Amazon: EDA & Recommendation System (EN)

Introduction

In this project we'll be performing an exploratory data analysis of Amazon sales data.

Amazon is an American multinational technology company, engaged in e-commerce, cloud computing, online advertising, digital streaming, and artificial intelligence. It is considered one of the Big Five American technology companies.

Data Source

The dataset is available on Kaggle at this
link

Analyzing Data

Initial Data Assessment

First of all, we'll analyze the data structure, see if there are any incorrect/null values, drop some values and modify data types.

For instance, we'll change the data type of values in **discounted_price**, **actual_price**, **discount percentage** to float and drop unnecessary symbols within them.

▼ Code

```
df['discounted_price'] = df['discounted_price'].str.replace('₹', '')
df['discounted_price'] = df['discounted_price'].str.replace(',', '')
df['discounted_price'] = df['discounted_price'].astype('float64')
```

```
df['actual_price'] = df['actual_price'].str.replace('₹', '')
df['actual_price'] = df['actual_price'].str.replace(',', '')
df['actual_price'] = df['actual_price'].astype('float64')
```

```
df['discount_percentage'] = df['discount_percentage'].str.replace('%', '')
df['discount_percentage'] = df['discount_percentage'].astype('float64')
df['discount_percentage'] = df['discount_percentage']/100
```

Exploring Categories

Category names look really inconvenient so we will extract product category and subcategory from the category columns

▼ Code

```
df['product_category'] = df['category'].str.split('|').str.get(0)
```

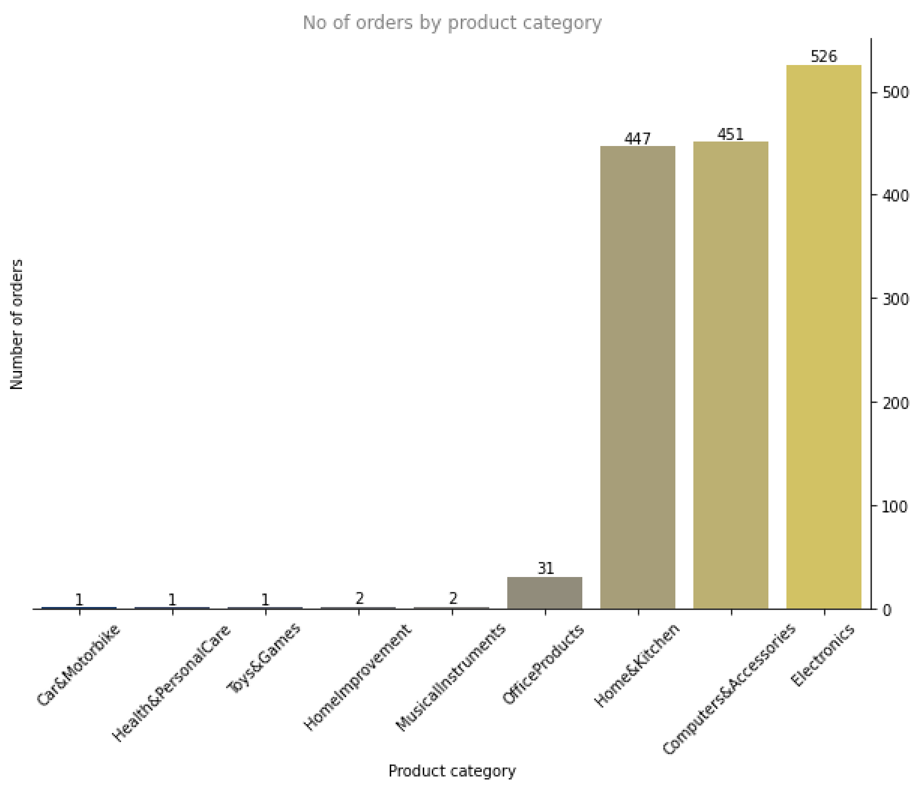
```
df['product_category'].unique()
```

```
df['product_subcategory'] = df['category'].str.split('|').str.get(1)
df['product_subcategory'].unique()
```

The most popular by number of purchases category is Electronics.

