mean_Expanding  0.225101

 80%|      | 8/10 [01:13<00:12,  6.33s/it]

                                                Cor
item_subtype_id_cnt_month_mean_Kfold      0.221293
item_subtype_id_cnt_month_mean_Smooth     0.250972
item_subtype_id_target_mean_LOO           0.251816
item_subtype_id_cnt_month_mean_Expanding  0.258824

 90%|      | 9/10 [01:17<00:05,  5.71s/it]

                                            Cor
cat_type_id_cnt_month_mean_Kfold      0.155011
cat_type_id_target_mean_LOO           0.169346
cat_type_id_cnt_month_mean_Smooth     0.169398
cat_type_id_cnt_month_mean_Expanding  0.174876

100%|      | 10/10 [01:21<00:00,  8.19s/it]

                                               Cor
cat_subtype_id_cnt_month_mean_Kfold      0.270167
cat_subtype_id_cnt_month_mean_Smooth     0.288706
cat_subtype_id_target_mean_LOO           0.288749
cat_subtype_id_cnt_month_mean_Expanding  0.290632
```

1.2.7 Combine test and train

[57]:
```
if Validation == False:
    test['date_block_num'] = 34
    all_data = pd.concat([sales, test], axis = 0)
    all_data = all_data.drop(columns = ['ID'])

else:
    all_data = sales
```

1.2.8 Feature Generation

1.2.9 1. Create time lagged features

[58]:
```
sales = downcast_dtypes(all_data)
```

[59]:
```
#function to create time lags
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
```

[60]:
```
#lagging sales for shop-item combinations per month
sales = lag_feature(sales, [1,2,3,4,5,6,7,8,9,10,11,12], 'item_cnt_month')

#as prediction months will have no mean encoded features and some of these vary␣
 ↪by month,
#they will be lagged with current ones removed

cols = ['shop_id_cnt_month_mean_Expanding',
        'item_id_cnt_month_mean_Expanding',
        'shop_item_id_cnt_month_mean_Smooth',
        'shop_type_id_cnt_month_mean_Expanding',
        'city_id_cnt_month_mean_Expanding',
        'item_category_id_cnt_month_mean_Expanding',
        'item_type_id_cnt_month_mean_Expanding',
        'item_subtype_id_cnt_month_mean_Expanding',
        'cat_type_id_cnt_month_mean_Expanding',
        'cat_subtype_id_cnt_month_mean_Expanding']

shift_range = [1, 2, 3, 4, 12]

for col in cols:
    sales = lag_feature(sales, shift_range, col)

sales.head()
```

[60]:
```
   cat_subtype_id  cat_subtype_id_cnt_month_mean_Expanding  cat_type_id  \
0               4                                 0.311493           11
1              10                                 0.311493            5
2              55                                 0.311493            8
3              16                                 0.311493            5
```

```
4              4                                0.000000            11

   cat_type_id_cnt_month_mean_Expanding  city_id  \
0                              0.311493        0
1                              0.311493        0
2                              0.311493        0
3                              1.000000        0
4                              0.000000        0

   city_id_cnt_month_mean_Expanding  date_block_num  item_category_id  \
0                          0.311493               0                40
1                          0.000000               0                19
2                          0.500000               0                30
3                          0.333333               0                23
4                          0.250000               0                40

   item_category_id_cnt_month_mean_Expanding  item_cnt_month  item_id  \
0                                   0.311493             0.0       19
1                                   0.311493             1.0       27
2                                   0.311493             0.0       28
3                                   0.311493             0.0       29
4                                   0.000000             0.0       32

   item_id_cnt_month_mean_Expanding  item_subtype_id  \
0                          0.311493               42
1                          0.311493               42
2                          0.311493               42
3                          0.311493               42
4                          0.311493               42

   item_subtype_id_cnt_month_mean_Expanding  item_type_id  \
0                                  0.311493             4
1                                  0.000000            77
2                                  0.500000           108
3                                  0.333333           124
4                                  0.250000             4

   item_type_id_cnt_month_mean_Expanding  shop_id  \
0                               0.311493        2
1                               0.311493        2
2                               0.311493        2
3                               0.311493        2
4                               0.000000        2

   shop_id_cnt_month_mean_Expanding shop_item_id  \
0                          0.311493         2_19
1                          0.000000         2_27
```

```
2                          0.500000         2_28
3                          0.333333         2_29
4                          0.250000         2_32

   shop_item_id_cnt_month_mean_Smooth  shop_type_id  \
0                            0.308409             5
1                            0.288254             5
2                            0.275657             5
3                            0.291115             5
4                            0.314547             5

   shop_type_id_cnt_month_mean_Expanding  item_cnt_month_lag_1  \
0                               0.311493                   NaN
1                               0.000000                   NaN
2                               0.500000                   NaN
3                               0.333333                   NaN
4                               0.250000                   NaN

   item_cnt_month_lag_2  item_cnt_month_lag_3  …  \
0                   NaN                   NaN  …
1                   NaN                   NaN  …
2                   NaN                   NaN  …
3                   NaN                   NaN  …
4                   NaN                   NaN  …

   item_category_id_cnt_month_mean_Expanding_lag_1  \
0                                              NaN
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

   item_category_id_cnt_month_mean_Expanding_lag_2  \
0                                              NaN
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

   item_category_id_cnt_month_mean_Expanding_lag_3  \
0                                              NaN
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

   item_category_id_cnt_month_mean_Expanding_lag_4  \
0                                                   NaN
1                                                   NaN
2                                                   NaN
3                                                   NaN
4                                                   NaN

   item_category_id_cnt_month_mean_Expanding_lag_12  \
0                                                   NaN
1                                                   NaN
2                                                   NaN
3                                                   NaN
4                                                   NaN

   item_type_id_cnt_month_mean_Expanding_lag_1  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   item_type_id_cnt_month_mean_Expanding_lag_2  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   item_type_id_cnt_month_mean_Expanding_lag_3  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   item_type_id_cnt_month_mean_Expanding_lag_4  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   item_type_id_cnt_month_mean_Expanding_lag_12  \
0                                            NaN
1                                            NaN
2                                            NaN
3                                            NaN
4                                            NaN
```

```
   item_subtype_id_cnt_month_mean_Expanding_lag_1  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_2  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_3  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_4  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_12  \
0                                              NaN
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

   cat_type_id_cnt_month_mean_Expanding_lag_1  \
0                                         NaN
1                                         NaN
2                                         NaN
3                                         NaN
4                                         NaN

   cat_type_id_cnt_month_mean_Expanding_lag_2  \
0                                         NaN
1                                         NaN
2                                         NaN
```

```
3                                        NaN
4                                        NaN

   cat_type_id_cnt_month_mean_Expanding_lag_3  \
0                                        NaN
1                                        NaN
2                                        NaN
3                                        NaN
4                                        NaN

   cat_type_id_cnt_month_mean_Expanding_lag_4  \
0                                        NaN
1                                        NaN
2                                        NaN
3                                        NaN
4                                        NaN

   cat_type_id_cnt_month_mean_Expanding_lag_12  \
0                                         NaN
1                                         NaN
2                                         NaN
3                                         NaN
4                                         NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_1  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_2  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_3  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_4  \
0                                           NaN
```

```
1                                                       NaN
2                                                       NaN
3                                                       NaN
4                                                       NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_12
0                                              NaN
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

[5 rows x 84 columns]
```

[61]:
```
#as these will not be present for test set, only the lagged ones will be␣
↪retained
sales = sales.drop(cols, axis=1)
```

[62]:
```
sales = downcast_dtypes(sales)
```

[63]:
```
sales.to_pickle('data_1.pickle.gzde', compression='gzip')
```

[64]:
```
del sales
```

1.2.10 2. Item price trend over the previous 6 months

Price will influence demand for a product, and thus sales.

[65]:
```
sales = pd.read_pickle('data_1.pickle.gzde', compression='gzip')
train = pd.read_pickle('train_alt.pickle.gzde', compression='gzip')
```

[66]:
```
# calculate mean item price
group = train.groupby(['item_id']).agg({'item_price': ['mean']})
group.columns = ['avg_item_price']
group.reset_index(inplace=True)

#add to df
sales = pd.merge(sales, group, on=['item_id'], how='left')
sales['avg_item_price'] = sales['avg_item_price'].astype(np.float16)
```

[67]:
```
# calculate mean item price per month
group = train.groupby(['date_block_num','item_id']).agg({'item_price':␣
↪['mean']})
group.columns = ['avg_item_price_month']
group.reset_index(inplace=True)

#add to df
```

```
sales = pd.merge(sales, group, on=['date_block_num','item_id'], how='left')
sales['avg_item_price_month'] = sales['avg_item_price_month'].astype(np.float16)
```

[68]:
```
#lag for price trend
lags = [1,2,3,4,5,6]
sales = lag_feature(sales, lags, 'avg_item_price_month')
```

[69]:
```
for i in lags:
    sales['delta_price_lag_'+str(i)] = \
        (sales['avg_item_price_month_lag_'+str(i)] - sales['avg_item_price']) /␣
 ↪sales['avg_item_price']
```

[70]:
```
def select_trend(row):
    for i in lags:
        if row['delta_price_lag_'+str(i)]:
            return row['delta_price_lag_'+str(i)]
    return 0
```

[71]:
```
sales['delta_price_lag'] = sales.apply(select_trend, axis=1)
sales['delta_price_lag'] = sales['delta_price_lag'].astype(np.float16)
sales['delta_price_lag'].fillna(0, inplace=True)

features_to_drop = ['avg_item_price_month', 'avg_item_price']
for i in lags:
    features_to_drop += ['avg_item_price_month_lag_'+str(i)]
    features_to_drop += ['delta_price_lag_'+str(i)]
```

[72]:
```
sales.drop(features_to_drop, axis=1, inplace=True)
```

[73]:
```
sales.to_pickle('data_2.pickle.gzde', compression='gzip')
```

[74]:
```
del sales
```

1.2.11 3. Shop revenue trends

[75]:
```
sales = pd.read_pickle('data_2.pickle.gzde', compression='gzip')
```

[76]:
```
#total revenue per shop per month
train['revenue'] = train['item_price'] *  train['item_cnt_day']
group = train.groupby(['date_block_num','shop_id']).agg({'revenue': ['sum']})
group.columns = ['shop_revenue_month']
group.reset_index(inplace=True)

sales = pd.merge(sales, group, on=['date_block_num','shop_id'], how='left')
sales['shop_revenue_month'] = sales['shop_revenue_month'].astype(np.float32)
```

[77]:
```
#average revenue per shop per month
group = group.groupby(['shop_id']).agg({'shop_revenue_month': ['mean']})
group.columns = ['shop_avg_revenue']
group.reset_index(inplace=True)

sales = pd.merge(sales, group, on=['shop_id'], how='left')
sales['shop_avg_revenue'] = sales['shop_avg_revenue'].astype(np.float32)

sales['delta_revenue'] = (sales['shop_revenue_month'] -␣
 ↪sales['shop_avg_revenue']) / sales['shop_avg_revenue']

sales['delta_revenue'] = sales['delta_revenue'].astype(np.float16)
```

[78]:
```
#lag revenue features
sales = lag_feature(sales, [1], 'delta_revenue')
```

[79]:
```
#drop present month revenue features
sales.drop(['shop_revenue_month','shop_avg_revenue','delta_revenue'], axis=1,␣
 ↪inplace=True)
```

[80]:
```
sales.to_pickle('data_3.pickle.gzde', compression='gzip')
```

[81]:
```
del sales
```

1.2.12 4. Date features

[82]:
```
sales = pd.read_pickle('data_3.pickle.gzde', compression='gzip')
```

[83]:
```
dates_train = train[['date', 'date_block_num']].drop_duplicates()
dates_test = dates_train[dates_train['date_block_num'] == 34-12]
dates_test['date_block_num'] = 34
dates_test['date'] = dates_test['date'] + pd.DateOffset(years=1)
dates_all = pd.concat([dates_train, dates_test])

dates_all['dow'] = dates_all['date'].dt.dayofweek
dates_all['year'] = dates_all['date'].dt.year
dates_all['month'] = dates_all['date'].dt.month
dates_all = pd.get_dummies(dates_all, columns=['dow'])
dow_col = ['dow_' + str(x) for x in range(7)]
date_features = dates_all.groupby(['year', 'month', 'date_block_num'])[dow_col].
 ↪agg('sum').reset_index()
date_features['days_of_month'] = date_features[dow_col].sum(axis=1)
date_features['year'] = date_features['year'] - 2013

date_features = date_features[['month', 'year', 'days_of_month',␣
 ↪'date_block_num']]
```

