DSIC

May 13, 2022

1 Fall 2022 Data Science Intern Challenge

1.0.1 Library Importation

[ ]:
```
import pandas as pd
import plotly.express as px
import seaborn as sns
import numpy as np
from matplotlib import pyplot as plt
```

1.0.2 Data Importation

[ ]:
```
data = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set - Sheet1.
↪csv')
data.head()
```

[ ]:
```
   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
```

1.0.3 Data Investigation

[ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   order_id        5000 non-null   int64
 1   shop_id         5000 non-null   int64
 2   user_id         5000 non-null   int64
 3   order_amount    5000 non-null   int64
 4   total_items     5000 non-null   int64
 5   payment_method  5000 non-null   object
 6   created_at      5000 non-null   object
dtypes: int64(5), object(2)
memory usage: 273.6+ KB
```

We see here that the dataset is very clean, no missing values. We can now see how we end up with an AOV of $3145.13

[ ]:
```python
print("Dates:", [data.created_at.min(),data.created_at.max()])
print("Rough Average or orfer amount:",data.order_amount.mean())
```

```
Dates: ['2017-03-01 0:08:09', '2017-03-30 9:55:00']
Rough Average or orfer amount: 3145.128
```

When roughly calculating the mean of the column "Order_Amount", we see that indeed, the AOV is $3145.128 over the 30 days period of time. Let's try to understand why this mean is so high

[ ]:
```python
data.order_amount.describe()
```

[ ]:
```
count      5000.000000
mean       3145.128000
std       41282.539349
min          90.000000
25%         163.000000
50%         284.000000
75%         390.000000
max      704000.000000
Name: order_amount, dtype: float64
```

We directly see here that something obvious shows up, the max amount at $+700k. We also understant that given the different quartiles, a small minority or orders is pushing high the mean of the rest. Let's investigate these outsiders.

[ ]:
```python
fig = px.scatter(data, x="created_at",
                 y="order_amount",
                 color="shop_id",
                 size_max=60,
                 hover_data=["payment_method", "total_items"],
                 labels={
                     "created_at": "Time",
                     "order_amount": "Order Amount",
                     "user_id" : "Buyer"
                 },
                 title="Sales Distribution" )
```

