ML Capstone Project - Ecommerce

Introduction

The AccioJob ML Capstone E-commerce Dataset available via Kaggle is a comprehensive collection of data tailored for machine learning projects in the e-commerce domain. This dataset encapsulates a wide array of variables, providing insights into various aspects of e-commerce operations and customer interactions. It's designed to facilitate exploration, analysis, and the development of predictive models within the realm of e-commerce.

Containing multiple tables and data points, this dataset offers ample opportunities for research, trend analysis, and the development of predictive models to enhance decision-making processes within the e-commerce industry. It includes information that spans user behavior, transactional data, product details, and more, offering a rich landscape for project development and analysis.

Business Problem

An electronics store aims to optimize its e-commerce platform by better understanding customer behaviors and preferences. The goal is to enhance marketing strategies, improve customer experience, and ultimately increase sales and customer loyalty.

Data Dictionary

event_time: Represents the timestamp indicating the occurrence of a purchase or related event (e.g., adding to cart, viewing). Vital for analyzing purchase patterns across time.

order_id: A unique identifier assigned to each order, facilitating individual transaction tracking and crucial for distinguishing between different orders during analysis.

product_id: Unique identification for each product purchased, pivotal for product-level analysis and identification of specific items.

category_id: An exclusive identifier for the category of each product. Aids in categorizing products for comprehensive analysis.

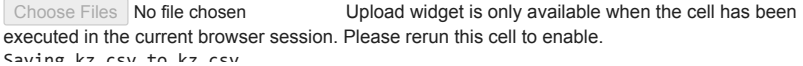
category_code: Possibly a textual or descriptive representation of the product category. It offers a more intuitive understanding than category IDs regarding product types.

brand: Signifies the brand of the product, important for brand-level analysis and understanding customer brand preferences.

price: The selling price of the product. Essential for revenue analysis and comprehending purchasing patterns concerning different price points.

user_id: A distinctive identifier assigned to each customer. Enables analysis on a customer level, including purchase history, frequency, and preferences.

```
from google.colab import files
uploaded = files.upload()
```




```
import pandas as pd
import io

df = pd.read_csv(io.BytesIO(uploaded['kz.csv']))
df.head()
```

	event_time	order_id	product_id	category_id	
0	2020-04-24 11:50:39 UTC	2294359932054530000	1515966223509080000	2.268105e+18	e
1	2020-04-24 11:50:39 UTC	2294359932054530000	1515966223509080000	2.268105e+18	e
	2020-04-24				

```
df.describe()
```

```
                order_id   product_id   category_id          price       user_id
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column         Non-Null Count    Dtype  
---  ------         --------------    -----  
 0   event_time     1048575 non-null  object 
 1   order_id       1048575 non-null  int64  
 2   product_id     1048575 non-null  int64  
 3   category_id    850228 non-null   float64
 4   category_code  789495 non-null   object 
 5   brand          814816 non-null   object 
 6   price          850228 non-null   float64
 7   user_id        107159 non-null   float64
dtypes: float64(3), int64(2), object(3)
memory usage: 64.0+ MB
```

```
df.isnull().sum()*100/1048575
```

```
event_time        0.000000
order_id          0.000000
product_id        0.000000
category_id      18.915862
category_code    24.707818
brand            22.293017
price            18.915862
user_id          89.780512
dtype: float64
```

```
df = df.dropna()
```

```
df.isnull().sum()
```

```
event_time       0
order_id         0
product_id       0
category_id      0
category_code    0
brand            0
price            0
user_id          0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 75651 entries, 0 to 1048444
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   event_time     75651 non-null  object 
 1   order_id       75651 non-null  int64  
 2   product_id     75651 non-null  int64  
 3   category_id    75651 non-null  float64
 4   category_code  75651 non-null  object 
 5   brand          75651 non-null  object 
 6   price          75651 non-null  float64
 7   user_id        75651 non-null  float64
dtypes: float64(3), int64(2), object(3)
memory usage: 5.2+ MB
```

```
df
```

