Shopify Data Science Intern Challenge Question

January 17, 2021

0.1 Question 1

0.1.1 Problem Statement and Answers:

On Shopify, we have exactly 100 sneaker shops, and each of these shops sells only one model of shoe. We want to do some analysis of the average order value (AOV). When we look at orders data over a 30 day window, we naively calculate an AOV of $3145.13. Given that we know these shops are selling sneakers, a relatively affordable item, something seems wrong with our analysis.

a. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data. The initial AOV is misleading and likely skewed by a couple of extremely high value orders.

b. What metric would you report for this dataset? I would report the median because the median value doesn't depend on all the values in the dataset. Even if some of the values are more extreme, the effect on the median is smaller.

c. What is its value? The Median order value is $284.

0.1.2 Thought Process and Detialed Solutions:

Before looking at the dataset, the first thing is to understand what is AOV, how to calculate it, why it's important, and eventually what kind of insights we could derive in order to help improve business efficiency.

After doing some researches, AOV is one of the key performance indicators for Ecommerce to get new traffics. It shows how much revenue each order brings in on average. Higher AOV means higher revenue per order. There are many industry reports around the tactics of increasing AOV like up-sells, cross-sells, loyalty programs, mix and match offers, bundle products, etc. Ultimately, we want to leverage this metric to help us identify the driving products among all and therefore develop the strategy to increase the average money people spent per order.

The formula of AOV is the following: Average Order Value = Product Revenue / Total Transactions

In the problem statement, an AOV of $3145.13 is obtained by: sum of all order_amonut / total number of orders (calculation shown below). The calculation used the correct formula. However, taken into consideration that the sneaker should be a relatively affordable item, $3145.13 is way too high in common sense. Therefore, this number is misleading and likely skewed by a couple of extremely high value orders.

[1]:
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
```

[2]:
```
#import dataset
df = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set.csv')
df.head()
```

[2]:
```
   order_id  shop_id  user_id  order_amount  total_items payment_method  \
0         1       53      746           224            2           cash
1         2       92      925            90            1           cash
2         3       44      861           144            1           cash
3         4       18      935           156            1    credit_card
4         5       18      883           156            1    credit_card

            created_at
0  2017-03-13 12:36:56
1  2017-03-03 17:38:52
2   2017-03-14 4:23:56
3  2017-03-26 12:43:37
4   2017-03-01 4:35:11
```

[3]:
```
#drop NAs
df1 = df.dropna()
#drop duplicates of order_id
df1 = df1.drop_duplicates(subset=['order_id'])
```

[4]:
```
#create a unit price column based on the order amount and total_items
df1['unit_price'] = df1['order_amount']/df1['total_items']
```

[5]:
```
df1.describe()
```

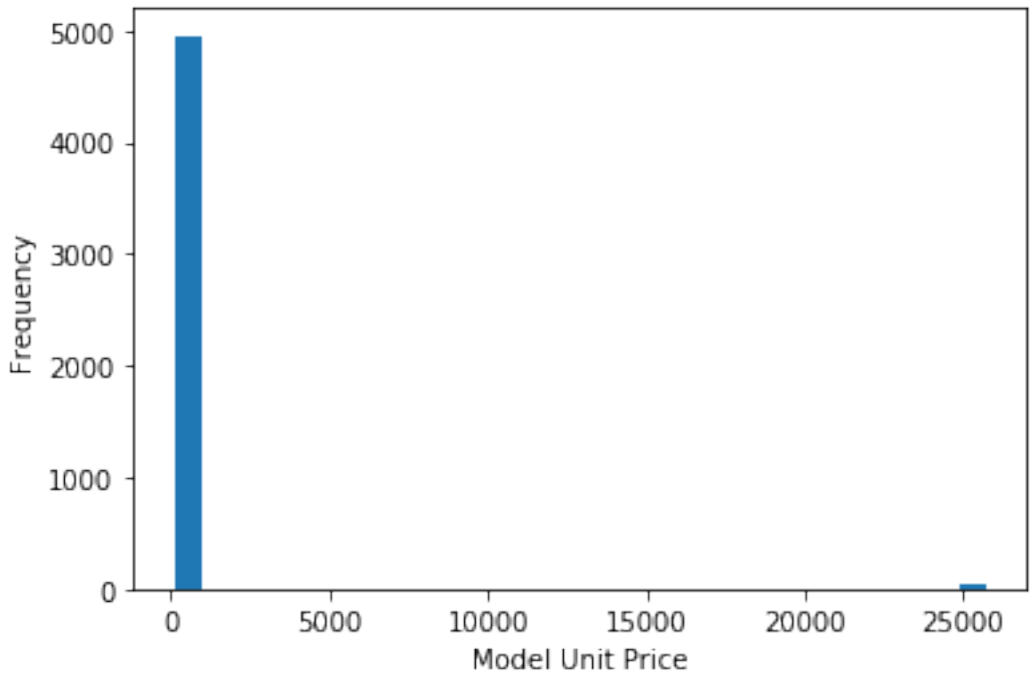
[5]:
```
          order_id      shop_id      user_id   order_amount  total_items  \
count  5000.000000  5000.000000  5000.000000    5000.000000   5000.00000
mean   2500.500000    50.078800   849.092400    3145.128000      8.78720
std    1443.520003    29.006118    87.798982   41282.539349    116.32032
min       1.000000     1.000000   607.000000      90.000000      1.00000
25%    1250.750000    24.000000   775.000000     163.000000      1.00000
50%    2500.500000    50.000000   849.000000     284.000000      2.00000
75%    3750.250000    75.000000   925.000000     390.000000      3.00000
max    5000.000000   100.000000   999.000000  704000.000000   2000.00000

