# EDA on E-commerce Behaviour Data From Cosmetics Online Store

Ahmed Shalaby Ahmed Mohamed

AyaTAllah Sabry Mohamed Abd-ElRahman

Mohamed Sulaiman Mohamed Mohamed

Mohamed Tarek Mahmoud Elsaid

Supervised by :

Eng. Rana Salah

Eng.Khaled Ellithy

Mar 24, 2023

# Table of contents

# Abstract

In the information age, consumers have access to virtually infinite possibilities. Knowing customer preferences and offering personalized recommendations are critical components of improving the user experience.Correct recommendations would increase customer loyalty and sales on the platforms.

Using datasets from a “medium” cosmetics online store hosted by the REES46 niche-specific personalization engine platform, we intend to create an association rule between brand and user as well as two recommender systems: one that recommends products to customers and the other that promotes brands. The retail platform, for example, might send emails advocating specific products or offering unique coupon codes for specific brands.

One of the most widely used models for recommender systems is collaborative filtering. predicts a person's choice.  According to user preference data collected from other users, users like similar products, and products attract individuals with related interests.

The three fundamental techniques for collaborative filtering are matrix factorization, model-based, and memory-based (neighborhood-based) methods.The memory-based method uses user-item data to determine how similar two persons or items are.  On the other hand, matrix factorization is one of the state-of-the-art solutions for collaborative filtering. It uses latent factors to decompose the user-item interaction data.

## • Business challenge

- There are various techniques to create recommendation systems for rating-based data, such as movies and songs, if we search online. Rating-based models have the drawback of being difficult to standardize for data with non-scaled target values, such as purchase or frequency data.
- For instance, ratings for music and movies are typically given on a scale of 0–5 or 0–10.
- **Purchase** data, however, is **continuous** and has **no upper bound.**
- Unfortunately, a lot of online resources present outcomes without analyzing their models.
- When we are  dealing with millions of data, this is a risky area for the majority of data scientists and engineers! Without any review, results alone won't advance our tools for industries.

## • Analysis Scope and Insights

- **Merging** the **2 data files** ( 8738120 rows × 9 columns) **Removing** the **Duplicates** ( 8278272 rows × 9 columns)  459,848 duplicated rows
- **Event Time** Min Date: 2019-10-01 Max Date : 2019-11-30
- **Event Type** :View - Purchase – Remove_from_Cart - Cart
- **Product_ID:** 45960 Unique Product_ID
- **Category_ID :**500 Unique Category_ID
- **Category_Code**:11 Unique Category_Code
- **Brand**:244 Unique Brands
- **Price**: Including the price of the items
- **User ID** :713100 Unique User_ID
- **User_Session**:1814075 Unique User_Session

# Data Preparation and EDA

1- **Event_time**
   - Changed it to date type
   - Extracting the month – week – day – hour

2- **Event_type**
   - Number of views 3938043
   - Number of Cart 2493850
   - Number of remove_from_cart 1278829
   - Number of purchases 567550

3- **Product_ID**
   - Changed its type to str. I don't want to make any calculations .

4- **Category_id**... > changed its type to str. I don't want to make any calculations.

5- **Category_Code**
   - Including 98% as null values ,We Decided to remove it . (98% nulls – its values not related to the Cosmetics such as furniture, Air condition and vacuums)

6- **User_ID** changed its type to str. I don't want to make any calculations.

7- **User_Session** We have 1329 null rows so we removed them

8- **Price**
   - There were ( 17517 of negative and zero rows ) so we decided to remove them .
   - Items with negative and zero prices are excluded since we can't promote them to customers because they are likely free gifts or transient items.

9- **Brand**
   - It has about 41% as null values ( 3451273 null rows ) .It has 244 brands
   - We decided to fill the null values according to the product _id , we filled about 8959 rows only ..
   - So we filled the rest of null values with FBX aka Fulfilled by X ( X referring to the website name )
   - Because every ecommerce store is divided into 2 sections ( Marketplace for large corporates and small sellers And the website itself as a seller in the website).

## More details about handling  our missing values

- These are some common methods for handling missing values, but there are also other methods such as hot deck imputation, cold deck imputation, and others. The choice of method should be based on the **specific characteristics** of the data and the analysis goals.

