# Product Analyst Relevance – Case Study

The present work is divided in three parts

- Part 1 – Dataset Exploration
- Part 2 – Data preprocessing
- Part 3 – Product recommendation model

Where we show how we achieve the final result (along plots and codes), the assumptions used, why it was selected and future work.

## Part 1 – Dataset Exploration

*Item_data.csv*

| Name | date_live_nk | category_sk | Listing_sk | User_sk encrypted | Last_reply_date | First_reply date |
|---|---|---|---|---|---|---|
| dtype | object | object | Object | object | object | Object |
| Unique | 81 | 54 | 2000 | 1935 | 8 | 8 |
| Nan | 0 | 0 | 0 | 0 | 0 | 0 |
| Top | 2017-09-19 | olx\|mea\|za\|362\|378 | 1051110968 | d36caa07b89dd70878ff87e35a8835aa | 2017-09-21 | 2017-09-14 |
| freq | 126 | 536 | 1 | 3 | 379 | 335 |

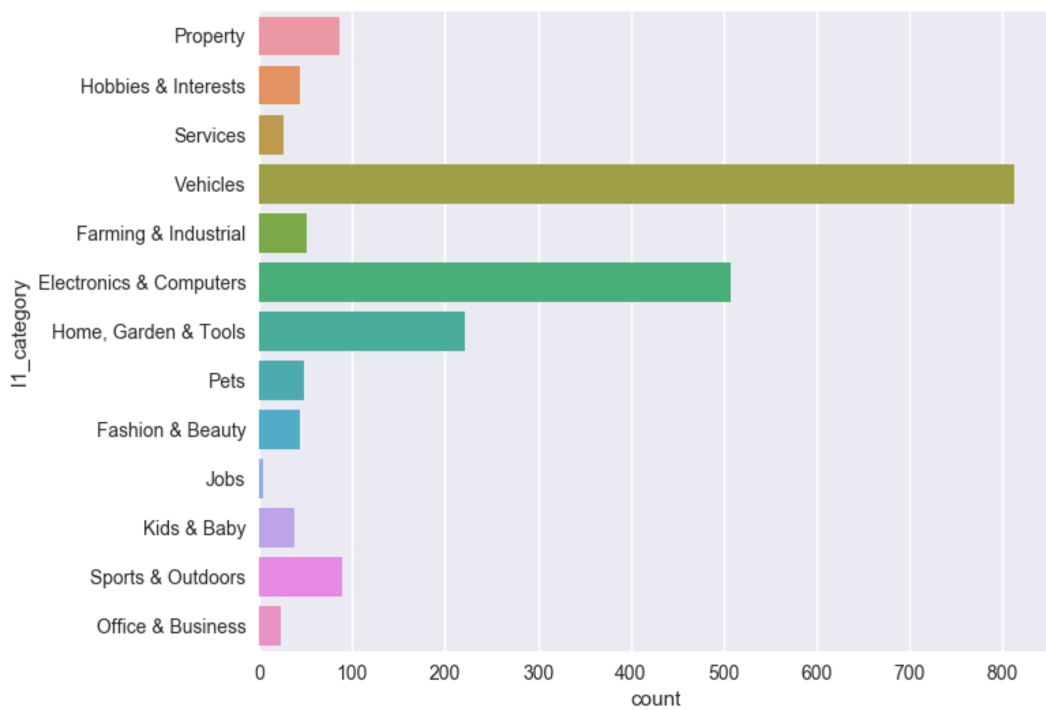
| Name | Successful_replies | Average_conversation length | replies |
|---|---|---|---|
| dtype | Float64 | Float64 | Float64 |
| Mean | 0.981 | 2.522 | 1.917 |
| Std | 1.639 | 3.874 | 2.4511 |
| Max | 21 | 81 | 41 |
| Min | 0 | 1 | 1 |
| Nan | 0 | 0 | 0 |