```
sales = sales.merge(date_features, on = 'date_block_num', how = 'left')
date_columns = date_features.columns.difference(set(index_cols))
```

[84]:
```
sales.head()
```

[84]:
```
   cat_subtype_id  cat_type_id  city_id  date_block_num  item_category_id  \
0               4           11        0               0                40
1              10            5        0               0                19
2              55            8        0               0                30
3              16            5        0               0                23
4               4           11        0               0                40

   item_cnt_month  item_id  item_subtype_id  item_type_id  shop_id  \
0             0.0       19               42             4        2
1             1.0       27               42            77        2
2             0.0       28               42           108        2
3             0.0       29               42           124        2
4             0.0       32               42             4        2

  shop_item_id  shop_type_id  item_cnt_month_lag_1  item_cnt_month_lag_2  \
0         2_19             5                   NaN                   NaN
1         2_27             5                   NaN                   NaN
2         2_28             5                   NaN                   NaN
3         2_29             5                   NaN                   NaN
4         2_32             5                   NaN                   NaN

   item_cnt_month_lag_3  item_cnt_month_lag_4  item_cnt_month_lag_5  \
0                   NaN                   NaN                   NaN
1                   NaN                   NaN                   NaN
2                   NaN                   NaN                   NaN
3                   NaN                   NaN                   NaN
4                   NaN                   NaN                   NaN

   item_cnt_month_lag_6  item_cnt_month_lag_7  item_cnt_month_lag_8  \
0                   NaN                   NaN                   NaN
1                   NaN                   NaN                   NaN
2                   NaN                   NaN                   NaN
3                   NaN                   NaN                   NaN
4                   NaN                   NaN                   NaN

   item_cnt_month_lag_9  item_cnt_month_lag_10  item_cnt_month_lag_11  \
0                   NaN                    NaN                    NaN
1                   NaN                    NaN                    NaN
2                   NaN                    NaN                    NaN
3                   NaN                    NaN                    NaN
4                   NaN                    NaN                    NaN
```

```
   item_cnt_month_lag_12  shop_id_cnt_month_mean_Expanding_lag_1  …  \
0                    NaN                                     NaN  …
1                    NaN                                     NaN  …
2                    NaN                                     NaN  …
3                    NaN                                     NaN  …
4                    NaN                                     NaN  …

   item_type_id_cnt_month_mean_Expanding_lag_1  \
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN

   item_type_id_cnt_month_mean_Expanding_lag_2  \
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN

   item_type_id_cnt_month_mean_Expanding_lag_3  \
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN

   item_type_id_cnt_month_mean_Expanding_lag_4  \
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN

   item_type_id_cnt_month_mean_Expanding_lag_12  \
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_1  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
```

```
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_2  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_3  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_4  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   item_subtype_id_cnt_month_mean_Expanding_lag_12  \
0                                              NaN
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

   cat_type_id_cnt_month_mean_Expanding_lag_1  \
0                                         NaN
1                                         NaN
2                                         NaN
3                                         NaN
4                                         NaN

   cat_type_id_cnt_month_mean_Expanding_lag_2  \
0                                         NaN
1                                         NaN
2                                         NaN
3                                         NaN
4                                         NaN

   cat_type_id_cnt_month_mean_Expanding_lag_3  \
0                                         NaN
1                                         NaN
```

```
2                                         NaN
3                                         NaN
4                                         NaN

   cat_type_id_cnt_month_mean_Expanding_lag_4  \
0                                         NaN
1                                         NaN
2                                         NaN
3                                         NaN
4                                         NaN

   cat_type_id_cnt_month_mean_Expanding_lag_12  \
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_1  \
0                                            NaN
1                                            NaN
2                                            NaN
3                                            NaN
4                                            NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_2  \
0                                            NaN
1                                            NaN
2                                            NaN
3                                            NaN
4                                            NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_3  \
0                                            NaN
1                                            NaN
2                                            NaN
3                                            NaN
4                                            NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_4  \
0                                            NaN
1                                            NaN
2                                            NaN
3                                            NaN
4                                            NaN

   cat_subtype_id_cnt_month_mean_Expanding_lag_12  delta_price_lag  \
0                                                    NaN                 0.0
1                                                    NaN                 0.0
2                                                    NaN                 0.0
3                                                    NaN                 0.0
4                                                    NaN                 0.0

   delta_revenue_lag_1  month  year  days_of_month
0                  NaN      1     0             31
1                  NaN      1     0             31
2                  NaN      1     0             31
3                  NaN      1     0             31
4                  NaN      1     0             31

[5 rows x 79 columns]
```

1.2.13 5. Months since the last sale and first sale for each shop-item combination

Each row will be iterated through, treating {shop_id, item_id} as the key date_block_num as values. If the key is not in the cache and not equal to 0, the key-value pair is added to the cache. If the key is in the cache already, the difference between the current and previous date_block_num value is calculated.

These will need to be lagged as if the present month is the first time sales for a shop-item combination are recorded, it will be reported as 0 months, which would not be possible for sales that have yet to be recorded for prediction purposes.

[85]:
```
#months since last sale of shop-item combination
cache = {}
sales['months_since_item_shop_last_sale'] = -1
sales['months_since_item_shop_last_sale'] =␣
 ↪sales['months_since_item_shop_last_sale'].astype(np.int8)
for idx, row in sales.iterrows():
    key = str(row.item_id)+' '+str(row.shop_id)
    if key not in cache:
        if row.item_cnt_month!=0:
            cache[key] = row.date_block_num
    else:
        last_date_block_num = cache[key]
        sales.at[idx, 'months_since_item_shop_last_sale'] = row.date_block_num␣
 ↪- last_date_block_num
        cache[key] = row.date_block_num
```

[86]:
```
train = pd.read_pickle('train_alt.pickle.gzde', compression='gzip')
```

[87]:
```
group = train.groupby(['date_block_num','shop_id','item_id']).
 ↪agg({'item_cnt_day': ['sum']})
group.columns = ['item_cnt_month']
group.reset_index(inplace=True)
```

[88]:
```
#months since first sale of shop-item combination
group['months_since_shop_item_first_sale'] = group['date_block_num'] - group.
 ↪groupby(['shop_id','item_id'])['date_block_num'].transform('min')
group1 =␣
 ↪group[['date_block_num','shop_id','item_id','months_since_shop_item_first_sale']]
```

[89]:
```
sales = pd.merge(sales, group1, on=['date_block_num','shop_id','item_id'],␣
 ↪how='left')
sales['months_since_shop_item_first_sale'] =␣
 ↪sales['months_since_shop_item_first_sale'].fillna(-1)
```

[90]:
```
#lag features
sales = lag_feature(sales, [1], 'months_since_item_shop_last_sale')
sales = lag_feature(sales, [1], 'months_since_shop_item_first_sale')

sales = sales.
 ↪drop(['months_since_item_shop_last_sale','months_since_shop_item_first_sale'],␣
 ↪axis=1)
```

[91]:
```
sales.to_pickle('data_4.pickle.gzde', compression='gzip')
```

[92]:
```
del sales
```

1.2.14 6. Months since the last and first sale for each item only

The same approach is used as above, but with the key only being item_id.

[93]:
```
sales = pd.read_pickle('data_4.pickle.gzde', compression='gzip')
```

[94]:
```
sales = downcast_dtypes(sales)
```

[95]:
```
#months since last sale of item
cache = {}
sales['months_since_item_last_sale'] = -1
sales['months_since_item_last_sale'] = sales['months_since_item_last_sale'].
 ↪astype(np.int8)
for idx, row in sales.iterrows():
    key = row.item_id
    if key not in cache:
        if row.item_cnt_month!=0:
            cache[key] = row.date_block_num
    else:
        last_date_block_num = cache[key]
        if row.date_block_num>last_date_block_num:
            sales.at[idx, 'months_since_item_last_sale'] = row.date_block_num -␣
↪last_date_block_num
            cache[key] = row.date_block_num


sales = lag_feature(sales, [1], 'months_since_item_last_sale')
sales = sales.drop(['months_since_item_last_sale'], axis=1)
sales = downcast_dtypes(sales)
```

[96]:
```
train = pd.read_pickle('train_alt.pickle.gzde', compression='gzip')
```

[97]:
```
group = train.groupby(['date_block_num','shop_id','item_id']).
↪agg({'item_cnt_day': ['sum']})
group.columns = ['item_cnt_month']
group.reset_index(inplace=True)

#months since first sale of item
group['months_since_item_first_sale'] = group['date_block_num'] - group.
↪groupby(['item_id'])['date_block_num'].transform('min')
group =␣
↪group[['date_block_num','shop_id','item_id','months_since_item_first_sale']]

sales = pd.merge(sales, group, on=['date_block_num','shop_id','item_id'],␣
↪how='left')
sales['months_since_item_first_sale'] = sales['months_since_item_first_sale'].
↪fillna(-1)
sales = lag_feature(sales, [1], 'months_since_item_first_sale')
sales = sales.drop(['months_since_item_first_sale'], axis=1)
```

1.2.15 Drop first 12 months of data as lags incomplete

[98]:
```
sales = sales[sales.date_block_num > 11]
```

1.2.16 Fill remaining null values with zero

[99]:
```
sales = sales.fillna(0)
sales = downcast_dtypes(sales)
sales.head()
```

[99]:
```
         cat_subtype_id  cat_type_id  city_id  date_block_num  \
3395293               4           11        0              12
3395294               1           11        0              12
3395295               4           11        0              12
3395296               1           11        0              12
3395297               4           11        0              12
```

```
         item_category_id  item_cnt_month  item_id  item_subtype_id  \
3395293                40             0.0       30               42   
3395294                37             0.0       31              562   
3395295                40             1.0       32               42   
3395296                37             1.0       33              562   
3395297                40             0.0       34             1367   

         item_type_id  shop_id shop_item_id  shop_type_id  \
3395293             4        2         2_30             5   
3395294             4        2         2_31             5   
3395295             4        2         2_32             5   
3395296             4        2         2_33             5   
3395297             4        2         2_34             5   

         item_cnt_month_lag_1  item_cnt_month_lag_2  item_cnt_month_lag_3  \
3395293                   0.0                   0.0                   0.0   
3395294                   0.0                   0.0                   0.0   
3395295                   0.0                   0.0                   0.0   
3395296                   1.0                   2.0                   0.0   
3395297                   0.0                   0.0                   0.0   

         item_cnt_month_lag_4  item_cnt_month_lag_5  item_cnt_month_lag_6  \
3395293                   0.0                   0.0                   0.0   
3395294                   0.0                   0.0                   0.0   
3395295                   0.0                   0.0                   0.0   
3395296                   0.0                   0.0                   0.0   
3395297                   0.0                   0.0                   0.0   

         item_cnt_month_lag_7  item_cnt_month_lag_8  item_cnt_month_lag_9  \
3395293                   1.0                   0.0                   0.0   
3395294                   0.0                   0.0                   1.0   
3395295                   0.0                   0.0                   0.0   
3395296                   0.0                   0.0                   0.0   
3395297                   0.0                   0.0                   0.0   

         item_cnt_month_lag_10  item_cnt_month_lag_11  item_cnt_month_lag_12  \
3395293                    1.0                    0.0                    0.0   
3395294                    1.0                    4.0                    0.0   
3395295                    0.0                    0.0                    0.0   
3395296                    0.0                    0.0                    1.0   
3395297                    0.0                    0.0                    0.0   

         shop_id_cnt_month_mean_Expanding_lag_1  …  \
3395293                                0.099054  …   
3395294                                0.099053  …   
3395295                                0.099052  …   
3395296                                0.099051  …   
```