- The dataset of "Ecommerce Behavior" had missing values in some columns  as in the following figure:

```
df.isna().sum()/len(df)
```

```
event_time       0.000000
event_type       0.000000
product_id       0.000000
category_id      0.000000
category_code    0.983551
brand            0.404473
price            0.000000
user_id          0.000000
user_session     0.000155
dtype: float64
```

- The column of category_code the percentage of lost data **is 98%**, it is difficult to replace the lost data. So the best solution is to delete this column.
- The column for user_session the percentage of data lost is very small, not exceeding 1%. The best solution is to delete the missing part of the data in this column

- In the column of brand, there is **41% missing data,** and here the imputation was several ways:
- The **first method** was random choice, and the imputation was more than the most purchased brands, while the rest of the brands were not affected by this imputation.
- The **second method** of imputation is theories such as ('backfill', 'bfill', 'pad', 'ffill'), but the result was largely **unsuccessful** due to the bias of most of the existing brands.
- The **third method:** is the use of the theory of machine learning, which is the (KNN:K-Nearest Neighbors) But after searching a lot, we found that we replace the lost data with another brand because it will certainly have an **error rate,** and this is not good in making a decision regarding business, and the proof of this is that we found that 21404 thousand product_id, they do not have a brand. While we found 24556 thousand product_id, they have a brand. This made us search for this topic more so **we searched on websites such as Amazon or Alibaba or Jumia and found that the percentage of sales on these sites is not the famous brands but the marketplace is what caused the increase in sales.**
- Because of the many sealers who compete with the brand. When the data entry found the large number of sellers, it replaced them with NaN. Therefore, the type of data missing in the data is (MNAR:Missing Not at Random). so we compensated for the lost data in this column with Fulfilled by X or FBX or seller ... As in the following figure,it is evidence that the percentage of sales of the sellers is greater than the **sales bran**

# Assignment  Answers

## Q1

## Which products should be featured in the next advertising campaigns and promotions?

- Products that need attention from the marketing team .. we extract **top 10** products in views with less number compared to the purchases  aka Conversion Rate .

**Products Need Attention from the Marketing Team**

In [37]:
```
#let's check the percentage of purchased products to the views
most_view_products = df.loc[df['event_type'].isin(['view', 'purchase'])].groupby(['product_id', 'event_type']).size()\
.unstack(fill_value=0)
```

In [38]:
```
#making sure there's no zero values in views or purchases to avoid dividing on zero
most_view_products=most_view_products[most_view_products['purchase']>=1]
most_view_products=most_view_products[most_view_products['view']>=1]
```

In [39]:
```
most_view_products['percentage_view/purchase']=round(most_view_products['view']/most_view_products['purchase'],0)
```

In [41]:
```
most_view_products['Conversion_rate']=most_view_products['purchase']/most_view_products['view']*100
```

In [42]:
```
#from this table we know the products that we are going to report to the marketing team from the conversion rate
most_view_products.sort_values(by='view',ascending=False).head(20)
```

Out[42]:

| event_type<br>product_id | purchase | view | percentage_view/purchase | Conversion_rate |
|---|---|---|---|---|
| 5809910 | 2449 | 27286 | 11.0 | 8.975299 |
| 5892179 | 238 | 12398 | 52.0 | 1.919664 |
| 5886282 | 205 | 12325 | 60.0 | 1.663286 |
| 5877454 | 432 | 11557 | 27.0 | 3.737994 |
| 5809912 | 1190 | 11316 | 10.0 | 10.516083 |
| 5751383 | 1368 | 9960 | 7.0 | 13.734940 |
| 5877456 | 31 | 9571 | 309.0 | 0.323895 |
| 5751422 | 1578 | 9298 | 6.0 | 16.971392 |
| 5900651 | 279 | 9135 | 33.0 | 3.054187 |
| 5856186 | 179 | 9055 | 51.0 | 1.976808 |

## b. Which brands are customers most loyal to?

Loyalty to the brands  has 2 sides

- Top sales per brand according to the number of purchases
- Most brands with number of customers

# Loyalty to brands

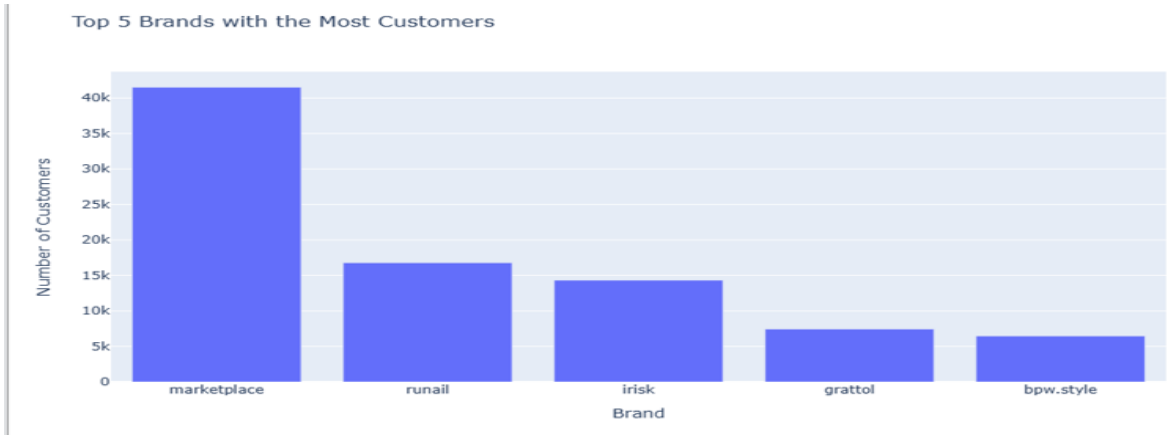
[77]:
```
#based on purchases only
customers_per_brand = most_sold.groupby('brand')['user_id'].nunique().sort_values(ascending=False)
```