```
fig.show()
```

There is a lot to take away from this plot.

First, we clearly see the outliers from the dataset that are adding a huge bias in our mean.

From these ourliers, we know that they are broadly sparsed through the month

There are 12 transactions of this sort with the **exact same amount** what makes me doubt about any "random" reason for these numbers.

All these transactions have been processed at the **same shop** (id = 42), and by credit card (What maks sense given the huge amount of the transaction)

During these transactions, the **exact same number of items** have been purchased.

[ ]:
```
px.histogram(data[data['shop_id'] == 42],x="order_amount" ,title = "Order␣
↪amount Histogram (shop_id = 42)",
histnorm = "percent",template = "ggplot2", nbins = 100)
```

We see here that if this shop is mainly responsible for the bias in the mean, it also records more reliable sellings that account fot 2/3 of this ship orders.

If we assume that we understood the reason of these wholesales in this shop, Let's see how the analysis changes when we drop this shop.

[ ]:
```
data2 = data.copy()
data2.drop(data2[data2['shop_id'] == 42].index, inplace = True) #Here we drop␣
↪the rows containing the orders from the 42th shop
fig = px.scatter(data2, x="created_at",
                 y="order_amount",
                 color="shop_id",
                 size_max=60,
                 hover_data=["payment_method", "total_items"],
                 labels={
                     "created_at": "Time",
                     "order_amount": "Order Amount",
                 },
                 title="Sales Distribution" )
fig.show()
```

Now, another Shop is outperforming the others in terms of order amount mean. The shop 78.

We see that this shop is selling expensive shoes. ($25.7k)

1.1 Assumptions over the outliers

Regarding the shop 42, $+700k orders of 2000 shoes, it can either be a mistake, but I will assume that another shop (outside shopify) is buying from shopify's shop wholesale for resale.

Then for the shop 78, I will assume that the shop is selling very exclusive shoes that cost $+25k/pair.

1.2 How to beter evaluate this data

Evaluating such a dataset will largly depend on the application and the goal behind. Either we are looking at the best-sellers items, the trends in the chops, the best shops, or trying to open new shops or close some.

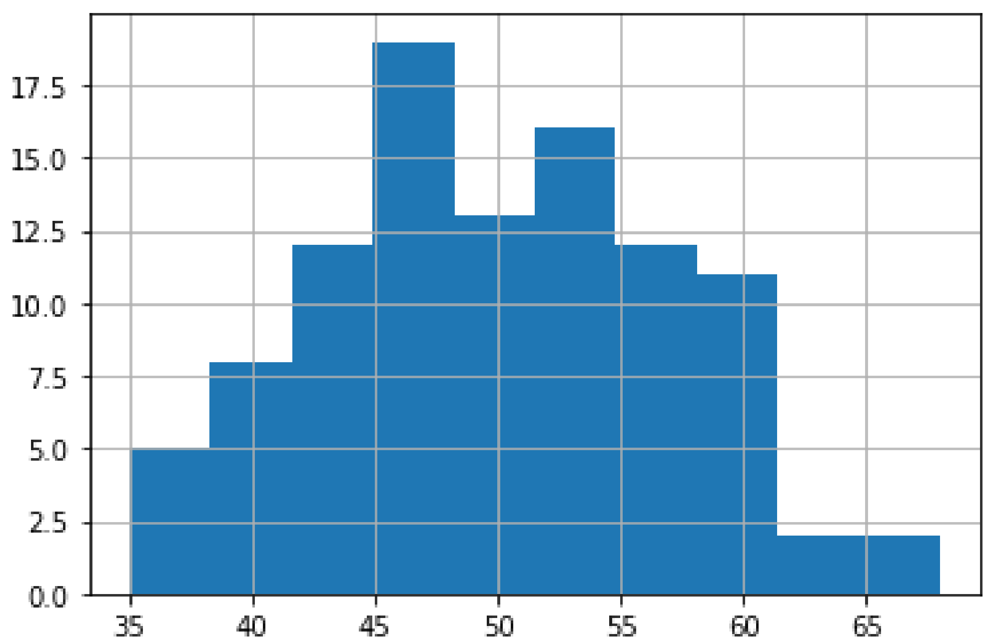
We could for example through this dataset sound each shop and know how they are going to maybe come up with new features, recommendations for the owners.

But I will assume for this Chalenge that we are looking for a clear overview at a higher level of the "sneakers" market on Spotify. Maybe to give insights for new customers who would want to open a sneakers shop.

1. We have seen that the mean is not a good metric since outliers tends to add a huge bias on this number.
2. A well known way to evaluate the global meaning of a dataset with outliers is the median and standard deviation.
3. Given the nature of the outliers, I would analyse the average amount per shop.
4. Another way to analyse this dataset would be to try to look at the sales distribution among the shops. (market share)
5. We can separate the types of shops (Small retailers / Large retailers)
6. We can separate the type of shoes (Average shoes / Exclusive Shoes)

[ ]:
```
data.groupby("shop_id").count()["order_amount"].hist()
```

[ ]: <AxesSubplot:>



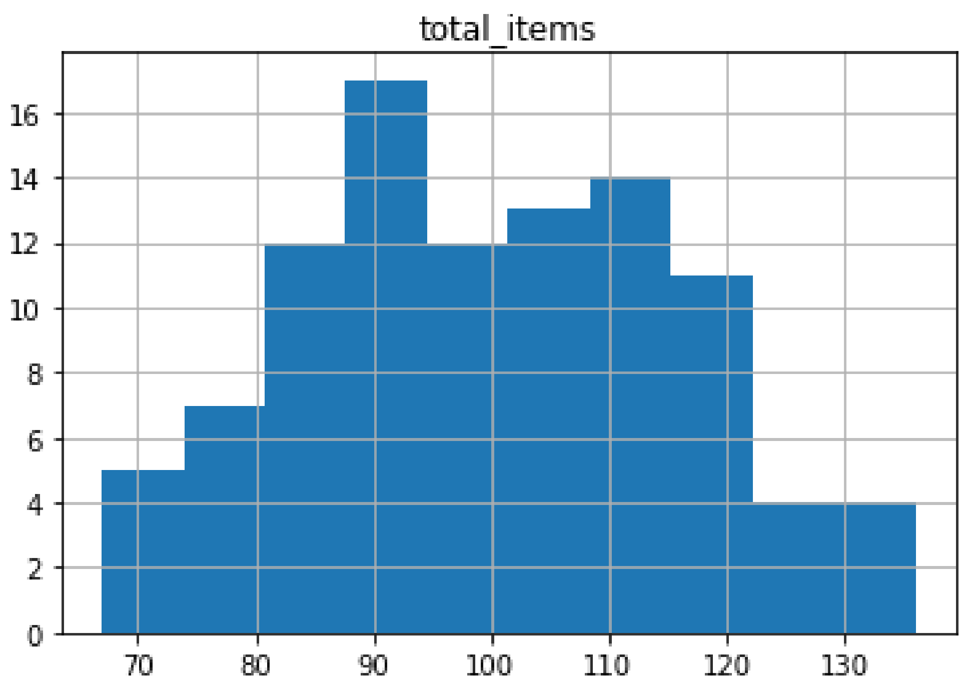
This graph shows that the distribution of the number of orders per shop shows no identifyable pattern.

Therefore, ther is no shop that we can categorize as huge seller in term of number of order.

Let's now analyse the number of items sold to categorize small and big shops

[ ]:
```
group = pd.DataFrame(data.groupby("shop_id").sum()["total_items"])
group[group["total_items"]<500].hist()
