eda

June 1, 2020

[1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
clrs = ['#025159', '#03A696', '#F28705', '#F25D27', '#F20505']
```

0.1 # Explorative Data Analysis for Challenge FS20C7 - Recommendations im Detailhandel

Author Roman Studer

This notebook is dedicated to the explorative data analysis of the data set used in the Challenge FS20C7 - Recommendations in Retail. Since the execution of the calculations to create the graphics in this notebook is rather long, a PDF and HTML version is available here. Please note that in the PDF the interactivity of the graphics is lost.

0.2 1. Dataset

Import and first look at Dataset

[ ]:
```
df = pd.read_csv('Recommender4Retail.csv', index_col='Unnamed: 0');
```

[3]:
```
df.head()
```

[3]:
```
   order_id  user_id eval_set  order_number  order_dow  order_hour_of_day  \
1   2539329        1    prior             1          2                  8
2   2539329        1    prior             1          2                  8
3   2539329        1    prior             1          2                  8
4   2539329        1    prior             1          2                  8
5   2539329        1    prior             1          2                  8

   days_since_prior_order  product_id  add_to_cart_order  reordered  \
1                     NaN         196                  1          0
2                     NaN       14084                  2          0
3                     NaN       12427                  3          0
4                     NaN       26088                  4          0
5                     NaN       26405                  5          0
```

```
                           product_name  aisle_id  department_id  \
1                                  Soda        77              7
2  Organic Unsweetened Vanilla Almond Milk        91             16
3                   Original Beef Jerky        23             19
4                Aged White Cheddar Popcorn        23             19
5          XL Pick-A-Size Paper Towel Rolls        54             17

   department             aisle
1   beverages       soft drinks
2  dairy eggs   soy lactosefree
3      snacks    popcorn jerky
4      snacks    popcorn jerky
5   household       paper goods
```

[4]:
```
df.tail()
```

[4]:
```
          order_id  user_id eval_set  order_number  order_dow  \
33819102    272231   206209    train            14          6
33819103    272231   206209    train            14          6
33819104    272231   206209    train            14          6
33819105    272231   206209    train            14          6
33819106    272231   206209    train            14          6

          order_hour_of_day  days_since_prior_order  product_id  \
33819102                 14                    30.0       40603
33819103                 14                    30.0       15655
33819104                 14                    30.0       42606
33819105                 14                    30.0       37966
33819106                 14                    30.0       39216

          add_to_cart_order  reordered  \
33819102                  4          0
33819103                  5          0
33819104                  6          0
33819105                  7          0
33819106                  8          1

                                     product_name  aisle_id  department_id  \
33819102                   Fabric Softener Sheets        75             17
33819103  Dark Chocolate Mint Snacking Chocolate        45             19
33819104                  Phish Food Frozen Yogurt        37              1
33819105                     French Baguette Bread       112              3
33819106    Original Multigrain Spoonfuls Cereal       121             14

         department             aisle
33819102  household           laundry
33819103     snacks   candy chocolate
```

```
33819104     frozen    ice cream ice
33819105     bakery            bread
33819106  breakfast           cereal
```

[5]:
```
df.columns
```

[5]:
```
Index(['order_id', 'user_id', 'eval_set', 'order_number', 'order_dow',
       'order_hour_of_day', 'days_since_prior_order', 'product_id',
       'add_to_cart_order', 'reordered', 'product_name', 'aisle_id',
       'department_id', 'department', 'aisle'],
      dtype='object')
```

0.3 2. Data description

The following columns are included in the dataset: - order_id: order identifier - user_id: customer identifier - eval_set: which evaluation set this order belongs in (see SET described below) - order_number: the order sequence number for this user (1 = first, n = nth) - order_dow: the day of the week the order was placed on - order_hour_of_day: the hour of the day the order was placed on - days_since_prior: days since the last order, capped at 30 (with NAs for order_number = 1) - product_id: product identifier - product_name: name of the product - aisle_id: aisle identifier - aisle: the name of the aisle - department_id: department identifier - department: the name of the department - reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

Where SET is one of the four following evaluation sets (eval_set in orders):

- "prior": orders prior to that users most recent order
- "train": training data supplied to participants
- "test": test data reserved for machine learning competitions

[6]:
```
# shape
f'Dimensionality of the dataset {df.shape}'
```

[6]:
```
'Dimensionality of the dataset (33819106, 15)'
```

0.4 3. Data cleaning

Dropping columns and deal with missing data

0.4.1 3.1. Dropping columns

The following columns are not included in the analysis: - eval_set: for the further use of the dataset a categorization in "prior", "train" or "test" of the data is not necessary. - order_number: the order of orders per user is not included. This information can also be read from the 'order_id'. - add_to_cart_order: the order of how a product has been placed is not relevant for the planned recommender - order_dow and order_hour_of_day: time of order is irrelevant for the planned recommender - days_since_prior_order, also does not flow into the planned recommender.

[7]:

```
df = df.
↪drop(['eval_set','order_number','add_to_cart_order','order_dow','order_hour_of_day',␣
↪'days_since_prior_order'], axis=1)
```

0.4.2 3.2. Data types

[8]:
```
# checking datatypes
df.dtypes
```

[8]:
```
order_id           int64
user_id            int64
product_id         int64
reordered          int64
product_name      object
aisle_id           int64
department_id      int64
department        object
aisle             object
dtype: object
```

Data types per feature are correct.

0.4.3 3.3. Missing values

Rows with missing values can no longer be used. Since these are individual transactions that at first glance do not appear to have any connection (apart from possible customer preferences) we do not impute missing data.

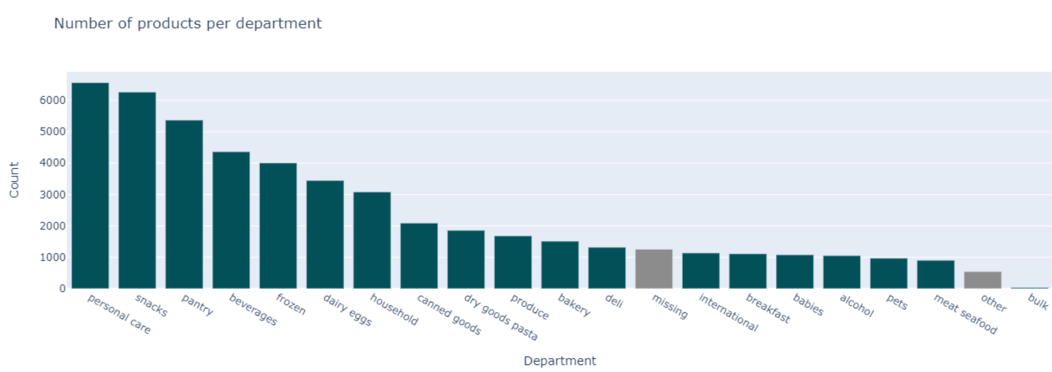
[9]:
```
df.drop_duplicates(inplace=True)
df.isna().sum()
```

[9]:
```
order_id         0
user_id          0
product_id       0
reordered        0
product_name     0
aisle_id         0
department_id    0
department       0
aisle            0
dtype: int64
```

0.5 4. Products

Visual analysis of the distribution of products in the dataset.

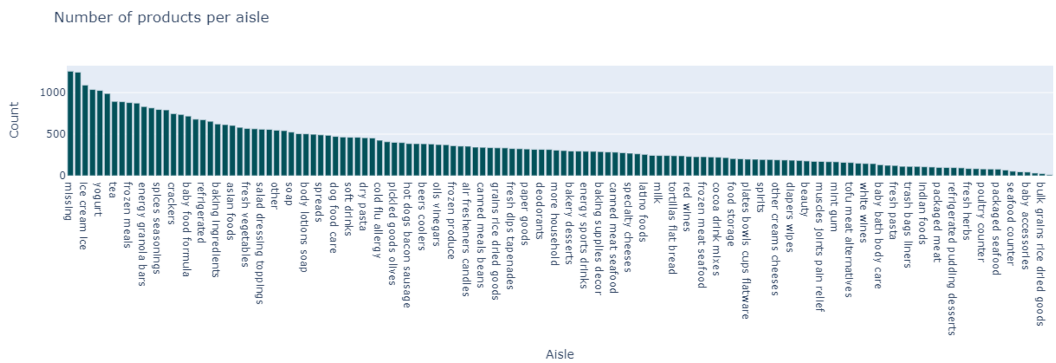
[10]:
```
df_department = df.groupby(by='department')['product_id'].nunique().
↪reset_index()

color = [clrs[0],]*len(df_department.department.unique())
color[14], color[15] = '#8c8c8c', '#8c8c8c'
fig = go.Figure(data=[go.Bar(
    x=df_department.department,
    y=df_department.product_id,
    marker_color=color, # marker color can be a single color value or an␣
↪iterable