[76]:
```
top_brands=pd.DataFrame(customers_per_brand).reset_index()
```

[78]:
```
top_brands.head(10)
```

:[78]:

| | brand | user_id |
|---|---|---|
| 0 | marketplace | 41518 |
| 1 | runail | 16790 |
| 2 | irisk | 14329 |
| 3 | grattol | 7434 |
| 4 | bpw.style | 6452 |
| 5 | masura | 6224 |
| 6 | ingarden | 6062 |
| 7 | uno | 4517 |
| 8 | domix | 4354 |
| 9 | estel | 3789 |



## c. Who are the most valuable customers based on their purchase history?

-**RFM (Recency, Frequency, Monetary**) analysis is a customer segmentation technique that groups customers based on their purchasing behavior.
It examines how recently and frequently people buy, as well as how much they spend, in order to identify high-value customers for targeted marketing and retention tactics.

### RFM analysis

```
[4]: #Let's check now for our valuable customers
     x=most_sold.copy() # most sold is filtering by the purchases only
     #x.head()
```

```
[4]: #getting the last day in the df as we will need it in the recency
     max_date=x['event_time'].max()
     max_date
```

```
[4]: Timestamp('2019-11-30 23:24:54+0000', tz='UTC')
```

```
[5]: #using lambda as first , second and third functions are for recnecy , frequency and monetary repectively(RFM analysis)
     df2=x.groupby('user_id').agg({'event_time':lambda x : (max_date-x.min()).days,
                                   'event_type': lambda x : len(x),
                                   'price': lambda x : x.sum()})
```

```
[2]: df2.columns = ['Recency', 'Frequency', 'Monetary']

     df2
```

```
[7]: #using qcut from pandas to give the customers scores based on their purchasing history and number of orders
     df2['Recency_score'] = pd.qcut(df2['Recency'], q=5, labels=['5', '4', '3', '2', '1'])
     #note the less recency score the better score for the cst ( means there's a small difference between days in min and max order)
     df2['Frequency_score'] = pd.qcut(df2['Frequency'], q=5, labels=['1', '2', '3', '4', '5'])
     df2['Monetary_score'] = pd.qcut(df2['Monetary'], q=5, labels=['1', '2', '3', '4', '5'])
```

```
[8]: #wanted to give the customers score based on the 3 columns after deep searching decided to take the average of them
     df2['cst_score'] = np.ceil((df2['Recency_score'].astype(int) + df2['Frequency_score'].astype(int) +
                                 df2['Monetary_score'].astype(int)) / 3).astype(int)
```

In [50]:
```
#applying a function to label our customers
def get_churn_score(cst_score):
    if cst_score in [1, 2]:
        return 'high churn'
    elif cst_score == 3:
        return 'Needs attention'
    else:
        return 'Loyal Customer'

df2['Loyalty'] = df2['cst_score'].apply(get_churn_score)
```

In [51]:
```
df2.sort_values(by='cst_score', ascending=False).head(50)
```

Out[51]:

| user_id | Recency | Frequency | Monetary | Recency_score | Frequency_score | Monetary_score | cst_score | Loyalty |
|---|---|---|---|---|---|---|---|---|
| 344995549 | 0 | 9 | 45.41 | 5 | 4 | 4 | 5 | Loyal Customer |
| 462478258 | 22 | 20 | 88.56 | 4 | 5 | 5 | 5 | Loyal Customer |
| 571014626 | 17 | 18 | 136.39 | 4 | 5 | 5 | 5 | Loyal Customer |
| 574326232 | 9 | 11 | 76.32 | 5 | 4 | 5 | 5 | Loyal Customer |
| 370772017 | 6 | 15 | 77.99 | 5 | 4 | 5 | 5 | Loyal Customer |
| 462510085 | 6 | 27 | 35.47 | 5 | 5 | 3 | 5 | Loyal Customer |
| 443496510 | 2 | 23 | 48.11 | 5 | 5 | 4 | 5 | Loyal Customer |
| 542303766 | 35 | 28 | 294.58 | 3 | 5 | 5 | 5 | Loyal Customer |
| 576108588 | 6 | 24 | 48.47 | 5 | 5 | 4 | 5 | Loyal Customer |
| 462521889 | 9 | 43 | 142.76 | 5 | 5 | 5 | 5 | Loyal Customer |
| 571013702 | 16 | 21 | 51.24 | 4 | 5 | 4 | 5 | Loyal Customer |
| 462523050 | 24 | 24 | 87.34 | 3 | 5 | 5 | 5 | Loyal Customer |
| 370624914 | 8 | 9 | 59.26 | 5 | 4 | 4 | 5 | Loyal Customer |
| 576111617 | 5 | 85 | 252.36 | 5 | 5 | 5 | 5 | Loyal Customer |
| 560376222 | 13 | 32 | 92.12 | 4 | 5 | 5 | 5 | Loyal Customer |
| 576113690 | 6 | 24 | 52.16 | 5 | 5 | 4 | 5 | Loyal Customer |