*category_data.csv*

Non NaN values found

| Category l1 code | category_l1_name_en | subcategories |
|---|---|---|
| 16 | Property | 388,368,367,363,301 |
| 185 | Hobbies & Interests | 911,820,243,214,211 |
| 191 | Services | 647,633,372,324,207,198 |
| 362 | Vehicles | 417,416,379,378,376,377 |
| 5170 | Office & Business | 5171,5172,5173,5174 |
| 600 | Farming & Industrial | 887,604,603,602 |
| 800 | Electronics & Computers | 912,870,805,804,803,802,801 |
| 806 | Home, Garden & Tools | 910,809,807,808 |
| 811 | Pets | 814,813,812 |
| 815 | Fashion & Beauty | 819,817,816 |
| 821 | Jobs | 823,822 |
| 853 | Kids & Baby | 856,855 |
| 881 | Sports & Outdoors | 889,883,882 |

Categories with more items:



## Part 2 – Data preprocessing

1- Date variables have been transformed to be able of handle them

```
item['first_reply_date'] = pd.to_datetime(item['first_reply_date'])
item['last_reply_date'] = pd.to_datetime(item['last_reply_date'])
item['date_live_nk'] = pd.to_datetime(item['date_live_nk'])
```

2- We desegregate the "category_sk" variable to get the variable in other columns and in numeric values

```
categories_item = item['category_sk'].str.split('|',expand = True)
item = pd.concat([item,categories_item[3],categories_item[4]], axis = 1)
item = item.rename(columns = {3:'l1_category',4:'l2_category'})
item['l1_category'] = pd.to_numeric(item['l1_category'])
item['l2_category'] = pd.to_numeric(item['l2_category'])
```

3- The category table is split in two datasets in order to have the categories and subcategories (called l1_category and l2_category) when it is necessary.

```
l1_category = category['category_sk'].str.split('|',expand = True)[3]
l1_category = pd.to_numeric(l1_category)
l1_category = pd.concat([l1_category, category['category_l1_name_en']], axis = 1).drop_duplicates([3], keep = 'last')
l1_category = l1_category.sort_values([3])

l2_category = category['category_sk'].str.split('|',expand = True)[4].fillna(0)
l2_category = pd.to_numeric(l2_category)
l2_category = pd.concat([l2_category, category['category_l2_name_en']], axis = 1).drop_duplicates([4], keep = 'last')
```

# Part 3 – Product recommendation model

**We are going to define an advertisement ranking based in the some indicators within the category of the reply. Once the potential BUYER makes a reply we offer the best advertisement.**

**Lemmas of the strategy**

- Once the BUYER made a reply, we already know which product the BUYER wants.
- A good SELLER improves the chances of closing a deal; we need reliable and better sellers.

**Metric: CTR, we are going to measure if more people "click" the recommendations and then make a "reply" in one of the recommendations (funnel analysis).**

**Assumptions:**

- We suppose that the data which was given is the total data we can have.
- The dataset is from the SELLER's point of view, we are going to reference BUYER's data (history, past buyers, similarities, etc.) as an outline work and not included in the present project. So the data is a proxy to buyer's chances of closing a deal.
- If the seller doesn't answer the deal won't close, so we reward sellers who answer message and with a good relevance.
- Reward good items (sellers), minimize the risk of disappointment.

**Step 1: Indicators selection**

1. Item antiquity
2. Conversation relevance
3. Rate of replies given by the seller
4. Number of items of the seller
5. Intensity of the item's replies
6. Category and subcategory