         unit_price
count   5000.000000
mean     387.742800
std     2441.963725
min       90.000000
25%      133.000000
50%      153.000000
75%      169.000000
max    25725.000000
```

According to above table, the mean of order_amount is $3145.13, which is where the initial

AOV came from.

[6]:
```
#plot the model unit price
plt.hist(df1['unit_price'], density=False, bins=30)
plt.ylabel('Frequency')
plt.xlabel('Model Unit Price')
```

[6]:
```
Text(0.5, 0, 'Model Unit Price')
```



From the summary table, the standard deviation of unit price is $2441.96 which indicates a wide price range. The max value of unit price is $25725, which is way higher than most of the shop price. And I also plotted the histogram for unit price to help visulize the distribution of the model unit price.

[7]:
```
#check the orders with the unit_price greater than 5000
df1[df1['unit_price'] > 5000].head(10)
```

[7]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	\
160	161	78	990	25725	1	credit_card	
490	491	78	936	51450	2	debit	
493	494	78	983	51450	2	cash	
511	512	78	967	51450	2	cash	
617	618	78	760	51450	2	cash	
691	692	78	878	154350	6	debit	
1056	1057	78	800	25725	1	debit	
1193	1194	78	944	25725	1	debit	
1204	1205	78	970	25725	1	credit_card	
1259	1260	78	775	77175	3	credit_card	