)]).update_xaxes(categoryorder='total descending')
fig.update_layout(title_text='Number of products per department',
                  xaxis_title="Department",
                 yaxis_title="Count")

fig.show()
```



The products are divided into 19 departments. In the graphic above, the departments Pantry, Personal Care, Snacks, Beverages and Frozen are particularly prominent. It can also be seen that there are products that cannot be assigned to a specific department. These 548 products fall into the category 'other'. A larger proportion (1258 products) cannot even be assigned to the category 'other' and were given the designation 'missing'. missing' and 'other' are greyed out in the graphic.

[11]:
```
# products per aisle
df_aisle = df.groupby('aisle')['product_id'].nunique().reset_index()

fig = px.bar(df_aisle,
            x='aisle',
            y='product_id',
            title='Number of products per aisle',
            labels = {'product_id':'Count', 'aisle':'Aisle'},
```

```
              color_discrete_sequence = [clrs[0]]).
↪update_xaxes(categoryorder='total descending')
fig.show()
```



The individual aisles of the departments, listed by number of products, show a range from 1258 up to 12 products per aisle.

[12]:
```
df_dep_aisle = df.groupby(by=['department','aisle'])['product_id'].nunique().
↪reset_index()
fig = px.treemap(df_dep_aisle,
                 path=['department','aisle'],
                 values='product_id',
                 title='Ratio of orders in departments and aisles',
                 color='department')
fig.show()
```

Product groups (Aisles) in the departments can be visualized well in a treemap. It is well visible that the top 5 departments contain slightly more than half of the available product range. To focus on a department, click on it.

0.5.1 Most ordered products

[13]:
```
# most ordered products
most_ordered = df['product_name'].value_counts()
# top 20 products
print(f'Top 20 Products: \n{most_ordered[:20]}')
```

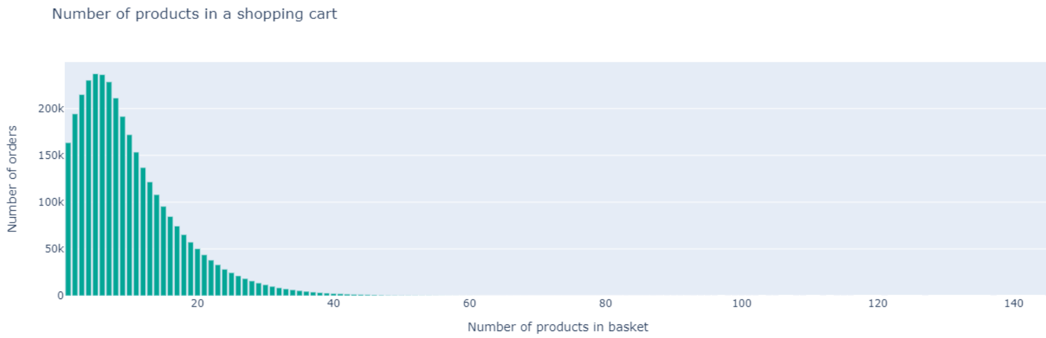
```
Top 20 Products:
Banana                        491291
Bag of Organic Bananas        394930
Organic Strawberries          275577
Organic Baby Spinach          251705
Organic Hass Avocado          220877
Organic Avocado               184224
Large Lemon                   160792
Strawberries                  149445
Limes                         146660
Organic Whole Milk            142813
Organic Raspberries           142603
Organic Yellow Onion          117716
Organic Garlic                113936
Organic Zucchini              109412
Organic Blueberries           105026
Cucumber Kirby                 99728
Organic Fuji Apple             92889
Organic Lemon                  91251
Organic Grape Tomatoes         88078
Apple Honeycrisp Organic       87272
Name: product_name, dtype: int64
```

At first glance, the top products appear to be organic products such as fruits and vegetables. At the very top, the banana.

0.6 Orders

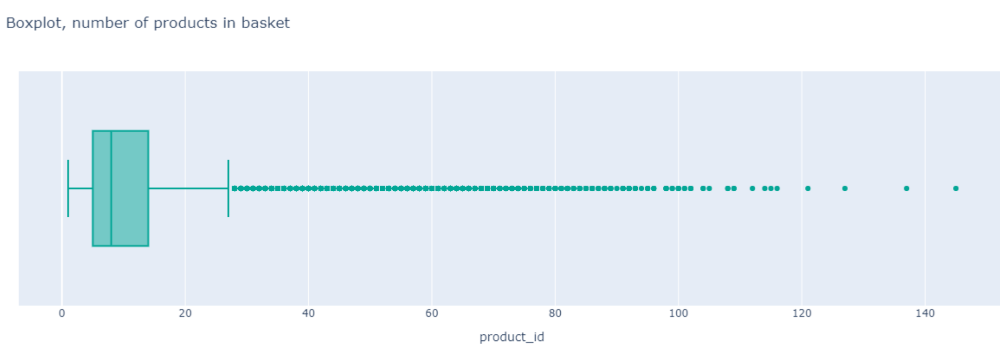
Visual analysis of the distribution of orders in the dataset.

[14]:
```
# Number of products in orders
df_n_order = df.groupby(by=['order_id'])['product_id'].count().reset_index()
n_orders = df_n_order.product_id.value_counts()