**Step 2: Indicators definition and exploratory analysis**

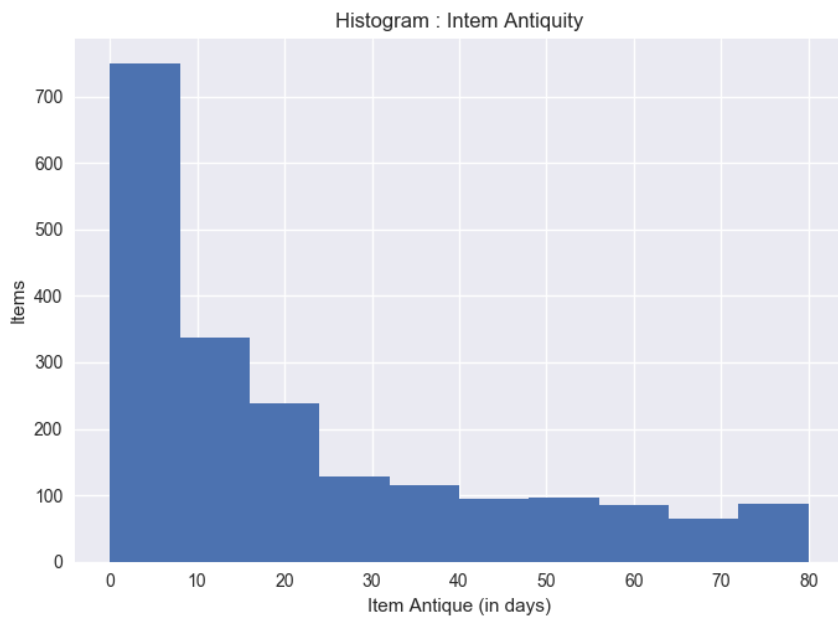
1. Item antiquity

$$ia = number\ of\ days\ since\ was\ posted$$

*Mean = 21.956*
*StdDev = 22.013*

We suppose that newest items are more probably to be sold because the seller recent activity and in order to avoid "forgotten items".

```
x1 = pd.to_datetime(item['date_live_nk'])
x2 =  pd.to_datetime(item['last_reply_date'][3]) #Last day of analysis 21-09-2017
ia = (x2 - x1).dt.days
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ia.plot(kind = 'hist')
ax2.set_xlabel('Item Antique (in days)')
ax2.set_ylabel('Items')
ax2.set_title('Histogram : Intem Antiquity')
```



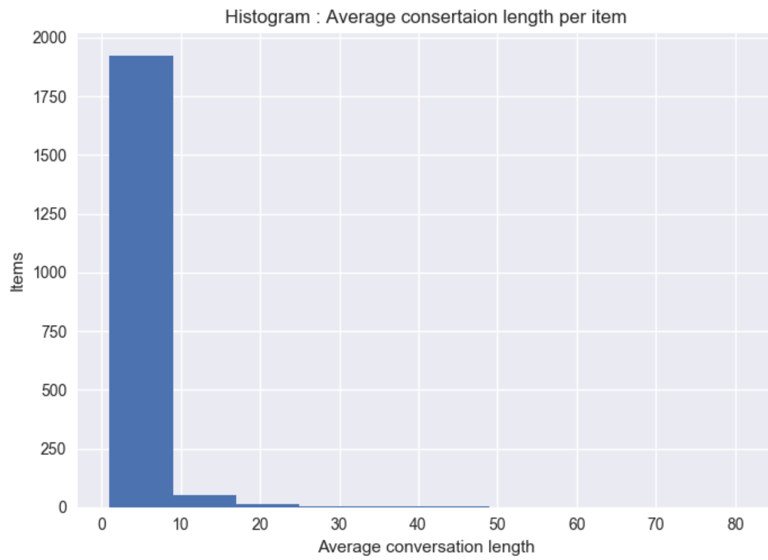

2. Conversation relevance

$$cr = average\ conversation\ length$$

*Mean = 2.522*
*StdDev = 3.87*

We suppose that a greater average of messages refers to a more active seller which is more willing to sell as being more descriptive. Also, it could be that lot of successful transactions are made with few messages.

```
fig4 = plt.figure()
ax4 = fig4.add_subplot(111)
item['average_conversation_length'].plot(kind = 'hist')
ax4.set_xlabel('Average conversation length')
ax4.set_ylabel('Items')
ax4.set_title('Histogram : Average consertaion length per item')
```



