E-COMMERCE BEHAVIOR - PYTHON

JONATHAN BORUNDA REYES

Table of Contents

Data Set

Analyzing ecommerce behavior allows for an understanding of the type of products that are purchased, what brands are purchased and the price the products are purchased at. The purpose of this analysis is to understand what is being purchased and at what price. The data set used is sourced from Kaggle. There are 7 months of data that total to over 40 GB of data. Within this analysis, I focused only on the month of October 2019 and limited the data set to a random 100,000 records because this data will be processed on a local machine. The data also consists of 9 unique fields, presented in the table below, that have been cleaned utilizing the dropping of columns, removing of null values, and applying delimiters.

Field Name	**Data Type**
event_time	object
event_type	object
category_id	float64
category_code	object
brand	object
price	float64
user_id	int64
user_session	object
product_id	int64

Data Set URL: https://www.kaggle.com/mkechinov/ecommerce-behavior-data-from-multi-category-store?select=2019-Oct.csv

Data Cleaning

Deletion of Columns

As the data set contains 9 columns, there isn't a need for every single one of them as this analysis will only be using 4 of those existing columns: event_type, category_code, brand, and price. Deletion of those columns was done by utilizing 'del' function. Presented below are images of the code, the data set prior to deletion of columns, and the data set after the deletion of columns.

Deletion of Columns Code

```
del df1['product_id']
del df1['category_id']
del df1['user_id']
del df1['event_time']
del df1['user_session']
print(df1.dtypes)
```

Columns Prior to Using 'del'

```
event_time        object
event_type        object
product_id         int64
category_id      float64
category_code     object
brand             object
price            float64
user_id            int64
user_session      object
dtype: object
```

Columns After Using 'del'

```
event_type        object
category_code     object
brand             object
price            float64
dtype: object
```

NaN and Null Values

Further cleaning of the data set, there is a need to remove NaN and Null values. There are several methods that were used to perform this. I identified ‘NaN’ as a float values, assigned it a variable, and replaced it and renamed it with a numeric value. Additionally, I utilized ‘dropna’ to delete rows that contain empty, NaN, and Null values.

NaN and Null Value Cleaning Code

```
nan_value = float("NaN")
df1.replace("", nan_value, inplace=True)
df1.dropna(inplace=True)
print(df1.head())
```

Records Prior to NaN and Null Cleaning

	event_type	category_code	brand	price
0	view	appliances.environment.vacuum	samsung	64.33
1	view	appliances.environment.vacuum	samsung	66.90
2	view	construction.tools.drill	NaN	39.10
3	view	construction.tools.drill	NaN	39.10
4	view	electronics.smartphone	apple	587.53

Records After NaN and Null Cleaning

	event_type	category_code	brand	price
0	view	appliances.environment.vacuum	samsung	64.33
1	view	appliances.environment.vacuum	samsung	66.90
4	view	electronics.smartphone	apple	587.53
5	view	electronics.smartphone	oppo	592.01
7	view	electronics.audio.headphone	xiaomi	22.39

Delimiting

The original data set contained a column of ‘category_code’, which consists of 3 subcategories. These three subcategories are split by a ‘.’ between them. By using a delimiting function, I can split the string into 3 additional columns, name them and later drop the original column. This allows me to analyze each subcategory and perform a much more focused analysis.

Splitting Column Code

```
category = df1['category_code'].str.split(".", n = 2, expand = True)
df1['category']= category[0]
df1['subcategory']= category[1]
df1['subtype']= category[2]
df1.drop(columns =['category_code'], inplace = True)
df1.dropna(inplace = True)
#print(df1.head())
```

Column, ‘category_code’ Prior to Splitting

	event_type	category_code	brand	price
0	view	appliances.environment.vacuum	samsung	64.33
1	view	appliances.environment.vacuum	samsung	66.90
4	view	electronics.smartphone	apple	587.53
5	view	electronics.smartphone	oppo	592.01
7	view	electronics.audio.headphone	xiaomi	22.39

New 3 Columns After Splitting

	event_type	brand	price	category	subcategory	subtype
0	view	samsung	64.33	appliances	environment	vacuum
1	view	samsung	66.90	appliances	environment	vacuum
7	view	xiaomi	22.39	electronics	audio	headphone
8	view	xiaomi	22.39	electronics	audio	headphone
9	view	xiaomi	22.39	electronics	audio	headphone

Summary Statistics

Using statistical functions within python we can better understand behavior of users on an ecommerce site. From the code below, we can see the counts, mean, median, and standard deviations are identified for different values. Of the values used, we can see that based on the event_type, most users view goods at a count of 225,117 compared to purchasing at 3,474. Further we can see that the average price of an item purchased, the mean, is $172.79. The maximum of an item purchased was $2,573.79.

