predict_future_sales

September 2, 2020

This kernel is going to solve Predict Future Sales Competition on Kaggle.

Competition Description:

This challenge serves as final project for the "How to win a data science competition" Coursera course.

In this competition you will work with a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company.

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

1 Import Libraries

First, we import necessary libraries, such as:

[273]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

2 Import The Data

[274]:
```
items = pd.read_csv('/kaggle/input/
 ↪competitive-data-science-predict-future-sales/items.csv')
shops = pd.read_csv('/kaggle/input/
 ↪competitive-data-science-predict-future-sales/shops.csv')
item_categories = pd.read_csv('/kaggle/input/
 ↪competitive-data-science-predict-future-sales/item_categories.csv')

train = pd.read_csv('/kaggle/input/
 ↪competitive-data-science-predict-future-sales/sales_train.csv')
```

```
test = pd.read_csv('/kaggle/input/competitive-data-science-predict-future-sales/
 ↪test.csv')

sample_submission = pd.read_csv('/kaggle/input/
 ↪competitive-data-science-predict-future-sales/sample_submission.csv')
```

3 Read The Data

[275]:
```
print('item_categories')
display(item_categories.head())

print('items')
display(items.head())

print('shops')
display(shops.head())

print('train')
display(train.head())

print('test')
display(test.head())

print('sample_submission')
display(sample_submission.head())
```

```
item_categories
```

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

```
items
```

	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
```

shops

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

train

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

test

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

sample_submission

```
   ID  item_cnt_month
0   0             0.5
1   1             0.5
2   2             0.5
3   3             0.5
4   4             0.5
```

- Check train info

[276]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
```

```
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   date            object
 1   date_block_num  int64
 2   shop_id         int64
 3   item_id         int64
 4   item_price      float64
 5   item_cnt_day    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB
```

- Check for missing values

[277]:
```
print('train')
display(train.isnull().sum())

print('test')
display(test.isnull().sum())
```

```
train
```

```
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
```

```
test
```

```
ID         0
shop_id    0
item_id    0
dtype: int64
```

- Quick look using `describe()` function

[278]:
```
print('train')
display(train.describe(include='all'))

print('test')
display(test.describe(include='all'))
```

```
train
```

```
                date  date_block_num       shop_id       item_id    item_price  \
count      2935849    2.935849e+06  2.935849e+06  2.935849e+06  2.935849e+06
```

	date	date_block_num	shop_id	item_id	item_price
unique	1034	NaN	NaN	NaN	NaN
top	28.12.2013	NaN	NaN	NaN	NaN
freq	9434	NaN	NaN	NaN	NaN
mean	NaN	1.456991e+01	3.300173e+01	1.019723e+04	8.908532e+02
std	NaN	9.422988e+00	1.622697e+01	6.324297e+03	1.729800e+03
min	NaN	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00
25%	NaN	7.000000e+00	2.200000e+01	4.476000e+03	2.490000e+02
50%	NaN	1.400000e+01	3.100000e+01	9.343000e+03	3.990000e+02
75%	NaN	2.300000e+01	4.700000e+01	1.568400e+04	9.990000e+02
max	NaN	3.300000e+01	5.900000e+01	2.216900e+04	3.079800e+05

	item_cnt_day
count	2.935849e+06
unique	NaN
top	NaN
freq	NaN
mean	1.242641e+00
std	2.618834e+00
min	-2.200000e+01
25%	1.000000e+00
50%	1.000000e+00
75%	1.000000e+00
max	2.169000e+03

test

	ID	shop_id	item_id
count	214200.000000	214200.000000	214200.000000
mean	107099.500000	31.642857	11019.398627
std	61834.358168	17.561933	6252.644590
min	0.000000	2.000000	30.000000
25%	53549.750000	16.000000	5381.500000
50%	107099.500000	34.500000	11203.000000
75%	160649.250000	47.000000	16071.500000
max	214199.000000	59.000000	22167.000000

Quick observations: - There are no missing values. - The train and test datasets did not match in term of features. - There is minus value(s) in item_price. - There is minus value(s) in item_cnt_day.

4 Exploratory Data Analysis

4.0.1 Removing Duplicates

[280]:
```
#drop duplicates
subset = ['date','date_block_num','shop_id','item_id','item_cnt_day']
print(train.duplicated(subset=subset).value_counts())
train.drop_duplicates(subset=subset, inplace=True)
```

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

4.0.2 Check negative values in item_price

[281]:
```
train[train['item_price'] < 0]
```

[281]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
484683	15.05.2013	4	32	2973	-1.0	1.0

Since there is only 1 negative value in item_price, we can just drop that because it won't affect the prediction too much.