```
3395297                                    0.099062  …

         item_type_id_cnt_month_mean_Expanding_lag_12  \
3395293                                      0.000000
3395294                                      0.000000
3395295                                      0.000000
3395296                                      0.000000
3395297                                      0.333333

         item_subtype_id_cnt_month_mean_Expanding_lag_1  \
3395293                                        0.384450
3395294                                        0.201874
3395295                                        0.384450
3395296                                        0.201874
3395297                                        0.192280

         item_subtype_id_cnt_month_mean_Expanding_lag_2  \
3395293                                        0.382351
3395294                                        0.204375
3395295                                        0.382351
3395296                                        0.204375
3395297                                        0.194044

         item_subtype_id_cnt_month_mean_Expanding_lag_3  \
3395293                                        0.382264
3395294                                        0.207749
3395295                                        0.382264
3395296                                        0.207748
3395297                                        0.195743

         item_subtype_id_cnt_month_mean_Expanding_lag_4  \
3395293                                        0.379615
3395294                                        0.207921
3395295                                        0.379615
3395296                                        0.207920
3395297                                        0.199872

         item_subtype_id_cnt_month_mean_Expanding_lag_12  \
3395293                                         0.000000
3395294                                         0.000000
3395295                                         0.250000
3395296                                         0.311493
3395297                                         0.311493

         cat_type_id_cnt_month_mean_Expanding_lag_1  \
3395293                                    0.238131
3395294                                    0.238130
```

```
3395295                                      0.238130
3395296                                      0.238130
3395297                                      0.238131

         cat_type_id_cnt_month_mean_Expanding_lag_2  \
3395293                                      0.238516
3395294                                      0.238516
3395295                                      0.238516
3395296                                      0.238515
3395297                                      0.238517

         cat_type_id_cnt_month_mean_Expanding_lag_3  \
3395293                                      0.240038
3395294                                      0.240038
3395295                                      0.240038
3395296                                      0.240037
3395297                                      0.240037

         cat_type_id_cnt_month_mean_Expanding_lag_4  \
3395293                                      0.242356
3395294                                      0.242356
3395295                                      0.242356
3395296                                      0.242355
3395297                                      0.242355

         cat_type_id_cnt_month_mean_Expanding_lag_12  \
3395293                                       0.000000
3395294                                       0.000000
3395295                                       0.000000
3395296                                       0.000000
3395297                                       0.333333

         cat_subtype_id_cnt_month_mean_Expanding_lag_1  \
3395293                                         0.254690
3395294                                         0.205493
3395295                                         0.254689
3395296                                         0.205492
3395297                                         0.254689

         cat_subtype_id_cnt_month_mean_Expanding_lag_2  \
3395293                                         0.254084
3395294                                         0.207230
3395295                                         0.254084
3395296                                         0.207229
3395297                                         0.254084

         cat_subtype_id_cnt_month_mean_Expanding_lag_3  \
3395293                                      0.254229
3395294                                      0.210747
3395295                                      0.254229
3395296                                      0.210746
3395297                                      0.254229

         cat_subtype_id_cnt_month_mean_Expanding_lag_4  \
3395293                                      0.256372
3395294                                      0.212243
3395295                                      0.256372
3395296                                      0.212243
3395297                                      0.256371

         cat_subtype_id_cnt_month_mean_Expanding_lag_12  delta_price_lag  \
3395293                                       0.000000        -0.478760
3395294                                       0.000000        -0.127563
3395295                                       0.000000        -0.407959
3395296                                       0.311493        -0.218750
3395297                                       0.000000         0.005058

         delta_revenue_lag_1  month  year  days_of_month  \
3395293             1.211914      1     1             31
3395294             1.211914      1     1             31
3395295             1.211914      1     1             31
3395296             1.211914      1     1             31
3395297             1.211914      1     1             31

         months_since_item_shop_last_sale_lag_1  \
3395293                                     1.0
3395294                                     1.0
3395295                                    -1.0
3395296                                     1.0
3395297                                    -1.0

         months_since_shop_item_first_sale_lag_1  \
3395293                                     -1.0
3395294                                     -1.0
3395295                                     -1.0
3395296                                     11.0
3395297                                     -1.0

         months_since_item_last_sale_lag_1  months_since_item_first_sale_lag_1
3395293                                1.0                                 -1.0
3395294                                1.0                                 -1.0
3395295                                1.0                                 -1.0
3395296                                1.0                                 11.0
3395297                                1.0                                 -1.0
```

```
[5 rows x 83 columns]
```

[100]:
```
sales.describe().round(2)
```

[100]:
```
       cat_subtype_id  cat_type_id     city_id  date_block_num  \
count      5465978.00   5465978.00  5465978.00      5465978.00
mean            19.74        10.96       15.38           22.53
std             20.96         3.01        8.29            6.60
min              0.00         0.00        0.00           12.00
25%              2.00        11.00       10.00           17.00
50%             10.00        11.00       15.00           22.00
75%             35.00        13.00       22.00           28.00
max             64.00        19.00       30.00           34.00

       item_category_id  item_cnt_month     item_id  item_subtype_id  \
count        5465978.00      5465978.00  5465978.00       5465978.00
mean              44.86            0.29    11211.40           404.73
std               15.84            1.16     6268.25           503.61
min                0.00            0.00        1.00             0.00
25%               37.00            0.00     5647.00            42.00
50%               40.00            0.00    11319.00            42.00
75%               55.00            0.00    16461.00           637.00
max               83.00           20.00    22169.00          1665.00

       item_type_id     shop_id  shop_type_id  item_cnt_month_lag_1  \
count    5465978.00  5465978.00    5465978.00            5465978.00
mean          20.14       31.45          3.58                  0.30
std           35.02       17.83          1.80                  1.21
min            0.00        2.00          0.00                  0.00
25%            4.00       16.00          3.00                  0.00
50%            4.00       34.00          4.00                  0.00
75%            4.00       47.00          5.00                  0.00
max          174.00       59.00          5.00                 20.00

       item_cnt_month_lag_2  item_cnt_month_lag_3  item_cnt_month_lag_4  \
count            5465978.00            5465978.00            5465978.00
mean                   0.30                  0.30                  0.30
std                    1.22                  1.22                  1.24
min                    0.00                  0.00                  0.00
25%                    0.00                  0.00                  0.00
50%                    0.00                  0.00                  0.00
75%                    0.00                  0.00                  0.00
max                   20.00                 20.00                 20.00

       item_cnt_month_lag_5  item_cnt_month_lag_6  item_cnt_month_lag_7  \
count            5465978.00            5465978.00            5465978.00
```

```
mean                  0.30                 0.29                 0.29
std                   1.24                 1.24                 1.23
min                   0.00                 0.00                 0.00
25%                   0.00                 0.00                 0.00
50%                   0.00                 0.00                 0.00
75%                   0.00                 0.00                 0.00
max                  20.00                20.00                20.00

       item_cnt_month_lag_8  item_cnt_month_lag_9  item_cnt_month_lag_10  \
count            5465978.00            5465978.00             5465978.00
mean                   0.28                  0.27                   0.27
std                    1.22                  1.21                   1.21
min                    0.00                  0.00                   0.00
25%                    0.00                  0.00                   0.00
50%                    0.00                  0.00                   0.00
75%                    0.00                  0.00                   0.00
max                   20.00                 20.00                  20.00

       item_cnt_month_lag_11  item_cnt_month_lag_12  \
count             5465978.00             5465978.00
mean                    0.26                   0.25
std                     1.20                   1.17
min                     0.00                   0.00
25%                     0.00                   0.00
50%                     0.00                   0.00
75%                     0.00                   0.00
max                    20.00                  20.00

       shop_id_cnt_month_mean_Expanding_lag_1  \
count                              5465978.00
mean                                     0.25
std                                      0.23
min                                      0.00
25%                                      0.11
50%                                      0.22
75%                                      0.29
max                                      1.21

       shop_id_cnt_month_mean_Expanding_lag_2  …  \
count                              5465978.00  …
mean                                     0.24  …
std                                      0.23  …
min                                      0.00  …
25%                                      0.04  …
50%                                      0.21  …
75%                                      0.29  …
max                                      1.21  …
```

```
       item_type_id_cnt_month_mean_Expanding_lag_12  \
count                                    5465978.00
mean                                           0.17
std                                            0.34
min                                            0.00
25%                                            0.00
50%                                            0.00
75%                                            0.24
max                                           19.84

       item_subtype_id_cnt_month_mean_Expanding_lag_1  \
count                                      5465978.00
mean                                             0.28
std                                              0.42
min                                              0.00
25%                                              0.10
50%                                              0.28
75%                                              0.39
max                                             20.00

       item_subtype_id_cnt_month_mean_Expanding_lag_2  \
count                                      5465978.00
mean                                             0.27
std                                              0.43
min                                              0.00
25%                                              0.04
50%                                              0.21
75%                                              0.39
max                                             20.00

       item_subtype_id_cnt_month_mean_Expanding_lag_3  \
count                                      5465978.00
mean                                             0.26
std                                              0.43
min                                              0.00
25%                                              0.00
50%                                              0.20
75%                                              0.39
max                                             20.00

       item_subtype_id_cnt_month_mean_Expanding_lag_4  \
count                                      5465978.00
mean                                             0.25
std                                              0.43
min                                              0.00
25%                                              0.00
```

```
50%                                            0.20
75%                                            0.39
max                                           20.00

       item_subtype_id_cnt_month_mean_Expanding_lag_12  \
count                                       5465978.00
mean                                              0.17
std                                               0.42
min                                               0.00
25%                                               0.00
50%                                               0.00
75%                                               0.38
max                                              20.00

       cat_type_id_cnt_month_mean_Expanding_lag_1  \
count                                  5465978.00
mean                                         0.27
std                                          0.27
min                                          0.00
25%                                          0.13
50%                                          0.23
75%                                          0.30
max                                          6.39

       cat_type_id_cnt_month_mean_Expanding_lag_2  \
count                                  5465978.00
mean                                         0.26
std                                          0.27
min                                          0.00
25%                                          0.12
50%                                          0.23
75%                                          0.30
max                                          6.39

       cat_type_id_cnt_month_mean_Expanding_lag_3  \
count                                  5465978.00
mean                                         0.25
std                                          0.27
min                                          0.00
25%                                          0.00
50%                                          0.23
75%                                          0.29
max                                          6.39

       cat_type_id_cnt_month_mean_Expanding_lag_4  \
count                                  5465978.00
mean                                         0.24
```

```
std                                              0.27
min                                              0.00
25%                                              0.00
50%                                              0.18
75%                                              0.28
max                                              6.39

       cat_type_id_cnt_month_mean_Expanding_lag_12  \
count                                   5465978.00
mean                                          0.16
std                                           0.26
min                                           0.00
25%                                           0.00
50%                                           0.00
75%                                           0.24
max                                           8.00

       cat_subtype_id_cnt_month_mean_Expanding_lag_1  \
count                                     5465978.00
mean                                            0.29
std                                             0.44
min                                             0.00
25%                                             0.07
50%                                             0.22
75%                                             0.26
max                                            15.33

       cat_subtype_id_cnt_month_mean_Expanding_lag_2  \
count                                     5465978.00
mean                                            0.28
std                                             0.43
min                                             0.00
25%                                             0.05
50%                                             0.21
75%                                             0.26
max                                            15.33

       cat_subtype_id_cnt_month_mean_Expanding_lag_3  \
count                                     5465978.00
mean                                            0.27
std                                             0.43
min                                             0.00
25%                                             0.00
50%                                             0.21
75%                                             0.26
max                                            15.33
```

```
       cat_subtype_id_cnt_month_mean_Expanding_lag_4  \
count                                     5465978.00
mean                                            0.25
std                                             0.42
min                                             0.00
25%                                             0.00
50%                                             0.20
75%                                             0.26
max                                            15.33

       cat_subtype_id_cnt_month_mean_Expanding_lag_12  delta_price_lag  \
count                                      5465978.00       5465978.00
mean                                             0.17              NaN
std                                              0.36             0.00
min                                              0.00            -1.00
25%                                              0.00            -0.04
50%                                              0.00             0.00
75%                                              0.22             0.01
max                                             15.00             2.99

       delta_revenue_lag_1       month        year  days_of_month  \
count           5465978.00  5465978.00  5465978.00     5465978.00
mean                   NaN        6.18        1.45          30.39
std                   0.00        3.37        0.50           0.88
min                  -1.00        1.00        1.00          28.00
25%                  -0.18        3.00        1.00          30.00
50%                   0.00        6.00        1.00          31.00
75%                   0.07        9.00        2.00          31.00
max                   4.20       12.00        2.00          31.00

       months_since_item_shop_last_sale_lag_1  \
count                              5465978.00
mean                                     0.17
std                                      1.01
min                                     -1.00
25%                                     -1.00
50%                                      0.00
75%                                      1.00
max                                     31.00

       months_since_shop_item_first_sale_lag_1  \
count                               5465978.00
mean                                      0.65
std                                       4.58
min                                      -1.00
25%                                      -1.00
50%                                      -1.00
```