Brands also have a great effect on the user's behavior. Looking at what prices brands are listed on an ecommerce site, one can see that it is very important to understand how one's price compares to that of another's. Analyzing, the mean, median and standard deviation of brand prices, we learn that specific brands' products listed have an average price. Zotac for example lists their products at the average price of $266.17, while ACD lists their products with an average price of $5.53. In comparison, the median pricing, that lies in the middle of all the different records for Zotac is $171.15 and for ACD is 5.53. The standard deviation for Zotac is $172.13 and ACD is 0. Zotac has a much greater distribution of pricing compared to ACD that is 0 meaning that all ACD products have the same pricing.

Statistics Code

```
#STATISTICS
print(df1[["event_type", "price"]].groupby("event_type").describe())
print(df1[["brand", "price"]].groupby("brand").median())
print(df1[["brand", "price"]].groupby("brand").mean())
print(df1[["brand", "price"]].groupby("brand").std(ddof=0))
```

Using ‘describe’ to Print Statistics of Price by Event_Type

event_type	count	mean	std	...	50%	75%	max
				...			
cart	2236.0	178.184199	143.199108	...	161.93	189.1000	1414.93
purchase	3474.0	172.795386	171.769292	...	128.70	210.2875	2573.79
view	225117.0	201.887920	252.111594	...	121.24	257.1500	2574.04

Median Price per Brand

brand	price
aardwolf	84.430
acd	5.530
acer	146.460
aces	160.390
acme	34.750
...	...
zlatek	51.460
zongshen	254.060
zoom	167.310
zorg	163.445
zotac	171.150

Mean Price per Brand

brand	price
aardwolf	84.430000
acd	5.530000
acer	218.894209
aces	243.636667
acme	39.894430
...	...
zlatek	45.665880
zongshen	252.156000
zoom	167.310000
zorg	194.473333
zotac	266.169231

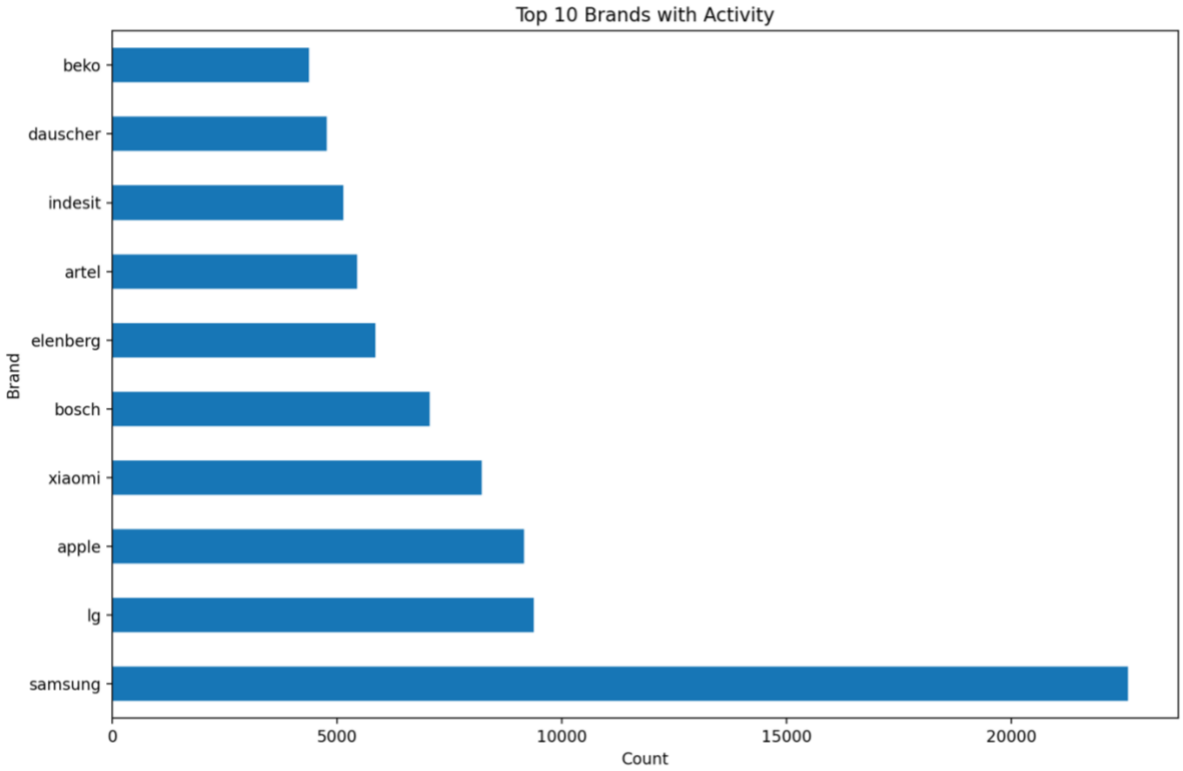
Standard Deviation of Price per Brand

brand	price
aardwolf	0.000000
acd	0.000000
acer	165.804350
aces	117.728565
acme	34.225040
...	...
zlatek	11.150840
zongshen	3.808000
zoom	0.000000
zorg	67.389104
zotac	173.129560