▼ Code

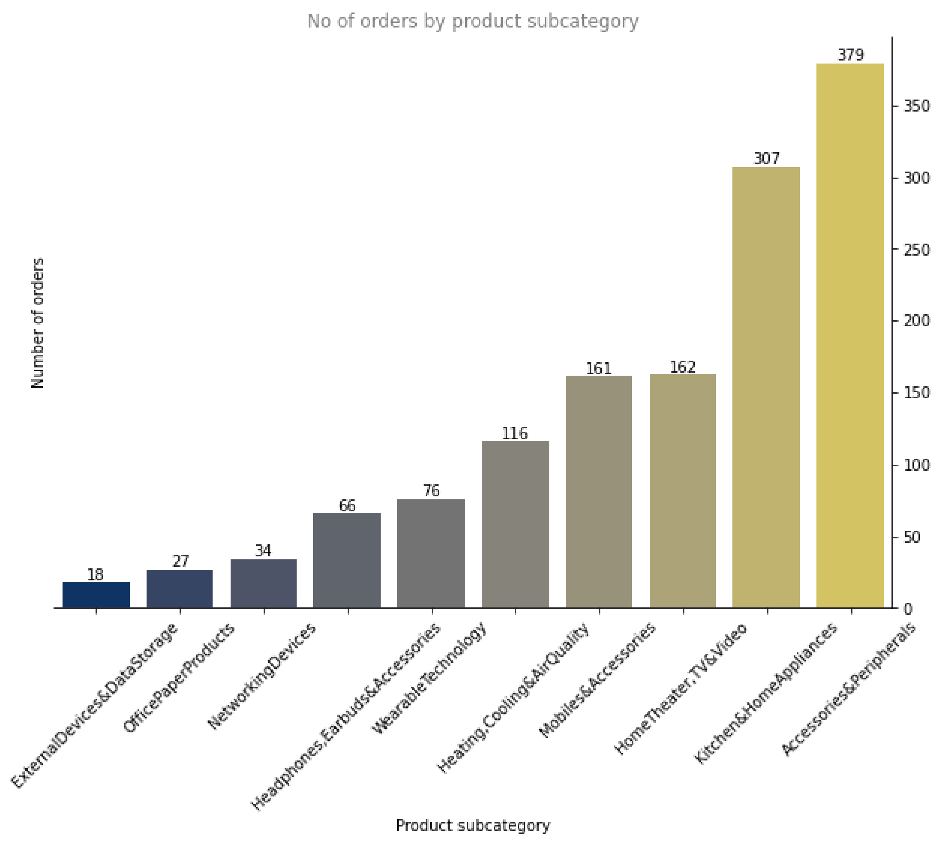
```
cat_count = df.groupby('product_category', as_index=False)['product_id'].coun
t().reset_index()

fig = plt.figure(figsize=(10,7))

ax = sns.barplot(data = cat_count, x = 'product_category', y = 'product_id',
palette='cividis', order=cat_count.sort_values('product_id').product_categor
y)

plt.xticks(rotation=45)

ax.set_xlabel('Категория')
ax.set_ylabel('Количество покупок')

sns.despine(left=True, right=False)

for i in ax.containers:
    ax.bar_label(i,)

plt.title('Количество покупок по категориям', color = 'gray')
```



The most popular subcategory is Accessories&Peripherals.

▼ Code

```
subcat_count = df.groupby('product_subcategory')['product_id'].count().sort_v
alues(ascending=False).reset_index().head(10)
fig = plt.figure(figsize=(10,7))

ax = sns.barplot(data = subcat_count, x = 'product_subcategory', y = 'product
_id', palette='cividis', order=subcat_count.sort_values('product_id').product
_subcategory)

plt.xticks(rotation=45)

ax.set_xlabel('Подкатегория')
ax.set_ylabel('Количество покупок')

sns.despine(left=True, right=False)

for i in ax.containers:
    ax.bar_label(i,)

plt.title('Количество покупок по подкатегориям', color = 'gray')
```



Setting up a Recommendation System

First, we will convert user id into categories to make processing easier

▼ Code

```
df['user_id_category'] = df['user_id'].astype('category')
```

Now we take a thousand of our customers and create a dictionary with numbers 0-1000 to replace their lengthy ids

▼ Code

```
target_cust = df['user_id_category'].unique()[:1000]
cust_dict = {id: i for i, id in enumerate(target_cust)}
```

Create a dataframe with those records that refer to our 1000 subject customers

▼ Code

```
rec = df[df['user_id_category'].isin(target_cust)].copy()
```

Replacing the user id using our previously created dictionary

▼ Code

```
def replace(x):
    n_id = 0
    try:
        n_id = cust_dict[x.user_id_category]
    except Exception as e:
        print(e)
        res = -1
    return n_id

rec['user_id'] = rec.apply(replace, axis=1)
rec.head()
```

Grouping the dataframe to then create a matrix

▼ Code

```
gr = rec.groupby(['user_id', 'product_name'])['product_id'].count()
```

Now, we create a matrix with user_id as the index and product name as columns

▼ Code

```
matrix_cust = gr.unstack('product_name')
```

Applying a lambda function so that the values are numeric

▼ Code

```
matrix_cust = matrix_cust.applymap(lambda x: 1 if x > 0 else 0)
```

We'll be using cosine similarity to identify customers with similar purchases

▼ Code

```
cos_sim = cosine_similarity(matrix_cust, matrix_cust)
cos_sim_df = pd.DataFrame(cos_sim)
cos_sim_df.columns = matrix_cust.index
cos_sim_df['user_id'] = matrix_cust.index
cos_sim_df = cos_sim_df.set_index('user_id')
```

Let's find user ids that have a high cosine similarity:

▼ Code

```
col_list = list(cos_sim_df)
cos_sim_df['sum'] = cos_sim_df[col_list].sum(axis=1).round(2)
cos_sim_df[cos_sim_df['sum']>1]
```

Setting up recommendations for user_id 296.

These are the products that user id 5 has bought:

▼ Code

```
cust_a = matrix_cust.loc[5]
```

```
cust_a_purchases = cust_a[cust_a > 0].index.tolist()
cust_a_purchases
```

These are the products that user id 296 has bought:

▼ Code

```
cust_b = matrix_cust.loc[296]

cust_b_purchases = cust_b[cust_b > 0].index.tolist()
cust_b_purchases
```

These are the products that we might recommend to user id 296:

▼ Code

```
b_recs = set(cust_a_purchases) - set(cust_b_purchases)
b_recs
```

Conclusions

In this mini-project we have taken a look at Amazon sales data and performed exploratory data analysis that have allowed us to identify the most popular product categories and subcategories.

We have also set up a simple recommendation system that can be used to promote certain products.

Links

Link to the full Jupyter Notebook (RU)

Link to the full Jupyter Notebook (EN)

Contacts

lianazaripovar@gmail.com

tg: @zaripova_liana