```
                created_at  unit_price
160    2017-03-12 5:56:57     25725.0
490   2017-03-26 17:08:19     25725.0
493   2017-03-16 21:39:35     25725.0
511    2017-03-09 7:23:14     25725.0
617   2017-03-18 11:18:42     25725.0
691   2017-03-27 22:51:43     25725.0
1056  2017-03-15 10:16:45     25725.0
1193  2017-03-16 16:38:26     25725.0
1204  2017-03-17 22:32:21     25725.0
1259   2017-03-27 9:27:20     25725.0
```

[8]:
```
len(df1[df1['unit_price'] > 5000])
```

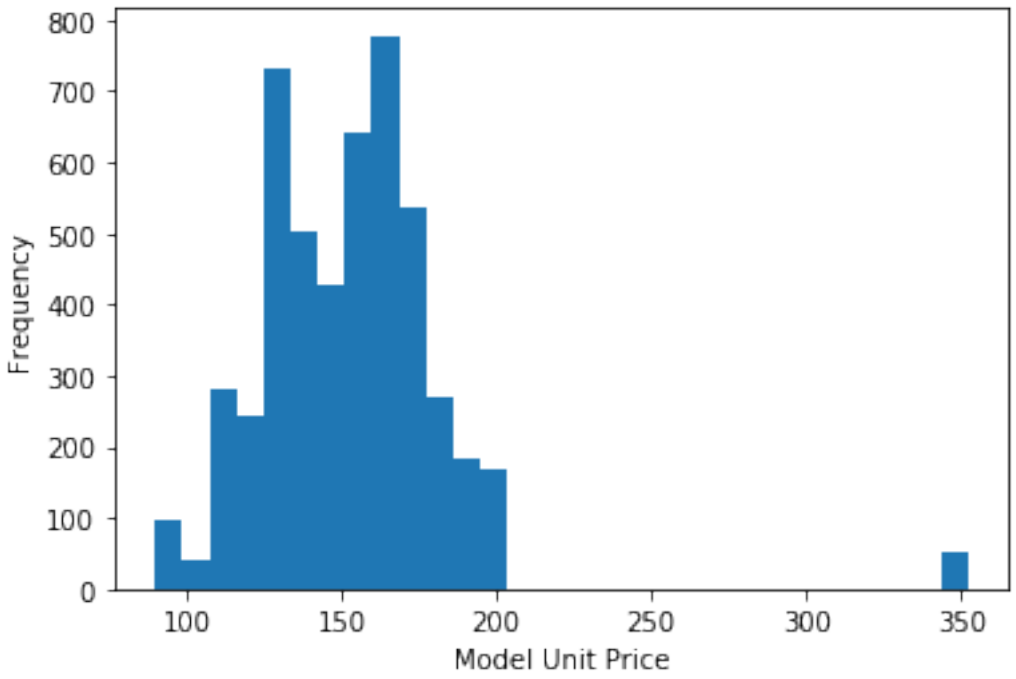
[8]:
```
46
```

[9]:
```
#take out orders with the unit price larger than 5000
df2 = df1[df1['unit_price'] <= 5000].reset_index(drop=True)
df2.head()
```

[9]:
```
   order_id  shop_id  user_id  order_amount  total_items payment_method  \
0         1       53      746           224            2           cash
1         2       92      925            90            1           cash
2         3       44      861           144            1           cash
3         4       18      935           156            1    credit_card
4         5       18      883           156            1    credit_card

            created_at  unit_price
0  2017-03-13 12:36:56       112.0
1  2017-03-03 17:38:52        90.0
2   2017-03-14 4:23:56       144.0
3  2017-03-26 12:43:37       156.0
4   2017-03-01 4:35:11       156.0
```

[10]:
```
#plot the model unit price for the updated dataset df2
plt.hist(df2['unit_price'], density=False, bins=30)
plt.ylabel('Frequency')
plt.xlabel('Model Unit Price')
```

[10]:
```
Text(0.5, 0, 'Model Unit Price')
```



According to the above analyses, there are 46 orders with the unit price of $25725 and they all come from shop 78. It only takes up around 0.92% of the total dataset. So, there wouldn't be a big impact if we remove them for our analyses.

[11]:
```
#original summary table
df1.describe()
```

[11]:

	order_id	shop_id	user_id	order_amount	total_items	unit_price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	5000.000000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720	387.742800
std	1443.520003	29.006118	87.798982	41282.539349	116.32032	2441.963725
min	1.000000	1.000000	607.000000	90.000000	1.00000	90.000000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000	133.000000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000	153.000000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000	169.000000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000	25725.000000

[12]:
```
#summary table after taking out orders having unit price greater than 5000
df2.describe()
```

[12]:

	order_id	shop_id	user_id	order_amount	total_items	unit_price
count	4954.000000	4954.000000	4954.000000	4954.000000	4954.000000	4954.000000
mean	2498.990916	49.819540	848.919257	2717.367784	8.851029	152.475575
std	1444.498907	29.014845	87.846007	41155.996469	116.857286	31.260218
min	1.000000	1.000000	607.000000	90.000000	1.000000	90.000000
25%	1248.250000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2494.500000	50.000000	849.000000	284.000000	2.000000	153.000000
75%	3750.750000	74.000000	925.000000	390.000000	3.000000	168.000000
max	5000.000000	100.000000	999.000000	704000.000000	2000.000000	352.000000

After taking out the orders with the unit price greater than 5000, we zoom in the price range. From the above plot, we can see that the majority of the unit price is below 250. Here, average is not a good metirc to look at as it is likely skewed by extreme values.

To better represent the whole picture, I would choose to report median order value instead. Unlike the mean, the median value doesn't depend on all the values in the dataset. Even if some of the values are more extreme, the effect on the median is smaller. And we can see that the median is the same for both df1 and df2 becuase those 46 orders only takes up a tiny portion in the dataset. Here, we have a right skewed distribution. The median is a better measure of central tendency than the mean.

From the above summary table, the Median order value is $284.