	event_time	order_id	product_id	category_id	
0	2020-04-24 11:50:39 UTC	2294359932054530000	1515966223509080000	2.268105e+18	
1	2020-04-24 11:50:39 UTC	2294359932054530000	1515966223509080000	2.268105e+18	
2	2020-04-24 14:37:43 UTC	2294444024058080000	2273948319057180000	2.268105e+18	electr
3	2020-04-24 14:37:43 UTC	2294444024058080000	2273948319057180000	2.268105e+18	electr
5	2020-04-26 08:45:57 UTC	2295716521449610000	1515966223509260000	2.268105e+18	
...	...	...	...	...	

```
import numpy as np


df['user_id'] = df['user_id'].astype(int)
df['category_id'] = df['category_id'].astype(int)


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 75651 entries, 0 to 1048444
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   event_time     75651 non-null  object
 1   order_id       75651 non-null  int64
 2   product_id     75651 non-null  int64
 3   category_id    75651 non-null  int64
 4   category_code  75651 non-null  object
 5   brand          75651 non-null  object
 6   price          75651 non-null  float64
 7   user_id        75651 non-null  int64
dtypes: float64(1), int64(4), object(3)
memory usage: 7.2+ MB
```

```
df
```

	event_time	order_id	product_id	category_i
0	2020-04-24 11:50:39 UTC	2294359932054530000	1515966223509080000	226810542664816998
1	2020-04-24 11:50:39 UTC	2294359932054530000	1515966223509080000	226810542664816998
2	2020-04-24 14:37:43 UTC	2294444024058080000	2273948319057180000	226810543016299008
3	2020-04-24 14:37:43 UTC	2294444024058080000	2273948319057180000	226810543016299008
5	2020-04-26 08:45:57 UTC	2295716521449610000	1515966223509260000	226810544263684992
...	...	...	...	...

Total Revenue By Brand

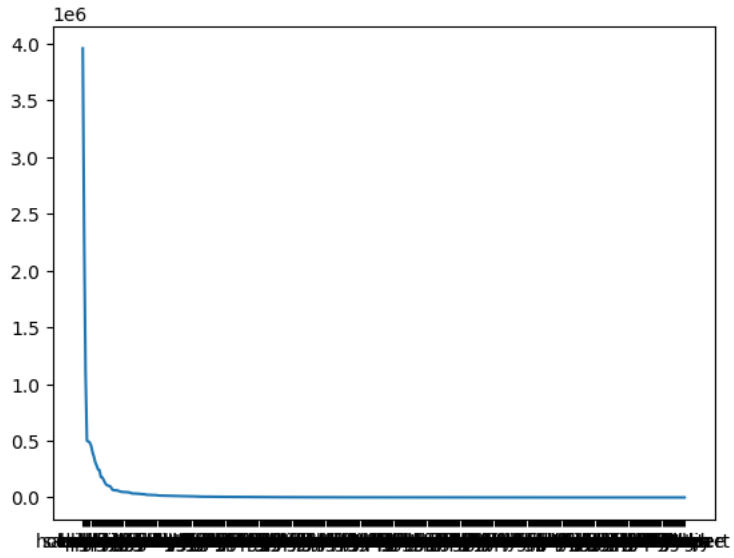
```
df.groupby('brand')['price'].sum().sort_values(ascending=False)
```

```
brand
samsung       3958377.79
apple         2291718.46
lg            1104532.28
bosch          500858.05
lenovo         493777.16
                 ...
att               0.92
cablexpert        0.90
rossija           0.88
reno              0.82
pedigree          0.23
Name: price, Length: 434, dtype: float64
```

Samsung has the highest revenue

```
import matplotlib.pyplot as plt


plt.plot(df.groupby('brand')['price'].sum().sort_values(ascending=False))
plt.show()
```




```
import seaborn as sns


sns.barplot(df.groupby('brand')['price'].sum().sort_values(ascending=False))
plt.show()
```


1e6
4.0
3.5
3.0
2.5
2.0
price