## d. Are there any price trends for a particular product over time?

In [82]:
```
# Filter the dataframe to include only events that occurred in October or November
oct_nov_df = df[(df['event_month'] == '2019-10') | (df['event_month'] == '2019-11')]

# Group the October-November dataframe by week and product, and get the mean price for each group
oct_nov_grouped = oct_nov_df.groupby(['week', 'product_id'])['price'].mean().reset_index()

# Sort the October-November grouped dataframe by price in descending order and get the top 3 rows for each week
top_3_products = oct_nov_grouped.sort_values(by=['week', 'price'], ascending=[True, False]).groupby('week').head(10)

# Reshape the top 3 products dataframe to have the weeks as columns and the product IDs and prices as rows
top_3_products_pivot = top_3_products.pivot(index='product_id', columns='week', values='price').reset_index()

# Rename the column names to include the "week" prefix
top_3_products_pivot.columns = ['product_id'] + ['week ' + str(w) + ' price' for w in top_3_products_pivot.columns[1:]]

# Print the top 3 products and their prices by week
pd.DataFrame(top_3_products_pivot)
```

Out[82]:

| | product_id | week 40 price | week 41 price | week 42 price | week 43 price | week 44 price | week 45 price | week 46 price | week 47 price | week 48 price |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4623 | 217.46 | 217.46 | 217.46 | 217.46 | NaN | 217.46 | NaN | 217.460000 | 217.460000 |
| 1 | 4851 | 252.38 | 252.38 | 252.38 | 252.38 | 252.38 | 252.38 | 252.38 | 252.380000 | 252.380000 |
| 2 | 5560754 | NaN | NaN | 194.44 | NaN | NaN | NaN | NaN | NaN | NaN |
| 3 | 5560756 | 207.94 | 207.94 | 207.94 | NaN | 207.94 | NaN | NaN | NaN | NaN |
| 4 | 5560760 | 201.59 | 201.59 | 201.59 | NaN | NaN | NaN | NaN | NaN | NaN |
| 5 | 5590822 | 194.44 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 6 | 5635474 | NaN | 307.60 | 307.60 | NaN | NaN | NaN | 307.60 | NaN | NaN |
| 7 | 5657946 | 277.78 | 277.78 | 277.78 | 277.78 | 277.78 | 277.78 | 277.78 | 277.780000 | 277.780000 |
| 8 | 5862067 | 297.14 | 297.14 | NaN | 297.14 | 297.14 | 297.14 | 297.14 | 220.734286 | 258.937143 |
| 9 | 5873654 | 273.02 | 273.02 | 273.02 | 273.02 | 273.02 | 273.02 | 273.02 | 265.954925 | 265.041479 |
| 10 | 5873656 | 273.02 | 273.02 | 273.02 | 273.02 | 273.02 | 273.02 | 273.02 | 273.020000 | 267.420000 |
| 11 | 5898227 | NaN | NaN | NaN | 236.51 | 236.51 | 236.51 | 236.51 | NaN | NaN |
| 12 | 5898228 | NaN | NaN | NaN | 236.51 | 236.51 | 236.51 | 236.51 | NaN | NaN |
| 13 | 5898229 | NaN | NaN | NaN | 236.51 | 236.51 | 236.51 | 236.51 | 236.510000 | 236.510000 |
| 14 | 5906220 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 264.117826 | 265.512500 |
| 15 | 5906221 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 316.343158 | 318.623333 |
| 16 | 89343 | 299.81 | 299.81 | 299.81 | 299.81 | 299.81 | 299.81 | 299.81 | 294.857532 | 296.035806 |

:

| | date | event | Revenu |
|---|---|---|---|
| 0 | 2019-10-30 | October Pay Day | 38832.06 |
| 1 | 2019-11-18 | Black Friday [sale starts] | 47100.96 |
| 2 | 2019-11-22 | Black Friday | 102854.76 |
| 3 | 2019-11-25 | Cyber Monday | 32914.67 |
| 4 | 2019-11-29 | November Pay Day | 78182.87 |