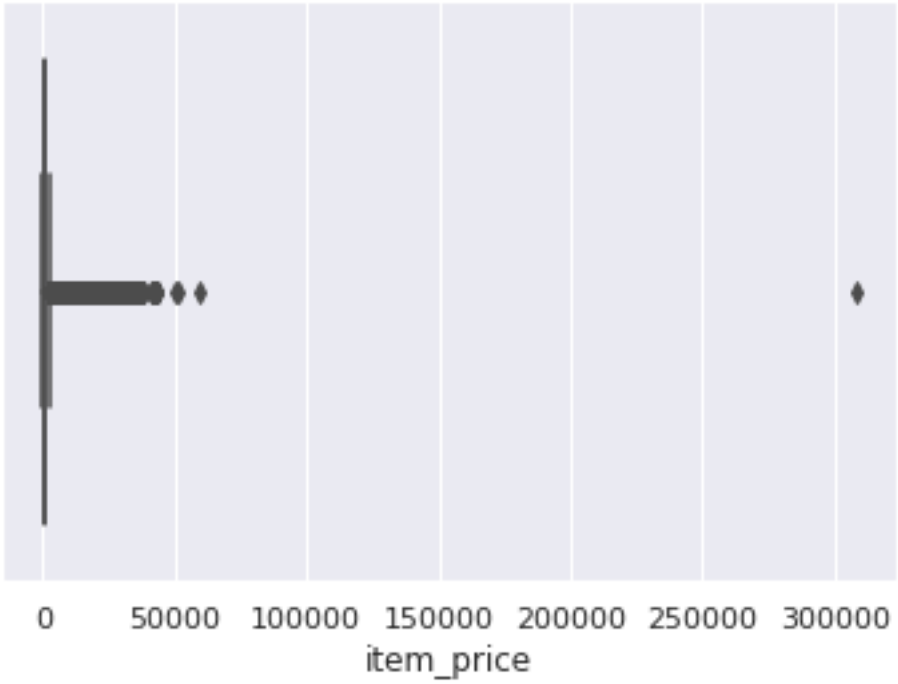
[282]:
```
#drop negative value in item_price
train = train[train['item_price'] > 0]
```

[283]:
```
train = train[train['item_cnt_day'] > 0]
```

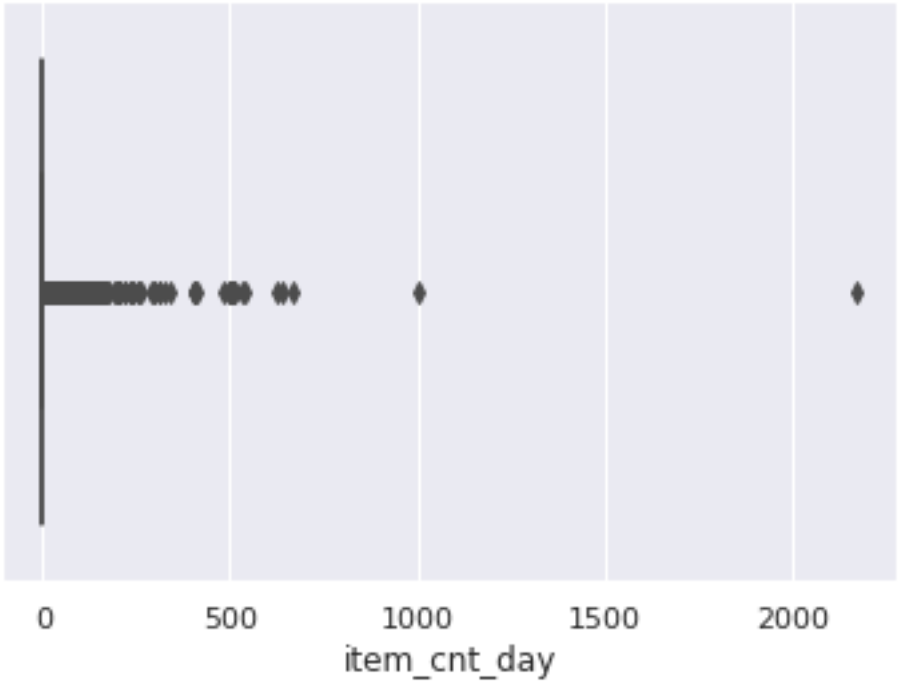
4.0.3 Cleaning item_price and item_cnt_day