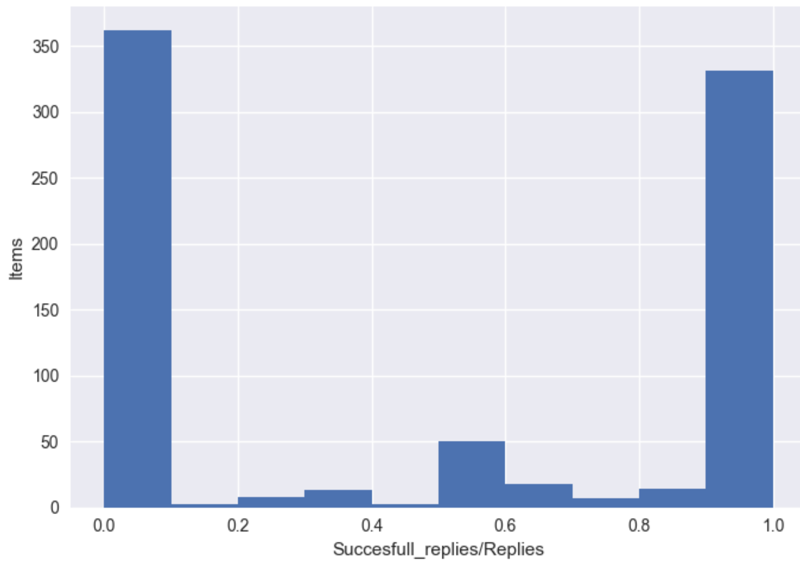

3. Rate of replies given by the seller

$$p = \frac{successful\ replies}{replies}$$

The rate is applied only if first *reply < current date (17/09/2017)*, given at least four days to the seller to respond. If still the seller doesn't respond, he will have a *p = 0*, which acts as a filter not considering the item.

```
item2 = item[item.first_reply_date < '2017-09-17'] #date considerer to give a margin to the seller
p = item2['successful_replies'] / item2['replies']

fig = plt.figure()
ax1 = fig.add_subplot(111)
p.plot(kind = 'hist')
ax1.set_xlabel('Succesfull_replies/Replies')
ax1.set_ylabel('Items')
#plt.yticks(y,item['listing_sk']
```



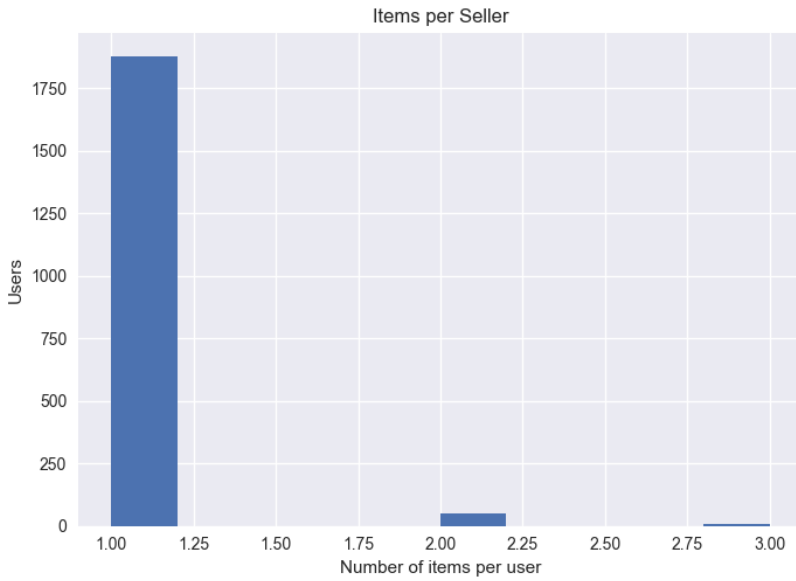

4. Number of items of the seller

$$is = number\ of\ items\ of\ the\ seller$$

*Mean = 1.033*

Most of them seem to be "casual sellers" (97%) but we decided to give a favor for those who sell more items in order to be more dedicated to the business.

```
is1 = item['user_sk_encrypted'].value_counts()
fig3 = plt.figure()
ax3 = fig3.add_subplot(111)
item['user_sk_encrypted'].value_counts().plot(kind = 'hist')
ax3.set_xlabel('Number of items per user')
ax3.set_ylabel('Items')
ax3.set_title('Items per Seller')
```