## Q2. Define a KPI that measures customer loyalty from the data

- **RFM (Recency, Frequency, and Monetary**) analysis is a customer segmentation technique that groups customers based on their purchasing behavior.
- It examines how recently and frequently people buy, as well as how much they spend, in order to identify high-value customers for targeted marketing and retention tactics.
- So **customer loyalty calculated with RFM with scores 4 and 5**
- Most sold products in oct , nov then overall
- Number of visits per day ( the peak in oct was 2 Oct 190k visits – peak in nov was 22 nov 255k visits)
- Using ecommerce tool in machine learning we could get ( total customers – number of orders – revenue- number of sold units – avg order value- avg revenue per cst )
- Number unique users id in oct and **nov 713100** .. this number has 2 sides ( negative and positive side ) as following:
- **Negative side :** 399k users in oct only 50k visited the website in nov and only 4k who bought that's very high churn number
- **Positive side:** we could get 368k new user_id in nov and they made more purchases more than oct we will discuss this with numbers and visualization in power bi
- Trends in some products and price variations from October to November.
  People who made a purchase in both oct and nov
- Number of people in oct was 399k only 25k who made a purchase in oct then 4k made a purchase in nov
- Number of people in nov was 368 only 31 who made a purchase

```
# Filter the dataframe to include only events that occurred in October and November
oct_nov_df = df[df['event_month'].isin(['2019-10', '2019-11'])]

# Filter to only include purchase events
purchases_df = oct_nov_df[oct_nov_df['event_type'] == 'purchase']

# Group by user_id and month to get the total number of purchases and purchase amount for each user and month
user_month_purchases = purchases_df.groupby(['user_id', 'event_month']).agg({'event_type': 'count',
                    'price': 'sum'}).reset_index()

# Filter to only include users who made at least one purchase in both October and November
loyal_customers = user_month_purchases.groupby('user_id').filter(lambda x: set(x['event_month']) == set(['2019-10', '2019-11']))

# Pivot the loyal_customers dataframe to have the months as columns and the total number of purchases
loyal_customers_pivot = loyal_customers.pivot(index='user_id', columns='event_month',
                        values=['event_type', 'price']).reset_index()

# Flatten the multi-level column names
loyal_customers_pivot.columns = [f'{y}_{x}' if x != '' else f'{y}' for x,y in loyal_customers_pivot.columns]

# Rename the columns to appropriate names
loyal_customers_pivot = loyal_customers_pivot.rename(columns={
    '2019-10_event_type': 'num_purchases_oct',
    '2019-11_event_type': 'num_purchases_nov',
    '2019-10_price': 'total_spent_oct',
    '2019-11_price': 'total_spent_nov'
})
loyal_customers_pivot
```

| | _user_id | num_purchases_oct | num_purchases_nov | total_spent_oct | total_spent_nov |
|---|---|---|---|---|---|
| **0** | 105075440 | 4.0 | 3.0 | 20.45 | 9.63 |
| **1** | 105118203 | 6.0 | 11.0 | 37.13 | 55.13 |
| **2** | 107945915 | 27.0 | 53.0 | 107.45 | 112.34 |
| **3** | 115398528 | 3.0 | 21.0 | 52.39 | 115.41 |
| **4** | 118821435 | 32.0 | 61.0 | 86.97 | 324.75 |
| **...** | ... | ... | ... | ... | ... |
| **4758** | 85993766 | 8.0 | 3.0 | 39.93 | 12.30 |
| **4759** | 88211255 | 2.0 | 6.0 | 21.25 | 34.77 |
| **4760** | 88940739 | 17.0 | 14.0 | 41.10 | 42.83 |
| **4761** | 92366748 | 17.0 | 10.0 | 30.70 | 67.48 |
| **4762** | 95758554 | 10.0 | 4.0 | 62.17 | 10.84 |

4763 rows × 5 columns

**After extracting some insights from the data, we will display them on a Jupyter notebook and in Power BI. We got some notes and recommendations**

1. **Customer lifetime value**

**CLTV** = (Avg Order price * Frequency of Purchase)* profit Margin * 36 months

2. ( we searched for the average customer life in ecommerce and it's 36 months)
3. To use RFM analysis with customer score 3 or less and send them to the retention team
4. Focus on the customers who made than **50 purchase** as they are **B2B (business to business)**and send them to the Sales team
5. For the **loyal** customer I suggest to make **loyalty program** to them to get points every purchase and he can exchange it later with a discount
6. To send top most views products with less purchases to the marketing team to check its price or discounts
7. To check the competitors' prices to avoid the mistake happened in **Oct and the high churn** rate
8. To get the data of every customer complain **last month** to analyze it and know the causes of the problem
9. Reasons for Churn and why only about 10% of people who bought in October bought again in November... Due to a lack of information in the data, we can summarize the reasons as follows:
    a. **Because of the poor quality of some items**
    b. **High prices comparing to the Competitors**
    c. **Customer support**
    d. **Logistics**
    e. **Lack of payment methods**
    f. **User interface**