- Check min and max

[284]:
```
sns.boxplot(train['item_price']);
```



[285]:
```
sns.boxplot(train['item_cnt_day']);
```



- Drop outliers

[286]:
```
#define a drop outliers function
def drop_outliers(df, feature, percentile_high = .99):
    '''df (dataframe)           : dataset
       feature (string)         : column
       percentile_high (float)  : upper limit
       ..................................................
    '''
    #train size before dropping values
    shape_init = df.shape[0]

    #get percentile value
    max_value = df[feature].quantile(percentile_high)

    #drop outliers
    print('dropping outliers...')
    df = df[df[feature] < max_value]

    print(str(shape_init - df.shape[0]) + ' ' + feature + ' values over ' +␣
 ↪str(max_value) + ' have been removed' )

    return df
```

[287]:
```
#drop outliers in item_price feature
train = drop_outliers(train, 'item_price')
```

```
dropping outliers…
29347 item_price values over 5999.0 have been removed
```

[288]:
```
#drop outliers in item_cnt_day
train = drop_outliers(train, 'item_cnt_day')
```

```
dropping outliers…
37347 item_cnt_day values over 5.0 have been removed
```

4.0.4 Price

Make a dataframe with item_price feature group by shop_id and item_id to get price for each item per shop. We can use this dataframe to create item_price feature for test dataset.

[289]:
```
prices_shop_df = train[['shop_id','item_id','item_price']]
prices_shop_df = prices_shop_df.groupby(['shop_id','item_id']).apply(lambda df:␣
 ↪df['item_price'][-2:].mean())
prices_shop_df = prices_shop_df.to_frame(name = 'item_price')

prices_shop_df
```

[289]:
```
                  item_price
shop_id item_id
0       30            265.0
        31            434.0
        32            221.0
        33            347.0
        35            247.0
…                     …
59      22154         999.0
        22155         149.0
        22162         349.0
        22164         699.0
        22167         299.0

[418764 rows x 1 columns]
```

Now we can merge this dataframe with test dataset to create item_price feature in test dataset.

[290]:
```
test = pd.merge(test, prices_shop_df, how='left',␣
↪left_on=['shop_id','item_id'], right_on=['shop_id','item_id'])

test.head()
```

[290]:

	ID	shop_id	item_id	item_price
0	0	5	5037	749.25
1	1	5	5320	NaN
2	2	5	5233	1199.00
3	3	5	5232	599.00
4	4	5	5268	NaN

[291]:
```
#check for missing values
test['item_price'].isnull().sum()
```

[291]:
```
104035
```

There are still missing values in test's item_price. We will fill this later by creating more features from item_categories.

4.0.5 Transform Data in Train Dataset As Monthly

[292]:
```
#split content in date into month and year
train['month'] = [date.split('.')[1] for date in train['date']]
train['year'] = [date.split('.')[2] for date in train['date']]

#drop date and date_block_num features
train.drop(['date','date_block_num'], axis=1, inplace=True)
```

```
#create month and year features fot test dataset
test['month'] = '11'
test['year'] = '2015'
```

[293]:
```
#change item_cnt_day into item_cnt_month
train_monthly = train.groupby(['year','month','shop_id','item_id'],␣
 ↪as_index=False)[['item_cnt_day']].sum()
train_monthly.rename(columns={'item_cnt_day': 'item_cnt_month'}, inplace=True)

train_monthly = pd.merge(train_monthly, prices_shop_df, how='left',␣
 ↪left_on=['shop_id','item_id'], right_on=['shop_id','item_id'])

train_monthly.head()
```

[293]:

	year	month	shop_id	item_id	item_cnt_month	item_price
0	2013	01	0	32	6.0	221.0
1	2013	01	0	33	3.0	347.0
2	2013	01	0	35	1.0	247.0
3	2013	01	0	43	1.0	221.0
4	2013	01	0	51	2.0	127.0

[294]:
```
train = train_monthly
```

4.0.6 Reindex test dataset

[295]:
```
test = test.
 ↪reindex(columns=['ID','year','month','shop_id','item_id','item_price'])

test.head()
```

[295]:

	ID	year	month	shop_id	item_id	item_price
0	0	2015	11	5	5037	749.25
1	1	2015	11	5	5320	NaN
2	2	2015	11	5	5233	1199.00
3	3	2015	11	5	5232	599.00
4	4	2015	11	5	5268	NaN

4.0.7 Exploring other datasets

- Exploring Item Categories dataset

[297]:
```
#extract main categories
item_categories['main_category'] = [x.split(' - ')[0] for x in␣
 ↪item_categories['item_category_name']]
```

```
#some items don't have sub-categories. For those, we will use None as a␣
 ↪sub-category (consider the main category as a sub)
sub_categories = []
for i in range(len(item_categories)):
    try:
        sub_categories.append(item_categories['item_category_name'][i].split('␣
 ↪- ')[1])

    except IndexError as e:
        sub_categories.append('None')
        #sub_categories.append(item_categories['main_category'][i])

item_categories['sub_category'] = sub_categories

#drop item_category_name
item_categories.drop(['item_category_name'], axis=1, inplace=True)

item_categories.head()
```

[297]:

	item_category_id	main_category	sub_category
0	0	PC	/
1	1		PS2
2	2		PS3
3	3		PS4
4	4		PSP

- Exploring Items Dataset

[298]:
```
#merge with item_categories
items = pd.merge(items, item_categories, how='left')

#drop item_name and item_category_id
items.drop(['item_name','item_category_id'], axis=1, inplace=True)

items.head()
```

[298]:

	item_id	main_category	sub_category
0	0		DVD
1	1	( )	
2	2		DVD
3	3		DVD
4	4		DVD

[299]:
```
#merge to train and test datasets
train = pd.merge(train, items, how='left')
test = pd.merge(test, items, how='left')
```

- Exploring Shops Dataset

[300]:
```
from string import punctuation

# replace all the punctuation in the shop_name columns
shops["shop_name_cleaned"] = shops["shop_name"].apply(lambda s: "".join([x for␣
↪x in s if x not in punctuation]))

# extract the city name
shops["shop_city"] = shops["shop_name_cleaned"].apply(lambda s: s.split()[0])

#extract the type
shops["shop_type"] = shops["shop_name_cleaned"].apply(lambda s: s.split()[1])

#extract shop's name
shops["shop_name"] = shops["shop_name_cleaned"].apply(lambda s: " ".join(s.
↪split()[2:]))

shops.drop(['shop_name_cleaned'], axis=1, inplace=True)

shops.head()
```

[300]:

	shop_name	shop_id	shop_city	shop_type
0	56	0		
1		1		
2		2		
3		3		
4		4		

[301]:
```
#merge to train and test datasets
train = pd.merge(train, shops, how='left')
test = pd.merge(test, shops, how='left')
```

Display current train and test datasets

[302]:
```
print('train')
display(train.head())

print('test')
display(test.head())
```

train

	year	month	shop_id	item_id	item_cnt_month	item_price	main_category	\
0	2013	01	0	32	6.0	221.0		
1	2013	01	0	33	3.0	347.0		
2	2013	01	0	35	1.0	247.0		
3	2013	01	0	43	1.0	221.0		
4	2013	01	0	51	2.0	127.0		

```
  sub_category shop_name shop_city     shop_type
0          DVD   56
1      Blu-Ray   56
2          DVD   56
3          DVD   56
4          MP3   56
```

```
test
```

```
   ID  year month  shop_id  item_id  item_price main_category  \
0   0  2015    11        5     5037      749.25
1   1  2015    11        5     5320         NaN
2   2  2015    11        5     5233     1199.00
3   3  2015    11        5     5232      599.00
4   4  2015    11        5     5268         NaN
```

```
                 sub_category shop_name shop_city shop_type
0                         PS3
1  CD
2                         PS3
3                    XBOX 360
4                         PS4
```

4.0.8 Fill missing values in item_price (by item categories)

[303]:
```
#fill missing values with median of each main_category and sub_category
test['item_price'] = test.
↪groupby(['main_category','sub_category'])['item_price'].apply(lambda df: df.
↪fillna(df.median()))
```

[304]:
```
test['item_price'].isnull().sum()
```

[304]: 840

[305]:
```
#fill missing values with median of each sub_category
test['item_price'] = test.groupby(['sub_category'])['item_price'].apply(lambda␣
↪df: df.fillna(df.median()))
```

[306]:
```
test['item_price'].isnull().sum()
```

[306]: 42

Show remaining missing values

[307]:
```
test[test['item_price'].isnull()]
```

[307]:

	ID	year	month	shop_id	item_id	item_price	main_category	\
3514	3514	2015	11	5	5441	NaN	PC	
8614	8614	2015	11	4	5441	NaN	PC	
13714	13714	2015	11	6	5441	NaN	PC	
18814	18814	2015	11	3	5441	NaN	PC	
23914	23914	2015	11	2	5441	NaN	PC	
29014	29014	2015	11	7	5441	NaN	PC	
34114	34114	2015	11	10	5441	NaN	PC	
39214	39214	2015	11	12	5441	NaN	PC	
44314	44314	2015	11	28	5441	NaN	PC	
49414	49414	2015	11	31	5441	NaN	PC	
54514	54514	2015	11	26	5441	NaN	PC	
59614	59614	2015	11	25	5441	NaN	PC	
64714	64714	2015	11	22	5441	NaN	PC	
69814	69814	2015	11	24	5441	NaN	PC	
74914	74914	2015	11	21	5441	NaN	PC	
80014	80014	2015	11	15	5441	NaN	PC	
85114	85114	2015	11	16	5441	NaN	PC	
90214	90214	2015	11	18	5441	NaN	PC	
95314	95314	2015	11	14	5441	NaN	PC	
100414	100414	2015	11	19	5441	NaN	PC	
105514	105514	2015	11	42	5441	NaN	PC	
110614	110614	2015	11	50	5441	NaN	PC	
115714	115714	2015	11	49	5441	NaN	PC	
120814	120814	2015	11	53	5441	NaN	PC	
125914	125914	2015	11	52	5441	NaN	PC	
131014	131014	2015	11	47	5441	NaN	PC	
136114	136114	2015	11	48	5441	NaN	PC	
141214	141214	2015	11	57	5441	NaN	PC	
146314	146314	2015	11	58	5441	NaN	PC	
151414	151414	2015	11	59	5441	NaN	PC	
156514	156514	2015	11	55	5441	NaN	PC	
161614	161614	2015	11	56	5441	NaN	PC	
166714	166714	2015	11	36	5441	NaN	PC	
171814	171814	2015	11	37	5441	NaN	PC	
176914	176914	2015	11	35	5441	NaN	PC	
182014	182014	2015	11	38	5441	NaN	PC	
187114	187114	2015	11	34	5441	NaN	PC	
192214	192214	2015	11	46	5441	NaN	PC	
197314	197314	2015	11	41	5441	NaN	PC	
202414	202414	2015	11	44	5441	NaN	PC	
207514	207514	2015	11	39	5441	NaN	PC	
212614	212614	2015	11	45	5441	NaN	PC	

	sub_category	shop_name	shop_city	shop_type
3514	/			
8614	/			

```
13714       /                      13
18814       /
23914       /
29014       /
34114       /                    39
39214       /
44314       /                  II
49414       /
54514       /
59614       /
64714       /                     21
69814       /                   7
74914       /
80014       /                   XXI
85114       /
90214       /
95314       /                     II
100414      /
105514      /
110614      /
115714      /
120814      /                     2
125914      /
131014      /
136114      /
141214      /                      56
146314      /
151414      /
156514      /                   1
161614      /
166714      /
171814      /
176914      /
182014      /
187114      /
192214      /                     7
197314      /
202414      /
207514      /
212614      /
```

All remaining item_price's missing values have same main_category and sub_category. This main and sub categories are not in the test dataset, but in train dataset.

[308]:
```
#fill missing values with median of main_category and sub_category from train␣
↪dataset
```

```
filler = train[(train['main_category'] == 'PC') & (train['sub_category'] ==␣
 ↪'     /     ')]['item_price'].median()

test['item_price'].fillna(filler, inplace=True)
```

[309]:
```
test['item_price'].isnull().sum()
```

[309]:
```
0
```

4.0.9 Exploratory Data Analysis: Epilogue

- From competition's evaluation note, target values are clipped into [0,20] range.