```
75%                                      0.00
max                                     33.00

       months_since_item_last_sale_lag_1  months_since_item_first_sale_lag_1
count                         5465978.00                          5465978.00
mean                               -0.75                                0.93
std                                 0.51                                5.12
min                                -1.00                               -1.00
25%                                -1.00                               -1.00
50%                                -1.00                               -1.00
75%                                -1.00                                0.00
max                                31.00                               33.00

[8 rows x 82 columns]
```

[101]:
```
sales = sales.drop(['shop_item_id'], axis=1)
```

[102]:
```
sales.to_pickle('data_5.pickle.gzde', compression='gzip')
```

[103]:
```
del sales
del group
del cache
```

1.2.17 Scaling data

The algorithms used to model are tree-based, therefore the data does not require normalisation.

[104]:
```
sales = pd.read_pickle('data_5.pickle.gzde', compression='gzip')
```

[105]:
```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()

feature_cols = list(sales)
feature_cols = [e for e in feature_cols if e not in␣
 ↪('date_block_num','item_cnt_month')]

train = sales[sales['date_block_num']!= sales['date_block_num'].max()]
test = sales[sales['date_block_num']== sales['date_block_num'].max()]
```

[106]:
```
train[feature_cols] = mms.fit_transform(train[feature_cols])
test[feature_cols] = mms.transform(test[feature_cols])
```

[107]:
```
sales = pd.concat([train, test], axis = 0)
sales = downcast_dtypes(sales)
```

[108]:
```
del train, test, feature_cols
gc.collect()
```

[108]: 20

1.2.18 Feature selection

Mulitple methods will be used to find the most important features in deriving a model to predict sales. The most consistently important features will be selected for the purposes of building the model.

[109]:
```
feat_sel = sales[sales.date_block_num < 33]
X = feat_sel.drop(['item_cnt_month','date_block_num'], axis=1)
Y = feat_sel['item_cnt_month']
feature_name = list(X.columns)
print(len(feature_name))
print(feature_name)
```

```
80
['cat_subtype_id', 'cat_type_id', 'city_id', 'item_category_id', 'item_id',
'item_subtype_id', 'item_type_id', 'shop_id', 'shop_type_id',
'item_cnt_month_lag_1', 'item_cnt_month_lag_2', 'item_cnt_month_lag_3',
'item_cnt_month_lag_4', 'item_cnt_month_lag_5', 'item_cnt_month_lag_6',
'item_cnt_month_lag_7', 'item_cnt_month_lag_8', 'item_cnt_month_lag_9',
'item_cnt_month_lag_10', 'item_cnt_month_lag_11', 'item_cnt_month_lag_12',
'shop_id_cnt_month_mean_Expanding_lag_1',
'shop_id_cnt_month_mean_Expanding_lag_2',
'shop_id_cnt_month_mean_Expanding_lag_3',
'shop_id_cnt_month_mean_Expanding_lag_4',
'shop_id_cnt_month_mean_Expanding_lag_12',
'item_id_cnt_month_mean_Expanding_lag_1',
'item_id_cnt_month_mean_Expanding_lag_2',
'item_id_cnt_month_mean_Expanding_lag_3',
'item_id_cnt_month_mean_Expanding_lag_4',
'item_id_cnt_month_mean_Expanding_lag_12',
'shop_item_id_cnt_month_mean_Smooth_lag_1',
'shop_item_id_cnt_month_mean_Smooth_lag_2',
'shop_item_id_cnt_month_mean_Smooth_lag_3',
'shop_item_id_cnt_month_mean_Smooth_lag_4',
'shop_item_id_cnt_month_mean_Smooth_lag_12',
'shop_type_id_cnt_month_mean_Expanding_lag_1',
'shop_type_id_cnt_month_mean_Expanding_lag_2',
'shop_type_id_cnt_month_mean_Expanding_lag_3',
'shop_type_id_cnt_month_mean_Expanding_lag_4',
'shop_type_id_cnt_month_mean_Expanding_lag_12',
'city_id_cnt_month_mean_Expanding_lag_1',
'city_id_cnt_month_mean_Expanding_lag_2',
'city_id_cnt_month_mean_Expanding_lag_3',
'city_id_cnt_month_mean_Expanding_lag_4',
'city_id_cnt_month_mean_Expanding_lag_12',
'item_category_id_cnt_month_mean_Expanding_lag_1',
'item_category_id_cnt_month_mean_Expanding_lag_2',
'item_category_id_cnt_month_mean_Expanding_lag_3',
'item_category_id_cnt_month_mean_Expanding_lag_4',
'item_category_id_cnt_month_mean_Expanding_lag_12',
'item_type_id_cnt_month_mean_Expanding_lag_1',
'item_type_id_cnt_month_mean_Expanding_lag_2',
'item_type_id_cnt_month_mean_Expanding_lag_3',
'item_type_id_cnt_month_mean_Expanding_lag_4',
'item_type_id_cnt_month_mean_Expanding_lag_12',
'item_subtype_id_cnt_month_mean_Expanding_lag_1',
'item_subtype_id_cnt_month_mean_Expanding_lag_2',
'item_subtype_id_cnt_month_mean_Expanding_lag_3',
'item_subtype_id_cnt_month_mean_Expanding_lag_4',
'item_subtype_id_cnt_month_mean_Expanding_lag_12',
'cat_type_id_cnt_month_mean_Expanding_lag_1',
'cat_type_id_cnt_month_mean_Expanding_lag_2',
'cat_type_id_cnt_month_mean_Expanding_lag_3',
'cat_type_id_cnt_month_mean_Expanding_lag_4',
'cat_type_id_cnt_month_mean_Expanding_lag_12',
'cat_subtype_id_cnt_month_mean_Expanding_lag_1',
'cat_subtype_id_cnt_month_mean_Expanding_lag_2',
'cat_subtype_id_cnt_month_mean_Expanding_lag_3',
'cat_subtype_id_cnt_month_mean_Expanding_lag_4',
'cat_subtype_id_cnt_month_mean_Expanding_lag_12', 'delta_price_lag',
'delta_revenue_lag_1', 'month', 'year', 'days_of_month',
'months_since_item_shop_last_sale_lag_1',
'months_since_shop_item_first_sale_lag_1', 'months_since_item_last_sale_lag_1',
'months_since_item_first_sale_lag_1']
```

[110]:
```
num_feats = 30
```

[111]:
```
#pearson correlation
def cor_selector(X, y,num_feats):
    cor_list = []
    feature_name = X.columns.tolist()
    # calculate the correlation with y for each feature
    for i in X.columns.tolist():
        cor = np.corrcoef(X[i], y)[0, 1]
        cor_list.append(cor)
    # replace NaN with 0
    cor_list = [0 if np.isnan(i) else i for i in cor_list]
    # feature name
    cor_feature = X.iloc[:,np.argsort(np.abs(cor_list))[-num_feats:]].columns.
 ↪tolist()
    # feature selection? 0 for not select, 1 for select
    cor_support = [True if i in cor_feature else False for i in feature_name]
    return cor_support, cor_feature
```

```
cor_support, cor_feature = cor_selector(X, Y, num_feats)
print(str(len(cor_feature)), 'selected features')
```

```
30 selected features
```

[112]:
```
cor_feature
```

[112]:
```
['item_subtype_id_cnt_month_mean_Expanding_lag_2',
 'shop_item_id_cnt_month_mean_Smooth_lag_12',
 'item_cnt_month_lag_12',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_3',
 'item_category_id_cnt_month_mean_Expanding_lag_3',
 'months_since_shop_item_first_sale_lag_1',
 'item_cnt_month_lag_11',
 'item_subtype_id_cnt_month_mean_Expanding_lag_1',
 'item_cnt_month_lag_10',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_2',
 'item_category_id_cnt_month_mean_Expanding_lag_2',
 'item_cnt_month_lag_9',
 'item_id_cnt_month_mean_Expanding_lag_4',
 'item_cnt_month_lag_8',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_1',
 'item_category_id_cnt_month_mean_Expanding_lag_1',
 'item_cnt_month_lag_7',
 'item_id_cnt_month_mean_Expanding_lag_3',
 'item_cnt_month_lag_6',
 'item_cnt_month_lag_5',
 'shop_item_id_cnt_month_mean_Smooth_lag_4',
 'item_id_cnt_month_mean_Expanding_lag_2',
 'item_cnt_month_lag_4',
 'shop_item_id_cnt_month_mean_Smooth_lag_3',
 'shop_item_id_cnt_month_mean_Smooth_lag_2',
 'item_cnt_month_lag_3',
 'item_id_cnt_month_mean_Expanding_lag_1',
 'shop_item_id_cnt_month_mean_Smooth_lag_1',
 'item_cnt_month_lag_2',
 'item_cnt_month_lag_1']
```

[113]:
```
#chi-squared
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
chi_selector = SelectKBest(chi2, k=num_feats)
chi_selector.fit(X, Y)
chi_support = chi_selector.get_support()
chi_feature = X.loc[:,chi_support].columns.tolist()
print(str(len(chi_feature)), 'selected features')
```

```
30 selected features
```

[114]:
```
chi_feature
```

[114]:
```
['item_cnt_month_lag_1',
 'item_cnt_month_lag_2',
 'item_cnt_month_lag_3',
 'item_cnt_month_lag_4',
 'item_cnt_month_lag_5',
 'item_cnt_month_lag_6',
 'item_cnt_month_lag_7',
 'item_cnt_month_lag_8',
 'item_cnt_month_lag_9',
 'item_cnt_month_lag_10',
 'item_cnt_month_lag_11',
 'item_cnt_month_lag_12',
 'shop_id_cnt_month_mean_Expanding_lag_1',
 'shop_id_cnt_month_mean_Expanding_lag_2',
 'shop_id_cnt_month_mean_Expanding_lag_3',
 'shop_id_cnt_month_mean_Expanding_lag_4',
 'item_id_cnt_month_mean_Expanding_lag_1',
 'item_id_cnt_month_mean_Expanding_lag_2',
 'item_id_cnt_month_mean_Expanding_lag_3',
 'item_id_cnt_month_mean_Expanding_lag_4',
 'item_id_cnt_month_mean_Expanding_lag_12',
 'shop_item_id_cnt_month_mean_Smooth_lag_1',
 'shop_item_id_cnt_month_mean_Smooth_lag_2',
 'shop_item_id_cnt_month_mean_Smooth_lag_3',
 'shop_item_id_cnt_month_mean_Smooth_lag_4',
 'city_id_cnt_month_mean_Expanding_lag_1',
 'city_id_cnt_month_mean_Expanding_lag_2',
 'city_id_cnt_month_mean_Expanding_lag_3',
 'months_since_shop_item_first_sale_lag_1',
 'months_since_item_first_sale_lag_1']
```

[115]:
```
#recursive feature elimination with linear regression
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

rfe_selector = RFE(estimator=LinearRegression(),␣
 ↪n_features_to_select=num_feats, step=10, verbose=5)
rfe_selector.fit(X, Y)
rfe_support = rfe_selector.get_support()
rfe_feature = X.loc[:,rfe_support].columns.tolist()
print(str(len(rfe_feature)), 'selected features')
```

```
Fitting estimator with 80 features.
Fitting estimator with 70 features.
Fitting estimator with 60 features.
Fitting estimator with 50 features.
Fitting estimator with 40 features.
30 selected features
```

[116]:
```
rfe_feature
```

[116]:
```
['item_cnt_month_lag_1',
 'item_cnt_month_lag_2',
 'item_cnt_month_lag_3',
 'item_id_cnt_month_mean_Expanding_lag_1',
 'item_id_cnt_month_mean_Expanding_lag_2',
 'shop_item_id_cnt_month_mean_Smooth_lag_1',
 'shop_item_id_cnt_month_mean_Smooth_lag_2',
 'shop_item_id_cnt_month_mean_Smooth_lag_3',
 'shop_item_id_cnt_month_mean_Smooth_lag_4',
 'shop_item_id_cnt_month_mean_Smooth_lag_12',
 'shop_type_id_cnt_month_mean_Expanding_lag_1',
 'shop_type_id_cnt_month_mean_Expanding_lag_12',
 'item_category_id_cnt_month_mean_Expanding_lag_1',
 'item_category_id_cnt_month_mean_Expanding_lag_2',
 'item_category_id_cnt_month_mean_Expanding_lag_3',
 'item_category_id_cnt_month_mean_Expanding_lag_4',
 'item_category_id_cnt_month_mean_Expanding_lag_12',
 'item_type_id_cnt_month_mean_Expanding_lag_1',
 'item_type_id_cnt_month_mean_Expanding_lag_2',
 'item_subtype_id_cnt_month_mean_Expanding_lag_1',
 'item_subtype_id_cnt_month_mean_Expanding_lag_2',
 'item_subtype_id_cnt_month_mean_Expanding_lag_3',
 'item_subtype_id_cnt_month_mean_Expanding_lag_4',
 'cat_type_id_cnt_month_mean_Expanding_lag_1',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_1',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_2',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_3',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_4',
 'cat_subtype_id_cnt_month_mean_Expanding_lag_12',
 'months_since_item_last_sale_lag_1']
```

[117]:
```
#recursive feature elimination with random forest
#the SelectFromModel method was attempted, but running time was too long
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators = 100,
                           n_jobs = -1,
                           oob_score = True,
                           bootstrap = True,
                           random_state = 42)
rf.fit(X, Y)
```