# • Association Rule and Recommendation system

## Data cleaning

- Duplicate rows were eliminated since they might have been acquired owing to some technical errors when the **two months datasets (Oct 2019 - Nov 2019)** were combined.
- Items with negative and zero prices are excluded since we can't promote them to customers because they are likely free gifts or transient items. To create our user-product matrix, only users who bought at least **10 different products** are preserved.
- The matrix's columns are **products,** while its rows are **individual users**.
- The **number of times that user I purchased product j is shown by the (i,j)-entry.**
- We eliminated rows with blank brand information when building the **user-brand matrix.** We didn't handle them all since more than **41% of the product brand information** was missing from the original datasets.
- Since this "no name" brand would be the most well-known and dominant, we avoided treating them all as a single brand.
- Users who have bought at least **five distinct brands** were used to create the user-brand matrix. This matrix's **columns** represent **distinct brands**, while its **rows** represent **distinct users**.
- The number of times that **user I bought brand j is shown by the (i,j)-entry.**
- In order to provide enough information for our recommender systems to learn from, we filtered the data to **only include people with a sufficient number of purchase histories.**

## Feature Engineering

- Using the raw browsing history of consumers, we first aggregated all of the browsing data from **October 2019 to November 2019** together, and then we sorted the entire dataframe by distinct user ids to produce a numerical quantification of users' preferences for specific brands or goods.
- For information on **user-brand relationships**, we added the **purchase counts** for every user-brand combination that was linked to **at least one "buy" event** in the combined dataframe.
- Also produced for the appropriate brand information is **a numerical brand id.**
- The same process used to create the user-brand connections is also used to create the **user-product relationships data.**
- We first create **user-brand and user-product relationships data**, after which we create the user-brand and user-product matrix, where each row represents the **number of times** the user has bought that **brand or product.**

# Association Rule

- Association rule mining is a popular technique in data mining and machine learning that aims to identify relationships and patterns between items in a dataset. It is used to discover strong associations or correlations between items in a transactional database or a set of features.
- Association rules are evaluated based on two metrics: **support and confidence**.
- **Support** measures the **frequency** of occurrence of the itemset in the dataset, while confidence measures the conditional probability that the consequent occurs given the antecedent.
- Other metrics, such as **lift and conviction,** can also be used to evaluate the strength of the association rules.
- Association rule mining has numerous applications in various fields, such as market basket analysis, customer segmentation, recommendation systems, and web usage mining. It can be used to identify cross-selling opportunities, understand customer behavior, and improve marketing strategies.
- The **absence** of a **column** for each **transaction** or **invoice** for a customer was the issue with this data.

- The solution was to **filter the data** for the brand that the **purchase process was based on, with the customer serving as the index,** so that if the customer **purchases** a particular brand, he receives the **number (1)**
  and if **he does not,** he **receives(0)**, so that I may know the length of the strength of the **relationships between the brands and each other,** as shown in the following figure:

```
df_test.groupby(['user_id','brand']).sum()['purchase_count'].unstack().reset_index().fillna(0).set_ind
ead()
```

| brand | airnails | art-visage | bluesky | cnd | concept | domix | elskin | f.o.x | fancy | freshbubble | ... | milv | oniq | profhenna | runail | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | | | | |
| 1120748 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 1.0 | |
| 1458813 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 21.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 4103071 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 5493470 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 5.0 | |
| 6217356 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |

5 rows × 32 columns

```
# Encode
def encoder(x):
    if(x <= 0): return 0
    if(x >= 1): return 1
```

```
# Apply to the dataframe
df_filter_ass_encoded = df_filter_ass.applymap(encoder)
df_filter_ass_encoded.head()
```

| ll | art-visage | artex | aura | balbcare | barbie | batiste | ... | uskusi | veraclara | vilenta | vosev | weaver | yoko | ypsed | yu-r | zeitun | zinger |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

- Then We used **Apriori**(a popular algorithm in association rule mining that aims to discover frequent itemsets in a transactional database).

- Then the value of **min support and min confidence** is determined so that it is strong rules as in the following figure

```
frequent_itemsets = apriori(df_filter_ass_encoded,min_support=0.02, use_colnames=True)
frequent_itemsets
```

```
C:\ANACONDA\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

| | support | itemsets |
|---|---|---|
| 0 | 0.049298 | (airnails) |
| 1 | 0.055990 | (art-visage) |
| 2 | 0.034782 | (artex) |
| 3 | 0.035159 | (beautix) |
| 4 | 0.030352 | (benovy) |
| ... | ... | ... |
| 809 | 0.021397 | (milv, irisk, bpw.style, runail, masura) |
| 810 | 0.021585 | (ingarden, irisk, runail, domix, grattol) |
| 811 | 0.021868 | (freedecor, ingarden, irisk, runail, grattol) |
| 812 | 0.026298 | (ingarden, irisk, runail, grattol, masura) |
| 813 | 0.023376 | (ingarden, irisk, runail, uno, grattol) |