fig = px.bar(x=n_orders.index,
             y=n_orders.values,
             title='Number of products in a shopping cart',
             labels={'x':'Number of products in basket','y':'Number of orders'},
             color_discrete_sequence=[clrs[1]])

fig.show()
```



The upper graphic shows how often a shopping cart contains a certain number of products. You can see here that there are orders with more than 140 products. The vast majority, however, have between one and twenty products in their shopping cart. The peak lies at five products with over 237 thousand orders containing this number of products.

[15]:
```
# orders per customer
df_c_order = df.groupby(by=['order_id'])['product_id'].count().reset_index()
fig = px.box(data_frame=df_c_order,
             x='product_id',
             title='Boxplot, number of products in basket',
             labels={'x':'Number of products in order'},
             color_discrete_sequence=[clrs[1]],
             orientation='h')

fig.show()
```

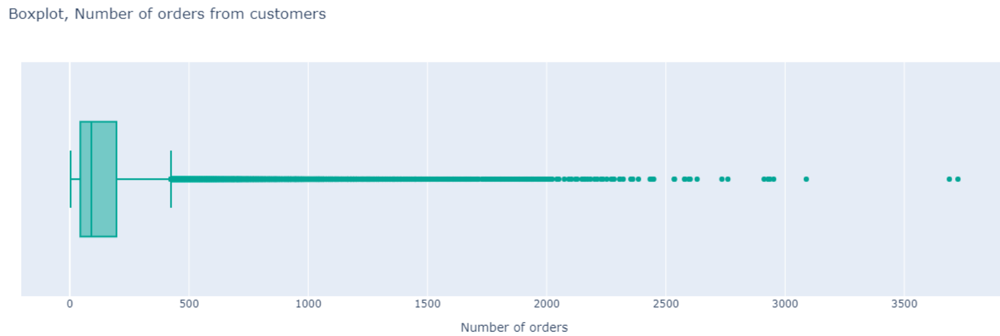


The box plot confirms the image of the upper bar plot. The minimum size of an order is one, since a smaller number would not trigger a transaction. The maximum size of an order is 145 and is an exception. The median is eight products, with 50% of orders containing between 5 and 14 products.

[16]:
```
# number of orders per customer
df_n_customer = df.groupby(by='user_id')['order_id'].count().reset_index()

fig = px.histogram(data_frame=df_n_customer,
                   x="order_id",
                   title='Number of customers with x orders',
                   labels={'order_id':'Number of orders','count':'Amount of␣
 ↪users'},
                   color_discrete_sequence=[clrs[1]],
                   nbins=150
                  )

fig.update_layout(yaxis_title="Amount of users")

fig.show()
```

The distribution of the number of orders has shifted strongly to the left. The histogram therefore moves to the lower left limit. The graphic shows how many customers (y-axis) carry out a certain number of orders (x-axis). If we look at the bar on the far left, we can see that almost 60 thousand customers placed between 1 and 50 orders. On the far right of the plot, barely visible, the highrollers with over 3500 orders.

[17]:
```
# number orders per customer
fig = px.box(data_frame=df_n_customer,
             x='order_id',
             title='Boxplot, Number of orders from customers',
             labels={'order_id':'Number of orders'},
             color_discrete_sequence=[clrs[1]],
             orientation='h')

fig.show()
```



The box plot to the upper histogram shows that 50% of the users place between 44 and 196 orders. The median is at 90.

1 Conclusion

The insights gained from this brief analysis are incorporated into the recommender system. Above all, the consequences of various types of product reduction are well illustrated here. For example, if I remove aisles with few products, I reduce the number of products only slightly, but lose a large part of the product variety. It is also interesting to see that the median number of orders per user is 90.