```
r_lst = []
for feature in sorted(zip(rf.feature_importances_, X.columns), reverse=True):
    r_lst.append(feature)
r_lst = r_lst[:30]
features_rf = []
for i in r_lst:
    features_rf.append(i[1])
```

[118]:
```
# Feature importance with light GBM
#lightGBM model fit
gbm = lgb.LGBMRegressor()
gbm.fit(X, Y)
gbm.booster_.feature_importance()

g_lst = []
for feature in sorted(zip(gbm.feature_importances_, X.columns), reverse=True):
    g_lst.append(feature)
g_lst = g_lst[:30]
features_gbm = []
for i in g_lst:
    features_gbm.append(i[1])
```

[119]:
```
xgbm = xgb.XGBRegressor()
xgbm.fit(X, Y)

x_lst = []
for feature in sorted(zip(xgbm.feature_importances_, X.columns), reverse=True):
    x_lst.append(feature)
x_lst
```

```
[21:48:58] WARNING: src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
[21:48:58] WARNING: src/learner.cc:686: Tree method is automatically selected to
be 'approx' for faster speed. To use old behavior (exact greedy algorithm on
single machine), set tree_method to 'exact'.
```

[119]:
```
[(0.29189757, 'item_cnt_month_lag_1'),
 (0.0972129, 'shop_item_id_cnt_month_mean_Smooth_lag_1'),
 (0.07092963, 'cat_subtype_id_cnt_month_mean_Expanding_lag_4'),
 (0.05207234, 'months_since_item_shop_last_sale_lag_1'),
 (0.03970951, 'shop_type_id_cnt_month_mean_Expanding_lag_4'),
 (0.032093115, 'item_cnt_month_lag_2'),
 (0.030606989, 'months_since_item_first_sale_lag_1'),
 (0.024894359, 'shop_type_id_cnt_month_mean_Expanding_lag_3'),
 (0.02302277, 'shop_id_cnt_month_mean_Expanding_lag_12'),
 (0.019293122, 'item_cnt_month_lag_3'),
 (0.01909757, 'item_category_id'),
 (0.018602934, 'item_cnt_month_lag_8'),
 (0.017617425, 'months_since_shop_item_first_sale_lag_1'),
 (0.01657576, 'item_type_id_cnt_month_mean_Expanding_lag_2'),
 (0.015775928, 'shop_item_id_cnt_month_mean_Smooth_lag_2'),
 (0.015747877, 'shop_item_id_cnt_month_mean_Smooth_lag_3'),
 (0.015213964, 'shop_item_id_cnt_month_mean_Smooth_lag_4'),
 (0.012944539, 'month'),
 (0.011845271, 'item_subtype_id_cnt_month_mean_Expanding_lag_4'),
 (0.011731102, 'shop_id_cnt_month_mean_Expanding_lag_4'),
 (0.009351417, 'item_cnt_month_lag_5'),
 (0.008769338, 'shop_item_id_cnt_month_mean_Smooth_lag_12'),
 (0.007908101, 'cat_type_id_cnt_month_mean_Expanding_lag_3'),
 (0.007446583, 'cat_type_id_cnt_month_mean_Expanding_lag_4'),
 (0.007249106, 'cat_type_id_cnt_month_mean_Expanding_lag_12'),
 (0.006701999, 'item_cnt_month_lag_6'),
 (0.0065668696, 'item_id_cnt_month_mean_Expanding_lag_2'),
 (0.005944074, 'item_id_cnt_month_mean_Expanding_lag_4'),
 (0.0058902176, 'item_id_cnt_month_mean_Expanding_lag_12'),
 (0.0057972954, 'city_id'),
 (0.0056893537, 'item_type_id_cnt_month_mean_Expanding_lag_12'),
 (0.00542173, 'shop_type_id_cnt_month_mean_Expanding_lag_2'),
 (0.0049782908, 'cat_subtype_id_cnt_month_mean_Expanding_lag_3'),
 (0.00486983, 'item_cnt_month_lag_10'),
 (0.0046798512, 'item_type_id_cnt_month_mean_Expanding_lag_3'),
 (0.0046731657, 'item_type_id'),
 (0.0046636546, 'cat_subtype_id'),
 (0.0046461974, 'cat_type_id_cnt_month_mean_Expanding_lag_1'),
 (0.004508725, 'item_cnt_month_lag_7'),
 (0.004317061, 'cat_type_id_cnt_month_mean_Expanding_lag_2'),
 (0.0042478885, 'item_cnt_month_lag_4'),
 (0.004143697, 'item_category_id_cnt_month_mean_Expanding_lag_1'),
 (0.003965197, 'cat_type_id'),
 (0.0037120741, 'item_id_cnt_month_mean_Expanding_lag_1'),
 (0.0031850059, 'delta_price_lag'),
 (0.0028749283, 'item_type_id_cnt_month_mean_Expanding_lag_1'),
 (0.0028671175, 'item_subtype_id_cnt_month_mean_Expanding_lag_3'),
 (0.0025235137, 'cat_subtype_id_cnt_month_mean_Expanding_lag_12'),
 (0.0020568229, 'shop_id_cnt_month_mean_Expanding_lag_1'),
 (0.0020163655, 'item_cnt_month_lag_9'),
 (0.0020153865, 'item_cnt_month_lag_11'),
 (0.0019541846, 'year'),
 (0.0019099995, 'item_id_cnt_month_mean_Expanding_lag_3'),
 (0.0015732666, 'days_of_month'),
 (0.0014899498, 'item_subtype_id'),
 (0.0013850734, 'item_cnt_month_lag_12'),
 (0.0011220238, 'item_id'),
 (0.0, 'shop_type_id_cnt_month_mean_Expanding_lag_12'),
 (0.0, 'shop_type_id_cnt_month_mean_Expanding_lag_1'),
 (0.0, 'shop_type_id'),
 (0.0, 'shop_id_cnt_month_mean_Expanding_lag_3'),
 (0.0, 'shop_id_cnt_month_mean_Expanding_lag_2'),
 (0.0, 'shop_id'),
 (0.0, 'months_since_item_last_sale_lag_1'),
 (0.0, 'item_type_id_cnt_month_mean_Expanding_lag_4'),
 (0.0, 'item_subtype_id_cnt_month_mean_Expanding_lag_2'),
 (0.0, 'item_subtype_id_cnt_month_mean_Expanding_lag_12'),
 (0.0, 'item_subtype_id_cnt_month_mean_Expanding_lag_1'),
 (0.0, 'item_category_id_cnt_month_mean_Expanding_lag_4'),
 (0.0, 'item_category_id_cnt_month_mean_Expanding_lag_3'),
 (0.0, 'item_category_id_cnt_month_mean_Expanding_lag_2'),
 (0.0, 'item_category_id_cnt_month_mean_Expanding_lag_12'),
 (0.0, 'delta_revenue_lag_1'),
 (0.0, 'city_id_cnt_month_mean_Expanding_lag_4'),
 (0.0, 'city_id_cnt_month_mean_Expanding_lag_3'),
 (0.0, 'city_id_cnt_month_mean_Expanding_lag_2'),
 (0.0, 'city_id_cnt_month_mean_Expanding_lag_12'),
 (0.0, 'city_id_cnt_month_mean_Expanding_lag_1'),
 (0.0, 'cat_subtype_id_cnt_month_mean_Expanding_lag_2'),
 (0.0, 'cat_subtype_id_cnt_month_mean_Expanding_lag_1')]
```

```
[120]: x_lst = x_lst[:30]
features_x = []
for i in x_lst:
    features_x.append(i[1])
```

```
[121]: pd.set_option('display.max_rows', None)
# put all selection together
feature_selection_df = pd.DataFrame({'Feature':feature_name, 'Pearson':
 ↪cor_support, 'Chi-2':chi_support, 'RFE':rfe_support})
```

```
[122]: rf_df = pd.DataFrame({'Feature': features_rf, 'RF': True})
gbm_df = pd.DataFrame({'Feature': features_gbm, 'LGBM': True})
xgb_df = pd.DataFrame({'Feature': features_x, 'XGB': True})
```

```
[123]: feature_selection_df = pd.merge(feature_selection_df, rf_df, how='left',␣
 ↪on='Feature')
feature_selection_df = pd.merge(feature_selection_df, gbm_df, how='left',␣
 ↪on='Feature')
feature_selection_df = pd.merge(feature_selection_df, xgb_df, how='left',␣
 ↪on='Feature')
feature_selection_df  = feature_selection_df.fillna(False)
```

[124]:
```
# count the selected times for each feature
feature_selection_df['Total'] = np.sum(feature_selection_df, axis=1)
# display the top
feature_selection_df = feature_selection_df.sort_values(['Total','Feature'] ,␣
 ↪ascending=False)
feature_selection_df.index = range(1, len(feature_selection_df)+1)
feature_selection_df
```

[124]:
```
                                           Feature  Pearson  Chi-2    RFE  \
1          shop_item_id_cnt_month_mean_Smooth_lag_2     True   True   True
2          shop_item_id_cnt_month_mean_Smooth_lag_1     True   True   True
3                              item_cnt_month_lag_2     True   True   True
4                              item_cnt_month_lag_1     True   True   True
5          shop_item_id_cnt_month_mean_Smooth_lag_4     True   True   True
6            months_since_shop_item_first_sale_lag_1     True   True  False
7             item_id_cnt_month_mean_Expanding_lag_4     True   True  False
8             item_id_cnt_month_mean_Expanding_lag_2     True   True   True
9             item_id_cnt_month_mean_Expanding_lag_1     True   True   True
10                             item_cnt_month_lag_3     True   True   True
11         shop_item_id_cnt_month_mean_Smooth_lag_3     True   True   True
12        shop_item_id_cnt_month_mean_Smooth_lag_12     True  False   True
13       item_type_id_cnt_month_mean_Expanding_lag_2    False  False   True
14    item_subtype_id_cnt_month_mean_Expanding_lag_2     True  False   True
15    item_subtype_id_cnt_month_mean_Expanding_lag_1     True  False   True
16                             item_cnt_month_lag_6     True   True  False
17           months_since_item_shop_last_sale_lag_1    False  False  False
18               months_since_item_first_sale_lag_1    False   True  False
19                                            month    False  False  False
20       item_type_id_cnt_month_mean_Expanding_lag_1    False  False   True
21           item_id_cnt_month_mean_Expanding_lag_12    False   True  False
22                             item_cnt_month_lag_8     True   True  False
23                             item_cnt_month_lag_5     True   True  False
24   item_category_id_cnt_month_mean_Expanding_lag_1     True  False   True
25                                 item_category_id    False  False  False
26        cat_type_id_cnt_month_mean_Expanding_lag_1    False  False   True
27       shop_type_id_cnt_month_mean_Expanding_lag_1    False  False   True
28            shop_id_cnt_month_mean_Expanding_lag_4    False   True  False
29                                     item_type_id    False  False  False
30    item_subtype_id_cnt_month_mean_Expanding_lag_4    False  False   True
31                                  item_subtype_id    False  False  False
32            item_id_cnt_month_mean_Expanding_lag_3     True   True  False
33                                          item_id    False  False  False
34                             item_cnt_month_lag_9     True   True  False
35                             item_cnt_month_lag_7     True   True  False
36                             item_cnt_month_lag_4     True   True  False
37                            item_cnt_month_lag_12     True   True  False
38                            item_cnt_month_lag_11     True   True  False
```