[311]:
```
train['item_cnt_month'] = train['item_cnt_month'].clip(0,20)
```

- Define target_array

[312]:
```
target_array = train['item_cnt_month']
train.drop(['item_cnt_month'], axis=1, inplace=True)

test_id = test['ID']
test.drop(['ID'], axis=1, inplace=True)
```

- Drop shop_id & item_id

[313]:
```
train.drop(['shop_id','item_id'], axis=1, inplace=True)
test.drop(['shop_id','item_id'], axis=1, inplace=True)
```

- Reduce memory usage

[315]:
```
def downcast_dtypes(df):
    '''df (dataframe)  : data
       Changes column types in the dataframe
            `float64` type to `float32`
            `int64`   type to `int32`
    '''

    # Select columns to downcast
    float_cols = [c for c in df if df[c].dtype == "float64"]
    int_cols =   [c for c in df if df[c].dtype == "int64"]

    # Downcast
    df[float_cols] = df[float_cols].astype(np.float32)
    df[int_cols]   = df[int_cols].astype(np.int32)

    return df
```

[316]:
```
#reduce memory
downcast_dtypes(train)
downcast_dtypes(test)
```

[316]:
```
        year month  item_price main_category                sub_category  \
0       2015    11      749.25                                      PS3
1       2015    11      299.00             CD
2       2015    11     1199.00                                      PS3
3       2015    11      599.00                                  XBOX 360
4       2015    11     2299.00                                      PS4
…       …   …          …            …                          …
214195  2015    11      149.00             CD
214196  2015    11      999.00
214197  2015    11      199.00             CD
214198  2015    11      169.00                                      DVD
214199  2015    11      549.00                                   Blu-Ray

       shop_name shop_city shop_type
0
1
2
3
4
…           …        …        …
214195
214196
214197
214198
214199

[214200 rows x 8 columns]
```

[317]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1592067 entries, 0 to 1592066
Data columns (total 8 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
 0   year           1592067 non-null  object
 1   month          1592067 non-null  object
 2   item_price     1592067 non-null  float32
 3   main_category  1592067 non-null  object
 4   sub_category   1592067 non-null  object
 5   shop_name      1592067 non-null  object
 6   shop_city      1592067 non-null  object
 7   shop_type      1592067 non-null  object
```

```
dtypes: float32(1), object(7)
memory usage: 103.2+ MB
```

- Check for missing values

[319]:
```
#check for any missing data
print('missing data in the train dataset : ', train.isnull().any().sum())
print('missing data in the test dataset : ', test.isnull().any().sum())
```

```
missing data in the train dataset :  0
missing data in the test dataset :  0
```

- Normality test

[320]:
```
#define a normality test function
def normalityTest(data, alpha=0.05):
    """data (array)   : The array containing the sample to be tested.
           alpha (float)  : Significance level.
           return True if data is normal distributed"""

    from scipy import stats

    statistic, p_value = stats.normaltest(data)

    #null hypothesis: array comes from a normal distribution
    if p_value < alpha:
        #The null hypothesis can be rejected
        is_normal_dist = False
    else:
        #The null hypothesis cannot be rejected
        is_normal_dist = True

    return is_normal_dist
```

[321]:
```
#check normality of all numericaal features and transform it if not normal␣
 ↪distributed
for feature in train.columns:
    if (train[feature].dtype != 'object'):
        if normalityTest(train[feature]) == False:
            train[feature] = np.log1p(train[feature])
            test[feature] = np.log1p(test[feature])
```

[322]:
```
#use numpy.log1p in order to target_array follows a normal distribution
target_array = np.log1p(target_array)
```

- Encoding

[323]:
```
from sklearn.preprocessing import OrdinalEncoder
```

```
enc = OrdinalEncoder()

X = enc.fit_transform(train)
y = target_array

X_predict = enc.fit_transform(test)
```

5 Creating a model

We begin by splitting data into two subsets: for training data and for testing data.

[324]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .1,␣
↪random_state = 0)
```

We will use XGBRegressor model to predict total sales for every product and store in the next month.