```

[ ]:
```
array([[<AxesSubplot:title={'center':'total_items'}>]], dtype=object)
```



[ ]:
```
group[group["total_items"]>500]
```

[ ]:
```
         total_items
shop_id
42             34063
```

This line of code shows us that only the shop 42 has sold more than 500 pairs. So when analysing the "small" shops we will just consider the dataset less the shop 42.

[ ]:
```
dataCateg = data.copy()
dataCateg['Shoes_Category']=np.where(dataCateg["unitPrice"]<=10000, "Average",␣
 ↪"Exclusive")
dataCateg['Shop_Category']=np.where(dataCateg["shop_id"]==42, "Big", "Small")
```

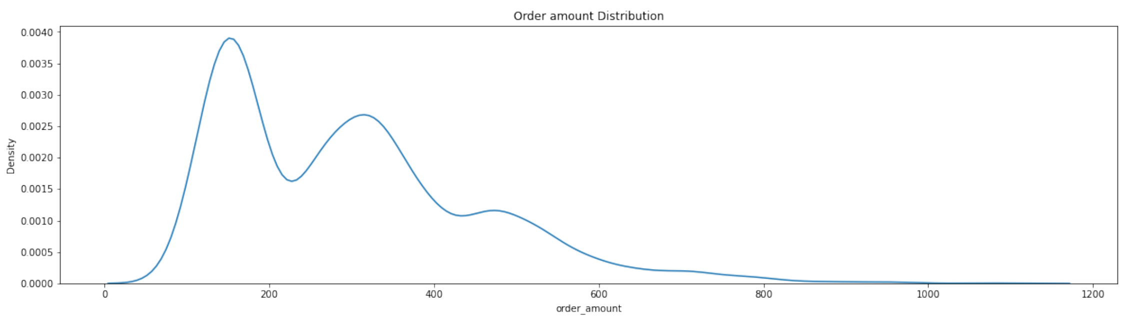
Now let's calculate the statistical metrics on the Small shops that are selling average shoes:

[ ]:
```
dataCateg[(dataCateg.Shoes_Category =='Average') & (dataCateg.
↪Shop_Category=='Small')]["order_amount"].describe()
```

[ ]:
```
count    4903.000000
mean      300.155823
std       155.941112
min        90.000000
25%       163.000000
50%       284.000000
75%       386.500000
max      1086.000000
Name: order_amount, dtype: float64
```

Now we are comparing comparable shops, and we see here that the metrics are more coherent with the business.

[ ]:
```
plt.figure(figsize=[20,5])
sns.kdeplot(dataCateg[(dataCateg.Shoes_Category =='Average') & (dataCateg.
↪Shop_Category=='Small')]["order_amount"])
plt.title("Order amount Distribution")
plt.show()
```



We observe a Pareto chart. (80/20 rule) That is a commun distribution in business. A large proportion of the small amount orders are responsible for a small proportion of the benefits

1.3 Conclusion

a. The calculation used to find an AOV were good, the problem is from the framework in with we have applied this method. Ineed, by looking at the dataset more in depth, we notice not only outlayers, but more importantly, shops are not all the same. Therefore taking a rough arithmetic mean of it does not make any business sens. A better way to evaluate this data is first to take into account the standart deviation of the distribution. This is the first hint that will tell us if we are comparing numbers that are comparable. We quickly see here that the SD of this distribution is very high due to the large variety of shops.

b. 2 different options are to consider when looking at useful metric. First, the median will give us a better idea of the “average” order value/ It is a way to deal with dataset with high variance. The drawback is that we just count the number above and below a certain number. In this way, we do not take into account the precise value of each observation and hence does not use all information available in the data.

Another alternative would be to give more sens to to the dataset instead of taking it as a whole. Indeed, in this dataset we have seen that couple of shops are outstanding the others and therefore are not comparable in terms of numbers. Even a normalization would tend to squash the others. So the second option is to separate the dataset in order to compare shops of the same size, or shops that represent the most the global vision of the market. This is why we have got rid of the outlayers (2 shops) in the second period that will give us a better understanding of the market in itself. This second analysis has more business value since we can draw conclusions about the expected profit, expected number of orders… of an upcoming shop for example.

The code is design such as if another outstanding shop is added to the dataset, the program will direcly put in in the category it belongs in order not to interfer with other analysis. I believe the idea to be more important than the actual tresholds that I have arbitrarily chosen. they are of course discussable.

c. Assuming that we go for the "Categorization" strategy, the AOV is \$300, with a Standard deviation of \$155.94, a median at \$284 and a max of \$1086