```
39                             item_cnt_month_lag_10     True   True  False
40   item_category_id_cnt_month_mean_Expanding_lag_3     True  False   True
41   item_category_id_cnt_month_mean_Expanding_lag_2     True  False   True
42                                   delta_price_lag    False  False  False
43                                     days_of_month    False  False  False
44           city_id_cnt_month_mean_Expanding_lag_2    False   True  False
45           city_id_cnt_month_mean_Expanding_lag_1    False   True  False
46                                           city_id    False  False  False
47     cat_subtype_id_cnt_month_mean_Expanding_lag_4    False  False   True
48     cat_subtype_id_cnt_month_mean_Expanding_lag_3     True  False   True
49     cat_subtype_id_cnt_month_mean_Expanding_lag_2     True  False   True
50     cat_subtype_id_cnt_month_mean_Expanding_lag_1     True  False   True
51                                    cat_subtype_id    False  False  False
52                                              year    False  False  False
53       shop_type_id_cnt_month_mean_Expanding_lag_4    False  False  False
54       shop_type_id_cnt_month_mean_Expanding_lag_3    False  False  False
55      shop_type_id_cnt_month_mean_Expanding_lag_12    False  False   True
56                                      shop_type_id    False  False  False
57           shop_id_cnt_month_mean_Expanding_lag_3    False   True  False
58           shop_id_cnt_month_mean_Expanding_lag_2    False   True  False
59          shop_id_cnt_month_mean_Expanding_lag_12    False  False  False
60           shop_id_cnt_month_mean_Expanding_lag_1    False   True  False
61                                           shop_id    False  False  False
62                 months_since_item_last_sale_lag_1    False  False   True
63    item_subtype_id_cnt_month_mean_Expanding_lag_3    False  False   True
64   item_category_id_cnt_month_mean_Expanding_lag_4    False  False   True
65  item_category_id_cnt_month_mean_Expanding_lag_12    False  False   True
66                               delta_revenue_lag_1    False  False  False
67           city_id_cnt_month_mean_Expanding_lag_3    False   True  False
68        cat_type_id_cnt_month_mean_Expanding_lag_4    False  False  False
69        cat_type_id_cnt_month_mean_Expanding_lag_3    False  False  False
70        cat_type_id_cnt_month_mean_Expanding_lag_2    False  False  False
71       cat_type_id_cnt_month_mean_Expanding_lag_12    False  False  False
72                                       cat_type_id    False  False  False
73    cat_subtype_id_cnt_month_mean_Expanding_lag_12    False  False   True
74       shop_type_id_cnt_month_mean_Expanding_lag_2    False  False  False
75       item_type_id_cnt_month_mean_Expanding_lag_4    False  False  False
76       item_type_id_cnt_month_mean_Expanding_lag_3    False  False  False
77      item_type_id_cnt_month_mean_Expanding_lag_12    False  False  False
78   item_subtype_id_cnt_month_mean_Expanding_lag_12    False  False  False
79           city_id_cnt_month_mean_Expanding_lag_4    False  False  False
80          city_id_cnt_month_mean_Expanding_lag_12    False  False  False

       RF   LGBM    XGB  Total
1    True   True   True      6
2    True   True   True      6
3    True   True   True      6
```

```
4    True   True   True      6
5   False   True   True      5
6    True   True   True      5
7    True   True   True      5
8    True  False   True      5
9    True   True  False      5
10  False   True   True      5
11  False  False   True      4
12  False   True   True      4
13   True   True   True      4
14   True   True  False      4
15   True   True  False      4
16  False   True   True      4
17   True   True   True      3
18   True  False   True      3
19   True   True   True      3
20   True   True  False      3
21  False   True   True      3
22  False  False   True      3
23  False  False   True      3
24  False   True  False      3
25   True   True   True      3
26   True   True  False      3
27   True  False  False      2
28  False  False   True      2
29   True   True  False      2
30  False  False   True      2
31   True   True  False      2
32  False  False  False      2
33   True   True  False      2
34  False  False  False      2
35  False  False  False      2
36  False  False  False      2
37  False  False  False      2
38  False  False  False      2
39  False  False  False      2
40  False  False  False      2
41  False  False  False      2
42   True   True  False      2
43   True   True  False      2
44   True  False  False      2
45   True  False  False      2
46   True  False   True      2
47  False  False   True      2
48  False  False  False      2
49  False  False  False      2
50  False  False  False      2
```

```
51   True   True  False      2
52  False   True  False      1
53  False  False   True      1
54  False  False   True      1
55  False  False  False      1
56   True  False  False      1
57  False  False  False      1
58  False  False  False      1
59  False  False   True      1
60  False  False  False      1
61   True  False  False      1
62  False  False  False      1
63  False  False  False      1
64  False  False  False      1
65  False  False  False      1
66   True  False  False      1
67  False  False  False      1
68  False  False   True      1
69  False  False   True      1
70  False   True  False      1
71  False  False   True      1
72  False   True  False      1
73  False  False  False      1
74  False  False  False      0
75  False  False  False      0
76  False  False  False      0
77  False  False  False      0
78  False  False  False      0
79  False  False  False      0
80  False  False  False      0
```

[128]:
```
#select from sales df features that will be used for modelling and drop␣
↪unwanted ones
unwanted = feature_selection_df.Feature[74:]
```

[129]:
```
to_drop = []
for i in unwanted:
    to_drop.append(i)
```

[130]:
```
sales_df = sales.drop(to_drop, axis=1)
```

[131]:
```
sales_df.head()
```

[131]:
```
         cat_subtype_id  cat_type_id  city_id  date_block_num  \
3395293        0.062500     0.555556      0.0              12
3395294        0.015625     0.555556      0.0              12
3395295        0.062500     0.555556      0.0              12
```

```
3395296       0.015625     0.555556      0.0              12
3395297       0.062500     0.555556      0.0              12

         item_category_id  item_cnt_month   item_id  item_subtype_id  \
3395293          0.469136             0.0  0.001308         0.025225
3395294          0.432099             0.0  0.001353         0.337538
3395295          0.469136             1.0  0.001398         0.025225
3395296          0.432099             1.0  0.001444         0.337538
3395297          0.469136             0.0  0.001489         0.821021

         item_type_id  shop_id  shop_type_id  item_cnt_month_lag_1  \
3395293      0.022989      0.0           1.0                  0.00
3395294      0.022989      0.0           1.0                  0.00
3395295      0.022989      0.0           1.0                  0.00
3395296      0.022989      0.0           1.0                  0.05
3395297      0.022989      0.0           1.0                  0.00

         item_cnt_month_lag_2  item_cnt_month_lag_3  item_cnt_month_lag_4  \
3395293                   0.0                   0.0                   0.0
3395294                   0.0                   0.0                   0.0
3395295                   0.0                   0.0                   0.0
3395296                   0.1                   0.0                   0.0
3395297                   0.0                   0.0                   0.0

         item_cnt_month_lag_5  item_cnt_month_lag_6  item_cnt_month_lag_7  \
3395293                   0.0                   0.0                  0.05
3395294                   0.0                   0.0                  0.00
3395295                   0.0                   0.0                  0.00
3395296                   0.0                   0.0                  0.00
3395297                   0.0                   0.0                  0.00

         item_cnt_month_lag_8  item_cnt_month_lag_9  item_cnt_month_lag_10  \
3395293                   0.0                  0.00                   0.05
3395294                   0.0                  0.05                   0.05
3395295                   0.0                  0.00                   0.00
3395296                   0.0                  0.00                   0.00
3395297                   0.0                  0.00                   0.00

         item_cnt_month_lag_11  item_cnt_month_lag_12  \
3395293                    0.0                   0.00
3395294                    0.2                   0.00
3395295                    0.0                   0.00
3395296                    0.0                   0.05
3395297                    0.0                   0.00

         shop_id_cnt_month_mean_Expanding_lag_1  \
3395293                                0.081879
```

```
3395294                               0.081878
3395295                               0.081877
3395296                               0.081877
3395297                               0.081885

         shop_id_cnt_month_mean_Expanding_lag_2  …  \
3395293                               0.081114  …
3395294                               0.081113  …
3395295                               0.081112  …
3395296                               0.081111  …
3395297                               0.081131  …

         item_type_id_cnt_month_mean_Expanding_lag_1  \
3395293                                    0.012491
3395294                                    0.012491
3395295                                    0.012491
3395296                                    0.012491
3395297                                    0.012491

         item_type_id_cnt_month_mean_Expanding_lag_2  \
3395293                                    0.012046
3395294                                    0.012046
3395295                                    0.012046
3395296                                    0.012046
3395297                                    0.012046

         item_subtype_id_cnt_month_mean_Expanding_lag_1  \
3395293                                       0.020679
3395294                                       0.010859
3395295                                       0.020679
3395296                                       0.010859
3395297                                       0.010343

         item_subtype_id_cnt_month_mean_Expanding_lag_2  \
3395293                                       0.019118
3395294                                       0.010219
3395295                                       0.019118
3395296                                       0.010219
3395297                                       0.009702

         item_subtype_id_cnt_month_mean_Expanding_lag_3  \
3395293                                       0.019113
3395294                                       0.010387
3395295                                       0.019113
3395296                                       0.010387
3395297                                       0.009787
```

```
         item_subtype_id_cnt_month_mean_Expanding_lag_4  \
3395293                                        0.018981
3395294                                        0.010396
3395295                                        0.018981
3395296                                        0.010396
3395297                                        0.009994

         cat_type_id_cnt_month_mean_Expanding_lag_1  \
3395293                                    0.037261
3395294                                    0.037261
3395295                                    0.037261
3395296                                    0.037261
3395297                                    0.037261

         cat_type_id_cnt_month_mean_Expanding_lag_2  \
3395293                                    0.037322
3395294                                    0.037322
3395295                                    0.037322
3395296                                    0.037322
3395297                                    0.037322

         cat_type_id_cnt_month_mean_Expanding_lag_3  \
3395293                                     0.03756
3395294                                     0.03756
3395295                                     0.03756
3395296                                     0.03756
3395297                                     0.03756

         cat_type_id_cnt_month_mean_Expanding_lag_4  \
3395293                                    0.037923
3395294                                    0.037923
3395295                                    0.037923
3395296                                    0.037923
3395297                                    0.037922

         cat_type_id_cnt_month_mean_Expanding_lag_12  \
3395293                                     0.000000
3395294                                     0.000000
3395295                                     0.000000
3395296                                     0.000000
3395297                                     0.041667

         cat_subtype_id_cnt_month_mean_Expanding_lag_1  \
3395293                                       0.016609
3395294                                       0.013401
3395295                                       0.016609
3395296                                       0.013401
```

```
3395297                                      0.016609

         cat_subtype_id_cnt_month_mean_Expanding_lag_2  \
3395293                                      0.016573
3395294                                      0.013517
3395295                                      0.016573
3395296                                      0.013517
3395297                                      0.016573

         cat_subtype_id_cnt_month_mean_Expanding_lag_3  \
3395293                                      0.016583
3395294                                      0.013746
3395295                                      0.016583
3395296                                      0.013746
3395297                                      0.016583

         cat_subtype_id_cnt_month_mean_Expanding_lag_4  \
3395293                                      0.016722
3395294                                      0.013844
3395295                                      0.016722
3395296                                      0.013844
3395297                                      0.016722

         cat_subtype_id_cnt_month_mean_Expanding_lag_12  delta_price_lag  \
3395293                                        0.000000         0.130395
3395294                                        0.000000         0.218334
3395295                                        0.000000         0.148123
3395296                                        0.020995         0.195501
3395297                                        0.000000         0.251542

         delta_revenue_lag_1  month  year  days_of_month  \
3395293             0.425059    0.0   0.0            1.0
3395294             0.425059    0.0   0.0            1.0
3395295             0.425059    0.0   0.0            1.0
3395296             0.425059    0.0   0.0            1.0
3395297             0.425059    0.0   0.0            1.0

         months_since_item_shop_last_sale_lag_1  \
3395293                                  0.0625
3395294                                  0.0625
3395295                                  0.0000
3395296                                  0.0625
3395297                                  0.0000

         months_since_shop_item_first_sale_lag_1  \
3395293                                 0.000000
3395294                                 0.000000
```

```
3395295                                        0.000000
3395296                                        0.363636
3395297                                        0.000000

         months_since_item_last_sale_lag_1  months_since_item_first_sale_lag_1
3395293                             0.0625                            0.000000
3395294                             0.0625                            0.000000
3395295                             0.0625                            0.000000
3395296                             0.0625                            0.363636
3395297                             0.0625                            0.000000