Analysis and Visualizations

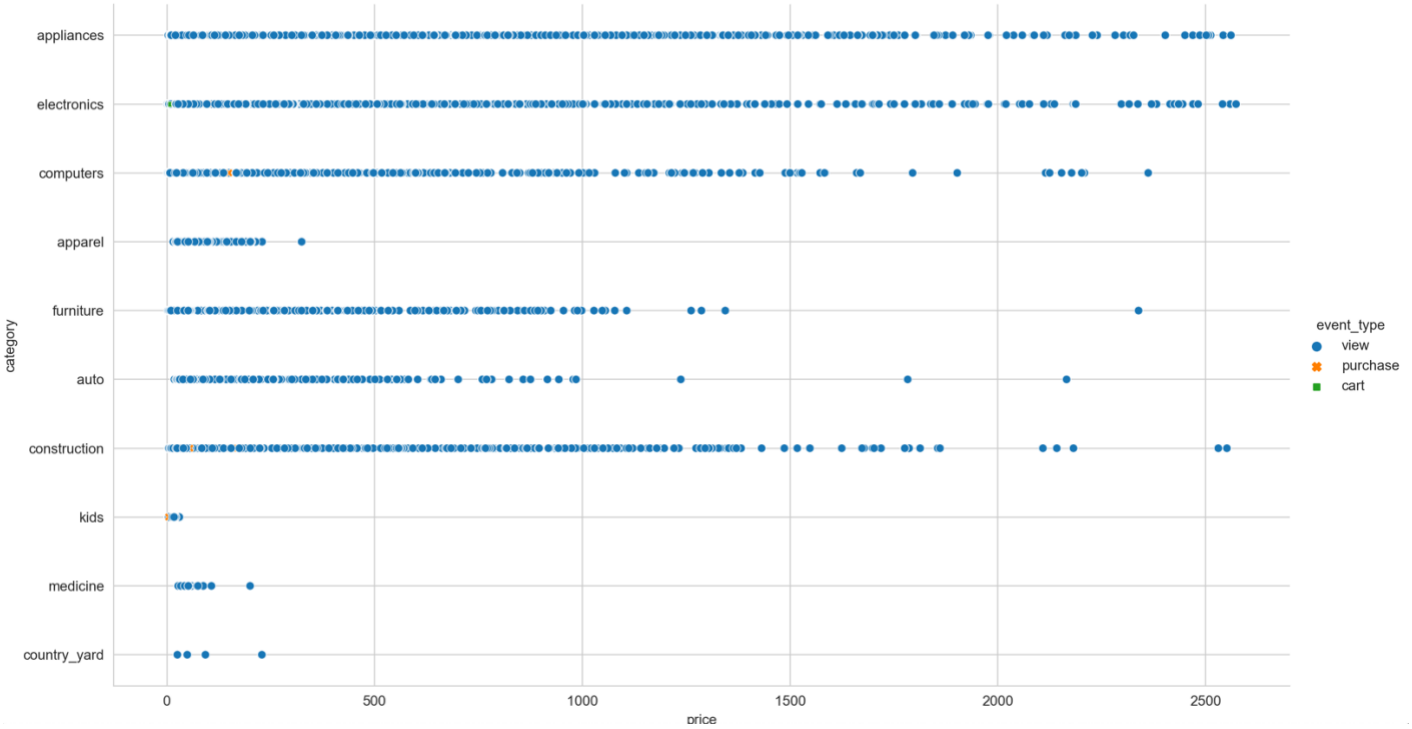
Utilizing python to further analyze data and create an appropriate visualization, I have prepared three visuals representation of the data to better understand their meaning and easily compare the data to with other variables.

What are the Top 10 Brands with Activity?