[325]:
```
from xgboost import XGBRegressor

#create a model
model = XGBRegressor()

#fitting
model.fit(
    X_train,
    y_train,
    eval_metric="rmse",
    eval_set=[(X_train, y_train), (X_test, y_test)],
    verbose=True,
    early_stopping_rounds = 20)
```

```
[0]     validation_0-rmse:0.53285       validation_1-rmse:0.53070
Multiple eval metrics have been passed: 'validation_1-rmse' will be used for
early stopping.

Will train until validation_1-rmse hasn't improved in 20 rounds.
[1]     validation_0-rmse:0.47582       validation_1-rmse:0.47366
[2]     validation_0-rmse:0.44368       validation_1-rmse:0.44160
[3]     validation_0-rmse:0.42537       validation_1-rmse:0.42344
[4]     validation_0-rmse:0.41539       validation_1-rmse:0.41357
[5]     validation_0-rmse:0.41000       validation_1-rmse:0.40845
[6]     validation_0-rmse:0.40615       validation_1-rmse:0.40471
[7]     validation_0-rmse:0.40421       validation_1-rmse:0.40290
[8]     validation_0-rmse:0.40289       validation_1-rmse:0.40167
```

```
[9]     validation_0-rmse:0.40201       validation_1-rmse:0.40083
[10]    validation_0-rmse:0.40087       validation_1-rmse:0.39975
[11]    validation_0-rmse:0.40006       validation_1-rmse:0.39892
[12]    validation_0-rmse:0.39914       validation_1-rmse:0.39806
[13]    validation_0-rmse:0.39864       validation_1-rmse:0.39762
[14]    validation_0-rmse:0.39760       validation_1-rmse:0.39667
[15]    validation_0-rmse:0.39679       validation_1-rmse:0.39598
[16]    validation_0-rmse:0.39637       validation_1-rmse:0.39561
[17]    validation_0-rmse:0.39551       validation_1-rmse:0.39474
[18]    validation_0-rmse:0.39484       validation_1-rmse:0.39414
[19]    validation_0-rmse:0.39433       validation_1-rmse:0.39366
[20]    validation_0-rmse:0.39382       validation_1-rmse:0.39321
[21]    validation_0-rmse:0.39329       validation_1-rmse:0.39269
[22]    validation_0-rmse:0.39289       validation_1-rmse:0.39229
[23]    validation_0-rmse:0.39260       validation_1-rmse:0.39200
[24]    validation_0-rmse:0.39224       validation_1-rmse:0.39166
[25]    validation_0-rmse:0.39186       validation_1-rmse:0.39133
[26]    validation_0-rmse:0.39146       validation_1-rmse:0.39095
[27]    validation_0-rmse:0.39099       validation_1-rmse:0.39053
[28]    validation_0-rmse:0.39086       validation_1-rmse:0.39040
[29]    validation_0-rmse:0.39061       validation_1-rmse:0.39019
[30]    validation_0-rmse:0.39031       validation_1-rmse:0.38991
[31]    validation_0-rmse:0.38992       validation_1-rmse:0.38958
[32]    validation_0-rmse:0.38964       validation_1-rmse:0.38936
[33]    validation_0-rmse:0.38941       validation_1-rmse:0.38914
[34]    validation_0-rmse:0.38910       validation_1-rmse:0.38885
[35]    validation_0-rmse:0.38875       validation_1-rmse:0.38855
[36]    validation_0-rmse:0.38828       validation_1-rmse:0.38811
[37]    validation_0-rmse:0.38800       validation_1-rmse:0.38787
[38]    validation_0-rmse:0.38784       validation_1-rmse:0.38769
[39]    validation_0-rmse:0.38776       validation_1-rmse:0.38761
[40]    validation_0-rmse:0.38766       validation_1-rmse:0.38755
[41]    validation_0-rmse:0.38748       validation_1-rmse:0.38739
[42]    validation_0-rmse:0.38720       validation_1-rmse:0.38713
[43]    validation_0-rmse:0.38712       validation_1-rmse:0.38709
[44]    validation_0-rmse:0.38691       validation_1-rmse:0.38690
[45]    validation_0-rmse:0.38672       validation_1-rmse:0.38677
[46]    validation_0-rmse:0.38655       validation_1-rmse:0.38659
[47]    validation_0-rmse:0.38638       validation_1-rmse:0.38647
[48]    validation_0-rmse:0.38611       validation_1-rmse:0.38624
[49]    validation_0-rmse:0.38603       validation_1-rmse:0.38619
[50]    validation_0-rmse:0.38578       validation_1-rmse:0.38595
[51]    validation_0-rmse:0.38560       validation_1-rmse:0.38577
[52]    validation_0-rmse:0.38535       validation_1-rmse:0.38553
[53]    validation_0-rmse:0.38521       validation_1-rmse:0.38537
[54]    validation_0-rmse:0.38505       validation_1-rmse:0.38524