```
# Create Association Rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Sort values based on lift
rules = rules.sort_values("lift",ascending=False).reset_index(drop= True)
rules.head(20)
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (bpw.style, milv) | (freedecor, de.lux) | 0.120652 | 0.060986 | 0.028089 | 0.232813 | 3.817477 | 0.020731 | 1.223969 |
| 1 | (freedecor, de.lux) | (bpw.style, milv) | 0.060986 | 0.120652 | 0.028089 | 0.460587 | 3.817477 | 0.020731 | 1.630195 |
| 2 | (kapous) | (concept) | 0.084834 | 0.071543 | 0.022528 | 0.265556 | 3.711830 | 0.016459 | 1.264162 |
| 3 | (concept) | (kapous) | 0.071543 | 0.084834 | 0.022528 | 0.314888 | 3.711830 | 0.016459 | 1.335791 |
| 4 | (freedecor, milv) | (bpw.style, de.lux) | 0.081629 | 0.095202 | 0.028089 | 0.344111 | 3.614527 | 0.020318 | 1.379498 |
| 5 | (bpw.style, de.lux) | (freedecor, milv) | 0.095202 | 0.081629 | 0.028089 | 0.295050 | 3.614527 | 0.020318 | 1.302746 |
| 6 | (estel) | (concept) | 0.091432 | 0.071543 | 0.022528 | 0.246392 | 3.443966 | 0.015987 | 1.232015 |
| 7 | (concept) | (estel) | 0.071543 | 0.091432 | 0.022528 | 0.314888 | 3.443966 | 0.015987 | 1.326160 |
| 8 | (bpw.style, milv, de.lux) | (freedecor) | 0.039401 | 0.207842 | 0.028089 | 0.712919 | 3.430093 | 0.019900 | 2.759349 |

## Recommendation

- We intend to create **two recommender systems**:
    - one that recommends **products for customers**
    - The other that **recommends brands**
- The online store can use such recommender systems to recommend **products or brands** to a specific customer. For example, the retail platform can send **emails advocating particular products or offering unique coupon codes on particular brands.**
- One of the most popular models for recommender systems is **collaborative filtering**. It predicts a **person's preference** using information from other users, assuming that a user favors comparable goods and that an item attracts others with similar interests.

## Target Variable

- We would like to know people's preferences for products and brands in order to construct the recommender system.
- We decided to rely on the variable **'event type'** to provide implicit feedback through users' actions of view, add to cart, purchase, and remove from cart because the datasets lack explicit feedback such as user ratings.
- We regard 'purchasing' as a statement of preference for specific products and brands.
- Consequently, we defined the target variable as **the number of purchases for both items and brands .**

## Evaluation Metrics

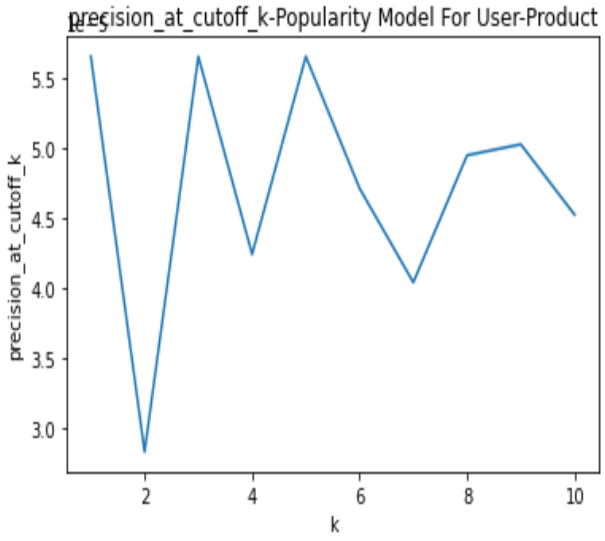
- For each customer, the recommendation systems generate a sorted list of items.
- The target is for the sorted list to accurately match the preferences of the customers.
- As a result, The **"precision at cutoff k"** is an evaluation parameter, where **K** is the number of **products/brands** to recommend for a specific **customer**. For a given user:

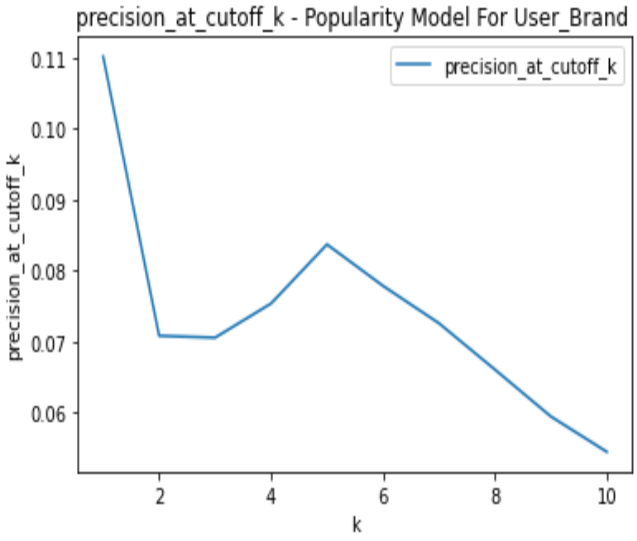
$$P(k) = \frac{|a \cap p_k|}{k},$$

- The metric is averaged across all customers to evaluate model performance. Business limitations determine the value of K.
- We calculated the mean "precision at cutoff k" for a list of K values.
- If a buyer received recommendations for 8 products, and later purchases 10 of them, then precision is 0.8.

# Models For Recommendation

- ## Baseline Model: Popularity Model
    - To compare and assess models, we would need to run a **baseline model**.
    - Techniques utilized beyond this method should be considered if they provide considerably **improved accuracy and complexity** because baseline normally uses a fairly simple approach. In this instance, we'll apply the **popularity model.**
    - The most well-liked **products or brands** are used in the popularity model's recommendations.
    - It can assist us in identifying potential popularity bias in other models. Moreover, popularity models partially address the industry-wide **cold-start issue**, particularly when only **implicit purchasing data** are available.



## Model 1: Item-Based Collaborative Filtering

- For data with far more users than objects, **item-based collaborative filtering** can be quite useful. With the **item-based** method, a **similarity matrix** between items is calculated.
- The expected **purchase counts** for an item that a consumer has not purchased are calculated for each customer as a weighted average of the purchase counts for **S nearby goods**.
- In our model, **three similarity** measures were used to fit the model (**cosine similarity, Pearson similarity, and Jaccard similarity**).
- We noted that, for the **User-Product matrix**, **Jaccard similarity offers** the least **RMSE** after hyper- parameterizing S.
- **Jaccard Similarity** produces the least **RMSE** on the validation set for the **User-Brand matrix.**

## Model 2: Matrix Factorization with Implicit Alternative Least Squares

- In contrast to **explicit feedback** like ratings, where a low rating indicates dislike and a high ranking indicates liking.
- It is difficult to infer user preference from the **purchase counts** of our objective variable. Another option is that a consumer passed up a product because he was unaware of its existence.
- The typical **matrix factorization** method performs badly on our dataset because it ignores the zero entries in the matrix during optimization. We used the implicit **Alternating Least Squares** approach to handle our implicit feedback dataset

## Model Results (performance on test data)

- With our selected model, we combined the training data with the validation data, and retained the model on the combined dataset. After predicting on the test set, we got a test **precision_at_cutoff_k**, which is  about 0.1, which is **lower** than the **precision_at_cutoff_k** on the training dataset.

## conclusion

- In this project, we constructed the **association rule** and **recommender systems** mainly using techniques that belong to **collaborative filtering and matrix factorization.**
- One issue of our model is that it can only suggest brands and goods to users who have **registered**.
- The new users may not **get good recommendations** because our model only considers the **purchase** event and  ignores `**view**` and `**add-to-cart**`.

# Screenshots of our recommendation system

## Product Recommendation using CF

Choose User to get recommended products

254751820

| user_id | product_id | score |
|---|---|---|
| 254751820 | 5854479 | 0.1343129741 |
| 254751820 | 5854482 | 0.1227561567 |
| 254751820 | 5800921 | 0.1138280365 |
| 254751820 | 5730210 | 0.100726578 |
| 254751820 | 5618229 | 0.100726578 |
| 254751820 | 5851216 | 0.100726578 |
| 254751820 | 5700080 | 0.0876433783 |
| 254751820 | 5847382 | 0.0555555556 |
| 254751820 | 5870124 | 0.0555555556 |
| 254751820 | 5614832 | 0.0555555556 |

Choose User to get recommended brands

105697630

| user_id | brand | score |
|---|---|---|
| 105697630 | runail | 0.2602356374 |
| 105697630 | irisk | 0.2575701773 |
| 105697630 | bpw.style | 0.1973702759 |
| 105697630 | metzger | 0.1838154793 |
| 105697630 | domix | 0.1760243028 |
| 105697630 | severina | 0.1705716997 |
| 105697630 | zinger | 0.1499827504 |
| 105697630 | freedecor | 0.1490627974 |
| 105697630 | milv | 0.1327797323 |
| 105697630 | de.lux | 0.1318692267 |

**And here is the link of deployed Streamlit Application**

- .https://aya-ai-2022-aya-app1-44d4h6.streamlit.app/