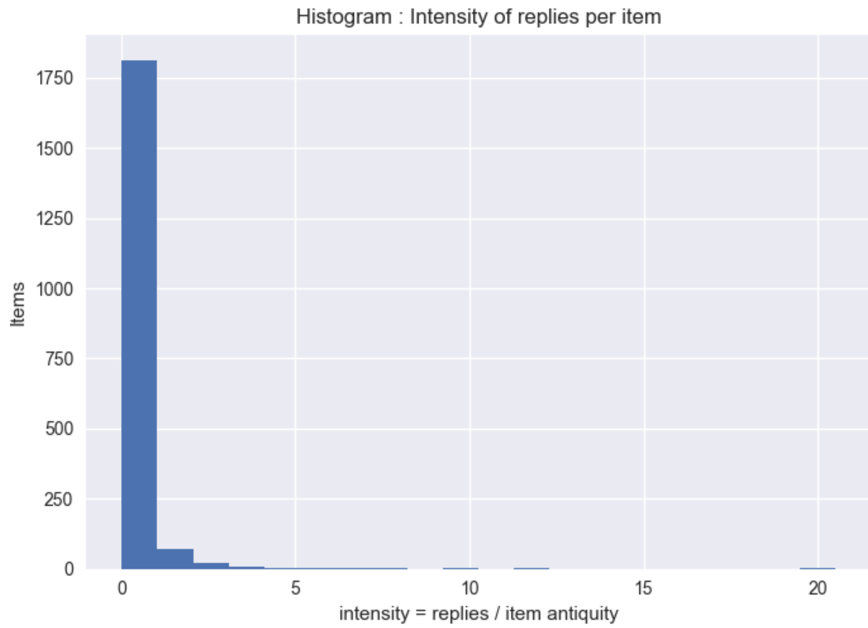

5. Intensity of the item's replies

For all items with more than one day since it was publish we take de intensity of replies

$$i = \frac{replies}{item\ antiquity}$$

With this factor we want to favor those articles which have a strong attention over the time

```
i = item[item.date_live_nk < '2017-09-21']['replies'] / ia[ia > 0]
fig5 = plt.figure()
ax5 = fig5.add_subplot(111)
i.plot(kind = 'hist', bins = 20)
ax5.set_xlabel('intensity = replies / item antiquity')
ax5.set_ylabel('Items')
ax5.set_title('Histogram : Intensity of replies per item')
```




6. Category and subcategory

We have found that 56 l2_categories that belongs to 13 l1_categories. The l2_category will be the filter to take the items and make the ranking.

## Step 3: Scoring function and scoring process definition

1. Simple multiplicative weighting

$$\boldsymbol{score = ia^{w1} * cr^{w2} * p^{w3} * is^{w4} * i^{w5} * category_1 * category_2}$$

- The non-ratio values are normalized centered to the mean.
- If the item is in the same categories => category1 = category2 = 1. If category2 is different it will take a value less than one.
- We can modify the weights in order to give more importance to certain variables, for example the rate of replies.
- We have to take some considerations about dates, for example in indicator p, to give the seller some days to answer the reply.

**Why?**

- The approach is simple, can be launch easily and used as benchmark of more complex approaches.
- An approach that it still found in sites as “Popular News”, “Most read articles”, “Amazon best sellers”.
- The approach can serve as a foundation of more complex systems.

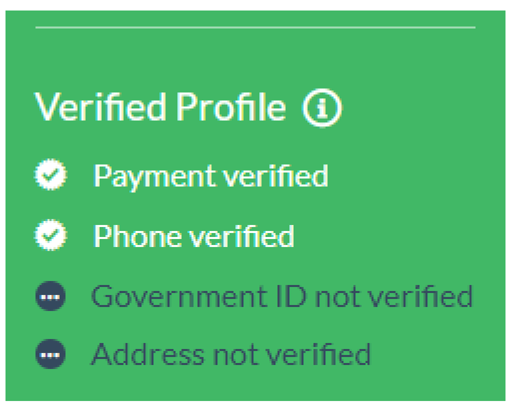

*Amazon best sellers*

**Weaknesses:**

- The method lack of any personalization per client given that the ranking activation occurs with a reply in a subcategory.
- It only works once the buyer makes a reply since it means (under the assumption) we know what he wants.
- Data handling: we rank over the entire number of items in a subcategory which we need to evaluate the computational time. It could be easily solved by adding other filters to discard items that we already know that are in the bottom of the rank.

**Outlook:**

- Adding more information as user information, this model can feed, for example, a SVM rank model as a training set for the algorithm, or a binary classifier to discard bad items.
- The key of the strategy is to improve the reliability of the SELLER, with this purpose a “couchsourfing approach” can be taken to have more genuine items.
- We can warn sellers of new messages.




*Couchsurfer’s verified profile example*