[5 rows x 76 columns]
```

[132]:
```
#save final dataframe
sales_df.to_pickle('data_6.pickle.gzde', compression='gzip')
```

[133]:
```
del sales_df
del sales
```

1.3 Modelling

[3]:
```
sales = pd.read_pickle('data_6.pickle.gzde', compression='gzip')
```

[18]:
```
#first-level model
#save date_block_num, as cannot use as feature, but need it to split dataset␣
 ↪into parts
feature_cols = list(sales)
feature_cols = [e for e in feature_cols if e not in␣
 ↪('date_block_num','item_cnt_month')]
num_first_level_models = 5

dates = sales['date_block_num']
last_block = dates.max()
print('Test `date_block_num` is %d' % last_block)
print(feature_cols)
```

```
Test `date_block_num` is 34
['cat_subtype_id', 'cat_type_id', 'city_id', 'item_category_id', 'item_id',
'item_subtype_id', 'item_type_id', 'shop_id', 'shop_type_id',
'item_cnt_month_lag_1', 'item_cnt_month_lag_2', 'item_cnt_month_lag_3',
'item_cnt_month_lag_4', 'item_cnt_month_lag_5', 'item_cnt_month_lag_6',
'item_cnt_month_lag_7', 'item_cnt_month_lag_8', 'item_cnt_month_lag_9',
'item_cnt_month_lag_10', 'item_cnt_month_lag_11', 'item_cnt_month_lag_12',
'shop_id_cnt_month_mean_Expanding_lag_1',
'shop_id_cnt_month_mean_Expanding_lag_2',
'shop_id_cnt_month_mean_Expanding_lag_3',
'shop_id_cnt_month_mean_Expanding_lag_4',
'shop_id_cnt_month_mean_Expanding_lag_12',
'item_id_cnt_month_mean_Expanding_lag_1',
'item_id_cnt_month_mean_Expanding_lag_2',
'item_id_cnt_month_mean_Expanding_lag_3',
'item_id_cnt_month_mean_Expanding_lag_4',
'item_id_cnt_month_mean_Expanding_lag_12',
'shop_item_id_cnt_month_mean_Smooth_lag_1',
'shop_item_id_cnt_month_mean_Smooth_lag_2',
'shop_item_id_cnt_month_mean_Smooth_lag_3',
'shop_item_id_cnt_month_mean_Smooth_lag_4',
'shop_item_id_cnt_month_mean_Smooth_lag_12',
'shop_type_id_cnt_month_mean_Expanding_lag_1',
'shop_type_id_cnt_month_mean_Expanding_lag_2',
'shop_type_id_cnt_month_mean_Expanding_lag_3',
'shop_type_id_cnt_month_mean_Expanding_lag_4',
'shop_type_id_cnt_month_mean_Expanding_lag_12',
'city_id_cnt_month_mean_Expanding_lag_1',
'city_id_cnt_month_mean_Expanding_lag_2',
'city_id_cnt_month_mean_Expanding_lag_3',
'item_category_id_cnt_month_mean_Expanding_lag_1',
'item_category_id_cnt_month_mean_Expanding_lag_2',
'item_category_id_cnt_month_mean_Expanding_lag_3',
'item_category_id_cnt_month_mean_Expanding_lag_4',
'item_category_id_cnt_month_mean_Expanding_lag_12',
'item_type_id_cnt_month_mean_Expanding_lag_1',
'item_type_id_cnt_month_mean_Expanding_lag_2',
'item_subtype_id_cnt_month_mean_Expanding_lag_1',
'item_subtype_id_cnt_month_mean_Expanding_lag_2',
'item_subtype_id_cnt_month_mean_Expanding_lag_3',
'item_subtype_id_cnt_month_mean_Expanding_lag_4',
'cat_type_id_cnt_month_mean_Expanding_lag_1',
'cat_type_id_cnt_month_mean_Expanding_lag_2',
'cat_type_id_cnt_month_mean_Expanding_lag_3',
'cat_type_id_cnt_month_mean_Expanding_lag_4',
'cat_type_id_cnt_month_mean_Expanding_lag_12',
'cat_subtype_id_cnt_month_mean_Expanding_lag_1',
'cat_subtype_id_cnt_month_mean_Expanding_lag_2',
'cat_subtype_id_cnt_month_mean_Expanding_lag_3',
'cat_subtype_id_cnt_month_mean_Expanding_lag_4',
'cat_subtype_id_cnt_month_mean_Expanding_lag_12', 'delta_price_lag',
'delta_revenue_lag_1', 'month', 'year', 'days_of_month',
'months_since_item_shop_last_sale_lag_1',
'months_since_shop_item_first_sale_lag_1', 'months_since_item_last_sale_lag_1',
'months_since_item_first_sale_lag_1']
```

[19]:
```
start_first_level_total = time.perf_counter()
```

```
scoringMethod = 'r2'; from sklearn.metrics import mean_squared_error; from math␣
↪import sqrt
```

[20]:
```
#train meta-features

months_to_generate_meta_features = range(24,last_block + 1)
mask = dates.isin(months_to_generate_meta_features)
Target = 'item_cnt_month'
y_all_level2 = sales[Target][mask].values
X_all_level2 = np.zeros([y_all_level2.shape[0], num_first_level_models])
```

Parameters tuned on separate notebook. The optimal parameters found are as follows:

XGBoost:
`{'colsample_bytree': 0.6, 'eta': 0.3, 'max_depth': 10, 'min_child_weight': 300, 'seed': 42, 'subsample': 1.0, 'n_estimators': 125}`

Light GBM:
`{'bagging_fraction': 0.2, 'feature_fraction': 0.8, 'learning_rate': 0.05, 'max_depth': -1, 'min_data_in_leaf': 300, 'num_leaves': 128, 'seed': 42, 'n_estimators': 200}`

Random Forest:
`{'bootstrap': True, 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 200}`

SGD:
`{'alpha': 0.0001, 'penalty': 'l1'}`

Keras:
`{'activation': 'tanh', 'activation2': 'relu', 'batch_size': 5000, 'dropout_rate': 0.0, 'epochs': 50, 'init': 'he_normal', 'neurons': 200, 'optimizer': 'Adam'}`

[21]:
```
#hyperparameters tuned on separate notebook and loaded here
#xgb params
with open("XBG_Params.pkl", 'rb') as file:
    xgb_params = pickle.load(file)

xgb_params
```

[21]:
```
{'colsample_bytree': 0.6,
 'eta': 0.3,
 'max_depth': 10,
 'min_child_weight': 300,
 'seed': 42,
 'subsample': 1.0,
 'n_estimators': 125}
```

[22]:
```
with open("LBG_Params.pkl", 'rb') as file:
    lgb_params = pickle.load(file)

lgb_params
```

[22]:
```
{'bagging_fraction': 0.2,
 'feature_fraction': 0.8,
 'learning_rate': 0.05,
 'max_depth': -1,
 'min_data_in_leaf': 300,
 'num_leaves': 128,
 'seed': 42,
 'n_estimators': 200}
```

[23]:
```
with open("RF_Params.pkl", 'rb') as file:
    rf_params = pickle.load(file)

rf_params
```

[23]:
```
{'bootstrap': True,
 'max_depth': 20,
 'max_features': 'sqrt',
 'min_samples_leaf': 4,
 'min_samples_split': 5,
 'n_estimators': 200}
```

[24]:
```
with open("SGD_Params.pkl", 'rb') as file:
    sgd_params = pickle.load(file)

sgd_params
```

[24]:
```
{'alpha': 0.0001, 'penalty': 'l1', 'random_state': 0}
```

[25]:
```
with open("Keras_Params.pkl", 'rb') as file:
    keras_params = pickle.load(file)

keras_params
```

[25]:
```
{'activation': 'tanh',
 'activation2': 'relu',
 'batch_size': 5000,
 'dropout_rate': 0.0,
 'epochs': 50,
 'init': 'he_normal',
 'neurons': 200,
 'optimizer': 'Adam'}
```

```
[26]: #fill `X_train_level2` with metafeatures
slice_start = 0

for cur_block_num in tqdm(months_to_generate_meta_features):
    print('-' * 50)
    print('Start training for month%d'% cur_block_num)
    start_cur_month = time.perf_counter()

    cur_X_train = sales.loc[dates <  cur_block_num][feature_cols]
    cur_X_test =  sales.loc[dates == cur_block_num][feature_cols]

    cur_y_train = sales.loc[dates <  cur_block_num, Target].values
    cur_y_test =  sales.loc[dates == cur_block_num, Target].values

    # Create Numpy arrays of train, test and target dataframes to feed into␣
 ↪models
    train_x = cur_X_train.values
    train_y = cur_y_train.ravel()
    test_x = cur_X_test.values
    test_y = cur_y_test.ravel()

    preds = []

    from sklearn.linear_model import (LinearRegression, SGDRegressor)
    import lightgbm as lgb
    import xgboost as xgb
    from sklearn.ensemble import RandomForestRegressor

    sgdr = SGDRegressor(penalty = 'l1', alpha = 0.0001, random_state = SEED )

    rf = RandomForestRegressor(
        bootstrap=True,
        max_depth=20,
        max_features='sqrt',
        min_samples_leaf=4,
        min_samples_split=5,
        n_estimators=200)

    lgb_params = {
        'feature_fraction': 0.8,
        'metric': 'rmse',
        'min_data_in_leaf': 300,
        'bagging_fraction': 0.2,
        'learning_rate': 0.05,
        'max_depth': -1,
        'objective': 'mse',
        'num_leaves': 128,
        'seed': 42,
        'n_estimators': 200,
        'verbose':0
    }

    xgb_params = {
        'colsample_bytree': 0.6,
        'eta': 0.3,
        'max_depth': 10,
        'min_child_weight': 300,
        'n_estimators': 125,
        'seed': 42,
        'subsample': 1.0,
        'verbosity': 0
    }

    print('Training Model %d: %s'%(len(preds), 'XGBoost'))

    start = time.perf_counter()

    estimator = xgb.train(xgb_params, xgb.DMatrix(data=train_x, label=train_y))
    pred_test = estimator.predict(xgb.DMatrix(test_x))
    preds.append(pred_test)

    run = time.perf_counter() - start

    print('{} runs for {:.2f} seconds.'.format('XGBoost', run))
    print()

    print('Training Model %d: %s'%(len(preds), 'Lightgbm'))

    start = time.perf_counter()

    estimator = lgb.train(lgb_params, lgb.Dataset(train_x, label=train_y))
    pred_test = estimator.predict(test_x)
    preds.append(pred_test)

    run = time.perf_counter() - start

    print('{} runs for {:.2f} seconds.'.format('Lightgbm', run))
    print()

    estimators = [sgdr, rf]
    for estimator in estimators:
        print('Training Model %d: %s'%(len(preds), estimator.__class__.
↪__name__))
        start = time.perf_counter()
        estimator.fit(train_x, train_y)
        pred_test = estimator.predict(test_x)
        preds.append(pred_test)

        run = time.perf_counter() - start
        print('{} runs for {:.2f} seconds.'.format(estimator.__class__.
↪__name__, run))
        print()


    print('Training Model %d: %s'%(len(preds), 'Keras'))

    start = time.perf_counter()

    from keras.models import Sequential
    from keras.layers import Dense
    from keras.wrappers.scikit_learn import KerasRegressor


    def baseline_model():
        # create model
        model = Sequential()
        model.add(Dense(200, input_dim=train_x.shape[1],␣
↪kernel_initializer='he_normal', activation='tanh'))
        model.add(Dense(1, kernel_initializer='he_normal', activation='relu'))
        # Compile model
        model.compile(loss='mse', optimizer='Adam', metrics=['mse'])
        return model


    estimator = KerasRegressor(build_fn=baseline_model, verbose=0, epochs=50,␣
↪batch_size = 5000)

    estimator.fit(train_x, train_y)
    pred_test = estimator.predict(test_x)
    preds.append(pred_test)

    run = time.perf_counter() - start
    print('{} runs for {:.2f} seconds.'.format('Keras', run))

    cur_month_run_total = time.perf_counter() - start_cur_month

    print('Total running time was {:.2f} minutes.'.format(cur_month_run_total/
↪60))
    print('-' * 50)

    slice_end = slice_start + cur_X_test.shape[0]
    X_all_level2[ slice_start : slice_end , :] = np.c_[preds].transpose()
    slice_start = slice_end
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

--------------------------------------------------
Start training for month24
Training Model 0: XGBoost
XGBoost runs for 46.14 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 57.14 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 16.68 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 3388.77 seconds.