[55]    validation_0-rmse:0.38497       validation_1-rmse:0.38517
[56]    validation_0-rmse:0.38468       validation_1-rmse:0.38489
```

```
[57]    validation_0-rmse:0.38446       validation_1-rmse:0.38470
[58]    validation_0-rmse:0.38426       validation_1-rmse:0.38451
[59]    validation_0-rmse:0.38421       validation_1-rmse:0.38448
[60]    validation_0-rmse:0.38412       validation_1-rmse:0.38439
[61]    validation_0-rmse:0.38397       validation_1-rmse:0.38428
[62]    validation_0-rmse:0.38374       validation_1-rmse:0.38406
[63]    validation_0-rmse:0.38356       validation_1-rmse:0.38393
[64]    validation_0-rmse:0.38352       validation_1-rmse:0.38390
[65]    validation_0-rmse:0.38336       validation_1-rmse:0.38373
[66]    validation_0-rmse:0.38324       validation_1-rmse:0.38363
[67]    validation_0-rmse:0.38312       validation_1-rmse:0.38352
[68]    validation_0-rmse:0.38299       validation_1-rmse:0.38344
[69]    validation_0-rmse:0.38281       validation_1-rmse:0.38327
[70]    validation_0-rmse:0.38270       validation_1-rmse:0.38316
[71]    validation_0-rmse:0.38263       validation_1-rmse:0.38311
[72]    validation_0-rmse:0.38247       validation_1-rmse:0.38293
[73]    validation_0-rmse:0.38237       validation_1-rmse:0.38285
[74]    validation_0-rmse:0.38232       validation_1-rmse:0.38280
[75]    validation_0-rmse:0.38222       validation_1-rmse:0.38272
[76]    validation_0-rmse:0.38214       validation_1-rmse:0.38265
[77]    validation_0-rmse:0.38209       validation_1-rmse:0.38261
[78]    validation_0-rmse:0.38192       validation_1-rmse:0.38246
[79]    validation_0-rmse:0.38181       validation_1-rmse:0.38235
[80]    validation_0-rmse:0.38167       validation_1-rmse:0.38221
[81]    validation_0-rmse:0.38159       validation_1-rmse:0.38216
[82]    validation_0-rmse:0.38151       validation_1-rmse:0.38208
[83]    validation_0-rmse:0.38144       validation_1-rmse:0.38205
[84]    validation_0-rmse:0.38138       validation_1-rmse:0.38197
[85]    validation_0-rmse:0.38125       validation_1-rmse:0.38188
[86]    validation_0-rmse:0.38112       validation_1-rmse:0.38179
[87]    validation_0-rmse:0.38106       validation_1-rmse:0.38174
[88]    validation_0-rmse:0.38099       validation_1-rmse:0.38169
[89]    validation_0-rmse:0.38093       validation_1-rmse:0.38164
[90]    validation_0-rmse:0.38090       validation_1-rmse:0.38163
[91]    validation_0-rmse:0.38066       validation_1-rmse:0.38140
[92]    validation_0-rmse:0.38060       validation_1-rmse:0.38135
[93]    validation_0-rmse:0.38043       validation_1-rmse:0.38118
[94]    validation_0-rmse:0.38031       validation_1-rmse:0.38107
[95]    validation_0-rmse:0.38023       validation_1-rmse:0.38101
[96]    validation_0-rmse:0.38013       validation_1-rmse:0.38093
[97]    validation_0-rmse:0.38008       validation_1-rmse:0.38089
[98]    validation_0-rmse:0.38002       validation_1-rmse:0.38084
[99]    validation_0-rmse:0.37995       validation_1-rmse:0.38079
```

```
[325]: XGBRegressor(base_score=0.5, booster=None, colsample_bylevel=1,
                    colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
                    importance_type='gain', interaction_constraints=None,
```

```
learning_rate=0.300000012, max_delta_step=0, max_depth=6,
min_child_weight=1, missing=nan, monotone_constraints=None,
n_estimators=100, n_jobs=0, num_parallel_tree=1,
objective='reg:squarederror', random_state=0, reg_alpha=0,
reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
validate_parameters=False, verbosity=None)
```

[326]:
```
#calculate Mean Squared Error
from sklearn.metrics import mean_squared_error

print('MSE : ', mean_squared_error(y_test, model.predict(X_test)))
```

```
MSE :  0.14499894862846432
```

[327]:
```
#make a prediction
y_predict = model.predict(X_predict)

#transform the values back
y_predict = np.expm1(y_predict)
```

[328]:
```
#sava results to a file
results = pd.DataFrame({'ID': test_id, 'item_cnt_month': y_predict})
results.to_csv('my_submission.csv', index=False)
```

[ ]:
```

```