Pandas Basics

Hai Bacti

January 20, 2021

Version 1.0.3

Hi! In this programming assignment you need to refresh your `pandas` knowledge. You will need to do several `groupby`s and `join`'s to solve the task.

[1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
DATA_FOLDER = '../readonly/final_project_data/'

transactions    = pd.read_csv(os.path.join(DATA_FOLDER, 'sales_train.csv.gz'))
items           = pd.read_csv(os.path.join(DATA_FOLDER, 'items.csv'))
item_categories = pd.read_csv(os.path.join(DATA_FOLDER, 'item_categories.csv'))
shops           = pd.read_csv(os.path.join(DATA_FOLDER, 'shops.csv'))
```

The dataset we are going to use is taken from the competition, that serves as the final project for this course. You can find complete data description at the competition web page. To join the competition use this link.

Let's start with a simple task.

Print the shape of the loaded dataframes and use `df.head` function to print several rows. Examine the features you are given.

[3]:
```
transactions.index = pd.to_datetime(transactions.date, format = '%d.%m.%Y')
transactions['revenue'] = transactions.item_price * transactions.item_cnt_day
```

[4]:
```
transactions.shape
```

[4]:
```
(2935849, 7)
```

[5]:
```
transactions.head(n = 3)
```

[5]:
```
                 date  date_block_num  shop_id  item_id  item_price  \
date
2013-01-02  02.01.2013               0       59    22154       999.0
2013-01-03  03.01.2013               0       25     2552       899.0
```

```
2013-01-05  05.01.2013               0        25      2552        899.0
```

```
            item_cnt_day  revenue
date
2013-01-02           1.0    999.0
2013-01-03           1.0    899.0
2013-01-05          -1.0   -899.0
```

Now use your `pandas` skills to get answers for the following questions. The first question is:

What was the maximum total revenue among all the shops in September, 2014?

Hereinafter *revenue* refers to total sales minus value of goods returned.

Sometimes items are returned, find such examples in the dataset.

It is handy to split `date` field into [`day`, `month`, `year`] components and use `df.year == 14` and `df.month == 9` in order to select target subset of dates.

You may work with `date` feature as with strings, or you may first convert it to `pd.datetime` type with `pd.to_datetime` function, but do not forget to set correct `format` argument.

[6]:
```
dt = transactions
dt = dt[dt.index.year == 2014]
dt = dt[dt.index.month == 9]
dt = dt.groupby('shop_id').sum()

max_revenue = dt.iloc[dt.revenue.argmax()].revenue
max_revenue
```

[6]:
```
7982852.199999956
```

Great! Let's move on and answer another question:

What item category generated the highest revenue in summer 2014?

Submit `id` of the category found.

Here we call "summer" the period from June to August.

Note, that for an object `x` of type `pd.Series`: `x.argmax()` returns **index** of the maximum element. `pd.Series` can have non-trivial index (not `[1, 2, 3, ... ]`).

[7]:
```
dt = transactions
dt = dt[(dt.index.year == 2014) & (dt.index.month >= 6) & (dt.index.month <= 8)]
dt = dt.set_index('item_id').join(other = items)
dt = dt.groupby('item_category_id').sum()

category_id_with_max_revenue = dt.index[dt.revenue.argmax()]
category_id_with_max_revenue
```

[7]:
```
20
```

How many items are there, such that their price stays constant (to the best of our knowledge) during the whole period of time?

Let's assume, that the items are returned for the same price as they had been sold.

[8]:
```
dt = transactions
num_items_constant_price = sum(dt.groupby('item_id').item_price.nunique() == 1)
num_items_constant_price
```

[8]:
```
5926
```

Remember, the data can sometimes be noisy.

What was the variance of the number of sold items per day sequence for the shop with `shop_id = 25` in December, 2014? Do not count the items, that were sold but returned back later.

Fill `total_num_items_sold` and `days` arrays, and plot the sequence with the code below.

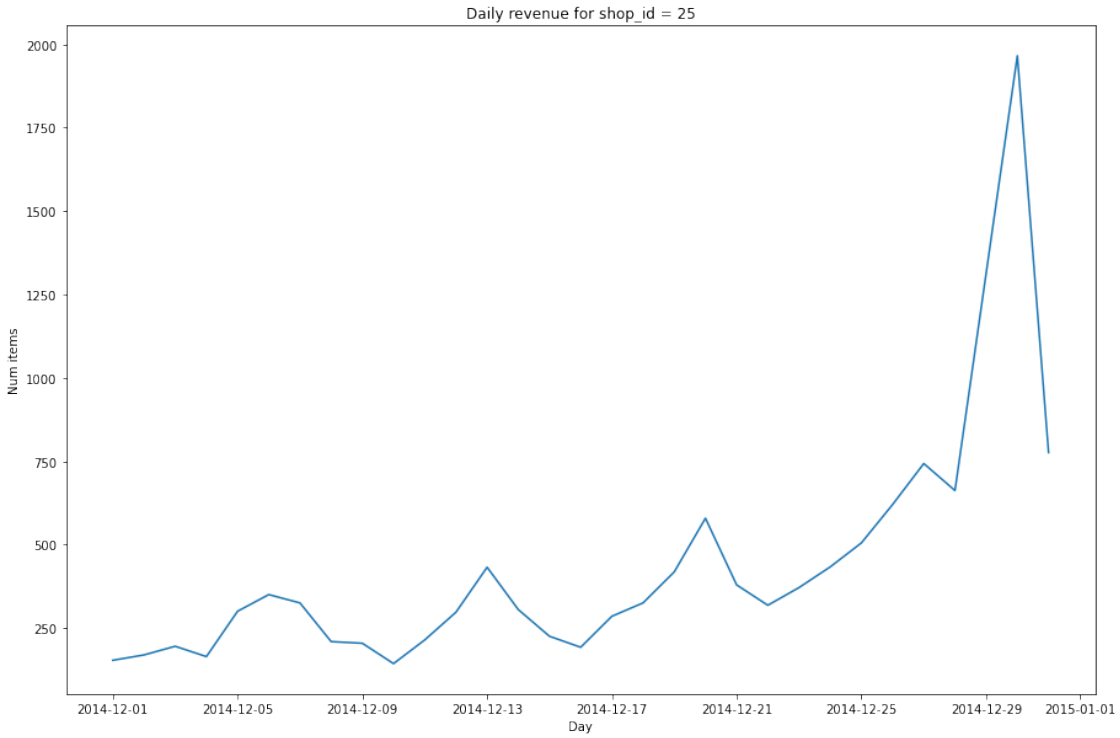
Then compute variance. Remember, there can be differences in how you normalize variance (biased or unbiased estimate, see link). Compute ***unbiased*** estimate (use the right value for `ddof` argument in `pd.var` or `np.var`).

If there were no sales at a given day, ***do not*** impute missing value with zero, just ignore that day

[9]:
```
dt = transactions
dt = dt[(dt.shop_id == 25) & (dt.index.year == 2014) & (dt.index.month == 12)]
dt = dt.groupby(dt.index).sum().item_cnt_day
days, total_num_items_sold = dt.index, dt

# Plot it
plt.figure(figsize = (15, 10))
plt.plot(days, total_num_items_sold)
plt.ylabel('Num items')
plt.xlabel('Day')
plt.title("Daily revenue for shop_id = 25")
plt.show()

total_num_items_sold_var = total_num_items_sold.var() # pandas uses 1/n-1 by␣
↪default, numpy uses 1/n
total_num_items_sold_var
```



[9]: 117167.70229885059

Well done! :)