Training Model 4: Keras


  9%|         | 1/11 [1:07:05<11:10:54, 4025.46s/it]

Keras runs for 515.55 seconds.
Total running time was 67.09 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month25
Training Model 0: XGBoost
XGBoost runs for 52.51 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 59.99 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 18.36 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 4280.31 seconds.

Training Model 4: Keras


 18%|        | 2/11 [3:37:06<13:47:43, 5518.18s/it]
```

```
Keras runs for 523.81 seconds.
Total running time was 82.27 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month26
Training Model 0: XGBoost
XGBoost runs for 53.21 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 59.85 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 18.45 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 3807.67 seconds.

Training Model 4: Keras



 27%|         | 3/11 [4:52:04<11:34:55, 5211.99s/it]

Keras runs for 556.77 seconds.
Total running time was 74.96 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month27
Training Model 0: XGBoost
XGBoost runs for 53.33 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 55.10 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 21.02 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 4319.26 seconds.

Training Model 4: Keras



 36%|        | 4/11 [6:16:06<10:02:08, 5161.17s/it]

Keras runs for 592.12 seconds.
Total running time was 84.04 minutes.
--------------------------------------------------
--------------------------------------------------
```

```
Start training for month28
Training Model 0: XGBoost
XGBoost runs for 55.97 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 56.19 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 22.67 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 4601.93 seconds.

Training Model 4: Keras



 45%|       | 5/11 [7:45:08<8:41:32, 5215.36s/it]

Keras runs for 603.41 seconds.
Total running time was 89.03 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month29
Training Model 0: XGBoost
XGBoost runs for 60.19 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 59.14 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 21.60 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 4891.86 seconds.

Training Model 4: Keras



 55%|       | 6/11 [9:19:45<7:26:09, 5353.92s/it]

Keras runs for 642.83 seconds.
Total running time was 94.62 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month30
Training Model 0: XGBoost
XGBoost runs for 82.18 seconds.
```

```
Training Model 1: Lightgbm
Lightgbm runs for 62.91 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 22.56 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 5224.10 seconds.

Training Model 4: Keras



 64%|       | 7/11 [11:01:01<6:11:22, 5570.53s/it]

Keras runs for 682.56 seconds.
Total running time was 101.27 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month31
Training Model 0: XGBoost
XGBoost runs for 85.40 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 65.56 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 23.92 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 5535.80 seconds.

Training Model 4: Keras



 73%|       | 8/11 [12:47:59<4:51:14, 5824.73s/it]

Keras runs for 705.44 seconds.
Total running time was 106.96 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month32
Training Model 0: XGBoost
XGBoost runs for 91.65 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 68.30 seconds.

Training Model 2: SGDRegressor
```

```
SGDRegressor runs for 24.95 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 5797.19 seconds.

Training Model 4: Keras



 82%|      | 9/11 [14:40:07<3:23:11, 6095.80s/it]

Keras runs for 744.41 seconds.
Total running time was 112.14 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month33
Training Model 0: XGBoost
XGBoost runs for 96.42 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 71.00 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 26.53 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 6080.74 seconds.

Training Model 4: Keras



 91%|      | 10/11 [16:37:54<1:46:26, 6386.90s/it]

Keras runs for 789.55 seconds.
Total running time was 117.77 minutes.
--------------------------------------------------
--------------------------------------------------
Start training for month34
Training Model 0: XGBoost
XGBoost runs for 101.11 seconds.

Training Model 1: Lightgbm
Lightgbm runs for 74.35 seconds.

Training Model 2: SGDRegressor
SGDRegressor runs for 27.23 seconds.

Training Model 3: RandomForestRegressor
RandomForestRegressor runs for 6312.06 seconds.
```

```
Training Model 4: Keras
```

```
100%|     | 11/11 [19:06:30<00:00, 6253.64s/it]

Keras runs for 884.63 seconds.
Total running time was 123.36 minutes.
--------------------------------------------------
```

[27]:
```
#split train and test

test_nrow = len(preds[0])
X_train_level2 = X_all_level2[ : -test_nrow, :]
X_test_level2 = X_all_level2[ -test_nrow: , :]
y_train_level2 = y_all_level2[ : -test_nrow]
y_test_level2 = y_all_level2[ -test_nrow : ]
```

[28]:
```
#RMSE for individual models
xgb_pred = X_train_level2[:, 0].clip(0, 20)
lgb_pred = X_train_level2[:, 1].clip(0, 20)
sgdr_pred = X_train_level2[:, 2].clip(0, 20)
rf_pred = X_train_level2[:, 3].clip(0, 20)
keras_pred = X_train_level2[:, 4].clip(0, 20)
print('Train RMSE for %s is %f' %('xgb_pred',␣
 ↪sqrt(mean_squared_error(y_train_level2, xgb_pred))))
print('Train RMSE for %s is %f' %('lgb_pred',␣
 ↪sqrt(mean_squared_error(y_train_level2, lgb_pred))))
print('Train RMSE for %s is %f' %('sgdr_pred',␣
 ↪sqrt(mean_squared_error(y_train_level2, sgdr_pred))))
print('Train RMSE for %s is %f' %('rf_pred',␣
 ↪sqrt(mean_squared_error(y_train_level2, rf_pred))))
print('Train RMSE for %s is %f' %('keras_pred',␣
 ↪sqrt(mean_squared_error(y_train_level2, keras_pred))))
```

```
Train RMSE for xgb_pred is 0.767315
Train RMSE for lgb_pred is 0.749203
Train RMSE for sgdr_pred is 0.856755
Train RMSE for rf_pred is 0.760995
Train RMSE for keras_pred is 0.866221
```

[29]:
```
#Ensembling
pred_list = {}

#second level learning model via linear regression
print('Training Second level learning model via linear regression')
```

```
from sklearn.linear_model import (LinearRegression, SGDRegressor)

lr = LinearRegression()
lr.fit(X_train_level2, y_train_level2)

#compute R-squared on train and test sets
# print('Train R-squared for %s is %f' %('test_preds_lr_stacking',␣
 ↪sqrt(mean_squared_error(y_train_level2, lr.predict(X_train_level2)))))

test_preds_lr_stacking = lr.predict(X_test_level2)
train_preds_lr_stacking = lr.predict(X_train_level2)
print('Train RMSE for %s is %f' %('train_preds_lr_stacking',␣
 ↪sqrt(mean_squared_error(y_train_level2, train_preds_lr_stacking))))

pred_list['test_preds_lr_stacking'] = test_preds_lr_stacking
if Validation:
    print('Test RMSE for %s is %f' %('test_preds_lr_stacking',␣
 ↪sqrt(mean_squared_error(y_test_level2, test_preds_lr_stacking))))
```

```
Training Second level learning model via linear regression
Train RMSE for train_preds_lr_stacking is 0.745656
```

[30]:
```
#second level learning model via SGDRegressor
print('Training Second level learning model via SGDRegressor')
sgdr = SGDRegressor(
    penalty = 'l2' ,
    random_state = SEED )

sgdr.fit(X_train_level2, y_train_level2)

#compute R-squared on train and test sets
# print('Train R-squared for %s is %f' %('test_preds_lr_stacking',␣
 ↪sqrt(mean_squared_error(y_train_level2, lr.predict(X_train_level2)))))
test_preds_sgdr_stacking = sgdr.predict(X_test_level2)
train_preds_sgdr_stacking = sgdr.predict(X_train_level2)
print('Train RMSE for %s is %f' %('train_preds_lr_stacking',␣
 ↪sqrt(mean_squared_error(y_train_level2, train_preds_sgdr_stacking))))

pred_list['test_preds_sgdr_stacking'] = test_preds_sgdr_stacking

if Validation:
    print('Test RMSE for %s is %f' %('test_preds_sgdr_stacking',␣
 ↪sqrt(mean_squared_error(y_test_level2, test_preds_sgdr_stacking))))

#print('%0.2f min: Finish training second level model'%((time.time() -␣
 ↪start_time)/60))
```

```
Training Second level learning model via SGDRegressor
Train RMSE for train_preds_lr_stacking is 0.761842
```

```
[31]: #submission
      if not Validation:
          submission = pd.read_csv('sample_submission.csv')
          ver = 3

          for pred_ver in ['lr_stacking', 'sgdr_stacking']:
              print(pred_list['test_preds_' + pred_ver].clip(0,20).mean())
              submission['item_cnt_month'] = pred_list['test_preds_' + pred_ver].
       ↪clip(0,20)
              submission[['ID', 'item_cnt_month']].to_csv('ver%d_%s.csv' % (ver,␣
       ↪pred_ver), index = False)
```

```
0.283260703911113
0.3396436425815545
```

```
[51]: #RF, LGBM and XGBoost produced the lowest RMSE overall, so these will now be␣
       ↪extracted to build an ensemble model to predict
      X_train_level2_v2 = X_train_level2[:,[0,1,3]]
      X_test_level2_v2 = X_test_level2[:,[0,1,3]]
```

```
[54]: #Ensembling v2
      pred_list = {}

      #second level learning model via linear regression
      print('Training Second level learning model via linear regression')

      from sklearn.linear_model import (LinearRegression, SGDRegressor)

      lr = LinearRegression()
      lr.fit(X_train_level2_v2, y_train_level2)

      #compute R-squared on train and test sets
      # print('Train R-squared for %s is %f' %('test_preds_lr_stacking',␣
       ↪sqrt(mean_squared_error(y_train_level2, lr.predict(X_train_level2))))

      test_preds_lr_stacking_v2 = lr.predict(X_test_level2_v2)
      train_preds_lr_stacking_v2 = lr.predict(X_train_level2_v2)
      print('Train RMSE for %s is %f' %('train_preds_lr_stacking_v2',␣
       ↪sqrt(mean_squared_error(y_train_level2, train_preds_lr_stacking_v2))))

      pred_list['test_preds_lr_stacking_v2'] = test_preds_lr_stacking_v2
      if Validation:
          print('Test RMSE for %s is %f' %('test_preds_lr_stacking_v2',␣
       ↪sqrt(mean_squared_error(y_test_level2, test_preds_lr_stacking_v2))))
```

```
Training Second level learning model via linear regression
Train RMSE for train_preds_lr_stacking_v2 is 0.747544
```

```
[55]: #second level learning model via SGDRegressor
print('Training Second level learning model via SGDRegressor')
sgdr = SGDRegressor(
    penalty = 'l2' ,
    random_state = SEED )

sgdr.fit(X_train_level2_v2, y_train_level2)

#compute R-squared on train and test sets
# print('Train R-squared for %s is %f' %('test_preds_lr_stacking',␣
 ↪sqrt(mean_squared_error(y_train_level2, lr.predict(X_train_level2)))))
test_preds_sgdr_stacking_v2 = sgdr.predict(X_test_level2_v2)
train_preds_sgdr_stacking_v2 = sgdr.predict(X_train_level2_v2)
print('Train RMSE for %s is %f' %('train_preds_lr_stacking_v2',␣
 ↪sqrt(mean_squared_error(y_train_level2, train_preds_sgdr_stacking_v2))))

pred_list['test_preds_sgdr_stacking_v2'] = test_preds_sgdr_stacking_v2

if Validation:
    print('Test RMSE for %s is %f' %('test_preds_sgdr_stacking_v2',␣
 ↪sqrt(mean_squared_error(y_test_level2, test_preds_sgdr_stacking_v2))))

#print('%0.2f min: Finish training second level model'%((time.time() -␣
 ↪start_time)/60))
```

```
Training Second level learning model via SGDRegressor
Train RMSE for train_preds_lr_stacking_v2 is 0.755024
```

```
[56]: #submission
if not Validation:
    submission = pd.read_csv('sample_submission.csv')
    ver = 3

    for pred_ver in ['lr_stacking_v2', 'sgdr_stacking_v2']:
        print(pred_list['test_preds_' + pred_ver].clip(0,20).mean())
        submission['item_cnt_month'] = pred_list['test_preds_' + pred_ver].
 ↪clip(0,20)
        submission[['ID', 'item_cnt_month']].to_csv('ver%d_%s.csv' % (ver,␣
 ↪pred_ver), index = False)
```

```
0.281080262941557
0.32307098985203775
```

Despite having a higher RMSE for the training set, the version 2 SGDR ensemble performed the best on the test set when submitted to Kaggle.

Public and private LB scores for this project were: 0.950299 and 0.949603.

```
[ ]:
```