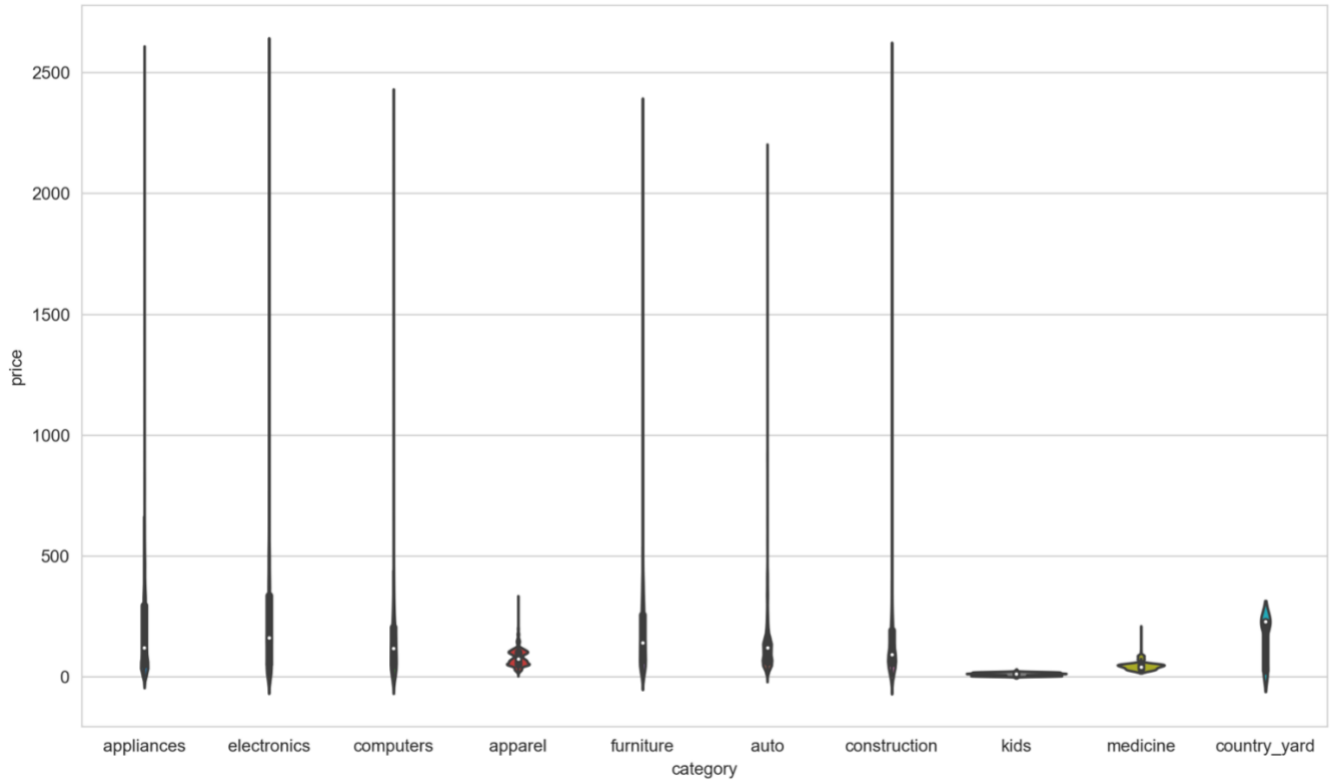
Understanding the top 10 brands and creating a sorted visualization allows for a clear depiction that Samsung has the highest count of items listed. The count of Samsung exceeds that of 20,000. Samsung is followed by LG, which has a count of almost 10,000, and Apple also has a count of almost 10,000 items listed. LG and Apple are viewed as competitive in their activity to each other, yet they are listing half the number of items as Samsung is. From this, we can determine that there is a greatly likelihood of interest in Samsung products and it is likely to due to their greater scope in types of products sold.

What Category of Items has the Most Events? What are those Events?



This scatter plot depicts a distribution of events per category. It provides an understanding of what events are most occurring, for what category are they occurring, and at what price. We can first see that the event that most occurs is ‘view.’ In comparison, there is a lot less purchasing and placing of items in one’s cart. Additionally, we can see that there is a greater concentration of activity in appliances and electronics. Further, we can see that the distribution is more concentrated at lower prices as well.

What is the Distribution of Categories by Price?



This violin distribution illustrates the prices per category and how goods are distributed per category. From this, we can understand how great of a range there is in prices per category, and in what range of pricing are most items in per category. Appliances, electronics, computers, furniture, auto, and construction have a greater distribution signifying that there are varying products listed at greatly varying prices. Apparel, kids, medicine, and country_yard have a much smaller distribution which reveals that the pricing of their goods is much more uniform. There is a higher probability that their listing of goods is smaller in types but also that the deviation of pricing is much lower unlike the others.

Visualizations Code

```
#VISUALIZATIONS

#Bar Chart
#df1['brand'].value_counts()[:10].plot(kind='barh')
#plt.xlabel("Count")
#plt.ylabel("Brand")
#plt.title("Top 10 Brands with Activity")

#Scatter
#sns.relplot(data=df1, x=df1['price'], y=df1['category'], hue="event_type", style="event_type")

#Violin
#sns.set_style('whitegrid')
#sns.violinplot(x='category',y='price', data=df1, figsize=(1000,1000))
plt.show()
```